
dynamicreports Documentation

Release 5.1.0

DynamicReports

Mar 15, 2021

1 Table of Contents

1

1.1 Home

1.1.1 Free and open source Java reporting tool

DynamicReports is based on JasperReports. It allows to create dynamic report designs and it doesn't need a visual report designer. You can very quickly create reports and produce documents that can be displayed, printed or exported into many popular formats such as PDF, Excel, Word and others.

1.1.2 Why should I use DynamicReports?

You can get a lot of benefits while using this library, such as

Easy to use

Very easy to use, which not only saves a lot of your time and money but also increases reporting productivity and decreases training time. Only a very small amount of code is needed to create a report and even more complex reports will be very clear, easy to maintain and understandable.

Dynamic report design

A design is created by a pure simple Java code. While it is generated at runtime, you can implement any logic in your code which decides how the report will look like, unlike the static reports (jrxml JasperReports templates) where the defined design cannot be changed at runtime. It's ideal for ad hoc reporting where it is needed to create reports dynamically.

Inherited report design

A design can even be inherited from another design. This is impossible in static report designs designed in visual designers.

No need for a visual designer

As mentioned above, a design is defined by a java code and hence you don't have to use a visual report designer. Visual designers are quite robust and require more effort and time for report creation. This library enables you to create reports in your favorite IDE at the same speed as you write a java code.

Ability to mix dynamic designs with static designs (jasper jrxml templates) You are able to design a part of the report in e.g. iReport and another part in DynamicReports. There are two ways how they can work together:

- static designs can be embedded in a dynamic design through subreports and placed into any band and any numbers of them
- static design as a base to which a dynamic design is added

1.1.3 Usage

DynamicReports is synchronized with a Maven central repository. For Maven projects you just add dependency to your maven configuration. In case you would like to use a development version, add a Sonatype Nexus snapshot repository to your maven configuration.

For non Maven projects you have to download a project with dependencies from the Download site.

To get started have a look at Getting started step-by-step tutorial which shows the basic usage of the tool. For more examples visit the Examples page where you will find a lot of useful examples.

In case you have questions regarding the usage, have a look at the Documentation and Forum. If you didn't get your answers there, feel free to post a question in the forum. To post in the forum you will need an account. If you don't have one, you can register at any time here.

If you think the library is missing a feature or has a bug, you can create a feature request or bug report in the Project tracker. Please provide a comprehensive description of your request when you are creating a new issue.

1.2 Getting Started

1.2.1 Overview

Creating a report with DynamicReports is very easy, take a look at the example below

```
1 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
2 public class Report {
3     private void build() {
4         try {
5             report() //create new report design
6                 .columns(...) //adds columns
7                 .groupBy(...) //adds groups
8                 .subtotalsAtSummary(...) //adds subtotals
9                 ...
10                //set datasource
11                .setDataSource(...)
```

(continues on next page)

(continued from previous page)

```

12      //export report
13      .toPdf(...) //export report to pdf
14      .toXls(...) //export report to excel
15      ...
16      //other outputs
17      .toJasperPrint() //creates jasperprint object
18      .show() //shows report
19      .print() //prints report
20      ...
21      } catch (DRException e) {
22          e.printStackTrace();
23      }
24  }
25  ...
26  }

```

See [ReportBuilder](#) class for all available features and [JasperReportBuilder](#) class for all available exports and outputs.

1.2.2 Simple report

Please note that this is a tutorial example and that this tutorial doesn't describe all features!

You will learn in this section how to create a simple report step by step. First we need to create an empty report desing, configure it and set a datasource. Afterwards the report will be ready to export into any format. It's important to remember the [DynamicReports](#) class, through which you will have available most of features. The class provides methods for building a particular piece of a report. Let's start!

Step 1 : Start

First define columns through [DynamicReports.col.column](#)(title, field name, datatype) (the field name must match with the name of a field contained in the datasource) and pass them into the report as follows:

```

1  .columns(//add columns
2      //          title,          field name          data type
3      col.column("Item",          "item",          type.stringType()),
4      col.column("Quantity",      "quantity",    type.integerType()),
5      col.column("Unit price",    "unitprice",  type.bigDecimalType()))

```

now define some text at the title and number of pages at the page footer as follows:

```

1  .title(cmp.text("Getting started"))//shows report title
2  .pageFooter(cmp.pageXofY())//shows number of page at page footer

```

[DynamicReports.cmp.text](#)(some text) - creates a component that shows a text [DynamicReports.cmp.pageXofY](#)() - creates a component that shows page X of Y Methods [title\(...\)](#) and [pageFooter\(...\)](#) allows to customize a particular report band by adding components to it

Step 2 : Styles

Each style can have a parent style from which it will inherit its properties. Empty style can be created by [DynamicReports.stl.style](#)()

```
1 StyleBuilder boldStyle          = stl.style().bold();
2 StyleBuilder boldCenteredStyle = stl.style(boldStyle)
3                               .setHorizontalAlignment(HorizontalAlignment.
4                               ↪CENTER);
5
6 StyleBuilder columnTitleStyle  = stl.style(boldCenteredStyle)
7                               .setBorder(stl.pen1Point())
8                               .setBackgroundColor(Color.LIGHT_GRAY);
9
10 report()
11     .setColumnTitleStyle(columnTitleStyle)
12     .highlightDetailEvenRows()
13     .title(cmp.text("Getting started").setStyle(boldCenteredStyle))
14     .pageFooter(cmp.pageXofY().setStyle(boldCenteredStyle))
```

Step 3 : Additional columns

You can very easy multiply, divide, add or subtract column of numbers by another column of numbers or by a number value

```
1 //price = unitPrice * quantity
2 TextColumnBuilder<BigDecimal> priceColumn = unitPriceColumn.multiply(quantityColumn)
3                               .setTitle("Price");
```

Adding percentage of any column of numbers is simple `DynamicReports.col.percentageColumn(title, column)`

```
1 PercentageColumnBuilder pricePercColumn = col.percentageColumn("Price %",
2                               ↪priceColumn);
```

`DynamicReports.col.reportRowNumberColumn(title)` creates a column that shows row number

```
1 TextColumnBuilder<Integer> rowNumberColumn =
2     col.reportRowNumberColumn("No.")
3     //sets the fixed width of a column, width = 2 * character width
4     .setFixedColumns(2)
5     .setHorizontalAlignment(HorizontalAlignment.CENTER);
```

Step 4 : Group

We continue by adding a group as shown below

```
1 .groupBy(itemColumn)
```

Step 5 : Subtotals

Now we can try to sum a column of numbers. Subtotal of sum is created through `DynamicReports.sbt.sum(column)`

```
1 .subtotalsAtSummary(
2     sbt.sum(unitPriceColumn), sbt.sum(priceColumn))
3 .subtotalsAtFirstGroupFooter(
4     sbt.sum(unitPriceColumn), sbt.sum(priceColumn))
```

Method `subtotalsAtSummary(...)` allows to add subtotals to the summary band Method `subtotalsAtFirstGroupFooter(...)` will find first defined group and add subtotals to the group footer band

Step 6 : Charts

DynamicReports.cht provide methods for building charts. Category and series are required.

```

1 Bar3DChartBuilder itemChart = cht.bar3DChart()
2   .setTitle("<a href='https://web.archive.org/web/
   ↳20180130194401/http://www.dynamicreports.org/examples/sales' title='Sales'>Sales</a>
   ↳ by item")
3   .setCategory(itemColumn)
4   .addSerie(
5     cht.serie(unitPriceColumn), cht.
   ↳serie(priceColumn));
6 Bar3DChartBuilder itemChart2 = cht.bar3DChart()
7   .setTitle("<a href='https://web.archive.org/web/
   ↳20180130194401/http://www.dynamicreports.org/examples/sales' title='Sales'>Sales</a>
   ↳ by item")
8   .setCategory(itemColumn)
9   .setUseSeriesAsCategory(true)
10  .addSerie(
11    cht.serie(unitPriceColumn), cht.serie(priceColumn));

```

Chart is a component and can be placed into any report band.

```

1 .summary(itemChart, itemChart2)

```

Step 7 : Column grid & Containers

Components inserted into the bands are arranged vertically, each component is below the previously added component. To arrange components horizontally it is needed to wrap these components with a horizontal container. Container is a component as well and therefore it can be added to any of the report bands.

```

1 .summary(
2   cmp.horizontalList(itemChart, itemChart2))

```

Columns layout can be modified by a column grid. The layout is applied to the columns title, details and subtotals.

```

1 .columnGrid(
2   rowNumberColumn, quantityColumn, unitPriceColumn,
3   grid.verticalColumnGridList(priceColumn, pricePercColumn))

```

Step 8 : Hide subtotal

Subtotal for the group notebook is unnecessary because contains only one row. We need to change the group declaration and set the boolean expression condition on which depends whether subtotal is printed.

DynamicReports.exp.printWhenGroupHasMoreThanOneRow(itemGroup) creates a boolean condition which returns true when itemGroup has more than one row.

```

1 ColumnGroupBuilder itemGroup = grp.group(itemColumn);
2
3 itemGroup.setPrintSubtotalsWhenExpression(
4   exp.printWhenGroupHasMoreThanOneRow(itemGroup));
5 .groupBy(itemGroup)

```

Step 9 : Title

First define a title style.

```
1 StyleBuilder titleStyle = stl.style(boldCenteredStyle)
2                               .setVerticalAlignment(VerticalAlignment.MIDDLE)
3                               .setFontSize(15);
```

DynamicReports.cmp.image() creates an image component

DynamicReports.cmp.filler() creates an empty component

```
1 .title(//shows report title
2     cmp.horizontalList()
3     .add(
4         cmp.image(getClass().getResourceAsStream("../images/dynamicreports.png")).
5         ↪setFixedDimension(80, 80),
6         cmp.text("DynamicReports").setStyle(titleStyle).
7         ↪setHorizontalAlignment(HorizontalAlignment.LEFT),
8         cmp.text("Getting started").setStyle(titleStyle).
9         ↪setHorizontalAlignment(HorizontalAlignment.RIGHT))
10    .newRow()
11    .add(cmp.filler().setStyle(stl.style().setTopBorder(stl.pen2Point()))).
12    ↪setFixedHeight(10))
```

The defined filler creates an additional blank space between the title and the column header.

Setting top border of a filler draws the line at the bottom of the title.

Horizontal list, as previously mentioned, arranges components horizontally in one row but by calling the method row() a new horizontal list is created which is located at the bottom of the previous horizontal list.

Step 10 : Currency data type

Unit price and price column are currency types. Showing currency is possible by setting pattern to both columns (via method setPattern()), but the best practice is to create a custom data type and apply it to the columns. The custom data type then can be used in other reports.

```
1 CurrencyType currencyType = new CurrencyType();
2
3 TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price", "unitprice",
4     ↪currencyType);
5 //price = unitPrice * quantity
6 TextColumnBuilder<BigDecimal> priceColumn = unitPriceColumn.
7     ↪multiply(quantityColumn).setTitle("Price")
8     .
9     ↪setDataType(currencyType);
10 private class CurrencyType extends BigDecimalType {
11     private static final long serialVersionUID = 1L;
12
13     @Override
14     public String getPattern() {
15         return "$ #,###.00";
16     }
17 }
```

(continues on next page)

(continued from previous page)

```

13     }
14 }

```

Step 11 : Detail row highlighters

```

1 ConditionalStyleBuilder condition1 = stl.conditionalStyle(cnd.greater(priceColumn, ↵
↵150))
2                                     .setBackgroundColor(new Color(210, 255, 210));
3 ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.smaller(priceColumn, ↵
↵30))
4                                     .setBackgroundColor(new Color(255, 210, 210));
5 .detailRowHighlighters(
6 condition1, condition2)

```

Condition1 is applied only if price is greater than 150 and sets background color of a row to green.

Condition2 is applied only if price is smaller than 30 and sets background color of a row to red.

Step 12 : Conditional styles

```

1 ConditionalStyleBuilder condition3 = stl.conditionalStyle(cnd.greater(priceColumn, ↵
↵200))
2                                     .setBackgroundColor(new Color(0, 190, 0))
3                                     .bold();
4 ConditionalStyleBuilder condition4 = stl.conditionalStyle(cnd.smaller(priceColumn, ↵
↵20))
5                                     .setBackgroundColor(new Color(190, 0, 0))
6                                     .bold();
7 StyleBuilder priceStyle = stl.style()
8     .conditionalStyles(
9         condition3, condition4);
10
11 priceColumn = unitPriceColumn.multiply(quantityColumn).setTitle("Price")
12     .setDataType(currencyType)
13     .setStyle(priceStyle);

```

Condition3 is applied only if price is greater than 200 and sets background color of a price to green.

Condition4 is applied only if price is smaller than 20 and sets background color of a price to red.

1.3 Syntax

Before you start creating reports with DynamicReports, you should consider which java syntax you will use. Java 5 added static imports that allow static methods or constants to be referenced without qualifying them with a class name. DynamicReports provides a lot of static methods of creating every part of a report and therefore you can consider using a static import for these methods. And DynamicReports also provides the ability to use [Method chaining](#) (also known as [Fluent interface](#)). Both are useful for code readability, the ability to replace multiple calls to a class with a single one, and for invoking multiple method calls. Let's see in detail the difference between using and not using a static import and method chaining:

1.3.1 Classic java Syntax

Classic java syntax (without using a static import and method chaining)

```
1 JasperReportBuilder report = DynamicReports.report();
2 report.addColumn(Columns.column("Item", "item", DataTypes.stringType()));
3 report.addColumn(Columns.column("Quantity", "quantity", DataTypes.integerType()));
4 report.addTitle(Components.text("Report title"));
5 report.setDataSource(dataSource);
6 report.show();
```

1.3.2 Using a static import

Using a static import for DynamicReports methods and method chaining

```
1 import static net.sf.dynamicreports.DynamicReports.*;
2 report()
3 .columns(
4     col.column("Item", "item", type.stringType()),
5     col.column("Quantity", "quantity", type.integerType())
6 .title(cmp.text("Report title"))
7 .setDataSource(dataSource)
8 .show();
```

As you can see, static imports and method chaining improve readability of the code but they are not mandatory. It's only up to you which code syntax you will prefer to use.

All examples and tutorials in the documentation use static imports and method chaining to provide more readability of the code.

1.3.3 Examples

[SimpleReport_ClassicSyntax](#), [SimpleReport_Step01](#)

1.4 Report Builder

1.4.1 JasperReportBuilder

The most used report builder for creating reports. It allows constructing and customizing the whole report content. A report consists of bands, columns, subtotals, groups, and other parts. Each part is created and configured using a particular builder method and it's passed to the report builder instance.

In the table below, you will find a list of all available builders:

Builder name	Using a static import for DynamicReports methods import static net.sf.dynamicreports.DynamicReports.*	Without a static import	Using a particular builder	Description
Column-Builders	col.*	DynamicReports.col.*	Columns.*	A set of methods of creating report columns
Grid-Builders	grid.*	DynamicReports.grid.*	Grids.*	A set of methods of customizing columns layout
Group-Builders	grp.*	DynamicReports.grp.*	Groups.*	A set of methods of creating report groups
Subtotal-Builders	sbt.*	DynamicReports.sbt.*	Subtotals.*	A set of methods of creating column subtotals
Style-Builders	stl.*	DynamicReports.stl.*	Styles.*	A set of methods of creating and customizing styles
Component-Builders	cmp.*	DynamicReports.cmp.*	Components.*	A set of methods of creating components
Expression-Builders	exp.*	DynamicReports.exp.*	Expressions.*	A set of build in expressions
Condition-Builders	cnd.*	DynamicReports.cnd.*	Conditions.*	A set of build in condition expressions
DataType-Builders	type.*	DynamicReports.type.*	DataTypes.*	A set of build in data types
Chart-Builders	cht.*	DynamicReports.cht.*	Charts.*	A set of methods of creating and customizing charts
Exporter-Builders	export.*	DynamicReports.export.*	Exporters.*	A set of methods of creating exporters
Barcode-Builders	bcode.*	DynamicReports.bcode.*	Barcodes.*	A set of methods of creating barcodes
Crosstab-Builders	ctab.*	DynamicReports.ctab.*	Crosstabs.*	A set of methods of creating and customizing crosstabs

Examples Quick usage:

```

1 report ()
2 .columns (
3     col.column("Item", "item", type.stringType()),
4     col.column("Quantity", "quantity", type.integerType()))
5 .title(cmp.text("Report title"))
6 .setDataSource(...)
7 .show();

```

Another example: [SimpleReport_Step01](#)

1.4.2 JasperConcatenatedBuilder

This report builder allows concatenating several separated reports into one single document. Each report starts on a new page with its own page dimension.

Examples Quick usage:

```
1 JasperReportBuilder report1 = ...;
2 JasperReportBuilder report2 = ...;
3 concatenatedReport()
4 .concatenate(report1, report2)
5 .toPdf(...)
```

Another examples: [ConcatenatedReport1](#), [ConcatenatedReport2](#)

1.5 Data source

The important part of reporting is the data source which fills the report design with given data. Data can be retrieved from various sources such as a database, a xml file or using an interface JRDataSource. JasperReports itself has a lot of implementations of JRDataSource interface and basically implementation of this interface by you is quite often unneeded. Data in reports can be additionally sorted or filtered and some components such as chart, subreport, and crosstab can have their own data source if it's needed. Please refer to the link below for examples. Configuration options

Method	Description
setFilterExpression(DRIExpression<Boolean> filterExpression)	Sets a dataset filter expression. The expression must be a type of Boolean
sortBy(TextColumnBuilder<?> ... sortCols), sortBy(SortBuilder ... sorts), addSort(SortBuilder ... sorts)	Adds a sort field to the dataset

1.5.1 Database

In this type of data source, data are retrieved from a database. You only need a properly configured database connector and a SQL query.

Configuration options

Method	Description
setDataSource(String sql, Connection connection), setDataSource(QueryBuilder query, Connection connection), etDataSource(ResultSet resultSet)	Sets a database data source

Examples

Quick usage:

```

1 java.sql.Connection connection = ...;
2 String query = "SELECT * FROM ...";
3 report()
4     .setDataSource(query, connection)

```

Another example: *DatabaseDatasourceReport*

1.5.2 Data source object

In this type, a JRDataSource object is expected as a data source. A good explanation of various implementations of data sources, as well as a few examples can be found using the following link: [JasperReports data sources](#)

Configuration options

Method	Description
setDataSource(JRDataSource dataSource), setDataSource(Collection<?> collection)	Sets a data source object

Examples

Quick usage:

```

1 List<JavaBean> data = new ArrayList<JavaBean>();
2 report()
3     .setDataSource(new JRBeanCollectionDataSource(data))

```

Another example: *CollectionDatasourceReport*

Tags: datasource

1.6 Report Bands

Each report consists of several sections (bands). Each band has its particular location, signification, dimension, and can contain components. The height of the band is based on the component's dimension that it contains and can grow when the components are stretched.

	Title	
	Page Header	
	Column Header	
	Detail 1	
	Column Footer	
	Page Footer	
	Summary	

Examples

Quick usage:

```
1 report ()
2   .title(cmp.text("This is a title band"))
3   .pageHeader(cmp.text("This is a page header band"))
4   .pageFooter(cmp.text("This is a page footer band"))
```

Another example: [BandReport](#)

A report contains these bands:

1.6.1 Title

The band is printed on the first page and only once.

Configuration options

#	method	description
1	<code>titleOnANewPage()</code>	Prints the title band on a separate page
2	<code>setTitleOnANewPage(Boolean titleOnANewPage)</code>	Sets whether or not the title band is printed on a separate page
3	<code>setTitleSplitType(SplitType splitType)</code>	Sets the title band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
4	<code>setTitleStyle(StyleBuilder style)</code>	Sets a title band style
5	<code>title(ComponentBuilder<?, ?> ... comp),</code> <code>addTitle(ComponentBuilder<?, ?> ... comp)</code>	Adds components to the title band

1.6.2 Page header

The band is printed on each page at the top of the page.

Configuration options

#	method	description
1	<code>setPageHeaderSplitType(SplitType splitType)</code>	Sets the page header band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setPageHeaderStyle(StyleBuilder style)</code>	Sets a page header band style
3	<code>pageHeader(ComponentBuilder<?, ?> ... co),</code> <code>addPageHeader(ComponentBuilder<?, ?> ... components)</code>	Adds components to the page header band

1.6.3 Page footer

The band is printed on each page at the bottom of the page.

Configuration options

#	method	description
1	<code>setPageFooterSplitType(SplitType splitType)</code>	Sets the page footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setPageFooterStyle(StyleBuilder style)</code>	Sets a page footer band style
3	<code>pageFooter(ComponentBuilder<?, ?> ... co),</code> <code>addPageFooter(ComponentBuilder<?, ?> ... components)</code>	Adds components to the page footer band

1.6.4 Column header

The band is printed on each page at the bottom of the page and it's placed above the page footer band.

Configuration options

#	method	description
1	<code>setColumnFooterSplitType(SplitType type)</code>	Sets the column footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>floatColumnFooter()</code>	Prints the column footer band at the bottom of the column
3	<code>setFloatColumnFooter(Boolean floatColFooter)</code>	Sets whether or not the column footer band is printed at the bottom of the column
4	<code>setColumnFooterStyle(StyleBuilder style)</code>	Sets a column footer band style
5	<code>columnFooter(ComponentBuilder<?, ?></code> <code>... components),</code> <code>addColumnFooter(ComponentBuilder<?, ?></code> <code>... components)</code>	Adds components to the column footer band

1.6.5 Group header

The band is printed for each data group. It's placed above the grouped data and between the column header and footer.

#	method	description
1	<code>setGroupHeaderSplitType(GroupBuilder<?> group, SplitType splitType)</code>	Sets the group header band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setGroupHeaderStyle(GroupBuilder<?> group, StyleBuilder style)</code>	Sets a group header band style
3	<code>groupHeader(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components), addGroupHeader(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components)</code>	Adds components to the group header band

1.6.6 Group footer

The band is printed for each data group. It's placed below the grouped data and between the column header and footer.

#	method	description
1	<code>setGroupFooterSplitType(GroupBuilder<?> group, SplitType splitType)</code>	Sets the group footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setGroupFooterStyle(GroupBuilder<?> group, StyleBuilder style)</code>	Sets a group footer band style
3	<code>groupFooter(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components), addGroupFooter(GroupBuilder<?> group, ComponentBuilder<?, ?> ... components)</code>	Adds components to the group footer band

1.6.7 Detail

The band is printed for each record row in the data source and it's placed between the column header and footer band.

Configuration options

#	method	description
1	<code>setDetailSplitType(SplitType splitType)</code>	Sets the detail band split type SplitType.PREVENT - prevent band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setDetailStyle(StyleBuilder style)</code>	Sets a detail band style
3	<code>detail(ComponentBuilder<?, ?> ... components),</code> <code>addDetail(ComponentBuilder<?, ?> ... components)</code>	Adds components to the detail band

1.6.8 Detail header

The band is printed for each record row in the data source and it's placed above the detail band.

Configuration options

#	method	description
1	<code>setDetailHeaderSplitType(SplitType type)</code>	<p>Sets the detail header band split type</p> <p><code>SplitType.PREVENT</code> - prevent band from splitting <code>SplitType.STRETCH</code> - the band can be split, but never within its declared height. <code>SplitType.IMMEDIATE</code> - the band can be split</p>
2	<code>setDetailHeaderStyle(StyleBuilder style)</code>	Sets a detail header band style
3	<code>detailHeader(ComponentBuilder<?, ?></code> <code>... components),</code> <code>addDetailHeader(ComponentBuilder<?, ?></code> <code>... components)</code>	Adds components to the detail header band

1.6.9 Detail footer

The band is printed for each record row in the data source and it's placed below the detail band.

Configuration options

#	method	description
1	<code>setDetailFooterSplitType(SplitType type)</code>	Sets the detail footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setDetailFooterStyle(StyleBuilder style)</code>	Sets a detail footer band style
3	<code>detailFooter(ComponentBuilder<?, ?> ... components),</code> <code>addDetailFooter(ComponentBuilder<?, ?> ... components)</code>	Adds components to the detail footer band

1.6.10 Last page footer

The band is printed only on the last page at the bottom of the page.

Configuration options

#	method	description
1	<code>setLastPageFooterSplitType(SplitType type)</code>	Sets the last page footer band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
2	<code>setLastPageFooterStyle(StyleBuilder style)</code>	Sets a last page footer band style
3	<code>lastPageFooter(ComponentBuilder<?, ?> ... components),</code> <code>addLastPageFooter(ComponentBuilder<?, ?> ... components)</code>	Adds components to the last page footer band

1.6.11 Summary

The band is printed on the last page and only once.

Configuration options

#	method	description
1	<code>summaryOnANewPage()</code>	Prints the summary band on a separate page
2	<code>setSummaryOnANewPage(Boolean sumOnANewPage)</code>	Sets whether or not the summary band is printed on a separate page
3	<code>summaryWithPageHeaderAndFooter()</code>	Prints the summary band with the page header and footer
4	<code>setSummaryWithPageHeaderAndFooter(Boolean summaryWithPageHeaderAndFooter)</code>	Sets whether or not the summary band is printed with the page header and footer
5	<code>setSummarySplitType(SplitType splitType)</code>	Sets the summary band split type SplitType.PREVENT - prevents the band from splitting SplitType.STRETCH - the band can be split, but never within its declared height. SplitType.IMMEDIATE - the band can be split
6	<code>setSummaryStyle(StyleBuilder style)</code>	Sets a summary band style
7	<code>summary(ComponentBuilder<?, ?> ... comps),</code> <code>addSummary(ComponentBuilder<?, ?> ... cos)</code>	Adds components to the summary band

1.6.12 No data

The band is printed only when the data source is empty. It's used to show the information that there are not any data in the report.

Configuration options

#	method	description
1	<code>setNoDataStyle(StyleBuilder style)</code>	Sets a no data band style
2	<code>noData(ComponentBuilder<?, ?> ... comps),</code> <code>addNoData(ComponentBuilder<?, ?> ... comp)</code>	Adds components to the no data band

1.6.13 Background

The band is printed on each page. It's mostly used for adding watermarks to the report.

Configuration options

#	method	description
1	<code>setBackgroundStyle(StyleBuilder style)</code>	Sets a background band style
2	<code>background(ComponentBuilder<?, ?> ... co),</code> <code>addBackground(ComponentBuilder<?, ?> ... components)</code>	Adds components to the background band

Tags: layout

1.7 Columns

1.7.1 Overview

DynamicReports has an ability to display data in a multi-column layout. There is no limit in number of columns added to the report, only keep in mind that the more columns you add to the report, the less space will be available for each column. All columns automatically fit the available page width.

Item	Quantity	Unit price	Column title
Book	3	11.00	Column values
Book	1	15.00	
Book	3	10.00	
Book	8	9.00	
	15	45.00	Column subtotal

Column

The picture above shows what is a column and how it is divided.

Configuration options of a column title

#	method	description
1	<code>setTitle(String title),</code> <code>setTitle(DRIExpression<?></code> <code>titleExpression)</code>	Sets the column title
2	<code>setTitleStyle(StyleBuilder</code> ti- <code>tileStyle)</code>	Sets the column title style
3	<code>setTitleRows(Integer rows),</code> <code>setTitleFixedRows(Integer rows),</code> <code>setTitleMinRows(Integer rows)</code>	This method is used to define the height of a column title. The height is set to the rows multiplied by height of the font
4	<code>setTitleHeight(Integer height),</code> <code>setTitleFixedHeight(Integer</code> <code>height),</code> <code>setTitleMinHeight(Integer height)</code>	Sets the height of a column title

Configuration options of column values

#	method	description
1	setStyle(StyleBuilder style)	Sets the column value style
2	setPrintWhenExpression(DRIExpression <Boolean> printWhenExpression)	Sets the print when expression. The expression must be a type of Boolean and it decides whether or not a column value will be print
3	setRows(Integer rows), setFixedRows(Integer rows), setMinRows(Integer rows)	This method is used to define the height of a column value. The height is set to the rows multiplied by height of the font
4	setHeight(Integer height), setFixedHeight(Integer height), setMinHeight(Integer height)	Sets the height of a column value
5	setPrintRepeatedDetailValues(Boolean printRepeatedDetailValues)	Specifies whether or not print a value if the value is the same as the previous value
6	setHorizontalAlignment(HorizontalAlignment horizontalAlignment)	Sets the column value horizontal alignment
7	setPattern(String pattern), setPattern(DRIExpression<String> patternExpression)	Sets the column value format pattern
8	setValueFormatter(DRIValueFormatter <?, ? super U> valueFormatter)	Sets the column value format expression
9	setHyperLink(HyperLinkBuilder hyperLink)	Sets the column value hyperlink
a	setStretchWithOverflow(Boolean stretchWithOverflow)	
1.7. Columns		25
b	addProperty(DRIPropertyExpression	Adds a jasper property to the column value

Configuration options of a column

#	method	description
1	<code>setColumns(Integer columns),</code> <code>setFixedColumns(Integer columns),</code> <code>setMinColumns(Integer columns)</code>	This method is used to define the width of a column. The width is set to the columns multiplied by width of the character m for the used font
2	<code>setWidth(Integer width),</code> <code>setFixedWidth(Integer width),</code> <code>setMinWidth(Integer width)</code>	Sets the width of a column

Examples

Column examples

1.7.2 Text column

It is represented by a `TextColumnBuilder` instance and it is used to show values from the data source.

Builders

#	method	description
1	col.column(String fieldName, Class<T> valueClass), col.column(String title, String fieldName, Class<T> valueClass)	Creates a new column - title (optional) - the column title - fieldName - the name of the field - valueClass - the field value class
2	col.column(String fieldName, DRIDDataType<? super T, T> dataType), col.column(String title, String fieldName, DRIDDataType<? super T, T> dataType)	Creates a new column - title (optional) - the column title - fieldName - the name of the field - dataType - the field data type
3	col.column(FieldBuilder<T> field), col.column(String title, FieldBuilder<T> field)	Creates a new column - title (optional) - the column title - field - the field definition
4	columnInstance.add(TextColumnBuilder<? extends Number> column), columnInstance.add(Number number)	Creates a new column by adding a value or a column value to the columnInstance column
5	columnInstance.subtract(TextColumnBuilder<? extends Number> column), columnInstance.subtract(Number number)	Creates a new column by subtracting a value or a column value from the columnInstance column
6	columnInstance.multiply(TextColumnBuilder<? extends Number> column), columnInstance.multiply(Number number)	Creates a new column by multiplying the columnInstance column with a value or a column value
7	columnInstance.divide(int scale, TextColumnBuilder<? extends Number> col), columnInstance.divide(int scale, Number number)	Creates a new column by dividing the columnInstance column with a value or a column value
1.7. Columns		27

Examples

Quick usage:

```
1 report ()
2   .columns (
3     col.column("Item", "item", type.stringType()),
4     col.column("Quantity", "quantity", type.integerType())
5   .setDataSource(...)
```

Another example: *ColumnDataTypesReport*

1.7.3 Expression column

It is represented by a `TextColumnBuilder` instance and the displayed values are defined in an expression.

Builders

#	method	description
1	<code>col.column(DRIExpression<T> expression),</code> <code>col.column(String title, DRIExpression<T> expression)</code>	Creates a new expression column - title (optional) - the column title - expression - the value expression

Examples

Quick usage:

```
1 report ()
2   .columns (
3     col.column("Expression column", new ExpressionColumn())
4   .setDataSource(...)
```

```
5 private class ExpressionColumn extends AbstractSimpleExpression<String> {
6   public String evaluate(ReportParameters reportParameters) {
7     return ...;
8   }
9 }
```

Another example: *ExpressionColumnReport*

1.7.4 Percentage column

It is represented by a `PercentageColumnBuilder` instance. It calculates percentage values from the field or column values.

Builders

#	method	description
1	<code>col.percentageColumn(ValueColumnBuilder<?, ? extends Number> column),</code> <code>col.percentageColumn(String title,</code> <code>ValueColumnBuilder<?, ? extends Number> col)</code>	Creates a new percentage column from the column values - title (optional) - the column title - column - the column definition
2	<code>col.percentageColumn(String field,</code> <code>Class<? extends Number> valueClass),</code> <code>col.percentageColumn(String fld,</code> <code>String s1,</code> <code>Class<? extends Number> valueClass)</code>	Creates a new percentage column from the field values - title (optional) - the column title - fieldName - the name of the field - valueClass - the field value class
3	<code>col.percentageColumn(FieldBuilder<? extends Number> field),</code> <code>col.percentageColumn(String title,</code> <code>FieldBuilder<? extends Number> field)</code>	Creates a new percentage column from the field values - title (optional) - the column title - field - the field definition

Configuration options

#	method	description
1	<code>setTotalType(PercentageTotalType totalType)</code>	Sets the total type. Has effect only when the report contains at least one group
2	<code>setTotalGroup(GroupBuilder<?> totalGroup)</code>	Sets the total group. Has effect only when the report contains at least one group

Examples

Quick usage:

```

1 TextColumnBuilder<Integer> quantityColumn = col.column("Quantity", "quantity", type.
  ↳ integerType());
2 PercentageColumnBuilder quantityPercColumn = col.percentageColumn("Quantity [%]",
  ↳ quantityColumn);
3 report()
4 .columns(
5     quantityColumn, quantityPercColumn)
6 .setDataSource(...)
```

Another example: *PercentageColumnsReport*

1.7.5 Row number column

It is represented by a `TextColumnBuilder` instance and displays row numbers. Builders

#	method	description
1	<code>col.columnRowNumberColumn()</code> , <code>col.columnRowNumberColumn(String title)</code>	Create a new row number column, the row number <code>l</code> is reset on each new column - title (optional) - the column title
2	<code>col.pageRowNumberColumn()</code> , <code>col.pageRowNumberColumn(String title)</code>	Create a new row number column, the row number is reset on each new page - title (optional) - the column title
3	<code>col.reportRowNumberColumn()</code> , <code>col.reportRowNumberColumn(String title)</code>	Creates a new row number column - title (optional) - the column title

Examples

Quick usage:

```
1 report ()
2 .columns (
3     col.reportRowNumberColumn("Report row")
4 .setDataSource(...)
```

Another example: *RowNumberColumnsReport*

1.7.6 Boolean column

It is represented by a `BooleanColumnBuilder` instance and shows a boolean value either as a text or as an image.

Builders

#	method	description
1	<code>col.booleanColumn(String fld),</code> <code>col.booleanColumn(String titl,</code> <code>String fld)</code>	Creates a new boolean column - titl (optional) - the column title - fld - the name of the field
2	<code>col.booleanColumn(FieldBuilder<Boolean></code> <code>fld),</code> <code>col.booleanColumn(String title,</code> <code>FieldBuilder</code> <code><Boolean> fld)</code>	Creates a new boolean column - title (optional) - the column title - fld - the field definition
3	<code>col.booleanColumn(DRIExpression<Boolean></code> <code>ex),</code> <code>col.booleanColumn(String title,</code> <code>DRIExpression</code> <code><Boolean> expression)</code>	Creates a new boolean column - title (optional) - the column title - expression - the boolean value expression

Configuration options

#	method	description
1	<code>setComponentType(BooleanComponentType</code> <code>booleanComponentType)</code>	Sets the boolean presentation type. <code>BooleanComponentType.TEXT_*</code> - shows a text value <code>BooleanComponentType.IMAGE_*</code> - shows an image
2	<code>setImageDimension(Integer we,</code> <code>Integer ht),</code> <code>setImageWidth(Integer width),</code> <code>setImageHeight(Integer height)</code>	Sets the boolean image dimension. Has effect only when the boolean value is presented as an image

Examples

Quick usage:

```
1 report()
2 .columns(
3     col.booleanColumn("Boolean", "boolean"),
4     col.booleanColumn("Boolean", "boolean").setComponentType(BooleanComponentType.IMAGE_
5     ↪STYLE_1))
6 .setDataSource(...)
```

Another example: *BooleanColumnReport*

1.7.7 Component column

It is represented by a `ComponentColumnBuilder` instance and is used to display custom components (e.g. images or complex content) in columns.

Builders

#	method	description
1	<code>col.componentColumn(ComponentBuilder<?, ?> component),</code> <code>col.componentColumn(String title, ComponentBuilder<?, ?> component)</code>	Creates a new component column - title (optional) - the column title - component - the component definition

Examples

Quick usage:

```
1 ImageBuilder image = cmp.image(...);
2 ComponentBuilder<?, ?> component = ...;
3 report()
4 .columns(
5     col.componentColumn("Image", image),
6     col.componentColumn("Component", component))
7 .setDataSource(...)
```

Another example: *ComponentColumnReport*

Tags: column

1.8 Expression

Expressions are used to define various calculations, conditions, text field content, specific report groups, etc. Every expression can access the declared report fields, variables and other expressions and get their values to calculate the expression value. It must be an instance of `DRIExpression`.

1.8.1 Simple expression

The basic and the simplest implementation of an expression.

A simple expression must be an instance of `DRISimpleExpression` and implement the inherited abstract method `evaluate`.

```
public Object evaluate(ReportParameters reportParameters);
```

- `reportParameters` - access to report fields, variables, parameters, expressions, and other report values
- the method returns the result of the expression evaluation

Examples

Quick usage:

```
1 private class SimpleExpression extends AbstractSimpleExpression<BigDecimal> {
2     public BigDecimal evaluate(ReportParameters reportParameters) {
3         Integer quantity = reportParameters.getValue("quantity");
4         BigDecimal unitPrice = reportParameters.getValue("unitprice");
5         return new BigDecimal(quantity).multiply(unitPrice);
6     }
7 }
```

Another example: *SimpleExpressionReport*

1.8.2 Complex expression

The difference between a simple and complex expression is that a complex expression allows registering additional fields or variables that are not defined in the report and are needed for calculating the value.

A complex expression must be an instance of `DRComplexExpression` and must implement the inherited abstract method `evaluate`.

```
public Object evaluate(List<?> values, ReportParameters reportParameters);
```

- `values` - the values of the registered expressions
- `reportParameters` - access to report fields, variables, parameters, expressions, and other report values
- the method returns the result of the expression evaluation

Examples

Quick usage:

```
1 private class ComplexExpression extends AbstractComplexExpression<BigDecimal> {
2     public ComplexExpression() {
3         addExpression(field("quantity", Integer.class));
4         addExpression(field("unitprice", BigDecimal.class));
5     }
6
7     @Override
```

(continues on next page)

(continued from previous page)

```

8  public BigDecimal evaluate(List<?> values, ReportParameters reportParameters) {
9      Integer quantity = (Integer) values.get(0);
10     BigDecimal unitPrice = (BigDecimal) values.get(1);
11     return new BigDecimal(quantity).multiply(unitPrice);
12 }
13 }

```

Another example: *ComplexExpressionReport*

1.8.3 Jasper expression

This expression allows declaring an expression in a Jasper native syntax. Knowledge of the jasper syntax is also required for proper use.

Configuration options

#	method	description
1	exp.jasperSyntax(String expression, Class class), exp.jasperSyntax(String expression)	Creates a new jasper expression - expression - the jasper expression - class - the expression class
2	exp.jasperSyntaxText(String text)	Creates a new jasper string expression, useful only for showing a static text. This method escapes the characters in a String using Java String rules. - text - text to be shown

Examples

Quick usage:

```

1  JasperExpression<BigDecimal> columnExpression = exp.jasperSyntax("new BigDecimal($F
↪{quantity}).multiply($F{unitprice})", BigDecimal.class);
2  JasperExpression<String> columnTitle = exp.jasperSyntaxText("Price");
3  TextColumnBuilder<BigDecimal> priceColumn = col.column(columnExpression).
↪setTitle(columnTitle);

```

Another example: *JasperExpressionReport*

1.8.4 Value formatter

The purpose of this expression is to format a value only. For instance, when it is necessary to display a currency next to the value or just show a value in another format. It can be applied in any report column, group, subtotal, or text field component.

A value formatter expression must be an instance of `DRValueFormatter` and must implement the inherited abstract method `format`.

```
public Object format(Object value, ReportParameters reportParameters);
```

- value - the value to be formatted
- reportParameters - access to report fields, variables, parameters, expressions, and other report values
- the method returns the formatted value

Examples

Quick usage:

The code below converts a big decimal value to a string and adds a currency symbol next to the value.

```
1 TextColumnBuilder<BigDecimal> column = ...;
2 column.setValueFormatter(new ValueFormatter())
3 private class ValueFormatter extends AbstractValueFormatter<String, BigDecimal> {
4     public String format(BigDecimal value, ReportParameters reportParameters) {
5         return value + " EUR";
6     }
7 }
```

Another example: *ValueFormatterReport*

Tags: expression

1.9 Fonts

Font configuration is very important for a proper report generation and exports. In pdf exports in particular, where it is possible to embed fonts into a document. Keep in mind that if a font is not found, a default system font is used, and this may cause a lot of problems with the layout of the report. The font configuration itself is not very complicated, just follow these few steps:

Say, we want to register a font named *FreeUniversal*. First we need to create a font configuration file `fonts.xml` and add it in the root of the classpath. Using this configuration, we should register all fonts that are used in the reports. It's recommended to register all font styles (normal, bold, italic and bolditalic). The configuration file will look like this:

```
1 <fontFamilies>
2   <fontFamily name="FreeUniversal">
3     <normal>net/sf/dynamicreports/examples/fonts/FreeUniversal-regular.ttf</
↪ normal>
4     <bold>net/sf/dynamicreports/examples/fonts/FreeUniversal-Bold.ttf</bold>
5     <italic>net/sf/dynamicreports/examples/fonts/FreeUniversal-Italic.ttf</
↪ italic>
6     <boldItalic>net/sf/dynamicreports/examples/fonts/FreeUniversal-BoldItalic.ttf
↪ </boldItalic>
```

(continues on next page)

(continued from previous page)

```
7      <pdfEncoding>Cp1252</pdfEncoding>
8      <pdfEmbedded>true</pdfEmbedded>
9  </fontFamily>
10 </fontFamilies>
```

Now we have to add the font files to the classpath or pack them into a separate jar file and put the jar to the classpath. The font `net/sf/dynamicreports/examples/fonts/FreeUniversal-regular.ttf` means that the font file `FreeUniversal-regular.ttf` has to be found in package `net.sf.dynamicreports.examples.fonts`.

And finally, we need to create a `jasperreports_extension.properties` file and register the font configuration file:

```
1 net.sf.jasperreports.extension.registry.factory.simple.font.families=net.sf.
  ↳jasperreports.engine.fonts.SimpleFontExtensionsRegistryFactory
2 net.sf.jasperreports.extension.simple.font.families.drfonts=fonts.xml
```

After applying the configuration above, you can use the font in this way:

```
1 StyleBuilder plainStyle = stl.style()
2 .setFontName("FreeUniversal");
3 report().
4     title(cmp.text("text - plain").setStyle(plainStyle))
```

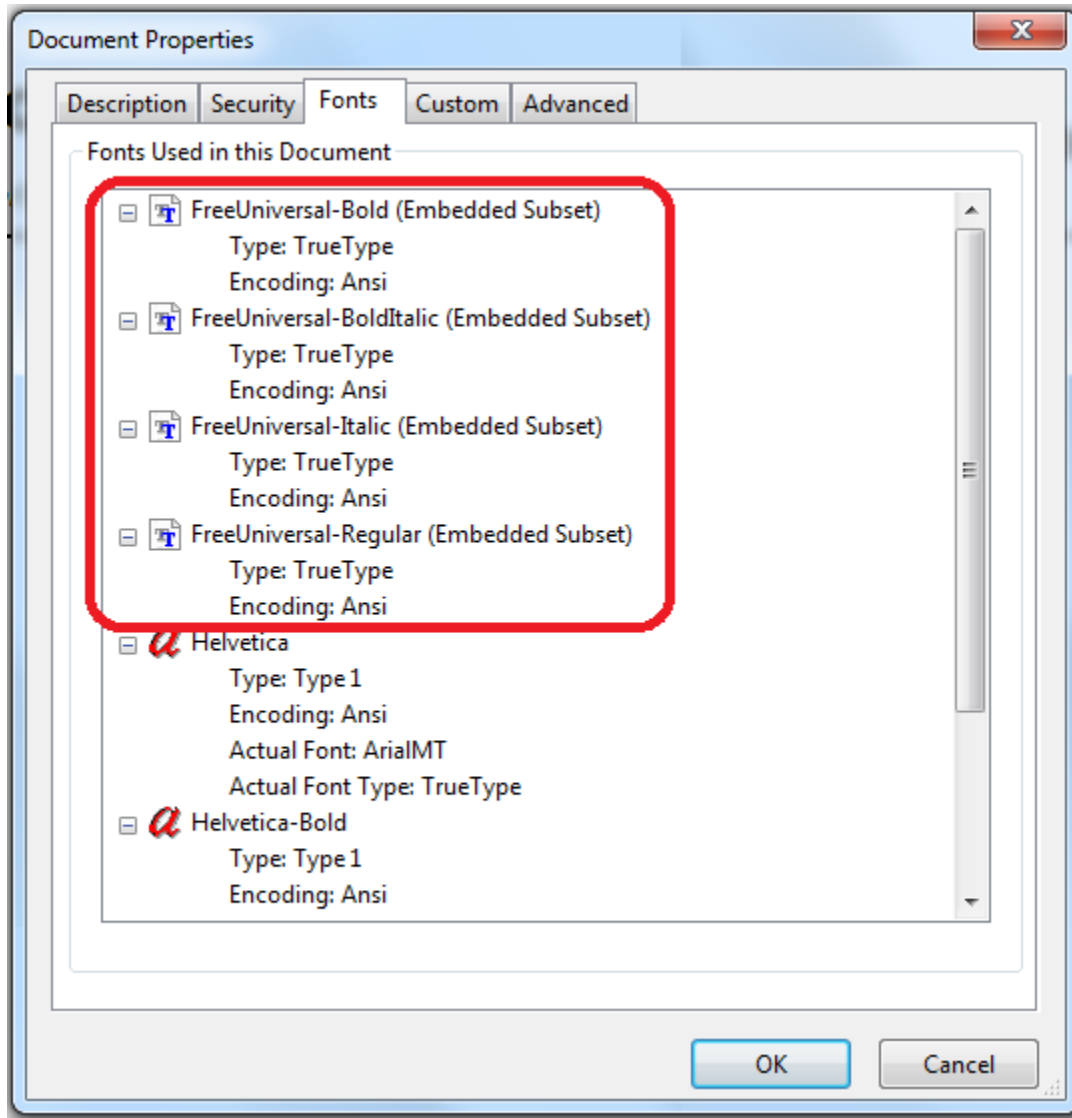
If you want to set the font `FreeUniversal` as a default font, create a `dynamicreports-defaults.xml` file and set the default font name:

```
1 <DynamicReports>
2 <!-- DEFAULT FONT -->
3     <font fontName="FreeUniversal" fontSize="10"/>
4 </DynamicReports>
```

If you don't set the style in a column or any other text component, the default font is used

```
1 report().
2     title(cmp.text("text - default font"))
```

You can check if the configuration is working properly by exporting the report into pdf. Open the pdf document and look at the document properties and select fonts, there should be a list of all embedded `FreeUniversal` fonts.



For more information about font configuration visit the following site: [Font extensions](#)

Examples

FontsReport

Tags: style

1.10 Servlets

Code below shows using the DynamicReports within a servlet. The servlet will display a pdf document on the web browser.

```

1  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
2  import java.io.IOException;
3  import java.io.OutputStream;
4  import javax.servlet.ServletException;

```

(continues on next page)

(continued from previous page)

```

5  import javax.servlet.http.HttpServlet;
6  import javax.servlet.http.HttpServletRequest;
7  import javax.servlet.http.HttpServletResponse;
8  import net.sf.dynamicreports.report.exception.DRException;
9
10 public class PdfReportServlet extends HttpServlet {
11     private static final long serialVersionUID = 1L;
12     @Override
13     protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
14     ↪ServletException, IOException {
15         resp.setContentType("application/pdf");
16         OutputStream out = resp.getOutputStream();
17         try {
18             report()
19             ...
20             .toPdf(out);
21         } catch (DRException e) {
22             throw new ServletException(e);
23         }
24         out.close();
25     }

```

1.11 Configuration

1.11.1 Default settings

DynamicReports automatically looks for a file named *dynamicreports-defaults.xml* on startup. In this file you can change the default settings (default font, pattern and horizontal alignment for all data types).

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <DynamicReports>
3      <!-- DEFAULT FONT -->
4      <font fontName="SansSerif" fontSize="10" pdfFontName="Helvetica" pdfEncoding=
5      ↪"Cp1252" pdfEmbedded="false"/>
6
7      <!-- DEFAULT DATA TYPES -->
8      <bigDecimalType pattern="#,##0.00#" horizontalAlignment="RIGHT"/>
9      <dateType pattern="MM/dd/yyyy" horizontalAlignment="RIGHT"/>
10     ...
11 </DynamicReports>

```

Copy this file in the root of the classpath.

1.11.2 Maven configuration

DynamicReports is synchronized with Maven Central Repository. Add the dependencies listed below to your pom.xml configuration file.

```

1  <dependency>
2      <groupId>net.sourceforge.dynamicreports</groupId>
3      <artifactId>dynamicreports-core</artifactId>

```

(continues on next page)

(continued from previous page)

```

4     <version>5.1.0</version>
5 </dependency>
6 <dependency>
7     <groupId>net.sourceforge.dynamicreports</groupId>
8     <artifactId>dynamicreports-adhoc</artifactId>
9     <version>5.1.0</version>
10 </dependency>
11 <dependency>
12     <groupId>net.sourceforge.dynamicreports</groupId>
13     <artifactId>dynamicreports-googlecharts</artifactId>
14     <version>5.1.0</version>
15 </dependency>

```

In some situations, maven may have a problem with downloading the itext-2.1.7.js6 dependency to your local maven repository. The problem may happen because the library itext-2.1.7.js6 is not available in maven central repository, it is only available through [JasperReports maven repository](#). If you are behind a firewall check if the firewall is not blocking that repository link or download the dependency and add it to your repository manually.

Maven snapshot repository configuration To use the development snapshots of DynamicReports, change your pom.xml configuration: just update the dynamicreports-core version

```

1 <dependency>
2     <groupId>net.sourceforge.dynamicreports</groupId>
3     <artifactId>dynamicreports-core</artifactId>
4     <version>5.1.1-SNAPSHOT</version>
5 </dependency>
6 <dependency>
7     <groupId>net.sourceforge.dynamicreports</groupId>
8     <artifactId>dynamicreports-adhoc</artifactId>
9     <version>5.1.1-SNAPSHOT</version>
10 </dependency>
11 <dependency>
12     <groupId>net.sourceforge.dynamicreports</groupId>
13     <artifactId>dynamicreports-googlecharts</artifactId>
14     <version>5.1.1-SNAPSHOT</version>
15 </dependency>

```

and add the Sonatype Nexus snapshot repository

```

1 <repositories>
2     <repository>
3         <id>sonatype-nexus-snapshots</id>
4         <name>Sonatype Nexus Snapshots</name>
5         <url>https://oss.sonatype.org/content/repositories/snapshots</url>
6         <releases>
7             <enabled>false</enabled>
8         </releases>
9         <snapshots>
10             <enabled>true</enabled>
11         </snapshots>
12     </repository>
13 </repositories>

```

1.11.3 dynamicreports-defaults.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <DynamicReports>
3      <!-- DEFAULT FONT -->
4      <!--
5      <font fontName="SansSerif" fontSize="10"/>
6      -->
7
8      <!-- DEFAULT DATA TYPES -->
9      <!--
10     <bigDecimalType pattern="#,##0.00#" horizontalAlignment="RIGHT"/>
11     <bigIntegerType pattern="#,##0" horizontalAlignment="RIGHT"/>
12     <byteType pattern="#,##0" horizontalAlignment="RIGHT"/>
13     <doubleType pattern="#,##0.#" horizontalAlignment="RIGHT"/>
14     <floatType pattern="#,##0.#" horizontalAlignment="RIGHT"/>
15     <integerType pattern="#,##0" horizontalAlignment="RIGHT"/>
16     <longType pattern="#,##0" horizontalAlignment="RIGHT"/>
17     <shortType pattern="#,##0" horizontalAlignment="RIGHT"/>
18     <dateType pattern="MM/dd/yyyy" horizontalAlignment="RIGHT"/>
19     <dateYearToMonthType pattern="MM/yyyy" horizontalAlignment="RIGHT"/>
20     <dateYearToHourType pattern="MM/dd/yyyy h a" horizontalAlignment="RIGHT"/>
21     <dateYearToMinuteType pattern="MM/dd/yyyy h:mm a" horizontalAlignment="RIGHT"/>
22     <dateYearToSecondType pattern="MM/dd/yyyy h:mm:ss a" horizontalAlignment="RIGHT"/>
23     <dateYearToFractionType pattern="MM/dd/yyyy h:mm:ss,SSS a" horizontalAlignment="
    ↪ "RIGHT"/>
24     <dateYearType pattern="yyyy" horizontalAlignment="RIGHT"/>
25     <dateMonthType pattern="MMMM" horizontalAlignment="RIGHT"/>
26     <dateDayType pattern="dd" horizontalAlignment="RIGHT"/>
27     <timeHourToMinuteType pattern="h:mm a" horizontalAlignment="RIGHT"/>
28     <timeHourToSecondType pattern="h:mm:ss a" horizontalAlignment="RIGHT"/>
29     <timeHourToFractionType pattern="h:mm:ss,SSS a" horizontalAlignment="RIGHT"/>
30     <percentageType pattern="#,##0.00%" horizontalAlignment="RIGHT"/>
31     <booleanType horizontalAlignment="CENTER"/>
32     <characterType horizontalAlignment="LEFT"/>
33     <stringType horizontalAlignment="LEFT"/>
34     -->
35
36     <!-- LOAD SYSTEM FONTS -->
37     <!--
38     <loadSystemFonts>true</loadSystemFonts>
39     -->
40 </DynamicReports>
```

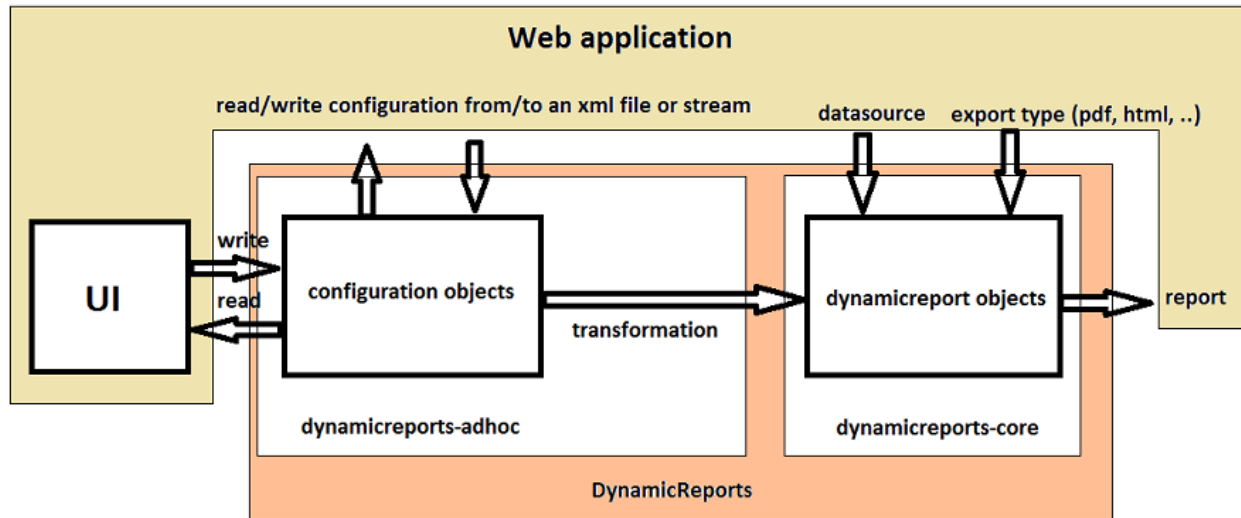
1.12 Adhoc Reports

Adhoc report applications allow users themselves to create specific reports and filter queries. A report is created on the fly, displaying values or charts that a user has specified via GUI. This project simplifies creating adhoc report applications. It allows to configure reports and filter queries and this configuration can be stored to an xml file or loaded from an xml file. The main configuration object is `AdhocConfiguration` that consists of `AdhocReport` and `AdhocFilter` object.

AdhocReport - a report configuration object that can be transformed to a dynamic report builder object.

AdhocFilter - a filter configuration object. This project doesn't generate sql queries from this object, it only allows you to store and load a filter configuration.

Another important class of this project is **AdhocManager**. It is a helper class that contains methods of loading/storing an AdhocConfiguration from/to a stream and methods of creating a report builder instance from a given AdhocReport object.



Examples

Quick usage:

```
1 AdhocConfiguration configuration = new AdhocConfiguration();
2 AdhocReport report = new AdhocReport();
3 configuration.setReport(report);
4 AdhocColumn column = new AdhocColumn();
5 column.setName("item");
6 report.addColumn(column);
```

The following code stores a configuration to an output stream

```
1 AdhocManager.saveConfiguration(configuration, outputStream);
```

The following code loads a configuration from an input stream

```
1 AdhocConfiguration configuration = AdhocManager.loadConfiguration(configuration,
↪ inputStream);
```

The following code converts a configuration to a report builder instance

```
1 JasperReportBuilder reportBuilder = AdhocManager.createReport(configuration.
↪ getReport());
2 reportBuilder.setDatasource(...);
```

Tags: `adhoc`

1.13 Features

- **Columns** - field column, expression column, percentage column, report row number column, page row number column, column row number column, component column, boolean column
- **Column grid** - for arranging columns layout
- **Groups** - field group, custom group, column group
- **Subtotals** - at title, page header/footer, column header/footer, group header/footer, first group header/footer, last group header/footer, last page footer, summary aggregation subtotal, custom subtotal, percentage subtotal
- **Components**
 - **Containers** - can hold components horizontal list, horizontal flow list, vertical list, multi page list
 - **Text fields** text field, current date field, page x of y field, page x slashed y field, page number field, total pages field
 - **Charts** area, stacked area, bar, layered bar, bar 3D, stacked bar, grouped stacked bar, stacked bar 3D, waterfall bar, line, pie, pie 3D, time series, difference, scatter, xy area, xy bar, xy line, xy step, spider, multi axis, xy block, bubble, candlestick, highlow, meter, thermometer, gantt
 - **Barcodes** *barcode4j* - codabar, code128, ean128, dataMatrix, code39, interleaved2Of5, upca, upce, ean13, ean8, uspsIntelligentMail, royalMailCustomer, postnet, pdf417 *barbecue* - 2of7, 3of9, bookland, codabar, code128, code128A, code128B, code128C, code39, code39 (Extended), ean128, ean13, globalTradeItem-Number, int2of5, monarch, nw7, pdf417, postNet, randomWeightUpca, scc14ShippingCode, shipmentIdentificationNumber, ssc18, std2of5, ucc128, upca, usd3, usd4, usps
 - **Other components** crosstab, subreports, map, image, line, filler, page break, column break, generic element, ellipse, rectangle
 - **Google charts** geo map
- **Report bands** - can hold components title, page header/footer, column header/footer, detail, detail header/footer, last page footer, summary, no data, background
- **Row highlighters** - conditional highlighter for highlighting detail rows
- **Styles** style, conditional style
- **Table of contents** - a table of contents generator
- **Others** concatenated reports, adhoc reporting, jrxml template design, sorting, **data types**, **expressions**, report template, template styles, parameters, fields, variables, scriptlets, data filters, ...

1.14 Requirements

1.14.1 Required libraries

dynamicreports
java 1.7 or above
jasperreports 6.5.1
commons-beanutils 1.9.3
commons-logging 1.1.1
commons-collections 3.2.2
commons-digester 2.1
commons-code 1.10

xml-apis 1.3.04
xml-apis-ext 1.3.04
ecj 4.4.2
jfreechart 1.0.19
jcommon 1.0.23
commons-lang 2.6
commons-lang3 3.1
jackson-core 2.1.4
jackson-databind 2.1.4
jackson-annotations 2.1.4
log4j 1.2.14
velocity 1.7

1.14.2 Required libraries for barcodes

barcode4j 2.0
barbecue 1.5-beta1
core 3.2.1
avalon-framework-impl 4.2.0
batik-anim 1.9
batik-awt-util 1.9
batik-bridge 1.9
batik-css 1.9
batik-dom 1.9
batik-ext 1.9
batik-gvt-1.9
batik-parser 1.9
batik-script 1.9
batik-svg-dom 1.9
batik-util 1.9
batik-xml 1.9
batik-i18n 1.9

1.14.3 Required libraries for exporting

PDF itext 2.1.7.js6
Excel poi 3.15

1.14.4 Optional

mlgraphics-commons 2.1
commons-collections4 4.1
bcprov-jdk15on 1.52
spring-core 3.2.18.RELEASE
spring-beans 3.2.18.RELEASE

hsqldb 1.8.0.10

xalan 2.7.2

serializer 2.7.2

1.15 Links & Resources

1.15.1 JasperReports documentation

JasperReports ultimate guide, legacy 3.0 version.

JasperStudio IDE Download

JasperStudio User guide

JasperReports Sample reference

JasperReports Configuration Reference

1.15.2 Ivan Lagunov's blog

DynamicReports and Spring MVC integration

DynamicReports and Cocoon integration

1.16 Adhoc

1.16.1 Adhoc Customizer

DynamicReports		AdhocCustomizer
 http://www.dynamicreports.org		
	Quantity	Unit price
Book		
	5	48.528
	9	34.258
	9	83.115
	4	32.754
	8	100.043
	7	16.088
	3	19.58
	8	24.563
	10	97.834
	9	87.248
	4	35.896
	3	89.088
	10	52.544
	8	41.508
	2	66.076
PDA		
	3	39.339
	5	69.458
	8	61.178
	9	17.146
	5	97.778
	3	86.814
	3	73.59
	4	22.153
	6	60.298
	8	11.158
	9	67.992
	4	5.126
	3	39.563
	3	37.604
	4	66.996
	7	59.63
	3	94.33
	3	67.909
	7	71.656
	10	65.888
	35	1.944.78
1 of 1		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.adhoc;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.Date;
29
30  import net.sf.dynamicreports.adhoc.AdhocManager;
31  import net.sf.dynamicreports.adhoc.configuration.AdhocCalculation;
32  import net.sf.dynamicreports.adhoc.configuration.AdhocColumn;
33  import net.sf.dynamicreports.adhoc.configuration.AdhocConfiguration;
34  import net.sf.dynamicreports.adhoc.configuration.AdhocGroup;
35  import net.sf.dynamicreports.adhoc.configuration.AdhocReport;
36  import net.sf.dynamicreports.adhoc.configuration.AdhocSort;
37  import net.sf.dynamicreports.adhoc.configuration.AdhocSubtotal;
38  import net.sf.dynamicreports.adhoc.report.DefaultAdhocReportCustomizer;
39  import net.sf.dynamicreports.examples.Templates;
40  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
41  import net.sf.dynamicreports.report.builder.ReportBuilder;
42  import net.sf.dynamicreports.report.datasource.DRDataSource;
43  import net.sf.dynamicreports.report.definition.datatype.DRIDataType;
44  import net.sf.dynamicreports.report.exception.DRException;
45  import net.sf.jasperreports.engine.JRDataSource;
46
47  /**
48   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
49   */
50  public class AdhocCustomizerReport {
51
52      public AdhocCustomizerReport() {
53          build();
54      }
55
56      private void build() {
57          AdhocConfiguration configuration = new AdhocConfiguration();

```

(continues on next page)

(continued from previous page)

```

58     AdhocReport report = new AdhocReport();
59     configuration.setReport(report);
60
61     // columns
62     AdhocColumn column = new AdhocColumn();
63     column.setName("quantity");
64     report.addColumn(column);
65     column = new AdhocColumn();
66     column.setName("unitprice");
67     report.addColumn(column);
68     // groups
69     AdhocGroup group = new AdhocGroup();
70     group.setName("item");
71     report.addGroup(group);
72     // subtotal
73     AdhocSubtotal subtotal = new AdhocSubtotal();
74     subtotal.setName("quantity");
75     subtotal.setCalculation(AdhocCalculation.COUNT);
76     report.addSubtotal(subtotal);
77     subtotal = new AdhocSubtotal();
78     subtotal.setCalculation(AdhocCalculation.SUM);
79     subtotal.setName("unitprice");
80     report.addSubtotal(subtotal);
81     // sorts
82     AdhocSort sort = new AdhocSort();
83     sort.setName("item");
84     report.addSort(sort);
85
86     try {
87         JasperReportBuilder reportBuilder = AdhocManager.
88 ↪ createReport(configuration.getReport(), new ReportCustomizer());
89         reportBuilder.setDataSource(createDataSource());
90         reportBuilder.show();
91     } catch (DRException e) {
92         e.printStackTrace();
93     }
94
95     private JRDataSource createDataSource() {
96         DRDataSource dataSource = new DRDataSource("item", "orderdate",
97 ↪ "quantity", "unitprice");
98         for (int i = 0; i < 15; i++) {
99             dataSource.add("Book", new Date(), (int) (Math.random() * 10)
100 ↪ + 1, new BigDecimal(Math.random() * 100 + 1));
101         }
102         for (int i = 0; i < 20; i++) {
103             dataSource.add("PDA", new Date(), (int) (Math.random() * 10)
104 ↪ + 1, new BigDecimal(Math.random() * 100 + 1));
105         }
106         return dataSource;
107     }
108
109     public static void main(String[] args) {
110         new AdhocCustomizerReport();
111
112     private class ReportCustomizer extends DefaultAdhocReportCustomizer {

```

(continues on next page)

(continued from previous page)

```

111      /**
112       * If you want to add some fixed content to a report that is not
113       ↪needed to store in the xml file.
114       * For example you can add default page header, footer, default fonts,
115       ↪...
116       */
117       @Override
118       public void customize(ReportBuilder<?> report, AdhocReport
119       ↪adhocReport) throws DRException {
120           super.customize(report, adhocReport);
121           // default report values
122           report.setTemplate(Templates.reportTemplate);
123           report.title(Templates.createTitleComponent("AdhocCustomizer
124       ↪"));
125           // a fixed page footer that user cannot change, this
126       ↪customization is not stored in the xml file
127           report.pageFooter(Templates.footerComponent);
128       }
129
130      /**
131       * This method returns a field type from a given field name.
132       */
133      @Override
134      protected DRIDataType<?, ?> getFieldTypes(String name) {
135          if (name.equals("item")) {
136              return type.stringType();
137          }
138          if (name.equals("orderdate")) {
139              return type.dateType();
140          }
141          if (name.equals("quantity")) {
142              return type.integerType();
143          }
144          if (name.equals("unitprice")) {
145              return type.bigDecimalType();
146          }
147          return super.getFieldTypes(name);
148      }
149
150      /**
151       * If a user doesn't specify a column title, getColumnTitle is
152       ↪evaluated and the return value is used as a column title.
153       */
154      @Override
155      protected String getFieldLabel(String name) {
156          if (name.equals("item")) {
157              return "Item";
158          }
159          if (name.equals("orderdate")) {
160              return "Order date";
161          }
162          if (name.equals("quantity")) {
163              return "Quantity";
164          }
165          if (name.equals("unitprice")) {
166              return "Unit price";

```

(continues on next page)

(continued from previous page)

```

162         }
163         return name;
164     }
165 }
166 }
167 }

```

1.16.2 Simple Adhoc

Item	quantity
Book	8
Book	7
Book	3
Book	10
Book	6
Book	1
Book	6
Book	9
Book	8
Book	1
Book	4
Book	8
Book	5
Book	10
Book	6
Book	3
Book	1
Book	5
Book	7
Book	8

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see http://www.gnu.org/licenses/.

```

(continues on next page)

(continued from previous page)

```

21  */
22
23  package net.sf.dynamicreports.examples.adhoc;
24
25  import java.io.FileInputStream;
26  import java.io.FileNotFoundException;
27  import java.io.FileOutputStream;
28  import java.math.BigDecimal;
29  import java.util.Date;
30
31  import net.sf.dynamicreports.adhoc.AdhocManager;
32  import net.sf.dynamicreports.adhoc.configuration.AdhocColumn;
33  import net.sf.dynamicreports.adhoc.configuration.AdhocConfiguration;
34  import net.sf.dynamicreports.adhoc.configuration.AdhocReport;
35  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
36  import net.sf.dynamicreports.report.datasource.DRDataSource;
37  import net.sf.dynamicreports.report.exception.DRException;
38  import net.sf.jasperreports.engine.JRDataSource;
39
40  /**
41   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42   */
43  public class SimpleAdhocReport {
44
45      public SimpleAdhocReport() {
46          build();
47      }
48
49      private void build() {
50          AdhocConfiguration configuration = new AdhocConfiguration();
51          AdhocReport report = new AdhocReport();
52          configuration.setReport(report);
53
54          AdhocColumn column = new AdhocColumn();
55          column.setName("item");
56          report.addColumn(column);
57
58          column = new AdhocColumn();
59          column.setName("quantity");
60          report.addColumn(column);
61
62          try {
63              // The following code stores the configuration to an xml file
64              AdhocManager.saveConfiguration(configuration, new
65 ↪FileOutputStream("c:/temp/configuration.xml"));
66              @SuppressWarnings("unused")
67              // The following code loads a configuration from an xml file
68              AdhocConfiguration loadedConfiguration = AdhocManager.
69 ↪loadConfiguration(new FileInputStream("c:/temp/configuration.xml"));
70
71              JasperReportBuilder reportBuilder = AdhocManager.
72 ↪createReport(configuration.getReport());
73              reportBuilder.setDataSource(createDataSource());
74              reportBuilder.show();
75          } catch (DRException e) {
76              e.printStackTrace();
77          } catch (FileNotFoundException e) {



```

(continues on next page)

(continued from previous page)

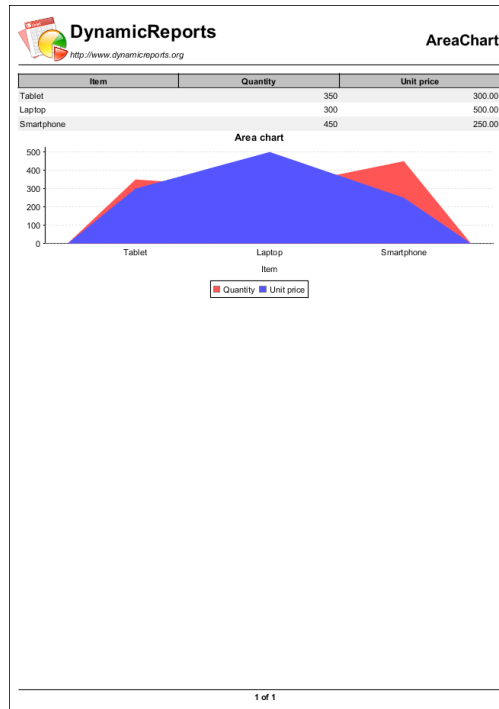
```
75         e.printStackTrace();
76     }
77 }
78
79 private JRDataSource createDataSource() {
80     DRDataSource dataSource = new DRDataSource("item", "orderdate",
81 ↪ "quantity", "unitprice");
82     for (int i = 0; i < 20; i++) {
83         dataSource.add("Book", new Date(), (int) (Math.random() * 10) ↪
84 ↪ + 1, new BigDecimal(Math.random() * 100 + 1));
85     }
86     return dataSource;
87 }
88
89 public static void main(String[] args) {
90     new SimpleAdhocReport();
91 }
```

Table 1: Adhoc Examples

	
AdhocCustomizer	SimpleAdhoc

1.17 Chart

1.17.1 Area Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;

```

(continues on next page)

(continued from previous page)

```

28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class AreaChartReport {
40
41     public AreaChartReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
49         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
50         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
51
52         try {
53             report()
54                 .setTemplate(Templates.reportTemplate)
55                 .columns(itemColumn, quantityColumn,
↪unitPriceColumn)
56                 .title(Templates.createTitleComponent(
↪"AreaChart"))
57                 .summary(
58                     cht.areaChart()
59                         .setTitle(
↪"Area chart")
60                         .
↪setTitleFont(boldFont)
61                         .
↪setCategory(itemColumn)
62                         .series(
↪serie(quantityColumn), cht.serie(unitPriceColumn))
63                         .
↪setCategoryAxisFormat(
↪axisFormat().setLabel("Item")))
64                 .pageFooter(Templates.footerComponent)
65                 .setDataSource(createDataSource())
66                 .show();
67         } catch (DRException e) {
68             e.printStackTrace();
69         }
70     }
71 }
72
73

```

(continues on next page)

(continued from previous page)

```

74     private JRDataSource createDataSource() {
75         DRDataSource dataSource = new DRDataSource("item", "quantity",
76         ↪ "unitprice");
77         dataSource.add("Tablet", 350, new BigDecimal(300));
78         dataSource.add("Laptop", 300, new BigDecimal(500));
79         dataSource.add("Smartphone", 450, new BigDecimal(250));
80         return dataSource;
81     }
82     public static void main(String[] args) {
83         new AreaChartReport();
84     }
85 }

```

1.17.2 Bar 3D Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *

```

(continues on next page)

(continued from previous page)

```

14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.builder.style.FontBuilder;
32  import net.sf.dynamicreports.report.datasource.DRDataSource;
33  import net.sf.dynamicreports.report.exception.DRException;
34  import net.sf.jasperreports.engine.JRDataSource;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class Bar3DChartReport {
40
41      public Bar3DChartReport() {
42          build();
43      }
44
45      private void build() {
46          FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳type.stringType());
49          TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳"quantity", type.integerType());
50          TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳", "unitprice", type.bigDecimalType());
51
52          try {
53              report()
54                  .setTemplate(Templates.reportTemplate)
55                  .columns(itemColumn, quantityColumn,
↳unitPriceColumn)
56                  .title(Templates.createTitleComponent(
↳"Bar3DChart"))
57                  .summary(
58                      cht.bar3DChart()
59                          .setTitle(
↳"Bar 3D chart")
60                          .
61                          .
62                          .series(

```

(continues on next page)

(continued from previous page)

```

63  cht.  

64  ↪serie(quantityColumn), cht.serie(unitPriceColumn))  

65  

66  ↪setCategoryAxisFormat ( cht.  

67  ↪axisFormat().setLabel("Item"))  

68  

69  .pageFooter(Templates.footerComponent)  

70  .setDataSource(createDataSource())  

71  .show();  

72  } catch (DRException e) {  

73  e.printStackTrace();  

74  }  

75  

76  private JRDataSource createDataSource() {  

77  DRDataSource dataSource = new DRDataSource("item", "quantity",  

78  ↪"unitprice");  

79  dataSource.add("Tablet", 350, new BigDecimal(300));  

80  dataSource.add("Laptop", 300, new BigDecimal(500));  

81  dataSource.add("Smartphone", 450, new BigDecimal(250));  

82  return dataSource;  

83  }  

84  

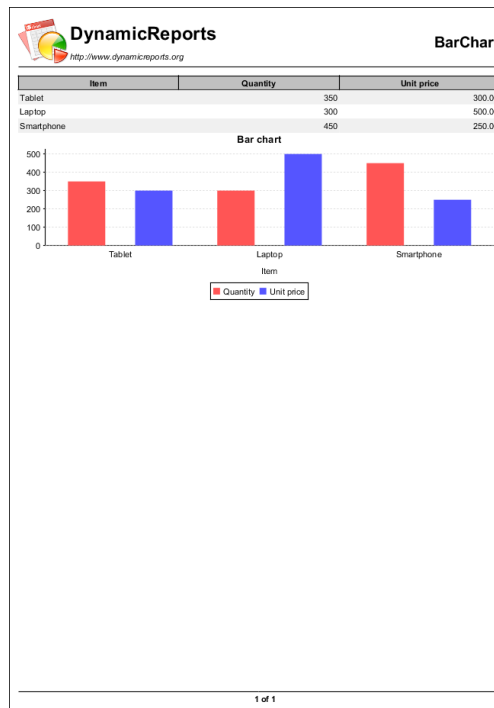
85  public static void main(String[] args) {  

86  new Bar3DChartReport();  

87  }  


```

1.17.3 Bar Chart



```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class BarChartReport {
40
41     public BarChartReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49 ↪type.stringType());
50         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
51 ↪"quantity", type.integerType());
52         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
53 ↪", "unitprice", type.bigDecimalType());
54
55         try {
56             report()
57                 .setTemplate(Templates.reportTemplate)
```

(continues on next page)

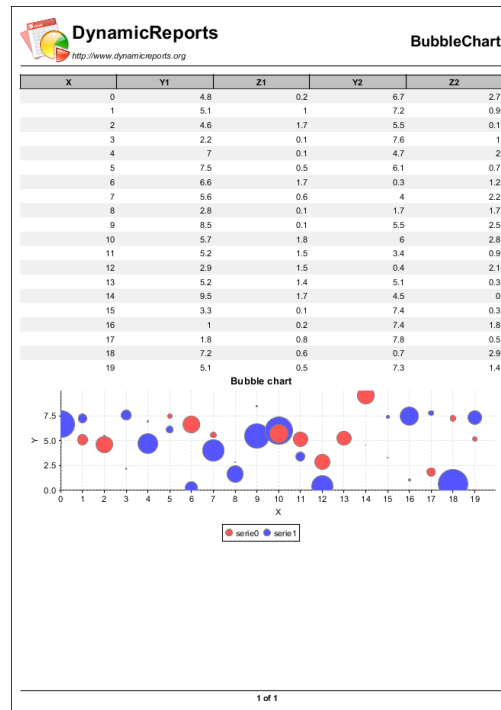
(continued from previous page)

```

55         .columns(itemColumn, quantityColumn, unitPriceColumn)
56         .title(Templates.createTitleComponent(
57             "BarChart"))
58         .summary(
59             cht.barChart()
60                 .setTitle(
61                     "Bar chart")
62                     .setTitleFont(boldFont)
63                     .setCategory(itemColumn)
64                     .series(
65                         serie(quantityColumn), cht.serie(unitPriceColumn))
66                     .setCategoryAxisFormat(
67                         axisFormat().setLabel("Item")))
68         .pageFooter(Templates.footerComponent)
69         .setDataSource(createDataSource())
70         .show();
71     } catch (DRException e) {
72         e.printStackTrace();
73     }
74 }
75
76 private JRDataSource createDataSource() {
77     DRDataSource dataSource = new DRDataSource("item", "quantity",
78         "unitprice");
79     dataSource.add("Tablet", 350, new BigDecimal(300));
80     dataSource.add("Laptop", 300, new BigDecimal(500));
81     dataSource.add("Smartphone", 450, new BigDecimal(250));
82     return dataSource;
83 }
84
85 public static void main(String[] args) {
86     new BarChartReport();
87 }

```

1.17.4 Bubble Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class BubbleChartReport {
37
38     public BubbleChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<Double> xColumn = col.column("X", "x", type.
↪doubleType());
46         TextColumnBuilder<Double> y1Column = col.column("Y1", "y1", type.
↪doubleType());
47         TextColumnBuilder<Double> z1Column = col.column("Z1", "z1", type.
↪doubleType());
48         TextColumnBuilder<Double> y2Column = col.column("Y2", "y2", type.
↪doubleType());
49         TextColumnBuilder<Double> z2Column = col.column("Z2", "z2", type.
↪doubleType());
50
51         try {
52             report()
53                 .setTemplate(Templates.reportTemplate)
54                 .columns(xColumn, y1Column, z1Column,
↪y2Column, z2Column)
55                 .title(Templates.createTitleComponent(
↪"BubbleChart"))
56                 .summary(
57                     cht.bubbleChart()
58                         .setTitle(
↪"Bubble chart")
59                         .
↪setTitleFont(boldFont)
60                         .
↪setXValue(xColumn)
61                         .series(
62                             cht.
↪xyzSerie().setYValue(y1Column).setZValue(z1Column),
63                             cht.
↪xyzSerie().setYValue(y2Column).setZValue(z2Column))
64                         .
↪setXAxisFormat(
65                             cht.
↪axisFormat().setLabel("X"))
66                         .
↪setYAxisFormat(
67                             cht.
↪axisFormat().setLabel("Y")))
68                 .pageFooter(Templates.footerComponent)
69                 .setDataSource(createDataSource())
70                 .show();
71         } catch (DRException e) {

```

(continues on next page)

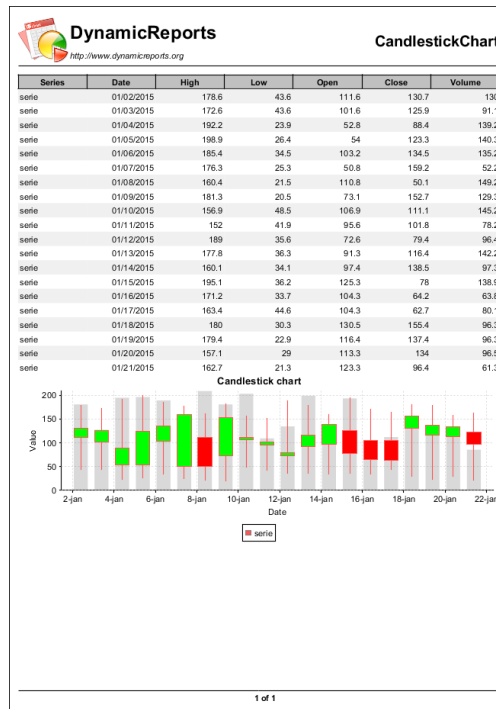
(continued from previous page)

```

72         e.printStackTrace();
73     }
74 }
75
76 private JRDataSource createDataSource() {
77     DRDataSource dataSource = new DRDataSource("x", "y1", "z1", "y2", "z2
↪");
78     for (int i = 0; i < 20; i++) {
79         dataSource.add((double) i, Math.random() * 10, Math.random()
↪
↪ * 2, Math.random() * 8, Math.random() * 3);
80     }
81     return dataSource;
82 }
83
84 public static void main(String[] args) {
85     new BubbleChartReport();
86 }
87 }

```

1.17.5 Candlestick Chart



```

1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *

```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.Calendar;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.style.FontBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class CandlestickChartReport {
41
42     public CandlestickChartReport() {
43         build();
44     }
45
46     private void build() {
47         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
48
49         TextColumnBuilder<String> seriesColumn = col.column("Series", "series
50 ↪", type.stringType());
51         TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
52 ↪dateType());
53         TextColumnBuilder<Double> highColumn = col.column("High", "high",
54 ↪type.doubleType());
55         TextColumnBuilder<Double> lowColumn = col.column("Low", "low", type.
56 ↪doubleType());
57         TextColumnBuilder<Double> openColumn = col.column("Open", "open",
58 ↪type.doubleType());
59         TextColumnBuilder<Double> closeColumn = col.column("Close", "close",
60 ↪type.doubleType());
61         TextColumnBuilder<Double> volumeColumn = col.column("Volume", "volume
62 ↪", type.doubleType());
63
64         try {
65             report()

```

(continues on next page)

(continued from previous page)

```

59         .setTemplate(Templates.reportTemplate)
60         .columns(seriesColumn, dateColumn, highColumn,
↪ lowColumn, openColumn, closeColumn, volumeColumn)
61         .title(Templates.createTitleComponent(
↪ "CandlestickChart"))
62         .summary(
63             cht.candlestickChart()
64                 .setTitle(
↪ "Candlestick chart")
65                 .
66                 .setTitleFont(boldFont)
67                 .
68                 .setSeries(seriesColumn)
69                 .
70                 .setDate(dateColumn)
71                 .
72                 .setHigh(highColumn)
73                 .
74                 .setLow(lowColumn)
75                 .
76                 .setOpen(openColumn)
77                 .
78                 .setClose(closeColumn)
79                 .
80                 .setVolume(volumeColumn)
81                 .
82                 .setTimeAxisFormat(
83                 cht.
84                 ↪ axisFormat().setLabel("Date"))
85                 .
86                 .setValueAxisFormat(
87                 cht.
88                 ↪ axisFormat().setLabel("Value")))
89         .pageFooter(Templates.footerComponent)
90         .setDataSource(createDataSource())
91         .show();
92     } catch (DRException e) {
93         e.printStackTrace();
94     }
95 }
96
97 private JRDataSource createDataSource() {
98     DRDataSource dataSource = new DRDataSource("series", "date", "high",
↪ "low", "open", "close", "volume");
99     Calendar c = Calendar.getInstance();
100     c.add(Calendar.DAY_OF_MONTH, -20);
101     for (int i = 0; i < 20; i++) {
102         dataSource.add("serie", c.getTime(), 150 + Math.random() * 50,
↪ 20 + Math.random() * 30, 50 + Math.random() * 90, 50 + Math.random() * 110,
103             50 + Math.random() * 100);
104         c.add(Calendar.DAY_OF_MONTH, 1);
105     }
106     return dataSource;
107 }
108
109 public static void main(String[] args) {
110     new CandlestickChartReport();

```

(continues on next page)

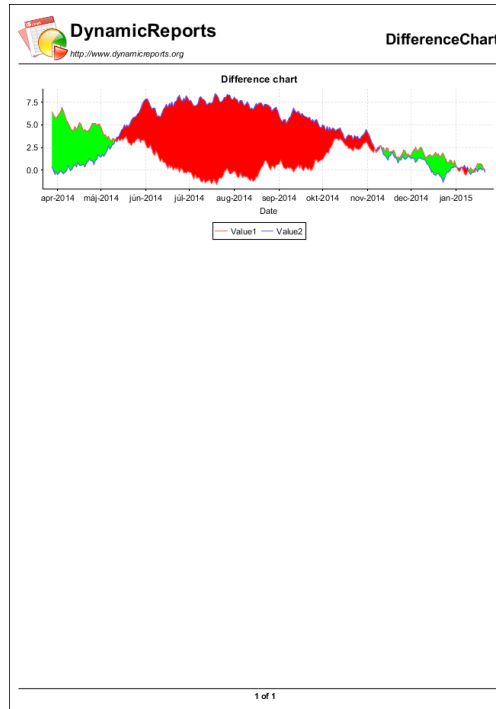
(continued from previous page)

```

99     }
100 }

```

1.17.6 Difference Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24

```

(continues on next page)

(continued from previous page)

```

25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.Calendar;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.DynamicReports;
32 import net.sf.dynamicreports.report.builder.style.FontBuilder;
33 import net.sf.dynamicreports.report.constant.TimePeriod;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class DifferenceChartReport {
42
43     public DifferenceChartReport() {
44         build();
45     }
46
47     private void build() {
48         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
49
50         try {
51             report()
52                 .setTemplate(Templates.reportTemplate)
53                 .title(Templates.createTitleComponent(
54                     ↪ "DifferenceChart"))
55                 .summary(
56                     ↪ cht.differenceChart()
57                         .setTitle(
58                             ↪ "Difference chart")
59                             .
60                             ↪ setTitleFont(boldFont)
61                             .
62                             ↪ setTimePeriod(DynamicReports.<Date>field("date", type.dateType()))
63                             .
64                             ↪ setTimePeriodType(TimePeriod.DAY)
65                             .series(
66                                 ↪ serie(DynamicReports.<Number>field("value1", type.doubleType())).setLabel("Value1"),
67                                 ↪ serie(DynamicReports.<Number>field("value2", type.doubleType())).setLabel("Value2"))
68                             .
69                             ↪ setTimeAxisFormat(
70                                 ↪ axisFormat().setLabel("Date")))
71                 .pageFooter(Templates.footerComponent)
72                 .setDataSource(createDataSource())
73                 .show();
74         } catch (DRException e) {
75             e.printStackTrace();
76         }
77     }
78 }

```

cht.
cht.
cht.

(continues on next page)

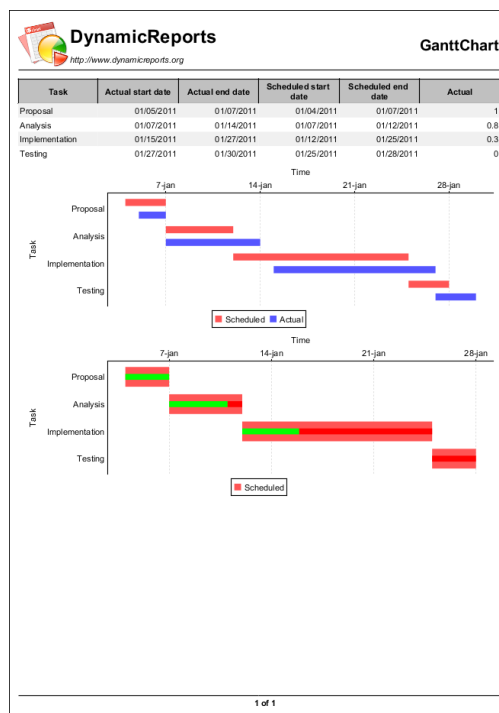
(continued from previous page)

```

73     private JRDataSource createDataSource() {
74         DRDataSource dataSource = new DRDataSource("date", "value1", "value2
↪");
75         double value1 = 0;
76         double value2 = 0;
77         Calendar c = Calendar.getInstance();
78         for (int i = 0; i < 300; i++) {
79             c.add(Calendar.DAY_OF_MONTH, -1);
80             value1 += Math.random() - 0.5;
81             value2 += Math.random() - 0.5;
82             dataSource.add(c.getTime(), value1, value2);
83         }
84         return dataSource;
85     }
86
87     public static void main(String[] args) {
88         new DifferenceChartReport();
89     }
90 }

```

1.17.7 Gantt Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.

```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.Calendar;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.chart.GanttChartBuilder;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class GanttChartReport {
41
42     public GanttChartReport() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> taskColumn = col.column("Task", "task",
48 ↪ type.stringType());
49         TextColumnBuilder<Date> actualStartDateColumn = col.column("Actual
50 ↪ start date", "actualstartdate", type.dateType());
51         TextColumnBuilder<Date> actualEndDateColumn = col.column("Actual end
52 ↪ date", "actualenddate", type.dateType());
53         TextColumnBuilder<Date> scheduledStartDateColumn = col.column(
54 ↪ "Scheduled start date", "scheduledstartdate", type.dateType());
55         TextColumnBuilder<Date> scheduledEndDateColumn = col.column(
56 ↪ "Scheduled end date", "scheduledenddate", type.dateType());
57         TextColumnBuilder<Double> actualColumn = col.column("Actual", "actual
58 ↪ ", type.doubleType());
59
60         GanttChartBuilder chart1 = cht.ganttChart()
61             .setTask(taskColumn)
62             .series(
63                 cht.ganttSerie()
64                 .setStartDate(scheduledStartDateColumn)

```

(continues on next page)

(continued from previous page)

```

59         ↪setEndDate(scheduledEndDateColumn)
60                                     .setLabel("Scheduled
61         ↪"),
62                                     cht.ganttSerie()
63                                     .
64         ↪setStartDate(actualStartDateColumn)
65                                     .
66         ↪setEndDate(actualEndDateColumn)
67                                     .setLabel("Actual"))
68                                     .setTimeAxisFormat(
69                                     cht.axisFormat().setLabel("Time"))
70                                     .setTaskAxisFormat(
71                                     cht.axisFormat().setLabel("Task"));
72
73     GanttChartBuilder chart2 = cht.ganttChart()
74         .setTask(taskColumn)
75         .series(
76             cht.ganttSerie()
77             .
78             ↪setStartDate(scheduledStartDateColumn)
79             .
80             ↪setEndDate(scheduledEndDateColumn)
81             .
82             ↪setPercent(actualColumn)
83             .setLabel("Scheduled
84             ↪"))
85             .setTimeAxisFormat(
86             cht.axisFormat().setLabel("Time"))
87             .setTaskAxisFormat(
88             cht.axisFormat().setLabel("Task"));
89
90     try {
91         report()
92             .setTemplate(Templates.reportTemplate)
93             .columns(taskColumn, actualStartDateColumn, ↪
94             ↪actualEndDateColumn, scheduledStartDateColumn, scheduledEndDateColumn, actualColumn)
95             .title(Templates.createTitleComponent(
96             ↪"GanttChart"))
97             .summary(chart1, chart2)
98             .pageFooter(Templates.footerComponent)
99             .setDataSource(createDataSource())
100             .show();
101     } catch (DRException e) {
102         e.printStackTrace();
103     }
104
105     private JRDataSource createDataSource() {
106         DRDataSource dataSource = new DRDataSource("task", "actualstartdate",
107             ↪"actualenddate", "scheduledstartdate", "scheduledenddate", "actual");
108         dataSource.add("Proposal", toDate(2011, 1, 5), toDate(2011, 1, 7), ↪
109             ↪toDate(2011, 1, 4), toDate(2011, 1, 7), 1d);
110         dataSource.add("Analysis", toDate(2011, 1, 7), toDate(2011, 1, 14), ↪
111             ↪toDate(2011, 1, 7), toDate(2011, 1, 12), 0.8d);
112         dataSource.add("Implementation", toDate(2011, 1, 15), toDate(2011, 1, ↪
113             ↪27), toDate(2011, 1, 12), toDate(2011, 1, 25), 0.3d);

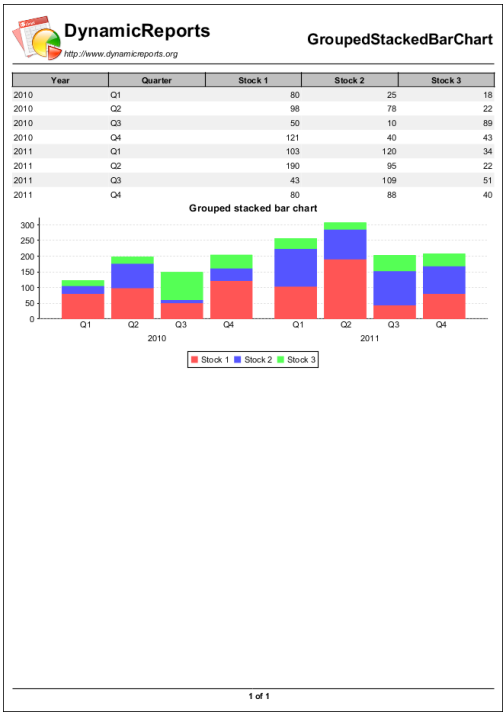
```

(continues on next page)

(continued from previous page)

```
102         dataSource.add("Testing", toDate(2011, 1, 27), toDate(2011, 1, 30),
103         ↪toDate(2011, 1, 25), toDate(2011, 1, 28), 0d);
104         return dataSource;
105     }
106
107     private Date toDate(int year, int month, int day) {
108         Calendar c = Calendar.getInstance();
109         c.clear();
110         c.set(Calendar.YEAR, year);
111         c.set(Calendar.MONTH, month - 1);
112         c.set(Calendar.DAY_OF_MONTH, day);
113         return c.getTime();
114     }
115
116     public static void main(String[] args) {
117         new GanttChartReport();
118     }
```

1.17.8 Grouped Stacked Bar Chart



```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class GroupedStackedBarChartReport {
37
38     public GroupedStackedBarChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<String> yearColumn = col.column("Year", "year",
↪ type.stringType());
46         TextColumnBuilder<String> quarterColumn = col.column("Quarter",
↪ "quarter", type.stringType());
47         TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪ "stock1", type.integerType());
48         TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪ "stock2", type.integerType());
49         TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪ "stock3", type.integerType());
50
51         try {
52             report()
53                 .setTemplate(Templates.reportTemplate)
54                 .columns(yearColumn, quarterColumn,
↪ stock1Column, stock2Column, stock3Column)
55                 .title(Templates.createTitleComponent(
↪ "GroupedStackedBarChart"))
56                 .summary(
57                     cht.groupedStackedBarChart()
58                         .setTitle(
↪ "Grouped stacked bar chart")

```

(continues on next page)

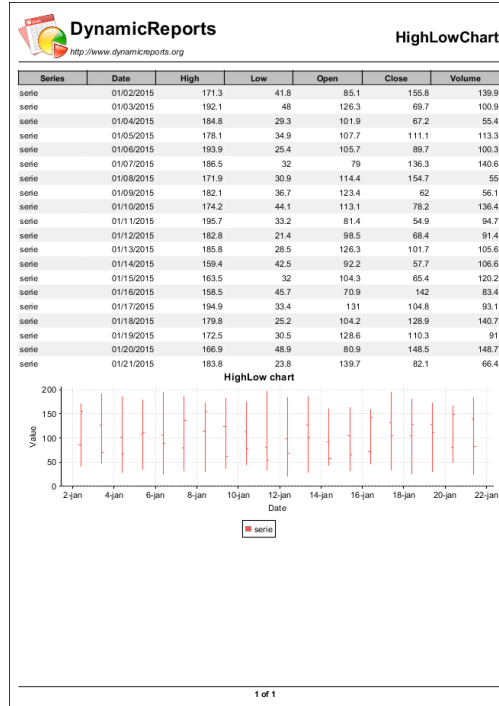
(continued from previous page)

```

59         ↪ setTitleFont (boldFont)
60
61         ↪ setCategory (yearColumn)
62                                     .series(
63                                     cht.
64         ↪ groupedSerie (stock1Column) .setGroup (quarterColumn),
65         ↪ groupedSerie (stock2Column) .setGroup (quarterColumn),
66         ↪ groupedSerie (stock3Column) .setGroup (quarterColumn))
67                                     .pageFooter (Templates.footerComponent)
68                                     .setDataSource (createDataSource ())
69                                     .show ();
70     } catch (DRException e) {
71         e.printStackTrace ();
72     }
73
74     private JRDataSource createDataSource () {
75         DRDataSource dataSource = new DRDataSource ("year", "quarter", "stock1
76         ↪", "stock2", "stock3");
77         dataSource.add ("2010", "Q1", 80, 25, 18);
78         dataSource.add ("2010", "Q2", 98, 78, 22);
79         dataSource.add ("2010", "Q3", 50, 10, 89);
80         dataSource.add ("2010", "Q4", 121, 40, 43);
81         dataSource.add ("2011", "Q1", 103, 120, 34);
82         dataSource.add ("2011", "Q2", 190, 95, 22);
83         dataSource.add ("2011", "Q3", 43, 109, 51);
84         dataSource.add ("2011", "Q4", 80, 88, 40);
85         return dataSource;
86     }
87
88     public static void main (String [] args) {
89         new GroupedStackedBarChartReport ();
90     }

```

1.17.9 High Low Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.util.Calendar;
28  import java.util.Date;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.style.FontBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class HighLowChartReport {
41
42     public HighLowChartReport() {
43         build();
44     }
45
46     private void build() {
47         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
48
49         TextColumnBuilder<String> seriesColumn = col.column("Series", "series
↳", type.stringType());
50         TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
↳dateType());
51         TextColumnBuilder<Double> highColumn = col.column("High", "high",
↳type.doubleType());
52         TextColumnBuilder<Double> lowColumn = col.column("Low", "low", type.
↳doubleType());
53         TextColumnBuilder<Double> openColumn = col.column("Open", "open",
↳type.doubleType());
54         TextColumnBuilder<Double> closeColumn = col.column("Close", "close",
↳type.doubleType());
55         TextColumnBuilder<Double> volumeColumn = col.column("Volume", "volume
↳", type.doubleType());
56
57         try {
58             report()
59                 .setTemplate(Templates.reportTemplate)
60                 .columns(seriesColumn, dateColumn, highColumn,
↳ lowColumn, openColumn, closeColumn, volumeColumn)
61                 .title(Templates.createTitleComponent(
↳ "HighLowChart"))
62                 .summary(
63                     cht.highLowChart()
64                         .setTitle(
↳ "HighLow chart")
65                         .
66                         .
67                         .
68                         .
69                         .
70                         .
71                         .
↳ setClose(closeColumn)

```

(continues on next page)

(continued from previous page)

```

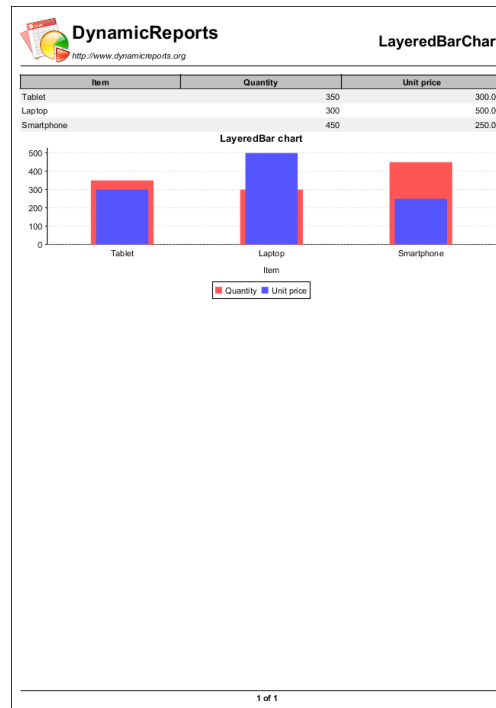
72         ↪setVolume(volumeColumn)
73         ↪setShowOpenTicks(true)
74         ↪setShowCloseTicks(true)
75         ↪setTimeAxisFormat(
76         ↪axisFormat().setLabel("Date"))
77         ↪setValueAxisFormat(
78         ↪axisFormat().setLabel("Value")))
79         .pageFooter(Templates.footerComponent)
80         .setDataSource(createDataSource())
81         .show();
82     } catch (DRException e) {
83         e.printStackTrace();
84     }
85 }
86
87 private JRDataSource createDataSource() {
88     DRDataSource dataSource = new DRDataSource("series", "date", "high",
89     ↪"low", "open", "close", "volume");
89     Calendar c = Calendar.getInstance();
90     c.add(Calendar.DAY_OF_MONTH, -20);
91     for (int i = 0; i < 20; i++) {
92         dataSource.add("serie", c.getTime(), 150 + Math.random() * 50,
93     ↪ 20 + Math.random() * 30, 50 + Math.random() * 90, 50 + Math.random() * 110,
94         50 + Math.random() * 100);
95         c.add(Calendar.DAY_OF_MONTH, 1);
96     }
97     return dataSource;
98 }
99
100 public static void main(String[] args) {
101     new HighLowChartReport();
102 }

```

cht.

cht.

1.17.10 Layered Bar Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class LayeredBarChartReport {
40
41     public LayeredBarChartReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49 ↪ type.stringType());
50         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
51 ↪ "quantity", type.integerType());
52         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
53 ↪", "unitprice", type.bigDecimalType());
54
55         try {
56             report()
57                 .setTemplate(Templates.reportTemplate)
58                 .columns(itemColumn, quantityColumn,
59 ↪ unitPriceColumn)
60                 .title(Templates.createTitleComponent(
61 ↪ "LayeredBarChart"))
62                 .summary(
63                     cht.layeredBarChart()
64                         .setTitle(
65 ↪ "LayeredBar chart")
66                         .
67                         .
68                         .
69                         .series(
70 ↪ serie(quantityColumn), cht.serie(unitPriceColumn))
71                         .
72                         .
73                         .setCategoryAxisFormat(
74 ↪ axisFormat().setLabel("Item"))
75                 .pageFooter(Templates.footerComponent)
76                 .setDataSource(createDataSource())
77                 .show();
78         } catch (DRException e) {
79             e.printStackTrace();
80         }
81     }
82
83     private JRDataSource createDataSource() {

```

(continues on next page)

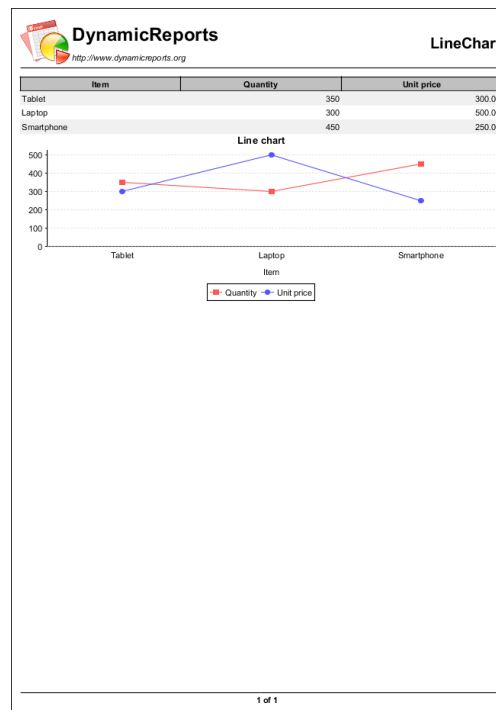
(continued from previous page)

```

76         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
77         dataSource.add("Tablet", 350, new BigDecimal(300));
78         dataSource.add("Laptop", 300, new BigDecimal(500));
79         dataSource.add("Smartphone", 450, new BigDecimal(250));
80         return dataSource;
81     }
82
83     public static void main(String[] args) {
84         new LayeredBarChartReport();
85     }
86 }

```

1.17.11 Line Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,

```

(continues on next page)

(continued from previous page)

```

15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.builder.style.FontBuilder;
32  import net.sf.dynamicreports.report.datasource.DRDataSource;
33  import net.sf.dynamicreports.report.exception.DRException;
34  import net.sf.jasperreports.engine.JRDataSource;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class LineChartReport {
40
41      public LineChartReport() {
42          build();
43      }
44
45      private void build() {
46          FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
49          TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
50          TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
51
52          try {
53              report()
54                  .setTemplate(Templates.reportTemplate)
55                  .columns(itemColumn, quantityColumn,
↪unitPriceColumn)
56                  .title(Templates.createTitleComponent(
↪"LineChart"))
57                  .summary(
58                      cht.lineChart()
59                          .setTitle(
↪"Line chart")
60                          .
61                          .
62                          .series(
↪serie(quantityColumn), cht.serie(unitPriceColumn))
63

```

cht.

(continues on next page)

(continued from previous page)

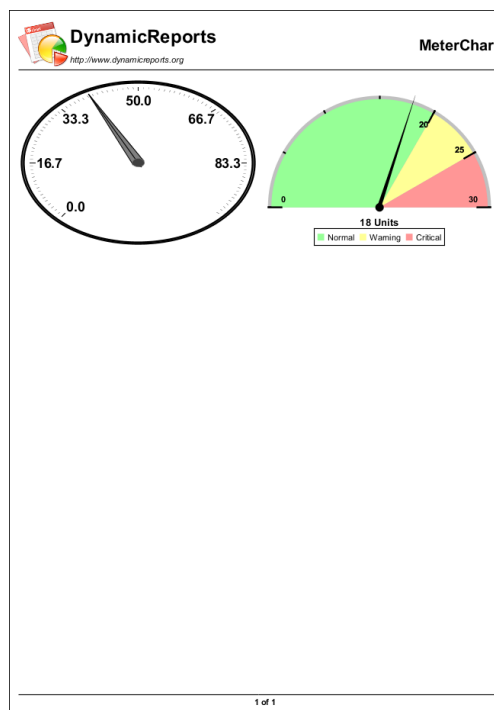
cht.

```

64  ↪setCategoryAxisFormat (
65
66  ↪axisFormat().setLabel("Item"))
67
68  .pageFooter(Templates.footerComponent)
69  .setDataSource(createDataSource())
70  .show();
71
72  } catch (DRException e) {
73      e.printStackTrace();
74  }
75
76  private JRDataSource createDataSource() {
77      DRDataSource dataSource = new DRDataSource("item", "quantity",
78  ↪"unitprice");
79      dataSource.add("Tablet", 350, new BigDecimal(300));
80      dataSource.add("Laptop", 300, new BigDecimal(500));
81      dataSource.add("Smartphone", 450, new BigDecimal(250));
82      return dataSource;
83  }
84
85  public static void main(String[] args) {
86      new LineChartReport();
87  }
88  }

```

1.17.12 Meter Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.awt.Color;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.DynamicReports;
31  import net.sf.dynamicreports.report.builder.chart.MeterChartBuilder;
32  import net.sf.dynamicreports.report.constant.MeterShape;
33  import net.sf.dynamicreports.report.datasource.DRDataSource;
34  import net.sf.dynamicreports.report.exception.DRException;
35  import net.sf.jasperreports.engine.JRDataSource;
36
37  /**
38   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39   */
40  public class MeterChartReport {
41
42      public MeterChartReport() {
43          build();
44      }
45
46      private void build() {
47          MeterChartBuilder chart1 = cht.meterChart()
48              .setValue(DynamicReports.<Number>field("value", type.
49  ↪integerType()))
50              .setShape(MeterShape.DIAL);
51
52          MeterChartBuilder chart2 = cht.meterChart()
53              .setValue(18)
54              .setDataRangeHighExpression(30)
55              .setTickInterval(5d)
56              .setTickColor(Color.BLACK)
57              .setNeedleColor(Color.BLACK)

```

(continues on next page)

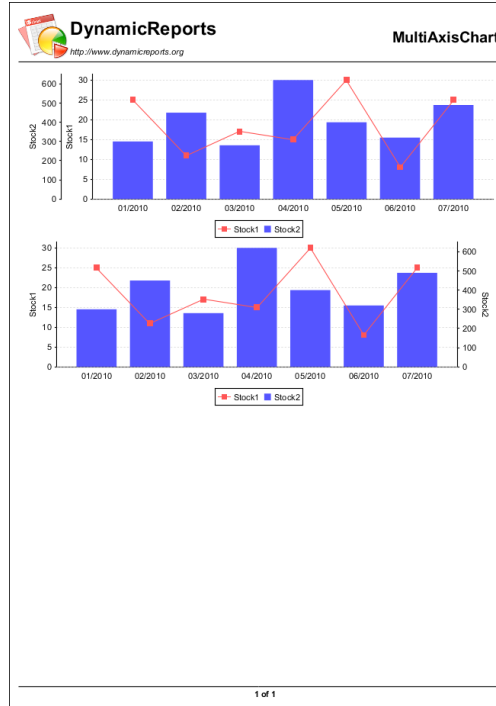
(continued from previous page)

```

57         .setValueColor (Color.BLACK)
58         .setMeterBackgroundColor (Color.LIGHT_GRAY)
59         .intervals (
60             cht.meterInterval ()
61                 .setLabel ("Normal")
62                 .
63             ↪ setBackgroundColor (new Color (150, 255, 150))
64                 .
65             ↪ setDataRangeLowExpression (0)
66                 .
67             ↪ setDataRangeHighExpression (20),
68                 cht.meterInterval ()
69                 .setLabel ("Warning")
70                 .
71             ↪ setBackgroundColor (new Color (255, 255, 150))
72                 .
73             ↪ setDataRangeLowExpression (20)
74                 .
75             ↪ setDataRangeHighExpression (25),
76                 cht.meterInterval ()
77                 .setLabel ("Critical")
78                 .
79             ↪ setBackgroundColor (new Color (255, 150, 150))
80                 .
81             ↪ setDataRangeLowExpression (25)
82                 .
83             ↪ setDataRangeHighExpression (30));
84
85     try {
86         report ()
87         .setTemplate (Templates.reportTemplate)
88         .title (Templates.createTitleComponent (
89             ↪ "MeterChart"))
90         .summary (
91             ↪ cmp.horizontalList (chart1, ↪
92             ↪ chart2))
93         .pageFooter (Templates.footerComponent)
94         .setDataSource (createDataSource ())
95         .show ();
96     } catch (DRException e) {
97         e.printStackTrace ();
98     }
99
100     private JRDataSource createDataSource () {
101         DRDataSource dataSource = new DRDataSource ("value");
102         dataSource.add (40);
103         return dataSource;
104     }
105
106     public static void main (String[] args) {
107         new MeterChartReport ();
108     }
109 }

```

1.17.13 Multi Axis Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.util.Calendar;
28  import java.util.Date;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
32 import net.sf.dynamicreports.report.builder.FieldBuilder;
33 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
34 import net.sf.dynamicreports.report.builder.chart.CategoryChartSerieBuilder;
35 import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
36 import net.sf.dynamicreports.report.constant.AxisPosition;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class MultiAxisChartReport {
46
47     public MultiAxisChartReport() {
48         build();
49     }
50
51     private void build() {
52         FieldBuilder<Date> dateField = field("date", type.dateType());
53         FieldBuilder<Integer> stock1Field = field("stock1", type.
↵integerType());
54         FieldBuilder<Integer> stock2Field = field("stock2", type.
↵integerType());
55
56         CategoryChartSerieBuilder stock1Serie = cht.series(stock1Field).
↵setLabel("Stock1");
57         CategoryChartSerieBuilder stock2Serie = cht.series(stock2Field).
↵setLabel("Stock2");
58
59         LineChartBuilder chart1 = cht.lineChart()
60             .setCategory(new CategoryExpression())
61             .series(stock1Serie)
62             .setValueAxisFormat(
63                 cht.axisFormat().setLabel("Stock1"));
64
65         BarChartBuilder chart2 = cht.barChart()
66             .setCategory(new CategoryExpression())
67             .series(stock2Serie)
68             .setValueAxisFormat(
69                 cht.axisFormat().setLabel("Stock2"));
70
71         try {
72             report()
73                 .setTemplate(Templates.reportTemplate)
74                 .fields(dateField)
75                 .title(
76                     Templates.
77                     ↵createTitleComponent("MultiAxisChart"),
78                     cht.multiAxisChart(chart1, ↵
79                     ↵chart2),
80                     cht.multiAxisChart()
81                     ↵
82                     ↵
83             ↵addChart(chart1, AxisPosition.LEFT_OR_TOP)
84             ↵addChart(chart2, AxisPosition.RIGHT_OR_BOTTOM)

```

(continues on next page)

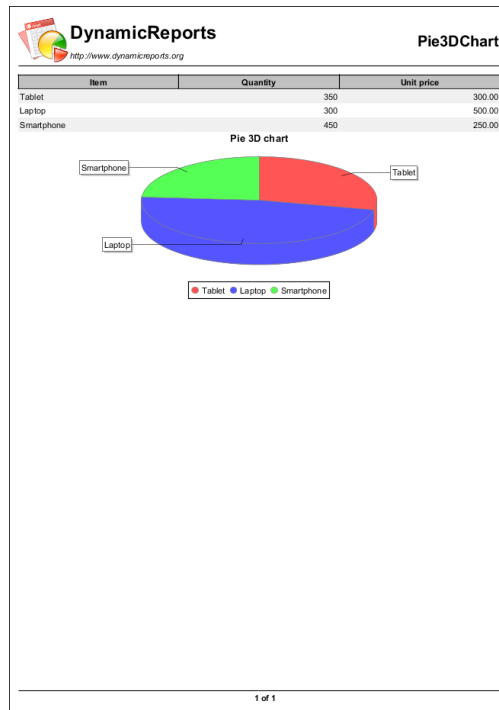
(continued from previous page)

```

81         .pageFooter(Templates.footerComponent)
82         .setDataSource(createDataSource())
83         .show();
84     } catch (DRException e) {
85         e.printStackTrace();
86     }
87 }
88
89 private JRDataSource createDataSource() {
90     DRDataSource dataSource = new DRDataSource("date", "stock1", "stock2
↪");
91     dataSource.add(toDate(2010, 1), 25, 300);
92     dataSource.add(toDate(2010, 2), 11, 450);
93     dataSource.add(toDate(2010, 3), 17, 280);
94     dataSource.add(toDate(2010, 4), 15, 620);
95     dataSource.add(toDate(2010, 5), 30, 400);
96     dataSource.add(toDate(2010, 6), 8, 320);
97     dataSource.add(toDate(2010, 7), 25, 490);
98     return dataSource;
99 }
100
101 private Date toDate(int year, int month) {
102     Calendar c = Calendar.getInstance();
103     c.clear();
104     c.set(Calendar.YEAR, year);
105     c.set(Calendar.MONTH, month - 1);
106     return c.getTime();
107 }
108
109 public static void main(String[] args) {
110     new MultiAxisChartReport();
111 }
112
113 private class CategoryExpression extends AbstractSimpleExpression<String> {
114     private static final long serialVersionUID = 1L;
115
116     @Override
117     public String evaluate(ReportParameters reportParameters) {
118         ↪return type.dateYearToMonthType().valueToString("date", ↪
↪reportParameters);
119     }
120 }
121 }

```

1.17.14 Pie 3D Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class Pie3DChartReport {
40
41     public Pie3DChartReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
49         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
50         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
51
52         try {
53             report()
54                 .setTemplate(Templates.reportTemplate)
55                 .columns(itemColumn, quantityColumn,
↪unitPriceColumn)
56                 .title(Templates.createTitleComponent(
↪"Pie3DChart"))
57                 .summary(
58                     cht.pie3DChart()
59                         .setTitle(
↪"Pie 3D chart")
60                         .
↪setTitleFont(boldFont)
61                         .
↪setKey(itemColumn)
62                         .series(
↪serie(unitPriceColumn))
63
64                 .pageFooter(Templates.footerComponent)
65                 .setDataSource(createDataSource())
66                 .show();
67         } catch (DRException e) {
68             e.printStackTrace();
69         }
70     }
71
72     private JRDataSource createDataSource() {
73         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
74         dataSource.add("Tablet", 350, new BigDecimal(300));
75         dataSource.add("Laptop", 300, new BigDecimal(500));
76         dataSource.add("Smartphone", 450, new BigDecimal(250));
77         return dataSource;

```

cht.

(continues on next page)

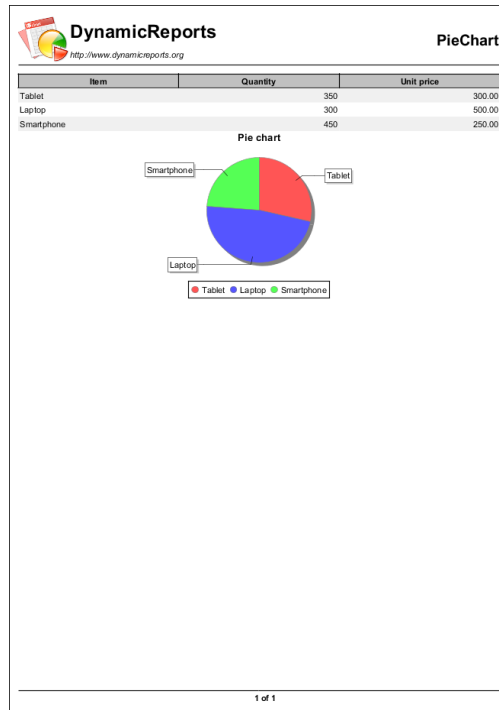
(continued from previous page)

```

78     }
79
80     public static void main(String[] args) {
81         new Pie3DChartReport();
82     }
83 }

```

1.17.15 Pie Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.

```

(continues on next page)

(continued from previous page)

```

21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.builder.style.FontBuilder;
32  import net.sf.dynamicreports.report.datasource.DRDataSource;
33  import net.sf.dynamicreports.report.exception.DRException;
34  import net.sf.jasperreports.engine.JRDataSource;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class PieChartReport {
40
41      public PieChartReport() {
42          build();
43      }
44
45      private void build() {
46          FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳type.stringType());
49          TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳"quantity", type.integerType());
50          TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳", "unitprice", type.bigDecimalType());
51
52          try {
53              report()
54                  .setTemplate(Templates.reportTemplate)
55                  .columns(itemColumn, quantityColumn,
↳unitPriceColumn)
56                  .title(Templates.createTitleComponent(
↳"PieChart"))
57                  .summary(
58                      cht.pieChart()
59                          .setTitle(
↳"Pie chart")
60                          .
↳setTitleFont(boldFont)
61                          .
↳setKey(itemColumn)
62                          .series(
↳serie(unitPriceColumn))
63
64                  .pageFooter(Templates.footerComponent)
65                  .setDataSource(createDataSource())
66                  .show();
67          } catch (DRException e) {
68              e.printStackTrace();

```

(continues on next page)

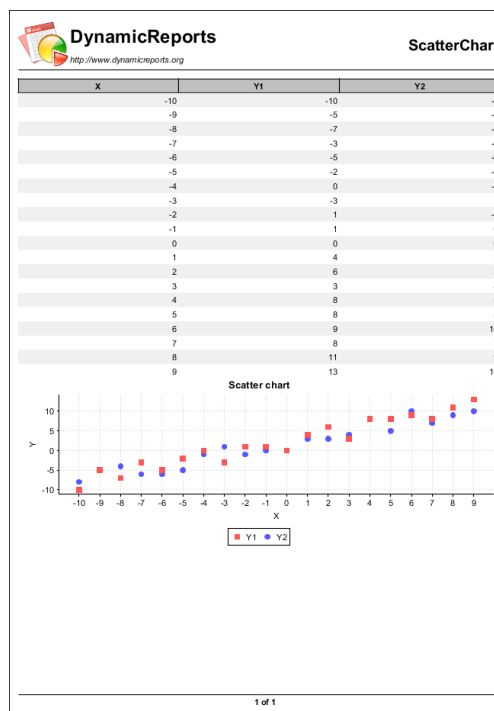
(continued from previous page)

```

69         }
70     }
71
72     private JRDataSource createDataSource() {
73         DRDataSource dataSource = new DRDataSource("item", "quantity",
74 ↪ "unitprice");
75         dataSource.add("Tablet", 350, new BigDecimal(300));
76         dataSource.add("Laptop", 300, new BigDecimal(500));
77         dataSource.add("Smartphone", 450, new BigDecimal(250));
78         return dataSource;
79     }
80
81     public static void main(String[] args) {
82         new PieChartReport();
83     }

```

1.17.16 Scatter Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class ScatterChartReport {
37
38     public ScatterChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↳ integerType());
46         TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↳ integerType());
47         TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↳ integerType());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(xColumn, y1Column, y2Column)
53                 .title(Templates.createTitleComponent(
↳ "ScatterChart"))
54                 .summary(
55                     cht.scatterChart()
56                         .setTitle(
↳ "Scatter chart")
57                         .
↳ setTitleFont(boldFont)
58                         .
↳ setShowLines(false)
59                         .
↳ setXValue(xColumn)

```

(continues on next page)

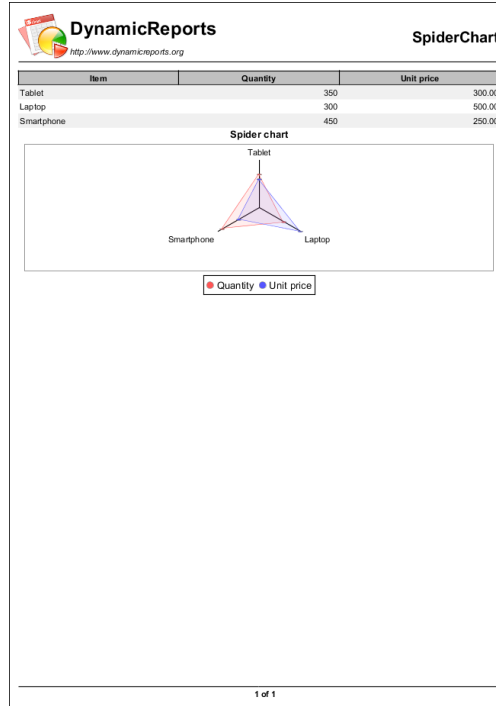
(continued from previous page)

```

60                                     .series (
61                                     cht.xySerie(y2Column))
62                                     .
63                                     cht.
64                                     .
65                                     cht.
66                                     .pageFooter(Templates.footerComponent)
67                                     .setDataSource(createDataSource())
68                                     .show();
69             } catch (DRException e) {
70                 e.printStackTrace();
71             }
72         }
73
74         private JRDataSource createDataSource() {
75             DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
76             for (int i = -10; i < 10; i++) {
77                 dataSource.add(i, i + (int) (Math.random() * 5), i + (int)
60 ↪ (Math.random() * 5));
78             }
79             return dataSource;
80         }
81
82         public static void main(String[] args) {
83             new ScatterChartReport();
84         }
85     }

```

1.17.17 Spider Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class SpiderChartReport {
40
41     public SpiderChartReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
47
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳type.stringType());
49         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳"quantity", type.integerType());
50         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳", "unitprice", type.bigDecimalType());
51
52         try {
53             report()
54                 .setTemplate(Templates.reportTemplate)
55                 .columns(itemColumn, quantityColumn,
↳unitPriceColumn)
56                 .title(Templates.createTitleComponent(
↳"SpiderChart"))
57                 .summary(
58                     cht.spiderChart()
59                         .setTitle(
↳"Spider chart")
60                         .
↳setTitleFont(boldFont)
61                         .
↳setCategory(itemColumn)
62                         .series(
63                             cht.
↳serie(quantityColumn), cht.serie(unitPriceColumn)))
64                 .pageFooter(Templates.footerComponent)
65                 .setDataSource(createDataSource())
66                 .show();
67         } catch (DRException e) {
68             e.printStackTrace();
69         }
70     }
71
72     private JRDataSource createDataSource() {
73         DRDataSource dataSource = new DRDataSource("item", "quantity",
↳"unitprice");
74         dataSource.add("Tablet", 350, new BigDecimal(300));
75         dataSource.add("Laptop", 300, new BigDecimal(500));
76         dataSource.add("Smartphone", 450, new BigDecimal(250));
77         return dataSource;

```

cht.

(continues on next page)

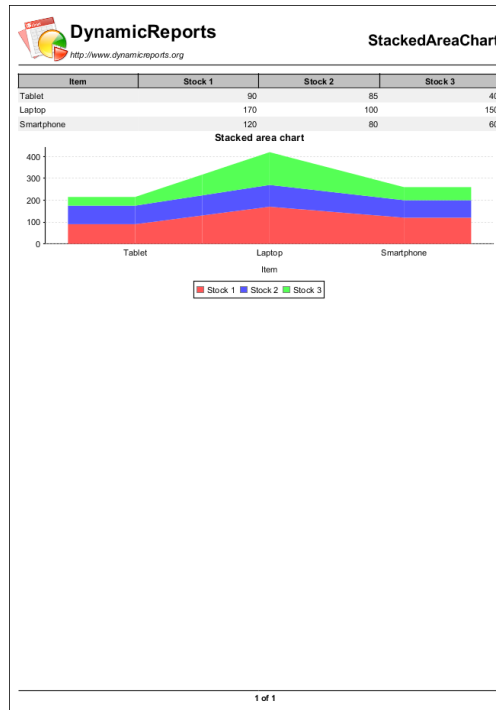
(continued from previous page)

```

78     }
79
80     public static void main(String[] args) {
81         new SpiderChartReport();
82     }
83 }

```

1.17.18 Stacked Area Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.

```

(continues on next page)

(continued from previous page)

```

21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28  import net.sf.dynamicreports.report.builder.style.FontBuilder;
29  import net.sf.dynamicreports.report.datasource.DRDataSource;
30  import net.sf.dynamicreports.report.exception.DRException;
31  import net.sf.jasperreports.engine.JRDataSource;
32
33  /**
34   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35   */
36  public class StackedAreaChartReport {
37
38      public StackedAreaChartReport() {
39          build();
40      }
41
42      private void build() {
43          FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
46          TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↳ "stock1", type.integerType());
47          TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↳ "stock2", type.integerType());
48          TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↳ "stock3", type.integerType());
49
50          try {
51              report()
52                  .setTemplate(Templates.reportTemplate)
53                  .columns(itemColumn, stock1Column,
↳ stock2Column, stock3Column)
54                  .title(Templates.createTitleComponent(
↳ "StackedAreaChart"))
55                  .summary(
56                      cht.stackedAreaChart()
57                          .setTitle(
↳ "Stacked area chart")
58                          .
↳ setTitleFont(boldFont)
59                          .
↳ setCategory(itemColumn)
60                          .series(
↳ serie(stock1Column), cht.serie(stock2Column), cht.serie(stock3Column))
61                          .
↳ setCategoryAxisFormat(
↳ axisFormat().setLabel("Item")))
62                  .pageFooter(Templates.footerComponent)
63                  .setDataSource(createDataSource())
64
65

```

(continues on next page)

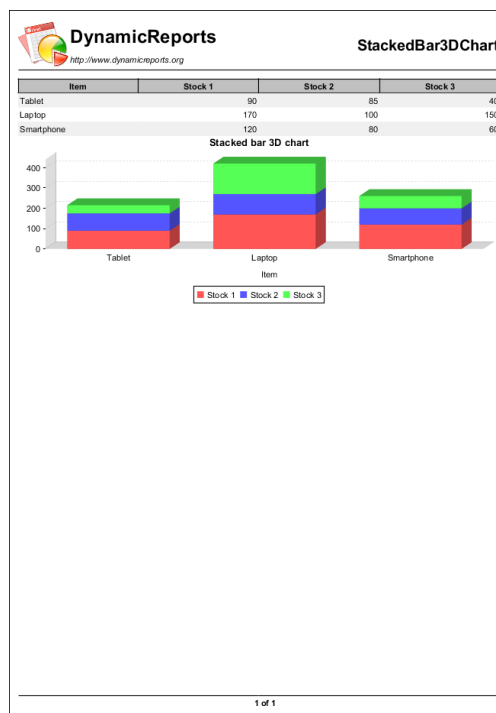
(continued from previous page)

```

66         .show();
67     } catch (DRException e) {
68         e.printStackTrace();
69     }
70 }
71
72 private JRDataSource createDataSource() {
73     DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↪ "stock3");
74     dataSource.add("Tablet", 90, 85, 40);
75     dataSource.add("Laptop", 170, 100, 150);
76     dataSource.add("Smartphone", 120, 80, 60);
77     return dataSource;
78 }
79
80 public static void main(String[] args) {
81     new StackedAreaChartReport();
82 }
83 }

```

1.17.19 Stacked Bar 3D Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.

```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class StackedBar3DChartReport {
37
38     public StackedBar3DChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
46         TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↳ "stock1", type.integerType());
47         TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↳ "stock2", type.integerType());
48         TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↳ "stock3", type.integerType());
49
50         try {
51             report()
52                 .setTemplate(Templates.reportTemplate)
53                 .columns(itemColumn, stock1Column,
↳ stock2Column, stock3Column)
54                 .title(Templates.createTitleComponent(
↳ "StackedBar3DChart"))
55                 .summary(
56                     cht.stackedBar3DChart()
57                         .setTitle(
↳ "Stacked bar 3D chart")

```

(continues on next page)

(continued from previous page)

```

58         ↪ setTitleFont (boldFont)
59         ↪ setCategory (itemColumn)
60                                     .series (
61                                     cht.
62         ↪ serie (stock1Column), cht.serie (stock2Column), cht.serie (stock3Column))
63         ↪ setCategoryAxisFormat (
64         ↪ axisFormat ().setLabel ("Item"))
65                                     .pageFooter (Templates.footerComponent)
66                                     .setDataSource (createDataSource ())
67                                     .show ();
68     } catch (DRException e) {
69         e.printStackTrace ();
70     }
71
72     private JRDataSource createDataSource () {
73         DRDataSource dataSource = new DRDataSource ("item", "stock1", "stock2",
74         ↪ "stock3");
75         dataSource.add ("Tablet", 90, 85, 40);
76         dataSource.add ("Laptop", 170, 100, 150);
77         dataSource.add ("Smartphone", 120, 80, 60);
78         return dataSource;
79     }
80
81     public static void main (String[] args) {
82         new StackedBar3DChartReport ();
83     }

```

1.17.20 Stacked Bar Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28  import net.sf.dynamicreports.report.builder.style.FontBuilder;
29  import net.sf.dynamicreports.report.datasource.DRDataSource;
30  import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class StackedBarChartReport {
37
38     public StackedBarChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
46         TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↪"stock1", type.integerType());
47         TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↪"stock2", type.integerType());
48         TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↪"stock3", type.integerType());
49
50         try {
51             report()
52                 .setTemplate(Templates.reportTemplate)
53                 .columns(itemColumn, stock1Column,
↪stock2Column, stock3Column)
54                 .title(Templates.createTitleComponent(
↪"StackedBarChart"))
55                 .summary(
56                     cht.stackedBarChart()
57                         .setTitle(
↪"Stacked bar chart")
58                         .
59                         .setTitleFont(boldFont)
60                         .
61                         .setCategory(itemColumn)
62                         .series(
↪serie(stock1Column), cht.serie(stock2Column), cht.serie(stock3Column))
63                         .
64                         .setCategoryAxisFormat(
↪axisFormat().setLabel("Item")))
65                 .pageFooter(Templates.footerComponent)
66                 .setDataSource(createDataSource())
67                 .show();
68         } catch (DRException e) {
69             e.printStackTrace();
70         }
71
72     private JRDataSource createDataSource() {
73         DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↪"stock3");
74         dataSource.add("Tablet", 90, 85, 40);

```

(continues on next page)

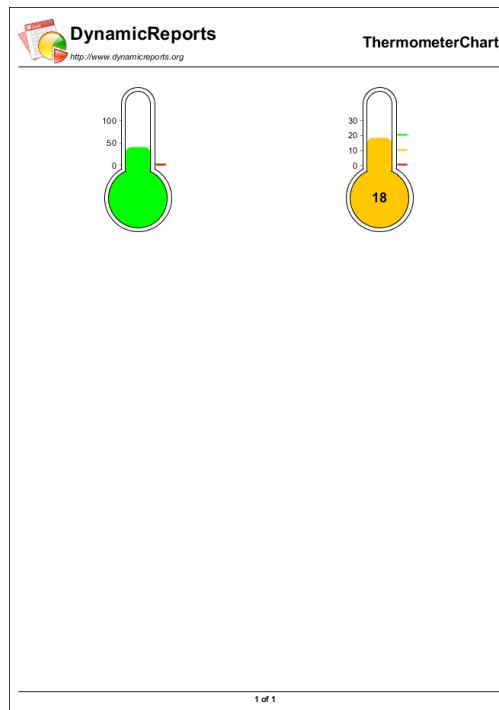
(continued from previous page)

```

75         dataSource.add("Laptop", 170, 100, 150);
76         dataSource.add("Smartphone", 120, 80, 60);
77         return dataSource;
78     }
79
80     public static void main(String[] args) {
81         new StackedBarChartReport();
82     }
83 }

```

1.17.21 Thermometer Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.

```

(continues on next page)

(continued from previous page)

```

18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.DynamicReports;
31 import net.sf.dynamicreports.report.builder.chart.ThermometerChartBuilder;
32 import net.sf.dynamicreports.report.constant.ValueLocation;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ThermometerChartReport {
41
42     public ThermometerChartReport() {
43         build();
44     }
45
46     private void build() {
47         ThermometerChartBuilder chart1 = cht.thermometerChart()
48             .setValue(DynamicReports.<Number>field("value", type.
49 ↪integerType()));
50
51         ThermometerChartBuilder chart2 = cht.thermometerChart()
52             .setValue(18)
53             .setDataRangeHighExpression(30)
54             .setValueColor(Color.BLACK)
55             .setValueLocation(ValueLocation.BULB)
56             .setLowDataRangeLowExpression(0)
57             .setLowDataRangeHighExpression(10)
58             .setMediumDataRangeLowExpression(10)
59             .setMediumDataRangeHighExpression(20)
60             .setHighDataRangeLowExpression(20)
61             .setHighDataRangeHighExpression(30);
62
63         try {
64             report()
65                 .setTemplate(Templates.reportTemplate)
66                 .title(Templates.createTitleComponent(
67 ↪"ThermometerChart"))
68                 .summary(
69                     cmp.horizontalList(chart1,
70 ↪chart2))
71                 .pageFooter(Templates.footerComponent)
72                 .setDataSource(createDataSource())
73                 .show();
74         } catch (DRException e) {

```

(continues on next page)

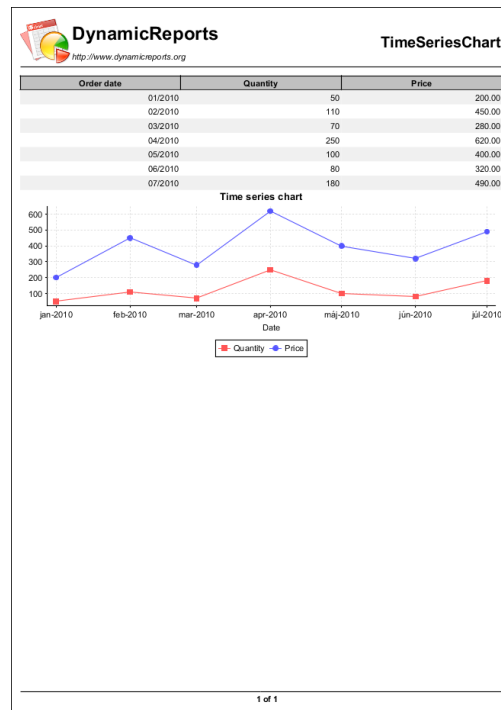
(continued from previous page)

```

72         e.printStackTrace();
73     }
74 }
75
76 private JRDataSource createDataSource() {
77     DRDataSource dataSource = new DRDataSource("value");
78     dataSource.add(40);
79     return dataSource;
80 }
81
82 public static void main(String[] args) {
83     new ThermometerChartReport();
84 }
85 }

```

1.17.22 Time-Series Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.

```

(continues on next page)

(continued from previous page)

```

13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.style.FontBuilder;
34 import net.sf.dynamicreports.report.constant.TimePeriod;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class TimeSeriesChartReport {
43
44     public TimeSeriesChartReport() {
45         build();
46     }
47
48     private void build() {
49         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
50
51         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪ "orderdate", type.dateYearToMonthType());
52         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪ "quantity", type.integerType());
53         TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
↪ ", type.bigDecimalType());
54
55         try {
56             report()
57                 .setTemplate(Templates.reportTemplate)
58                 .columns(orderDateColumn, quantityColumn,
↪ priceColumn)
59                 .title(Templates.createTitleComponent(
↪ "TimeSeriesChart"))
60                 .summary(
61                     cht.timeSeriesChart()
62                         .setTitle(
↪ "Time series chart")
63                         .setTitleFont(boldFont)

```

(continues on next page)

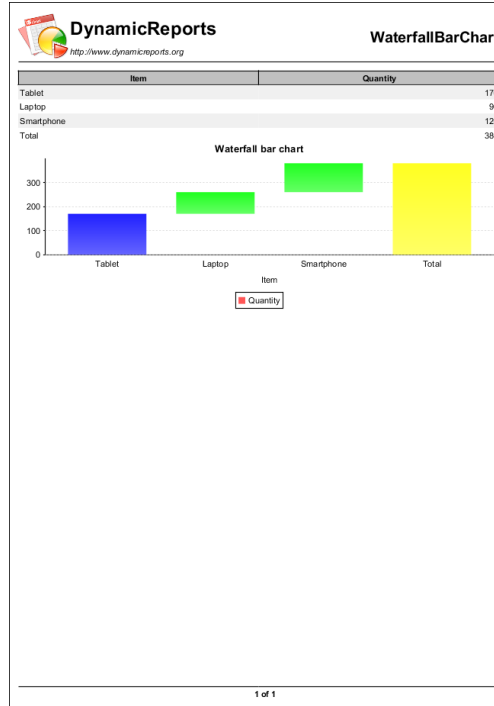
(continued from previous page)

```

64         ↪ setTimePeriod(orderDateColumn)
65
66         ↪ setTimePeriodType(TimePeriod.MONTH)
67
68         ↪ serie(quantityColumn), cht.serie(priceColumn))
69
70         ↪ setTimeAxisFormat(
71         ↪ axisFormat().setLabel("Date"))
72
73         .pageFooter(Templates.footerComponent)
74         .setDataSource(createDataSource())
75         .show();
76     } catch (DRException e) {
77         e.printStackTrace();
78     }
79
80     private JRDataSource createDataSource() {
81         DRDataSource dataSource = new DRDataSource("orderdate", "quantity",
82         ↪ "price");
83         dataSource.add(toDate(2010, 1), 50, new BigDecimal(200));
84         dataSource.add(toDate(2010, 2), 110, new BigDecimal(450));
85         dataSource.add(toDate(2010, 3), 70, new BigDecimal(280));
86         dataSource.add(toDate(2010, 4), 250, new BigDecimal(620));
87         dataSource.add(toDate(2010, 5), 100, new BigDecimal(400));
88         dataSource.add(toDate(2010, 6), 80, new BigDecimal(320));
89         dataSource.add(toDate(2010, 7), 180, new BigDecimal(490));
90         return dataSource;
91     }
92
93     private Date toDate(int year, int month) {
94         Calendar c = Calendar.getInstance();
95         c.clear();
96         c.set(Calendar.YEAR, year);
97         c.set(Calendar.MONTH, month - 1);
98         return c.getTime();
99     }
100
101     public static void main(String[] args) {
102         new TimeSeriesChartReport();
103     }

```

1.17.23 Waterfall Bar Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class WaterfallBarChartReport {
37
38     public WaterfallBarChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
46         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
47
48         try {
49             report()
50                 .setTemplate(Templates.reportTemplate)
51                 .columns(itemColumn, quantityColumn)
52                 .title(Templates.createTitleComponent(
↪"WaterfallBarChart"))
53                 .summary(
54                     cht.waterfallBarChart()
55                         .setTitle(
↪"Waterfall bar chart")
56                         .
57                         .
58                         .series(cht.
↪serie(quantityColumn))
59                         .
60                         .setCategoryAxisFormat(cht.
↪axisFormat().setLabel("Item")))
61                 .pageFooter(Templates.footerComponent)
62                 .setDataSource(createDataSource())
63                 .show();
64         } catch (DRException e) {
65             e.printStackTrace();
66         }
67     }
68
69     private JRDataSource createDataSource() {
70         DRDataSource dataSource = new DRDataSource("item", "quantity");
71         dataSource.add("Tablet", 170);
72         dataSource.add("Laptop", 90);
73         dataSource.add("Smartphone", 120);
74         dataSource.add("Total", 380);
75         return dataSource;
76     }
77
78     public static void main(String[] args) {

```

(continues on next page)

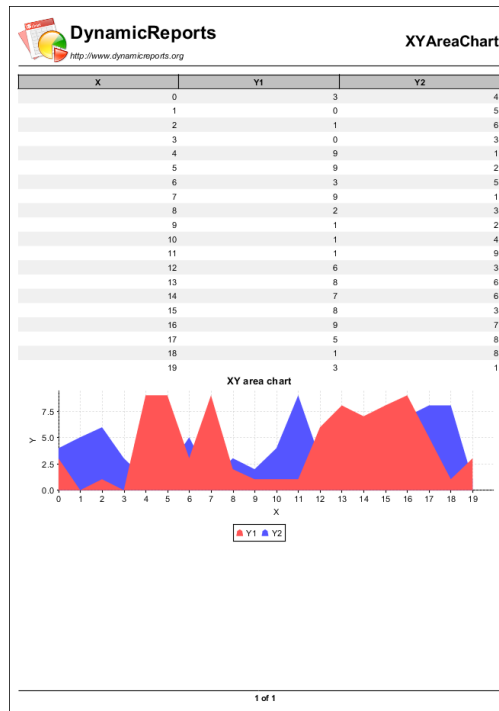
(continued from previous page)

```

79         new WaterfallBarChartReport();
80     }
81 }

```

1.17.24 XY-Area Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;

```

(continues on next page)

(continued from previous page)

```

24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class XYAreaChartReport {
37
38     public XYAreaChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
46         TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↪integerType());
47         TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↪integerType());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(xColumn, y1Column, y2Column)
53                 .title(Templates.createTitleComponent(
↪"XYAreaChart"))
54                 .summary(
55                     cht.xyAreaChart()
56                         .setTitle("XY
↪area chart")
57                         .
58                         ↪setTitleFont(boldFont)
59                         .
60                         ↪setXValue(xColumn)
61                         .series(
62                             cht.
↪xySerie(y1Column), cht.xySerie(y2Column))
63                         .
64                         ↪setXAxisFormat(
65                             cht.
↪axisFormat().setLabel("X"))
66                         .
67                         ↪setYAxisFormat(
68                             cht.
↪axisFormat().setLabel("Y"))
69                 .pageFooter(Templates.footerComponent)
70                 .setDataSource(createDataSource())
71                 .show();
72         } catch (DRException e) {

```

(continues on next page)

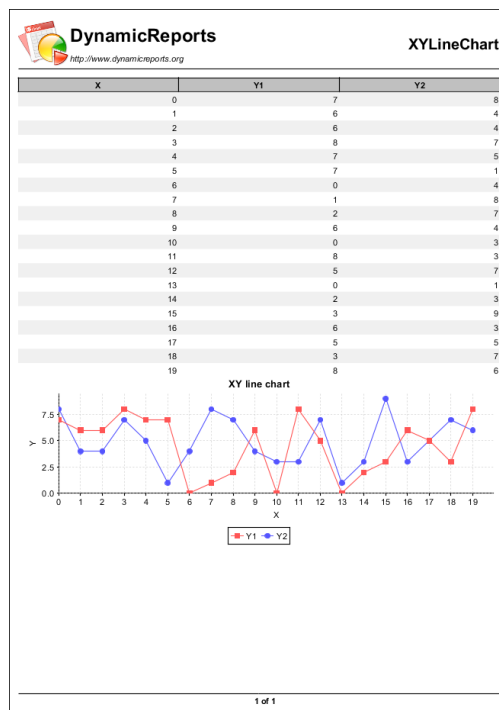
(continued from previous page)

```

69         e.printStackTrace();
70     }
71 }
72
73 private JRDataSource createDataSource() {
74     DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
75     for (int i = 0; i < 20; i++) {
76         dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
77 ↪random() * 10));
78     }
79     return dataSource;
80 }
81
82 public static void main(String[] args) {
83     new XYAreaChartReport();
84 }

```

1.17.25 XY-Line Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify

```

(continues on next page)

(continued from previous page)

```

10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class XYLineChartReport {
37
38     public XYLineChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
46         TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↪integerType());
47         TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↪integerType());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(xColumn, y1Column, y2Column)
53                 .title(Templates.createTitleComponent(
↪"XYLineChart"))
54                 .summary(
55                     cht.xyLineChart()
56                         .setTitle("XY_
↪line chart")
57                         .
58                         .
59                         .setTitleFont(boldFont)
60                         .setXValue(xColumn)
61                         .series(

```

(continues on next page)

(continued from previous page)

```

60  ↪xySerie(y1Column), cht.xySerie(y2Column))
61
62  ↪setXAxisFormat (
63
64  ↪axisFormat().setLabel("X"))
65
66  ↪setYAxisFormat (
67
68  ↪axisFormat().setLabel("Y"))
69
70  .pageFooter(Templates.footerComponent)
71  .setDataSource(createDataSource())
72  .show();
73  } catch (DRException e) {
74      e.printStackTrace();
75  }
76
77  private JRDataSource createDataSource() {
78      DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
79      for (int i = 0; i < 20; i++) {
80          dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
81  ↪random() * 10));
82      }
83      return dataSource;
84  }
85
86  public static void main(String[] args) {
87      new XYLineChartReport();
88  }
89

```

cht.

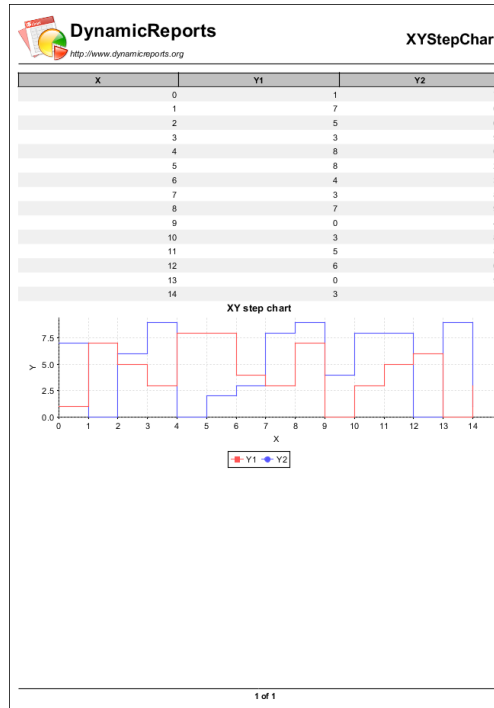
.

cht.

.

cht.

1.17.26 XY-Step Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class XYStepChartReport {
37
38     public XYStepChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
46         TextColumnBuilder<Integer> y1Column = col.column("Y1", "y1", type.
↪integerType());
47         TextColumnBuilder<Integer> y2Column = col.column("Y2", "y2", type.
↪integerType());
48
49         try {
50             report()
51
52                 .setTemplate(Templates.reportTemplate)
53                 .columns(xColumn, y1Column, y2Column)
54                 .title(Templates.createTitleComponent(
↪"XYStepChart"))
55                 .summary(
56                     cht.xyStepChart()
57                         .setTitle("XY_
↪step chart")
58                         .
59                         .setTitleFont(boldFont)
60                         .
61                         .setXValue(xColumn)
62                         .series(
↪cht.
63                             xySerie(y1Column), cht.xySerie(y2Column))
64                             .
65                             .setXAxisFormat(
↪cht.
66                             axisFormat().setLabel("X"))
67                             .
68                             .setYAxisFormat(
↪cht.
69                             axisFormat().setLabel("Y"))
70
71                 .pageFooter(Templates.footerComponent)
72                 .setDataSource(createDataSource())
73                 .show();
74             } catch (DRException e) {
75                 e.printStackTrace();
76             }
77
78     private JRDataSource createDataSource() {
79         DRDataSource dataSource = new DRDataSource("x", "y1", "y2");
80         for (int i = 0; i < 15; i++) {

```

(continues on next page)

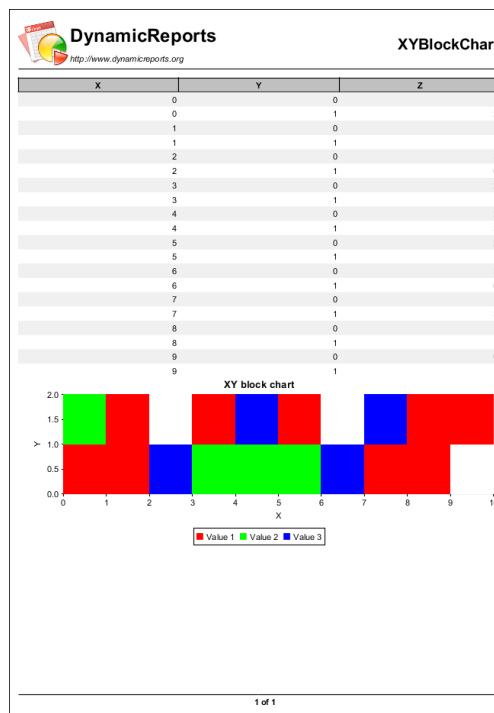
(continued from previous page)

```

76         dataSource.add(i, (int) (Math.random() * 10), (int) (Math.
↪random() * 10));
77     }
78     return dataSource;
79 }
80
81 public static void main(String[] args) {
82     new XYStepChartReport();
83 }
84 }

```

1.17.27 XY-Block Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

```

(continues on next page)

(continued from previous page)

```

17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chart;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.style.FontBuilder;
32 import net.sf.dynamicreports.report.constant.RectangleAnchor;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class XyBlockChartReport {
41
42     public XyBlockChartReport() {
43         build();
44     }
45
46     private void build() {
47         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
48
49         TextColumnBuilder<Integer> xColumn = col.column("X", "x", type.
↪integerType());
50         TextColumnBuilder<Integer> yColumn = col.column("Y", "y", type.
↪integerType());
51         TextColumnBuilder<Integer> zColumn = col.column("Z", "z", type.
↪integerType());
52
53         try {
54             report()
55                 .setTemplate(Templates.reportTemplate)
56                 .columns(xColumn, yColumn, zColumn)
57                 .title(Templates.createTitleComponent(
↪"XYBlockChart"))
58                 .summary(
59                     cht.xyBlockChart(0, 5, Color.
↪WHITE)
60                     .setTitle("XY_
↪block chart")
61                     .
62                     .setTitleFont(boldFont)
63                     .
64                     .setBlockAnchor(RectangleAnchor.BOTTOM_LEFT)
65                     .paintScales(
↪cht.
↪paintScale("Value 1", 1, Color.RED),

```

(continues on next page)

(continued from previous page)

```

65     ↪paintScale("Value 2", 2, Color.GREEN),
66     ↪paintScale("Value 3", 3, Color.BLUE))
67     ↪setXValue(xColumn)
68                                     .series(
69     ↪xyzSerie().setYValue(yColumn).setZValue(zColumn))
70                                     .
71     ↪setXAxisFormat(
72     ↪axisFormat().setLabel("X"))
73                                     .
74     ↪setYAxisFormat(
75     ↪axisFormat().setLabel("Y"))
76                                     .pageFooter(Templates.footerComponent)
77                                     .setDataSource(createDataSource())
78                                     .show();
79     } catch (DRException e) {
80         e.printStackTrace();
81     }
82     private JRDataSource createDataSource() {
83         DRDataSource dataSource = new DRDataSource("x", "y", "z");
84         for (int i = 0; i < 10; i++) {
85             dataSource.add(i, 0, (int) (Math.random() * 4));
86             dataSource.add(i, 1, (int) (Math.random() * 4));
87         }
88         return dataSource;
89     }
90
91     public static void main(String[] args) {
92         new XyBlockChartReport();
93     }
94 }

```

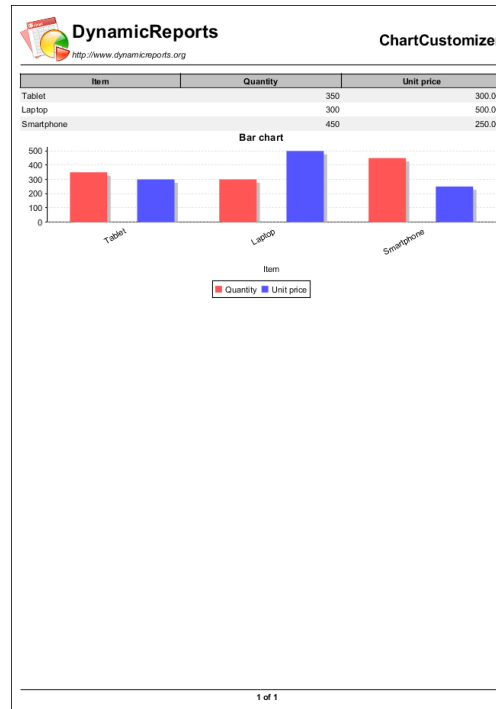
cht.
cht.
.
cht.
.
cht.
.
cht.

Table 2: Chart Examples

			
AreaChart	Bar3DChart	BarChart	BubbleChart
			
CandlestickChart	DifferenceChart	GanttChart	GroupedStackedBar
			
HighLowChart	LayeredBarChart	LineChart	MeterChart
			
MultiAxisChart	Pie3DChart	PieChart	ScatterChart
			
SpiderChart	StackedAreaChart	StackedBar3DChart	StackedBarChart
			
ThermometerChart	TimeSeriesChart	WaterfallBarChart	XYAreaChart
			
XYBarChart	XYBarChart	XYBarChart	

1.18 Chart customization

1.18.1 Chart Customizer



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;

```

(continues on next page)

(continued from previous page)

```

28 import java.io.Serializable;
29 import java.math.BigDecimal;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.style.FontBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.definition.ReportParameters;
36 import net.sf.dynamicreports.report.definition.chart.DRChartCustomizer;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 import org.jfree.chart.JFreeChart;
41 import org.jfree.chart.axis.CategoryAxis;
42 import org.jfree.chart.axis.CategoryLabelPositions;
43 import org.jfree.chart.renderer.category.BarRenderer;
44
45 /**
46  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
47  */
48 public class ChartCustomizerReport {
49
50     public ChartCustomizerReport() {
51         build();
52     }
53
54     private void build() {
55         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
56
57         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
58         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳ "quantity", type.integerType());
59         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳ ", "unitprice", type.bigDecimalType());
60
61         try {
62             report()
63                 .setTemplate(Templates.reportTemplate)
64                 .columns(itemColumn, quantityColumn,
↳ unitPriceColumn)
65                 .title(Templates.createTitleComponent(
↳ "ChartCustomizer"))
66                 .summary(
67                     cht.barChart()
68                         .
↳ customizers(new ChartCustomizer())
69                         .setTitle(
↳ "Bar chart")
70                         .
↳ setTitleFont(boldFont)
71                         .
↳ setCategory(itemColumn)
72                         .series(
↳ serie(quantityColumn), cht.serie(unitPriceColumn))
73                         .
↳ setCategoryAxisFormat (

```

(continues on next page)

(continued from previous page)

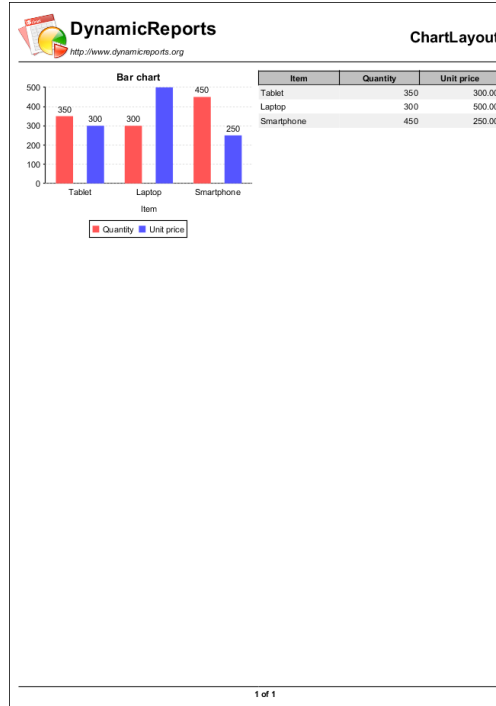
cht.

```

75  ↪axisFormat().setLabel("Item"))
76                                     .pageFooter(Templates.footerComponent)
77                                     .setDataSource(createDataSource())
78                                     .show();
79      } catch (DRException e) {
80          e.printStackTrace();
81      }
82  }
83
84  private class ChartCustomizer implements DRChartCustomizer, Serializable {
85      private static final long serialVersionUID = 1L;
86
87      @Override
88      ↪public void customize(JFreeChart chart, ReportParameters_
89  ↪reportParameters) {
89          BarRenderer renderer = (BarRenderer) chart.getCategoryPlot().
90  ↪getRenderer();
91          renderer.setShadowPaint(Color.LIGHT_GRAY);
92          renderer.setShadowVisible(true);
93          CategoryAxis domainAxis = chart.getCategoryPlot().
94  ↪getDomainAxis();
95          domainAxis.setCategoryLabelPositions(CategoryLabelPositions.
96  ↪createUpRotationLabelPositions(Math.PI / 6.0));
97      }
98
99      private JRDataSource createDataSource() {
100          DRDataSource dataSource = new DRDataSource("item", "quantity",
101  ↪"unitprice");
102          dataSource.add("Tablet", 350, new BigDecimal(300));
103          dataSource.add("Laptop", 300, new BigDecimal(500));
104          dataSource.add("Smartphone", 450, new BigDecimal(250));
105          return dataSource;
106      }
107
108      public static void main(String[] args) {
109          new ChartCustomizerReport();
110      }
111  }

```

1.18.2 Chart Layout



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.Serializable;
28 import java.math.BigDecimal;
29
30 import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
32 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.style.FontBuilder;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.definition.ReportParameters;
37 import net.sf.dynamicreports.report.definition.chart.DRChartCustomizer;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40
41 import org.jfree.chart.JFreeChart;
42 import org.jfree.chart.labels.StandardCategoryItemLabelGenerator;
43 import org.jfree.chart.renderer.category.BarRenderer;
44
45 /**
46  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
47  */
48 public class ChartLayoutReport {
49
50     public ChartLayoutReport() {
51         build();
52     }
53
54     private void build() {
55         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
56
57         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
58         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳ "quantity", type.integerType());
59         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳ ", "unitprice", type.bigDecimalType());
60
61         JasperReportBuilder subreport = report();
62         subreport
63             .setTemplate(Templates.reportTemplate)
64             .columns(itemColumn, quantityColumn, unitPriceColumn)
65             .setDataSource(createDataSource());
66
67         BarChartBuilder chart = cht.barChart()
68             .customizers(new ChartCustomizer())
69             .setTitle("Bar chart")
70             .setTitleFont(boldFont)
71             .setCategory(itemColumn)
72             .series(
73                 cht.serie(quantityColumn), cht.
↳ serie(unitPriceColumn))
74             .setValueAxisFormat(
75                 cht.axisFormat().
↳ setRangeMaxValueExpression(500))
76             .setCategoryAxisFormat(
77                 cht.axisFormat().setLabel("Item"));
78
79         try {
80             report()
81                 .setTemplate(Templates.reportTemplate)
82                 .title(Templates.createTitleComponent(
↳ "ChartLayout"))

```

(continues on next page)

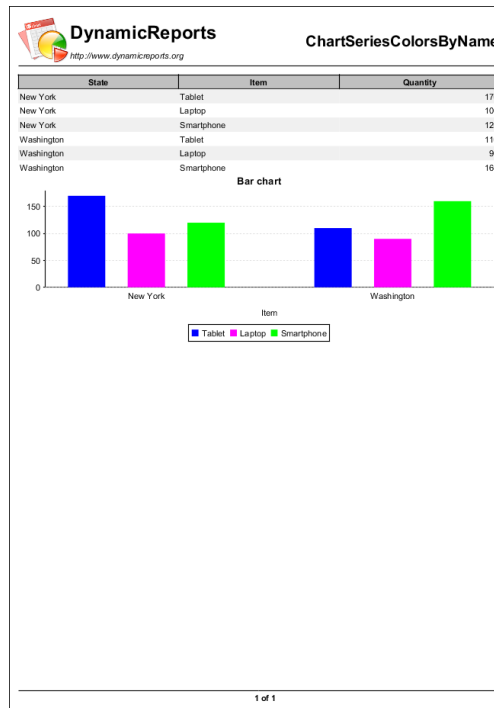
(continued from previous page)

```

83         .summary(
84             cmp.horizontalList(chart, cmp.
↪subreport(subreport)))
85         .pageFooter(Templates.footerComponent)
86         .setDataSource(createDataSource())
87         .show();
88     } catch (DRException e) {
89         e.printStackTrace();
90     }
91 }
92
93 private class ChartCustomizer implements DRICustomizer, Serializable {
94     private static final long serialVersionUID = 1L;
95
96     @Override
97     ↪reportParameters) {
98         BarRenderer renderer = (BarRenderer) chart.getCategoryPlot().
↪getRenderer();
99         renderer.setBaseItemLabelGenerator(new
↪StandardCategoryItemLabelGenerator());
100         renderer.setBaseItemLabelsVisible(true);
101     }
102
103     private JRDataSource createDataSource() {
104         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
105         dataSource.add("Tablet", 350, new BigDecimal(300));
106         dataSource.add("Laptop", 300, new BigDecimal(500));
107         dataSource.add("Smartphone", 450, new BigDecimal(250));
108         return dataSource;
109     }
110
111     public static void main(String[] args) {
112         new ChartLayoutReport();
113     }
114 }
115 }

```

1.18.3 Chart Series Colors By Name



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chartcustomization;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.awt.Color;
28  import java.util.HashMap;
29  import java.util.Map;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.style.FontBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class ChartSeriesColorsByNameReport {
42
43     public ChartSeriesColorsByNameReport() {
44         build();
45     }
46
47     private void build() {
48         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
49
50         TextColumnBuilder<String> stateColumn = col.column("State", "state",
↳ type.stringType());
51         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
52         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳ "quantity", type.integerType());
53
54         Map<String, Color> seriesColors = new HashMap<String, Color>();
55         seriesColors.put("Tablet", Color.BLUE);
56         seriesColors.put("Smartphone", Color.GREEN);
57         seriesColors.put("Laptop", Color.MAGENTA);
58
59         try {
60             report()
61                 .setTemplate(Templates.reportTemplate)
62                 .columns(stateColumn, itemColumn,
↳ quantityColumn)
63                 .title(Templates.createTitleComponent(
↳ "ChartSeriesColorsByName"))
64                 .summary(
65                     cht.barChart()
66                         .setTitle(
↳ "Bar chart")
67                         .setTitleFont(boldFont)
68                         .seriesColorsByName(seriesColors)
69                         .setCategory(stateColumn)
70                         .series(
↳ serie(quantityColumn).setSeries(itemColumn))
71                         .setCategoryAxisFormat(
↳ axisFormat().setLabel("Item"))
72                 .pageFooter(Templates.footerComponent)
73                 .setDataSource(createDataSource())
74
75

```

(continues on next page)

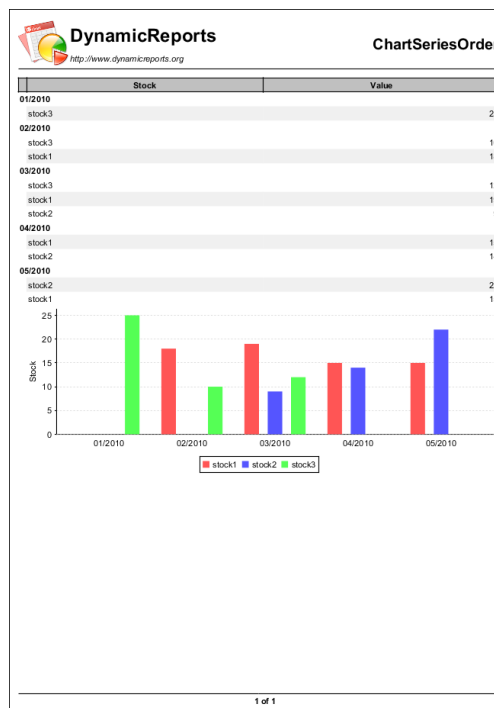
(continued from previous page)

```

76         .show();
77     } catch (DRException e) {
78         e.printStackTrace();
79     }
80 }
81
82 private JRDataSource createDataSource() {
83     DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪");
84
85     dataSource.add("New York", "Tablet", 170);
86     dataSource.add("New York", "Laptop", 100);
87     dataSource.add("New York", "Smartphone", 120);
88     dataSource.add("Washington", "Tablet", 110);
89     dataSource.add("Washington", "Laptop", 90);
90     dataSource.add("Washington", "Smartphone", 160);
91     return dataSource;
92 }
93
94 public static void main(String[] args) {
95     new ChartSeriesColorsByNameReport();
96 }

```

1.18.4 Chart Series Order



```

1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca

```

(continues on next page)

(continued from previous page)

```

5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.Calendar;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
32 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.constant.OrderType;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.definition.ReportParameters;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class ChartSeriesOrderReport {
44
45     public ChartSeriesOrderReport() {
46         build();
47     }
48
49     private void build() {
50         TextColumnBuilder<Date> dateColumn = col.column("Date", "date", type.
51 ↪dateYearToMonthType());
52         TextColumnBuilder<String> stockColumn = col.column("Stock", "stock",
53 ↪type.stringType());
54         TextColumnBuilder<Integer> valueColumn = col.column("Value", "value",
55 ↪type.integerType());
56
57         BarChartBuilder chart = cht.barChart()
58             .setCategory(new CategoryExpression())
59             .series(cht.serie(valueColumn).setSeries(stockColumn))
60             .setSeriesOrderType(OrderType.ASCENDING)
61             .setValueAxisFormat(

```

(continues on next page)

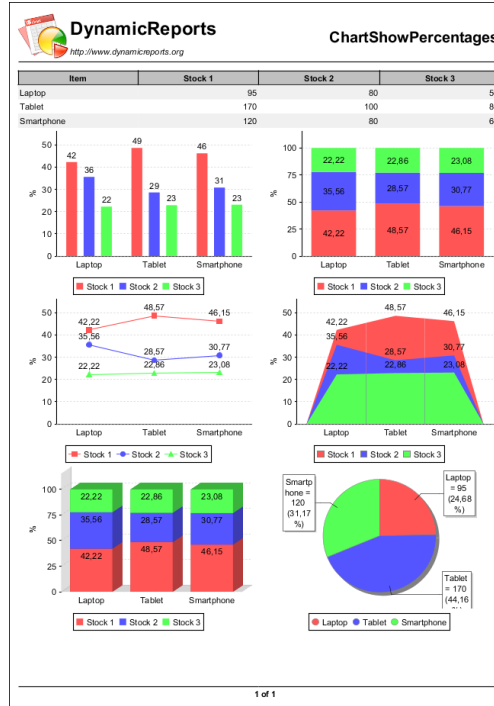
(continued from previous page)

```

59         cht.axisFormat().setLabel("Stock"));
60
61         try {
62             report()
63                 .setTemplate(Templates.reportTemplate)
64                 .columns(dateColumn, stockColumn, valueColumn)
65                 .title(Templates.createTitleComponent(
66                     ↪ "ChartSeriesOrder"))
67                 .groupBy(dateColumn)
68                 .summary(chart)
69                 .pageFooter(Templates.footerComponent)
70                 .setDataSource(createDataSource())
71                 .show();
72         } catch (DRException e) {
73             e.printStackTrace();
74         }
75
76     private JRDataSource createDataSource() {
77         DRDataSource dataSource = new DRDataSource("date", "stock", "value");
78         dataSource.add(toDate(2010, 1), "stock3", 25);
79         dataSource.add(toDate(2010, 2), "stock3", 10);
80         dataSource.add(toDate(2010, 2), "stock1", 18);
81         dataSource.add(toDate(2010, 3), "stock3", 12);
82         dataSource.add(toDate(2010, 3), "stock1", 19);
83         dataSource.add(toDate(2010, 3), "stock2", 9);
84         dataSource.add(toDate(2010, 4), "stock1", 15);
85         dataSource.add(toDate(2010, 4), "stock2", 14);
86         dataSource.add(toDate(2010, 5), "stock2", 22);
87         dataSource.add(toDate(2010, 5), "stock1", 15);
88         return dataSource;
89     }
90
91     private Date toDate(int year, int month) {
92         Calendar c = Calendar.getInstance();
93         c.clear();
94         c.set(Calendar.YEAR, year);
95         c.set(Calendar.MONTH, month - 1);
96         return c.getTime();
97     }
98
99     public static void main(String[] args) {
100         new ChartSeriesOrderReport();
101     }
102
103     private class CategoryExpression extends AbstractSimpleExpression<String> {
104         private static final long serialVersionUID = 1L;
105
106         @Override
107         public String evaluate(ReportParameters reportParameters) {
108             ↪ return type.dateYearToMonthType().valueToString("date", ↪
109                 ↪ reportParameters);
110         }
111     }

```

1.18.5 Chart Show Percentages



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.chart.AreaChartBuilder;
28 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
29 import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
30 import net.sf.dynamicreports.report.builder.chart.PieChartBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.chart.StackedBar3DChartBuilder;
32 import net.sf.dynamicreports.report.builder.chart.StackedBarChartBuilder;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class ChartShowPercentagesReport {
42
43     public ChartShowPercentagesReport() {
44         build();
45     }
46
47     private void build() {
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
49         TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↳ "stock1", type.integerType());
50         TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↳ "stock2", type.integerType());
51         TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↳ "stock3", type.integerType());
52
53         BarChartBuilder barChart = cht.barChart()
54             .setShowPercentages(true)
55             .setShowValues(true)
56             .setPercentValuePattern("#,##0")
57             .setCategory(itemColumn)
58             .series(cht.series(stock1Column), cht.
↳ serie(stock2Column), cht.series(stock3Column));
59         StackedBarChartBuilder stackedBarChart = cht.stackedBarChart()
60             .setShowPercentages(true)
61             .setShowValues(true)
62             .setCategory(itemColumn)
63             .series(cht.series(stock1Column), cht.
↳ serie(stock2Column), cht.series(stock3Column));
64         LineChartBuilder lineChart = cht.lineChart()
65             .setShowPercentages(true)
66             .setShowValues(true)
67             .setCategory(itemColumn)
68             .series(cht.series(stock1Column), cht.
↳ serie(stock2Column), cht.series(stock3Column));
69         AreaChartBuilder areaChart = cht.areaChart()
70             .setShowPercentages(true)
71             .setShowValues(true)
72             .setCategory(itemColumn)
73             .series(cht.series(stock1Column), cht.
↳ serie(stock2Column), cht.series(stock3Column));
74         StackedBar3DChartBuilder stackedBar3DChart = cht.stackedBar3DChart()
75             .setShowPercentages(true)
76             .setShowValues(true)
77             .setCategory(itemColumn)
78             .series(cht.series(stock1Column), cht.
↳ serie(stock2Column), cht.series(stock3Column));

```

(continues on next page)

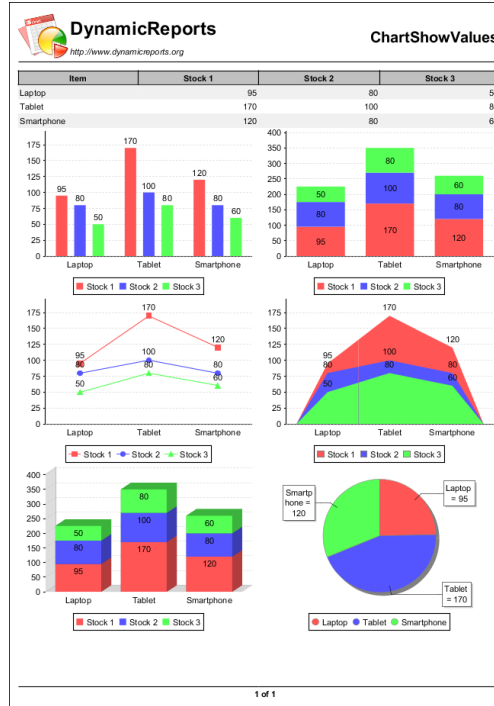
(continued from previous page)

```

79         PieChartBuilder pieChart = cht.pieChart()
80             .setShowPercentages(true)
81             .setShowValues(true)
82             .setKey(itemColumn)
83             .series(cht.serie(stock1Column));
84
85         try {
86             report()
87                 .setTemplate(Templates.reportTemplate)
88                 .columns(itemColumn, stock1Column, ↵
↵stock2Column, stock3Column)
89                 .title(Templates.createTitleComponent(
↵"ChartShowPercentages"))
90                 .summary(
91                     cmp.horizontalList(barChart, ↵
↵stackedBarChart),
92                     cmp.horizontalList(lineChart, ↵
↵areaChart),
93                     cmp.
↵horizontalList(stackedBar3DChart, pieChart))
94                     .pageFooter(Templates.footerComponent)
95                     .setDataSource(createDataSource())
96                     .show();
97         } catch (DRException e) {
98             e.printStackTrace();
99         }
100     }
101
102     private JRDataSource createDataSource() {
103         DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↵"stock3");
104         dataSource.add("Laptop", 95, 80, 50);
105         dataSource.add("Tablet", 170, 100, 80);
106         dataSource.add("Smartphone", 120, 80, 60);
107         return dataSource;
108     }
109
110     public static void main(String[] args) {
111         new ChartShowPercentagesReport();
112     }
113 }

```

1.18.6 Chart Show Values



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chartcustomization;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.chart.AreaChartBuilder;
28  import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
29  import net.sf.dynamicreports.report.builder.chart.LineChartBuilder;
30  import net.sf.dynamicreports.report.builder.chart.PieChartBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.chart.StackedBar3DChartBuilder;
32 import net.sf.dynamicreports.report.builder.chart.StackedBarChartBuilder;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class ChartShowValuesReport {
42
43     public ChartShowValuesReport() {
44         build();
45     }
46
47     private void build() {
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
49         TextColumnBuilder<Integer> stock1Column = col.column("Stock 1",
↳ "stock1", type.integerType());
50         TextColumnBuilder<Integer> stock2Column = col.column("Stock 2",
↳ "stock2", type.integerType());
51         TextColumnBuilder<Integer> stock3Column = col.column("Stock 3",
↳ "stock3", type.integerType());
52
53         BarChartBuilder barChart = cht.barChart()
54             .setShowValues(true)
55             .setCategory(itemColumn)
56             .series(cht.serie(stock1Column), cht.
↳ serie(stock2Column), cht.serie(stock3Column));
57         StackedBarChartBuilder stackedBarChart = cht.stackedBarChart()
58             .setShowValues(true)
59             .setCategory(itemColumn)
60             .series(cht.serie(stock1Column), cht.
↳ serie(stock2Column), cht.serie(stock3Column));
61         LineChartBuilder lineChart = cht.lineChart()
62             .setShowValues(true)
63             .setCategory(itemColumn)
64             .series(cht.serie(stock1Column), cht.
↳ serie(stock2Column), cht.serie(stock3Column));
65         AreaChartBuilder areaChart = cht.areaChart()
66             .setShowValues(true)
67             .setCategory(itemColumn)
68             .series(cht.serie(stock1Column), cht.
↳ serie(stock2Column), cht.serie(stock3Column));
69         StackedBar3DChartBuilder stackedBar3DChart = cht.stackedBar3DChart()
70             .setShowValues(true)
71             .setCategory(itemColumn)
72             .series(cht.serie(stock1Column), cht.
↳ serie(stock2Column), cht.serie(stock3Column));
73         PieChartBuilder pieChart = cht.pieChart()
74             .setShowValues(true)
75             .setKey(itemColumn)
76             .series(cht.serie(stock1Column));
77
78         try {

```

(continues on next page)

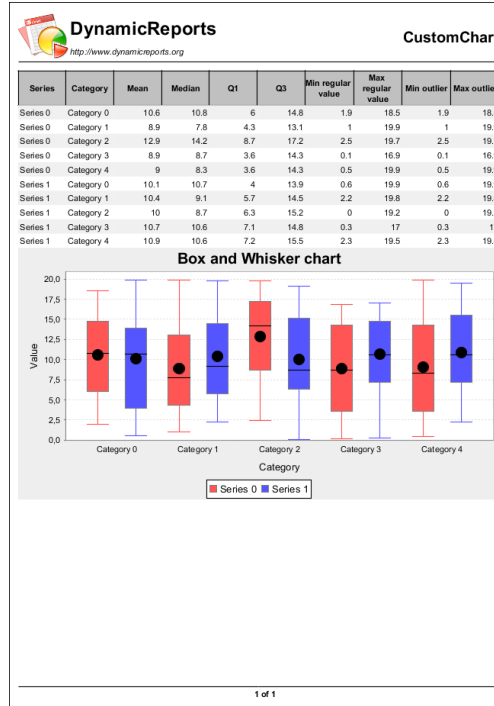
(continued from previous page)

```

79         report()
80             .setTemplate(Templates.reportTemplate)
81             .columns(itemColumn, stock1Column, ↵
↵ stock2Column, stock3Column)
82             .title(Templates.createTitleComponent(
↵ "ChartShowValues"))
83             .summary(
84                 cmp.horizontalList(barChart, ↵
↵ stackedBarChart),
85                 cmp.horizontalList(lineChart, ↵
↵ areaChart),
86                 cmp.
↵ horizontalList(stackedBar3DChart, pieChart))
87             .pageFooter(Templates.footerComponent)
88             .setDataSource(createDataSource())
89             .show();
90     } catch (DRException e) {
91         e.printStackTrace();
92     }
93 }
94
95 private JRDataSource createDataSource() {
96     DRDataSource dataSource = new DRDataSource("item", "stock1", "stock2",
↵ "stock3");
97     dataSource.add("Laptop", 95, 80, 50);
98     dataSource.add("Tablet", 170, 100, 80);
99     dataSource.add("Smartphone", 120, 80, 60);
100     return dataSource;
101 }
102
103 public static void main(String[] args) {
104     new ChartShowValuesReport();
105 }
106 }

```

1.18.7 Custom Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import org.jfree.chart.ChartFactory;
28 import org.jfree.chart.JFreeChart;
29 import org.jfree.data.statistics.BoxAndWhiskerItem;
30 import org.jfree.data.statistics.DefaultBoxAndWhiskerCategoryDataset;

```

(continues on next page)

(continued from previous page)

```

31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.report.base.AbstractScriptlet;
34 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
35 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.definition.ReportParameters;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.charts.util.DrawChartRendererImpl;
40 import net.sf.jasperreports.engine.JRDataSource;
41 import net.sf.jasperreports.renderers.Renderable;
42
43 /**
44  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
45  */
46 public class CustomChartReport {
47     private DefaultBoxAndWhiskerCategoryDataset dataset = new
↳DefaultBoxAndWhiskerCategoryDataset();
48
49     public CustomChartReport() {
50         build();
51     }
52
53     private void build() {
54         TextColumnBuilder<String> seriesColumn = col.column("Series", "series
↳", type.stringType());
55         TextColumnBuilder<String> categoryColumn = col.column("Category",
↳"category", type.stringType());
56         TextColumnBuilder<Double> meanColumn = col.column("Mean", "mean",
↳type.doubleType());
57         TextColumnBuilder<Double> medianColumn = col.column("Median", "median
↳", type.doubleType());
58         TextColumnBuilder<Double> q1Column = col.column("Q1", "q1", type.
↳doubleType());
59         TextColumnBuilder<Double> q3Column = col.column("Q3", "q3", type.
↳doubleType());
60         TextColumnBuilder<Double> minRegularValueColumn = col.column("Min
↳regular value", "minregularvalue", type.doubleType());
61         TextColumnBuilder<Double> maxRegularValueColumn = col.column("Max
↳regular value", "maxregularvalue", type.doubleType());
62         TextColumnBuilder<Double> minOutlierColumn = col.column("Min outlier",
↳ "minoutlier", type.doubleType());
63         TextColumnBuilder<Double> maxOutlierColumn = col.column("Max outlier",
↳ "maxoutlier", type.doubleType());
64
65         try {
66             report()
67                 .setTemplate(Templates.reportTemplate)
68                 .scriptlets(new ReportScriptlet())
69                 .columns(seriesColumn, categoryColumn,
↳meanColumn, medianColumn, q1Column, q3Column, minRegularValueColumn,
↳maxRegularValueColumn, minOutlierColumn,
70                             maxOutlierColumn)
71                 .title(Templates.createTitleComponent(
↳"CustomChart"))
72                 .summary(
73                             cmp.image(new
↳ChartExpression()).setFixedHeight(300))

```

(continues on next page)

(continued from previous page)

```

74         .pageFooter(Templates.footerComponent)
75         .setDataSource(createDataSource())
76         .show();
77     } catch (DRException e) {
78         e.printStackTrace();
79     }
80 }
81
82 private JRDataSource createDataSource() {
83     DRDataSource dataSource = new DRDataSource("series", "category", "mean
↪", "median", "q1", "q3", "minregularvalue", "maxregularvalue", "minoutlier",
84         "maxoutlier");
85     dataSource.add("Series 0", "Category 0", 10.55d, 10.75d, 6.05d, 14.
↪76d, 1.93d, 18.51d, 1.93d, 18.51d);
86     dataSource.add("Series 0", "Category 1", 8.92d, 7.78d, 4.32d, 13.07d,
↪1.01d, 19.89d, 1.01d, 19.89d);
87     dataSource.add("Series 0", "Category 2", 12.88d, 14.19d, 8.72d, 17.
↪23d, 2.48d, 19.74d, 2.48d, 19.74d);
88     dataSource.add("Series 0", "Category 3", 8.87d, 8.68d, 3.55d, 14.26d,
↪0.13d, 16.87d, 0.13d, 16.87d);
89     dataSource.add("Series 0", "Category 4", 9.05d, 8.33d, 3.61d, 14.28d,
↪0.46d, 19.86d, 0.46d, 19.86d);
90     dataSource.add("Series 1", "Category 0", 10.11d, 10.69d, 3.96d, 13.
↪93d, 0.58d, 19.9d, 0.58d, 19.9d);
91     dataSource.add("Series 1", "Category 1", 10.43d, 9.13d, 5.72d, 14.46d,
↪2.25d, 19.79d, 2.25d, 19.79d);
92     dataSource.add("Series 1", "Category 2", 10.04d, 8.71d, 6.32d, 15.15d,
↪0.04d, 19.15d, 0.04d, 19.15d);
93     dataSource.add("Series 1", "Category 3", 10.67d, 10.56d, 7.14d, 14.
↪76d, 0.27d, 16.99d, 0.27d, 16.99d);
94     dataSource.add("Series 1", "Category 4", 10.91d, 10.63d, 7.18d, 15.
↪51d, 2.3d, 19.53d, 2.3d, 19.53d);
95     return dataSource;
96 }
97
98 public static void main(String[] args) {
99     new CustomChartReport();
100 }
101
102 private class ReportScriptlet extends AbstractScriptlet {
103
104     @Override
105     public void afterDetailEval(ReportParameters reportParameters) {
106         super.afterDetailEval(reportParameters);
107         String series = reportParameters.getValue("series");
108         String category = reportParameters.getValue("category");
109         Double mean = reportParameters.getValue("mean");
110         Double median = reportParameters.getValue("median");
111         Double q1 = reportParameters.getValue("q1");
112         Double q3 = reportParameters.getValue("q3");
113         Double minRegularValue = reportParameters.getValue(
↪"minregularvalue");
114         Double maxRegularValue = reportParameters.getValue(
↪"maxregularvalue");
115         Double minOutlier = reportParameters.getValue("minoutlier");
116         Double maxOutlier = reportParameters.getValue("maxoutlier");
117         BoxAndWhiskerItem item = new BoxAndWhiskerItem(mean, median,
↪q1, q3, minRegularValue, maxRegularValue, minOutlier, maxOutlier, null(continues on next page)

```

(continued from previous page)

```

118         dataset.add(item, series, category);
119     }
120 }
121
122 private class ChartExpression extends AbstractSimpleExpression<Renderable> {
123     private static final long serialVersionUID = 1L;
124
125     @Override
126     public Renderable evaluate(ReportParameters reportParameters) {
127         JFreeChart chart = ChartFactory.createBoxAndWhiskerChart("Box_
128 and Whisker chart", "Category", "Value", dataset, true);
129         return new DrawChartRendererImpl(chart, null);
130     }
131 }
132 }

```

1.18.8 Group Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.chart.Bar3DChartBuilder;
31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class GroupChartReport {
41
42     public GroupChartReport() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> countryColumn = col.column("Country",
48 ↪ "country", type.stringType());
49         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
50 ↪ type.stringType());
51         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
52 ↪ "quantity", type.integerType());
53         TextColumnBuilder<BigDecimal> salesColumn = col.column("Sales", "sales
54 ↪", type.bigDecimalType());
55
56         Bar3DChartBuilder chart = cht.bar3DChart()
57             .setFixedHeight(180)
58             .setCategory(itemColumn)
59             .series(cht.serie(quantityColumn), cht
60 ↪ serie(salesColumn))
61             .setCategoryAxisFormat(
62                 cht.axisFormat().setLabel("Item"));
63
64         ColumnGroupBuilder countryGroup = grp.group(countryColumn)
65             .footer(chart);
66
67         try {

```

(continues on next page)

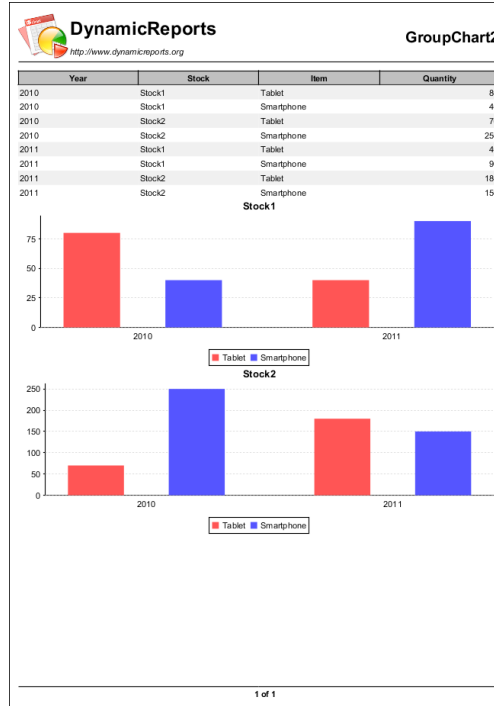
(continued from previous page)

```

63         report()
64             .setTemplate(Templates.reportTemplate)
65             .columns(countryColumn, itemColumn, ↵
↵quantityColumn, salesColumn)
66             .title(Templates.createTitleComponent(
↵"GroupChart"))
67             .groupBy(countryGroup)
68             .summary(
69                 cmp.text("All countries").
↵setStyle(Templates.bold12CenteredStyle),
70                 chart)
71             .pageFooter(Templates.footerComponent)
72             .setDataSource(createDataSource())
73             .show();
74     } catch (DRException e) {
75         e.printStackTrace();
76     }
77 }
78
79 private JRDataSource createDataSource() {
80     DRDataSource dataSource = new DRDataSource("country", "item",
↵"quantity", "sales");
81     dataSource.add("USA", "Tablet", 170, new BigDecimal(100));
82     dataSource.add("USA", "Laptop", 90, new BigDecimal(280));
83     dataSource.add("USA", "Smartphone", 120, new BigDecimal(250));
84     dataSource.add("Canada", "Tablet", 120, new BigDecimal(80));
85     dataSource.add("Canada", "Laptop", 150, new BigDecimal(310));
86     dataSource.add("Canada", "Smartphone", 100, new BigDecimal(180));
87     return dataSource;
88 }
89
90 public static void main(String[] args) {
91     new GroupChartReport();
92 }
93 }

```

1.18.9 Group Chart 2



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.List;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
34 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
35 import net.sf.dynamicreports.report.builder.style.FontBuilder;
36 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class GroupChartReport2 {
46
47     public GroupChartReport2() {
48         build();
49     }
50
51     private void build() {
52         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
53
54         TextColumnBuilder<String> yearColumn = col.column("Year", "year",
↳ type.stringType());
55         TextColumnBuilder<String> stockColumn = col.column("Stock", "stock",
↳ type.stringType());
56         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
57         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳ "quantity", type.integerType());
58
59         BarChartBuilder chart1 = cht.barChart()
60             .setTitle(new ChartTitleExpression(stockColumn))
61             .setTitleFont(boldFont)
62             .setCategory(yearColumn)
63             .series(
64                 cht.serie(quantityColumn).
↳ setSeries(itemColumn));
65
66         ColumnGroupBuilder stockGroup = grp.group(stockColumn)
67             .setHeaderLayout(GroupHeaderLayout.EMPTY)
68             .footer(chart1);
69
70         JasperReportBuilder subReport = report()
71             .sortBy(stockColumn)
72             .groupBy(stockGroup)
73             .setDataSource(createDataSource());
74
75         try {
76             report()
77                 .setTemplate(Templates.reportTemplate)
78                 .columns(yearColumn, stockColumn, itemColumn,
↳ quantityColumn)
79                 .title(Templates.createTitleComponent(
↳ "GroupChart2"))
80                 .summary(cmp.subreport(subReport))

```

(continues on next page)

(continued from previous page)

```

81         .pageFooter(Templates.footerComponent)
82         .setDataSource(createDataSource())
83         .show();
84     } catch (DRException e) {
85         e.printStackTrace();
86     }
87 }
88
89 private JRDataSource createDataSource() {
90     DRDataSource dataSource = new DRDataSource("year", "stock", "item",
↪ "quantity");
91     dataSource.add("2010", "Stock1", "Tablet", 80);
92     dataSource.add("2010", "Stock1", "Smartphone", 40);
93     dataSource.add("2010", "Stock2", "Tablet", 70);
94     dataSource.add("2010", "Stock2", "Smartphone", 250);
95     dataSource.add("2011", "Stock1", "Tablet", 40);
96     dataSource.add("2011", "Stock1", "Smartphone", 90);
97     dataSource.add("2011", "Stock2", "Tablet", 180);
98     dataSource.add("2011", "Stock2", "Smartphone", 150);
99     return dataSource;
100 }
101
102 public static void main(String[] args) {
103     new GroupChartReport2();
104 }
105
106 private class ChartTitleExpression extends AbstractComplexExpression<String> {
107     private static final long serialVersionUID = 1L;
108
109     public ChartTitleExpression(TextColumnBuilder<String> stockColumn) {
110         addExpression(stockColumn);
111     }
112
113     @Override
114     public String evaluate(List<?> values, ReportParameters_
↪ reportParameters) {
115         return (String) values.get(0);
116     }
117 }
118 }

```

1.18.10 Group Count Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.chartcustomization;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.VariableBuilder;
28  import net.sf.dynamicreports.report.builder.chart.Bar3DChartBuilder;
29  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
30  import net.sf.dynamicreports.report.constant.Calculation;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.constant.Evaluation;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class GroupCountChartReport {
40
41     public GroupCountChartReport() {
42         build();
43     }
44
45     private void build() {
46         TextColumnBuilder<String> countryColumn = col.column("Country",
47 ↪ "country", type.stringType());
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49 ↪ type.stringType());
50         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
51 ↪ "quantity", type.integerType());
52
53         Bar3DChartBuilder chart1 = cht.bar3DChart()
54             .setFixedHeight(180)
55             .setCategory(countryColumn)
56             .series(cht.serie(exp.number(1)).setLabel("Items_
57 ↪ (group count)"))
58             .setCategoryAxisFormat(
59                 cht.axisFormat().setLabel("Country"));
60
61         VariableBuilder<Integer> itemVariable = variable(itemColumn,
62 ↪ Calculation.DISTINCT_COUNT);
63         itemVariable.setResetType(Evaluation.FIRST_GROUP);
64
65         Bar3DChartBuilder chart2 = cht.bar3DChart()
66             .setFixedHeight(180)
67             .setCategory(countryColumn)
68             .series(cht.serie(itemVariable).setLabel("Items_
69 ↪ (group distinct count)"))
70             .setCategoryAxisFormat(
71                 cht.axisFormat().setLabel("Country"));
72
73         try {
74             report()
75                 .setTemplate(Templates.reportTemplate)
76                 .columns(countryColumn, itemColumn,
77 ↪ quantityColumn)
78                 .title(Templates.createTitleComponent(
79 ↪ "GroupCountChart"))
80                 .groupBy(grp.group(countryColumn))
81                 .summary(cmp.horizontalList(chart1, chart2))
82                 .pageFooter(Templates.footerComponent)
83                 .setDataSource(createDataSource())
84                 .show();
85         } catch (DRException e) {
86             e.printStackTrace();
87         }
88     }
89 }

```

(continues on next page)

(continued from previous page)

```

80     }
81
82     private JRDataSource createDataSource() {
83         DRDataSource dataSource = new DRDataSource("country", "item",
84         ↪ "quantity");
85         dataSource.add("USA", "Tablet", 170);
86         dataSource.add("USA", "Tablet", 80);
87         dataSource.add("USA", "Laptop", 90);
88         dataSource.add("USA", "Smartphone", 120);
89         dataSource.add("Canada", "Tablet", 120);
90         dataSource.add("Canada", "Laptop", 150);
91         return dataSource;
92     }
93
94     public static void main(String[] args) {
95         new GroupCountChartReport();
96     }

```

1.18.11 Series Bar Chart



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *

```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.chartcustomization;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
28 import net.sf.dynamicreports.report.builder.style.FontBuilder;
29 import net.sf.dynamicreports.report.datasource.DRDataSource;
30 import net.sf.dynamicreports.report.exception.DRException;
31 import net.sf.jasperreports.engine.JRDataSource;
32
33 /**
34  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
35  */
36 public class SeriesBarChartReport {
37
38     public SeriesBarChartReport() {
39         build();
40     }
41
42     private void build() {
43         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
44
45         TextColumnBuilder<String> stateColumn = col.column("State", "state",
↪ type.stringType());
46         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪ type.stringType());
47         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪ "quantity", type.integerType());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(stateColumn, itemColumn,
↪ quantityColumn)
53                 .title(Templates.createTitleComponent(
↪ "SeriesBarChart"))
54                 .summary(
55                     cht.barChart()
56                         .setTitle(
↪ "Bar chart")
57                         .
↪ setTitleFont(boldFont)
58                         .
↪ setCategory(stateColumn)

```

(continues on next page)


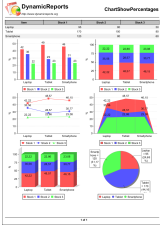
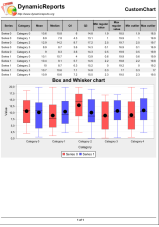




(continued from previous page)

```

59                                     .series (
60                                     cht.
↪serie (quantityColumn) .setSeries (itemColumn))
61                                     .
62                                     cht.
↪setCategoryAxisFormat (
63                                     .pageFooter (Templates.footerComponent)
64                                     .setDataSource (createDataSource ())
65                                     .show ();
66                                     } catch (DRException e) {
67                                     e.printStackTrace ();
68                                     }
69     }
70
71     private JRDataSource createDataSource () {
72         DRDataSource dataSource = new DRDataSource ("state", "item", "quantity
↪");
73         dataSource.add ("New York", "Tablet", 170);
74         dataSource.add ("New York", "Laptop", 100);
75         dataSource.add ("New York", "Smartphone", 120);
76         dataSource.add ("Washington", "Tablet", 110);
77         dataSource.add ("Washington", "Laptop", 90);
78         dataSource.add ("Washington", "Smartphone", 160);
79         return dataSource;
80     }
81
82     public static void main (String [] args) {
83         new SeriesBarChartReport ();
84     }
85 }

```

Table 3: Chart customization Examples

 ChartCustomizerReport	 ChartLayoutReport	 ChartSeriesColorsBy- NameReport	 ChartSeriesOrderRe- port
 ChartShowPercentagesRe- port	 ChartShowValuesRe- port	 CustomChartReport	 GroupChartReport
 GroupChartReport2	 GroupCountChartRe- port	 SeriesBarChartReport	

1.19 Column

1.19.1 Boolean Column

BooleanColumn										
Boolean	TEXT_STYLE	TEXT_STYLE_2	TEXT_STYLE_3	TEXT_STYLE_4	TEXT_STYLE_5	TEXT_STYLE_6	TEXT_STYLE_7	TEXT_STYLE_8	TEXT_STYLE_9	TEXT_STYLE_10
True	True	True	True	True	True	True	True	True	True	True
False	False	False	False	False	False	False	False	False	False	False

```
/**
 * DynamicReports - Free Java reporting library for creating reports dynamically
 *
 * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

```

5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.constant.BooleanComponentType;
28 import net.sf.dynamicreports.report.constant.PageOrientation;
29 import net.sf.dynamicreports.report.constant.PageType;
30 import net.sf.dynamicreports.report.datasource.DRDataSource;
31 import net.sf.dynamicreports.report.exception.DRException;
32 import net.sf.jasperreports.engine.JRDataSource;
33
34 /**
35  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36  */
37 public class BooleanColumnReport {
38
39     public BooleanColumnReport() {
40         build();
41     }
42
43     private void build() {
44         try {
45             report()
46                 .setTemplate(Templates.reportTemplate)
47                 .setPageFormat(PageType.A3, PageOrientation.
↳LANDSCAPE)
48                 .columns(
49                     col.booleanColumn(
↳"Boolean\ndefault", "boolean"),
50                     col.booleanColumn(
↳"Boolean\nTEXT_TRUE_FALSE", "boolean").setComponentType(BooleanComponentType.TEXT_
↳TRUE_FALSE).setEmptyWhenNullValue(true),
51                     col.booleanColumn(
↳"Boolean\nTEXT_YES_NO", "boolean").setComponentType(BooleanComponentType.TEXT_YES_
↳NO),
52                     col.booleanColumn(
↳"Boolean\nIMAGE_STYLE_1", "boolean").setComponentType(BooleanComponentType.IMAGE_
↳STYLE_1).setEmptyWhenNullValue(true),
53                     col.booleanColumn(
↳"Boolean\nIMAGE_STYLE_2", "boolean").setComponentType(BooleanComponentType.IMAGE_
↳STYLE_2),

```

(continues on next page)


(continued from previous page)

```

54         col.booleanColumn (
↪ "Boolean\nIMAGE_STYLE_3", "boolean").setComponentType (BooleanComponentType.IMAGE_
↪ STYLE_3),
55         col.booleanColumn (
↪ "Boolean\nIMAGE_STYLE_4", "boolean").setComponentType (BooleanComponentType.IMAGE_
↪ STYLE_4),
56         col.booleanColumn (
↪ "Boolean\nIMAGE_CHECKBOX_1", "boolean").setComponentType (BooleanComponentType.IMAGE_
↪ CHECKBOX_1),
57         col.booleanColumn (
↪ "Boolean\nIMAGE_CHECKBOX_2", "boolean").setComponentType (BooleanComponentType.IMAGE_
↪ CHECKBOX_2),
58         col.booleanColumn (
↪ "Boolean\nIMAGE_BALL", "boolean").setComponentType (BooleanComponentType.IMAGE_BALL))
59         .title (Templates.createTitleComponent (
↪ "BooleanColumn"))
60         .pageFooter (Templates.footerComponent)
61         .setDataSource (createDataSource ())
62         .show ();
63     } catch (DRException e) {
64         e.printStackTrace ();
65     }
66 }
67
68 private JRDataSource createDataSource () {
69     DRDataSource dataSource = new DRDataSource ("boolean");
70     dataSource.add (true);
71     dataSource.add (false);
72     dataSource.add ();
73     dataSource.add (false);
74     return dataSource;
75 }
76
77 public static void main (String[] args) {
78     new BooleanColumnReport ();
79 }
80 }

```

1.19.2 Calculated Column

DynamicReports		CalculatedColumn					
 http://www.dynamicreports.org							
A	B	A * B	A / B	A + B	A - B	A * B + 6	(A*B+6) / 5 + 1
10	5	50.00	2.00	15.00	5.00	56.00	12.20

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.column;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)


(continued from previous page)

```

31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class CalculatedColumnReport {
39
40     public CalculatedColumnReport() {
41         build();
42     }
43
44     private void build() {
45         TextColumnBuilder<Integer> column1 = col.column("A", "field1", type.
46 ↪integerType());
47         TextColumnBuilder<Integer> column2 = col.column("B", "field2", type.
48 ↪integerType());
49         TextColumnBuilder<BigDecimal> column3 = column1.multiply(column2).
50 ↪setTitle("A * B");
51         TextColumnBuilder<BigDecimal> column4 = column1.divide(2, column2).
52 ↪setTitle("A / B");
53         TextColumnBuilder<BigDecimal> column5 = column1.add(column2).setTitle(
54 ↪"A + B");
55         TextColumnBuilder<BigDecimal> column6 = column1.subtract(column2).
56 ↪setTitle("A - B");
57         TextColumnBuilder<BigDecimal> column7 = column3.add(6).setTitle("A *
58 ↪B + 6");
59         TextColumnBuilder<BigDecimal> column8 = column7.divide(2, 5).add(1).
60 ↪setTitle("(A*B+6) / 5 + 1");
61
62         try {
63             report()
64                 .setTemplate(Templates.reportTemplate)
65                 .columns(
66 ↪column1, column2, column3,
67 ↪column4, column5, column6, column7, column8)
68                 .title(Templates.createTitleComponent(
69 ↪"CalculatedColumn"))
70                 .pageFooter(Templates.footerComponent)
71                 .setDataSource(createDataSource())
72                 .show();
73         } catch (DRException e) {
74             e.printStackTrace();
75         }
76     }
77
78     private JRDataSource createDataSource() {
79         DRDataSource dataSource = new DRDataSource("field1", "field2");
80         dataSource.add(10, 5);
81         return dataSource;
82     }
83
84     public static void main(String[] args) {
85         new CalculatedColumnReport();
86     }
87 }

```

1.19.3 Column Data Types

DynamicReports		ColumnDataTypes					
 http://www.dynamicreports.org							
Item	Quantity	Unit price	Order date	Order date	Order year	Order month	Order day
Notebook	1	500.00	01/22/2015	01/22/2015 9:18:22,840 AM	2015	January	22
1 of 1							

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.column;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.Date;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)


(continued from previous page)

```

31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class ColumnDataTypesReport {
39
40     public ColumnDataTypesReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             report()
47                 .setTemplate(Templates.reportTemplate)
48                 .columns(
49                     col.column("Item", "item",
50 ↪ type.stringType()),
51                     col.column("Quantity",
52 ↪ "quantity", type.integerType()),
53                     col.column("Unit price",
54 ↪ "unitprice", type.bigDecimalType()),
55                     col.column("Order date",
56 ↪ "orderdate", type.dateType()),
57                     col.column("Order date",
58 ↪ "orderdate", type.dateYearToFractionType()),
59                     col.column("Order year",
60 ↪ "orderdate", type.dateYearType()),
61                     col.column("Order month",
62 ↪ "orderdate", type.dateMonthType()),
63                     col.column("Order day",
64 ↪ "orderdate", type.dateDayType()))
65                 .title(Templates.createTitleComponent(
66 ↪ "ColumnDataTypes"))
67                 .pageFooter(Templates.footerComponent)
68                 .setDataSource(createDataSource())
69                 .show();
70         } catch (DRException e) {
71             e.printStackTrace();
72         }
73     }
74
75     private JRDataSource createDataSource() {
76         DRDataSource dataSource = new DRDataSource("item", "orderdate",
77 ↪ "quantity", "unitprice");
78         dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
79         return dataSource;
80     }
81
82     public static void main(String[] args) {
83         new ColumnDataTypesReport();
84     }
85 }

```

1.19.4 Column Detect Data Type

DynamicReports		ColumnDetectDataTypes					
		http://www.dynamicreports.org					
Item	Quantity	Unit price	Order date	Order date	Order year	Order month	Order day
Notebook	1	500.00	01/22/2015	01/22/2015 9:18:22,877 AM	2015	January	22
1 of 1							

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.column;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.ArrayList;
29  import java.util.Date;
30  import java.util.List;

```

(continues on next page)

(continued from previous page)

```

31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.definition.datatype.DRIDataType;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class ColumnDetectDataTypeReport {
43
44     public ColumnDetectDataTypeReport() {
45         build();
46     }
47
48     @SuppressWarnings({ "rawtypes", "unchecked" })
49     private void build() {
50         try {
51             JasperReportBuilder report = report();
52             for (Column column : createColumns()) {
53                 report.addColumn(col.column(column.title, column.
↳field, (DRIDataType) type.detectType(column.dataType)));
54             }
55             report
56                 .setTemplate(Templates.reportTemplate)
57                 .title(Templates.createTitleComponent(
↳"ColumnDetectDataTypes"))
58                 .pageFooter(Templates.footerComponent)
59                 .setDataSource(createDataSource())
60                 .show();
61         } catch (DRException e) {
62             e.printStackTrace();
63         }
64     }
65
66     private JRDataSource createDataSource() {
67         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↳"quantity", "unitprice");
68         dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
69         return dataSource;
70     }
71
72     private List<Column> createColumns() {
73         List<Column> columns = new ArrayList<Column>();
74         columns.add(new Column("Item", "item", "string")); // dataType =
↳"String", "STRING", "java.lang.String", "text"
75         columns.add(new Column("Quantity", "quantity", "integer")); //
↳dataType = "Integer", "INTEGER", "java.lang.Integer"
76         columns.add(new Column("Unit price", "unitprice", "bigDecimal")); //
↳dataType = "bigdecimal", "BIGDECIMAL", "java.math.BigDecimal"
77         columns.add(new Column("Order date", "orderdate", "date")); //
↳dataType = "Date", "DATE", "java.util.Date"
78         columns.add(new Column("Order date", "orderdate", "dateYearToFraction
↳")); // dataType = "dateyeartofraction", "DATEYEARTOFRACTION"
79         columns.add(new Column("Order year", "orderdate", "dateYear")); //
↳dataType = "DateYear", "dateyear", "DATEYEAR"


```

(continues on next page)

(continued from previous page)

```
80         columns.add(new Column("Order month", "orderdate", "dateMonth")); //
      ↳ dataType = "DateMonth", "datemonth", "DATEMONTH"
81         columns.add(new Column("Order day", "orderdate", "dateDay")); //
      ↳ dataType = "DateDay", "dateday", "DATEDAY"
82         return columns;
83     }
84
85     private class Column {
86         private String title;
87         private String field;
88         private String dataType;
89
90         private Column(String title, String field, String dataType) {
91             this.title = title;
92             this.field = field;
93             this.dataType = dataType;
94         }
95     }
96
97     public static void main(String[] args) {
98         new ColumnDetectDataTypeReport();
99     }
100 }
```

1.19.5 Column List Data Type



DynamicReports
<http://www.dynamicreports.org>

ColumnListDataType

Item	Quantity	Comments
Book	10	comment1 comment2 comment3
Notebook	20	comment1 comment2

1 of 1

```
1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
```

(continues on next page)

(continued from previous page)

```

4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.util.ArrayList;
28 import java.util.List;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.exception.DRException;
32 import net.sf.jasperreports.engine.JRDataSource;
33 import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class ColumnListDataTypeReport {
39
40     public ColumnListDataTypeReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             report()
47                 .setTemplate(Templates.reportTemplate)
48                 .columns(
49                     col.column("Item", "item",
50 ↪ type.stringType()),
51                     col.column("Quantity",
52 ↪ "quantity", type.integerType()),
53                     col.column("Comments",
54 ↪ "comments", type.listType())
55                 .title(Templates.createTitleComponent(
56 ↪ "ColumnListDataType"))
57                 .pageFooter(Templates.footerComponent)
58                 .setDataSource(createDataSource())
59                 .show();
60         } catch (DRException e) {

```

(continues on next page)

(continued from previous page)

```

57         e.printStackTrace();
58     }
59 }
60
61 private JRDataSource createDataSource() {
62     List<ReportData> datasource = new ArrayList<ReportData>();
63
64     ReportData data = new ReportData();
65     List<String> comments = new ArrayList<String>();
66     comments.add("comment1");
67     comments.add("comment2");
68     comments.add("comment3");
69     data.setItem("Book");
70     data.setQuantity(new Integer(10));
71     data.setComments(comments);
72     datasource.add(data);
73
74     data = new ReportData();
75     comments = new ArrayList<String>();
76     comments.add("comment1");
77     comments.add("comment2");
78     data.setItem("Notebook");
79     data.setQuantity(new Integer(20));
80     data.setComments(comments);
81     datasource.add(data);
82
83     return new JRBeanCollectionDataSource(datasource);
84 }
85
86 public class ReportData {
87     private String item;
88     private Integer quantity;
89     private List<String> comments;
90
91     public String getItem() {
92         return item;
93     }
94
95     public void setItem(String item) {
96         this.item = item;
97     }
98
99     public Integer getQuantity() {
100         return quantity;
101     }
102
103     public void setQuantity(Integer quantity) {
104         this.quantity = quantity;
105     }
106
107     public List<String> getComments() {
108         return comments;
109     }
110
111     public void setComments(List<String> comments) {
112         this.comments = comments;
113     }

```

(continues on next page)

(continued from previous page)

```

114     }
115
116     public static void main(String[] args) {
117         new ColumnListDataTypeReport();
118     }
119 }

```

1.19.6 Column Subreport Data

DynamicReports <small>http://www.dynamicreports.org</small>		ColumnSubreportData
Item	Quantity	Comments
Book	10	comment1 comment2 comment3
Notebook	20	comment1 comment2

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.

```

(continues on next page)

(continued from previous page)

```

21  */
22
23  package net.sf.dynamicreports.examples.column;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.util.ArrayList;
28  import java.util.Collection;
29  import java.util.HashMap;
30  import java.util.List;
31  import java.util.Map;
32
33  import net.sf.dynamicreports.examples.Templates;
34  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
35  import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
36  import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
37  import net.sf.dynamicreports.report.definition.ReportParameters;
38  import net.sf.dynamicreports.report.exception.DRException;
39  import net.sf.jasperreports.engine.JRDataSource;
40  import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
41  import net.sf.jasperreports.engine.data.JRMapCollectionDataSource;
42
43  /**
44   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
45   */
46  public class ColumnSubreportDataReport {
47
48      public ColumnSubreportDataReport() {
49          build();
50      }
51
52      private void build() {
53          SubreportBuilder subreport = cmp.subreport(new SubreportDesign())
54              .setDataSource(new SubreportData());
55
56          try {
57              report()
58                  .setTemplate(Templates.reportTemplate)
59                  .fields(field("comments", List.class))
60                  .columns(
61                      col.column("Item", "item",
62                          ↪ type.stringType()),
63                      col.column("Quantity",
64                          ↪ "quantity", type.integerType()),
65                      col.componentColumn("Comments",
66                          ↪ " ", subreport))
67                  .title(Templates.createTitleComponent(
68                      ↪ "ColumnSubreportData"))
69                  .pageFooter(Templates.footerComponent)
70                  .setDataSource(createDataSource())
71                  .show();
72          } catch (DRException e) {
73              e.printStackTrace();
74          }
75      }
76
77      private class SubreportDesign extends AbstractSimpleExpression
78          ↪ <JasperReportBuilder> {

```

(continues on next page)

(continued from previous page)

```

73         private static final long serialVersionUID = 1L;
74
75         @Override
76         public JasperReportBuilder evaluate(ReportParameters_
77 ↪reportParameters) {
78             JasperReportBuilder report = report()
79                 .columns(col.column("comment", type.
80 ↪stringType()));
81             return report;
82         }
83
84         private class SubreportData extends AbstractSimpleExpression<JRDataSource> {
85             private static final long serialVersionUID = 1L;
86
87             @Override
88             public JRDataSource evaluate(ReportParameters reportParameters) {
89                 Collection<Map<String, ?>> value = reportParameters.getValue(
90 ↪"comments");
91                 return new JRMapCollectionDataSource(value);
92             }
93
94             private JRDataSource createDataSource() {
95                 List<ReportData> datasource = new ArrayList<ReportData>();
96
97                 ReportData data = new ReportData();
98                 List<Map<String, Object>> comments = new ArrayList<Map<String, Object>>
99 ↪();
100                 Map<String, Object> values = new HashMap<String, Object>();
101                 values.put("comment", "comment1");
102                 comments.add(values);
103                 values = new HashMap<String, Object>();
104                 values.put("comment", "comment2");
105                 comments.add(values);
106                 values = new HashMap<String, Object>();
107                 values.put("comment", "comment3");
108                 comments.add(values);
109                 data.setItem("Book");
110                 data.setQuantity(new Integer(10));
111                 data.setComments(comments);
112                 datasource.add(data);
113
114                 data = new ReportData();
115                 comments = new ArrayList<Map<String, Object>>();
116                 values = new HashMap<String, Object>();
117                 values.put("comment", "comment1");
118                 comments.add(values);
119                 values = new HashMap<String, Object>();
120                 values.put("comment", "comment2");
121                 comments.add(values);
122                 data.setItem("Notebook");
123                 data.setQuantity(new Integer(20));
124                 data.setComments(comments);
125                 datasource.add(data);
126
127                 return new JRBeanCollectionDataSource(datasource);













```

(continues on next page)

(continued from previous page)

```
126     }
127
128     public class ReportData {
129         private String item;
130         private Integer quantity;
131         private List<Map<String, Object>> comments;
132
133         public String getItem() {
134             return item;
135         }
136
137         public void setItem(String item) {
138             this.item = item;
139         }
140
141         public Integer getQuantity() {
142             return quantity;
143         }
144
145         public void setQuantity(Integer quantity) {
146             this.quantity = quantity;
147         }
148
149         public List<Map<String, Object>> getComments() {
150             return comments;
151         }
152
153         public void setComments(List<Map<String, Object>> comments) {
154             this.comments = comments;
155         }
156     }
157
158     public static void main(String[] args) {
159         new ColumnSubreportDataReport();
160     }
161 }
```

1.19.7 Component Column

DynamicReports <small>http://www.dynamicreports.org</small>		ComponentColumn	
Image	Item	Image	Item
	PDA		PDA 1100064882171958
	Camera		Camera 1100064882171958
	Camera		Camera 1100064882171958
	USB		USB 1100064882171958
	PDA		PDA 1100064882171958
	USB		USB 1100064882171958

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.InputStream;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
32 import net.sf.dynamicreports.report.builder.component.ImageBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.definition.ReportParameters;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class ComponentColumnReport {
42
43     public ComponentColumnReport() {
44         build();
45     }
46
47     private void build() {
48         try {
49             ImageBuilder image = cmp.image(new ImageExpression()).
↪ setFixedDimension(48, 48);
50             HorizontalListBuilder itemComponent = cmp.horizontalList(
51                 image,
52                 cmp.verticalList(
53                     cmp.text(new
↪
54                     BarcodeExpression()).setFixedHeight(24)));
55
56             report()
57                 .setTemplate(Templates.reportTemplate)
58                 .fields(
59                     field("image", String.class),
60                     field("barcode", String.
↪ class))
61                 .columns(
62                     col.componentColumn("Image",
↪
63                     col.column("Item", "item",
↪
64                     col.componentColumn("Item",
↪
65                 .title(Templates.createTitleComponent(
66                     "ComponentColumn"))
67                 .pageFooter(Templates.footerComponent)
68                 .setDataSource(createDataSource())
69                 .show();
70         } catch (DRException e) {
71             e.printStackTrace();
72         }
73
74     public class ImageExpression extends AbstractSimpleExpression<InputStream> {
75         private static final long serialVersionUID = 1L;
76
77         @Override
78         public InputStream evaluate(ReportParameters reportParameters) {
79             return Templates.class.getResourceAsStream("images/" +
↪ reportParameters.getValue("image") + ".png");

```

(continues on next page)


(continued from previous page)

```

80         }
81     }
82
83     public class ItemExpression extends AbstractSimpleExpression<String> {
84         private static final long serialVersionUID = 1L;
85
86         @Override
87         public String evaluate(ReportParameters reportParameters) {
88             return reportParameters.getValue("item");
89         }
90     }
91
92     public class BarcodeExpression extends AbstractSimpleExpression<String> {
93         private static final long serialVersionUID = 1L;
94
95         @Override
96         public String evaluate(ReportParameters reportParameters) {
97             return reportParameters.getValue("barcode");
98         }
99     }
100
101     private JRDataSource createDataSource() {
102         DRDataSource dataSource = new DRDataSource("item", "image", "barcode
103         ↪");
104         dataSource.add("PDA", "pda", "100264832717658");
105         dataSource.add("Camera", "camera", "100364875790352");
106         dataSource.add("Camera", "camera", "100764935316351");
107         dataSource.add("USB", "usb", "100864565780343");
108         dataSource.add("PDA", "pda", "100264865712551");
109         dataSource.add("USB", "usb", "100268834723431");
110         return dataSource;
111     }
112
113     public static void main(String[] args) {
114         new ComponentColumnReport();
115     }

```

1.19.8 Conversion Column

DynamicReports		ConversionColumn		
 http://www.dynamicreports.org				
Item	Order date	Order date	Quantity	Quantity
Notebook	01/01/2010	01/01/2010	100.90	100.90
Notebook	02/02/2010	02/02/2010	100.20	100.20
	01/01/2010	01/01/2010	201.10	201.10

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.text.DecimalFormat;
29 import java.text.DecimalFormatSymbols;
30 import java.text.ParseException;

```

(continues on next page)

(continued from previous page)

```

31 import java.text.SimpleDateFormat;
32 import java.util.Date;
33
34 import net.sf.dynamicreports.examples.Templates;
35 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
36 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class ConversionColumnReport {
46
47     public ConversionColumnReport() {
48         build();
49     }
50
51     private void build() {
52         try {
53             TextColumnBuilder<Date> orderDateColumn1 = col.column("Order_
↪date", new OrderDateColumn1())
54                 .setDataType(type.dateType());
55             TextColumnBuilder<Date> orderDateColumn2 = col.column("Order_
↪date", new OrderDateColumn2())
56                 .setDataType(type.dateType());
57             TextColumnBuilder<BigDecimal> quantityColumn1 = col.column(
↪"Quantity", new QuantityColumn1())
58                 .setDataType(type.bigDecimalType());
59             TextColumnBuilder<BigDecimal> quantityColumn2 = col.column(
↪"Quantity", new QuantityColumn2())
60                 .setDataType(type.bigDecimalType());
61
62             report()
63                 .setTemplate(Templates.reportTemplate)
64                 .fields(
65                     field("orderdate", String.
↪class),
66                     field("quantity", String.
↪class))
67                 .columns(
68                     col.column("Item", "item",
↪type.stringType()),
69                     orderDateColumn1,
70                     orderDateColumn2, quantityColumn1, quantityColumn2)
71                 .title(Templates.createTitleComponent(
↪"ConversionColumn"))
72                 .pageFooter(Templates.footerComponent)
73                 .subtotalsAtSummary(
74                     sbt.min(orderDateColumn1),
↪sbt.min(orderDateColumn2), sbt.sum(quantityColumn1), sbt.sum(quantityColumn2))
75                 .setDataSource(createDataSource())
76                 .show();
77         } catch (DRException e) {
78             e.printStackTrace();
79         }
80     }
81 }

```

(continues on next page)

(continued from previous page)

```

78         }
79     }
80
81     private JRDataSource createDataSource() {
82         DRDataSource dataSource = new DRDataSource("item", "orderdate",
83 ↪ "quantity");
84         dataSource.add("Notebook", "1/1/2010", "100.9");
85         dataSource.add("Notebook", "2/2/2010", "100.2");
86         return dataSource;
87     }
88
89     public static void main(String[] args) {
90         new ConversionColumnReport();
91     }
92
93     private class OrderDateColumn1 extends AbstractSimpleExpression<Date> {
94         private static final long serialVersionUID = 1L;
95
96         @Override
97         public Date evaluate(ReportParameters reportParameters) {
98             String value = reportParameters.getValue("orderdate");
99             try {
100                 return new SimpleDateFormat("MM/dd/yyyy",
101 ↪ reportParameters.getLocale()).parse(value);
102             } catch (ParseException e) {
103                 e.printStackTrace();
104             }
105             return null;
106         }
107     }
108
109     private class OrderDateColumn2 extends AbstractSimpleExpression<Date> {
110         private static final long serialVersionUID = 1L;
111
112         @Override
113         public Date evaluate(ReportParameters reportParameters) {
114             try {
115                 return type.dateType().stringToValue("orderdate",
116 ↪ reportParameters);
117             } catch (DRException e) {
118                 e.printStackTrace();
119             }
120             return null;
121         }
122     }
123
124     private class QuantityColumn1 extends AbstractSimpleExpression<BigDecimal> {
125         private static final long serialVersionUID = 1L;
126
127         @Override
128         public BigDecimal evaluate(ReportParameters reportParameters) {
129             String value = reportParameters.getValue("quantity");
130             try {
131                 Number number = new DecimalFormat("#,###.##", new
132 ↪ DecimalFormatSymbols(reportParameters.getLocale())).parse(value);
133                 return new BigDecimal(number.doubleValue());
134             } catch (ParseException e) {

```

(continues on next page)


(continued from previous page)

```

131         e.printStackTrace();
132     }
133     return null;
134 }
135 }
136
137 private class QuantityColumn2 extends AbstractSimpleExpression<BigDecimal> {
138     private static final long serialVersionUID = 1L;
139
140     @Override
141     public BigDecimal evaluate(ReportParameters reportParameters) {
142         try {
143             return type.bigDecimalType().stringValue("quantity",
144 ↪ reportParameters);
145         } catch (DRException e) {
146             e.printStackTrace();
147         }
148         return null;
149     }
150 }

```

1.19.9 Empty Column



DynamicReports

<http://www.dynamicreports.org>

EmptyColumn

Item	Quantity	Unit price	Order date
Notebook	1	500.00	01/22/2011

1 of 1

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
```

(continues on next page)

(continued from previous page)

```

6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class EmptyColumnReport {
39
40     public EmptyColumnReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             report()
47                 .setTemplate(Templates.reportTemplate)
48                 .columns(
49                     col.column("Item", "item",
50 ↪type.stringType()),
51                     col.emptyColumn(),
52 ↪setFixedWidth(30),
53                     col.column("Quantity",
54 ↪"quantity", type.integerType()),
55                     col.emptyColumn(false, true).
56 ↪setFixedWidth(20),
57                     col.column("Unit price",
58 ↪"unitprice", type.bigDecimalType()),
59                     col.emptyColumn(true, true).
60 ↪setFixedWidth(20),
61                     col.column("Order date",
62 ↪"orderdate", type.dateType()))

```

(continues on next page)

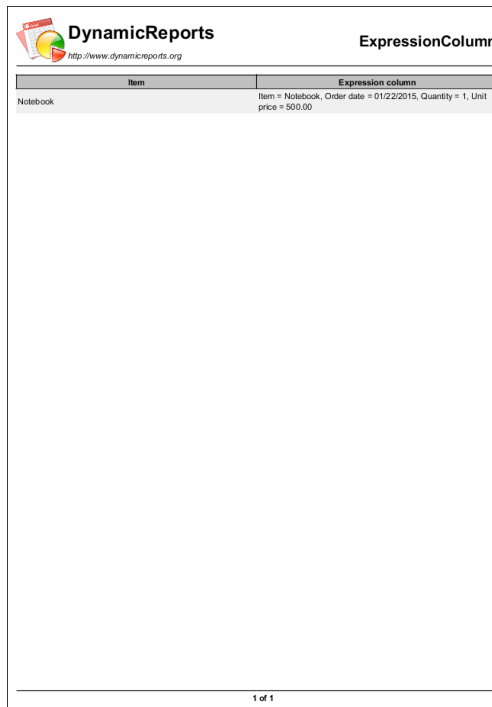
(continued from previous page)

```

56         .title(Templates.createTitleComponent(
    ↪ "EmptyColumn"))
57         .pageFooter(Templates.footerComponent)
58         .setDataSource(createDataSource())
59         .show();
60     } catch (DRException e) {
61         e.printStackTrace();
62     }
63 }
64
65 private JRDataSource createDataSource() {
66     DRDataSource dataSource = new DRDataSource("item", "orderdate",
    ↪ "quantity", "unitprice");
67     dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
68     return dataSource;
69 }
70
71 public static void main(String[] args) {
72     new EmptyColumnReport();
73 }
74 }

```

1.19.10 Expression Column



Item	Expression column
Notebook	Item = Notebook, Order date = 01/22/2015, Quantity = 1, Unit price = 500.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org

```

(continues on next page)

(continued from previous page)

```

6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.column;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
32 import net.sf.dynamicreports.report.builder.FieldBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.definition.ReportParameters;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class ExpressionColumnReport {
42     private FieldBuilder<Integer> quantityField;
43
44     public ExpressionColumnReport() {
45         build();
46     }
47
48     private void build() {
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .fields(
53                     field("orderdate", Date.
54 ↪ class),
55                     quantityField = field(
56 ↪ "quantity", Integer.class),
57                     field("unitprice", BigDecimal.
58 ↪ class))
59                 .columns(
60                     col.column("Item", "item",
61 ↪ type.stringType()),
62                     col.column("Expression column
63 ↪ ", new ExpressionColumn())

```

(continues on next page)


(continued from previous page)

```

59         .title(Templates.createTitleComponent(
↪ "ExpressionColumn"))
60         .pageFooter(Templates.footerComponent)
61         .setDataSource(createDataSource())
62         .show();
63     } catch (DRException e) {
64         e.printStackTrace();
65     }
66 }
67
68 private JRDataSource createDataSource() {
69     DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪ "quantity", "unitprice");
70     dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
71     return dataSource;
72 }
73
74 public static void main(String[] args) {
75     new ExpressionColumnReport();
76 }
77
78 private class ExpressionColumn extends AbstractSimpleExpression<String> {
79     private static final long serialVersionUID = 1L;
80
81     @Override
82     public String evaluate(ReportParameters reportParameters) {
83         return "Item = " + reportParameters.getValue("item") + ", " +
84             "Order date = " + type.dateType().
↪ valueToString("orderdate", reportParameters) + ", " +
85             "Quantity = " + type.integerType().
↪ valueToString(quantityField, reportParameters) + ", " +
86             "Unit price = " + type.bigDecimalType().
↪ valueToString("unitprice", reportParameters);
87     }
88 }
89 }

```

1.19.11 Percentage Columns

DynamicReports		PercentageColumns	
		http://www.dynamicreports.org	
Item	Quantity	Quantity [%]	Unit price [%]
Book	3	17.65%	24.44%
Book	1	5.88%	33.33%
Book	5	29.41%	22.22%
Book	8	47.06%	20.00%

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.column;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.FieldBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.column.PercentageColumnBuilder;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class PercentageColumnsReport {
41     private FieldBuilder<BigDecimal> unitPriceField;
42
43     public PercentageColumnsReport() {
44         build();
45     }
46
47     private void build() {
48         try {
49             unitPriceField = field("unitprice", BigDecimal.class);
50
51             TextColumnBuilder<String> itemColumn = col.column("Item",
↪ "item", type.stringType());
52             TextColumnBuilder<Integer> quantityColumn = col.column(
↪ "Quantity", "quantity", type.integerType());
53             PercentageColumnBuilder quantityPercColumn = col.
↪ percentageColumn("Quantity [%]", quantityColumn);
54             PercentageColumnBuilder unitPricePercColumn = col.
↪ percentageColumn("Unit price [%]", unitPriceField);
55
56             report()
57                 .setTemplate(Templates.reportTemplate)
58                 .fields(
59                     unitPriceField)
60                 .columns(
61                     itemColumn, quantityColumn,
↪ quantityPercColumn, unitPricePercColumn)
62                 .title(Templates.createTitleComponent(
↪ "PercentageColumns"))
63                 .pageFooter(Templates.footerComponent)
64                 .setDataSource(createDataSource())
65                 .show();
66         } catch (DRException e) {
67             e.printStackTrace();
68         }
69     }
70
71     private JRDataSource createDataSource() {
72         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
73         dataSource.add("Book", 3, new BigDecimal(11));
74         dataSource.add("Book", 1, new BigDecimal(15));
75         dataSource.add("Book", 5, new BigDecimal(10));
76         dataSource.add("Book", 8, new BigDecimal(9));
77         return dataSource;
78     }
79
80     public static void main(String[] args) {

```

(continues on next page)

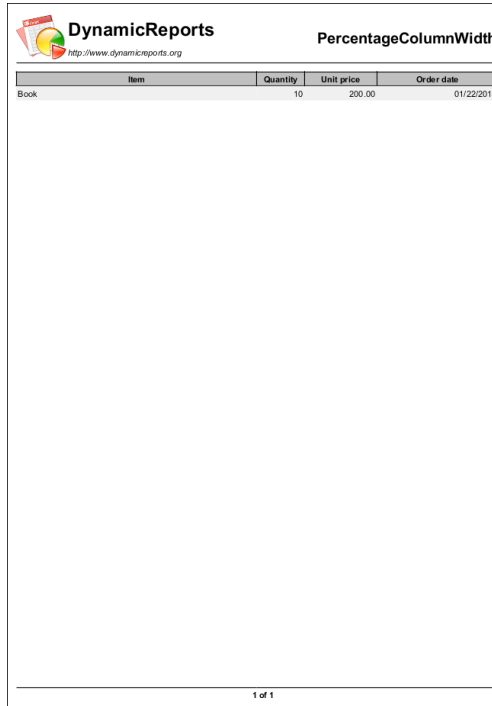
(continued from previous page)

```

81         new PercentageColumnsReport ();
82     }
83 }

```

1.19.12 Percentage Column Width



Item	Quantity	Unit price	Order date
Book	10	200.00	01/22/2015

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.column;

```

(continues on next page)

(continued from previous page)

```

24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class PercentageColumnWidthReport {
39
40     public PercentageColumnWidthReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             report()
47                 .setTemplate(Templates.reportTemplate)
48                 .columns(
49                     col.column("Item", "item",
50 ↪ type.stringType()).setWidth(50), // 50%
51                     col.column("Quantity",
52 ↪ "quantity", type.integerType()).setWidth(10), // 10%
53                     col.column("Unit price",
54 ↪ "unitprice", type.bigDecimalType()).setWidth(15), // 15%
55                     col.column("Order date",
56 ↪ "orderdate", type.dateType()).setWidth(25)) // 25%
57                 .title(Templates.createTitleComponent(
58 ↪ "PercentageColumnWidth"))
59                 .pageFooter(Templates.footerComponent)
60                 .setDataSource(createDataSource())
61                 .show();
62         } catch (DRException e) {
63             e.printStackTrace();
64         }
65     }
66
67     private JRDataSource createDataSource() {
68         DRDataSource dataSource = new DRDataSource("item", "orderdate",
69 ↪ "quantity", "unitprice");
70         dataSource.add("Book", new Date(), 10, new BigDecimal(200));
71         return dataSource;
72     }
73
74     public static void main(String[] args) {
75         new PercentageColumnWidthReport();
76     }
77 }

```

1.19.13 Row Number Columns

DynamicReports				RowNumberColumns				PageNumber				PageContent			
Report	Page	PageContent	PageContent	Report	Page	PageContent	PageContent	Report	Page	PageContent	PageContent	Report	Page	PageContent	PageContent
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10
11	11	11	11	11	11	11	11	11	11	11	11	11	11	11	11
12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
13	13	13	13	13	13	13	13	13	13	13	13	13	13	13	13
14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
15	15	15	15	15	15	15	15	15	15	15	15	15	15	15	15
16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
17	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
18	18	18	18	18	18	18	18	18	18	18	18	18	18	18	18
19	19	19	19	19	19	19	19	19	19	19	19	19	19	19	19
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21
22	22	22	22	22	22	22	22	22	22	22	22	22	22	22	22
23	23	23	23	23	23	23	23	23	23	23	23	23	23	23	23
24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
25	25	25	25	25	25	25	25	25	25	25	25	25	25	25	25
26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26
27	27	27	27	27	27	27	27	27	27	27	27	27	27	27	27
28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
29	29	29	29	29	29	29	29	29	29	29	29	29	29	29	29
30	30	30	30	30	30	30	30	30	30	30	30	30	30	30	30
31	31	31	31	31	31	31	31	31	31	31	31	31	31	31	31
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
33	33	33	33	33	33	33	33	33	33	33	33	33	33	33	33
34	34	34	34	34	34	34	34	34	34	34	34	34	34	34	34
35	35	35	35	35	35	35	35	35	35	35	35	35	35	35	35
36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
37	37	37	37	37	37	37	37	37	37	37	37	37	37	37	37
38	38	38	38	38	38	38	38	38	38	38	38	38	38	38	38
39	39	39	39	39	39	39	39	39	39	39	39	39	39	39	39
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
41	41	41	41	41	41	41	41	41	41	41	41	41	41	41	41
42	42	42	42	42	42	42	42	42	42	42	42	42	42	42	42

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */

```

```

22 package net.sf.dynamicreports.examples.column;
23
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.exception.DRException;
28 import net.sf.jasperreports.engine.JREmptyDataSource;
29
30 /**
31  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
32  */
33 public class RowNumberColumnsReport {
34
35     public RowNumberColumnsReport() {
36         build();
37     }
38
39     private void build() {
40         try {
41             report()
42                 .setTemplate(Templates.reportTemplate)


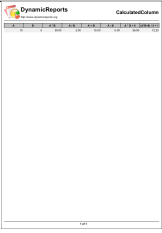
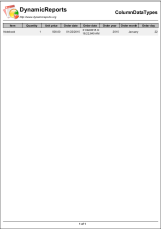
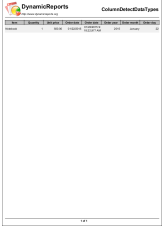



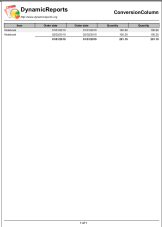
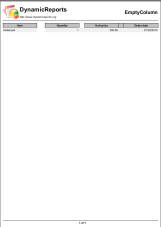

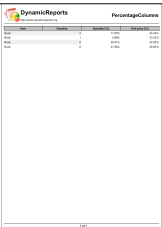

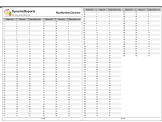
```

(continues on next page)

(continued from previous page)


```
43         .setPageColumnsPerPage(2)
44         .setPageColumnSpace(10)
45         .columns(
46             col.reportRowNumberColumn(
↪ "Report row"),
47             col.pageRowNumberColumn("Page_
↪ row"),
48             col.columnRowNumberColumn(
↪ "Page column row"))
49         .title(Templates.createTitleComponent(
↪ "RowNumberColumns"))
50         .pageFooter(Templates.footerComponent)
51         .setDataSource(new JREmptyDataSource(150))
52         .show();
53     } catch (DRException e) {
54         e.printStackTrace();
55     }
56 }
57
58 public static void main(String[] args) {
59     new RowNumberColumnsReport();
60 }
61 }
```

Table 4: Column Examples

		
BooleanColumnReport	CalculatedColumnReport	ColumnDataTypesReport
		
ColumnDetectDataTypeReport	ColumnListDataTypeReport	ColumnSubreportDataReport
		
ComponentColumnReport	ConversionColumnReport	EmptyColumnReport
		
ExpressionColumnReport	PercentageColumnsReport	PercentageColumnWidthReport
		
RowNumberColumnsReport		

1.20 Column Grid

1.20.1 Column Grid

DynamicReports		ColumnGrid	
 http://www.dynamicreports.org			
Item		Order date	
Quantity	Unit price	Order date Order month	Order year Order day
Notebook		01/22/2015 9:18:25:238 AM	2015
1	500.00	January	22
Book		01/22/2015 9:18:25:238 AM	2015
7	300.00	January	22
PDA		01/22/2015 9:18:25:238 AM	2015
2	250.00	January	22

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;

```

(continues on next page)

(continued from previous page)

```

28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ColumnGridReport {
41
42     public ColumnGridReport() {
43         build();
44     }
45
46     private void build() {
47         StyleBuilder textStyle = stl.style(Templates.columnStyle)
48             .setBorder(stl.pen1Point());
49
50         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
51 ↪ type.stringType());
52         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
53 ↪ "quantity", type.integerType());
54         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
55 ↪ ", "unitprice", type.bigDecimalType());
56         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
57 ↪ "orderdate", type.dateType());
58         TextColumnBuilder<Date> orderDateFColumn = col.column("Order date",
59 ↪ "orderdate", type.dateYearToFractionType());
60         TextColumnBuilder<Date> orderYearColumn = col.column("Order year",
61 ↪ "orderdate", type.dateYearType());
62         TextColumnBuilder<Date> orderMonthColumn = col.column("Order month",
63 ↪ "orderdate", type.dateMonthType());
64         TextColumnBuilder<Date> orderDayColumn = col.column("Order day",
65 ↪ "orderdate", type.dateDayType());
66
67         try {
68             report()
69                 .setTemplate(Templates.reportTemplate)
70                 .setColumnStyle(textStyle)
71                 .columns(
72                     itemColumn, quantityColumn,
73                     unitPriceColumn, orderDateColumn,
74                     orderDateFColumn,
75                     orderYearColumn, orderMonthColumn, orderDayColumn)
76                 .columnGrid(
77                     grid.verticalColumnGridList(
78                         itemColumn,
79                         grid.
80                     ↪ horizontalColumnGridList(quantityColumn, unitPriceColumn)),
81                     grid.verticalColumnGridList(
82                         orderDateColumn,
83                         grid.
84                     ↪ horizontalColumnGridList(orderDateFColumn, orderYearColumn)),

```

(continues on next page)


(continued from previous page)

```

73                                     grid.
↪horizontalColumnGridList (orderMonthColumn, orderDayColumn))
74                                     .title(Templates.createTitleComponent (
↪"ColumnGrid"))
75                                     .pageFooter(Templates.footerComponent)
76                                     .setDataSource(createDataSource())
77                                     .show();
78     } catch (DRException e) {
79         e.printStackTrace();
80     }
81 }
82
83 private JRDataSource createDataSource() {
84     DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
85     dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
86     dataSource.add("Book", new Date(), 7, new BigDecimal(300));
87     dataSource.add("PDA", new Date(), 2, new BigDecimal(250));
88     return dataSource;
89 }
90
91 public static void main(String[] args) {
92     new ColumnGridReport();
93 }
94 }

```

1.20.2 Column Title Group

DynamicReports		ColumnTitleGroup	
 http://www.dynamicreports.org			
Item	Order date	Group 1	
		Quantity	Unit price
Notebook	01/22/2015	1	500.00

1 of 1

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.grid.ColumnTitleGroupBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ColumnTitleGroupReport {
41
42     public ColumnTitleGroupReport() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
48 ↪ type.stringType());
49         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
50 ↪ "orderdate", type.dateType());
51         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
52 ↪ "quantity", type.integerType()).setFixedWidth(50);
53         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price",
54 ↪ "unitprice", type.bigDecimalType());
55
56         ColumnTitleGroupBuilder titleGroup2 = grid.titleGroup("Group 2",
57 ↪ quantityColumn, unitPriceColumn);
```

(continues on next page)


(continued from previous page)

```

53      ColumnTitleGroupBuilder titleGroup1 = grid.titleGroup("Group 1",
↳orderDateColumn, titleGroup2).setTitleFixedWidth(450);
54
55      try {
56          report()
57              .setTemplate(Templates.reportTemplate)
58              .columnGrid(
59                  itemColumn, titleGroup1)
60              .columns(
61                  itemColumn, orderDateColumn,
↳quantityColumn, unitPriceColumn)
62              .title(Templates.createTitleComponent(
↳"ColumnTitleGroup"))
63              .pageFooter(Templates.footerComponent)
64              .setDataSource(createDataSource())
65              .show();
66      } catch (DRException e) {
67          e.printStackTrace();
68      }
69  }
70
71  private JRDataSource createDataSource() {
72      DRDataSource dataSource = new DRDataSource("item", "orderdate",
↳"quantity", "unitprice");
73      dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
74      return dataSource;
75  }
76
77  public static void main(String[] args) {
78      new ColumnTitleGroupReport();
79  }
80  }

```

1.20.3 Flow Column Pairs



DynamicReports

<http://www.dynamicreports.org>

FlowColumnPairs

Id	Item	Quantity	Unit price
5	Notebook	1	500.0
Order date	Order year	Order month	Order day
01/22/2015	2015	January	22

Id	Item	Quantity	Unit price
8	Book	7	300.0
Order date	Order year	Order month	Order day
01/22/2015	2015	January	22

Id	Item	Quantity	Unit price
15	PDIA	2	250.0
Order date	Order year	Order month	Order day
01/22/2015	2015	January	22

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.FieldBuilder;
32 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
33 import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
34 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
35 import net.sf.dynamicreports.report.constant.ListType;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class FlowColumnPairsReport {
44
45     public FlowColumnPairsReport() {
46         build();
47     }
48
49     private void build() {
50         StyleBuilder textStyle = stl.style(Templates.columnStyle)
51             .setBorder(stl.pen1Point());
52
53         FieldBuilder<Integer> idField = field("id", type.integerType());
54         FieldBuilder<String> itemField = field("item", type.stringType());
55         FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
56         FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());
57         FieldBuilder<Date> orderDateField = field("orderdate", type.
↪dateType());
58         FieldBuilder<Date> orderYearField = field("orderdate", type.
↪dateYearType());
59         FieldBuilder<Date> orderMonthField = field("orderdate", type.
↪dateMonthType());
60         FieldBuilder<Date> orderDayField = field("orderdate", type.
↪dateDayType());
61
62         try {
63             report()
64                 .setTemplate(Templates.reportTemplate)
65                 .setColumnStyle(textStyle)
66                 .columnGrid(ListType.HORIZONTAL_FLOW)
67                 .fields(idField, itemField, quantityField,
↪unitPriceField, orderDateField, orderYearField, orderMonthField, orderDayField)
68                 .columns(
69                     col.
↪componentColumn(columnPair("Id", idField)),
70                     col.
↪componentColumn(columnPair("Item", itemField).setFixedWidth(200)),
71                     col.
↪componentColumn(columnPair("Quantity", quantityField)),
72                     col.
↪componentColumn(columnPair("Unit price", unitPriceField)),
73                     col.
↪componentColumn(columnPair("Order date", orderDateField)),
74                     col.
↪componentColumn(columnPair("Order year", orderYearField)),

```

(continues on next page)


(continued from previous page)

```

75         col.
↪componentColumn(columnPair("Order month", orderMonthField)),
76         col.
↪componentColumn(columnPair("Order day", orderDayField)))
77         .title(Templates.createTitleComponent(
↪"FlowColumnPairs"))
78         .detailFooter(cmp.verticalGap(20))
79         .pageFooter(Templates.footerComponent)
80         .setDataSource(createDataSource())
81         .show();
82     } catch (DRException e) {
83         e.printStackTrace();
84     }
85 }
86
87 private VerticalListBuilder columnPair(String title, FieldBuilder<?> value) {
88     TextFieldBuilder<String> titleCmp = cmp.text(title)
89         .setStyle(Templates.columnTitleStyle);
90     TextFieldBuilder<?> valueCmp = cmp.text(value);
91     return cmp.verticalList(titleCmp, valueCmp);
92 }
93
94 private JRDataSource createDataSource() {
95     DRDataSource dataSource = new DRDataSource("id", "item", "orderdate",
↪"quantity", "unitprice");
96     dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));
97     dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
98     dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
99     return dataSource;
100 }
101
102 public static void main(String[] args) {
103     new FlowColumnPairsReport();
104 }
105 }

```

1.20.4 Many Columns

DynamicReports					ManyColumns
 http://www.dynamicreports.org					
ID	Item	Quantity	Unit price	Order date	
Order date	Order year	Order month	Order day		
5	Notebook	1	500.00	01/22/2015	22
01/22/2015 9:18:25,714 AM	2015	January			
8	Book	7	300.00	01/22/2015	22
01/22/2015 9:18:25,714 AM	2015	January			
15	PDA	2	250.00	01/22/2015	22
01/22/2015 9:18:25,714 AM	2015	January			

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
32 import net.sf.dynamicreports.report.constant.ListType;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ManyColumnsReport {
41
42     public ManyColumnsReport() {
43         build();
44     }
45
46     private void build() {
47         StyleBuilder textStyle = stl.style(Templates.columnStyle)
48             .setBorder(stl.pen1Point());
49
50         try {
51             report()
52                 .setTemplate(Templates.reportTemplate)
53                 .setColumnStyle(textStyle)
54                 .columnGrid(ListType.HORIZONTAL_FLOW)
55                 .columns(
56                     col.column("ID", "id", type.
57 ↪integerType()),
58                     col.column("Item", "item",
59 ↪type.stringType()),
60                     col.column("Quantity",
61 ↪"quantity", type.integerType()),
62                     col.column("Unit price",
63 ↪"unitprice", type.bigDecimalType()),
64                     col.column("Order date",
65 ↪"orderdate", type.dateType()),
66                     col.column("Order date",
67 ↪"orderdate", type.dateYearToFractionType()),
68                     col.column("Order year",
69 ↪"orderdate", type.dateYearType()),
70                     col.column("Order month",
71 ↪"orderdate", type.dateMonthType()),
72                     col.column("Order day",
73 ↪"orderdate", type.dateDayType()))
74                 .title(Templates.createTitleComponent(
75 ↪"ManyColumns"))
76                 .pageFooter(Templates.footerComponent)
77                 .setDataSource(createDataSource())
78                 .show();
79         } catch (DRException e) {
80             e.printStackTrace();
81         }
82     }
83
84     private JRDataSource createDataSource() {
85         DRDataSource dataSource = new DRDataSource("id", "item", "orderdate",
86 ↪"quantity", "unitprice");
87         dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));

```

(continues on next page)


(continued from previous page)

```

77         dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
78         dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
79         return dataSource;
80     }
81
82     public static void main(String[] args) {
83         new ManyColumnsReport();
84     }
85 }

```

1.20.5 Vertical Columns



DynamicReports

<http://www.dynamicreports.org>

VerticalColumns

Item
Quantity
Unit price
Notebook
Book
PDA

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.

```

(continues on next page)

(continued from previous page)

```

18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
31 import net.sf.dynamicreports.report.constant.ListType;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class VerticalColumnsReport {
40
41     public VerticalColumnsReport() {
42         build();
43     }
44
45     private void build() {
46         StyleBuilder textStyle = stl.style(Templates.columnStyle)
47             .setBorder(stl.pen1Point());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .setColumnStyle(textStyle)
53                 .columnGrid(ListType.VERTICAL)
54                 .columns(
55                     col.column("Item", "item",
56 ↪ type.stringType()),
57                     col.column("Quantity",
58 ↪ "quantity", type.integerType()),
59                     col.column("Unit price",
60 ↪ "unitprice", type.bigDecimalType()))
61                 .title(Templates.createTitleComponent(
62 ↪ "VerticalColumns"))
63                 .pageFooter(Templates.footerComponent)
64                 .setDataSource(createDataSource())
65                 .show();
66         } catch (DRException e) {
67             e.printStackTrace();
68         }
69     }
70
71     private JRDataSource createDataSource() {
72         DRDataSource dataSource = new DRDataSource("item", "quantity",
73 ↪ "unitprice");
74         dataSource.add("Notebook", 1, new BigDecimal(500));
75     }
76 }

```

(continues on next page)


(continued from previous page)

```

70         dataSource.add("Book", 7, new BigDecimal(300));
71         dataSource.add("PDA", 2, new BigDecimal(250));
72         return dataSource;
73     }
74
75     public static void main(String[] args) {
76         new VerticalColumnsReport();
77     }
78 }

```

1.20.6 Vertical Values



DynamicReports

<http://www.dynamicreports.org>

VerticalValues

ues

Name	Value
Item:	Notebook
Quantity:	1
Unit price:	500.00
Order date:	01/22/2015
Item:	Book
Quantity:	4
Unit price:	25.00
Order date:	01/22/2015
Item:	PDA
Quantity:	2
Unit price:	120.00
Order date:	01/22/2015
Unit price sum =	
645.00	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.

```

(continues on next page)

(continued from previous page)

```

18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.columngrid;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.FieldBuilder;
32 import net.sf.dynamicreports.report.builder.column.ComponentColumnBuilder;
33 import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
34 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
35 import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
36 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
37 import net.sf.dynamicreports.report.constant.PageType;
38 import net.sf.dynamicreports.report.datasource.DRDataSource;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class VerticalValuesReport {
46
47     public VerticalValuesReport() {
48         build();
49     }
50
51     private void build() {
52         StyleBuilder nameStyle = stl.style().bold();
53         StyleBuilder valueStyle = stl.style().
54 ↪setHorizontalTextAlignment(HorizontalTextAlignment.LEFT);
55
56         FieldBuilder<String> itemField = field("item", type.stringType());
57         FieldBuilder<Integer> quantityField = field("quantity", type.
58 ↪integerType());
59         FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
60 ↪bigDecimalType());
61         FieldBuilder<Date> orderDateField = field("orderdate", type.
62 ↪dateType());
63
64         VerticalListBuilder nameList = cmp.verticalList(
65             cmp.text("Item:").setStyle(nameStyle),
66             cmp.text("Quantity:").setStyle(nameStyle),
67             cmp.text("Unit price:").setStyle(nameStyle),
68             cmp.text("Order date:").setStyle(nameStyle));
69         VerticalListBuilder valueList = cmp.verticalList(
70             cmp.text(itemField).setStyle(valueStyle),
71             cmp.text(quantityField).setStyle(valueStyle),
72             cmp.text(unitPriceField).setStyle(valueStyle),
73             cmp.text(orderDateField).setStyle(valueStyle));

```

(continues on next page)

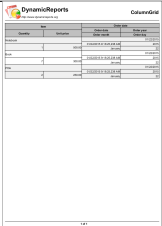

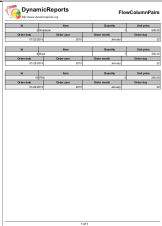
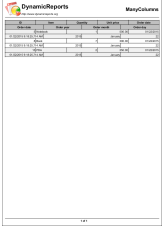
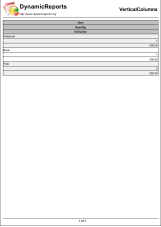
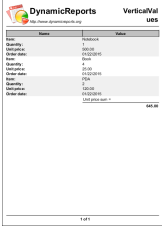
(continued from previous page)

```

71         ComponentColumnBuilder nameColumn = col.componentColumn("Name", ↵
↵nameList);
72         ComponentColumnBuilder valueColumn = col.componentColumn("Value", ↵
↵valueList);
73
74         AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.
↵sum(unitPriceField, valueColumn)
75             .setLabel("Unit price sum =");
76
77         try {
78             report()
79                 .setTemplate(Templates.reportTemplate)
80                 .setPageFormat(PageType.A5)
81                 .fields(itemField, quantityField, ↵
↵unitPriceField, orderDateField)
82                 .columns(nameColumn, valueColumn)
83                 .subtotalsAtSummary(unitPriceSum)
84                 .title(Templates.createTitleComponent(
↵"VerticalValues"))
85                 .pageFooter(Templates.footerComponent)
86                 .setDataSource(createDataSource())
87                 .show();
88         } catch (DRException e) {
89             e.printStackTrace();
90         }
91     }
92
93     private JRDataSource createDataSource() {
94         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↵"quantity", "unitprice");
95         dataSource.add("Notebook", new Date(), 1, new BigDecimal(500));
96         dataSource.add("Book", new Date(), 4, new BigDecimal(25));
97         dataSource.add("PDA", new Date(), 2, new BigDecimal(120));
98         return dataSource;
99     }
100
101     public static void main(String[] args) {
102         new VerticalValuesReport();
103     }
104 }

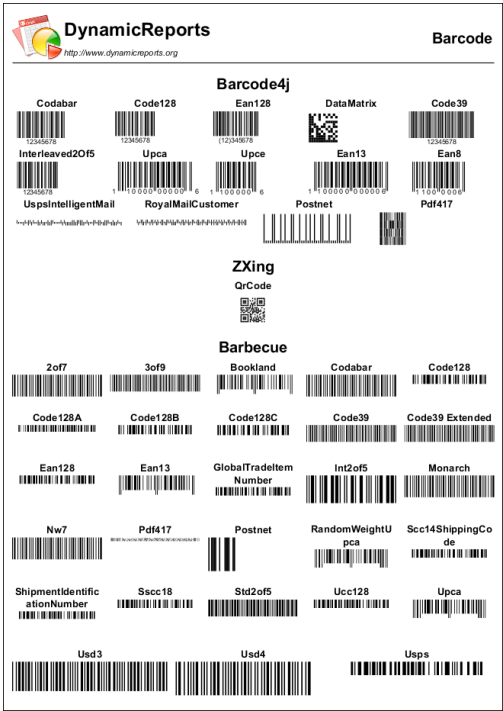
```

Table 5: Chart Examples

		
ColumnGridReport	ColumnTitleGroupReport	FlowColumnPairsReport
		
ManyColumnsReport	VerticalColumnsReport	VerticalValuesReport

1.21 Component

1.21.1 Barcode



```
1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
```

(continues on next page)

(continued from previous page)

```

5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.component;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
28 import net.sf.dynamicreports.report.builder.component.DimensionComponentBuilder;
29 import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
30 import net.sf.dynamicreports.report.exception.DRException;
31
32 /**
33  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
34  */
35 public class BarcodeReport {
36
37     public BarcodeReport() {
38         build();
39     }
40
41     private void build() {
42         try {
43             report()
44                 .setTemplate(template().setBarcodeHeight(40))
45                 .title(
46                     Templates.
47                         ↳createTitleComponent("Barcode"),
48                         cmp.text("Barcode4j").
49                         ↳setStyle(Templates.bold18CenteredStyle),
50                         barcode4j(),
51                         cmp.verticalGap(10),
52                         cmp.text("ZXing").
53                         ↳setStyle(Templates.bold18CenteredStyle),
54                         barcode("QRCode", bcode.
55                             cmp.verticalGap(10),
56                             cmp.text("Barcode").
57                             ↳setStyle(Templates.bold18CenteredStyle),
58                             barbecue())
59                 .show();
60         } catch (DRException e) {

```

(continues on next page)

(continued from previous page)

```

57         e.printStackTrace();
58     }
59 }
60
61 private ComponentBuilder<?, ?> barcode4j() {
62     HorizontalListBuilder list = cmp.horizontalFlowList();
63     list.setGap(10);
64     list.add(
65         barcode("Codabar", bcode.codabar("12345678")),
66         barcode("Code128", bcode.code128("12345678")),
67         barcode("Ean128", bcode.ean128("12345678")),
68         barcode("DataMatrix", bcode.dataMatrix("12345678")),
69         barcode("Code39", bcode.code39("12345678")),
70         barcode("Interleaved2Of5", bcode.interleaved2Of5(
71             ↪ "12345678")),
72         barcode("Upca", bcode.upca("110000000000")),
73         barcode("Upce", bcode.upce("1100000")),
74         barcode("Ean13", bcode.ean13("1100000000000")),
75         barcode("Ean8", bcode.ean8("1100000")),
76         barcode("UspsIntelligentMail", bcode.
77             ↪ uspsIntelligentMail("34160265194042788110")),
78         barcode("RoyalMailCustomer", bcode.royalMailCustomer(
79             ↪ "34160265194042788110")),
80         barcode("Postnet", bcode.postnet("12345678")),
81         barcode("Pdf417", bcode.pdf417("12345678")));
82     return list;
83 }
84
85 private ComponentBuilder<?, ?> barbecue() {
86     HorizontalListBuilder list = cmp.horizontalFlowList();
87     list.setGap(10);
88     list.add(
89         barcode("2of7", bcode.barbecue_2of7("12345678")),
90         barcode("3of9", bcode.barbecue_3of9("12345678")),
91         barcode("Bookland", bcode.barbecue_bookland(
92             ↪ "1234567890")),
93         barcode("Codabar", bcode.barbecue_codabar("12345678
94             ↪ "))),
95         barcode("Code128", bcode.barbecue_code128("12345678
96             ↪ "))),
97         barcode("Code128A", bcode.barbecue_code128A("12345678
98             ↪ "))),
99         barcode("Code128B", bcode.barbecue_code128B("12345678
100            ↪ "))),
101         barcode("Code128C", bcode.barbecue_code128C("12345678
102            ↪ "))),
103         barcode("Code39", bcode.barbecue_code39("12345678")),
104         barcode("Code39 Extended", bcode.barbecue_
105             ↪ code39Extended("12345678")),
106         barcode("Ean128", bcode.barbecue_ean128("12345678")),
107         barcode("Ean13", bcode.barbecue_ean13("123456789012
108             ↪ "))),
109         barcode("GlobalTradeItemNumber", bcode.barbecue_
110             ↪ globalTradeItemNumber("12345678")),
111         barcode("Int2of5", bcode.barbecue_int2of5("12345678
112             ↪ "))),
113         barcode("Monarch", bcode.barbecue_monarch("12345678
114             ↪ "))),

```

(continues on next page)

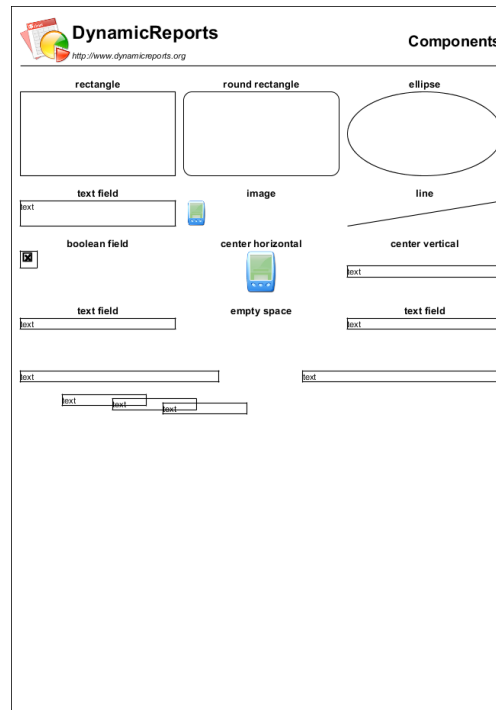
(continued from previous page)

```

101         barcode("Nw7", bcode.barbecue_nw7("12345678")),
102         barcode("Pdf417", bcode.barbecue_pdf417("12345678")),
103         barcode("Postnet", bcode.barbecue_postnet("12345678
↳")),
104         barcode("RandomWeightUpca", bcode.barbecue_
↳randomWeightUpca("12345678901")),
105         barcode("Scc14ShippingCode", bcode.barbecue_
↳scc14ShippingCode("12345678")),
106         barcode("ShipmentIdentificationNumber", bcode.
↳barbecue_shipmentIdentificationNumber("12345678")),
107         barcode("Scc18", bcode.barbecue_scc18("12345678")),
108         barcode("Std2of5", bcode.barbecue_std2of5("12345678
↳")),
109         barcode("Ucc128", bcode.barbecue_ucc128("12345678").
↳setApplicationIdentifierExpression("123")),
110         barcode("Upca", bcode.barbecue_upca("12345678901")),
111         barcode("Usd3", bcode.barbecue_usd3("12345678")),
112         barcode("Usd4", bcode.barbecue_usd4("12345678")),
113         barcode("Usps", bcode.barbecue_usps("12345678")));
114     return list;
115 }
116
117 private ComponentBuilder<?, ?> barcode(String label, DimensionComponentBuilder
↳<?, ?> barcode) {
118     return cmp.verticalList(cmp.text(label).setStyle(Templates.
↳bold12CenteredStyle), barcode);
119 }
120
121 public static void main(String[] args) {
122     new BarcodeReport();
123 }
124 }

```

1.21.2 Components



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.component;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.net.URL;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.component.ComponentBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
32 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
33 import net.sf.dynamicreports.report.constant.BooleanComponentType;
34 import net.sf.dynamicreports.report.exception.DRException;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class ComponentsReport {
40
41     public ComponentsReport() {
42         build();
43     }
44
45     private void build() {
46         URL image = Templates.class.getResource("images/pda.png");
47
48         try {
49             report()
50                 .setTemplate(template().setBarcodeHeight(50))
51                 .setTextStyle(stl.style(stl.pen1Point()))
52                 .title(
53                     Templates.
54                     createTitleComponent("Components"),
55                     components(
56                         cmp.rectangle(),
57                         cmp.rectangle(),
58                         cmp.ellipse(),
59                         cmp.verticalGap(10),
60                         cmp.text("text"),
61                         cmp.image(image).setFixedDimension(30, 30),
62                         cmp.line(),
63                         cmp.verticalGap(10),
64                         cmp.components(
65                             cmp.booleanField(true).setComponentType(BooleanComponentType.IMAGE_CHECKBOX_2).
66                             setFixedDimension(20, 20),
67                             cmp.horizontal(),
68                             cmp.vertical(),
69                             cmp.text("text"),
70                             cmp.text("text"),
71                             cmp.filler(),
72                             cmp.text("text")),
73                     cmp.verticalGap(50),

```

(continues on next page)

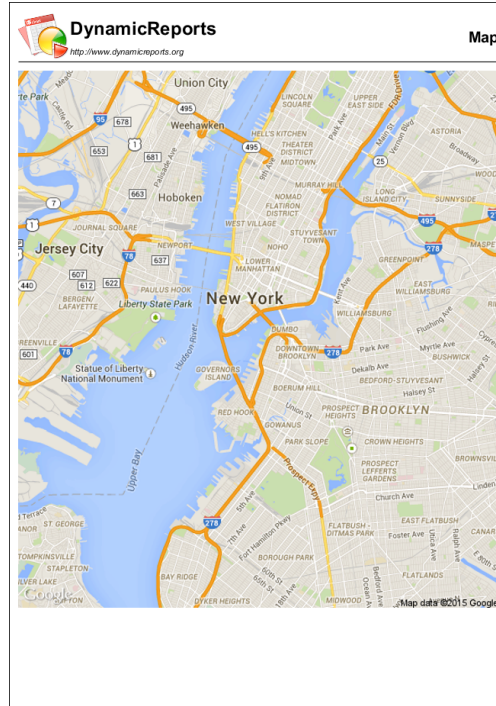
(continued from previous page)

```

74         cmp.horizontalList(cmp.text(
↪ "text"), cmp.horizontalGap(100), cmp.text("text")),
75         cmp.xyList()
76             .add(50, 15, ↵
↪ cmp.text("text"))
77             .add(110, 20, ↵
↪ cmp.text("text"))
78             .add(170, 25, ↵
↪ cmp.text("text")))
79         .show();
80     } catch (DRException e) {
81         e.printStackTrace();
82     }
83 }
84
85 private ComponentBuilder<?, ?> components(String label1, ComponentBuilder<?, ?
↪ > component1, String label2, ComponentBuilder<?, ?> component2, String label3,
86     ComponentBuilder<?, ?> component3) {
87     HorizontalListBuilder list = cmp.horizontalList()
88         .setGap(10);
89     list.add(component(label1, component1));
90     list.add(component(label2, component2));
91     list.add(component(label3, component3));
92     return list;
93 }
94
95 private ComponentBuilder<?, ?> component(String label, ComponentBuilder<?, ?> ↵
↪ component) {
96     TextFieldBuilder<String> labelField = cmp.text(label)
97         .setFixedRows(1)
98         .setStyle(Templates.bold12CenteredStyle);
99     return cmp.verticalList(labelField, component);
100 }
101
102 public static void main(String[] args) {
103     new ComponentsReport();
104 }
105 }

```

1.21.3 Map



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.component;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
28  import net.sf.dynamicreports.report.exception.DRException;
29
30  /**

```

(continues on next page)

(continued from previous page)

```

31  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
32  */
33  public class MapReport {
34
35      public MapReport() {
36          build();
37      }
38
39      private void build() {
40          try {
41              JasperHtmlExporterBuilder htmlExporter = export.htmlExporter(
42                  ↪ "c:/report.html")
43                      .setImagesDirName("c:/images")
44                      .setOutputImagesToDir(true);
45
46              report()
47                  .setTemplate(template())
48                  .title(
49                      ↪ createTitleComponent("Map"),
50                      ↪ cmp.map(40.7f, -74f, 12).
51                      ↪ setFixedHeight(750))
52                      .toHtml(htmlExporter);
53          } catch (DRException e) {
54              e.printStackTrace();
55          }
56
57      public static void main(String[] args) {
58          new MapReport();
59      }

```

1.21.4 Multi Page List

DynamicReports Free Java reporting library for creating reports dynamically	
MultiPageList	
Title component 1	Title component 1
Title component 2	Title component 2
Title component 3	Title component 3
Title component 4	
Title component 5	
Title component 6	
Title component 7	
Page 1	

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *

```

(continues on next page)

(continued from previous page)

```

7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.component;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.component.MultiPageListBuilder;
28 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
29 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
30 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
31 import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
32 import net.sf.dynamicreports.report.exception.DRException;
33
34 /**
35  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36  */
37 public class MultiPageListReport {
38
39     public MultiPageListReport() {
40         build();
41     }
42
43     private void build() {
44         StyleBuilder style = stl.style(stl.pen1Point())
45             .setTextAlignment(HorizontalTextAlignment.CENTER,
46 ↪ VerticalTextAlignment.MIDDLE);
47
48         MultiPageListBuilder multiPageList = cmp.multiPageList();
49         for (int i = 0; i < 10; i++) {
50             TextFieldBuilder<String> textField = cmp.text("Title_
51 ↪ component " + (i + 1))
52                 .setFixedHeight(100)
53                 .setStyle(style);
54             multiPageList.add(textField);
55         }
56
57         try {
58             report()
59                 .setTemplate(Templates.reportTemplate)
60                 .title(Templates.createTitleComponent(
61 ↪ "MultiPageList"))
62                 .summary(multiPageList)
63                 .pageFooter(Templates.footerComponent)





```

(continues on next page)

(continued from previous page)


```
61         .show();
62     } catch (DREException e) {
63         e.printStackTrace();
64     }
65 }
66
67 public static void main(String[] args) {
68     new MultiPageListReport();
69 }
70 }
```

Table 6: Component Examples

 BarcodeReport	 ComponentsReport	 MapReport
 MultiPageListReport		

1.22 Crosstab

1.22.1 Crosstab

**DynamicReports**
<http://www.dynamicreports.org>

Crosstab

State / Item	Book		DVD		Notebook		Total	
	Quantity	Unit price	Quantity	Unit price	Quantity	Unit price	Quantity	Unit price
Florida	2	51.00	8	128.00	1	480.00	31	659.00
New York	10	54.00	18	152.00	1	500.00	33	686.00
Washington	21	44.00	10	163.00	1	610.00	37	817.00
Total for sales	33	529.00	46	443.00	3	1,590.00	96	2,142.00

1 of 1

```
1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.crosstab;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
31 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
32 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
33 import net.sf.dynamicreports.report.constant.Calculation;
34 import net.sf.dynamicreports.report.constant.PageOrientation;
35 import net.sf.dynamicreports.report.constant.PageType;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class CrosstabReport {
44
45     public CrosstabReport() {
46         build();
47     }
48
49     private void build() {
50         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state", ↵
51 ↵String.class)
52             .setTotalHeader("Total for state");
53
54         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
55 ↵"item", String.class);
56
57         CrosstabBuilder crosstab = ctab.crosstab()
58             .headerCell(cmp.text("State / Item"))
59             .setStyle(Templates.boldCenteredStyle())
60             .rowGroups(rowGroup)
61             .columnGroups(columnGroup)
62             .measures(
63                 ctab.measure("Quantity", "quantity", ↵
64 ↵Integer.class, Calculation.SUM),

```

(continues on next page)


(continued from previous page)

```

61                                     ctab.measure("Unit price", "unitprice
↪", BigDecimal.class, Calculation.SUM));
62
63         try {
64             report()
65                                     .setPageFormat(PageType.A4, PageOrientation.
↪LANDSCAPE)
66                                     .setTemplate(Templates.reportTemplate)
67                                     .title(Templates.createTitleComponent(
↪"Crosstab"))
68                                     .summary(crosstab)
69                                     .pageFooter(Templates.footerComponent)
70                                     .setDataSource(createDataSource())
71                                     .show();
72         } catch (DRException e) {
73             e.printStackTrace();
74         }
75     }
76
77     private JRDataSource createDataSource() {
78         DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪", "unitprice");
79         dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
80         dataSource.add("New York", "DVD", 5, new BigDecimal(30));
81         dataSource.add("New York", "DVD", 2, new BigDecimal(45));
82         dataSource.add("New York", "DVD", 4, new BigDecimal(36));
83         dataSource.add("New York", "DVD", 5, new BigDecimal(41));
84         dataSource.add("New York", "Book", 2, new BigDecimal(11));
85         dataSource.add("New York", "Book", 8, new BigDecimal(9));
86         dataSource.add("New York", "Book", 6, new BigDecimal(14));
87
88         dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
89         dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
90         dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
91         dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
92         dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
93         dataSource.add("Washington", "Book", 3, new BigDecimal(12));
94         dataSource.add("Washington", "Book", 9, new BigDecimal(8));
95         dataSource.add("Washington", "Book", 4, new BigDecimal(14));
96         dataSource.add("Washington", "Book", 5, new BigDecimal(10));
97
98         dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
99         dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
100        dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
101        dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
102        dataSource.add("Florida", "Book", 4, new BigDecimal(11));
103        dataSource.add("Florida", "Book", 8, new BigDecimal(6));
104        dataSource.add("Florida", "Book", 6, new BigDecimal(16));
105        dataSource.add("Florida", "Book", 3, new BigDecimal(18));
106        return dataSource;
107    }
108
109    public static void main(String[] args) {
110        new CrosstabReport();
111    }
112 }

```

1.22.2 Custom Percentage Crosstab



DynamicReports
http://www.dynamicreports.org

CustomPercentageCrosstab

State / Item	Book		DVD		Notebook		Total	
	Unit price	%	Unit price	%	Unit price	%	Unit price	%
Florida	51.10	8	128.00	20	480.70	72	659.80	100
New York	34.80	5.1	100.00	16.2	500.00	72.2	634.80	100
Washington	44.00	6.8	100.40	16	610.00	74.8	814.40	100
Total for state	129.90	6.1	448.40	72.2	1,590.70	73.8	2,168.00	100

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.crosstab;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.List;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.FieldBuilder;
32 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
33 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
34 import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
35 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
36 import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
37 import net.sf.dynamicreports.report.constant.Calculation;
38 import net.sf.dynamicreports.report.constant.PageOrientation;
39 import net.sf.dynamicreports.report.constant.PageType;
40 import net.sf.dynamicreports.report.datasource.DRDataSource;
41 import net.sf.dynamicreports.report.definition.ReportParameters;
42 import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

(continued from previous page)

```

43 import net.sf.jasperreports.engine.JRDataSource;
44
45 /**
46  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
47  */
48 public class CustomPercentageCrosstabReport {
49
50     public CustomPercentageCrosstabReport() {
51         build();
52     }
53
54     private void build() {
55         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",
↳String.class)
56             .setTotalHeader("Total for state");
57
58         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↳"item", String.class);
59
60         FieldBuilder<BigDecimal> quantityField = field("unitprice",
↳BigDecimal.class);
61
62         CrosstabMeasureBuilder<BigDecimal> unitPriceMeasure = ctab.measure(
↳"Unit price", quantityField, Calculation.SUM);
63         CrosstabMeasureBuilder<BigDecimal> percentageMeasure = ctab.measure("%
↳", new PercentageExpression(unitPriceMeasure, columnGroup));
64         percentageMeasure.setDataType(type.doubleType());
65         CrosstabBuilder crosstab = ctab.crosstab()
66             .headerCell(cmp.text("State / Item"))
↳setStyle(Templates.boldCenteredStyle))
67             .rowGroups(rowGroup)
68             .columnGroups(columnGroup)
69             .measures(unitPriceMeasure, percentageMeasure);
70
71         try {
72             report()
73                 .setPageFormat(PageType.A4, PageOrientation.
↳LANDSCAPE)
74                 .setTemplate(Templates.reportTemplate)
75                 .title(Templates.createTitleComponent(
↳"CustomPercentageCrosstab"))
76                 .summary(crosstab)
77                 .pageFooter(Templates.footerComponent)
78                 .setDataSource(createDataSource())
79                 .show();
80         } catch (DRException e) {
81             e.printStackTrace();
82         }
83     }
84
85     private JRDataSource createDataSource() {
86         DRDataSource dataSource = new DRDataSource("state", "item", "unitprice
↳");
87         dataSource.add("New York", "Notebook", new BigDecimal(500));
88         dataSource.add("New York", "DVD", new BigDecimal(30));
89         dataSource.add("New York", "DVD", new BigDecimal(45.6));
90         dataSource.add("New York", "DVD", new BigDecimal(36));

```

(continues on next page)


(continued from previous page)

```

91     dataSource.add("New York", "DVD", new BigDecimal(41));
92     dataSource.add("New York", "Book", new BigDecimal(11));
93     dataSource.add("New York", "Book", new BigDecimal(9));
94     dataSource.add("New York", "Book", new BigDecimal(14.8));
95
96     dataSource.add("Washington", "Notebook", new BigDecimal(610));
97     dataSource.add("Washington", "DVD", new BigDecimal(40));
98     dataSource.add("Washington", "DVD", new BigDecimal(35));
99     dataSource.add("Washington", "DVD", new BigDecimal(46.4));
100    dataSource.add("Washington", "DVD", new BigDecimal(42));
101    dataSource.add("Washington", "Book", new BigDecimal(12));
102    dataSource.add("Washington", "Book", new BigDecimal(8));
103    dataSource.add("Washington", "Book", new BigDecimal(14));
104    dataSource.add("Washington", "Book", new BigDecimal(10));
105
106    dataSource.add("Florida", "Notebook", new BigDecimal(460.7));
107    dataSource.add("Florida", "DVD", new BigDecimal(49));
108    dataSource.add("Florida", "DVD", new BigDecimal(32));
109    dataSource.add("Florida", "DVD", new BigDecimal(47));
110    dataSource.add("Florida", "Book", new BigDecimal(11));
111    dataSource.add("Florida", "Book", new BigDecimal(6.1));
112    dataSource.add("Florida", "Book", new BigDecimal(16));
113    dataSource.add("Florida", "Book", new BigDecimal(18));
114    return dataSource;
115 }
116
117 public static void main(String[] args) {
118     new CustomPercentageCrosstabReport();
119 }
120
121 private class PercentageExpression extends AbstractComplexExpression
↪<BigDecimal> {
122     private static final long serialVersionUID = 1L;
123
124     private PercentageExpression(CrosstabMeasureBuilder<BigDecimal> ↪
↪unitPriceMeasure, CrosstabColumnGroupBuilder<String> columnGroup) {
125         addExpression(unitPriceMeasure);
126         addExpression(exp.crosstabValue(unitPriceMeasure, ↪
↪columnGroup));
127     }
128
129     @Override
130     public BigDecimal evaluate(List<?> values, ReportParameters ↪
↪reportParameters) {
131         BigDecimal unitPrice = (BigDecimal) values.get(0);
132         BigDecimal unitPriceTotal = (BigDecimal) values.get(1);
133         return unitPrice.divide(unitPriceTotal, 4, BigDecimal.ROUND_
↪HALF_UP).multiply(new BigDecimal(100));
134     }
135 }
136 }

```

1.22.3 Measure Expression Crosstab


DynamicReports
<http://www.dynamicreports.org>

MeasureExpressionCrosstab

Price1 = SUM(Quantity * unitPrice)

Price2 = SUM(Quantity * unitPrice)

State / Item	Book			DVD			Headbook			Total		
	Quantity	Price1	Price2	Quantity	Price1	Price2	Quantity	Price1	Price2	Quantity	Price1	Price2
Florida	21	1,071.00	242.00	9	1,152.00	309.00	1	480.00	480.00	31	19,899.00	1,071.00
New York	10	564.00	178.00	10	2,432.00	599.00	1	500.00	500.00	21	22,636.00	1,267.00
Washington	21	924.00	214.00	10	2,440.00	599.00	1	490.00	490.00	32	19,259.00	1,458.00
Total for state	52	7,462.00	634.00	29	11,722.00	1,505.00	3	4,710.00	1,470.00	101	218,342.00	3,754.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.crosstab;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
32 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
33 import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
34 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
35 import net.sf.dynamicreports.report.builder.crosstab.CrosstabVariableBuilder;
36 import net.sf.dynamicreports.report.constant.Calculation;
37 import net.sf.dynamicreports.report.constant.PageOrientation;
38 import net.sf.dynamicreports.report.constant.PageType;
39 import net.sf.dynamicreports.report.datasource.DRDataSource;
40 import net.sf.dynamicreports.report.definition.ReportParameters;
41 import net.sf.dynamicreports.report.exception.DRException;
42 import net.sf.jasperreports.engine.JRDataSource;

```

(continues on next page)

(continued from previous page)

```

43
44 /**
45  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46  */
47 public class MeasureExpressionCrosstabReport {
48     private CrosstabMeasureBuilder<Integer> quantityMeasure;
49     private CrosstabVariableBuilder<BigDecimal> unitPriceVariable;
50     private CrosstabVariableBuilder<BigDecimal> priceVariable;
51
52     public MeasureExpressionCrosstabReport() {
53         build();
54     }
55
56     private void build() {
57         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",
↪String.class)
58             .setTotalHeader("Total for state");
59
60         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↪"item", String.class);
61
62         unitPriceVariable = ctab.variable("unitprice", BigDecimal.class,
↪Calculation.SUM);
63         priceVariable = ctab.variable(new PriceExpression(), Calculation.SUM);
64         quantityMeasure = ctab.measure("Quantity", "quantity", Integer.class,
↪Calculation.SUM);
65         // price1 = sum(unitprice) * sum(quantity)
66         CrosstabMeasureBuilder<BigDecimal> priceMeasure1 = ctab.measure(
↪"Price1", new PriceMeasure1Expression());
67         priceMeasure1.setDataType(type.bigDecimalType());
68         // price2 = sum(unitprice * quantity)
69         CrosstabMeasureBuilder<BigDecimal> priceMeasure2 = ctab.measure(
↪"Price2", new PriceMeasure2Expression());
70         priceMeasure2.setDataType(type.bigDecimalType());
71
72         CrosstabBuilder crosstab = ctab.crosstab()
73             .headerCell(cmp.text("State / Item").
↪setStyle(Templates.boldCenteredStyle))
74             .setCellWidth(180)
75             .rowGroups(rowGroup)
76             .columnGroups(columnGroup)
77             .variables(unitPriceVariable, priceVariable)
78             .measures(quantityMeasure, priceMeasure1,
↪priceMeasure2);
79
80         try {
81             report()
82                 .setPageFormat(PageType.A4, PageOrientation.
↪LANDSCAPE)
83                 .setTemplate(Templates.reportTemplate)
84                 .title(
85                     Templates.
↪createTitleComponent("MeasureExpressionCrosstab"),
86                     cmp.text("Price1 =
↪SUM(quantity) * SUM(unitPrice)").setStyle(Templates.boldStyle),
87                     cmp.text("Price2 =
↪SUM(quantity * unitPrice)").setStyle(Templates.boldStyle))

```

(continues on next page)

(continued from previous page)

```

88         .summary(crosstab)
89         .pageFooter(Templates.footerComponent)
90         .setDataSource(createDataSource())
91         .show();
92     } catch (DRException e) {
93         e.printStackTrace();
94     }
95 }
96
97 private class PriceExpression extends AbstractSimpleExpression<BigDecimal> {
98     private static final long serialVersionUID = 1L;
99
100     @Override
101     public BigDecimal evaluate(ReportParameters reportParameters) {
102         Integer quantity = reportParameters.getValue("quantity");
103         BigDecimal unitPrice = reportParameters.getValue("unitprice");
104         return unitPrice.multiply(new BigDecimal(quantity));
105     }
106 }
107
108 private class PriceMeasure1Expression extends AbstractSimpleExpression
109 <BigDecimal> {
110     private static final long serialVersionUID = 1L;
111
112     @Override
113     public BigDecimal evaluate(ReportParameters reportParameters) {
114         Integer quantity = reportParameters.getValue(quantityMeasure);
115         BigDecimal unitPrice = reportParameters.
116         <getValue>(unitPriceVariable);
117         return unitPrice.multiply(new BigDecimal(quantity));
118     }
119 }
120
121 private class PriceMeasure2Expression extends AbstractSimpleExpression
122 <BigDecimal> {
123     private static final long serialVersionUID = 1L;
124
125     @Override
126     public BigDecimal evaluate(ReportParameters reportParameters) {
127         return reportParameters.getValue(priceVariable);
128     }
129 }
130
131 private JRDataSource createDataSource() {
132     DRDataSource dataSource = new DRDataSource("state", "item", "quantity
133     <","unitprice");
134     dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
135     dataSource.add("New York", "DVD", 5, new BigDecimal(30));
136     dataSource.add("New York", "DVD", 2, new BigDecimal(45));
137     dataSource.add("New York", "DVD", 4, new BigDecimal(36));
138     dataSource.add("New York", "DVD", 5, new BigDecimal(41));
139     dataSource.add("New York", "Book", 2, new BigDecimal(11));
140     dataSource.add("New York", "Book", 8, new BigDecimal(9));
141     dataSource.add("New York", "Book", 6, new BigDecimal(14));
142
143     dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
144     dataSource.add("Washington", "DVD", 4, new BigDecimal(40));

```

(continues on next page)

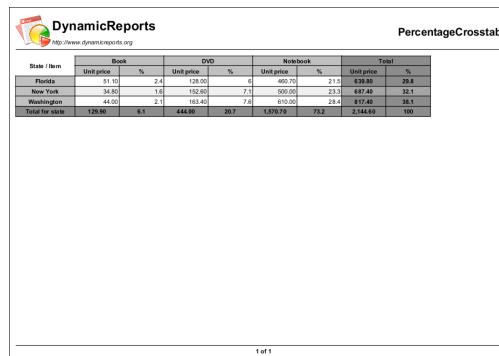
(continued from previous page)

```

141     dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
142     dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
143     dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
144     dataSource.add("Washington", "Book", 3, new BigDecimal(12));
145     dataSource.add("Washington", "Book", 9, new BigDecimal(8));
146     dataSource.add("Washington", "Book", 4, new BigDecimal(14));
147     dataSource.add("Washington", "Book", 5, new BigDecimal(10));
148
149     dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
150     dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
151     dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
152     dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
153     dataSource.add("Florida", "Book", 4, new BigDecimal(11));
154     dataSource.add("Florida", "Book", 8, new BigDecimal(6));
155     dataSource.add("Florida", "Book", 6, new BigDecimal(16));
156     dataSource.add("Florida", "Book", 3, new BigDecimal(18));
157     return dataSource;
158 }
159
160 public static void main(String[] args) {
161     new MeasureExpressionCrosstabReport();
162 }
163 }

```

1.22.4 Percentage Crosstab



State / Item	Book		DVD		Notebook		Total	
	Unit price	%	Unit price	%	Unit price	%	Unit price	%
Florida	51.10	2.4	128.00	6	460.70	21.0	639.80	29.4
New York	34.80	1.6	103.00	4.7	500.00	23.1	637.80	29.1
Washington	44.00	2.0	183.40	7.8	610.00	28.4	837.40	38.1
Total for state	129.90	6.1	414.40	20.7	1,570.70	72.4	2,114.00	100

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of

```

(continues on next page)

(continued from previous page)

```

16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.crosstab;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.FieldBuilder;
31 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
32 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
33 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
34 import net.sf.dynamicreports.report.constant.Calculation;
35 import net.sf.dynamicreports.report.constant.CrosstabPercentageType;
36 import net.sf.dynamicreports.report.constant.PageOrientation;
37 import net.sf.dynamicreports.report.constant.PageType;
38 import net.sf.dynamicreports.report.datasource.DRDataSource;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class PercentageCrosstabReport {
46
47     public PercentageCrosstabReport() {
48         build();
49     }
50
51     private void build() {
52         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state", ↵
↵String.class)
53             .setTotalHeader("Total for state");
54
55         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↵"item", String.class);
56
57         FieldBuilder<BigDecimal> unitPriceField = field("unitprice", ↵
↵BigDecimal.class);
58
59         CrosstabBuilder crosstab = ctab.crosstab()
60             .headerCell(cmp.text("State / Item").
↵setStyle(Templates.boldCenteredStyle))
61             .rowGroups(rowGroup)
62             .columnGroups(columnGroup)
63             .measures(
64                 ctab.measure("Unit price", ↵
↵unitPriceField, Calculation.SUM),
65                 ctab.measure("%", unitPriceField, ↵
↵Calculation.SUM).setPercentageType(CrosstabPercentageType.GRAND_TOTAL));
66

```

(continues on next page)

(continued from previous page)

```

67         try {
68             report()
69                 .setPageFormat(PageType.A4, PageOrientation.
↪ LANDSCAPE)
70                 .setTemplate(Templates.reportTemplate)
71                 .title(Templates.createTitleComponent(
↪ "PercentageCrosstab"))
72                 .summary(crosstab)
73                 .pageFooter(Templates.footerComponent)
74                 .setDataSource(createDataSource())
75                 .show();
76         } catch (DRException e) {
77             e.printStackTrace();
78         }
79     }
80
81     private JRDataSource createDataSource() {
82         DRDataSource dataSource = new DRDataSource("state", "item", "unitprice
↪ ");
83         dataSource.add("New York", "Notebook", new BigDecimal(500));
84         dataSource.add("New York", "DVD", new BigDecimal(30));
85         dataSource.add("New York", "DVD", new BigDecimal(45.6));
86         dataSource.add("New York", "DVD", new BigDecimal(36));
87         dataSource.add("New York", "DVD", new BigDecimal(41));
88         dataSource.add("New York", "Book", new BigDecimal(11));
89         dataSource.add("New York", "Book", new BigDecimal(9));
90         dataSource.add("New York", "Book", new BigDecimal(14.8));
91
92         dataSource.add("Washington", "Notebook", new BigDecimal(610));
93         dataSource.add("Washington", "DVD", new BigDecimal(40));
94         dataSource.add("Washington", "DVD", new BigDecimal(35));
95         dataSource.add("Washington", "DVD", new BigDecimal(46.4));
96         dataSource.add("Washington", "DVD", new BigDecimal(42));
97         dataSource.add("Washington", "Book", new BigDecimal(12));
98         dataSource.add("Washington", "Book", new BigDecimal(8));
99         dataSource.add("Washington", "Book", new BigDecimal(14));
100        dataSource.add("Washington", "Book", new BigDecimal(10));
101
102        dataSource.add("Florida", "Notebook", new BigDecimal(460.7));
103        dataSource.add("Florida", "DVD", new BigDecimal(49));
104        dataSource.add("Florida", "DVD", new BigDecimal(32));
105        dataSource.add("Florida", "DVD", new BigDecimal(47));
106        dataSource.add("Florida", "Book", new BigDecimal(11));
107        dataSource.add("Florida", "Book", new BigDecimal(6.1));
108        dataSource.add("Florida", "Book", new BigDecimal(16));
109        dataSource.add("Florida", "Book", new BigDecimal(18));
110        return dataSource;
111    }
112
113    public static void main(String[] args) {
114        new PercentageCrosstabReport();
115    }
116 }

```

1.22.5 Style Crosstab

DynamicReports
http://www.dynamicreports.org

StyleCrosstab

State / Item	Quantity	Unit price	Quantity	Unit price	Quantity	Unit price	Quantity	Unit price
Florida	21	\$1.00	5	120.00	1	400.00	21	630.00
New York	16	\$1.00	10	120.00	1	500.00	27	680.00
Washington	21	\$4.00	10	160.00	1	610.00	32	810.00
Total for state	58	120.00	25	440.00	3	1,510.00	80	2,120.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.crosstab;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28 import java.math.BigDecimal;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
32 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
33 import net.sf.dynamicreports.report.builder.crosstab.CrosstabMeasureBuilder;
34 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
35 import net.sf.dynamicreports.report.builder.style.ConditionalStyleBuilder;
36 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
37 import net.sf.dynamicreports.report.constant.Calculation;
38 import net.sf.dynamicreports.report.constant.PageOrientation;
39 import net.sf.dynamicreports.report.constant.PageType;
40 import net.sf.dynamicreports.report.datasource.DRDataSource;
41 import net.sf.dynamicreports.report.exception.DRException;
42 import net.sf.jasperreports.engine.JRDataSource;

```

(continues on next page)

(continued from previous page)

```

43
44 /**
45  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46  */
47 public class StyleCrosstabReport {
48
49     public StyleCrosstabReport() {
50         build();
51     }
52
53     private void build() {
54         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state",
↪String.class)
55             .setTotalHeader("Total for state");
56
57         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↪"item", String.class);
58
59         CrosstabMeasureBuilder<Integer> quantityMeasure = ctab.measure(
↪"Quantity", "quantity", Integer.class, Calculation.SUM);
60         CrosstabMeasureBuilder<BigDecimal> unitPriceMeasure = ctab.measure(
↪"Unit price", "unitprice", BigDecimal.class, Calculation.SUM);
61
62         ConditionalStyleBuilder condition1 = stl.conditionalStyle(cnd.
↪greater(unitPriceMeasure, 600))
63             .setBackgroundColor(new Color(210, 255, 210))
64             .setBorder(stl.pen1Point());
65         ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.
↪smaller(unitPriceMeasure, 150))
66             .setBackgroundColor(new Color(255, 210, 210))
67             .setBorder(stl.pen1Point());
68
69         StyleBuilder unitPriceStyle = stl.style()
70             .conditionalStyles(condition1, condition2)
71             .setBorder(stl.pen1Point());
72         StyleBuilder totalCellStyle = stl.style()
73             .setBackgroundColor(new Color(200, 200, 255))
74             .setBorder(stl.pen1Point());
75
76         unitPriceMeasure.setStyle(unitPriceStyle)
77             .setStyle(totalCellStyle, rowGroup)
78             .setStyle(totalCellStyle, rowGroup, columnGroup);
79         quantityMeasure.setStyle(totalCellStyle, rowGroup)
80             .setStyle(totalCellStyle, rowGroup, columnGroup);
81
82         CrosstabBuilder crosstab = ctab.crosstab()
83             .headerCell(cmp.text("State / Item").
↪setStyle(Templates.boldCenteredStyle))
84             .rowGroups(rowGroup)
85             .columnGroups(columnGroup)
86             .measures(quantityMeasure, unitPriceMeasure);
87
88         try {
89             report()
90                 .setPageFormat(PageType.A4, PageOrientation.
↪LANDSCAPE)
91                 .setTemplate(Templates.reportTemplate)

```

(continues on next page)






(continued from previous page)

```

92         .title(Templates.createTitleComponent(
↪ "StyleCrosstab"))
93         .summary(crosstab)
94         .pageFooter(Templates.footerComponent)
95         .setDataSource(createDataSource())
96         .show();
97     } catch (DRException e) {
98         e.printStackTrace();
99     }
100 }
101
102 private JRDataSource createDataSource() {
103     DRDataSource dataSource = new DRDataSource("state", "item", "quantity
↪ ", "unitprice");
104     dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
105     dataSource.add("New York", "DVD", 5, new BigDecimal(30));
106     dataSource.add("New York", "DVD", 2, new BigDecimal(45));
107     dataSource.add("New York", "DVD", 4, new BigDecimal(36));
108     dataSource.add("New York", "DVD", 5, new BigDecimal(41));
109     dataSource.add("New York", "Book", 2, new BigDecimal(11));
110     dataSource.add("New York", "Book", 8, new BigDecimal(9));
111     dataSource.add("New York", "Book", 6, new BigDecimal(14));
112
113     dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
114     dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
115     dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
116     dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
117     dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
118     dataSource.add("Washington", "Book", 3, new BigDecimal(12));
119     dataSource.add("Washington", "Book", 9, new BigDecimal(8));
120     dataSource.add("Washington", "Book", 4, new BigDecimal(14));
121     dataSource.add("Washington", "Book", 5, new BigDecimal(10));
122
123     dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
124     dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
125     dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
126     dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
127     dataSource.add("Florida", "Book", 4, new BigDecimal(11));
128     dataSource.add("Florida", "Book", 8, new BigDecimal(6));
129     dataSource.add("Florida", "Book", 6, new BigDecimal(16));
130     dataSource.add("Florida", "Book", 3, new BigDecimal(18));
131     return dataSource;
132 }
133
134 public static void main(String[] args) {
135     new StyleCrosstabReport();
136 }
137 }


```

Table 7: Column Examples

 CrosstabReport	 CustomPercentageCrosstabReport	 MeasureExpressionCrosstabReport
 PercentageCrosstabReport	 StyleCrosstabReport	

1.23 Datasource

1.23.1 Collection Datasource


DynamicReports
<http://www.dynamicreports.org>

CollectionDatasource

Item	Quantity	Unit price
DVD	5	30.00
SubData		
Order date	Quantity	
12/10/2010	2	
12/15/2010	3	
Book	8	11.00
SubData		
Order date	Quantity	
12/11/2010	1	
12/12/2010	3	
12/16/2010	4	
PDA	2	15.00
SubData		
Order date	Quantity	
12/09/2010	1	
12/18/2010	1	

1 of 1

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.ArrayList;
29 import java.util.Calendar;
30 import java.util.Date;
31 import java.util.List;
32
33 import net.sf.dynamicreports.examples.Templates;
34 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
35 import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38 import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class CollectionDatasourceReport {
44
45     public CollectionDatasourceReport() {
46         build();
47     }
48
49     private void build() {
50         SubreportBuilder subreport = cmp.subreport(createSubreport())
51             .setDataSource(exp.subDataSourceBeanCollection(
52 ↪ "subData"));
53
54         try {
55             report()
56                 .setTemplate(Templates.reportTemplate)
57                 .columns(
58                     col.column("Item", "item", ↪
59 ↪ type.stringType()),
60                     col.column("Quantity",
61 ↪ "quantity", type.integerType()),
62                     col.column("Unit price",
63 ↪ "unitPrice", type.bigDecimalType()))
64                 .title(Templates.createTitleComponent(
65 ↪ "CollectionDatasource"))

```

(continues on next page)

(continued from previous page)

```

61         .detailFooter(
62             cmp.horizontalList(cmp.
↪horizontalGap(150), subreport, cmp.horizontalGap(150)),
63             cmp.line())
64         .pageFooter(Templates.footerComponent)
65         .setDataSource(createDataSource())
66         .show();
67     } catch (DRException e) {
68         e.printStackTrace();
69     }
70 }
71
72 private JasperReportBuilder createSubreport() {
73     JasperReportBuilder report = report();
74     report
75         .setTemplate(Templates.reportTemplate)
76         .title(cmp.text("SubData").setStyle(Templates.
↪boldCenteredStyle))
77         .columns(
78             col.column("Order date", "orderDate", ↪
↪type.dateType()),
79             col.column("Quantity", "quantity", ↪
↪type.integerType()));
80
81     return report;
82 }
83
84 private JRDataSource createDataSource() {
85     List<Data> data = new ArrayList<Data>();
86
87     List<SubData> subData = new ArrayList<SubData>();
88     subData.add(new SubData(toDate(2011, 0, 10), 2));
89     subData.add(new SubData(toDate(2011, 0, 15), 3));
90     data.add(new Data("DVD", 5, new BigDecimal(30), subData));
91
92     subData = new ArrayList<SubData>();
93     subData.add(new SubData(toDate(2011, 0, 11), 1));
94     subData.add(new SubData(toDate(2011, 0, 12), 3));
95     subData.add(new SubData(toDate(2011, 0, 16), 4));
96     data.add(new Data("Book", 8, new BigDecimal(11), subData));
97
98     subData = new ArrayList<SubData>();
99     subData.add(new SubData(toDate(2011, 0, 9), 1));
100    subData.add(new SubData(toDate(2011, 0, 18), 1));
101    data.add(new Data("PDA", 2, new BigDecimal(15), subData));
102
103    return new JRBeanCollectionDataSource(data);
104 }
105
106 private Date toDate(int year, int month, int day) {
107     Calendar c = Calendar.getInstance();
108     c.set(Calendar.YEAR, year);
109     c.set(Calendar.MONTH, month - 1);
110     c.set(Calendar.DAY_OF_MONTH, day);
111     return c.getTime();
112 }
113

```

(continues on next page)

(continued from previous page)

```

114     public static void main(String[] args) {
115         new CollectionDatasourceReport();
116     }
117
118     public class Data {
119         private String item;
120         private Integer quantity;
121         private BigDecimal unitPrice;
122         private List<SubData> subData;
123
124         public Data(String item, Integer quantity, BigDecimal unitPrice, List
125         ↪<SubData> subData) {
126             this.item = item;
127             this.quantity = quantity;
128             this.unitPrice = unitPrice;
129             this.subData = subData;
130         }
131
132         public String getItem() {
133             return item;
134         }
135
136         public void setItem(String item) {
137             this.item = item;
138         }
139
140         public Integer getQuantity() {
141             return quantity;
142         }
143
144         public void setQuantity(Integer quantity) {
145             this.quantity = quantity;
146         }
147
148         public BigDecimal getUnitPrice() {
149             return unitPrice;
150         }
151
152         public void setUnitPrice(BigDecimal unitPrice) {
153             this.unitPrice = unitPrice;
154         }
155
156         public List<SubData> getSubData() {
157             return subData;
158         }
159
160         public void setSubData(List<SubData> subData) {
161             this.subData = subData;
162         }
163
164     }
165
166     public class SubData {
167         private Date orderDate;
168         private Integer quantity;
169
170         public SubData(Date orderDate, Integer quantity) {
171             this.orderDate = orderDate;

```

(continues on next page)

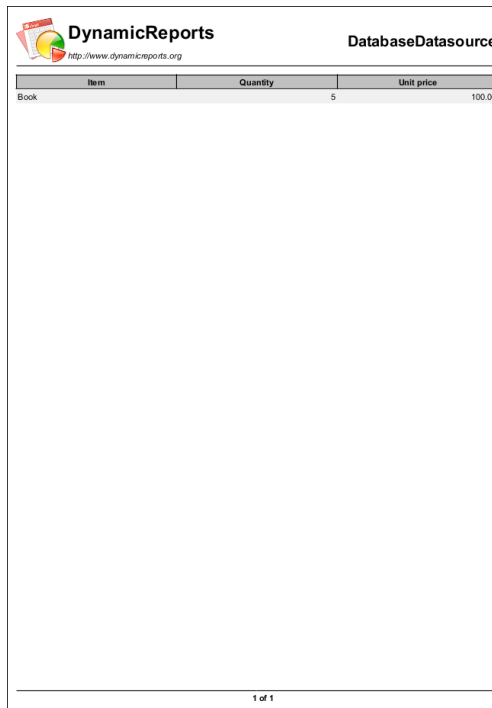
(continued from previous page)

```

170         this.quantity = quantity;
171     }
172
173     public Date getOrderDate() {
174         return orderDate;
175     }
176
177     public void setOrderDate(Date orderDate) {
178         this.orderDate = orderDate;
179     }
180
181     public Integer getQuantity() {
182         return quantity;
183     }
184
185     public void setQuantity(Integer quantity) {
186         this.quantity = quantity;
187     }
188 }
189

```

1.23.2 Database Datasource



The image shows a preview of a report generated by DynamicReports. The report has a header section with the DynamicReports logo and URL on the left, and the title 'DatabaseDatasource' on the right. Below the header is a table with three columns: 'Item', 'Quantity', and 'Unit price'. The table contains one data row with the values 'Book', '5', and '100.00'. At the bottom of the report, there is a footer indicating '1 of 1'.

Item	Quantity	Unit price
Book	5	100.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *

```

(continues on next page)

(continued from previous page)

```

7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.sql.Connection;
28 import java.sql.DriverManager;
29 import java.sql.SQLException;
30 import java.sql.Statement;
31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.report.exception.DRException;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class DatabaseDatasourceReport {
39     private Connection connection;
40
41     public DatabaseDatasourceReport() {
42         try {
43             Class.forName("org.hsqldb.jdbcDriver");
44             connection = DriverManager.getConnection("jdbc:hsqldb:mem:test
↳");
45             createTable();
46             build();
47         } catch (SQLException e) {
48             e.printStackTrace();
49         } catch (ClassNotFoundException e) {
50             e.printStackTrace();
51         }
52     }
53
54     private void build() {
55         try {
56             report()
57                 .setTemplate(Templates.reportTemplate)
58                 .columns(
59                     col.column("Item", "item",
↳type.stringType()),
60                     col.column("Quantity",
↳"quantity", type.integerType()),

```

(continues on next page)


(continued from previous page)

```

61                                     col.column("Unit price",
↪ "unitprice", type.bigDecimalType()))
62                                     .title(Templates.createTitleComponent(
↪ "DatabaseDatasource"))
63                                     .pageFooter(Templates.footerComponent)
64                                     .setDataSource("SELECT * FROM sales", ↪
↪ connection)
65                                     .show();
66     } catch (DRException e) {
67         e.printStackTrace();
68     }
69 }
70
71 private void createTable() throws SQLException {
72     Statement st = connection.createStatement();
73     st.execute("CREATE TABLE sales (item VARCHAR(50), quantity INTEGER, ↪
↪ unitprice DECIMAL)");
74     st.execute("INSERT INTO sales VALUES ('Book', 5, 100)");
75 }
76
77 public static void main(String[] args) {
78     new DatabaseDatasourceReport();
79 }
80 }

```

1.23.3 Data Filter



DynamicReports

<http://www.dynamicreports.org>

DataFilter

Item	Quantity	Unit price
Book	3	11.0
Book	1	15.0
Book	5	10.0
Book	8	9.0

1 of 1

```

1  /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically

```

(continues on next page)

(continued from previous page)

```

3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.definition.ReportParameters;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class DataFilterReport {
40
41     public DataFilterReport() {
42         build();
43     }
44
45     private void build() {
46         try {
47             report()
48                 .setTemplate(Templates.reportTemplate)
49                 .columns(
50                     col.column("Item", "item",
51 ↪ type.stringType()),
52                     col.column("Quantity",
53 ↪ "quantity", type.integerType()),
54                     col.column("Unit price",
55 ↪ "unitprice", type.bigDecimalType()))
56                 .title(Templates.createTitleComponent(
57 ↪ "DataFilter"))
58                 .pageFooter(Templates.footerComponent)
59                 .setFilterExpression(new FilterExpression())

```

(continues on next page)


(continued from previous page)

```

56         .setDataSource(createDataSource())
57         .show();
58     } catch (DRException e) {
59         e.printStackTrace();
60     }
61 }
62
63 private JRDataSource createDataSource() {
64     DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
65     dataSource.add("DVD", 5, new BigDecimal(30));
66     dataSource.add("DVD", 1, new BigDecimal(28));
67     dataSource.add("DVD", 5, new BigDecimal(32));
68     dataSource.add("Book", 3, new BigDecimal(11));
69     dataSource.add("Book", 1, new BigDecimal(15));
70     dataSource.add("Book", 5, new BigDecimal(10));
71     dataSource.add("Book", 8, new BigDecimal(9));
72     return dataSource;
73 }
74
75 public static void main(String[] args) {
76     new DataFilterReport();
77 }
78
79 private class FilterExpression extends AbstractSimpleExpression<Boolean> {
80     private static final long serialVersionUID = 1L;
81
82     @Override
83     public Boolean evaluate(ReportParameters reportParameters) {
84         return reportParameters.getValue("item").equals("Book");
85     }
86 }
87 }

```

1.23.4 Sort



DynamicReports

<http://www.dynamicreports.org>

Sort

Item	Quantity	Unit price	
Book	1		15.00
Book	3		11.00
Book	5		10.00
Book	8		9.00
DVD	5		32.00
DVD	5		30.00
DVD	1		28.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)

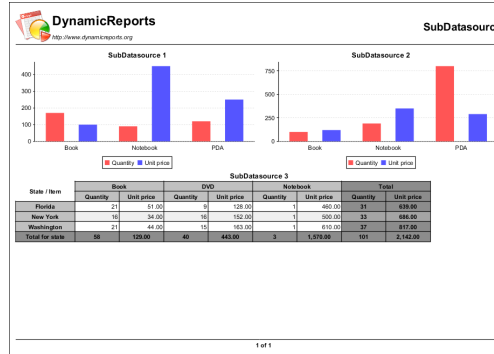
(continued from previous page)

```

31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class SortReport {
39
40     public SortReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             TextColumnBuilder<String> itemColumn = col.column("Item",
↪ "item", type.stringType());
47             TextColumnBuilder<Integer> quantityColumn = col.column(
↪ "Quantity", "quantity", type.integerType());
48             TextColumnBuilder<BigDecimal> unitPriceColumn = col.column(
↪ "Unit price", "unitprice", type.bigDecimalType());
49             report()
50                 .setTemplate(Templates.reportTemplate)
51                 .columns(itemColumn, quantityColumn, ↪
↪ unitPriceColumn)
52                 .title(Templates.createTitleComponent("Sort"))
53                 .pageFooter(Templates.footerComponent)
54                 .sortBy(
55                     asc(itemColumn), ↪
↪ desc(unitPriceColumn))
56                 .setDataSource(createDataSource())
57                 .show();
58         } catch (DRException e) {
59             e.printStackTrace();
60         }
61     }
62
63     private JRDataSource createDataSource() {
64         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
65         dataSource.add("Book", 1, new BigDecimal(15));
66         dataSource.add("DVD", 5, new BigDecimal(30));
67         dataSource.add("DVD", 1, new BigDecimal(28));
68         dataSource.add("Book", 5, new BigDecimal(10));
69         dataSource.add("DVD", 5, new BigDecimal(32));
70         dataSource.add("Book", 3, new BigDecimal(11));
71         dataSource.add("Book", 8, new BigDecimal(9));
72         return dataSource;
73     }
74
75     public static void main(String[] args) {
76         new SortReport();
77     }
78 }

```

1.23.5 Sub Datasource



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.FieldBuilder;
31 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
32 import net.sf.dynamicreports.report.builder.chart.CategoryChartSerieBuilder;
33 import net.sf.dynamicreports.report.builder.crosstab.CrosstabBuilder;
34 import net.sf.dynamicreports.report.builder.crosstab.CrosstabColumnGroupBuilder;
35 import net.sf.dynamicreports.report.builder.crosstab.CrosstabRowGroupBuilder;
36 import net.sf.dynamicreports.report.builder.style.FontBuilder;
37 import net.sf.dynamicreports.report.constant.Calculation;
38 import net.sf.dynamicreports.report.constant.PageOrientation;
39 import net.sf.dynamicreports.report.constant.PageType;
40 import net.sf.dynamicreports.report.datasource.DRDataSource;
41 import net.sf.dynamicreports.report.exception.DRException;
42 import net.sf.jasperreports.engine.JRDataSource;

```

(continues on next page)

(continued from previous page)

```

43
44 /**
45  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46  */
47 public class SubDataSourceReport {
48
49     public SubDataSourceReport() {
50         build();
51     }
52
53     private void build() {
54         FontBuilder boldFont = stl.fontArialBold().setFontSize(12);
55
56         FieldBuilder<String> itemField = field("item", type.stringType());
57         FieldBuilder<Integer> quantityField = field("quantity", type.
↪integerType());
58         FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
↪bigDecimalType());
59
60         CategoryChartSerieBuilder quantitySerie = cht.serie(quantityField).
↪setLabel("Quantity");
61         CategoryChartSerieBuilder unitPriceSerie = cht.serie(unitPriceField).
↪setLabel("Unit price");
62
63         BarChartBuilder chart1 = cht.barChart()
64             .setDataSource(createDataSource1())
65             .setTitle("SubDataSource 1")
66             .setTitleFont(boldFont)
67             .setCategory(itemField)
68             .series(
69                 quantitySerie, unitPriceSerie);
70
71         BarChartBuilder chart2 = cht.barChart()
72             .setDataSource(createDataSource2())
73             .setTitle("SubDataSource 2")
74             .setTitleFont(boldFont)
75             .setCategory(itemField)
76             .series(
77                 quantitySerie, unitPriceSerie);
78
79         CrosstabRowGroupBuilder<String> rowGroup = ctab.rowGroup("state", ↪
↪String.class)
80             .setTotalHeader("Total for state");
81
82         CrosstabColumnGroupBuilder<String> columnGroup = ctab.columnGroup(
↪"item", String.class);
83
84         CrosstabBuilder crosstab = ctab.crosstab()
85             .setDataSource(createDataSource3())
86             .headerCell(cmp.text("State / Item").
↪setStyle(Templates.boldCenteredStyle))
87             .rowGroups(rowGroup)
88             .columnGroups(columnGroup)
89             .measures(
90                 ctab.measure("Quantity", "quantity", ↪
↪Integer.class, Calculation.SUM),
91                 ctab.measure("Unit price", "unitprice
↪", BigDecimal.class, Calculation.SUM));

```

(continues on next page)

(continued from previous page)

```

92         try {
93             report()
94                 .setPageFormat(PageType.A4, PageOrientation.
95 ↪ LANDSCAPE)
96                 .setTemplate(Templates.reportTemplate)
97                 .title(
98                     Templates.
99 ↪ createTitleComponent("SubDatasource"),
100                     cmp.horizontalList(chart1, ↪
101 ↪ chart2),
102                     cmp.text("SubDatasource 3").
103 ↪ setStyle(Templates.bold12CenteredStyle),
104                     crosstab)
105                 .pageFooter(Templates.footerComponent)
106                 .show();
107         } catch (DRException e) {
108             e.printStackTrace();
109         }
110     }
111
112     private JRDataSource createDataSource1() {
113         DRDataSource dataSource = new DRDataSource("item", "quantity",
114 ↪ "unitprice");
115         dataSource.add("Book", 170, new BigDecimal(100));
116         dataSource.add("Notebook", 90, new BigDecimal(450));
117         dataSource.add("PDA", 120, new BigDecimal(250));
118         return dataSource;
119     }
120
121     private JRDataSource createDataSource2() {
122         DRDataSource dataSource = new DRDataSource("item", "quantity",
123 ↪ "unitprice");
124         dataSource.add("Book", 100, new BigDecimal(120));
125         dataSource.add("Notebook", 190, new BigDecimal(350));
126         dataSource.add("PDA", 800, new BigDecimal(290));
127         return dataSource;
128     }
129
130     private JRDataSource createDataSource3() {
131         DRDataSource dataSource = new DRDataSource("state", "item", "quantity
132 ↪ ", "unitprice");
133         dataSource.add("New York", "Notebook", 1, new BigDecimal(500));
134         dataSource.add("New York", "DVD", 5, new BigDecimal(30));
135         dataSource.add("New York", "DVD", 2, new BigDecimal(45));
136         dataSource.add("New York", "DVD", 4, new BigDecimal(36));
137         dataSource.add("New York", "DVD", 5, new BigDecimal(41));
138         dataSource.add("New York", "Book", 2, new BigDecimal(11));
139         dataSource.add("New York", "Book", 8, new BigDecimal(9));
140         dataSource.add("New York", "Book", 6, new BigDecimal(14));
141
142         dataSource.add("Washington", "Notebook", 1, new BigDecimal(610));
143         dataSource.add("Washington", "DVD", 4, new BigDecimal(40));
144         dataSource.add("Washington", "DVD", 6, new BigDecimal(35));
145         dataSource.add("Washington", "DVD", 3, new BigDecimal(46));
146         dataSource.add("Washington", "DVD", 2, new BigDecimal(42));
147         dataSource.add("Washington", "Book", 3, new BigDecimal(12));

```

(continues on next page)

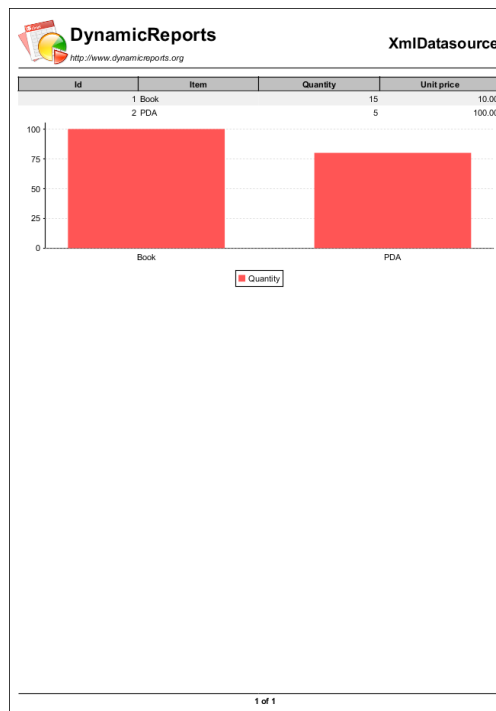
(continued from previous page)

```

142     dataSource.add("Washington", "Book", 9, new BigDecimal(8));
143     dataSource.add("Washington", "Book", 4, new BigDecimal(14));
144     dataSource.add("Washington", "Book", 5, new BigDecimal(10));
145
146     dataSource.add("Florida", "Notebook", 1, new BigDecimal(460));
147     dataSource.add("Florida", "DVD", 3, new BigDecimal(49));
148     dataSource.add("Florida", "DVD", 4, new BigDecimal(32));
149     dataSource.add("Florida", "DVD", 2, new BigDecimal(47));
150     dataSource.add("Florida", "Book", 4, new BigDecimal(11));
151     dataSource.add("Florida", "Book", 8, new BigDecimal(6));
152     dataSource.add("Florida", "Book", 6, new BigDecimal(16));
153     dataSource.add("Florida", "Book", 3, new BigDecimal(18));
154     return dataSource;
155 }
156
157 public static void main(String[] args) {
158     new SubDatasourceReport();
159 }
160 }

```

1.23.6 Xml Datasource



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.

```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.datasource;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.FieldBuilder;
31 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRException;
34 import net.sf.jasperreports.engine.data.JRXmlDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class XmlDataSourceReport {
40
41     public XmlDataSourceReport() {
42         build();
43     }
44
45     private void build() {
46         try {
47             JRXmlDataSource dataSource = new
48 ↪ JRXmlDataSource(XmlDataSourceReport.class.getResourceAsStream("sales.xml"), "/sales/
49 ↪ item");
50
51             JRXmlDataSource chartDataSource = dataSource.dataSource("/
52 ↪ sales/chart/item");
53
54             FieldBuilder<Object> idField = field("id", type.integerType())
55                 .setDescription("@id");
56             FieldBuilder<String> itemField = field("item", type.
57 ↪ stringType());
58             FieldBuilder<Integer> quantityField = field("quantity", type.
59 ↪ integerType());
60             FieldBuilder<BigDecimal> unitPriceField = field("unitprice",
61 ↪ type.bigDecimalType());
62
63             BarChartBuilder barChart = cht.barChart()
64                 .setDataSource(chartDataSource)
65                 .setCategory(itemField)

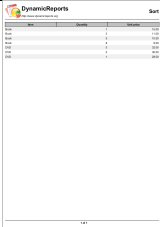
```

(continues on next page)

(continued from previous page)

```
59         .series(  
60             cht.series(quantityField).  
61             ↪setLabel("Quantity"));  
62     report()  
63         .setTemplate(Templates.reportTemplate)  
64         .columns(  
65             col.column("Id", idField),  
66             col.column("Item", itemField),  
67             ↪quantityField),  
68             ↪unitPriceField))  
69         .title(Templates.createTitleComponent(  
70             ↪"XmlDatasource"))  
71         .summary(barChart)  
72         .pageFooter(Templates.footerComponent)  
73         .setDataSource(dataSource)  
74         .show();  
75     } catch (DRException e) {  
76         e.printStackTrace();  
77     } catch (JRException e) {  
78         e.printStackTrace();  
79     }  
80 }  
81 public static void main(String[] args) {  
82     new XmlDatasourceReport();  
83 }  
84 }
```

Table 8: Column Examples

		
CollectionDatasourceReport	DatabaseDatasourceReport	DataFilterReport
		
SortReport	SubDatasourceReport	XmlDatasourceReport

1.24 Exporter

1.24.1 Encrypted Pdf

DynamicReports			EncryptedPdfReport
http://www.dynamicreports.org			
Item	Quantity	Unit price	
Book	7	92.096	
Book	2	97.569	
Book	3	65.609	
Book	1	93.808	
Book	7	20.141	
Book	6	67.518	
Book	3	78.866	
Book	4	91.625	
Book	1	10.757	
Book	3	41.087	
Book	4	15.957	
Book	6	94.28	
Book	7	3.074	
Book	2	40.161	
Book	4	100.975	
Book	7	95.14	
Book	8	39.90	
Book	9	77.445	
Book	9	57.166	
Book	5	28.725	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.exporter;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;

```

(continues on next page)

(continued from previous page)

```

28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.export.JasperPdfExporterBuilder;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class EncryptedPdfReport {
39
40     public EncryptedPdfReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             JasperPdfExporterBuilder pdfExporter = export.pdfExporter("c:/
↪report.pdf")
47                 .setEncrypted(true)
48                 .setUserPassword("1234");
49
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(
53                     col.column("Item", "item",
↪type.stringType()),
54                     col.column("Quantity",
↪"quantity", type.integerType()),
55                     col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
56                 .title(Templates.createTitleComponent(
↪"EncryptedPdfReport"))
57                 .pageFooter(Templates.footerComponent)
58                 .setDataSource(createDataSource())
59                 .toPdf(pdfExporter);
60         } catch (DRException e) {
61             e.printStackTrace();
62         }
63     }
64
65     private JRDataSource createDataSource() {
66         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
67         for (int i = 0; i < 20; i++) {
68             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪BigDecimal(Math.random() * 100 + 1));
69         }
70         return dataSource;
71     }
72
73     public static void main(String[] args) {
74         new EncryptedPdfReport();
75     }
76 }

```

1.24.2 Excel 1

Item	Quantity	Unit price
Book	4	24,238
Book	2	91,841
Book	3	39,703
Book	6	31,667
Book	2	68,379
Book	6	72,355
Book	8	28,255
Book	4	63,426
Book	5	73,515
Book	10	27,053
Book	2	8,264
Book	9	29,859
Book	1	7,125
Book	5	85,24
Book	4	46,506
Book	2	50,246
Book	10	21,545
Book	2	66,284
Book	5	48,059
Book	1	81,291
Book	3	26,244
Book	10	82,535
Book	10	73,472
Book	3	63,822
Book	7	17,185
Book	2	48,646
Book	10	75,732
Book	2	91,706
Book	6	44,922
Book	3	85,189
Book	10	51,142
Book	2	3,389
Book	9	28,057
Book	8	29,451
Book	9	23,268
Book	1	21,425
Book	10	43,70
Book	5	91,311
Book	8	4,085
Book	6	90,414
Book	7	51,592
Book	10	17,074
Book	7	1,471
Book	2	88,715
Book	3	33,508
Book	10	49,169
Book	10	67,184
Book	5	14,346
Book	7	54,586
Book	6	94,589

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.exporter;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.export.JasperXlsExporterBuilder;
31 import net.sf.dynamicreports.jasper.constant.JasperProperty;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class ExcelReport1 {
40
41     public ExcelReport1() {
42         build();
43     }
44
45     private void build() {
46         try {
47             JasperXlsExporterBuilder xlsExporter = export.xlsExporter("c:/
↪report.xls")
48
49                 .setDetectCellType(true)
50                 .setIgnorePageMargins(true)
51                 .setWhitePageBackground(false)
52                 .setRemoveEmptySpaceBetweenColumns(true);
53
54             report()
55                 .setColumnTitleStyle(Templates.
↪columnTitleStyle)
56                 .addProperty(JasperProperty.EXPORT_XLS_FREEZE_
↪ROW, "2")
57                 .ignorePageWidth()
58                 .ignorePagination()
59                 .columns(
60                     col.column("Item", "item",
↪type.stringType()),
61                     col.column("Quantity",
↪"quantity", type.integerType()),

```

(continues on next page)

(continued from previous page)

```

61                                     col.column("Unit price",
↪ "unitprice", type.bigDecimalType()))
62                                     .setDataSource(createDataSource())
63                                     .toXls(xlsExporter);
64         } catch (DRException e) {
65             e.printStackTrace();
66         }
67     }
68
69     private JRDataSource createDataSource() {
70         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
71         for (int i = 0; i < 50; i++) {
72             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
73         }
74         return dataSource;
75     }
76
77     public static void main(String[] args) {
78         new ExcelReport1();
79     }
80 }

```

1.24.3 Excel 2

Item	Quantity	Unit price
Very	5	4,462
Very	2	70,434
Very	10	78,238
Very	1	39,965
Very	3	79,392

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License

```

(continues on next page)

(continued from previous page)

```

20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.exporter;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.jasper.builder.export.JasperXlsExporterBuilder;
31  import net.sf.dynamicreports.jasper.constant.JasperProperty;
32  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33  import net.sf.dynamicreports.report.datasource.DRDataSource;
34  import net.sf.dynamicreports.report.exception.DRException;
35  import net.sf.jasperreports.engine.JRDataSource;
36
37  /**
38   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39   */
40  public class ExcelReport2 {
41
42      public ExcelReport2() {
43          build();
44      }
45
46      private void build() {
47          try {
48              JasperXlsExporterBuilder xlsExporter = export.xlsExporter("c:/
↳report.xls")
49
50                  .setDetectCellType(true)
51                  .setIgnorePageMargins(true)
52                  .setWhitePageBackground(false)
53                  .setRemoveEmptySpaceBetweenColumns(true);
54
55              TextColumnBuilder<String> itemColumn = col.column("Item",
↳"item", type.stringType())
56
57                  .setFixedWidth(30)
58                  .setStretchWithOverflow(false)
59                  .addProperty(JasperProperty.PRINT_KEEP_FULL_
↳TEXT, "true");
60
61              report()
62
63                  .setColumnTitleStyle(Templates.
↳columnTitleStyle)
64
65                  .addProperty(JasperProperty.EXPORT_XLS_FREEZE_
↳ROW, "2")
66
67                  .ignorePageWidth()
68                  .ignorePagination()
69                  .columns(
70
71                      itemColumn,
72                      col.column("Quantity",
↳"quantity", type.integerType()),
73
74                      col.column("Unit price",
↳"unitprice", type.bigDecimalType()))
75
76                  .setDataSource(createDataSource())
77                  .toXls(xlsExporter);

```

(continues on next page)

(continued from previous page)

```

70         } catch (DRException e) {
71             e.printStackTrace();
72         }
73     }
74
75     private JRDataSource createDataSource() {
76         DRDataSource dataSource = new DRDataSource("item", "quantity",
77 ↪ "unitprice");
78         for (int i = 0; i < 5; i++) {
79             dataSource.add("Very long book name", (int) (Math.random() *
80 ↪ 10) + 1, new BigDecimal(Math.random() * 100 + 1));
81         }
82         return dataSource;
83     }
84
85     public static void main(String[] args) {
86         new ExcelReport2();
87     }

```

1.24.4 Html

DynamicReports			HtmlReport
 http://www.dynamicreports.org			
Item	Quantity	Unit price	
Book	2	100.641	
Book	3	91.262	
Book	1	76.172	
Book	6	5.185	
Book	6	49.926	
Book	2	20.813	
Book	6	20.356	
Book	1	93.498	
Book	7	51.048	
Book	9	49.961	
Book	9	48.262	
Book	5	91.94	
Book	9	62.20	
Book	2	62.785	
Book	6	74.764	
Book	8	51.913	
Book	3	6.159	
Book	2	77.74	
Book	5	45.491	
Book	7	95.918	
			1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.

```

(continues on next page)

(continued from previous page)

```

8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.exporter;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class HtmlReport {
39
40     public HtmlReport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             JasperHtmlExporterBuilder htmlExporter = export.htmlExporter(
47 ↪ "c:/report.html")
48                 .setImagesDirName("c:/images")
49                 .setOutputImagesToDir(true);
50
51             report()
52                 .setTemplate(Templates.reportTemplate)
53                 .columns(
54                     col.column("Item", "item",
55 ↪ type.stringType()),
56                     col.column("Quantity",
57 ↪ "quantity", type.integerType()),
58                     col.column("Unit price",
59 ↪ "unitprice", type.bigDecimalType()))
60                 .title(Templates.createTitleComponent(
61 ↪ "HtmlReport"))
62                 .pageFooter(Templates.footerComponent)
63                 .setDataSource(createDataSource())
64                 .toHtml(htmlExporter);
65         } catch (DRException e) {
66             // ...
67         }
68     }
69 }

```

(continues on next page)

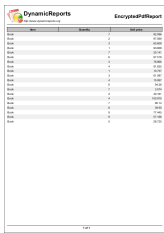
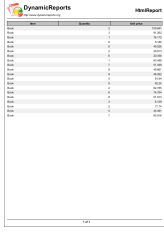
(continued from previous page)

```

60         } catch (DRException e) {
61             e.printStackTrace();
62         }
63     }
64
65     private JRDataSource createDataSource() {
66         DRDataSource dataSource = new DRDataSource("item", "quantity",
67 ↪ "unitprice");
68         for (int i = 0; i < 20; i++) {
69             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
70 ↪ BigDecimal(Math.random() * 100 + 1));
71         }
72         return dataSource;
73     }
74
75     public static void main(String[] args) {
76         new HtmlReport();
77     }


```

Table 9: Exporter Examples

 <p>EncryptedPdfReport</p>	<table border="1"> <thead> <tr> <th>Item</th><th>Quantity</th><th>Unit price</th></tr> </thead> <tbody> <tr><td>Book</td><td>4</td><td>24.228</td></tr> <tr><td>Book</td><td>2</td><td>37.841</td></tr> <tr><td>Book</td><td>3</td><td>36.703</td></tr> <tr><td>Book</td><td>6</td><td>37.667</td></tr> <tr><td>Book</td><td>2</td><td>68.379</td></tr> <tr><td>Book</td><td>6</td><td>72.399</td></tr> <tr><td>Book</td><td>8</td><td>26.235</td></tr> <tr><td>Book</td><td>4</td><td>60.698</td></tr> <tr><td>Book</td><td>5</td><td>73.615</td></tr> <tr><td>Book</td><td>10</td><td>27.059</td></tr> <tr><td>Book</td><td>2</td><td>4.094</td></tr> <tr><td>Book</td><td>9</td><td>29.039</td></tr> <tr><td>Book</td><td>1</td><td>71.525</td></tr> <tr><td>Book</td><td>6</td><td>45.24</td></tr> <tr><td>Book</td><td>4</td><td>46.266</td></tr> <tr><td>Book</td><td>2</td><td>32.586</td></tr> <tr><td>Book</td><td>10</td><td>27.455</td></tr> <tr><td>Book</td><td>2</td><td>46.264</td></tr> <tr><td>Book</td><td>5</td><td>46.225</td></tr> <tr><td>Book</td><td>1</td><td>81.291</td></tr> <tr><td>Book</td><td>5</td><td>30.244</td></tr> <tr><td>Book</td><td>10</td><td>62.535</td></tr> <tr><td>Book</td><td>10</td><td>75.472</td></tr> <tr><td>Book</td><td>3</td><td>63.622</td></tr> <tr><td>Book</td><td>2</td><td>17.186</td></tr> <tr><td>Book</td><td>2</td><td>46.646</td></tr> <tr><td>Book</td><td>10</td><td>70.732</td></tr> <tr><td>Book</td><td>2</td><td>37.739</td></tr> <tr><td>Book</td><td>6</td><td>44.622</td></tr> <tr><td>Book</td><td>3</td><td>66.189</td></tr> <tr><td>Book</td><td>10</td><td>51.162</td></tr> <tr><td>Book</td><td>6</td><td>3.386</td></tr> <tr><td>Book</td><td>9</td><td>28.017</td></tr> <tr><td>Book</td><td>8</td><td>20.471</td></tr> <tr><td>Book</td><td>9</td><td>23.268</td></tr> <tr><td>Book</td><td>1</td><td>27.425</td></tr> <tr><td>Book</td><td>10</td><td>45.775</td></tr> <tr><td>Book</td><td>5</td><td>30.313</td></tr> <tr><td>Book</td><td>6</td><td>4.695</td></tr> <tr><td>Book</td><td>6</td><td>36.414</td></tr> <tr><td>Book</td><td>7</td><td>61.562</td></tr> <tr><td>Book</td><td>10</td><td>77.074</td></tr> <tr><td>Book</td><td>2</td><td>1.471</td></tr> <tr><td>Book</td><td>2</td><td>80.715</td></tr> <tr><td>Book</td><td>5</td><td>30.556</td></tr> <tr><td>Book</td><td>10</td><td>40.109</td></tr> <tr><td>Book</td><td>10</td><td>67.184</td></tr> <tr><td>Book</td><td>5</td><td>14.266</td></tr> <tr><td>Book</td><td>7</td><td>56.686</td></tr> <tr><td>Book</td><td>6</td><td>34.580</td></tr> </tbody> </table> <p>ExcelReport1</p>	Item	Quantity	Unit price	Book	4	24.228	Book	2	37.841	Book	3	36.703	Book	6	37.667	Book	2	68.379	Book	6	72.399	Book	8	26.235	Book	4	60.698	Book	5	73.615	Book	10	27.059	Book	2	4.094	Book	9	29.039	Book	1	71.525	Book	6	45.24	Book	4	46.266	Book	2	32.586	Book	10	27.455	Book	2	46.264	Book	5	46.225	Book	1	81.291	Book	5	30.244	Book	10	62.535	Book	10	75.472	Book	3	63.622	Book	2	17.186	Book	2	46.646	Book	10	70.732	Book	2	37.739	Book	6	44.622	Book	3	66.189	Book	10	51.162	Book	6	3.386	Book	9	28.017	Book	8	20.471	Book	9	23.268	Book	1	27.425	Book	10	45.775	Book	5	30.313	Book	6	4.695	Book	6	36.414	Book	7	61.562	Book	10	77.074	Book	2	1.471	Book	2	80.715	Book	5	30.556	Book	10	40.109	Book	10	67.184	Book	5	14.266	Book	7	56.686	Book	6	34.580	<table border="1"> <thead> <tr> <th>Item</th><th>Quantity</th><th>Unit price</th></tr> </thead> <tbody> <tr><td>Vary</td><td>5</td><td>4.452</td></tr> <tr><td>Vary</td><td>2</td><td>70.534</td></tr> <tr><td>Vary</td><td>10</td><td>79.238</td></tr> <tr><td>Vary</td><td>1</td><td>38.965</td></tr> <tr><td>Vary</td><td>3</td><td>79.352</td></tr> </tbody> </table> <p>ExcelReport2</p>	Item	Quantity	Unit price	Vary	5	4.452	Vary	2	70.534	Vary	10	79.238	Vary	1	38.965	Vary	3	79.352
Item	Quantity	Unit price																																																																																																																																																																											
Book	4	24.228																																																																																																																																																																											
Book	2	37.841																																																																																																																																																																											
Book	3	36.703																																																																																																																																																																											
Book	6	37.667																																																																																																																																																																											
Book	2	68.379																																																																																																																																																																											
Book	6	72.399																																																																																																																																																																											
Book	8	26.235																																																																																																																																																																											
Book	4	60.698																																																																																																																																																																											
Book	5	73.615																																																																																																																																																																											
Book	10	27.059																																																																																																																																																																											
Book	2	4.094																																																																																																																																																																											
Book	9	29.039																																																																																																																																																																											
Book	1	71.525																																																																																																																																																																											
Book	6	45.24																																																																																																																																																																											
Book	4	46.266																																																																																																																																																																											
Book	2	32.586																																																																																																																																																																											
Book	10	27.455																																																																																																																																																																											
Book	2	46.264																																																																																																																																																																											
Book	5	46.225																																																																																																																																																																											
Book	1	81.291																																																																																																																																																																											
Book	5	30.244																																																																																																																																																																											
Book	10	62.535																																																																																																																																																																											
Book	10	75.472																																																																																																																																																																											
Book	3	63.622																																																																																																																																																																											
Book	2	17.186																																																																																																																																																																											
Book	2	46.646																																																																																																																																																																											
Book	10	70.732																																																																																																																																																																											
Book	2	37.739																																																																																																																																																																											
Book	6	44.622																																																																																																																																																																											
Book	3	66.189																																																																																																																																																																											
Book	10	51.162																																																																																																																																																																											
Book	6	3.386																																																																																																																																																																											
Book	9	28.017																																																																																																																																																																											
Book	8	20.471																																																																																																																																																																											
Book	9	23.268																																																																																																																																																																											
Book	1	27.425																																																																																																																																																																											
Book	10	45.775																																																																																																																																																																											
Book	5	30.313																																																																																																																																																																											
Book	6	4.695																																																																																																																																																																											
Book	6	36.414																																																																																																																																																																											
Book	7	61.562																																																																																																																																																																											
Book	10	77.074																																																																																																																																																																											
Book	2	1.471																																																																																																																																																																											
Book	2	80.715																																																																																																																																																																											
Book	5	30.556																																																																																																																																																																											
Book	10	40.109																																																																																																																																																																											
Book	10	67.184																																																																																																																																																																											
Book	5	14.266																																																																																																																																																																											
Book	7	56.686																																																																																																																																																																											
Book	6	34.580																																																																																																																																																																											
Item	Quantity	Unit price																																																																																																																																																																											
Vary	5	4.452																																																																																																																																																																											
Vary	2	70.534																																																																																																																																																																											
Vary	10	79.238																																																																																																																																																																											
Vary	1	38.965																																																																																																																																																																											
Vary	3	79.352																																																																																																																																																																											
 <p>HtmlReport</p>																																																																																																																																																																													

1.25 Expression

1.25.1 Complex Expression



DynamicReports

<http://www.dynamicreports.org>

ComplexExpression

Item	Price
Book	200

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.expression;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;

```

(continues on next page)

(continued from previous page)

```

28 import java.util.List;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.definition.ReportParameters;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ComplexExpressionReport {
41
42     public ComplexExpressionReport() {
43         build();
44     }
45
46     private void build() {
47         try {
48             report()
49                 .setTemplate(Templates.reportTemplate)
50                 .columns(
51                     col.column("Item", "item",
52 ↪type.stringType()),
53                     col.column("Price", new
54 ↪ComplexExpression()))
55                 .title(Templates.createTitleComponent(
56 ↪"ComplexExpression"))
57                 .pageFooter(Templates.footerComponent)
58                 .setDataSource(createDataSource())
59                 .show();
60         } catch (DRException e) {
61             e.printStackTrace();
62         }
63     }
64
65     private JRDataSource createDataSource() {
66         DRDataSource dataSource = new DRDataSource("item", "quantity",
67 ↪"unitprice");
68         dataSource.add("Book", 20, new BigDecimal(10));
69         return dataSource;
70     }
71
72     public static void main(String[] args) {
73         new ComplexExpressionReport();
74     }
75
76     private class ComplexExpression extends AbstractComplexExpression<BigDecimal>
77 ↪{
78         private static final long serialVersionUID = 1L;
79
80         public ComplexExpression() {
81             addExpression(field("quantity", Integer.class));
82             addExpression(field("unitprice", BigDecimal.class));
83         }
84     }

```

(continues on next page)

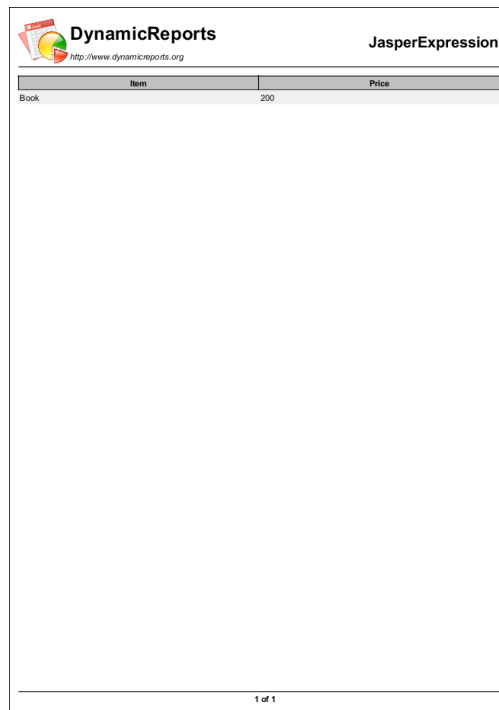
(continued from previous page)

```

80         @Override
81         public BigDecimal evaluate(List<?> values, ReportParameters_
↪reportParameters) {
82             Integer quantity = (Integer) values.get(0);
83             BigDecimal unitPrice = (BigDecimal) values.get(1);
84             return new BigDecimal(quantity).multiply(unitPrice);
85         }
86     }
87 }

```

1.25.2 Jasper Expression



DynamicReports		JasperExpression	
Item		Price	
Book		200	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.

```

(continues on next page)

(continued from previous page)

```

18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.expression;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.expression.JasperExpression;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class JasperExpressionReport {
40
41     public JasperExpressionReport() {
42         build();
43     }
44
45     private void build() {
46         try {
47             TextColumnBuilder<String> itemColumn = col.column("item",
↳type.stringType())
48                 .setTitle(exp.jasperSyntaxText("Item"));
49             JasperExpression<BigDecimal> priceExpression = exp.
↳jasperSyntax("new BigDecimal(${quantity}).multiply(${unitprice})", BigDecimal.
↳class);
50             TextColumnBuilder<BigDecimal> priceColumn = col.
↳column(priceExpression)
51                 .setTitle(exp.jasperSyntaxText("Price"));
52
53             report()
54                 .setTemplate(Templates.reportTemplate)
55                 .fields(
56                     field("quantity", Integer.
↳class),
57                     field("unitprice", BigDecimal.
↳class))
58                 .columns(itemColumn, priceColumn)
59                 .title(Templates.createTitleComponent(
↳"JasperExpression"))
60                 .pageFooter(Templates.footerComponent)
61                 .setDataSource(createDataSource())
62                 .show();
63         } catch (DRException e) {
64             e.printStackTrace();
65         }
66     }
67

```

(continues on next page)

(continued from previous page)

```

68     private JRDataSource createDataSource() {
69         DRDataSource dataSource = new DRDataSource("item", "quantity",
    ↪ "unitprice");
70         dataSource.add("Book", 20, new BigDecimal(10));
71         return dataSource;
72     }
73
74     public static void main(String[] args) {
75         new JasperExpressionReport();
76     }
77 }

```

1.25.3 Simple Expression

[illegible]

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
```

(continues on next page)

(continued from previous page)

```

16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.expression;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.definition.ReportParameters;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class SimpleExpressionReport {
40
41     public SimpleExpressionReport() {
42         build();
43     }
44
45     private void build() {
46         try {
47             report()
48                 .setTemplate(Templates.reportTemplate)
49                 .fields(
50                     field("quantity", Integer.
51 ↪class),
52                     field("unitprice", BigDecimal.
53 ↪class))
54                 .columns(
55                     col.column("Item", "item",
56 ↪type.stringType()),
57                     col.column("Price", new
58 ↪SimpleExpression()))
59                 .title(Templates.createTitleComponent(
60 ↪"SimpleExpression"))
61                 .pageFooter(Templates.footerComponent)
62                 .setDataSource(createDataSource())
63                 .show();
64         } catch (DRException e) {
65             e.printStackTrace();
66         }
67     }
68
69     private JRDataSource createDataSource() {
70         DRDataSource dataSource = new DRDataSource("item", "quantity",
71 ↪"unitprice");
72         dataSource.add("Book", 20, new BigDecimal(10));

```

(continues on next page)


(continued from previous page)

```

67         return dataSource;
68     }
69
70     public static void main(String[] args) {
71         new SimpleExpressionReport();
72     }
73
74     private class SimpleExpression extends AbstractSimpleExpression<BigDecimal> {
75         private static final long serialVersionUID = 1L;
76
77         @Override
78         public BigDecimal evaluate(ReportParameters reportParameters) {
79             Integer quantity = reportParameters.getValue("quantity");
80             BigDecimal unitPrice = reportParameters.getValue("unitprice");
81             return new BigDecimal(quantity).multiply(unitPrice);
82         }
83     }
84 }

```

1.25.4 Value Formatter



DynamicReports

<http://www.dynamicreports.org>

ValueFormatter

Item	Quantity	Unit price
Book	20	10.00 EUR

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *

```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.expression;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractValueFormatter;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.definition.ReportParameters;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class ValueFormatterReport {
40
41     public ValueFormatterReport() {
42         build();
43     }
44
45     private void build() {
46         try {
47             report()
48                 .setTemplate(Templates.reportTemplate)
49                 .fields(
50                     field("unitprice", BigDecimal.
↳class))
51                 .columns(
52                     col.column("Item", "item",
↳type.stringType()),
53                     col.column("Quantity",
↳"quantity", type.integerType()),
54                     col.column("Unit price",
↳"unitprice", type.bigDecimalType()).setValueFormatter(new ValueFormatter())
55                 .title(Templates.createTitleComponent(
↳"ValueFormatter"))
56                 .pageFooter(Templates.footerComponent)
57                 .setDataSource(createDataSource())
58                 .show();
59         } catch (DRException e) {
60             e.printStackTrace();

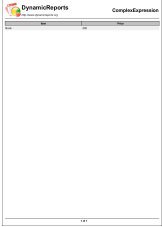
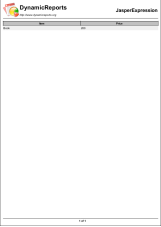
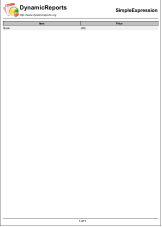
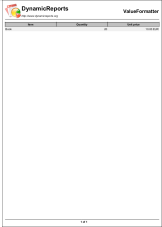
```

(continues on next page)

(continued from previous page)

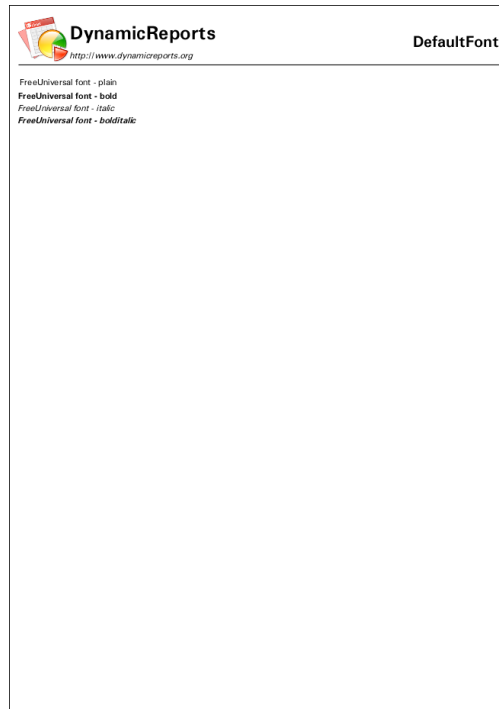
```
61         }
62     }
63
64     private JRDataSource createDataSource() {
65         DRDataSource dataSource = new DRDataSource("item", "quantity",
66 ↪ "unitprice");
67         dataSource.add("Book", 20, new BigDecimal(10));
68         return dataSource;
69     }
70
71     public static void main(String[] args) {
72         new ValueFormatterReport();
73     }
74
75     private static class ValueFormatter extends AbstractValueFormatter<String, ↪
76 ↪ Number> {
77         private static final long serialVersionUID = 1L;
78
79         @Override
80         public String format(Number value, ReportParameters reportParameters)
81 ↪ {
82             return type.bigDecimalType().valueToString(value, ↪
83 ↪ reportParameters.getLocale()) + " EUR";
84         }
85     }
86 }
```

Table 10: Expression Examples

		
ComplexExpressionReport	JasperExpressionReport	SimpleExpressionReport
		
ValueFormatterReport		

1.26 Font

1.26.1 Default Font



```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.fonts;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.style.FontBuilder;
```

(continues on next page)

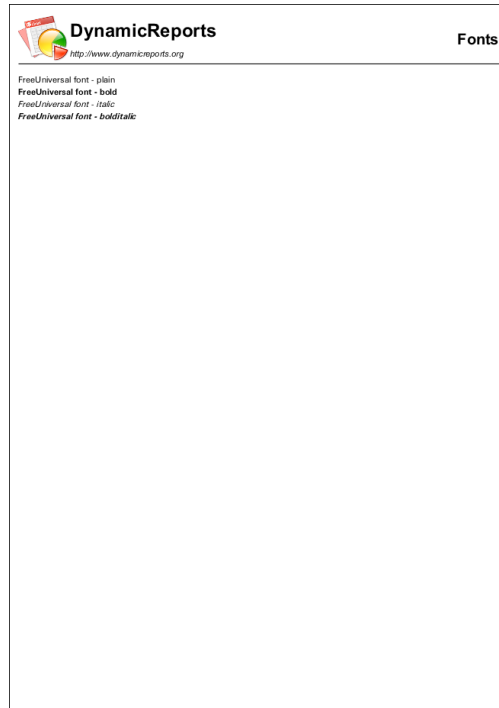
(continued from previous page)

```

28 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
29 import net.sf.dynamicreports.report.exception.DRException;
30
31 /**
32  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
33  *
34  * This report is working properly only when the font "FreeUniversal" is
35  * registered.
36  * The font is registered in these files: customfonts.xml and jasperreports_
37  * extension.properties. The files are located in the root of the classpath.
38  */
39 public class DefaultFontReport {
40
41     public DefaultFontReport() {
42         build();
43     }
44
45     private void build() {
46         FontBuilder defaultFont = stl.font()
47             .setFontName("FreeUniversal");
48         StyleBuilder boldStyle = stl.style()
49             .bold();
50         StyleBuilder italicStyle = stl.style()
51             .italic();
52         StyleBuilder boldItalicStyle = stl.style()
53             .boldItalic();
54
55         try {
56             report()
57                 .setDefaultFont(defaultFont)
58                 .title(
59                     Templates.
60                     createTitleComponent("DefaultFont"),
61                     cmp.text("FreeUniversal font -
62                     plain"),
63                     cmp.text("FreeUniversal font -
64                     bold").setStyle(boldStyle),
65                     cmp.text("FreeUniversal font -
66                     italic").setStyle(italicStyle),
67                     cmp.text("FreeUniversal font -
68                     bolditalic").setStyle(boldItalicStyle))
69                     .show();
70         } catch (DRException e) {
71             e.printStackTrace();
72         }
73     }
74
75     public static void main(String[] args) {
76         new DefaultFontReport();
77     }
78 }

```

1.26.2 Fonts



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.fonts;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.Templates;
27  import net.sf.dynamicreports.report.builder.style.StyleBuilder;
28  import net.sf.dynamicreports.report.exception.DRException;
29
30  /**

```

(continues on next page)



(continued from previous page)

```

31  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
32  *
33  *      This report is working properly only when the font "FreeUniversal" is
↪registered.
34  *      The font is registered in these files: customfonts.xml and jasperreports_
↪extension.properties. The files are located in the root of the classpath.
35  */
36  public class FontsReport {
37
38      public FontsReport() {
39          build();
40      }
41
42      private void build() {
43          StyleBuilder plainStyle = stl.style()
44              .setFontName("FreeUniversal");
45          StyleBuilder boldStyle = stl.style(plainStyle)
46              .bold();
47          StyleBuilder italicStyle = stl.style(plainStyle)
48              .italic();
49          StyleBuilder boldItalicStyle = stl.style(plainStyle)
50              .boldItalic();
51
52          try {
53              report()
54                  .title(
55                      Templates.
56                      cmp.text("FreeUniversal font -
↪createTitleComponent("Fonts"),
57                      cmp.text("FreeUniversal font -
↪ plain").setStyle(plainStyle),
58                      cmp.text("FreeUniversal font -
↪ bold").setStyle(boldStyle),
59                      cmp.text("FreeUniversal font -
↪ italic").setStyle(italicStyle),
60                      cmp.text("FreeUniversal font -
↪ bolditalic").setStyle(boldItalicStyle))
61                      .show();
62          } catch (DRException e) {
63              e.printStackTrace();
64          }
65
66      public static void main(String[] args) {
67          new FontsReport();
68      }
69  }

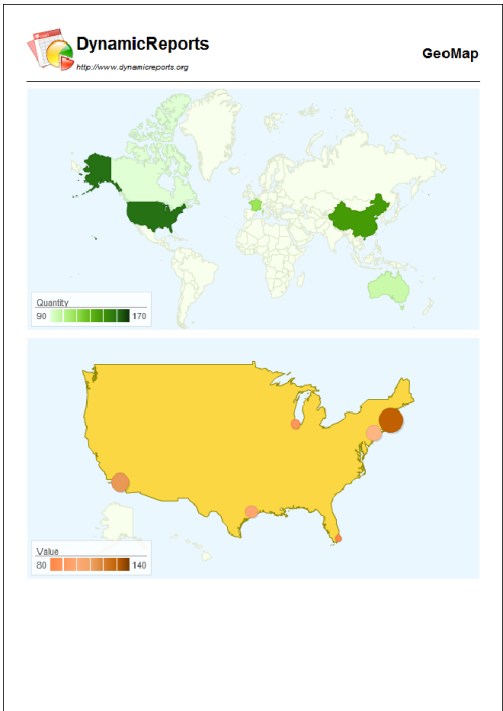
```

Table 11: Font Examples

	
DefaultFontReport	FontsReport

1.27 GeoMap

1.27.1 Geo Map



```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
```

(continues on next page)

(continued from previous page)

```

14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.googlechart;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.awt.Color;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.googlecharts.report.GoogleCharts;
31  import net.sf.dynamicreports.googlecharts.report.geomap.GeoMapBuilder;
32  import net.sf.dynamicreports.googlecharts.report.geomap.GeoMapDataMode;
33  import net.sf.dynamicreports.jasper.builder.export.JasperHtmlExporterBuilder;
34  import net.sf.dynamicreports.report.datasource.DRDataSource;
35  import net.sf.dynamicreports.report.exception.DRException;
36  import net.sf.jasperreports.engine.JRDataSource;
37
38  /**
39   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40   *
41   *      GeoMap component works only for html export.
42   *      Visit the following link for more information about geo map parameters
43   *      http://code.google.com/apis/chart/interactive/docs/gallery/geomap.html
44   */
45  public class GeoMapReport {
46
47      public GeoMapReport() {
48          build();
49      }
50
51      private void build() {
52          GeoMapBuilder geoMap1 = GoogleCharts.geoMap()
53              .setDataSource(createDataSource1())
54              .setLocation("location", String.class)
55              .setValue("quantity", Integer.class)
56              .setLabel("label", String.class)
57              .setValueLabel("Quantity")
58              .setFixedHeight(300);
59
60          GeoMapBuilder geoMap2 = GoogleCharts.geoMap()
61              .setDataSource(createDataSource2())
62              .setDataMode(GeoMapDataMode.MARKERS)
63              .setRegion("US")
64              .colors(Color.decode("#FF8747"), Color.decode("#FFB581
65  ↪"), Color.decode("#C06000"))
66              .setLocation("location", String.class)
67              .setValue("quantity", Integer.class)
68              .setFixedHeight(300);
69
70      try {

```

(continues on next page)

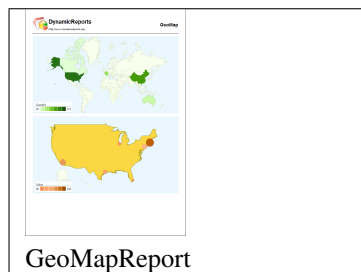
(continued from previous page)

```

70         JasperHtmlExporterBuilder htmlExporter = export.htmlExporter(
71             ↪ "c:/report.html")
72             .setImagesDirName("c:/images")
73             .setOutputImagesToDir(true);
74
75         report()
76             .setTemplate(Templates.reportTemplate)
77             .title(Templates.createTitleComponent("GeoMap
78             ↪"))
79             .summary(
80                 ↪ geoMap1, cmp.verticalGap(10), ↪
81                 ↪ geoMap2)
82             .toHtml(htmlExporter);
83     } catch (DRException e) {
84         e.printStackTrace();
85     }
86 }
87
88 private JRDataSource createDataSource1() {
89     DRDataSource dataSource = new DRDataSource("location", "quantity",
90     ↪ "label");
91     dataSource.add("US", 170, "United States");
92     dataSource.add("CA", 90, "Canada");
93     dataSource.add("FR", 120, "France");
94     dataSource.add("AU", 100, "Australia");
95     dataSource.add("CN", 150, "China");
96     return dataSource;
97 }
98
99 private JRDataSource createDataSource2() {
100     DRDataSource dataSource = new DRDataSource("location", "quantity");
101     dataSource.add("New York", 110);
102     dataSource.add("Boston", 140);
103     dataSource.add("Miami", 80);
104     dataSource.add("Chicago", 90);
105     dataSource.add("Los Angeles", 120);
106     dataSource.add("Houston", 100);
107     return dataSource;
108 }
109
110 public static void main(String[] args) {
111     new GeoMapReport();
112 }


```

Table 12: GeoMap Examples



1.28 Group

1.28.1 Column Group

DynamicReports				Group
 http://www.dynamicreports.org				
Item	Tablet			
Order date	Quantity	Unit price		
01/01/2010	5	300.00		
01/03/2010	1	280.00		
01/19/2010	5	320.00		
Item	Laptop			
Order date	Quantity	Unit price		
01/05/2010	3	580.00		
01/08/2010	1	620.00		
01/15/2010	5	600.00		
Item	Smartphone			
Order date	Quantity	Unit price		
01/18/2010	8	150.00		
01/20/2010	8	210.00		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;

```

(continues on next page)

(continued from previous page)

```

28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class ColumnGroupReport {
43
44     public ColumnGroupReport() {
45         build();
46     }
47
48     private void build() {
49         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
50
51         ColumnGroupBuilder itemGroup = grp.group(itemColumn)
52             .setTitleWidth(30)
53             .setHeaderLayout(GroupHeaderLayout.TITLE_AND_VALUE)
54             .showColumnHeaderAndFooter();
55
56         try {
57             report()
58                 .setTemplate(Templates.reportTemplate)
59                 .setShowColumnTitle(false)
60                 .columns(
61                     itemColumn,
62                     col.column("Order date",
↳ "orderdate", type.dateType()),
63                     col.column("Quantity",
↳ "quantity", type.integerType()),
64                     col.column("Unit price",
↳ "unitprice", type.bigDecimalType()))
65                 .groupBy(itemGroup)
66                 .title(Templates.createTitleComponent("Group
↳ "))
67                 .pageFooter(Templates.footerComponent)
68                 .setDataSource(createDataSource())
69                 .show();
70         } catch (DRException e) {
71             e.printStackTrace();
72         }
73     }
74
75     private JRDataSource createDataSource() {
76         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↳ "quantity", "unitprice");
77         dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
78         dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));


```

(continues on next page)

(continued from previous page)

```
79         dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
80         dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
81         dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
82         dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
83         dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new
↪BigDecimal(150));
84         dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new
↪BigDecimal(210));
85         return dataSource;
86     }
87
88     private Date toDate(int year, int month, int day) {
89         Calendar c = Calendar.getInstance();
90         c.set(Calendar.YEAR, year);
91         c.set(Calendar.MONTH, month - 1);
92         c.set(Calendar.DAY_OF_MONTH, day);
93         return c.getTime();
94     }
95
96     public static void main(String[] args) {
97         new ColumnGroupReport();
98     }
99 }
```

1.28.2 Custom Group

**DynamicReports**
<http://www.dynamicreports.org>

CustomGroup

Order date	Item	Quantity	Unit price
2009			
2009-11-01	Tablet	5	250.00
2009-11-01	Laptop	3	480.00
2009-12-01	Smartphone	1	280.00
2009-12-01	Tablet	1	190.00
2010			
2010-01-01	Tablet	4	230.00
2010-01-01	Laptop	2	650.00
2010-02-01	Laptop	3	550.00
2010-02-01	Smartphone	5	210.00

1 of 1

```
1 /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically
3  *
```

(continues on next page)

(continued from previous page)

```

4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31 import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.definition.ReportParameters;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class CustomGroupReport {
41
42     public CustomGroupReport() {
43         build();
44     }
45
46     private void build() {
47         CustomGroupBuilder yearGroup = grp.group(new YearExpression());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(
53                     col.column("Order date",
54 ↪ "orderdate", type.stringType()),
55                     col.column("Item", "item", ↪
56 ↪ type.stringType()),
57                     col.column("Quantity",
58 ↪ "quantity", type.integerType()),
59                     col.column("Unit price",
60 ↪ "unitprice", type.bigDecimalType()))

```

(continues on next page)

(continued from previous page)

```

57         .groupBy(yearGroup)
58         .title(Templates.createTitleComponent(
↪ "CustomGroup"))
59         .pageFooter(Templates.footerComponent)
60         .setDataSource(createDataSource())
61         .show();
62     } catch (DRException e) {
63         e.printStackTrace();
64     }
65 }
66
67 private class YearExpression extends AbstractSimpleExpression<String> {
68     private static final long serialVersionUID = 1L;
69
70     @Override
71     public String evaluate(ReportParameters reportParameters) {
72         String orderDate = reportParameters.getValue("orderdate");
73         return orderDate.split("-")[0];
74     }
75 }
76
77 private JRDataSource createDataSource() {
78     DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪ "quantity", "unitprice");
79     dataSource.add("2009-11-01", "Tablet", 5, new BigDecimal(250));
80     dataSource.add("2009-11-01", "Laptop", 3, new BigDecimal(480));
81     dataSource.add("2009-12-01", "Smartphone", 1, new BigDecimal(280));
82     dataSource.add("2009-12-01", "Tablet", 1, new BigDecimal(190));
83     dataSource.add("2010-01-01", "Tablet", 4, new BigDecimal(230));
84     dataSource.add("2010-01-01", "Laptop", 2, new BigDecimal(650));
85     dataSource.add("2010-02-01", "Laptop", 3, new BigDecimal(550));
86     dataSource.add("2010-02-01", "Smartphone", 5, new BigDecimal(210));
87     return dataSource;
88 }
89
90 public static void main(String[] args) {
91     new CustomGroupReport();
92 }
93 }

```

1.28.3 Date Group

DynamicReports		DateGroup	
		http://www.dynamicreports.org	
Order date	Item	Quantity	Unit price
2009			
November			
11/01/2009	Tablet	5	250.00
11/01/2009	Laptop	3	480.00
December			
12/01/2009	Smartphone	1	280.00
12/01/2009	Tablet	1	190.00
2010			
January			
01/01/2010	Tablet	4	230.00
01/01/2010	Laptop	2	650.00
February			
02/01/2010	Laptop	3	550.00
02/01/2010	Smartphone	5	210.00

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class DateGroupReport {
42
43     public DateGroupReport() {
44         build();
45     }
46
47     private void build() {
48         TextColumnBuilder<Date> yearColumn = col.column("Order year",
49 ↪ "orderdate", type.dateYearType());
50         TextColumnBuilder<Date> monthColumn = col.column("Order month",
51 ↪ "orderdate", type.dateMonthType());
52
53         ColumnGroupBuilder yearGroup = grp.group(yearColumn)
54             .groupByDataType();
55         ColumnGroupBuilder monthGroup = grp.group(monthColumn)
56             .groupByDataType();
57
58         try {
59             report()
60                 .setTemplate(Templates.reportTemplate)
61                 .columns(
62                     yearColumn,
63                     monthColumn,
64                     col.column("Order date",
65 ↪ "orderdate", type.dateType()),
66                     col.column("Item", "item",
67 ↪ type.stringType()),
68                     col.column("Quantity",
69 ↪ "quantity", type.integerType()),
70                     col.column("Unit price",
71 ↪ "unitprice", type.bigDecimalType()))
72                 .groupBy(
73                     yearGroup, monthGroup)
74                 .title(Templates.createTitleComponent(
75 ↪ "DateGroup"))
76                 .pageFooter(Templates.footerComponent)
77                 .setDataSource(createDataSource())
78                 .show();
79         } catch (DRException e) {
80             e.printStackTrace();
81         }
82
83     private JRDataSource createDataSource() {
84         DRDataSource dataSource = new DRDataSource("orderdate", "item",
85 ↪ "quantity", "unitprice");
86         dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
87     }
88 }

```

(continues on next page)


(continued from previous page)

```

80         dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
81         dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new
↳BigDecimal(280));
82         dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
83         dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
84         dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
85         dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
86         dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new
↳BigDecimal(210));
87         return dataSource;
88     }
89
90     private Date toDate(int year, int month, int day) {
91         Calendar c = Calendar.getInstance();
92         c.set(Calendar.YEAR, year);
93         c.set(Calendar.MONTH, month - 1);
94         c.set(Calendar.DAY_OF_MONTH, day);
95         return c.getTime();
96     }
97
98     public static void main(String[] args) {
99         new DateGroupReport();
100     }
101 }

```

1.28.4 Field Group

DynamicReports		FieldGroup	
 http://www.dynamicreports.org			
	Order date	Quantity	Unit price
Item Tablet	01/01/2010	5	300.00
	01/03/2010	1	280.00
	01/19/2010	5	320.00
Item Laptop	01/05/2010	3	580.00
	01/08/2010	1	620.00
	01/15/2010	5	600.00
Item Smartphone	01/18/2010	8	150.00
	01/20/2010	8	210.00

1 of 1

```

1  /**
2  * DynamicReports - Free Java reporting library for creating reports dynamically

```

(continues on next page)

(continued from previous page)

```

3  *
4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
33 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
34 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class FieldGroupReport {
43
44     public FieldGroupReport() {
45         build();
46     }
47
48     private void build() {
49         StyleBuilder groupStyle = stl.style().bold();
50
51         CustomGroupBuilder itemGroup = grp.group("item", String.class)
52             .setStyle(groupStyle)
53             .setTitle("Item")
54             .setTitleStyle(groupStyle)
55             .setTitleWidth(30)
56             .setHeaderLayout(GroupHeaderLayout.TITLE_AND_VALUE);
57
58         try {
59             report()

```

(continues on next page)

(continued from previous page)

```

60         .setTemplate(Templates.reportTemplate)
61         .columns(
62             col.column("Order date",
↪ "orderdate", type.dateType()),
63             col.column("Quantity",
↪ "quantity", type.integerType()),
64             col.column("Unit price",
↪ "unitprice", type.bigDecimalType()))
65         .groupBy(itemGroup)
66         .title(Templates.createTitleComponent(
↪ "FieldGroup"))
67         .pageFooter(Templates.footerComponent)
68         .setDataSource(createDataSource())
69         .show();
70     } catch (DRException e) {
71         e.printStackTrace();
72     }
73 }
74
75 private JRDataSource createDataSource() {
76     DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪ "quantity", "unitprice");
77     dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
78     dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));
79     dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
80     dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
81     dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
82     dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
83     dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new_
↪ BigDecimal(150));
84     dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new_
↪ BigDecimal(210));
85     return dataSource;
86 }
87
88 private Date toDate(int year, int month, int day) {
89     Calendar c = Calendar.getInstance();
90     c.set(Calendar.YEAR, year);
91     c.set(Calendar.MONTH, month - 1);
92     c.set(Calendar.DAY_OF_MONTH, day);
93     return c.getTime();
94 }
95
96 public static void main(String[] args) {
97     new FieldGroupReport();
98 }
99 }

```

1.28.5 Group Header With Subtotal

DynamicReports		GroupHeaderWithSubtotal		
		http://www.dynamicreports.org		
Order year	Order date	Item	Quantity	Unit price
2009			10	1,200.00
	11/01/2009	Tablet	5	250.00
	11/01/2009	Laptop	3	480.00
	12/01/2009	Smartphone	1	280.00
	12/01/2009	Tablet	1	190.00
2010			14	1,640.00
	01/01/2010	Tablet	4	230.00
	01/01/2010	Laptop	2	650.00
	02/01/2010	Laptop	3	550.00
	02/01/2010	Smartphone	5	210.00
Total			24	2,840.00

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
35 import net.sf.dynamicreports.report.constant.Calculation;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.definition.ReportParameters;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40
41 /**
42  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
43  */
44 public class GroupHeaderWithSubtotalReport {
45
46     public GroupHeaderWithSubtotalReport() {
47         build();
48     }
49
50     private void build() {
51         TextColumnBuilder<String> yearColumn = col.column("Order year", exp.
↪text(""));
52         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateType());
53         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
54         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
55         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
56
57         CustomGroupBuilder yearGroup = grp.group(new YearExpression())
58             .groupByDataType()
59             .headerWithSubtotal();
60
61         try {
62             report()
63                 .setTemplate(Templates.reportTemplate)
64                 .setSubtotalStyle(stl.style(Templates.
↪boldStyle))
65                 .fields(
66                     field("orderdate", type.
↪dateYearType()))
67                 .columns(
68                     yearColumn, orderDateColumn,
↪itemColumn, quantityColumn, unitPriceColumn)
69                 .groupBy(yearGroup)
70                 .subtotalsAtGroupHeader(yearGroup,
71                     sbt.sum(quantityColumn), sbt.
↪sum(unitPriceColumn))
72                 .subtotalsAtSummary(
73                     sbt.aggregate(exp.text("Total
↪"), yearColumn, Calculation.NOTHING), sbt.sum(quantityColumn), sbt.
↪sum(unitPriceColumn))
74                 .title(Templates.createTitleComponent(
↪"GroupHeaderWithSubtotal"))
75                 .pageFooter(Templates.footerComponent)

```

(continues on next page)

(continued from previous page)

```

76         .setDataSource(createDataSource())
77         .show();
78     } catch (DRException e) {
79         e.printStackTrace();
80     }
81 }
82
83 private class YearExpression extends AbstractSimpleExpression<String> {
84     private static final long serialVersionUID = 1L;
85
86     @Override
87     public String evaluate(ReportParameters reportParameters) {
88         return type.dateYearType().valueToString("orderdate",
↪reportParameters);
89     }
90 }
91
92 private JRDataSource createDataSource() {
93     DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪"quantity", "unitprice");
94     dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
95     dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
96     dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new
↪BigDecimal(280));
97     dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
98     dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
99     dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
100    dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
101    dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new
↪BigDecimal(210));
102    return dataSource;
103 }
104
105 private Date toDate(int year, int month, int day) {
106     Calendar c = Calendar.getInstance();
107     c.set(Calendar.YEAR, year);
108     c.set(Calendar.MONTH, month - 1);
109     c.set(Calendar.DAY_OF_MONTH, day);
110     return c.getTime();
111 }
112
113 public static void main(String[] args) {
114     new GroupHeaderWithSubtotalReport();
115 }
116 }

```

1.28.6 Group In New Page



```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.group;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.Calendar;
29  import java.util.Date;
30
31  import net.sf.dynamicreports.examples.Templates;
32  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33  import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34  import net.sf.dynamicreports.report.datasource.DRDataSource;
35  import net.sf.dynamicreports.report.exception.DRException;
36  import net.sf.jasperreports.engine.JRDataSource;
37
38  /**
39   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40   */
41  public class GroupInNewPageReport {
42
43      public GroupInNewPageReport() {
44          build();
45      }
46  }
```

(continues on next page)

(continued from previous page)

```

47     private void build() {
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType())
49             .setWidth(30);
50
51         ColumnGroupBuilder itemGroup = grp.group(itemColumn)
52             .startInNewPage();
53
54         try {
55             report()
56                 .setTemplate(Templates.reportTemplate)
57                 .columns(
58                     itemColumn,
59                     col.column("Order date",
↳ "orderdate", type.dateType()),
60                     col.column("Quantity",
↳ "quantity", type.integerType()),
61                     col.column("Unit price",
↳ "unitprice", type.bigDecimalType()))
62                 .groupBy(itemGroup)
63                 .title(Templates.createTitleComponent(
↳ "GroupInNewPage"))
64                 .pageFooter(Templates.footerComponent)
65                 .setDataSource(createDataSource())
66                 .show();
67         } catch (DRException e) {
68             e.printStackTrace();
69         }
70     }
71
72     private JRDataSource createDataSource() {
73         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↳ "quantity", "unitprice");
74         dataSource.add("Tablet", toDate(2010, 1, 1), 5, new BigDecimal(300));
75         dataSource.add("Tablet", toDate(2010, 1, 3), 1, new BigDecimal(280));
76         dataSource.add("Tablet", toDate(2010, 1, 19), 5, new BigDecimal(320));
77         dataSource.add("Laptop", toDate(2010, 1, 5), 3, new BigDecimal(580));
78         dataSource.add("Laptop", toDate(2010, 1, 8), 1, new BigDecimal(620));
79         dataSource.add("Laptop", toDate(2010, 1, 15), 5, new BigDecimal(600));
80         dataSource.add("Smartphone", toDate(2010, 1, 18), 8, new
↳ BigDecimal(150));
81         dataSource.add("Smartphone", toDate(2010, 1, 20), 8, new
↳ BigDecimal(310));
82         return dataSource;
83     }
84
85     private Date toDate(int year, int month, int day) {
86         Calendar c = Calendar.getInstance();
87         c.set(Calendar.YEAR, year);
88         c.set(Calendar.MONTH, month - 1);
89         c.set(Calendar.DAY_OF_MONTH, day);
90         return c.getTime();
91     }
92
93     public static void main(String[] args) {
94         new GroupInNewPageReport();
95     }

```

(continues on next page)

(continued from previous page)

96 }

1.28.7 Group Layout

DynamicReports		GroupLayout		
http://www.dynamicreports.org				
Order year	Order date	Item	Quantity	Unit price
2009			10	1,200.00
	11/01/2009	Tablet	5	250.00
	11/01/2009	Laptop	3	480.00
	12/01/2009	Smartphone	1	280.00
	12/01/2009	Tablet	1	190.00
2010			14	1,640.00
	01/01/2010	Tablet	4	230.00
	01/01/2010	Laptop	2	650.00
	02/01/2010	Laptop	3	550.00
	02/01/2010	Smartphone	5	210.00
Total			24	2,840.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;

```

(continues on next page)

(continued from previous page)

```

26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
35 import net.sf.dynamicreports.report.constant.Calculation;
36 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class GroupLayoutReport {
46
47     public GroupLayoutReport() {
48         build();
49     }
50
51     private void build() {
52         TextColumnBuilder<String> yearColumn = col.column("Order year", exp.
↳text(""));
53         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↳"orderdate", type.dateType());
54         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳type.stringType());
55         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳"quantity", type.integerType());
56         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳", "unitprice", type.bigDecimalType());
57
58         CustomGroupBuilder yearGroup = grp.group(new YearExpression())
59             .groupByDataType()
60             .setHeaderLayout(GroupHeaderLayout.EMPTY)
61             .setPadding(0);
62
63         try {
64             report()
65                 .setTemplate(Templates.reportTemplate)
66                 .setSubtotalStyle(stl.style(Templates.
↳boldStyle))
67                 .fields(
68                     field("orderdate", type.
↳dateYearType()))
69                 .columns(
70                     yearColumn, orderDateColumn,
↳itemColumn, quantityColumn, unitPriceColumn)
71                 .groupBy(yearGroup)
72                 .subtotalsAtGroupHeader(yearGroup,
73                     sbt.first(new
↳YearExpression(), yearColumn), sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))

```

(continues on next page)


(continued from previous page)

```

74         .subtotalsAtSummary (
75             sbt.aggregate(exp.text("Total
↪"), yearColumn, Calculation.NOTHING), sbt.sum(quantityColumn), sbt.
↪sum(unitPriceColumn))
76         .title(Templates.createTitleComponent (
↪"GroupLayout"))
77         .pageFooter(Templates.footerComponent)
78         .setDataSource(createDataSource())
79         .show();
80     } catch (DRException e) {
81         e.printStackTrace();
82     }
83 }
84
85 private class YearExpression extends AbstractSimpleExpression<String> {
86     private static final long serialVersionUID = 1L;
87
88     @Override
89     public String evaluate(ReportParameters reportParameters) {
90         ↪return type.dateYearType().valueToString("orderdate", ↪
↪reportParameters);
91     }
92 }
93
94 private JRDataSource createDataSource() {
95     DRDataSource dataSource = new DRDataSource("orderdate", "item",
↪"quantity", "unitprice");
96     dataSource.add(toDate(2009, 11, 1), "Tablet", 5, new BigDecimal(250));
97     dataSource.add(toDate(2009, 11, 1), "Laptop", 3, new BigDecimal(480));
98     dataSource.add(toDate(2009, 12, 1), "Smartphone", 1, new ↪
↪BigDecimal(280));
99     dataSource.add(toDate(2009, 12, 1), "Tablet", 1, new BigDecimal(190));
100    dataSource.add(toDate(2010, 1, 1), "Tablet", 4, new BigDecimal(230));
101    dataSource.add(toDate(2010, 1, 1), "Laptop", 2, new BigDecimal(650));
102    dataSource.add(toDate(2010, 2, 1), "Laptop", 3, new BigDecimal(550));
103    dataSource.add(toDate(2010, 2, 1), "Smartphone", 5, new ↪
↪BigDecimal(210));
104    return dataSource;
105 }
106
107 private Date toDate(int year, int month, int day) {
108     Calendar c = Calendar.getInstance();
109     c.set(Calendar.YEAR, year);
110     c.set(Calendar.MONTH, month - 1);
111     c.set(Calendar.DAY_OF_MONTH, day);
112     return c.getTime();
113 }
114
115 public static void main(String[] args) {
116     new GroupLayoutReport();
117 }
118 }

```

1.28.8 Group Subtotals

DynamicReports		GroupSubtotals	
 http://www.dynamicreports.org			
	Order month	Quantity	Unit price
2014			
	January	63	599.513
	February	151	1.889.478
	March	203	1.536.809
	April	170	1.302.148
	May	202	1.858.564
	June	157	1.225.739
	July	160	1.548.704
	August	56	529.743
1 of 1			

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.group;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class GroupSubtotalsReport {
43
44     public GroupSubtotalsReport() {
45         build();
46     }
47
48     private void build() {
49         TextColumnBuilder<Date> yearColumn = col.column("Order year",
50 ↪ "orderdate", type.dateYearType());
51         TextColumnBuilder<Date> monthColumn = col.column("Order month",
52 ↪ "orderdate", type.dateMonthType());
53         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
54 ↪ "quantity", type.integerType());
55         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
56 ↪ ", "unitprice", type.bigDecimalType());
57
58         ColumnGroupBuilder yearGroup = grp.group(yearColumn)
59             .groupByDataType();
60         ColumnGroupBuilder monthGroup = grp.group(monthColumn)
61             .groupByDataType()
62             .setHeaderLayout(GroupHeaderLayout.EMPTY)
63             .setHideColumn(false);
64
65         try {
66             report()
67                 .setTemplate(Templates.reportTemplate)
68                 .setSubtotalStyle(Templates.columnStyle)
69                 .setShowColumnValues(false)
70                 .columns(yearColumn, monthColumn,
71 ↪ quantityColumn, unitPriceColumn)
72                 .groupBy(yearGroup, monthGroup)
73                 .subtotalsAtGroupFooter(monthGroup, sbt.
74 ↪ first(monthColumn), sbt.sum(quantityColumn), sbt.sum(unitPriceColumn))
75                 .title(Templates.createTitleComponent(
76 ↪ "GroupSubtotals"))
77                 .pageFooter(Templates.footerComponent)
78                 .setDataSource(createDataSource())
79                 .show();
80         } catch (DRException e) {
81             e.printStackTrace();
82         }
83     }
84
85     private JRDataSource createDataSource() {
86         DRDataSource dataSource = new DRDataSource("orderdate", "quantity",
87 ↪ "unitprice");

```

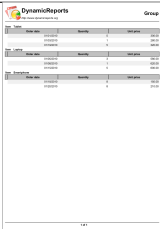
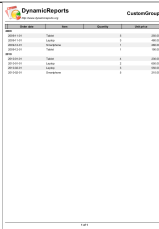
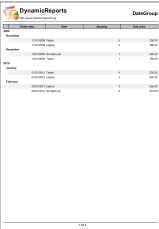
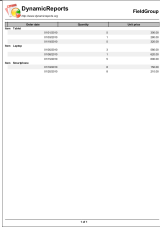
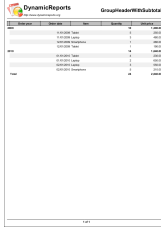

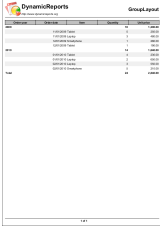
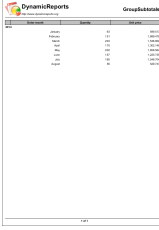
(continues on next page)

(continued from previous page)

```
Calendar c = Calendar.getInstance();
c.add(Calendar.YEAR, -1);
for (int i = 0; i < 200; i++) {
    Date date = c.getTime();
    dataSource.add(date, (int) (Math.random() * 10) + 1, new
↳BigDecimal(Math.random() * 100 + 1));
    c.add(Calendar.DAY_OF_MONTH, 1);
}
return dataSource;
}


public static void main(String[] args) {
    new GroupSubtotalsReport();
}
}
```

Table 13: Group Examples

		
ColumnGroupReport	CustomGroupReport	DateGroupReport
		
FieldGroupReport	GroupHeaderWithSubtotalReport	GroupInNewPageReport
		
GroupLayoutReport	GroupSubtotalsReport	











1.29 Miscellaneous

1.29.1 Card



DynamicReports

<http://www.dynamicreports.org>

		Card
	Name: Peter Marsh Address: 23 Baden Av. City: New York	 Name: Peter Marsh Address: 23 Baden Av. City: New York
	Name: Peter Marsh Address: 23 Baden Av. City: New York	 Name: Peter Marsh Address: 23 Baden Av. City: New York
	Name: Peter Marsh Address: 23 Baden Av. City: New York	 Name: Peter Marsh Address: 23 Baden Av. City: New York
	Name: Peter Marsh Address: 23 Baden Av. City: New York	 Name: Peter Marsh Address: 23 Baden Av. City: New York
	Name: Peter Marsh Address: 23 Baden Av. City: New York	 Name: Peter Marsh Address: 23 Baden Av. City: New York

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.report.builder.component.ComponentBuilder;

```

(continues on next page)

(continued from previous page)

```

28 import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
29 import net.sf.dynamicreports.report.builder.component.ImageBuilder;
30 import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
31 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
32 import net.sf.dynamicreports.report.constant.PageType;
33 import net.sf.dynamicreports.report.exception.DRException;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class CardReport {
39
40     public CardReport() {
41         build();
42     }
43
44     private void build() {
45         ComponentBuilder<?, ?> cardComponent = createCardComponent();
46         HorizontalListBuilder cards = cmp.horizontalFlowList();
47         for (int i = 0; i < 10; i++) {
48             cards.add(cardComponent);
49         }
50
51         try {
52             report()
53                 .setTemplate(Templates.reportTemplate)
54                 .setTextStyle(stl.style())
55                 .setPageFormat(PageType.A5)
56                 .title(
57                     Templates.
58                     createTitleComponent("Card"),
59                     cards)
60                 .show();
61         } catch (DRException e) {
62             e.printStackTrace();
63         }
64
65         private ComponentBuilder<?, ?> createCardComponent() {
66             HorizontalListBuilder cardComponent = cmp.horizontalList();
67             StyleBuilder cardStyle = stl.style(stl.pen1Point())
68                 .setPadding(10);
69             cardComponent.setStyle(cardStyle);
70
71             ImageBuilder image = cmp.image(Templates.class.getResource("images/
72             ↪user_male.png")).setFixedDimension(60, 60);
73             cardComponent.add(cmp.hListCell(image).heightFixedOnMiddle());
74             cardComponent.add(cmp.horizontalGap(10));
75
76             StyleBuilder boldStyle = stl.style().bold();
77             VerticalListBuilder content = cmp.verticalList(
78                 cmp.text("Name:").setStyle(boldStyle),
79                 cmp.text("Peter Marsh"),
80                 cmp.text("Address:").setStyle(boldStyle),
81                 cmp.text("23 Baden Av."),
82                 cmp.text("City:").setStyle(boldStyle),
83                 cmp.text("New York"));

```

(continues on next page)

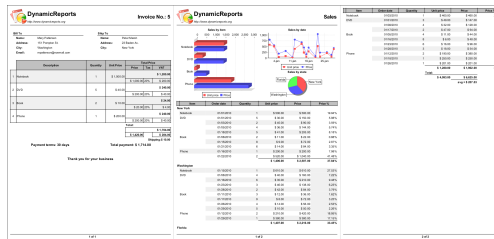
(continued from previous page)

```

83         cardComponent.add(content);
84         return cardComponent;
85     }
86
87
88     public static void main(String[] args) {
89         new CardReport();
90     }
91 }

```

1.29.2 Concatenated 1



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26  import net.sf.dynamicreports.examples.complex.invoice.InvoiceDesign;
27  import net.sf.dynamicreports.examples.complex.sales.SalesDesign;
28  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
29  import net.sf.dynamicreports.jasper.builder.export.Exporters;
30  import net.sf.dynamicreports.report.exception.DRException;
31
32  /**
33   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)

```

(continues on next page)

(continued from previous page)

```

34  */
35  public class ConcatenatedReport1 {
36
37      public ConcatenatedReport1() {
38          build();
39      }
40
41      private void build() {
42          try {
43              concatenatedReport()
44                  .concatenate(
45                  ↪salesReport())
46                  .toPdf(Exporters.pdfExporter("c:/report.pdf",
47                  ↪"));
48          } catch (DRException e) {
49              e.printStackTrace();
50          }
51
52      private JasperReportBuilder invoiceReport() throws DRException {
53          return new InvoiceDesign().build();
54      }
55
56      private JasperReportBuilder salesReport() throws DRException {
57          return new SalesDesign().build();
58      }
59
60      public static void main(String[] args) {
61          new ConcatenatedReport1();
62      }
63  }

```

1.29.3 Concatenated 2

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *

```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
31 import net.sf.dynamicreports.jasper.builder.export.Exporters;
32 import net.sf.dynamicreports.report.constant.PageType;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class ConcatenatedReport2 {
41
42     public ConcatenatedReport2() {
43         build();
44     }
45
46     private void build() {
47         try {
48             concatenatedReport()
49                 .continuousPageNumbering()
50                 .concatenate(
51                     createReport(PageType.A4),
52                     createReport(PageType.A3),
53                     createReport(PageType.A5))
54                 .toPdf(Exporters.pdfExporter("c:/report.pdf
55 ↪"));
56         } catch (DRException e) {
57             e.printStackTrace();
58         }
59
60     private JasperReportBuilder createReport(PageType pageType) {
61         JasperReportBuilder report = report();
62         report
63             .setTemplate(Templates.reportTemplate)
64             .setPageFormat(pageType)

```

(continues on next page)

(continued from previous page)

```

65         .columns (
66             col.column("Item", "item", type.
↳stringType()),
67             col.column("Quantity", "quantity",
↳type.integerType()),
68             col.column("Unit price", "unitprice",
↳type.bigDecimalType()))
69         .title(Templates.createTitleComponent(pageType.name()
↳+ "Report"))
70         .pageFooter (
71             cmp.line(),
72             cmp.pageNumber().setStyle(Templates.
↳boldCenteredStyle))
73         .setDataSource(createDataSource());
74
75     return report;
76 }
77
78     private JRDataSource createDataSource() {
79         DRDataSource dataSource = new DRDataSource("item", "quantity",
↳"unitprice");
80         for (int i = 0; i < 20; i++) {
81             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↳BigDecimal(Math.random() * 100 + 1));
82         }
83         return dataSource;
84     }
85
86     public static void main(String[] args) {
87         new ConcatenatedReport2();
88     }
89 }

```

1.29.4 Dynamic Page Width

DynamicReports		DynamicPageWidth						
ID	Item	Quantity	Unit price	Order date	Order date	Order year	Order month	Order day
5 Notebook	1	300.00	01/02/2015	01/02/2015 9:16	42.037 A\$	2015	January	02
8 Book	7	300.00	01/02/2015	01/02/2015 9:16	42.037 A\$	2015	January	02
10 PCN	2	300.00	01/02/2015	01/02/2015 9:16	42.037 A\$	2015	January	02

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *

```

(continues on next page)

(continued from previous page)

```

4  * Copyright (C) 2010 - 2018 Ricardo Mariaca
5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Date;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class DynamicPageWidthReport {
39
40     public DynamicPageWidthReport() {
41         build();
42     }
43
44     private void build() {
45
46         try {
47             report()
48                 .setTemplate(Templates.reportTemplate)
49                 .ignorePageWidth()
50                 .columns(
51                     col.column("ID", "id", type.
52 ↪integerType()),
53                     col.column("Item", "item",
54 ↪type.stringType()),
55                     col.column("Quantity",
56 ↪"quantity", type.integerType()),
57                     col.column("Unit price",
58 ↪"unitprice", type.bigDecimalType()),
59                     col.column("Order date",
60 ↪"orderdate", type.dateType()),

```

(continues on next page)

(continued from previous page)

```

56         col.column("Order date",
↳ "orderdate", type.dateYearToFractionType()),
57         col.column("Order year",
↳ "orderdate", type.dateYearType()),
58         col.column("Order month",
↳ "orderdate", type.dateMonthType()),
59         col.column("Order day",
↳ "orderdate", type.dateDayType()))
60         .title(Templates.createTitleComponent(
↳ "DynamicPageWidth"))
61         .pageFooter(Templates.footerComponent)
62         .setDataSource(createDataSource())
63         .show();
64     } catch (DREException e) {
65         e.printStackTrace();
66     }
67 }
68
69 private JRDataSource createDataSource() {
70     DRDataSource dataSource = new DRDataSource("id", "item", "orderdate",
↳ "quantity", "unitprice");
71     dataSource.add(5, "Notebook", new Date(), 1, new BigDecimal(500));
72     dataSource.add(8, "Book", new Date(), 7, new BigDecimal(300));
73     dataSource.add(15, "PDA", new Date(), 2, new BigDecimal(250));
74     return dataSource;
75 }
76
77 public static void main(String[] args) {
78     new DynamicPageWidthReport();
79 }
80 }

```

1.29.5 Inheritance

Report A			Report B extends A		
Item	Quantity	Total	Item	Quantity	Unit price
Book	7	2100	Book	7	300
Book	10	3000	Book	3	75000
Book	8	2400	Book	4	12000
Book	2	1000	Book	8	80000
Book	5	1500	Book	6	72000
Book	4	1600	Book	10	20000
Book	9	2700	Book	9	14400
Book	2	800	Book	7	10500
Book	10	3000	Book	7	40000

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
31 import net.sf.dynamicreports.jasper.builder.export.Exporters;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.constant.PageOrientation;
34 import net.sf.dynamicreports.report.constant.PageType;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class InheritanceReport {
43
44     public InheritanceReport() {
45         build();
46     }
47
48     private void build() {
49         try {
50             concatenatedReport()
51                 .concatenate(
52                     new ReportA().report,
53                     new ReportB().report)
54             .toPdf(Exporters.pdfExporter("c:/report.pdf
55     ↪"));
56         } catch (DRException e) {
57             e.printStackTrace();
58         }
59     }
60
61     private class ReportA {
62         protected JasperReportBuilder report = report();
63
64         private ReportA() {
65             configure();
66         }

```

(continues on next page)

(continued from previous page)

```

67         protected void configure() {
68             report
69                 .setTemplate(Templates.reportTemplate)
70                 .setPageFormat(PageType.A5)
71                 .columns(
72                     col.column("Item", "item",
↪type.stringType()),
73                     col.column("Quantity",
↪"quantity", type.integerType()))
74                 .title(Templates.
↪createTitleComponent(getTitle()))
75                 .pageFooter(Templates.footerComponent)
76                 .setDataSource(createDataSource());
77         }
78
79         protected String getTitle() {
80             return "Report A";
81         }
82     }
83
84     private class ReportB extends ReportA {
85
86         @Override
87         protected void configure() {
88             super.configure();
89             TextColumnBuilder<BigDecimal> unitPriceColumn = col.column(
↪"Unit price", "unitprice", type.bigDecimalType());
90             report
91                 .setPageFormat(PageType.A5, PageOrientation.
↪LANDSCAPE)
92                 .addColumn(unitPriceColumn)
93                 .subtotalsAtSummary(sbt.sum(unitPriceColumn));
94         }
95
96         @Override
97         protected String getTitle() {
98             return "Report B extends A";
99         }
100     }
101
102     private JRDataSource createDataSource() {
103         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
104         for (int i = 0; i < 10; i++) {
105             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪BigDecimal(Math.random() * 100 + 1));
106         }
107         return dataSource;
108     }
109
110     public static void main(String[] args) {
111         new InheritanceReport();
112     }
113 }

```

1.29.6 List Background

Item	Quantity	Unit price	Order date
Book	10	51.971	01/22/2015
Book	3	90.351	01/22/2015
Book	9	62.146	01/22/2015
Book	10	1.473	01/22/2015
Book	1	3.034	01/22/2015
Book	9	98.768	01/22/2015
Book	6	16.267	01/22/2015
Book	4	90.879	01/22/2015
Book	7	89.275	01/22/2015
Book	4	64.252	01/22/2015
Book	7	77.759	01/22/2015
Book	4	19.173	01/22/2015
Book	4	92.068	01/22/2015
Book	8	14.219	01/22/2015
Book	6	60.05	01/22/2015
Book	9	94.561	01/22/2015
Book	3	92.289	01/22/2015
Book	9	81.319	01/22/2015
Book	2	92.182	01/22/2015
Book	10	55.274	01/22/2015
Book	3	72.896	01/22/2015
Book	5	76.685	01/22/2015
Book	2	66.051	01/22/2015
Book	4	44.827	01/22/2015
Book	9	32.795	01/22/2015
Book	6	15.803	01/22/2015
Book	10	17.00	01/22/2015
Book	8	65.271	01/22/2015
Book	9	12.382	01/22/2015
Book	6	3.881	01/22/2015

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28 import java.math.BigDecimal;
29 import java.util.Date;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.component.HorizontalListBuilder;
33 import net.sf.dynamicreports.report.builder.component.ImageBuilder;
34 import net.sf.dynamicreports.report.builder.component.RectangleBuilder;
35 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
36 import net.sf.dynamicreports.report.constant.ImageScale;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40
41 /**
42  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
43  */
44 public class ListBackgroundReport {
45
46     public ListBackgroundReport() {
47         build();
48     }
49
50     private void build() {
51         StyleBuilder style1 = stl.style()
52             .setRadius(10)
53             .setBackground(new Color(230, 230, 230))
54             .setLinePen(stl.pen().setLineColor(Color.LIGHT_GRAY));
55         StyleBuilder style2 = stl.style()
56             .setRadius(5);
57
58         ImageBuilder background1 = cmp.image(Templates.class.getResource(
59             ↪ "images/background.gif"))
60             .setImageScale(ImageScale.CLIP)
61             .setStyle(style1);
62         RectangleBuilder background2 = cmp.rectangle()
63             .setStyle(style2);
64         RectangleBuilder background3 = cmp.rectangle()
65             .setStyle(style1)
66             .setPrintWhenExpression(exp.printInOddRow());
67
68         HorizontalListBuilder title1 = cmp.horizontalList()
69             .add(cmp.text("title1"))
70             .setBackgroundComponent(background2);
71         HorizontalListBuilder title2 = cmp.horizontalList()
72             .add(cmp.text("title2"))
73             .setBackgroundComponent(background2);
74         HorizontalListBuilder title = cmp.horizontalList()
75             .add(title1, cmp.horizontalGap(20), title2)
76             .setStyle(stl.style(10));
77
78         try {
79             report()
80                 .setColumnStyle(Templates.columnStyle)
81                 .setColumnTitleStyle(Templates.
82                 ↪ boldCenteredStyle)
83                 .setTitleBackgroundComponent(background1)
84                 .setColumnHeaderBackgroundComponent(background2)
85                 .setPageFooterBackgroundComponent(background1)
86                 .setDetailBackgroundComponent(background3)

```

(continues on next page)

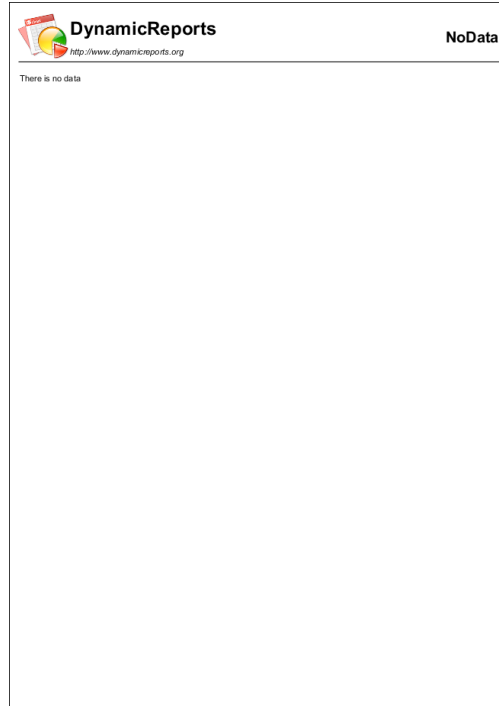
(continued from previous page)

```

85         .columns(
86             col.column("Item", "item",
87                 ↪type.stringType()),
88             col.column("Quantity",
89                 ↪"quantity", type.integerType()),
90             col.column("Unit price",
91                 ↪"unitprice", type.bigDecimalType()),
92             col.column("Order date",
93                 ↪"orderdate", type.dateType()))
94         .title(
95             Templates.
96             ↪createTitleComponent("ListBackground"),
97             title)
98         .pageFooter(cmp.pageXofY().setStyle(Templates.
99             ↪boldCenteredStyle))
100         .setDataSource(createDataSource())
101         .show();
102     } catch (DRException e) {
103         e.printStackTrace();
104     }
105 }
106
107 private JRDataSource createDataSource() {
108     DRDataSource dataSource = new DRDataSource("item", "orderdate",
109     ↪"quantity", "unitprice");
110     for (int i = 0; i < 30; i++) {
111         dataSource.add("Book", new Date(), (int) (Math.random() * 10)
112     ↪+ 1, new BigDecimal(Math.random() * 100 + 1));
113     }
114     return dataSource;
115 }
116
117 public static void main(String[] args) {
118     new ListBackgroundReport();
119 }
120 }

```

1.29.7 No Data



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)


(continued from previous page)

```

31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class NoDataReport {
39
40     public NoDataReport() {
41         build();
42     }
43
44     private void build() {
45         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
46         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
47         TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
↪"unitprice", type.bigDecimalType());
48
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .columns(
53                     itemColumn, quantityColumn,
↪priceColumn)
54                 .noData(Templates.createTitleComponent("NoData
↪"), cmp.text("There is no data"))
55                 .setDataSource(createDataSource())
56                 .show();
57         } catch (DRException e) {
58             e.printStackTrace();
59         }
60     }
61
62     private JRDataSource createDataSource() {
63         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
64         // dataSource.add("Book", 10, new BigDecimal(100));
65         return dataSource;
66     }
67
68     public static void main(String[] args) {
69         new NoDataReport();
70     }
71 }

```

1.29.8 Page Format

DynamicReports			PageFormat		
 http://www.dynamicreports.org					
Item	Quantity	Unit price	Item	Quantity	Unit price
Book	3	80.651	Book	3	79.31
Book	6	78.936	Book	10	36.909
Book	2	31.039	Book	5	36.36
Book	6	69.948	Book	10	89.858
Book	5	49.423	Book	1	19.372
Book	7	17.279	Book	1	73.225
Book	1	80.222	Book	6	5.863
Book	6	31.903	Book	8	87.538
Book	3	27.166	Book	2	54.031
Book	10	90.244	Book	7	32.767
Book	8	61.759	Book	9	33.47
Book	1	14.168	Book	6	91.333
Book	10	94.834	Book	7	18.003
Book	7	13.613	Book	8	1.123
Book	9	52.908	Book	6	95.672
Book	5	94.20	Book	10	48.972
			1 of 1		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.constant.PageOrientation;
32  import net.sf.dynamicreports.report.constant.PageType;
33  import net.sf.dynamicreports.report.datasource.JRDataSource;
34  import net.sf.dynamicreports.report.exception.DRException;
35  import net.sf.jasperreports.engine.JRDataSource;
36
37  /**
38   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39   */
40  public class PageFormatReport {
41
42      public PageFormatReport() {

```

(continues on next page)

(continued from previous page)

```

43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳type.stringType());
48         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳"quantity", type.integerType());
49         TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
↳"unitprice", type.bigDecimalType());
50
51         try {
52             report()
53                 .setTemplate(Templates.reportTemplate)
54                 .setPageFormat(PageType.A5, PageOrientation.
↳LANDSCAPE)
55                 .setPageColumnsPerPage(3)
56                 .setPageColumnSpace(15)
57                 .setPageMargin(margin(20))
58                 .columns(
59                     itemColumn, quantityColumn,
↳priceColumn)
60                 .title(Templates.createTitleComponent(
↳"PageFormat"))
61                 .pageFooter(Templates.footerComponent)
62                 .setDataSource(createDataSource())
63                 .show();
64         } catch (DRException e) {
65             e.printStackTrace();
66         }
67     }
68
69     private JRDataSource createDataSource() {
70         DRDataSource dataSource = new DRDataSource("item", "quantity",
↳"unitprice");
71         for (int i = 0; i < 40; i++) {
72             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↳BigDecimal(Math.random() * 100 + 1));
73         }
74         return dataSource;
75     }
76
77     public static void main(String[] args) {
78         new PageFormatReport();
79     }
80 }

```

1.29.9 Print When Expression

DynamicReports					PrintWhenExpression				
Old page header					New page header				
id	Quantity	Unit price	Quantity	Unit price	id	Quantity	Unit price	Quantity	Unit price
1	5	22.125	5	22.125	1	5	22.125	5	22.125
2	5	22.125	5	22.125	2	5	22.125	5	22.125
3	5	22.125	5	22.125	3	5	22.125	5	22.125
4	5	22.125	5	22.125	4	5	22.125	5	22.125
5	5	22.125	5	22.125	5	5	22.125	5	22.125
6	5	22.125	5	22.125	6	5	22.125	5	22.125
7	5	22.125	5	22.125	7	5	22.125	5	22.125
8	5	22.125	5	22.125	8	5	22.125	5	22.125
9	5	22.125	5	22.125	9	5	22.125	5	22.125
10	5	22.125	5	22.125	10	5	22.125	5	22.125
11	5	22.125	5	22.125	11	5	22.125	5	22.125
12	5	22.125	5	22.125	12	5	22.125	5	22.125
13	5	22.125	5	22.125	13	5	22.125	5	22.125
14	5	22.125	5	22.125	14	5	22.125	5	22.125
15	5	22.125	5	22.125	15	5	22.125	5	22.125
16	5	22.125	5	22.125	16	5	22.125	5	22.125
17	5	22.125	5	22.125	17	5	22.125	5	22.125
18	5	22.125	5	22.125	18	5	22.125	5	22.125
19	5	22.125	5	22.125	19	5	22.125	5	22.125
20	5	22.125	5	22.125	20	5	22.125	5	22.125
21	5	22.125	5	22.125	21	5	22.125	5	22.125
22	5	22.125	5	22.125	22	5	22.125	5	22.125
23	5	22.125	5	22.125	23	5	22.125	5	22.125
24	5	22.125	5	22.125	24	5	22.125	5	22.125
25	5	22.125	5	22.125	25	5	22.125	5	22.125
26	5	22.125	5	22.125	26	5	22.125	5	22.125
27	5	22.125	5	22.125	27	5	22.125	5	22.125
28	5	22.125	5	22.125	28	5	22.125	5	22.125
29	5	22.125	5	22.125	29	5	22.125	5	22.125
30	5	22.125	5	22.125	30	5	22.125	5	22.125
31	5	22.125	5	22.125	31	5	22.125	5	22.125
32	5	22.125	5	22.125	32	5	22.125	5	22.125
33	5	22.125	5	22.125	33	5	22.125	5	22.125
34	5	22.125	5	22.125	34	5	22.125	5	22.125
35	5	22.125	5	22.125	35	5	22.125	5	22.125
36	5	22.125	5	22.125	36	5	22.125	5	22.125
37	5	22.125	5	22.125	37	5	22.125	5	22.125
38	5	22.125	5	22.125	38	5	22.125	5	22.125
39	5	22.125	5	22.125	39	5	22.125	5	22.125
40	5	22.125	5	22.125	40	5	22.125	5	22.125
41	5	22.125	5	22.125	41	5	22.125	5	22.125
42	5	22.125	5	22.125	42	5	22.125	5	22.125

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */

```

```

22 package net.sf.dynamicreports.examples.miscellaneous;
23
24 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
25
26 import java.math.BigDecimal;
27
28 import net.sf.dynamicreports.examples.Templates;
29 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
32 import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
33 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34 import net.sf.dynamicreports.report.constant.Constants;
35 import net.sf.dynamicreports.report.constant.GroupHeaderLayout;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.definition.ReportParameters;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40

```

```

41 /**
42

```

(continues on next page)

(continued from previous page)

```

43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45  public class PrintWhenExpressionReport {
46
47      public PrintWhenExpressionReport() {
48          build();
49      }
50
51      private void build() {
52          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
53
54          ColumnGroupBuilder itemGroup = grp.group("itemGroup", itemColumn)
55              .setHeaderLayout(GroupHeaderLayout.EMPTY);
56
57          TextFieldBuilder<String> groupHeader = cmp.text(new
↪GroupHeaderExpression())
58              .setStyle(Templates.groupStyle)
59              .setPrintWhenExpression(new
↪PrintGroupHeaderExpression())
60              .removeLineWhenBlank();
61
62          VerticalListBuilder oddPageHeader = cmp.verticalList()
63              .add(
64                  cmp.text("Odd page header").
↪setStyle(Templates.bold12CenteredStyle),
65                  cmp.line())
66              .setPrintWhenExpression(new
↪PrintInOddPageExpression())
67              .removeLineWhenBlank();
68
69          VerticalListBuilder evenPageHeader = cmp.verticalList()
70              .add(
71                  cmp.line(),
72                  cmp.text("Even page header").
↪setStyle(Templates.bold12CenteredStyle),
73                  cmp.line())
74              .setPrintWhenExpression(new
↪PrintInEvenPageExpression())
75              .removeLineWhenBlank();
76
77          try {
78              report()
79                  .setTemplate(Templates.reportTemplate)
80                  .setPageColumnsPerPage(2)
81                  .setPageColumnSpace(5)
82                  .columns(
83                      itemColumn,
84                      col.column("Quantity",
↪"quantity", type.integerType()),
85                      col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
86                  .groupBy(itemGroup)
87                  .title(Templates.createTitleComponent(
↪"PrintWhenExpression"))
88                  .detailHeader(
89                      cmp.columnBreak().
↪setPrintWhenExpression(new PrintGroupHeaderColumnBreakExpression()),

```

(continues on next page)

(continued from previous page)

```

90                                     groupHeader)
91                                     .pageHeader (
92                                     oddPageHeader, evenPageHeader,
↪ cmp.verticalGap(10))
93                                     .pageFooter(Templates.footerComponent)
94                                     .setDataSource(createDataSource())
95                                     .show();
96         } catch (DRException e) {
97             e.printStackTrace();
98         }
99     }
100
101     private JRDataSource createDataSource() {
102         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
103         for (int i = 0; i < 50; i++) {
104             dataSource.add("DVD", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
105         }
106         for (int i = 0; i < 50; i++) {
107             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
108         }
109         return dataSource;
110     }
111
112     public class GroupHeaderExpression extends AbstractSimpleExpression<String> {
113         private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪ UID;
114
115         @Override
116         public String evaluate(ReportParameters reportParameters) {
117             return reportParameters.getValue("item");
118         }
119     }
120
121     public class PrintGroupHeaderExpression extends AbstractSimpleExpression
↪ <Boolean> {
122         private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪ UID;
123
124         @Override
125         public boolean evaluate(ReportParameters reportParameters) {
126             return reportParameters.getColumnRowNumber() == 1 ||
↪ reportParameters.getGroupCount("itemGroup") == 1;
127         }
128     }
129
130     public class PrintGroupHeaderColumnBreakExpression extends
↪ AbstractSimpleExpression<Boolean> {
131         private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪ UID;
132
133         @Override
134         public boolean evaluate(ReportParameters reportParameters) {
135             return reportParameters.getColumnRowNumber() == 1 &&
↪ reportParameters.getGroupCount("itemGroup") != 1;


```

(continues on next page)

(continued from previous page)

```
136         }
137     }
138
139     public class PrintInOddPageExpression extends AbstractSimpleExpression
140     <<Boolean> {
141         private static final long serialVersionUID = Constants.SERIAL_VERSION_
142         UID;
143
144         @Override
145         public Boolean evaluate(ReportParameters reportParameters) {
146             return reportParameters.getPageNumber().doubleValue() % 2 !=
147             0;
148         }
149     }
150
151     public class PrintInEvenPageExpression extends AbstractSimpleExpression
152     <<Boolean> {
153         private static final long serialVersionUID = Constants.SERIAL_VERSION_
154         UID;
155
156         @Override
157         public Boolean evaluate(ReportParameters reportParameters) {
158             return reportParameters.getPageNumber().doubleValue() % 2 ==
159             0;
160         }
161     }
162
163     public static void main(String[] args) {
164         new PrintWhenExpressionReport();
165     }
166 }
```

1.29.10 Resource Bundle



DynamicReports

<http://www.dynamicreports.org>

ResourceBundle

Report

Quantity: min=1, max=10

Item	Quantity	Unit Price	
Book	3		16.28
Book	9		24.59
Book	6		63.66
Book	1		20.62
Book	5		2.94
Book	10		9.76
Book	6		50.91
Book	3		50.85
Book	10		62.67
Book	6		78.04

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.List;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
33 import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
34 import net.sf.dynamicreports.report.constant.Calculation;
35 import net.sf.dynamicreports.report.constant.Constants;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.definition.ReportParameters;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40
41 /**
42  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
43  */
44 public class ResourceBundleReport {
45
46     public ResourceBundleReport() {
47         build();
48     }
49
50     private void build() {
51         try {
52             TextColumnBuilder<String> itemColumn = col.column("item",
↪type.stringType())
53                                     .setTitle(exp.message("item_title"));
54             TextColumnBuilder<Integer> quantityColumn = col.column(
↪"quantity", type.integerType())
55                                     .setTitle(exp.message("quantity_title"));
56             TextColumnBuilder<BigDecimal> priceColumn = col.column(
↪"unitprice", type.bigDecimalType())
57                                     .setTitle(exp.message("unitprice_title"));
58
59             TextFieldBuilder<String> title = cmp.text(exp.message("report_
↪title"))
60                                     .setStyle(Templates.bold12CenteredStyle);
61             TextFieldBuilder<String> subtitle = cmp.text(new
↪SubtitleExpression())
62                                     .setStyle(Templates.bold12CenteredStyle);
63
64             report()
65                 .setTemplate(Templates.reportTemplate)
66                 .setResourceBundle(ResourceBundleReport.class.
↪getName())
67                 .columns(
68                     itemColumn, quantityColumn,
↪priceColumn)
69                 .title(
70                     Templates.
↪createTitleComponent("ResourceBundle"),
71                     title, subtitle)
72                 .pageFooter(Templates.footerComponent)
73                 .setDataSource(createDataSource())
74                 .show();
75         } catch (DRException e) {
76             e.printStackTrace();
77         }
78     }
79

```

(continues on next page)


(continued from previous page)

```

80     private JRDataSource createDataSource() {
81         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
82         for (int i = 0; i < 10; i++) {
83             dataSource.add("Book", (int) (Math.random() * 10) + 1, new_
↪ BigDecimal(Math.random() * 100 + 1));
84         }
85         return dataSource;
86     }
87
88     public static void main(String[] args) {
89         new ResourceBundleReport();
90     }
91
92     public class SubtitleExpression extends AbstractComplexExpression<String> {
93         private static final long serialVersionUID = Constants.SERIAL_VERSION_
↪ UID;
94
95         public SubtitleExpression() {
96             addExpression(variable("quantity", Integer.class, Calculation.
↪ LOWEST));
97             addExpression(variable("quantity", Integer.class, Calculation.
↪ HIGHEST));
98         }
99
100         @Override
101         public String evaluate(List<?> values, ReportParameters_
↪ reportParameters) {
102             return reportParameters.getMessage("report_subtitle", new_
↪ Object[] { values.get(0), values.get(1) });
103         }
104     }
105 }

```

1.29.11 Scriptlet

DynamicReports		Scriptlet
		http://www.dynamicreports.org
	Item	
USA		
	Book	
	DVD	
	Book	
	Book	
	DVD	
	Book	
	DVD	
	Book = 4	
	DVD = 3	
Canada		
	Book	
	Book	
	DVD	
	Book	
	DVD	
	Phone	
	Book = 3	
	Phone = 1	
	DVD = 2	
1 of 1		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.util.HashMap;
28  import java.util.Map;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.base.AbstractScriptlet;
32 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.group.CustomGroupBuilder;
35 import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.definition.ReportParameters;
38 import net.sf.dynamicreports.report.exception.DRException;
39 import net.sf.jasperreports.engine.JRDataSource;
40
41 import org.apache.commons.lang3.StringUtils;
42
43 /**
44  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
45  */
46 public class ScriptletReport {
47     private Map<String, Integer> itemsCount;
48
49     public ScriptletReport() {
50         build();
51     }
52
53     private void build() {
54         itemsCount = new HashMap<String, Integer>();
55
56         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
57 ↪type.stringType());
58         CustomSubtotalBuilder<String> itemSbt = sbt.customValue(new
59 ↪ItemSubtotal(), itemColumn);
60         CustomGroupBuilder group = grp.group("country", String.class);
61
62         try {
63             report()
64                 .setTemplate(Templates.reportTemplate)
65                 .scriptlets(new ReportScriptlet())
66                 .columns(itemColumn)
67                 .groupBy(group)
68                 .subtotalsAtGroupFooter(group, itemSbt)
69                 .title(Templates.createTitleComponent(
70 ↪"Scriptlet"))
71                 .pageFooter(Templates.footerComponent)
72                 .setDataSource(createDataSource())
73                 .show();
74         } catch (DRException e) {
75             e.printStackTrace();
76         }
77     }
78
79     private class ItemSubtotal extends AbstractSimpleExpression<String> {
80         private static final long serialVersionUID = 1L;
81
82         @Override
83         public String evaluate(ReportParameters reportParameters) {
84             String result = "";
85             for (String item : itemsCount.keySet()) {
86                 result += item + " = " + itemsCount.get(item) + "\n";
87             }
88         }
89     }

```

(continues on next page)

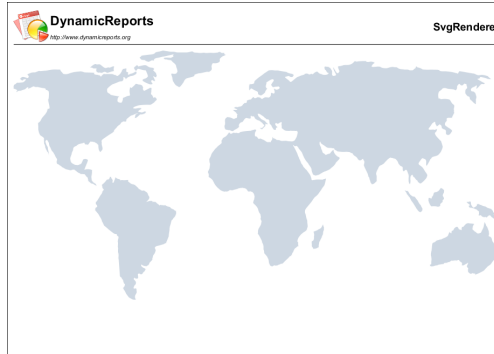
(continued from previous page)

```

85         return StringUtils.removeEnd(result, "\n");
86     }
87 }
88
89 private class ReportScriptlet extends AbstractScriptlet {
90
91     @Override
92     public void afterDetailEval(ReportParameters reportParameters) {
93         super.afterDetailEval(reportParameters);
94         String item = reportParameters.getValue("item");
95         Integer count;
96         if (itemsCount.containsKey(item)) {
97             count = itemsCount.get(item);
98         } else {
99             count = 0;
100         }
101         itemsCount.put(item, ++count);
102     }
103
104     @Override
105     public void afterGroupInit(String groupName, ReportParameters_
↪ reportParameters) {
106         super.afterGroupInit(groupName, reportParameters);
107         itemsCount.clear();
108     }
109 }
110
111 private JRDataSource createDataSource() {
112     DRDataSource dataSource = new DRDataSource("country", "item");
113     dataSource.add("USA", "Book");
114     dataSource.add("USA", "DVD");
115     dataSource.add("USA", "Book");
116     dataSource.add("USA", "Book");
117     dataSource.add("USA", "DVD");
118     dataSource.add("USA", "Book");
119     dataSource.add("USA", "DVD");
120
121     dataSource.add("Canada", "Book");
122     dataSource.add("Canada", "Book");
123     dataSource.add("Canada", "DVD");
124     dataSource.add("Canada", "Book");
125     dataSource.add("Canada", "DVD");
126     dataSource.add("Canada", "Phone");
127     return dataSource;
128 }
129
130 public static void main(String[] args) {
131     new ScriptletReport();
132 }
133 }

```

1.29.12 Svg Renderer



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import net.sf.dynamicreports.examples.Templates;
28  import net.sf.dynamicreports.report.constant.PageOrientation;
29  import net.sf.dynamicreports.report.constant.PageType;
30  import net.sf.dynamicreports.report.exception.DRException;
31  import net.sf.jasperreports.engine.JRException;
32  import net.sf.jasperreports.engine.util.JRLoader;
33  import net.sf.jasperreports.renderers.Renderable;
34  import net.sf.jasperreports.renderers.SimpleDataRenderer;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class SvgRendererReport {
40
41      public SvgRendererReport() {
42          build();

```

(continues on next page)


(continued from previous page)

```

43     }
44
45     private void build() {
46         try {
47             Renderable image = new SimpleDataRenderer(JRLoader.
↳loadBytes(Templates.class.getResource("images/map.svg"), null));
48
49             report()
50                 .setTemplate(Templates.reportTemplate)
51                 .setPageFormat(PageType.A4, PageOrientation.
↳LANDSCAPE)
52                 .title(
53                     Templates.
↳createTitleComponent("SvgRenderer"),
54                     cmp.image(image).
↳setHeight(500))
55                 .show();
56         } catch (DRException e) {
57             e.printStackTrace();
58         } catch (JRException e) {
59             e.printStackTrace();
60         }
61     }
62
63     public static void main(String[] args) {
64         new SvgRendererReport();
65     }
66 }

```

1.29.13 Units

DynamicReports			Units
 http://www.dynamicreports.org			
width = 120 pixels			
width = 10cm			
width = 5 inches			
width = 150mm			
Item	Quantity	Unit price	
Book	9	43.88	
Book	2	18.521	
Book	4	59.609	
Book	7	36.61	
Book	2	31.454	
Book	4	94.065	
Book	4	17.026	
Book	7	18.25	
Book	1	36.002	
Book	6	1.987	
Book	9	98.422	
Book	8	58.485	
Book	1	97.324	
Book	10	75.556	
Book	6	87.967	
Book	10	27.73	
Book	9	18.175	
Book	10	98.74	
Book	2	44.735	
Book	3	38.628	
			1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.miscellaneous;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
32 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class UnitsReport {
41
42     public UnitsReport() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
48 ↪ type.stringType())
49             .setFixedWidth(cm(10));
50         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
51 ↪ "quantity", type.integerType());
52         TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
53 ↪ "unitprice", type.bigDecimalType());
54
55         StyleBuilder style = stl.style(Templates.bold12CenteredStyle)
56             .setBorder(stl.pen1Point());
57         TextFieldBuilder<String> text1 = cmp.text("width = 120 pixels")

```

(continues on next page)

(continued from previous page)

```

55         .setFixedWidth(120)
56         .setStyle(style);
57     TextFieldBuilder<String> text2 = cmp.text("width = 10cm")
58         .setFixedWidth(cm(10))
59         .setStyle(style);
60     TextFieldBuilder<String> text3 = cmp.text("width = 5 inches")
61         .setFixedWidth(inch(5))
62         .setStyle(style);
63     TextFieldBuilder<String> text4 = cmp.text("width = 150mm")
64         .setFixedWidth(mm(150))
65         .setStyle(style);
66
67     try {
68         report()
69             .setTemplate(Templates.reportTemplate)
70             .columns(
71                 itemColumn, quantityColumn,
↪ priceColumn)
72             .title(
73                 Templates.
↪ createTitleComponent("Units"),
74                 text1, text2, text3, text4,
↪ cmp.verticalGap(cm(1)))
75             .pageFooter(Templates.footerComponent)
76             .setDataSource(createDataSource())
77             .show();
78     } catch (DRException e) {
79         e.printStackTrace();
80     }
81 }
82
83 private JRDataSource createDataSource() {
84     DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
85     for (int i = 0; i < 20; i++) {
86         dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
87     }
88     return dataSource;
89 }
90
91 public static void main(String[] args) {
92     new UnitsReport();
93 }
94 }

```

1.29.14 Variable

DynamicReports			Variable
 http://www.dynamicreports.org			
Item count = 30 Quantity sum = 164 Unit price sum = 1,421.198 SUM(quantity * unit price) = 7,858.64			
Item	Quantity	Unit price	
Book	7	23.171	
Book	4	95.938	
Book	6	15.632	
Book	8	21.607	
Book	4	51.546	
Book	7	89.092	
Book	1	82.333	
Book	8	15.141	
Book	7	74.277	
Book	8	34.87	
Book	3	22.198	
Book	4	99.545	
Book	4	70.132	
Book	8	69.893	
Book	1	10.024	
Book	2	11.148	
Book	10	50.622	
Book	10	31.273	
Book	7	23.076	
Book	1	42.216	
Book	4	17.194	
Book	9	65.533	
Book	8	22.87	
Book	5	48.337	
Book	2	84.884	
Book	3	33.916	
Book	7	66.106	
Book	3	37.409	
Book	8	54.065	
Book	7	87.751	
			1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.List;
29
30  import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
32 import net.sf.dynamicreports.report.builder.VariableBuilder;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.expression.AbstractComplexExpression;
35 import net.sf.dynamicreports.report.constant.Calculation;
36 import net.sf.dynamicreports.report.constant.Evaluation;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class VariableReport {
46
47     public VariableReport() {
48         build();
49     }
50
51     private void build() {
52         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType());
53         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
54         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
55
56         VariableBuilder<Integer> itemCount = variable(itemColumn, Calculation.
↪COUNT);
57         VariableBuilder<Integer> quantitySum = variable("quantitySum",
↪quantityColumn, Calculation.SUM);
58         VariableBuilder<Integer> priceSum = variable(new
↪PriceExpression(quantityColumn, unitPriceColumn), Calculation.SUM);
59
60         try {
61             report()
62                 .setTemplate(Templates.reportTemplate)
63                 .variables(quantitySum)
64                 .columns(
65                     itemColumn, quantityColumn,
↪unitPriceColumn)
66                 .title(
67                     Templates.
↪createTitleComponent("Variable"),
68                     cmp.horizontalList(cmp.text(
↪"Item count =").setFixedWidth(80), cmp.text(itemCount)),
69                     cmp.text(new
↪QuantitySumTextExpression()).setEvaluationTime(Evaluation.REPORT),
70                     cmp.text(new
↪UnitPriceSumTextExpression(unitPriceColumn)),
71                     cmp.horizontalList(cmp.text(
↪"SUM(quantity * unit price) =").setFixedWidth(150), cmp.text(priceSum).setPattern("
↪#,###.00")))
72                 .pageFooter(Templates.footerComponent)
73                 .setDataSource(createDataSource())
74                 .show();

```

(continues on next page)

(continued from previous page)

```

75         } catch (DRException e) {
76             e.printStackTrace();
77         }
78     }
79
80     private JRDataSource createDataSource() {
81         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
82         for (int i = 0; i < 30; i++) {
83             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
84         }
85         return dataSource;
86     }
87
88     public static void main(String[] args) {
89         new VariableReport();
90     }
91
92     private class QuantitySumTextExpression extends AbstractSimpleExpression
↪ <String> {
93         private static final long serialVersionUID = 1L;
94
95         @Override
96         public String evaluate(ReportParameters reportParameters) {
97             Integer quantitySum = reportParameters.getValue("quantitySum
↪ ");
98             return "Quantity sum = " + quantitySum;
99         }
100     }
101
102     private class UnitPriceSumTextExpression extends AbstractComplexExpression
↪ <String> {
103         private static final long serialVersionUID = 1L;
104
105         public UnitPriceSumTextExpression(TextColumnBuilder<BigDecimal>
↪ unitPriceColumn) {
106             addExpression(variable(unitPriceColumn, Calculation.SUM));
107         }
108
109         @Override
110         public String evaluate(List<?> values, ReportParameters
↪ reportParameters) {
111             BigDecimal unitPriceSum = (BigDecimal) values.get(0);
112             return "Unit price sum = " + type.bigDecimalType().
↪ valueToString(unitPriceSum, reportParameters.getLocale());
113         }
114     }
115
116     private class PriceExpression extends AbstractSimpleExpression<BigDecimal> {
117         private static final long serialVersionUID = 1L;
118
119         private TextColumnBuilder<Integer> quantityColumn;
120         private TextColumnBuilder<BigDecimal> unitPriceColumn;
121
122         public PriceExpression(TextColumnBuilder<Integer> quantityColumn,
↪ TextColumnBuilder<BigDecimal> unitPriceColumn) {

```

(continues on next page)

(continued from previous page)

```

123         this.quantityColumn = quantityColumn;
124         this.unitPriceColumn = unitPriceColumn;
125     }
126
127     @Override
128     public BigDecimal evaluate(ReportParameters reportParameters) {
129         Integer quantity = reportParameters.getValue(quantityColumn);
130         BigDecimal unitPrice = reportParameters.
131         ↪getValue(unitPriceColumn);
132         return unitPrice.multiply(new BigDecimal(quantity));
133     }
134 }

```

1.29.15 Virtualizer

DynamicReports		Virtualizer	
		http://www.dynamicreports.org	
Item	Quantity	Unit price	
Book	4	33.225	
Book	4	92.622	
Book	4	37.251	
Book	7	17.117	
Book	4	7.52	
Book	9	27.717	
Book	7	56.475	
Book	1	72.15	
Book	2	2.63	
Book	6	67.902	
Book	8	24.766	
Book	3	43.178	
Book	3	22.731	
Book	3	88.036	
Book	2	27.155	
Book	6	78.17	
Book	8	47.386	
Book	7	7.171	
Book	7	72.001	
Book	2	64.733	
Book	7	39.521	
Book	6	28.072	
Book	2	78.273	
Book	10	91.743	
Book	5	73.819	
Book	9	47.042	
Book	10	59.998	
Book	3	36.381	
Book	5	45.296	
Book	7	50.283	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *

```

(continues on next page)

(continued from previous page)

```

14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.miscellaneous;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
31  import net.sf.dynamicreports.report.datasource.DRDataSource;
32  import net.sf.dynamicreports.report.exception.DRException;
33  import net.sf.jasperreports.engine.JRDataSource;
34  import net.sf.jasperreports.engine.fill.JRFileVirtualizer;
35
36  /**
37   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38   */
39  public class VirtualizerReport {
40
41      public VirtualizerReport() {
42          build();
43      }
44
45      private void build() {
46          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
47 ↪ type.stringType());
48          TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
49 ↪ "quantity", type.integerType());
50          TextColumnBuilder<BigDecimal> priceColumn = col.column("Unit price",
51 ↪ "unitprice", type.bigDecimalType());
52
53          try {
54              report()
55                  .setTemplate(Templates.reportTemplate)
56                  .columns(
57                      itemColumn, quantityColumn,
58 ↪ priceColumn)
59                  .title(Templates.createTitleComponent(
60 ↪ "Virtualizer"))
61                  .pageFooter(Templates.footerComponent)
62                  .setDataSource(createDataSource())
63                  .setVirtualizer(new JRFileVirtualizer(2))
64                  .show();
65          } catch (DRException e) {
66              e.printStackTrace();
67          }
68      }
69
70      private JRDataSource createDataSource() {

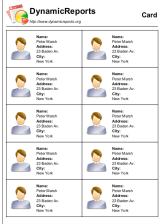


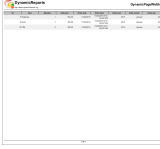

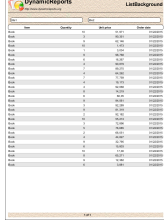
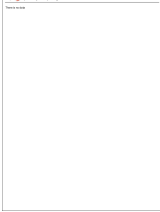
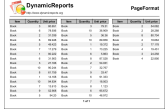
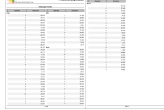
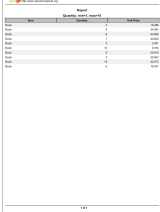



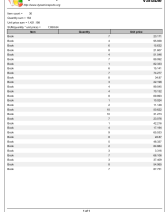

```

(continues on next page)

(continued from previous page)


```
66         DRDataSource dataSource = new DRDataSource("item", "quantity",  
↪ "unitprice");  
67         for (int i = 0; i < 30; i++) {  
68             // for (int i = 0; i < 100000; i++) {  
69                 dataSource.add("Book", (int) (Math.random() * 10) + 1, new_  
↪ BigDecimal(Math.random() * 100 + 1));  
70             }  
71             return dataSource;  
72         }  
73  
74         public static void main(String[] args) {  
75             new VirtualizerReport();  
76         }  
77     }
```

Table 14: Miscellaneous Examples

 <p>CardReport</p>	 <p>ConcatenatedReport1</p>	 <p>ConcatenatedReport2</p>
 <p>DynamicPageWidthReport</p>	 <p>InheritanceReport</p>	 <p>ListBackgroundReport</p>
 <p>NoDataReport</p>	 <p>PageFormatReport</p>	 <p>PrintWhenExpressionReport</p>
 <p>ResourceBundleReport</p>	 <p>ScriptletReport</p>	 <p>SvgRendererReport</p>
 <p>UnitsReport</p>	 <p>VariableReport</p>	 <p>VirtualizerReport</p>

1.30 Style

1.30.1 Conditional Style

DynamicReports		ConditionalStyle	
		http://www.dynamicreports.org	
Item	Order date	Quantity	Unit price
DVD	01/01/2010	5	30.00
DVD	01/03/2010	1	28.00
DVD	01/19/2010	5	32.00
Book	01/05/2010	3	11.00
Book	01/11/2010	1	15.00
Book	01/15/2010	5	10.00
Book	01/20/2010	8	9.00

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.style;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;

```

(continues on next page)

(continued from previous page)

```

28 import java.math.BigDecimal;
29 import java.util.Calendar;
30 import java.util.Date;
31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
34 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
35 import net.sf.dynamicreports.report.builder.style.ConditionalStyleBuilder;
36 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
37 import net.sf.dynamicreports.report.datasource.DRDataSource;
38 import net.sf.dynamicreports.report.definition.ReportParameters;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class ConditionalStyleReport {
46
47     public ConditionalStyleReport() {
48         build();
49     }
50
51     private void build() {
52         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
53         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↳ "orderdate", type.dateType());
54         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↳ "quantity", type.integerType());
55         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↳ ", "unitprice", type.bigDecimalType());
56
57         ConditionalStyleBuilder condition1 = stl.conditionalStyle(new
↳ OrderDateConditionExpression())
58             .setBackgroundColor(new Color(255, 210, 210));
59         ConditionalStyleBuilder condition2 = stl.conditionalStyle(cnd.
↳ greater(unitPriceColumn, 20))
60             .setBackgroundColor(new Color(210, 255, 210));
61
62         StyleBuilder orderDateStyle = stl.style()
63             .conditionalStyles(condition1);
64         orderDateColumn.setStyle(orderDateStyle);
65
66         StyleBuilder unitPriceStyle = stl.style()
67             .conditionalStyles(condition2);
68         unitPriceColumn.setStyle(unitPriceStyle);
69
70         try {
71             report()
72                 .setTemplate(Templates.reportTemplate)
73                 .columns(
74                     itemColumn, orderDateColumn,
↳ quantityColumn, unitPriceColumn)
75                 .title(Templates.createTitleComponent(
↳ "ConditionalStyle"))
76                 .pageFooter(Templates.footerComponent)

```

(continues on next page)

(continued from previous page)

```

77         .setDataSource(createDataSource())
78         .show();
79     } catch (DRException e) {
80         e.printStackTrace();
81     }
82 }
83
84 private JRDataSource createDataSource() {
85     DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪ "quantity", "unitprice");
86     dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
87     dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
88     dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
89     dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
90     dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
91     dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));
92     dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
93     return dataSource;
94 }
95
96 private Date toDate(int year, int month, int day) {
97     Calendar c = Calendar.getInstance();
98     c.set(Calendar.YEAR, year);
99     c.set(Calendar.MONTH, month - 1);
100    c.set(Calendar.DAY_OF_MONTH, day);
101    return c.getTime();
102 }
103
104 public static void main(String[] args) {
105     new ConditionalStyleReport();
106 }
107
108 private class OrderDateConditionExpression extends AbstractSimpleExpression
↪ <Boolean> {
109     private static final long serialVersionUID = 1L;
110
111     @Override
112     public Boolean evaluate(ReportParameters reportParameters) {
113         Date orderDate = reportParameters.getValue("orderdate");
114         Integer quantity = reportParameters.getValue("quantity");
115         return orderDate.after(toDate(2010, 1, 10)) && quantity > 1;
116     }
117 }
118 }

```

1.30.2 Template Style File

DynamicReports		TemplateStyleFile		
http://www.dynamicreports.org				
Item	Order date	Quantity	Unit price	
DVD	01/01/2010	5	30,00	
DVD	01/03/2010	1	28,00	
DVD	01/19/2010	5	32,00	
Book	01/05/2010	3	11,00	
Book	01/11/2010	1	15,00	
Book	01/15/2010	5	10,00	
Book	01/20/2010	8	9,00	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.style;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class TemplateStyleFileReport {
41
42     public TemplateStyleFileReport() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪type.stringType())
48             .setStyle(stl.templateStyle("style1"));
49         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↪"orderdate", type.dateType())
50             .setStyle(stl.templateStyle("style2"));
51         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪"quantity", type.integerType());
52         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↪", "unitprice", type.bigDecimalType());
53
54         try {
55             report()
56                 .templateStyles(stl.
↪loadStyles(TemplateStyleFileReport.class.getResource("TemplateStyles.jrtx")))
57                 .setColumnStyle(stl.templateStyle("columnStyle
↪"))
58                 .setColumnTitleStyle(stl.templateStyle(
↪"columnTitleStyle"))
59                 .columns(
60                     itemColumn, orderDateColumn,
↪quantityColumn, unitPriceColumn)
61                 .title(Templates.createTitleComponent(
↪"TemplateStyleFile"))
62                 .pageFooter(Templates.footerComponent)
63                 .setDataSource(createDataSource())
64                 .show();
65         } catch (DRException e) {
66             e.printStackTrace();
67         }
68     }
69
70     private JRDataSource createDataSource() {
71         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
72         dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
73         dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
74         dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
75         dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
76         dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
77         dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));

```

(continues on next page)

(continued from previous page)

```

78         dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
79         return dataSource;
80     }
81
82     private Date toDate(int year, int month, int day) {
83         Calendar c = Calendar.getInstance();
84         c.set(Calendar.YEAR, year);
85         c.set(Calendar.MONTH, month - 1);
86         c.set(Calendar.DAY_OF_MONTH, day);
87         return c.getTime();
88     }
89
90     public static void main(String[] args) {
91         new TemplateStyleFileReport();
92     }
93 }

```

1.30.3 Template Style

DynamicReports		TemplateStyle		
http://www.dynamicreports.org				
Item	Order date	Quantity	Unit price	
DVD	01/01/2010	5	30,00	
DVD	01/03/2010	1	28,00	
DVD	01/19/2010	5	32,00	
Book	01/05/2010	3	11,00	
Book	01/11/2010	1	15,00	
Book	01/15/2010	5	10,00	
Book	01/20/2010	8	9,00	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.style;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28 import java.math.BigDecimal;
29 import java.util.Calendar;
30 import java.util.Date;
31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.report.builder.ReportTemplateBuilder;
34 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
35 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
36 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
37 import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
38 import net.sf.dynamicreports.report.datasource.DRDataSource;
39 import net.sf.dynamicreports.report.exception.DRException;
40 import net.sf.jasperreports.engine.JRDataSource;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class TemplateStyleReport {
46
47     public TemplateStyleReport() {
48         build();
49     }
50
51     private void build() {
52         StyleBuilder style1 = stl.style()
53             .setName("style1")
54             .bold();
55         StyleBuilder style2 = stl.style(style1)
56             .setName("style2")
57             .italic();
58         StyleBuilder columnStyle = stl.style()
59             .setName("columnStyle")
60             .setVerticalTextAlignment(VerticalTextAlignment.
61 ↪MIDDLE);
62         StyleBuilder columnTitleStyle = stl.style(columnStyle)
63             .setName("columnTitleStyle")
64             .setBorder(stl.pen1Point())
65             .setHorizontalTextAlignment(HorizontalTextAlignment.
66 ↪CENTER)
67             .setBackgroundColor(Color.LIGHT_GRAY);

```

(continues on next page)

(continued from previous page)

```

66         ReportTemplateBuilder template = template()
67             .templateStyles(style1, style2, columnStyle, ↵
↵columnTitleStyle);
68
69         TextColumnBuilder<String> itemColumn = col.column("Item", "item", ↵
↵type.stringType())
70             .setStyle(stl.templateStyle("style1"));
71         TextColumnBuilder<Date> orderDateColumn = col.column("Order date",
↵"orderdate", type.dateType())
72             .setStyle(stl.templateStyle("style2"));
73         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↵"quantity", type.integerType());
74         TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
↵", "unitprice", type.bigDecimalType());
75
76         try {
77             report()
78                 .setTemplate(template)
79                 .setColumnStyle(stl.templateStyle("columnStyle
↵"))
80                 .setColumnTitleStyle(stl.templateStyle(
↵"columnTitleStyle"))
81                 .columns(
82                     itemColumn, orderDateColumn, ↵
↵quantityColumn, unitPriceColumn)
83                 .title(Templates.createTitleComponent(
↵"TemplateStyle"))
84                 .pageFooter(Templates.footerComponent)
85                 .setDataSource(createDataSource())
86                 .show();
87         } catch (DRException e) {
88             e.printStackTrace();
89         }
90     }
91
92     private JRDataSource createDataSource() {
93         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↵"quantity", "unitprice");
94         dataSource.add("DVD", toDate(2010, 1, 1), 5, new BigDecimal(30));
95         dataSource.add("DVD", toDate(2010, 1, 3), 1, new BigDecimal(28));
96         dataSource.add("DVD", toDate(2010, 1, 19), 5, new BigDecimal(32));
97         dataSource.add("Book", toDate(2010, 1, 5), 3, new BigDecimal(11));
98         dataSource.add("Book", toDate(2010, 1, 11), 1, new BigDecimal(15));
99         dataSource.add("Book", toDate(2010, 1, 15), 5, new BigDecimal(10));
100        dataSource.add("Book", toDate(2010, 1, 20), 8, new BigDecimal(9));
101        return dataSource;
102    }
103
104    private Date toDate(int year, int month, int day) {
105        Calendar c = Calendar.getInstance();
106        c.set(Calendar.YEAR, year);
107        c.set(Calendar.MONTH, month - 1);
108        c.set(Calendar.DAY_OF_MONTH, day);
109        return c.getTime();
110    }
111
112    public static void main(String[] args) {

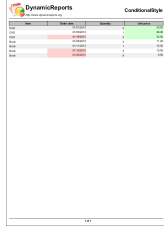
```

(continues on next page)

(continued from previous page)


```
113         new TemplateStyleReport();
114     }
115 }
```

Table 15: Style Examples

		
ConditionalStyleReport	TemplateStyleFileReport	TemplateStyleReport

1.31 Subreport

1.31.1 Detail Dynamic Subreport



DynamicReports

http://www.dynamicreports.org

DetailDynamicSubreport

Subreport1

Column1
row1_column1

Subreport2

Column1	Column2
row1_column1	row1_column2
row2_column1	row2_column2

Subreport3

Column1	Column2	Column3
row1_column1	row1_column2	row1_column3
row2_column1	row2_column2	row2_column3
row3_column1	row3_column2	row3_column3

Subreport4

Column1	Column2	Column3	Column4
row1_column1	row1_column2	row1_column3	row1_column4
row2_column1	row2_column2	row2_column3	row2_column4
row3_column1	row3_column2	row3_column3	row3_column4
row4_column1	row4_column2	row4_column3	row4_column4

Subreport5

Column1	Column2	Column3	Column4	Column5
row1_column1	row1_column2	row1_column3	row1_column4	row1_column5
row2_column1	row2_column2	row2_column3	row2_column4	row2_column5
row3_column1	row3_column2	row3_column3	row3_column4	row3_column5
row4_column1	row4_column2	row4_column3	row4_column4	row4_column5
row5_column1	row5_column2	row5_column3	row5_column4	row5_column5

1 of 1

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
```

(continues on next page)

(continued from previous page)

```

9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.subreport;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26 import net.sf.dynamicreports.examples.Templates;
27 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
28 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
29 import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
30 import net.sf.dynamicreports.report.datasource.DRDataSource;
31 import net.sf.dynamicreports.report.definition.ReportParameters;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34 import net.sf.jasperreports.engine.JREmptyDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class DetailDynamicSubreport {
40
41     public DetailDynamicSubreport() {
42         build();
43     }
44
45     private void build() {
46         SubreportBuilder subreport = cmp.subreport(new SubreportExpression())
47             .setDataSource(new SubreportDataSourceExpression());
48
49         try {
50             report()
51                 .title(Templates.createTitleComponent(
52                     ↪ "DetailDynamicSubreport"))
53                 .detail(
54                     subreport,
55                     cmp.verticalGap(20))
56                 .pageFooter(Templates.footerComponent)
57                 .setDataSource(createDataSource())
58                 .show();
59         } catch (DRException e) {
60             e.printStackTrace();
61         }
62
63         private JRDataSource createDataSource() {
64             return new JREmptyDataSource(5);
65         }
66     }

```

(continues on next page)


(continued from previous page)

```

65     }
66
67     private class SubreportExpression extends AbstractSimpleExpression
68     ↪<JasperReportBuilder> {
69         private static final long serialVersionUID = 1L;
70
71         @Override
72         ↪public JasperReportBuilder evaluate(ReportParameters_
73         ↪reportParameters) {
74             int masterRowNumber = reportParameters.getReportRowNumber();
75             JasperReportBuilder report = report();
76             report
77                 .setTemplate(Templates.reportTemplate)
78                 .title(cmp.text("Subreport" +
79         ↪masterRowNumber).setStyle(Templates.bold12CenteredStyle));
80
81             for (int i = 1; i <= masterRowNumber; i++) {
82                 report.addColumn(col.column("Column" + i, "column" +
83         ↪i, type.stringType()));
84             }
85
86             return report;
87         }
88     }
89
90     private class SubreportDataSourceExpression extends AbstractSimpleExpression
91     ↪<JRDataSource> {
92         private static final long serialVersionUID = 1L;
93
94         @Override
95         ↪public JRDataSource evaluate(ReportParameters reportParameters) {
96             int masterRowNumber = reportParameters.getReportRowNumber();
97             String[] columns = new String[masterRowNumber];
98             for (int i = 1; i <= masterRowNumber; i++) {
99                 columns[i - 1] = "column" + i;
100             }
101             DRDataSource dataSource = new DRDataSource(columns);
102
103             for (int i = 1; i <= masterRowNumber; i++) {
104                 Object[] values = new Object[masterRowNumber];
105                 for (int j = 1; j <= masterRowNumber; j++) {
106                     values[j - 1] = "row" + i + "_column" + j;
107                 }
108                 dataSource.add(values);
109             }
110
111             return dataSource;
112         }
113     }
114
115     public static void main(String[] args) {
116         new DetailDynamicSubreport();
117     }
118 }

```

1.31.2 Detail Jasper Subreport



DynamicReports

<http://www.dynamicreports.org>

DetailJasperSubreport

Subreport1		
Item	Quantity	Unit price
Book	3	37,797
Book	6	71,291
Book	5	81,823
Book	6	97,376
Book	1	40,854
Subreport2		
Item	Quantity	Unit price
Book	5	45,04
Book	5	62,616
Book	3	39,726
Book	3	24,033
Book	2	37,98
Subreport3		
Item	Quantity	Unit price
Book	5	79,095
Book	3	27,424
Book	10	87,85
Book	4	65,914
Book	9	40,358

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.subreport;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.InputStream;
28 import java.math.BigDecimal;
29
30 import net.sf.dynamicreports.examples.Templates;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
32 import net.sf.dynamicreports.report.builder.component.SubreportBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.definition.ReportParameters;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37 import net.sf.jasperreports.engine.JREmptyDataSource;
38 import net.sf.jasperreports.engine.JRException;
39 import net.sf.jasperreports.engine.JasperCompileManager;
40 import net.sf.jasperreports.engine.JasperReport;
41
42 /**
43  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
44  */
45 public class DetailJasperSubreport {
46
47     public DetailJasperSubreport() {
48         build();
49     }
50
51     private void build() {
52         try {
53             SubreportBuilder subreport = cmp.
↳ subreport(getJasperSubreport())
54                                     .setDataSource(new
↳ SubreportDataSourceExpression());
55
56             report()
57                                     .title(Templates.createTitleComponent(
↳ "DetailJasperSubreport"))
58                                     .detail(
59                                         subreport,
60                                         cmp.verticalGap(20))
61                                     .pageFooter(Templates.footerComponent)
62                                     .setDataSource(createDataSource())
63                                     .show();
64         } catch (DRException e) {
65             e.printStackTrace();
66         } catch (JRException e) {
67             e.printStackTrace();
68         }
69     }
70
71     private JRDataSource createDataSource() {
72         return new JREmptyDataSource(3);
73     }
74
75     private JasperReport getJasperSubreport() throws JRException {
76         InputStream is = DetailJasperSubreport.class.getResourceAsStream(
↳ "subreport.jrxml");
77         return JasperCompileManager.compileReport(is);
78     }
79
80     private class SubreportDataSourceExpression extends AbstractSimpleExpression
↳ <JRDataSource> {
81         private static final long serialVersionUID = 1L;
82

```

(continues on next page)

(continued from previous page)

```

83         @Override
84         public JRDataSource evaluate(ReportParameters reportParameters) {
85             DRDataSource dataSource = new DRDataSource("item", "quantity",
86 ↪ "unitprice");
87             for (int i = 0; i < 5; i++) {
88                 dataSource.add("Book", (int) (Math.random() * 10) + 1,
89 ↪ new BigDecimal(Math.random() * 100 + 1));
90             }
91             return dataSource;
92         }
93         public static void main(String[] args) {
94             new DetailJasperSubreport();
95         }
96     }

```

1.31.3 Jasper Subreport

Item	Quantity	Unit price
Book	5	25.76
Book	2	13.003
Book	1	83.946
Book	2	46.186
Book	2	52.508
Book	2	12.238
Book	1	74.598
Book	8	50.296
Book	3	21.217
Book	3	80.844

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.subreport;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.InputStream;
28 import java.math.BigDecimal;
29
30 import net.sf.dynamicreports.examples.Templates;
31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34 import net.sf.jasperreports.engine.JRException;
35 import net.sf.jasperreports.engine.JasperCompileManager;
36 import net.sf.jasperreports.engine.JasperReport;
37 import net.sf.jasperreports.engine.util.JRLoader;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class JasperSubreport {
43
44     public JasperSubreport() {
45         build();
46     }
47
48     private void build() {
49         try {
50             report()
51                 .setTemplate(Templates.reportTemplate)
52                 .title(
53                     Templates.
54                     ↳ createTitleComponent("JasperSubreport"),
55                     cmp.
56                     ↳ subreport(getJasperTitleSubreport()))
57                 .columns(
58                     col.column("Item", "item",
59                     ↳ type.stringType()),
60                     col.column("Quantity",
61                     ↳ "quantity", type.integerType()),
62                     col.column("Unit price",
63                     ↳ "unitprice", type.bigDecimalType()))
64                 .pageFooter(Templates.footerComponent)
65                 .summary(cmp.
66                     ↳ subreport(getJasperSummarySubreport()))
67                 .setDataSource(createDataSource())

```

(continues on next page)


(continued from previous page)

```

62         .show();
63     } catch (DRException e) {
64         e.printStackTrace();
65     } catch (JRException e) {
66         e.printStackTrace();
67     }
68 }
69
70 private JRDataSource createDataSource() {
71     DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");
72     for (int i = 0; i < 10; i++) {
73         dataSource.add("Book", (int) (Math.random() * 10) + 1, new
↪ BigDecimal(Math.random() * 100 + 1));
74     }
75     return dataSource;
76 }
77
78 private JasperReport getJasperTitleSubreport() throws JRException {
79     InputStream is = JasperSubreport.class.getResourceAsStream(
↪ "titlesubreport.jrxml");
80     return JasperCompileManager.compileReport(is);
81 }
82
83 private JasperReport getJasperSummarySubreport() throws JRException {
84     InputStream is = JasperSubreport.class.getResourceAsStream(
↪ "summarysubreport.jasper");
85     return (JasperReport) JRLoader.loadObject(is);
86 }
87
88 public static void main(String[] args) {
89     new JasperSubreport();
90 }
91 }

```

1.31.4 Title Subreport



DynamicReports

<http://www.dynamicreports.org>

TitleSubreport

Subreport in title		
Item	Quantity	Unit price
Book	2	40.616
Book	8	12.87
Book	5	14.806
Book	7	70.941
Book	7	61.123
Book	2	100.718
Book	7	40.14
Book	7	61.167
Book	8	97.213
Book	8	20.503

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.subreport;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.datasource.DRDataSource;
32 import net.sf.dynamicreports.report.exception.DRException;
33 import net.sf.jasperreports.engine.JRDataSource;
34
35 /**
36  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
37  */
38 public class TitleSubreport {
39
40     public TitleSubreport() {
41         build();
42     }
43
44     private void build() {
45         try {
46             report()
47                 .title(
48                     Templates.
49                     ↳createTitleComponent("TitleSubreport"),
50                     cmp.
51                     ↳subreport(createSubreport()))
52                 .pageFooter(Templates.footerComponent)
53                 .show();
54         } catch (DRException e) {
55             e.printStackTrace();
56         }
57
58     private JasperReportBuilder createSubreport() {
59         JasperReportBuilder report = report();
60         report
61             .setTemplate(Templates.reportTemplate)
62             .title(cmp.text("Subreport in title").
63                 ↳setStyle(Templates.bold12CenteredStyle))
64             .columns(
65                 col.column("Item", "item", type.
66                     ↳stringType()),
67                 col.column("Quantity", "quantity",
68                     ↳type.integerType()),
69                 col.column("Unit price", "unitprice",
70                     ↳type.bigDecimalType()))
71             .setDataSource(createSubreportDataSource());
72
73         return report;
74     }
75
76     private JRDataSource createSubreportDataSource() {
77         DRDataSource dataSource = new DRDataSource("item", "quantity",
78             ↳"unitprice");
79         for (int i = 0; i < 10; i++) {
80             dataSource.add("Book", (int) (Math.random() * 10) + 1, new
81                 ↳BigDecimal(Math.random() * 100 + 1));
82         }
83         return dataSource;
84     }
85
86     public static void main(String[] args) {

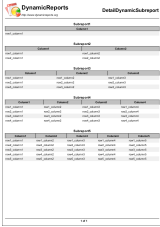
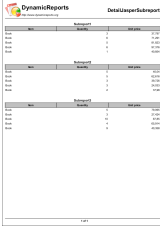

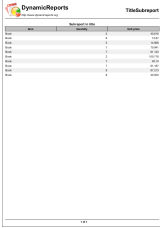
```

(continues on next page)

(continued from previous page)


```
80         new TitleSubreport ();
81     }
82 }
```

Table 16: Subreport Examples

 DetailDynamicSubreport	 DetailJasperSubreport	 JasperSubreport
 TitleSubreport		

1.32 Subtotal

1.32.1 Aggregation Subtotal

 DynamicReports http://www.dynamicreports.org		AggregationSubtotal	
Item	Order Date	Quantity	Unit price
Tablet	01/01/2010	3	110.00
Tablet	02/01/2010	1	150.00
Laptop	02/01/2010	3	300.00
Smartphone	04/01/2010	8	90.00
Smartphone	05/01/2010	6	120.00
unit price sum	count	sum	sum
770.00	5	21	770.00
count	distinct count	avg	avg
5	4	4.2	154
distinct count	min value	count	count
3	01/01/2010	5	5
	max value	distinct count	distinct count
	05/01/2010	4	5
		min value	first value
		1	110.00
		max value	standard deviation
		8	75.5
			variance
			5.704
1 of 1			

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.subtotal;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28  import java.util.Calendar;
29  import java.util.Date;
30
31  import net.sf.dynamicreports.examples.Templates;
32  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33  import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
34  import net.sf.dynamicreports.report.datasource.DRDataSource;
35  import net.sf.dynamicreports.report.exception.DRException;
36  import net.sf.jasperreports.engine.JRDataSource;
37
38  /**
39   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40   */
41  public class AggregationSubtotalReport {
42
43      public AggregationSubtotalReport() {
44          build();
45      }
46
47      private void build() {
48          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49  ↪ type.stringType());
50          TextColumnBuilder<Date> orderDateColumn = col.column("Order Date",
51  ↪ "orderdate", type.dateType());
52          TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
53  ↪ "quantity", type.integerType());
54          TextColumnBuilder<BigDecimal> unitPriceColumn = col.column("Unit price
55  ↪ ", "unitprice", type.bigDecimalType());
56
57          AggregationSubtotalBuilder<Long> itemCount = sbt.count(itemColumn)

```

(continues on next page)

(continued from previous page)

```

54         .setLabel("count");
55         AggregationSubtotalBuilder<Long> itemDistCount = sbt.
↪distinctCount(itemColumn)
56         .setLabel("distinct count");
57         AggregationSubtotalBuilder<BigDecimal> itemSum = sbt.
↪sum(unitPriceColumn)
58         .setLabel("unit price sum")
59         .setShowInColumn(itemColumn);
60
61         AggregationSubtotalBuilder<Long> orderDateCount = sbt.
↪count(orderDateColumn)
62         .setLabel("count");
63         AggregationSubtotalBuilder<Long> orderDateDistCount = sbt.
↪distinctCount(orderDateColumn)
64         .setLabel("distinct count");
65         AggregationSubtotalBuilder<Date> orderDateMin = sbt.
↪min(orderDateColumn)
66         .setLabel("min value");
67         AggregationSubtotalBuilder<Date> orderDateMax = sbt.
↪max(orderDateColumn)
68         .setLabel("max value");
69
70         AggregationSubtotalBuilder<Integer> quantitySum = sbt.
↪sum(quantityColumn)
71         .setLabel("sum");
72         AggregationSubtotalBuilder<Number> quantityAvg = sbt.
↪avg(quantityColumn)
73         .setLabel("avg");
74         AggregationSubtotalBuilder<Long> quantityCount = sbt.
↪count(quantityColumn)
75         .setLabel("count");
76         AggregationSubtotalBuilder<Long> quantityDistCount = sbt.
↪distinctCount(quantityColumn)
77         .setLabel("distinct count");
78         AggregationSubtotalBuilder<Integer> quantityMin = sbt.
↪min(quantityColumn)
79         .setLabel("min value");
80         AggregationSubtotalBuilder<Integer> quantityMax = sbt.
↪max(quantityColumn)
81         .setLabel("max value");
82
83         AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.
↪sum(unitPriceColumn)
84         .setLabel("sum");
85         AggregationSubtotalBuilder<Number> unitPriceAvg = sbt.
↪avg(unitPriceColumn)
86         .setLabel("avg");
87         AggregationSubtotalBuilder<Long> unitPriceCount = sbt.
↪count(unitPriceColumn)
88         .setLabel("count");
89         AggregationSubtotalBuilder<Long> unitPriceDistCount = sbt.
↪distinctCount(unitPriceColumn)
90         .setLabel("distinct count");
91         AggregationSubtotalBuilder<BigDecimal> unitPriceFirst = sbt.
↪first(unitPriceColumn)
92         .setLabel("first value");
93         AggregationSubtotalBuilder<Number> unitPriceStdDev = sbt.
↪stdDev(unitPriceColumn)

```

(continues on next page)


(continued from previous page)

```

94         .setLabel("standard deviation");
95         AggregationSubtotalBuilder<Number> unitPriceVar = sbt.
↪var(unitPriceColumn)
96         .setLabel("variance");
97
98         try {
99             report()
100                 .setTemplate(Templates.reportTemplate)
101                 .columns(
102                     itemColumn, orderDateColumn, ↪
↪quantityColumn, unitPriceColumn)
103                 .subtotalsAtSummary(
104                     itemSum, itemCount, ↪
↪itemDistCount,
105                     orderDateCount, ↪
↪orderDateDistCount, orderDateMin, orderDateMax,
106                     quantitySum, quantityAvg, ↪
↪quantityCount, quantityDistCount, quantityMin, quantityMax,
107                     unitPriceSum, unitPriceAvg, ↪
↪unitPriceCount, unitPriceDistCount, unitPriceFirst, unitPriceStdDev, unitPriceVar)
108                 .title(Templates.createTitleComponent(
↪"AggregationSubtotal"))
109                 .pageFooter(Templates.footerComponent)
110                 .setDataSource(createDataSource())
111                 .show();
112         } catch (DRException e) {
113             e.printStackTrace();
114         }
115     }
116
117     private JRDataSource createDataSource() {
118         DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪"quantity", "unitprice");
119         dataSource.add("Tablet", toDate(2010, 1, 1), 3, new BigDecimal(110));
120         dataSource.add("Tablet", toDate(2010, 2, 1), 1, new BigDecimal(150));
121         dataSource.add("Laptop", toDate(2010, 2, 1), 3, new BigDecimal(300));
122         dataSource.add("Smartphone", toDate(2010, 4, 1), 8, new ↪
↪BigDecimal(90));
123         dataSource.add("Smartphone", toDate(2010, 5, 1), 6, new ↪
↪BigDecimal(120));
124         return dataSource;
125     }
126
127     private Date toDate(int year, int month, int day) {
128         Calendar c = Calendar.getInstance();
129         c.set(Calendar.YEAR, year);
130         c.set(Calendar.MONTH, month - 1);
131         c.set(Calendar.DAY_OF_MONTH, day);
132         return c.getTime();
133     }
134
135     public static void main(String[] args) {
136         new AggregationSubtotalReport();
137     }
138 }

```

1.32.2 Custom Subtotal

DynamicReports		CustomSubtotal		
		http://www.dynamicreports.org		
Item	Quantity	Price	Price / Quantity	
Tablet	3	330.00	110.00	
Tablet	1	150.00	150.00	
Laptop	3	900.00	300.00	
Smartphone	8	720.00	90.00	
Smartphone	6	720.00	120.00	
sum		sum	sum(price) / sum(quantity)	
		21	2,820.00	
			134.29	

1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.subtotal;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32 import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
33 import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.definition.ReportParameters;
36 import net.sf.dynamicreports.report.exception.DRException;
37 import net.sf.jasperreports.engine.JRDataSource;
38
39 /**
40  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
41  */
42 public class CustomSubtotalReport {
43     private AggregationSubtotalBuilder<Integer> quantitySum;
44     private AggregationSubtotalBuilder<BigDecimal> priceSum;
45
46     public CustomSubtotalReport() {
47         build();
48     }
49
50     private void build() {
51         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↪ type.stringType());
52         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
↪ "quantity", type.integerType());
53         TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price",
↪ type.bigDecimalType());
54         TextColumnBuilder<BigDecimal> unitPriceColumn = priceColumn.divide(2,
↪ quantityColumn)
55             .setTitle("Price / Quantity");
56
57         quantitySum = sbt.sum(quantityColumn)
58             .setLabel("sum");
59         priceSum = sbt.sum(priceColumn)
60             .setLabel("sum");
61         CustomSubtotalBuilder<BigDecimal> unitPriceSbt = sbt.customValue(new
↪ UnitPriceSubtotal(), unitPriceColumn)
62             .setLabel("sum(price) / sum(quantity)")
63             .setDataType(type.bigDecimalType());
64
65         try {
66             report()
67                 .setTemplate(Templates.reportTemplate)
68                 .columns(
69                     itemColumn, quantityColumn,
↪ priceColumn, unitPriceColumn)
70                 .subtotalsAtSummary(
71                     quantitySum, priceSum,
↪ unitPriceSbt)
72                 .title(Templates.createTitleComponent(
↪ "CustomSubtotal"))
73                 .pageFooter(Templates.footerComponent)
74                 .setDataSource(createDataSource())
75                 .show();
76         } catch (DRException e) {
77             e.printStackTrace();
78         }
79     }

```

(continues on next page)


(continued from previous page)

```

80
81     private class UnitPriceSubtotal extends AbstractSimpleExpression<BigDecimal> {
82         private static final long serialVersionUID = 1L;
83
84         @Override
85         public BigDecimal evaluate(ReportParameters reportParameters) {
86             Integer quantitySumValue = reportParameters.
↪getValue(quantitySum);
87             BigDecimal priceSumValue = reportParameters.
↪getValue(priceSum);
88             return priceSumValue.divide(new BigDecimal(quantitySumValue),
↪2, BigDecimal.ROUND_HALF_UP);
89         }
90     }
91
92     private JRDataSource createDataSource() {
93         DRDataSource dataSource = new DRDataSource("item", "quantity", "price
↪");
94         dataSource.add("Tablet", 3, new BigDecimal(330));
95         dataSource.add("Tablet", 1, new BigDecimal(150));
96         dataSource.add("Laptop", 3, new BigDecimal(900));
97         dataSource.add("Smartphone", 8, new BigDecimal(720));
98         dataSource.add("Smartphone", 6, new BigDecimal(720));
99         return dataSource;
100     }
101
102     public static void main(String[] args) {
103         new CustomSubtotalReport();
104     }
105 }

```

1.32.3 Custom Text Subtotal

DynamicReports			CustomTextSubtotal
 http://www.dynamicreports.org			
Item	Quantity	Price	
USA			
Tablet	4	600.00	
Tablet	3	570.00	
Laptop	2	500.00	
Laptop	1	420.00	
sum(quantity) = 10, sum(price) = 2,090.00, sum(price) / sum(quantity) = 209.00			
Canada			
Tablet	6	720.00	
Tablet	2	360.00	
Laptop	3	900.00	
Laptop	2	780.00	
sum(quantity) = 13, sum(price) = 2,760.00, sum(price) / sum(quantity) = 212.31			
sum(quantity) = 23, sum(price) = 4,850.00, sum(price) / sum(quantity) = 210.87			
1 of 1			

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.subtotal;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.math.BigDecimal;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.VariableBuilder;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
34 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
35 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
36 import net.sf.dynamicreports.report.constant.Calculation;
37 import net.sf.dynamicreports.report.constant.Evaluation;
38 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
39 import net.sf.dynamicreports.report.datasource.DRDataSource;
40 import net.sf.dynamicreports.report.definition.ReportParameters;
41 import net.sf.dynamicreports.report.exception.DRException;
42 import net.sf.jasperreports.engine.JRDataSource;
43
44 /**
45  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
46  */
47 public class CustomTextSubtotalReport {
48
49     public CustomTextSubtotalReport() {
50         build();
51     }
52
53     private void build() {
54         TextColumnBuilder<String> countryColumn = col.column("Country",
55 ↪ "country", type.stringType());
56         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
57 ↪ type.stringType());
58         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
59 ↪ "quantity", type.integerType());
60         TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
61 ↪ ", type.bigDecimalType());
62
63         ColumnGroupBuilder countryGroup = grp.group(countryColumn);
64
65         VariableBuilder<Integer> quantitySum = variable(quantityColumn,
66 ↪ Calculation.SUM);
67         VariableBuilder<BigDecimal> priceSum = variable(priceColumn,
68 ↪ Calculation.SUM);
69
70         VariableBuilder<Integer> quantityGrpSum = variable(quantityColumn,
71 ↪ Calculation.SUM);
72         quantityGrpSum.setResetGroup(countryGroup);
73         VariableBuilder<BigDecimal> priceGrpSum = variable(priceColumn,
74 ↪ Calculation.SUM);
75         priceGrpSum.setResetType(Evaluation.FIRST_GROUP);
76
77         StyleBuilder subtotalStyle = stl.style()
78             .bold()
79             .setTopBorder(stl.pen1Point())
80             .setHorizontalTextAlignment(HorizontalTextAlignment.
81 ↪ CENTER);
82
83         TextFieldBuilder<String> summarySbt = cmp.text(new
84 ↪ CustomTextSubtotal(quantitySum, priceSum))
85             .setStyle(subtotalStyle);
86
87         TextFieldBuilder<String> groupSbt = cmp.text(new
88 ↪ CustomTextSubtotal(quantityGrpSum, priceGrpSum))

```

(continues on next page)

(continued from previous page)

```

78         .setStyle(subtotalStyle);
79
80         countryGroup.footer(groupSbt);
81
82         try {
83             report()
84                 .setTemplate(Templates.reportTemplate)
85                 .variables(
86                     ↪ quantitySum, priceSum, ↪
87                     ↪ quantityGrpSum, priceGrpSum)
88                 .columns(
89                     ↪ quantityColumn, priceColumn)
90                 .groupBy(
91                     ↪ countryGroup)
92                 .summary(
93                     ↪ summarySbt)
94                 .title(Templates.createTitleComponent(
95                     ↪ "CustomTextSubtotal"))
96                 .pageFooter(Templates.footerComponent)
97                 .setDataSource(createDataSource())
98                 .show();
99         } catch (DRException e) {
100             e.printStackTrace();
101         }
102     }
103
104     private class CustomTextSubtotal extends AbstractSimpleExpression<String> {
105         private static final long serialVersionUID = 1L;
106
107         private VariableBuilder<Integer> quantitySum;
108         private VariableBuilder<BigDecimal> priceSum;
109
110         public CustomTextSubtotal(VariableBuilder<Integer> quantitySum, ↪
111             ↪ VariableBuilder<BigDecimal> priceSum) {
112             this.quantitySum = quantitySum;
113             this.priceSum = priceSum;
114         }
115
116         @Override
117         public String evaluate(ReportParameters reportParameters) {
118             Integer quantitySumValue = reportParameters.
119             ↪ getValue(quantitySum);
120             BigDecimal priceSumValue = reportParameters.
121             ↪ getValue(priceSum);
122             BigDecimal unitPriceSbt = priceSumValue.divide(new ↪
123             ↪ BigDecimal(quantitySumValue), 2, BigDecimal.ROUND_HALF_UP);
124             return "sum(quantity) = " + type.integerType().
125             ↪ valueToString(quantitySum, reportParameters) + ", " +
126             ↪ "sum(price) = " + type.bigDecimalType().
127             ↪ valueToString(priceSum, reportParameters) + ", " +
128             ↪ "sum(price) / sum(quantity) = " + type.
129             ↪ bigDecimalType().valueToString(unitPriceSbt, reportParameters.getLocale());
130         }
131
132         private JRDataSource createDataSource() {

```

(continues on next page)


(continued from previous page)

```

125         DRDataSource dataSource = new DRDataSource("country", "item",
126             ↪ "quantity", "price");
127         dataSource.add("USA", "Tablet", 4, new BigDecimal(600));
128         dataSource.add("USA", "Tablet", 3, new BigDecimal(570));
129         dataSource.add("USA", "Laptop", 2, new BigDecimal(500));
130         dataSource.add("USA", "Laptop", 1, new BigDecimal(420));
131         dataSource.add("Canada", "Tablet", 6, new BigDecimal(720));
132         dataSource.add("Canada", "Tablet", 2, new BigDecimal(360));
133         dataSource.add("Canada", "Laptop", 3, new BigDecimal(900));
134         dataSource.add("Canada", "Laptop", 2, new BigDecimal(780));
135         return dataSource;
136     }
137     public static void main(String[] args) {
138         new CustomTextSubtotalReport();
139     }
140 }

```

1.32.4 Field Subtotal

DynamicReports		FieldSubtotal
 http://www.dynamicreports.org		
Item	Order Date	
Tablet	01/01/2010	
Tablet	02/01/2010	
Laptop	02/01/2010	
Smartphone	04/01/2010	
Smartphone	05/01/2010	
quantity sum		21
unitPrice sum		770.00
		1 of 1

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify

```

(continues on next page)

(continued from previous page)

```

10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.subtotal;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.subtotal.AggregationSubtotalBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class FieldSubtotalReport {
42
43     public FieldSubtotalReport() {
44         build();
45     }
46
47     private void build() {
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49 ↪ type.stringType());
50         TextColumnBuilder<Date> orderDateColumn = col.column("Order Date",
51 ↪ "orderdate", type.dateType());
52
53         AggregationSubtotalBuilder<Integer> quantitySum = sbt.sum("quantity",
54 ↪ Integer.class, itemColumn)
55             .setLabel("quantity sum");
56         AggregationSubtotalBuilder<BigDecimal> unitPriceSum = sbt.sum(
57 ↪ "unitprice", BigDecimal.class, itemColumn)
58             .setLabel("unitPrice sum");
59
60         try {
61             report()
62                 .setTemplate(Templates.reportTemplate)
63                 .columns(
64                     itemColumn, orderDateColumn)
65                 .subtotalsAtSummary(
66                     quantitySum, unitPriceSum)

```

(continues on next page)


(continued from previous page)

```

63         .title(Templates.createTitleComponent(
↪ "FieldSubtotal"))
64         .pageFooter(Templates.footerComponent)
65         .setDataSource(createDataSource())
66         .show();
67     } catch (DRException e) {
68         e.printStackTrace();
69     }
70 }
71
72 private JRDataSource createDataSource() {
73     DRDataSource dataSource = new DRDataSource("item", "orderdate",
↪ "quantity", "unitprice");
74     dataSource.add("Tablet", toDate(2010, 1, 1), 3, new BigDecimal(110));
75     dataSource.add("Tablet", toDate(2010, 2, 1), 1, new BigDecimal(150));
76     dataSource.add("Laptop", toDate(2010, 2, 1), 3, new BigDecimal(300));
77     dataSource.add("Smartphone", toDate(2010, 4, 1), 8, new
↪ BigDecimal(90));
78     dataSource.add("Smartphone", toDate(2010, 5, 1), 6, new
↪ BigDecimal(120));
79     return dataSource;
80 }
81
82 private Date toDate(int year, int month, int day) {
83     Calendar c = Calendar.getInstance();
84     c.set(Calendar.YEAR, year);
85     c.set(Calendar.MONTH, month - 1);
86     c.set(Calendar.DAY_OF_MONTH, day);
87     return c.getTime();
88 }
89
90 public static void main(String[] args) {
91     new FieldSubtotalReport();
92 }
93 }

```

1.32.5 Group Subtotal

DynamicReports		GroupSubtotal
		http://www.dynamicreports.org
Item	Quantity	Price
USA		
Tablet	4	150.00
Tablet	3	190.00
Laptop	2	250.00
Laptop	1	420.00
	10	1,010.00
Canada		
Tablet	6	120.00
Tablet	2	180.00
Laptop	3	300.00
Laptop	2	390.00
	13	990.00
Total	23	2,000.00

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.subtotal;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
32 import net.sf.dynamicreports.report.datasource.DRDataSource;
33 import net.sf.dynamicreports.report.exception.DRException;
34 import net.sf.jasperreports.engine.JRDataSource;
35
36 /**
37  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
38  */
39 public class GroupSubtotalReport {
40
41     public GroupSubtotalReport() {
42         build();
43     }
44
45     private void build() {
46         TextColumnBuilder<String> countryColumn = col.column("Country",
47 ↪ "country", type.stringType());
48         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
49 ↪ type.stringType());
50         TextColumnBuilder<Integer> quantityColumn = col.column("Quantity",
51 ↪ "quantity", type.integerType());
52         TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price
53 ↪ ", type.bigDecimalType());
54
55         ColumnGroupBuilder countryGroup = grp.group(countryColumn);
56
57         try {
58             report()
59                 .setTemplate(Templates.reportTemplate)
60                 .columns(
61                     countryColumn, itemColumn,
62 ↪ quantityColumn, priceColumn)
63                 .groupBy(
64                     countryGroup)
65                 .subtotalsAtFirstGroupFooter(
66                     sbt.sum(quantityColumn))
67                 .subtotalsAtGroupFooter(
68                     countryGroup, sbt.
69 ↪ sum(priceColumn))
70                 .subtotalsAtSummary(
71                     sbt.text("Total", itemColumn),
72 ↪ sbt.sum(quantityColumn), sbt.sum(priceColumn))
73                 .title(Templates.createTitleComponent(
74 ↪ "GroupSubtotal"))
75                 .pageFooter(Templates.footerComponent)
76                 .setDataSource(createDataSource())
77                 .show();
78         } catch (DRException e) {
79             e.printStackTrace();
80         }
81
82     private JRDataSource createDataSource() {
83         DRDataSource dataSource = new DRDataSource("country", "item",
84 ↪ "quantity", "price");
85         dataSource.add("USA", "Tablet", 4, new BigDecimal(150));
86         dataSource.add("USA", "Tablet", 3, new BigDecimal(190));
87     }
88 }

```

(continues on next page)

(continued from previous page)

```

79      dataSource.add("USA", "Laptop", 2, new BigDecimal(250));
80      dataSource.add("USA", "Laptop", 1, new BigDecimal(420));
81      dataSource.add("Canada", "Tablet", 6, new BigDecimal(120));
82      dataSource.add("Canada", "Tablet", 2, new BigDecimal(180));
83      dataSource.add("Canada", "Laptop", 3, new BigDecimal(300));
84      dataSource.add("Canada", "Laptop", 2, new BigDecimal(390));
85      return dataSource;
86  }
87
88  public static void main(String[] args) {
89      new GroupSubtotalReport();
90  }
91  }

```

1.32.6 Percentage Subtotal

DynamicReports		PercentageSubtotal
		http://www.dynamicreports.org
Price		
USA		
Tablet		
	150.00	
	190.00	
Item in country price[%]		33.66%
Item price[%]		17.00%
Laptop		
	250.00	
	420.00	
Item in country price[%]		66.34%
Item price[%]		33.50%
country price [%]		50.50%
Canada		
Tablet		
	120.00	
	180.00	
Item in country price[%]		30.30%
Item price[%]		15.00%
Laptop		
	300.00	
	390.00	
Item in country price[%]		69.70%
Item price[%]		34.50%
country price [%]		49.50%
1 of 1		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *

```

(continues on next page)

(continued from previous page)

```

14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.subtotal;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.awt.Color;
28  import java.math.BigDecimal;
29
30  import net.sf.dynamicreports.examples.Templates;
31  import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
32  import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
33  import net.sf.dynamicreports.report.builder.style.StyleBuilder;
34  import net.sf.dynamicreports.report.builder.subtotal.PercentageSubtotalBuilder;
35  import net.sf.dynamicreports.report.constant.PercentageTotalType;
36  import net.sf.dynamicreports.report.datasource.DRDataSource;
37  import net.sf.dynamicreports.report.exception.DRException;
38  import net.sf.jasperreports.engine.JRDataSource;
39
40  /**
41   * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42   */
43  public class PercentageSubtotalReport {
44
45      public PercentageSubtotalReport() {
46          build();
47      }
48
49      private void build() {
50          TextColumnBuilder<String> countryColumn = col.column("Country",
51 ↪ "country", type.stringType());
52          TextColumnBuilder<String> itemColumn = col.column("Item", "item",
53 ↪ type.stringType());
54          TextColumnBuilder<BigDecimal> priceColumn = col.column("Price", "price",
55 ↪ type.bigDecimalType());
56
57          ColumnGroupBuilder countryGroup = grp.group(countryColumn);
58          ColumnGroupBuilder itemGroup = grp.group(itemColumn);
59
60          StyleBuilder countryLabelStyle = stl.style()
61              .setForegroundColor(Color.RED);
62          StyleBuilder countryStyle = stl.style(countryLabelStyle)
63              .setTopBorder(stl.pen1Point());
64          StyleBuilder itemInCountryLabelStyle = stl.style()
65              .setForegroundColor(Color.GREEN);
66          StyleBuilder itemInCountryStyle = stl.style(itemInCountryLabelStyle)
67              .setTopBorder(stl.pen1Point());
68          StyleBuilder itemLabelStyle = stl.style()
69              .setForegroundColor(Color.BLUE);
70          StyleBuilder itemStyle = stl.style(itemLabelStyle)

```

(continues on next page)

(continued from previous page)

```

68         .setTopBorder(stl.pen1Point());
69
70         PercentageSubtotalBuilder countryPercentage = sbt.
↪percentage(priceColumn)
71             .setLabel("country price [%]")
72             .setLabelStyle(countryLabelStyle)
73             .setStyle(countryStyle);
74
75         PercentageSubtotalBuilder itemInCountryPercentage = sbt.
↪percentage(priceColumn)
76             .setLabel("item in country price[%]")
77             .setLabelStyle(itemInCountryLabelStyle)
78             .setStyle(itemInCountryStyle);
79
80         PercentageSubtotalBuilder itemPercentage = sbt.percentage(priceColumn)
81             .setLabel("item price[%]")
82             .setLabelStyle(itemLabelStyle)
83             .setStyle(itemStyle)
84             .setTotalType(PercentageTotalType.REPORT);
85
86         try {
87             report()
88                 .setTemplate(Templates.reportTemplate)
89                 .columns(
90                     countryColumn, itemColumn, ↪
↪priceColumn)
91                 .groupBy(
92                     countryGroup, itemGroup)
93                 .subtotalsOfPercentageAtGroupFooter(
94                     countryGroup, ↪
↪countryPercentage)
95                 .subtotalsOfPercentageAtGroupFooter(
96                     itemGroup, ↪
↪itemInCountryPercentage, itemPercentage)
97                 .title(Templates.createTitleComponent(
↪"PercentageSubtotal"))
98                 .pageFooter(Templates.footerComponent)
99                 .setDataSource(createDataSource())
100                 .show();
101         } catch (DRException e) {
102             e.printStackTrace();
103         }
104     }
105
106     private JRDataSource createDataSource() {
107         DRDataSource dataSource = new DRDataSource("country", "item", "price
↪");
108         dataSource.add("USA", "Tablet", new BigDecimal(150));
109         dataSource.add("USA", "Tablet", new BigDecimal(190));
110         dataSource.add("USA", "Laptop", new BigDecimal(250));
111         dataSource.add("USA", "Laptop", new BigDecimal(420));
112         dataSource.add("Canada", "Tablet", new BigDecimal(120));
113         dataSource.add("Canada", "Tablet", new BigDecimal(180));
114         dataSource.add("Canada", "Laptop", new BigDecimal(300));
115         dataSource.add("Canada", "Laptop", new BigDecimal(390));
116         return dataSource;
117     }

```

(continues on next page)

(continued from previous page)

```

118
119     public static void main(String[] args) {
120         new PercentageSubtotalReport();
121     }
122 }

```

1.32.7 Variable Subtotal

DynamicReports		VariableSubtotal
 http://www.dynamicreports.org		
Item		
Tablet		
Tablet		
Laptop		
Smartphone		
Smartphone		
sum(price) / sum(quantity)		134.29
1 of 1		

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */

```

(continues on next page)

(continued from previous page)

```

22
23 package net.sf.dynamicreports.examples.subtotal;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.base.expression.AbstractSimpleExpression;
31 import net.sf.dynamicreports.report.builder.VariableBuilder;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.subtotal.CustomSubtotalBuilder;
34 import net.sf.dynamicreports.report.constant.Calculation;
35 import net.sf.dynamicreports.report.datasource.DRDataSource;
36 import net.sf.dynamicreports.report.definition.ReportParameters;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class VariableSubtotalReport {
44     private VariableBuilder<Integer> quantitySum;
45     private VariableBuilder<BigDecimal> priceSum;
46
47     public VariableSubtotalReport() {
48         build();
49     }
50
51     private void build() {
52         quantitySum = variable("quantity", Integer.class, Calculation.SUM);
53         priceSum = variable("price", BigDecimal.class, Calculation.SUM);
54
55         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
↳ type.stringType());
56
57         CustomSubtotalBuilder<BigDecimal> unitPriceSbt = sbt.customValue(new
↳ UnitPriceSubtotal(), itemColumn)
58             .setLabel("sum(price) / sum(quantity)")
59             .setDataType(type.bigDecimalType());
60
61         try {
62             report()
63                 .setTemplate(Templates.reportTemplate)
64                 .variables(
65                     quantitySum, priceSum)
66                 .columns(
67                     itemColumn)
68                 .subtotalsAtSummary(
69                     unitPriceSbt)
70                 .title(Templates.createTitleComponent(
↳ "VariableSubtotal"))
71                 .pageFooter(Templates.footerComponent)
72                 .setDataSource(createDataSource())
73                 .show();
74         } catch (DRException e) {
75             e.printStackTrace();

```

(continues on next page)

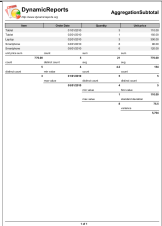
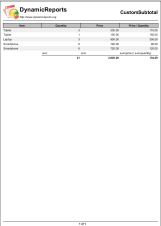
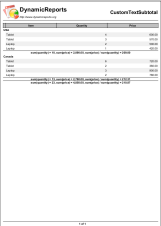
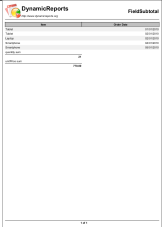
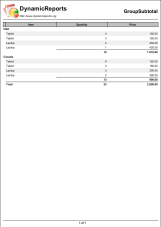


(continued from previous page)

```

76         }
77     }
78
79     private class UnitPriceSubtotal extends AbstractSimpleExpression<BigDecimal> {
80         private static final long serialVersionUID = 1L;
81
82         @Override
83         public BigDecimal evaluate(ReportParameters reportParameters) {
84             Integer quantitySumValue = reportParameters.
↪getValue(quantitySum);
85             BigDecimal priceSumValue = reportParameters.
↪getValue(priceSum);
86             return priceSumValue.divide(new BigDecimal(quantitySumValue),
↪2, BigDecimal.ROUND_HALF_UP);
87         }
88     }
89
90     private JRDataSource createDataSource() {
91         DRDataSource dataSource = new DRDataSource("item", "quantity", "price
↪");
92         dataSource.add("Tablet", 3, new BigDecimal(330));
93         dataSource.add("Tablet", 1, new BigDecimal(150));
94         dataSource.add("Laptop", 3, new BigDecimal(900));
95         dataSource.add("Smartphone", 8, new BigDecimal(720));
96         dataSource.add("Smartphone", 6, new BigDecimal(720));
97         return dataSource;
98     }
99
100     public static void main(String[] args) {
101         new VariableSubtotalReport();
102     }
103 }

```

Table 17: Subtotal Examples

		
AggregationSubtotalReport	CustomSubtotalReport	CustomTextSubtotalReport
		
FieldSubtotalReport	GroupSubtotalReport	PercentageSubtotalReport
		
VariableSubtotalReport		

1.33 Table Of Contents

1.33.1 Custom Table Of Contents

Table of contents	DynamicReports	CustomTableOfContents
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11

```
1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
```

(continues on next page)

(continued from previous page)

```

12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.tableofcontents;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.awt.Color;
28 import java.math.BigDecimal;
29 import java.util.Calendar;
30 import java.util.Date;
31
32 import net.sf.dynamicreports.examples.Templates;
33 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
34 import net.sf.dynamicreports.report.builder.component.ComponentBuilder;
35 import net.sf.dynamicreports.report.builder.component.VerticalListBuilder;
36 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
37 import net.sf.dynamicreports.report.builder.tableofcontents.TableOfContentsCustomizer;
38 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
39 import net.sf.dynamicreports.report.datasource.DRDataSource;
40 import net.sf.dynamicreports.report.exception.DRException;
41 import net.sf.jasperreports.engine.JRDataSource;
42
43 /**
44  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
45  */
46 public class CustomTableOfContentsReport {
47
48     public CustomTableOfContentsReport() {
49         build();
50     }
51
52     private void build() {
53         TextColumnBuilder<String> countryColumn = col.column("Country",
54 ↪ "country", type.stringType());
55         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
56 ↪ type.stringType());
57
58         StyleBuilder titleTocStyle = stl.style()
59             .setForegroundColor(Color.BLUE)
60             .setFontSize(18)
61             .bold()
62             .setHorizontalTextAlignment(HorizontalTextAlignment.
63 ↪ CENTER);
64
65         StyleBuilder headingToc0Style = stl.style(Templates.rootStyle)
66             .setFontSize(12)
67             .bold();
68
69         StyleBuilder headingToc1Style = stl.style(Templates.rootStyle)
70             .italic();

```

(continues on next page)

(continued from previous page)

```

66
67         CustomTableOfContentsCustomizer tableOfContentsCustomizer = new
↪CustomTableOfContentsCustomizer();
68         tableOfContentsCustomizer.setTitleStyle(titleTocStyle);
69         tableOfContentsCustomizer.setHeadingStyle(0, headingToc0Style);
70         tableOfContentsCustomizer.setHeadingStyle(1, headingToc1Style);
71         tableOfContentsCustomizer.setTextFixedWidth(100);
72         tableOfContentsCustomizer.setPageIndexFixedWidth(30);
73
74         try {
75             report()
76
77                 .setTemplate(Templates.reportTemplate)
78                 .tableOfContents(tableOfContentsCustomizer)
79                 .columns(
80                     countryColumn,
81                     itemColumn,
82                     col.column("Order date",
↪"orderdate", type.dateType()),
83                     col.column("Quantity",
↪"quantity", type.integerType()),
84                     col.column("Unit price",
↪"unitprice", type.bigDecimalType()))
85                 .groupBy(countryColumn, itemColumn)
86                 .title(Templates.createTitleComponent(
↪"CustomTableOfContents"))
87                 .pageFooter(Templates.footerComponent)
88                 .setDataSource(createDataSource())
89                 .show();
90         } catch (DRException e) {
91             e.printStackTrace();
92         }
93
94         private JRDataSource createDataSource() {
95             String[] countries = new String[] { "USA", "Canada", "Mexico" };
96             String[] items = new String[] { "Book", "Notebook", "PDA" };
97             DRDataSource dataSource = new DRDataSource("country", "item",
↪"orderdate", "quantity", "unitprice");
98             for (String country : countries) {
99                 for (String item : items) {
100                     for (int i = 0; i < 8; i++) {
101                         dataSource.add(country, item, toDate(2010, 1,
↪(int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
102                             new BigDecimal(Math.random()
↪* 100 + 1));
103                     }
104                 }
105             }
106             return dataSource;
107         }
108
109         private Date toDate(int year, int month, int day) {
110             Calendar c = Calendar.getInstance();
111             c.set(Calendar.YEAR, year);
112             c.set(Calendar.MONTH, month - 1);
113             c.set(Calendar.DAY_OF_MONTH, day);
114             return c.getTime();

```

(continues on next page)

(continued from previous page)

```

115     }
116
117     public static void main(String[] args) {
118         new CustomTableOfContentsReport();
119     }
120
121     private class CustomTableOfContentsCustomizer extends
122 ↪TableOfContentsCustomizer {
123         private static final long serialVersionUID = 1L;
124
125         @Override
126         protected ComponentBuilder<?, ?> title() {
127             VerticalListBuilder verticalList = cmp.verticalList();
128             verticalList.add(cmp.line());
129             verticalList.add(super.title());
130             verticalList.add(cmp.line());
131             return verticalList;
132         }
133
134         @Override
135         protected ComponentBuilder<?, ?> headingComponent(int level) {
136             if (level == 0) {
137                 VerticalListBuilder verticalList = cmp.verticalList();
138                 verticalList.add(super.headingComponent(level));
139                 verticalList.add(cmp.line());
140                 return verticalList;
141             } else {
142                 return super.headingComponent(level);
143             }
144         }
145     }

```

1.33.2 Table Of Contents

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by

```

(continues on next page)

(continued from previous page)

```

11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.tableofcontents;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.datasource.DRDataSource;
34 import net.sf.dynamicreports.report.exception.DRException;
35 import net.sf.jasperreports.engine.JRDataSource;
36
37 /**
38  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
39  */
40 public class TableOfContentsReport1 {
41
42     public TableOfContentsReport1() {
43         build();
44     }
45
46     private void build() {
47         TextColumnBuilder<String> countryColumn = col.column("Country",
48 ↪ "country", type.stringType());
49         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
50 ↪ type.stringType());
51
52         try {
53             report()
54                 .setTemplate(Templates.reportTemplate)
55                 .tableOfContents()
56                 .columns(
57                     countryColumn,
58                     itemColumn,
59                     col.column("Order date",
60 ↪ "orderdate", type.dateType()),
61                     col.column("Quantity",
62 ↪ "quantity", type.integerType()),
63                     col.column("Unit price",
64 ↪ "unitprice", type.bigDecimalType()))
65                 .groupBy(countryColumn, itemColumn)
66                 .title(Templates.createTitleComponent(
67 ↪ "TableOfContents1"))

```

(continues on next page)

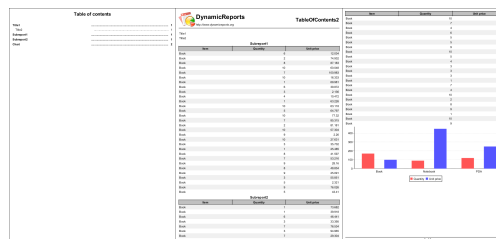
(continued from previous page)

```

62         .pageFooter(Templates.footerComponent)
63         .setDataSource(createDataSource())
64         .show();
65     } catch (DRException e) {
66         e.printStackTrace();
67     }
68 }
69
70 private JRDataSource createDataSource() {
71     String[] countries = new String[] { "USA", "Canada", "Mexico" };
72     String[] items = new String[] { "Book", "Notebook", "PDA" };
73     DRDataSource dataSource = new DRDataSource("country", "item",
74     ↪ "orderdate", "quantity", "unitprice");
75     for (String country : countries) {
76         for (String item : items) {
77             for (int i = 0; i < 8; i++) {
78                 dataSource.add(country, item, toDate(2010, 1,
79     ↪ (int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
80                 new BigDecimal(Math.random()
81     ↪ * 100 + 1));
82             }
83         }
84     }
85     return dataSource;
86 }
87
88 private Date toDate(int year, int month, int day) {
89     Calendar c = Calendar.getInstance();
90     c.set(Calendar.YEAR, year);
91     c.set(Calendar.MONTH, month - 1);
92     c.set(Calendar.DAY_OF_MONTH, day);
93     return c.getTime();
94 }
95
96 public static void main(String[] args) {
97     new TableOfContentsReport1();
98 }

```

1.33.3 Table Of Contents 2



```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca

```

(continues on next page)

(continued from previous page)

```

5  * http://www.dynamicreports.org
6  *
7  * This file is part of DynamicReports.
8  *
9  * DynamicReports is free software: you can redistribute it and/or modify
10 * it under the terms of the GNU Lesser General Public License as published by
11 * the Free Software Foundation, either version 3 of the License, or
12 * (at your option) any later version.
13 *
14 * DynamicReports is distributed in the hope that it will be useful,
15 * but WITHOUT ANY WARRANTY; without even the implied warranty of
16 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17 * GNU Lesser General Public License for more details.
18 *
19 * You should have received a copy of the GNU Lesser General Public License
20 * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21 */
22
23 package net.sf.dynamicreports.examples.tableofcontents;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;
31 import net.sf.dynamicreports.report.builder.FieldBuilder;
32 import net.sf.dynamicreports.report.builder.chart.BarChartBuilder;
33 import net.sf.dynamicreports.report.builder.component.TextFieldBuilder;
34 import net.sf.dynamicreports.report.builder.tableofcontents.
35     ↳TableOfContentsHeadingBuilder;
36 import net.sf.dynamicreports.report.datasource.DRDataSource;
37 import net.sf.dynamicreports.report.exception.DRException;
38 import net.sf.jasperreports.engine.JRDataSource;
39
40 /**
41  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
42  */
43 public class TableOfContentsReport2 {
44
45     public TableOfContentsReport2() {
46         build();
47     }
48
49     private void build() {
50         FieldBuilder<String> itemField = field("item", type.stringType());
51         FieldBuilder<Integer> quantityField = field("quantity", type.
52     ↳integerType());
53         FieldBuilder<BigDecimal> unitPriceField = field("unitprice", type.
54     ↳bigDecimalType());
55
56         TableOfContentsHeadingBuilder tocHeading1 = tableOfContentsHeading();
57         TextFieldBuilder<String> title1 = cmp.text("Title1")
58     ↳.setTableOfContentsHeading(tocHeading1);
59
60         TableOfContentsHeadingBuilder tocHeading2 = tableOfContentsHeading()
61     ↳.setParentHeading(tocHeading1);

```

(continues on next page)

(continued from previous page)

```

59         TextFieldBuilder<String> title2 = cmp.text("Title2")
60             .setTableOfContentsHeading(tocHeading2);
61
62         BarChartBuilder chart = cht.barChart()
63             .setDataSource(createChartDataSource())
64             .setCategory(itemField)
65             .series(
66                 cht.serie(quantityField).setLabel(
↪ "Quantity"),
67                 cht.serie(unitPriceField).setLabel(
↪ "Unit price"))
68             .setTableOfContentsHeading("Chart");
69
70         try {
71             report()
72                 .setTemplate(Templates.reportTemplate)
73                 .tableOfContents()
74                 .title(
75                     Templates.
↪ createTitleComponent("TableOfContents2"),
76                     title1, title2,
77                     cmp.
↪ subreport(createSubreport(1)),
78                     cmp.
↪ subreport(createSubreport(2)),
79                     chart)
80                 .pageFooter(Templates.footerComponent)
81                 .show();
82         } catch (DRException e) {
83             e.printStackTrace();
84         }
85     }
86
87     private JasperReportBuilder createSubreport(int index) {
88         TextFieldBuilder<String> title = cmp.text("Subreport" + index)
89             .setStyle(Templates.bold12CenteredStyle)
90             .setTableOfContentsHeading(tableOfContentsHeading());
91
92         JasperReportBuilder report = report();
93         report
94             .setTemplate(Templates.reportTemplate)
95             .title(title)
96             .columns(
97                 col.column("Item", "item", type.
↪ stringType()),
98                 col.column("Quantity", "quantity",
↪ type.integerType()),
99                 col.column("Unit price", "unitprice",
↪ type.bigDecimalType()))
100             .setDataSource(createSubreportDataSource());
101
102         return report;
103     }
104
105     private JRDataSource createSubreportDataSource() {
106         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪ "unitprice");

```

(continues on next page)

(continued from previous page)

```

107         for (int i = 0; i < 30; i++) {
108             dataSource.add("Book", (int) (Math.random() * 10) + 1, new_
↪BigDecimal(Math.random() * 100 + 1));
109         }
110         return dataSource;
111     }
112
113     private JRDataSource createChartDataSource() {
114         DRDataSource dataSource = new DRDataSource("item", "quantity",
↪"unitprice");
115         dataSource.add("Book", 170, new BigDecimal(100));
116         dataSource.add("Notebook", 90, new BigDecimal(450));
117         dataSource.add("PDA", 120, new BigDecimal(250));
118         return dataSource;
119     }
120
121     public static void main(String[] args) {
122         new TableOfContentsReport2();
123     }
124 }

```

1.33.4 Table Of Contents 3

Item	Quantity	Unit Price	Total
Book	170	100	17000
Notebook	90	450	40500
PDA	120	250	30000
Total	380		87500

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22

```

(continues on next page)

(continued from previous page)

```

23 package net.sf.dynamicreports.examples.tableofcontents;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.math.BigDecimal;
28 import java.util.Calendar;
29 import java.util.Date;
30
31 import net.sf.dynamicreports.examples.Templates;
32 import net.sf.dynamicreports.report.builder.column.TextColumnBuilder;
33 import net.sf.dynamicreports.report.builder.group.ColumnGroupBuilder;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37
38 /**
39  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
40  */
41 public class TableOfContentsReport3 {
42
43     public TableOfContentsReport3() {
44         build();
45     }
46
47     private void build() {
48         TextColumnBuilder<String> countryColumn = col.column("Country",
49 ↪ "country", type.stringType());
50         TextColumnBuilder<String> itemColumn = col.column("Item", "item",
51 ↪ type.stringType());
52
53         ColumnGroupBuilder countryGroup = grp.group(countryColumn);
54         ColumnGroupBuilder itemGroup = grp.group(itemColumn)
55             .setAddToTableOfContents(false);
56
57         try {
58             report()
59                 .setTemplate(Templates.reportTemplate)
60                 .tableOfContents()
61                 .columns(
62                     countryColumn,
63                     itemColumn,
64                     col.column("Order date",
65 ↪ "orderdate", type.dateType()),
66                     col.column("Quantity",
67 ↪ "quantity", type.integerType()),
68                     col.column("Unit price",
69 ↪ "unitprice", type.bigDecimalType()))
70                 .groupBy(countryGroup, itemGroup)
71                 .title(Templates.createTitleComponent(
72 ↪ "TableOfContents3"))
73                 .pageFooter(Templates.footerComponent)
74                 .setDataSource(createDataSource())
75                 .show();
76         } catch (DRException e) {
77             e.printStackTrace();
78         }
79     }
80 }



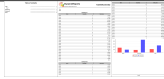

```

(continues on next page)

(continued from previous page)

```
74
75     private JRDataSource createDataSource() {
76         String[] countries = new String[] { "USA", "Canada", "Mexico" };
77         String[] items = new String[] { "Book", "Notebook", "PDA" };
78         DRDataSource dataSource = new DRDataSource("country", "item",
79 ↪ "orderdate", "quantity", "unitprice");
80         for (String country : countries) {
81             for (String item : items) {
82                 for (int i = 0; i < 8; i++) {
83                     dataSource.add(country, item, toDate(2010, 1, ↪
84 ↪ (int) (Math.random() * 10) + 1), (int) (Math.random() * 10) + 1,
85                                     new BigDecimal(Math.random() ↪
86 ↪ * 100 + 1));
87                 }
88             }
89         }
90         return dataSource;
91     }
92
93     private Date toDate(int year, int month, int day) {
94         Calendar c = Calendar.getInstance();
95         c.set(Calendar.YEAR, year);
96         c.set(Calendar.MONTH, month - 1);
97         c.set(Calendar.DAY_OF_MONTH, day);
98         return c.getTime();
99     }
100
101     public static void main(String[] args) {
102         new TableOfContentsReport3();
103     }
104 }
```

Table 18: Table Of Contents Examples

		
CustomTableOfContentsReport	TableOfContentsReport1	TableOfContentsReport2
		
TableOfContentsReport3		

1.34 Template Design

1.34.1 Jasper Template Design 1

DynamicReports

<http://www.dynamicreports.org>

JasperTemplateDesign1

Page header defined in jrxml file

Item	Quantity	Unit price
row 1, Item = Book, quantity = 10, unit price = 38, defined in jrxml file		
Book	10	38
row 2, Item = Book, quantity = 1, unit price = 6, defined in jrxml file		
Book	1	6
row 3, Item = Book, quantity = 4, unit price = 94, defined in jrxml file		
Book	4	94
row 4, Item = Book, quantity = 6, unit price = 65, defined in jrxml file		
Book	6	65
row 5, Item = Book, quantity = 8, unit price = 85, defined in jrxml file		
Book	8	85
row 6, Item = Book, quantity = 3, unit price = 13, defined in jrxml file		
Book	3	13
row 7, Item = Book, quantity = 1, unit price = 22, defined in jrxml file		
Book	1	22
row 8, Item = Book, quantity = 2, unit price = 9, defined in jrxml file		
Book	2	9
row 9, Item = Book, quantity = 8, unit price = 29, defined in jrxml file		
Book	8	29
row 10, Item = Book, quantity = 10, unit price = 67, defined in jrxml file		
Book	10	67

Summary defined in jrxml file

Page footer defined in jrxml file

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.templateDesign;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.InputStream;

```

(continues on next page)


(continued from previous page)

```

28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.datasource.DRDataSource;
31 import net.sf.dynamicreports.report.exception.DRException;
32 import net.sf.jasperreports.engine.JRDataSource;
33
34 /**
35  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36  */
37 public class JasperTemplateDesignReport1 {
38
39     public JasperTemplateDesignReport1() {
40         build();
41     }
42
43     private void build() {
44         InputStream is = JasperTemplateDesignReport1.class.
↳getResourceAsStream("templatedesign1.jrxml");
45
46         try {
47             report()
48                 .setTemplate(Templates.reportTemplate)
49                 .setTemplateDesign(is)
50                 .columns(
51                     col.column("Item", "item",
↳type.stringType()),
52                     col.column("Quantity",
↳"quantity", type.integerType()),
53                     col.column("Unit price",
↳"unitprice", type.integerType()))
54                 .title(Templates.createTitleComponent(
↳"JasperTemplateDesign1"))
55                 .setDataSource(createDataSource())
56                 .show();
57         } catch (DRException e) {
58             e.printStackTrace();
59         }
60     }
61
62     private JRDataSource createDataSource() {
63         DRDataSource dataSource = new DRDataSource("item", "quantity",
↳"unitprice");
64         for (int i = 0; i < 10; i++) {
65             dataSource.add("Book", (int) (Math.random() * 10) + 1, (int)
↳(Math.random() * 100) + 1);
66         }
67         return dataSource;
68     }
69
70     public static void main(String[] args) {
71         new JasperTemplateDesignReport1();
72     }
73 }

```

1.34.2 Jasper Template Design 2

DynamicReports			JasperTemplateDesign2
			http://www.dynamicreports.org
Subreport - dynamic design			
Subreport - static design defined in subreport.xml file			
Subreport - static and dynamic design			
Subreport - static and dynamic design defined in subreporttemplatedesign.xml file			
Page header defined in templatedesign2.xml file			
Item	Quantity	Unit price	
Book	7	39	
Book	7	78	
Book	4	80	
Book	1	73	
Book	1	49	
Summary defined in templatedesign2.xml file			
Page footer defined in templatedesign2.xml file			

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23  package net.sf.dynamicreports.examples.templatedesign;
24
25  import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27  import java.io.InputStream;
28
29  import net.sf.dynamicreports.examples.Templates;
30  import net.sf.dynamicreports.jasper.builder.JasperReportBuilder;

```

(continues on next page)

(continued from previous page)

```

31 import net.sf.dynamicreports.report.builder.style.StyleBuilder;
32 import net.sf.dynamicreports.report.constant.HorizontalTextAlignment;
33 import net.sf.dynamicreports.report.constant.VerticalTextAlignment;
34 import net.sf.dynamicreports.report.datasource.DRDataSource;
35 import net.sf.dynamicreports.report.exception.DRException;
36 import net.sf.jasperreports.engine.JRDataSource;
37 import net.sf.jasperreports.engine.JRException;
38 import net.sf.jasperreports.engine.JasperCompileManager;
39 import net.sf.jasperreports.engine.JasperReport;
40
41 /**
42  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
43  */
44 public class JasperTemplateDesignReport2 {
45
46     public JasperTemplateDesignReport2() {
47         build();
48     }
49
50     private void build() {
51         InputStream is = JasperTemplateDesignReport2.class.
↳getResourceAsStream("templatedesign2.jrxml");
52
53         try {
54             report()
55                 .setTemplate(Templates.reportTemplate)
56                 .setTemplateDesign(is)
57                 .columns(
58                     col.column("Item", "item",
↳type.stringType()),
59                     col.column("Quantity",
↳"quantity", type.integerType()),
60                     col.column("Unit price",
↳"unitprice", type.integerType()))
61                 .title(
62                     Templates.
↳createTitleComponent("JasperTemplateDesign2"),
63                     cmp.
↳subreport(createDynamicSubreport()),
64                     cmp.
↳subreport(createStaticSubreport()),
65                     cmp.
↳subreport(createStaticAndDynamicSubreport()))
66                 .setDataSource(createDataSource())
67                 .show();
68         } catch (DRException e) {
69             e.printStackTrace();
70         } catch (JRException e) {
71             e.printStackTrace();
72         }
73     }
74
75     private JasperReportBuilder createDynamicSubreport() {
76         return createSubreport("Subreport - dynamic design");
77     }
78
79     private JasperReportBuilder createStaticAndDynamicSubreport() throws
↳DRException {

```

(continues on next page)


(continued from previous page)

```

80         InputStream is = JasperTemplateDesignReport2.class.
↳getResourceAsStream("subreporttemplatedesign.jrxml");
81         JasperReportBuilder report = createSubreport("Subreport - static and
↳dynamic design");
82         report.setTemplateDesign(is);
83         return report;
84     }
85
86     private JasperReportBuilder createSubreport(String title) {
87         StyleBuilder style = stl.style()
88             .setHorizontalTextAlignment(HorizontalTextAlignment.
↳CENTER)
89             .setVerticalTextAlignment(VerticalTextAlignment.
↳MIDDLE)
90             .setBorder(stl.pen1Point());
91
92         JasperReportBuilder report = report();
93         report
94             .setTemplate(Templates.reportTemplate)
95             .title(
96                 cmp.horizontalList(
97                     cmp.gap(30, 47),
98                     cmp.text(title).
↳setStyle(style),
99                     cmp.gap(30, 47)));
100
101         return report;
102     }
103
104     private JasperReport createStaticSubreport() throws JRException {
105         InputStream is = JasperTemplateDesignReport2.class.
↳getResourceAsStream("subreport.jrxml");
106         return JasperCompileManager.compileReport(is);
107     }
108
109     private JRDataSource createDataSource() {
110         DRDataSource dataSource = new DRDataSource("item", "quantity",
↳"unitprice");
111         for (int i = 0; i < 5; i++) {
112             dataSource.add("Book", (int) (Math.random() * 10) + 1, (int)
↳(Math.random() * 100) + 1);
113         }
114         return dataSource;
115     }
116
117     public static void main(String[] args) {
118         new JasperTemplateDesignReport2();
119     }
120 }

```

1.34.3 Jasper Template Design 3



DynamicReports

<http://www.dynamicreports.org>

JasperTemplateDesign3

Page header defined in jxmt file

Detail defined in jxmt file

Item	Quantity	Unit price
Book	2	68
Book	1	43
Book	7	18
Book	1	58
Book	2	36

Detail defined in jxmt file

Item	Quantity	Unit price
Book	6	13
Book	4	71
Book	7	88
Book	6	99
Book	5	85

Page footer defined in jxmt file

```

1  /**
2   * DynamicReports - Free Java reporting library for creating reports dynamically
3   *
4   * Copyright (C) 2010 - 2018 Ricardo Mariaca
5   * http://www.dynamicreports.org
6   *
7   * This file is part of DynamicReports.
8   *
9   * DynamicReports is free software: you can redistribute it and/or modify
10  * it under the terms of the GNU Lesser General Public License as published by
11  * the Free Software Foundation, either version 3 of the License, or
12  * (at your option) any later version.
13  *
14  * DynamicReports is distributed in the hope that it will be useful,
15  * but WITHOUT ANY WARRANTY; without even the implied warranty of
16  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
17  * GNU Lesser General Public License for more details.
18  *
19  * You should have received a copy of the GNU Lesser General Public License
20  * along with DynamicReports. If not, see <http://www.gnu.org/licenses/>.
21  */
22
23 package net.sf.dynamicreports.examples.templatedesign;
24
25 import static net.sf.dynamicreports.report.builder.DynamicReports.*;
26
27 import java.io.InputStream;
28
29 import net.sf.dynamicreports.examples.Templates;
30 import net.sf.dynamicreports.report.exception.DRException;

```

(continues on next page)

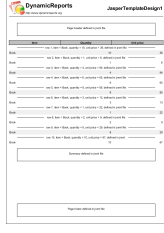


(continued from previous page)

```

31 import net.sf.jasperreports.engine.JRDataSource;
32 import net.sf.jasperreports.engine.JREmptyDataSource;
33
34 /**
35  * @author Ricardo Mariaca (r.mariaca@dynamicreports.org)
36  */
37 public class JasperTemplateDesignReport3 {
38
39     public JasperTemplateDesignReport3() {
40         build();
41     }
42
43     private void build() {
44         InputStream is = JasperTemplateDesignReport3.class.
45         ↪getResourceAsStream("templatedesign3.jrxml");
46
47         try {
48             report()
49                 .setTemplateDesign(is)
50                 .title(Templates.createTitleComponent (
51                 ↪"JasperTemplateDesign3"))
52                 .setDataSource(createDataSource())
53                 .show();
54         } catch (DRException e) {
55             e.printStackTrace();
56         }
57
58     private JRDataSource createDataSource() {
59         return new JREmptyDataSource(2);
60     }
61
62     public static void main(String[] args) {
63         new JasperTemplateDesignReport3();
64     }
65 }

```

Table 19: Template Design

 <p>JasperTemplateDesignReport1</p>	 <p>JasperTemplateDesignReport1</p>	 <p>JasperTemplateDesignReport1</p>
--	--	--