TIBCO JASPERSOFT STUDIO® USER GUIDE

RELEASE 6.1

http://www.jaspersoft.com

This is version 0515-JSS61-07 of the *Jaspersoft Studio User Guide*.

TABLE OF CONTENTS

CHAPTER 1 GETTING STARTED WITH JASPERSOFT STUDIO

Jaspersoft Studio is the latest incarnation of the well-known iReport Editor. Because it is built on the Eclipse platform, Jaspersoft Studio is a more complete solution that allows users to extend its capabilities and functionality.

This chapter contains the following sections:

1.1 Introduction to Jaspersoft Studio

TIBCO™ JasperReports® Server builds on TIBCO™ JasperReports® Library as a comprehensive family of Business Intelligence (BI) products, providing robust static and interactive reporting, report server, and data analysis capabilities. These capabilities are available as either stand-alone products, or as part of an integrated end-to-end BI suite utilizing common metadata and provide shared services, such as security, a repository, and scheduling. The server exposes comprehensive public interfaces enabling seamless integration with other applications and the capability to easily add custom functionality.

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

The heart of the TIBCO™ Jaspersoft® BI Suite is the server, which provides the ability to:

- Easily create new reports based on views designed in an intuitive, web-based, drag and drop Ad Hoc Editor.
- Efficiently and securely manage many reports.
- Interact with reports, including sorting, changing formatting, entering parameters, and drilling on data.
- Schedule reports for distribution through email and storage in the repository.
- Arrange reports and web content to create appealing, data-rich Jaspersoft Dashboards that quickly convey business trends.

For business intelligence users, Jaspersoft offers TIBCO™ Jaspersoft® OLAP, which runs on the server.

While the Ad Hoc Editor lets users create simple reports, more complex reports can be created outside of the server. You can either use TIBCO™ Jaspersoft® Studio or manually write JRXML code to create a report that can be run in the server. We recommend that you use Jaspersoft Studio unless you have a thorough understanding of the JasperReports file structure.

Jaspersoft Studio is an Eclipse-based report designer for JasperReports Library and JasperReports Server; it's available as an Eclipse plug-in or as a stand-alone application. Jaspersoft Studio allows you to create sophisticated layouts containing charts, images, subreports, crosstabs, and more. You can access your data through a variety of sources including JDBC, TableModels, JavaBeans, XML, Hibernate, Big Data (such as Hive), CSV, XML/A, as well as custom sources, then publish your reports as PDF, RTF, XML, XLS, CSV, HTML, XHTML, text, DOCX, or OpenOffice.

You can use the following sources of information to extend your knowledge of JasperReports Server:

- Our core documentation describes how to install, administer, and use JasperReports Server. Core documentation is available as PDFs in the doc subdirectory of your JasperReports Server installation. You can also access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.
- Our Ultimate Guides document advanced features and configuration. They also include best practice recommendations and numerous examples. You can access PDF and HTML versions of these guides online from the Documentation section of the Jaspersoft Community website.
- Our Online Learning Portal lets you learn at your own pace, and covers topics for developers, system administrators, business users, and data integration users. The Portal is available online from Professional Services section of our website.
- Our free samples, which are installed with JasperReports, Jaspersoft Studio, and JasperReports Server, are documented online. OPTIONAL TEXT HERE

JasperReports Server is a component of both a community project and commercial offerings. Each integrates the standard features such as security, scheduling, a web services interface, and much more for running and sharing reports. Commercial editions provide additional features, including Ad Hoc charts, flash charts, dashboards, Domains, auditing, and a multi-organization architecture for hosting large BI deployments.

1.2 User Interface

Jaspersoft Studio is offered in two different versions: a stand-alone Rich Client Platform (RCP) product, and an Eclipse plug-in version. If you have worked with Eclipse, you are likely familiar with the user interface; those familiar with iReport will find that the layout is different in the new designer. Both the stand-alone and plug-in versions have a similar user interface. **Figure 1-1**, shows a preview of the Jaspersoft Studio interface, with the main areas highlighted.

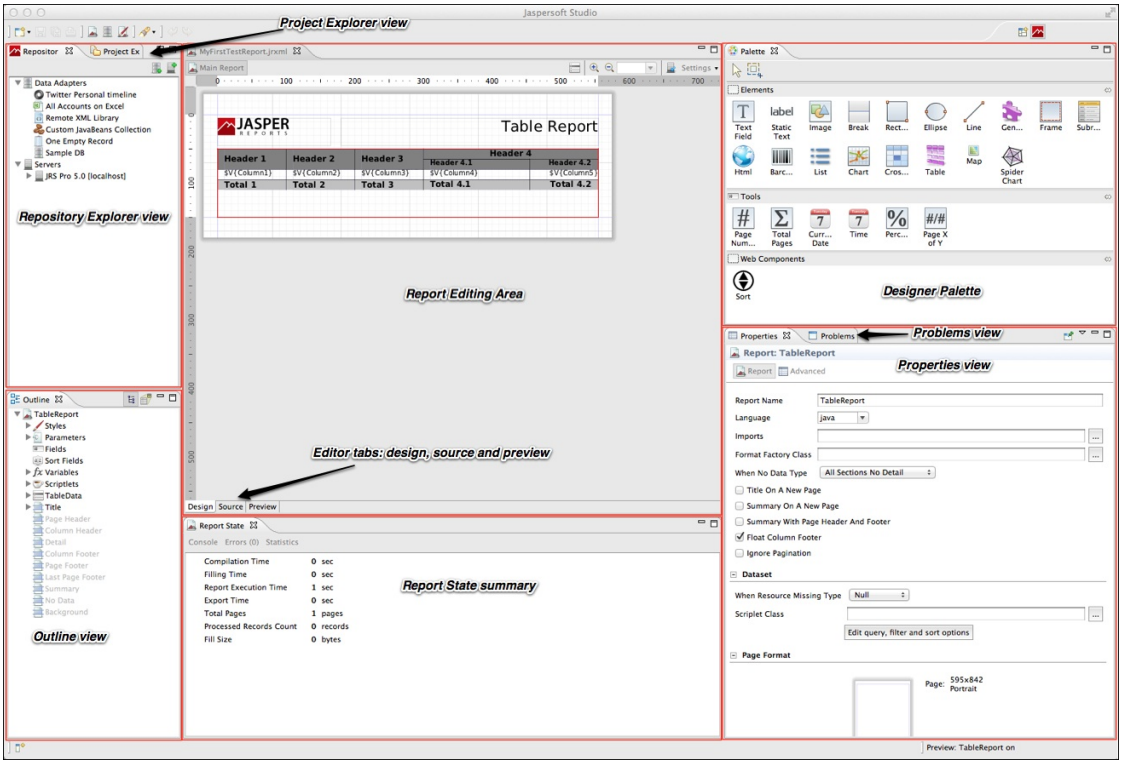


Figure 1-1 Jaspersoft Studio User Interface

1.2.1 User Interface Components

Jaspersoft Studio has a multi-tab editor, which includes three tabs: Design, Source, and Preview:

- The Design tab is the main one selected when you open a report file and it allows you to graphically create your report.
- The Source tab contains the JRXML source code for your report.
- The Preview tab lets you run the report preview after having selected a data source and output format.

You can explore the data using the following views:

- The Repository Explorer view maintains the list of JasperReports Server connections and available data adapters.
- The Project Explorer view maintains the list of the projects in the current workspace, usually JasperReports Server project.
- The Outline view shows the complete structure of the report in a tree-form way.
- The Properties view is common to most Eclipse-based plug-ins. It contains properties information for the selected element. When you select a report element from the main design area (such as a text field) or from the Outline, the Properties view shows information about it. Some of these properties are read-only, but most are editable and their modification change the elements that are drawn (such as element width or height).
- The Problems view shows a list of problems and errors that can, for example, block the correct compilation of a report.

- The Report state summary provides statistics on report compilation/filling/execution. Errors are shown here as well.

This comparison table shows the differences in terminology between iReport and Jaspersoft Studio.

Table 1-1 Comparison of Features in iReport and Jaspersoft Studio

iReport	Jaspersoft Studio
JasperReports Server Repository	Repository Explorer
Report Inspector	Outline view
Report Designer	Report editing area
Problems List	Problems view
Elements palette	Designer Palette
Formatting tools	Available via context menu on the element
Property sheet	Properties view
Styles library	---
---	Project Explorer
iReport Output window	Report State summary

1.3 Hardware Requirements

Jaspersoft Studio needs a 64-bit or 32-bit processor and at least 500 MB of Hard Disk space. The amount of RAM needed is dependent upon report complexity. A value of 1 GB dedicated to Jaspersoft Studio is recommended, 2 GB is suggested.

1.4 Software Requirements

Jaspersoft Studio requires the Java Runtime Environment (JRE). To compile the report scriptlets, a full distribution of Java is required. The JSS installer includes the required version of Java.

During the JSS download, you must accept the Java license agreement and select the correct operating system. Jaspersoft Studio supports the following common operating systems:

- Windows 7/8, 32 or 64 bit
- Linux, 32 or 64 bit
- MacOS X, 64 bit

You can download Jaspersoft Studio here:

http://community.jaspersoft.com/project/jaspersoft-studio/releases.

A page that lists all the releases of Jaspersoft Studio appears. Click the latest version, and a grid with the following links appears:

- jaspersoftstudio_x.x.x_i386.deb for the 32-bit version for Linux;
- jaspersoftstudio_x.x.x_amd64.deb for the 64-bit version for Linux;
- jaspersoftstudio_x.x.x_windows-installer-x86_64.exe for the 64-bit version of windows;
- jaspersoftstudio_x.x.x_windows-installer-x86.exe for the 32-bit version of windows;
- jaspersoftstudio-x.x.x-mac-x86_64.dmg for the 64-bit version of MacOS X.

x.x.x represents the version number of Jaspersoft Studio. If your distribution of Linux doesn't support the deb format there are also tar versions.

Note that on a 64-bit operating system you can install the 32-bit version of Jaspersoft Studio (although the 64-bit is recommended), but you cannot install the 64- bit version of Jaspersoft Studio on a 32-bit operating system.

1.5 Compatibility Between Versions

When a new version of JasperReports is distributed, some classes usually change. These modified classes typically impact the XML syntax and the `JASPER` file structure.

Before JasperReports 1.1.0, this was a serious problem and a major upgrade deterrent, since it required recompiling all the `JRXML` files in order to be used with the new library version. Things changed after the release of Version 1.1.0, in which JasperReports assured backwards compatibility, that is, the library is able to understand and execute any `JASPER` file generated with a previous version of JasperReports.

With JasperReports 3.1, the `JRXML` syntax moved from a DTD-based definition to XML-based schema. The XML source declaration syntax now references a schema file, rather than a DTD. Based on what we said previously, this is not a problem since JasperReports assures backwards compatibility. However, many people are used to designing reports with early versions of iReport then generating the reports by compiling `JRXML` in JasperReports. This was always a risky operation, but it was still valid because the user was not using a new tag in the XML. With the move to an XML schema, the `JRXML` output of iReport 3.1.1 and newer can only be compiled with a JasperReports 3.1.0 or later. All versions of Jaspersoft Studio produce output that is only compatible with later version of JasperReports Library.

1.6 Accessing the Source Code

The last version of the source code is available from http://community.jaspersoft.com/project/jaspersoft-studio/releases by clicking **Browse Source Code**, which lets you access the Subversion (SVN) repository (read only mode) where the most up-to-date version is available. You can download and compile this source code, but since it is a work in progress it might contain new, unreleased features and bugs. All the information necessary to download the Source Code, configure a develop environment on the Eclipse IDE and compile\run the source code are described in the tutorial Contributing to Jaspersoft Studio and building from sources.

1.7 Report Structure in Jaspersoft Studio

A report is defined by means of a type page. This is divided into different horizontal portions named bands. When the report is joined with the data generating the print, this section are printed many times according to their function (and according to the rules that the report designer has set). For instance, the page header is repeated at the beginning of every page, while the detail band is repeated for each record.

1.7.1 The Report Lifecycle

The report lifecycle starts with report design. Designing a report means creating a template, such as a form containing blank space, that can be filled with data. Some portions of a page defined in this way are reused, others stretch to fit the content, etc..

This template is saved as an XML file sub-type called JRXML ("JR" for JasperReports). This file contains information about the report layout, including complex formulas to perform calculations, an optional query to retrieve data out of a data source, and other functionality.

The life cycle can be divided into two distinct action sets:

- Tasks executed during the development phase: design and planning of the report, and compilation of a Jasper file source, the JRXML.
- Tasks that must be executed in run time: loading of the Jasper file, filling of the report, and export of the print in a final format.

The main role of Jaspersoft Studio in the cycle is to design a report and create an associated `JASPER` file, though it is able to preview the result and export it in all the supported formats. Jaspersoft Studio provides support for a wide range of data sources and allows users to create custom data sources, thereby becoming a complete environment for report development and testing.

When you design a report using Jaspersoft Studio, you are creating a JRXML file, which is an XML document that contains the definition of the report layout. The layout is completely visual, so you can ignore the underlying structure of the JRXML file. Before executing a report, the JRXML must be compiled in a binary object called a Jasper file. Jasper files are what you need to ship with your application in order to run the reports.

Report execution is performed by passing a Jasper file and a data source to JasperReports. There are many data source types. You can fill a Jasper file from an SQL query, an XML file, a .csv file, an HQL (Hibernate Query Language) query, a collection of JavaBeans, and others. If you don't have a suitable data source, JasperReports allows you to write your own custom data source. With a Jasper file and a data source, JasperReports is able to generate the final document in the format you want.

Jaspersoft Studio also lets you configure data sources and use them to test your reports. In many cases, data-driven wizards can help you design your reports much quicker. Jaspersoft Studio includes the JasperReports engine itself to let you preview your report output, test, and refine your reports.

1.7.2 Understanding Bands

The type page is divided into nine predefined bands to which new groups are added. In addition, Jaspersoft Studio manages a heading band (group header) and a recapitulation band (group footer) for every group.

A band is as wide as the page width (right and left margins excluded). However, its height, even if it is established during the design phase, can vary during print creation according to the contained elements; it can "lengthen" toward the bottom of a page in an arbitrary way. This typically occurs when bands contain subreports or text fields that have to adapt to the content vertically. Generally, the height specified by the user should be considered "the minimal height" of the band. Not all bands can be stretched dynamically according to content; in particular the column footer, page footer, and last page footer bands are statically sized.

The sum of all band heights (except for the background) has to always be less than or equal to the page height minus the top and bottom margins.

1.7.3 Band Types

The following table contains brief descriptions of the available bands:

Band Name	Description
Title	The title band is the first visible band. It is created only once and can be printed on a separate page. It is not possible during design to exceed the report page height (top and bottom margins are included). If the title is printed on a separate page, this band height is not included in the calculation of the total sum of all band heights.
Page Header	The page header band allows you to define a page header. The height specified during the design phase usually does not change during the creation process, except for the insertion of vertically resizable components such as a text fields. The page header appears on all printed pages in the position defined during the design phase. Title and summary bands do not include the page header when printed on a separate page.
Column Header	The column header band is printed at the beginning of each detail column. Usually labels containing the column names of a tabular report are inserted in this band.
Group Header	A report can contains zero or more group bands which permit the collection of detail records in real groups. A group header is always accompanied by a group footer (both can be independently visible or not). Different properties are associated with a group. They determine its behavior from the graphic point of view. It is possible to always force a group header on a new page or in a new column and to print this band on all pages if the bands below it overflow the single page (as a page header, but at group level). It is possible to fix a minimum height required to print a group header: if it exceeds this height, the group header band is printed on a new page (please note that a value too large for this property can create an infinite loop during printing).
Group Footer	The group footer band completes a group. Usually it contains fields to view subtotals or separation graphic elements, such as lines.
Column Footer	The column footer band appears on at the end of every column. Its dimension are not resizable at run time (not even if it contains resizable elements such as subreports or text fields with a variable number of text lines).
Page Footer	The page footer band appears on every page where there is a page header. Like the column footer, it is not resizable at run time.
Last Page Footer	If you want to make the last page footer different from the other footers, it is possible to use the special last page footer band. If the band height is 0, it is completely ignored, and the layout established for the common page is used for the last page.
Summary	The summary band allows you to insert fields containing total calculations, means, or any other information you want to include at the end of the report.
Background	The background enables you to create watermarks and similar effects, such as a frame around the whole page. It can have a maximum height equal to the page height.

1.7.4 Specifying Report Properties

To edit report properties, click on the **Edit page format** button in the properties view of the report, or right click on the report itself, and select **Page Format** from the contextual menu, to see the page properties.

In the dialog that appears you can change the page dimensions.

The unit of measurement used by Jaspersoft Studio and JasperReports is the pixel. However, it is possible to specify report dimension using other units of measurement, such as centimeters, millimeters, or inches. Note that because the dimensions management is based on pixels, some rough adjustments can take place when viewing the same data using different units of measurement.In the following table there is a list of standard measures and their dimensions in pixels.

Page Type	Dimensions in Pixels
Letter	612x792
Note	540x720
Legal	612x1008
A0	2380x3368
A1	1684x3368
A2	1190x1684
A3	842x1190
A4	595x842
A5	421x595
A6	297x421
A7	210x297
A8	148x210
A9	105X148
A10	74X105
B0	2836x4008
B1	2004x2836
B2	1418x2004
B3	1002x1418
B4	709x1002

Page Type	Dimensions in Pixels
B5	501x709
ARCH_E	2592x3456
ARCH_D	1728x2593
ARCH_C	1296x1728
ARCH_B	864x1296
ARCH_A	648x864
FLSA	612x936
FLSE	612x936
HALFLETTER	396x612
11X17	792x1224
LEDGER	1224x792

By modifying width and height, it is possible to create a report of whatever size you like. Although Jaspersoft enables you to create pixel-perfect reports, the page orientation options, Landscape or Portrait, are there because they are used by certain report exporters. The page margin dimensions are set by means of the four options on the **Page Margin** tab.

1.7.5 Columns

Pages, one or more of which make up a report, presents bands are independent from the data (such as the title or the page footers) and other bands that are printed only if there are one or more data records to print (such as the group headers and the detail band). These last sections can be divided into vertical columns in order to take advantage of the available space on the page. A column does not concern the record fields, but it does concern the detail band. This means that if you have record with ten fields and you desire a table view, ten columns are not needed. However, the element must be placed correctly to have a table effect. Ten columns are returned when long records lists (that are horizontally very narrow) are printed.

Next, let's set up columns in a report as an example. Create a new report from **File > New > Jasper Report**. Choose as template **BlankA4** and name it `ColumnExample`. Use **Sample DB - Database JBDC Connection** for the data adapter, with the following SQL query: `select * from orders`. Fields from the database are discovered. Double-click `SHIPNAME`, to add it to the report field and click **Next** twice. Finally, click **Finish**.From

the outline view drag the `SHIPNAME` field in the report in the detail band, resize the detail band, and remove the unused bands. Go to the Preview tab to see the report compiled.

By default the number of columns is 1, and its width is equal to the entire page, except the margins. The space between columns is zero by default. Most of the page is unused. If multiple columns are used, this report would

look better. On the Page Format dialog set the number of columns to two and compile the report to see the changes.

Jaspersoft Studio automatically calculates maximum column width according to the margins and the page width. If you want to increase the space between the columns, increase the value of the **Space** field.

The restricted area is used to mark every column after the first, to show that all the elements should be placed in the first column; the other column's is replicated automatically during compilation. If you want you can also put elements in the other columns, but in most cases you need only the first. It is not recommended that you use parts of the report as margins and columns after the first, if they have to be considered as though they were a continuation of the first.

Multiple columns are commonly used for print-outs of very long lists (for example, a phone directory). It is important to remember that when you have more than one column, the width of the detail band and of linked bands is reduced to the width of the columns.

The sum of the margins, column widths, and space between columns has to be less or equal to the page width. If this condition is not met, the compilation results in an error.

1.7.6 Advanced Options

From the properties tab of the report there are many other options for the report configuration. Select the report root node from the outline view, and in the **Properties** tab you see:

- **Report Name**: It is a logical name, independent form the source file's name, and is used only by the JasperReports library (for example, to name the produced java file when a report is compiled).
- **Title on a new page**: This option specifies that the title band is to be printed on a new page, which forces a page break at the end of the title band. In the first page only the title band is printed. However this page is still included in total page count.
- **Summary on a new page**: This option is similar to **"**Title on a new page**"** except that the summary band is printed as the last page. If you need to print this band on a new page, the new page only contains the summary band.
- **Summary with page header and footer**: This option specifies if the summary band is to be accompanied by the page header and the page footer.
- **Floating column footer**: This option forces the printing of the column footer band immediately after the last detail band (or group footer) rather than the end of the column. This option is used, for example, when you want to create tables using the report elements.
- **When no data type**: When an empty data is supplied as the print number (or the SQL associated with the report returns no records), an empty file is created (or a stream of zero bytes is returned). This default behavior can be modified by specifying what to do in the case of absence of data. The possible values for this field are:
  - **NoPages**: This is the default value; the final result is an empty buffer.
  - **BlankPage**: This returns an empty page.
  - **AllSectionNoDetails**: This returns a page containing all bands except for the detail band.

1.8 Units of Measure in Jaspersoft Studio

Jaspersoft Studio can handle many units of measure, including pixels, centimeters, millimeters and inches. To accomplish this, we included a measure component in Jaspersoft Studio. This component looks like a standard text box with a place to enter a measure unit to the right of the value.

This component can handle a different measure unit for each field, if needed.

1.8.1 Configuration

You can set two preferred (default) units of measure, one at the field level, the other at the report level. The report level unit is used wherever there is not a preferred field unit of measure. The report's default unit of measure is the pixel.

To change the report level unit:

1. Select **Window** > **Preferences** to open the Preferences window.
2. Expand **Jaspersoft Studio** and select **Report Designer**.
3. Use the Default Unit drop-down menu to select one of the following units of measure:
   - Pixels
   - Inches
   - Millimeters
   - Centimeters
   - Meters

1.8.2 Changing the Field Unit of Measure

To change a field's local unit of measure select the field, double-click the unit of measure in the Properties tab, and select a supported unit from the pop-up menu:

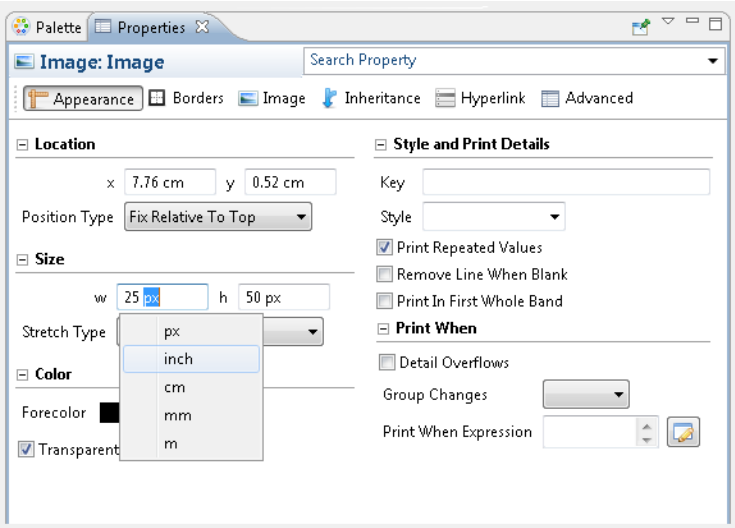


Figure 1-2 Updating a field's measure unit

1.8.3 Alias and Auto-complete

Jaspersoft Studio has included alias and auto-complete services for units of measure. The following table shows your options.

Unit	Accepted Values
centimeter	centimeter, centimeters, cm
millimeter	millimeter, millimeters, mm
meter	meter, meters, m
pixel	pixel, pixels, px
inch	inch, inches, " (double quote)

Enter a value and begin typing a unit of measure. Auto-complete will list the matching supported values for you to choose from.

1.8.4 Approximations

Even through Jaspersoft Studio handles many units of measure, JasperReports works only with pixels. So pixels are the only unit allowed in the project file. Jaspersoft Studio approximates measurements and converts them to pixels. For example, 5 cm is converted to the nearest whole-number value in pixels. In this case the 5 centimeters is converted to 139 pixels (approximately 4.97 cm).

1.9 Exporting reports with Jaspersoft Studio

In addition to generating and viewing reports, Jaspersoft Studio allows you to export reports into many formats, including PDF, XLS, HTML and others.

1.9.1 Compiling the Report

When you select the Preview tab in the designer bottom bar, Jaspersoft Studio performs a set of operations to create the final report. The first operation compiles the JRXML source file in a Jasper file. This first step can fail if the elements are not correctly positioned (for example, if an element is placed outside of a band), or if an expression in the report has errors and cannot be compiled.

If the compilation runs successfully, the produced Jasper file is loaded and filled using the active connection or data source. This second operation can also lead to errors. This can happen if the referenced database is not active, an invalid query has been provided, or a null field produced an error in an expression during the filling process. If all operations complete without error, the report is displayed in the integrated viewer. Errors are shown in the Report State window, after clicking the Errors button.

If errors occur during the compilation, the tab focus changes from Preview to Design.

1.9.2 Preview and Exporting

If the compilation completes and there are no errors in the file, the preview is shown. From there you can browse the generated report and change its visualization, change the data source or export the report. Note that after changing the data source the report is recompiled automatically. You can also change the preview format as well as save the report in different formats.

When you set a preview format, the report is automatically regenerated in the chosen format, and the corresponding viewer application is opened.

CHAPTER 2 CONCEPTS OF JASPERREPORTS

This chapter illustrates JasperReports' base concepts for a better understanding of how Jaspersoft Studio works.

The JasperReports API, the XML syntax for report definition, and details for using the library in your own programs are documented in the *JasperReports Library Ultimate Guide*. This guide, along with other information and examples, is directly available on the Jaspersoft community site at http://community.jaspersoft.com.

JasperReports is published under the LGPL license, which is a less restrictive GPL license. JasperReports can be freely used on commercial programs without buying expensive software licenses and without remaining trapped in the complicated net of open source licenses. This is important when reports created with Jaspersoft Studio are used in a commercial product; in fact, programs only need the JasperReports library to produce prints, which work something like run time executables.

This chapter contains the following sections:

2.1 JRXML Sources and Jasper Files

JasperReports defines a report with an XML file. A `JRXML` file is composed of a set of sections; some concerned with the report's physical characteristics (such as the dimensions of the page, positioning of the fields, and height of the bands), and some concerned with the logical characteristics (such as the declaration of the parameters and variables and the definition of a query for data selection).

The following figure shows sample report source code:

Table 2-1 A simple JRMXL file example

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
        http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
      name="My first report" pageWidth="595" pageHeight="842" columnWidth="535"
      leftMargin="20" rightMargin="20" topMargin="20" bottomMargin="20">
  <queryString language="SQL">
    <![CDATA[select * from address order by city]]>
  </queryString>
  <field name="ID" class="java.lang.Integer">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="FIRSTNAME" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="LASTNAME" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="STREET" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>
  <field name="CITY" class="java.lang.String">
    <fieldDescription><![CDATA[]]></fieldDescription>
  </field>

  <group name="CITY">
    <groupExpression><![CDATA[$F{CITY}]]></groupExpression>
    <groupHeader>
      <band height="27">
        <staticText>
          <reportElement mode="Opaque" x="0" y="0" width="139" height="27"
          forecolor="#FFFFFF" backcolor="#000000"/>
          <textElement>
            <font size="18"/>
          </textElement>
        <text><![CDATA[CITY]]></text>
      </staticText>

      <textField hyperlinkType="None">
        <reportElement mode="Opaque" x="139" y="0" width="416" height="27"
        forecolor="#FFFFFF" backcolor="#000000"/>
        <textElement>
          <font size="18" isBold="true"/>
        </textElement>
        <textFieldExpression class="java.lang.String"><![CDATA[$F{CITY}]]>
        </textFieldExpression>
      </textField>
    </band>
  </groupHeader>
  <groupFooter>
    <band height="8">
      <line direction="BottomUp">
        <reportElement key="line" x="1" y="4" width="554" height="1"/>
      </line>
```

```
    </band>
  </groupFooter>
</group>


<background>
  <band/>
</background>
  <title>
    <band height="58">
      <line>
        <reportElement x="0" y="8" width="555" height="1"/>
      </line>
      <line>
        <reportElement positionType="FixRelativeToBottom" x="0" y="51" width="555"
            height="1"/>
      </line>

      <staticText>
        <reportElement x=”65” y=”13” width ”424” height=”35”/>
        <textElement textAlignment=”Center”>
          <font size=”26” isBold=”true”/>
        </textElement>
      <text><![CDATA[Classic template]]> </text>
    </staticText>
</band>
</title>


  <pageHeader>
    <band/>
  </pageHeader>
  <columnHeader>
    <band height="18">
      <staticText>
        <reportElement mode="Opaque" x="0" y="0" width="138" height="18"
        forecolor="#FFFFFF" backcolor="#999999"/>
        <textElement>
          <font size="12"/>
        </textElement>
        <text><![CDATA[ID]]></text>
      </staticText>
      <staticText>
        <reportElement mode="Opaque" x="138" y="0" width="138" height="18"
        forecolor="#FFFFFF" backcolor="#999999"/>
        <textElement>
          <font size="12"/>
        </textElement>
        <text><![CDATA[FIRSTNAME]]></text>
      </staticText>
      <staticText>
        <reportElement mode="Opaque" x="276" y="0" width="138" height="18"
        forecolor="#FFFFFF" backcolor="#999999"/>
        <textElement>
          <font size="12"/>
        </textElement>
        <text><![CDATA[LASTNAME]]></text>
```

```
    </staticText>
    <staticText>
      <reportElement mode="Opaque" x="414" y="0" width="138" height="18"
      forecolor="#FFFFFF" backcolor="#999999"/>
      <textElement>
        <font size="12"/>
      </textElement>
      <text><![CDATA[STREET]]></text>
    </staticText>
  </band>
</columnHeader>

<detail>
  <band height="20">
    <textField hyperlinkType="None">
      <reportElement x="0" y="0" width="138" height="20"/>
      <textElement>
        <font size="12"/>
      </textElement>
      <textFieldExpression class="java.lang.Integer"><![CDATA[$F{ID}]]>
      </textFieldExpression>
    </textField>
    <textField hyperlinkType="None">
      <reportElement x="138" y="0" width="138" height="20"/>
    </textField>
      <textElement>
        <font size="12"/>
      </textElement>
      <textFieldExpression class="java.lang.String"><![CDATA[$F{FIRSTNAME}]]>
      </textFieldExpression>
    <textField hyperlinkType="None">
      <reportElement x="276" y="0" width="138" height="20"/>
      <textElement>
        <font size="12"/>
      </textElement>
      <textFieldExpression class="java.lang.String"><![CDATA[$F{LASTNAME}]]>
      </textFieldExpression>
    </textField>
    <textField hyperlinkType="None">
      <reportElement x="414" y="0" width="138" height="20"/>
      <textElement>
        <font size="12"/>
      </textElement>
      <textFieldExpression class="java.lang.String"><![CDATA[$F{STREET}]]>
      </textFieldExpression>
    </textField>
  </band>
</detail>

<columnFooter>
  <band/>
</columnFooter>
<pageFooter>
  <band height="26">
    <textField evaluationTime="Report" pattern="" isBlankWhenNull="false"
    hyperlinkType="None">
      <reportElement key="textField" x="516" y="6" width="36" height="19"
      forecolor="#000000" backcolor="#FFFFFF"/>
```

```
        <textElement>
          <font size="10"/>
        </textElement>

        <textFieldExpression class="java.lang.String"><![CDATA["" +
        $V{PAGE_NUMBER}]]></textFieldExpression>
      </textField>
      <textField pattern="" isBlankWhenNull="false" hyperlinkType="None">
        <reportElement key="textField" x="342" y="6" width="170" height="19"
        forecolor="#000000" backcolor="#FFFFFF"/>
        <box>
          <topPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <leftPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <bottomPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <rightPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
        </box>
        <textElement textAlignment="Right">
          <font size="10"/>
        </textElement>
        <textFieldExpression class="java.lang.String"><![CDATA["Page " +
        $V{PAGE_NUMBER} + " of "]]></textFieldExpression>
      </textField>

      <textField pattern="" isBlankWhenNull="false" hyperlinkType="None">
        <reportElement key="textField" x="1" y="6" width="209" height="19"
        forecolor="#000000" backcolor="#FFFFFF"/>
        <box>
          <topPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <leftPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <bottomPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
          <rightPen lineWidth="0.0" lineStyle="Solid" lineColor="#000000"/>
        </box>
        <textElement>
          <font size="10"/>
        </textElement>
        <textFieldExpression class="java.util.Date"><![CDATA[new Date()]]>
        </textFieldExpression>
      </textField>
    </band>
  </pageFooter>
  <summary>
    <band/>
  </summary>
</jasperReport>
```

During compilation of the JRXML file (using some JasperReports classes) the XML is parsed and loaded in a JasperDesign object, which is a rich data structure that allows you to represent the exact XML contents in memory. Regardless of the language used for expressions inside the JRXML, JasperReports creates a special Java class that represents the whole report. The report is then compiled, instanced, and serialized in a JASPER file, ready for loading at any time.

JasperReports' speedy operation is due to all of a report's formulas being compiled into Java-native bytecode and the report structure being verified during compilation instead of at run time. The JASPER file contains no extraneous resources, such as images used in the report, resource bundles to run the report in different languages, or extra scriptlets and external style definitions. All these resources must be provided by the host application and located at run time.

2.2 Data Sources and Print Formats

Without a means of supplying content from a dynamic data source, even the most sophisticated and appealing report would be useless. JasperReports gives you two ways to specify fill data for the output report: parameters and data sources. Both kinds of data are presented by means of a generic interface named `JRDataSource`.

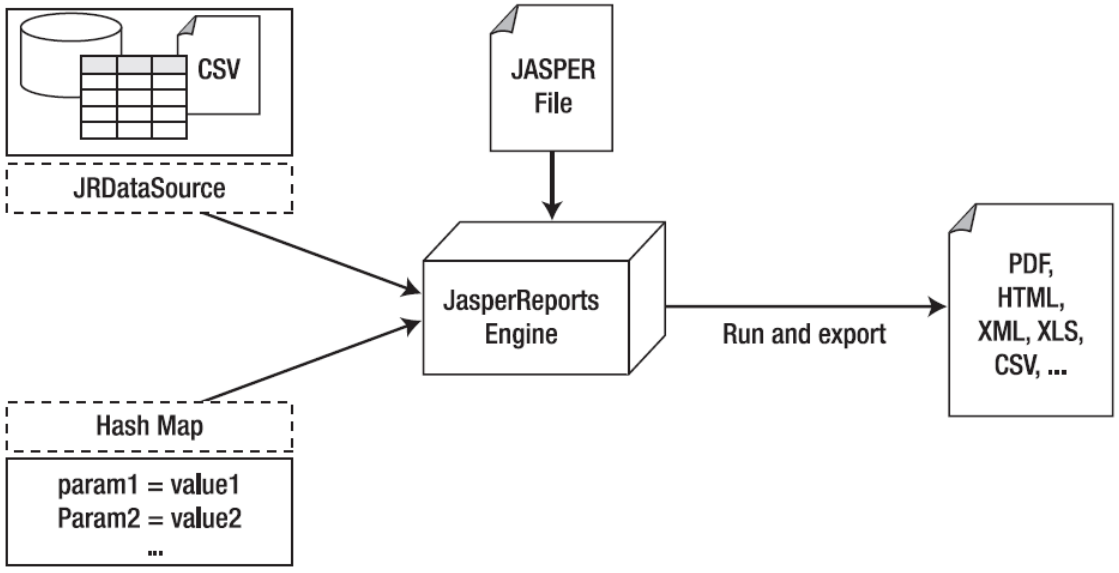


Figure 2-1 Data Source and Parameter Flows for Report Creation

`JRDataSource` allows a set of records organized in tables (rows and columns) to be read. It enables JasperReports to fill a report with data from an explicit data source, using a JDBC connection (already instanced and opened) to whichever relational database you want to run an SQL query on (which is specified in the report).

If the data don't meet your requirements, you may need to specify values to condition the report's execution; you can create name/value pairs to pass to the print engine. These pairs are named parameters, and they have to be preventively declared in the report. Through `fillManager`, you can join a `JASPER` file and a data source in a JasperPrint object. This object is a meta-print that can create a real print after you export it in the format of your choice through appropriate classes that implement the `JRExporter` interface.

JasperReports give you pre-defined exporters, such as those for creating files formatted as `PDF`, `XLS`, CVS, XML, RTF, ODF, text, HTML and SWF. Through the `JRViewer` class, you can view the print directly on the screen and print a hardcopy.

2.3 Expressions

Many settings in a report are defined by formulas, such as conditions that can hide an element, special calculations, or text processing that requires knowledge of a scripting language.

Formulas can be written in at least three languages, two of which (JavaScript and Groovy) can be used without knowledge of programming methods.

All formulas in JasperReports are defined through expressions. The default expression language is Java, but if you're not a programmer, we recommend JavaScript or Groovy, because those languages hide a lot of the Java complexity. The language is a property of the document. To set it, select the document root node in the Outline view and choose your language in the `Language` property in the Properties view.

An expression is a formula that operates on some values and returns a result, like a formula in a spreadsheet cell. A cell can have a simple value or a complex formula that refers to other values. In a spreadsheet you refer to values contained in other cells; in JasperReports you use the report fields, parameters, and variables. Whatever is in your expression, when it's computed, it returns a value (which can be null).

2.3.1 Expression Types

An expression's type is determined by the context in which the expression is used. For example, if your expression is used to evaluate a condition, the expression should be Boolean (true or false); if you're creating an expression to display in a text field, it's probably a String or a number (Integer or Double). Using the right type is crucial; JasperReports requires precision when choosing an expression type.

Some of the most important Java types are:

`java.lang.Boolean`	Defines an Object that represents a boolean value such as true and false
`java.lang.Byte`	Defines an Object that represents a byte
`java.lang.Short`	Defines an Object that represents an short integer
`java.lang.Integer`	Defines an Object that represents integer numbers
`java.lang.Long`	Defines an Object that represents long integer numbers
`java.lang.Float`	Defines an Object that represents floating point numbers
`java.lang.Double`	Defines an Object that represents real numbers
`java.lang.String`	Defines an Object that represents a text
`java.util.Date`	Defines an Object that represents a date or a timestamp
`java.lang.Object`	A generic java Object

If an expression is used to determine the value of a condition that determines, for instance, whether an element should be printed, the return type is `java.lang.Boolean`. To create it, you need an expression that returns an instance of a Boolean object. Similarly, if an expression shows a number in a text field, the return type is `java.lang.Integer` or `java.lang.Double`.

Neither JavaScript nor Groovy is particularly formal about types, since they are not typed languages; the language itself treats a value in the best way by trying to guess the value type or by performing implicit casts (conversion of the type).

2.3.2 Expression Operators and Object Methods

Operators in Java, Groovy and JavaScript are similar because these languages share the same basic syntax. Operators can be applied to a single operand (unary operators) or on two operands (binary operators). The

following table shows a number of operators, but it is not a complete list. For example, there is a unary operator to add 1 to a variable (++), but it is easier to use x + 1.

Table 2-2 Expression operators

Operator	Description	Example
+	Sum (it can be used to sum two numbers or to concatenate two strings)	A + B
-	Subtraction	A - B
/	Division	A / B
%	Rest, it returns the rest of an integer division	A % B
||	Boolean operator OR	A || B
&&	Boolean operator AND	A && B
==	Equals	A == B
!=	Not equals	A != B
!	Boolean operator NOT	!A

Regarding the Equals operator: in Java, the == operator can only be used to compare two primitive values. With objects, you need to use the special method “equals”; for example, you cannot write an expression like “test” == “test”, you need to write “test”.equals(“test”).

Regarding the Equals operator: in Java, the != operator can only be used to compare two primitive values.

Within an expression, you can use the syntax summarized in **Table 2-3, “Syntax for referring to report objects,” on page 30** to refer to the parameters, variables, and fields defined in the report.

Table 2-3 Syntax for referring to report objects

Syntax	Description
`$F{name_field}`	Specifies the `name_field` field ("F" means field).
`$V{name_variable}`	Specifies the `name_variable` variable.
`$P{name_parameter}`	Specifies the `name_parameter` parameter.
`$P!{name_parameter}`	Special syntax used in the report SQL query to indicate that the parameter does not have to be dealt as a value to transfer to a prepared statement, but that it represents a little piece of the query.

Syntax	Description
`$R{resource_key}`	Special syntax for localization of strings.

`$X{functionName, col_name, param1, [param2]}`	Syntax for complex queries, such as comparing a column value to a parameter value. Based on the function in the first argument, JasperReports constructs a SQL clause. The following functions are available:
	• Functions expecting three arguments for `$X{}` – `EQUAL`, `NOTEQUAL`, `LESS`, `LESS]` (less than or equal to), `GREATER`, `[GREATER` (greater than or equal to), `IN`, `NOTIN`. For example:
	`$X{EQUAL, order_date, date_parameter}`
	• Functions expecting four arguments for `$X{}` –
	`BETWEEN` (excludes both endpoints)
	`BETWEEN]` (includes right endpoint)
	`[BETWEEN` (includes left endpoint)
	`[BETWEEN]` (includes both endpoints)
	For example:
	`$X{BETWEEN, order_date, start_date_param, end_date_param}`

In summary, fields, variables and parameters represent objects; specify their type when you declare them within a report.

Although expressions can be complicated, usually it is a simple operation that returns a value. There is a simple if-else expression that is very useful in many situations. An expression is just an arbitrary operation that any stage must represent a value. In Java, these operators can be applied only to primitive values, except for the sum operator (+). The sum operator can be applied to a String expression with the special meaning of concatenate. For example:

```
$F{city} + ", " + $F{state}
```

results in a string like:

```
San Francisco, California
```

Any object in an expression can include methods. A method can accept zero or more arguments, and it can return or not a value. In an expression you can use only methods that return a value; otherwise, you would have nothing to return from your expression. The syntax of a method call is:

```
Object.method(argument1, argument2, <etc.>)
```

Some examples:

Expression	Result
`"test".length()`	`4`
`"test".substring(0, 3)`	`"tes"`
`"test".startsWith("A")`	`false`
`"test".substring(1, 2).startsWith("e")`	`true`

The methods of each object are usually explained in the JasperReports Library Javadocs, which that are available at http://jasperreports.sourceforge.net/api/.

You can use parentheses to isolate expressions and make the overall expression more readable.

2.3.3 Using an If-Else Construct in an Expression

A way to create an if-else-like expression is by using the special question mark operator. For example:

```
(($F{name}.length() > 50) ? $F{name}.substring(0,50) : $F{name})
```

The syntax is `(<condition>) ? <value on true> : <value on false>`. It is extremely useful, and can be recursive, meaning that the value on `true` and `false` can be represented by another expression which can be a new condition:

```
(($F{name}.length() > 50) ?
(($F{name}.startsWidth("A")) ? "AAAA" : "BBB")
 :
 $F{name})
```

This expression returns the String `AAAA` when the value of the field name is longer than 50 characters and starts with A, returns BBB if it is longer than 50 characters but does not start with A, and, finally, returns the original field value if neither of these conditions is true.

Despite the possible complexity of an expression, it can be insufficient to define a needed value. For example, if you want to print a number in Roman numerals or return the name of the weekday of a date, it is possible to transfer the elaborations to an external Java class method, which must be declared as static, as shown in the following example:

```
MyFormatter.toRomanNumber( $F{MyInteger}.intValue() )
```

The function operand `toRomanNumber` is a static method of the `MyFormatter` class, which takes an `int` as argument (the conversion from Integer to `int` is done by means of the `intValue()` method; it is required only when using Java as language) and gives back the Roman version of a number in a lace.

This technique can be used for many purposes; for example, to read the text from a CLOB field or to add a value into a HashMap (a Java object that represents a set of key/value pairs).

2.4 Using Java as a Language for Expressions

Java was the first language supported by JasperReports and is still the most commonly-used language as well as being the default.

Following are some examples of Java expressions:

- `"This is an expression"`
- `new Boolean(true)`
- `new Integer(3)`
- `(($P{MyParam}.equals("S")) ? "Yes" : "No")`

The first thing to note is that each of these expressions represents a Java Object, meaning that the result of each expression is a non-primitive value. The difference between an object and a primitive value makes sense only in Java, but it is very important: a primitive value is a pure value like the number 5 or the Boolean value true.

Operations between primitive values have as a result a new primitive value, so the expression:

```
5+5
```

results in the primitive value 10. Objects are complex types that can have methods, can be null, and must be "instanced" with the keyword "new" most of the time. In the second example above, for instance (`new Boolean (true)`), we must wrap the primitive value `true` in an object that represents it.

By contrast, in a scripting language such as Groovy and JavaScript, primitive values are automatically wrapped into objects, so the distinction between primitive values and objects wanes. When using Java, the result of our expression must be an object, which is why the expression 5+5 is not legal as-is but must be fixed with something like this:

```
new Integer( 5 + 5 )
```

The fix creates a new object of type `Integer` representing the primitive value 10.

So, if you use Java as the default language for your expressions, remember that expressions like the following are not valid:

- `3 + 2 * 5`
- `true`
- `(($P{MyParam} == 1) ? "Yes" : "No")`

These expressions don't make the correct use of objects. In particular, the first and the second expressions are not valid because they are of primitive types (`integer` in the first case and `boolean` in the second case) which do not produce an object as a result. The third expression is not valid because it assumes that the `MyParam` parameter is a primitive type and that it can be compared through the == operator with an `int`, but it cannot. In fact, we said that parameters, variables, and fields are always objects and primitive values cannot be compared or used directly in a mathematical expression with an object.

2.5 Using Groovy as a Language for Expressions

The modular architecture of JasperReports provides a way to plug in support for languages other than Java. By default, the library supports bsh, Groovy and JavaScript.

Groovy is a full language for the Java 2 Platform. Inside the Groovy language you can use all classes and JARs that are available for Java. The following table compares some typical JasperReports expressions written in Java and Groovy:

Table 2-4 Groovy and Java code samples

Expression	Java	Groovy
Field	`$F{field_name}`	`$F{field_name}`
Sum of two double fields	`new Double($F{f1}.doubleValue() + $F{f2}.doubleValue())`	`$F{f1} + $F {f2}`
Comparison of numbers	`new Boolean($F{f}.intValue() == 1)`	`$F{f} == 1`
Comparison of strings	`new Boolean($F{f} != null && $F{f}.equals("test"))`	`$F{f} == "test"`

The following is a correct Groovy expression:

```
new JREmptyDataSource($F{num_of_void_records})
```

`JREmptyDataSource` is a class of JasperReports that creates an empty record set (meaning with the all fields set to null). You can see how you can instance this class (a pure Java class) in Groovy without any problem. At the same time, Groovy allows you to use a simple expression like this one:

```
5+5
```

The language automatically encapsulates the primitive value 10 (the result of that expression) in a proper object. Actually, you can do more: you can treat this value as an object of type `String` and create an expression such as:

```
5 + 5+ "my value"
```

Whether or not such an expression resolves to a rational value, it is still a legal expression and the result is an object of type `String` with the value:

```
10 my value
```

Hiding the difference between objects and primitive values, Groovy allows the comparison of different types of objects and primitive values, such as the legal expression:

```
$F{Name} == "John"
```

This expression returns true or false, or, again:

`$F{Age} > 18`	Returns true if the `Age` object interpreted as a number is greater than 18.
`"340" < 100`	Always returns false.
`"340".substring(0,2) < 100`	Always returns true (since the substring method call produces the string "34", which is less than 100).

Groovy provides a way to greatly simplify expressions and never complains about null objects that can crash a Java expression throwing a `NullPointerException`. It really does open the doors of JasperReports to people who don't know Java.

2.6 Using JavaScript as a Language for Expressions

JavaScript is a popular scripting language with a syntax very similar to Java and Groovy. JavaScript has a set of functions and object methods that in some cases differ from Java and Groovy. For example, the method `String.startsWith(...)` does not exist in JavaScript. You can still use Java objects in JavaScript. An example is:

```
(new java.lang.String("test")).startsWith("t")
```

This is a valid JavaScript expression creating a Java object (in this case a `java.lang.String`) and using its methods.

JavaScript is the best choice for users who have no knowledge of other languages. The other significant advantage of JavaScript is that it is not interpreted at run time, but instead generates pure Java byte-code. As a result, it offers almost the same performance as Java itself.

2.7 Using JasperReports Extensions in Jaspersoft Studio

JasperReports provides several ways to extend its functionality. In general, extensions (like components, fonts, query executors, chart themes, and so on) are packaged in JARs. To use these extensions in Jaspersoft Studio, just add the required JARs to the Jaspersoft Studio classpath. The Jaspersoft Studio classpath is composed of static and reloadable paths. Extensions must be set as static paths, while objects that don't require a proper descriptor or special loading mechanism (such as scriptlets and custom data sources) can be reloadable.

2.8 A Simple Program

In conclusion, following is an example of a simple program that shows how to produce a PDF file from a Jasper file using a data source named `JREmptyDataSource`, a utility data source that provides zero or more records without fields. The file test.jasper, referenced in the example, is the compiled version of the code in **Table 2-1 on page 24**.

Table 2-5 JasperTest.java

```
import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.export.*;
import java.util.*;
public class JasperTest
{
       public static void main(String[] args)
       {
               String fileName = "/devel/examples/test.jasper";
               String outFileName = "/devel/examples/test.pdf";
               HashMap hm = new HashMap();
               try
               {
                       JasperPrint print = JasperFillManager.fillReport(
                               fileName,
                               hm,
                               new JREmptyDataSource());
                       JRExporter exporter =
                               new net.sf.jasperreports.engine.export.JRPdfExporter();

                        exporter.setParameter(
                               JRExporterParameter.OUTPUT_FILE_NAME,
                               outFileName);
                       exporter.setParameter(
                       JRExporterParameter.JASPER_PRINT,print);
                       exporter.exportReport();
                       System.out.println("Created file: " + outFileName);
               }
               catch (JRException e)
               {
                       e.printStackTrace();
                       System.exit(1);
               }
               catch (Exception e)
               {
                       e.printStackTrace();
                       System.exit(1);
               }
       }
}
```

CHAPTER 3 REPORT ELEMENTS

The basic building block of a report is the element. An element is a graphical object, such as a text string or a rectangle. In Jaspersoft Studio, the concept of line or paragraph does not exist, as it does in word processing programs. Everything is created by means of elements, which can contain text, create tables, display images, and so on. This approach follows the model used by the majority of report authoring tools.

Jaspersoft Studio relies on all the basic elements provided in the JasperReports library:

- Line
- Rectangle
- Ellipse
- Static text
- Text field (or simply Field)
- Image
- Frame
- Subreport
- Crosstab
- Chart
- Break

Combining these elements,you can produce every kind of report. JasperReports also allows developers to implement their own generic elements and custom components for which they can add support in Jaspersoft Studio to create a proper plug-in.

This chapter contains the following sections:

3.1 Basic Element Properties

All the elements have a set of common properties. Other properties are specific to element type. An element's properties determine its appearance and position on the page. You can access the properties of a selected element in the Properties tab (by default in the upper right area of the UI). In Jaspersoft Studio you place elements within bands (containers). Depending on the elements it contains, you can change the vertical size of a band.

Elements appear on the Palette tab, bu default, in the top right of the UI.

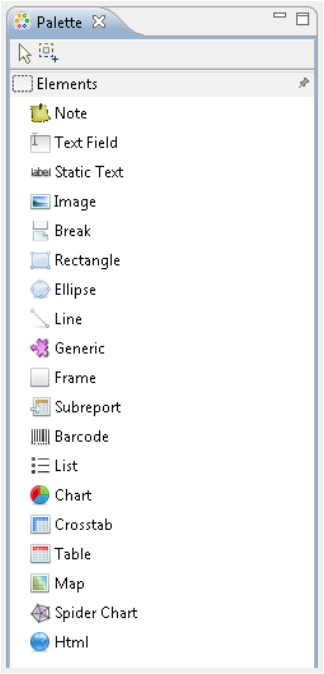


Figure 3-1 Elements in the Designer Palette

Element properties are divided into categories, visible via tabs in the Properties view. The attributes available depend on the element type.

- The **Appearance** tab allows you to set the location, size, color, and text style of the field.
- The **Borders** tab allows you to set the padding and border style, color, and width of the field.
- The **Static Text** tab allows you to define unchangeable text for a field, and control its appearance.
- The **Text Field** tab allows you to place text in an element.
- The **Inheritance** tab allows you to view any attributes inherited from another level, and override those attributes when possible.
- The **Hyperlink** tab allows you to define a hyperlink in an element.
- The **Advanced** tab displays detailed information about the element.

Appearance and Borders are shared among many elements and contain the attributes that define the graphical appearance of the elements, such the width of its borders, the background color, and so on. The Image category is a special one, because it is present only for the elements of the type Image.

Commonly, the value of an attribute is undefined, and it has a common default value. This means that the element does not have a specific behavior defined, but gets a behavior from somewhere else. For example, the

default value of the attribute "background color" is undefined in most of cases, but when a non-transparent element is added to the designer, we can see that it has a white background. The value of the background color attribute is inherited from a lower level.

3.2 Inserting, Selecting, and Positioning Elements

3.2.1 Inserting Elements

Drag an element from the palette to place it in the report editing area.

3.2.2 Selecting Elements

Click to select an element in the report editing area. Drag to adjust the element's position or change its size by selecting it and dragging a corner of the selection frame. To select several elements at the same time draw a rectangle around them; shift-clicking works as well.

When two or more elements are selected, only their common properties are displayed in the Properties view. If the values of the properties are different, the value fields are blank (usually the field is shown empty). To edit properties unique to one element, select only that element.

3.2.3 Positioning Elements

Jaspersoft Studio offers a number of ways to place the elements in your report with precision.

3.2.3.1 Using the Grid

To show a grid for aligning elements in the page, go to **View > Show Grid** from the main menu. To force the elements to snap to the grid, also select **Snap to Grid**.

3.2.3.2 Using Bands

The top and left values that define the element’s position are always relative to the parent container (a band or frame).

If you want to move an element from one band to another or to a frame, drag the element node from the Outline view to the new band (or frame) node.

In the report editing area, you can drag an element from one band to another band, but the element’s parent band does not change. In general, an element must stay in its band, but there are exceptions to this rule. For example, you can move an element anywhere in the report without changing or updating the parent band.

3.2.3.3 Guides

When dragging or resizing an element, Jaspersoft Studio suggests places to align it based on the elements currently in the design pane, the band bounds, and any guides. When the element you're moving or resizing is in line with another element in the report, a guideline appears, allowing you align the elements. To force elements to align with guidelines, select **View > Snap to Guides** from the main menu.

You can drag and change the position of a guideline at any time with no effect on the element’s position.

To remove a guideline, drag it to the upper-left corner of the report editing area.

3.2.3.4 The Properties View

You can use the Properties tab to edit an element's properties. By default the Properties tab is at the right side of the UI. The Properties view for more than just for elements. You'll use it to edit all the components of a report. When you select something in the designer or the Outline tab, the Properties view shows the options specific to that object.

3.3 Formatting Elements

Formatting tools help organize the elements in the report. Right-click the element you want to work on and select a tool from the context menu.

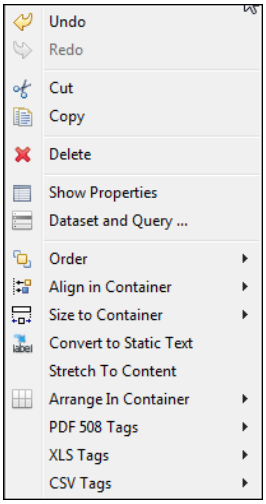


Figure 3-2 Formatting Tools Menu

The tools in the context menu are specific to the selected item(s). The following tables explain the tools.

Table 3-1 Formatting Tools

Icon	Tool Name	Description	Multiple Select?
Order Tools			
	Send Backward	Moves the element behind its current layer.	Yes
	Send to Back	Moves the element to the bottom layer.	Yes
Align in Container Tools			
	Align to Left	Aligns the left sides to that of the primary element.	Yes

Icon	Tool Name	Description	Multiple Select?
	Align to Center	Aligns the centers to that of the primary element.	Yes
	Align to Right	Aligns the right sides to that of the primary element.	Yes
	Align to Top	Aligns the top sides (or the upper part) to that of the primary element.	Yes
	Align to Middle	Aligns the middles to that of the primary element.	Yes
	Align to Bottom	Aligns the bottom sides (or the lower part) to that of the primary element.	Yes
Size Components Tools			
	Match Width	Adjusts width to that of primary element.	Yes
	Match Height	Adjusts height to that of primary element.	Yes
	Match Size	Resizes to that of primary element.	Yes
Size to Container Tools			
	Fit to Width	Adjusts elements to fill width of container.	Yes
	Fit to Height	Adjusts elements to fill height of container.	Yes
	Fit to Both	Adjusts elements to fill width and height of container.	Yes
Arrange in Container Tools			
	Horizontal Layout	Centers selected elements vertically.	Yes
	Vertical Layout	Centers selected elements horizontally	Yes

Icon	Tool Name	Description	Multiple Select?
Miscellaneous Tools			
	Stretch to Content	Resizes element to fit the content	N/A
	PDF 508 Tags		
	XLS Tags		

3.4 Graphic Elements

Graphic elements like lines and shapes are used to make reports more attractive and readable. You can also add these by dragging them from the Designer palette to the report editing area.

3.4.1 Line

In Jaspersoft Studio, a line is defined by a rectangle for which the line represents the diagonal. By default, the foreground color is used as the default color and a 1-pixel-width line is used as the line style.

You can customize the look, style, and direction of the line in the element's Properties view.

3.4.2 Rectangle and Ellipse

The rectangle element is usually used to draw frames around other elements. By default, the foreground color setting is used and a normal 1 pixel width

The ellipse is the only element that has no attributes specific to it. The ellipse is drawn in a rectangle that defines the maximum height and width. By default, the foreground color is used, and a normal 1-pixel-width line is used as line style. The background is filled with the background color setting if the element has not been defined as transparent.

3.4.3 Images

An image is the most complex of the graphic elements. You can insert raster images (such as GIF, PNG and JPEG images) in the report, but you can also use an image element as a canvas object to render, for example, a Swing component, or to leverage some custom rendering code.

Dragging an image element from the Palette into the report editing area launches the **Create new image element** dialog. This is the most convenient way to specify an image to use in the report. Jaspersoft Studio does not save or store the selected image anywhere, it just uses the file location, translating the absolute path of the selected image into an expression to locate the file when the report is executed. The expression is then set as the value for the `Image Expression` property.

You can add an image by explicitly defining the full absolute path of the image file in your expression. This is an easy way to add an image to the report, but, overall, it has a big impact on the report's portability, since the file may not be found on another machine (for instance, after deploying the report on a web server or running the report on a different computer).

3.4.4 Padding and Borders

For the image and text elements you can visualize a frame or define a particular padding (the space between the element border and its content) for the four sides. Border and padding are specified by selecting the element in the report editing area, and using the Properties view.

In the Properties view, click the **Borders** option. This includes the following controls:

- Padding allows you to define padding widths for each of the four sides, or to apply the same value to all sides.
- Borders allow you to select their color, style, and width, as well as choose where it appears.

As always, all the measurements are shown in pixels.

3.5 Text Elements

Two elements are specifically designed to display text in a report: static text and text field. Static text is used for creating labels or to print static text set at design time, that is not meant to change when the report is generated. That said, in some cases you still use a text field to print labels too, since the nature of the static text elements prevents the ability to display text dynamically translated in different languages when the report is executed with a specific locale and it is configured to use a resource bundle leveraging the JasperReports internationalization capabilities.

A text field is similar to a static text string, but the content (the text in the field) is provided using an expression (which can be a simple static text string itself). That expression can return several kinds of value types, allowing the user to specify a pattern to format that value. Since the text specified dynamically can have an arbitrary length, a text field provides several options about how the text must be treated regarding alignment, position, line breaks and so on. Optionally, the text field is able to grow vertically to fit the content when required.

By default both text elements are transparent with no border, with a black text color. The most used text properties can be modified using the text tool bar displayed when a text element is selected. Text element properties can also be modified using the Properties view.

Text fields support hyperlinks as well. See **3.8.4, "Defining Hyperlinks," on page 49** for more information.

3.5.1 Static Text

The static text element is used to show non-dynamic text in reports. The only parameter that distinguishes this element from a generic text element is the Text property, where the text to view is specified: it is normal text, not an expression, and so it is not necessary to enclose it in double quotes in order to respect the conventions of Java, Groovy, or JavaScript syntax.

3.5.2 Text Fields

A text field allows you to print an arbitrary section of text (or a number or a date) created using an expression. The simplest case of use of a text field is to print a constant string (`java.lang.String`) created using an expression like this:

```
"This is a text"
```

A text field that prints a constant value like the one returned by this expression can be easily replaced by a static field; actually, the use of an expression to define the content of a text field provides a high level of control on the generated text (even if it's just constant text). A common case is when labels have to be internationalized and loaded from a resource bundle. In general, an expression can contain fields, variables and parameters, so you can print in a text field the value of a field and set the format of the value to present. For this purpose, a text field expression does not have to return necessarily a string (that's a text value): the `text field expression class name` property specifies what type of value is returned by the expression. It can be one of the following:

Valid Expression Types		
`java.lang.Object`	`java.sql.Time`	`java.lang.Long`
`java.lang.Boolean`	`java.lang.Double`	`java.lang.Short`
`java.lang.Byte`	`java.lang.Float`	`java.math.BigDecimal`
`java.util.Date`	`java.lang.Integer`	`java.lang.String`
`java.sql.Timestamp`	`java.io.InputStream`	

An incorrect expression class is frequently the cause of compilation errors. If you use Groovy or JavaScript you can choose `String` as expression type without causing an error when the report is compiled. The side effect is that without specifying the right expression class, the pattern (if set) is not applied to the value.

Let's see what properties can be set for a text field:

`Blank when null`	If set to true, this option avoids printing the text field content if the expression result is a null object that would be produce the text `"null"` when converted in a string.
`Evaluation time`	Determines in which phase of the report creation the `Text field Expression` has to be elaborated.
`Evaluation group`	The group to which the evaluation time is referred if it is set to `Group`.
`Stretch with overflow`	When it is selected, this option allows the text field to adapt vertically to the content, if the element is not sufficient to contain all the text lines.
`Pattern`	The pattern property allows you to set a mask to format a value. It is used only when the expression class is congruent with the pattern to apply, meaning you need a numeric value to apply a mask to format a number, or a date to use a date pattern.

3.6 Using Frames

A frame is an element that can contain other elements and optionally draw a border around them. Since a frame is a container of other elements, in the document outline view the frame is represented as a node containing other elements.

A frame can contain other frames, and so on recursively. To add an element to a frame, just drag the new element from the palette inside the frame. Alternatively you can use the outline view and drag elements from a band into the frame and so on. The position of an element is always relative to the container position. If the container is a band, the element position is relative to the top of the band and to the left margin. If the container (or element parent) is a frame, the element coordinates are relative to the top left corner of the frame. Since an element dragged from a container to another does not change its top/left properties, when moving an element from a container to another its position is recalculated based on the new container location.

The advantages of using a frame to draw a border around a set of elements, with respect to using a simple rectangle element, are:

- When you move a frame, all the elements contained in the frame move.
- While using a rectangle to overlap some elements, the elements inside the rectangle are not treated as if they overlap (respect to the frame), so you don't have problems when exporting in HTML (which does not support overlapped elements).
- Finally, the frame automatically stretches according to its content, and the element `position type` property of its elements refer to the frame itself, not to the band, making the design a bit easier to manage.

3.7 Inserting Page and Column Breaks

Page and column breaks are used to force the report engine to make a jump to the next page or column. A column break in a single column report has the same effect as a page break.

In the design view they are represented as a small line. If you try to resize them, the size is reset to the default, this because they are used just to set a particular vertical position in the page (or better, in the band) at which Jaspersoft Studio forces a page or column break.

The type of break can be changed in the Properties view.

3.8 Anchors, Bookmarks, and Hyperlinks

This section describes the creation of hyperlinks for most types of elements. For HTML5 charts, they are defined in a different way, as described in **3.10, "Hyperlinks in HTML5 Charts," on page 52**.

Image, textfield, and chart elements can be used both as anchors into a document and as hypertext links to external sources or other local anchors. JasperReports provides a powerful combination of settings to define hyperlinks. While a hyperlink usually opens a specific URL, JasperReports broadens the concept, extending it to a more complex object that can be used for more complicated functionality, such as executing a report in JasperReports to perform a drill-down or drill-up operation, pointing to a page within a PDF document, and so on.

To set anchor, bookmark, or hyperlink properties, go to the **Hyperlink** tab in the **Properties** view.

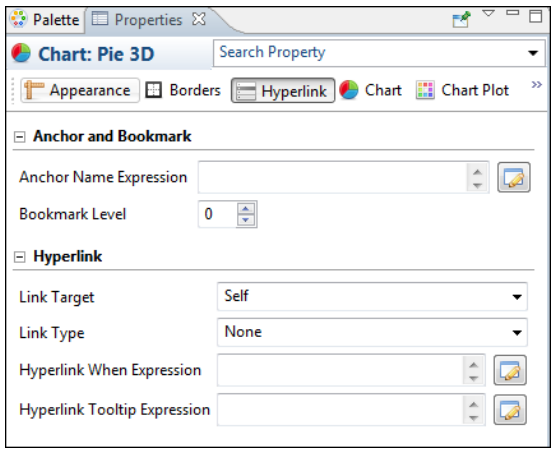


Figure 3-3 Anchor, Bookmark, and Hyperlink Properties

This window is divided in two sections:

- Anchor and Bookmark
- Hyperlink details

Use Anchor and Bookmark to create an anchor inside the current document; it is only available for elements that support hyperlinks (textfield, image, and chart). The name of the anchor is specified by means of an expression, and the bookmark level is used when the report is exported in PDF to create the Table of Content. Bookmarks can also be displayed in JasperReports Server when the report is displayed in the interactive viewer.

Note for developers:

In JasperReports 5.6 and later the same table of content is also available in the JasperPrint object, and can be explored by calling the method:

```
List<PrintBookmark> getBookmarks()
```

3.8.1 Anchors and Bookmarks

An anchor is a kind of label that identifies a specific position in a document. The **Hyperlink** tab includes a text area where you can enter an expression to be the name of the anchor. This name can be referenced by other links. Clicking the  button opens the **Expression Editor**, in which you write or build the expression.

If you plan to export your report as a PDF, set a bookmark level to populate the bookmark tree, making the final document navigation much easier. To make an anchor available in a bookmark, simply choose a bookmark level higher than 1. Defining different levels creates nested bookmarks.

3.8.2 Overview of Hyperlink Configuration

The definition of a hyperlink, given the number of options, may look a little bit cumbersome, but it is very simple in most of the use cases, in example if we really just want to link to an URL.

The Link Target allows to specify where the link should be open. This is probably the less important option, but since it is the first we encounter, it makes sense start from here (note: we may think to change its position, and move it at the bottom). The Link Target is very similar to the target attribute of an HTML anchor. The drop down box to set it, proposes the most common options (Self, Blank, Top, Parent), but it is still possible to

manually specify a target name, which actually makes sense only when the hyperlink is used in a web environment (which is when the final export format is HTML like).

There are several type of hyperlink, and each of them can be used for different scopes. The built-in types provided by JasperReports are: Reference, Local Anchor, Local Page, Remote Anchor Remote Page. The first type, Reference, is the one used to create a typical URL link to some web address like http://www.jaspersoft.com/ and it is rendered in all the output formats supporting web link (include for instance Microsoft Excel and Word).

The other four are used to point to a position within the current or an external document, and primarily work in PDF and HTML.

Selecting a type, the user interface change accordingly to the information required by the type itself.

Please note the difference between URL parameters and hyperlink parameters. In the panel to configure a hyperlink there is a section called parameters. When working with a hyperlink of type Reference, these parameters are ignored: they don't have anything to do with the parameters that appear in a regular URL (in particular, the part of the URL following the question mark). The URL should contain a parameter; you can use the Reference Expression. For example:

```
"http://www.someurl.com/search?city=" + $F{city} + "&country= " + $F{country}
```

The values of the fields city and country are used to dynamically build a URL.

So why are these parameters interesting?Because they can be used by URL Handlers, especial JasperReports Library extensions, to manipulate and generate a hyperlinks.

The hyperlink type `ReportExecution` is one of these custom hyperlink extensions. It is used to execute a JasperReports Server report from another JasperReports Server report. This hyperlink type requires that the parameter `_report` is defined and points to the JasperReports Server report to execute. If this destination report contains one or more input controls, their value can be set by specifying the name of the input control as the parameter name and providing a value.

A `ReportExecution` hyperlink is configured in Jaspersoft Studio by simply dragging a report unit in the JasperReports Server repository explorer over an element that supports hyperlink (such as a `textfield`).

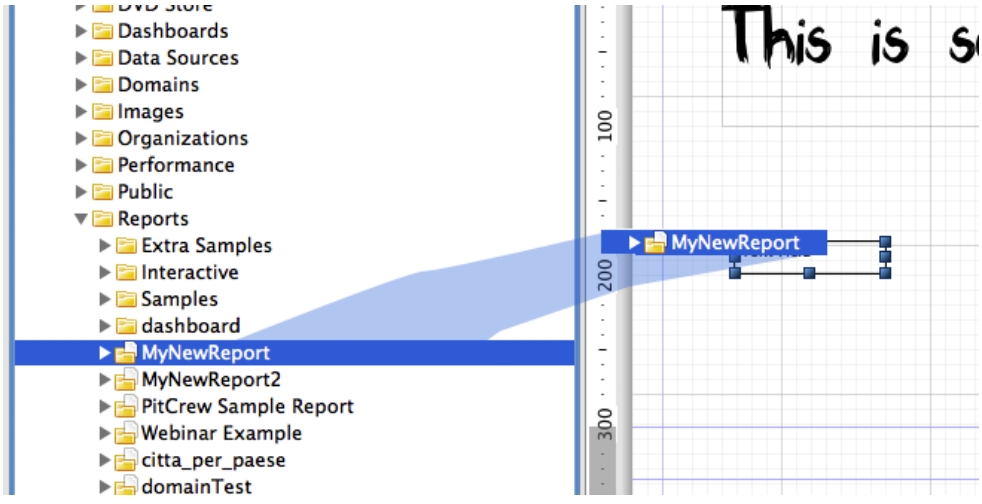


Figure 3-4 Dragging a report from the server into the Report Design

The auto configured hyperlink has the type set to ReportExecution and all the _report parameter set. Moreover, if the JasperReports Server Report has input controls, they are added to the list of parameters, ready to be populated with a proper value expression, as shown below.

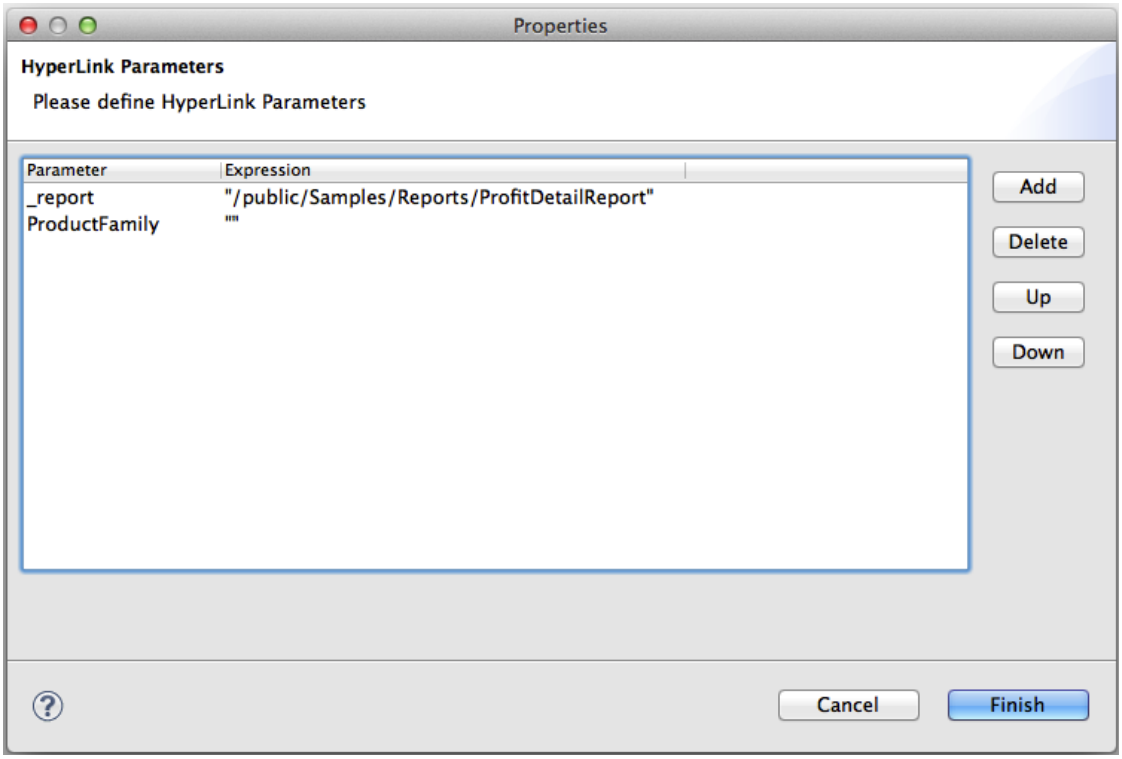


Figure 3-5 Configuring Hyperlink Parameters

So, in general a hyperlink is defined by a type, one or more expressions (in case of Reference, just the Reference Expression that contains the URL to invoke), and zero or more parameters, depending by the type (in a regular Jaspersoft Studio installation, only the type ReportExecution does actually accept parameters).

Additionally, a hyperlink may also be completed with a ToolTip (by using the Tooltip Expression) and a visibility condition (Print When Expression) which can be used to prevent the link to be considered if the print when expression does not return the boolean value True (note that an empty print when expression is considered as True).

3.8.3 Hyperlink Types

Jaspersoft Studio provides six types of built-in hypertext links: Reference, LocalAnchor, LocalPage, RemoteAnchor and RemotePage. Other types of hyperlinks can be implemented and plugged into JasperReports.

The following table describes the hyperlink types.

Reference	The Reference link indicates an external source that is identified by a normal URL. This is ideal to point, for example, to a servility to manage a record drill-down tools. The only expression required is the hyperlink reference expression.
LocalAnchor	To point to a local anchor means to create a link between two locations into the same document. It can be used, for example, to link the titles of a summary to the chapters to which they refer. To define the local anchor, specify a hyperlink anchor expression that produces a valid anchor name.
LocalPage	If instead of pointing to an anchor you want to point to a specific current report page, you need to create a LocalPage link. In this case, it is necessary to specify the page number you are pointing to by means of a hyperlink page expression (the expression has to return an Integer object)
RemoteAnchor	If you want to point to a particular anchor that resides in an external document, you use the RemoteAnchor link. In this case, the URL of the external file referenced must be specified in the Hyperlink Reference Expression field, and the name of the anchor must be specified in the Hyperlink Anchor Expression field.
RemotePage	This link allows you to point to a particular page of an external document. Similarly, in this case the URL of the external file pointed to must be specified in the **Hyperlink Reference Expression** field, and the page number must be specified by means of the hyperlink page expression. Some export formats have no support for hypertext links.
ReportExecution	This type of hyperlink is used to implement JasperReports Server's drill-down feature.

Some export formats have no support for hypertext links.

3.8.4 Defining Hyperlinks

Some types of datasets let you assign a hyperlink to the value represented in the chart; in the report output, clicking the chart opens a web page or navigates to a different location in the report.

The click-enabled area depends on the chart type. For example, in pie charts, the hyperlink is linked to each slice of the pie; in bar charts, the click-enabled areas are the bars themselves.

Image, text field, and chart elements can be used both as anchors into a document and as hypertext links to external sources or local anchors.

To create hyperlinks:

1. Click the **Hyperlink** tab in the **Properties** view.
2. In the **Link Target** drop-down, choose one of the following target types:
   - **Self**: This is the default setting. It opens the link in the current window.
   - **Blank**: Opens the target in a new window. Used for output formats such as HTML and PDF.
   - **Top**: Opens the target in the current window but outside eventually frames. Used for output formats such as HTML and PDF.

- **Parent**: Opens the target in the parent window (if available). Used for output formats such as HTML and PDF.

3. In the **Link Type** drop-down, choose whether the link type is None, Reference, LocalAnchor, LocalPage, RemoteAnchor, RemotePage, or ReportExecution.

   See **"Hyperlink Types" on page 48** for an explanation of the different choices.

4. Click the  button next to **Hyperlink When Expression** to create a tooltip for your hyperlink.
5. Save your report.

3.8.5 Hyperlink Parameters

Sometimes you need to define some parameters that must be attached to the link. The Link parameters table provides a convenient way to define them. The parameter value can be set using an expression. The parameter expression is supposed to be a string (since it is encoded in the URL). But when using custom link types it makes sense to set different types for parameters.

3.8.6 Hyperlink Tooltips

The tooltip expression is used to set a text to display as tooltip when the mouse is over the element that represents the hyperlink (this only works when the document is exported in a format that supports this type of interactive use).

3.8.7 Working with Report Units

Up to now we focused on creating links of type Reference. Now let's create a hyperlink to invoke a JasperReports Server report, specifying which report, and setting values for its input controls. This mechanism requires that the report showing the report is executed in JasperReports server, so it may be considered a way to move from a report unit to another, realizing what is more generally called drill-down and drill-up (for example, I may want to click a bar in a bar chart related to a country to run a report that shows more about this country, or just details the number represented by the bar itself).

The execution of report units can be done by using hyperlink of type ReportExecution. This type requires to define a hyperlink parameter called "_report", which holds the location of the report unit inside the JasperReports server repository.

Other hyperlink parameters can be defined to set values for input controls exposed by the report unit.

The next tutorial shows how to create a link to the report unit 04. Product Results by Store Type Report", located at /public/Samples/Reports/4_Product_Results_by_Store_Type_Report.

When executed, this report can be filtered by country, anyway, since this report unit does not expose through input controls defined at report unit the country parameter, we cannot pass the country value we have.

1. Create a new report, following the steps described at the beginning of this chapter (when we created a static link to http://www.jaspersoft.com)
2. Modify the # of orders measure by adding the following properties in the advanced properties tab:
   a. hyperlinkType (contributor: SeriesItemHyperlink, Static value: ReportExecution)
   b. hyperlinkTarget (contributor: SeriesItemHyperlink, Static value: Blank)
   c. 3_report (contributor: SeriesItemHyperlink, Static value: /public/Samples/Reports/4_Product_Results_by_Store_Type_Report)

3. Publish the report in JasperReports Server and preview it on the web. Click a bar column to open the report units we defined in the previous step.

Note that there is no special syntax to define a parameter, just use as property name the name of the parameter, and select the value (static, bucket or measure based).

The report unit 06. Profit Detail Report (/public/Samples/Reports/ProfitDetailReport) is a good candidate for this test. It exposes the parameter ProductFamily, so by adding the ProductFamily hyperlink property, and by setting it to a value like Drink, can pre-filter its results.

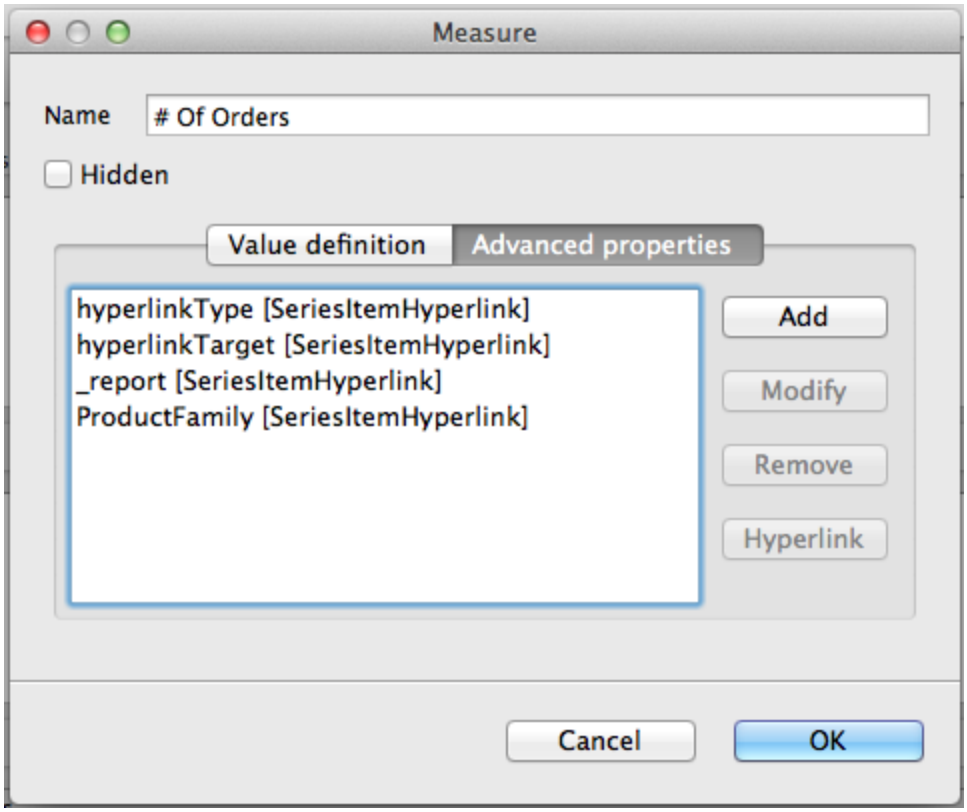


Figure 3-6 Measure Configuration for # of Orders

3.9 Advanced Elements and Custom Components

Besides the built-in elements seen up to now, JasperReports supports two technologies that enable you to plug-in new JasperReport objects respectively called “custom components” and “generic elements.” Both are supported by Jaspersoft Studio. Without a specific plug-in offered by the custom element provider, there is not much you can do with it; you can just set the common element properties. Therefore, a custom element developer should provide a plug-in for Jaspersoft Studio through which you can, at least, add the element to a report (maybe adding a palette item) and modify the element properties (implementing what is required to display the additional properties in the Properties view when the element is selected.

3.10 Hyperlinks in HTML5 Charts

This section describes functionality that can be restricted by the software license for Jaspersoft Studio. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

Hyperlinks in HTML 5 charts are created by a different process than that described in **3.8, "Anchors, Bookmarks, and Hyperlinks," on page 45**.

To follow a link in an HTML 5 chart, the user clicks a chart component that represents a measure. This hyperlink is created by a specific extension to that chart component, which evaluates properties of the measure

3.10.1 Creating Hyperlinks in HTML5 Charts

1. Create an empty report using the Sample DB data adapter and the query:
   `SELECT * FROM ORDER`
2. Add an HTML5 Bar chart to the title band.
3. Double click the chart to access the chart configuration, click the Chart Data tab and select the Configuration sub-tab to setup the multi-dimensional dataset.
4. Modify the Level 1 category by renaming it Country and setting the Expression to
   `$F{SHIPCOUNTRY}`
5. Click **OK**

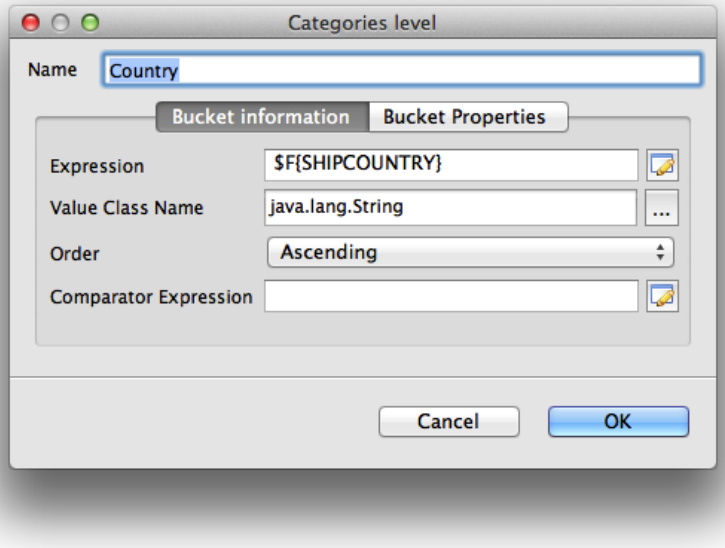


Figure 3-7 Editing Categories in the Chart Data Configuration Window

6. On the Configuration tab of the Chart Properties dialog, select the default value in **Measures** and click **Modify**.

7. Change the measure to count the number of orders as shown below:

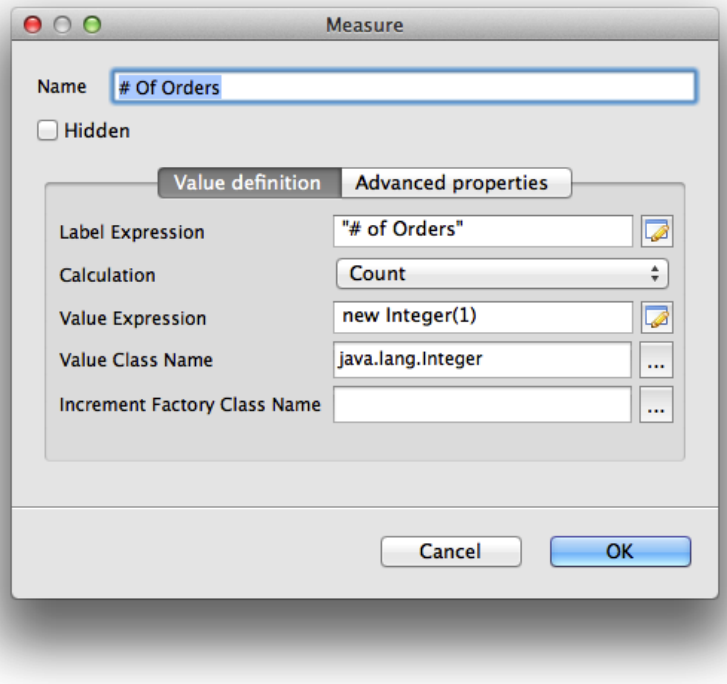


Figure 3-8 Editing the Measure to Count Orders

8. Click **OK** twice.
   At this point the chart is configured, run the report to test it.

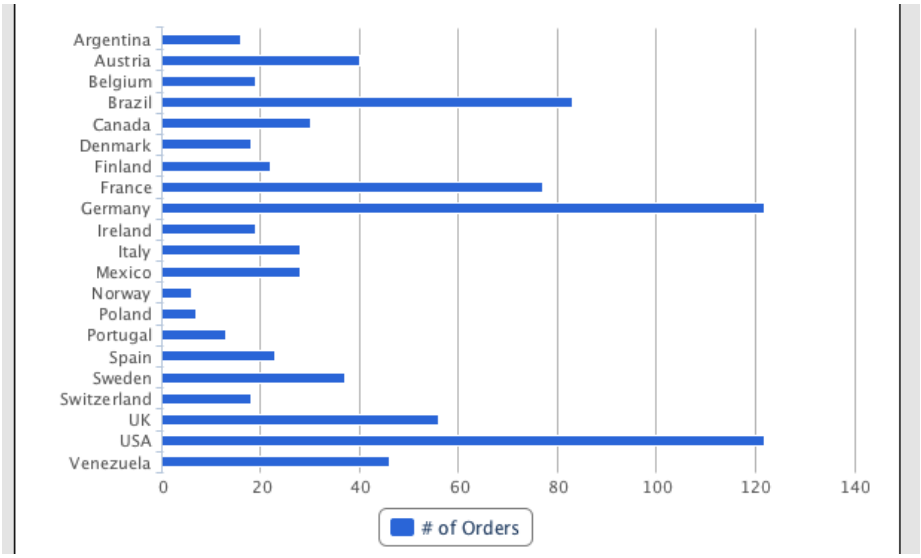


Figure 3-9 Successful Test of the Example Report

Now, let's configure the hyperlink.

9. In design mode, double click the chart and again click the Chart Data tab and select the Configuration sub-tab.
10. Select the measure # of orders for editing.

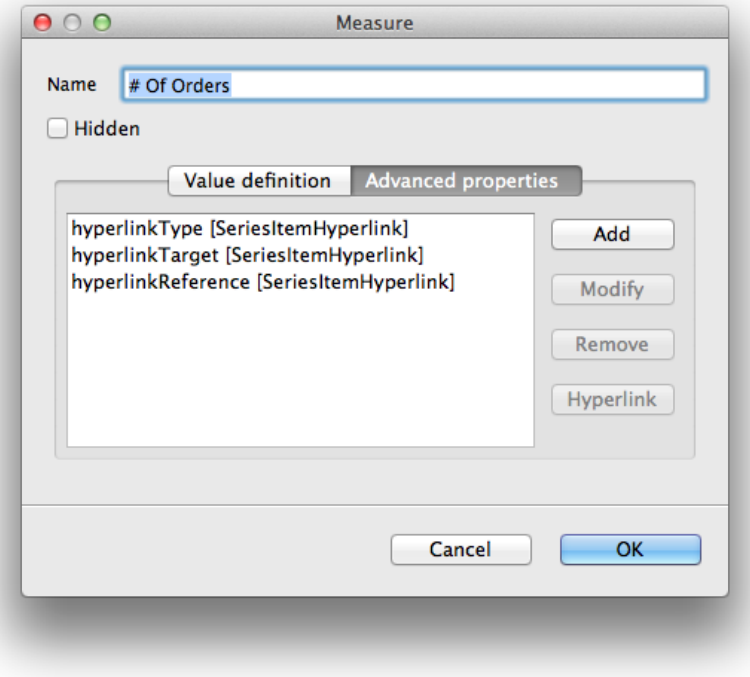


Figure 3-10 Editing the Number of Orders a Second Time

11. Click Advanced properties, then the Hyperlink button.

    Jaspersoft Studio creates the hyperlink's basic property configuration: `hyperlinkType`, `hyperlinkTarget`, and `hyperlinkReference`.
12. Double-click `hyperlinkReference`. You'll see the Edit property dialog, which defaults to a constant: `http://www.jaspersoft.com`

Click **OK**.

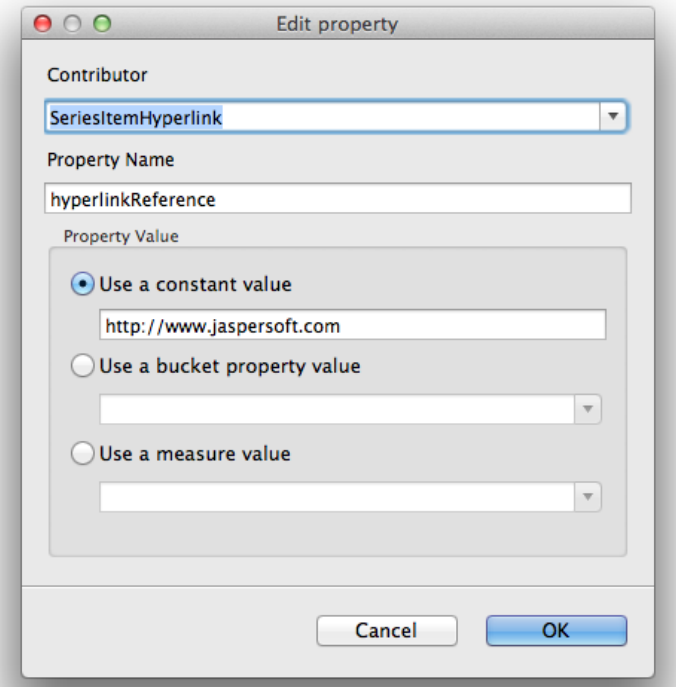


Figure 3-11 Editing the Property Definition for hyperlinkReference

13. Preview your chart using the interactive report viewer to see that the columns are actually clickable and point to http://www.jaspersoft.com

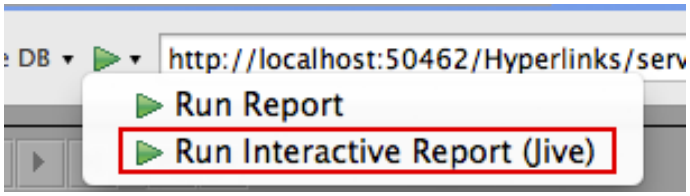


Figure 3-12 Previewing the Report

Property values can come from a static value, a bucket property value (explained in **"Working with Bucket Properties and Hidden Measures" on page 55**) or from the value of a measure (which lets us create values for our hyperlink, like the URL, combined with using hidden measures, to create an interesting user experience).

3.10.2 Working with Bucket Properties and Hidden Measures

Bucket properties are user defined key/value pairs associated with a category or series level. The user can name the property and define the value using an expression.

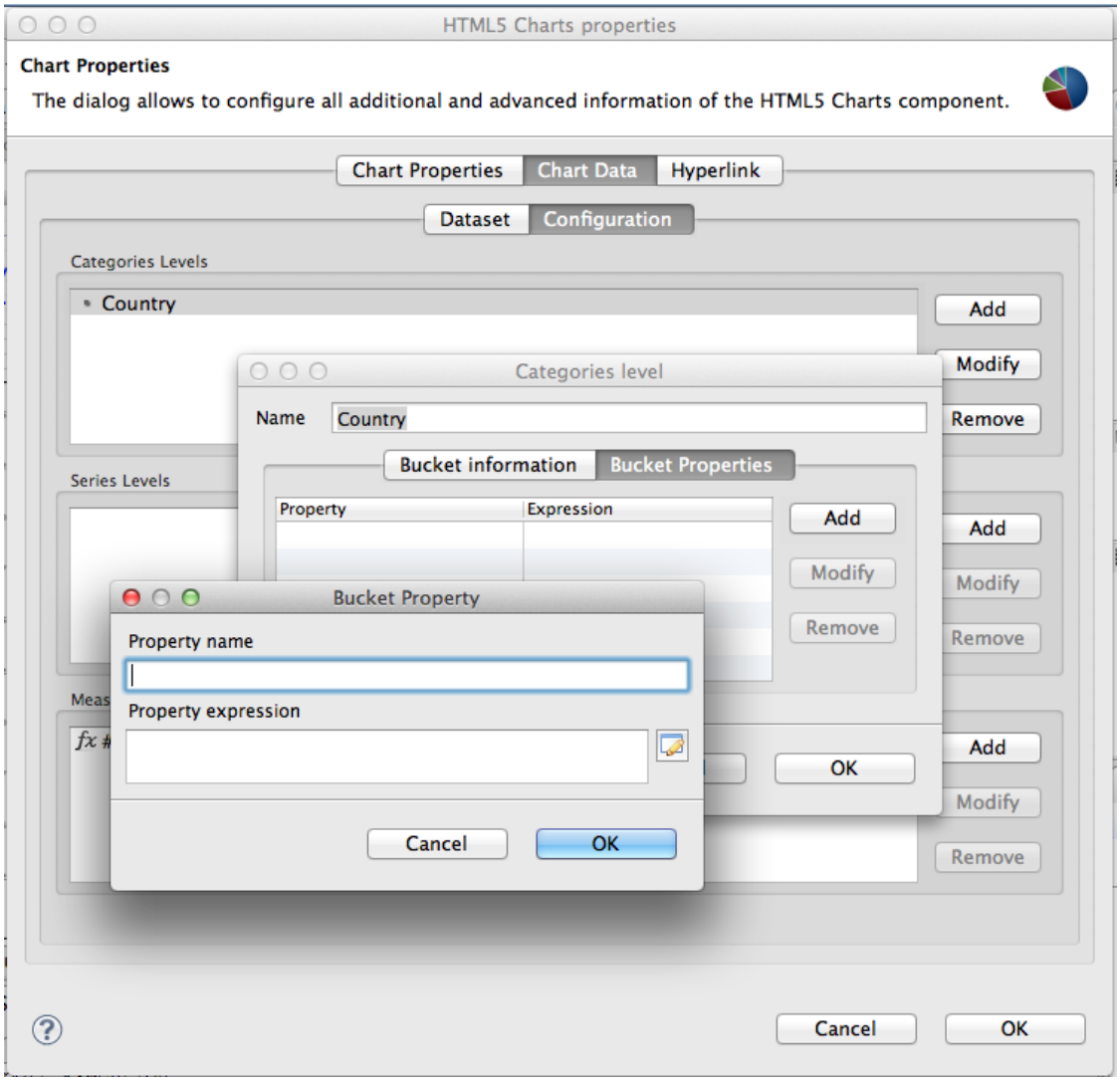


Figure 3-13 Defining Bucket Properties for HTML5 Chart Hyplerlinking

Let's define a new property, called country_url, storing a link specific to the country (something like http://en.wikipedia.org/wiki/Italy and use this link in our chart measure hyperlink definition so that, when a user clicks on a column, the link of the country that the column references is opened.

1. Create a new bucket property for the Country category level and use the following settings:
   - Property name: country_link
   - Expression: "http://en.wikipedia.org/wiki/" + SF{SHIPCOUNTRY}
2. Open the configuration dialog of your measure, move to Advanced Properties, and double click the `HyperlinkReference` property.
3. Set the **Use a bucket property value** to the newly created property: country_link.

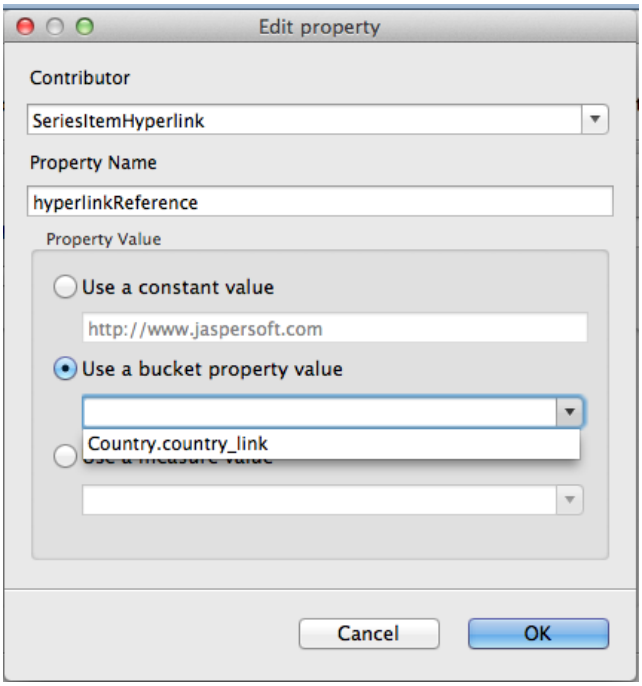


Figure 3-14 Configuring Measure Properties to set the HyperlinkReference Value

4. Click **OK**.
5. Save the report and preview. Now, when you click a column, you should link to a Wikipedia page related to our country.

Using bucket properties, you can create links connected to a dimension. In this case it was the Country dimension (the Country category), but the same approach can be used to create a link related to series.

To make this example a little bit more useful, and to see how we can really use hyperlink information form different dimensions, let's add an extra dimension called year, so the chart can display the number of orders placed in a specific country in a specific year. Then we can create a hyperlink that contains both the country and the year.

1. Add the new dimension by creating a new series level configured as shown below:
   - Name: Year
   - Expression: YEAR($F{ORDERDATE})

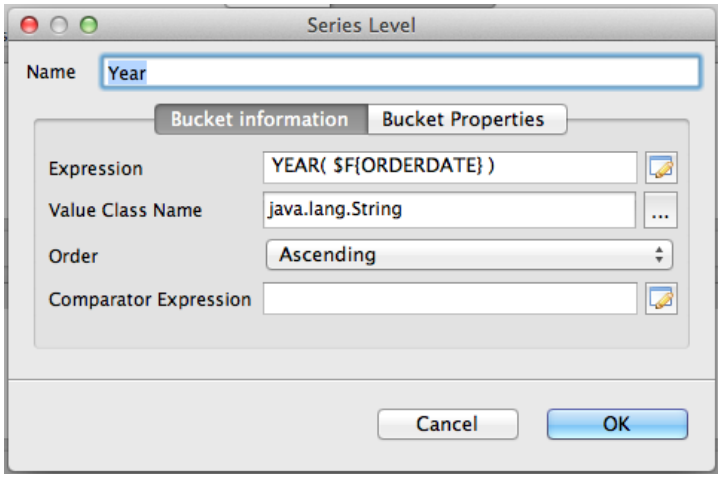


Figure 3-15 Editing Series Level to Define Buckets

Now,for each country, the chart shows the number of orders split by year.

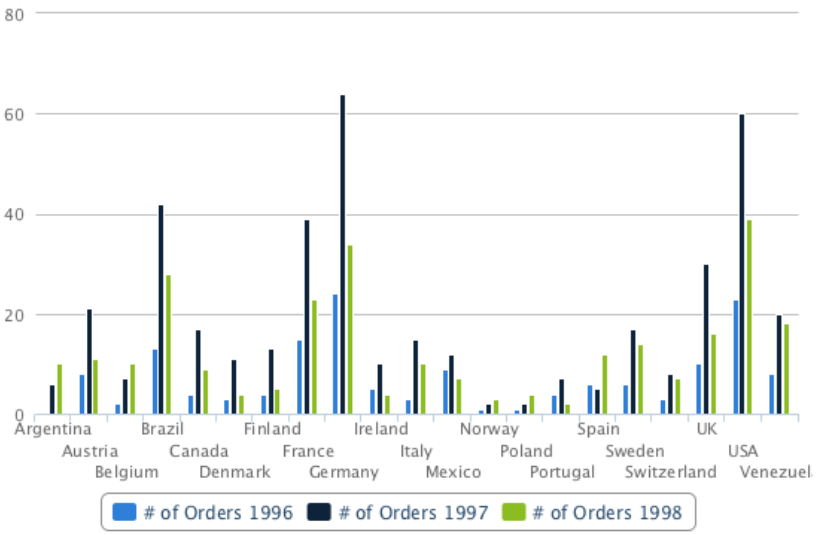


Figure 3-16 Bar Chart with Order Numbers Split by Year

Let's create a new measure to build our URL string. Measures are usually designed to hold numeric values, most of the time the result of some aggregation function (in our example we show the count of orders). But a measure can actually be anything; in our case it's a string built using an expression like the following:

- **Name**: URL Measure
- **Hidden**: true (it is very important to check the checkbox, because we are not going to display this measure, it would not make sense as it's not numeric)
- **Calculation**: Nothing
- **Value Expression**: "http://en.wikipedia.org/wiki/" + $F{SHIPCOUNTRY} + "/" + YEAR($F {ORDERDATE})
- **Value Class Name**: java.lang.String

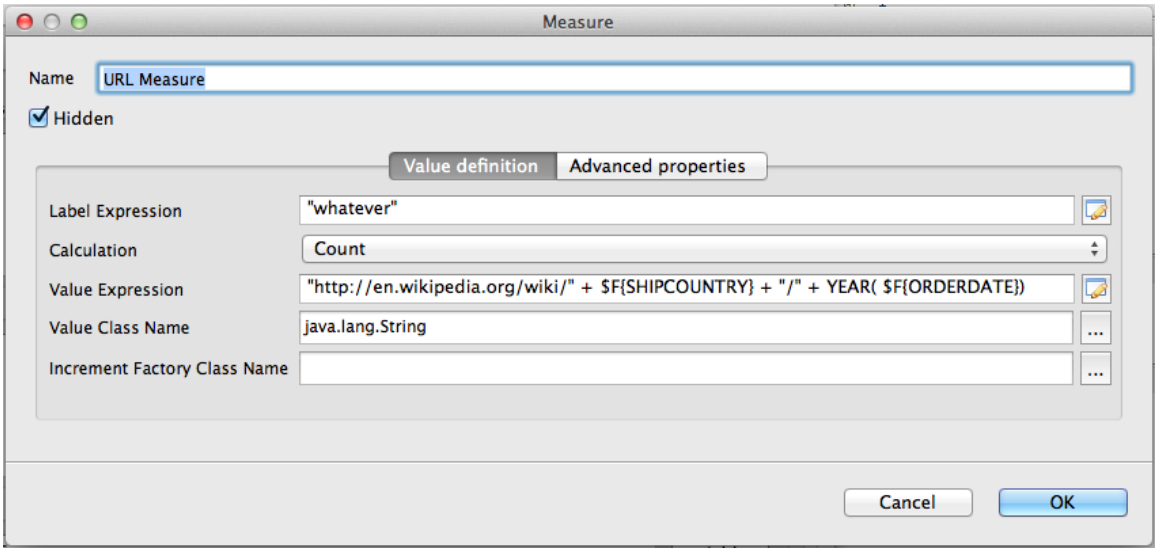


Figure 3-17 Defining the URL to Open when the Measure is Clicked

The most important things to note are:

- the measure is marked as hidden
- the calculation type is set to Nothing
- the class of the measure is String (accordingly to what the expression produces)

The expression produces a String that represent a URL with a reference to both the country and the year of the order.

Now, let's use this measure.

2. Open the # of orders measure to edit the `hyperlinkReference` and set the measure we just created as the value.

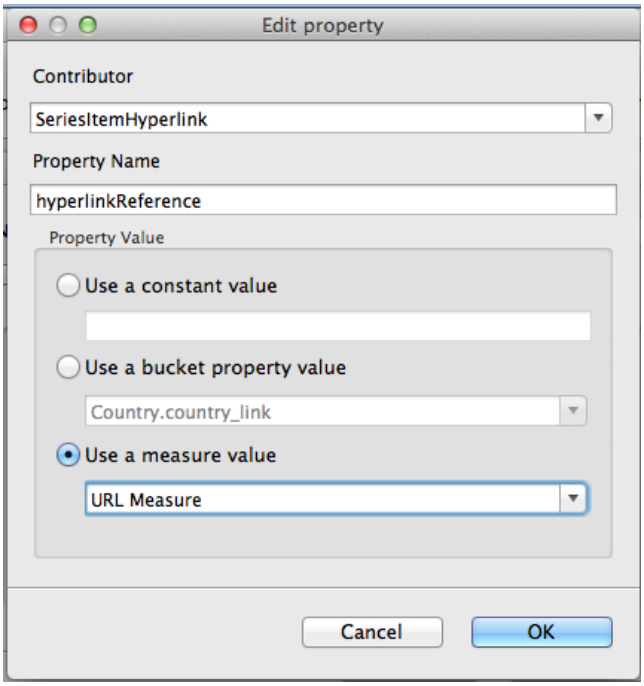


Figure 3-18 Editing the hyperlinkReference to use a Measure Value

3. Save all and preview.

Now, each column, slice, bar, and column section in the stacked column chart points to the country/year the object is representing.

3.11 Subreports

The subreport element includes a subreport inside a report. The subreport is represented by an external Jasper file. It can use the same database connection as the parent report or a data source specified in the subreport properties. Subreports are useful for charts that don't need all the data in the main report.

The following briefly describes the characteristics of subreports:

`Subreport Expression`	This expression returns a subreport expression class object at run time. According to the return type, the expression is evaluated in order to recover a Jasper object to be used to produce the subreport. In case the expression class is set to `java.lang.String`, JasperReports looks for a file following the same approach explained for the `Image Expression` of the Image element.
`Subreport Expression Class`	This is the class type of the expression; there are several options, each for a different way to load the JasperReport object used for the subreport.

`Using cache`	This specifies whether the subreport's report object is kept in memory or reloaded each time it's used. It is common for a subreport element to be printed more than once (or once for each record in the main dataset). The cache works only if the subreport expression type is `String`, because that string is used as key for the cache.
`Connection/Datasource Expression`	This identifies the expression returned at run time when a JDBC connection or a JRDataSource used to fill in the subreport. Alternatively you can choose not to pass any data. This option is useful at times. However, a subreport, like any report, returns an empty document if data are not provided. So in this case, the subreport should have the document property `When No Data Type` set to something like `All Sections, No Detail`.
`Parameters Map Expression`	This optional expression can be used to produce a `java.util.Map`object at run time. The map must contain a set of coupled names/objects that are passed to the subreport to set a value for its parameters.
`Subreport parameters`	This table allows you to define some coupled names/expressions that are useful to dynamically set a value for the subreport parameters using calculated expressions.
`Subreport return values`	This table allows you to define how to store values in local variables calculated or processed in the subreport (such as totals and record count).

To create a subreport:

1. Drag the **Subreport** element from the **Palette** to the area of your report where you want to use it.
   The **Subreport** wizard provides three options:
   - **Create a new report**: Use this option when you need to use data or a query not available in an existing report.
   - **Select an existing report**: Use this option when you want to choose a report from the repository.
   - **Just create the subreport element**: Use this option to create a placeholder to be used later.

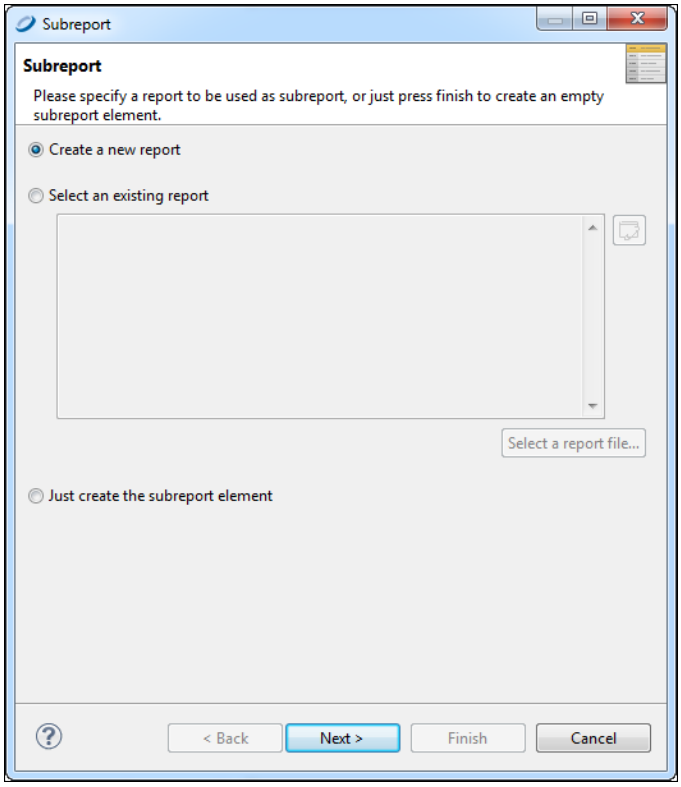


Figure 3-19 Subreport Wizard

2. Select **Create a new report** and click **Next**. The **New Report Wizard > Report Templates** window appears.
3. Select a template for your subreport. For this example, select one of the blank templates. Click **Next**. The **New Report Wizard > Report file** window opens.
4. Select a location for your subreport, and name it. Click **Next**. The **Data Source** window opens.

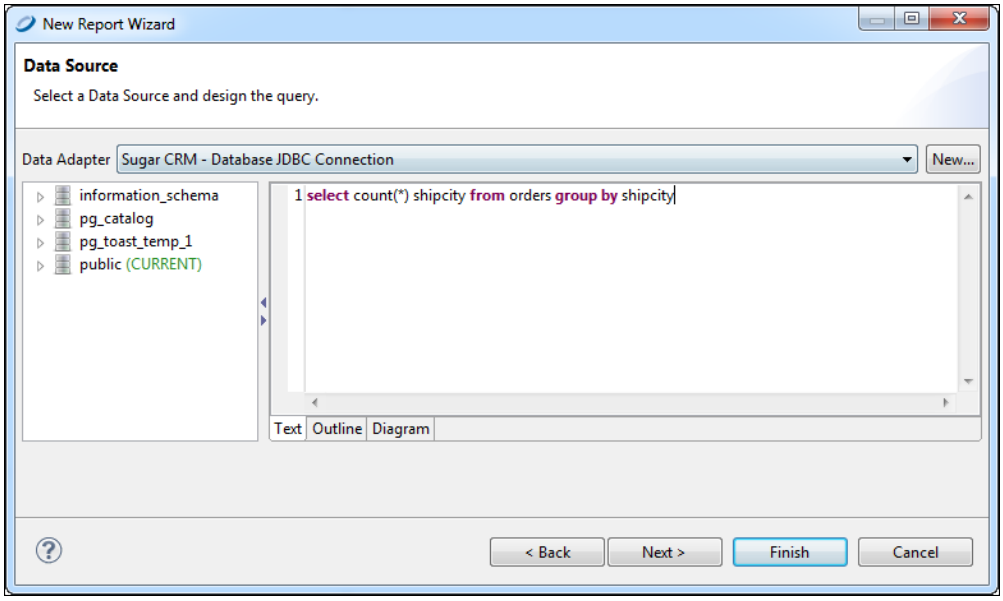


Figure 3-20 Data Source and Query

5. Choose to use the same data adapter as the main report, or a different data adapter.
   For this example, choose the same adapter (Sugar CRM). Enter the following SQL query:
   `select count (*), shipcity from orders group by shipcity`
   Add all the fields to the list on the right. Click **Next**.
6. Click **Next** to skip the **Group By** step. The **Subreport > Connection** window opens.

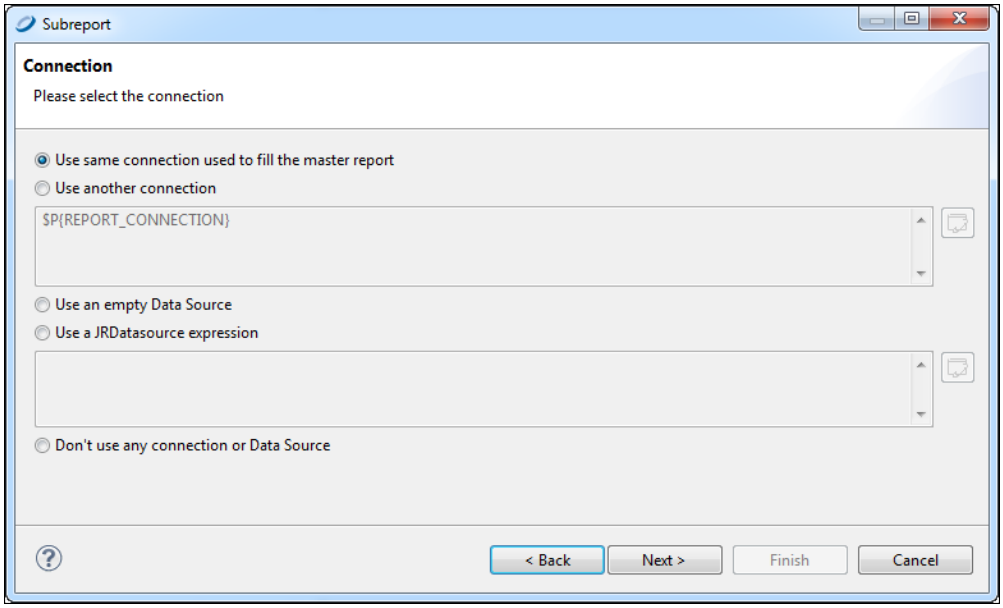


Figure 3-21 Subreport > Connection Window

7. Choose to connect either to the same database as the main report or to a different database. For this example, click **Use same connection used to fill the master report**.
8. Click **Next**. The **Subreport Parameters** window opens.

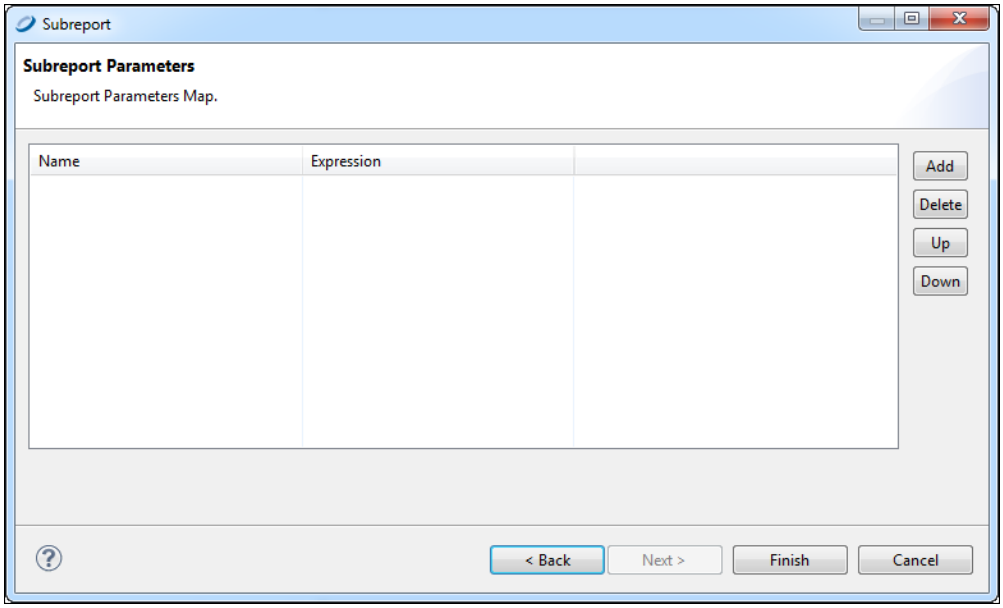


Figure 3-22 Subreport Parameters Window

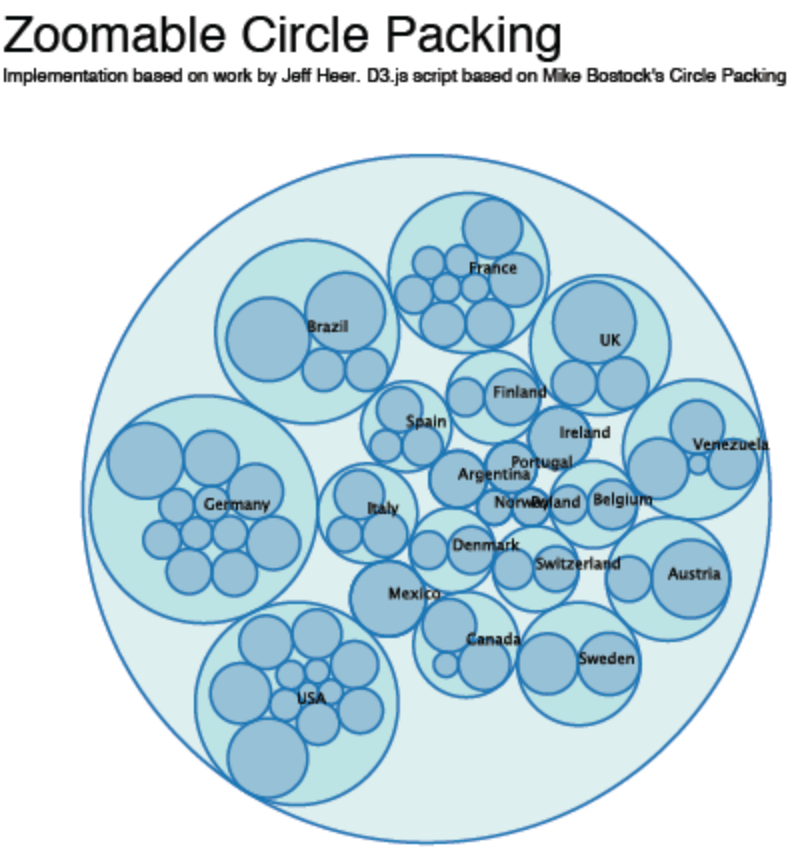
9. For this example, skip this window and click **Finish**. A new report opens containing all bands.
10. Delete all bands but the Title or Summary band to eliminate extra white space in your report.

    You now have a location into which to place your table, chart, or other element attached to the new subreport.

3.12 Custom Visualization Component

The custom visualization component lets you leverage JavaScript in Jaspersoft Studio and JasperReports Server. You can create JavaScript modules and use them in your reports to extend their functionality. The main purpose of the custom visualization component is to render SVG (Scalable Vector Graphics) images. Your modules can leverage third-party JavaScript libraries, such as JQuery. For example, perhaps you want to create reports with dynamic behaviors for interaction and animation; you could leverage D3 (d3js.org) to bind data to the Document Object Model (DOM) and manipulate the SVG. Such a report is shown in **, "D3-enabled report," on page 65**.

Please note that this component is only supported by the Jaspersoft Community (community.jaspersoft.com). For this release, TIBCO Jaspersoft Technical Support and Engineering do not support it.



D3-enabled report

The custom visualization component is a powerful and flexible feature, suitable for advanced users of JasperReports Library. Using the component requires advanced coding skills in the following technologies:

- JavaScript
- CSS
- HTML/DHTML
- Optionally, any third-party library you want to expose in Jaspersoft Studio and JasperReports Server.

Before creating reports that use your third-party library, you must configure various applications to work together; you must:

1. Install PhantomJS (phantomjs.org), which renders your JavaScript module's visual component.
2. Configure Jaspersoft Studio and JasperReports Server to use PhantomJS.
3. Create a JavaScript module that renders an SVG. This main module, as well as any JavaScript it relies on, are optimized ans combines into a single JavaScript file (using a RequireJS build file (build.js). Jaspersoft Studio simplifies the optimization process to generate the JavaScript implementation of the component. Place your JavaScript files somewhere that Jaspersoft Studio can find it, such as attaching it to the report unit or placing it on the classpath.

Next, create and deploy reports that rely on your custom visualization component to render SVG images. Drag the Custom Visualization component from the Elements palette into the Design pane, and create a report-level property of type `com.jaspersoft.jasperreports.components.customvisualization.require.js`; it

must specify the main JavaScript file for your custom visualization. The custom visualization component appears as a rectangular area of your report. A single custom visualization component can be used by any number of reports that require the same JavaScript functionality. When exported to HTML format, these reports can be interactive (assuming that your module attaches events to the DOM).

This component can help you leverage any number of JavaScript libraries, such as:

- D3.js
- Raphäel
- HighCharts
- JQuery

For more in-depth information, please see the articles on our Community wiki that describe the custom visualization component.

CHAPTER 4 CREATING A SIMPLE REPORT

JasperReports Library is a powerful tool, and Jaspersoft Studio exposes much of its functionality to help you design reports. This chapter introduces the basic steps for defining a report and includes the following sections:

- **Creating a New Report**
- **Adding and Deleting Report Elements**
- **Previewing a Report**
- **Creating a Project Folder**

4.1 Creating a New Report

You can add your own templates, or use one of the supplied templates to create a report from scratch. You have three data source options:

- **One Empty Record - Empty rows**: This is default data adapter for creating a report without data. You might use this option to define the layout of a report and connect it to a data source later.
- **Sample DB - Database JBDC Connection**: This is an SQL database provided with the Jaspersoft Studio installation.
- **Create a new data adapter**: Jaspersoft Studio supports a wide variety of data sources.

To create a new report:

1. Go to **File > New > Jasper Report**. The **New Report Wizard** window displays the **Report Templates** page.

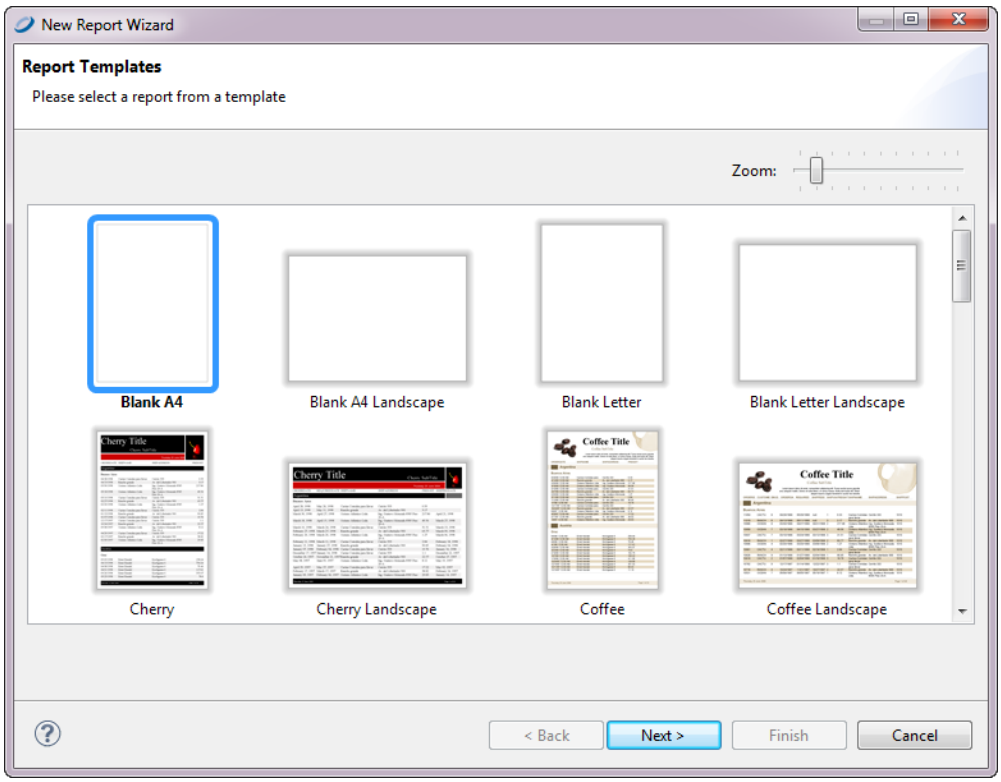


Figure 4-1 New Report Wizard

2. Select the Coffee template and click **Next**. The **New Report Wizard** shows the **Report file** page.

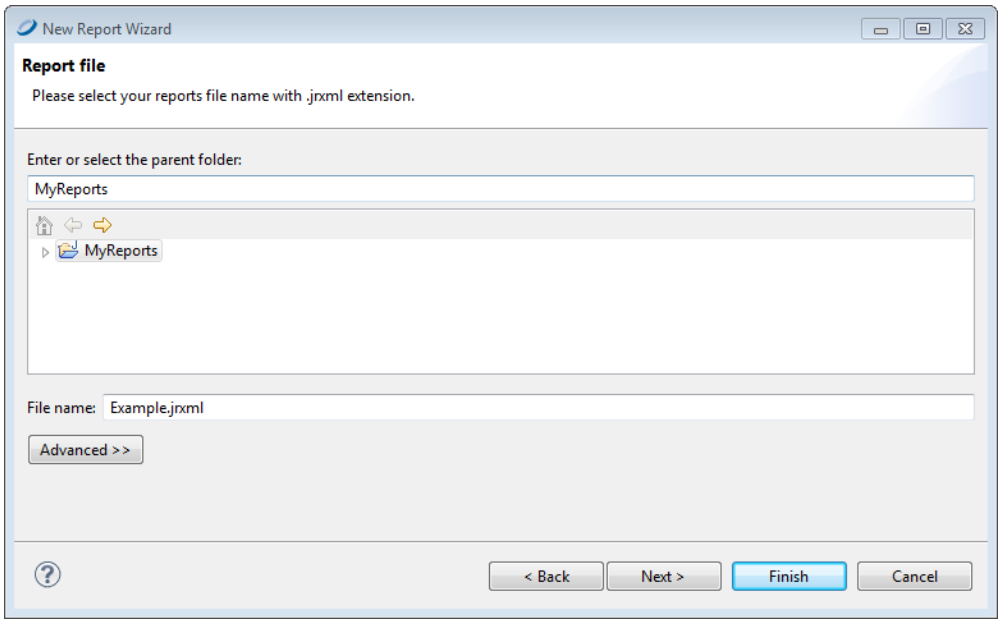


Figure 4-2 New Report Wizard > Report file

3. Navigate to the folder you want the report in and name the report. To create a new folder, see **"Creating a Project Folder" on page 75**.
4. Click **Next**. The **Datasource** window appears.
5. Choose **Sample DB - Database JDBC Connection**. Enter the query `select * from orders`

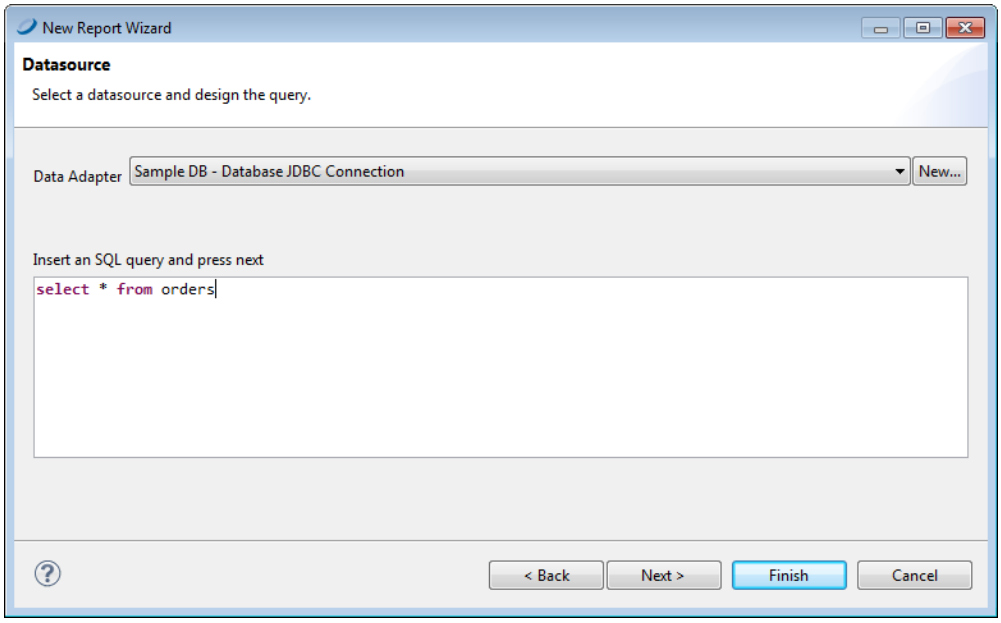


Figure 4-3 New Report Wizard > Datasource

6. Click **Next**. The **Fields** window appears. The Dataset list shows all the discovered fields.

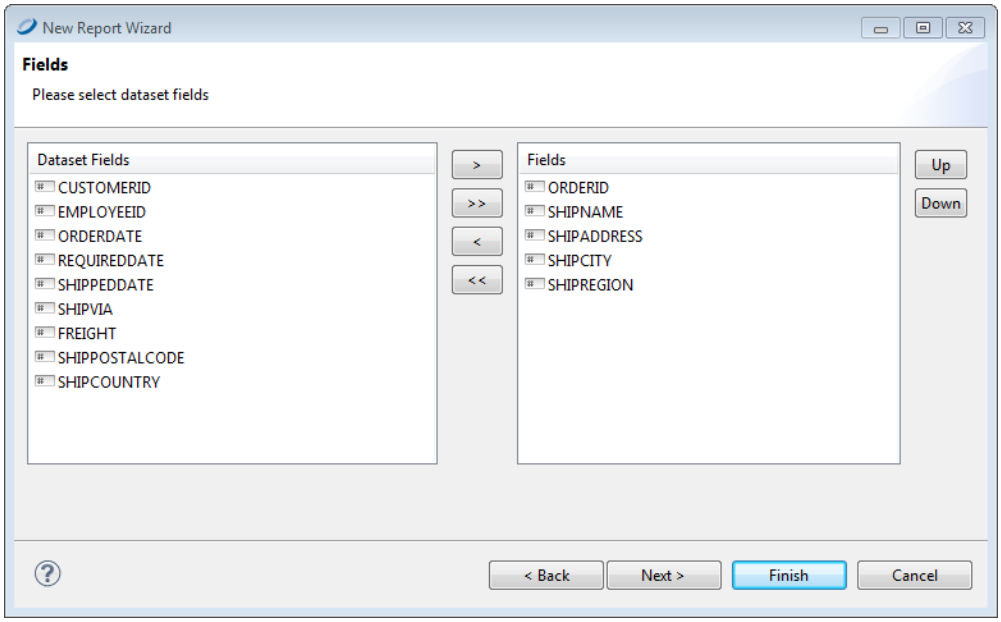


Figure 4-4 New Report Wizard > Fields

7. Select the following fields and click the right arrow to add them to your report.
   - ORDERID
   - SHIPNAME
   - SHIPADDRESS
   - SHIPCITY
   - SHIPREGION
8. Click **Next**. The **Grouping** window appears.
9. Click **Next** and **Finish**.

   Jaspersoft Studio now builds the report layout with the selected fields included as shown.

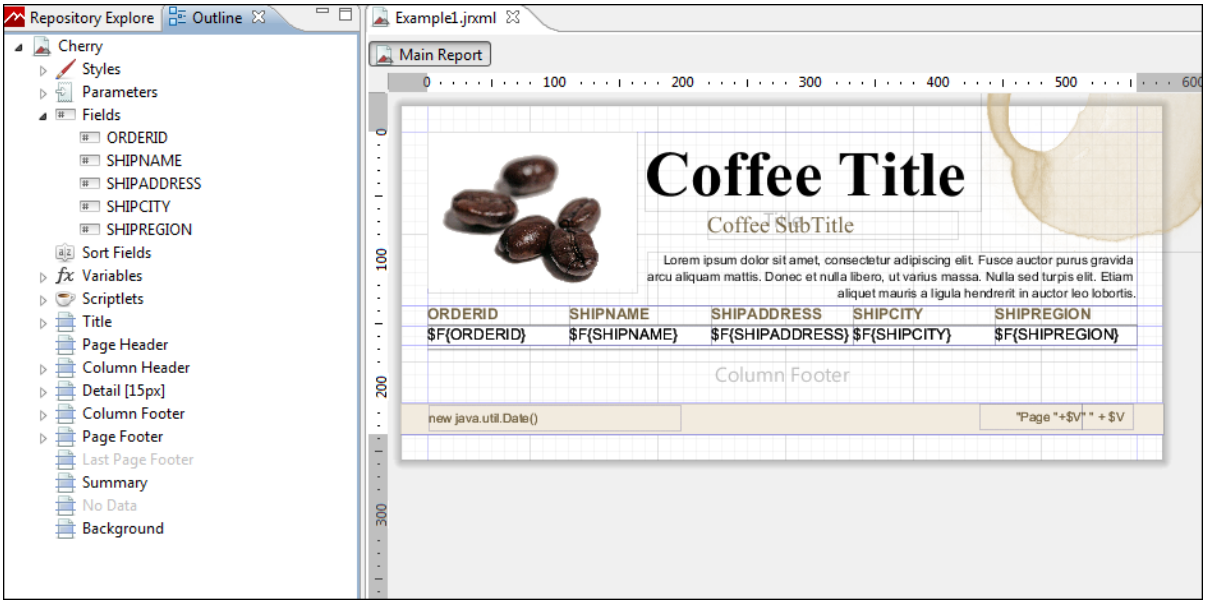


Figure 4-5 New Report in Design View

4.2 Adding and Deleting Report Elements

You can add and delete fields and other elements to your report.

4.2.1 Adding Fields to a Report

To add fields to an already created report:

1. Select the main node of the report from the **Outline** view.
2. Select the **Properties** tab.

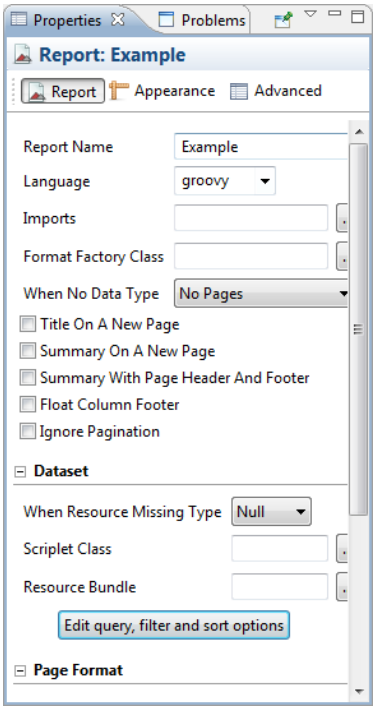


Figure 4-6 Properties Tab

3. On the **Properties** tab click the **Report** button.
4. Click the **Edit query, filter and sort option** button. The **Dataset and Query Dialog** opens.

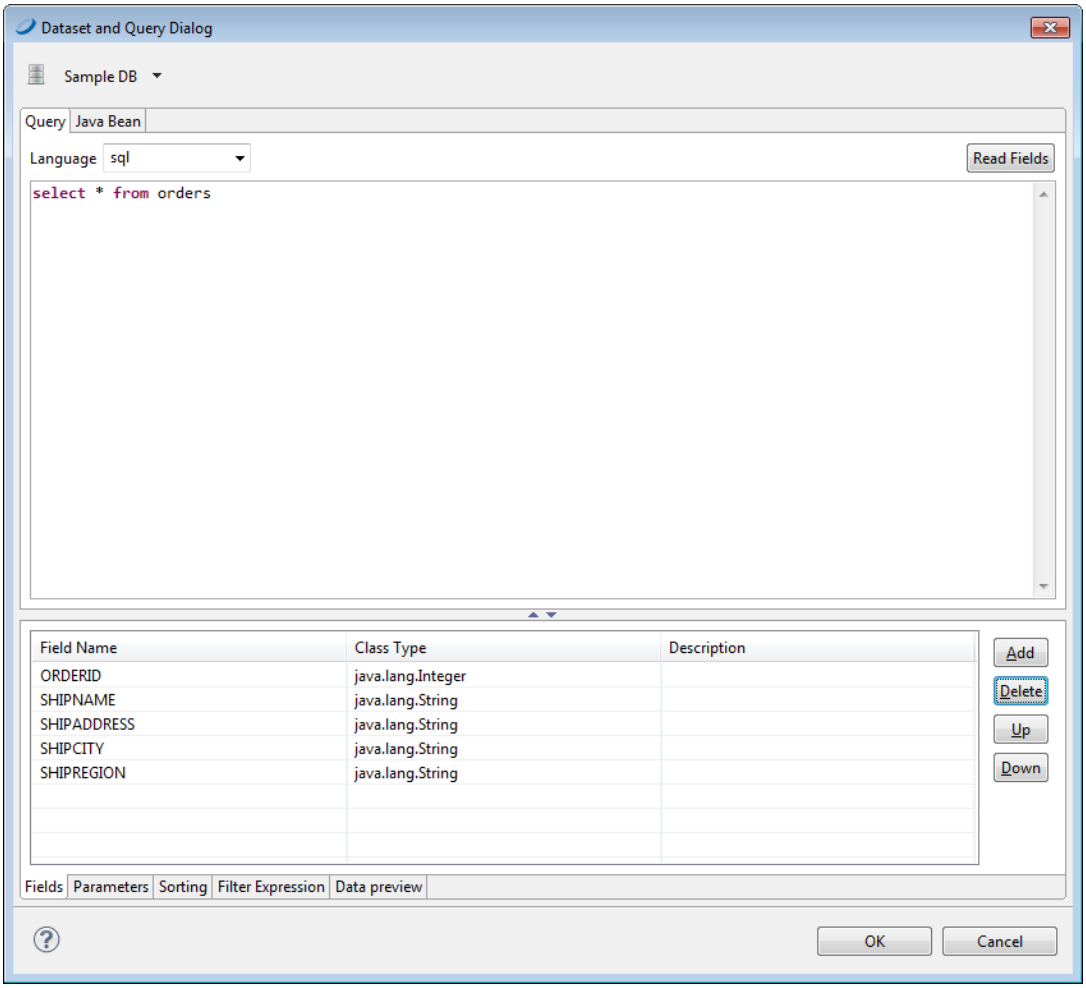


Figure 4-7 Dataset and Query Dialog

5. Add more fields by clicking the **Read Fields** button. All the fields discovered are added as new fields in the report.

You can also change your query in the same dialog. If a new query discovers fewer fields than used in the existing report, the fields not included the new query are removed from your report.

6. Click **OK** to return to the **Design** view.
7. Expand **Fields** in the **Outline** view to see all the fields now available for your report.

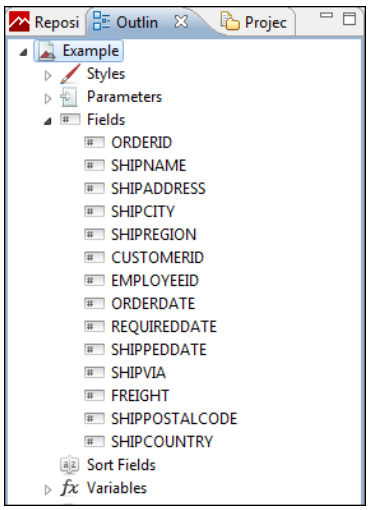


Figure 4-8 Fields

8. To add a field to your report, click the field and drag it into the **Design**.
   When the field object is dragged inside the detail band, Jaspersoft Studio creates a text field element and sets the text field expression for that element.

4.2.2 Deleting Fields

To delete a field from a report, right click the field in the **Design** and select **Delete**.

4.2.3 Adding Other Elements

To add other elements, such as lines, images or charts, drag the element from the Palette into the **Design**.

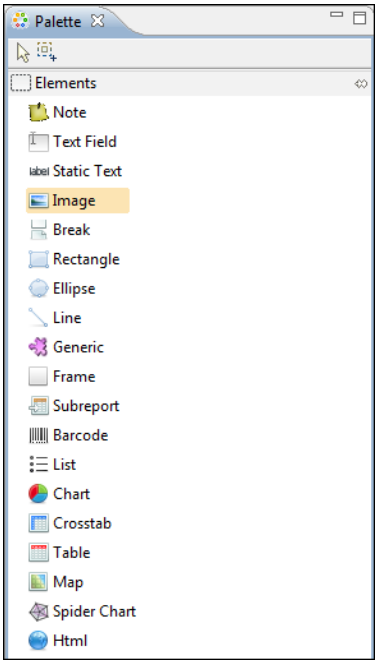
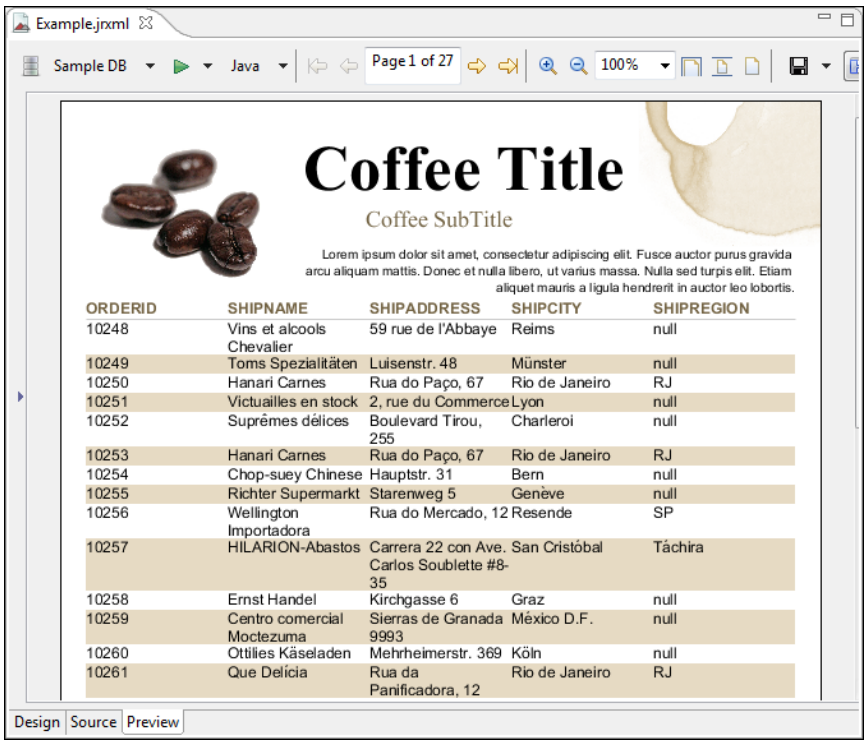


Figure 4-9 Elements Palette

4.3 Previewing a Report

Click the **Preview** tab at the bottom of the **Design** view. The preview compiles the report in the background with data retrieved by the query through your JDBC connection.The Detail band repeats for every row in the query results, creating a simple table report:



ORDERID	SHIPNAME	SHIPADDRESS	SHIPCITY	SHIPREGION
10248	Vins et alcools Chevalier	59 rue de l'Abbaye	Reims	null
10249	Toms Spezialitäten	Luisenstr. 48	Münster	null
10250	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	RJ
10251	Victuailles en stock	2, rue du Commerce	Lyon	null
10252	Suprêmes délices	Boulevard Tirou, 255	Charleroi	null
10253	Hanari Carnes	Rua do Paço, 67	Rio de Janeiro	RJ
10254	Chop-suey Chinese	Hauptstr. 31	Bern	null
10255	Richter Supermarkt	Starenweg 5	Genève	null
10256	Wellington Importadora	Rua do Mercado, 12	Resende	SP
10257	HILARION-Abastos	Carrera 22 con Ave. Carlos Soublette #8-35	San Cristóbal	Táchira
10258	Ernst Handel	Kirchgasse 6	Graz	null
10259	Centro comercial Moctezuma	Sierras de Granada 9993	México D.F.	null
10260	Ottilies Käseladen	Mehrheimerstr. 369	Köln	null
10261	Que Delícia	Rua da Panificadora, 12	Rio de Janeiro	RJ

Figure 4-10 Report Preview

4.4 Creating a Project Folder

Project folders help you organize your reports.

To create a project folder:

1. Choose **File > New > Project**. The **Select a wizard** dialog appears.

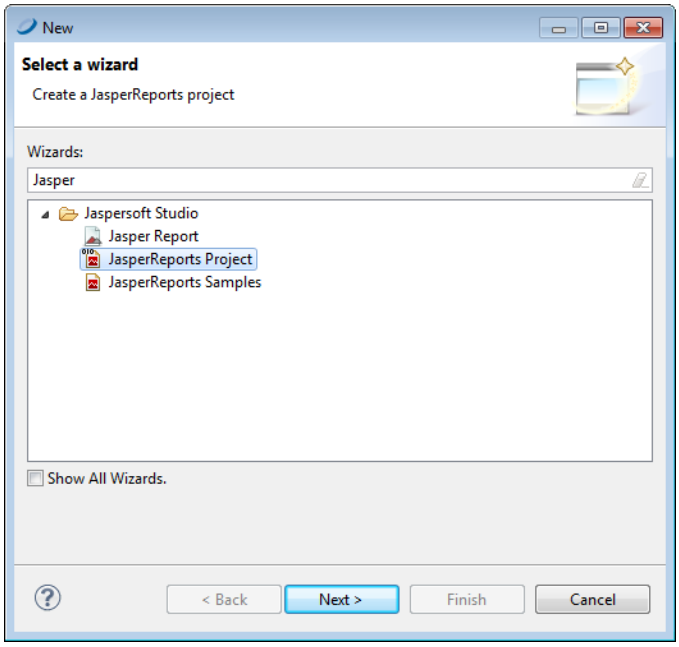


Figure 4-11 Select a Wizard

2. Enter **Jasper** in the Wizards bar to filter actions to those related to Jaspersoft Studio
3. Select **JasperReports Project**. Click **Next**. The **New JasperReports Project** wizard appears.
4. Enter a name for your project and click **Finish**. The **Project Explorer** displays your project.

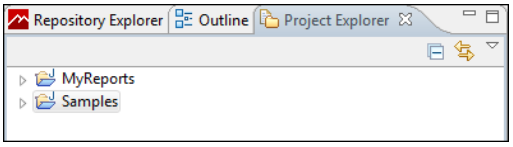


Figure 4-12

CHAPTER 5 ACCESSING JASPERREPORTS SERVER FROM JASPERSOFT STUDIO

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

You can connect Jaspersoft Studio to JasperReports Serverand move reports between the two. Jaspersoft Studio uses web services to interact with the server.

Connecting with JasperReports Server enables you to:

- Browse the repository on the server from Jaspersoft Studio.
- Add reports and subreports to the repository from Jaspersoft Studio.
- Drag and drop images and other resources from the repository to Jaspersoft Studio.
- Add and delete folders and resources on the server from Jaspersoft Studio.
- Modify resource properties on the server from Jaspersoft Studio.
- Link input controls to reports on the server.
- Import and export data sources (JDBC, JNDI, and JavaBean).
- Download, edit, and upload JRXML files.
- Connect to multiple servers for access to both test and production environments.
- Create a report in Jaspersoft Studio based on a Domain in JasperReports Server (commercial edition only).

This chapter contains the following sections:

5.1 Connecting to JasperReports Server

To connect Jaspersoft Studio to the server:

1. Start Jaspersoft Studio.
2. Open the **Repository Explorer**, then click the **Create a JasperReports Server Connection** icon  .

The Server profile wizard appears.

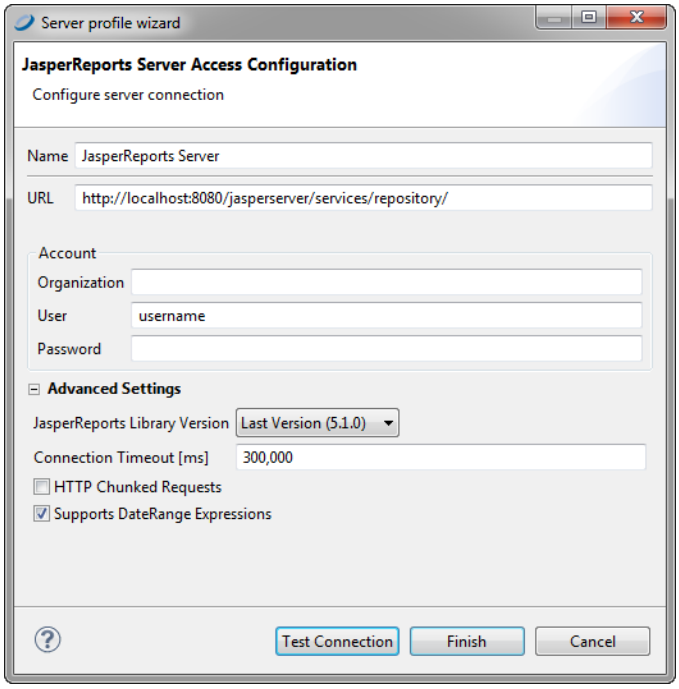


Figure 5-1 Server Profile Wizard

3. Enter the URL, user names and password for your server. If the server hosts multiple organization, enter the name of your organization as well.

   The defaults are:

   - URL:
     - Commercial editions: `http://localhost:8080/jasperserver-pro/`
     - Community edition: `http://localhost:8080/jasperserver/`
   - Organization: There is no default value for this field. If the server hosts multiple organizations, enter the ID of the organization to which you belong.
   - User name: `jasperadmin`
   - Password: `jasperadmin`

   Note that if you are upgrading from a previous version of Jaspersoft Studio, the old URL (`http://localhost:8080/jasperserver-pro/services/repository`) still works.
4. Click **Test Connection**.
5. If the test fails, check your URL, organization, user name, and password.

   Connection problems can sometimes be caused by Eclipse's secure storage feature, which improves your security by storing passwords in an encrypted format. For more information, refer to our Eclipse Secure Storage in Jaspersoft Studio Community wiki page.
6. If the test is successful, click **Finish**.

   The server appears in the **Repository Explorer**.

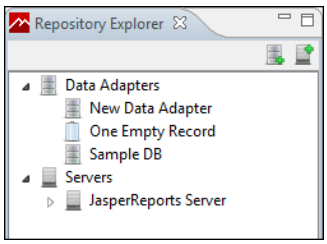


Figure 5-2 Repository Explorer

5.2 Publishing a Report to JasperReports Server

Exporting a report is quite simple, but exporting its data source is more complicated. You have three options for exporting the data source:

- Data source from repository: the server uses a data source already there it to fill the report, so we only need to browse the server for our data source.
- Local data source
- Don't use a data source: the report is exported without a data source connection

To publish a report to the server:

1. Open a report.
2. Click the **Publish Report** button  in the upper-right corner of the Designer. The **Report Publishing Wizard** opens.

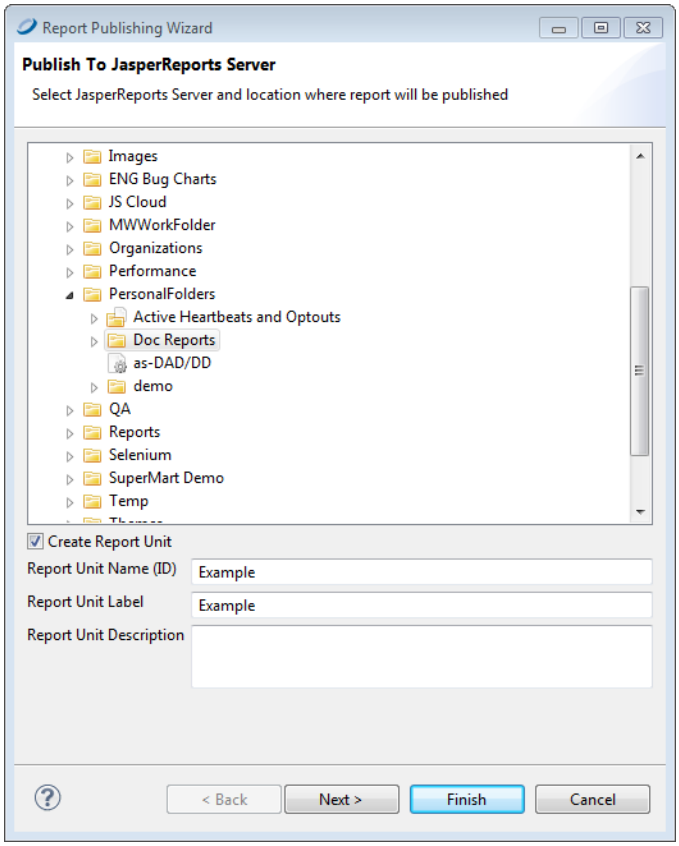


Figure 5-3 Report Publishing Wizard

3. Locate the directory for storing your report.
4. Name the report unit. The report unit contains all report files.
5. Click **Next**. The **Select Resources** window opens.

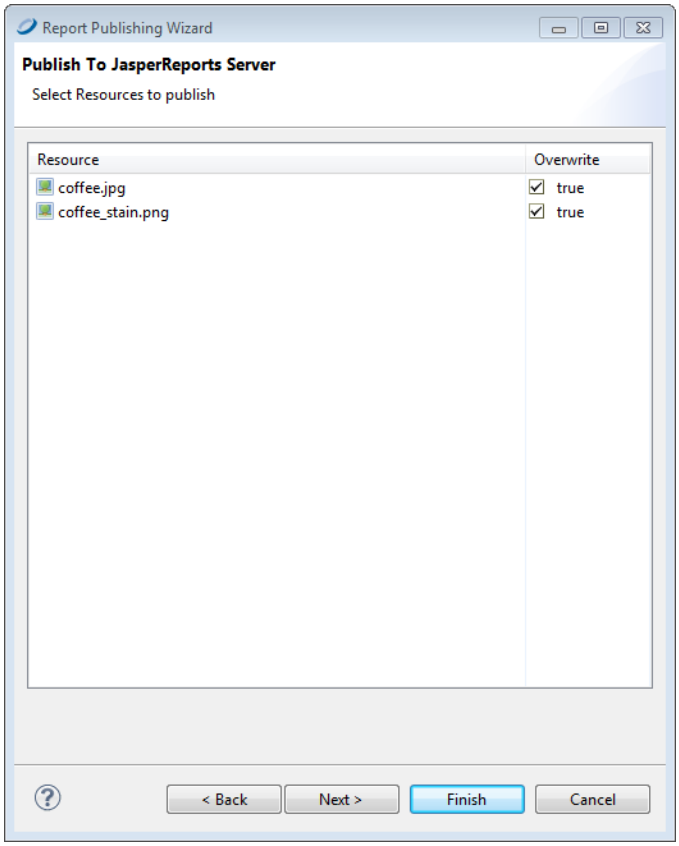


Figure 5-4 Select Resources

6. Select any resources you want to upload with your report and check the box if you want to overwrite previous versions of those resources. Click **Next**. The **Configure the data source** window opens.

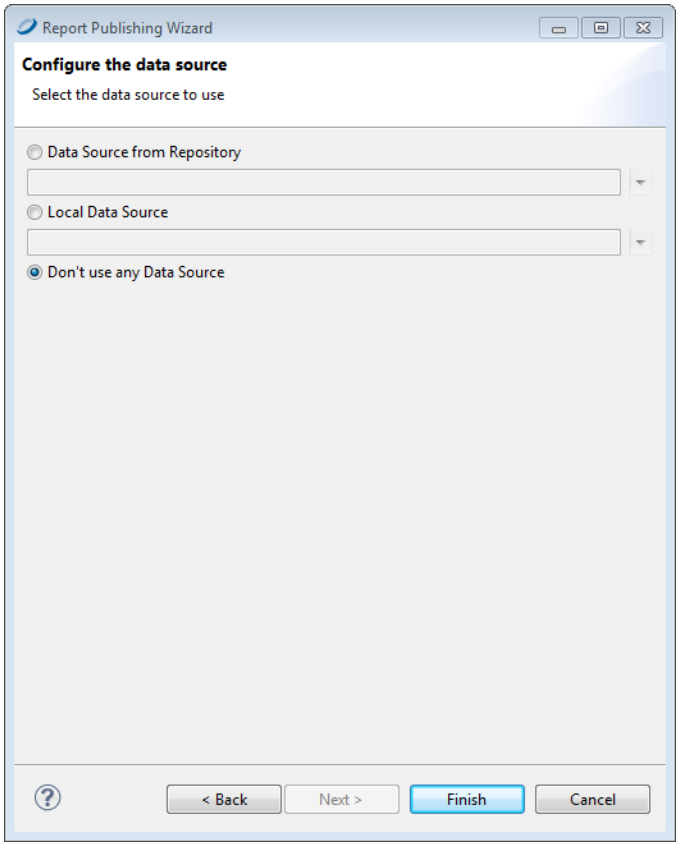


Figure 5-5 Configure Data Source

7. Select a data source, or no data source. Click **Finish**. The report is uploaded to the server. If there are no errors, an appropriate message is shown.

5.3 Publishing a Report Template

This section describes functionality that can be restricted by the software license for Jaspersoft Studio. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

You can add custom report templates to your JasperReports Server instance by uploading a JRXML file to a Templates directory. In addition to font and color choice, reports templates can contain images such as logos. In a template, the absolute path of an image is in the repository and cannot be overwritten. Other users can apply your template by selecting Custom Report Template when they create a report from an Ad Hoc View.

5.3.1 Creating a Report Template based on a Template in the Server

It's easiest to start with a template in JasperReports Server and change its properties (such as colors, fonts, and logos) in Jaspersoft Studio. Then, publish the new template to the server. This example shows how to change the font for a chart title.

To create a template:

1. Connect to JasperReports Server as `superuser`.
2. In the **Repository Explorer**, navigate to the Public/Templates directory.

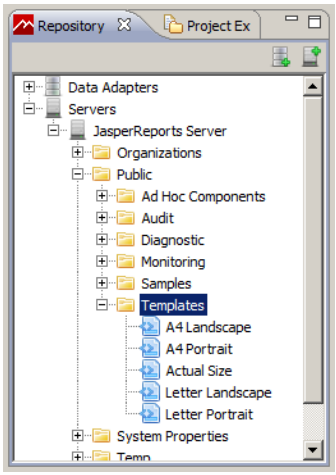


Figure 5-6 Accessing the Templates directory from Jaspersoft Studio

3. Right-click **A4 Landscape** and choose **Open in Editor**.

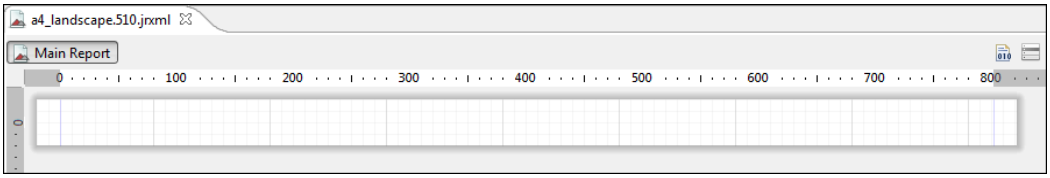


Figure 5-7 Default A4 landscape template

The document will look empty, but if you click the **Source** tab, you see that attributes are set at the JRXML level. Note the attributes for ChartTitle:

```
<style name="ChartTitle" forecolor="#000000" fontName="DejaVu Sans" fontSize="12"
isBold="true"/>
```

You can edit styles directly on the **Source** tab if you choose.

4. Click the **Design** tab and in the **Outline** view, click the arrow next to **Styles**.
5. Click **ChartTitle**. The styles open in the **Properties** tab.

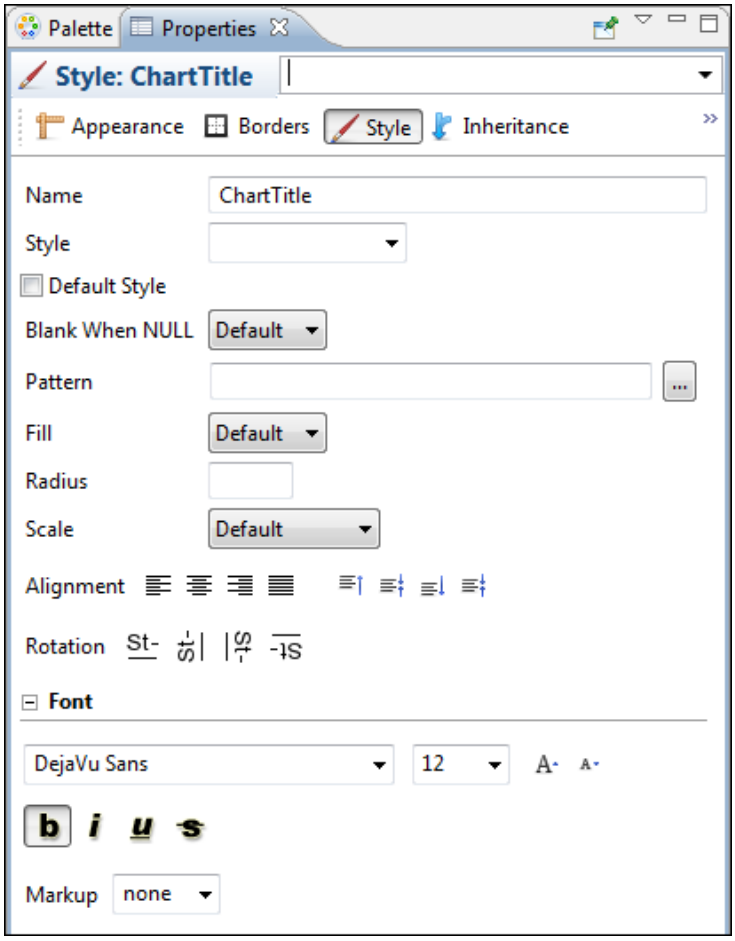


Figure 5-8 Style view in Properties tab

6. Make these changes:
   a. Font: Century Gothic, 14 pt, bold, italic.
   b. Change the forecolor to red (Click **Appearance** for that change.)
   c. Alignment: Center

You can link to an image in your template by uploading the image to the repository and then dragging it into the appropriate band in the template. The template uses the absolute path to the image in the repository and the image cannot be changed or overwritten.

To save and publish a template:

1. Save the template with a new name.
2. Click **Yes** in the pop-up to publish the report to JasperReports Server.
   The **Report Publishing** wizard appears.

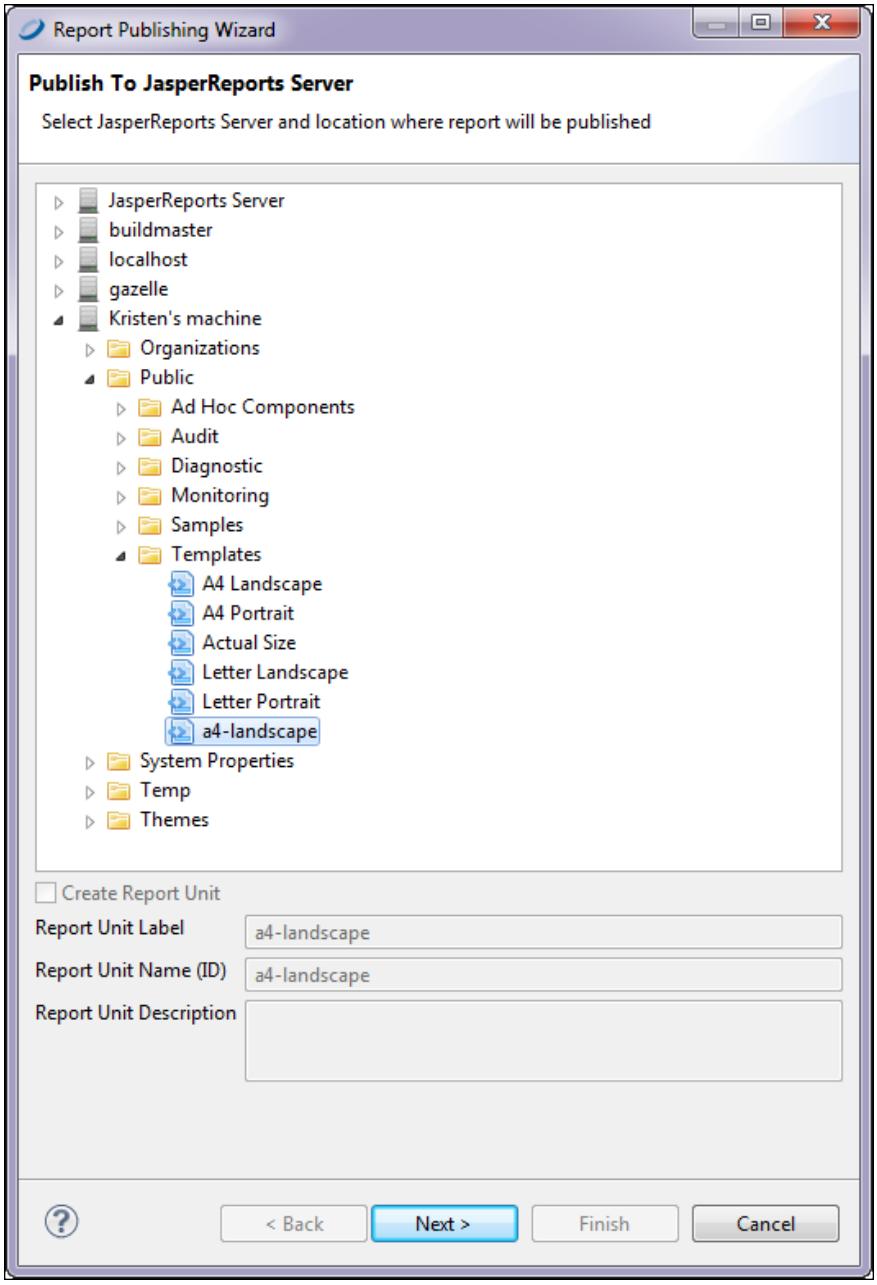


Figure 5-9 Report Publishing wizard

3. Select a folder to store your template, and click **Next**.

5.3.2 Report Template Styles in Jaspersoft Studio

In Jaspersoft Studio a report template includes styles that inherit attributes from other styles or from default values. Open one of the default templates in Jaspersoft Studio to see the available styles listed in the Outline view. When a report template is applied to a report that includes a basic chart, only the ChartTitle style is

applied to the chart. In general, report styles control the general look of the report, while chart themes control the look of basic charts (implemented through JFreeCharts). For more information on chart themes, see **12.5, "Chart Themes," on page 190**.

The Inheritance view in the Properties tab shows you which styles are inherited and from where. That makes it easy if you want to change a style at a higher level, or have an attribute inherited by more styles.

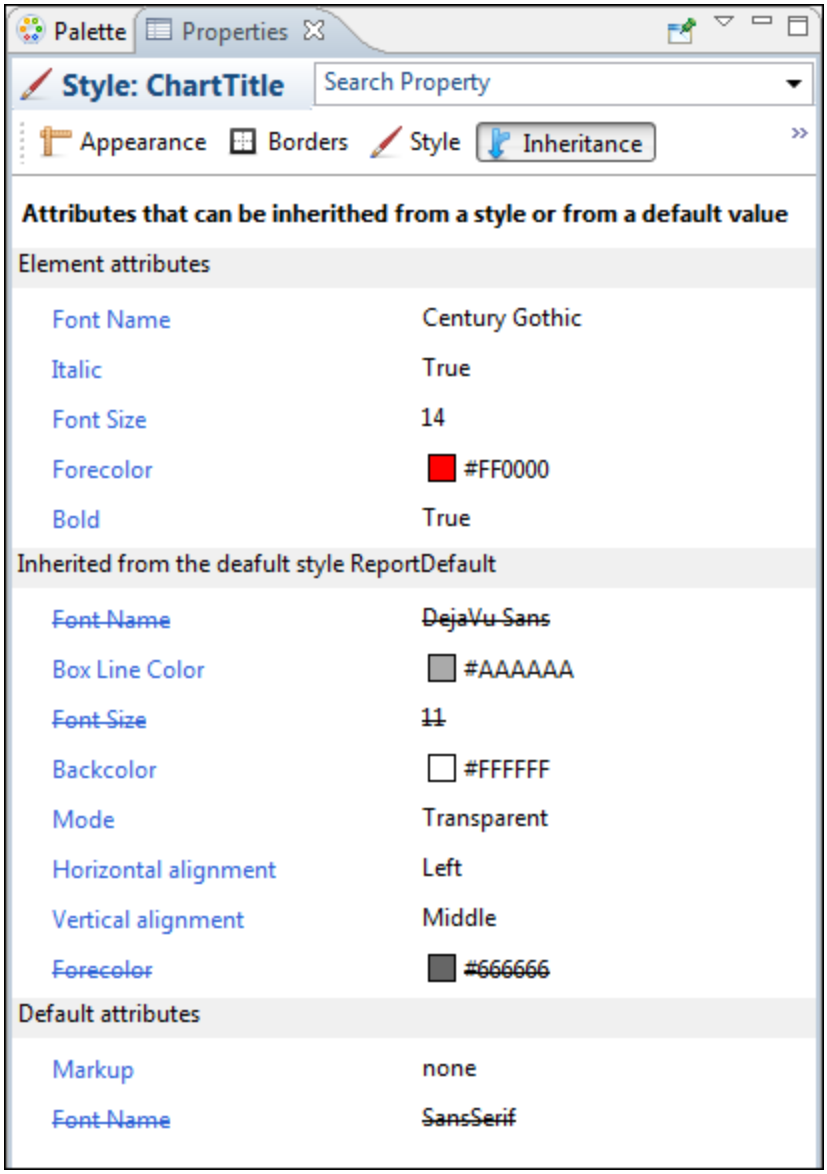


Figure 5-10 Inheritance tab

5.4 Creating and Uploading a Topic for Ad Hoc Views

When a JasperReports Server user creates an Ad Hoc view, she can select a Topic, Domain, or OLAP client connection to provide data to the view. This determines the data presented in the editor and the features available to the user. A Topic is the simplest to create, because it's just a report unit that defines a query. Most other elements of the report unit (such as its layout) are ignored when it's used as a Topic, though input controls you define in the report unit are carried over to the topic's Ad Hoc views and their dependent reports. Topics also support resource bundles and localization.

The following steps show how to create a new Topic based on the sample database provided with Jaspersoft Studio, and then upload it to the Topics folder in the JasperReports Serverrepository.

To create a Topic's JRXML:

1. In Jaspersoft Studio, click **File** > **New** > **Jasper Report**. The report wizard appears.
2. Select a report template and click **Next**.
3. Select a location to save the JRXML, enter a name for it, and click **Next**.
   In this case, name the file my-topic.jrxml.
4. Select a data adapter for the Topic. In this case, select the Sample DB - database JDBC connection.
   This sample includes the same data as the SugarCRM data source in the JasperReports Server samples.
5. In the text field, enter a query. In this case, enter:
   ```
   select * from orders
   ```
6. Click **Next**.
7. Move all the fields in the left list into the right list and click **Next**.
8. Click **Next** again to skip past the Group By option.
9. Click **Finish**. The wizard closes and Jaspersoft Studio displays the JRXML, which has been saved in the location specified in **step 3**.

To upload the Topic:

1. In the Repository Explorer, expand the Servers node and select the JasperReports Server instance where you want to put the Topic.
   If you haven't created any server connections, create one before proceeding. For more information, see **5.1, "Connecting to JasperReports Server," on page 77**.
2. Navigate to the Topic folder. For example, if you are logged in as jasperadmin, navigate to **Ad Hoc Components** > **Topics**.
3. Right-click the Topics folder and select **New**. The Add Resource wizard appears.
4. Click **Report Unit** and click **Next**.
5. Enter a name and optional description it and click **Next**.
6. Enable the **Local Resource** radio button it and click  to locate and select the JRXML you created above. For example, click **Upload/Download Resource**, click **Upload from Workspace**, select the my-topic.jrxml file.
7. Click **OK** to close the upload window and click **Next**.
8. Click the **Data Source from Repository** radio button and click  to its right.
9. Navigate to **Analysis Components** > **Analysis connections**, select the SugarCRM data source, and click **OK**.

10. Click **Finish** to upload the report unit to the Topics folder so it can be used in the JasperReports Server Ad Hoc Editor.

To test the Topic:

1. Log in to JasperReports Server.
2. Click **Create** > **Ad Hoc View**.
3. On the topics tab of the Data Chooser, open the Topics folder. For example, navigate to **Organizations** > **Topics**.
4. Click the Topic you created above and click **Table**, **Chart**, or **Crosstab**.
5. Verify that the fields you selected in Jaspersoft Studio all appear in the list of available fields.

If the Topic includes fields with unusual datatypes, those fields don't appear in the Ad Hoc Editor because JasperReports Server is not equipped to manage them properly. For example, if the Topic is defined against a MongoDB instance that returns data of type array, this field isn't available in the Ad Hoc Editor. For more information on datatype support in the Ad hoc editor, see the *JasperReports Server Administrator Guide*.

When you create a JRXML file for use as a Topic, you can specify the name to display for each field the Topic returns. To do so, define a field property named `adhoc.display` for each field declared in the JRXML. The `adhoc.display` field property must have the following syntax:

```
<property name="adhoc.display" value="Any Name"/>
```

For example, this JRXML code declares a StoreState field displayed in reports as Store State:

```
<field name="StoreState" class="java.lang.String">
 <property name="adhoc.display" value="Store State"/>
</field>
```

Topics also support the $R expressions for field names; for more information, see **2.3, "Expressions," on page 28**.

For fields in a non-domain topic the following properties may be of interest:

- `dimensionOrMeasure`, which marks a field as a field or a measure
- `defaultAgg`, the aggregation to use for this measure (for example, `avg`)
- `semantic.item.desc`, a description of the field
- `DefaultMask`, which sets a measure as a $ or date

5.5 Managing Repository Objects through Jaspersoft Studio

The Repository Explorer lets you create, view, modify, and delete reports units and the resources they reply on.

This section describes these tasks:

- **Adding, Modifying and Deleting Resources**
- **Running a Report**
- **Editing a Report**
- **Creating and Uploading Chart Themes**

5.5.1 Adding, Modifying and Deleting Resources

You need to create and manage the resources associated with your reports, such as images, JARs, JRXML files, property files for localized reports, input controls, datatypes, lists of values, style templates (JRTX), and data sources. If you're maintaining existing reports, you may need to modify existing resources. You can also change the location, name, or description of the repository folders.

You can add, modify, or delete repository resources from Jaspersoft Studio. In the Repository panel, expand your JasperReports Server repository and take one of the following actions:

- To add a resource, right-click a folder, select **New**, then select the type of object you want to add.

  If you choose to add an item other than a JasperReport, a dialog appears for entering information about the object. If you choose to add a JasperReport, a wizard guides you through the process. For the best results when adding a JasperReport, open the JRXML in Jaspersoft Studio and click .

  Follow the steps in the wizard to publish your report. See **5.2, "Publishing a Report to JasperReports Server," on page 79**

- To change the location of a repository resource, drag it to a new location.
- To delete a resource from the repository, right-click it and select **Delete**.
- To modify a repository resource, right-click it, select **Properties**, and make your changes in the Properties dialog. On the **General** tab, you can view the object's repository ID, name, and description. (Available tabs depend on the selected resource.)

  If you are logged in as a user with sufficient access rights (such as jasperadmin or superuser), you can modify property values and save them back to the repository.

- To change an input control, use the **Input Control Resource** tab.

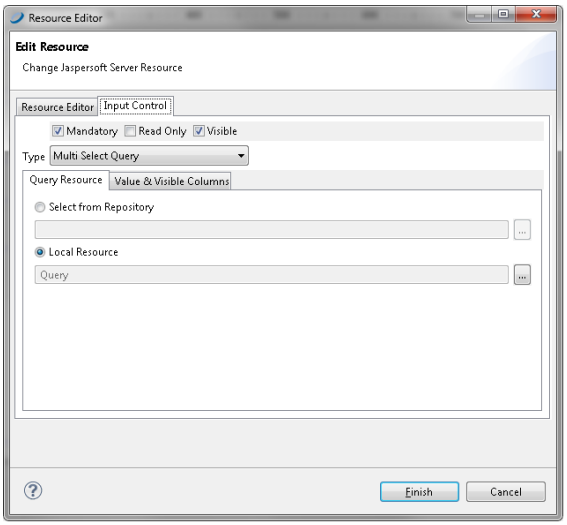


Figure 5-11 Properties of an Input Control Resource

5.5.2 Running a Report

Connect to JasperReports Server to test report changes you make in Jaspersoft Studio. See **5.1, "Connecting to JasperReports Server," on page 77**.

Navigate to your report's JRXML, and click **Run Report Unit**. If prompted to save the report unit, specify a location on your local computer and click **OK**. If the report has input controls that require values, you'll be prompted to specify them.. The report appears in a browser window.

5.5.3 Editing a Report

In the **Repository Explorer**, the  icon means a report unit, and  means a JRXML file. When you work with a JRXML file in the Repository, Jaspersoft Studiooperates on a copy of the file. You need to upload the JRXML file to put it back into the repository when finished.

To edit a JRXML file in the Repository:

1. In the **Repository Explorer**, right-click the JRXML file, and select **Open in Editor**. The JRXML appears in the Design tab.

   The JRXML is stored locally in your workspace. The default location is in the user directory of your operating system. For example, in Windows, the default workspace is C:\Users\<username>\JaspersoftWorkspace\MyReports.
2. Edit the file, either in the **Design** tab or in the **Source** tab. For example, in the **Repository Explorer** navigate to the Images\JR Logo image resource, and drag it into to the report's Title band. The logo appears in the **Design** tab.
3. Click **Save**. If you're prompted to publish the report, click **Yes**.
4. Specify a server and a repository location. To save the JRXML to the same report unit where you opened it, click **Next**.

   If the report relies on resources, you're asked if you want to overwrite the resources currently in the repository. If you added resources to the report, you are prompted to add them to the repository.
5. Click **Next**. and specify a data source for the report. You can't change a data source through the Publish wizard.

   Click **Finish**. Your changes are saved to the repository.

To edit a Report Unit in the Repository:

1. In the **Repository Explorer**, right-click the report unit and select **Properties**.
2. On the **Resource Editor** tab, change the name and description.
3. On the **Report Unit** tab, you can change the JRXML file for the report, either by selecting one from the repository, or uploading one through Jaspersoft Studio.
4. On the **Data Source** tab, select the data source from the repository or from Jaspersoft Studio.
5. On the **Input Controls** tab, set the display properties for any input controls:
   - Pop-up Screen: the controls are shown on-top of the report viewer.
   - Separate Page: the controls are shown in a different page than the report viewer.
   - Top of Page: the controls are shown at the top of the report viewer.
   - In Page: the controls are shown next to the report viewer.

6. You can also use the JSP field to modify the appearance of the controls. Specify a name of a JSP file in WEB-INF of the server's host to define the page that displays input controls.
7. Click **Finish**.

5.6 Creating and Uploading Chart Themes

Using Jaspersoft Studio, you can create new chart themes to give a custom look to any JFreeChart. You can set the fonts, colors, line widths, and other settings that determine the appearance of charts. Then upload the chart theme for use in reports generated on the server, either on a report-by-report basis or as a global setting for all charts that don't provide their own theme.

To create a new chart theme in Jaspersoft Studio:

1. Select **File > New > Other**. The New wizard appears.
2. Expand **Jaspersoft Studio**, select **Chart Themes**, and click **Next**. The new file dialog suggests a default file name. Chart themes use the .jrctx file extension.

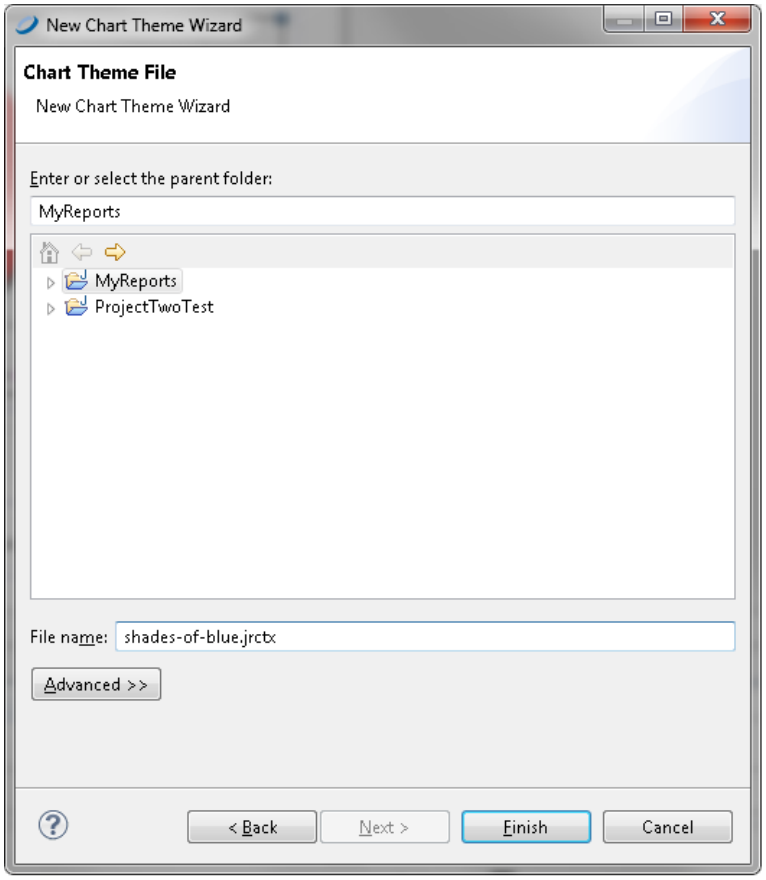


Figure 5-12 New Chart Theme in Jaspersoft Studio

3. Specify a location, enter a name, and click **Finish**.

The chart theme editor appears; it displays several types of charts to help you understand how the theme will be applied to each.

The available options are based on the JFreeChart library used to generate charts.

Jaspersoft Studio supports only the most common options provided by JFreeCharts.

4. In the Outline pane, select each category and review the available options in the Properties tab (typically found beneath the Preview tab).
5. Select a property to change its value.

   Depending on the nature of the property, you might type text, select a color, check or clear a check box, or select a value from a drop-down. As you update the chart theme, the Preview tab shows your changes. For example, select Title in the Outline pane and choose Bottom from the Position drop-down to move the title beneath the chart.
6. Click a chart type in the Preview pane to zoom in to examine the effects of your changes more closely. Click again to zoom out.

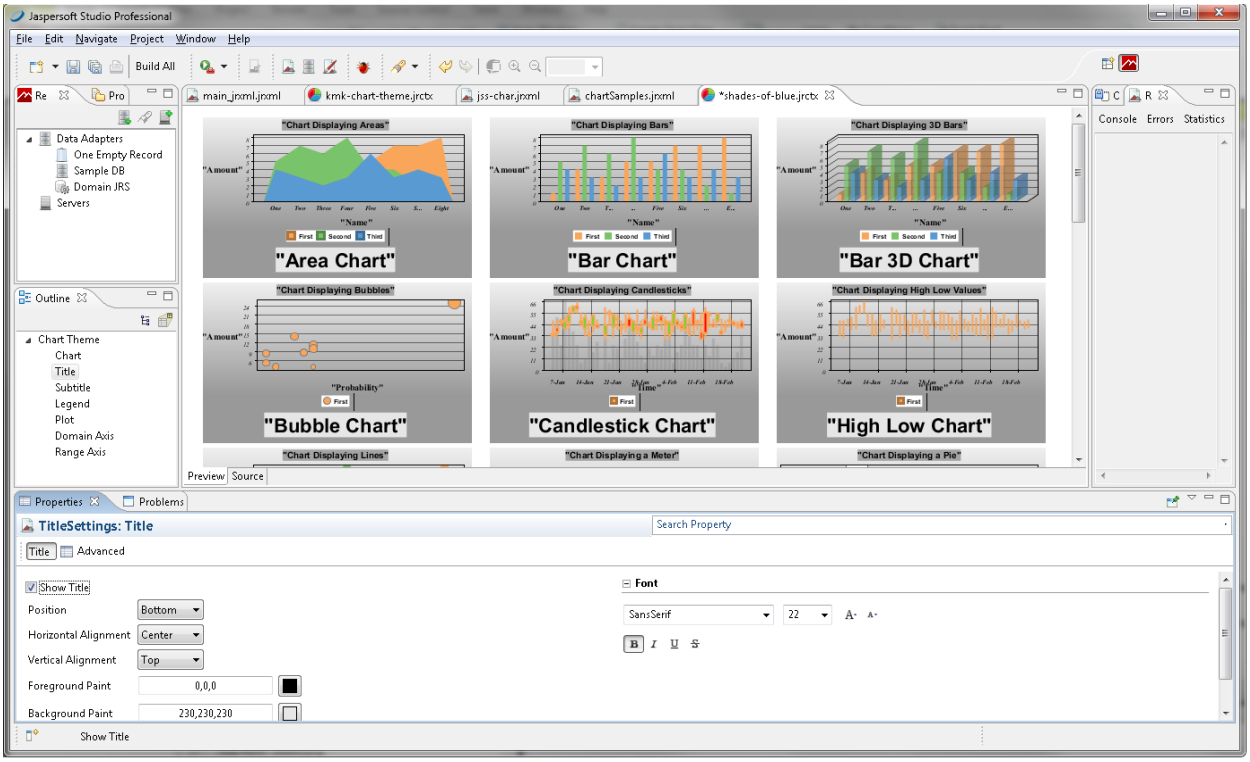


Figure 5-13 A Chart Theme Edited in Jaspersoft Studio

7. To view the XML that defines the chart theme's appearance, click the **Source** tab.
8. When you are satisfied with the chart theme, click **File > Save** to save the chart theme. This saves the chart theme to your local hard drive.
9. In the Project Explorer, locate and right-click your chart theme, and click **Export.**

10. In the dialog that appears, expand **Java**, choose **JAR file**, and click **Next**. If **Java** doesn't appear in the dialog, you may need to enable the option: click **Preferences>Capabilities** and enable the **Development > Java** option.
11. In the Export dialog, specify a location and a file name for your chart theme JAR.
12. Click **Finish**. The chart theme is exported.
13. To use the chart theme in Jaspersoft Studio, you must add the JAR to your project's classpath. In the Project Explorer, right-click the JAR and choose **Build Path** > **Add to Build Path**. If your Project Explorer isn't displayed, click **Window** > **Show View** > **Other** > **Project Explorer**.

The theme can be used at the report or server level in JasperReports Server. For more information, refer to the *JasperReports Server Administrator Guide*. In Jaspersoft Studio, you can upload the chart theme to the server using the Repository Explorer; right-click the folder in the repository where the theme should reside, select **New > Jar**and click **Next**. You'll need to enter a name and ID, to specify the JAR to upload. Once uploaded, you can use the repo: syntax in a report to specify this JAR as your chart theme.

5.7 Working with Domains

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

You can create reports based on Domains defined in JasperReports Server. Such reports use data adapters to load data stored in the Domains. To create a Domain-based report, create a data adapter and design a report with its data. Before you create these objects, you'll need a connection to the server. See **5.1, "Connecting to JasperReports Server," on page 77**.

To create a Domain-based report:

1. Click **File** > **New** > **JasperReport**. The New Report Wizard appears.
2. Select a template and click **Next**.
3. Select a location to save your report, enter its name, and click **Next**.
4. When prompted for a data adapter, select Domain JRS - Query Executor adapter.
5. Select a server connection. Ignore the notice that changing the server resets the query. As this is a new report, there's no query yet.
6. Click **Yes**. The **Domain** drop-down is populated with a list of Domains defined on the server.
7. Click the Domain containing the data for your report. Ignore the notice that changing the Domain resets the query. As this is a new report, there's no query yet.
8. Click **Yes**.
9. Select items in the Domain on the left-side of the dialog, and drag them to the right-side to create fields and filters.
10. When you have the fields and filters you need for your report, click **Next**. You are prompted to select fields for your dataset.
11. Select the fields to include, and click **Next**. You are prompted to define grouping in your dataset.
12. Select a field for grouping your data and click **Next**.

13. Click **Finish** to complete your report. The blank report appears in the Design tab. Use the **Palette** and **Outline** to define your report as usual.
14. Click **Preview** to test your report. Jaspersoft Studiocompiles your report; if it's successful your report is filled and displayed.
15. Review your report, make any needed changes, and save it.
16. Click  to publish your report. For more information, see **5.2, "Publishing a Report to JasperReports Server," on page 79**.

CHAPTER 6 WORKING WITH FIELDS

In a report, there are three groups of objects that can store values:

- Fields
- Parameters
- Variables

Jaspersoft Studio uses these objects in data source queries. In order to use these objects in a report, they must be declared with a discrete type that corresponds to a Java class, such as `String` or `Double`. After they have been declared in a report design, the objects can be modified or updated during the report generation process.

This chapter contains the following sections:

6.1 Understanding Fields

A print is commonly created starting from a data source that provides a set of records composed of a series of fields. This behavior is exactly like obtaining the results of an SQL query.

Jaspersoft Studio displays available fields as children of the **Fields** node in the document outline view. To create a field, right-click the **Fields** node and select **Create Field**. The new field is included as an undefined entry on the **Properties** tab. You can configure the field properties by selecting it.

Press the **Object** button to name your field, enter a description, and choose a class.

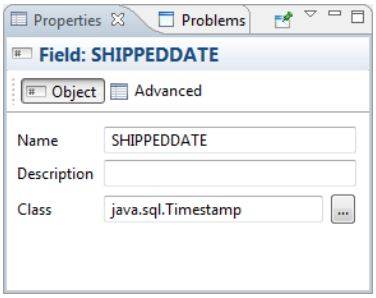


Figure 6-1 Field Properties Tab - Object

Press the **Advanced** button to enter advanced properties for the field.

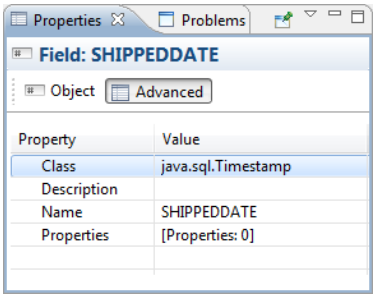


Figure 6-2 Field Properties Tab - Advanced

A field is identified by a unique name, a type, and an optional description. You can also define a set of name/value pair properties for each field. These custom properties are not generally used by JasperReports, but they can be used by external applications or by some custom modules of JasperReports (such as a special query executor).

Jaspersoft Studio determines the value for a field based on your data source. For example, when using an SQL query to fill a report, Jaspersoft Studio assumes that the name of the field matches the name of a field in the query result set. You must ensure that the field name and type match the field name and type in the data source. You can systematically declare and configure large numbers of fields using the tools provided by Jaspersoft Studio. Because the number of fields in a report can be quite large (possibly reaching the hundreds), Jaspersoft Studio provides different tools for handling declaration fields retrieved from particular types of data sources.

Inside each report expression (like the one used to set the content of a textfield) Jaspersoft Studio specifies a field object, using the following syntax:

```
$F{<field name>}
```

where `<field name>` must be replaced with the name of the field. When using a field expression (for example, calling a method on it), keep in mind that it can have a value of `null`, so you should check for that condition. An example of a Java expression that checks for a `null` value is:

```
($F{myField} != null) ? $F{myFiled}.doSomething() : null
```

This method is generally valid for all the objects, not just fields. Using Groovy or JavaScript this is rarely a problem, since those languages handle a null value exception in a more transparent way, usually returning an empty string.

In some cases a field can be a complex object, like a JavaBean, not just a simple value like a String or an Integer. A trick to convert a generic object to a String is to concatenate it to an empty string this way:

```
$F{myfield}+ ""
```

All Java objects can be converted to a string; the result of this expression depends on the individual object implementation (specifically, by the implementation of the `toString()` method). If the object is null, the result returns the literal text string "`null`" as a value.

6.2 Registration of Fields from a SQL Query

An SQL query is the most common way to fill a report. Jaspersoft Studio provides several tools for working with SQL, including a query designer and a way to automatically retrieve and register the fields derived from a query in the report.

Before opening the query dialog, be sure you select the correct connection/data source. All operations performed by the tools in the query dialog use this data source.

To open the query dialog (**Figure 6-3**) right-click the name of your report in the **Outline** view and choose **Dataset and Query...**.

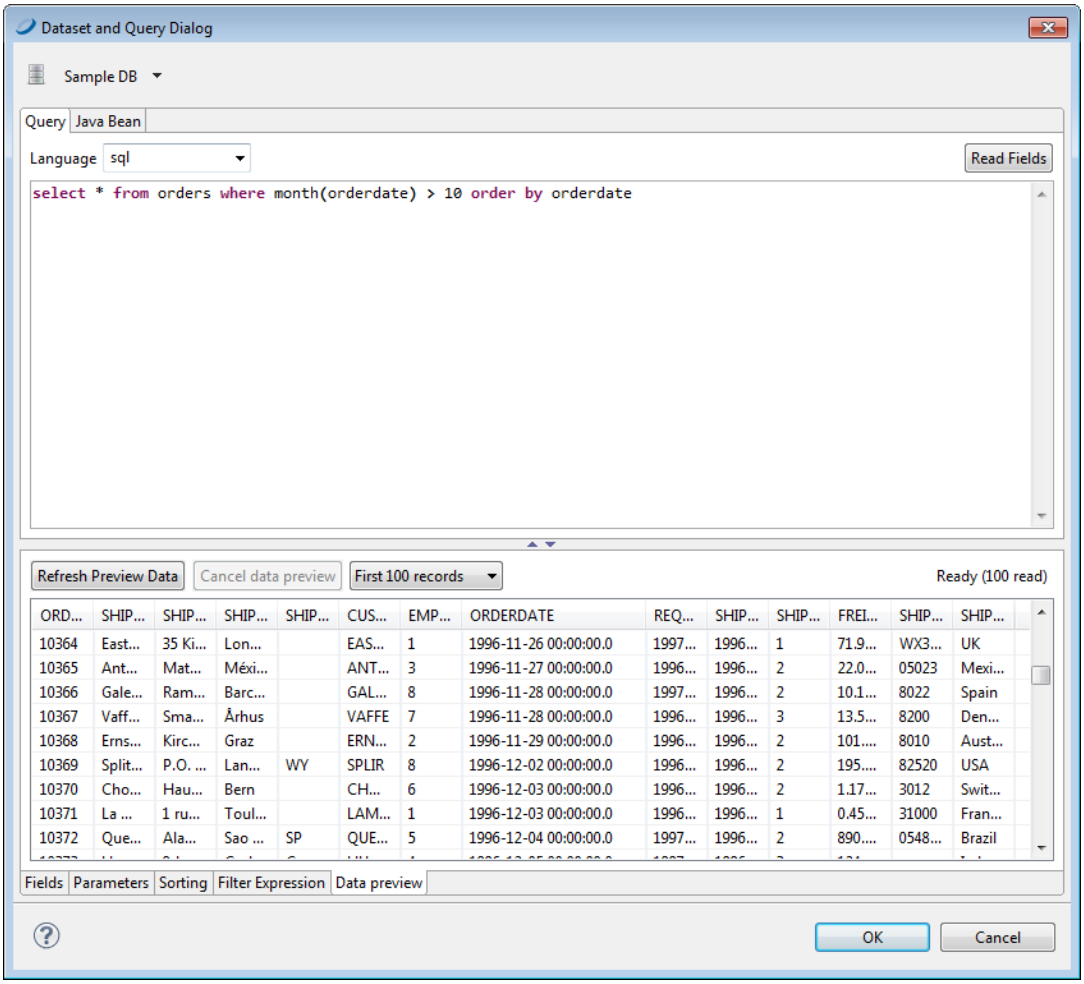


Figure 6-3 Query Dialog

Jaspersoft Studio doesn't require a query to generate a report. It can obtain data from a data source that is not defined by a query execution. JasperReports supports multiple query languages including:

- CQL
- HiveQL
- JSON
- MongoDBQuery
- PLSQL
- SQL
- XLS
- XPath

If the selected data source is a JDBC connection, Jaspersoft Studio tests the access connection to the data source as you define the query. This allows Jaspersoft Studio to identify the fields using the query metadata in the result set. The design tool lists the discovered fields in the bottom portion of the window. For each field, Jaspersoft Studio determines the name and Java type specified by the JDBC driver.

If your query accesses tables containing large amounts of data, scanning the data source for field names could take a while. In this case you might consider disabling the **Automatically Retrieve Fields** option to quickly finish your query definition. When you've completed the query, click the **Read Fields** button to start the fields discovery scan.

All fields used in a query must have a unique name. Use alias field names in the query for fields having the same name.

The field name scan may return a large number of field names if you are working with complex tables. To reduce unnecessary complexity, we suggest that you review the list of discovered names and remove fields you're not using in your report.When you click **OK** all the fields in the list are included in the report design. Although you can remove them later in the outline view, it's a good idea at this point in the design process to remove any field names that you won't be using.

6.3 Registration of JavaBean Fields

One of the most advanced features of JasperReports is its ability to manage data sources that aren't based on simple SQL queries. One example of this is JavaBean collections. In a JavaBean collection, each item in the collection represents a record. JasperReports assumes that all objects in the collection are instances of the same Java class. In this case the "fields" are the object attributes (or even attributes of attributes).

By selecting the **Java Bean** tab in the query designer, you can register the fields that correspond to the specified Java classes. We assume you know the Java classes that correspond to the objects that you use in your report.

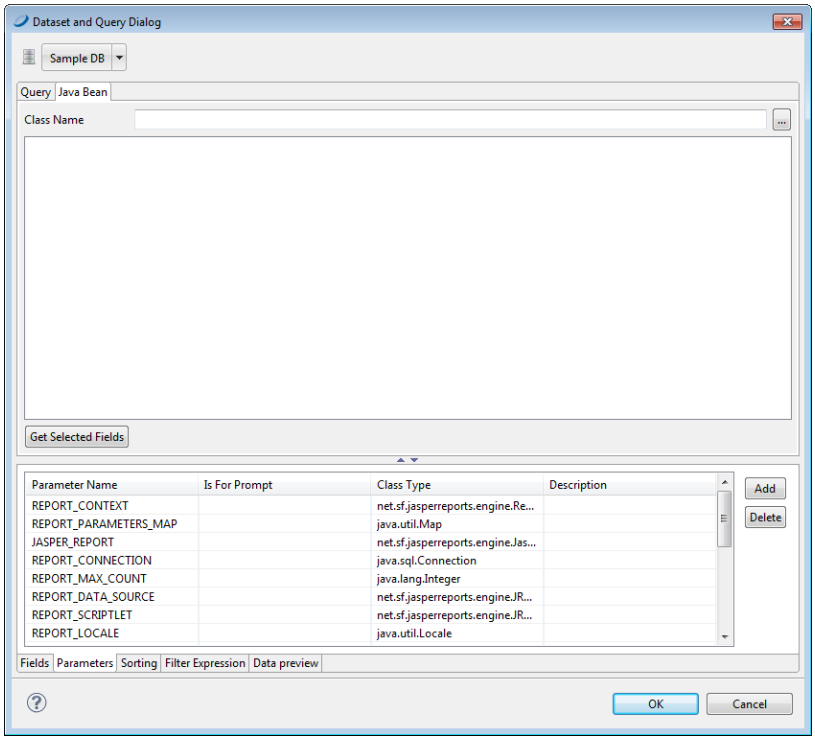


Figure 6-4 JavaBeans Tab

Suppose you're using objects of this Java class:

```
com.jaspersoft.ireport.examples.beans.PersonBean
```

To register fields for the class:

1. Put the class name in the name field and click **Read attributes**. Jaspersoft Studio scans the class.
2. Check the scan results to make sure Jaspersoft Studio has captured the correct object attributes for the class type.
3. Select the fields you want to use in your report and click **Add**.

Jaspersoft Studio creates new fields corresponding to the selected attributes and adhesion to the list. The description, in this case, stores the method that the data source must invoke to retrieve the value for the specified field.

Jaspersoft Studio parses a description such as `address.state` (with a period between the two attributes) as an attribute path. This attribute path is passed to the function `getAddress()` to locate the target attribute, and then to `getState()` to query the status of the attribute. Paths may be arbitrary and long, and Jaspersoft Studio can recursively parse attribute trees within complex JavaBeans and in order to register very specific fields.

We have just discussed the two tools used most frequently to register fields, but we're not done yet. There are many other tools you can use to discover and register fields, for instance, the HQL and XML node mapping tools.

6.4 Fields and Textfields

To print a field in a text element, you must set the expression and the `textfield` class type correctly. You may also need to define a formatting pattern for the field.

To create a corresponding textfield, drag the field you want to display from the Outline view into the design panel. Jaspersoft Studio creates a new textfield with the correct expression (for example, `$F{fieldname}`) and assigns the correct class name.

6.5 Data Centric Exporters

Jaspersoft Studio supports two data-oriented export formats designed to be used programmatically when another application embeds TIBCO Jaspersoft products:

- CSV: exports the report's data to a list of comma-separated values (CSV).
- JSON: exports the report's data to a JavaScript Object Notation (JSON) object.

In both cases, the metadata defines the structure of the exported data.

Jaspersoft Studio also supports other types of field-level metadata:

- PDF 508 Tags are used when creating report output in Adobe Acrobat format that complies with the Americans with Disabilities 508 (c) specification.
- XLS Tags are used to define how data is exported to the Microsoft Excel format. In addition to numerous layout settings, you can define XLS metadata that define the structure of the data when exported.

This section describes how to work with metadata for the JSON exporter.

6.5.1 Configuring a Report's Metadata for Use With the JSON Data Exporter

JasperReports Server's REST API includes a JSON (JavaScript Object Notation) data exporter that enables you to feed pure data into applications you integrate with the server. During report generation, this exporter skips all layout-related steps and returns a dataset. The structure of this data is determined by metadata you define in your report. You can also define expressions to determine how data from a specific filed is exported.

Note: The ability to define metadata and export data in JSON format is sometimes referred to as the JasperReports Data API.

You can define a structure by separating the names of the levels you want to create with periods (.). For example, consider a report with three fields configured with these JSON properties:

Field Expression	JSON Path
$F{salesamount}	store.sale.amount
$F{salesyear}	store.sale.year
$F{cust.name}	store.cust.name

When exported to JSON, the data is structured with three distinct paths:

```
store
   sale
      amount
      year
   cust
      name
```

Example exported data would be similar to:

```
[
   {store:
      [
         {
            sale:[{amount:"19000", year:2014}],
            cust:[{name:"Acme"}],
         }
      ]
   }
]
```

Note that when you preview your report as JSON, the data is not formatted to be human readabable, as above. You may want to use one of the many JSON formatting tools to review the output of your JSON tagged report, you can copy the JSON output from the **Preview** tab.

It's important to define paths that create a structure that the application receiving the data can interpret.

To define JSON export object metadata in your report:

1. Open a report that includes the fields you want to export to your application.
2. Right-click a field in the **Design** tab, and select **JSON tags > JSON Metadata Path**.

   If the field you selected appears in a frame, you're warned that JasperReports Library may ignore the property. This warning relates only to older versions of the library; it remains in the product for backwards-compatibility. For current versions of JasperReports Server, JasperReports Server, and Jaspersoft Studio, properties defined in frames aren't ignored.
3. If you receive this warning, click **OK**. The JSON Exporter Property Configuration window appears.
4. In the **Path** field, enter a string that specifies the way the data from this field should be exported. For example, if you are working with a field that returns a sales amount value, you might enter store.sale.amount.
5. If the data being returned necessitates it, check the **Repeat value if missing** check box.

   This option is helpful if your source data doesn't include values for every row of data returned. Selecting this option instructs Jaspersoft Studio to simply use the last value passed when a value is missing, which may prevent problems in the application receiving the JSON object.
6. If you want to manipulate the data being exported, check the **Use custom expression for exported value** check box, click  , and define an expression.
7. Click **OK**.
8. Select each field you want to export to JSON and define its metadata.
9. Click **File > Save**.
10. Click **Preview**.
11. If the JSON Metadata preview isn't selected, click the arrow next to the current preview format, and select **JSON Metadata**.

Figure 6-5 Selecting the JSON Metadata Preview

12. Review the structure of the data to ensure your application can interpret it.
13. If the data isn't structured correctly, click **Design** and edit each field's JSON export properties.
14. When you're satisfied with the data returned by Jaspersoft Studio, you can publish your report to JasperReports Server and begin testing your own application's ability to use the data passed by the server.

CHAPTER 7 REPORT TEMPLATES

Templates is one of the most useful tools of Jaspersoft Studio. You can use the provided templates as the basis for new reports. You can also use a template as a model and add fields, text fields, and groups in the Report Wizard.

This chapter explains how to build custom templates that will appear in the Template Chooser. It has the following sections:

- **Template Structure**
- **Creating and Customizing Templates**
- **Saving Templates**
- **Adding Templates to Jaspersoft Studio**

7.1 Template Structure

A template is a JRXML file. When you create a new report,Jaspersoft Studio loads the selected template's JRXML file with any modifications you've specified in the wizard. If you don't use the wizard, your template is just copied along with all the referenced images in the location the you've specified.

When you launch the Report Template dialog (**File > New > Jasper Report**) scans all paths specified as template directories. Any valid JRXML files found are included in the Report Template dialog. If a template provides a preview image, the image is displayed. Otherwise, a white box appears.

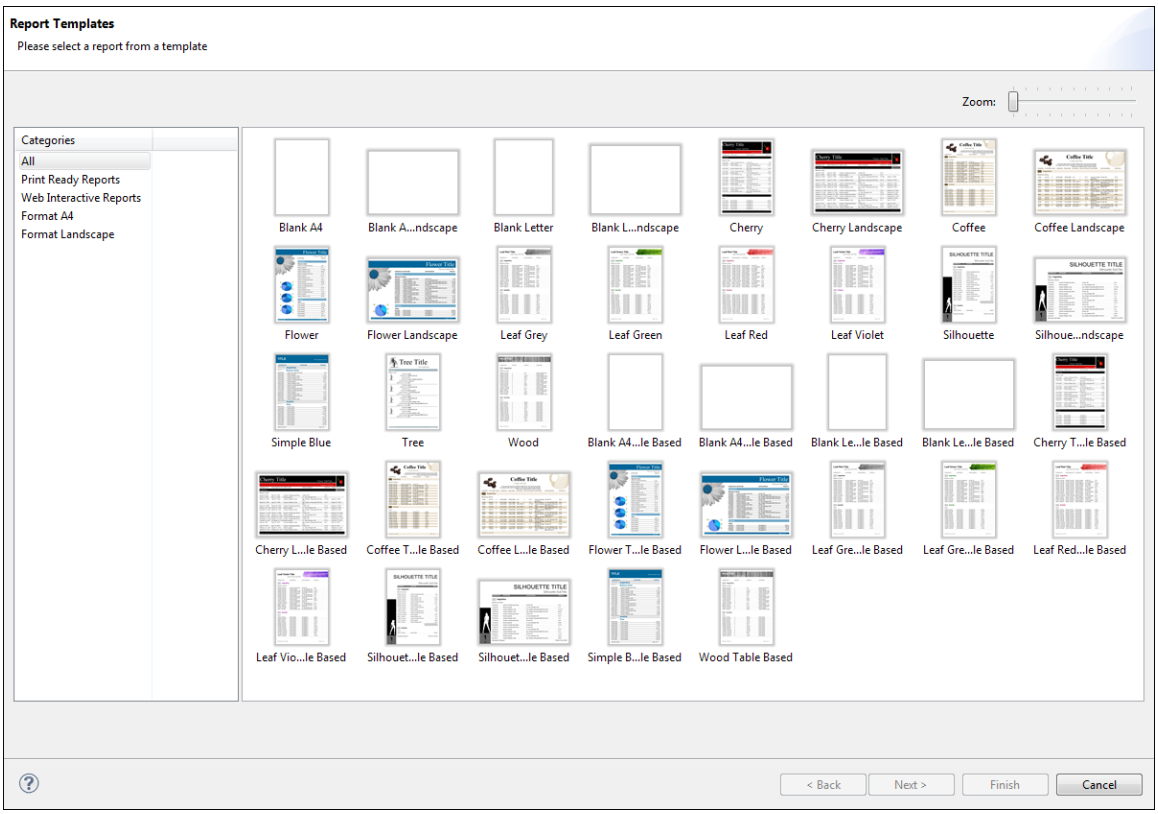


Figure 7-1 Default Report Templates

A template contains all or some of the same parts as a report. Remember the following when creating a new template or editing an existing template:

- A report's page formatting is the page formatting of its template.
- Each band in a report is, by default, the same size as that band in its template.
- Every element placed in the **Summary**, **Title**, **Page Header** and **Page Footer** bands, of a template appears in every report that uses that template.
- The **Column Header** band should contain only a **Static Text** element, and its text content must be `Label`. The appearance, font and the other attributes of this label create every label inserted in this band.
- The **Group Header** band should contain only a **Text Field** with the string "`GroupField`" (including the double quotes). As with the **Column Header** this assumes an example to generate every field that goes in this band.
- The **Detail** band should contain only a **Text Filed** with the string "`Field`" (including the double quotes). Again, this is used to generate every field that goes in this band.

When you group data using the wizard, the wizard creates all the necessary report structures to produce your groups. The Report Wizard supports up to four groups, with a group header and group footer associated with each. If the template defines one or more groups and you group the data, the wizard tries to use any existing groups before creating new ones. By default, groups in the template are deleted if they're not used. For each group, the wizard sets the group expression and adds a label for the name and a text field showing the value of the group expression (which is always a field name, because the grouping criteria set using the wizard is one of the selected fields).

7.2 Creating and Customizing Templates

You can create templates from scratch or edit existing templates and save them as new templates.

7.2.1 Creating a New Template

If you want to start fresh with your template, create a new one.

To create a new template:

1. Go to **File > New > Jasper Report** to launch the New Report Wizard.
2. Select a template to start. Click **Blank Letter** and **Next**.
3. Choose where you want to store the file, name the new template, and click **Next**.
   For creating a template, there is no need to connect to a data source.
4. Select **One Empty Record - Empty rows** and click **Finish**.

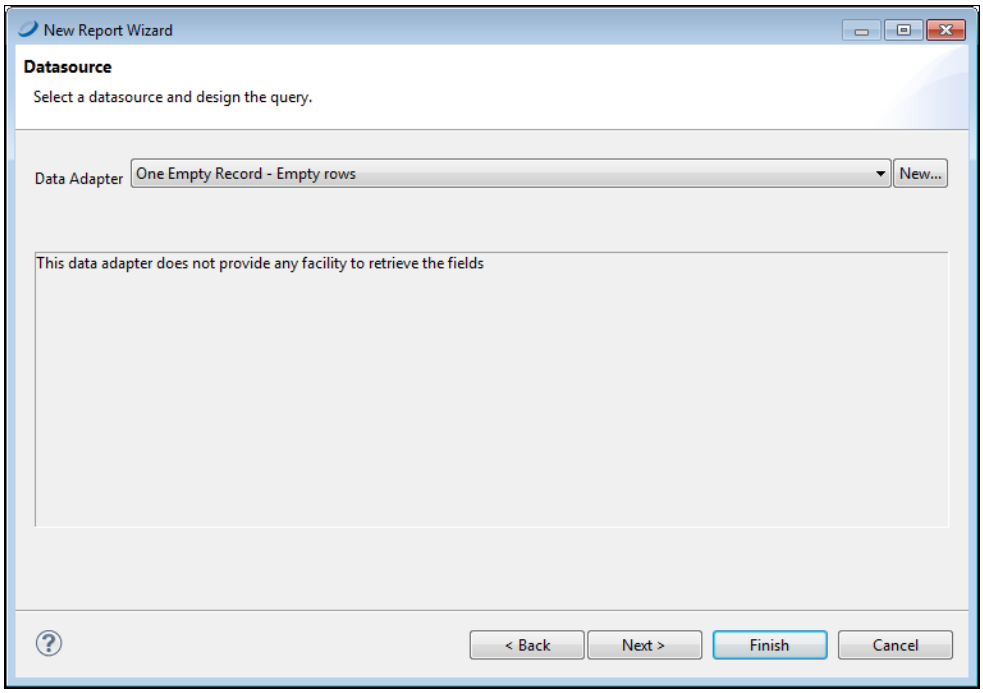


Figure 7-2 One Empty Record Data Source

An empty report opens, containing the following bands:

- Title
- Page Header
- Column Header
- Detail
- Column Footer
- Page Footer
- Summary

5. Right-click on the report root node in the **Outline**, and select **Create Group**. The **Group Band** dialog appears.

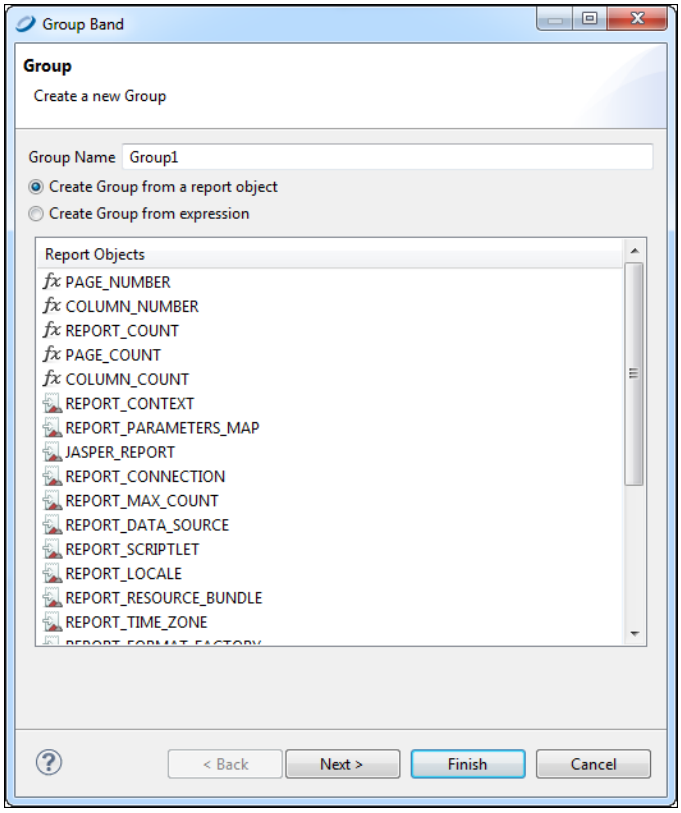


Figure 7-3 Group Band Dialog

6. Name your group and click **Next**. The **Group Layout** dialog appears.

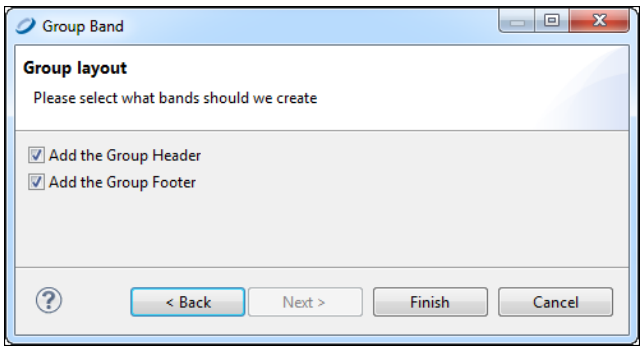


Figure 7-4 Group Layout Dialog

7. Leave both **Add the Group Header** and **Add the Group Footer** checked, and click **Finish**. Your report is similar to the one in **Figure 7-5**.

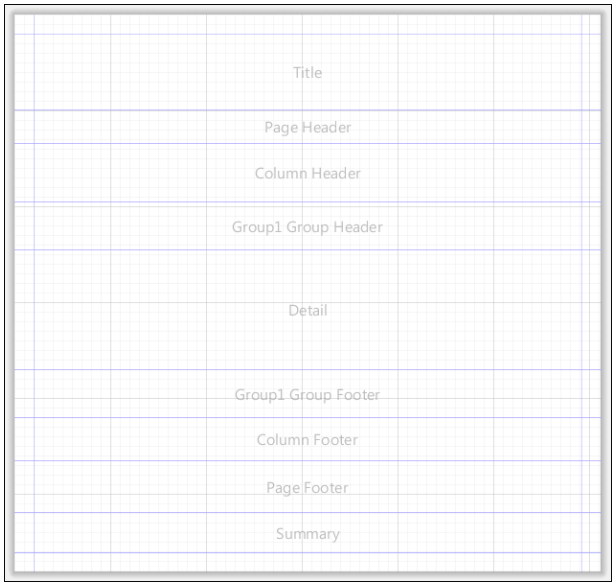


Figure 7-5 Report Containing Group Header and Footer

7.2.2 Customizing a Template

Now that you have a blank template, you can customize it to suit your preference. For example, you can add your company name and logo, page numbering, add a background for your report, and set band and column sizes. You also use this procedure to make changes to an existing template.

To customize a template:

1. Add a graphic: Drag an **Image** element where you want the image to appear. This is usually the Title band. For more information about the Image element, see **"Graphic Elements" on page 42**.
2. Add a title: Drag a **Static Text** element to the Title band. Style the text in the Properties tab. For more information about Static Text elements, see **"Text Elements" on page 43**.
3. Want the background to cover the entire page? Right-click the element in the **Outline** and choose **Maximize Band Height**. Otherwise, set the Background band to the size you want. Drag an **Image** element into the Background band to create your background.
4. Add page numbering to the Page Footer band: Drag a **Page Number** element into the band, and place it where you want it. You can also add a **Page X of Y** element if you prefer.
5. Want a label in the Column Header band? Add a **Static Text** element with the text "Label".
6. Set styles for your report's text: Add a **Text** field to the Group Header and a **Text** field to the Detail band. Set the text of the first **Text** field to `"GroupField"` and the text of the second **Text** field to `"Field"`. Format the text as you like.
7. Save your template file.
8. Click the **Preview** tab. Your template should like something like the one in **Figure 7-6**.

Figure 7-6 Template Preview

7.3 Saving Templates

Jaspersoft Studio templates require a flat folder structure (resources and report in the same folder). This way, when you export a template, the paths and resources in the exported report point to the same directory.

7.3.1 Creating a Template Directory

You can specify one ore more directories for your custom templates.

1. Go to **Window > Preferences > Jaspersoft Studio > Templates Locations**.

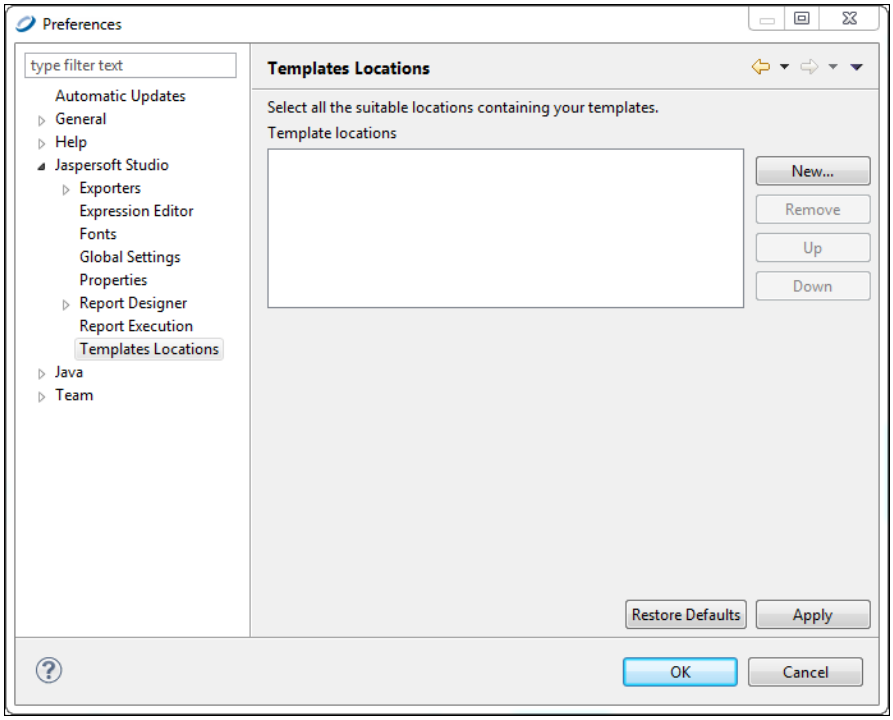


Figure 7-7 Template Location Preferences

2. Click the **New...** button and navigate to the directory in which you want to store your template.
3. Click the **Apply** button.
4. Click **OK**.

7.3.2 Exporting a Template

Save your template for future use.

1. Go to **File > Export as Report Template**. The **Template Export** dialog opens.

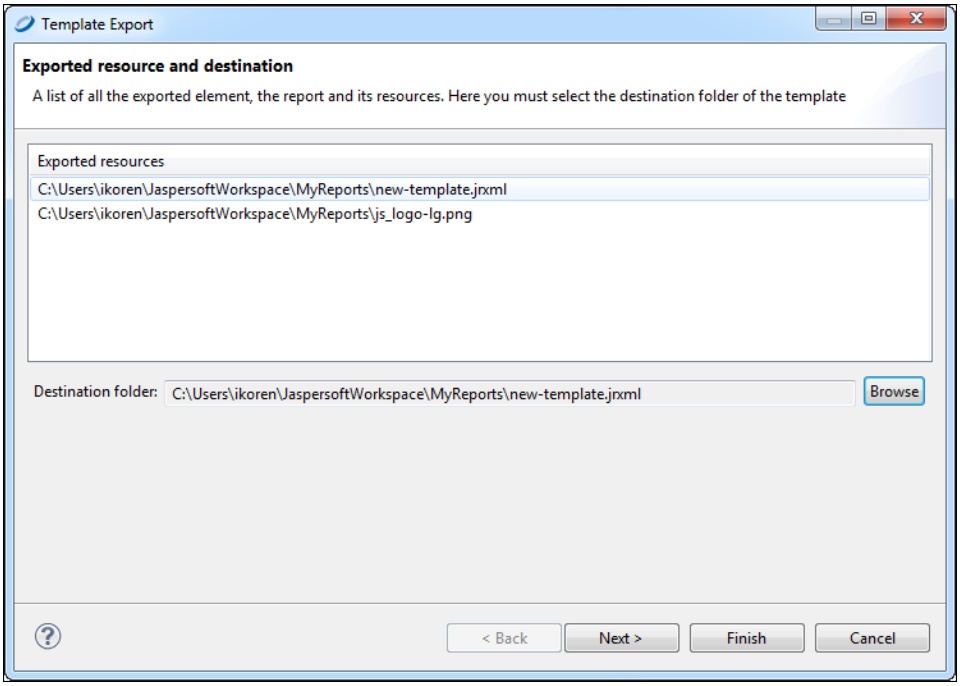


Figure 7-8 Template Export Dialog

2. Click the **Browse** button and navigate to the directory where you want to save your template. Click **Next**. The **Define Type and Categories** dialog opens.

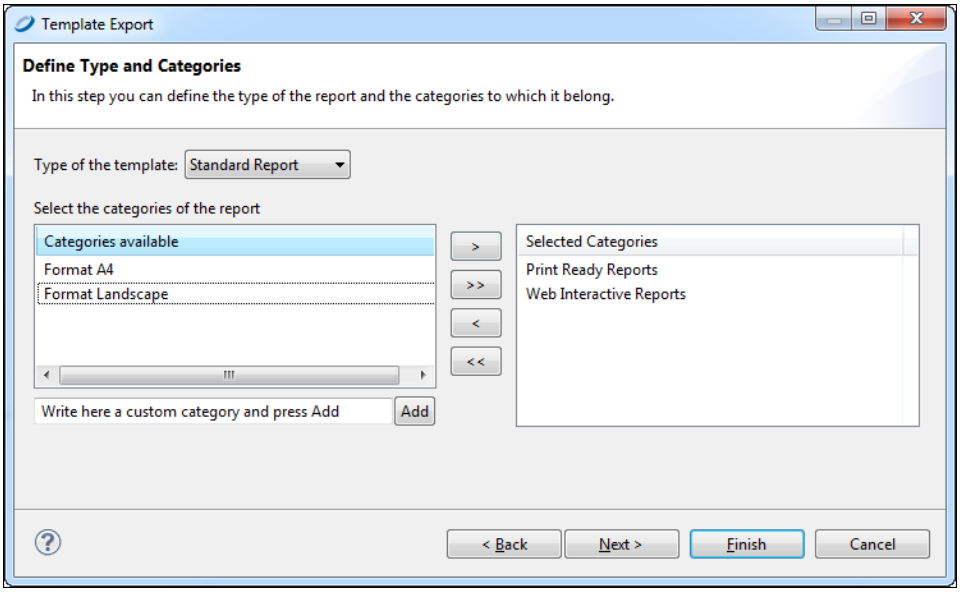


Figure 7-9 Define Type and Categories Dialog

3. In the drop-down, choose whether the template type is a Standard Report or a Table-Based report.

This selection is used to validate the report. For example. by selecting **Standard Report**, the validation process searches for the group field with the text group or for the column header label with the text label. If any of the required fields are not found, an error message is displayed.

4. Select the categories for the template available and use the arrow button to add them to the **Selected Categories**.
5. Click **Finish**.

7.3.3 Creating a Template Thumbnail

Saving a thumbnail is not required, but can be helpful if more than one person uses your templates.

1. Go to the **Preview** tab.
2. Click the Export Image button. 
3. Save the image in the same directory and with the same name as your template.

7.4 Adding Templates to Jaspersoft Studio

Once you've created a custom template, you need to add it to Jaspersoft Studio to use it.

To add a template to Jaspersoft Studio:

1. Go to **Window > Preferences > Jaspersoft Studio > Templates Locations**.

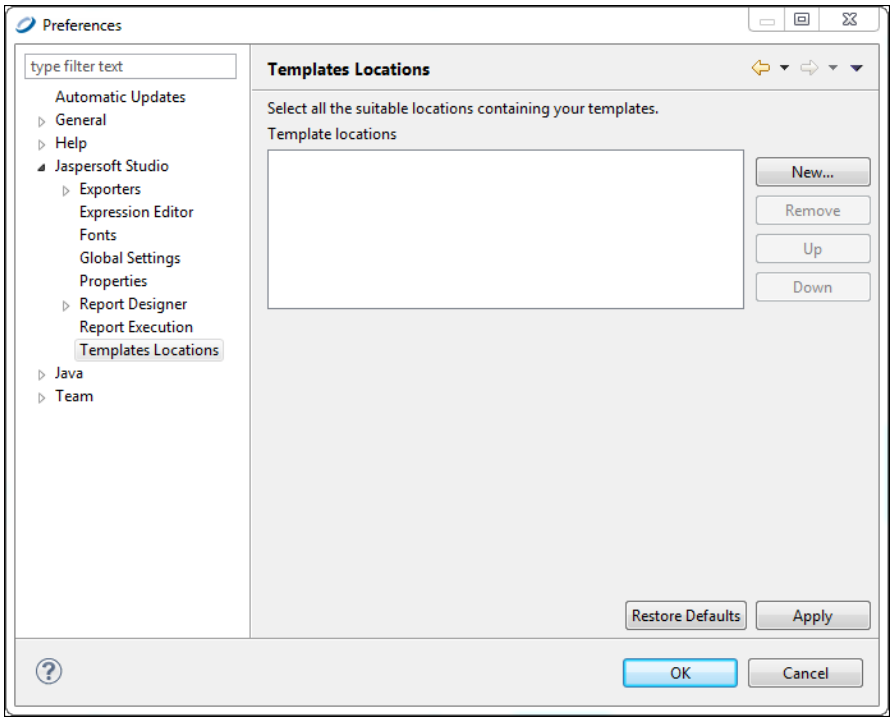


Figure 7-10 Template Location Preferences

2. Navigate to the directory in which you stored your template.

3. Click **Apply**.
4. Click **OK**.

When you go to **File > New > Jasper Report**, your new template appears, along with the default templates.

CHAPTER 8 USING PARAMETERS

Parameters are values usually passed to the report from the application that originally requested it. They can be used for configuring report features at generation (like the value to use in an SQL query), or maybe to supply additional data to the report that's not provided by the data source (like a custom report title, an application-specific path for images).

Report parameters have many functions in a report. They can be used in the "where" condition of an SQL query, or to provide additional data to the report (like the value of a title or name of the user that executed the report).

A parameter is defined by a name and a Class, which is a Java class type. For example a parameter of type java.sql.Connection may be used to populate a subreport, while a simple java.lang.Boolean parameter may be used to show or hide a section of the report.

This chapter contains the following sections:

8.1 Managing Parameters

Parameters are the best communication channel between the report engine and the execution environment (your application).

A parameter can have a default value defined by means of the default expression property. This expression is evaluated by JasperReports only when a value for the parameter has not been provided by the user at run time.

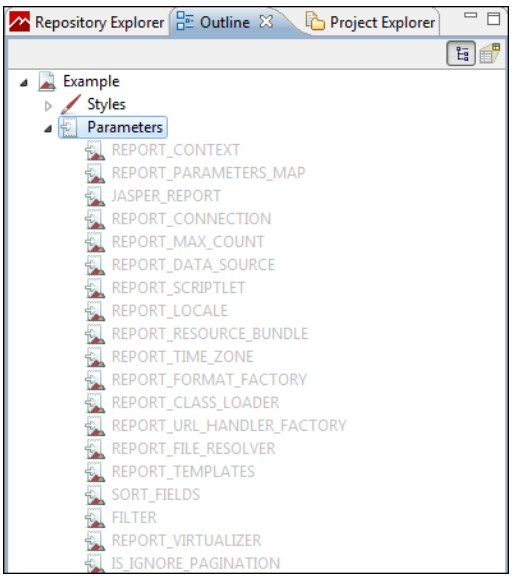


Figure 8-1 Parameters in Outline View

To manage parameters, use the outline view. Add a parameter by right-clicking on the item **Parameters** and choosing **Create Parameter**. To delete a parameter from the outline view right click on it and select **Delete**. The parameters with light gray names are created by the system and can not be deleted or edited.

Right-click any parameter and choose **Show Properties** to view and edit the properties of the parameter.

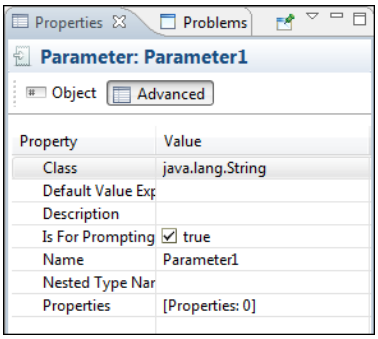


Figure 8-2 Parameters - Advanced Properties

The **Default Value Expression** field allows you to set a pre-defined value for the parameter. This value is used if no value is provided for the parameter from the application that executes the report. The type of value must match the type declared in the **Class** field.

You may legally define another parameter as the value of **Default Value Expression**, but this method requires careful report design. Jaspersoft Studio parses parameters in the same order in which they are declared, so a default value parameter must be declared before the current parameter.

The parameter **Description** is not used directly by JasperReports, but like the **Is for Prompting**, may be passed to an external application.

As with fields, you may specify pairs of type name/value as properties for each parameter. This is a way to add extra information to the parameter for use by external applications. For example, the designer can use properties to include the description of the parameter in different languages or perhaps add instructions about the format of the input prompt.

8.2 Default Parameters

JasperReports provides some built-in parameters (internal to the reporting engine). You can read the built-in parameters, but you can't modify or delete them. Some important built-in parameters are:

- `REPORT_CONNECTION` - For a report using JDBC, this parameter holds the JDBC connection used to run the SQL query.
- `REPORT_DATA_SOURCE` - This parameter contains the data source used to fill the report (if available)
- `REPORT_LOCALE` - This parameter contains the Locale used to fill the report.

Some built-in parameters are specific to a query language. For example if you're using the Hibernate query language, the reports automatically includes the `HIBERNATE_SESSION` parameter that holds the Hibernate session for the HQL query.

The built-in parameters are:

Table 8-1 JasperReports Default Parameters

Parameter	Description
`REPORT_CONTEXT`	
`REPORT_PARAMETERS_MAP`	This is the `java.util.Map` passed to the `fillReport` method; it contains the parameter values defined by the user.
`JASPER_REPORT`	
`REPORT_CONNECTION`	This is the JDBC connection passed to the report when the report is created through an SQL query.
`REPORT_MAX_COUNT`	This is limits the number of records filling a report. If no value is provided, no limit is set.
`REPORT_DATA_SOURCE`	This is the data source used by the report when it's not using a JDBC connection.
`REPORT_SCRIPTLET`	This represents the scriptlet instance used during report creation. If no scriptlet is specified, this parameter uses an instance of `net.sf.jasperreports.engine.JRDefaultScriptlet`.
`REPORT_LOCALE`	This is specifies the locale used to fill the report. If no locale is provided, the system default is used.
`REPORT_RESOURCE_BUNDLE`	This is the resource bundle loaded for this report.

Parameter	Description
REPORT_TIME_ZONE	This is used to set the time zone used to fill the report. If no value is provided, the system default is used.
REPORT_FORMAT_FACTORY	This is an instance of a net.sf.jasperreports.engine.util.FormatFactory. The user can replace the default one and specify a custom version using a parameter. Another way to use a particular format factory is by setting the report property format factory class.
REPORT_CLASS_LOADER	This parameter can be used to set the class loader to use when filling the report.
REPORT_URL_HANDLER_FACTORY	Class used to create URL handlers. If specified, it replaces the default.
REPORT_FILE_RESOLVER	This is an instance of net.sf.jasperreports.engine.util.FileResolver used to resolve resource locations that can be passed to the report in order to replace the default implementation.
REPORT_TEMPLATES	This is an optional collection of styles (JRTemplate) that can be used in addition to those defined in the report.
SORT_FIELDS	
FILTER	
REPORT_VIRTUALIZER	This defines the class for the report filler that implements the JRVirtualizer interface for filling the report.
IS_IGNORE_PAGINATION	You can switch the pagination system on and off with this parameter (it must be a Boolean object). By default, pagination is used except when exporting to HTML and Excel formats.

8.3 Using Parameters in Queries

Generally, you can use parameters in a report query whether or not the language supports them.

JasperReports executes queries, passing the value of each parameter used in the query to the statement.

This approach has a major advantage with respect to concatenating the parameter value to the query string—you don't have to take care of special characters or sanitize your parameter. The database can do it for you. At the same time, this method limits your control of the query structure. For example, you can't specify a portion of a query with a parameter.

8.3.1 Using Parameters in a SQL Query

You can use parameters in SQL queries to filter records in a where condition or to add/replace pieces of raw SQL or even to pass the entire SQL string to execute.

In the first case the parameters act as standard SQL parameters. For example:

```
SELECT * FROM ORDERS WHERE ORDER_ID = $P{my_order_id}
```

In this example the `my_order_id` parameter contains the ID of the order to be read. This parameter can be passed to the report from the application running it to select only a specific order. Please note that the parameter here is a valid SQL parameter, meaning that the query can be executed using a prepared statement like:

```
SELECT * FROM ORDERS WHERE ORDER_ID = ?
```

and the value of the parameter `my_order_id` is then passed to the statement.

In this query:

```
SELECT * FROM ORDERS ORDER BY $P!{my_order_field}
```

`my_order_field` cannot be treated as an SQL parameter. JasperReports considers this parameter a placeholder (note the special syntax `$P!{})` is replaced with the text value of the parameter.

Using the same logic, a query can be fully passed using a parameter. The query string would look like:

```
$P!{my_query}
```

A query can contain any number of parameters. When passing a value using the `$P!{}` syntax, the value of the parameter is taken as is, the user is responsible of the correctness of the passed value (SQL escaping is not performed by JasperReports in this case). When using a parameter in a query, a default value must be set for the parameter to allow Jaspersoft Studio to execute the query to retrieve the available fields.

8.3.2 Using Parameters with Null Values

The parameter form `$P{parametername}` doesn't work correctly with null values. In an operation in which your value could be null, use the form `$X{EQUAL,fieldname,parametername}`.

For example:

1. `$P{param}: "select * where num_column > $P{num_param}"`

   In this case $P should be used, because we don't have $X{GREATER,..}, and Null has no meaning for the operation “greater than”.

2. `$X{EQUAL, column_name, param_name}`

   Let's compare two expressions:

   `"select * where num_column = $P{num_param}"`

   and

   `"select * where $X{EQUAL, num_column, num_param}"`

   Both generate the same output if parameter value is not Null: `"select * where num_column = 1"`

   However, if the parameter has a Null value the output is different:

   - `$P: "select * where num_column = null"`
   - `$X: "select * where num_column IS null"`

   Databases don't understand the key difference `"= null"`, but `"is null"`. So if you want your query with the condition "=" to work with null values, you need to use `$X{EQUAL/NOTEQUAL, column, parameter}`.

8.3.3 IN and NOTIN Clauses

JasperReports provides a special syntax to use with a `where` condition: the `IN` and `NOTIN` clauses.

The `IN` clause checks whether a particular value is present in a discrete set of values. Here is an example:

```
SELECT * FROM ORDERS WHERE SHIPCOUNTRY IS IN ('USA','Italy','Germany')
```

The set here is defined by the countries USA, Italy and Germany. Assuming we are passing the set of countries in a list (or better a `java.util.Collection`) or in an array, the syntax to make the previous query dynamic in reference to the set of countries is:

```
SELECT * FROM ORDERS WHERE $X{IN, SHIPCOUNTRY, myCountries}
```

where `myCountries` is the name of the parameter that contains the set of country names. The `$X{}` clause recognizes three parameters:

- Type of function to apply (`IN` or `NOTIN`)
- Field name to be evaluated (`SHIPCOUNTRY`)
- Parameter name (`myCountries`)

JasperReports handles special characters in each value. If the parameter is `null` or contains an empty list, meaning no value has been set for the parameter, the entire `$X{}` clause is evaluated as the always true statement "`0 = 0`".

8.3.4 Relative Dates

You can create a report that filters information based on a date range relative to the current system date using a parameter of type `DateRange`. A date range parameter can take either a date or a text expression that specifies a date range relative to the current system date.

A relative date expression is always calculated in the time zone of the logged-in user. However, the start day of the week can be configured independent of locale.

8.3.4.1 Relative Date Keywords

The text expression for the relative date must be in the format `<Keyword>+/-<N>` where:

- `<Keyword>` – Specifies the time span you want to use. Options include: `DAY`, `WEEK`, `MONTH`, `QUARTER`, `SEMI`, and `YEAR`.
- `<+/->` – Specifies whether the time span occurs before (-) or after (+) the chosen date.
- `<N>` – Specifies the number of the above-mentioned time spans you want to include in the filter.

For example, if you want to look at Sales for the prior month, your expression would be `MONTH - 1`.

Relative dates don't currently support keywords like "Week-To-Date" (from the start of the current week to the end of the current day). However, you can set a relative date period in a query in JRXML using `BETWEEN`, which has the syntax:

```
$X{BETWEEN, column, startParam, endParam}
```

For example, to create a week-to-date query, set `startParam` to `WEEK` and `endParam` to `DAY`. You can do this for other time ranges, such as Year-To-Day, Year-To-Week, and so forth.

8.3.4.2 Creating a Date Range Parameter

The `class` attribute of a JasperReports date range parameter must have one of the following values:

- `net.sf.jasperreports.types.date.DateRange` (Date only) – Accepts text strings with relative date keywords as described above and date strings in YYYY-MM-DD format. For example:
  `<parameter name="myParameter" class="net.sf.jasperreports.types.date.DateRange">`
- `net.sf.jasperreports.types.date.TimestampRange` (Date and Time) – Accepts text strings with relative date keywords as described above and date strings in YYYY-MM-DD HH:mm:ss format. For example:
  `<parameter name="myParam" class="net.sf.jasperreports.types.date.TimestampRange">`

8.3.4.3 Using Date Ranges in Queries

You must use `$X{}` functions with date ranges, because `$P{}` does not support the date-range types (`DateRange` and `TimestampRange`).

To use date ranges, create a parameter with type date range and use it as the third argument in the $X{} function. To set the default value expression of a date range parameter, use the `DateRangeBuilder()` class to cast the expression to the correct type:

- `new net.sf.jasperreports.types.date.DateRangeBuilder("DAY-1").toDateRange()` – casts a keyword text string to a `DateRange`.
- `new net.sf.jasperreports.types.date.DateRangeBuilder("WEEK").set(Timestamp.class).toDateRange()` – casts a keyword text string to a `TimestampRange`.
- `new net.sf.jasperreports.types.date.DateRangeBuilder("2012-08-01").toDateRange()`– casts a date in YYYY-MM-DD format to a `DateRange`.
- `new net.sf.jasperreports.types.date.DateRangeBuilder("2012-08-01 12:34:56").toDateRange()`– casts a date in YYYY-MM-DD HH:mm:ss format to a `TimestampRange`.

The following JRXML example shows data from the previous day:

```
<parameter name="myParameter" class="net.sf.jasperreports.types.date.DateRange">
  <defaultValueExpression>
    <![CDATA[new DateRangeBuilder("DAY-1").toDateRange()]]>
  </defaultValueExpression>
</parameter>
<queryString>
  <![CDATA[Select * from account where $X{EQUAL, OpportunityCloseDate, myParameter}]]>
</queryString>
```

This JRXML example shows results prior to the end of last month:

```
<parameter class="net.sf.jasperreports.types.date.DateRange" name="EndDate">
  <defaultValueExpression>
    <![CDATA[new net.sf.jasperreports.types.date.DateRangeBuilder("MONTH-1").toDateRange().getEnd()]]>
    </defaultValueExpression>
</parameter>
<queryString>
  <![CDATA[SELECT * FROM orders WHERE $X{LESS, order_date, EndDate}]]>
</queryString>
```

The following table shows two additional examples of relative dates.

Problem	Solution
Set up a relative date parameter called `StartDate` that takes the value: `QUARTER`. `QUARTER` evaluates to the first day (the first instant, really) of this quarter.	
Find all purchases made previous to this quarter	`SQL: select * from orders where $X{LESS, order_date, StartDate}`
Find all purchases made in this quarter	`select * from orders where $X{EQUAL, order_date, StartDate}`

8.3.4.4 Using Relative Dates in Input Controls

When you create an input control for a `DateRange` or `TimestampRange` parameter, the user can either type a relative date expression or enter a specific date (either by typing or by using the calendar widget).

Use `BETWEEN` to set up input controls that allow the user to specify a range (other than a day) using either a relative date expression or actual dates. To do this:

- Define two date range parameters, for example, `StartDate` and `EndDate`.
- Optionally, set default values for one or both parameters using `defaultValueExpression`.
- Use a `$X{}` expression with a `BETWEEN` function in your query.
- Create a date type input control for each parameter, for example, `StartDate` and `EndDate`.

The following JRXML example uses the `BETWEEN` keyword in the `$X()` function to find all data from the previous 20 years:

```
<parameter name="StartDate" class="net.sf.jasperreports.types.date.DateRange">
  <defaultValueExpression>
    <![CDATA[(new net.sf.jasperreports.types.date.DateRangeBuilder("YEAR-20")).toDateRange()]]>
  </defaultValueExpression>
</parameter>
<parameter name="EndDate" class="net.sf.jasperreports.types.date.DateRange">
  <defaultValueExpression>
    <![CDATA[(new net.sf.jasperreports.types.date.DateRangeBuilder("DAY")).toDateRange()]]>
  </defaultValueExpression>
</parameter>
<queryString language="SQL">
  <![CDATA[select HIRE_DATE, MANAGEMENT_ROLE, GENDER, SUPERVISOR_ID,SALARY from employee where
        $X{BETWEEN, HIRE_DATE, StartDate, EndDate} limit 200]]>
</queryString>
```

You can use the `getStart()` and `getEnd()` methods to get the precise beginning and end of a relative date. Both of these methods return a date instead of a date range. The following example shows how to get the precise start date as a default value expression:

```
<parameter name="StartDate" class="java.util.Date" nestedType="java.util.Date">
  <defaultValueExpression><![CDATA[$P{UserPeriod}.getStart()]]></defaultValueExpression>
</parameter>
```

8.3.4.5 Publishing Reports with Relative Dates to JasperReports Server

Jaspersoft Studio automatically enables support for date range expressions on connections to JasperReports Server 5.0 and higher. To verify that date range expressions are enabled:

1. Right-click on the server connection in the Repository and select Edit.

2. In the Server profile wizard, display the Advanced settings and select **Supports DateRange Expressions**.

When **Supports DateRange Expressions** is enabled, input controls for date range parameters work correctly when published to JasperReports Server.

8.3.5 Passing Parameters from a Program

Jaspersoft Studio passes parameters from a program "caller" to the print generator using a class that extends the `java.util.Map` interface. For example:

```
...
        HashMap hm = new HashMap();
                ...
        JasperPrint print = JasperFillManager.fillReport(
                fileName,
                hm,
        new JREmptyDataSource());
...
```

`fillReport` is a key method that allows you to create a report instance by specifying the file name as a parameter, a parameter map, and a data source. (This example uses a dummy data source created with the class `JREmptyDataSource` and an empty parameter map created using a `java.util.HashMap` object.)

Let's see how to pass a simple parameter to a reporting order to specify the title of a report.

The first step is to create a parameter in the report to host the title (that is a String). We can name this parameter `REPORT_TITLE` and the class is `java.lang.String` (**Figure 8-3**).

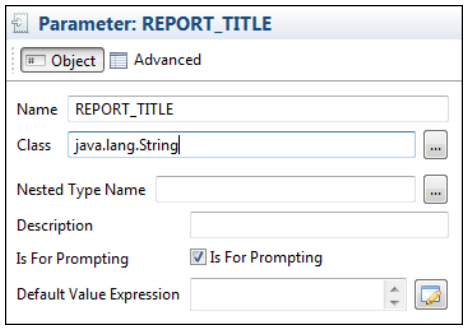


Figure 8-3 Definition of `REPORT_TITLE`

All the other properties can be left as they are. Drag the parameter into the Title band to create a text field to display the `REPORT_TITLE` parameter.

Figure 8-4 Design Panel with `REPORT_TITLE` in the Title Band

To set the value of the `REPORT_TITLE` parameter in our application, modify the code of the previous source code example by adding:

```
 ...
HashMap hm = new HashMap();
hm.put("REPORT_TITLE","This is the title of the report");
...
JasperPrint print = JasperFillManager.fillReport(
fileName,
hm,
new JREmptyDataSource());
...
```

We have included a value for the `REPORT_TITLE` parameter in the parameter map. You don't need to pass values for all the parameters. If you don't provide a value for a certain parameter, JasperReports assigns the value of `Default Value Expression` to the parameter with the empty expression evaluated as null.

When printing the report, Jaspersoft Studio includes the String `This is the title of the report` in the Title band. In this case we just used a simple String. But you can pass much more complex objects as parameters, such as an image (`java.awt.Image`) or a data source instance configured to provide a specified subreport with data. The most important thing to remember is that the object passed in the map as the value for a certain parameter must have the same type (or at least be a super class) of the type of the parameter in the report. Otherwise, Jaspersoft Studio fails to generate the report and returns a `ClassCastException` error.

8.4 Parameters Prompt

If you set a parameter to be used as a prompt, when executing the report, Jaspersoft Studio asks for the value of the parameter.

To create a parameter prompt:

Create a simple report with the template **Blank A4**, name `ParameterExample` and data adapter **One Empty Record - Empty Rows**.

1. In this report create a parameter and rename it (from its properties tab) to `MESSAGE`, with type `java.lang.String`, and select the **Is For Prompting**) checkbox.
2. Drag the parameter from the outline view into the **Title** band. Jaspersoft Studio creates a text field to display the parameter value. You should have something like the following image.

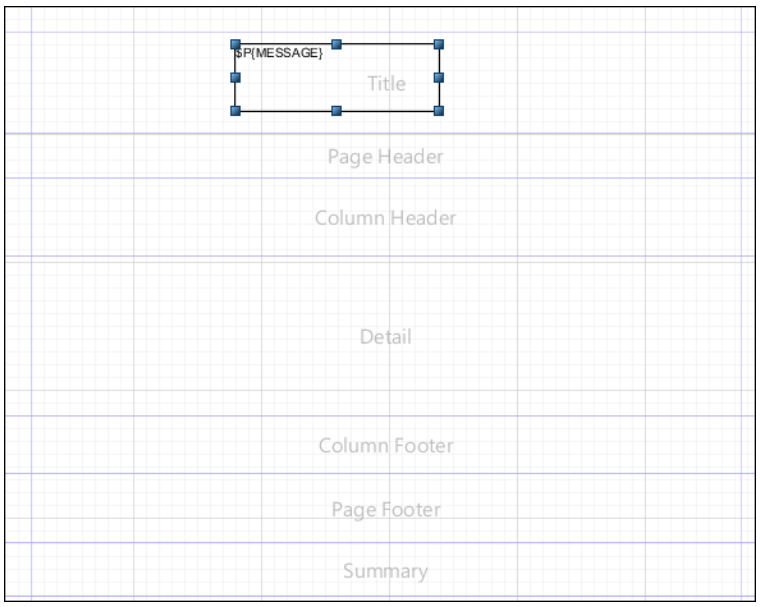


Figure 8-5 Parameter in Title Band

To compile and preview the report:

1. Click the **Preview** tab.

   Be sure the **Input Parameter** window is open. If not, click the gray right-arrow to the left of the **Preview** screen.
2. Add a value for the `MESSAGE` parameter. For this example, type `Parameter Example`.
3. Press the **Play** button. The message is printed in the title band.

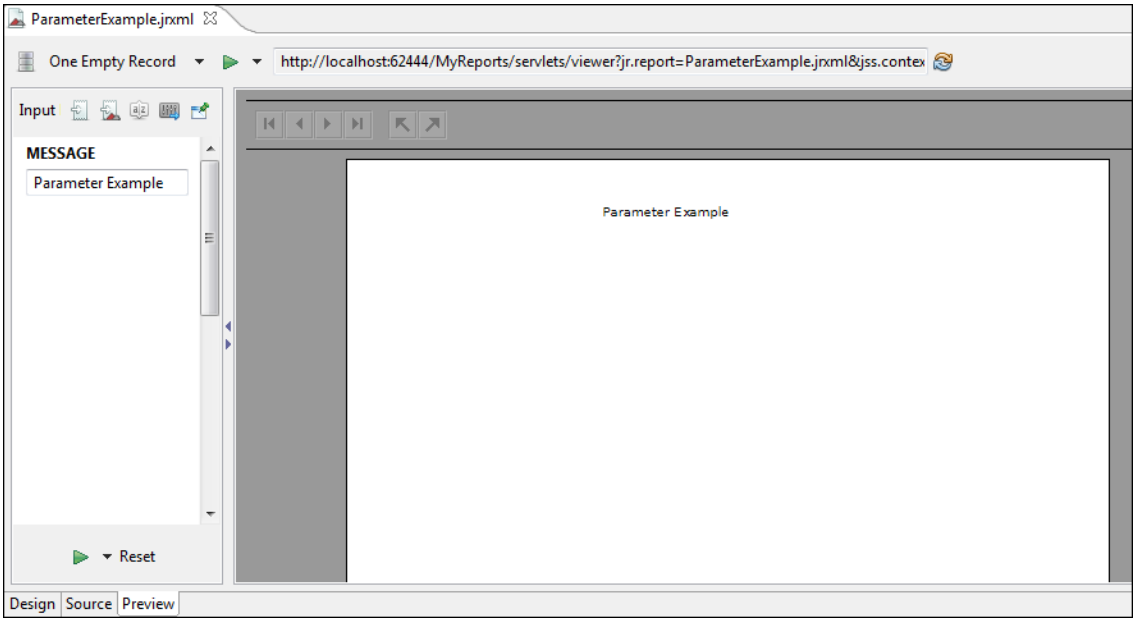


Figure 8-6 Preview Tab with Parameter Value

Jaspersoft Studio provides input dialogs for parameters of type String, Date, Time, Number, and Collection.

CHAPTER 9 VARIABLES

You can use variables to store partial results and do complex calculations with the data extracted from a data source. You can then use these values in other parts of the report, including other variables.

This chapter contains the following sections:

9.1 Defining or Editing a Variable

As with many other elements, all defined variables are visible in the Outline view, where you can create a new variable and edit its properties in the Properties view.

To define a new variable:

1. In the Outline view, right click the **Variables** item and select **Create Variable**. A new variable is added to the list of variables.
2. Click to select the new variable. The Properties view shows information about the new variable.
3. In the Properties view, click the **Object tab**.
4. Update the variable properties. See **"Base Properties of a Variable" on page 127** for information on these options.

Jaspersoft Studio has some built-in variables that are present in every report. These variables can't be edited. You'll notice their names are light gray; other variables are black. For more information see **"Built-In Variables" on page 130**

9.2 Base Properties of a Variable

At a minimum, all variables must have the following defined:

- **Name**: A string used to refer to a variable. It is necessary to use this variable inside other expressions like the valorization of a Text Field or the computation of another variable). Use the following syntax to refer to a variable: `$V{variable_name}`.

- **Type**: Necessary because a variable is an object that is probably used in other expressions, so its type must be known to be manipulated correctly.
- **Expression**: The function used to define the variable value, it can be composed of more fields and variables, which could be logic operators, math operators and so on. Jaspersoft Studio provides an expression editor. To open it, click the button to the right of the expression text field. The expression is evaluated on each iteration (every time a record is read from the data source). If no calculation function is defined, the result of the expression is assigned to the variable. So it's important that the result has a type compatible with the one in the variable.
- **Initial Value**: The value assumed from the variable at the beginning, before the first computation of its expression. The initial value is an expression itself, so it can be defined through the expression editor.
- It's not mandatory, but it's good practice to define an initial value. For example, if you have a variable called `variable1` with the expression `new Integer(5)`, at every iteration, the variable is assigned the integer value 5. In this context the initial value isn't important. But if you change the expression to `$V{variable1}+5`, at every iteration the variable is incremented by 5. In this case, an initial value is necessary because if the `variable1` is undefined at the first iteration, all future evaluations will break.

9.3 Other Properties of a Variable

The most complex property of a variable is its temporal value. Because its expression is evaluated at every iteration, it's important to understand which value has a variable, and at which time. This can be complicated, considering that a variable can use other variables inside its expression. For these reasons there are mechanisms that can be used to simplify the evaluation or reading of the variable value during iterations.

9.3.1 Evaluation Time

Evaluation time is not an attribute of the variable but of the elements that can use the variable in their expressions (like a Text Field). Evaluation time determines when the value of the variable should be read. A variable can potentially change value at every iteration, so a value read at one time may be different from the value read another time.

For every element using a variable in its expression, it's possible to say when to evaluate the variable. And because an expression can contain multiple variables, this parameter also influences when these variables are read.

The possible evaluation times are:

- **Report**: The expression is evaluated ad the end of the report.
- **Page**: The expression is evaluated at the end of every page of the report.
- **Column**: The expression is evaluated at the end of each column (for a single column report, this is the same as Page).
- **Group**: The expression is evaluated after the break of the specified group (available only if at least one group is defined).
- **Band**: The expression is evaluated after the end of the band where the element with this evaluation time is placed.
- This is a very specific case, introduced to wait until the other elements in the band are completely created. Typically the value of the variables are read at the start of the band. But suppose, for example, you have a subreport with an output parameter to print in the main report. To print this parameter it must be read after the subreport is computed, so the value can be printed when the band is completely created. In this case the Band evaluation time is necessary.

- **Auto**: This is used when the expression contains variables and fields that need to be evaluated at different times. The variables are evaluated at a time corresponding to their Reset Type (see below for more information), instead the fields are always evaluated at time -now. This type is useful when report elements have expressions that combine values evaluated at different times (for example, percentage out of a total).
- **Now**: The value of the expression is evaluated after the read of every record, so at every iteration, this is the default behavior.

9.3.2 Calculation Function

A calculation function is an attribute that specifies when a variable can be used in association with the expression to determine the value of the variable. When using a calculation function, the value of the variable is not determined directly by its expression. Instead, it's passed to the calculation function that uses it to determine its value.

There are many calculation functions built-in to Jaspersoft Studio:

- **Sum**: At every iteration the variable value is summed. This is one of the cases where the initial value is really important.
- **Count**: At every iteration the variable value is incremented by one unit (this is only if the expression isn't null).
- **Distinct Count:** At every iteration the variable value is incremented by one unit, but only if the value of the expression was never returned before.
- **Average**: The value of the variable is the arithmetic average of all values received in input from the expression.
- **Lowest**: The variable takes the value of the lowest element received from the expression.
- **Highest**: The variable takes the value of the highest element received from the expression.
- **Standard Deviation**: The standard deviation of all the values received from the expression.
- **First**: The variable takes the value from the first value returned by the expression.
- **System**: No calculation is done and the expression is not evaluated, the value of the variable is the last value set on it. This is useful to store partial results or the final result of a computation.

9.3.3 Increment Type

As stated above, when a calculation function is defined, the value of the expression passed to the function that calculates the variable. By default this occurs for every record read, but sometimes a different behavior is desired. The increment type parameter enables you to change the "time" at which the calculation function is used.

The possible values for this attribute are:

- **Report**: The Calculation Function is called only at the end of the report, passing to it the expression's value at that moment.
- **Page**: The Calculation Function is called at the end of each page, passing to it expression's value at each of those moments.
- **Column**: The Calculation Function is called at the end of each column (for a one-column report, this is the same as Page).
- **Group**: The Calculation Function is called at the start of every occurrence of the specified group. This option is visible only if at least one group is defined.
- **None**: The Calculation Function is called after the read of every record, this is the default behavior.

Remember the expression is evaluated at every record read, independent of the increment type selected, but the calculation function is used only when the times match those defined in the increment type.

9.3.4 Reset Type

The reset type specifies when a variable should be reset to its initial value (or to null if no initial value is defined). This is useful when the variable is used to compute a partial value, like a sum or an average of only some of the records read.

The possible values for this attribute are:

- **Report**: The variable is initialized only one time at the beginning of the report creation.
- **Page**: The variable is initialized on each page.
- **Column**: The variable is initialized again in each new column (for a one-column report, this is the same as Page).
- **Group**: The variable is initialized at the start of every occurrence of the specified group. This option is available only if at least one group is defined.
- **None**: The variable is never initialized, so the initial value expression is ignored.

9.3.5 Incrementer Factory Class Name

The calculation functions are useful, but limited to numeric types. You may have a case where something more specific is needed. Suppose you have a String type field and you want to concatenate the value read. You can do this by defining a new Incrementer. An incrementer is a piece of java code that extends the JRIncrementerFactory interface, and can build a personalized calculation function to do what you need. Every calculation function receives the expression value and the variable value and returns the result of the increment, so there is everything needed to do the calculation and return the right value.

9.4 Built-In Variables

Jaspersoft Studio makes some built-in variables available to you. See the table below. These variables can't be edited. You'll notice their names are light gray; other variables are black.

Variable Name	Description
PAGE_NUMBER	Contains the current number of pages in the report at report time.
COLUMN_NUMBER	Contains the current number of columns.
REPORT_COUNT	Contains the number of records processed.
PAGE_COUNT	Contains the current number of records processed in the current page.
COLUMN_COUNT	Contains the current number of records processed during the current column creation.

9.5 Tips & Tricks

Pay attention to the variable type. For example if your expression returns a number but the variable type is string (the default type) then its value is always zero.

The form of the expression is very important for the computation of a value, especially when the variable itself is used in the expression. Consider the following example:

A field with name `Money_Gained` with an integer value, which could be null, is read from the data source.

A variable `Total1` with the expression:
`IF(EQUALS($F{Money_Gained}, null), $V{Total1}, $V{Total1}+$F{Money_Gained})`
initial value zero, and no calculation function;

A variable `Total2` with the expression
`$V{Money_Gained} == null ? $V{Total2} : $V{Total2}+$F{Money_Gained}`
initial value zero, and no calculation function;

The two expressions seem equivalent. They both add the money gained to the variable when it's not null (remember that if there's no calculation function, the value of the expression is assigned to the variable). The check if the `Money_Gained` has a null value is necessary because the sum of a number with the value null is null. So adding null to Total1 or Total2 changes the variable to null. But even with this check when `Money_Gained` becomes null for the first time even `Total1` is null, instead `Total2` has the correct value.

This happens because these two expressions have different interpreters, the first is interpreted by Groovy, the second by Java. The Java behavior evaluates the condition and then selects the correct branch. The Groovy behavior computes the two branches, then evaluates the expression, and finally returns the correct branch. Doing this adds the null value to `Total1` before doing the check, and makes `Total1` null. To avoid this, use the variable in the main branch only, for example `Total1` could be rewritten as:
`$V{Total1} + IF(EQUALS($F{Money_Gained}, null),0,F{Money_Gained}).`

The syntax is still interpreted by Groovy, but now the variable is out of the `IF` branches, so even if they are both evaluated, the variable maintains its value.

CHAPTER 10 DATA SOURCES

JasperReports can fill a report with data in a number of ways. For example, you can put an SQL query directly inside a report and provide a connection to a database against which to execute the query and read the resulting record set, or you can use more sophisticated custom technology to provide a table-like set of values.

Jaspersoft Studio provides direct support for a rich set of query languages, including SQL, HQL, EJBQL, and MDX, and supports other languages like XPath (XML Path Language). If you don't want to use a query language or put your query inside your report, you can use a JasperReports data adapter. Basically, a JasperReports data adapter is an object that iterates on a record set that's organized like a simple table.

All the data adapters implement the `JRDataSource` interface. JasperReports provides many ready-to-use implementations of data sources to wrap generic data structures, such as arrays or collections of JavaBeans, result sets, table models, CSV and XML files. This chapter documents some of these data sources.

Jaspersoft Studio provides support for all these things: you can define JDBC connections to execute SQL queries, set up Hibernate connections using Spring, and test your own `JRDataSource` or your custom query language.

This chapter has the following sections:

10.1 Understanding Data Adapters and Connections in Jaspersoft Studio

In Jaspersoft Studio, a data adapter is an XML file that connects to your data. This information includes, URL, user, password, paths, etc. Data adapters also contain the logic to prepare all parameters for JasperReports to run the query and iterate data. A data adapter itself contains no data, the data is located in the JRDataSource.

Data adapters in Jaspersoft Studio are used to:

- Define the data connection type. The connection is an SQL connection rather than a JasperReports or Jaspersoft Studio object.
- Bind information to connect your data to Jaspersoft Studio.

- Create the connection and necessary parameters for JasperReports library.

Inside Jaspersoft Studio, data adapters can be stored in your Eclipse settings or in files. Files are the preferred method, because they're easier to deploy later.

The most frequently used connection in Jaspersoft Studio is the JDBC data adapter. Other connection types provided by Jaspersoft Studio include:

- JDBC connection
- JavaBean collection data source
- XML data source
- CSV data source
- Hibernate connection
- Spring-loaded Hibernate connection
- Hadoop Hive data source
- JRDataSourceProvider
- Custom data source
- Mondrian OLAP connection
- XML/A connection
- EJBQL connection
- Empty data source

All the connections are opened and passed directly to JasperReports during report generation. For many connections, JasperReports provides one or more built-in parameters that can be used inside the report for several purposes (for example, to fill a subreport that needs the same connection as the parent).

- The XML data source allows you to take data from an XML document.
- A CSV data adapter allows you to open a CSV (comma-separated values) file for use in a report.
- The JavaBean set data source, custom data adapter, and JRDataSourceProvider allow you to print data using purposely written Java classes.
- The Hibernate connection provides the environment to execute HQL (Hibernate Query Language) queries. This connection can be configured using Spring, as well.
- EJBQL (Enterprise JavaBean Query Language) queries can be used with an EJB connection.
- MDX queries can be used with either type of OLAP client connection (native direct connection to Mondrian servers or standard XML/A providers (such as JasperReports Server or Microsoft Analysis Services) to interrogate OLAP databases.

An empty data adapter is something like a generator of records having zero fields. Use this for testing or to meet specific needs, such as static content reports or subreports.

You manage connections and data adapters through the **DataAdapter Wizard**. To set up a new data source, right-click **Data Adapters** in the Repository view and choose **Create Data Adapter**.

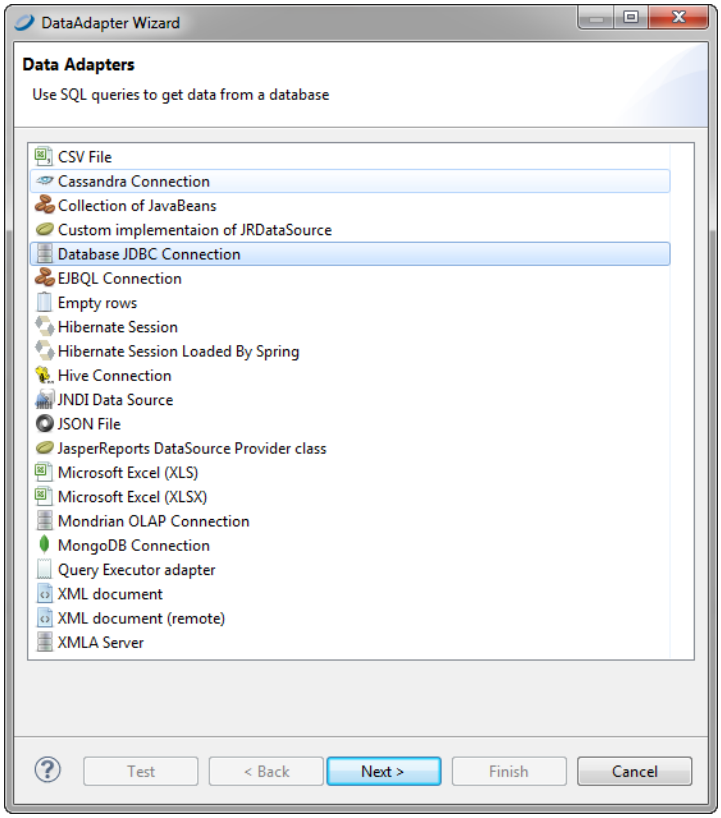


Figure 10-1 DataAdapter Wizard

A connection and a data adapter are different objects. However, this document uses the two terms interchangeably because their functions are so similar.

You can set the active data adapter in several ways. The easiest and most intuitive way is to select the data adapter from the Repository view.

If no data adapter is selected, the report can't be filled, so by default Jaspersoft Studio uses a pre-configured empty data adapter. Data adapters can also be used in conjunction with the Report Wizard. But it's always better to start your report with the data you want to fill it. That's why configuring the connection to your data is usually the first step when starting with Jaspersoft Studio.

10.2 Creating and Using Database JDBC Connections

A JDBC connection allows you to use a relational DBMS (or, in general, whatever databases are accessible through a JDBC driver) as a data source. To set a new JDBC connection, click the **New** button in the Connections/Datasources dialog (shown earlier in **Figure 10-1**) to open the interface for creation of a new connection (or data source). From the list, select **Database JDBC connection** to open the Data Adapter dialog.

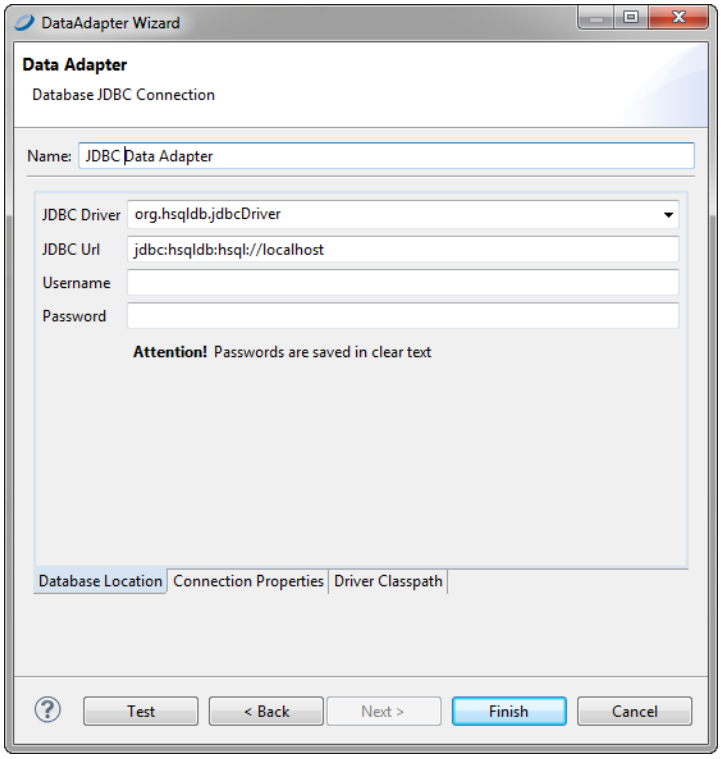


Figure 10-2 Configuring a JDBC Connection

First, name the connection (use a significant name like `Mysql - Test`). Jaspersoft Studio will use this name to refer to this connection.

In the **JDBC Driver** field, specify the JDBC driver to connect to the database. The combo box proposes the names of the most common JDBC drivers .

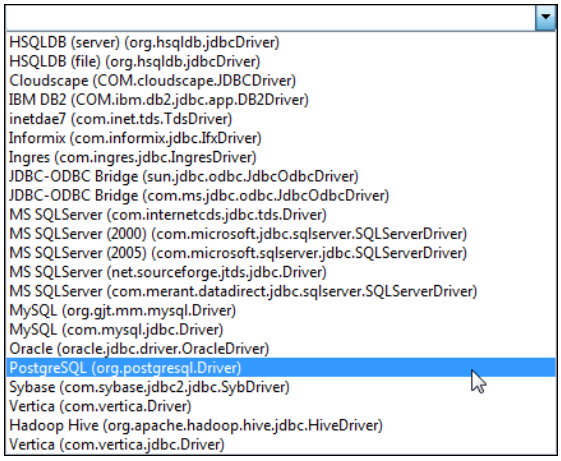


Figure 10-3 JDBC Drivers List

If a driver is displayed in red, the JDBC driver class for that driver is not present in the classpath and you must obtain and install the driver before using it. See **10.3, "Working with Your JDBC Connection," on page 138**.

Thanks to the JDBC URL Wizard, you can automatically construct the JDBC URL to use the connection to the database by inserting the server name and the database name in the correct text fields. Click the **Wizard** button to create the URL.

Enter a username and password to access the database. A check box enables you to save the password.

Jaspersoft Studio saves passwords in Eclipse secure storage. In some cases, this can cause problems when trying to test and save data adapters. For more information, refer to our Eclipse Secure Storage in Jaspersoft Studio Community wiki page.

If the password is empty, it is better if you specify that it be saved.

After you've inserted all the data, click the **Test** button to verify the connection. If everything's okay, you'll see this message:

Figure 10-4 Test Confirmation Dialog

When tests fails, the most common exceptions are:

- A `ClassNotFoundError` was thrown.
- The URL is not correct.
- Parameters are not correct for the connection (database is not found, the username or password is wrong, etc.).

10.2.1 ClassNotFoundError

The `ClassNotFoundError` exception occurs when the required JDBC driver is not present in the classpath. For example, suppose you want to create a connection to an Oracle database. Jaspersoft Studio has no driver for this database. If you mistakenly choose the `oracle.jdbc.driver.OracleDriver` driver, when you test the connection,you'll see the `ClassNotFoundException`, as shown in **Figure 10-5**.

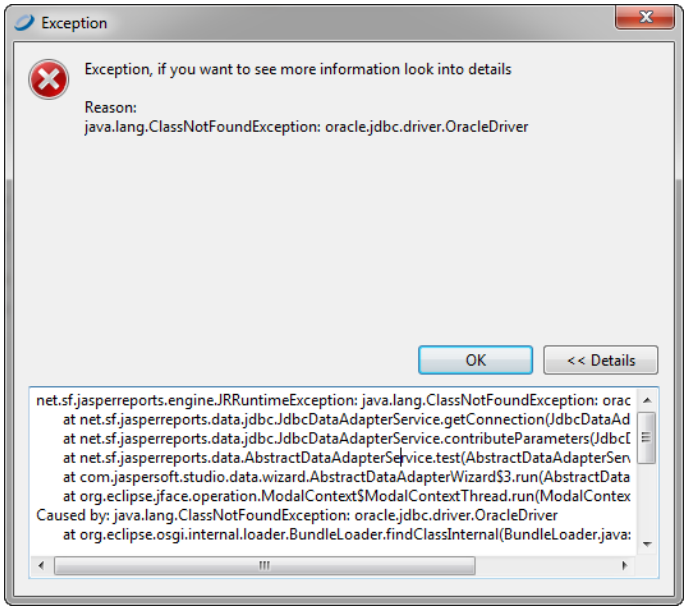


Figure 10-5 `ClassNotFoundError` **exception**

You'll need to add the JDBC driver for Oracle to the classpath. As Jaspersoft Studio uses its own class loader, it's enough add the ojdbc14.jar file to the Jaspersoft Studio classpath, which is defined in your Eclipse project; the same can be done for directories containing classes and other resources. To edit the classpath, click **Project** > **Properties** > **Java Build Path**>**Libraries**, and click **Add JARs**. If you need to use the driver only for this data adapter, you can instead add the driver on the data adapter's Driver Classpath tab.

10.2.2 URL Not Correct

If a wrong URL is specified, you'll get an exception when you click the **Test** button. You can find the exact cause of the error using the stack trace provided in the exception.

In this case, if possible, it is better to use the JDBC URL Wizard to build the JDBC URL and try again.

10.2.3 Parameters Not Correct for the Connection

If you try to establish a connection to a database with the wrong parameters (for example invalid credentials or inaccessible database), the same database returns a message is fairly explicit about the reason behind the failure of the connection.

10.3 Working with Your JDBC Connection

When you create a report with a JDBC connection, you specify an SQL query to extract records from the database. You can also use this connection for a subreport or a personalized lookup function for decoding specific data. For this reason, JasperReports provides a `java.sql.Connection` parameter called `REPORT_CONNECTION`. You can use this parameter in any expression you like, with this parameters syntax:

```
$P{REPORT_CONNECTION}
```

This parameter contains the `java.sql.Connection` class passed to JasperReports from the calling program.

The use of JDBC or SQL connections is the simplest and easiest way to fill a report.

10.3.1 Fields Registration

In order to use SQL query fields in a report, you need to register them. You don't need to register all the selected fields—only those effectively used in the report. For each field, specify name and type. **Table 10-1, "Conversion of SQL and JAVA types ," on page 139** shows SQL types and the Java objects they map to.

Table 10-1 Conversion of SQL and JAVA types

SQL Type	Java Object	SQL Type	Java Object
CHAR	`String`	REAL	`Float`
VARCHAR	`String`	FLOAT	`Double`
LONGVARCHAR	`String`	DOUBLE	`Double`
NUMERIC	`java.math.BigDecimal`	BINARY	`byte[]`
DECIMAL	`java.math.BigDecimal`	VARBINARY	`byte[]`
BIT	`Boolean`	LONGVARBINARY	`byte[]`
TINYINT	`Integer`	DATE	`java.sql.Date`
SMALLINT	`Integer`	TIME	`java.sql.Time`
INTEGER	`Integer`	TIMESTAMP	`java.sql.Timestamp`
BIGINT	`Long`		

The table doesn't include special types like BLOB, CLOB, ARRAY, STRUCT, and REF, because these types cannot be managed automatically by JasperReports. However, you can use them by declaring them generically as `Object` and managing them by writing supporting static methods. The BINARY, VARBINARY, and LONGBINARY types should be dealt with in a similar way. With many databases, BLOB and CLOB can be declared as `java.io.InputStream`.

Whether an SQL type is converted to a Java object depends on the JDBC driver used.

For the automatic registration of SQL query fields, Jaspersoft Studio relies on the type proposed for each field by the driver itself.

10.3.2 Filtering Records

The records retrieved from a data source (or from the execution of a query through a connection) can be ordered and filtered. Set sort and filter options in the Report query dialog by clicking the **Dataset and Query** button  .

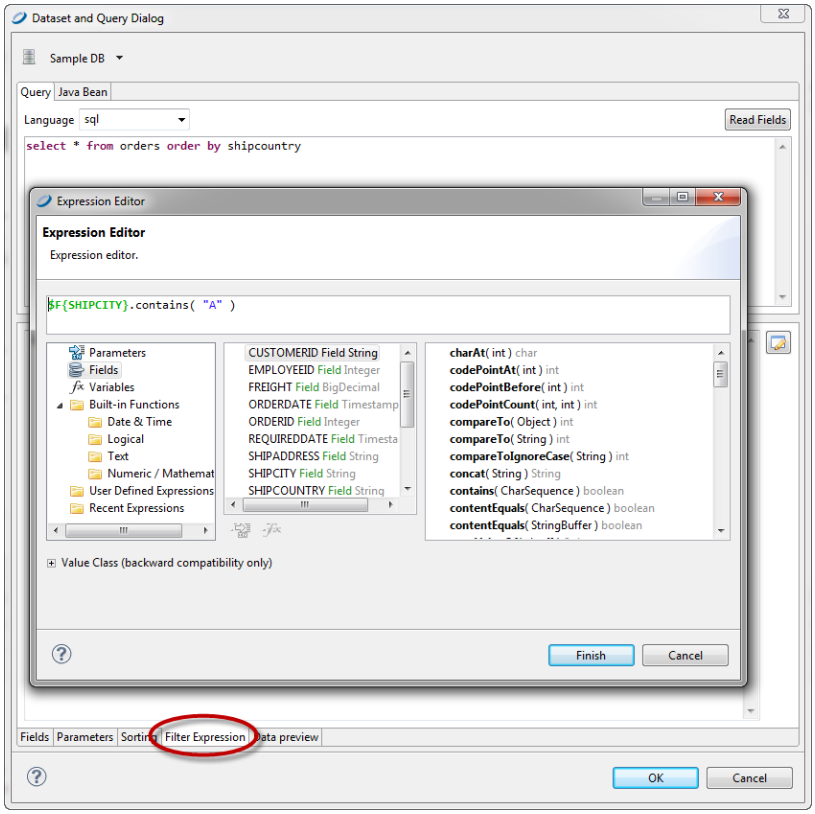


Figure 10-6 Filter Expression Tab and Expression Editor

Clicking the **Data Preview** tab shows your filtered data.The filter expression must return a Boolean object: true if a particular record can be kept, false otherwise.

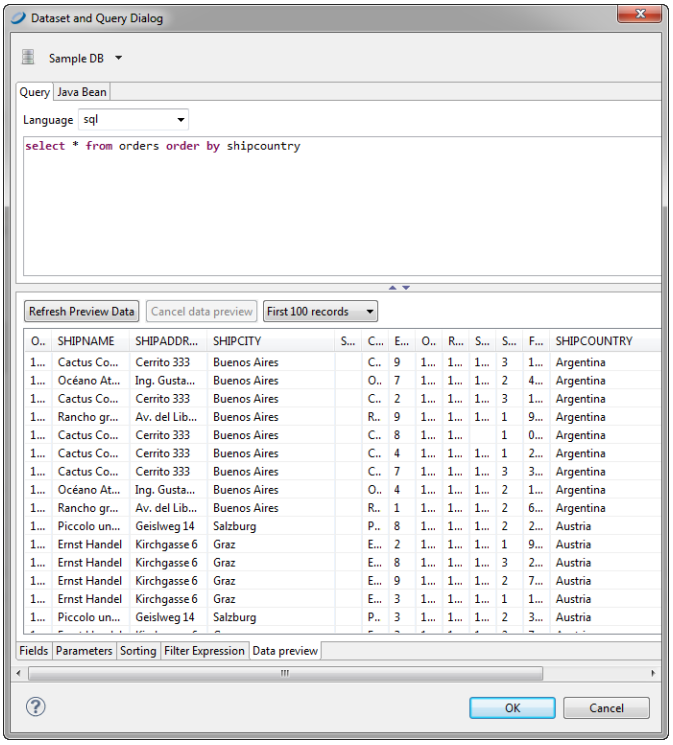


Figure 10-7 Data Preview

If no fields can be selected with the **Add field** button, check to see if the report contains fields. If not, close the query dialog, register the fields, and resume the sorting.

10.4 Understanding the JRDataSource Interface

Before exploring the data sources provided by Jaspersoft Studio, it's important to understand how the `JRDataSource` interface works. Every JRDataSource must implement both these methods:

```
public boolean next()
public Object getFieldValue(JRField jrField)
```

The `public boolean next()` method returns true if the cursor is positioned correctly in the subsequent record, false if no more records are available The `public Object getFieldValue(JRField jrField)` method is useful for moving a virtual cursor to the next record. In fact, data supplied by a JRDataSource is ideally organized into records as in a table. .

Every time JasperReports executes the `public boolean next()` method, all the fields declared in the report are filled and all the expressions (starting from those associated with the variables) are calculated again. Subsequently, JasperReports determines whether to print the header of a new group, to go to a new page, and so on. When `next` returns false, the report is ended by printing all final bands (Group Footer, Column Footer, Last Page Footer, and Summary). The method can be called as many times as there are records present (or represented) from the data source instance.

The method `public Object getFieldValue(JRField jrField)` is called by JasperReports after a call to `next` results in a true value. In particular, it's executed for every single field declared in the report. In the call, a

`JRField` object is passed as a parameter. It's used to specify the name, the description and the type of the field from which to obtain the value (all this information, depending on the specific data source implementation, can be combined to extract the field value).

The type of the value returned by the `public Object getFieldValue(JRField jrField)` method has to be adequate for that declared in the `JRField` parameter, except when a `null` is returned. If the type of the field was declared as `java.lang.Object`, the method can return an arbitrary type. In this case, if required, a cast can be used in the expressions. A cast is a way to dynamically indicate the type on an object, the syntax of a cast is:

```
(type)object
```

in example:

```
(com.jaspersoft.ireport.examples.beans.PersonBean)$F{my_person}
```

Usually a cast is required when you need to call a method on the object that belongs to a particular class.

10.5 Data Source Types

10.5.1 Using Collection of JavaBeans Data Sources

A JavaBeans set data source allows you to use JavaBeans as data for a report. In this context, a JavaBean is a Java class that exposes its attributes with a series of `get` methods, with the following syntax:

```
public <returnType> getXXX()
```

where <*returnType*> (the return value) is a generic Java class or a primitive type (such as `int`, `double`, and so on).

To create a connection to handle JavaBeans, select **Collection of JavaBeans** in the list of data source types to bring up the dialog shown in **Figure 10-8**.

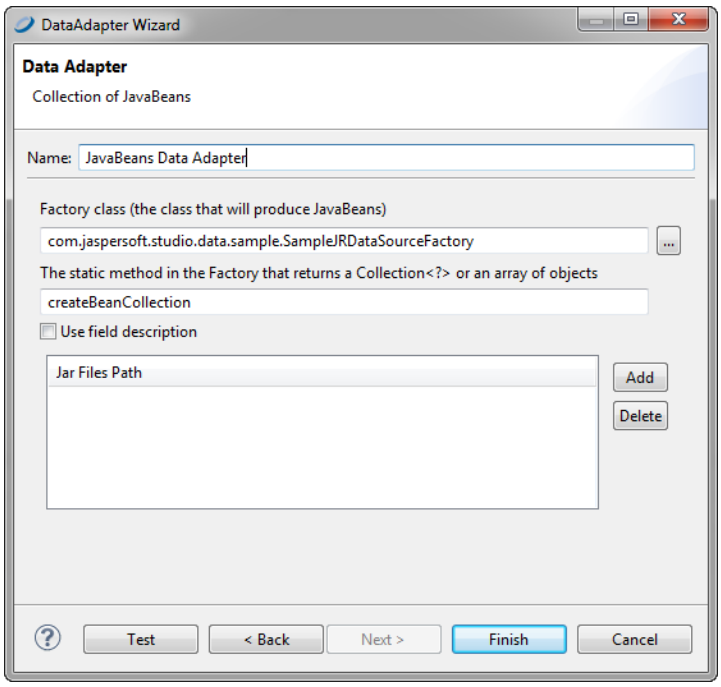


Figure 10-8 JavaBeans set data source

Once again, the first thing to do is specify the name of the new data source.

10.5.2 Registering the Fields of a JavaBean Set Data Source

One peculiarity of a JavaBeans set data source is that the fields are exposed through `get` methods. This means that if the JavaBean has a `getXyz()` method, `xyz` is the name of a record field (the JavaBean represent the record).

In this example, the `PersonBean` object shows two fields: name and age. Register them in the fields list as a `String` and an `Integer`, respectively.

Create a new empty report and add the two fields by right-clicking the **Fields** node in the outline view and selecting **Add field**. The field names and the types of the fields are: name (`java.lang.String`) and age (`java.lang.Integer`).

Drag the fields into the **Detail** band and run the report. (Make sure the active connection is the Test Factory.) To refer to an attribute of an attribute, you can separate with periods. For example, to access the `street` attribute of an `Address` class contained in the PersonBean, the syntax would be `address.street`. The real call would be `<someBean>.getAddress().getStreet()`.

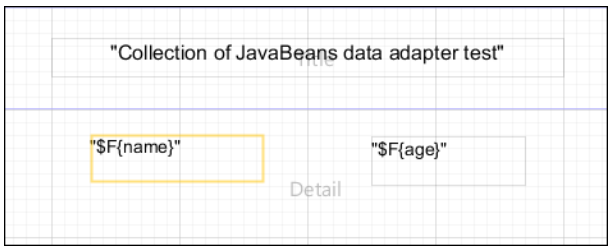


Figure 10-9 Layout of a JavaBeans-Based Report

If the flag `Use field description` is set when you're specifying the properties of your JavaBeans set data source, mapping between JavaBean attribute and field value uses the field description instead of the field name. The data source considers only the description to determine the field value, and the field can have any name.

Jaspersoft Studio provides a visual tool to map JavaBean attributes to report fields. To use it, open the query window, go to the tab **JavaBean Data Source**, insert the full class name of the bean you want to explore, and click **Read attributes**. The tab displays the attributes of the specified bean class.

- If an attribute is also a Java object, you can double-click the object to display its other attributes.
- To map a field, select an attribute name and click the **Add Selected Field(s)** button.

10.5.3 Using XML Data Adapters

JasperReports supports XML as a data adapter in three different ways: XML documents, remote XML documents, and XML/A servers.

An XML document is typically organized as a tree, and doesn't match the table-like form required by JasperReports. For this reason, you have to use an XPath expression to define a node set. The specifications of the XPath language are available at http://www.w3.org/TR/xpath. Some examples can help you understand how to define the nodes.

The XML file below is an address book in which people are grouped in categories, followed by a second list of favorite objects. In this case, you can define different node set types. First you'll need to decide how you want to organize the data in your report.

Table 10-2 Example XML file

```
<addressbook>
       <category name="home">
               <person id="1">
                       <lastname>Davolio</lastname>
                       <firstname>Nancy</firstname>
               </person>
               <person id="2">
                       <lastname>Fuller</lastname>
                       <firstname>Andrew</firstname>
               </person>
               <person id="3">
                       <lastname>Leverling</lastname>
               </person>       </category>

        <category name="work">
```

```
                <person id="4">
                        <lastname>Peacock</lastname>
                        <firstname>Margaret</firstname>
                </person>
        </category>
        <favorites>
                <person id="1"/>
                <person id="3"/>
        </favorites>
</addressbook>
```

To select only the people contained in the categories (that is, all the people in the address book), use the following expression:

```
/addressbook/category/person
```

Four nodes are returned as shown.

Table 10-3 Node set with expression `/addressbook/category/person`

```
<person id="1">
        <lastname>Davolio</lastname>
        <firstname>Nancy</firstname>
</person>
<person id="2">
        <lastname>Fuller</lastname>
        <firstname>Andrew</firstname>
</person>
<person id="3">
        <lastname>Leverling</lastname>
</person>
<person id="4">
        <lastname>Peacock</lastname>
        <firstname>Margaret</firstname>
</person>
```

If you want to select the people appearing in the `favorites` node, the expression to use is

```
/addressbook/favorites/person
```

Two nodes are returned.

```
<person id="1"/>
<person id="3"/>
```

Here's another expression. It's a bit more complex, but it shows all the power of the Xpath language. The idea is to select the person nodes belonging to the work category. The expression to use is the following:

```
/addressbook/category[@name = "work"]/person
```

The expression returns only one node, the one with an ID equal to 4, as shown here:

```
<person id="4">
        <lastname>Peacock</lastname>
        <firstname>Margaret</firstname>
</person>
```

After you've created an expression to select a node set, you can create an XML data source.

Open the window for creating a new data source and select **XML File** data source from the list of connection types to display the dialog below.

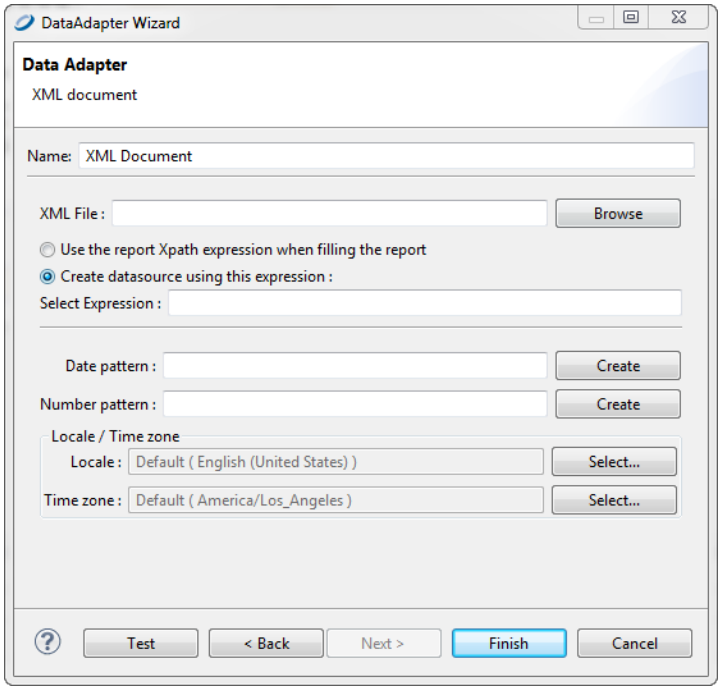


Figure 10-10 Configuring an XML Data Adapter

XML file is the only required field. You have the option to provide a set of nodes, using a pre-defined static XPath expression. Or you can set the XPath expression directly in the report.

We suggest using a report-defined XPath expression. This enables you to use parameters inside the XPath expression, which acts like a real query on the supplied XML data.

Optionally, you can specify Java patterns to convert dates and numbers from plain strings to more appropriate Java objects (like **Date** and **Double**). For the same purpose, you can define a specific locale and time zone to use when parsing the XML stream.

10.5.4 Registration of Fields for an XML Data Source

In addition to the type and name, the definition of a field in a report using an XML data source requires an expression inserted as a field description. As the data source aims always to be one node of the selected node set, the expressions are relative to the current node.

To select the value of an attribute of the current node, use the following syntax:

```
@<name attribute>
```

For example, to define a field that must point to the `id` attribute of a person (attribute id of the node person), it's sufficient to create a new field, name it, and set the description to:

```
@id
```

It's also possible to get to the child nodes of the current node. For example, if you want to refer to the `lastname` node, child of `person`, use the following syntax:

```
lastname
```

To move to the parent value of the current node (for example, to determine the category to which a person belongs), use a slightly different syntax:

```
ancestor::category/@name
```

The ancestor keyword indicates that you're referring to a parent node of the current node. Specifically, the first parent of category type, of which you want to know the value of the name attribute.

Now, let's see everything in action. Prepare a simple report with the registered fields shown here:

Field name	Description	Type
`id`	@id	`Integer`
`lastname`	lastname	`String`
`firstname`	firstname	`String`
`name of category`	ancestor::category/@name	`String`

Jaspersoft Studio provides a visual tool to map XML nodes to report fields; to use it, open the query window and select **XPath** as the query language. If the active connection is a valid XML data source, the associated XML document is shown in a tree view. To register the fields, set the record node by right-clicking a Person node and selecting the menu item **Set record node**. The record nodes are displayed in bold.

Then one by one, select the nodes or attributes and select the pop-up menu item**Add node as field** to map them to report fields. Jaspersoft Studio determines the correct XPath expression to use and creates the fields for you. You can modify the generated field name and set a more suitable field type after the registration of the field in the report (which happens when you close the query dialog).

Insert the different fields in the Detail band. The XML file used to fill the report is that shown:

The XPath expression for the node set selection specified in the query dialog is:

```
/addressbook/category/person
```

10.5.5 XML Data Source and Subreports

A node set allows you to identify a series of nodes that represent records from a `JRDataSource` point of view. However, due to the tree-like nature of an XML document, it may be necessary to see other node sets that are subordinate to the main nodes.

Consider the XML in **Table 10-4, "Complex XML example," on page 147**. This is a slightly modified version of **Table 10-2, "Example XML file," on page 144**. For each person node, a hobbies node is added which contains a series of hobby nodes and one or more e-mail addresses.

Table 10-4 Complex XML example

```
<addressbook>
```

```
	<category name="home">
		<person id="1">
			<lastname>Davolio</lastname>
			<firstname>Nancy</firstname>
			<email>davolio1@sf.net</email>
			<email>davolio2@sf.net</email>
			<hobbies>
<hobby>Music</hobby>
<hobby>Sport</hobby>
			</hobbies>
		</person>
		<person id="2">
			<lastname>Fuller</lastname>
			<firstname>Andrew</firstname>
			<email>af@test.net</email>
			<email>afullera@fuller.org</email>
			<hobbies>
<hobby>Cinema</hobby>
<hobby>Sport</hobby>
			</hobbies>
		</person>
	</category>

	 <category name="work">
		<person id="3">
			<lastname>Leverling</lastname>
			<email>leverling@xyz.it</email>
		</person>
		<person id="4">
			<lastname>Peacock</lastname>
			<firstname>Margaret</firstname>
			<email>margaret@foo.org</email>
			<hobbies>
<hobby>Food</hobby>
<hobby>Books</hobby>
			</hobbies>
		</person>
	</category>
	<favorites>
		<person id="1"/>
		<person id="3"/>
	</favorites>
</addressbook>
```

What we want to produce is a document that is more elaborate than those you have seen so far. For each person, we want to present their e-mail addresses, hobbies, and favorite people.

You can create this document using subreports. You'll need a subreport for the e-mail address list, one for hobbies, and one for favorite people (that is a set of nodes out of the scope of the XPath query we used). To generate these subreports, you need to understand how to produce new data sources to feed them. In this case, you'll use the `JRXmlDataSource`, which exposes two extremely useful methods:

```
public JRXmlDataSource dataSource(String selectExpression)
public JRXmlDataSource subDataSource(String selectExpression)
```

The first method processes the expression by applying it to the whole document, starting from the actual root. The second assumes the current node is the root.

Both methods can be used in the data source expression of a subreport element to dynamically produce the data source to pass to the element. The most important thing to note is that this mechanism allows you to make both the data source production and the expression of node selection dynamic.

The expression to create the data source that feeds the subreport of the e-mail addresses is:

```
((net.sf.jasperreports.engine.data.JRXmlDataSource)
      $P{REPORT_DATA_SOURCE}).subDataSource("/person/email")
```

This code returns all the e-mail nodes that descend directly from the present node (person).

The expression for the hobbies subreport is similar, except for the node selection:

```
((net.sf.jasperreports.engine.data.JRXmlDataSource)
   $P{REPORT_DATA_SOURCE}).subDataSource("/person/hobbies/hobby")
```

Next, declare the master report's fields. In the subreport, you have to refer to the current node value, so the field expression is simply a dot (.),

Proceed with building your three reports: `xml_addressbook.jasper`, `xml_addresses.jasper`, and `xml_hobbies.jasper`.

In the master report, `xml_addressbook.jrxml`, insert a group named "Name of category," in which you associate the expression for the category field (`$F{name of category}`). In the header band for Name of category, insert a field to display the category name. By doing this, the names of the people are grouped by category (as in the XML file).

In the Detail band, position the `id`, `lastname`, and `firstname` fields. Below these fields, add the two Subreport elements, the first for the e-mail addresses, the second for the hobbies.

The e-mail and hobby subreports are identical except for the name of the field in each one. The two reports should be as large as the Subreport elements in the master report, so remove the margins and set the report width accordingly.

Preview both the subreports just to compile them and generate the relative `.jasper` files. Jaspersoft Studio returns an error during the fill process, but that's expected. We haven't set an Xpath query, so JasperReports can't get any data. You can resolve the problem by setting a simple Xpath query (it isn't used in the final report), or you can preview the subreport using an empty data source (select it from the combo box in the tool bar).

When the subreports are done, execute the master report. If everything is okay, the report groups people by home and work categories and the subreports associated with each person.

As this example demonstrates, the real power of the XML data source is the versatility of XPath, which allows navigation of the node selection in a refined manner.

10.5.6 Using XML/A Data Adapters

XML/A (XML for Analysis) is an XML standard for accessing remote data in an OLAP schema. Jaspersoft Studio supports an XML/A data adapter that can connect various XML/A providers such as JasperReports Server and Microsoft SQL Server Analytic Services (SSAS). Because Jaspersoft Studio uses OLAP4J (http://www.olap4j.org/), it may also be able to connect to other types of XML/A providers.

The remote server must also be configured for XML/A. For more information, including instructions for configuring Jaspersoft OLAP, see the *Jaspersoft OLAP User Guide* .

To create an XML/A Data Adapter:

1. Right-click **Data Adapters** in the Repository Explorer, and select **Create Data Adapter**.

2. Select **XML/A Server** and click **Next**.
3. Enter a name for the data adapter.
4. Enter the URL for your XML/A provider. The type of server determines the value. For example:
   - If the XML/A server is JasperReports Server, the URL is something like: http://<hostname>:<port>/jasperserver-pro/xmla
   - If the XML/A server is Microsoft SSAS 2012, the URL is something like: http://<hostname>/MSSQL_2012/msmdpump.dll
5. Enter a user name and password of a user that has sufficient access in the report server to return your data.
6. Click **Get Metadata**.

   Jaspersoft Studio attempts to connect to the server and return information about its data sources, catalogs, and cubes. If it's successful, default values appear in the drop-downs. If the connection fails, check the URL, ensure that the remote server is available, and try again.
7. Select the data source, catalog, and cube that stores the data you want for your report.
8. Click **Test**.

   Jaspersoft Studio connects to the server and read the cube you selected. If the connection fails, check the URL, ensure that the remote server is available, and try again.
9. When the test succeeds, click **OK** to close the message and click **Finish** to close the New Data Adapter wizard.

When you create a report using this data adapter, you may see a message indicating that the data adapter doesn't support the ability to retrieve fields. This means Jaspersoft Studio doesn't have enough information to preview your data. After you provide an MDX query, Jaspersoft Studio can automatically read fields and suggest their datatypes.

10.5.7 Registration of fields in XML/A Providers

When you create an XML/A data adapter, you define the cube from which to read data. Jaspersoft Studio can then inspect the remote server and suggest datatypes for the fields returned.

To register fields returned by an XML/A data adapter:

1. With your report open in the design pane, click  to open the Dataset and Query window.
2. Select the data adapter that points to your XML/A provider from the drop-down in the upper-left corner.
3. Select MDX from the **Language** drop-down.
4. Enter a valid MDX query in the text field.

   To create a good MDX query, you must be familiar with both the language itself and the data you want to work with. You can also use a tool (such as the Jaspersoft OLAP Workbench) to load your OLAP schema and automatically generate MDX queries from it.
5. Click **Read Fields**.

   Jaspersoft Studio returns the XML/A provider's data, including fields and parameters, and populates the window's tabs with information. For more on how these tabs can be used to define the data in your report, see **Chapter 15, "Creating Queries," on page 227**.

   Jaspersoft Studio also sets the class type of each field to an appropriate Java datatype. If Jaspersoft Studio sets an incorrect datatype, you can set the correct type after the fields are added to your report.
6. When the data in the Data Preview tab looks like the data you want to fill your report, click OK to close the Dataset and Query window.

10.5.8 Using CSV Data Sources

To create a connection based on a CSV file, click the **New** button in the Connections/Datasources dialog and select **File CSV data source** from the data source types list to display the dialog.

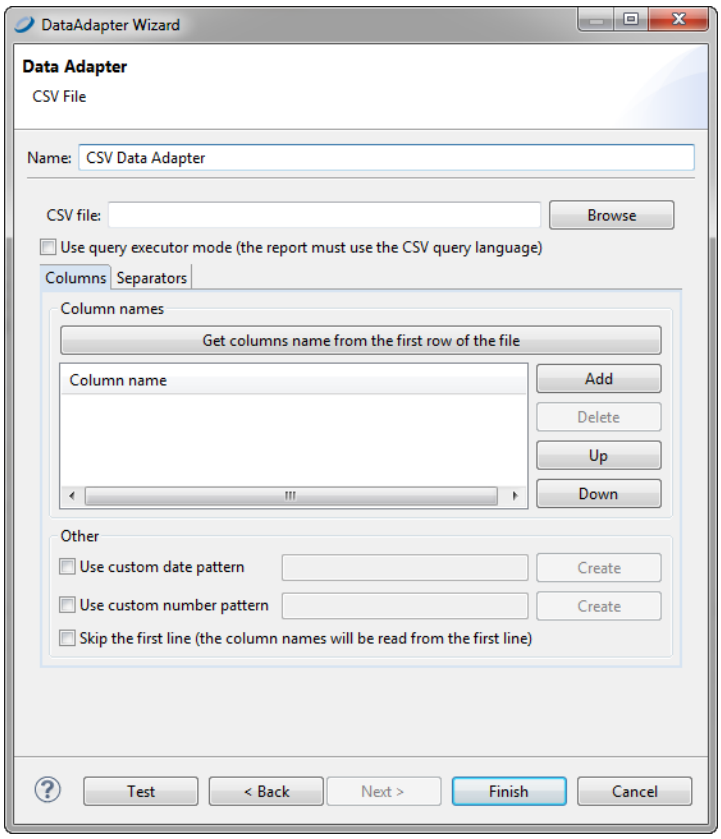


Figure 10-11 CSV Data Adapter

Set a name for the connection and choose a CSV file. Then declare the fields in the data source.

- If the first line in your file contains the names of the columns, click **Get column names from the first row of the file** and select the **Skip the first line** check box . This forces JasperReports to skip the first line (the one containing your column labels). In any case, the column names read from the file are used instead of the declared ones, so avoid modifying the names found with the **Get column names** button.
- If the first line of your CSV file *doesn't* contain the column names, set a name for each column using the syntax `COLUMN_0, COLUMN_1`, and so on.

If you define more columns than the ones available, you'll get an exception at report filling time.

JasperReports assumes that for each row all the columns have a value (even if they are empty).

If your CSV file uses nonstandard characters to separate fields and rows, you can adjust the default setting for separators using the Separators tab.

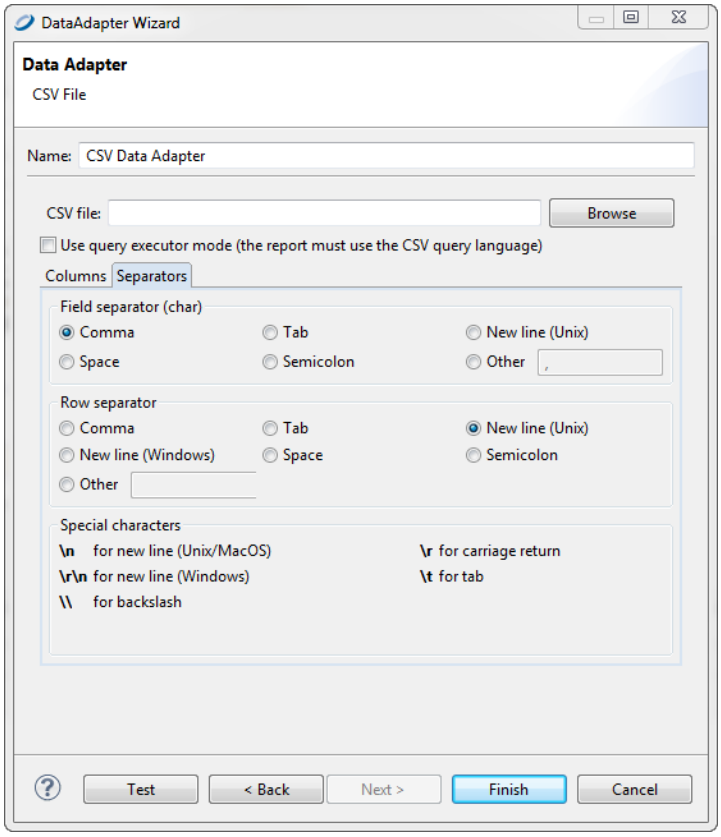


Figure 10-12 Separators Tab

10.5.9 Registration of the Fields for a CSV Data Source

When you create a CSV data source, you must define a set of column names as fields for your report. To add them to the fields list, set your CSV data source as the active connection and open the Report query dialog. Open the **Dataset and Query** Dialog and click the **Read Fields** button.

By default, Jaspersoft Studio sets the class type of all fields to `java.lang.String`. If you're sure the text of a particular column can be easily converted to a number, a date, or a Boolean value, set the correct field type yourself after the fields are added to your report.

The pattern used to recognize a timestamp (or date) object can be configured at the data source level by selecting the **Use custom date format** check box.

10.5.10 Using JREmptyDataSource

JasperReports provides a special data source named `JREmptyDataSource`.

This source returns true to the `next` method for the record number (by default only one), and always returns `null` to every call of the `getFieldValue` method. It's like having records without fields, that is, an empty data source.

The two constructors of this class are:

```
public JREmptyDataSource(int count)public JREmptyDataSource()
```

The first constructor indicates how many records to return, and the second sets the number of records to one.

By default, Jaspersoft Studio provides a pre-configured empty data source that returns a single record.

To create a new empty data source with more records:

1. Double-click **One Empty Record** in the Repository View. The **Data Adapter Wizard** appears with Empty rows.

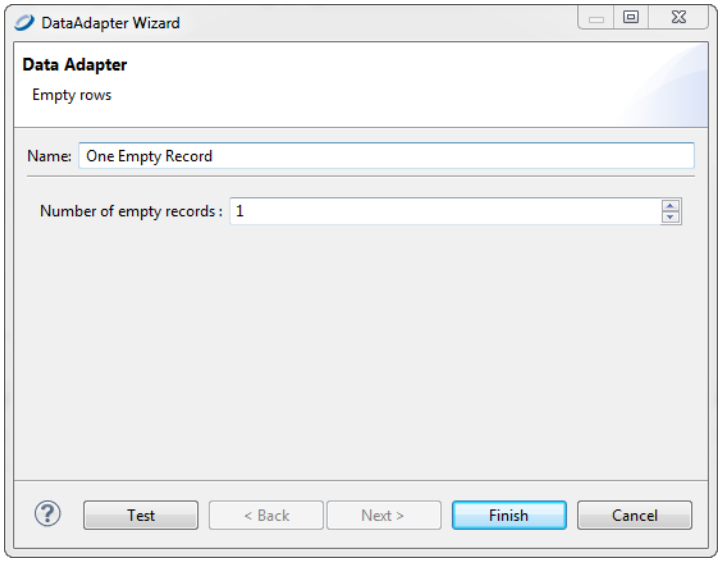


Figure 10-13 Data Adapter Wizard > Empty Record

2. Set the number or empty records you need. Remember, whatever field you add to the report, its value is set to `null`. Since this data source doesn't care about field names or types, this is a perfect way to test any report (keeping in mind that the fields are always set to `null`).
3. Click **Finish**.

10.5.11 Using HQL and Hibernate Connections

JasperReports provides a way to use HQL directly in your report. To do so, first set up a Hibernate connection. Expand your classpath to include all classes, JARs, and configuration files used by your Hibernate mapping. In other words, Jaspersoft Studio must be able to access all the *.hbm.xml files you plan to use, the JavaBeans declared in those files, the `hibernate.cfg.xml` file, and any other JARs used (for example, JARs that access the database under Hibernate).

To add these objects to the classpath:

1. Right-click **Data Adapters** in the Repository Explorer, and select **Create Data Adapter**.
2. Click the **Driver Classpath** tab.

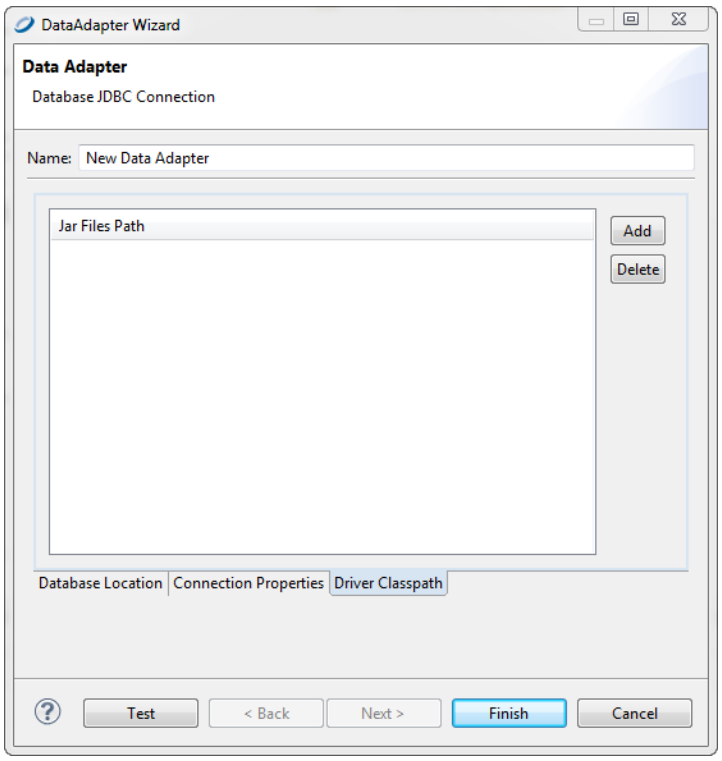


Figure 10-14 Data Adapter Wizard > Add JAR Files

3. Click the **Add** button.
4. Navigate to your JAR file.
5. Click **Open**.
6. Repeat for any additional files.
7. Give your data adapter a unique name.
8. Click the **Test** button to check the path resolution to be certain that `hibernate.cfg.xml` is in the classpath.

   Currently Jaspersoft Studio works only with the first hibernate.cfg.xml file found in the classpath.
9. Click **Finish**.

If you use the Spring framework, you can use a Spring configuration file to define your connection. In this case, you'll need to set the configuration file name and the Session Factory Bean ID.

Now that a Hibernate connection is available, use an HQL query to select the data to print. You can use HQL in the same way as SQL: open the Report query dialog and choose HQL as the query language.

When you enter an HQL query, Jaspersoft Studio tries to retrieve the available fields. According to the JasperReports documentation, the field mappings are resolved as follows:

- If the query returns one object per row, a field mapping can be one of the following ways:
  - If the object's type is a Hibernate entity or component type, the field mappings are resolved as the property names of the entity/component. If a select alias is present, it can be used to map a field to the whole entity/component object.
  - Otherwise, the object type is considered scalar, and only one field can be mapped to its value.

- If the query returns a tuple (object array) per row, a field mapping can be one of the following:
  - A select alias. The field is mapped to the value corresponding to the alias.
  - A property name prefixed by a select alias and a period (.). The field is mapped to the value of the property for the object corresponding to the alias. The type corresponding to the select alias has to be an entity or component.

If you don't understand this field mapping information, simply accept the fields listed by Jaspersoft Studio when the query is parsed.

Jaspersoft Studio provides a mapping tool to map objects and attributes to report fields. The objects (or JavaBeans) available in each record are listed in the combo box on top of the object tree.

To add a field from the tree, select the corresponding node and click the **Add selected field(s)** button.

10.5.12 Using a Hadoop Hive Connection

JasperReports provides a way to use Hive in your reports. Unlike traditional databases, Hadoop systems support huge amounts of data, generally called big data. However, this capability has a cost: high latency with access times between 30 seconds and 2 minutes.

Because of the latency, reports based on Hadoop-Hive data sources are best run in the background or scheduled. For example, the report could be run at 6 a.m. and the HTML or PDF could be exported and stored for anyone wanting to access the report during the day.

To use Hadoop Hive with Jaspersoft Studio:

1. Double-click **New Data Adapter** in the Repository view to display the Data Adapter Wizard.

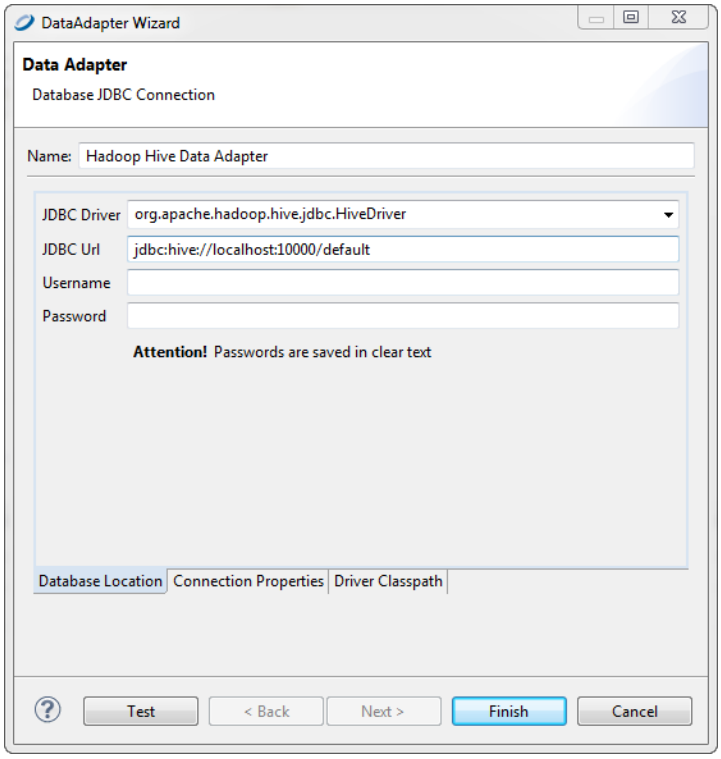


Figure 10-15 Hadoop Hive Connection

2. Choose Hadoop Hive connection as your JDBC driver.
3. Enter the path to your Hive JDBC. Usually it looks something like `jdbc:hive://localhost:10000/default`.
4. Click **Test** to check the path resolution.
5. Click **Finish**.

Use a HiveQL query to retrieve data for your report. You can use HiveQL the same way that you use SQL. To use a HiveQL query, open the Report Query dialog and choose HiveQL as the query language.

When you enter a HiveQL query, Jaspersoft Studio retrieves all available fields. on the next two screens, select the fields you want to display in your report, and how you want to group them. After defining your HiveQL query and choosing your fields, your report is configured to receive data from your Hadoop Hive data source.

The Hive JDBC driver does not yet fully implement the JDBC specifications, and it does not yet work correctly with the JasperReports Server metadata layer (Data Domains). If you're reporting against a Hive data source using JasperReports Server, use Topics rather than Domains.

10.5.13 Implementing a New JRDataSource

If the `JRDataSource` supplied with JasperReports doesn't meet your requirements, you can write a new `JRDataSource`. This is not a complex operation. In fact, all you have to do is create a class that implements the `JRDataSource` interface that exposes two simple methods: `next` and `getFieldValue`.

Table 10-5 The JRDataSource interface

```
package net.sf.jasperreports.engine;
public interface JRDataSource
{
                        public boolean next() throws JRException;
                        public Object getFieldValue(JRField jrField) throws JRException;
}
```

The `next` method is used to set the current record into the data source. It has to return true if a new record to elaborate exists; otherwise it returns false.

If the `next` method has been called positively, the `getFieldValue` method has to return the value of the requested field or `null`. Specifically, the requested field name is contained in the `JRField` object passed as a parameter. Also, `JRField` is an interface through which you can get information associated with a field—the name, description, and Java type that represents it (as mentioned previously in **10.4, "Understanding the JRDataSource Interface," on page 141**).

Now try writing your personalized data source. You have to write a data source that explores the directory of a file system and returns the found objects (files or directories). The fields you create to manage your data source are the same as the file name, which should be named FILENAME; a flag that indicates whether the object is a file or a directory, which should be named IS_DIRECTORY; and the file size, if available, which should be named SIZE.

You data source should have two constructors: the first receives the directory to scan as a parameter; the second has no parameters and uses the current directory to scan.

Once instantiated, the data source looks for the files and the directories present in the way you indicate and fills the array files.

The `next` method increases the index variable you use to keep track of the position reached in the array files, and returns true until you reach the end of the array.

Table 10-6 Sample personalized data source

```
import net.sf.jasperreports.engine.*;
import java.io.*;
public class JRFileSystemDataSource implements JRDataSource
{
File[] files = null;
int    index = -1;
public JRFileSystemDataSource(String path)
{
File dir = new File(path);
if (dir.exists() && dir.isDirectory())
{
files = dir.listFiles();
}
}
public JRFileSystemDataSource()
{
this(".");

}
public boolean next() throws JRException
```

```
{
index++;
if (files != null && index < files.length)
{
return true;
}
return false;
}
                                public Object  getFieldValue(JRField jrField) throws JRException
                                {
                                File f = files[index];
                                if (f == null) return null;
                                if (jrField.getName().equals("FILENAME"))
                                {
                                return f.getName();
                                }
                                else if (jrField.getName().equals("IS_DIRECTORY"))
                                {
                                return new Boolean(f.isDirectory());
                                }

                                 else if (jrField.getName().equals("SIZE"))
                                {
                                return new Long(f.length());
                                }
                                // Field not found...
                                return null;
                                }
}
```

The `getFieldValue` method returns the requested file information. Your implementation doesn't use the information regarding the return type expected by the caller of the method. It assumes the name has to be returned as a string. The flag `IS_DIRECTORY` as a Boolean object, and the file size as a `Long` object.

The next section shows how to use your personalized data source in Jaspersoft Studio and test it.

10.5.14 Using a Custom JasperReports Data Source with Jaspersoft Studio

Jaspersoft Studio provides a special connection for your personalized data sources. It's useful for employing whatever `JRDataSource` you want to use through some kind of factory class that provides an instance of that `JRDataSource` implementation. The factory is just a simple Java class useful for testing your data source and filling a report in Jaspersoft Studio. The idea is the same as what you have seen for the JavaBeans set data source—You need to write a Java class that creates the data source through a static method and returns it. For example, if you want to test the `JRFileSystemDataSource` in the previous section, you need to create a simple class like that shown in this code sample:

Table 10-7 Class for testing a custom data source

```
import net.sf.jasperreports.engine.*;
public class FileSystemDataSourceFactory {
       public static JRDataSource createDatasource()
{
return new JRFileSystemDataSource("/");
}
}
```

This class, and in particular the static method that's called, executes all the necessary code for instancing the data source correctly. In this case, you create a new `JRFileSystemDataSource` object by specifying a way to scan the directory root (`"/"`).

Now that you have defined the way to obtain the `JRDataSource` you prepared and the data source is ready to be used, you can create the connection through which it can be used.

Create a new connection as you normally would (see **"Creating and Using Database JDBC Connections" on page 135**), then select **Custom implementation of JRDataSource** from the list and specify a data source name like `TestFileSystemDataSource` (or whatever name you want), as shown below

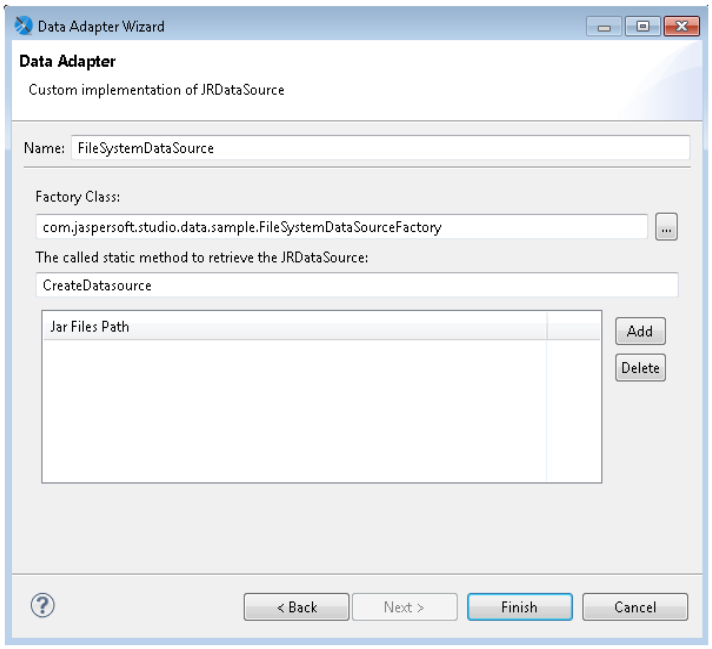


Figure 10-16 Configuring a Custom Data Adapter

Next, specify the class and method to obtain an instance of your `JRFileSystemDataSource`, that is, `TestFileSystemDataSource` and `test`.

There is no automatic method to find the fields managed by a custom data source.

In this case, you know that the `JRFileSystemDataSource` provides three fields: `FILENAME` (String), `IS_DIRECTORY` (Boolean), and `SIZE` (Long). After you have created these fields, insert them in the report's Detail band.

Divide the report into two columns and in the Column Header band, insert Filename and Size tags. Then add two images, one representing a document and the other an open folder. In the `Print when` expression setting of the Image element placed in the foreground, insert the expression `$F{IS_DIRECTORY}`, or use as your image expression a condition like the following:

```
($F{IS_DIRECTORY}) ? "folder.png" : "file.png"
```

In this example, the class that instantiated the `JRFileSystemDataSource` was very simple. But you can use more complex classes, such as one that obtains the data source by calling an Enterprise JavaBean or by calling a web service.

10.6 Importing and Exporting Data Sources

Jaspersoft Studio enables you to import and export data source definitions to simplify the process of sharing data source configurations.

To export one or more data sources, right-click the data adapter and choose **Export to File**. Jaspersoft Studio prompts you to name the file and select the destination for the exported information. A simple XML file is created.

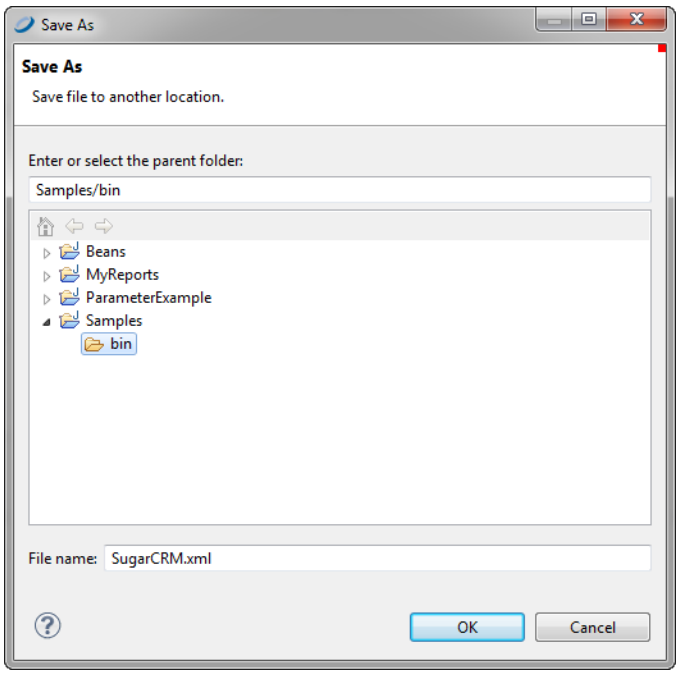


Figure 10-17 Export to File Dialog

Any file exported with Jaspersoft Studio can also be imported. Right-click in the Repository view, and choose **Import from Workspace**. Because exported files can contain more than one data source or connection definition, the import process adds all the data sources found in the specified file to the current list.

If a duplicate data source name is found during the import, Jaspersoft Studio appends a number to the imported data source name.

10.7 A Look at TIBCO Spotfire Information Links

You can populate reports created in Jaspersoft Studio with data from TIBCO Spotfire. You can navigate your Spotfire library, select Information Links and Spotfire Binary Data Files (SBDFs), and inspect their data. To load Spotfire data, create a data adapter to connect to the data. The data adapter will return data in tabular form.

Before you publish the report to JasperReports Server, export your data adapter as an XML file so you can add it to the server's repository.

This version of Jaspersoft Studio supports TIBCO Spotfire 6.5 and above. Earlier versions may also work but have not been tested extensively.

To create a data adapter for a Spotfire Information Link:

1. In the Repository Explorer, right-click Data Adapters and select **Create Data Adapter** to display the Data Adapter wizard.
2. Enter a name for the data adapter.
3. Enter the URL to your Spotfire Web Player in this form:

   http://<web-player-host>/SpotfireWeb

   where <web-player-host> is the IP address or name of the computer hosting the Spotfire Web Player where you access your Information Link.
4. Enter your Spotfire user name and password.
5. Click **Browse** next to the **Resource ID** field to locate and select your Information Link in the Spotfire library.

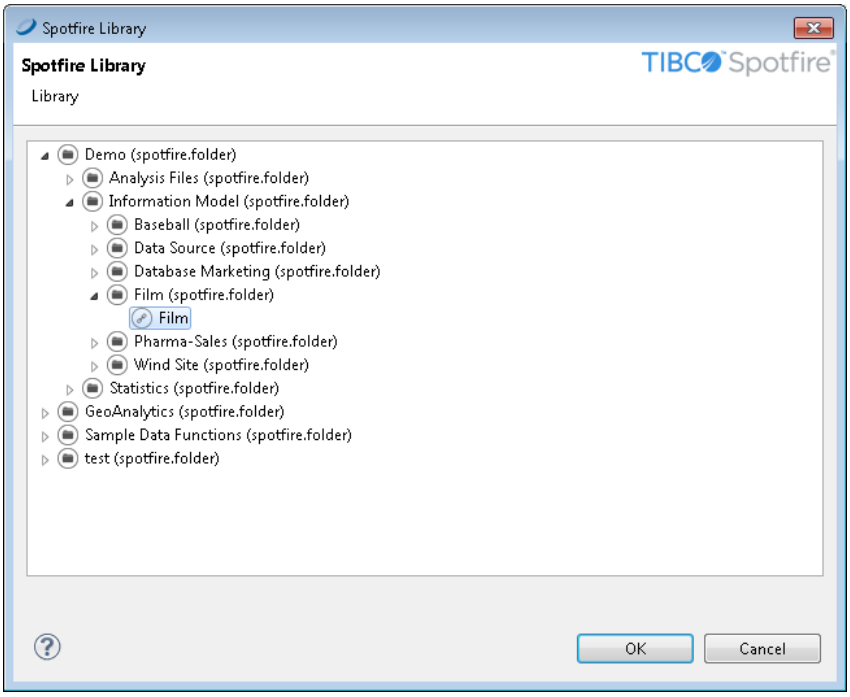


Figure 10-18 Spotfire Library displayed in Jaspersoft Studio

You can also create reports against SBDFs; to do so, select one from your Spotfire library.

6. Click **OK**.
7. Click **Test** to test your connection.
8. If the test fails, check your URL, credentials, and resource ID.
9. When the test succeeds, click **Finish**.

It isn't uncommon for an Information Link to return millions of rows of data, which may take a some time for Jaspersoft Studio to process when data is loaded, such as when previewing the report; the same may hold true in JasperReports Server.

To create your report:

1. Click **File** > **New** > **JasperReport**.
2. Select template and enter a name for your report.
3. Click **Next**. The Data Source dialog appears.
4. Select the Spotfire Information Link data adapter you created above.
5. Select the fields to include in your data set.
6. Click **Next**.
7. Optionally select a field to define grouping.
8. Click **Finish**. Jaspersoft Studio displays the report in the **Design** tab.
9. Edit the report as needed. For example, add fields and components and configure your query and dataset.
10. Click **Preview** to ensure that the report is correctly configured.
11. When your report is ready, click **File** > **Save**.

To export your data adapter as an XML file:

1. In the Repository Explorer, right-click your Spotfire Information Link data adapter and select **Export to File**.
2. Select the folder in your Jaspersoft Studio workspace that contains your report, enter a name for the data adapter, and click **OK**.

   The data adapter and report don't have to be in the same place, but it can make it easier to publish your report.

To configure the report to use the exported data adapter:

1. In the Outline pane, click the root node of your report to display the report properties.
2. Click **Advanced**.
3. Expand Misc and click the ellipsis to the right of Properties to open the Properties window.
4. Click **Add** to create a property that indicates the data adapter to use.
5. In the **Property Name** field, enter `net.sf.jasperreports.data.adapter`.
6. In the **Value** field, enter the name of the data adapter you exported (this is an XML file).
7. Click **OK**.

   You're ready to publish your report.

To publish your report:

1. Click  at the top of the **Design** tab. You're prompted to select a location for publishing the report.
2. Select a server connection and navigate its repository to the desired location.
3. Optionally enter a new label, name (ID), and description of the report unit.
4. Click **Next**. You're prompted to select a resource used by the report, including the data adapter you exported above.
5. Click **Next**. You're prompted to select a data source.

6. Select **Don't use any Data Source**. Since the data adapter has already been defined, the report doesn't need a separate data source.
7. click **Finish**. Jaspersoft Studio adds your report to the repository.

While it's uploaded, Jaspersoft Studio modifies your JRXML so that it will run properly on the server. In particular, it changes how the data adapter is specified. On the server, the data adapter is uploaded to the folder you selected when you published the report.

To test your report, open the server's web UI, locate your report, and click it to run it.

CHAPTER 11 USING TABLES

The Table component displays data coming from a secondary dataset. This powerful component can often make subreports unnecessary.

The Table wizard allows you to create a complex table with a few clicks. Each table cell can be simple as a text element or it can contain a set of report elements including nested tables, creating very sophisticated layouts.

This chapter contains the following sections:

11.1 Creating a Table

Drag the Table element from the Elements palette into any band of the report. The Table Wizard enables you to create a table from a new or existing dataset.

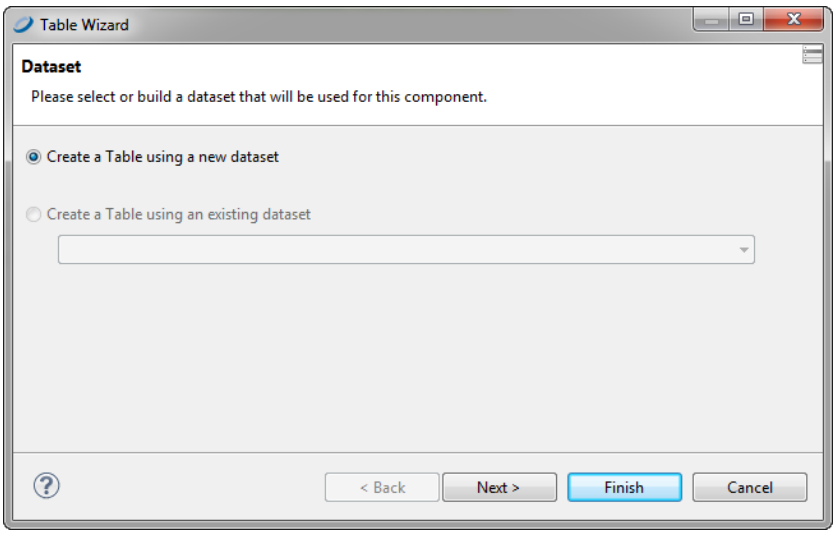


Figure 11-1 Table Wizard - New Table

To create a table from a new dataset:

1. Select **Create a Table from a new dataset** and click **Next**. The Dataset window appears.

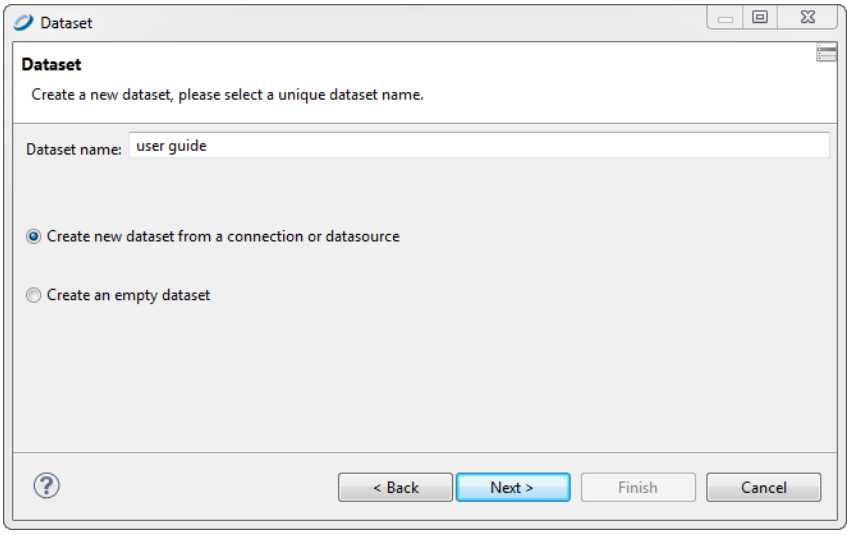


Figure 11-2 Table Wizard - Dataset

2. Name your dataset and select your option: **Create new dataset from a connection or datasource** or **Create an empty dataset**. For this example, choose the first option and click **Next**. You'll be prompted to select a datasource and design query.

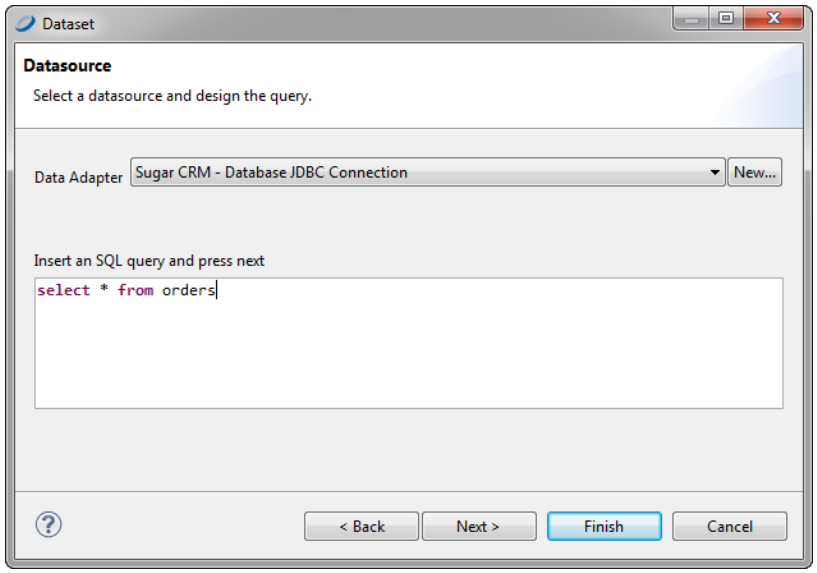


Figure 11-3 Table Wizard - Dataset Datasource

3. Select a data source and enter an SQL query such as: `select * from orders` and click **Next**. You'll be prompted to select dataset fields.

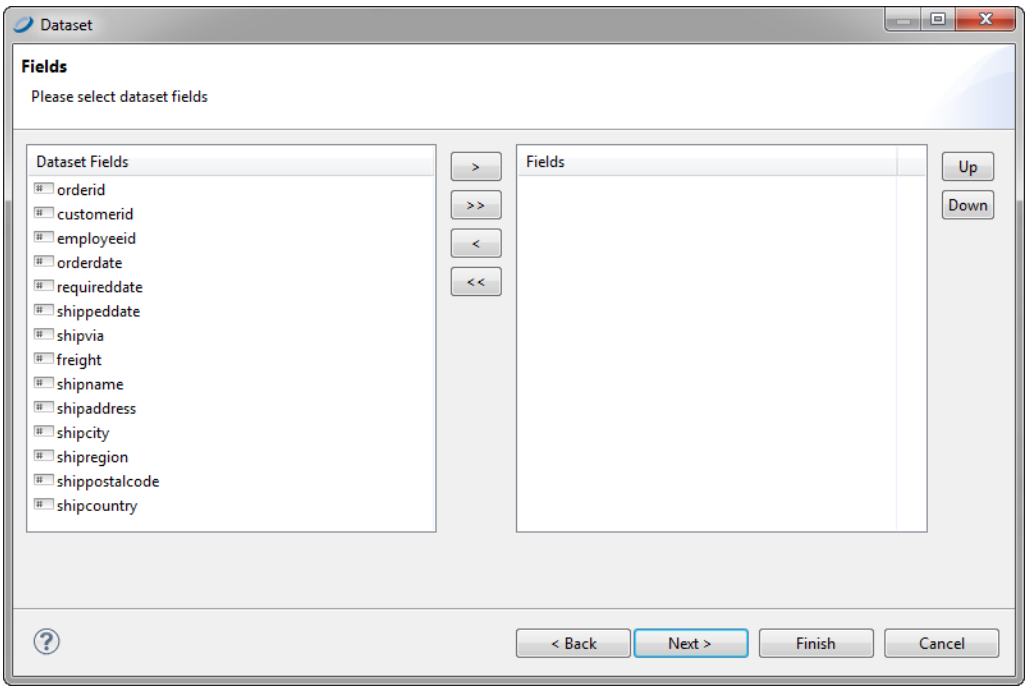


Figure 11-4 Table Wizard - Dataset Fields

4. Select the fields you want in your table and add them to the **Fields** list on the right. Then click **Next**. You'll be prompted to select the fields to group by from among your chosen fields.

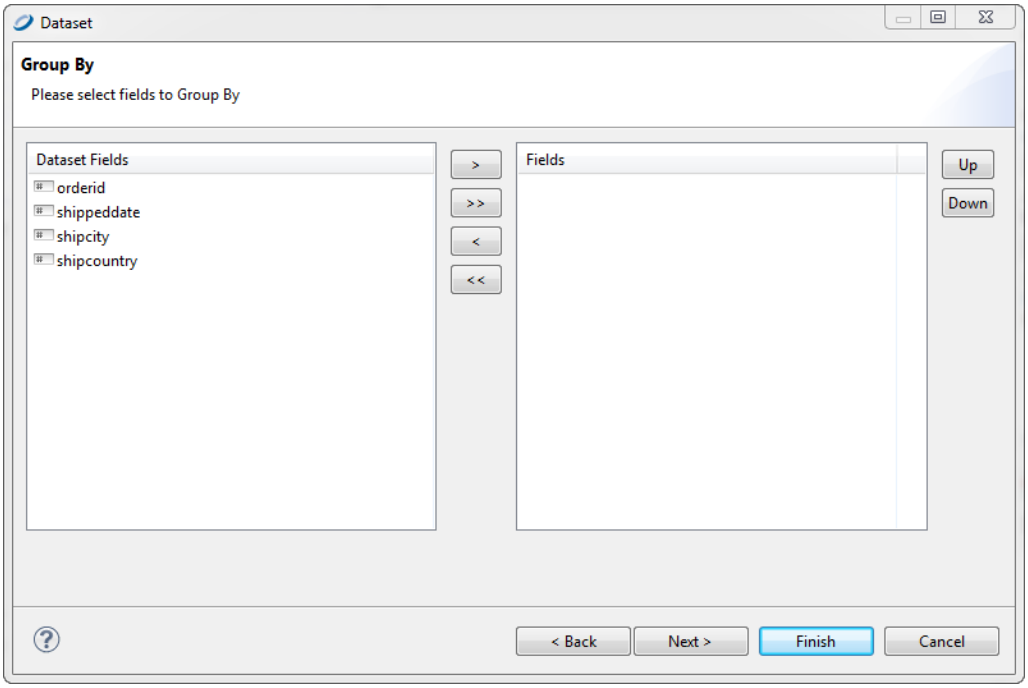


Figure 11-5 Table Wizard - Dataset > Group By

5. Select one or more fields to group by and move them to the Fields list on the right. Click **Next**. You'll be prompted to select a connection.

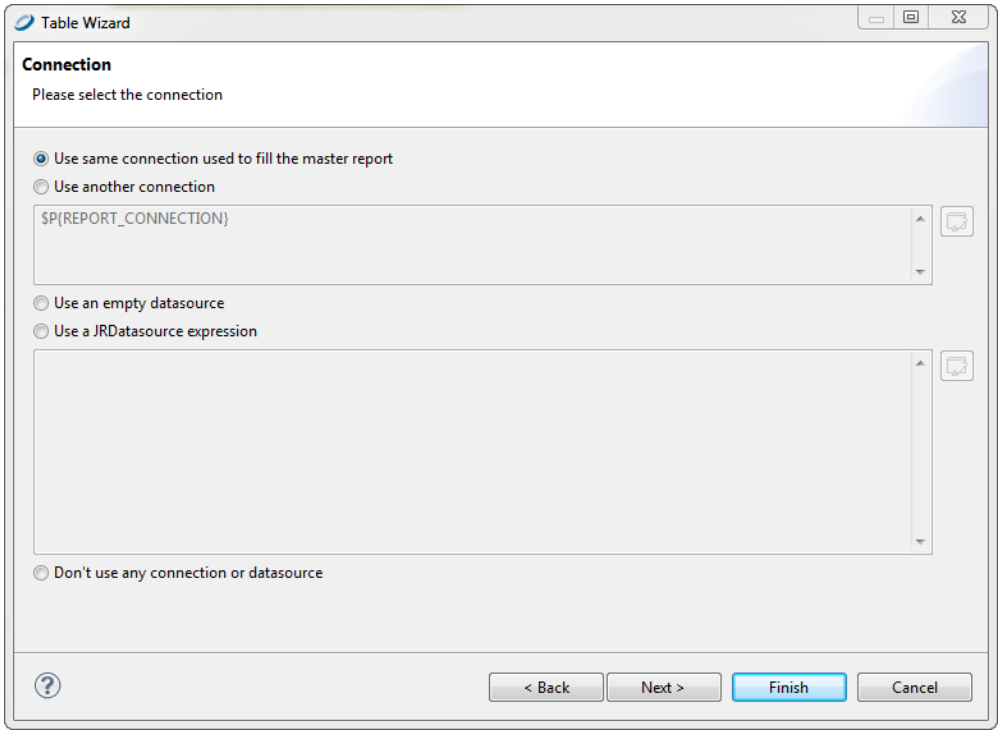


Figure 11-6 Table Wizard - Connection

6. Select a data connection option. Your options are:
   - Use the same connection used to fill the master report (the option used in this example)
   - Use another connection (you'll provide a connection)
   - Use an empty datasource
   - Use a JRDatasource expression (you'll enter a JRDatasource expression)
   - Don't use any datasource or connection
7. Click **Next**. You'll be prompted to choose the fields for produce table columns.

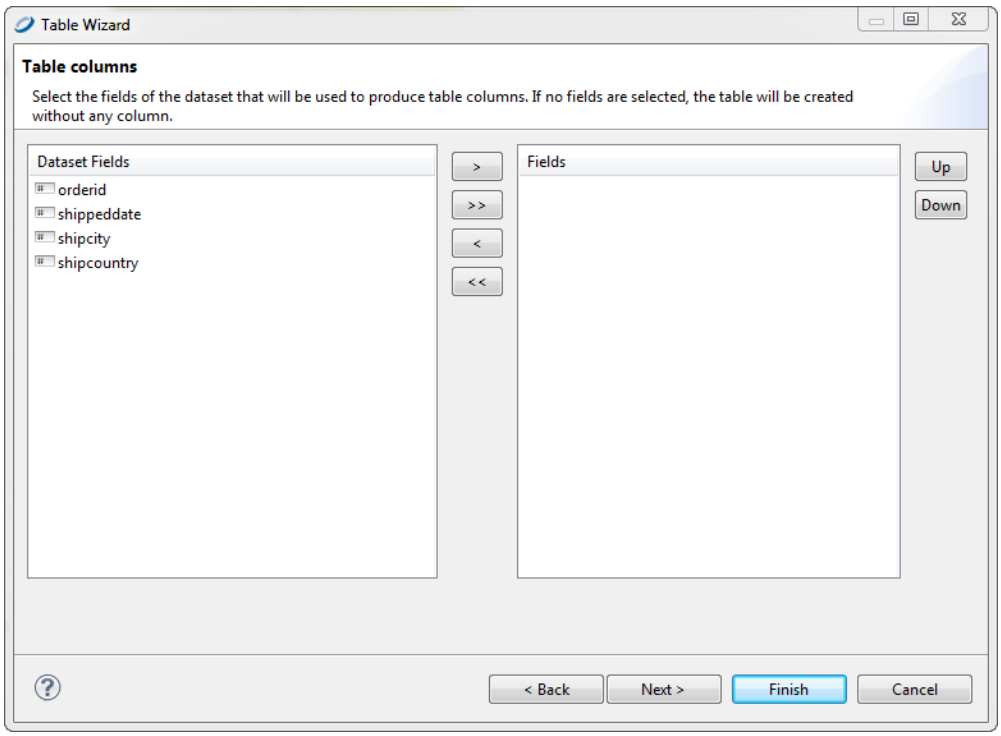


Figure 11-7 Table Wizard - Table Columns

8. Select one or more fields to for table columns and move them to the Fields list on the right. Click **Next**. You'll be prompted to select a layout.

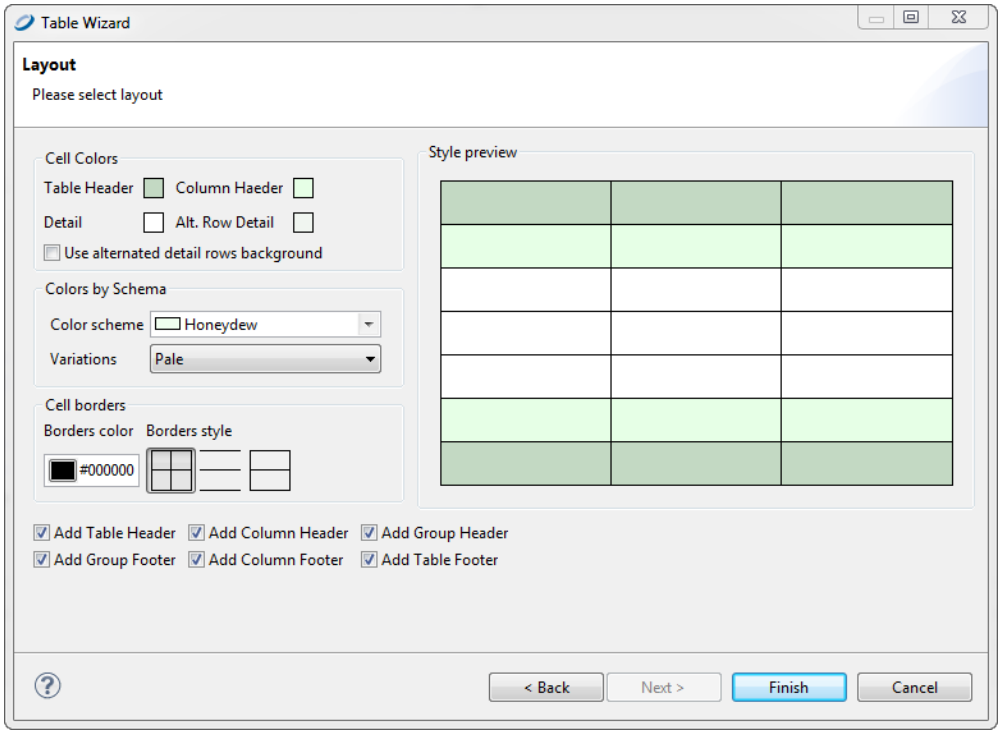


Figure 11-8 Table Wizard - Layout

9. Select the layout for your table, and click **Finish**. The table appears where you dragged the table element in your report.

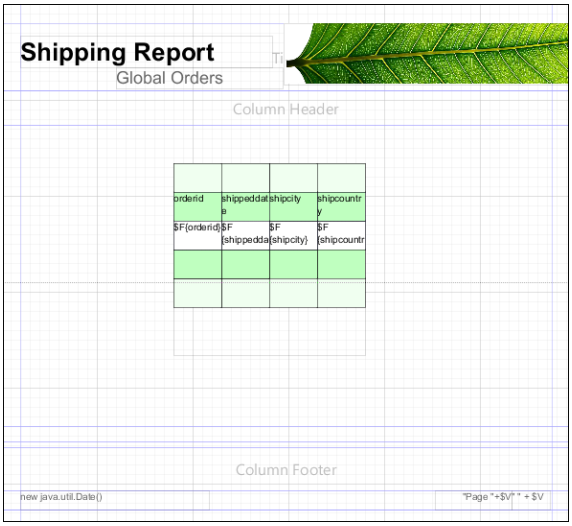


Figure 11-9 Report Containing a Table

To create a table using an existing dataset:

Creating a table using an existing dataset is largely the same as creating a table using a new dataset.

1. In the Dataset window of the Table Wizard, select **Create a Table using an existing dataset**.
2. Select a dataset from the drop-down.

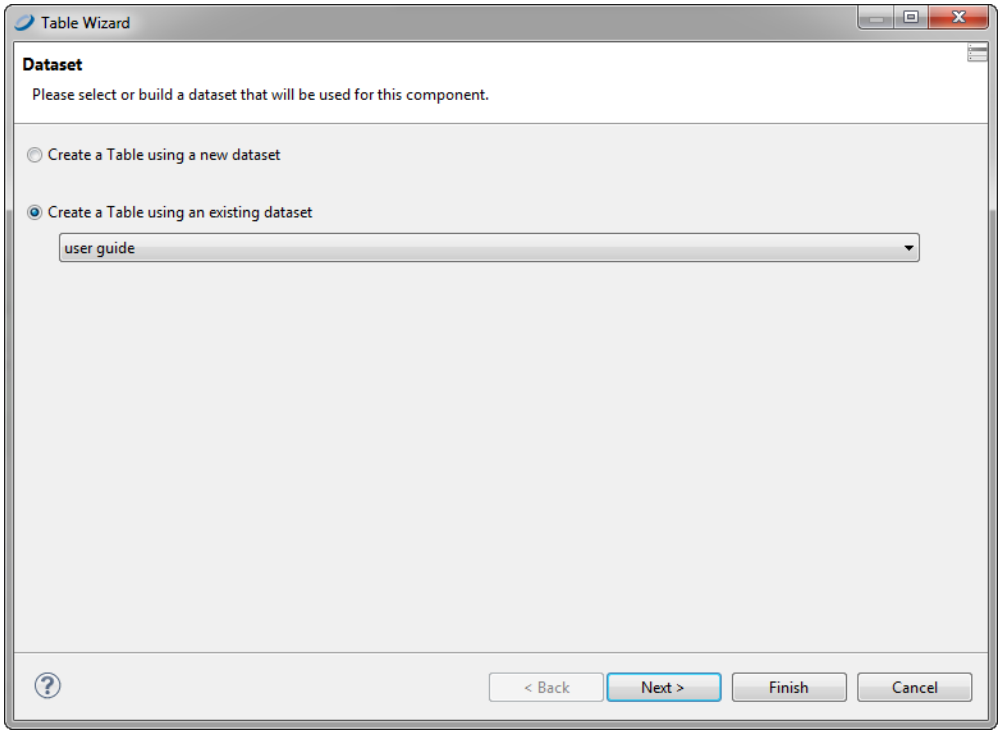


Figure 11-10 Table Wizard - Dataset

3. Click **Next**. You'll be prompted to select the connection.

From this point the steps are the same as creating a table using a new dataset.

11.2 Editing a Table

You can edit tables on the **Design** tab or the **Source** tab. In the source the tags are labeled:

- `Table`: External border of the table.
- `Table_TH`: Table header background color and cell borders.
- `Table_CH`: Table column background.
- `Table_TD`: Detailed cell style. `Table_TD` can be nested to display alternating background color for the detail rows.

11.2.1 Editing Table Styles

You can edit the look and feel of a table by choosing elements from the **Table Styles** tab. To create a style, click the new style button  . In the Table Wizard Layout window, you can select a style to add to your palette or create a new style to save to your palette.

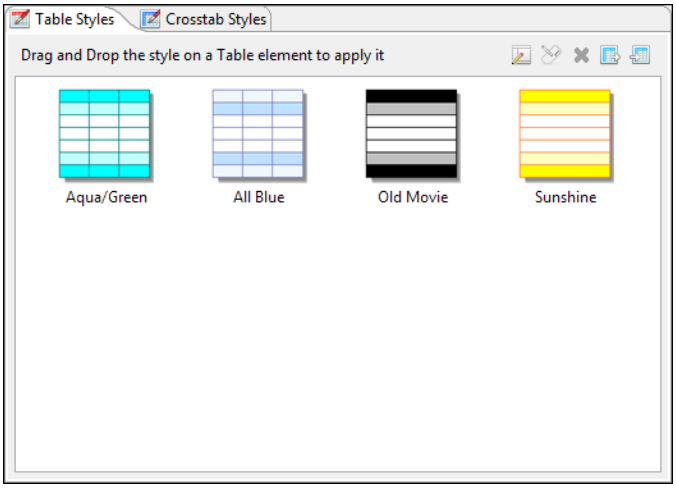


Figure 11-11 Table Styles Tab

To edit a style in the palette, double-click the style and edit it in the Layout window. You can save it either as a new style or with the same name. To edit table layout in the **Design** view, right-click the table and choose **Change Table Style**.

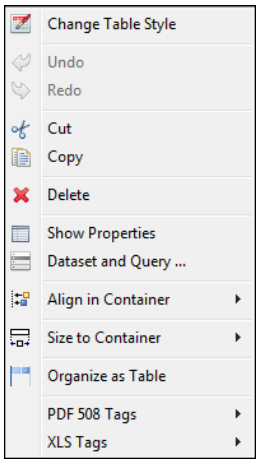


Figure 11-12 Change Table Style

You can also decide which table sections to create. If the dataset for the table contains groups, it can be convenient to select the **Add Group Headers** and **Add Group Footers** check boxes.

To delete a style right click it in the Table Styles tab and choose **Delete Style**.

11.2.2 Editing Cell Contents

You can edit the content, style, and size of each cell or group of cells in your table. To edit table content, double-click your table. The table opens in a separate tab in **Design** view.

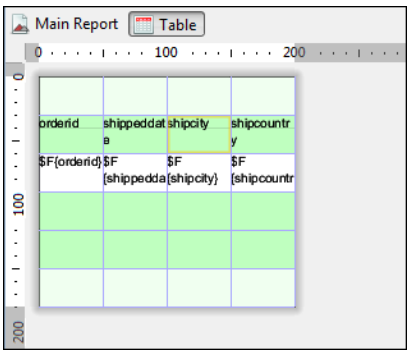


Figure 11-13 Table Editing Tab

To edit the content and style of a cell, click the cell. Switch to the **Properties** tab where you can edit location, size, color, style, and print details for the cell.

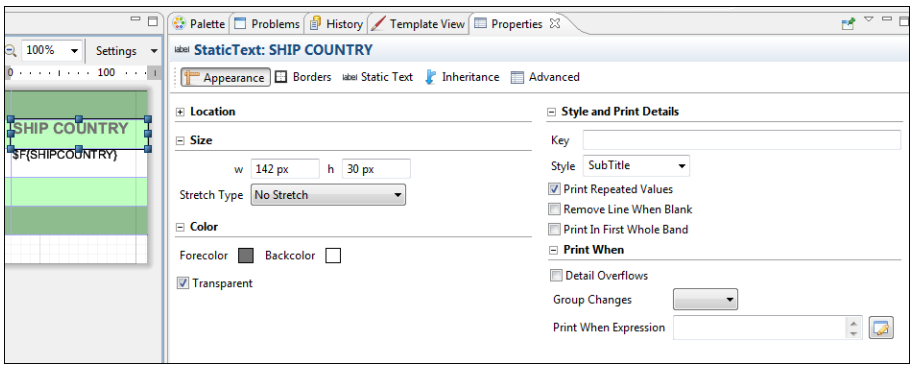


Figure 11-14 Cell with Open Properties Tab

You can edit the size and position of a cell, or copy or delete it, by right-clicking on the cell.

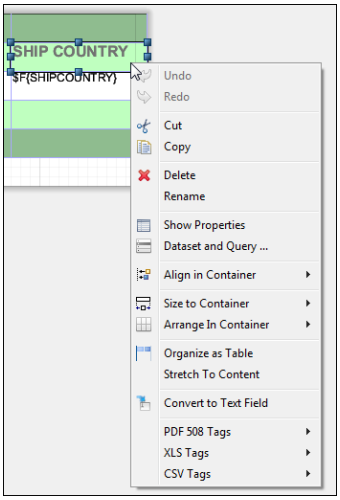
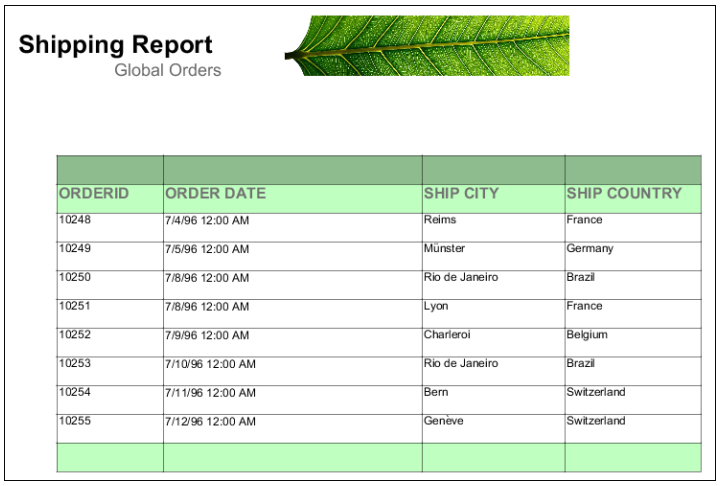


Figure 11-15 Cell Right-Click Menu

Use shift-click to select an all cells in a row. See **"Working with Columns" on page 177** for information about columns.

The following figure is the table created in **"Creating a Table" on page 165**, after formatting and editing:



ORDERID	ORDER DATE	SHIP CITY	SHIP COUNTRY
10248	7/4/96 12:00 AM	Reims	France
10249	7/5/96 12:00 AM	Münster	Germany
10250	7/8/96 12:00 AM	Rio de Janeiro	Brazil
10251	7/8/96 12:00 AM	Lyon	France
10252	7/9/96 12:00 AM	Charleroi	Belgium
10253	7/10/96 12:00 AM	Rio de Janeiro	Brazil
10254	7/11/96 12:00 AM	Bern	Switzerland
10255	7/12/96 12:00 AM	Genève	Switzerland

Figure 11-16 Formatted Table

11.2.3 Editing Table Data

You can edit the dataset used for the table by right-clicking the table and selecting **Dataset and Query** to display the Dataset and Query Dialog.

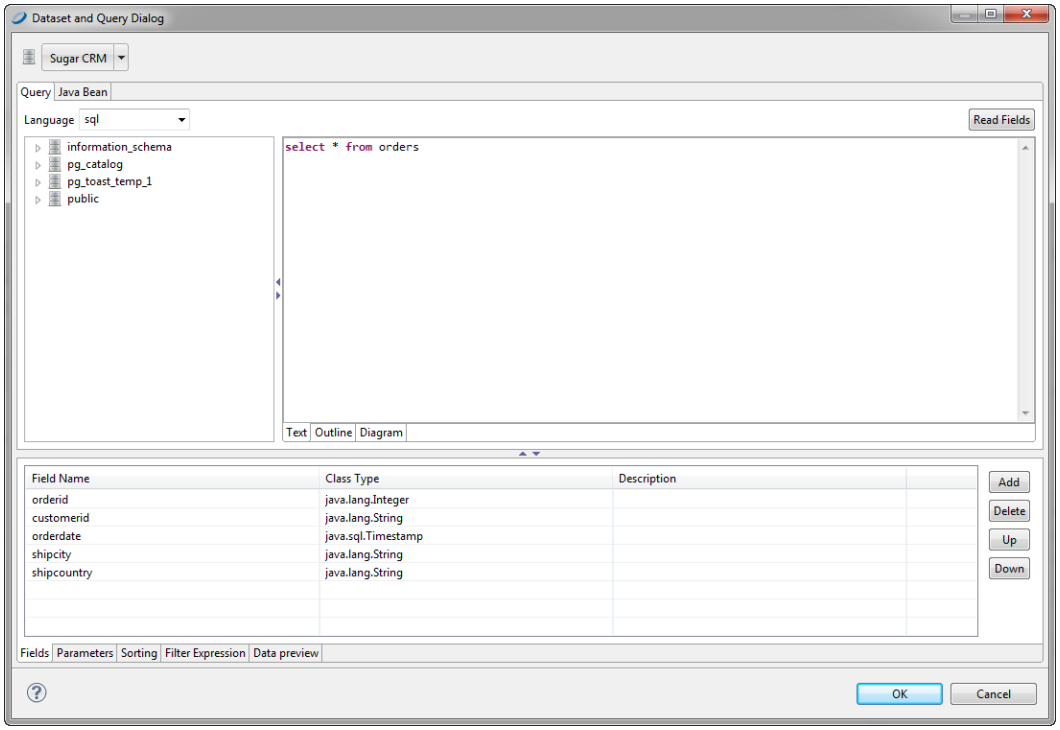


Figure 11-17 Dataset and Query Dialog

Here you can set the dataset parameters to dynamically filter the data used in the table.

Suppose, for instance, you have a report that displays order details in a table, you can use a parameter in your SQL query to specify the order ID to filter the order details.

Unlike charts and crosstabs, a table always requires a subdataset. Tables cannot use the main dataset.

11.3 Table Structure

Tables consist of cells and columns. This section provides more information about working with each of these elements.

11.3.1 Table Elements

A table must have at least one column, but it can have any number. A set of columns can collected into a column group with a heading that spans several all those columns.

Each table is divided into sections similar the main document bands:

- **Table header and footer** - each printed only once.
- **Column header and footer** - repeated on each page the table spans. For column groups, the table can display a group header and footer section for each group and for each column.

- **Detail** - repeated for each record of the table. Each column contains only one detail section, and the section cannot span multiple columns.

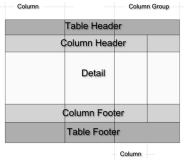


Figure 11-18 Table Structure

In the **Outline** view, table sections are shown as child nodes of the table element node.

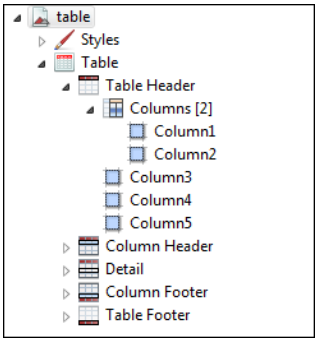


Figure 11-19 Table in Outline View

In the Design view, each column has a cell for each section (for example, one cell for the table header section, another for the table footer, etc.). A cell can be undefined. If all the cells of a section are undefined, the section is not printed. If the height of all the cells of a section is zero, the section is printed but is not visible in the Design view.

11.3.2 Table Cells

A cell can contain any JasperReports element. When an element is dropped on a cell, Jaspersoft Studio automatically arranges the element to fit the cell size. To change the arrangement of the elements, right-click the cell (or element) and select an option from the context menu. Alternatively, you can insert a frame element in the cell and add all the required elements in that frame.

You can delete a cell by right-clicking and choosing **Delete cell**. If the cell is the only defined cell in the column, the entire column is removed. Similarly, if a cell is undefined, right-click it and select **Add cell** to create the cell. An undefined cell is automatically created when the user drags an element into it (see **"Working with Columns" on page 177**).

Edit cell properties from the **Properties** tab:

- The **Appearance** sub-tab allows you to set location, size, color, style and print details.
- You can set cell padding as well as borders from the **Properties > Borders** tab.
- Cell height defines the vertical dimension of a cell. When its value is changed, the new dimension is propagated to all the cells in the row.

11.4 Working with Columns

To add or delete a column to a table, right-click a column and choose an action from the context menu. By default, when Jaspersoft Studio adds a column to a table, the new column inherits the properties of the other columns.

You can drag a column to any position, inside or outside a group. Move a column within the same section by dragging the nodes that represent the columns in the Report Inspector.

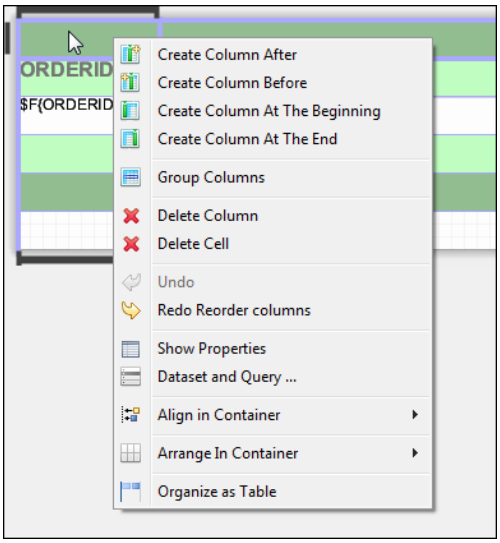


Figure 11-20 Column Context Menu

11.4.1 Column Groups

A column is composed of a set of cells. If you create a column group, a column heading can span all columns in the group. A column group can include other column groups.

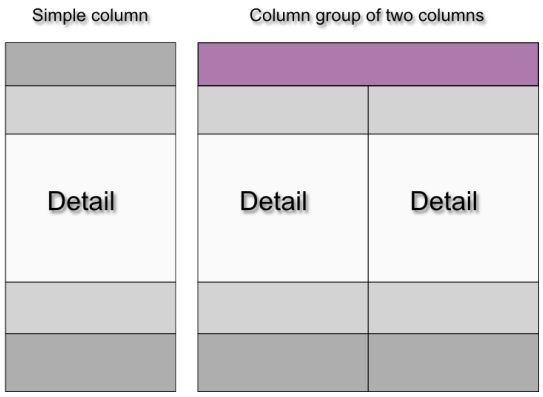


Figure 11-21 Simple Column vs. Column Group

A column group acts as a single when you drag it. If you drag the last column out of a column group, the column becomes a simple column and the remaining group cells are deleted.

When you create a column group, every column section gets a group heading, as shown in **Figure 11-22**, but you can remove unnecessary headings. On the left of the figure there are two columns (most of the sections in the columns have only one record; one section has two records). When the columns are grouped, each column section gets a group heading, as shown in the center. However, most of the group headings are unnecessary, so their heights have been set to zero to hide them, as shown on the right.

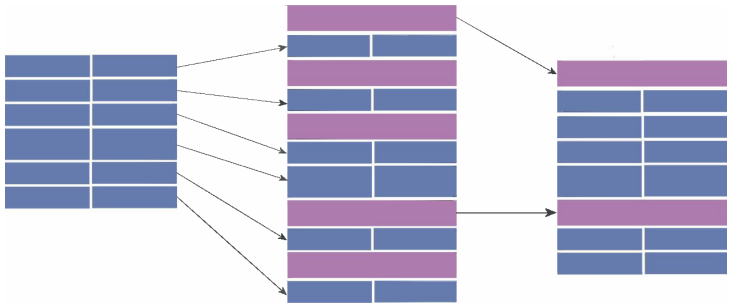


Figure 11-22 Group headings

CHAPTER 12 WORKING WITH CHARTS

JasperReports Server supports a wide variety of chart types. In Jaspersoft Studio you can render charts inside a report two different ways. You can print the data coming from the main dataset or use a subdataset. This allows you to include many different charts in one document without using subreports.

This chapter has the following sections:

12.1 Datasets

When you generate a report, chart data is collected and stored within the chart's dataset.

The dataset types are:

- Pie
- Category
- Time period
- Time series
- XY
- XYZ
- High low
- Value

Think of a dataset as a table. Each dataset has different columns (fields). When a new record is put in the dataset, values are added to the fields.

Every report has a main dataset defined during its creation. But sometimes we need fields that are not returned by a query from the main dataset, or fields in a different data source. The dataset element allows you to define many datasets inside a report, each with its own fields and data source. Every dataset is independent, so its fields are separated from those of the main dataset, and from those of other datasets.

12.1.1 Creating a Dataset

1. Right-click the report root node in the outline view and choose Create Dataset:

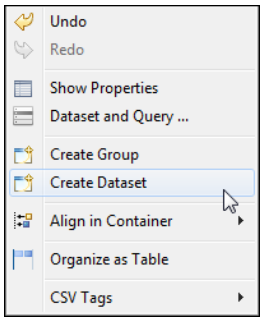


Figure 12-1 Create Dataset

2. In the Dataset wizard, name your dataset, choose **Create new dataset from a connection** or Data Source and click **Next**. The Dataset - Data Source window opens.

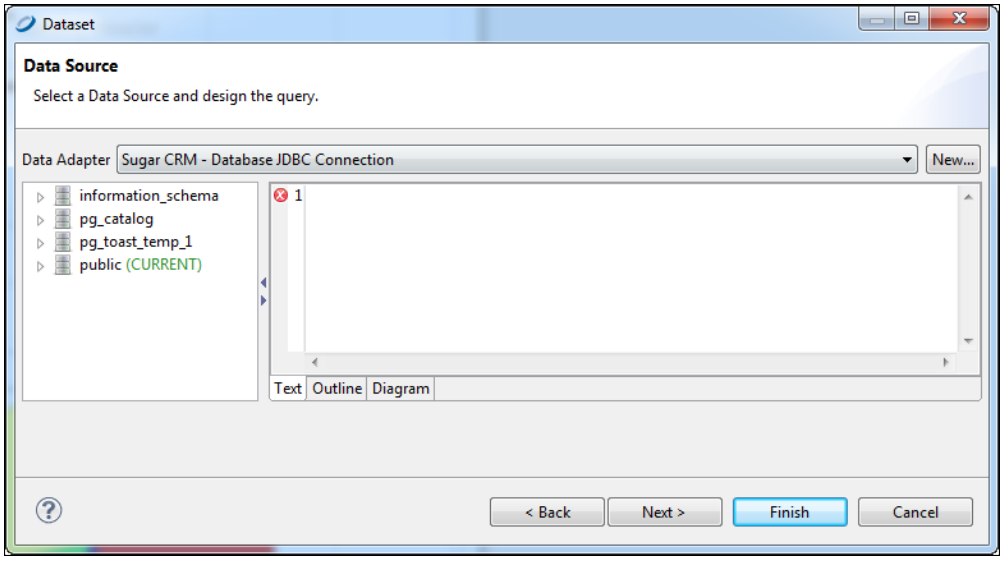


Figure 12-2 Dataset - Data Source Window

3. Select **Sugar CRM - Database JDBC Connection**.
4. Enter the following query:

```
select * from orders
```

5. Click **Next**. The Dataset - Fields window opens.

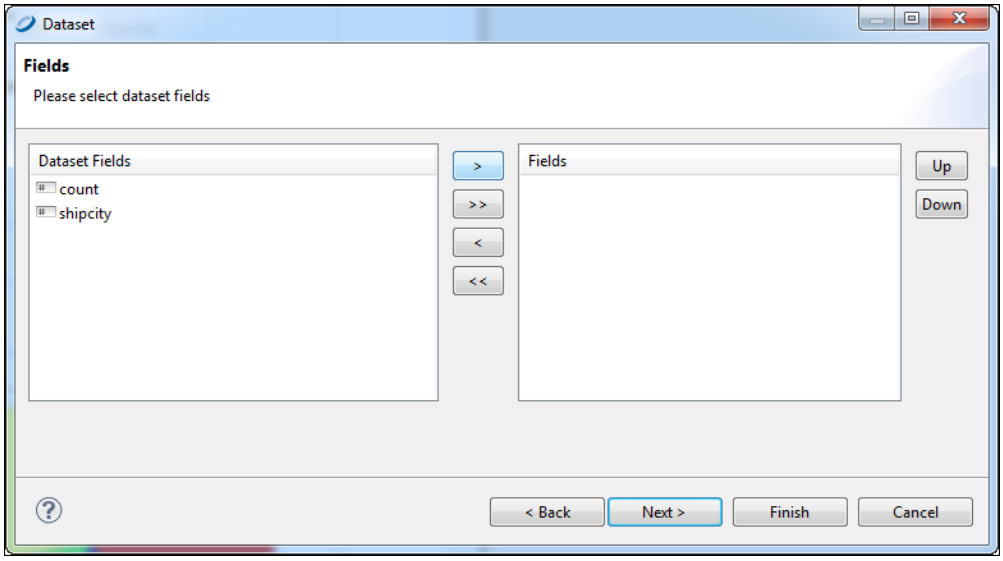


Figure 12-3 Dataset - Fields

6. Select the dataset fields you want and move them to the Fields list on the right and click **Next**.
   The **Dataset - Group By** window opens. This is where you can choose fields to group by, and choose whether to use group fields as sort fields.
7. Do not select any fields to group by. Click **Finish**. The dataset you created in the **Outline** view appears.

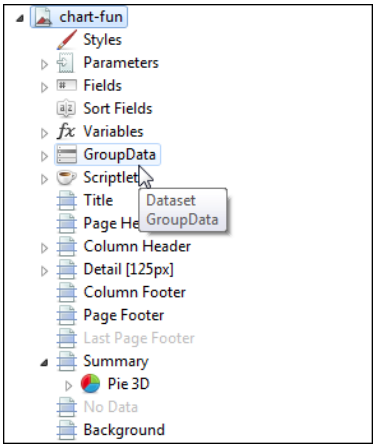


Figure 12-4 GroupData Data Set

12.2 Creating a Simple Chart

This section shows you how to use the Chart tool to build a report containing a Pie 3D chart and explore chart configuration.

To add a chart to a report:

1. Create a new report using the Sugar CRM data source.
2. Use this query to display the count of orders in different countries:

```
select COUNT(*) as orders, shipcountry from orders group by shipcountry
```

3. Drag the fields from the Outline to the Detail band to create a small table of values to display in the chart.

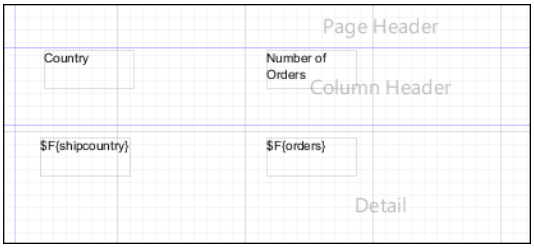


Figure 12-5 Initial Report Design

4. Expand the Summary band to 378 pixels.

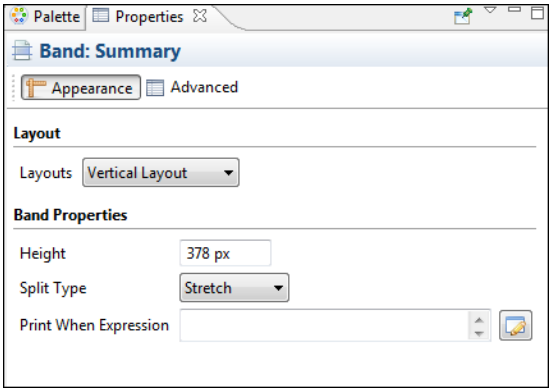


Figure 12-6 Summary Band Properties

5. Drag the Chart tool from the Palette into the Summary band. Jaspersoft Studio shows the Chart Wizard.

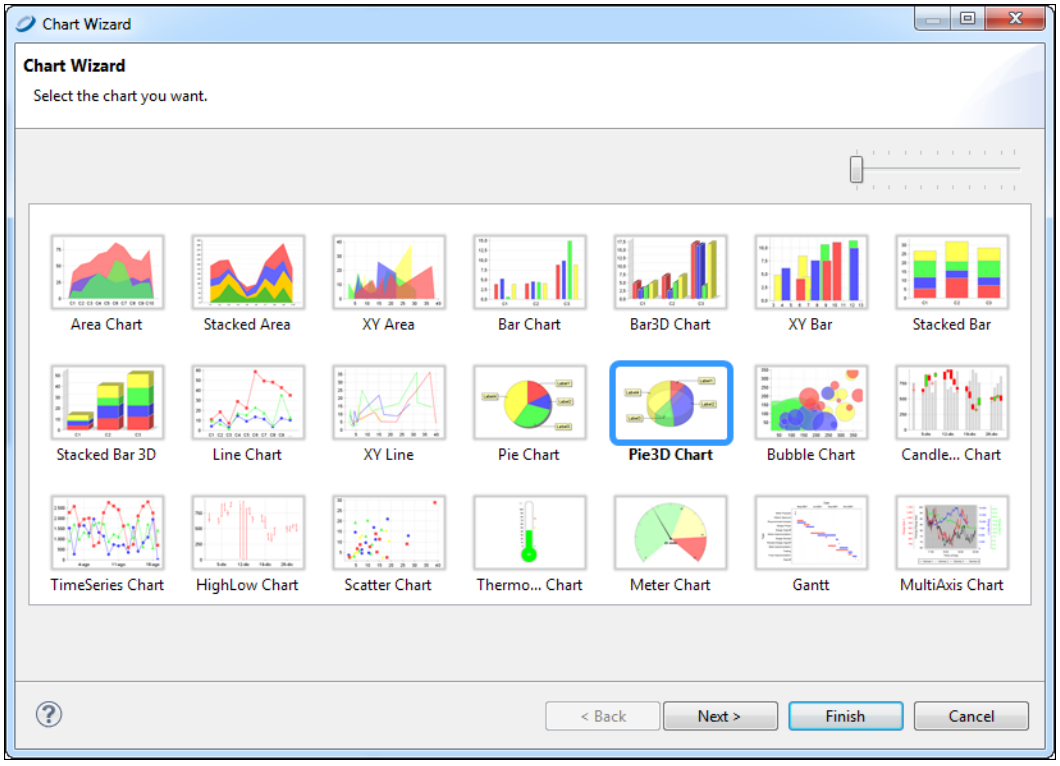


Figure 12-7 Chart Wizard

6. Select the **Pie 3D Chart** and click **Next**.
7. Accept the default configuration and click **Finish**.
8. Expand the chart to fit the Summary band.

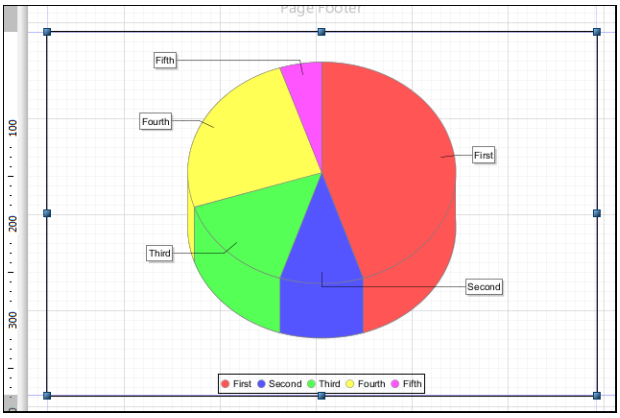


Figure 12-8 Chart in Summary Band

In Design view, the chart is a placeholder and doesn't display your data.

To configure a chart:

1. Double click the chart. The Chart Data Configuration window opens.

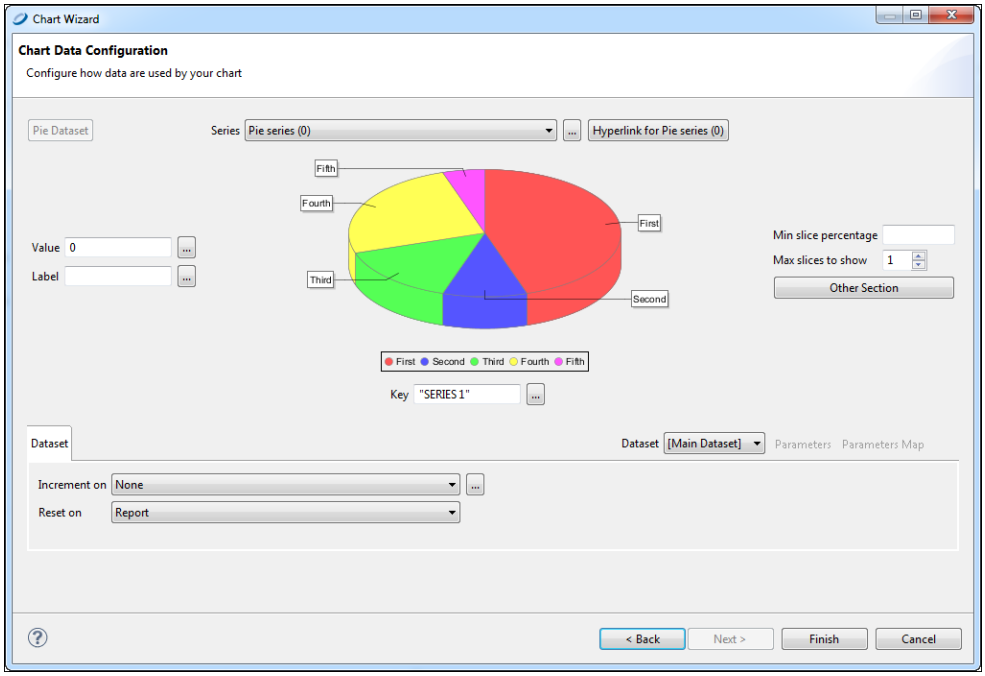


Figure 12-9 Chart Wizard - Chart Data Configuration

2. Select data to use in your chart.

The **Chart Data** tab shows the fields within the specified dataset. You'll find detailed descriptions of field types and their functionality in the *JasperReports Library Ultimate Guide*.

3. Set `10` for **Max slices to show** For a chart of many slices, this field specifies the number to show. A chart slice labeled Other contains the slices not shown.
4. On the **Dataset** tab, you can define the dataset within the context of the report.

   You can use the **Reset on** controls to periodically reset the dataset. This is useful, for example, when summarizing data relative to a special grouping. Use the **Increment on** control to specify the events that trigger addition of new values to the dataset. By default, each record of the chart's dataset corresponds to a value printed in the chart. You can change this behavior and force the engine to collect data at a specific time (for instance, every time the end of a group is reached).

   Set **Reset on** to **Report** since you don't want the data to be reset, and leave **Increment Type** set to **None** so that each record is appended to your dataset.

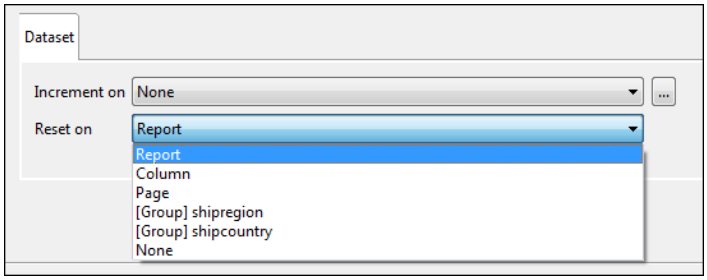


Figure 12-10 Dataset Tab

5. Also in the **Chart Data Configuration** dialog, enter an expression to associate with each value in the data source. For a Pie 3D chart, three expressions can be entered: `key`, `value`, and `label`.
   - **Key expression** identifies a slice of the chart. Each key expression must be a unique. Any repeated key simply overwrites the duplicate key. A key can never be `null`.
   - **Value expression** specifies the numeric value of the key.
   - **Label expression** specifies the label of a pie chart slice. This is the key expression by default.

   Next to each field, click the ... button. Enter the following:

   Value: `$F{orders}`

   Label: `$F{shipcountry}`

   Key: `$F{shipcountry}`
6. Click **Finish**.
7. Save your report, and preview it to see the result:

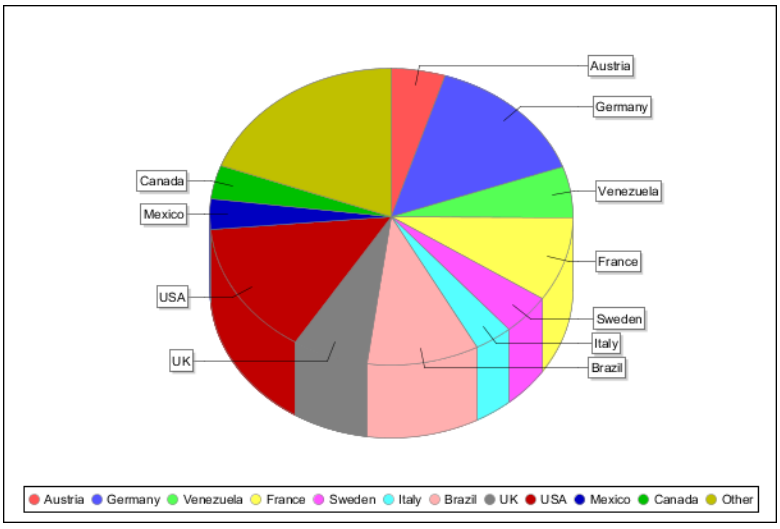


Figure 12-11 Final Chart

In this chart, each slice represents a country and the shipping total for that country.

12.3 Setting Chart Properties

In the Design view, the Properties tab enables you to define the appearance of your chart.

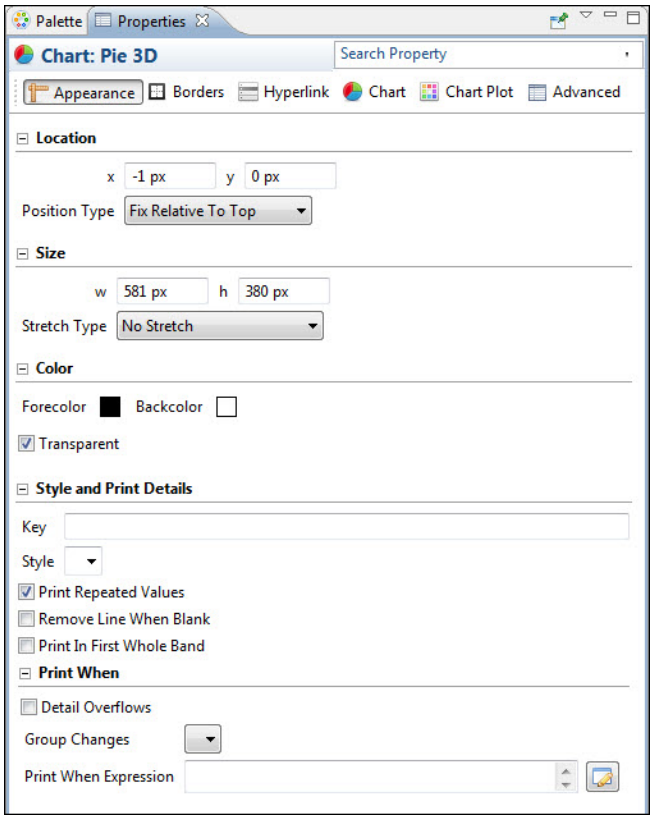


Figure 12-12 Properties View

For more information about setting hyperlinks, see **"Anchors, Bookmarks, and Hyperlinks" on page 45**.

Currently, JasperReports Server takes advantage of only a small portion of the capabilities of the `JFreeChart` library. To customize a graph, you must write a class that implements the following interface:

```
net.sf.jasperreports.engine.JRChartCustomizer
```

The only method available from this interface is the following:

```
public void customize(JFreeChart chart, JRChart jasperChart);
```

It takes a `JFreeChart` object and a `JRChart` object as its arguments. The first object is used to actually produce the image, while the second contains all the features you specify during the design phase that are relevant to the customize method.

12.4 Spider Charts

Spider charts (or radar charts) are two-dimensional charts designed to plot series of values over multiple common quantitative variables by providing an axis for each variable, arranged as spokes around a central point. The values for adjacent variables in a single series are connected by lines. Frequently the shape created by these lines is filled in with color.

In Jaspersoft Studio, spider charts are separate from the rest of the charts available in the Community Project, because they're a separate component in JasperReports Library. But you use them just as other JFree Charts.

To create the report for the chart:

1. Open a new, blank report using a landscape template and the Sugar CRM database. Use the following query:

   ```
   select * from orders
   ```

   Click **Next**.
2. Move all the fields into the right box. Click **Next** and **Finish**.
3. Delete all bands except **Title** and **Summary**.
4. Drag a **Static Text** element into the title bar, and name your report something like “Employee Orders by Month and Country”.
5. Enlarge the Summary band to 350 pixels by changing the **Height** entry in the **Band Properties** view.

To create a spider chart:

1. Drag the **Spider Chart** element from the **Palette** into your report.

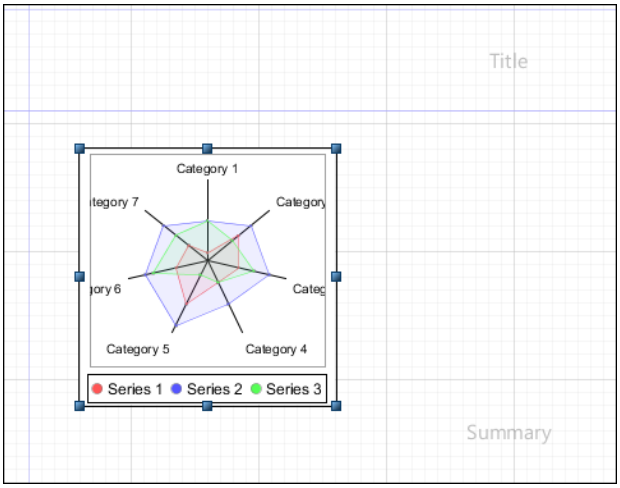


Figure 12-13 Spider Chart

2. Select the report in the Outline view, and in the Properties tab, click the **Edit query, filter, and sort options** button. The Dataset and Query dialog opens.

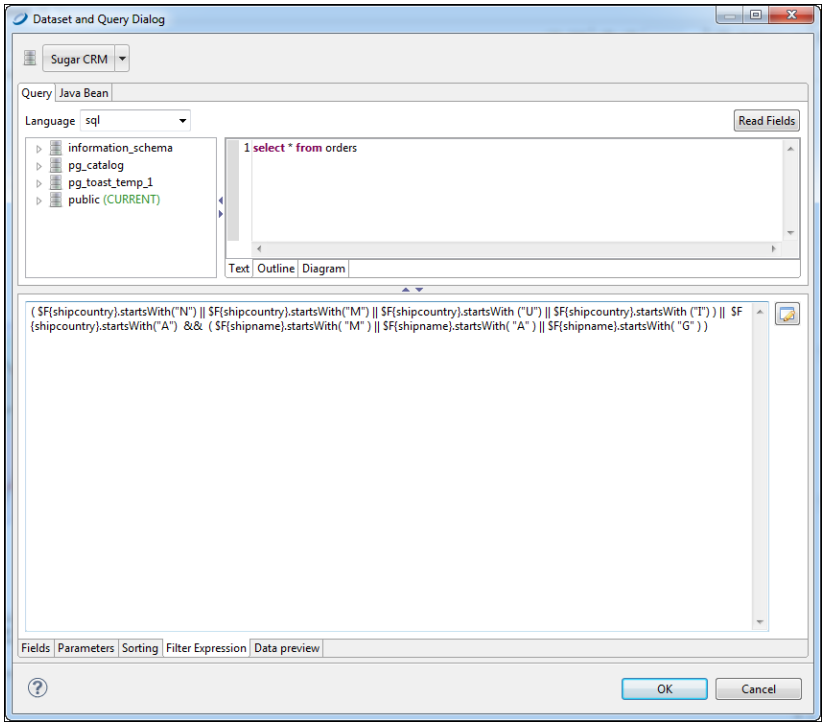


Figure 12-14 Filter Expression for Spider Chart

To filter your data in a Spider Chart, you must filter it in the **Dataset and Query** dialog.

3. To filter data for a more readable chart, click the **Filter Expression** tab and enter the expression below with no manual line breaks, then click **OK**.

```
( $F{shipcountry}.startsWith("N") || $F{shipcountry}.startsWith("M") || $F
{shipcountry}.startsWith ("U") || $F{shipcountry}.startsWith ("I") ) || $F
{shipcountry}.startsWith("A") && ( $F{shipname}.startsWith( "M" ) || $F
{shipname}.startsWith( "A" ) || $F{shipname}.startsWith( "G" ) )
```

4. Double-click the chart to display the Chart Data Configuration dialog.
5. Click the **Series** button and create the series `$F{employeeid}`.
6. Click the **Value** button and create the value `MONTH($F{orderdate})`.
7. Click the **Category** button and create the category `$F{shipcountry}`.
8. Click **Finish**.

To customize the look of your chart:

1. Single-click your chart, and click the **Properties** tab.
2. Click the **Legend** category and select **True** in the **Show Legend** drop-down menu.
3. Use the **Position** drop-down, to move the legend to the left.
4. Drag a Static Text element just to the left and above the legend. Label the legend **Employee Number**.
5. Save your report. The Design view should look like the following figure.

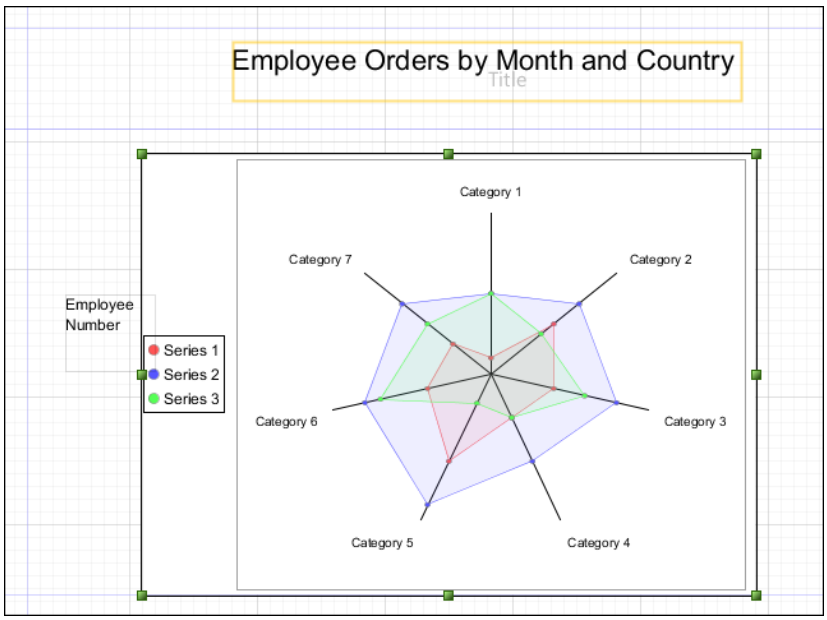


Figure 12-15 Spider Chart Design

6. Preview your report. It should look like the following:

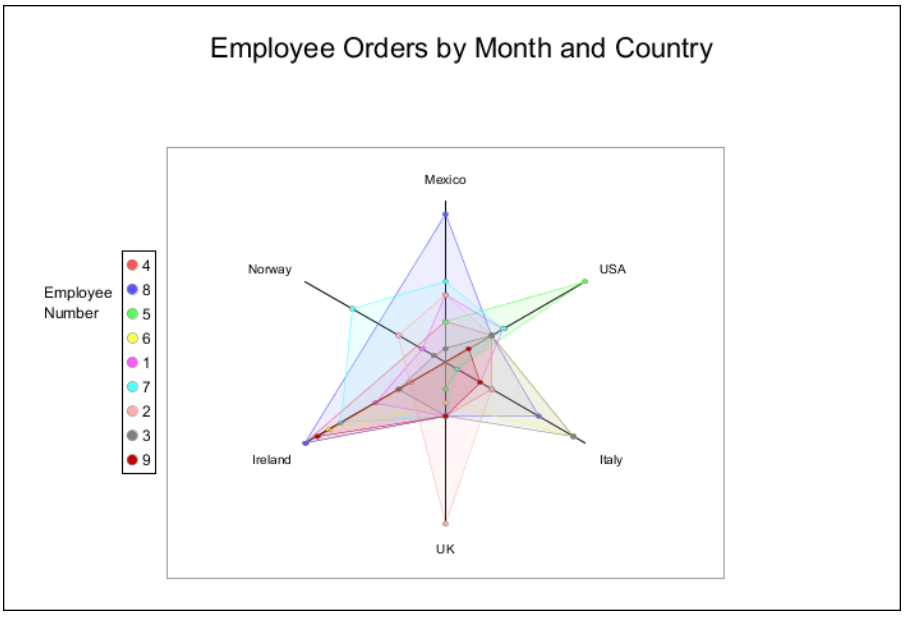


Figure 12-16 Spider Chart Preview

12.5 Chart Themes

Chart themes give you full control over the style of your charts. You can create a chart theme and use it in multiple reports for a uniform look. And you update that look for all those reports simply by updating the chart theme. Jaspersoft Studio is the supported tool for creating chart themes.

To create a JasperReports chart theme:

1. Select **File > New > Other** to open the Select a Wizard window.
2. Select the **Chart Themes** wizard and click **Next** to open the New Chart Theme Wizard.

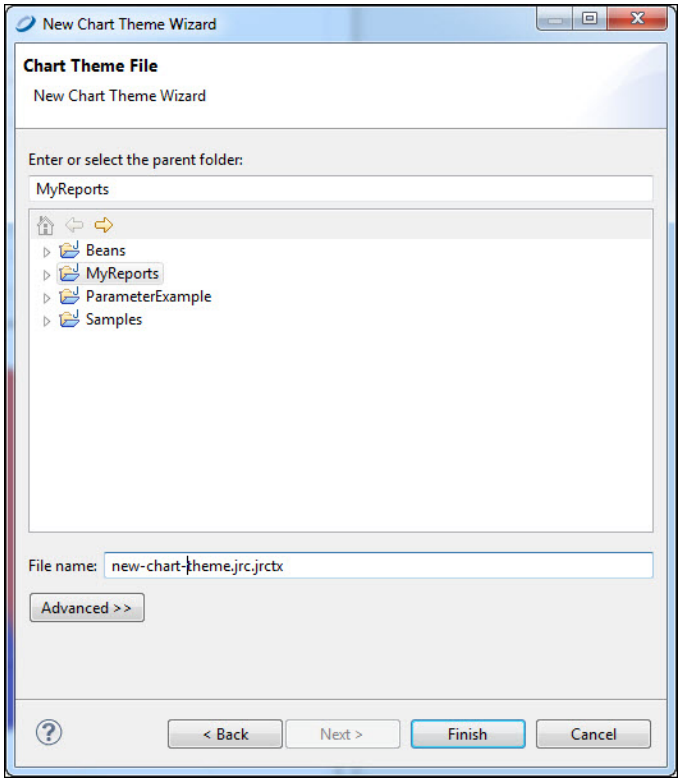


Figure 12-17

3. Select a folder and enter a file name for your theme. The file extension must be `.jrctx`.
4. Click **Finish**. The Chart Theme Designer opens.

12.5.1 Using the Chart Theme Designer

Click the Outline tab and expand the Chart Theme list to view the subsections of a theme design.

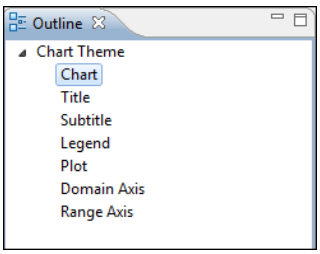


Figure 12-18 Chart Theme Designer Outline View

- **Chart**: Set properties for borders, color, background, and padding.
- **Title**: Set properties for position, color, alignment, padding, and font.
- **Subtitle**: Set properties for subtitles. These can be set to `INHERITED` on the **Advanced** tab.
- **Legend**: Specify whether to show a legend and, if so, its configuration.
- **Plot**: Set properties for label rotation, foreground, orientation, colors, image alignment, padding, grid lines, chart outline, series stroke and colors, and display and tick label fonts.
- **Domain Axis**: Set properties for elements along the domain axis.
- **Range Axis**: Set properties for elements along the range axis.

Your settings are applied to all chart types. If you want to see one of the chart types close up, click that chart. Click the close up view to all charts.

12.5.2 Editing Chart Theme XML

You can see and edit the chart theme XML from the **Source** tab. However, the XML appears as one long line. It is better paste it into a text editor for editing.

12.5.3 Creating a JasperReports Extension for a Chart Theme

To be used in a report, the `.jrtx` file must be exported to a JasperReports extension JAR file

To export the theme as a JAR:

1. On the **Preview** tab, click the  button. The Save As window opens.

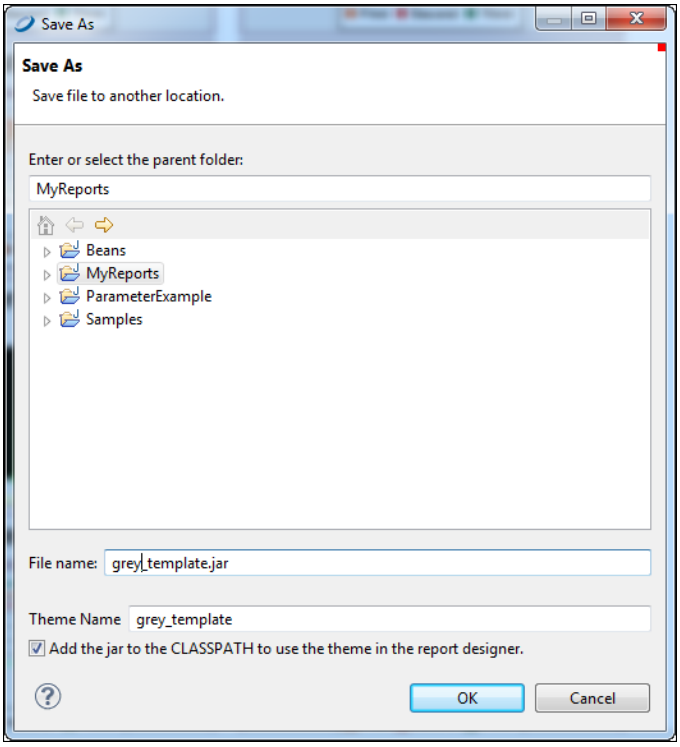


Figure 12-19 Save As JAR

2. Enter or select the parent folder, name the file, and name your theme. Click **OK**. A dialog appears, indicating that the Chart Theme was generated.

12.5.4 Applying a Chart Theme

Add the theme to your classpath to make it available in the list of themes.

CHAPTER 13 WORKING WITH TIBCO GEOANALYTICS MAPS

Jaspersoft Studio leverages TIBCO GeoAnalytics Maps to produce data-rich maps. This section describes their set-up and configuration, including:

- **Configuring a Basic Map**
- **Using Expressions for Properties**
- **Understanding Layers**
- **Working with Markers**
- **Working with Paths**

This section describes functionality that can be restricted by the software license for JasperReports Server. If you don’t see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

In addition to the other types of map component that Jaspersoft Studio supports, TIBCO GeoAnalytics Maps are also supported. These multi-layer maps are designed for use in interactive web environments, and support both markers and paths. They also support the ability to provide a street address and resolve it to the correct latitude and longitude (sometimes called geolocation).

Because these components download content from either TIBCO's service or from Google Maps, they require a connection to the Internet. While the maps themselves are freely available, using the GeoAnalytics geolocation serve to resolve street addresses requires an additional license.

These maps are well suited to web-based environments, such as HTML export or when viewed through an interactive viewer such as JasperReports Server; however, limitations in the underlying technology prevent some TIBCO GeoAnalytics Map features from working in static formats, such as PDF. In this case, the map is converted to an image, which is always downloaded from Google Maps instead of TIBCO's server. In addition, if the map's location is resolved from a street address, the canvas may be blank (or blue); this happens when the address's latitude and longitude aren't available.

If your target output format is something other than HTML, consider using the standard map component.

The map component consists of three layers: a map, a set of paths, and a set of markers. The lowest layer contains the map itself, rendered by your choice of providers: TIBCO Maps or Google Maps. In both cases, the image is formed of tiles retrieved from a remote server. The next two layers (first paths then markers) can contain paths and markers or shapes.

Figure 13-1 Basic structure of the TIBCO GeoAnalytics Map component

13.1 Configuring a Basic Map

To create a TIBCO Map component:

1. First locate the component in the Palette; it uses this icon:  ;drag it onto the canvas.

At a minimum, the TIBCO Map component requires the location of the area to display, which can be defined by these manually-exclusive options:

a. The latitude and longitude of the location.

b. The street address of the location (assuming you have a license for TIBCO GeoAnalytics geolocation services). To use this option, you must also provide credentials for TIBCO's geolocation service. You can either enter these in the Maparama Credentials section of the TIBCO map component's properties, or by defining them in the jasperreports.properties file so that they can shared across multiple reports. These properties are:

- `com.jaspersoft.jasperreports.tibco.maps.customer` - the customer name used with TIBCO GeoAnalytics Maps
- `com.jaspersoft.jasperreports.tibco.maps.key` - the corresponding license key for the specified user

2. To define a location, edit the TIBCO Maps component's Location properties. Entering a latitude/longitude pair or address defines a static location. You can also use parameters to dynamically define the components location as well as all other TIBCO Map properties.

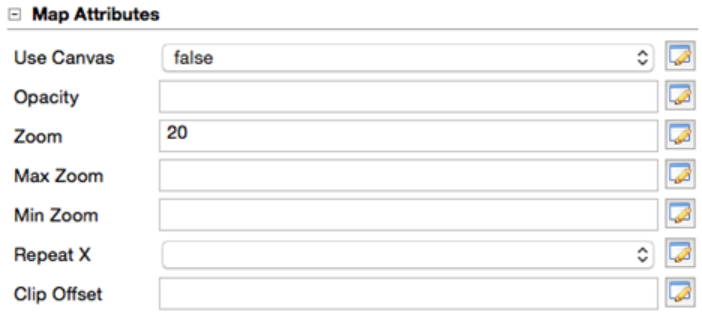


Figure 13-2 TIBCO Map Attributes

Map attributes determine how the map layer of the component is rendered. The attributes are all optional:

Property	Property Value
Use Canvas true	false This property refers to the way the map is rendered (by using a canvas or SVG layers)
Opacity	0.0- 1.0 Level of opacity of the map.
Max Zoom 1 - 18	The maximum allowed zoom

Min Zoom 1 - 18	The minimum allowed zoom
Repeat X true | false	Specifies whether tiles are repeated when the world's bounds are exceeded horizontally
Clip Offset integer	The clip offset of the map

13.2 Using Expressions for Properties

The simplest way to define the map layer's properties is to set them to static values; however, this is a much more limited approach than using expressions to pass parameters to the component dynamically, which allows you to evaluate data in your data set and use the results to populate the map layer's properties. If you don't specify a different data set, the component uses the main dataset of the report. In the components properties, properties based on expressions are indicated by displaying *f(x)* next to the field, as shown below.

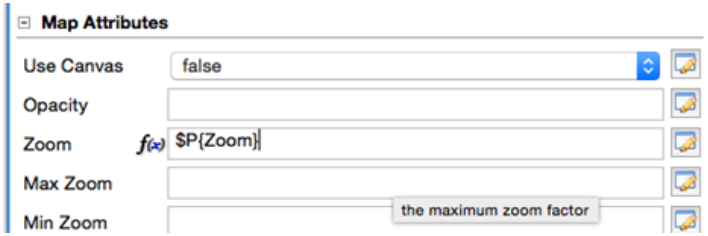


Figure 13-3 Map properties showing Zoom defined by an expression

To specify a different dataset to resolve the map attributes based on expressions, click the **Use Dataset** check box to select it, and select the dataset to use in the Dataset Run.

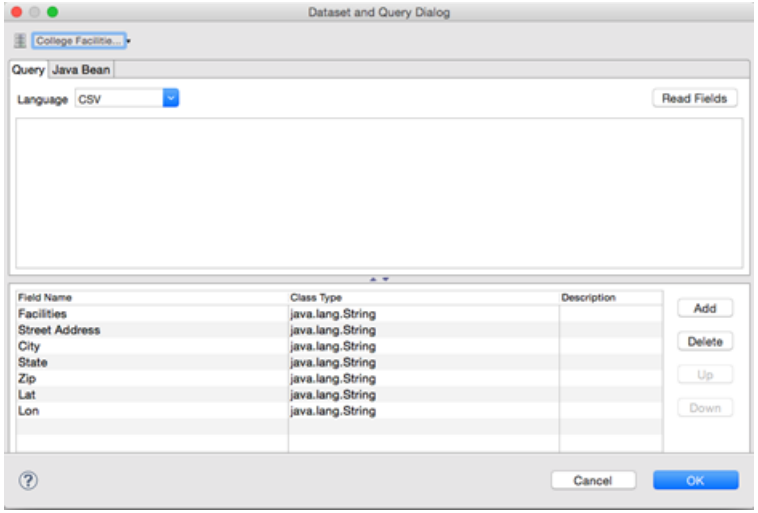


Figure 13-4 Defining the Dataset to use to resolve map attribute expressions

Data Runs are used throughout Jaspersoft Studio and its related products when a report includes a subdataset. Use a Data Run to define values for the subdataset's parameters.

13.3 Understanding Layers

Each layer in the map component controls different aspects of the final map rendered in your report:

- The maps layer defines the map tiles that are displayed by the component's image, which are determined by its location and zoom, the maximum and minimum zoom allowed in the component, and the image's opacity.
- The marker layer defines locations on the map that display an image you select.
- The path layer defines lines between locations on the map.

Each layer can be named uniquely; these names can be displayed in the JasperReports Serverinteractive report viewer in the Layers drop down; this allows your user to select which layers to drawn.

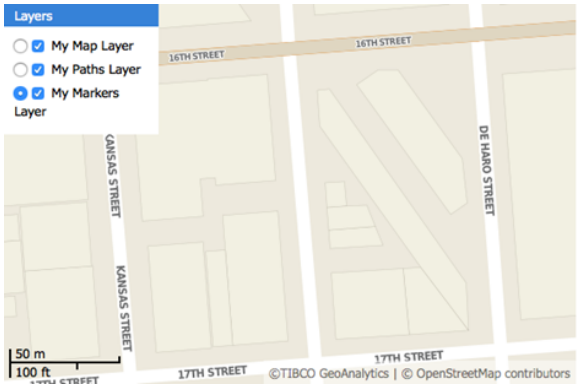


Figure 13-5 Layer names defined in the component can control the layers drawn in the final report

13.4 Working with Markers

Markers are points rendered on the second layer of the TIBCO Map component.

This section describes:

- **Static Markers**
- **Dynamic Markers**

13.4.1 Static Markers

To define a marker in Jaspersoft Studio:

1. Edit the map component's properties.
2. On the Markers tab, click Add.
3. Define your markers by specifying a location and icon. The list of properties for a marker includes:
   - target
   - string
   - optional
   - _blank

- the hyperlink target for the marker

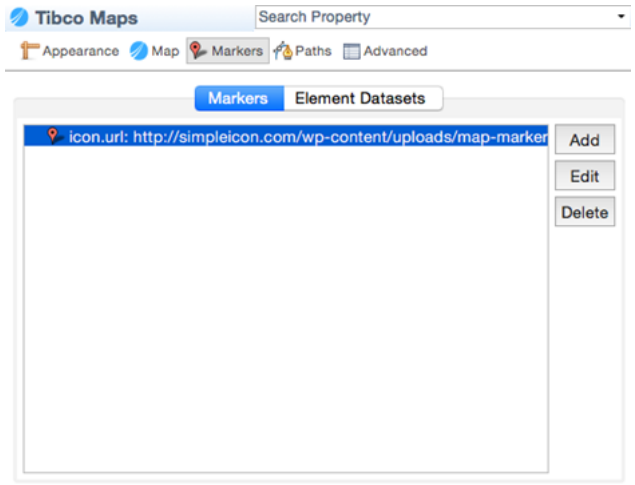


Figure 13-6 Defining a map's markers

4. Specify the icon as a URL that points to the image to use; it's loaded by the JavaScript API.
   Jaspersoft doesn't currently support loading an image directly from the repository, or as a resource local to the report.
   The location can be set by latitude/longitude coordinates or an address to be geolocated, as described above.

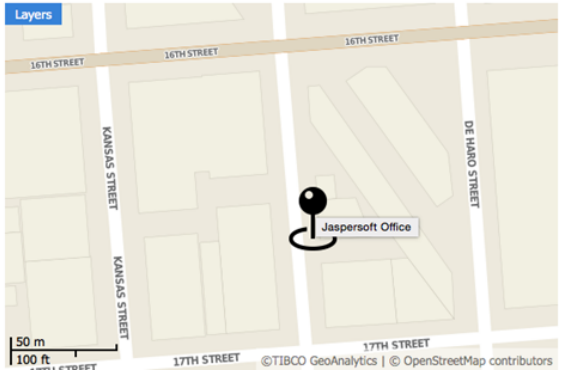


Figure 13-7 A map with a marker

5. For the addresses, set each property to form the address: country, state, zip, city, street.

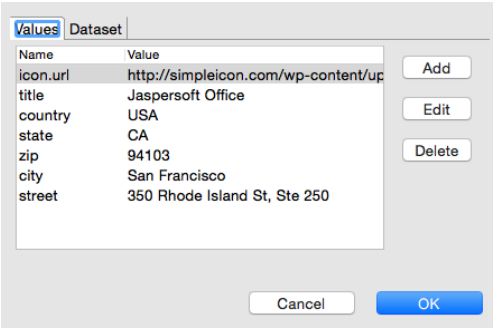


Figure 13-8 Marker properties set to a static location

Marker properties include:

Property Name	Type	Required?	Possible Values	Description
xoffset	Numeric (integer)	Optional	0	The horizontal offset of the marker icon measured in pixels
yoffset	Numeric (integer)	Optional	0	The vertical offset of the marker icon measured in pixels
anchor	String	Optional	bottom-left, bottom-right, bottom-center	The anchor point of the marker icon
draggble	Boolean	Optional	false	Specifies whether users can drag the marker
icon.url	String	Required	N/A	URL for the icon
title	String	Optional	N/A	The ToolTip for the marker icon; works in conjunction with icon.url
hyperlink	String	Optional	N/A	The hyperlink text for the marker
target	String	Optional	N/A	The hyperlink target for the marker. The default value is _ blank.

This is a simplified example; the more common scenario is to read location data from the database.

13.4.2 Dynamic Markers

The steps above define the marker as a static address known when the report was created. But it is far more useful to dynamically define the markers based on locations defined in your report's data. A single map can use both static and dynamic marker, and locations can be based on data from more than one data source.

In this example, we'll use data from City College of San Francisco's public facilities data set that we've saved as an Excel file:

Facilities	Street Address	City	State	Zip	Latitude	Longitude
Airport	SF International Airport, North Access Road, Building 928	San Francisco	CA	94128	37.622511278000445	-122.39519978799973
Civic Center	750 Eddy Street	San Francisco	CA	94109	37.783008003000475	-122.42000284399973
Castro	450 Castro Street	San Francisco	CA	94114	37.76168744600045	-122.43511354199973
Chinatown/North Beach	808 Kearny Street	San Francisco	CA	94108	37.79551773600048	-122.40496557999973
Downtown	88 4th Street	San Francisco	CA	94103	37.784588004000454	-122.40438877299971
Evans	1400 Evans Avenue	San Francisco	CA	94124	37.74169579700049	-122.38589540299972
Fort Mason	Laguna Street & Marina Boulevard, Building B	San Francisco	CA	94123	37.80001344200048	-122.43517818299972
John Adams	1860 Hayes Street	San Francisco	CA	94117	37.77385843900049	-122.4469528999997
Mission	1125 Valencia Street	San Francisco	CA	94110	37.754794183000456	-122.42088522899968
Ocean	50 Phelan Avenue	San Francisco	CA	94112	37.72408746400049	-122.45231737299969
Southeast	1800 Oakdale Avenue	San Francisco	CA	94124	37.73684564000047	-122.39424548999972
Gough Street	31/33 Gough Street	San Francisco	CA	94103	37.772268634000454	-122.42098248799971

Since the data set includes both street addresses and latitude/longitude pairs, we can explore both functions.

To use dynamic locations:

1. Create an Excel file with the data provided above and a data adapter that points to it. Export the data adapter to the project folder; name it CollegeFacilities.xml.
2. In the report, create a new dataset: right-click the root in the outline view and select **Create Dataset**.

3. Right-click the new dataset and select **Dataset and Query**.
4. In the **Query** dialog, select the CollegeFacilities.xml data adapter and click **Read Fields**.
5. Use the data adapter to populate the dataset. With the CollegeFacilities dataset selected in the outline view, click the **Advanced** tab in the Properties view, then select the property **Properties** and click the button ellipsis to open the properties dialog.
6. Add a new property: `net.sf.jasperreports.data.adapter`, and specify the name of the data adapter file saved earlier (CollegeFacilities.xml).

   We can use this new dataset to set markers on the map.
7. Select the map in the design view, click the **Markers** tab, and click **Add**.
8. Click **Dataset**, check the **Use Dataset** check box, and click **Add**.
9. Select the CollegeFacilities dataset and accept the defaults. Studio uses the data adapter referenced by the `net.sf.jasperreports.data.adapter` property set previously for this dataset.
10. Click **OK**.

    The dataset is added to the list of datasets we'll use for markers.
11. Click **Values** and create an expression for each marker property: for example, provide the title, street, city, state, country, and so forth.

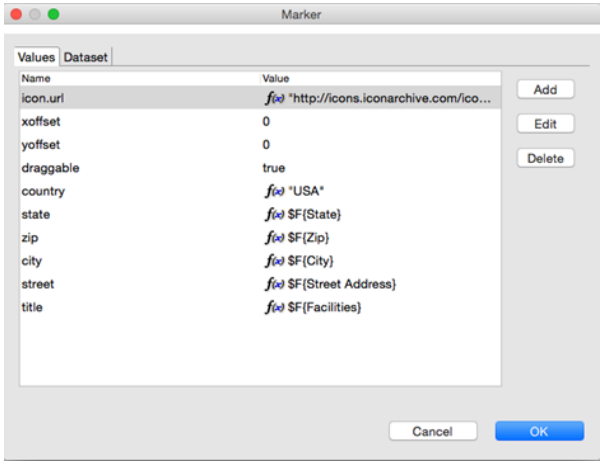


Figure 13-9 Location values defined as expressions

This example uses an icon from the web: Pink Push Pin.

12. Click **OK**.

13. Preview your report in HTML.

Figure 13-10 San Francisco City College facilities marked on a map

13.5 Working with Paths

Paths are lines rendered on the third layer of the TIBCO Map component.A path is defined by:

- A name that serves as a path identifier in case different paths will appear on the map
- A style that specifies various style configuration properties, such as line and fill color, line weight, and opacity
- A collection of places (points) on the map defined by latitude/longitude coordinates or addresses; these are connected to form the path

To define a path in Jaspersoft Studio:

1. On the Paths tab, use the Styles section to define a style to associate with the path: click Add to do so.
   Style properties can be added manually or by specifying a dataset. The style name sets the style property when adding points to the path.
2. Use the Paths section to add points to the path: click **Add** in this UI area to do so. For each point, specify:
   a. the path name (to identify which path includes the point)
   b. the style property (to identify the style associated with this path)
   c. the latitude/longitude coordinates or the address of the point

Like styles, points can be added manually or by using a specific dataset. Pay close attention when adding points: they are connected on the map in the order that they are declared in the JRXML file. If they aren't declared in a sensible order, the path won't make sense, either.

CHAPTER 14 ADDITIONAL CHART OPTIONS IN COMMERCIAL EDITIONS

All editions of Jaspersoft Studio include basic charting functionality in the form of JFree Charts. Commercial editions provide more advanced HTML5 charts implemented through Highcharts.

Written in pure HTML5 and JavaScript, Highcharts offer sophisticated, interactive charts that animate automatically. Jaspersoft Studio currently supports many of the charts available in the Highcharts library.

This section describes functionality that can be restricted by the software license for Jaspersoft Studio. If you don't see some of the options described in this section, your license may prohibit you from using them. To find out what you're licensed to use, or to upgrade your license, contact Jaspersoft.

This chapter has the following sections:

- **Overview of HTML5 Charts**
- **Simple HTML5 Charts**
- **Scatter Charts**
- **Dual-Axis, Multi-Axis, and Combination Charts**

For details about charts implemented through JFreeCharts, see **"Working with Charts" on page 179**.

14.1 Overview of HTML5 Charts

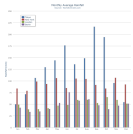
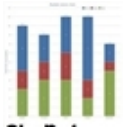
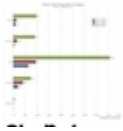
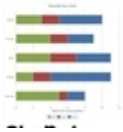
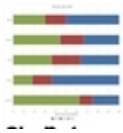
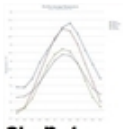
You can create interactive reports with HTML5 charts. And they're more attractive than the basic charts available in Jaspersoft Studio. This section shows how to build a report containing a simple HTML5 chart and how to change chart types and edit charts.

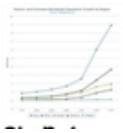
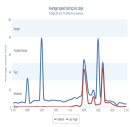
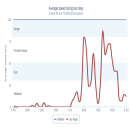
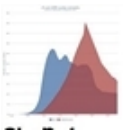
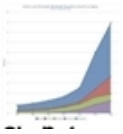
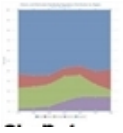
In Jaspersoft Studio HTML5 Charts work like Ad Hoc charts. Levels, for example, are the equivalent of what you would use the Ad Hoc Slider to see. Other explanations include:

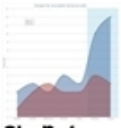
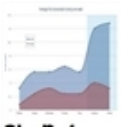
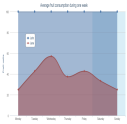
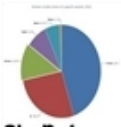
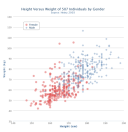
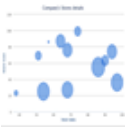
- **Values**: Static properties.
- **Expressions**: Dynamic properties.
- **Categories**: Rows. In a pie chart, the categories are the slices.
- **Measures**: The same as in Ad Hoc. In a pie chart, these are the size of the slice.
- **Series Contributors**: In the Design view, these are defined at the measure level. In JRXML these are defined as Series.

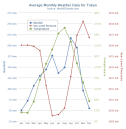
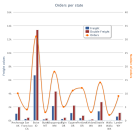
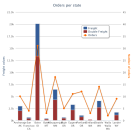
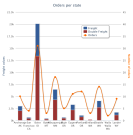
Before you add a chart to your report, consider the best way to display your data. The following table describes the available chart types.

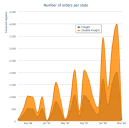
Table 14-1 HTML5 Chart Types

Icon	Description
Column charts - Compare values displayed as columns	
	**Column**. Multiple measures of a group are depicted as individual columns.
	**Stacked Column**. Multiple measures of a group are depicted as portions of a single column whose size reflects the aggregate value of the group.
	**Percent Column**. Multiple measures of a group are depicted as portions of a single column of fixed size representing 100% of the amounts for a category. Used when you have three or more data series and want to compare distributions within categories and at the same time display the differences between categories.
Bar charts - Compare values displayed as bars	
	**Bar**. Graphically summarize and display categories of data to let users easily compare amounts or values among categories.
	**Stacked Bar**. Multiple measures of a group are depicted as portions of a single bar whose size reflects the aggregate value of the group.
	**Percent Bar**. Multiple measures of a group are depicted as portions of a single bar of fixed size representing 100% of the amounts for a category. Used when you have three or more data series and want to compare distributions within categories and at the same time display the differences between categories.
Line charts - Compare values displayed as points connected by lines	
	**Line**. Displays data points connected with straight lines, typically to show trends.

Icon	Description
	**Spline**. Displays data points connected with a fitted curve. Allow you to take a limited set of known data points and approximate intervening values.
	**Stacked Line**. Displays series as a set of points connected by a line. Values are represented on the y-axis and categories are displayed on the x-axis. Lines do not overlap because they are cumulative at each point.
	**Stacked Spline**. Displays series as a set of points connected with a fitted curve. Values are represented on the y-axis and categories are displayed on the x-axis. Lines do not overlap because they are cumulative at each point
	**Percent Line**. A variation of a line chart in which each series adjoins but does not overlap the preceding series.
	**Percent Spline**. A variation of a spline chart in which each series adjoins but does not overlap the preceding series.
Area charts - Compare values displayed as shaded areas. Compared to line charts, area charts emphasize quantities rather than trends.	
	**Area**. Displays data points connected with a straight line and a color below the line; groups are displayed as transparent overlays.
	**Stacked Area**. Displays data points connected with a straight line and a solid color below the line; groups are displayed as solid areas arranged vertically, one on top of another.
	**Percent Area**. Displays data points connected with a straight line and a solid color below the line; groups are displayed as portions of an area of fixed sized, and arranged vertically one on top of the another.

Icon	Description
	**Area Spline**. Displays data points connected with a fitted curve and a color below the line; groups are displayed as transparent overlays.
	**Stacked Area Spline**. Displays a series as a set of points connected by a smooth line with the area below the line filled in. Values are represented on the y-axis and categories are displayed on the x-axis. Areas do not overlap because they are cumulative at each point.
	**Percent Area Spline**. A variation of area spline charts that present values as trends for percentages, totaling 100% for each category.
Pie charts - Compare values displayed as slices of a circular graph	
	**Pie**. Multiple items of a single group are displayed as sectors of a circle.
	**Dual-Level Pie**. A variation of pie charts that present grouped values in two concentric circles; the inner circle represents the coarsest grouping level in the data. In Jaspersoft Studio, note these rules about data configuration for dual-level pie charts: • Only one measure is displayed (the first) • The last row level is rendered as the outer pie • The next to last row level is rendered as the inner pie; if only one row level is defined, the inner pie consists of a single section representing the total
Scatter and Bubble Charts - Show the extent of correlation, if any, between the values of observed quantities.	
	**Scatter**. Displays a single point for each point in a data series without connecting the points.
	**Bubble**. Compares the relationships between three measures displayed on the x-y axis. The location and size of each bubble indicates the relative values of each data point

Icon	Description
Multi-Axis Charts - Compare trends in two or more data sets whose numeric range differ greatly.	
	**Multi-Axis Column**. A column chart with two series and two axis ranges.
	**Multi-Axis Line**. A line chart with two series and two axis ranges.
	**Multi-Axis Spline**. A spline chart with two series and two axis ranges.
Combination Charts - Display multiple data series in a single chart, combining the features of a area, bar, column, or line charts.	
	**Column Line**. Combines the features of a column chart with a line chart.
	**Column Spline**. Combines the features of a column chart with a spline chart.
	**Stacked Column Line**. Combines the features of a stacked column chart with a line chart.
	**Stacked Column Spline**. Combines the features of a stacked column chart with a line chart.

Icon	Description
Time Series Charts - Illustrate data points at successive time intervals. Also called Fever Chart.	
	**Time Series Area**. Displays data points over time connected with a straight line and a color below the line.
	**Time Series Area Spline**. Displays data points over time connected with a fitted curve and a color below the line.
	**Time Series Line**. Displays data points over time connected with straight lines.
	**Time Series Spline**. Displays data points over time connected with a fitted curve.
Spider Charts - Display data in line or data bars arranged on a circular spider web chart. Also called a Radar Chart.	
	**Spider Column**. Plots one or more series over multiple common quantitative variables by providing axes for each variable arranged as spokes around a central point. The column variation of spider charts displays values as bars that extend out from the central point towards the edges of the circular web. The bar's length indicates the relative value.
	**Spider Line**. Plots one or more series over multiple common quantitative variables by providing axes for each variable arranged as spokes around a central point. The line variation of spider charts displays values as points arranged around the circular web. The data points are joined by a line. Each point's distance from the central point indicates the relative value.
	**Spider Area**. Plots one or more series over multiple common quantitative variables by providing axes for each variable arranged as spokes around a central point. The area variation of spider charts is similar to the line variation, but the shape defined by the line that connects each series' points is filled with color.

14.2 Simple HTML5 Charts

Before you add a chart, consider the best way to display your data. HTML5 Chart Types can help you decide.

14.2.1 Creating an HTML5 chart

1. Create a new report using the SugarCRM data source in the JasperReports Server repository. See **"Accessing JasperReports Server from Jaspersoft Studio" on page 77**.
2. Start with the query:
   ```
   select * from orders
   ```
3. Choose **HTML5 Charts** from the **Components Pro** section of the **Palette**, and drag it into your report.

Figure 14-1 Palette

4. Select a chart type based on the information you want to display. See**Table 14-1** for help.
   For this example, we'll choose **Pie chart**.

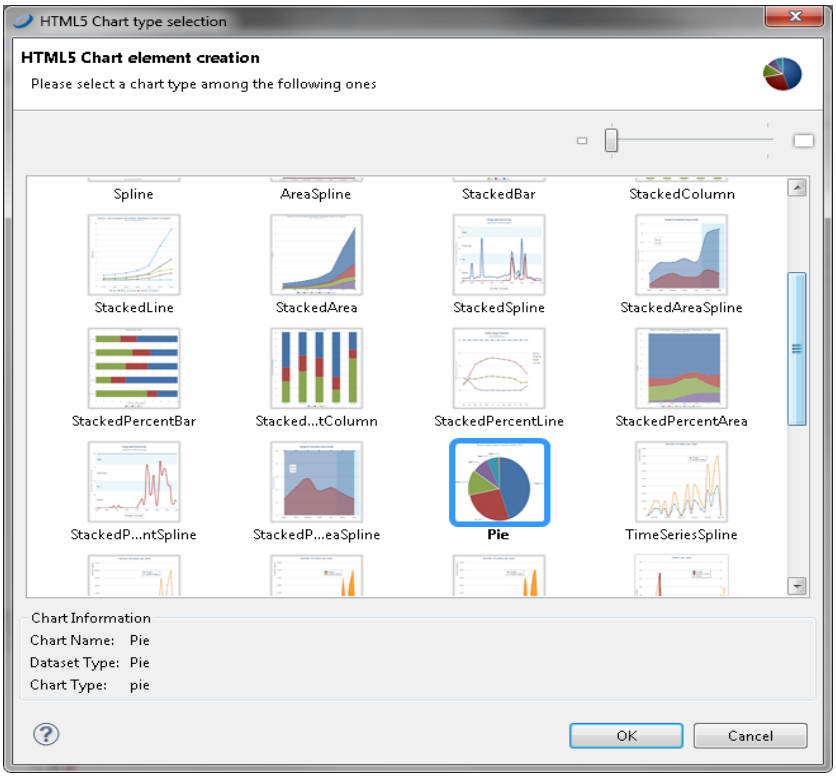


Figure 14-2 Chart Types

Your report now includes a sample chart. The Design view doesn't display live data. After you configure your data for use with an HTML5 chart, you can click **Preview** to see the chart with your data.

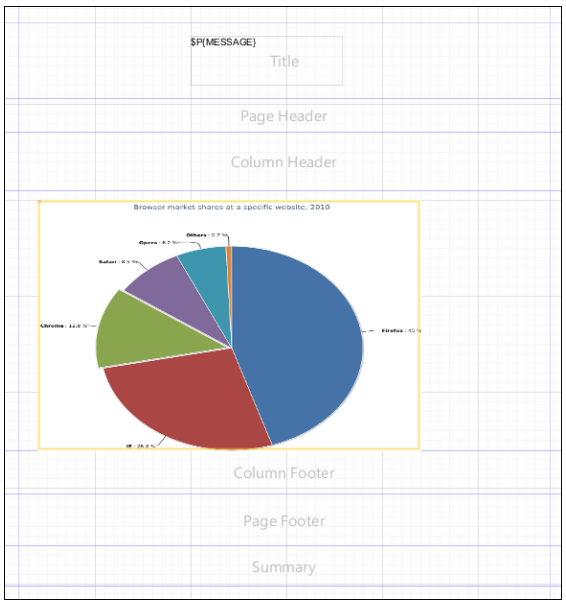


Figure 14-3 Pie Chart Example

5. Right-click the chart, and choose **Edit Chart Properties**. The Chart Properties dialog includes tabs for configuring chart properties, chart data, and hyperlinks.

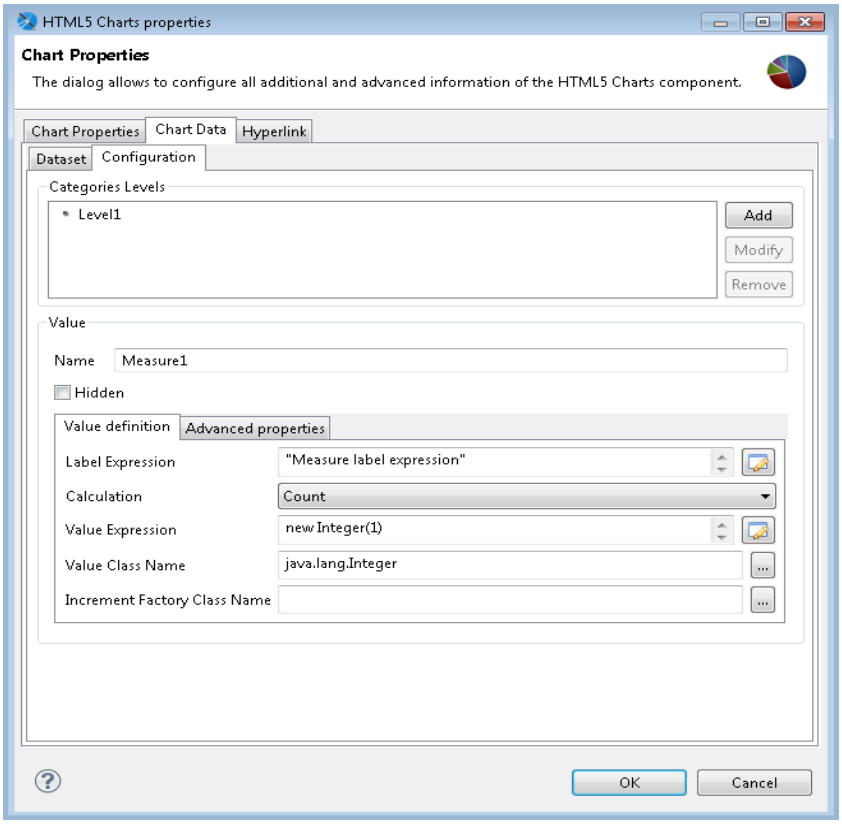


Figure 14-4 HTML5 Charts Properties > Chart Data > Configuration

6. Go to the **Chart Data** tab and click the **Configuration** tab.
7. Highlight **Level1**, and click **Modify**.
8. Name the Expression `Country`.
9. Click  to open the Expression Editor and change the default to the `shipcountry` field. Click **OK**.
10. In the **Values** field, create a measure called **Total Orders**.
    - The **Label Expression** should be `"Total Orders"`.
    - The **Calculation** type is `DistinctCount`.
    - In the **Value Expression** field, add the `orderid` field.
    - The **Value Class Name** should be `java.lang.Integer`.
11. Click **OK** to close the **Chart Properties** dialog.
12. Save, then click the **Preview** tab to see your chart. To preview your chart in a web browser, choose XHTML Preview from the Preview drop-down. HTML5 charts require XHTML for web preview.

    Your HTML chart preview is interactive. Hover over a pie segment to see the exact number of orders.

14.2.2 Editing HTML5 Charts

Continue using the HTML5 pie chart from the previous example to complete the following tasks.

To edit the title of an HTML5 chart:

1. Right-click the chart and select **Edit Chart properties**.
2. Click the **Chart Properties** tab.
3. Click **Title** and enter `Orders by Country` in the text box. You can also customize alignment, color, and font.

To filter the data in the chart:

1. Click the **Chart Data** tab, then click its **Dataset** tab.
2. By the Filter expression text box, click  to ope the Expression Editor.
3. Filter your data set by adding a Filter expression such as:

```
$F{SHIPCOUNTRY}.startsWith("A") ||
$F{SHIPCOUNTRY}.startsWith ("U") ||
$F{SHIPCOUNTRY}.startsWith ("I") ||
$F{SHIPCOUNTRY}.startsWith ("S")
```

4. Click **Finish**.
5. Save and click **Preview** again to view the edited chart.

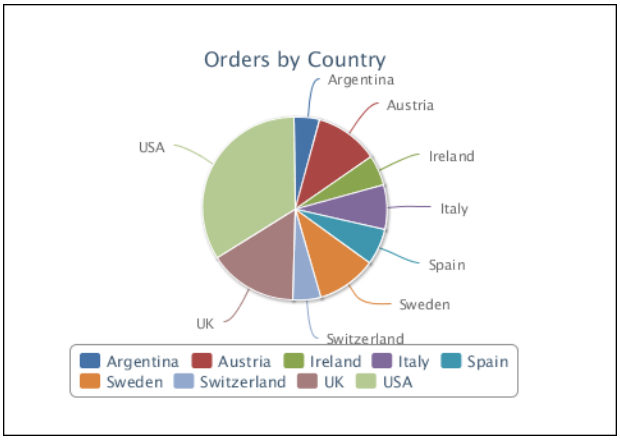


Figure 14-5 Filtered Pie Chart

To change chart type:

Before beginning this task, make sure your data is appropriate for the new chart type.

1. Right-click your chart and choose **Edit Chart properties**.
2. In the lower-left section of the dialog box, click the **Change Chart Type** button to open the **HTML5 Chart type selection** window. This window provides some guidance about which dataset type is used by each type of chart.
3. For this example, choose **Bar** and click **OK**.

   Jaspersoft Studio discards data that's not right for the new chart type.
4. Click **OK**.
5. Save and preview your report. Because it was originally designed as a pie chart, there's no label for the Y-values.

To add an additional measure to a bar chart:

Editing works similarly for all charts, depending on the type of data presented.

1. On the **Design** tab, right-click your chart and choose **Edit Chart properties**.
2. Click the **Chart Data** tab, then click its **Configuration** tab. Because this is now a bar chart, there are sections to define Series Levels and additional measures.
3. In the Measures section, click **Add**. Enter the following information:
   - **Name**: `Average Freight`
   - **Label Expression**: `"Average Freight"`
   - **Calculation**: `Average`
   - **Value Expression**: `$F{FREIGHT}`
   - **Value Class Name**: `java.lang.Integer`

   Click **OK** and Close.
4. Save and Preview the chart. HTML charts are interactive.

To add a series to a bar chart:

1. On the **Design** tab, right-click your chart and choose **Edit Chart properties**.
2. Click the Chart Data tab, then click its **Configuration** tab.
3. In the Series Levels section, click **Add**. Enter the following information:
   - **Name**: `Year`
   - **Expression**: `new java.text.SimpleDateFormat("yyyy").format( $F{ORDERDATE} )`
   - **Value Class Name**: `java.lang.Comparable`
   - **Order**: `Ascending`

   Click **OK** and Close.
4. Save and Preview your report. You see two measures per year, Total Orders and Average Freight, grouped by Country.

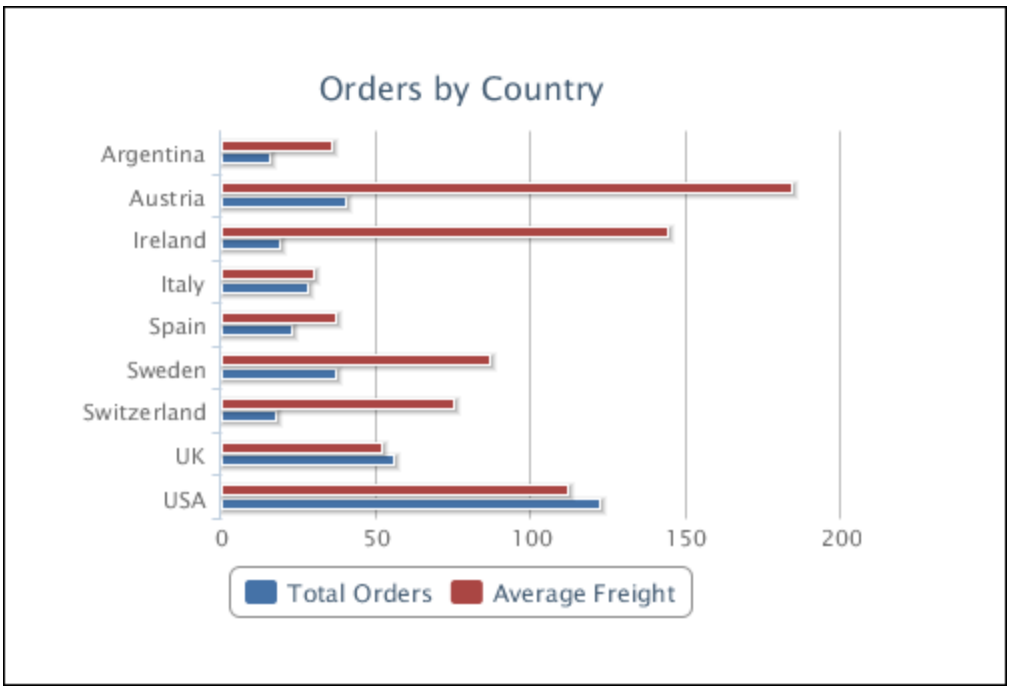


Figure 14-6 Bar Chart with Second Measure and Series

14.2.3 Creating Hyperlinks

With HTML5 charts (Highcharts) functionality added to JasperReports, using hyperlinks in charts is similar to using JFreeCharts (regular Charts) and Fusion charts (Charts Pro). You open the chart properties, select the **Hyperlink** tab, and add your link.

Because HTML5 charts are so different from other chart types supported by Jaspersoft Studio,their chart items (bars or pie section) support hyperlinks differently than other types of chart. This section gives some basic steps for defining hyperlinks for HTML5 charts. For more detailed instructions, see **3.10.1, "Creating Hyperlinks in HTML5 Charts," on page 52**.

1. Right-click your bar chart and select **Edit Chart properties**.
2. Click the **Chart Data** tab, then click its **Configuration** tab.
3. In the Categories Levels section, double-click **Country**.
4. Click the **Bucket Properties** tab, and click **Add**.
5. In the **Property name** text entry box, enter `url`.
6. Click  and add the following expression:

```
"http://www.ask.com/web?q=" + $F{SHIPCOUNTRY}
```

7. Click **OK**.

To add the hyperlink:

1. Back in the **Chart Data** tab, click its **Configuration** tab.
2. Double-click the measure **Total Orders**.

3. Click the **Advanced Properties** tab, then click the **Hyperlink** button. That provides a shortcut to the two most important properties needed to create a hyperlink.
4. Double-click **hyperlinkReference (SeriesItemHyperlink)**. The **Edit property** dialog opens with some information filled in. You customize this information to the values you need.
5. Click the radio button for **Use a bucket property value** and enter `Country.url`.
6. Click **OK**.
7. Save and preview your report. In the HTML preview, click the bar for any country to open the Ask.com page for that country.

To create hyperlinks that use expressions:

Bucket-level properties are the only place where you can create expressions to be used with your link. If you're not using an expression, you must either use static values or reference a bucket level property.

1. Identify the series you want to use for your link.
2. Create a bucket property for that series. For example, if you are using a pie chart, the category bucket may have a property called myUrl which contains your url built with the category value.
   This can be done in **Edit Chart Properties > Chart Data > Configuration**.
3. In the Categories Levels section, click **Add**.
4. Click the Bucket Properties tab, and click **Add**.
5. Create the myUrl property, then click **OK**.
6. Next, Identify the measure to which you need to add a link.
   This can be done in **Edit Chart Properties > Chart Data > Configuration**.
7. In the Measures section, click select the measure you want to link to, and click **Modify**.
8. In the advanced properties of your measure click **Hyperlink**.
   This creates a couple of properties. Edit them by assigning the proper value (again, you need to either use static values or reference a bucket level property).

To add a ReportExecution hyperlink:

If you want a ReportExecution hyperlink, you need to rename one of the measure's advanced properties (see step 4 in the previous procedure) `to _report` and enter the corresponding value, and if you need to pass parameter values you also need to add them as measure properties.

For more information about setting hyperlinks, see **"Anchors, Bookmarks, and Hyperlinks" on page 45**.

14.3 Scatter Charts

Scatter charts plot two or more data series as points. The charts are intended to show raw data distribution of the series. The data is represented as follows:

1. The first data series is represented as the x values.
2. The second data series is represented as correlated y values.
3. Any additional data series are plotted as y values correlated to the x values provided by the first series.

As a result, a scatter chart always displays one less plot than the number of data series included.

Now let's create a scatter chart that shows maximum and average freight in each city for each year.

To create the report for the chart:

1. Open a new, blank report using the SugarCRM database and the following query:
   ```
   select * from orders
   ```
   Click **Next**.
2. Move all the fields into the right box. Click **Next** then **Finish**.
3. Delete all bands except for **Title** and **Summary**.
4. Enlarge the Summary band to 500 pixels by changing the **Height** entry in the **Band Properties** view.

To create the chart:

1. Give your report a title like “Maximum and Average Freight in Years for City”.
2. Drag the **HTML5 Charts** element into the summary band, and select **Scatter**.
3. In the Outline view, right click the chart element, and choose **Edit chart properties**.
4. In **HTML5 Chart Properties > Chart Data > Configuration** create a Category with the following information:
   - **Name**: `ShipCountry`
   - **Expression**: `$F{SHIPCOUNTRY}`
   - **Value Class Name**: `java.lang.String`
   - **Order**: `Ascending`

   Click **OK**.
5. Create a second Category with the following:
   - **Name**: `ShipCity`
   - **Expression**: `$F{SHIPCOUNTRY}`
   - **Value Class Name**: `java.lang.String`
   - **Order**: `Ascending`

   Click **OK**.
6. Create a Series with the following:
   - **Name**: `Orderdate Year`
   - **Expression**: `YEAR($F{ORDERDATE})`
   - **Value Class Name**: `java.lang.Integer`
   - **Order**: `Ascending`

   Click **OK**.
7. Add a Measure for maximum freight:
   - **Name**: `Max Freight`
   - **Label Expression**: `“Max Freight”`
   - **Calculation**: `Highest`
   - **Value Expression**: `${freight}`
   - **Value Class Name**: `java.math.BigDecimal`

   Click **OK**.
8. Add a Measure for average freight:
   - **Name**: `Average Freight`
   - **Label Expression**: `“Average Freight”`
   - **Calculation**: `Average`
   - **Value Expression**: `${freight}`

- **Value Class Name**: `java.math.BigDecimal`

Click **OK**.

Your **Chart Properties** dialog should look like this:

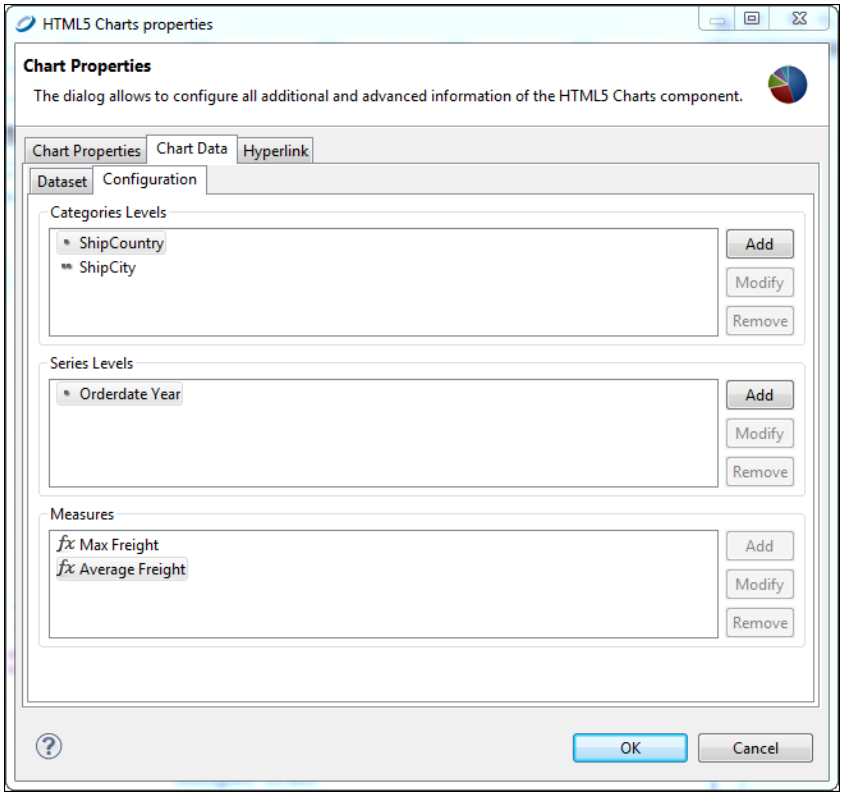


Figure 14-7 Scatter Chart Properties

To change the position or layout of the legend:

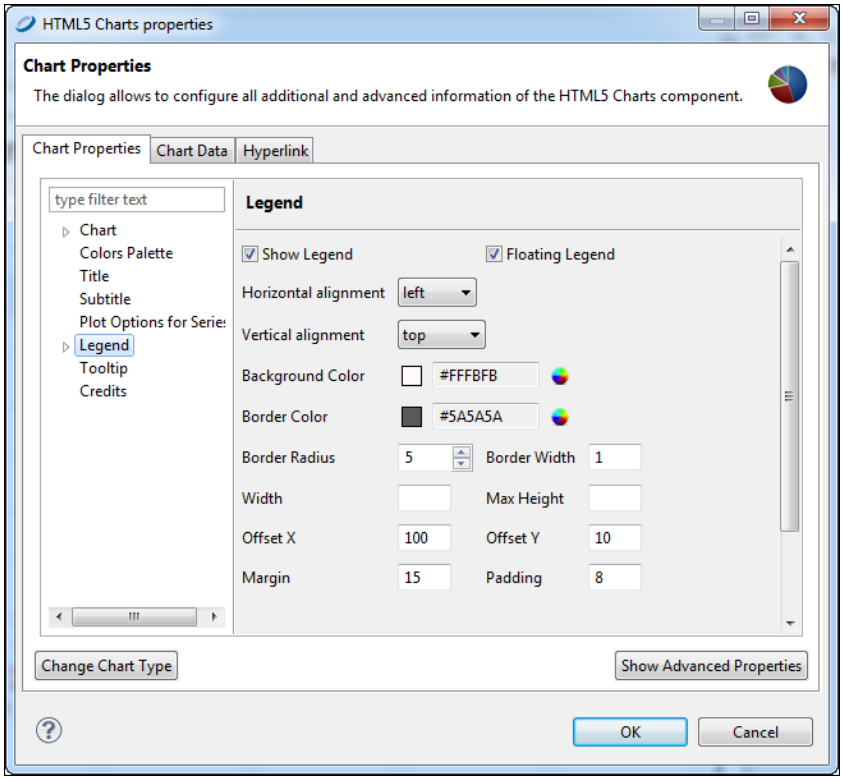


Figure 14-8 Legend Properties

1. Open **HTML5 Chart Properties** and click the **Chart Properties** tab.
2. Highlight **Legend**.
3. Check the boxes for **Show Legend** and **Floating Legend**.
4. Use the drop-down menus to arrange the legend to appear in the top left by setting Offset X and Offset Y to appear within the plot area.

   Click **OK**.

To add a border to the chart:

1. Open **HTML5 Chart Properties** and click the **Chart Properties** tab.
2. Click the arrow next to **Chart** and highlight **Borders and Plot Area**.
3. Create a border for your chart by using the arrows next to **Plot Border Width** or by typing the number of pixels.
4. Select a **Border Color**, **Border Radius**, and **Border Width**.
5. Click **OK**.

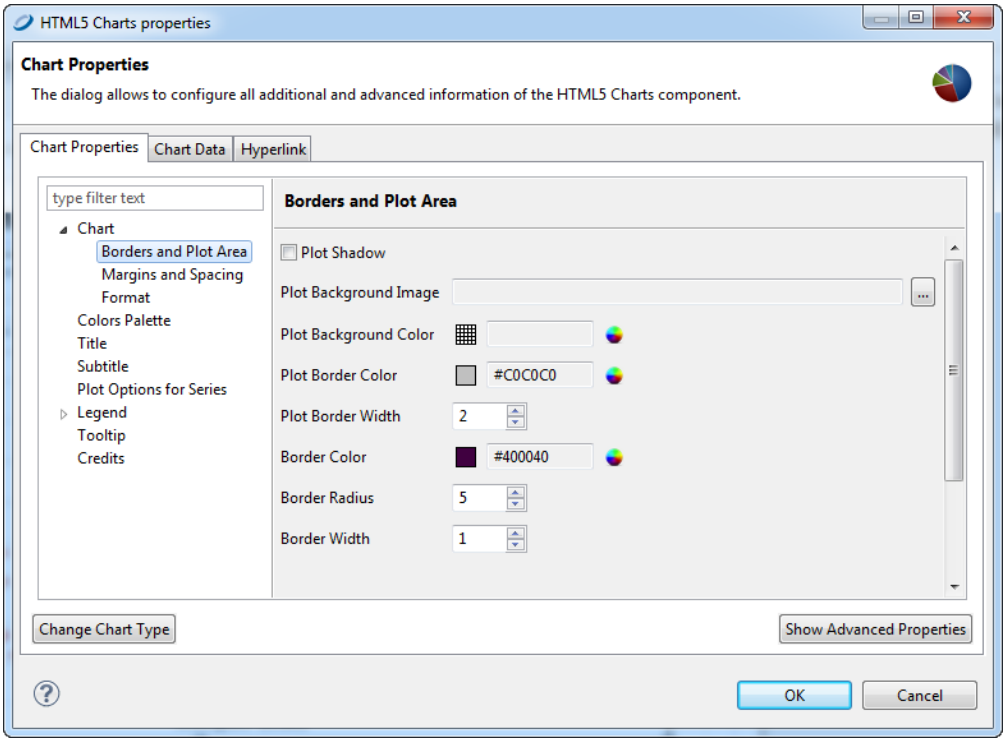


Figure 14-9 Border Properties

6. Save and preview your report. It should look like this:

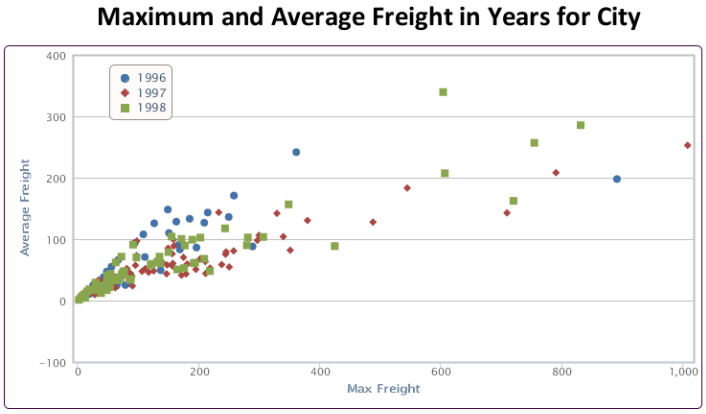


Figure 14-10 Scatter Chart

14.4 Dual-Axis, Multi-Axis, and Combination Charts

Dual- and multi-axis charts are useful when you want to plot multiple chart types on the same chart or use two different scales for each y-axis. This enables you to easily compare data items with very different scales.

Similarly, combination charts are used to display multiple data series in a single chart that combines the features of two different charts.

In this example, we create a column spline chart that plots freight and orders per country, per year.

To create the report for the chart:

1. Open a new, blank report using the SugarCRM database and the following query:

   ```
   select * from orders
   ```

   Click **Next**.
2. Move all the fields into the right box. Click **Next** and **Finish**.
3. Delete all bands except for **Title** and **Summary**.
4. Enlarge the Summary band to 500 pixels by changing the **Height** entry in the **Band Properties** view.

To create the chart:

1. Drag the **HTML5 Charts** element into the summary band, and select **ColumnSpline**.
2. In **HTML5 Chart Properties > Chart Properties > Chart Data > Configuration** create a Category with the following:
   - **Name**: `Country`
   - **Expression**: `$F{SHIPCOUNTRY}`
   - **Value Class Name**: `java.lang.String`
   - **Order**: `None`

   Click **OK**.
3. Create a Series with the following:
   - **Name**: `Year`
   - **Expression**: `new java.text.SimpleDateFormat ("yyyy").format( $F{ORDERDATE} )`
   - **Value Class Name**: `java.lang.Comparable`
   - **Order**: `Ascending`

   Click **OK**.
4. Add a Measure for freight:
   - **Name**: `Freight Values`
   - **Label Expression**: `"Freight"`
   - **Calculation**: `Count`
   - **Value Expression**: `${freight}`
   - **Value Class Name**: `java.lang.integer`

   Click **OK**.
5. Add a second Measure for orders:
   - **Name**: `Total Orders`
   - **Label Expression**: `"Total Orders"`
   - **Calculation**: `DistinctCount`
   - **Value Expression**: `$F{ORDERID}`
   - **Value Class Name**: `java.lang.integer`

   Click **OK**.
6. To see fewer countries in your chart, select **Edit Chart Properties > Chart Data > Dataset**.
7. Add a Filter expression such as:

   ```
   $F{SHIPCOUNTRY}.startsWith("A") ||
   $F{SHIPCOUNTRY}.startsWith ("U") ||
   ```

```
$F{SHIPCOUNTRY}.startsWith ("I")
```

Click **OK**.

8. To change the position or layout of the legend:

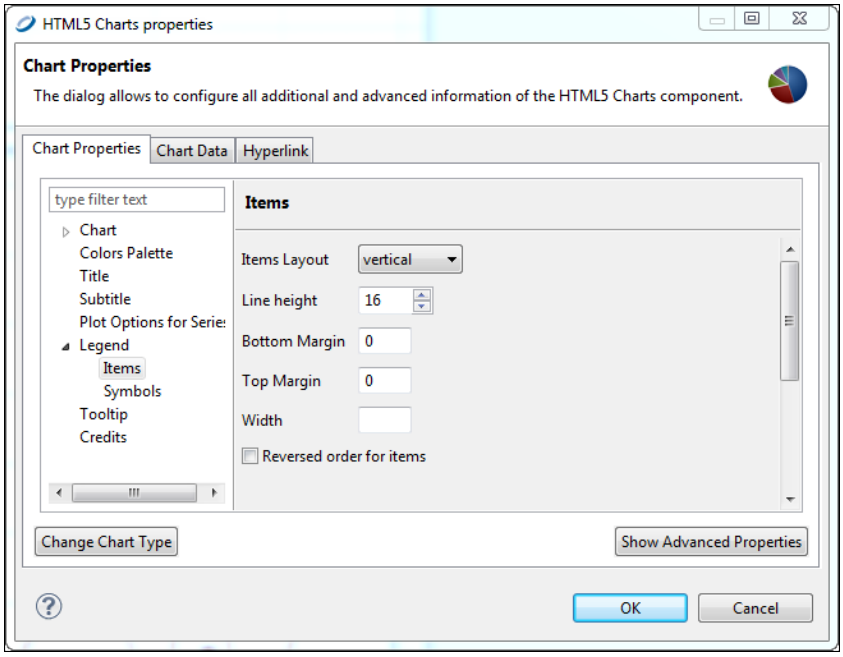


Figure 14-11 Legend Properties

   a. Go to **Chart Properties > Chart Properties**.
   b. Click the down arrow next to **Legend** on the left.
   c. Highlight **Items**.
   d. Arrange the **Items Layout** to be Horizontal.
   e. Click **OK**.

9. Save and preview your report. It should look like the following figure.

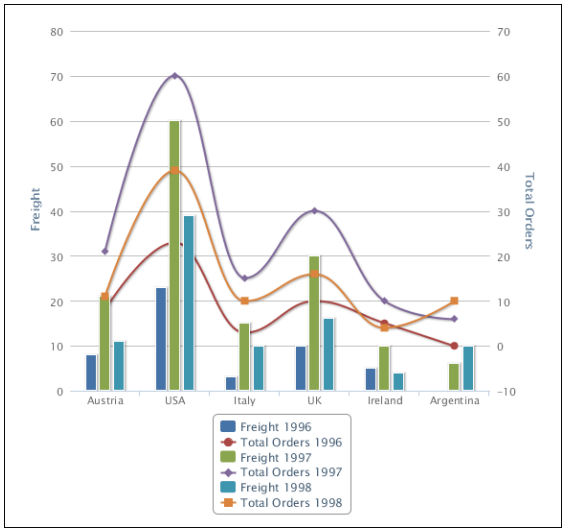


Figure 14-12 Column Spline Chart

CHAPTER 15 CREATING QUERIES

Jaspersoft Studio provides tools to help you define report fields and create a proper query (if a query is needed to fetch the report data). You'll find these tools in the **Dataset and Query** dialog.

It also provides a drag-and-drop query builder for easily creating SQL queries. This allows users who aren't familiar with SQL to quickly join tables and produce complex data filters and where conditions. SQL Builder also provides a way for skilled users to explore the database and list the metadata such as schemas and available tables.

This chapter contains the following sections:

- **Using the Dataset and Query Dialog**
- **Working with the Query Builder**

15.1 Using the Dataset and Query Dialog

Click the **Dataset and Query** icon .

When working with a sub-dataset, open the dialog by right-clicking the dataset name inside the Outline view, and selecting **Dataset and Query...**.

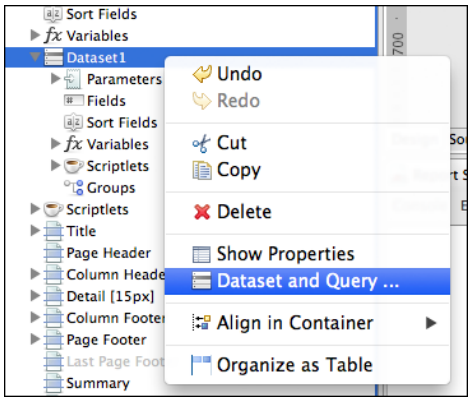


Figure 15-1 Right-Click Menu

Use the **Dataset and Query** dialog to:

- Select a data adapter with which to configure the dataset. Usually a data adapter is selected, but you can change it.
- Select a query language for the dataset you're editing. (This can be the main dataset or a sub-dataset that populates a chart or a table.)
- Enter a query. A tool is available for several languages including SQL, XPath and JSON.
- Retrieve the fields from the selected Data Adapter. These can be provided directly by the Data Adapter or by executing the query and reading the response metadata.
- Add, edit, or remove fields and parameters.
- Edit sort options.
- Provide an expression to filter dataset records.
- Preview your data, if supported by the selected data adapter.

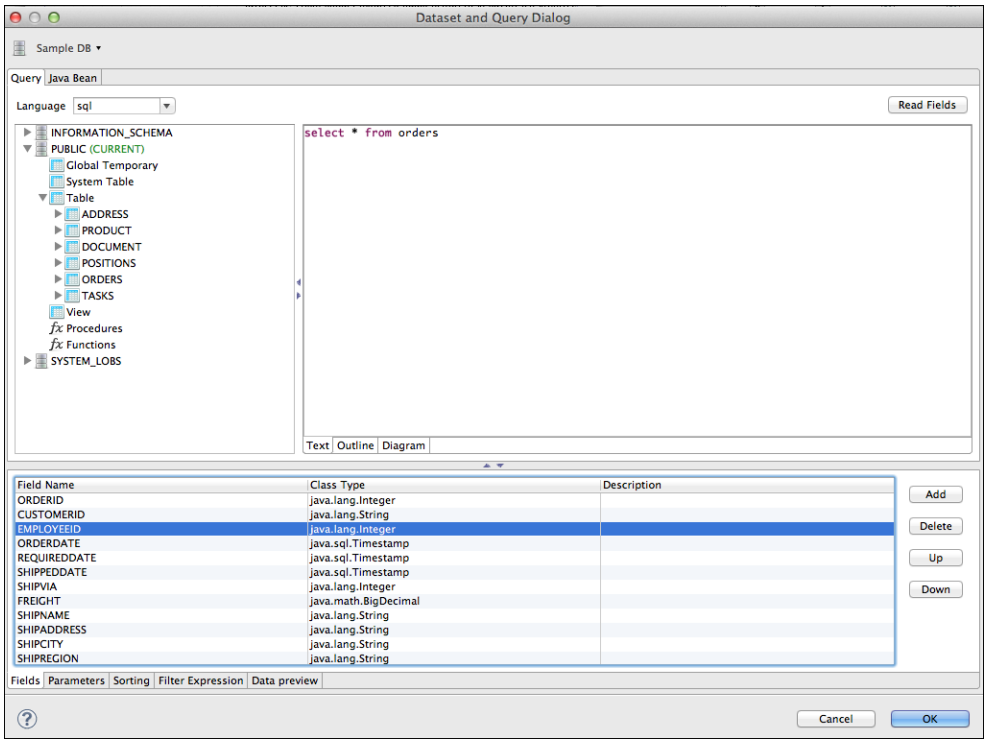


Figure 15-2 Data Preview Tab

Defining all the fields of a report by hand can be tedious. JasperReports requires all report fields to be named and configured with a proper class type. The Dataset and Query dialog simplifies the process by automatically discovering the available fields provided by a data adapter, without using a query. To execute a query you need to use the proper data adapter: for example to execute an SQL query you must use a JDBC data adapter. The **Read Fields** button starts the discovery process: the fields found are listed in the **Fields** tab and added to the report. Click **OK** to close this window.

Some query languages, like XPath, do not produce a result that resemble a table, but more complex structures that require extra steps to correctly map result data to report fields. An example of this is the multidimensional result coming from MDX results (MDX is the query language used with OLAP and XML/A connections): in this case each field is mapped by specifying an expression stored in the field description. For some languages,

such as instance XPath and JSON query, the Query dialog displays a tool that simplifies the creation of both the query and the mapping.

15.2 Working with the Query Builder

When your language is SQL, the Query Builder is called the SQL Builder. The tool requires a JDBC data adapter.

The builder has two parts. On the left a tree-view shows all the available schemas and relative objects like tables, views, etc. found by using the JDBC connection provided by the data adapter. On the right there are three tabs that present the query in different ways.

The first **Text** tab contains a text area for writing a query. You can drag tables and other objects from the metadata view into the text area, so you don't have to write the entire qualified names of those objects. Although the SQL builder doesn't support arbitrary complex queries (which may use database-specific syntax), this text area can be used for any supported query, including stored procedures if supported by the report query executer.

If a query has already been set for the report, this text area shows it when the query dialog is open.

Use the **Outline** and **Diagram** tabs to visually build the query. The current version of Jaspersoft Studio doesn't support back-parsing of SQL. For this reason, you should use Outline and Diagram editing mode only to create new queries, otherwise the new query replaces any existing query. If you do attempt to overwrite an existing query, Jaspersoft Studio alerts you.

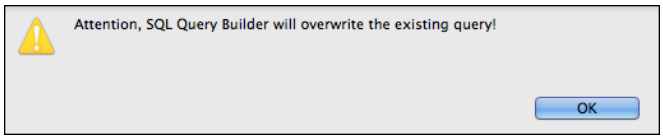


Figure 15-3 Query Overwrite Warning

15.2.1 Query Outline View and Diagram View

The purpose of SQL is to select data from the tables of the database. SQL allows you to join tables, so that you can get data from more than one table at a time.

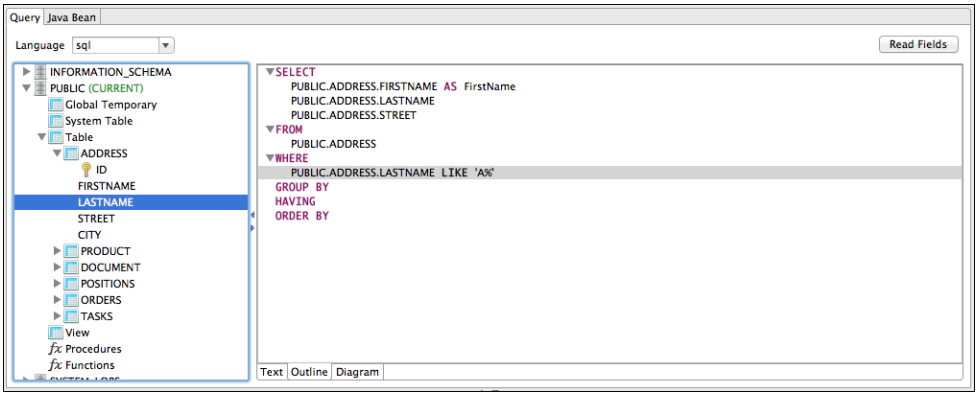


Figure 15-4 Outline View

The **Outline** view is a good tool for people with a basic understanding of SQL. It works in conjunction with the **Diagram** view, which represents the simplest way to design a query. In the outline view the query is split in its main parts introduced by the relative keyword. Those parts include:

`SELECT` introduces the portion of query where are listed all the columns that form a record of the final result set (which is basically a list of records).

`FROM` contains the list of tables involved in our queries and if specified the rules to join that tables.

`WHERE` is the portion of query that describes the filters and conditions that the data must satisfy in order to be part of the final result, conditions can be combined by using the `OR` and `AND` logical operators and can be aggregated by using parentheses.

`GROUP BY` indicates a set of fields used to aggregate data and is used when an aggregation function is present in the `SELECT`. For example, the following query counts the number of orders placed in each country.

```
SELECT
        count(*) as number_of_orders,
        Orders.country
FROM
        Orders
GROUP BY
        Orders.country
```

`HAVING` works in a i bit like `WHERE`, but it's used with aggregate functions. For example the following query filters the records by showing only the countries that have at least 40 orders:

`ORDER BY` specifies a set of columns for sorting the result set.

```
SELECT
        count(*) as number_of_orders,
        Orders.country
```

```
FROM

        Orders

GROUP BY

        Orders.country

HAVING

        count(*) > 40
```

The `Diagram` view shows the tables in the query with the relative join connections and the selected fields. This view is very useful especially to select the relevant fields and easily edit the table joins by double-clicking the connection arrows.

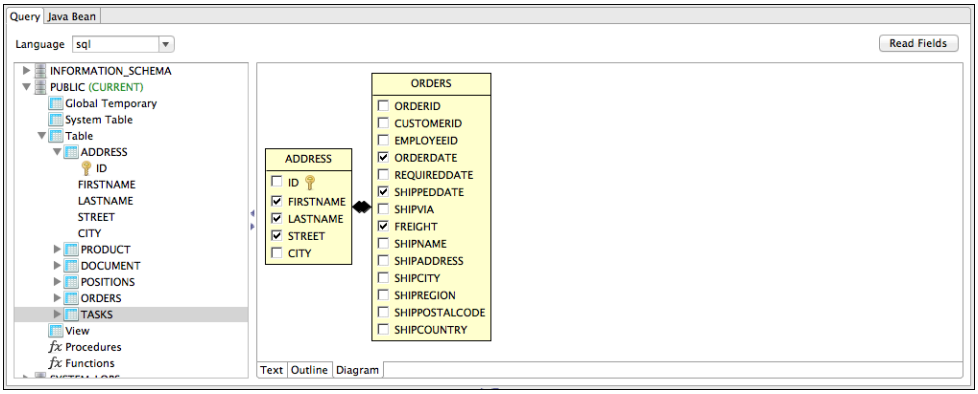


Figure 15-5 Diagram View

15.2.2 Selecting Columns

You can drag columns from the database explorer into the `SELECT` node or other nodes in the outline view. Make sure the columns you select are from tables present in the `FROM` part of your query.

You can also select fields by their check boxes in the diagram view.

Finally you can add a column by right-clicking the **SELECT** node in the outline view and selecting **Add Column**. A new dialog prompts you to pick the column from the those in the tables mentioned in the `FROM` clause. If a column you want to add is not a table column, but a more complex expression, for instance an aggregation function like `COUNT(*)`, add it by right-clicking the `SELECT` node in the outline view and selecting **Add Expression**.

When a column is added to the query as part of the `SELECT` section, you can set an alias for it by double-clicking it.

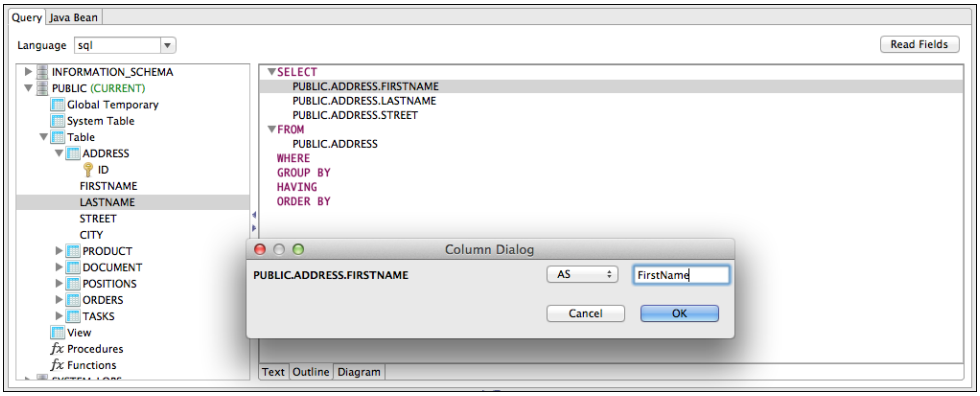


Figure 15-6 Setting an Alias

Aliases are useful when you have several fields with the same name coming from two different tables.

15.2.3 Joining Tables

You can join tables you've added by selecting shared fields. You can create the relationship the Diagram view by dragging a column of the first table onto the column of the table you're joining to. You can edit this type of join by double-clicking the join arrows. Currently a single join condition is supported.

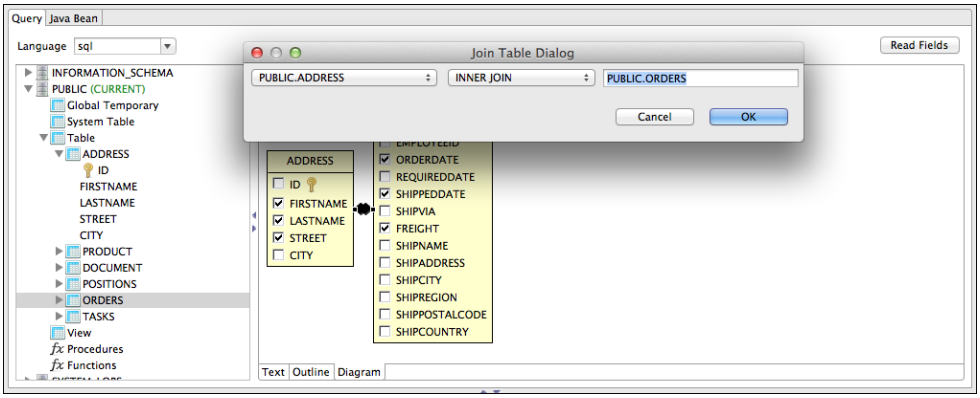


Figure 15-7 Column Dialog

You can also edit joins in the outline view by right-clicking a table name and selecting **Add or Edit Table Join**.

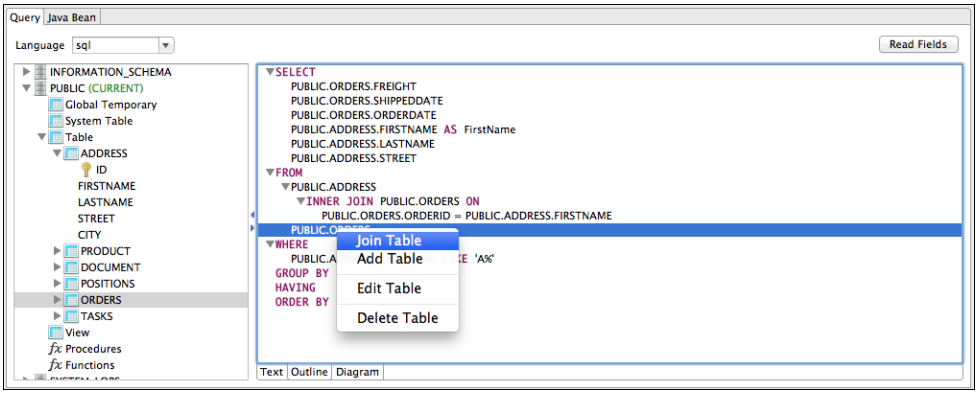


Figure 15-8 Join Table

15.2.4 Data Selection Criteria (WHERE Conditions)

Use a `WHERE` clause to specify the criteria to be for each record's part in the query result. Right click the `WHERE` node in the outline view and select **Add Condition** to add a new condition.

If you know in advance which column should be involved in the condition, you can drag this column on the **WHERE** node to create the condition. In both cases the condition dialog appears.

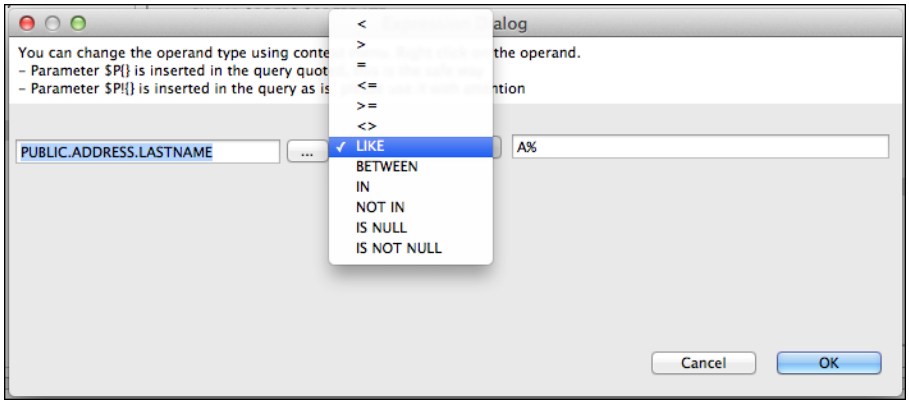


Figure 15-9 Add Condition

You can organize conditions by creating condition groups, and then combining them with `or` and `and` operators. At least one condition must be true for an `OR` group. All conditions must be true for the `AND` operator. You can double-click to change the value of either of these group operators.

If the value of a condition is not fixed, you can express it by using a parameter with the expression: `$P{parameter_name}`.

When using collection type parameters, the `$X{}`expressions allow you to use operators like `IN` and `NOT IN`, which determine whether a value is present in the provided list. The `$X{}` syntax also allows you to use other operators like `BETWEEN` for numbers, Date Range and Time Range type of parameters. See **2.3.2, "Expression Operators and Object Methods," on page 29** for more information about the `$X{}` syntax.

15.2.5 Acquiring Fields

When your query is ready, it's time map the columns of the result set to fields of the report. When using SQL, this is a pretty straight forward operation.

Press the Read button to execute the query. If the query is valid and no errors occur Jaspersoft Studio adds a field for each column with the proper class type to the fields list.

15.2.6 Data Preview

Use the **Data Preview** tab to generate a ghost report that maps the fields to the fields tab using your query and selected Data Adapter. This tool is independent of the query language and a good way to debug a query or check which records the dataset returns.

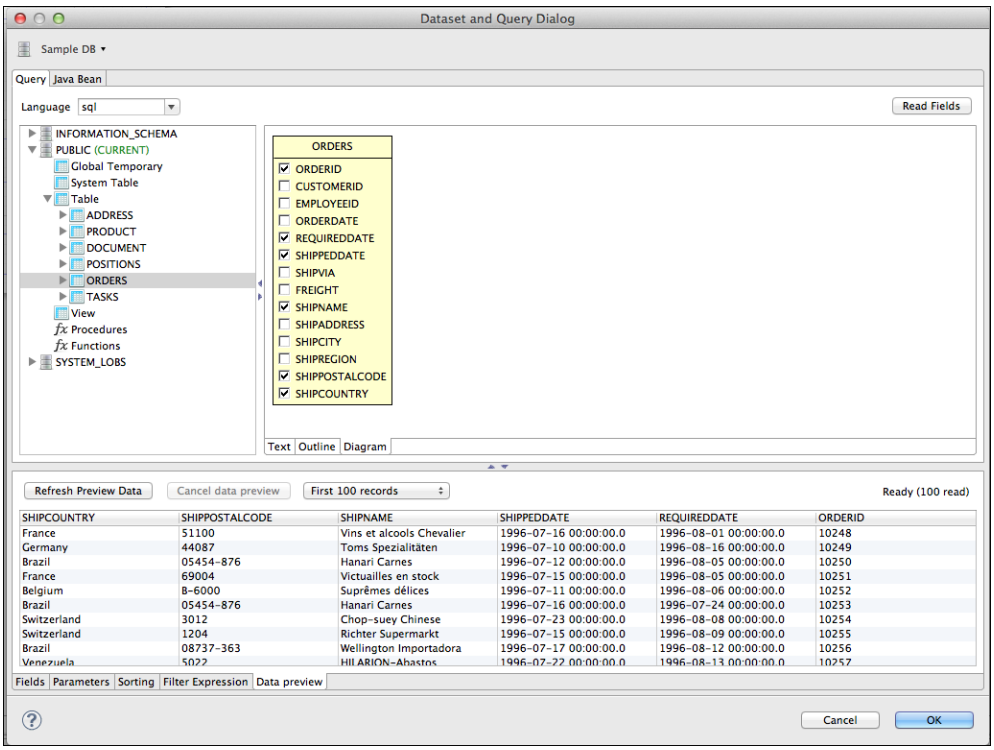


Figure 15-10 Data Preview Tab

CHAPTER 16 REPORT BOOKS

A report book is a single .jrxml that bundles multiple reports into a single object. Like a single report, a report book is driven by a data set that allows you to define the flow of the book's sections, and the parts within those sections.

This section provides a walkthrough of the report book creation process, using the sample.db included with your JasperReports Server installation. You'll create a cover page, table of contents, and subreports to build into your report book.

Creating a report book has a number of separate tasks, including:

- **Creating the Report Book Framework**
- **Creating and Adding Reports to the Report Book**
- **Refining the Report Book**
- **Configuring the Table of Contents**
- **Report Book Pagination**
- **Publishing the Report Book**

16.1 Creating the Report Book Framework

The first step is to create your report book jrxml. This is the framework in which you organize the book's parts.

To create the report book framework:

1. In Jaspersoft Studio, click  and select **Other...** to open the Wizard selection window.
2. Expand the Jaspersoft Studio folder, select **Jasper Report**, and click **Next**.
3. In the Categories panel, select **Report Books**.
4. Click to select **Wave Book** then click **Next**.
5. In the Project Explorer, select the **My Reports** folder, change the file name to `Sample_Book.jrxml`, and click **Next**.
6. In the Data Source window, select a data adapter. For our walkthrough, use **Sample DB – Database JDBC Connection**.
7. In the text pane, enter the following query then click **Next**:

```
select distinct shipcountry from orders order by shipcountry
```

8. In the Fields window, move **SHIPCOUNTRY** from the Dataset Fields panel to the Fields panel and click **Next**.

9. In the Book Sections window, make sure all three options are selected:
   - **Create Cover Section**
   - **Create Table of Contents**
   - **Create Back Cover Section**
10. Click **Finish**.

Your Report Book project opens in Jaspersoft Studio.

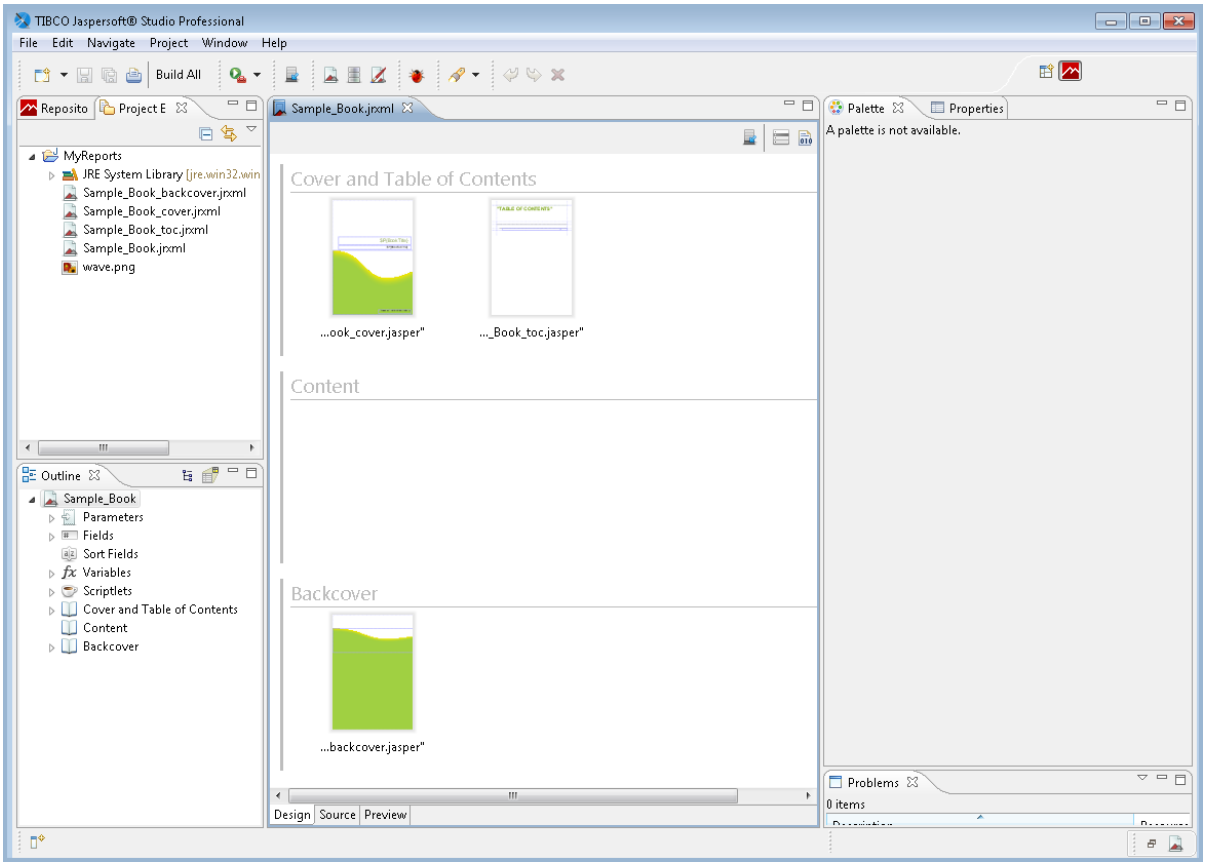


Figure 16-1 Report Book Framework

In Jaspersoft Studio, open the Project Explorer and expand the My Reports folder. There, you can see the jrxml files you just created:

- Sample_Book_backcover.jrxml
- Sample_Book_cover.jrxml
- Sample_Book_toc.jrxml
- Sample_Book.jrxml

Sample_Book.jrxml is open in the main Design tab. This is the file in which you'll organize the report parts. You'll notice three groups for the book part types:

- **Cover and Table of Contents** contains Sample_Book_cover.jrxml and Sample_Book_toc.jrxml.
- **Content** is currently empty.
- **Backcover** contains Sample_Book_backcover.jrxml

When you select each these book parts in the design window, you can view and edit their properties in the Properties View, as you can with standard reports and subreports.

Next, you'll create a subreport and add it to your report book.

16.2 Creating and Adding Reports to the Report Book

Now that your framework is established, you can create reports and/or subreports to include in your book.

You create a report or subreport as described here, and in **Creating a New Report**. You can also include previously-created reports. See **Adding a Report to the Report Book**.

For our walkthrough, you'll create a report, then add it to your report book.

16.2.1 Creating a Report for the Report Book

To create a report:

1. Click  and select **Other...** to open the Wizard selection window.
2. Expand the Jaspersoft Studio folder and select **Jasper Report**. Click **Next**.
3. In the Report Templates window, scroll to and select the **Leaf Green** template. Click **Next**.
4. In the Report file window, select the **MyReports** folder and change the Leaf_Green.file name to `Content_Page_One.jrxml`. Click **Next**.
5. In the Data Source window, select **Sample DB – Database JDBC Connection**, and enter the following query:

   ```
   Select * from orders order by shipcity
   ```
6. Click **Next**.
7. In the Fields window, move the following Dataset fields to the Fields panel on the right to include then in your subreport:
   - **ORDERID**
   - **CUSTOMERID**
   - **FREIGHT**
   - **SHIPCITY**
   - **SHIPCOUNTRY**
8. Click **Next**.
9. In the Group By window, move the **SHIPCITY** dataset field into the Fields pane.
10. Click **Finish**. The Content_Page_One.jrxml appears in the Design tab.
11. In the Project Explorer, right-click **Content_Page_One.jrxml** and select **Compile Report**. The resulting file, Content_Page_One.jasper, appears in the Project Explorer.

16.2.2 Adding a Report to the Report Book

Now you can add this new subreport to your report book. During this process, you'll assign a datasource for executing the subreport.

To add a report to your report book:

1. Double-click the **Sample_Book.jrxml** in the Project Explorer to open it in the Design tab.

2. Drag **Content_Page_One.jasper** from the Project Explorer into the Content group. In the Connection dialog, click **Finish**.
3. In the Design tab, click to select **Content_Page_One.jasper**.
4. In the Properties tab, click the **Data** button at the top, then click **Edit Parameters**.
5. In the Report Part Parameters window, click **Add** to open the Parameter Configuration dialog.
6. In the Parameter Name field, enter `REPORT_CONNECTION`.
7. Click  to open the Expression Editor.
8. In the first column, select **Parameters**.
9. In the center column, double-click **REPORT_CONNECTION parameter connection** to add it to the editor field, and click **Finish**. The expression appears in the Parameter Configuration field.
10. Click **OK** and confirm the parameter has been added to the Part Parameters list, then click **Finish**.

16.3 Refining the Report Book

You can further refine the report data to make your report easier to use, by sorting on additional fields, and by adding pages to the book to introduce each of the sorted sections.

16.3.1 Sorting on Additional Fields

At this point, you could run the report, but the data it returns is sorted by City, which you established in **Creating and Adding Reports to the Report Book**. To better organize the data, you can now modify the report query to sort by Country as well.

To add a filter to a report in a report book:

1. In the Project Explorer, double-click to open **content_page_one.jrxml** in the designer.
2. In the Outline tab, right-click the Parameters folder and select **Create Parameter**.
3. In the Properties tab, change the name to `country`.
4. In the Designer, click  and modify the query to say:

```
Select * from orders where shipcountry = $P{country} order by shipcity
```

5. Click **OK** to return to the designer.
6. Click the **Preview** tab at the bottom of the designer to open Input Parameters.
7. In the country field, enter `Italy` and run the report. The report preview displays only data related to Italy, sorted by city.
8. Click the Design tab, then save and compile your report.
9. Open **Sample_Book.jrxml** in the design tab, and click to select **Content_Page_One.jasper** in the Content group.
10. In the Properties tab (Data section), click **Edit Parameters** to open the Report Part Parameters window, and click **Add**.
11. In the Parameter Configuration Dialog, enter `country` as the parameter name.
12. Click  to open the Expression Editor, and click **Fields** in the left panel.
13. Double-click **SHIPCOUNTRY Field String** to add it to the expression, then click **Finish**.

14. Click **OK** in the Parameter Configuration Dialog, then **Finish** in the Report Part Parameters window.

16.3.2 Adding Section Introductory Pages

You can insert pages in your reports to introduce each section of data, as determined in **Sorting on Additional Fields**. These pages can include text, images, charts, or any number of other elements, pulled from a data source.

We'll place an introductory page before each country section and include the country name and a chart representing the number of orders for each city in the country.

To add introductory pages to your report:

1. Click  to open the Wizard selection window.
2. Expand the Jaspersoft Studio folder and select **Jasper Report**. Click **Next**.
3. In the Report Templates window, select the **Blank A4 Landscape** template. Click **Next**.
4. In the Report file window, select the **MyReports** folder and change the Blank_A4_Landscape.jrxml file name to `Country_Intro.jrxml`. Click **Next**.
5. In the Data Source window, select **Sample DB – Database JDBC Connection**, and enter the following query and click **Next**:

```
select count(*) c, shipcity from orders group by shipcity
```

6. In the Fields window, add the following Dataset fields to the Fields panel and click **Next**.
   - C
   - SHIPCITY
7. In the Group By window, click **Finish**. The Country_Intro.jrxml appears in the Design view.

Now you can determine what data appears on the intro pages.

To modify the data on the intro pages:

1. With Country_Intro.jrxml open in the Design view, click the **Title** band and increase its height to 350 pixels.
2. In the Outline view, right-click Parameters and select **Create Parameter**.
3. In the Properties view, change the Name from Parameter 1 to `Country`.
4. In the Designer view, click  to open the Dataset and Query Dialog.
5. Modify the query to say:

```
select count(*) c, shipcity from orders where shipcountry = $P{Country} group by
shipcity
```

6. Click **OK**.
7. Save Country_Intro.jrxml.
8. In the Outline view, drag **Country** from the Parameters list into the Title band.
9. Click the Country parameter ($P{Country}).
10. In the Properties view, click **Text Field**. Increase the font size to 26.
11. Click outside the parameter element.

Next, you can add a chart to the intro pages, that provides a graphical representation of the data in the section.

To add a chart to the intro pages:

1. In the Palette view, select and drag **HTML5 Charts** from the Components Pro section and place it under the parameter element in the designer view.
2. In the Chart type selection dialog, scroll down and select **Pie**. Click **OK**.
3. Resize the pie chart to fit the space. See **Creating a Simple Chart** for more information.
4. Double-click the chart element to open the Chart Properties.
5. Click the **Chart Data** tab, then click the **Configuration** tab.
6. In the Categories Levels section, double-click **Level1**.
7. In the Expression text box, delete "Change Me" and click  .
8. Select Fields from the first column, and double-click **SHIPCITY Field String** to add it to the expression.
9. Click **Finish**.
10. Update the **Name** field to `"City"` and click **OK**.
11. Back in the Chart Properties dialog, update the following information:
    - **Name**: `Number of orders.`
    - **Label Expression**: `"Number of orders"`
    - **Calculation**: Nothing
    - Value Expression: Delete new Integer1, click  , and double-click C Field Long, then click **Finish**.
12. Click **OK**, then save the report.
13. Compile **Country_Intro.jrxml** to create a .jasper file.

Now, you can add the Country_Intro page to your book, and configure it to display the correct data.

To add and configure the intro page::

1. Open Sample_Book.jrxml in the Design tab.
2. Drag **Country_Intro.jasper** from the Project Explorer into the Content group of Sample_Book.jrxml, and place it to the left of the Content_Page_One.jasper file.
3. In the Design tab, click to select **Country_Intro.jasper**.
4. In the Properties view, click **Edit Parameters**.
5. In the Report Part Parameters window, click **Add** to open the Parameter Configuration dialog.
6. In the Parameter Name field, enter `REPORT_CONNECTION`.
7. Click  to open the Expression Editor.
8. In the first column, select **Parameters**.
9. From the center column, double-click **REPORT_CONNECTION parameter connection** to add it to the editor field, and click **Finish**. The expression appears in the Parameter Expression field.
10. Click **OK** and confirm the parameter has been added to the Part Parameters list.
11. Click **Add** to open the Parameter Configuration dialog again.
12. In the Parameter Name field, enter `Country`.
13. Click  to open the Expression Editor.
14. In the first column, select **Fields**.
15. From the center column, double-click **SHIPCOUNTRY Field String** to add it to the editor field, and click **Finish**. The expression appears in the Parameter Expression field.

16. Click **OK** and confirm the parameter has been added to the Part Parameters list, click **Finish**.

16.4 Configuring the Table of Contents

Your report book is organized and the reports are populated with data. Now you can configure your Table of Contents so your users can find the information they need.

The Table of Contents is derived from a special data source created by JasperReports and included as a property in the report book. This property, `net.sf.jasperreports.bookmarks.data.source.parameter`, collects bookmarks from the report book's content pages. So you'll need to add bookmarks to your reports.

To add bookmarks:

1. Open **Country_Intro.jrxml** in the Design tab.
2. Click the **Country** parameter in the Title band.
3. In the Properties view, click **Hyperlink**.
4. Expand the **Anchor and Bookmark** section.
5. Click  to open the Expression Editor, and click **Parameters**.
6. Double-click **Country Parameter String** to add it to the expression, then click **Finish**.
7. In the Properties view, change the Bookmark Level to 1.
8. Click outside the design space in the Design tab, then click **Report** in the Properties view.
9. Click to enable **Create bookmarks**.
10. Open the **Content_Page_One.jrxml** in the Design tab.
11. Click the **$F{SHIPCITY}** text band.
12. In the Properties view, click **Hyperlink**.
13. Expand the **Anchor and Bookmark** section.
14. Click  to open the Expression Editor, and click **Fields**.
15. Double-click **SHIPCITY Field String** to add it to the expression, then click **Finish**.
16. In the Properties view, change the Bookmark Level to 2.
17. Save all files, and compile the Sample_Book.jrxml.

16.5 Report Book Pagination

To ensure that the report book's pagination increments correctly, you need to modify a few variables.

To establish the report book pagination:

1. In the Project Explorer, double-click to open **Content_Page_One.jrxml**.
2. On the Design tab,double-click the text field containing the expression **" "+$V{PAGE_NUMBER}**.
3. In the Expression Editor, and click **Variables** in the left panel.
4. Update the expression to the following:

```
"Page "+$V{MASTER_CURRENT_PAGE}+" of"
```

5. Click **Finish**.

6. Click to select the text field you just updated. Then in the Properties view, select **Text Field**.
7. Use the Evaluation Time drop-down menu to select **Master**.
8. Back in the Design tab, double-click the text field containing the expression **"Page "+$V{PAGE_NUMBER}**.
9. In the Expression Editor, and click **Variables** in the left panel.
10. Update the expression to the following:

```
" "+$V{MASTER_TOTAL_PAGES}
```

11. Click **Finish**.
12. As you did in steps 6-7, use the Evaluation Time drop-down menu to select **Master**.
13. Save your project.

16.6 Publishing the Report Book

Now that your report book has been created, tested, refined, configured, and paginated, you can publish it to JasperReports Server, so it is available to users

To publish your report book to JasperReports Server:

1. In the Project Explorer, double-click to open **Sample_Book.jrxml** in the Design tab.
2. Click  to open the Report Publishing wizard.
3. Browse to **JS Server** > **Public** > **Samples** > **Reports**, and click **Next**.
4. In the Select Resources dialog, verify that the following resources are listed:
   - Content_Page_One.jrxml
   - Country_Intro.jrxml
   - Sample_Book_backcover.jrxml
   - Sample_Book_cover.jrxml
   - Sample_Book_toc.jrxml
   - wave.png
5. Click **Next**.
6. Specify your data source, JServer JNDI Data Source, and click **Finish**.

   Your report book is published, and is available for use in JasperReports Server

GLOSSARY

Ad Hoc Editor

The interactive data explorer in JasperReports Server Professional and Enterprise editions. Starting from a predefined collection of fields, the Ad Hoc Editor lets you drag and drop fields, dimensions, and measures to explore data and create tables, charts, and crosstabs. These Ad Hoc views can be saved as reports.

Ad Hoc Report

In previous versions of JasperReports Server, a report created through the Ad Hoc Editor. Such reports could be added to dashboards and be scheduled, but when edited in iReport, lost their grouping and sorting. In the current version, the Ad Hoc Editor is used to explore views which in turn can be saved as reports. Such reports can be edited in iReport and Jaspersoft Studio without loss, and can be scheduled and added to dashboards.

Ad Hoc View

A view of data that is based on a Domain, Topic, or OLAP client connection. An Ad Hoc view can be a table, chart, or crosstab and is the entry point to analysis operations such as slice and dice, drill down, and drill through. **Compare OLAP View.** You can save an Ad Hoc view as a report in order to edit it in the interactive viewer, schedule it, or add it to a dashboard.

Aggregate Function

An aggregate function is one that is computed using a group of values; for example, Sum or Average. Aggregate functions can be used to create calculated fields in Ad Hoc views. Calculated fields containing aggregate functions cannot be used as fields or added to groups in an Ad Hoc view and should not be used as filters. Aggregate functions allow you to set a level, which specifies the scope of the calculation; level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total

Analysis View

See OLAP View.

Audit Archiving

To prevent audit logs from growing too large to be easily accessed, the installer configures JasperReports Server to move current audit logs to an archive after a certain number of days, and to delete logs in the archive after a certain age. The archive is another table in the JasperReports Server's repository database.

Audit Domains

A Domain that accesses audit data in the repository and lets administrators create Ad Hoc reports of server activity. There is one Domain for current audit logs and one for archived logs.

Audit Logging

When auditing is enabled, audit logging is the active recording of who used JasperReports Server to do what when. The system installer can configure what activities to log, the amount of detail gathered, and when to archive the data. Audit logs are stored in the same private database that JasperReports Server uses to store the repository, but the data is only accessible through the audit Domains.

Auditing

A feature of JasperReports Server Enterprise edition that records all server activity and allows administrators to view the data.

Calculated Field

In an Ad Hoc view or a Domain, a field whose value is calculated from a user-defined formula that may include any number of fields, operators, and constants. For Domains, a calculated field becomes one of the items to which the Domain's security file and locale bundles can apply. There are more functions available for Ad Hoc view calculations than for Domains.

CRM

Customer Relationship Management. The practice of managing every facet of a company's interactions with its clientele. CRM applications help businesses track and support their customers.

CrossJoin

An MDX function that combines two or more dimensions into a single axis (column or row).

Cube

The basis of most OLAP applications, a cube is a data structure that contains three or more dimensions that categorize the cube's quantitative data. When you navigate the data displayed in an OLAP view, you are exploring a cube.

Custom Field

In the Ad Hoc Editor, a field that is created through menu items as a simple function of one or two available fields, including other custom fields. When a custom field becomes too complex or needs to be used in many reports, it is best to define it as a calculated field in a Domain.

Dashboard

A collection of reports, input controls, graphics, labels, and web content displayed in a single, integrated view. Dashboards often present a high level view of your data, but input controls can parametrize the data to display. For example, you can narrow down the data to a specific date range. Embedded web content, such as other web-based applications or maps, make dashboards more interactive and functional.

Data Island

A single join tree or a table without joins in a Domain. A Domain may contain several data islands, but when creating an Ad Hoc view from a Domain, you can only select one of them to be available in the view.

Data Policy

In JasperReports Server, a setting that determines how the server processes and caches data used by Ad Hoc reports. Select your data policies by clicking **Manage > Ad Hoc Settings**.

Data Source

Defines the connection properties that JasperReports Server needs to access data. The server transmits queries to data sources and obtains datasets in return for use in filling reports and previewing Ad Hoc reports. JasperReports Server supports JDBC, JNDI, and Bean data sources; custom data sources can be defined as well.

Dataset

A collection of data arranged in columns and rows. Datasets are equivalent to relational results sets and the `JRDataSource` type in the JasperReports Library.

Datatype

In JasperReports Server, a datatype is used to characterize a value entered through an input control. A datatype must be of type text, number, date, or date-time. It can include constraints on the value of the input, for example maximum and minimum values. As such, a datatype in JasperReports Server is more structured than a datatype in most programming languages.

Denormalize

A process for creating table joins that speeds up data retrieval at the cost of having duplicate row values between some columns.

Derived Table

In a Domain, a derived table is defined by an additional query whose result becomes another set of items available in the Domain. For example, with a JDBC data source, you can write an SQL query that includes complex functions for selecting data. You can use the items in a derived table for other operations on the Domain, such as joining tables, defining a calculated field, or filtering. The items in a derived table can also be referenced in the Domain's security file and locale bundles.

Dice

An OLAP operation to select columns.

Dimension

A categorization of the data in a cube. For example, a cube that stores data about sales figures might include dimensions such as time, product, region, and customer's industry.

Domain

A virtual view of a data source that presents the data in business terms, allows for localization, and provides data-level security. A Domain is not a view of the database in relational terms, but it implements the same functionality within JasperReports Server. The design of a Domain specifies tables in the database, join clauses, calculated fields, display names, and default properties, all of which define items and sets of items for creating Ad Hoc reports.

Domain Topic

A Topic that is created from a Domain by the Data Chooser. A Domain Topic is based on the data source and items in a Domain, but it allows further filtering, user input, and selection of items. Unlike a JRXML-based Topic, a Domain Topic can be edited in JasperReports Server by users with the appropriate permissions.

Drill

To click on an element of an OLAP view to change the data that is displayed:

- Drill down. An OLAP operation that exposes more detailed information down the hierarchy levels by delving deeper into the hierarchy and updating the contents of the navigation table.
- Drill through. An OLAP operation that displays detailed transactional data for a given aggregate measure. Click a fact to open a new table beneath the main navigation table; the new table displays the low-level data that constitutes the data that was clicked.
- Drill up. An OLAP operation for returning the parent hierarchy level to view to summary information.

Eclipse

An open source Integrated Development Environment (IDE) for Java and other programming languages, such as C/C++.

ETL

Extract, Transform, Load. A process that retrieves data from transactional systems, and filters and aggregates the data to create a multidimensional database. Generally, ETL prepares the database that your reports will access. The Jaspersoft ETL product lets you define and schedule ETL processes.

Fact

The specific value or aggregate value of a measure for a particular member of a dimension. Facts are typically numeric.

Field

A field is equivalent to a column in the relational database model. Fields originate in the structure of the data source, but you may define calculated fields in a Domain or custom fields in the Ad Hoc Editor. Any type of field, along with its display name and default formatting properties, is called an item and may be used in the Ad Hoc Editor.

Frame

A dashboard element that displays reports or custom URLs. Frames can be mapped to input controls if their content can accept parameters.

Group

In a report, a group is a set of data rows that have an identical value in a designated field.

- In a table, the value appears in a header and footer around the rows of the group, while the other fields appear as columns.
- In a chart, the field chosen to define the group becomes the independent variable on the X axis, while the other fields of each group are used to compute the dependent value on the Y axis.

Hierarchy Level

In an OLAP cube, a member of a dimension containing a group of members.

Input Control

A button, check box, drop-down list, text field, or calendar icon that allows users to enter a value when running a report or viewing a dashboard that accepts input parameters. For JRXML reports, input controls and their associated datatypes must be defined as repository objects and explicitly associated with the report. For Domain-based reports that prompt for filter values, the input controls are defined internally. When either type of report is used in a dashboard, its input controls are available to be added as special content.

iReport Designer

An open source tool for graphically designing reports that leverage all features of the JasperReports Library. The Jaspersoft iReport Designer lets you drag and drop fields, charts, and sub-reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in iReport, or upload it to JasperReports Server. iReport is implemented in NetBeans.

Item

When designing a Domain or creating a Topic based on a Domain, an item is the representation of a database field or a calculated field along with its display name and formatting properties defined in the Domain. Items can be grouped in sets and are available for use in the creation of Ad Hoc reports.

JasperReport

A combination of a report template and data that produces a complex document for viewing, printing, or archiving information. In the server, a JasperReport references other resources in the repository:

- The report template (in the form of a JRXML file)
- Information about the data source that supplies data for the report
- Any additional resources, such as images, fonts, and resource bundles referenced by the report template.

The collection of all the resources that are referenced in a JasperReport is sometimes called a report unit. End users usually see and interact with a JasperReport as a single resource in the repository, but report creators must define all of the components in the report unit.

Level

Specifies the scope of an aggregate function in an Ad Hoc view. Level values include Current (not available for PercentOf), ColumnGroup, ColumnTotal, RowGroup, RowTotal, Total.

JasperReports Library

An embeddable, open source, Java API for generating a report, filling it with current data, drawing charts and tables, and exporting to any standard format (HTML, PDF, Excel, CSV, and others). JasperReports processes reports defined in JRXML, an open XML format that allows the report to contain expressions and logic to control report output based on run-time data.

JasperReports Server

A commercial open source, server-based application that calls the JasperReports Library to generate and share reports securely. JasperReports Server authenticates users and lets them upload, run, view, schedule, and send reports from a web browser. Commercial versions provide metadata layers, interactive report and dashboard creation, and enterprise features such as organizations and auditing.

Jaspersoft ETL

A graphical tool for designing and implementing your data extraction, transforming, and loading (ETL) tasks. It provides hundreds of data source connectors to extract data from many relational and non-relational systems. Then, it schedules and performs data aggregation and integration into data marts or data warehouses that you use for reporting.

Jaspersoft OLAP

A relational OLAP server integrated into JasperReports Server that performs data analysis with MDX queries. The product includes query builders and visualization clients that help users explore and make sense of multidimensional data. Jaspersoft OLAP also supports XML/A connections to remote servers.

Jaspersoft Studio

An open source tool for graphically designing reports that leverage all features of the JasperReports Library. Jaspersoft Studio lets you drag and drop fields, charts, and sub-reports onto a canvas, and also define parameters or expressions for each object to create pixel-perfect reports. You can generate the JRXML of the report directly in Jaspersoft Studio, or upload it to JasperReports Server. Jaspersoft Studio is implemented in Eclipse.

JavaBean

A reusable Java component that can be dropped into an application container to provide standard functionality.

JDBC

Java Database Connectivity. A standard interface that Java applications use to access databases.

JNDI

Java Naming and Directory Interface. A standard interface that Java applications use to access naming and directory services.

Join Tree

In Domains, a collection of joined tables from the actual data source. A join is the relational operation that associates the rows of one table with the rows of another table based on a common value in given field of each table. Only the fields in a same join tree or calculated from the fields in a same join tree may appear together in a report.

JPivot

An open source graphical user interface for OLAP operations. For more information, visit http://jpivot.sourceforge.net/.

JRXML

An XML file format for saving and sharing reports created for the JasperReports Library and the applications that use it, such as iReport Designer and JasperReports Server. JRXML is an open format that uses the XML standard to define precisely all the structure and configuration of a report.

MDX

Multidimensional Expression Language. A language for querying multidimensional objects, such as OLAP (On Line Analytical Processing) cubes, and returning cube data for analytical processing. An MDX query is the query that determines the data displayed in an OLAP view.

Measure

Depending on the context:

- In a report, a formula that calculates the values displayed in a table's columns, a crosstab's data values, or a chart's dependent variable (such as the slices in a pie).
- In an OLAP view, a formula that calculates the facts that constitute the quantitative data in a cube.

Mondrian

A Java-based, open source multidimensional database application.

Mondrian Connection

An OLAP client connection that consists of an OLAP schema and a data source. OLAP client connections populate OLAP views.

Mondrian Schema Editor

An open source Eclipse plug-in for creating Mondrian OLAP schemas.

Mondrian XML/A Source

A server-side XML/A source definition of a remote client-side XML/A connection used to populate an OLAP view using the XML/A standard.

MySQL

An open source relational database management system. For information, visit http://www.mysql.com/.

Navigation Table

The main table in an OLAP view that displays measures and dimensions as columns and rows.

ODBO Connect

Jaspersoft ODBO Connect enables Microsoft Excel 2003 and 2007 Pivot Tables to work with Jaspersoft OLAP and other OLAP servers that support the XML/A protocol. After setting up the Jaspersoft ODBO data source, business analysts can use Excel Pivot Tables as a front-end for OLAP analysis.

OLAP

On Line Analytical Processing. Provides multidimensional views of data that help users analyze current and past performance and model future scenarios.

OLAP Client Connection

A definition for retrieving data to populate an OLAP view. An OLAP client connection is either a direct Java connection (Mondrian connection) or an XML-based API connection (XML/A connection).

OLAP Schema

A metadata definition of a multidimensional database. In Jaspersoft OLAP, schemas are stored in the repository as XML file resources.

OLAP View

Also called an analysis view. A view of multidimensional data that is based on an OLAP client connection and an MDX query. Unlike Ad Hoc views, you can directly edit an OLAP view's MDX query to change the data and the way they are displayed. An OLAP view is the entry point for advanced analysis users who want to write their own queries. **Compare Ad Hoc View.**

Organization

A set of users that share folders and resources in the repository. An organization has its own user accounts, roles, and root folder in the repository to securely isolate it from other organizations that may be hosted on the same instance of JasperReports Server.

Organization Admin

Also called the organization administrator. A user in an organization with the privileges to manage the organization's user accounts and roles, repository permissions, and repository content. An organization admin can also create suborganizations and mange all of their accounts, roles, and repository objects. The default organization admin in each organization is the `jasperadmin` account.

Outlier

A fact that seems incongruous when compared to other member's facts. For example, a very low sales figure or a very high number of helpdesk tickets. Such outliers may indicate a problem (or an important achievement) in your business. The analysis features of Jaspersoft OLAP excel at revealing outliers.

Parameter

Named values that are passed to the engine at report-filling time to control the data returned or the appearance and formatting of the report. A report parameter is defined by its name and type. In JasperReports Server, parameters can be mapped to input controls that users can interact with.

Pivot

To rotate a crosstab such that its row groups become column groups and its column groups become rows. In the Ad Hoc Editor, pivot a crosstab by clicking  .

Pivot Table

A table with two physical dimensions (for example, X and Y axis) for organizing information containing more than two logical dimensions (for example, PRODUCT, CUSTOMER, TIME, and LOCATION), such that each physical dimension is capable of representing one or more logical dimensions, where the values described by the dimensions are aggregated using a function such as SUM. Pivot tables are used in Jaspersoft OLAP.

Properties

Settings associated with an object. The settings determine certain features of the object, such as its color and label. Properties are normally editable. In Java, properties can be set in files listing objects and their settings.

Report

In casual usage, *report* may refer to:

- A JasperReport. **See JasperReport.**
- The main JRXML in a JasperReport.
- The file generated when a JasperReport is scheduled. Such files are also called content resources or output files.
- The file generated when a JasperReport is run and then exported.
- In previous JasperReports Server versions, a report created in the Ad Hoc Editor. **See Ad Hoc Report.**

Report Run

An execution of a report, Ad Hoc view, or dashboard, or a view or dashboard designer session, it measures and limits usage of Freemium instances of JasperReports Server. The executions apply to resources no matter how they are run (either in the web interface or through the various APIs, such as REST web services). Users of our Community Project and our full-use commercial licenses are not affected by the limit. For more information, please contact sales@jaspersoft.com.

Repository

The tree structure of folders that contain all saved reports, dashboards, OLAP views, and resources. Users access the repository through the JasperReports Server web interface or through iReport. Applications can access the repository through the web service API. Administrators use the import and export utilities to back up the repository contents.

Resource

In JasperReports Server, anything residing in the repository, such as an image, file, font, data source, Topic, Domain, report element, saved report, report output, dashboard, or OLAP view. Resources also include the folders in the repository. Administrators set user and role-based access permissions on repository resources to establish a security policy.

Role

A security feature of JasperReports Server. Administrators create named roles, assign them to user accounts, and then set access permissions to repository objects based on those roles. Certain roles also determine what functionality and menu options are displayed to users in the JasperReports Server interface.

Schema

A logical model that determines how data is stored. For example, the schema in a relational database is a description of the relationships between tables, views, and indexes. In Jaspersoft OLAP, an OLAP schema is the logical model of the data that appears in an OLAP view; they are uploaded to the repository as resources. For Domains, schemas are represented in XML design files.

Schema Workbench

A graphical tool for easily designing OLAP schemas, data security schemas, and MDX queries. The resulting cube and query definitions can then be used in Jaspersoft OLAP to perform simple but powerful analysis of large quantities of multi-dimensional data stored in standard RDBMS systems.

Set

In Domains and Domain Topics, a named collection of items grouped together for ease of use in the Ad Hoc Editor. A set can be based on the fields in a table or entirely defined by the Domain creator, but all items in a set must originate in the same join tree. The order of items in a set is preserved.

Slice

An OLAP operation for filtering data rows.

SQL

Structured Query Language. A standard language used to access and manipulate data and schemas in a relational database.

System Admin

Also called the system administrator. A user who has unlimited access to manage all organizations, users, roles, repository permissions, and repository objects across the entire JasperReports Server instance. The system admin can create root-level organizations and manage all server settings. The default system admin is the `superuser` account.

Topic

A JRXML file created externally and uploaded to JasperReports Server as a basis for Ad Hoc reports. Topics are created by business analysts to specify a data source and a list of fields with which business users can create reports in the Ad Hoc Editor. Topics are stored in the Ad Hoc Components folder of the repository and displayed when a user launches the Ad Hoc Editor.

Transactional Data

Data that describe measurable aspects of an event, such as a retail transaction, relevant to your business. Transactional data are often stored in relational databases, with one row for each event and a table column or field for each measure.

User

Depending on the context:

- A person who interacts with JasperReports Server through the web interface. There are generally three categories of users: administrators who install and configure JasperReports Server, database experts or business analysts who create data sources and Domains, and business users who create and view reports and dashboards.
- A user account that has an ID and password to enforce authentication. Both people and API calls accessing the server must provide the ID and password of a valid user account. Roles are assigned to user accounts to determine access to objects in the repository.

View

Several meanings pertain to JasperReports Server:

- An Ad Hoc view. **See Ad Hoc View.**
- An OLAP view. **See OLAP View.**
- A database view. See http://en.wikipedia.org/wiki/View_%28database%29.

Virtual Data Source

A virtual data source allows you to combine data residing in multiple JDBC and/or JNDI data sources into a single data source that can query the combined data. Once you have created a virtual data source, you create Domains that join tables across the data sources to define the relationships between the data sources.

WCF

Web Component Framework. A low-level GUI component of JPivot. For more information, see http://jpivot.sourceforge.net/wcf/index.html.

Web Services

A SOAP (Simple Object Access Protocol) API that enables applications to access certain features of JasperReports Server. The features include repository, scheduling and user administration tasks.

XML

eXtensible Markup language. A standard for defining, transferring, and interpreting data for use across any number of XML-enabled applications.

XML/A

XML for Analysis. An XML standard that uses Simple Object Access protocol (SOAP) to access remote data sources. For more information, see http://www.xmla.org/.

XML/A Connection

A type of OLAP client connection that consists of Simple Object Access Protocol (SOAP) definitions used to access data on a remote server. OLAP client connections populate OLAP views.

INDEX

F

G

H

I

J

L

M