

TIBCO  [®] Analytics

TIBCO JASPERSOFT[®] STUDIO USER GUIDE

RELEASE 6.3

<http://www.jaspersoft.com>

Copyright ©2005-2016, TIBCO Software Inc. All rights reserved. Printed in the U.S.A. TIBCO, the TIBCO logo, TIBCO JasperSoft, the TIBCO JasperSoft logo, TIBCO JasperSoft iReport Designer, TIBCO JasperReports Library, TIBCO JasperReports Server, TIBCO JasperSoft OLAP, TIBCO JasperSoft Studio, and TIBCO JasperSoft ETL are trademarks and/or registered trademarks of TIBCO Software Inc. in the United States and in jurisdictions throughout the world. All other company and product names are or may be trade names or trademarks of their respective owners.

This is version 0616-JSP63-11 of the *TIBCO JasperSoft Studio User Guide*.

TABLE OF CONTENTS

Chapter 1 Getting Started with Jaspersoft Studio	11
1.1 Introduction to Jaspersoft Studio	11
1.2 Installing Jaspersoft Studio	12
1.2.1 Software Requirements	12
1.2.2 Hardware Requirements	13
1.2.3 Available Packages	13
1.2.4 Updating Your Workspace to Jaspersoft Studio 6.2 and Higher	13
1.2.5 Compatibility Between Versions	17
1.2.6 Accessing the Source Code	18
Chapter 2 Creating a Simple Report	19
2.1 Creating a New Report	19
2.2 Adding and Deleting Report Elements	23
2.2.1 Adding Fields to a Report	23
2.2.2 Deleting Fields	24
2.2.3 Adding Other Elements	24
2.3 Previewing a Report	24
2.4 Creating a Project Folder	25
Chapter 3 User Interface and Design View	27
3.1 Eclipse Interface	28
3.1.1 Learning More About Eclipse	28
3.2 User Interface Components	28
3.3 The Design Tab	29
3.4 Understanding Bands	30
3.4.1 Band Types	30
3.5 Specifying Report Properties	31
3.5.1 Columns	33
3.5.2 Advanced Options	33
3.6 Exporting Reports with Jaspersoft Studio	34
3.6.1 Compiling the Report	34
3.6.2 Preview and Exporting	34
3.6.3 Choosing Report Templates for PDF	35

Chapter 4 Report Elements	37
4.1 Common Element Properties	38
4.1.1 The Palette	38
4.1.2 Element Properties	38
4.2 Inserting, Selecting, and Positioning Elements	39
4.2.1 Inserting Elements	39
4.2.2 Selecting Elements	40
4.2.3 Positioning Elements	40
4.2.4 Positioning Elements in Containers	41
4.3 Formatting Elements	45
4.4 Graphic Elements	48
4.4.1 Line	48
4.4.2 Rectangle and Ellipse	48
4.4.3 Images	48
4.4.4 Padding and Borders	48
4.5 Text Elements	49
4.5.1 Static Text	49
4.5.2 Text Fields	49
4.6 Frames	50
4.7 Inserting Page and Column Breaks	51
4.8 Working with Composite Elements	51
4.8.1 Creating and Editing Composite Elements	51
4.8.2 Exporting and Importing Composite Elements	54
4.9 Anchors, Bookmarks, and Hyperlinks	55
4.9.1 Anchors and Bookmarks	56
4.9.2 Hyperlinks	57
4.9.3 Hyperlink Types	59
4.9.4 Creating a Hyperlink	60
4.10 Advanced Elements and Custom Components	60
4.11 Custom Visualization Component	61
Chapter 5 Fields	63
5.1 Understanding Fields	63
5.2 Registration of Fields from a SQL Query	65
5.3 Registration of JavaBean Fields	66
5.4 Fields and Text Fields	68
5.5 Data Centric Exporters	68
5.5.1 Configuring a Report's Metadata for PDF 508 Tags	68
5.5.2 Configuring a Report's Metadata for Use With the JSON Data Exporter	71
Chapter 6 Parameters	75
6.1 Managing Parameters	75
6.2 Default Parameters	77
6.3 Using Parameters in Queries	79
6.3.1 Using Parameters in a SQL Query	79
6.3.2 Using Parameters with Null Values	79

6.3.3 IN and NOTIN Clauses	80
6.3.4 Relative Dates	80
6.3.5 Passing Parameters from a Program	83
6.4 Parameters Prompt	84
Chapter 7 Variables	87
7.1 Defining or Editing a Variable	87
7.2 Base Properties of a Variable	87
7.3 Other Properties of a Variable	88
7.3.1 Evaluation Time	88
7.3.2 Calculation Function	89
7.3.3 Increment Type	89
7.3.4 Reset Type	90
7.3.5 Incrementer Factory Class Name	90
7.4 Built-In Variables	90
7.5 Tips & Tricks	91
Chapter 8 Expressions	93
8.1 Expression Types	93
8.2 Expression Operators and Object Methods	94
8.3 Using an If-Else Construct in an Expression	96
8.4 Using Unicode Characters in Expressions	97
8.5 Using Java as a Language for Expressions	97
8.6 Using Groovy as a Language for Expressions	98
8.7 Using JavaScript as a Language for Expressions	99
Chapter 9 Fonts	101
9.1 Font Extensions Reference	101
9.1.1 The Fonts Page	101
9.1.2 The Font Family Dialog	103
9.1.3 Font Sets	106
9.2 Example of Using Font Extensions	107
9.2.1 Creating Font Extensions and Font Sets	108
9.2.2 Using Font Extensions in a Report	112
9.3 Deploying Font Extensions to JasperReports Server	115
Chapter 10 Data Adapters	119
10.1 Creating and Editing Data Adapters	120
10.1.1 Creating a Data Adapter	120
10.1.2 Importing and Exporting Data Adapters	121
10.1.3 Copying a Data Adapter	122
10.2 Using Data Adapters in Reports and Datasets	122
10.2.1 Data Adapter For a Report	122
10.2.2 Data Adapters and Report Deployment	123
10.2.3 Default Data Adapter	123
10.3 Working with Database JDBC Connections	125
10.3.1 Creating a Database JDBC Connection	125
10.3.2 Troubleshooting a Database JDBC Connection	127

10.3.3 Using a Database JDBC Connection	129
10.4 Working with a MongoDB Data Adapter	131
10.4.1 Creating a Native MongoDB Connection	132
10.4.2 Creating a MongoDB JDBC Data Source	134
10.5 Working with a Native Cassandra Connection	136
10.5.1 Creating a Native Cassandra Data Adapter	136
10.5.2 Using a Cassandra Connection	138
10.6 Working with a Collection of JavaBeans Data Adapter	138
10.6.1 Implementing the Factory Class for a Collection of JavaBeans	139
10.6.2 Creating a Data Adapter from a Factory Class	140
10.6.3 Registering the Fields	141
10.7 Working with XML Data Adapters	141
10.7.1 Creating a Node Set for an XML Document	141
10.7.2 Creating an XML Data Adapter	143
10.7.3 Registration of Fields for an XML Data Adapter	145
10.7.4 XML Data Adapters and Subreports	146
10.8 Working with XML/A Data Adapters	148
10.8.1 Registration of fields in XML/A Providers	149
10.9 Working with CSV Data Adapters	149
10.9.1 Registration of the Fields for a CSV Data Adapter	152
10.10 Using the Empty Record Data Adapter	152
10.10.1 Understanding the Empty Record Implementation	153
10.11 Working with the JRDataSource Interface	153
10.11.1 Understanding the JRDataSource Interface	154
10.11.2 Implementing a New JRDataSource	154
10.11.3 Using a Custom JasperReports Data Source with Jaspersoft Studio	156
10.12 A Look at TIBCO Spotfire Information Links	157
Chapter 11 Creating Queries	161
11.1 Using the Dataset and Query Dialog	161
11.2 Working with the Query Builder	163
11.2.1 Query Outline View and Diagram View	163
11.2.2 Selecting Columns	165
11.2.3 Joining Tables	166
11.2.4 Data Selection Criteria (WHERE Conditions)	167
11.2.5 Acquiring Fields	168
11.2.6 Data Preview	168
Chapter 12 Accessing JasperReports Server from Jaspersoft Studio	169
12.1 Connecting to JasperReports Server	170
12.1.1 Advanced Connection Settings	171
12.1.2 Using Single Sign-on with JasperReports Server	172
12.2 Publishing a Report to JasperReports Server	174
12.2.1 Publishing Report Resources	174
12.2.2 Choosing a Data Source for a Published Report	174
12.2.3 Example of Publishing a Report	177
12.3 Working with JasperReports Server Templates	179

12.3.1	Creating a Custom JasperReports Server Template	179
12.3.2	Report Template Styles in Jaspersoft Studio	182
12.4	Creating and Uploading a Topic for Ad Hoc Views	183
12.5	Managing Repository Objects through Jaspersoft Studio	184
12.5.1	Adding, Modifying and Deleting Resources	185
12.5.2	Running a Report	186
12.5.3	Editing a Report	186
12.6	Creating and Uploading Chart Themes	187
12.7	Working with Domains	189
12.8	Understanding the repo: Syntax	190
12.9	Adding a Date/Time Stamp to Scheduled Output in JasperReports Server	191
Chapter 13	Working with Tables	195
13.1	Creating a Table	195
13.2	Editing a Table	201
13.2.1	Editing Table Properties	201
13.2.2	Editing Table Styles	201
13.2.3	Editing Cell Contents	202
13.2.4	Editing Table Data	203
13.2.5	Editing Table Source	204
13.3	Table Structure	204
13.3.1	Table Elements	204
13.3.2	Table Cells	205
13.4	Working with Columns	206
13.4.1	Table Properties for Managing Columns	206
13.4.2	Working with Individual Columns	206
13.4.3	Column Groups	207
Chapter 14	Working with Charts	209
14.1	Creating a Simple Chart	209
14.2	Setting Chart Properties	214
14.3	Spider Charts	214
14.4	Chart Themes	218
14.4.1	Using the Chart Theme Designer	218
14.4.2	Editing Chart Theme XML	218
14.4.3	Creating a JasperReports Extension for a Chart Theme	218
14.4.4	Applying a Chart Theme	219
Chapter 15	HTML5 Charts in Commercial Editions	221
15.1	Overview of HTML5 Charts	221
15.2	Simple HTML5 Charts	227
15.2.1	Creating an HTML5 chart	227
15.2.2	Editing HTML5 Charts	230
15.2.3	Creating Hyperlinks	232
15.2.4	Setting Advanced Options for HTML5 Charts	233
15.3	Scatter Charts	234
15.4	Dual-Axis, Multi-Axis, and Combination Charts	238

15.5 Hyperlinks in HTML5 Charts	241
15.5.1 Creating Hyperlinks in HTML5 Charts	241
15.5.2 Working with Bucket Properties and Hidden Measures	245
15.5.3 Working with Report Units	249
Chapter 16 Working with Crosstabs	251
16.1 Example of Creating a Crosstab	252
16.2 Working with Crosstab Properties	257
16.3 Using the Crosstab Editor	258
16.3.1 Formatting Columns, Rows, and Cells	258
16.3.2 Editing Row or Column Group Properties	259
16.3.3 Adding and Deleting Row and Column Groups	261
16.3.4 Working with Measures	263
16.4 Working with Crosstab Parameters	267
Chapter 17 Working With the Map Component	269
17.1 Working with Map Properties	269
17.2 Viewing Authentication Properties	271
17.3 Working with Markers	272
17.3.1 Marker Properties	273
17.3.2 Adding Markers Manually	273
17.3.3 Adding Markers Using the Map	275
17.3.4 Adding Markers Using a Dataset	276
17.3.5 Modifying Markers	280
17.4 Working with Paths	281
17.4.1 Defining Path Styles	281
17.4.2 Defining a Path Manually	283
17.4.3 Defining a Path Using a Dataset	284
17.4.4 Modifying Paths and Path Styles	285
17.5 Properties for Markers and Paths	285
Chapter 18 Working with TIBCO GeoAnalytics Maps	289
18.1 Configuring a Basic Map	290
18.2 Using Expressions for Properties	292
18.3 Understanding Layers	293
18.4 Working with Markers	294
18.4.1 Static Markers	294
18.4.2 Dynamic Markers	297
18.5 Working with Paths	300
Chapter 19 Working with Subreports	303
19.1 Creating a New Report via the Subreport Wizard	303
19.2 Understanding Subreports	306
19.2.1 Subreports	306
19.2.2 Subreport Elements	307
19.2.3 The Expression Property	308
19.2.4 Specifying the Data Source	309
19.2.5 Subreport Parameters	309

Chapter 20 Report Templates	313
20.1 Template Structure	313
20.2 Creating and Customizing Templates	315
20.2.1 Creating a New Template	315
20.2.2 Customizing a Template	317
20.3 Saving Templates	318
20.3.1 Creating a Template Directory	318
20.3.2 Exporting a Template	319
20.3.3 Creating a Template Thumbnail	321
20.4 Adding Templates to Jaspersoft Studio	321
Chapter 21 Report Books	323
21.1 Creating the Report Book Framework	323
21.2 Creating and Adding Reports to the Report Book	325
21.2.1 Creating a Report for the Report Book	325
21.2.2 Adding a Report to the Report Book	325
21.3 Refining the Report Book	326
21.3.1 Sorting on Additional Fields	326
21.3.2 Adding Section Introductory Pages	327
21.4 Configuring the Table of Contents	328
21.5 Report Book Pagination	329
21.6 Publishing the Report Book	330
Chapter 22 Preferences and Configuration	331
22.1 Properties	331
22.2 JasperReports Samples	331
22.3 Units of Measure in Jaspersoft Studio	331
22.3.1 Configuration	332
22.3.2 Changing the Field Unit of Measure	332
22.3.3 Alias and Auto-complete	332
22.3.4 Approximations	333
22.4 Export and Import	333
22.4.1	335
22.5 Setting Compatibility with Earlier Versions of JasperReports Library	335
Appendix A Concepts of JasperReports	339
A.1 JRXML Sources and Jasper Files	339
A.1.1 The Report Lifecycle	339
A.2 Data Sources and Print Formats	345
A.3 Using JasperReports Extensions in Jaspersoft Studio	345
A.4 A Simple Program	346
Glossary	347
Index	357

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:

- **Introduction to Jaspersoft Studio**
- **Installing Jaspersoft Studio**
- **Exporting Reports with Jaspersoft Studio**

1.1 Introduction to Jaspersoft Studio

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.

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 users interested in multi-dimensional modeling, we offer Jaspersoft® OLAP, which runs as part of 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 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.

You can use the following sources of information to learn about JasperReports Server:

- Our core documentation describes how to install, administer, and use JasperReports Server and Jaspersoft Studio. 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 Library, Jaspersoft Studio, and JasperReports Server, are documented online.

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 Installing Jaspersoft Studio

Jaspersoft Studio is available as an Eclipse Rich Client Package (RCP), downloadable from the following location:

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

1.2.1 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

1.2.2 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.

1.2.3 Available Packages

The Eclipse RCP package is available in the following formats for community and commercial versions.

Linux versions:

- TIBCOJaspersoftStudio-x.x.x.final-linux-x86.tgz
- TIBCOJaspersoftStudio-x.x.x.final-linux-x86_64.tgz

Mac:

- TIBCOJaspersoftStudio-x.x.x.final-mac-x86_64.dmg

Windows:

- TIBCOJaspersoftStudio-x.x.x.final-windows-installer-x86.exe
- TIBCOJaspersoftStudio-x.x.x.final-windows-installer-x86_64.exe

x.x.x represents the version number of Jaspersoft Studio.

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.

For community only, unsupported versions for the Eclipse RCP on Windows and Debian are available as a convenience for users who are in a restricted environment and can't download or install an .exe file:

- TIBCOJaspersoftstudio_x.x.x.final_amd64.deb
- TIBCOJaspersoftstudio_x.x.x.final_i386.deb
- TIBCOJaspersoftStudio-x.x.x.final.win32.x86_64.zip
- TIBCOJaspersoftStudio-x.x.x.final.win32.x86.zip

The community version is also available as an unsupported Eclipse plug-in, called the Jaspersoft Studio plugin. You can install it from the Eclipse Marketplace or download using the Eclipse Update Manager. See the following article on the community website for more information about working with the Jaspersoft Studio plugin.

<http://community.jaspersoft.com/wiki/contributing-jaspersoft-studio-and-building-sources>

1.2.4 Updating Your Workspace to Jaspersoft Studio 6.2 and Higher

Due to incompatibilities between Eclipse 4.5 and earlier versions of the Jaspersoft Studio workspace, the workspace format has been updated in Jaspersoft Studio 6.2. The new workspace format can't be used with Jaspersoft Studio 6.1.1 or earlier.

If you are updating to 6.2 or higher from Jaspersoft Studio 6.1.1 or earlier, you are prompted to choose a new workspace when you launch Jaspersoft Studio. When you choose a new workspace, a new, empty workspace is created and set as the workspace for your Jaspersoft Studio instance. This workspace will be used for versions 6.2 and higher. Your previous workspace remains unchanged and can still be used for versions 6.1.1 or earlier.

To update the reports and data from your earlier version of Jaspersoft Studio to version 6.2 or higher, you can import some or all of your projects, server connections, data adapters, and project settings into your new workspace.

Importing projects:

1. (Optional) To import a version of the MyReports project, you must first delete the existing MyReports folder from your current workspace. You can do this, for example, if you have just upgraded to 6.2 or higher and have created a new empty workspace. To delete MyReports in your current workspace, navigate to the workspace location in your file system and delete or move the MyReports directory.
2. Select **File > Import ...**
3. Select **Existing Projects into Workspace** from the **General** category.

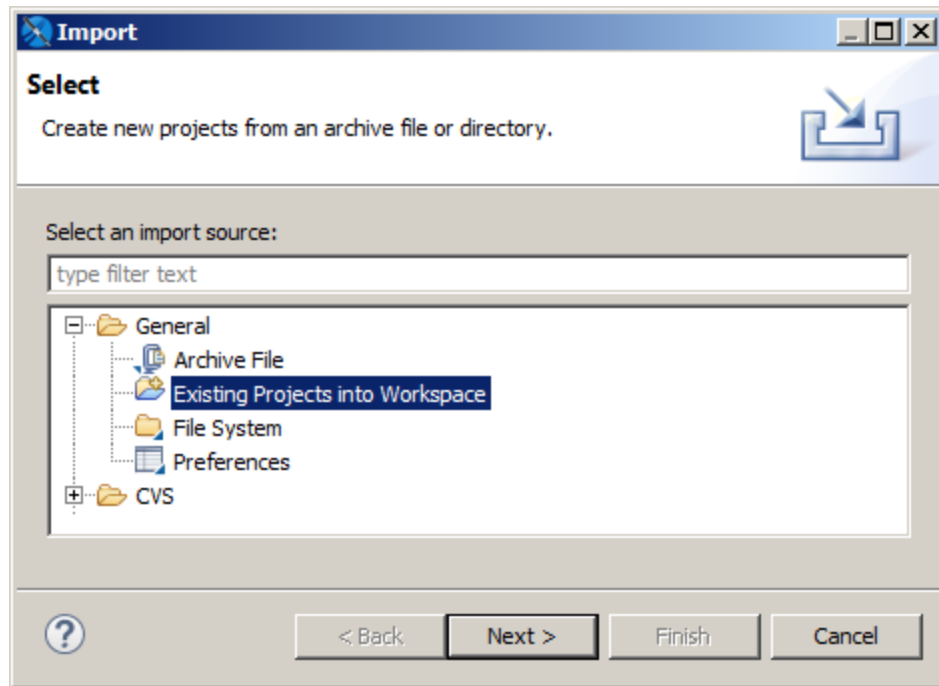


Figure 1-1 Selecting projects in Import dialog

4. Browse to the workspace with you want, click **OK**, and then click **Next**.
The Import dialog opens.

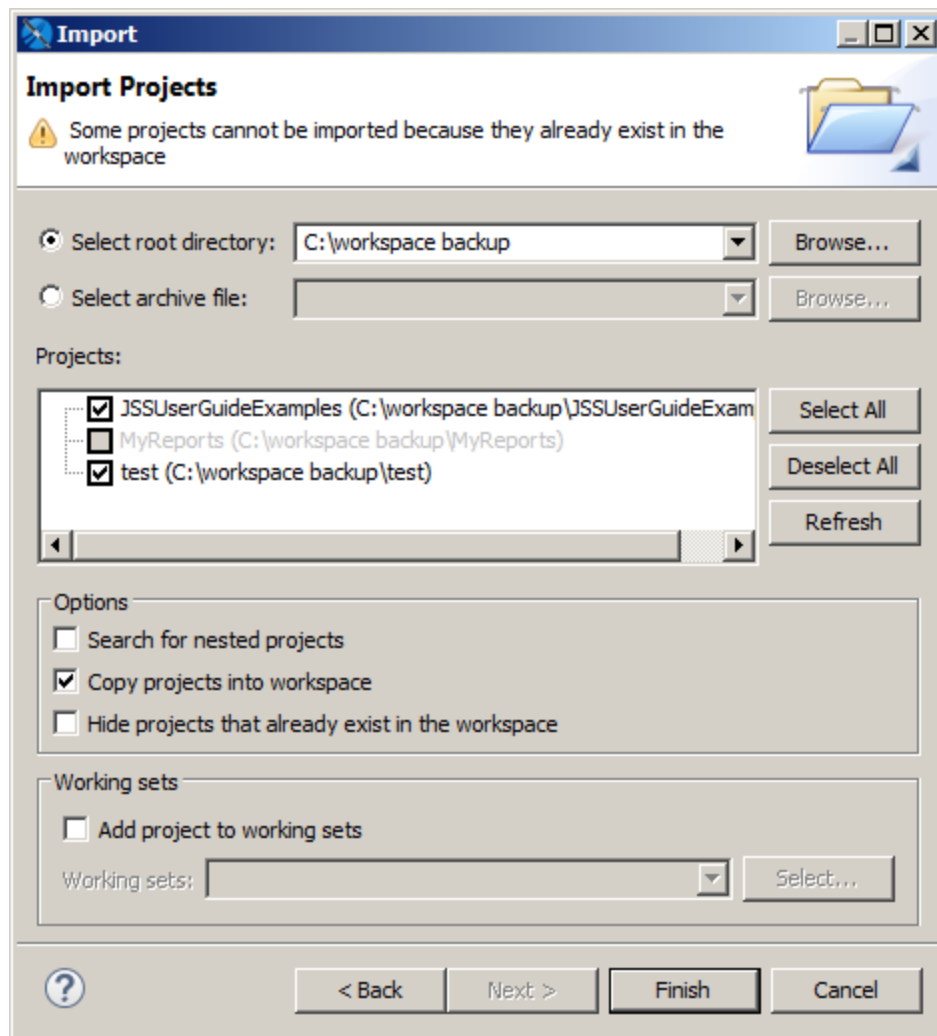


Figure 1-2 Import Projects dialog

5. To work on a copy without modifying the originals, select **Copy projects into workspace**.
6. Click **Finish**.

The projects you selected are imported into your current workspace.

Your workspace contains server connections, global data adapters, and your Jaspersoft Studio preferences in addition to your projects. You must import each type separately.

Importing server connections:

1. Select **File > Import ...**.
2. Select **External JasperReports Server Connections** from the **Jaspersoft Studio** category.
3. Browse to the workspace with you want, click **OK**, and then click **Next**.

The Select the Server Connections dialog opens.

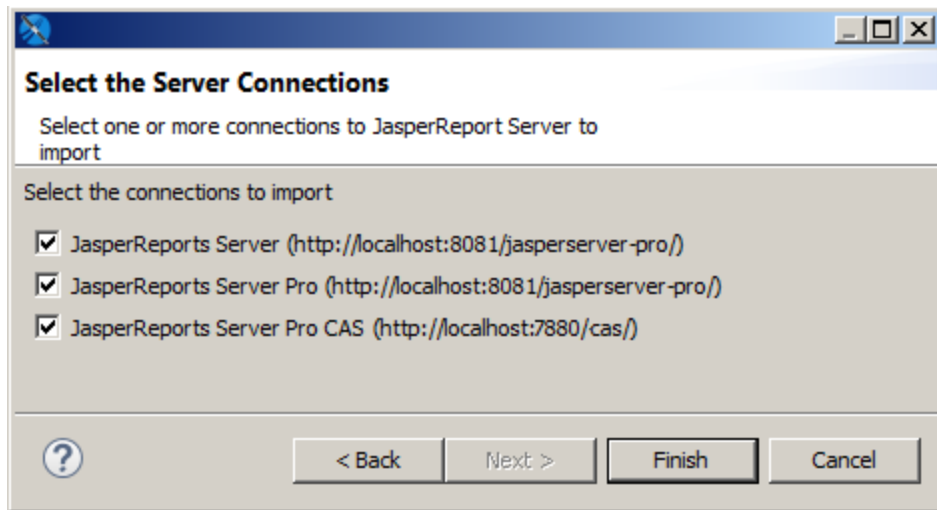


Figure 1-3 Select the Server Connections dialog

4. Choose the connections you want.
5. Click **Finish** to import the connections.

The selected server connections are imported into your Jaspersoft Studio instance.

Importing data adapters and settings:

1. Select **File > Import ...**.
2. Select **External Properties and Data Adapters** from the **Jaspersoft Studio** category.
3. Browse to the workspace with you want, click **OK**, and then click **Next**.

The Select the Data Adapters dialog opens.

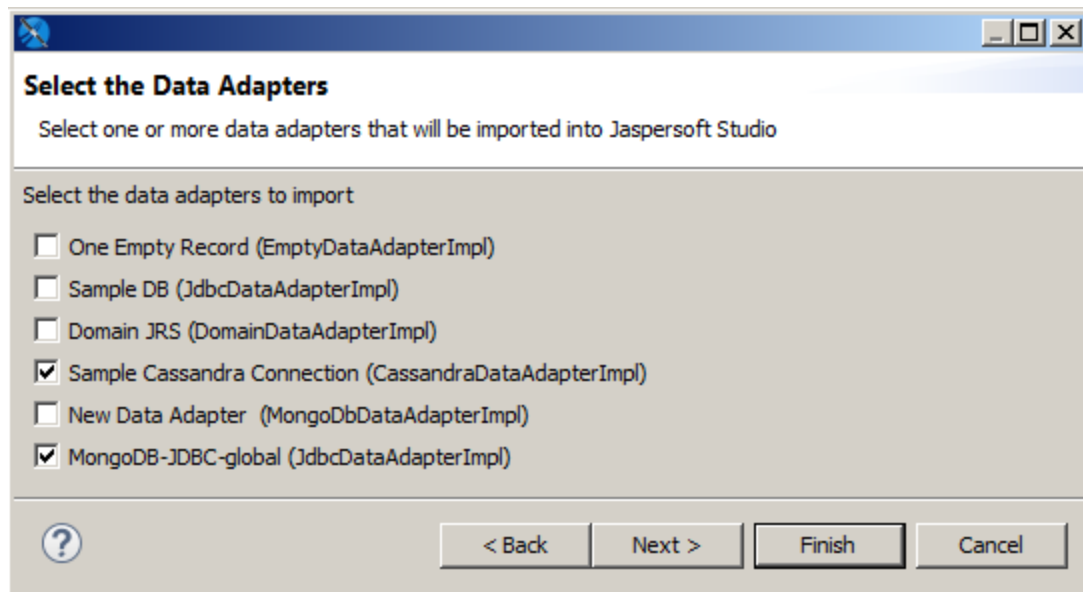


Figure 1-4 Select the Data Adapters dialog

4. Choose the data adapters you want. You do not need to import the built-in adapters (One Empty Record and Sample DB).
5. Click **Next**.

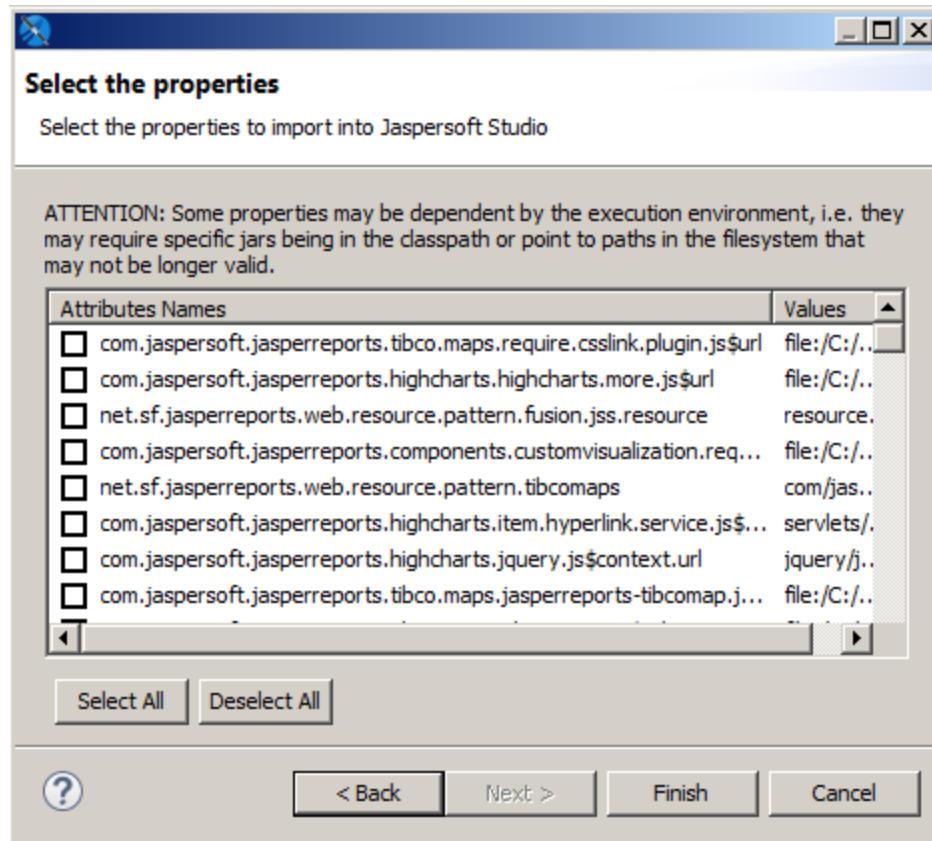


Figure 1-5 Select the properties dialog

6. Choose the properties you want and click **Finish**.
The selected data adapters and properties are imported into your JasperSoft Studio instance.

1.2.5 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 versions of JasperReports Library.

For information on exporting or compiling a report to an earlier version of JasperReports Library, see **22.5, “Setting Compatibility with Earlier Versions of JasperReports Library,” on page 335.**

1.2.6 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 development environment on the Eclipse IDE, and compile and run the source code are described in the tutorial "Contributing to Jaspersoft Studio and building from sources".

CHAPTER 2 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](#)

2.1 Creating a New Report

To create a new report:

1. Go to **File > New > Jasper Report** or click  on the main toolbar.

The **New Report Wizard** window displays the **Report Templates** page. Jaspersoft Studio includes a number of pre-installed templates; you can also create your own. See [Chapter 20, “Report Templates,” on page 313](#) for more information.

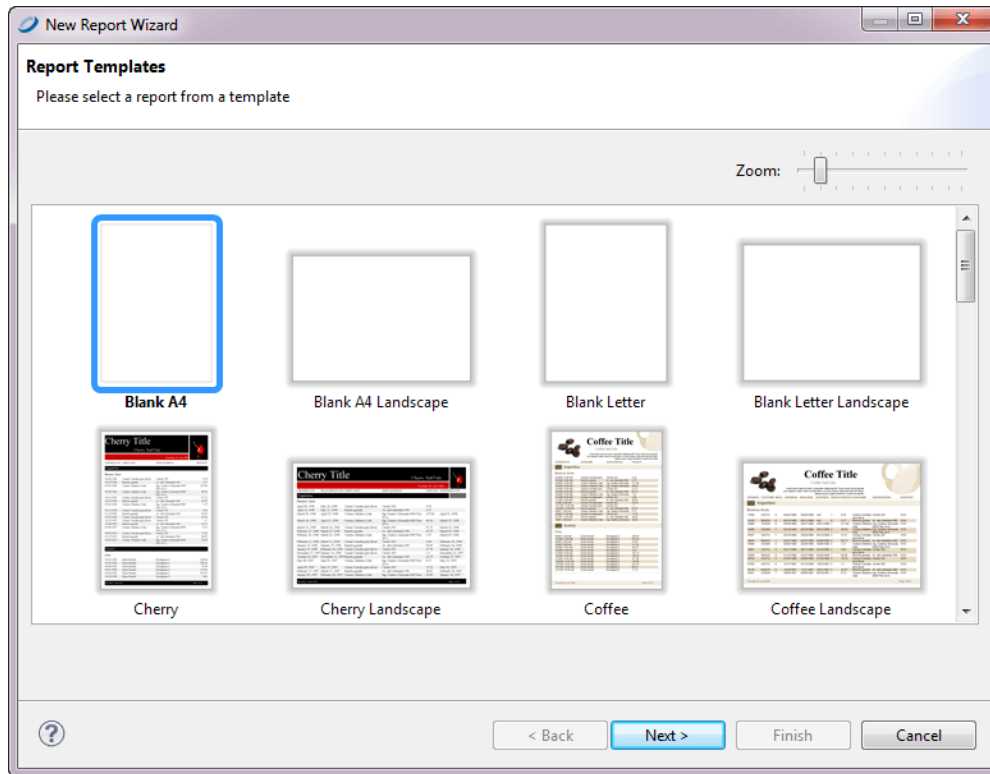


Figure 2-1 New Report Wizard

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

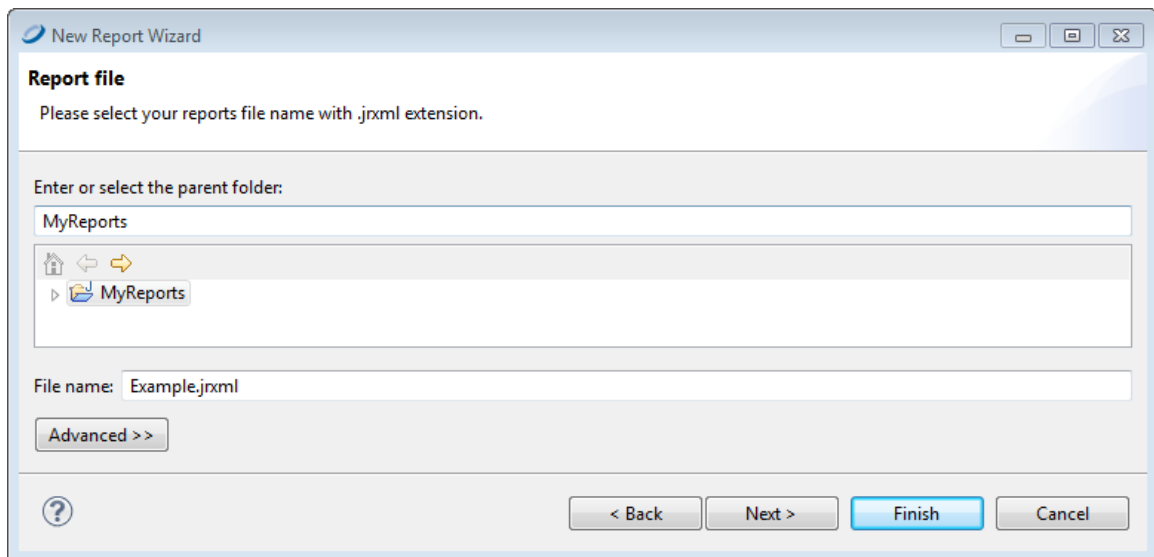



Figure 2-2 New Report Wizard > Report file

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 25**.
 4. Click **Next**.
The **Data Source** dialog opens. This is where you choose the data that will fill the report. The drop-down menu shows the pre-installed data adapters as well any data adapters you have added. The following adapters are pre-installed:
 - **One Empty Record - Empty rows**: Data adapter that lets you create 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 JDBC Connection**: Data adapter that connects to an SQL database provided with the JasperSoft Studio installation. If you are getting your data from a JDBC database, you must also supply an SQL query.
-  You can create a data adapter separately or click **New...** to create a data adapter directly from this dialog. Adapters can be created globally (embedded in the workspace) or local to a specific project. Using a local adapter makes it easier to deploy the report to JasperReports Server. See **10.1, “Creating and Editing Data Adapters,” on page 120** for more information.
5. Choose **Sample DB - Database JDBC Connection**. Enter the query `select * from orders`.

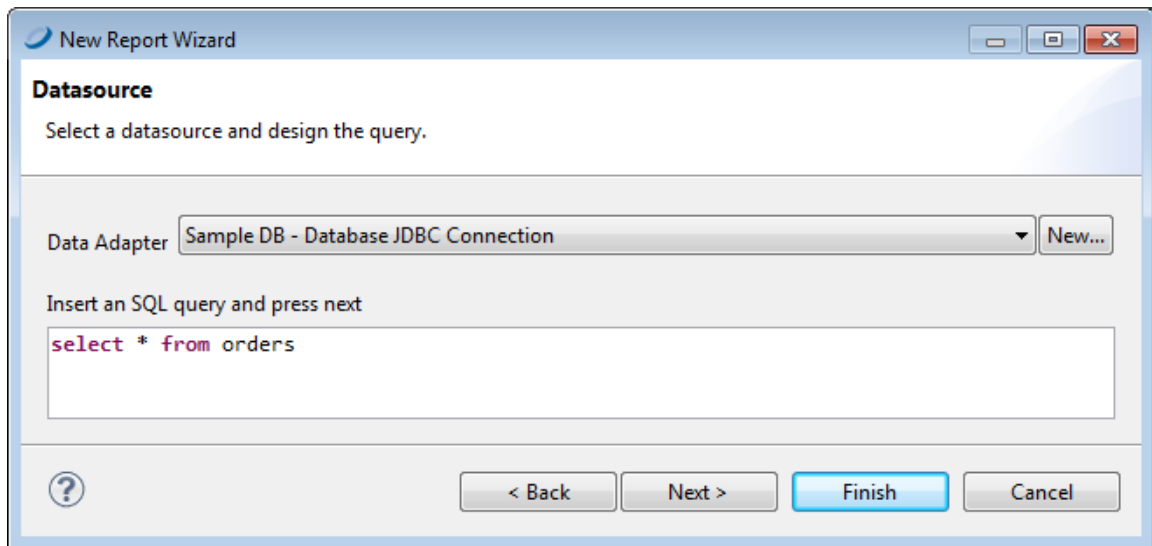


Figure 2-3 New Report Wizard > Data Source

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

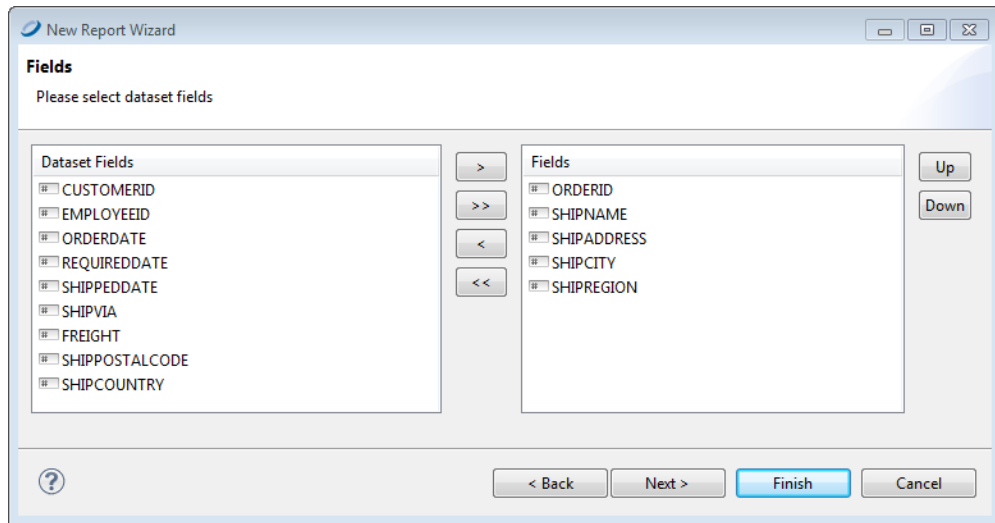


Figure 2-4 New Report Wizard > Fields

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.

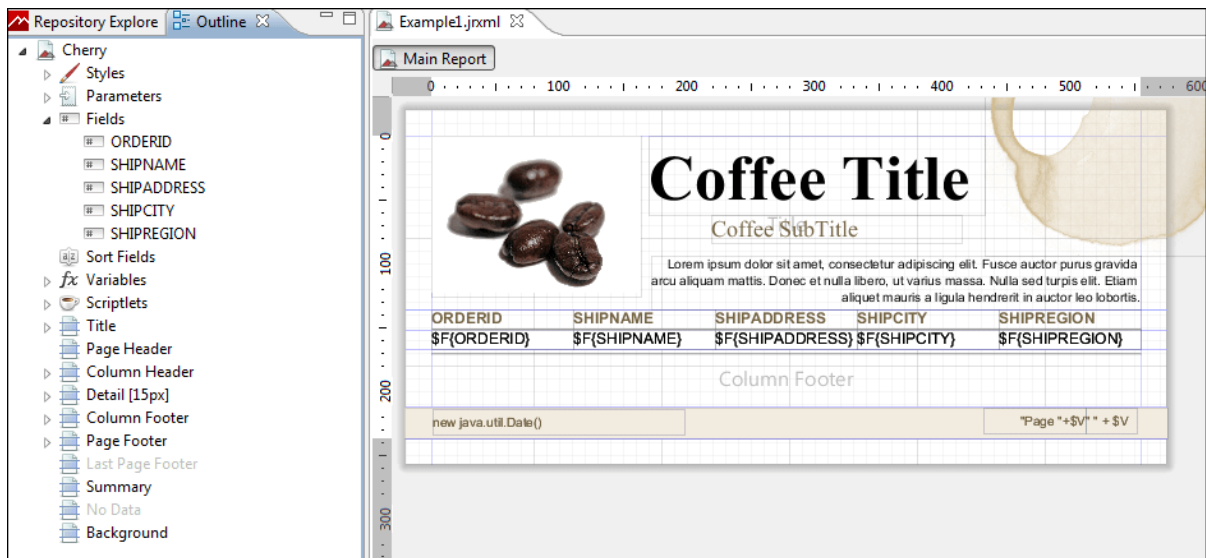


Figure 2-5 New Report in the Design Tab

2.2 Adding and Deleting Report Elements

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

2.2.1 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 **Report** tab in the Properties view and click the **Edit query, filter and sort options** button in the Dataset section.

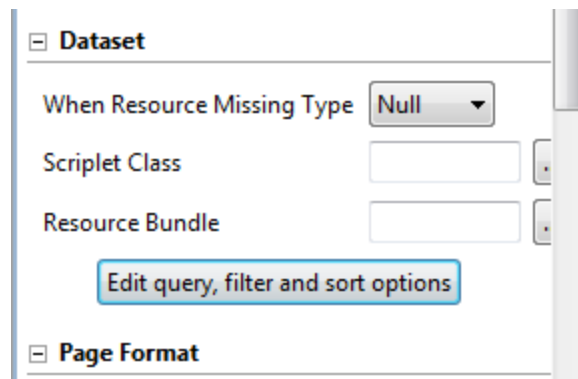


Figure 2-6 Dataset section in properties view

The **Dataset and Query Dialog** opens.

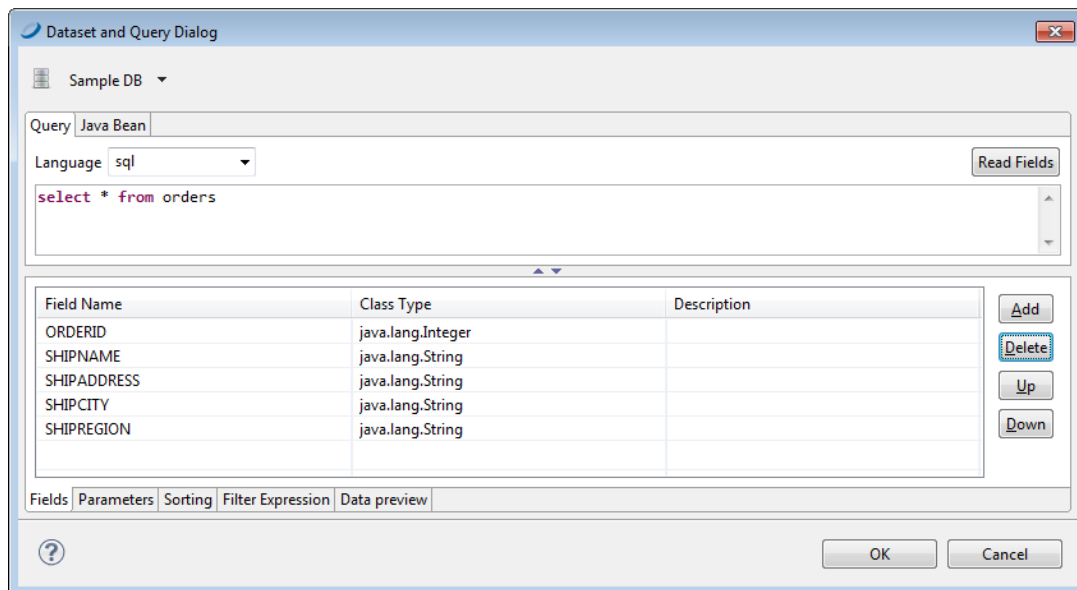


Figure 2-7 Dataset and Query Dialog

3. 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.

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

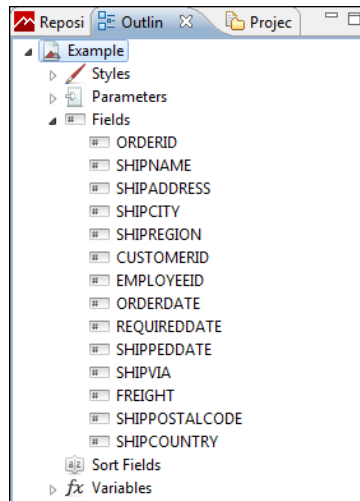


Figure 2-8 Fields

6. 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.

2.2.2 Deleting Fields

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

2.2.3 Adding Other Elements

To add other elements, such as lines, images, or charts, drag the element from the Palette into the **Design**. See [4.2.1, “Inserting Elements,” on page 39](#) for more information.

2.3 Previewing a Report

Click the **Preview** tab at the bottom of the report. 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	Victualles 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. San Cristóbal Carlos Soublette #8-35	Táchira	
10258	Ernst Handel	Kirchgasse 6	Graz	null
10259	Centro comercial Moctezuma	Sierras de Granada 9993	México D.F.	null
10260	Otilies Käseladen	Mehrheimerstr. 369	Köln	null
10261	Que Delícia	Rua da Panificadora, 12	Rio de Janeiro	RJ

Figure 2-9 Report Preview



Each subreport is saved in a separate report file. Reflecting standard Eclipse design, saving or previewing a report that contains subreports does not update the subreports. When you edit a subreport, you must first build the subreport and then save the file in order for the subreport changes to be visible when you preview the report that contains it.

- To build a subreport explicitly, use the **Build All** button on the toolbar, or type **Ctrl-B**. Alternatively, select **Project > Build Automatically** to have Jaspersoft Studio do it for you.
- To save a subreport, use **File > Save** or **File > Save As**.

2.4 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.

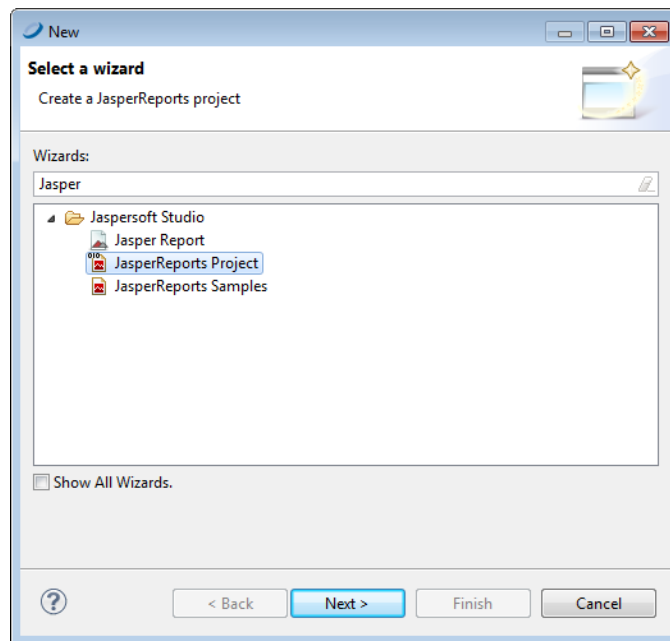


Figure 2-10 Select a Wizard

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.

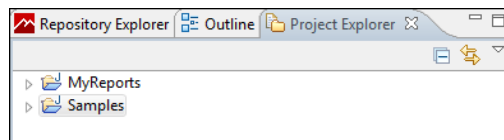


Figure 2-11

CHAPTER 3 USER INTERFACE AND DESIGN VIEW

Jaspersoft Studio is based on the Eclipse platform. If you have worked with Eclipse, you are likely familiar with the user interface. **Figure 3-1** shows a preview of the Jaspersoft Studio interface, with the main areas highlighted. Some views have additional menus and actions, accessed through icons in the upper right of the view.

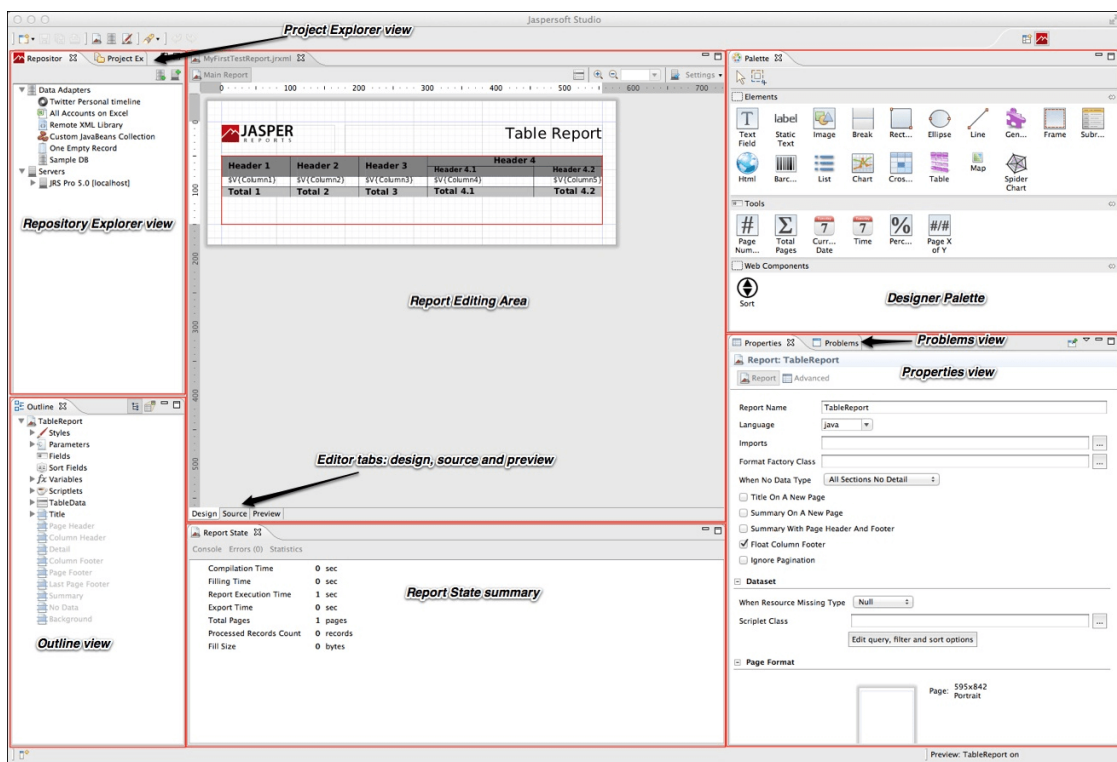


Figure 3-1 Jaspersoft Studio User Interface



3.1 Eclipse Interface

In Eclipse terminology, the initial layout of the Jaspersoft Studio interface is called a perspective. The default Jaspersoft Studio perspective contains an editor area and views. Some views appear by themselves, while others are stacked together in tabbed notebooks. You can open and close views and drag them to different positions in the Eclipse workbench.

- To open a window you have closed, select **Window > Open View** from the menu. Select the window you want to open from the drop-down list.
- To reset the interface to the default perspective, select **Window > Reset Perspective**.
- To save a perspective, select **Window > Save Perspective As** and enter a name for your perspective.

3.1.1 Learning More About Eclipse

If you are not familiar with Eclipse, there are many excellent resources available:

- The Eclipse help is a good place to start. You can access Eclipse help by selecting **Help > Subclipse - Subversion Eclipse Plugin**.
- If you are setting up Eclipse for a team, search the internet for a phrase such as "configuration management for Eclipse".
- To work with version control such as CSV, Git, or SVN, use the corresponding Eclipse perspective included with Jaspersoft Studio. To add a different perspective, click  at the upper right of the Eclipse interface and select the perspective you want from the Open Perspective dialog. Once a perspective has been added, you will see an icon for it at the top right. Use this perspective for all interactions with your version control repository, such as checking out projects, synchronizing files, and resolving conflicts. For information about working with a particular package, use an internet search such as "Eclipse Subversion". To return to the Jaspersoft Studio perspective, click .

3.2 User Interface Components

Jaspersoft Studio has a multi-tab editor, which includes three tabs that allow you to interact with your reports in different ways: 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 a Jaspersoft Studio project.
- The Outline view shows the complete structure of the report in a tree. When the Design or Source tab is active, clicking an element in the Outline view highlights that element in the editor. The Outline tab is empty when the Preview tab is active.
- The Properties view lets you view and edit the properties of the element that is currently selected in the report editor or in the Outline view. The properties shown depend on the type of element. For example, the Properties view for a table shows four tabs: Appearance, Dataset, Table, and Advanced, while the Properties view for a line shows Appearance, Borders, Line, Inheritance, and Advanced. Some properties are read-only,

but most are editable. When the root node of a report is selected in Outline view, the Properties view shows the properties for the report.

Unlike many other views, you can open multiple instances of the Properties view at one time and you can pin a selection to a specific Properties view instance. This allows you to view or edit the properties for a specific element while working with other elements in your report, or with another report entirely.

- 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 3-1 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

3.3 The Design Tab

You design a report using the Design tab, which is divided into different horizontal portions, named bands, where you can place report elements. When the report design is combined with the data to generate the print, each band is printed multiple times based on its 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.

Jaspersoft Studio provides a graphical interface for creating JRXML files. The layout is visual, so you can ignore the underlying structure of the JRXML. You can specify the precise page locations of different types of text and data, such as title, footers, detailed records, groups, and summary information. Some portions of a page defined in this way are reused, others stretch to fit the content, and so on. Additional tools let you add charts and subreports, set up an optional query retrieve data out of a data source, and more.

3.4 Understanding Bands

The Design tab 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.

3.4.1 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 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.

Band Name	Description
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.

3.5 Specifying Report Properties

To view or edit report properties, select the report root node in the Outline view. The report properties are shown in the Properties view.

To change the page dimensions of a report, click the Report tab in the Properties view for the report, then click **Edit Page Format**. In the Page Format dialog that appears, you can edit the width, height, units, orientation and margins of the report.

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. The following table shows standard page sizes and their dimensions in pixels.

Page Type	Dimensions in Pixels
Letter	612x792
Note	540x720
Legal	612x1008
A0	2380x3368
A1	1684x3368
A2	1190x1684

Page Type	Dimensions in Pixels
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
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.

3.5.1 Columns

Pages, one or more of which make up a report, present bands that 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 a 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 JDBC 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 compiled report.

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 columns are 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 than or equal to the page width. If this condition is not met, the compilation results in an error.

3.5.2 Advanced Options

From the **Properties** view 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** view you see:

- **Report Name:** It is a logical name, independent from 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:
 - **No Pages:** This is the default value; the final result is an empty buffer.
 - **Blank Page:** This returns an empty page.
 - **All Sections No Detail:** This returns a page containing all bands except for the detail band.

3.6 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.

3.6.1 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.

3.6.2 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.

3.6.3 Choosing Report Templates for PDF

If you are exporting your report to PDF, choose a report template based on the size of the output.

- For most PDF exports, you can use Actual Size, which supports a maximum size of 14400px by 14400px.
- For reports with an output height exceeding 14400 px, use a paginated report template that is wide enough for your report. For example, if you have a long report with width less than 842px, you can use the paginated A4 Landscape theme. A report designer can create additional custom templates in JasperSoft Studio.
- Reports with output width exceeding 14400 px will be truncated in PDF.

CHAPTER 4 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:

- **Common Element Properties**
- **Inserting, Selecting, and Positioning Elements**
- **Formatting Elements**
- **Graphic Elements**
- **Text Elements**
- **Frames**
- **Working with Composite Elements**
- **Anchors, Bookmarks, and Hyperlinks**
- **Custom Visualization Component**

4.1 Common Element Properties

All 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 view (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.

4.1.1 The Palette

Elements appear in the Palette, located by default in the top right of the UI.

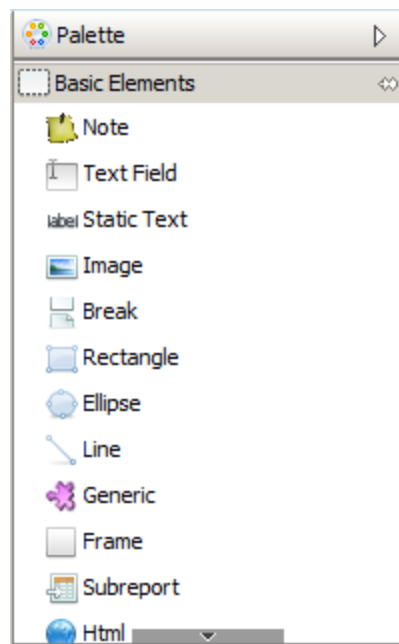


Figure 4-1 Elements in the Palette

The palette contains three subpalettes:

- Basic Elements contains the elements and components available in all editions of Jaspersoft Studio.
- Composite Elements contains elements created as combinations of other elements, such as Page Number and Time. You can add your own composite elements to any palette.
- Components Pro contains elements only available in commercial versions of Jaspersoft Studio. This subpalette is not visible in the community edition.

4.1.2 Element Properties

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

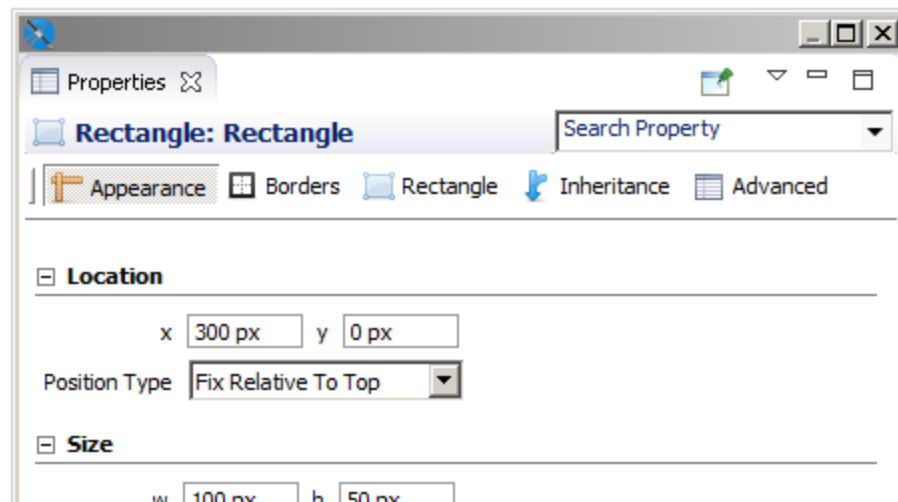


Figure 4-2 Properties view for a rectangle

- The **Appearance** tab allows you to set the location, size, color, and text style of the element.
- The **Borders** tab allows you to set the padding and border style, color, and width of the element.
- An element tab allows you to set evaluation time along with properties specific to the element type. For example:
 - The **Static Text** tab allows you to define unchangeable text for a field, and control its appearance.
 - The **Text Field** tab allows you to format and position a text field element.
 - The **Image** tab allows you to set image alignment, fill, and scale properties.
- Some elements have more than one element-specific tab. For example, the Chart component has the **Chart** and **Chart Plot** tabs, and the Map component has the **Map**, **Authentication**, **Markers**, and **Paths** tabs.
- The **Inheritance** tab allows you to view any attributes inherited from another level, and override those attributes when possible.
- The **Hyperlink** tab, available for image, text field, and chart elements, allows you to define a hyperlink in an element.
- The **Advanced** tab displays detailed information about the element.

Frequently, 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 a report in the design tab, you can see that it has a white background. The value of the background color attribute is inherited from a lower level.

4.2 Inserting, Selecting, and Positioning Elements


4.2.1 Inserting Elements

When you insert an element, you can let Jaspersoft Studio autosize it, or you can size it as you insert it. Setting the size of an element when you insert it is useful for tabular elements such as tables and crosstabs.

To let Jaspersoft Studio autosize an element:

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

To size an element at insertion time:

- Click on the element in the palette. The cursor changes  to show that an element is selected. Click and drag in the report editing area to size and place the element. If you insert a crosstab or table using click and drag, the columns fill the whole crosstab or table.

4.2.2 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 drag the cursor in a rectangle around them. 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.
- Shift-click to select the parent of the current object. For example, shift-click an element contained directly in a band to select the band.

4.2.3 Positioning Elements

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

4.2.3.1 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**.

4.2.3.2 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.

4.2.3.3 Guides

When dragging or resizing an element, Jaspersoft Studio suggests places to align it based on the elements currently in the Design tab, 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.

4.2.3.4 The Properties View

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

4.2.4 Positioning Elements in Containers

Some elements that can contain many other elements are called containers. Containers include bands, frames, table cells, and crosstab cells. The following tools help you position items inside containers:

- Sizing tools – Let you size an element to fit the height, width, or entire container.
- Container layouts – Let you set how elements are automatically arranged in a container.

Elements inside containers must obey the following rules.

- Elements in table cell, and crosstab cells must be fully contained by the parent in the design time. Otherwise, an error will occur at compilation time.
- Elements in bands can extend horizontally past the document margins and/or overflow the top of the band. Otherwise, an error will occur at compilation time.
- Frames are able to adapt their size to content.

4.2.4.1 Container Layouts

A container layout is a design-time tool that adjusts the size and the position of elements when they are added to or removed from a container. The concept of layout is specific to Jaspersoft Studio and works only at design time. Layouts don't make a report stretchable or resizable. At run-time, depending on the design, JasperReports Library may still let elements overlap or change their position relative to other elements.

There are four container layouts:

- Free layout (default)
- Horizontal layout
- Vertical layout
- Grid layout

To choose a layout:

- Right-click in the container select Arrange in Container from the menu, then select the layout you want.
or
- Click on the container and then select the option you want from the Layouts menu on the Appearance tab of the Properties view. This is the only way to return to Free Layout after you have selected a different layout.

4.2.4.2 Working with Grid Layout

Grid layout positions elements in a container in a grid of rows and columns. By setting properties on individual elements, you can control the element's placement in the grid, as well as influence the overall height of the rows and width of the columns. Elements can span multiple rows and/or columns. If you resize the container during design, the elements are resized based on their properties.

When grid layout is selected for a container, such as a band, elements inside the container have a Layout section on the Appearance tab of the Properties dialog. The following table shows the properties you can set on an element in a container with grid layout. The property name to use in source view is included in the description.

Property	Value	Description
Row Number	Relative (default) or an integer between 0 and 1000	Number of the row from which this element starts. 0 is the first row. When set to Relative, increments the last evaluated row by 1. com.jaspersoft.layout.grid.y
Column Number	Relative (default) or an integer between 0 and 250	Number of the column from which this element starts. 0 is the first column. When set to Relative, increments the last evaluated column by 1. com.jaspersoft.layout.grid.x
Row Span	Integer between 0 and 1000; default = 1	Number of rows that the element spans. com.jaspersoft.layout.grid.colspan
Column Span	Integer between 0 and 250; default = 1	Number of columns that the element spans com.jaspersoft.layout.grid.rowspan
Fixed Size	Boolean; default = false.	Set to true to manually size the element. Set to false to have the element size automatically using the element's settings com.jaspersoft.layout.grid.fixed
Row Weight	Number; default = 1	Number that specifies how much space the element's row takes relative to other rows. Not available when Fixed Size is True. com.jaspersoft.layout.grid.weight.x
Column Weight	Number; default = 1	Number that specifies how much space the element's column takes relative to other rows. Not available when Fixed Size is True. com.jaspersoft.layout.grid.weight.y

To use grid layout:

1. This example uses a vertical image, that is, an image much taller than it is wide. You can use any vertical image, for example, a company logo rotated vertically. To create the exact image used in this example, create a report with the Green Leaf template. This creates a leaf_banner_green.png file in your workspace. In your file system, use a graphics editor to rotate the image 90°. Note that this will rotate the image in any report where it is used.
2. Create a report using the BlankA4 template and the Empty data source. Do not reuse the report created in the previous step.
3. Add your vertical image to the title band of your report.
4. Add a chart to the title band of your report, to the right of your image.
5. Resize the title band to fit a chart.

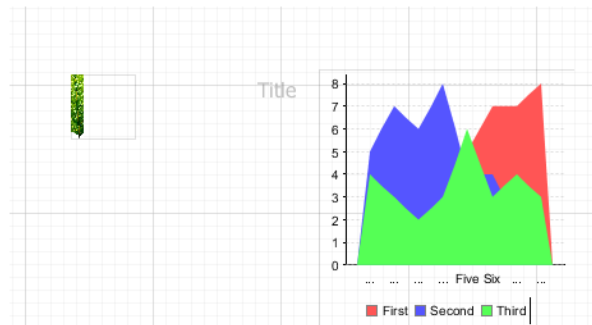


Figure 4-3 Title band before applying grid layout

6. Right-click in a blank space in the Title band and select **Arrange in Container > Grid Layout**, or select the Title band and select **Grid Layout** in the Properties view.
The two elements are arranged to fill the band equally.

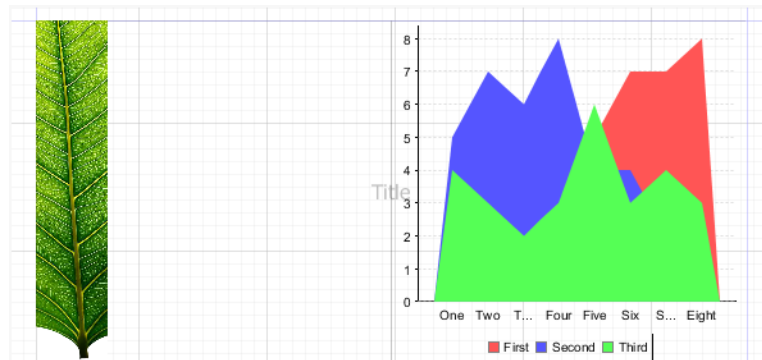


Figure 4-4 Title band with grid layout

7. Resize the elements so that the chart takes up most of the space. To do this, select the chart. In Properties view, in the Layout section of the Appearance tab, set **Column Weight** to 5.
The elements adjust so that the chart width is five times the image width.

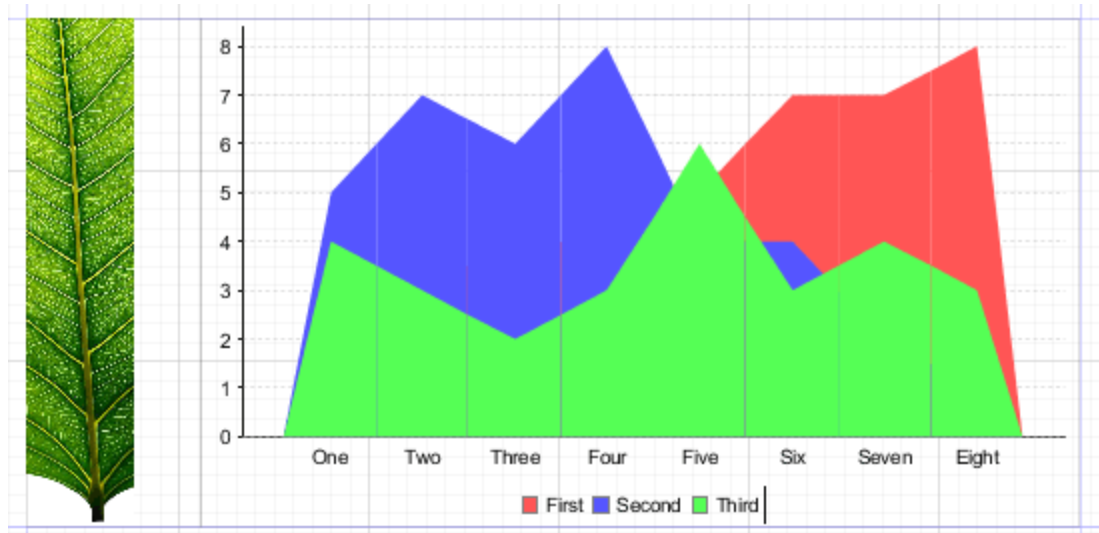


Figure 4-5 Grid layout with column weight

8. Now add a static text element to the far right of the title band.
The static text is added at the end of the first row.

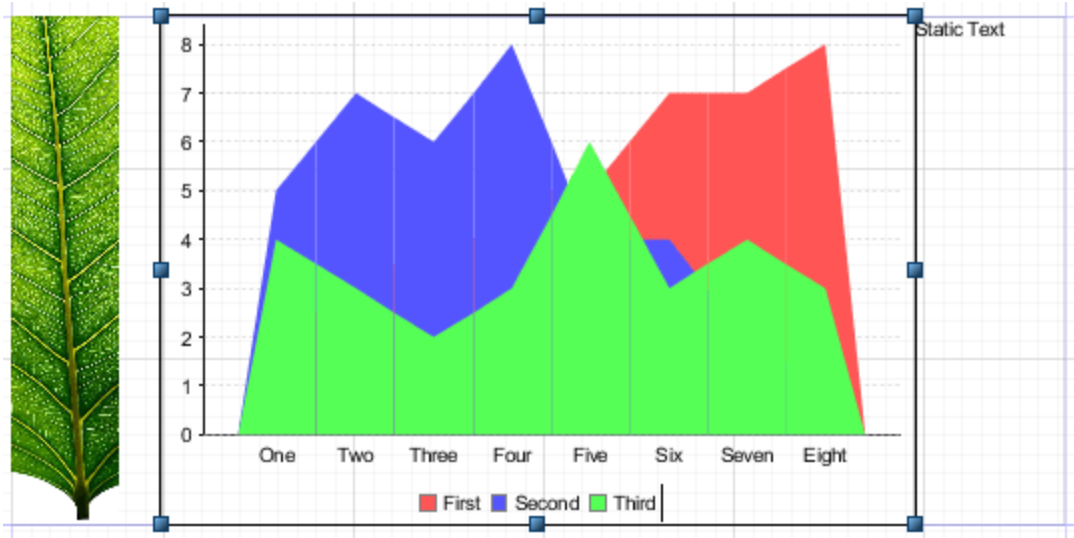


Figure 4-6 Adding an element to a grid layout

9. Position the static text. To do this, select the static text. In Properties view, in the Layout section of the Appearance tab, set the following:
 - Set **Row Number** to 1 to move the element to the second row. You could also have added the static text directly below the first row, but setting the row explicitly gives you more control.
 - Set **Column Span** to 2 to have the element span both columns. You could instead set the Column Number to 2 to move the static text under the chart.

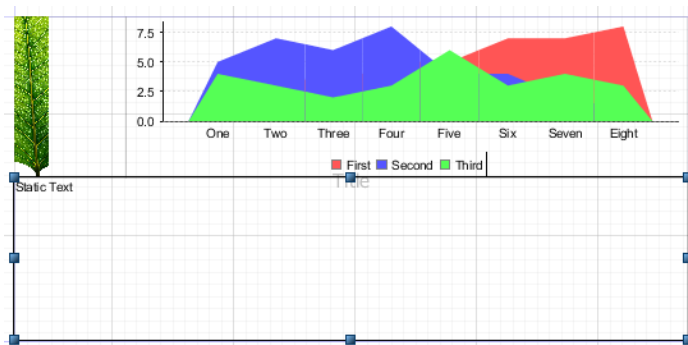


Figure 4-7 Using two rows in grid layout

- Set the relative heights of the rows. To do this, select the chart and set **Row Weight** to 10 in the Layout section of the Appearance tab of Properties view. You could actually do this by changing the settings on any of the three elements, but in this case, the chart is the main element and you want other elements to adjust to it.

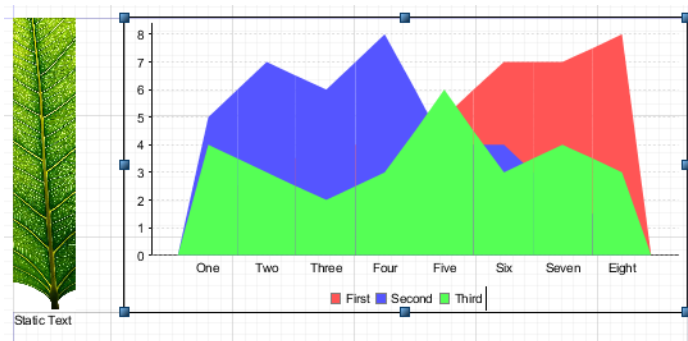


Figure 4-8 Using row weight in grid layout

4.3 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.

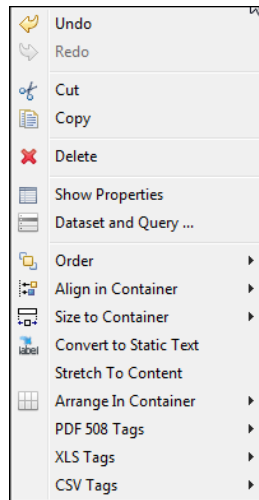


Figure 4-9 Formatting Tools Menu

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



















 A container is the band, frame, or cell that contains the element.

Table 4-1 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
	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

Icon	Tool Name	Description	Multiple Select?
	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
	Grid Layout	Positions elements in a grid based on properties set on each element.	Yes
Miscellaneous Tools			
	Stretch to Content	Resizes element to fit the content	N/A
	PDF 508 Tags	Adds tags required for PDF 508C compliance	
	XLS Tags	Adds tags that define how data is exported to the Microsoft Excel format	

4.4 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 palette to the report editing area.

4.4.1 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.

4.4.2 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.

4.4.3 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).

4.4.4 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.

4.5 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 [4.9.4, “Creating a Hyperlink,” on page 60](#) for more information.

4.5.1 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.

4.5.2 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		
<code>java.lang.Object</code>	<code>java.sql.Time</code>	<code>java.lang.Long</code>

Valid Expression Types		
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 <code>Text field Expression</code> has to be elaborated.
Evaluation group	The group to which the evaluation time is referred if it is set to <code>Group</code> .
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.

4.6 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 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 Outline view and drag elements from a band into the frame. 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.

4.7 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 tab, 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.

4.8 Working with Composite Elements

Composite elements are one or more pre-configured elements that you can use in your reports. You can configure properties such as the size, color, or font of an element, or create a text field with a complex expression you frequently use, and then save it as a composite element. Jaspersoft Studio also ships with several pre-existing composite elements, such as page number and total pages.

Some element types are hard to reuse in other reports, in particular, those elements that depend on the availability of a specific subdataset having certain fields. You cannot include elements based on a dataset in a composite element. In particular, composite elements can not include charts, tables, lists and crosstabs. Composite elements can include notes, text fields, static text, images, breaks, rectangles, ellipses, lines, frames (must contain only permitted elements), barcodes, HTML elements, and other composite elements.

If your composite element contains elements that use expressions (text-field expressions or print-when expressions), the objects you are referencing in those expressions (such as variables, fields or parameters) should be available in the report in which you use the composite elements. If the objects are not present, you may receive an error when compiling or previewing the report.




Composite elements cannot include elements based on a dataset, such as charts or crosstabs.

4.8.1 Creating and Editing Composite Elements

To create a composite element:

1. Open or create a report.

For example:


- a. Go to **File > New > Jasper Report** or click  on the main toolbar.
- b. In the New Report Wizard, select Blank A4 in the Report Templates window and click **Next**.
- c. Select a name and location for your file (for example, Composite Element Sample Report in MyReports) and click **Next**.

- d. Choose **One Empty Record** in the Data Source window and click **Finish**.
2. Place the elements you want in the Title band and format and position them.



Composite elements must be created from the Title band.

For example, to create a footer that includes your company name and the page number:

- a. Drag the Static Text element to the Title band in your report and type My Company. Then align the company name to the left by right-clicking the Static Text element and selecting **Align in Container > Align to Left Margin**.
- b. Drag the Page Number element to the Title band in your report. Align the Page Number to the right by right-clicking the Page Number element and selecting **Align in Container > Align to Right Margin**. Then, with the Page Number element selected, go to the **Text Field** tab in the Properties view and click  to align the text right.
- c. Select both elements, right-click, and choose **Align Components > Align Top**.
3. Select all the elements you want in your composite.
4. (Optional) To have your elements move together, right-click and select **Enclose into Frame** from the context menu.
5. Make sure all elements are still selected.

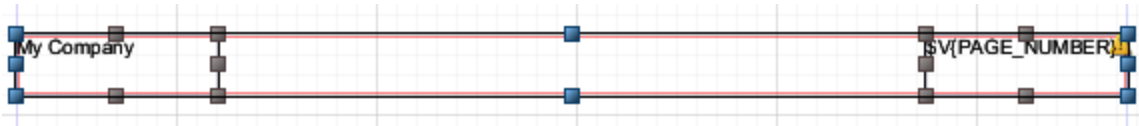


Figure 4-10 Selected Elements For Composite Element Creation

6. Right-click and select **Save as Composite Element**.
The Composite Element Settings dialog box opens.

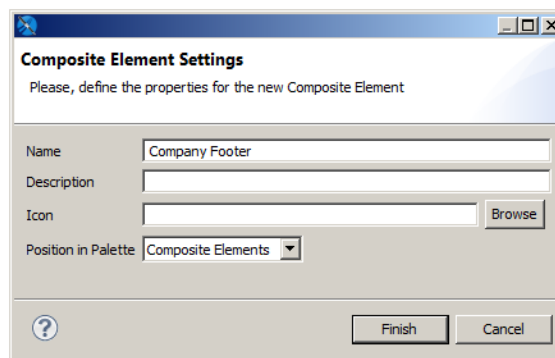



Figure 4-11 Composite Element Settings Dialog Box

7. Enter the following information:
 - **Name:** Enter a unique name you want to appear in the palette.

- **Description** (optional): Enter a description. If the element uses text fields or expressions, it may be useful to mention these, or the expected data adapter, in the description.
 - **Icon** (optional): Choose the icon that will show in the palette for this composite element. You can choose an icon in JPG, PNG, or GIF format. If you click Browse to locate an icon, and you want to use a PNG or a GIF, you must choose the correct format at the bottom right of the file open dialog. If you do not choose an icon, Jaspersoft Studio uses the default icon .
 - **Position in Palette**: Select one of Basic Elements, Composite Elements, or Components Pro.
8. Click **Finish**.
 9. Click **OK** on the confirmation message.
- The new composite element is saved as a .jrtool file in the same location as your report. An icon is added to the bottom of the subpalette you selected.

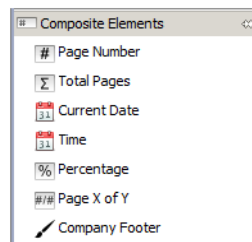


Figure 4-12 Composite Element in the Palette

To edit the contents of a composite element:

1. Right-click on the composite element in the palette and select **Open in Designer**.
The composite element opens in the Designer as a .jrtool file.
2. You can add and remove elements and change element formatting. If you add elements, remember to select all elements and create a frame.
3. Save the file.

To edit the name or location of a composite element:

1. Right-click on the composite element in the palette and select **Edit**.
The Composite Element Settings dialog box opens.
2. Change the name, description, icon, or position in palette.
3. Click **Finish**.
4. Click **OK** on the confirmation message.

To delete a composite element you have created:

1. Right-click on the composite element in the palette and select **Delete**.
The element is removed from the palette.




You cannot delete the composite elements that are shipped with Jaspersoft Studio.

4.8.2 Exporting and Importing Composite Elements

You can share composite elements between Jaspersoft Studio installations using import/export.

To export one or more composite elements:

1. Right-click on a custom composite element in the palette.
2. Select  **Export Composite Elements** from the context menu.

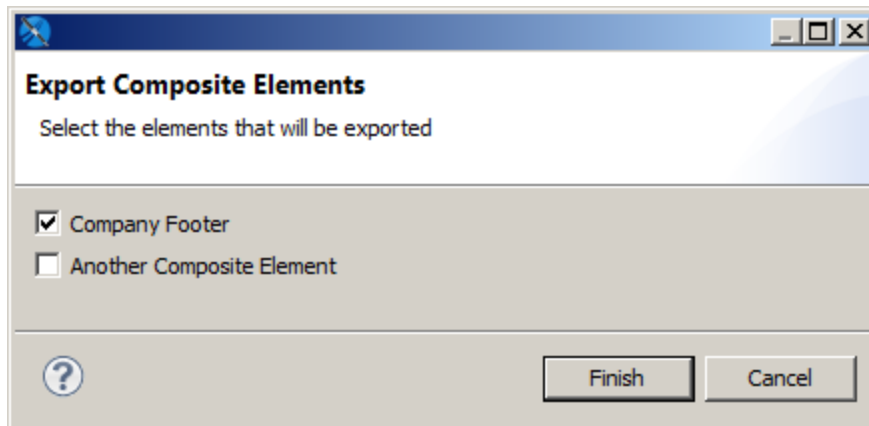



Figure 4-13 Exporting Composite Elements

3. Select the elements you want to export in the Export Composite Elements dialog.
4. Click **Finish**.
5. When prompted, navigate to the location where you want to save the export file, and click OK.
The selected composite elements are saved as a .zip file in the location you chose.

To import a composite element set:

1. Right-click on any element in the palette.
2. Select  **Import Composite Elements** from the context menu.
3. Navigate to the location where your zip file is stored, select the file you want to upload, and click **Open**.
4. Choose a name, optional description, and optional icon in the Import Composite Element dialog and select the palette where you want the composite element to appear. The name in the palette must be unique.

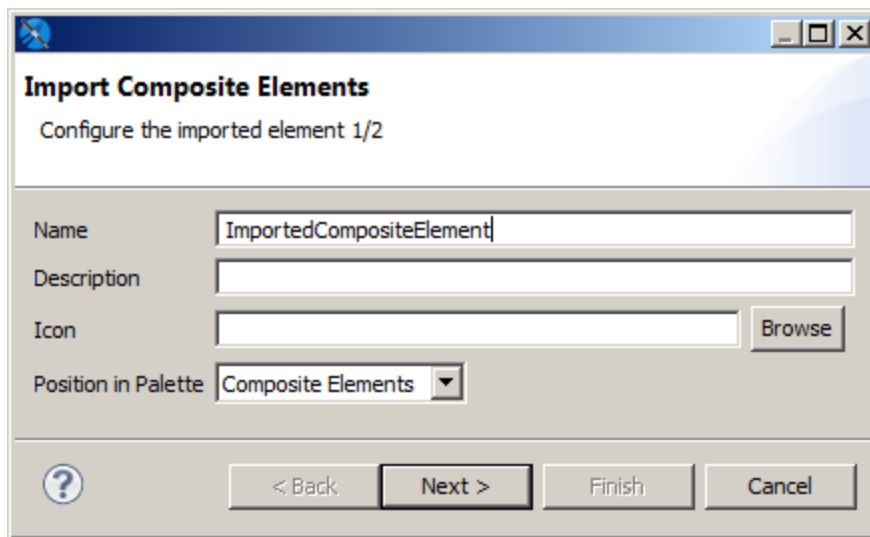


Figure 4-14 Importing Composite Elements

5. If there are additional elements in the file, click **Next** and configure the next element as in the previous step.
6. When you have configured all your imported elements, click **Finish**.
The composite elements are placed in the palette with the settings you configured.

4.9 Anchors, Bookmarks, and Hyperlinks

JasperReports Library 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.

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



This section describes hyperlinks for images, text fields, and charts. Hyperlinks for HTML5 charts are defined differently, as described in [15.5, “Hyperlinks in HTML5 Charts,” on page 241](#).

To set anchor, bookmark, or hyperlink properties, select an image, text field, or chart element and go to the **Hyperlink** tab in the Properties view.

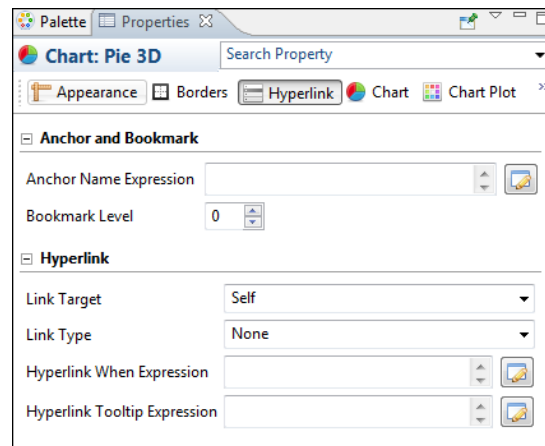



Figure 4-15 Anchor, Bookmark, and Hyperlink Properties

This window is divided in two sections:

- Anchor and Bookmark
- Hyperlink

4.9.1 Anchors and Bookmarks

An anchor identifies a specific position in a document. If you plan to export your report to PDF, you can optionally set a bookmark level to have the anchor show up as a PDF bookmark. The Anchor and Bookmark area lets you set the following:

- **Anchor Name Expression** – Expression for the name of the anchor. This name can be referenced by other hyperlinks. Click the  button to open the **Expression Editor**, where you can write or build the expression.
- **Bookmark Level** – Bookmark level when the report is exported to PDF. 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 as a bookmark, simply choose a bookmark level higher than 1. Defining different levels creates nested bookmarks.

Bookmarks can also be displayed in JasperReports Server when the report is displayed in the interactive viewer. To display bookmarks in the server, set the following property:

```
<property name="net.sf.jasperreports.print.create.bookmarks" value="true"/>
```

This property can be set on the report level or globally in the JasperReports Server WEB-INF\classes\jasperreports.properties file.



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

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

4.9.2 Hyperlinks

Hyperlinks let you link a location in a report to another destination. The most important property of a hyperlink is its type, which determines the format of the target. JasperSoft Studio supports the following types of hyperlink: The exact properties of a hyperlink depend on the hyperlink type. The following properties may appear for a hyperlink:

- **Link Target** – Specifies where to open the link target. The Link Target is similar to the target attribute of an HTML link. The dropdown box shows the following options: Self, Blank, Top, Parent. You can also possibly specify a target name, which actually makes sense only when the hyperlink is used in a web environment.
- **Link Type** – The following link types are supported in JasperSoft Studio: Reference, Local Anchor, Local Page, Remote Anchor, Remote Page, and ReportExecution. You can also define your own custom hyperlink types.
- **Target Expressions** – Expressions that determine the location of the link target. May include:
 - **Hyperlink Anchor Expression** – Anchor in document to use as hyperlink target.
 - **Hyperlink Page Expression** – Page in document to use as hyperlink target.
 - **Hyperlink Reference Expression** – Location of remote document. For link of type Reference, use a URL; for a link of type Remote Anchor or Remote Page, use a file reference.
- **Hyperlink When Expression** – Expression that determines when the hyperlink is implemented. A hyperlink is only available if the Hyperlink When expression returns the Boolean value `True` (default).
- **Tooltip Expression** – String to use as a tooltip when a user hovers the cursor over the hyperlink.
- **Parameters** – Parameters that specify information about the target; only available for ReportExecution hyperlinks and custom hyperlink types.

ReportExecution is implemented as a custom hyperlink type in JasperReports Library.

4.9.2.1 Linking to a URL

To link to a URL, <http://www.jaspersoft.com/>, select **Reference** from the Link Type dropdown in the Properties view. Hyperlinks of type Reference are rendered in all output formats that support web links, including Microsoft Excel and Word.

When working with a hyperlink of type Reference, you can add parameters via the Hyperlink Reference Expression. For example, the following expression uses the values of the `city` and `country` fields to dynamically build a URL:

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

4.9.2.2 Linking to a Report

Several hyperlink types link to an existing report. These types are primarily supported in PDF and HTML formats:

- **LocalAnchor** – Links 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, it is necessary to specify a hyperlink anchor expression, which will have to produce a valid anchor name, for example, "title".
- **LocalPage** – Point to a specific page in the current report. In this case, it is necessary to specify the page number you are pointing to by means of a hyperlink page expression, for example `Integer.valueOf(2)`. The expression must return an Integer object.
- **RemoteAnchor** – Points to an anchor that resides in an external document. In this case, the location of the external file 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** – Points to a particular page of an external document. In this case the location of the external file must be specified in the Hyperlink Reference Expression field, and the page number must be specified in the Hyperlink Reference Expression.

Similar to links of type Reference, you can specify additional parameters for these hyperlink types by appending them to the expression string.

4.9.2.3 Creating a Link Between Reports on a JasperReports Server Instance

Hyperlinks of type ReportExecution execute one JasperReports Server report from another JasperReports Server report, for example, when drilling down to a report in the context of JasperReports Server. Instead of a hyperlink reference or similar expression, ReportExecution hyperlinks use JasperReports parameters to specify the target. The following report-execution parameters are available:

- **_report** (required) – String that points to the JasperReports Server report to execute. Usually a path on JasperReports Server, enclosed in quotes, such as `"/public/Samples/Reports/myReport"`
- **_page** (optional) – Specifies a page to display in the target report. Only one of **_page** and **_anchor** should be used. If both are used, **_page** takes precedence and **_anchor** is ignored.
- **_anchor** (optional) – Specifies a named anchor to display in the target report.
- **_output** (optional) – Specifies an output format for the report, such as PDF, DOCX, etc. Default is HTML.
- If the destination report contains one or more input controls, their value can be set by specifying the name of the input control as a parameter name and providing a value.



ReportExecution hyperlinks are an example of custom hyperlink extensions in JasperReports Library. More generally, these parameters can be used by URL Handlers, especially JasperReports Library extensions, to manipulate and generate hyperlinks.

Creating a ReportExecution hyperlink by dragging:

The easiest way to configure a ReportExecution hyperlink in JasperSoft Studio is to drag a report unit from the JasperReports Server repository explorer over an element that supports hyperlinks (such as a textfield).

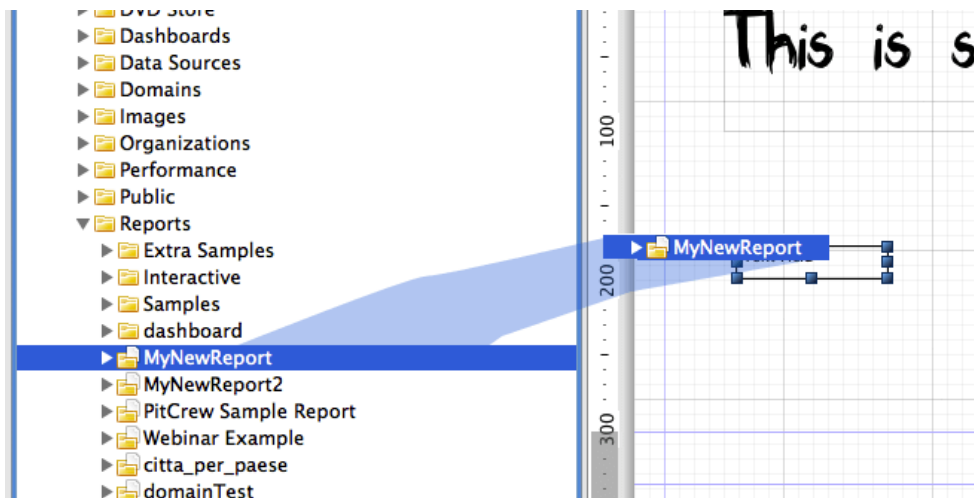


Figure 4-16 Dragging a report from the server into the Report Design

The auto-configured hyperlink automatically sets the type to ReportExecution and configures the `_report` parameter. 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.

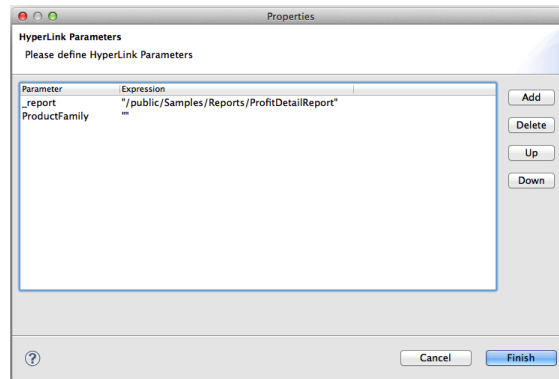


Figure 4-17 Configuring Hyperlink Parameters

4.9.3 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 servlet to manage a record drill-down tool. 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.


4.9.4 Creating a Hyperlink

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 a hyperlink:

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 59** for an explanation of the different choices.
4. Click the  button next to **Hyperlink Tool Expression** to create a tooltip for your hyperlink.
5. Save your report.

4.10 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).

4.11 Custom Visualization Component

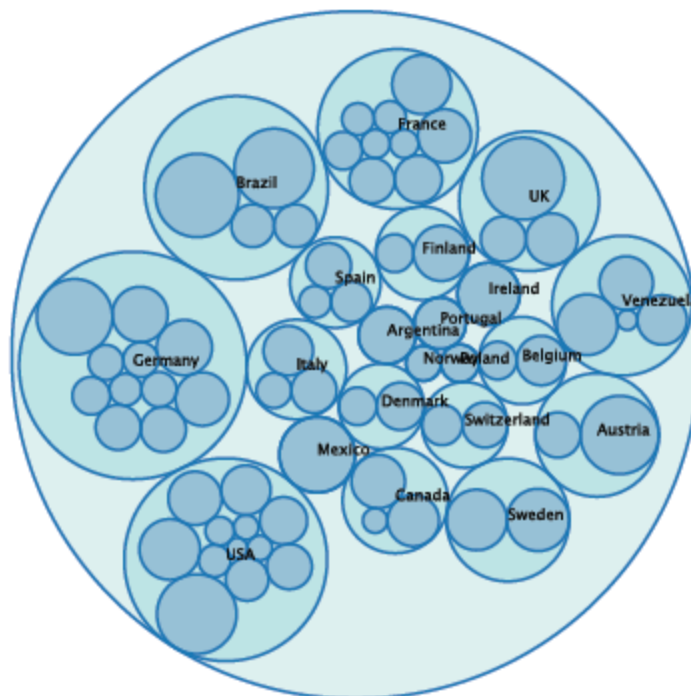
The custom visualization component lets you leverage JavaScript in Jaspersoft Studio and JasperReports Server. You can create JavaScript modules and use them in your reports to extend their functionality. The main purpose of the custom visualization component is to render SVG (Scalable Vector Graphics) 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 61.



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.

Zoomable Circle Packing

Implementation based on work by Jeff Heer. D3.js script based on Mike Bostock's Circle Packing



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 and combined 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 tab, and create a report-level property of type `com.tibco.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).

You can also create a custom component descriptor in JSON, which lets you add the following to your component:

- A component UI – You can specify the property names and types and the data items used by the component.
- A thumbnail image – Used when the component is presented in the component choose, which appears when a component is dragged into the design view.
- Location of implementation files – You can specify the location of the JavaScript file and CSS file which implement the component.

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

- D3.js
- Raphaël
- Highcharts
- JQuery

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

CHAPTER 5 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:

- **Understanding Fields**
- **Registration of Fields from a SQL Query**
- **Registration of JavaBean Fields**
- **Fields and Text Fields**

5.1 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.

Select the **Object** tab to name your field, enter a description, and choose a class.

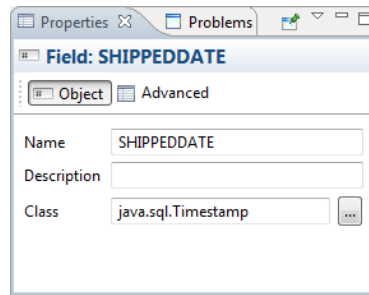


Figure 5-1 Object Tab in Properties View of a Field

Select the **Advanced** tab to enter advanced properties for the field.

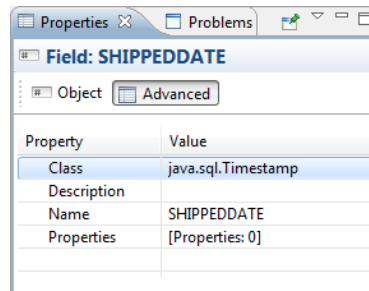


Figure 5-2 Advanced Tab in Properties View of a Field

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 text field) Jaspersoft Studio specifies a field object, using the following syntax:

```
#{<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:

```
(#{myField} != null) ? #{myField}.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:

```
`${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.

5.2 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 5-3) right-click the name of your report in the **Outline** view and choose **Dataset and Query...**

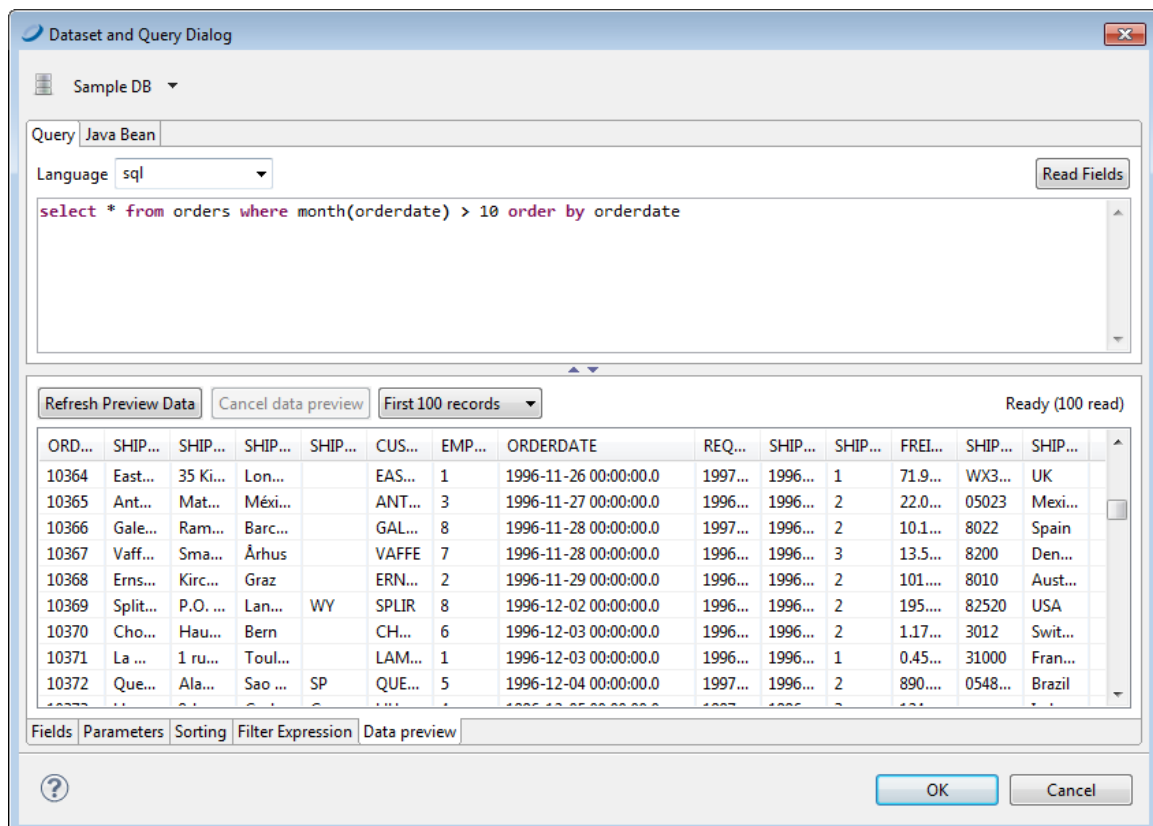


Figure 5-3 Query Dialog

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.

5.3 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.

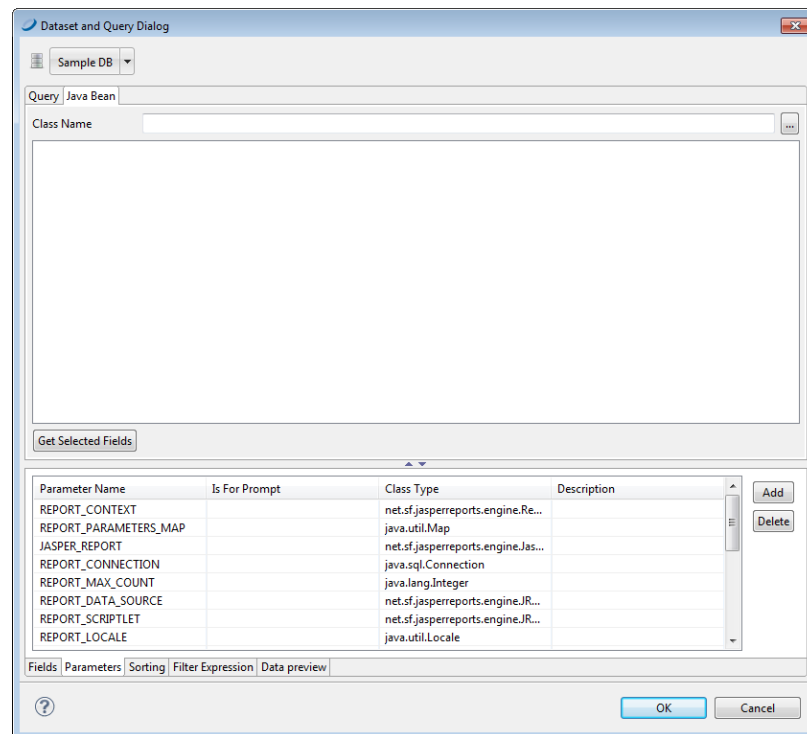


Figure 5-4 JavaBeans Tab

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.

5.4 Fields and Text Fields

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

5.5 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 to create report output in Adobe Acrobat format that provides functionality in accordance with the Americans with Disabilities 508 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 PDF 508 Tags and for the JSON exporter.

5.5.1 Configuring a Report's Metadata for PDF 508 Tags

To add 508 functionality to a report, you must add tags to the report elements. Jaspersoft Studio has menu items that let you explicitly insert tags for headings, tables, and table-like elements. For more information about tags for 508 functionality in JasperReports Library, see the *JasperReports Library Ultimate Guide*.

5.5.1.1 Tagging Headings

You can tag text fields or static text elements as headings. You can include a range of static text elements and/or text fields in the same heading.

To tag a single element as a heading:

1. Right-click the text field or static text and select **PDF 508 Tags > Heading > Heading n > Full** from the context menu.

The setting is displayed in the upper left-hand corner of the element in design view. It is underlined to show the element is the full heading.

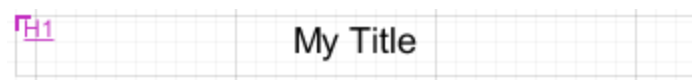


Figure 5-5 A static text element tagged as Full

To tag multiple elements as a heading:

1. Right-click the first text field or static text element in your heading section and select **PDF 508 Tags > Heading > Heading n > Start** from the context menu.
2. Right-click the last text field or static text element in your heading section and select **PDF 508 Tags > Heading > Heading n > End** from the context menu.

In design view, the start of a multi-element heading is shown in the upper left-hand corner of the Start element, and the end is shown in the lower right-hand corner of the End element.

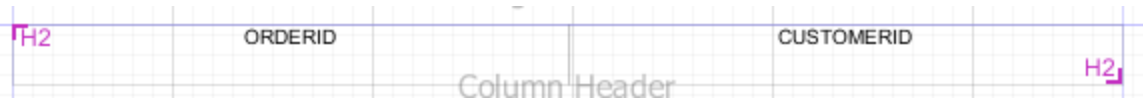


Figure 5-6 Text fields tagged as start and end

To remove a heading tag from an element:

1. Right-click the text field or static text element in your heading section and select **PDF 508 Tags > Heading > Heading n > None** from the context menu.

5.5.1.2 Using Automatic Table Tagging

In version 6.2, we added the `net.sf.jasperreports.components.table.generate.pdf.tags` property. When this property is enabled, tags are automatically generated for the tables in your report. This property can be set at the global, report, or table level.

To set this property globally:

1. Select **Window > Preferences** to open the Preferences dialog box.
2. Navigate to Jaspersoft Studio > Properties.
3. Click **Add** to open the Properties dialog.
4. Enter the following values:
 - **Property Name** – `net.sf.jasperreports.components.table.generate.pdf.tags`
 - **Value** – Enter `true` to enable table tagging or `false` to disable table tagging.



Setting the property globally inserts tags when you export a report to PDF directly from Jaspersoft Studio. If you are publishing your reports to another environment, such as JasperReports Server, you must enable this property in the `jasperreports.properties` file in your environment. See the *JasperReports Server Administrator Guide* for more information about enabling this property for JasperReports Server.

To set this property for a report or table:

1. For a table, right-click in the table. For a report, right-click on the root node in outline view.
2. Select **PDF 508 Tags > Autotag Table** from the context menu.
3. Select one of the following options:
 - **Default** – Inherits the property settings from a higher level. If the property has been set explicitly at a higher level, the current setting is shown on the menu, for example **Default (Enabled)**.
 - **Enabled** – Enables the property for this table or report. This setting overrides any value set at a higher level.
 - **Disabled** – Disables the property for this table or report. This setting overrides any value set at a higher level.

When the `net.sf.jasperreports.components.table.generate.pdf.tags` is set at the table level, the setting is displayed in the upper left-hand corner of the table in design view.

PDF ON					
ORDERID	CUSTOMERID	EMPLOYEEID	ORDERDATE	REQUIREDDATE	SHIPPEDDATE
`\${ORDERID}`	`\${CUSTOMERID}`	`\${EMPLOYEEID}`	`\${ORDERDATE}`	`\${REQUIREDDATE}`	`\${SHIPPEDDATE}`

Figure 5-7 Table with tagging enabled

5.5.1.3 Manually Tagging Tables and Lists

Automatic table tagging only works with table elements. If you have a table-like element in your report, such as a list or a tabular arrangement of fields, it cannot be tagged automatically. However, you can manually insert list or table tags using the context menu. Like tables, manual tagging only works with text fields and static text. You manually tag lists and tables using a CSS-type structure. The general steps necessary to tag tables are shown.

To manually tag a tabular arrangement of elements as a table:


1. Tag the first element in your table: **PDF 508 Tags > Table > Start.**
2. Tag the start and end of each row:
 - a. Tag the first element in your row: **PDF 508 Tags > Table Row > Start.**
 - b. Tag the last element in your row: **PDF 508 Tags > Table Row > End.**
3. To make a row a header row, add header tags to the start and end:
 - a. Tag the first element in each header row: **PDF 508 Tags > Table Header > Start.**
 - b. Tag the last element in each header row: **PDF 508 Tags > Table Header > End.**
4. Tag each detail element in each row: **PDF 508 Tags > Table Details > Full.**
5. Tag the final element in your table: **PDF 508 Tags > Table > End.**

To manually tag elements as a list:

1. Tag the first element in your list: **PDF 508 Tags > List > Start.**
2. Tag each list item: **PDF 508 Tags > List Item > Full.**
3. Tag the final element in your list: **PDF 508 Tags > List > End.**

5.5.1.4 Setting Export Parameters

Once you have inserted your 508C tags correctly, you must set the PDF export parameters in your report.

1. Click on the report preview.
2. The first time the report runs, it will not have tags. To speed up the initial run, select One Empty Record.
3. Select PDF from the format menu and wait for the report to run.
4. If necessary, click ▶ to open the panel on the left of the preview window.
5. Click  to view the exporter parameters.

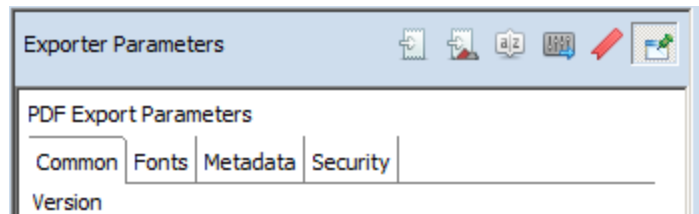


Figure 5-8 PDF Export Parameters tab in report preview

6. Select **Is Tagged**.
7. Enter a language code in the **Tag Language** field.

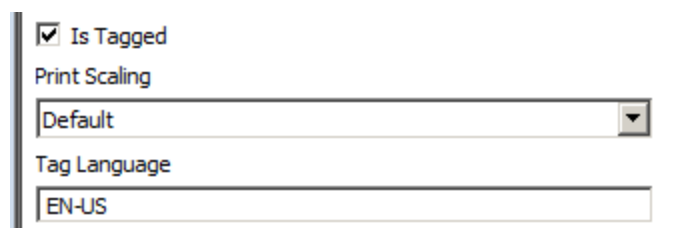


Figure 5-9 508C tags in PDF Export Parameters tab

8. Select the correct data adapter and run your report.

5.5.2 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 field 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
<code>\$F{salesamount}</code>	<code>store.sale.amount</code>
<code>\$F{salesyear}</code>	<code>store.sale.year</code>
<code>\$F{cust.name}</code>	<code>store.cust.name</code>

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 readable, 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 5-10 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 6 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 time (like the value to use in an SQL query), or to supply additional data 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:

- **Managing Parameters**
- **Default Parameters**
- **Using Parameters in Queries**
- **Parameters Prompt**

6.1 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.

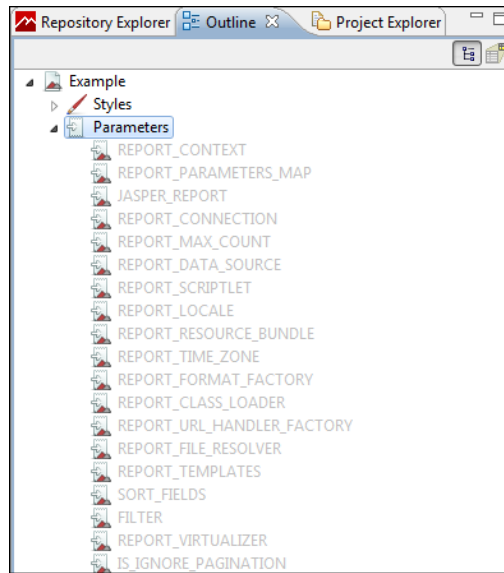


Figure 6-1 Parameters in Outline View

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.

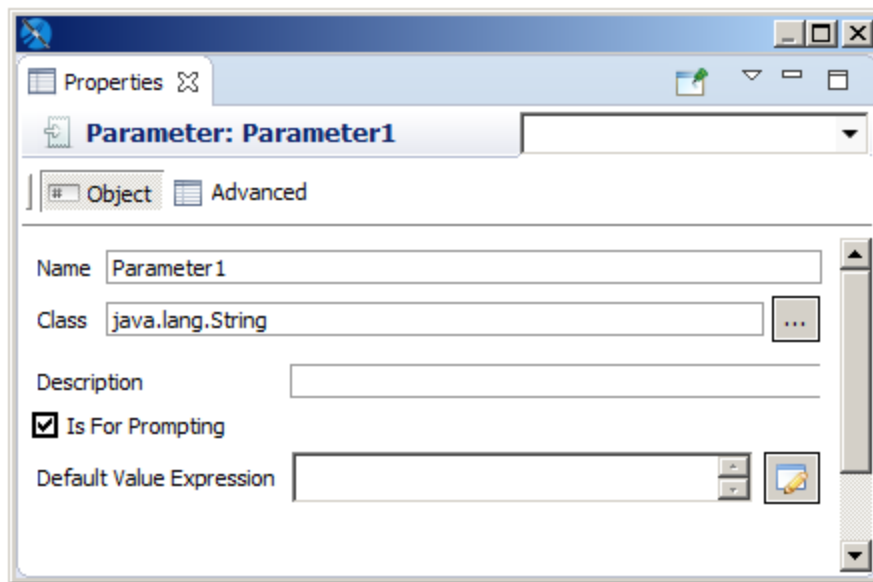


Figure 6-2 Parameters - Properties

Parameters have the following properties on the Object tab:

- **Name** – Name of the parameter.
- **Class** – Class type of the parameter.

- **Default Value Expression** – 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.
- **Description** – A string describing the parameter. The description is not used directly by JasperReports, but may be passed to an external application.
- **Is for Prompting** – Enable this to have JasperSoft Studio prompt for the parameter when the report is executed. May also be passed to an external application.

On the Advanced tab, you can use the **Properties** field to 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, you can use properties to include the description of the parameter in different languages or to add instructions about the format of the input prompt.

6.2 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 6-1 JasperReports Default Parameters

Parameter	Description
<code>REPORT_CONTEXT</code>	
<code>REPORT_PARAMETERS_MAP</code>	This is the <code>java.util.Map</code> passed to the <code>fillReport</code> method; it contains the parameter values defined by the user.
<code>JASPER_REPORT</code>	
<code>REPORT_CONNECTION</code>	This is the JDBC connection passed to the report when the report is created through an SQL query.
<code>REPORT_MAX_COUNT</code>	This is limits the number of records filling a report. If no value is provided, no limit is set.

Parameter	Description
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 <code>net.sf.jasperreports.engine.JRDefaultScriptlet</code> .
REPORT_LOCALE	This 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.
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 <code>net.sf.jasperreports.engine.util.FormatFactory</code> . 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 <code>format factory class</code> .
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 <code>net.sf.jasperreports.engine.util.FileResolver</code> 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 (<code>JRTemplate</code>) 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 <code>JRVirtualizer</code> 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.

6.3 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.

6.3.1 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.

6.3.2 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 ${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 `${EQUAL/NOTEQUAL, column, parameter}`.

6.3.3 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 ${IN, SHIPCOUNTRY, myCountries}
```

where `myCountries` is the name of the parameter that contains the set of country names. The `${}` 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 `${}` clause is evaluated as the always true statement "`0 = 0`".

6.3.4 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.

6.3.4.1 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.

6.3.4.2 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">
```

6.3.4.3 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 ${LESS, order_date, EndDate}]]>
</queryString>

```

The following table shows two additional examples of relative dates.

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

6.3.4.4 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 `${}` 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 `${}` 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
    ${BETWEEN, HIRE_DATE, StartDate, EndDate} limit 20]]>
</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>
```

6.3.4.5 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.

6.3.5 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 6-3**).

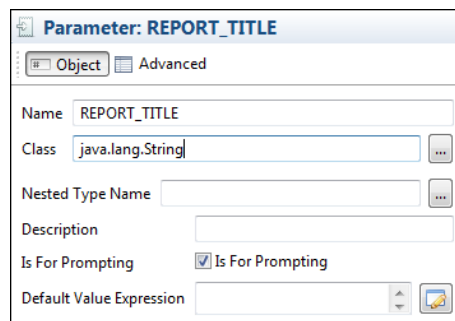


Figure 6-3 Definition of `REPORT_TITLE`

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.

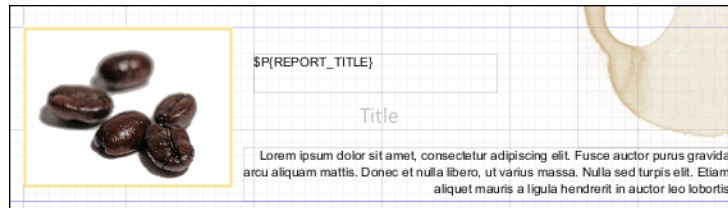


Figure 6-4 Design Panel with `REPORT_TITLE` in the Title Band

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.

6.4 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 view) 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.

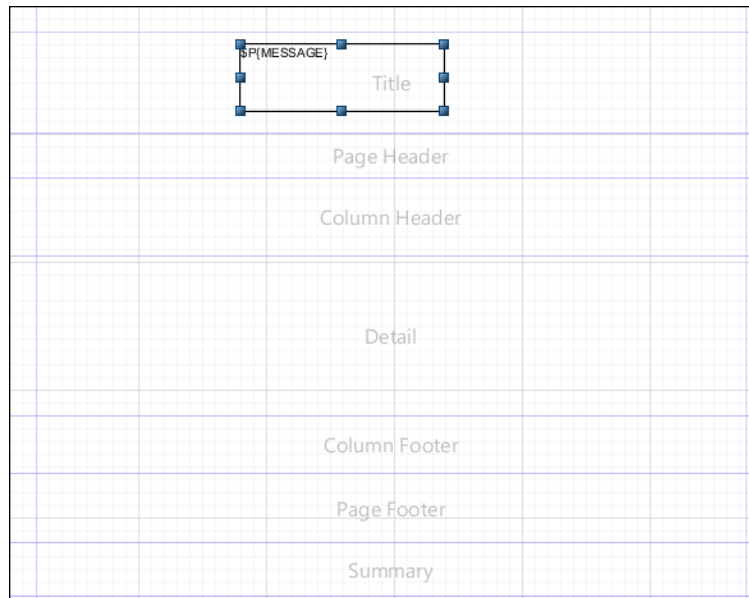


Figure 6-5 Parameter in Title Band

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.

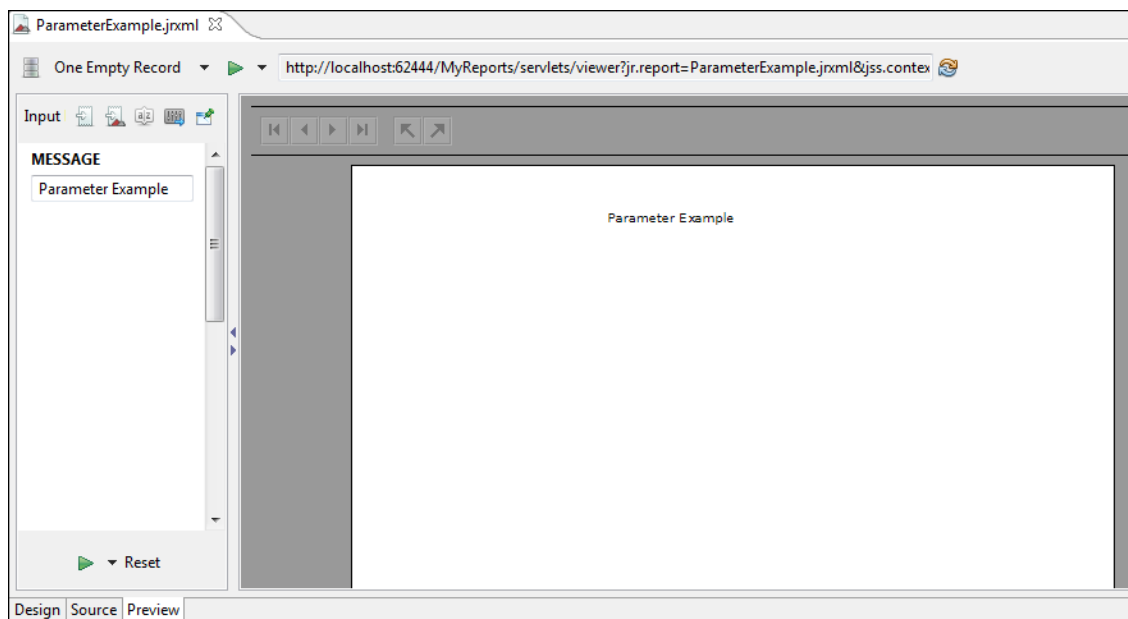


Figure 6-6 Preview Tab with Parameter Value

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

CHAPTER 7 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:

- **Defining or Editing a Variable**
- **Base Properties of a Variable**
- **Other Properties of a Variable**
- **Built-In Variables**
- **Tips & Tricks**

7.1 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 87** 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 90**

7.2 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 evaluation 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.

7.3 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.

7.3.1 Evaluation Time

Evaluation time is not an attribute of the variable but of 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 at 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.

7.3.2 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.

7.3.3 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.

7.3.4 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.

7.3.5 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 incrementation, so there is everything needed to do the calculation and return the right value.

7.4 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.

7.5 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 8 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).

8.1 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:

<code>java.lang.Boolean</code>	Defines an Object that represents a Boolean value such as true and false
<code>java.lang.Byte</code>	Defines an Object that represents a byte
<code>java.lang.Short</code>	Defines an Object that represents an short integer
<code>java.lang.Integer</code>	Defines an Object that represents integer numbers
<code>java.lang.Long</code>	Defines an Object that represents long integer numbers
<code>java.lang.Float</code>	Defines an Object that represents floating point numbers
<code>java.lang.Double</code>	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).

8.2 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 8-1 Expression operators

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



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 8-2, “Syntax for referring to report objects,” on page 95** to refer to the parameters, variables, and fields defined in the report.

Table 8-2 Syntax for referring to report objects

Syntax	Description
<code>\$F{name_field}</code>	Specifies the <code>name_field</code> field ("F" means field).
<code>\$V{name_variable}</code>	Specifies the <code>name_variable</code> variable.
<code>\$P{name_parameter}</code>	Specifies the <code>name_parameter</code> parameter.
<code>\$P!{name_parameter}</code>	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.
<code>\$R{resource_key}</code>	Special syntax for localization of strings.
<code>\$X{functionName, col_name, param1, [param2]}</code>	<p>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:</p> <ul style="list-style-type: none"> • Functions expecting three arguments for <code>\$X{}</code> – <code>EQUAL</code>, <code>NOTEQUAL</code>, <code>LESS</code>, <code>LESS]</code> (less than or equal to), <code>GREATER</code>, <code>[GREATER</code> (greater than or equal to), <code>IN</code>, <code>NOTIN</code>. For example: <pre>\$X{EQUAL, order_date, date_parameter}</pre> • Functions expecting four arguments for <code>\$X{}</code> – <code>BETWEEN</code> (excludes both endpoints), <code>BETWEEN]</code> (includes right endpoint), <code>[BETWEEN</code> (includes left endpoint), <code>[BETWEEN]</code> (includes both endpoints). For example: <pre>\$X{BETWEEN, order_date, start_date_param, end_date_param}</pre>

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
<code>"test".length()</code>	4
<code>"test".substring(0, 3)</code>	"tes"
<code>"test".startsWith("A")</code>	false
<code>"test".substring(1, 2).startsWith("e")</code>	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.

8.3 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}}.startsWith("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).

8.4 Using Unicode Characters in Expressions

You can use Unicode syntax to write non-Latin-based characters (such as Greek, Cyrillic, and Asian characters). For these characters, specify the Unicode code in the expression that identifies the field text. For example, to print the Euro symbol, use the Unicode `\u20ac` character escape. The expression `\u20ac` is not simple text; it is a Java expression that identifies a string containing the € character.



If you use this character in a static text element, “`\u20ac`” will appear. The value of a static field is not interpreted as a Java expression.

8.5 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)`
- `((${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 “instantiated” 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+3` is not legal as-is but must be fixed with something like this:

```
new Integer( 5 + 3 )
```

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`
- `((${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.

8.6 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 8-3 Groovy and Java code samples

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

The following is a correct Groovy expression:

```
new JREmptyDataSource(${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:

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

This expression returns true or false, or, again:

```
${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.

8.7 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.

CHAPTER 9 FONTS

The best way to define and use a font in JasperReports Library is to create and use a font extension. Font extensions force JasperReports to work with external TTF, SVG, WOFF, or EOT fonts instead of using built-in or system fonts. This ensures that a specific font behaves in the same way wherever the report is executed.

Using system fonts usually results in unacceptable changes in report format when the report is deployed on another system. Subtle differences in font size and spacing can affect not only the appearance of the text but the layout of the report itself. You may lose part of the text in a text element or the font might not be available at all. Font extensions help avoid these problems:

- A font can be available in one operating system but not in another. In this case, the default font is used for the element, but it may not support the expected character set.
- The Java virtual machine can map logical font family names to different physical fonts.
- A font that is available in different operating systems can be slightly different from one operating system to another.

You can incorporate additional information in a font extension, such as: bold and italic fonts, the text encoding for the font, and a list of locales where the font should be used. You can also use font extensions to embed your fonts in PDF files.

In addition, you can combine several font extensions into a font set. For example, if you have data in English and Japanese, you can create a single font set that combines fonts for those languages.

9.1 Font Extensions Reference

You work with font extensions using the Fonts page in the Preferences dialog.

9.1.1 The Fonts Page

The Fonts page displays all your font extensions and font sets, and lets you create, copy, edit, delete, and reorder fonts and font sets.

To access the Fonts page:

1. Select **Window > Preferences**. The Preferences dialog is displayed.
2. In the Preferences dialog, select **Jaspersoft Studio > Fonts**.

The Fonts page is displayed.

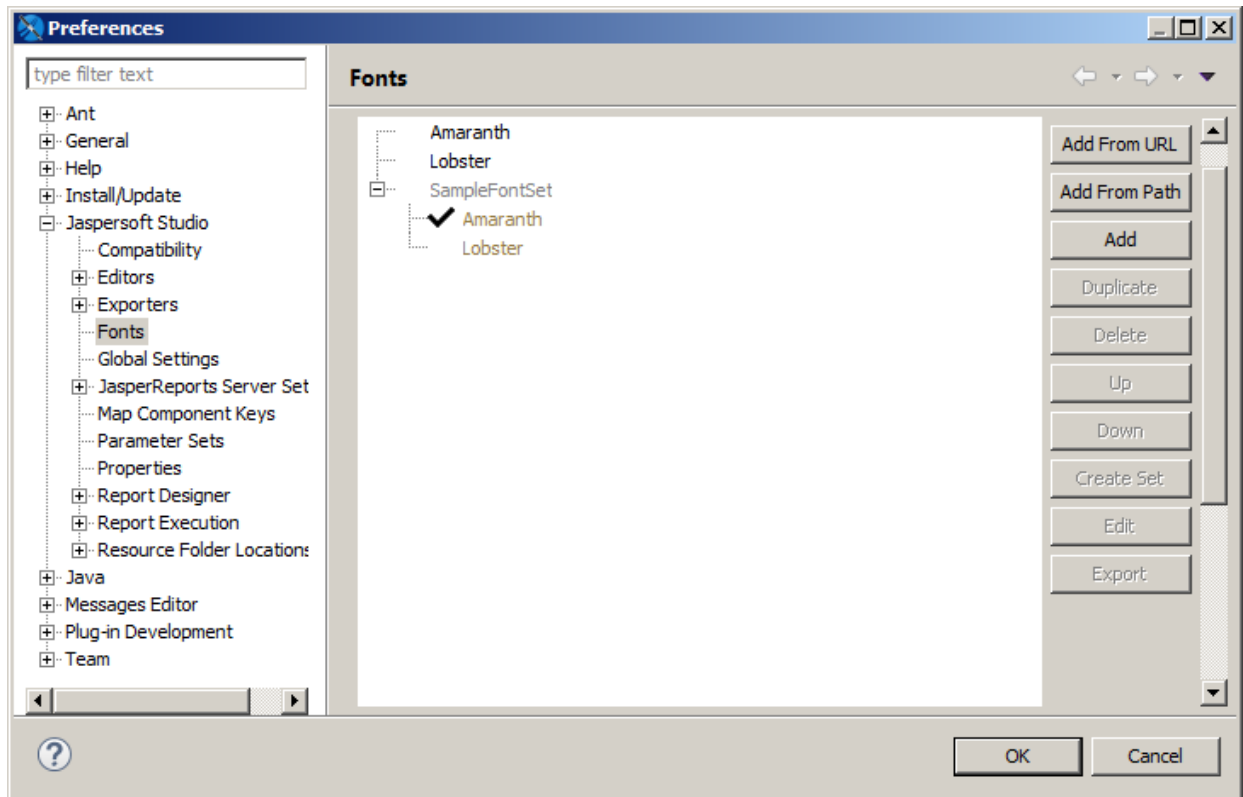


Figure 9-1 Fonts page

The Fonts page shows the following:

- **Font List** – The list of font extensions and font sets. Font sets are prepended with **+**; click to expand and show the font extensions contained in the font set. When you create a font set, the default font for the set is the font that appears highest in the list when the font set is created. Inside a font set, non-default fonts are applied in the order they appear in the font list.
- **Add from URL** – Specify a URL location for a zip file of the fonts you want to add as extensions. To add all the Noto fonts from Google, click **▼** and select the Noto URL.
- **Add from Path** – Specify a folder containing the fonts you want to add as extensions.
- **Duplicate** – Create a copy of the selected font extension(s) and/or font set(s). Each copy has a unique name, which can be edited.
- **Delete** – Delete the selected font extension(s) and/or font set(s). This does not remove the original font files from your system.
- **Up** – Move the selected font extension and/or font set up in the list of fonts.
- **Down** – Move the selected font extension and/or font set down in the list of fonts.
- **Create Set** – Combine the selected font extensions into a font set.
- **Edit** – Edit the selected font extension or set. For a font extension, displays **The Font Family Dialog**. For a font set, displays the Font Set dialog.
- **Export** – Export the selected font extension or set.

9.1.2 The Font Family Dialog

The Font Family dialog lets you configure an existing font extension and create and configure font sets. This dialog has three pages.

To access the Font Family dialog:

Select a font extension in the Fonts list and click **Edit**.

9.1.2.1 Font Family Page

The Font Family page lets you define the basic configuration of the font or font set.

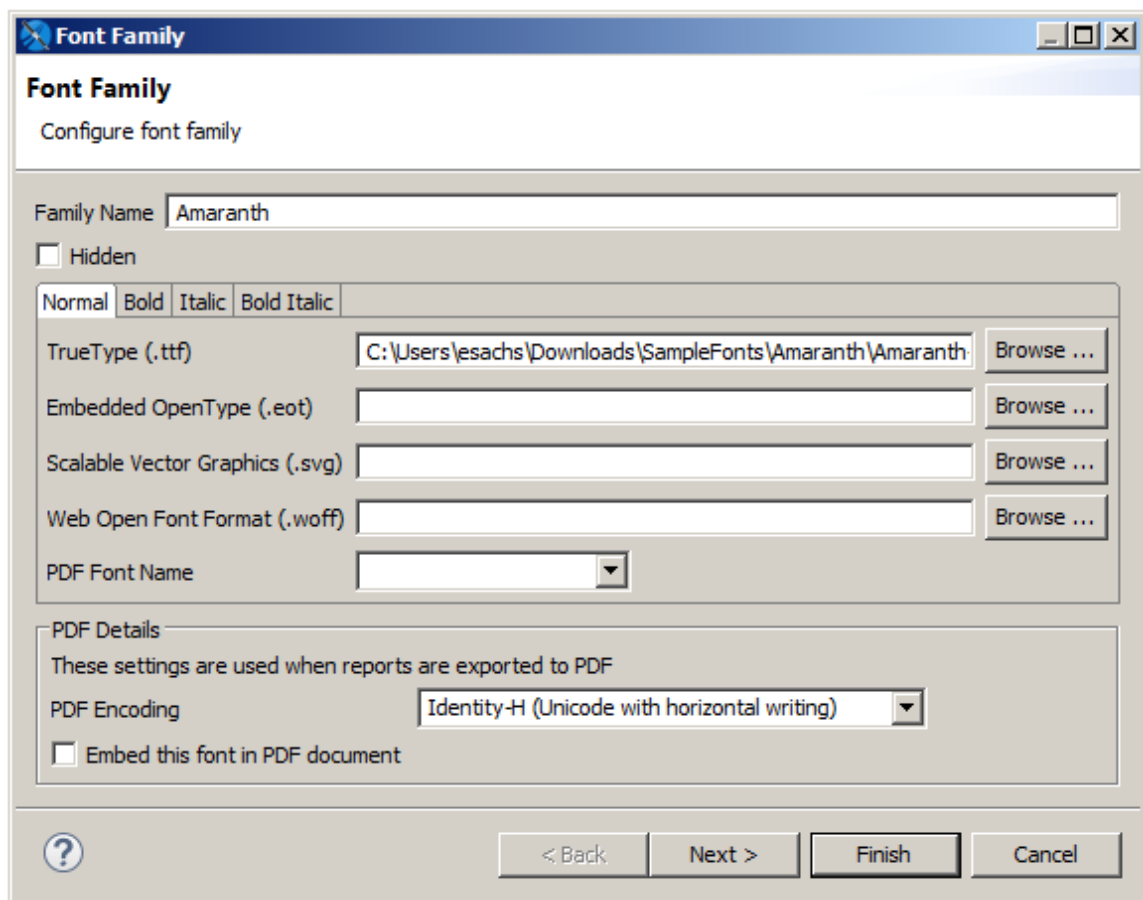


Figure 9-2 Font Family dialog - Font Family

The Font Family page shows the following:

- **Family Name** – Name JasperReports Library uses to identify the font extension or font set. When you create a font extension from an external font file, by default, the font name is used for the family name. This can be edited.
- **Hidden** – Flag that determines whether the font is shown as an available font extension "above the line" in the **Font** property of a report element. Use the Hidden flag to hide internal fonts from the user. See [9.2.2, “Using Font Extensions in a Report,” on page 112](#) for more information.

- **Normal/Bold/Italic/Bold Italic** – Tabs that let you configure the individual files that define the specified attributes. For each attribute, you can configure the following:
 - **[Font Format]** – File location for the specific font and attribute. To change or add a file, use the **Browse** button to navigate to the file location for the specific font and attribute. You can only select one file for each tab.
 - **PDF Font Name** (deprecated) – The name of the font when exported to PDF. This can be a pre-defined PDF font or the name of the font file. Not necessary when using font extensions.
- **PDF Details** – Settings used for the font when exported to PDF. Deprecated for static text and textfields.
 - **PDF Encoding** (deprecated) – The font encoding to use for the font. Defaults to Identity-H. To avoid Identity-H printing issues, set this to the correct encoding for your font.
 - **Embed this font in PDF document** (deprecated) – Flag that specifies whether to embed the font in a generated PDF or not. Embedding fonts is recommended to ensure consistency across platforms.

9.1.2.2 Font Mapping Page

The Font Mapping page lets you specify a font mapping to use when one or more font files are not installed in the target environment. The order of the font names indicates the order in which the web browser should search for the replacement font. Once a replacement font is found, the browser stops searching and renders the text. If no font mapping is available for HTML, then web fonts are used.

To access the Font Mapping Page:

Click **Next** on the Font Family page.

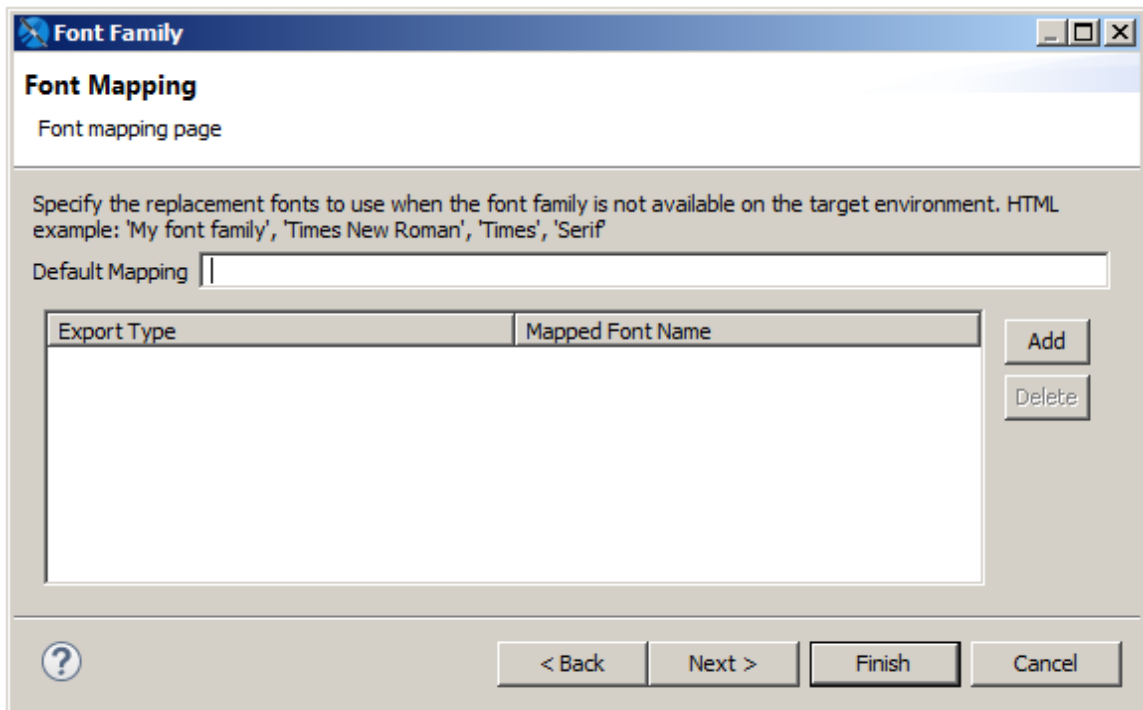


Figure 9-3 Font Family dialog - Font Mapping



If a mapping is present, then web fonts are not used. Do not define a font mapping unless you want to override the web font functionality.

The Font Mapping page shows the following:

- **Default Mapping** – A string that defines a mapping for the font. For example, a serif font might be set to: 'My font family', 'Times New Roman', 'Times', 'Serif'.
- **Add** button – Adds the string in the Default Mapping field to the list of mappings.
- **Delete** button – Deletes the currently highlighted mapping.
- **Export Type** – Sets the file type (html, xhtml, or rtf). Click on a value to display a down arrow, then click on the arrow to select a type from the cascading menu. Defaults to html. You can define different mappings for different file types.
- **Mapped Font Name** – Click on a value to edit the name of the mapped font.

9.1.2.3 Locale Mapping Page

The Locale Mapping page lets you set the locales where this font family is used. If you have different font extensions that support different languages, (for example, a font for Western European and a font for Japanese), you can set the locales for these fonts. It is preferable to use font sets.

To access the Font Locale Page:

Click **Next** on the Font Mapping page.

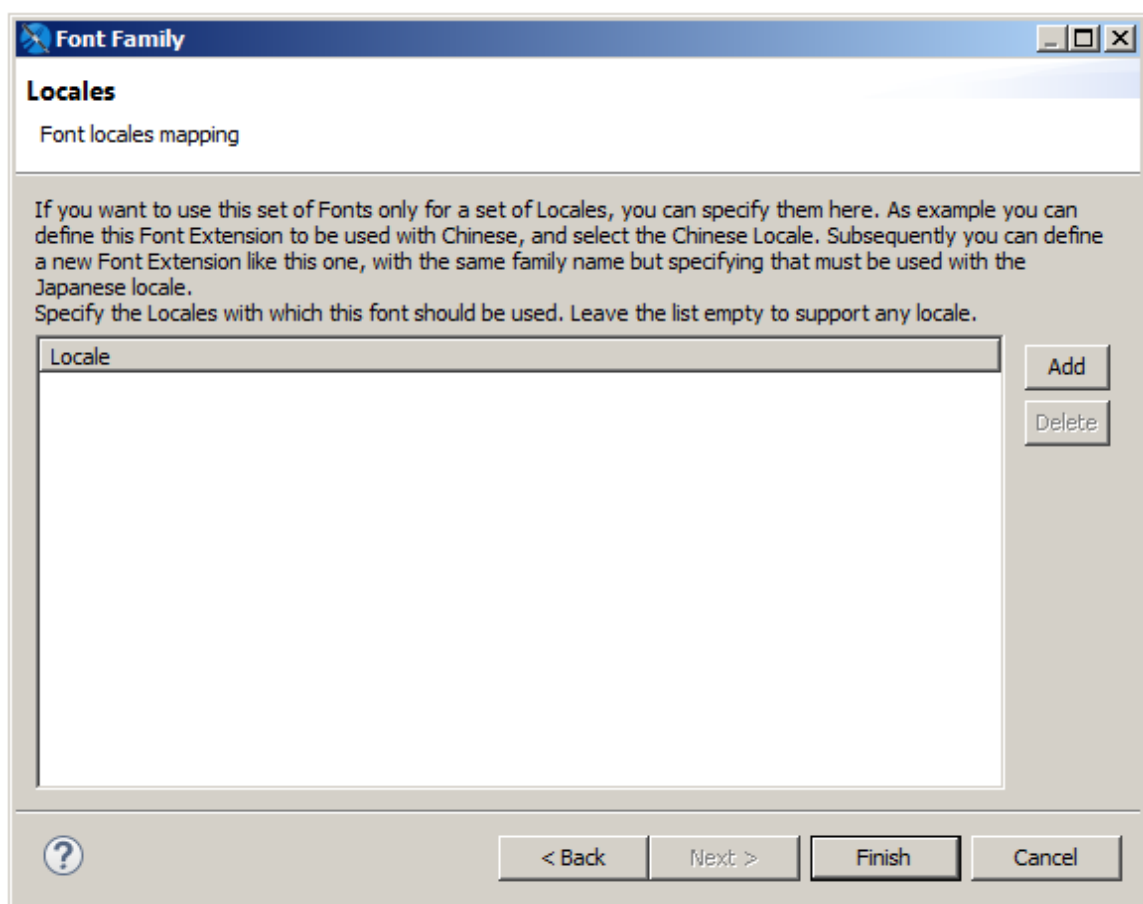


Figure 9-4 Font Family dialog - Locale

The Locales page shows the following:

- **Locale** – List of locales supported by the font.
- **Add** button – Add a locale where the font is supported.
- **Delete** button – Delete the selected locale.

9.1.3 Font Sets

Font sets let you group font extensions in supersets that can include several languages and/or scripts. When you use a font set, the font family resolution occurs during text processing on a per-character basis, allowing you to include characters from different languages or character sets in the same text element. For example, if you have data that includes Japanese and English entries, you can create a single font set that contains Western European and Japanese fonts and use that font set for your data.

9.1.3.1 The Font Set Dialog

When you create or edit a font set, the Font Set dialog lets you edit the name for the font set.

Accessing the font set dialog:

- To create a font set, select the font extensions you want to combine, and click **Create Font Set**. Enter a name for the set and click **OK**.
- To edit the name of an existing font set, select the set and click **Edit**.

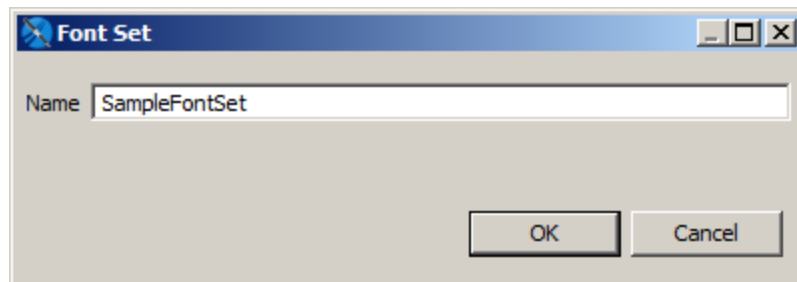


Figure 9-5 Font Set dialog

9.1.3.2 The Font Set Family Dialog

The Font Set Family dialog lets you control the mapping between fonts and characters sets for a font set. In many cases, multiple fonts inside a font family may support the same character set or "script". In this case, JasperReports Library uses a greedy algorithm to display as much contiguous text as possible in a single font. If you want to ensure that a specific font extension is used for a specific script, you can set the included/excluded scripts for the fonts in your font set.

To access the Font Set Family dialog:

Expand a font set in the Fonts list, then double-click a font inside the set.

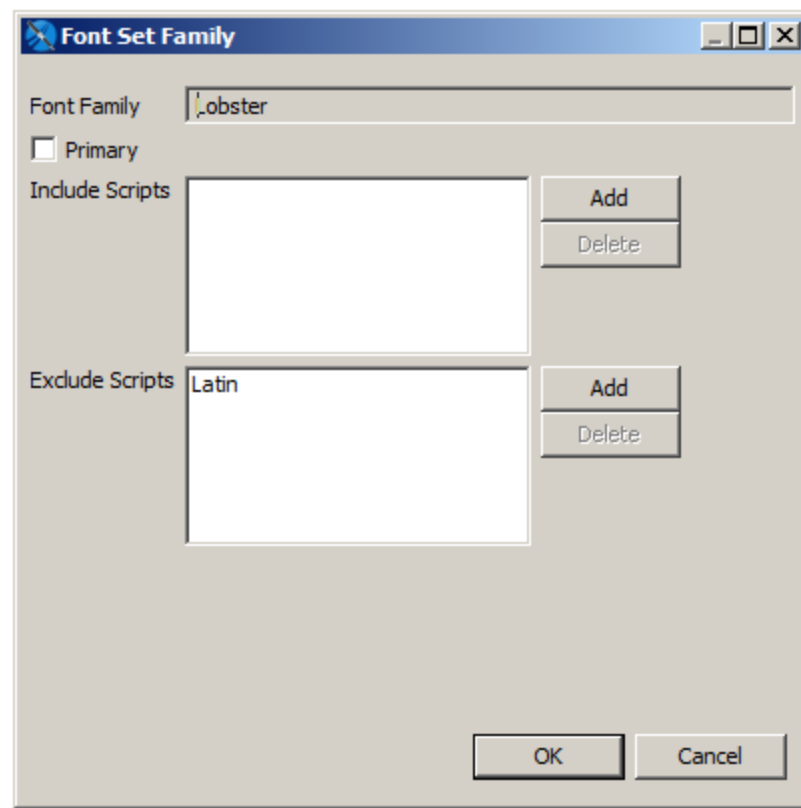


Figure 9-6 Font Set Family dialog

The Font Set Family dialog shows the following:

- **Primary** – Toggle that sets the primary font for the font set.
- **Include Scripts** – Language/character sets that should use this font extension when the font set is applied. Click **Add** to add a script; highlight a dis
- **Exclude Scripts** – Language/character sets that should never use this font extension when the font set is applied.

9.2 Example of Using Font Extensions

This example shows how to create font extensions for two font and then combine them in a font set.

9.2.1 Creating Font Extensions and Font Sets

To access the Fonts page:

1. Select **Window > Preferences**. The Preferences dialog box is displayed.
2. In the Preferences dialog box, select **Jaspersoft Studio > Fonts**.

The Fonts page is displayed:

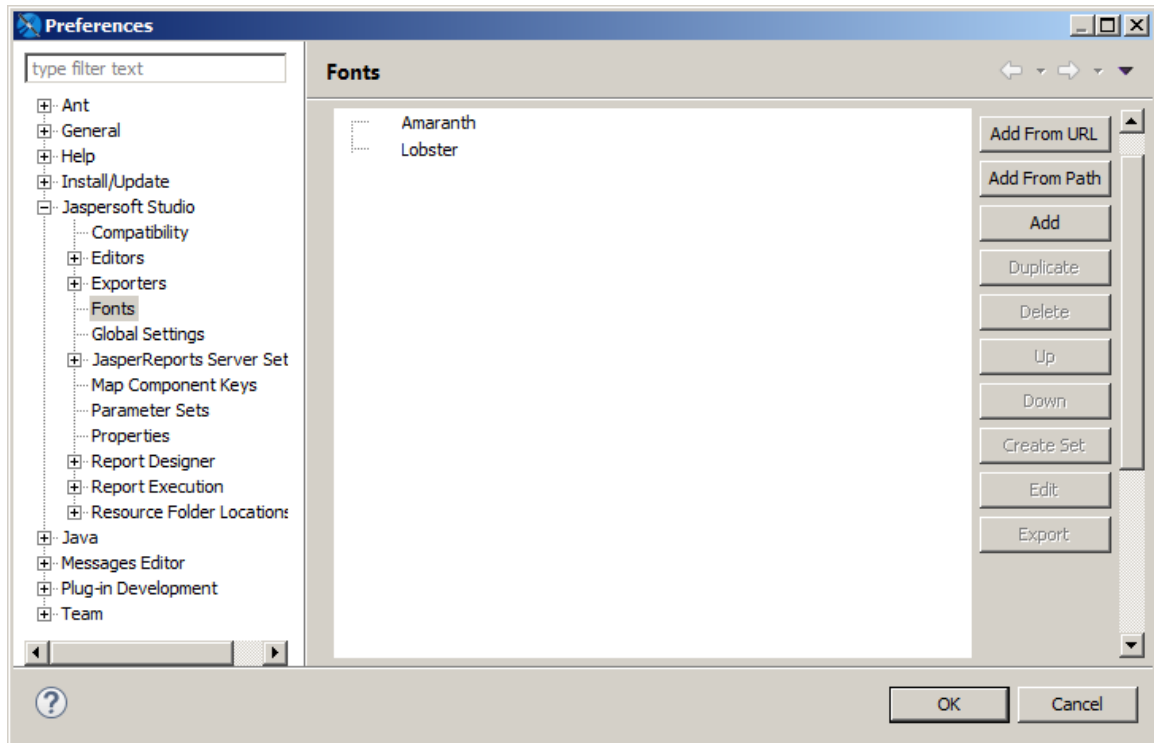


Figure 9-7 Fonts preferences

To create a font extension:

This example uses two external Google fonts, Amaranth (a Latin-only font) and Lobster (supports Latin and Cyrillic characters). If you don't have these fonts available, you can work with other fonts. However, you must have the correct license to embed your fonts in a PDF.

1. Make sure the font files for the fonts have been downloaded and decompressed. You can also use fonts from a URL.
2. Click **Add from Path** to open the Fonts path dialog.

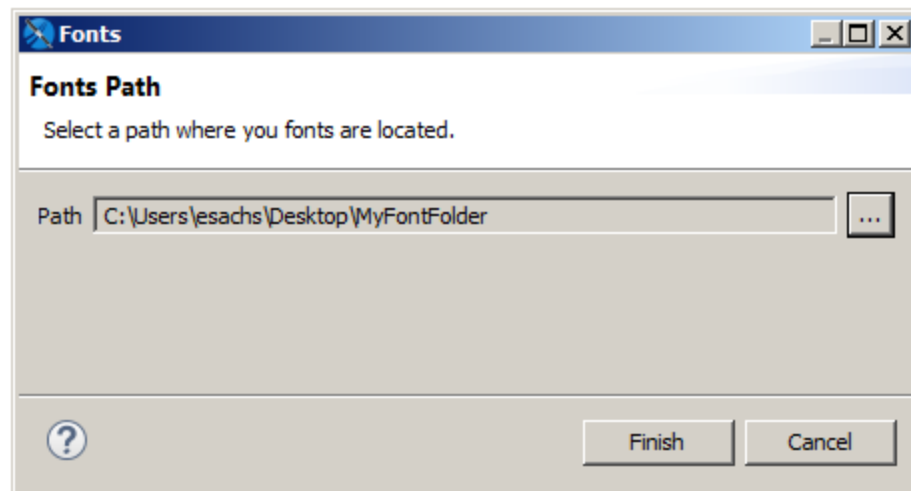


Figure 9-8 Adding fonts from a path

3. Click ... and browse to the folder that contains the fonts you want, then click **Finish**.
Jaspersoft Studio loads all the fonts at that location, extracts the font family name embedded in the font files, and displays all the extracted fonts in the Preferences dialog.



Once you have create a font extension, you can edit it by double-clicking its name in the font list or by selecting it and clicking **Edit**. See [9.1.2, “The Font Family Dialog,” on page 103](#) for more information.

Next, combine these two font extensions in a single font set.

To create a font set:

1. In the Font section of the Preferences dialog, select the fonts you want in your set.

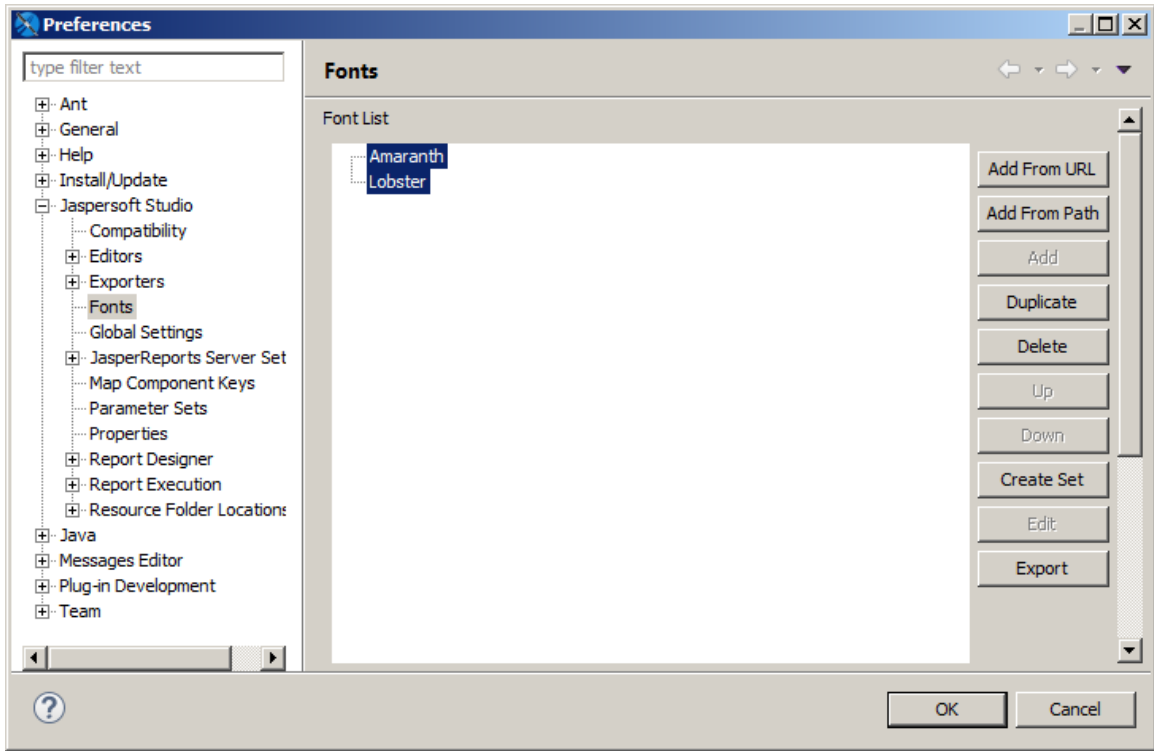


Figure 9-9 Selecting font extensions

2. Click **Create Font Set**. The Font Set dialog is displayed.

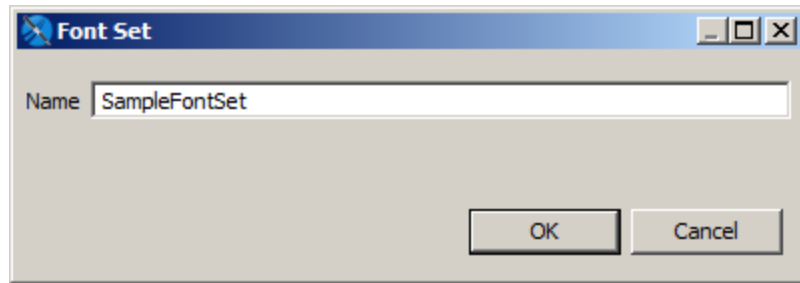


Figure 9-10 Font Set dialog

3. Enter a name for your font set and click **OK**. This example uses SampleFontSet.
The new font set is displayed in the font list.

Next, configure the fonts in the font set so that Lobster is only used for Cyrillic characters, even though it supports Latin characters.

Configure fonts in a font set:

1. Expand the font set to display the names of the individual font extensions.

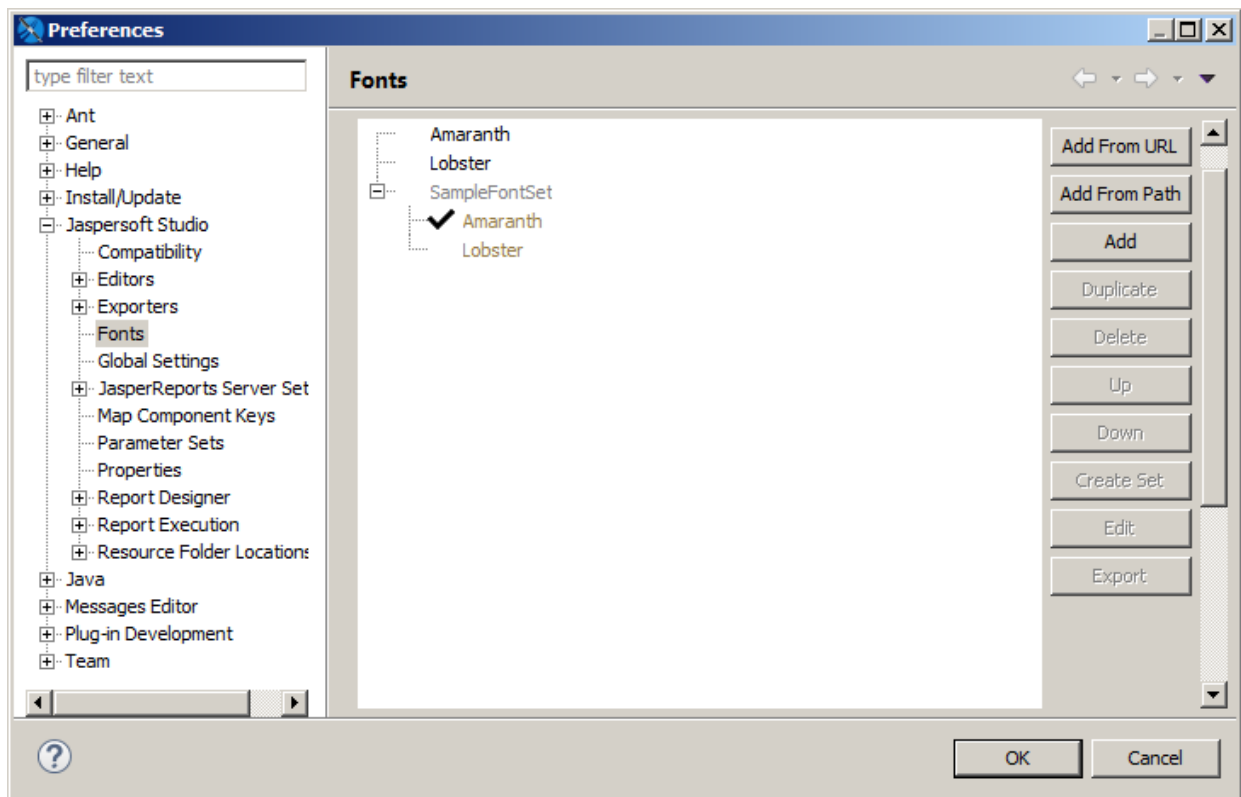


Figure 9-11 Fonts window with expanded font set

2. Select **Lobster** and click **Edit** or double-click **Lobster**.
The Font Set Family dialog is displayed.

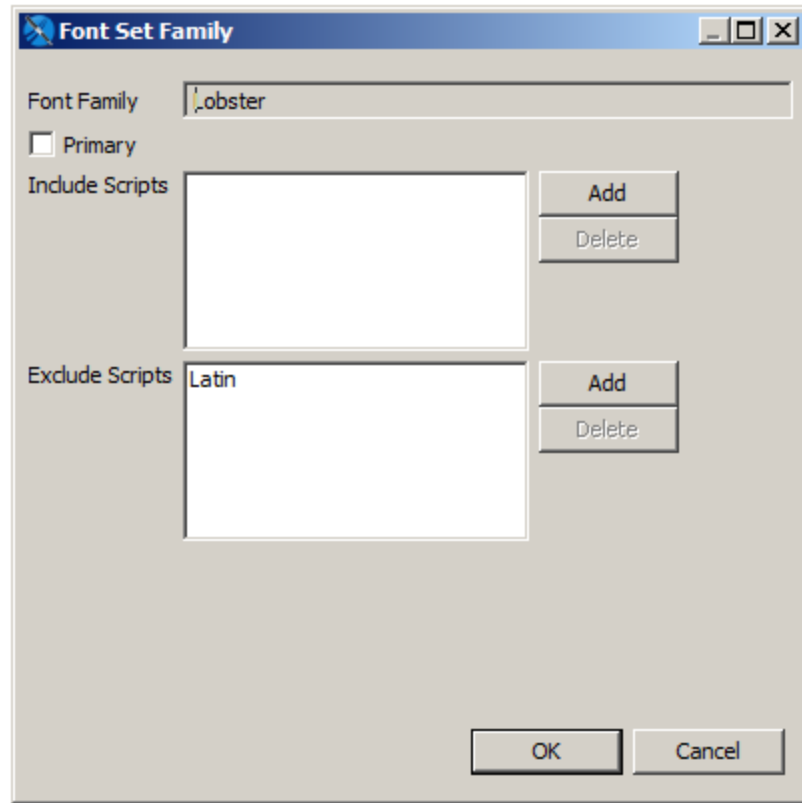


Figure 9-12 Font Set Family dialog

3. To prevent Lobster from being used by Latin characters, click **Add** next to the Exclude Scripts list. The Scripts name dialog is displayed.
4. Select **Latin** in the Scripts Name dialog and click **OK**. Latin is added to the list of excluded scripts.
5. Click **OK** to close the Scripts Name dialog; click **OK** again to close the Font Set Family dialog and click **OK** a third time to close the Preference dialog.

9.2.2 Using Font Extensions in a Report

Once you have set up your font set, you can use it in a report.

Create a report with a local data adapter:

1. Export the One Empty Record adapter to your project. To do this:
 - a. In the Repository Explorer, right-click the One Empty Record adapter and select **Export to File**.

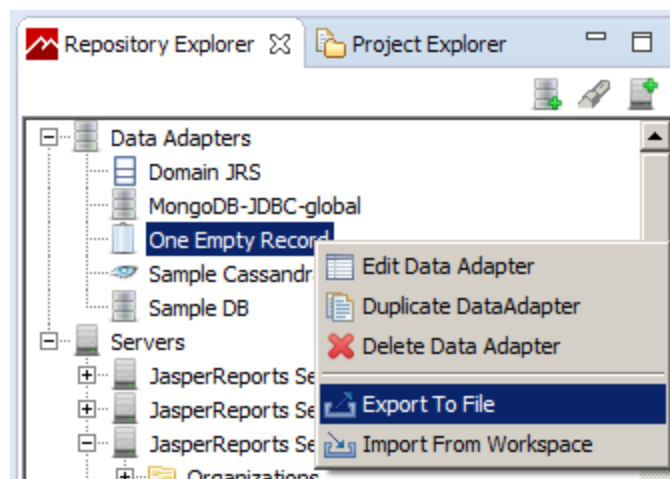



Figure 9-13 Exporting a global data adapter

- b. Select the project you want and click **OK**.
A data adapter file is created in your project.
2. Go to **File > New > Jasper Report** or click  on the main toolbar.
3. In the **New Report Wizard** window, select a blank template, such as the **Blank A4** template, then click **Next**.
4. Select the project folder with the data adapter file you just created, give the report a name, and click **Next**.
5. On the Data Source page, select the **One Empty Record - [OneEmptyRecord.xml]** adapter. Make sure to select this adapter, which is local, and not the One Empty Record adapter which is selected by default.

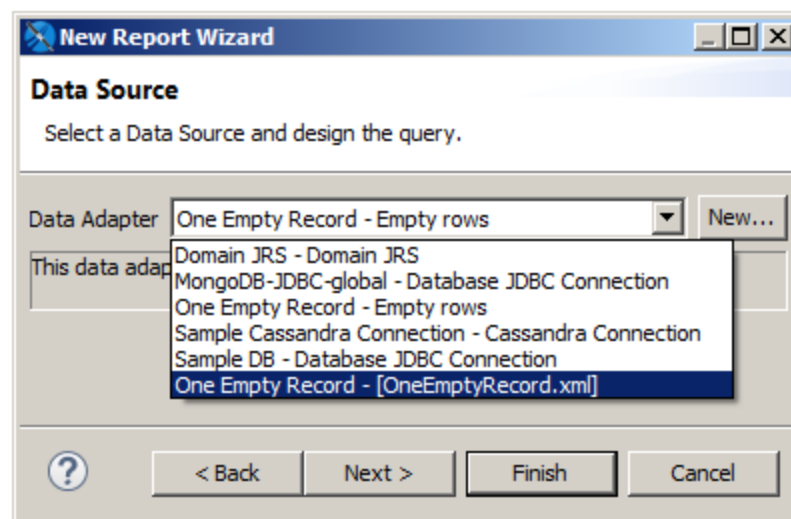


Figure 9-14 Selecting the local data adapter

6. Click Finish.

7. Set the default data adapter for the report:
 - a. Select the report node in Outline view.
 - b. In the Properties view for the report, on the Report tab, scroll down to **Dataset > Default Data Adapter** and click ...
 - c. In the Open Data Adapter dialog, select **Custom Value**.
 - d. Enter OneEmptyRecord.xml in the **Path** entry box.

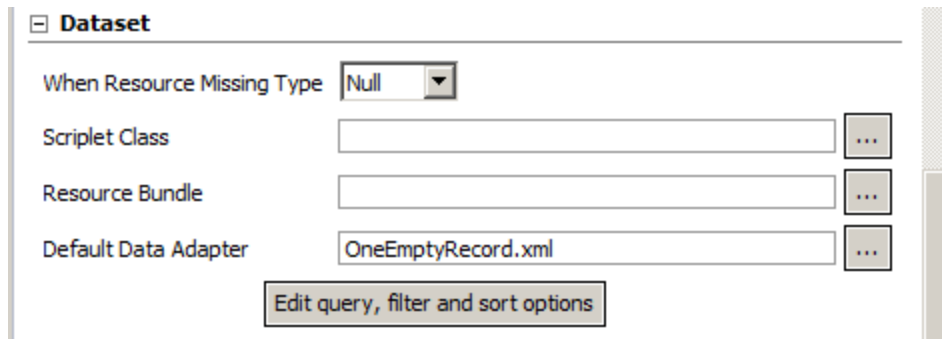



Figure 9-15 Default Data Adapter

- e. Click **Finish**.

Create a report with multi-lingual text:

1. Create a new report with a blank template
2. Drag the Static Text element  into the Title band of the report.
3. Enter English and Cyrillic text in the element you just created:
Report Отчет
4. Select the element.
5. Expand the **Font** menu on the Static Text tab of the Properties View for the static text element.
The menu is divided into two sections. Installed font extensions or font sets appear above the line. Fonts below the line are not installed as font extensions. In this example, Amaranth, Lobster, and SampleFontSet are all above the line.

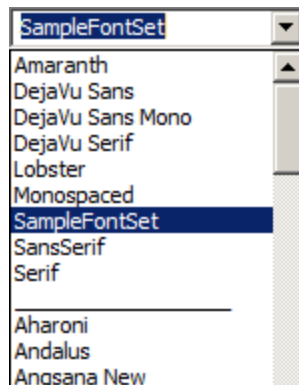


Figure 9-16 Font menu with font extensions

The default font used for a new static text element is SansSerif. This font does support for extended characters, but because it's a Java logical font that is translated to a physical font by the JVM, you won't know what font will be selected when the report is run.

6. Select **SampleFontSet** from the **Font** menu and **24** from the size menu next to it. Then resize the static text element so it is large enough to display the text.



Figure 9-17 Font set in design view

7. Save and preview the report. The license for these fonts let you preview the report as a PDF.


Report Омчëm

Figure 9-18 Font set in preview

9.3 Deploying Font Extensions to JasperReports Server

When use font extensions in a report, the font extensions are not automatically available on the server. You need to export your font extensions as a jar and upload them to the server. For reports in HTML, add the jar to the server classpath and enable font support in `jasperreports.properties`. For reports in PDF, add the jar to the report as a resource.

Deploy the report to JasperReports Server:

1. Click  on the main menu bar.
2. In the Publish To JasperReports Server dialog, select the JasperReports Server instance you want and choose a location for the report. This example uses **Public > Samples > Reports**.
3. Enter a name for the report on JasperReports Server. This example uses `SampleFontSetReport`.
4. Click **Next**.
5. Verify that `OneEmptyRecord` appears as a resource to publish on the next page, then click **Next**.
6. Verify that **Don't use any Data Source** is selected on the Configure the Data Source page. Making this selection ensures that the uploaded adapter will be used for the report.
7. Click **Finish**. If the publishing process succeeds, a success dialog is displayed.
8. Click **OK** to exit the success dialog.

View the report on JasperReports Server:

1. Log in to the server, navigate to the report you just created, and run the report.
You will see that the report does not use the correct fonts. You need to export the fonts in Jaspersoft Studio and upload them to the server.



When working with fonts, look carefully at your uploaded reports. For some fonts or character sets, you will not see misformatted text; instead, the text will not be displayed at all.

Export the font set in Jaspersoft Studio:

1. In **Window > Preferences > Jaspersoft Studio > Fonts**, select the font set and the fonts within it and click **Export**. For this example, select Amaranth, Lobster, and SampleFontSet.
2. In the Export Font to Jar dialog, select a name and location for the exported file and click **Save**. For this example, use SampleFontSet.jar.

The font set is exported as a jar in the location you chose. This is not a regular font jar; it is a jar file that includes additional information used by Jaspersoft.

Add the font set as a jar on your JasperReports Server instance:

1. On the machine hosting your JasperReports Server instance, enable font support by adding the following to your `<js-install>\WEB-INF\classes\jasperreports.properties` file:

```
net.sf.jasperreports.web.resource.pattern.fonts=fonts/*.*
```



You only have to enable font support once.

2. Add the exported font set jar to your `<js-install>WEB-INF\lib` directory.
3. Stop and restart your JasperReports Server instance. See the *JasperReports Server Installation Guide* for more information.

Upload the font set as a resource:

You can attach the resource directly to the report, or you can upload it to another location, for example the report directory and link the report to it. Uploading a resource to another location makes it easier to reuse the resource.

1. In the Repository Explorer in Jaspersoft Studio, navigate to the folder on your JasperReports Server instance where you want to add this resource. For this example, it is the **Public > Samples > Resources** folder.
2. Right-click the folder and select **New** from the cascading menu.

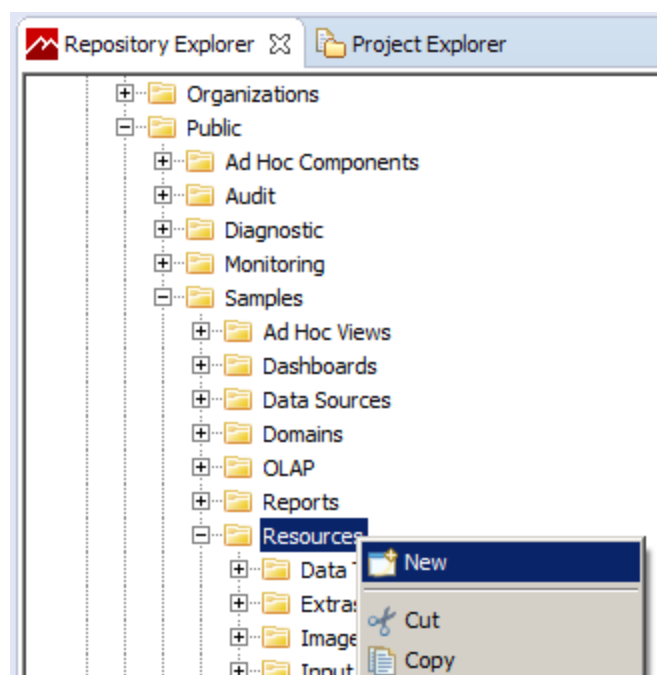


Figure 9-19 Adding a resource in the Repository Explorer

3. Select **Jar** in the Add Resource wizard and click **Next**.
4. Enter a name and ID for your jar and click **Next**.
5. Select **Upload from File System**, select the jar you want, and click **Open**.
6. Click **Finish** to upload the selected jar.

Add the resource to your report:

1. Right-click your report in the Repository Explorer and select **New** from the cascading menu.
2. Select **Link** in the Add Resource Wizard and click **Next**.
3. Enter a name and ID for the link and click **Next**.
4. Click ▼ to open the Find Resource dialog.
5. Navigate to the jar you uploaded and click **Open**.
6. Click **Finish** to attach the jar to the report as a resource.

View the report on JasperReports Server:

1. Open a web browser, log in to the server, navigate to the report you just created, and run the report. You should see the correct fonts. If you do not see your fonts, there has been a problem with uploading the jar to the file system.
2. Export the report as PDF. You should see the updated fonts. If not, there has been some problem uploading and linking the resource.



When verifying font extensions and sets, it is best to run the report in JasperReports Server from a web browser. Running the report from Repository Explorer in Jaspersoft Studio may not show the fonts correctly.

CHAPTER 10 DATA ADAPTERS

A data adapter is a resource that specifies how and where to obtain data. Specifically, it is an object that contains information about how to connect to or retrieve the data, and the logic to do that. Data adapters are stored in XML files and simplify porting of the report configuration and data source creation between JasperReports environments. Whether you use a report with a data adapter XML file in Jaspersoft Studio, publish it to JasperReports Server or deploy it to a custom JasperReports environment, JasperReports Library can use it to obtain the data you specify.

This chapter starts by telling you how to create and use data adapters based on the data adapter types available in Jaspersoft Studio. Data adapters are designed to simplify the complexities of working with data in JasperReports Library. However, data adapters are only one of the ways that JasperReports Library can get data from a data source. As you get more familiar with Jaspersoft Studio, you may want to go a little deeper and learn about data in JasperReports Library and the `JRDataSource` interface.

Usually data adapters are stored as XML files in the same project as the report to simplify deployment to JasperReports Server or another environment. In Jaspersoft Studio, data adapters can also be stored in the Repository Explorer, in which case they are visible from all the projects. If you plan to deploy the report outside Jaspersoft Studio, it is better to store it in the project from the beginning.

This chapter has the following sections:

- **[Creating and Editing Data Adapters](#)**
- **[Using Data Adapters in Reports and Datasets](#)**
- **[Working with Database JDBC Connections](#)**
- **[Working with a MongoDB Data Adapter](#)**
- **[Working with a Native Cassandra Connection](#)**
- **[Working with a Collection of JavaBeans Data Adapter](#)**
- **[Working with XML Data Adapters](#)**
- **[Working with XML/A Data Adapters](#)**
- **[Working with CSV Data Adapters](#)**
- **[Using the Empty Record Data Adapter](#)**
- **[Working with the JRDataSource Interface](#)**
- **[A Look at TIBCO Spotfire Information Links](#)**

10.1 Creating and Editing Data Adapters

10.1.1 Creating a Data Adapter


You create data adapters using the **Data Adapter Wizard**. The exact steps and information you need to provide vary with the type of adapter that you select. However, the initial steps are similar.

Data adapters can be created locally in projects or globally in the Repository Explorer.

- Data adapters in projects are stored as XML files, which simplifies deployment to JasperReports Server. A project-level data adapter cannot be seen from other projects, but you can easily copy it from one project to another.
- Global data adapters are saved as Eclipse settings and are visible to all projects. Global data adapters are saved as Eclipse settings. To deploy a global adapter to JasperReports Server, you need to export it to an XML file located in the same project as your report.

To create a data adapter in a project:

When you create a data adapter in a project, it is saved as an XML file in that project. Saving the XML file in the same project as your reports makes it easier to deploy the data adapter to JasperReports Server.

1. Click  on the main toolbar OR right-click a project in the Project Explorer and select **New > Data Adapter**.
2. In the DataAdapter File window, choose the project where you want to save the data adapter file. This should be the project that contains the report(s) you want to use with your data adapter.
3. Enter a name for your adapter and click **Next**.

The Data Adapters Wizard opens.

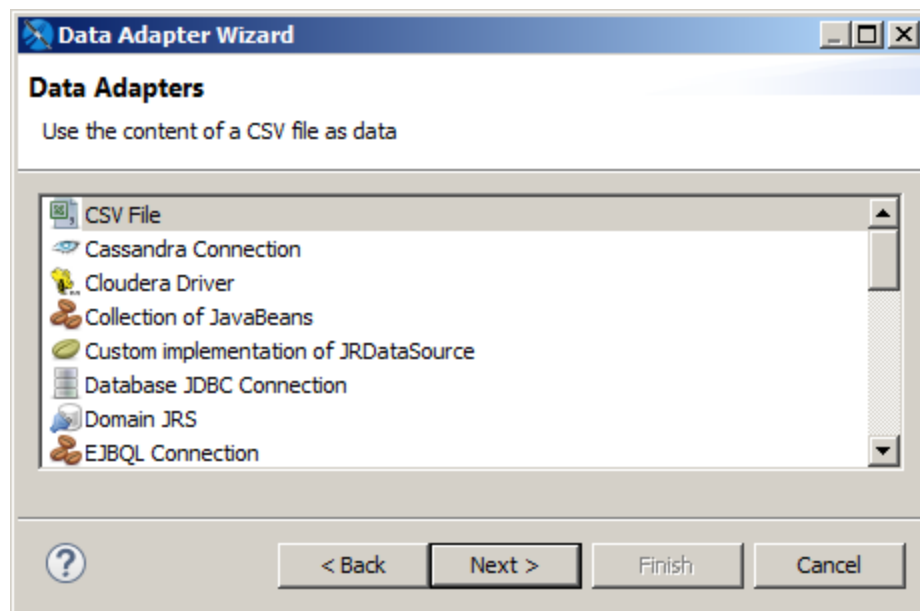



Figure 10-1 Data Adapter Wizard

4. Select the data adapter type you want and click **Next**.

5. Enter a name for your adapter. This name is used when you select an adapter for a report.
6. Enter the properties needed by the adapter type you selected. For example, for a database JDBC connection you need to select a JDBC driver and set the URL and database username and password. For a CSV file, you need to enter a filename, column names, and the column separator.
7. (Optional) If you want to test the connection, click the **Test** button if available.
8. Click **Finish** to create the adapter.
The adapter is saved as an XML file in the project location you selected.

To create a global data adapter:

When you create a global data adapter, it is available to all reports. However, if you want to deploy it to JasperReports Server, you must export it explicitly.

1. Click  in the Repository Explorer OR right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
The Data Adapters Wizard opens.
2. Select the data adapter type you want and click **Next**.
3. Enter a name for your adapter and the properties needed by the adapter type you selected. To optionally test the adapter, click the **Test** button.
4. Click **Finish** to create the adapter.

10.1.2 Importing and Exporting Data Adapters

Jaspersoft Studio enables you to import and export data adapter definitions to simplify the process of sharing data source configurations.

To export a global data adapter as an XML file:

1. In the Repository Explorer, right-click your data adapter and select **Export to File**.
Jaspersoft Studio prompts you to name the file and select the destination for the exported information.

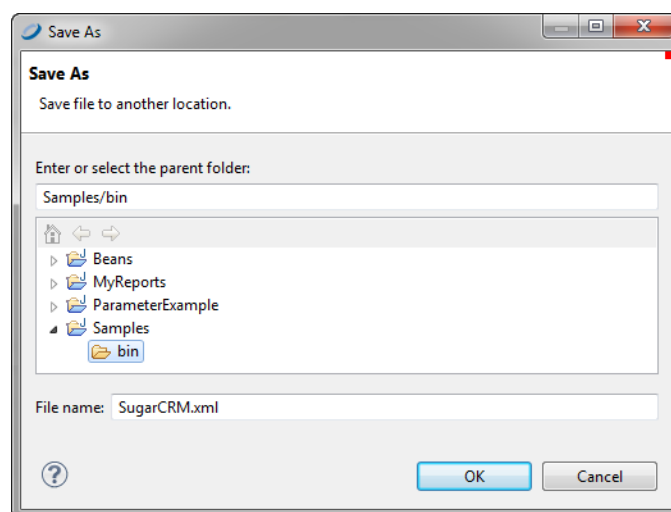


Figure 10-2 Export to File Dialog

2. Select a location in the same project as the report that will be using this adapter, enter a name for the file, and click **OK**.

A simple XML file is created in the location you chose. The data adapter must be in the same project as your report. To use the same adapter in more than one project, see [10.1.3, “Copying a Data Adapter,” on page 122](#).

To promote a data adapter file to a global data adapter:

You can make any file-based data adapter into a global adapter by importing it.

1. Right-click on the Data Adapter node in the Repository Explorer, and choose **Import from Workspace**.
2. Select the data adapter(s) you want to import.
3. You can optionally select **Overwrite Data Adapter if exists**. Otherwise, if a duplicate data source name is found during the import, Jaspersoft Studio appends a number to the imported data source name.
4. Click **OK**.

The import process adds all the selected data adapters to the current list.

10.1.3 Copying a Data Adapter

If you have saved your data adapter as an XML file, you can easily copy it between projects.

To copy a data adapter from one project to another:

1. In the Project Explorer, right-click your data adapter and select **Copy** OR use **Ctrl-C**.
2. Still in the Project Explorer, right-click the project or folder in your Jaspersoft Studio workspace that you want to use and select **Paste** OR **Ctrl-V**.

The data adapter is copied to the new location.

10.2 Using Data Adapters in Reports and Datasets

You can use the Jaspersoft Studio user interface to select the data adapter to use for previewing reports and datasets. However, this selection is specific to Jaspersoft Studio. When you want to deploy your reports to JasperReports Server or to a custom JasperReports deployment, you must specify the data adapter or data source you want to use.

10.2.1 Data Adapter For a Report

When you choose a data adapter during report creation, the drop-down lists all available adapters, with global adapters on top and project file-based adapters below. The example below shows the global adapters available with Jaspersoft Studio followed by a sample file-based adapter local to the project, named `SampleDataAdapter.xml`.

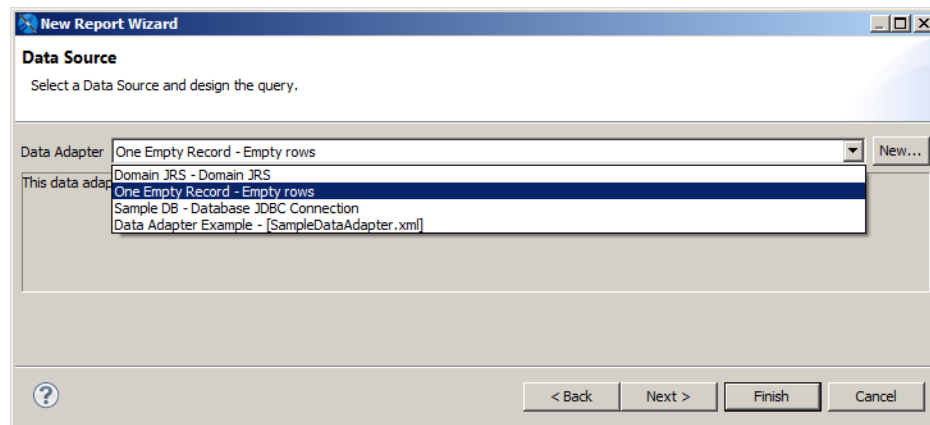


Figure 10-3 List of Data Adapters During Report Creation

Data adapters are hierarchical. That is, if no adapter is directly defined for a subdataset, it looks for the adapter of its parent dataset, then its parent's parent, and so forth.

10.2.2 Data Adapters and Report Deployment

When you use a drop-down to select a data adapter for a report or dataset, you are just setting the current default data adapter used for preview. As you continue to design your report, you can easily change this data adapter by selecting a different data adapter during preview, or by editing the dataset and changing the adapter. If you do not select a data adapter during preview, Jaspersoft Studio defaults to whichever data adapter was used most recently. If no data adapter is selected when you create a report, Jaspersoft Studio defaults to the pre-configured empty data adapter.

This data adapter is internal to Jaspersoft Studio. It is stored in an internal property (`com.jaspersoft.studio.data.defaultdataadapter`) which cannot be used in JasperReports Server or JasperReports Library. Therefore, when you publish or deploy a report, you need to specify the data source you want to use in the deployed report. You can do this in the following ways:

- When you publish a report to JasperReports Server, you can select a JasperReports Server data source to use. See [12.2, “Publishing a Report to JasperReports Server,” on page 174](#) for more information. If you choose this method to select a data source, any subdatasets in the report must use the same data source.
- You can choose to set the default data adapter explicitly for the report and/or any subdatasets. You can set this property separately for any dataset in the report. If this property is present, you cannot choose a different adapter to preview the report.

10.2.3 Default Data Adapter

You can explicitly set the data adapter for a report or dataset using the `net.sf.jasperreports.data.adapter` property.

Setting the default data adapter:

1. In Outline view, to set the data adapter for the report, click the report's root node. To set the data adapter for a dataset, click the dataset.

2. In the Properties view, to set the data adapter for a report, go to the Report tab. To set the data adapter for a dataset, go to the Dataset tab.
3. Click ... at the right of the Default Data Adapter property.
The Open Data Adapter dialog opens.

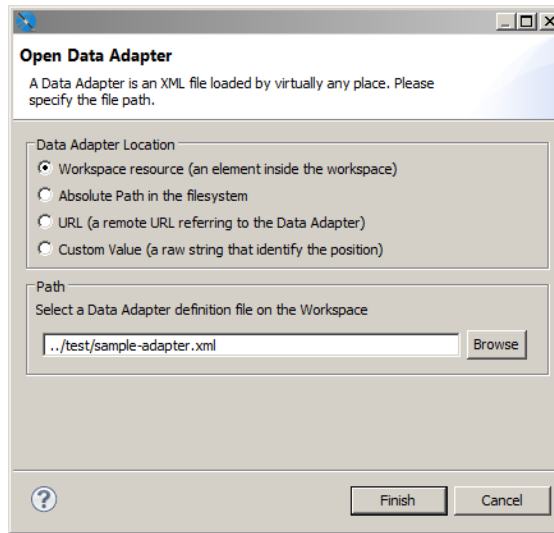


Figure 10-4 Open Data Adapter Dialog

4. Choose the format to use for specifying the data adapter location:
 - **Workspace resource** – A file in your workspace, for example, `value="../test/sample-adapter.xml"/>`. This should be a file in the same project as your report. If you want to use a global adapter, you need to export it to a file first. See [10.1.2, “Importing and Exporting Data Adapters,” on page 121](#) for more information.
 - **Absolute Path in the file system** – A file path, for example, `value="file:///C:/Adapters/sample-adapter.xml"`
 - **URL** – A remote URL that hosts the data adapter file, for example, `value="http://myserver:8080/sample-adapter.xml"`
 - **Custom value** – A free-form string that identifies the location of the data adapter to use. You could use this if you wanted to enter a string in the `repo:` syntax, for example, `value="repo:/reports/interactive/CustomersDataAdapter"` See [12.8, “Understanding the repo: Syntax,” on page 190](#) for more information.
5. If you selected **Workspace resource** or **Absolute Path**, click **Browse** to locate the file in the workspace or in your file system. Otherwise, enter the URL or free-form string.
6. Click **Finish**.
The default data adapter is set for the dataset. It is represented in the JRXML file using the `net.sf.jasperreports.data.adapter` property.

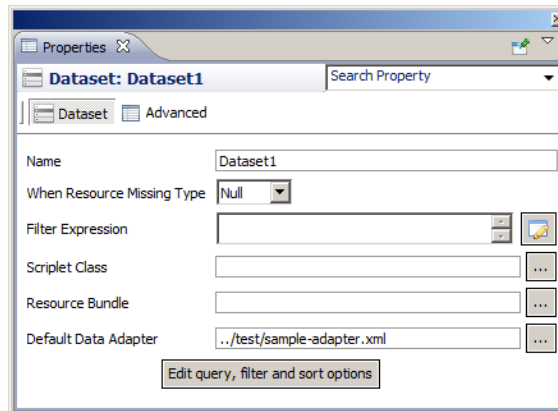


Figure 10-5 Dataset with Default Data Adapter

10.2.3.1 The JasperReports Data Adapter in the UI

When the JasperReports data adapter is present, it is shown as the bottom adapter on the list of available adapters. In the example below, New Data Adapter has been set as the default data adapter:

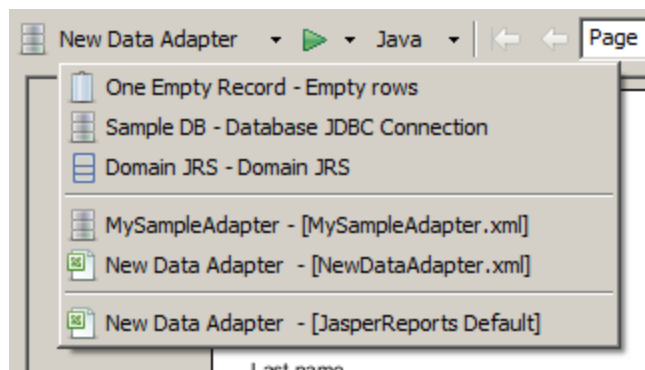


Figure 10-6 List of Data Adapters Including Default Data Adapter


10.3 Working with Database JDBC Connections

A JDBC connection lets you use a database accessed through a JDBC driver (such as a relational DBMS). When you use a JDBC connection in a report, you must specify a query.

10.3.1 Creating a Database JDBC Connection

To create a new JDBC connection:

1. Create the connection globally or locally:
 - To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.

- To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**. The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).
- From the list, select **Database JDBC connection** to open the Data Adapter dialog.

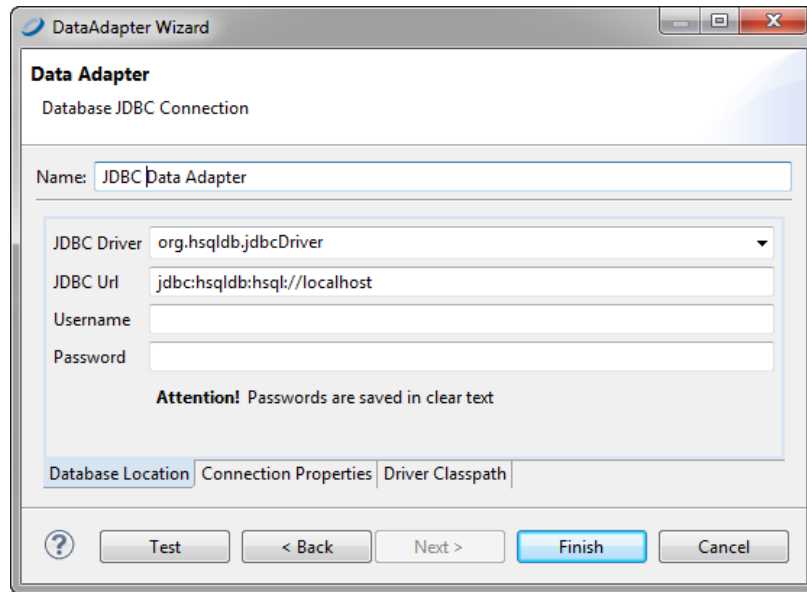


Figure 10-7 Configuring a JDBC Connection

- Name the connection (use a significant name like `MySQL - Test`). This is the name that will appear on the list of available connections when you create a report.
- In the **JDBC Driver** field, specify the JDBC driver to use for your database connection. The drop-down displays the names of the most common JDBC drivers.

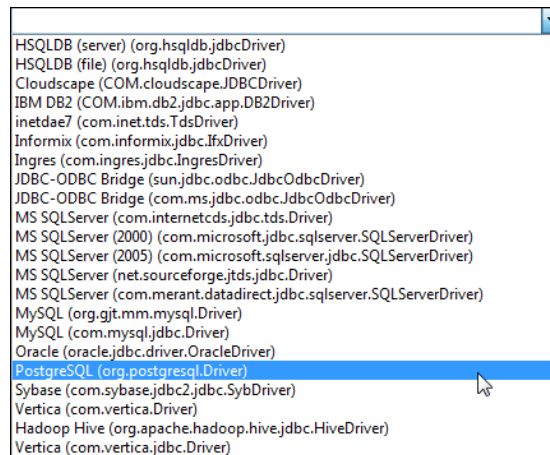


Figure 10-8 JDBC Drivers List

If a driver is displayed in red, the JDBC driver class for that driver is not present in the class path and you must obtain and install the driver before using it. See [10.3.3, “Using a Database JDBC Connection,” on page 129](#).



As of version 5.6.1, JasperReports Server includes the TIBCO JDBC drivers for the following commercial databases: Oracle, MS SQLServer and DB2. In some cases, these drivers provide functionality not provided by the vendors' driver. However, there may be some differences in queries between the two drivers. You can use the TIBCO drivers, or you can choose to install and use the driver supplied by the database vendor.

If you upload your reports to JasperReports Server, make sure to use the same driver in both JasperReports Server and Jaspersoft Studio.

5. Enter the connection URL. To have Jaspersoft Studio construct the URL, click the **Wizard** button. The JDBC URL Wizard inserts the server name and the database name in the correct text fields.
6. Enter a username and password to access the database. If the password is empty, it is better if you specify that it be saved. You can choose to save the password in one of two ways:
 - Clear text – This is not secure, but can sometimes be convenient when working in a developer or staging environment.
 - Eclipse secure storage – This is the correct option for security, but can be difficult to work with when testing and saving adapters. In addition, it can make it difficult to share adapters with other developers or deploy data adapters to JasperReports Server.
7. After you've inserted all the data, click the **Test** button to verify the connection. If everything's okay, you'll see a message that the test was successful.
8. Click **OK** to exit the message.
9. Click **Finish** to create the connection.

10.3.2 Troubleshooting a Database JDBC Connection

When the tests fail, the most common exceptions are:

- A `ClassNotFoundException` 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.).

10.3.2.1 `ClassNotFoundException`

The `ClassNotFoundException` exception occurs whenever a data adapter fails to load a class it requires. In the context of JDBC connections, the most likely cause is that the required JDBC driver is not present in the classpath. In general, a data adapter has two classpaths it uses to find libraries. First the adapter looks at any paths that were specified inside the data adapter when it was created. If it cannot load the libraries or classes it needs using its internal paths, the data adapter uses the Jaspersoft Studio classpath to look for them.

The Jaspersoft Studio classpath is defined in your Eclipse project. As Jaspersoft Studio uses its own class loader, it's enough to add resources such as jar files and directories containing classes to the Jaspersoft Studio classpath.

For example, suppose you want to create a connection to an Oracle database. Jaspersoft Studio does not ship the vendor's driver for this database. If you choose the `oracle.jdbc.driver.OracleDriver` driver, when you test the connection, you'll see the `ClassNotFoundException`, as shown in [Figure 10-9](#). You need to add the JDBC driver for Oracle, `ojdbc14.jar`, to the classpath.

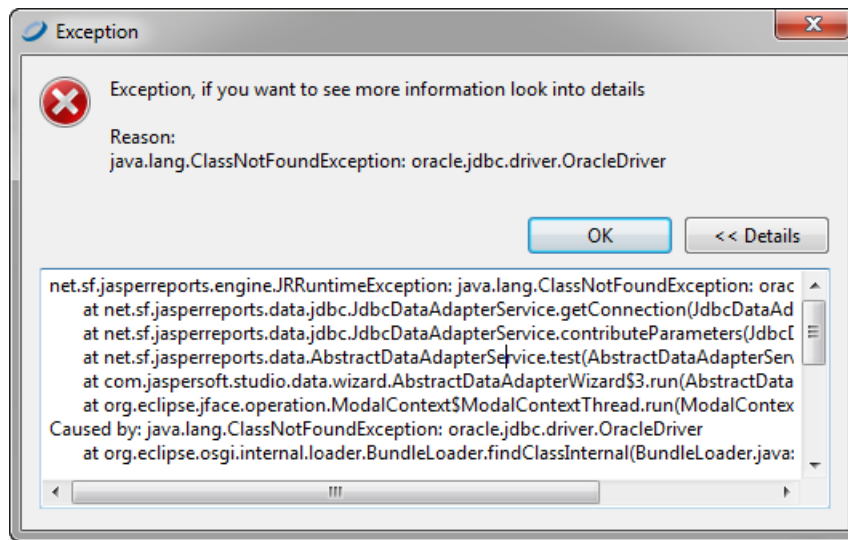


Figure 10-9 `ClassNotFoundException` exception

To add a resource to the Jaspersoft Studio classpath:

If you add a resource to the Jaspersoft Studio classpath, it will be available to all data adapters. In addition to JARs, you can add variables, libraries, class folders, and external class folders. To add a jar to the Jaspersoft Studio classpath:

1. Click **Project > Properties > Java Build Path > Libraries**, and click **Add JARs** or **Add External JARs**.
2. Browse to locate the jar you want to add.
3. Select the file you want to add to the classpath.
4. Click **OK**.

To add a JAR file to a data adapter's classpath:

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.

1. If the adapter is not already open, double-click its icon in the Repository Explorer or Project Explorer to open it.
2. Click on the Driver Classpath tab.
3. Click Add and browse to locate the jar you want to add. If you want to add a different file type, use the menu at the bottom right.
4. Click on the jar and **Open**.

The location of the file you chose is added to the driver classpath.

10.3.2.2 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.

Use the JDBC URL Wizard to build the JDBC URL and try again.

10.3.2.3 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 database returns a message is fairly explicit about the reason behind the failure of the connection.

10.3.3 Using a Database JDBC Connection

When you create a report with a JDBC connection, you specify a query to extract records from the database. The query language you use depends on the connection type; the most common query type is an SQL query.

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

10.3.3.1 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 actually used in the report. For each field, specify name and type. **Table 10-1, “Conversion of SQL and JAVA types ,” on page 129** 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.

10.3.3.2 Filtering Records

The records in a report can be ordered and filtered. Set sort and filter options in the Report query dialog by clicking the **Dataset and Query** button .

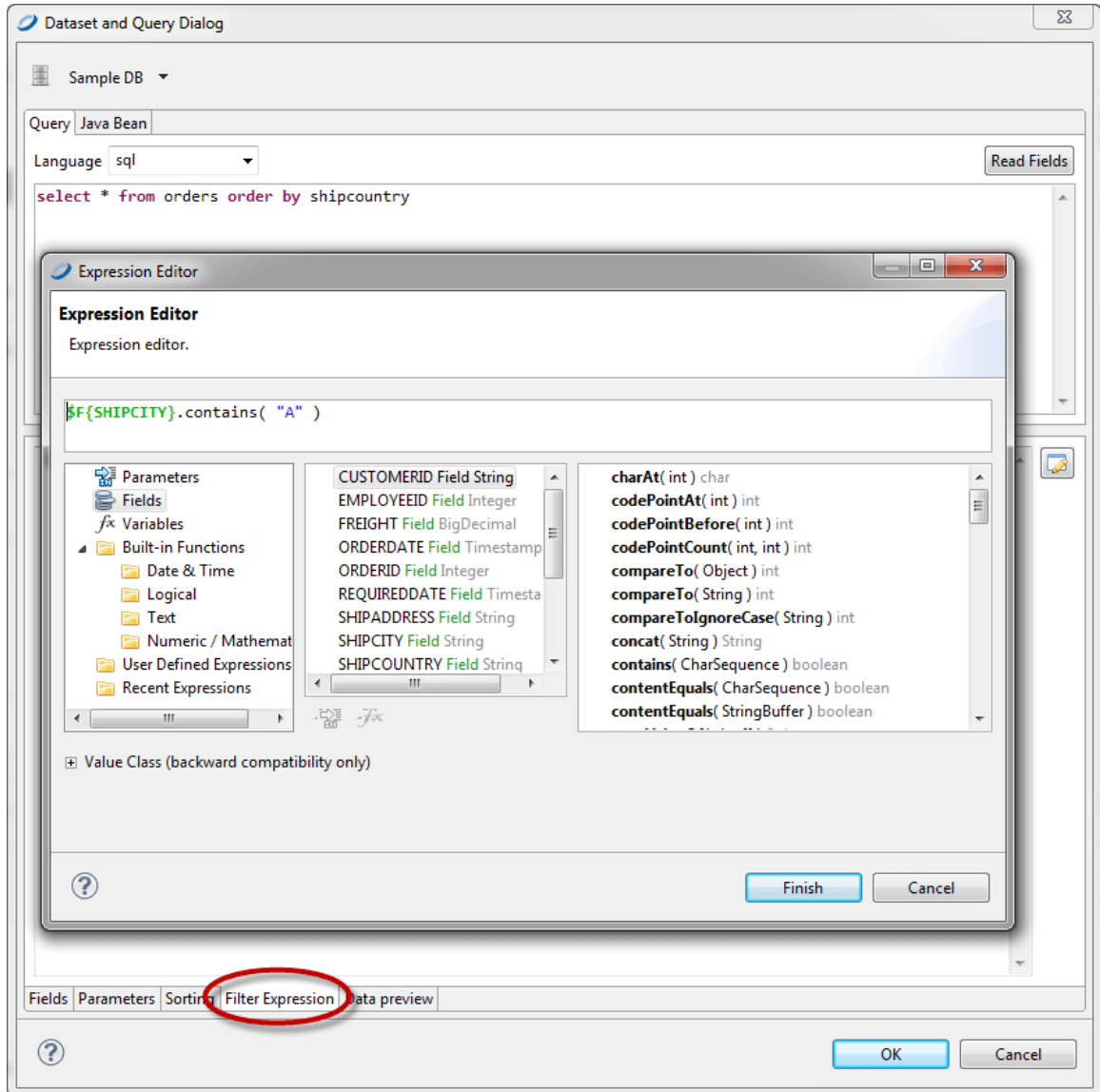


Figure 10-10 Filter Expression Tab and Expression Editor

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.

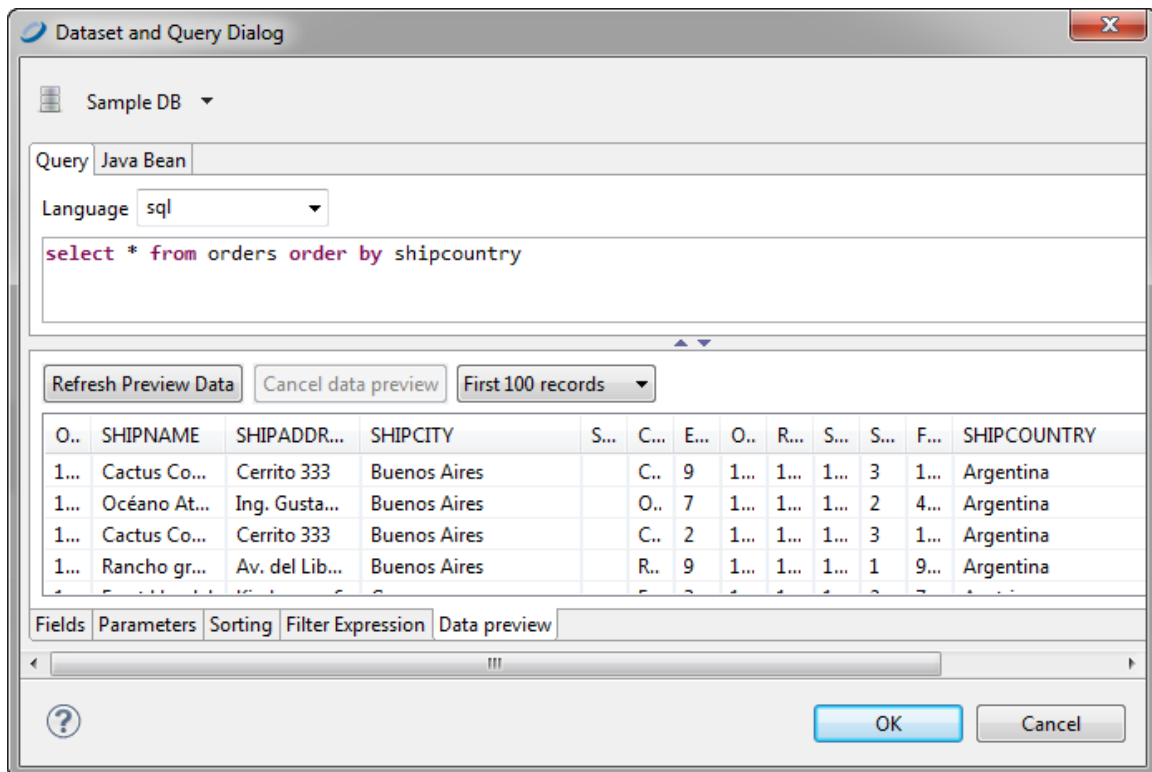


Figure 10-11 Data Preview

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.

10.3.3.3 Using JDBC Connections for Subreports

You can also use a JDBC 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.


10.4 Working with a MongoDB Data Adapter

MongoDB is a big data architecture based on the NoSQL model that is neither relational nor SQL-based. Jaspersoft Studio includes data adapters that allow reports to use a native MongoDB data connection or a MongoDB JDBC data adapter. As of version 6.1.1 JasperReports Server also supports SSL and x509 authentication for MongoDB.

10.4.1 Creating a Native MongoDB Connection

To create a MongoDB data adapter with the native driver:

Follow these steps to create a MongoDB data source with the native MongoDB driver.

1. Create the connection globally or locally:
 - To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
 - To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**.

The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).

2. From the list, select **MongoDB Connection** to open the Data Adapter dialog.

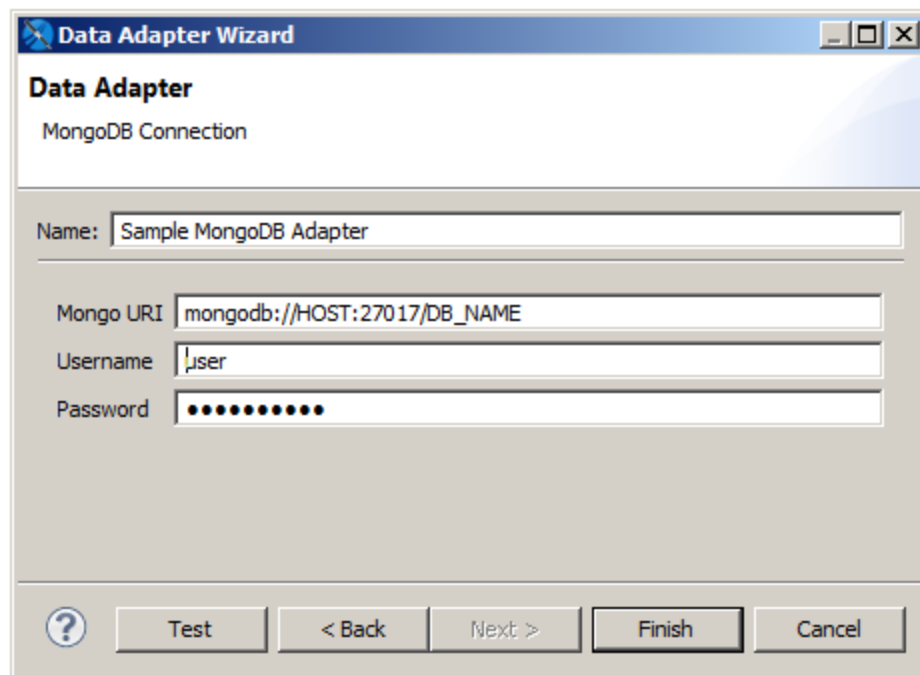


Figure 10-12 Configuring a MongoDB Connection

3. Fill in the required fields:
 - **Name:** The name that will appear on the list of available data adapters when you create or run a report.
 - **Mongo URI:** The URI of your MongoDB data.
4. If you have configured your MongoDB source to be password protected, specify a valid username and password.
5. Click **Test** to check the values you entered. If everything's okay, you'll see a success message.
6. Click **OK** to exit the message.
7. Click **Finish** to create the connection.



If you get a `ClassNotFoundException` exception, the most likely cause is that the required driver is not present in the classpath. See [10.3.2.1, “ClassNotFoundException,” on page 127](#) for more information.

10.4.1.1 The Jaspersoft MongoDB Query Language

MongoDB is designed to be accessed through API calls in an application or a command shell. As a consequence, it does not have a defined query language. In order to write queries for MongoDB data sources, we have developed a query language based on the JSON-like objects upon which MongoDB operates. JSON is the JavaScript Object Notation, a textual representation of data structures that is both human- and machine-readable.

The Jaspersoft MongoDB Query Language is a declarative language for specifying what data to retrieve from MongoDB. The connector converts this query into the appropriate API calls and uses the MongoDB Java connector to query the MongoDB instance. The following examples give an overview of the Jaspersoft MongoDB Query Language, with SQL-equivalent terms in parentheses:

- Retrieve all documents (rows) in the given collection (table):

```
{ 'collectionName' : 'accounts' }
```

- From all documents in the given collection, select the named fields (columns) and sort the results:

```
{
  'collectionName' : 'accounts',
  'findFields' : { 'name':1, 'phone_office':1, 'billing_address_city':1,
                  'billing_address_street':1, 'billing_address_country':1},
  'sort' : { 'billing_address_country':-1, 'billing_address_city':1}
}
```

- Retrieve only the documents (rows) in the given collection (table) that match the query (where clause). In this case, the date is greater-than-or-equal to the input parameter, and the name matches a string (starts with N):

```
{
  'collectionName' : 'accounts',
  'findQuery' : {
    'status_date' : { '$gte' : $P{StartDate} },
    'name' : { '$regex' : '^N', '$options' : '' }
  }
}
```

The Jaspersoft MongoDB Query Language also supports advanced features of MongoDB such as map-reduce functions and aggregation that are beyond the scope of this document. For more information, see the [language reference](#) on the Community website.

When you create a report or subdatasource from a native MongoDB connection, Jaspersoft Studio automatically selects MongoDBQuery as the query language. You can explicitly view and set the query language using the Dataset and Query Dialog.

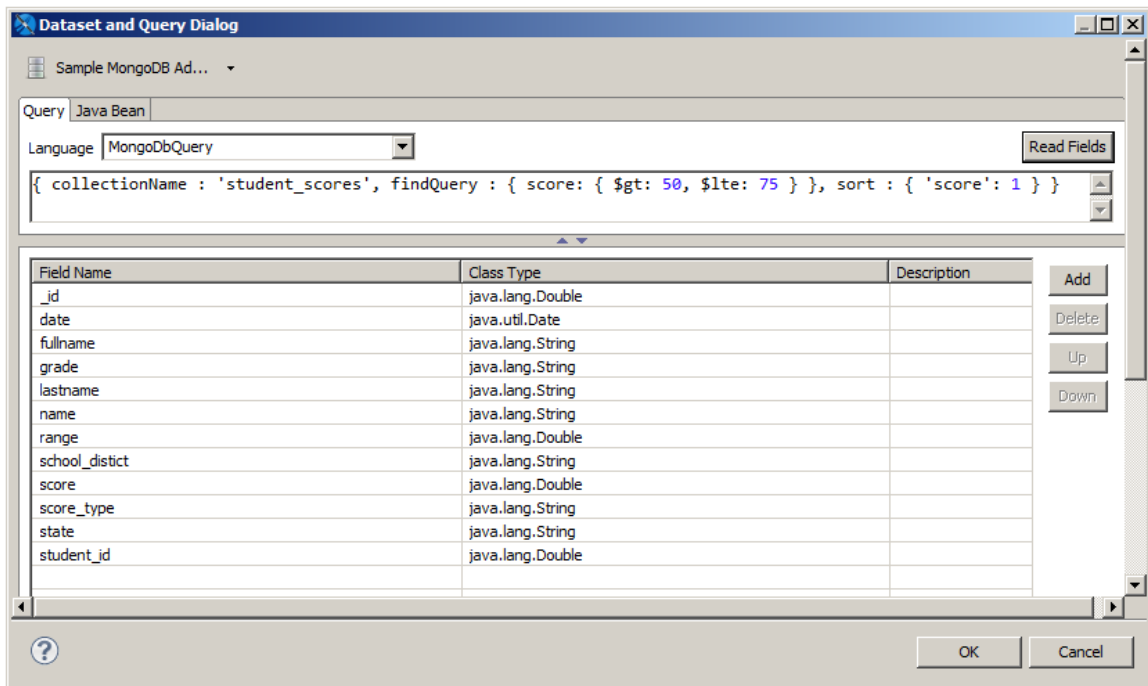


Figure 10-13 Example Dataset and Query Dialog for a MongoDB Data Adapter


10.4.2 Creating a MongoDB JDBC Data Source

If you want to wrap your MongoDB data source in a domain or virtual data source, create a MongoDB JDBC data source. The MongoDB JDBC driver can create a default normalized schema for your data or, if you prefer, you can load a schema from the repository or your server file system.



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.

To create a MongoDB JDBC data source:

1. Create the connection globally or locally:
 - To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
 - To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**.
The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).
2. From the list, select **Database JDBC Connection** to open the Data Adapter dialog.
3. In the JDBC Driver field, select **MongoDB (TIBCO Jaspersoft)**.

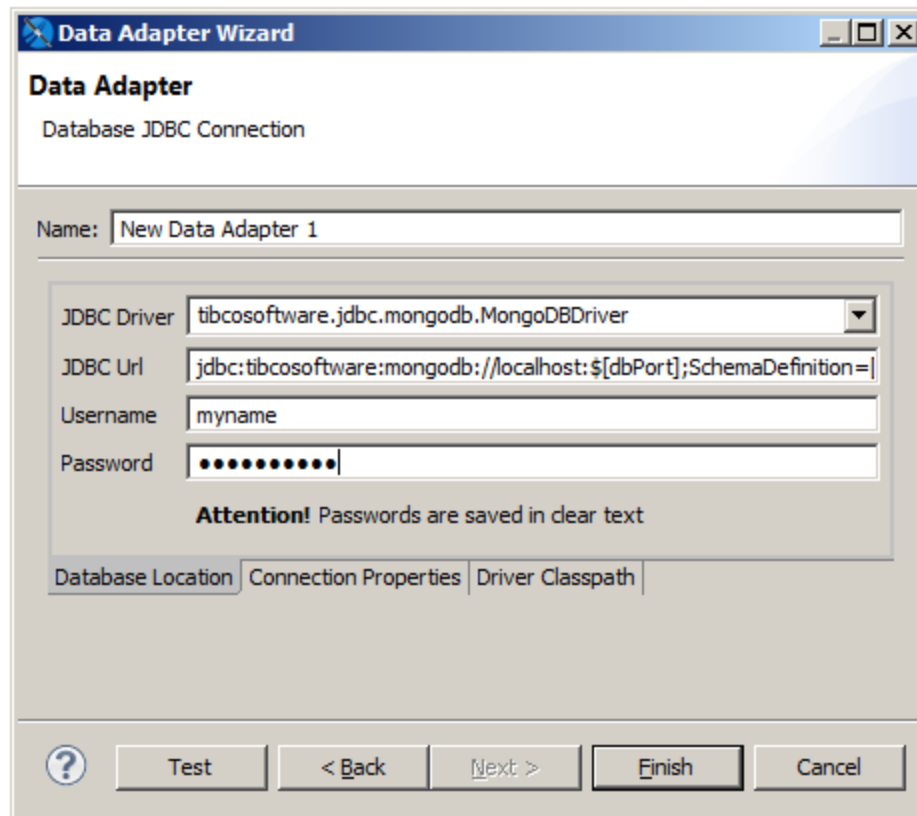


Figure 10-14 MongoDB JDBC Data Adapter Dialog

4. Fill in the required fields:
 - **Name:** The name that will appear on the list of available data adapters when you create or run a report.
 - **JDBC URL:** The URL for your MongoDB instance. This is in the following format:
`jdbc:tibcosoftware:mongodb://host:port;databaseName=db;SchemaDefinition=file`
 where:
 - **host:port** is the host and port of your MongoDB instance
 - **db** is the name of the database to use in your MongoDB instance
 - **file** (optional) is a location on your local disk where you want to store the schema definition. When you first connect to a MongoDB server, the driver automatically creates a normalized schema of the data and generates a SchemaDefinition for housing and sharing the normalized schema. You can also specify the path of an existing schema on your local disk.
5. If you have configured your MongoDB source to be password protected, specify a valid username and password.
6. (Optional) Click **Connection Properties** to specify any additional properties for your connection. For example, if you're using MongoDB 3.0 and you want to enable SSL, enter:
`EncryptionMethod=SSL;ValidateServerCertificate=false`
 To enable both SSL encryption and self-signed CA, enter the TrustStore and KeyStore paths and the KeyStore password. For example:
`EncryptionMethod=SSL;TrustStore=<path>;KeyStore=<path>;KeyStorePassword=<password>;`

- Click **Test** to check the values you entered. If everything's okay, you'll see this message:

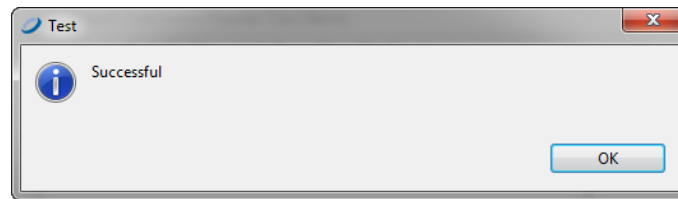


Figure 10-15 Test Confirmation Dialog

- Click **OK** to exit the message.
- Click **Finish** to create the connection.



If you get a `ClassNotFoundException` exception, it may be due to one of the following:

- You are not licensed to use the TIBCO MongoDB JDBC driver. This driver is only available in commercial editions.
- The required driver is not present in the classpath. See [10.3.2.1, “ClassNotFoundException,” on page 127](#) for more information.

10.5 Working with a Native Cassandra Connection

The Apache Cassandra database provides scalability and high availability for certain applications of big data. For more information about Cassandra, see <http://cassandra.apache.org/>.

The Cassandra data adapter relies on a driver that has certain limitations on how your data can be structured and accessed:


- The current version of Cassandra does not support NULL values in the data. All required fields must have non-NULL default values.
- The current version of the driver does not support aggregate functions (sum, min, max).
- For query parameters, the current version of the driver supports `$X(IN...)`, but no other `$X` functions.

The Cassandra data adapter supports queries in the Cassandra Query Language 3 (CQL3). To improve performance, design your Cassandra data using the following guidelines:

- Specify the `ALLOW FILTERING` suffix to speed up queries.
- All fields referenced in `WHERE` clauses of a query should be indexed.

10.5.1 Creating a Native Cassandra Data Adapter

To create a new Cassandra adapter:

- Create the connection globally or locally:
 - To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
 - To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**.

The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).

- From the list, select **Cassandra Connection** to open the Data Adapter dialog.

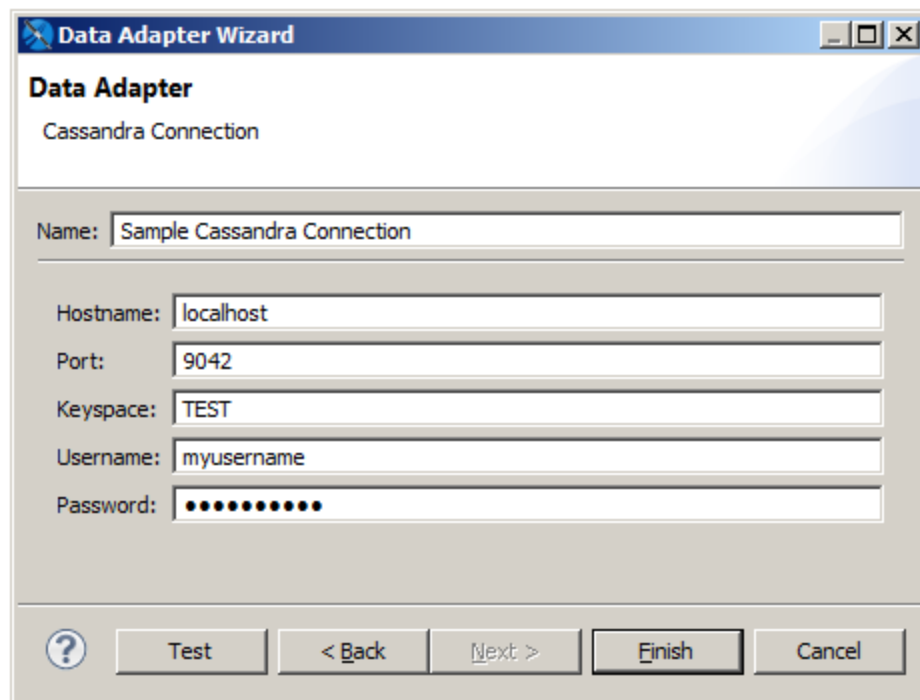


Figure 10-16 Configuring a Cassandra Native Connection

- Fill in the required fields:
 - Name: The name that will appear on the list of available data adapters when you create a report.
 - Port: Use port 9042 with the Cassandra data source. Cassandra's default port of 9160 is for the Thrift client that is commonly used with Cassandra. To use the Cassandra Query Language (CQL) with your Cassandra data source, you may need to configure your Cassandra instance as follows:


```
start_native_transport: true
native_transport_port: 9042
```
 - Keyspace: The keyspace of your Cassandra instance.
- If you have configured your Cassandra source to be password protected, specify a valid username and password. Due to compatibility issues, Cassandra authentication is supported only when you use Cassandra 1.12.18 and above. If the password is empty, it is better if you specify that it be saved. You can choose to save the password in one of two ways:
 - Clear text – This is not secure, but can sometimes be convenient when working in a developer or staging environment.
 - Eclipse secure storage – This is the correct option for security, but can be difficult to work with when testing and saving adapters. In addition, it can make it difficult to share adapters with other developers or deploy data adapters to JasperReports Server.
- Click **Test** to check the values you entered. Make sure that the port is set to 9042, because the connection test will also work with the wrong port (9160). If everything's okay, you'll see a success message.
- Click **OK** to exit the message.
- Click **Finish** to create the connection.



If you get a `ClassNotFoundException` exception, the most likely cause is that the required driver is not present in the classpath. See 10.3.2.1, “[ClassNotFoundException](#),” on page 127 for more information.

10.5.2 Using a Cassandra Connection

When you create a report or subdatasource from a native Cassandra connection, specify a CQL query to extract records from the database.

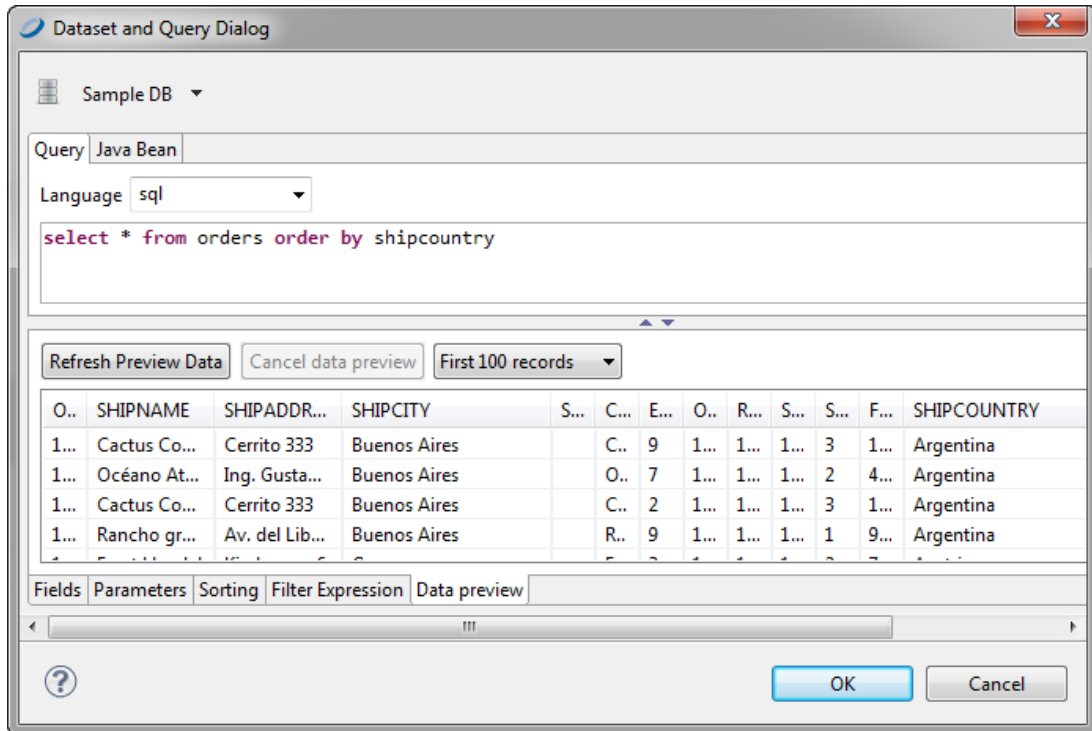


Figure 10-17 Data Preview

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.

10.6 Working with a Collection of JavaBeans Data Adapter

A collection of JavaBeans data adapter 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).

10.6.1 Implementing the Factory Class for a Collection of JavaBeans

The collection of JavaBeans data adapter uses an external class (named `Factory`) to produce some objects (the JavaBeans) that constitute the data to pass to the report. To use a collection of JavaBeans as a data adapter in Jaspersoft Studio, you must create an instance of the `Factory` class and provide a static method to instantiate different JavaBeans and to return them as a collection (`java.util.Collection`) or an array (`Object[]`). The following example shows how you might create write an instance of the `Factory` class.

Suppose that you have an collection of JavaBeans, where the data is represented by a set of objects of type `PersonBean`. The following table shows the code for `PersonBean`, which contains two fields: `name` (the person's name) and `age`:

Table 10-2 `PersonBean` example

```
public class PersonBean
{
    private String name = "";
    private int age = 0;

    public PersonBean(String name, int age)
    {
        this.name = name;
        this.age = age;
    }
    public int getAge()
    {
        return age;
    }

    public String getName()
    {
        return name;
    }
}
```

To use this collection of beans, you need to create an instance of the `Factory` class. Your class, named `TestFactory`, must contain the actual data that is used by the report. In this case, it will be something similar to this:

Table 10-3 `PersonBean` example - Class result

```
public class TestFactory
{

    public static java.util.Collection generateCollection()
    {
        java.util.Vector collection = new java.util.Vector();
        collection.add(new PersonBean("Ted", 20) );
        collection.add(new PersonBean("Jack", 34) );
        collection.add(new PersonBean("Bob", 56) );
        collection.add(new PersonBean("Alice",12) );
        collection.add(new PersonBean("Robin",22) );
        collection.add(new PersonBean("Peter",28) );


        return collection;
    }
}
```

A data adapter based on this class would represent five JavaBeans of `PersonBean` type.

10.6.2 Creating a Data Adapter from a Factory Class

Once you have created your `Factory` class instance, you can create a data adapter that uses your collection of JavaBeans.

1. Create the connection globally or locally:

- To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
- To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**.

The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).

2. To create a connection to handle JavaBeans, select **Collection of JavaBeans** in the list of data adapter types.

The fields necessary to create a collection JavaBeans appear.

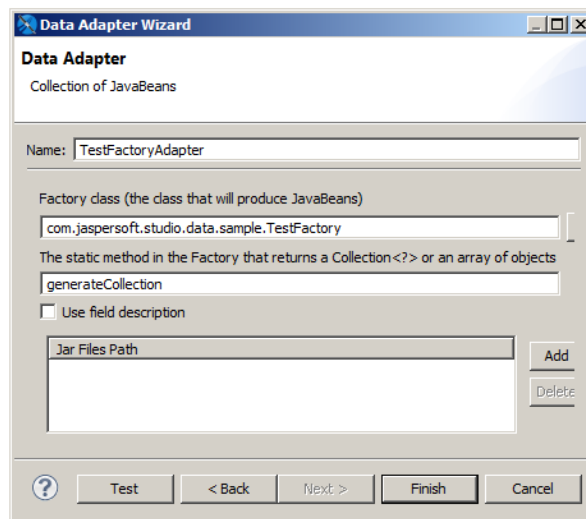


Figure 10-18 Collection of JavaBeans Data Adapter

3. Create a name for your adapter.
4. Enter the name of your Java class in the Factory class. For the example above, you would need to specify the class name for `TestFactory`.
5. Enter the name of the static method in your Factory class. In the example above, this is `generateCollection`.
6. By default, the field names in your JavaBeans become the field names in your data adapter. If your JavaBeans definition has field descriptions, and you want to use these as names in JasperSoft Studio, select **Use field description**.
7. If necessary, you can add the path to your jar files.

10.6.3 Registering the Fields

One peculiarity of a collection of JavaBeans data adapter is that the fields are exposed through `get` methods. This means that if the JavaBean has a `getXYZ()` method, `xyz` becomes the name of a record field (the JavaBean represents 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 `TestFactoryAdapter`.) To refer to an attribute of an attribute, use periods as a separator. 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()`.

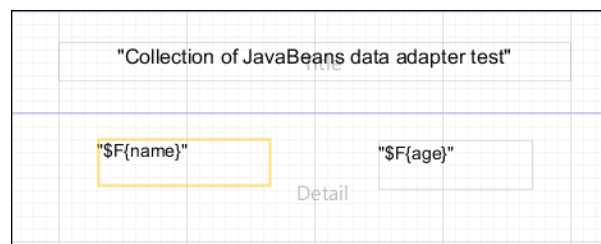


Figure 10-19 Layout of a Report Based on JavaBeans

If you selected **Use field description** when you specified the properties of your data adapter, the mapping between JavaBean attribute and field value uses the field description instead of the field 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.

10.7 Working with XML Data Adapters

JasperReports supports data adapters for XML documents.

10.7.1 Creating a Node Set for an XML Document

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 10-4 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 in the following table.

Table 10-5 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>
```

10.7.2 Creating an XML Data Adapter

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

1. Create the connection globally or locally:

- To create the connection globally, right-click **Data Adapters** in the Repository Explorer and choose **Create Data Adapter**.
- To create the connection local to a project, click , enter a name and location for the data adapter in the DataAdapter File dialog box, and then click **Next**.

The Data Adapter wizard appears (see [Figure 10-1, “Data Adapter Wizard,” on page 120](#)).

2. From the list, select **XML document** to open the Data Adapter dialog.

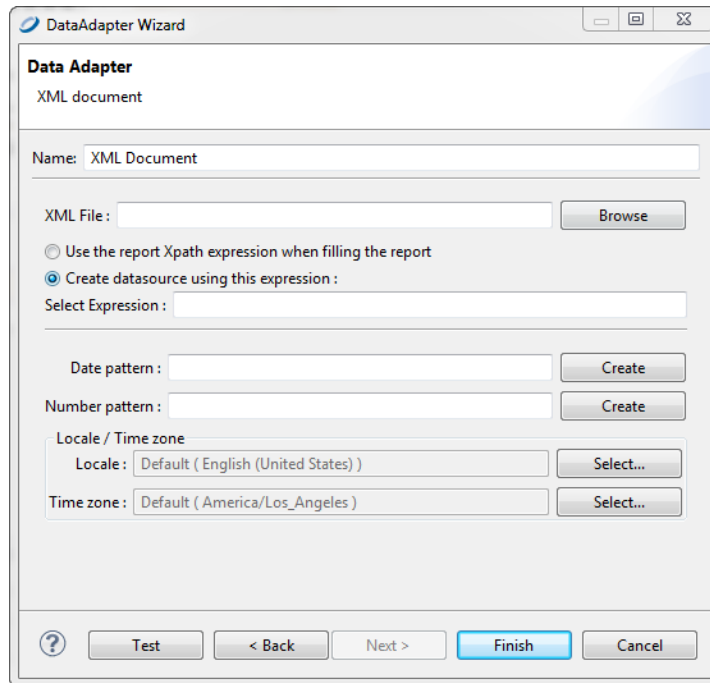


Figure 10-20 Configuring an XML Data Adapter

3. Enter a name for your adapter.
4. **XML file** is the only required field. Choose an XML file or enter the URL where your XML data is located.
5. (URL only.) If you entered a URL in the **XML file** field, click the **Options** button to open the Http Connection Options dialog box.

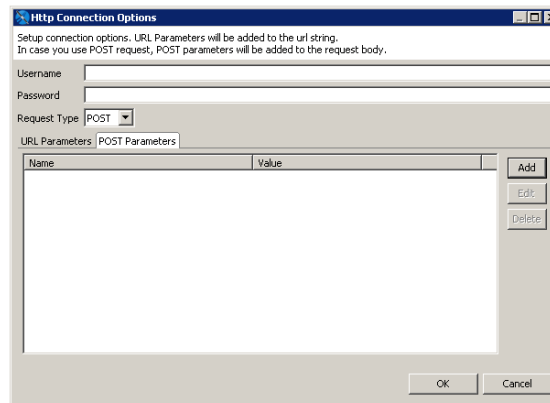


Figure 10-21 HTTP Connection Options

In this dialog you can enter the following options:

- **Username** and **Password** (optional) – The username and password to use if your CSV location requires authentication.
- **Request Type** – Select GET (default) or POST.

- To add a parameter to the request URL, click **Add** in the URL Parameters tab. Enter the name and value of your parameters in the Parameter dialog and click **OK**. For multiple parameters, add each parameter separately.
- For a POST request, to add parameters to the body of the POST, click **Add** in the POST Parameters tab. Enter the name and value of your parameters in the Parameter dialog and click **OK**. For multiple parameters, add each parameter separately.

When you have configured your request, click **OK**.

6. Choose whether to provide a set of nodes using a pre-defined static XPath expression, or set the XPath expression directly in the report.

We recommend 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.

10.7.3 Registration of Fields for an XML Data Adapter

In addition to the type and name, the definition of a field in a report using an XML data adapter requires an expression inserted as a field description. As the data adapter 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
<code>id</code>	<code>@id</code>	Integer
<code>lastname</code>	<code>lastname</code>	String
<code>firstname</code>	<code>firstname</code>	String
<code>name of category</code>	<code>ancestor::category/@name</code>	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 adapter, 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
```

10.7.4 XML Data Adapters 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 10-6, “Complex XML example,” on page 146](#). This is a slightly modified version of [Table 10-4, “Example XML file,” on page 142](#). For each person node, a hobbies node is added which contains a series of hobby nodes and one or more e-mail addresses.

Table 10-6 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)
    ${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)
    ${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 (`{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 the empty data adapter (select it from the drop-down 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 adapter is the versatility of XPath, which allows navigation of the node selection in a refined manner.

10.8 Working with 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.

10.8.1 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 tab, 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 11, "Creating Queries," on page 161](#).

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.

10.9 Working with CSV Data Adapters

You can create a connection based on a CSV file or URL location.

To create a connection based on a CSV file:

1. Click the **New** button in the Connections/Datasources dialog and select **CSV File** from the list of data adapter types.

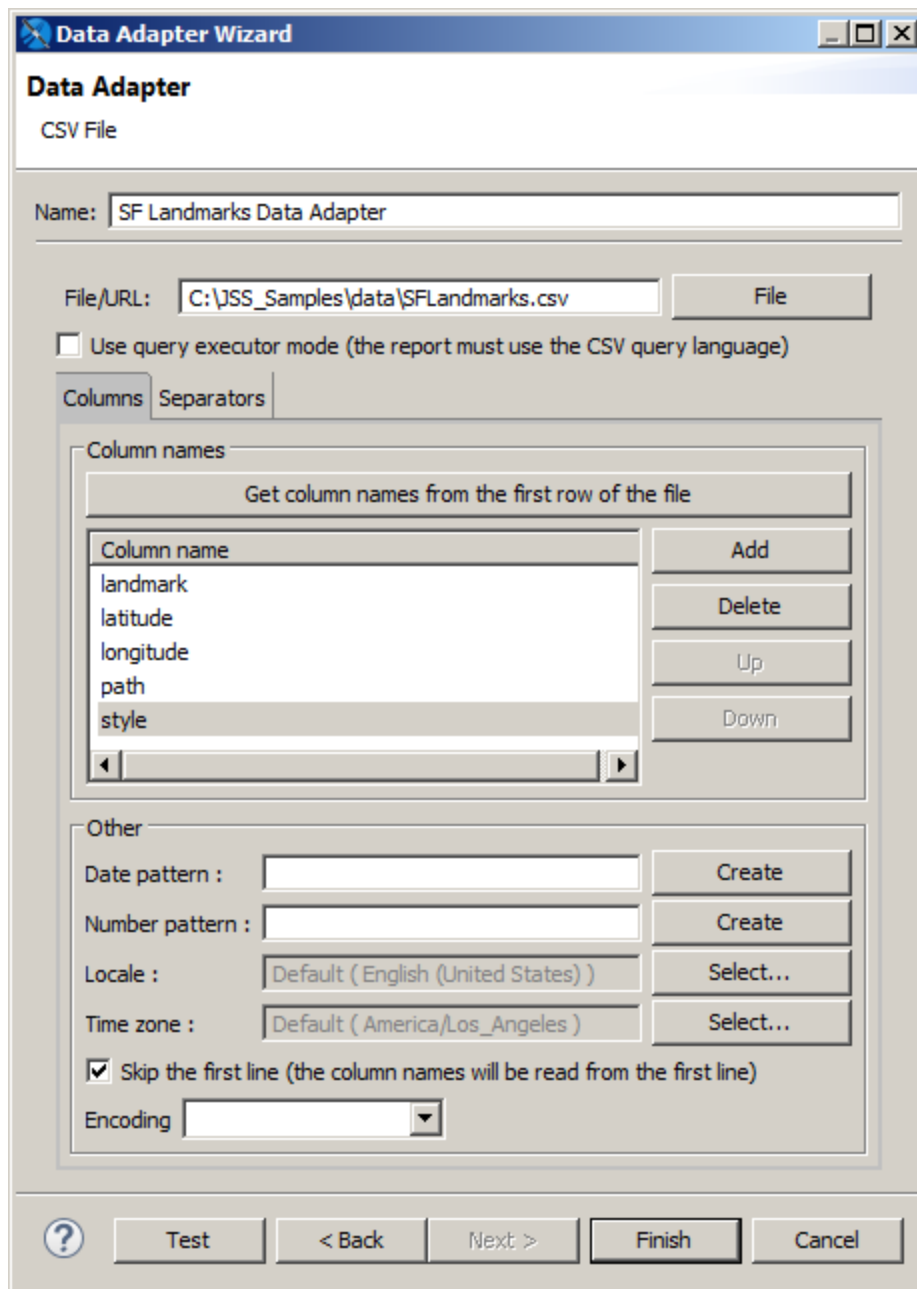


Figure 10-22 CSV Data Adapter

2. Set a name for the connection.
3. In the **CSV file** field, choose a CSV file or enter the URL where your CSV data is located.
4. (URL only.) If you entered a URL in the **CSV file** field, click the **Options** button to open the Http Connection Options dialog box.

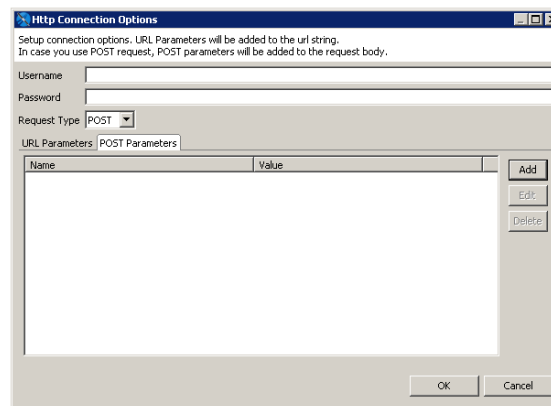


Figure 10-23 HTTP Connection Options

In this dialog you can enter the following options:

- **Username** and **Password** (optional) – The username and password to use if your CSV location requires authentication.
- **Request Type** – Select GET (default) or POST.
- To add a parameter to the request URL, click **Add** in the URL Parameters tab. Enter the name and value of your parameters in the Parameter dialog and click **OK**. For multiple parameters, add each parameter separately.
- For a POST request, to add parameters to the body of the POST, click **Add** in the POST Parameters tab. Enter the name and value of your parameters in the Parameter dialog and click **OK**. For multiple parameters, add each parameter separately.

When you have configured your request, click **OK**.

5. Declare the fields in the data adapter.
 - 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).

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

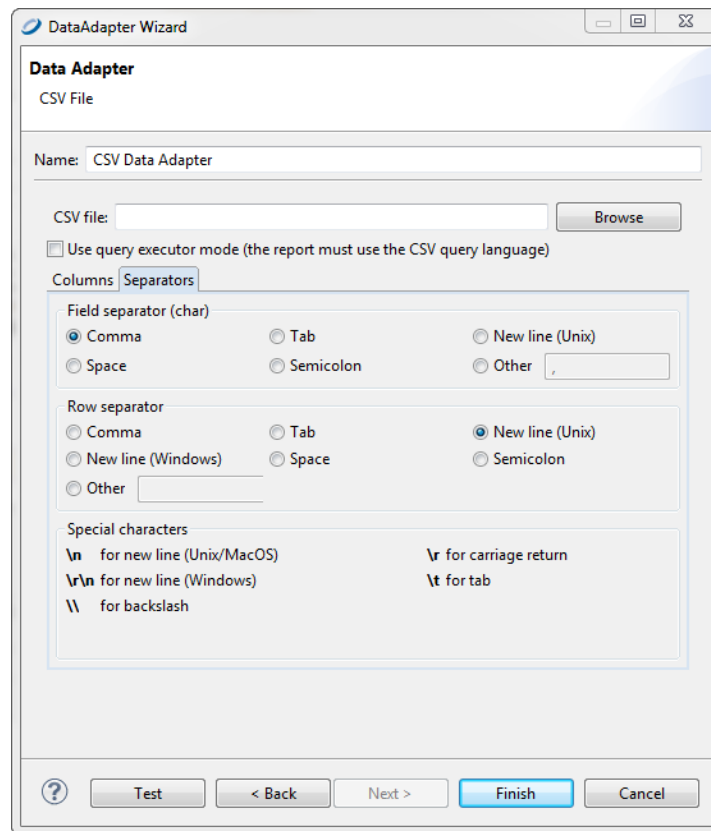


Figure 10-24 Separators Tab

7. Click **Finish**.

10.9.1 Registration of the Fields for a CSV Data Adapter

When you create a CSV data adapter, you must define a set of column names as fields for your report. To add them to the fields list, set your CSV data adapter 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 adapter level by selecting the **Use custom date format** check box.

10.10 Using the Empty Record Data Adapter

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 Explorer. The **Data Adapter Wizard** appears with Empty rows.

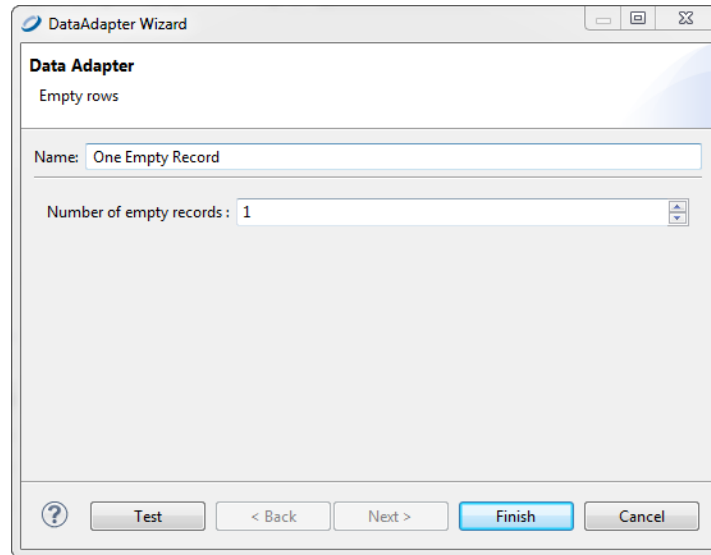


Figure 10-25 Data Adapter Wizard > Empty Record

2. Set the number of empty records you need. Remember, whatever field you add to the report, its value is set to `null`. Since this data adapter 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**.

10.10.1 Understanding the Empty Record Implementation

The empty record data adapter in JasperSoft Studio uses the special JasperReports data source `JREmptyDataSource`. This data 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.

10.11 Working with the JRDataSource Interface

All data adapters implement the `JRDataSource` interface. Some data adapters, such as the JDBC data connections, do this indirectly using a connection and a query; other data adapters, such as adapters for CSV files, XML documents, and collections of JavaBeans, do this directly. This section is useful if you want to understand more about the direct data adapters, or if you are interested in creating a custom data adapter.

10.11.1 Understanding the JRDataSource Interface

Data supplied by a `JRDataSource` is ideally organized into records as in a table. Every `JRDataSource` must implement the following two methods:

`public boolean next()` – Returns true if the cursor is positioned correctly in the subsequent record, false if no more records are available.

`public Object getFieldValue(JRField jrField)` – Moves a virtual cursor to the next record

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.

10.11.2 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 10-7 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.

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`.

Your 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 10-8 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.

10.11.3 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 collection of JavaBeans data adapter — 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 10-9 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 instantiating 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 [“Working with Database JDBC Connections” on page 125](#)), 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.

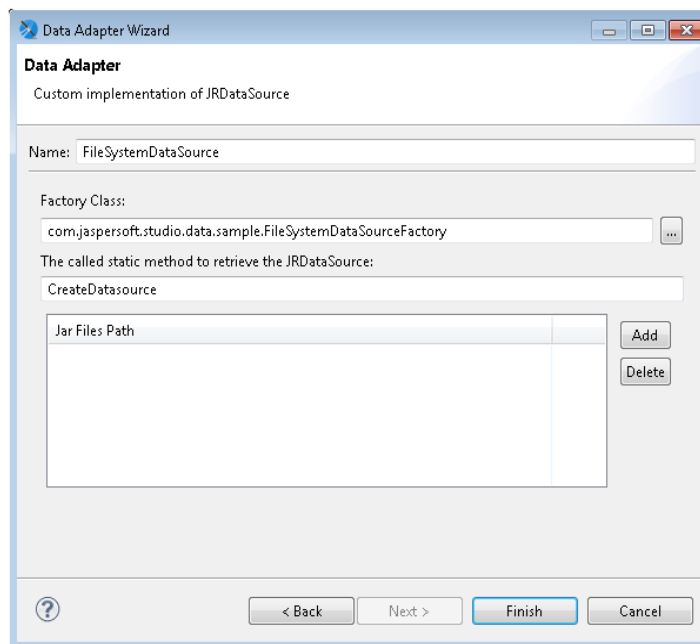


Figure 10-26 Configuring a Custom Data Adapter

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 `#{IS_DIRECTORY}`, or use as your image expression a condition like the following:

```
#{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.

10.12 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.

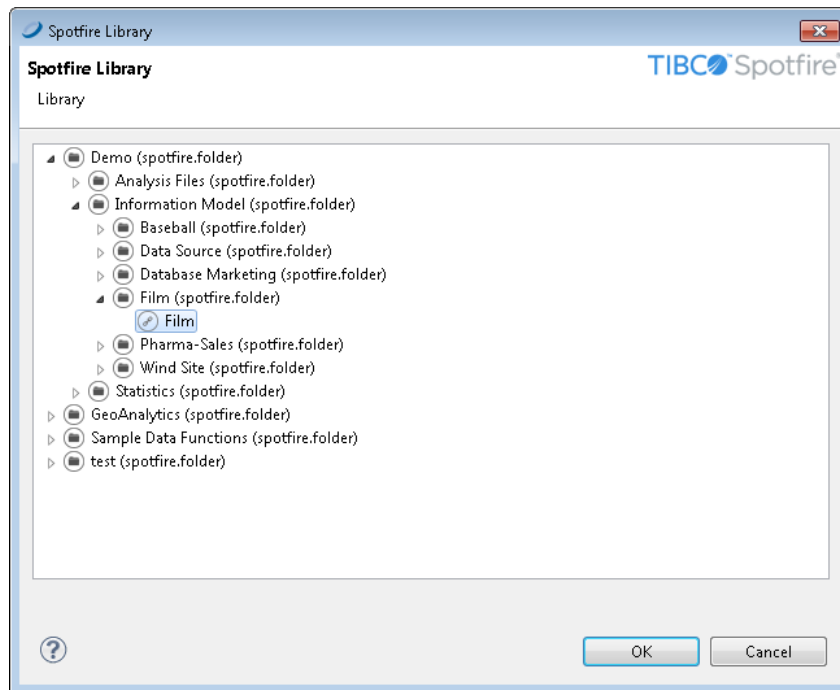


Figure 10-27 Spotfire Library displayed in Jaspersoft Studio

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**.

To configure the report to use the exported data adapter:

1. In the Outline view, 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 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**

11.1 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...**

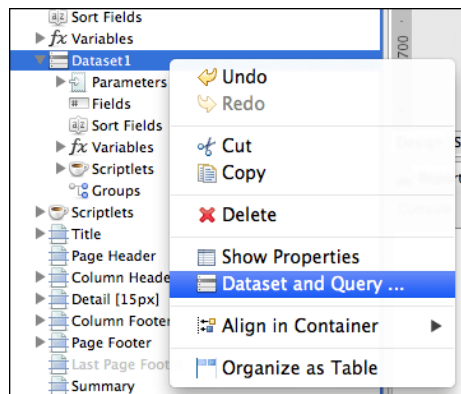


Figure 11-1 Right-Click Menu

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.

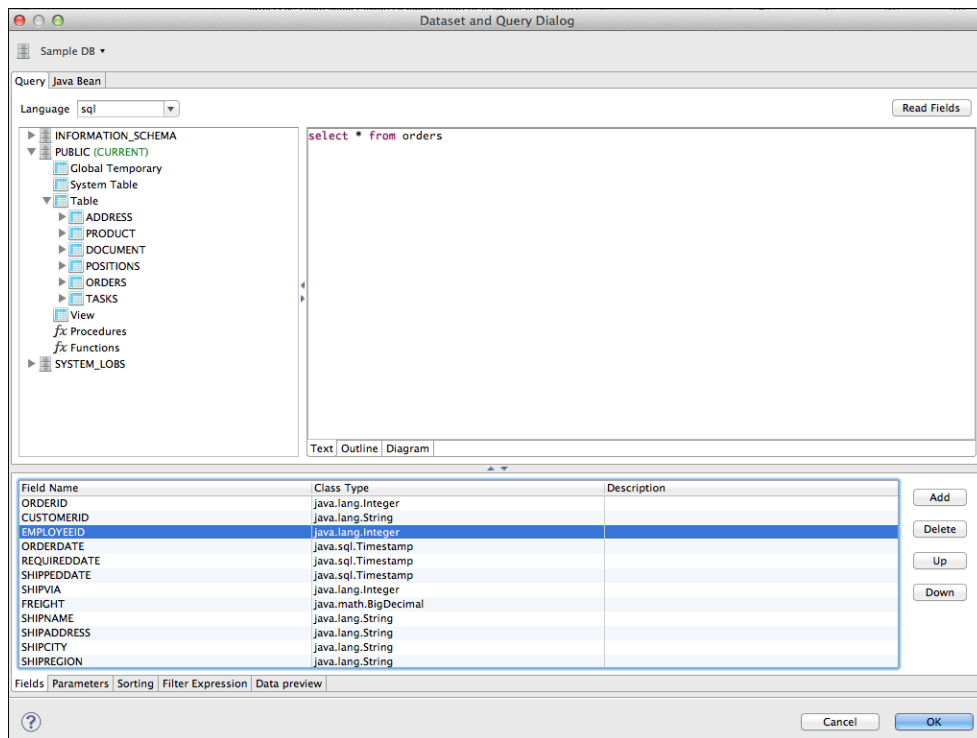


Figure 11-2 Data Preview Tab

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.

11.2 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 executor.

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.

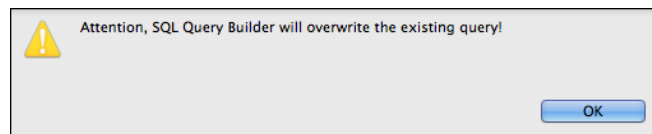


Figure 11-3 Query Overwrite Warning

11.2.1 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.

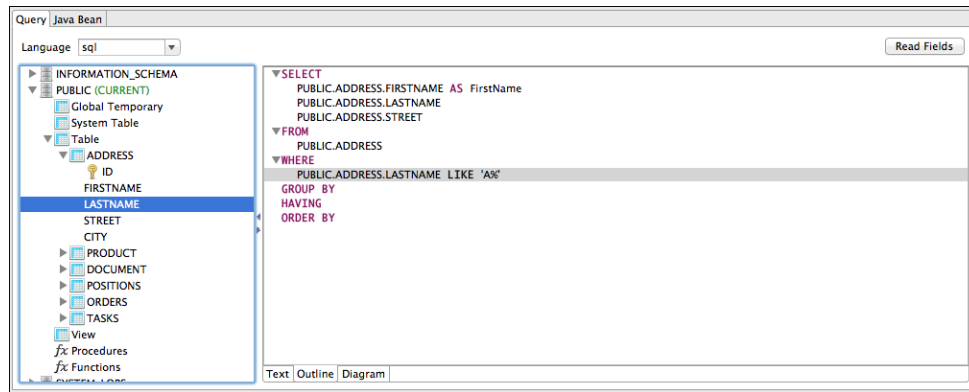


Figure 11-4 Outline View

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 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.

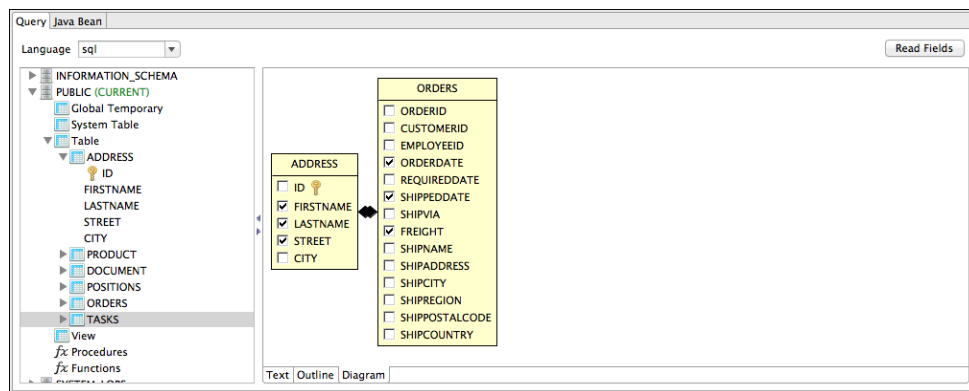


Figure 11-5 Diagram View

11.2.2 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.

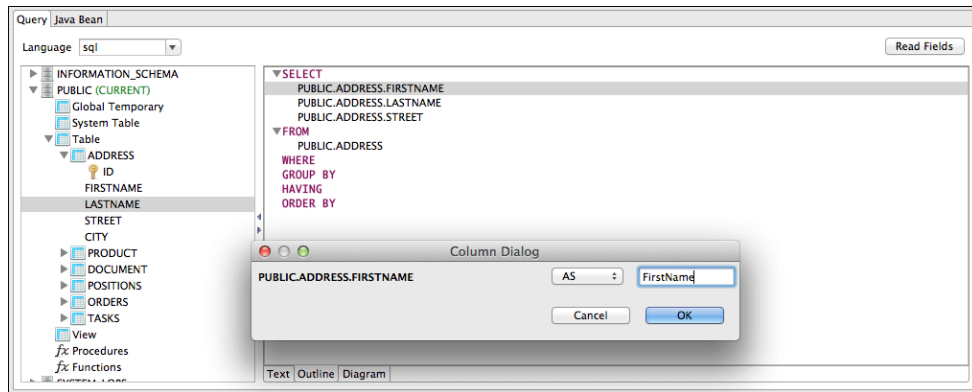


Figure 11-6 Setting an Alias

Aliases are useful when you have several fields with the same name coming from two different tables.

11.2.3 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.

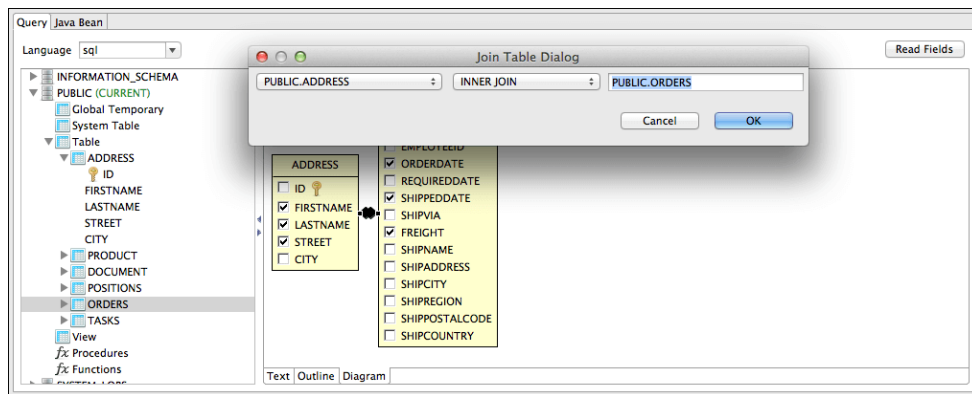


Figure 11-7 Column Dialog

You can also edit joins in the outline view by right-clicking a table name and selecting **Add or Edit Table Join**.

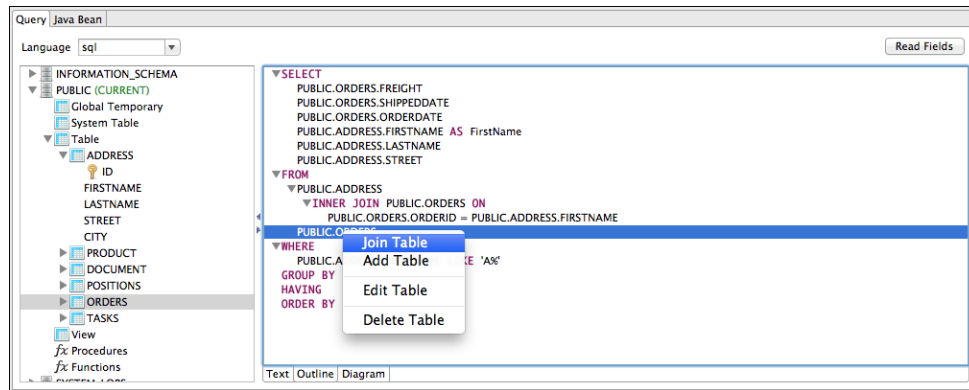


Figure 11-8 Join Table

11.2.4 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.

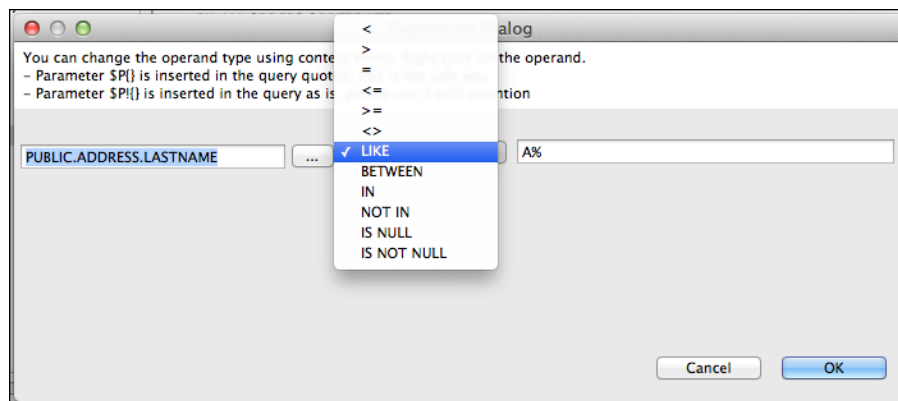


Figure 11-9 Add Condition

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 [8.2, “Expression Operators and Object Methods,” on page 94](#) for more information about the `$X{}` syntax.

11.2.5 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.

11.2.6 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.

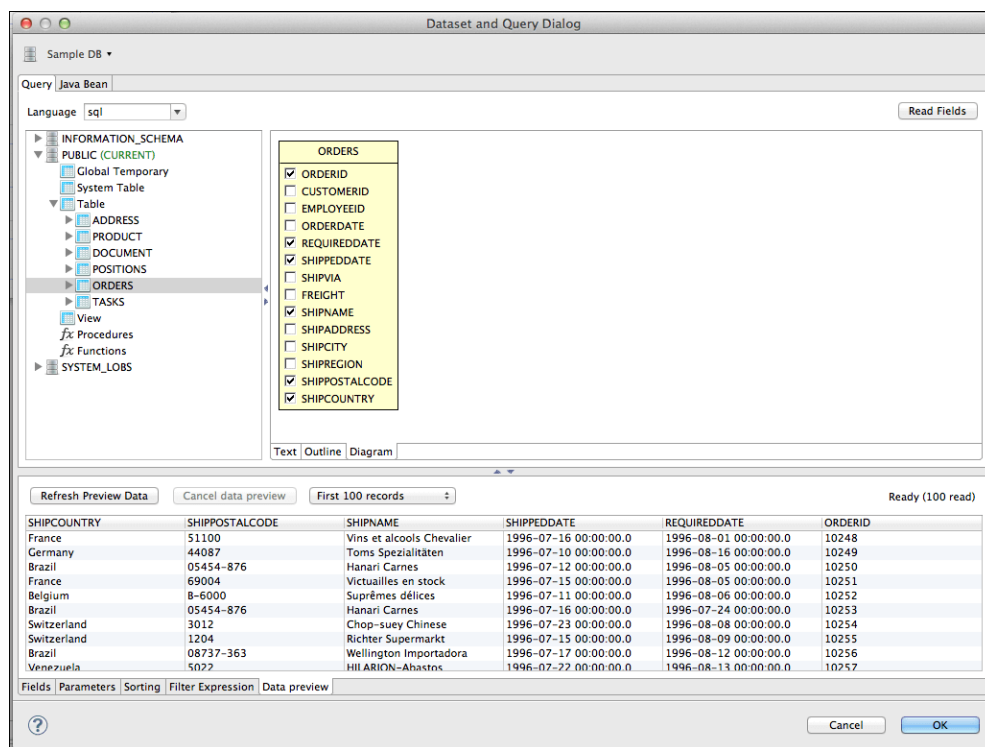


Figure 11-10 Data Preview Tab

CHAPTER 12 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 Server and 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:

- **Connecting to JasperReports Server**
- **Publishing a Report to JasperReports Server**
- **Working with JasperReports Server Templates**
- **Creating and Uploading a Topic for Ad Hoc Views**
- **Managing Repository Objects through Jaspersoft Studio**
- **Creating and Uploading Chart Themes**
- **Working with Domains**
- **Understanding the repo: Syntax**
- **Adding a Date/Time Stamp to Scheduled Output in JasperReports Server**

12.1 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.

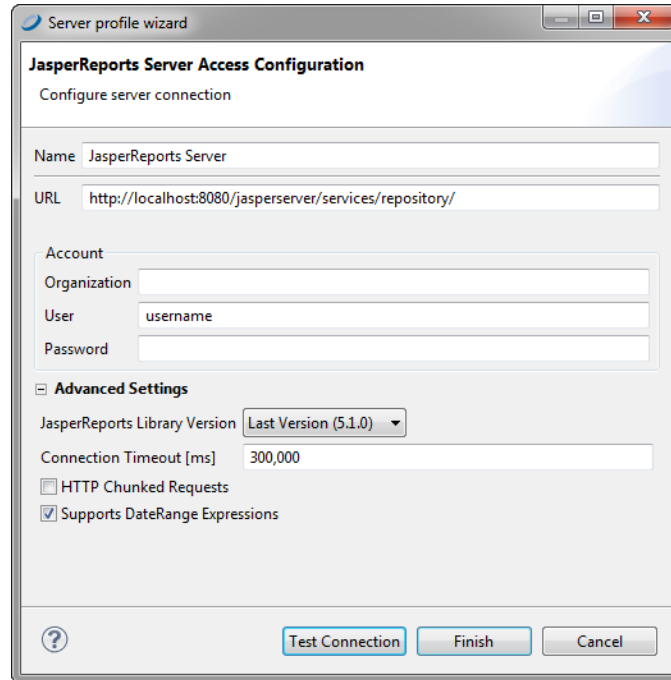


Figure 12-1 Server Profile Wizard

3. Enter the URL, user names and password for your server. If the server hosts multiple organizations, 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**.

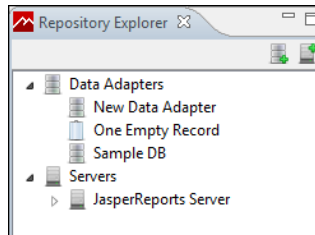


Figure 12-2 Repository Explorer

12.1.1 Advanced Connection Settings

You can configure additional properties for your JasperReports Server connection using the Advanced Settings in the Server profile wizard. The following options are available:

- **JasperReports Library Version** – Sets the version of the JRXML output generated by Jaspersoft Studio. When you connect to a JasperReports Server instance that is earlier than your version of Jaspersoft Studio, use this setting to convert your JRXML to the version used by the server. You can view server information, including the server version, on the info tab.
- **Connection Timeout [ms]** – Sets the connection timeout in milliseconds.
- **HTTP Chunked Requests** – Enables chunked transfer encoding. Disable this option if your firewall blocks incomplete response chunks.
- **Use SOAP protocol only** – Sets communication between Jaspersoft Studio and the server to use the SOAP format. Can't be used with single sign-on.
- **Use Single Sign On** – Lets you connect to a JasperReports Server instance that has been configured to work with the Central Authentication Service (CAS) protocol. See [12.1.2, “Using Single Sign-on with JasperReports Server,” on page 172](#) for more information.
- **Supports DateRange Expressions** – Lets you upload queries with relative dates to JasperReports Server. See [6.3.4, “Relative Dates ,” on page 80](#) for more information.
- **Synchronize DataAdapter Properties** – Synchronizes the `net.sf.jasperreports.data.adapter` property with the Jaspersoft Studio data adapter setting.
- **Format of Attachments** – Sets the message attachment format to MIME or DIME.
- **Workspace Folder** – Lets you designate the folder in the workspace where you want to store files downloaded from this JasperReports Server instance.
- **Locale** – Sets the locale to use for the Jaspersoft Studio connection to the server.
- **Time Zone** – Sets the timezone to use for the Jaspersoft Studio connection to the server.
- **Info tab** – Shows information about the JasperReports Server instance, including version number, edition, license expiration, licensed features, and data and timestamp format.

12.1.2 Using Single Sign-on with JasperReports Server

You can use single sign-on (SSO) to connect to a JasperReports Server instance that has been configured to work with the Central Authentication Service (CAS) protocol. For information about configuring CAS for JasperReports Server, see the *JasperReports Server External Authentication Cookbook*.

Configure JasperReports Server:

1. Before you begin, configure your JasperReports Server instance for CAS, as described in the *JasperReports Server External Authentication Cookbook*.

Add the CAS server to your JasperSoft Studio workspace:

1. In JasperSoft Studio, select **Window > Preferences**.
2. In the Preferences window, navigate to **JasperSoft Studio > JasperSoft Server Setting > Single Sign On Server**.

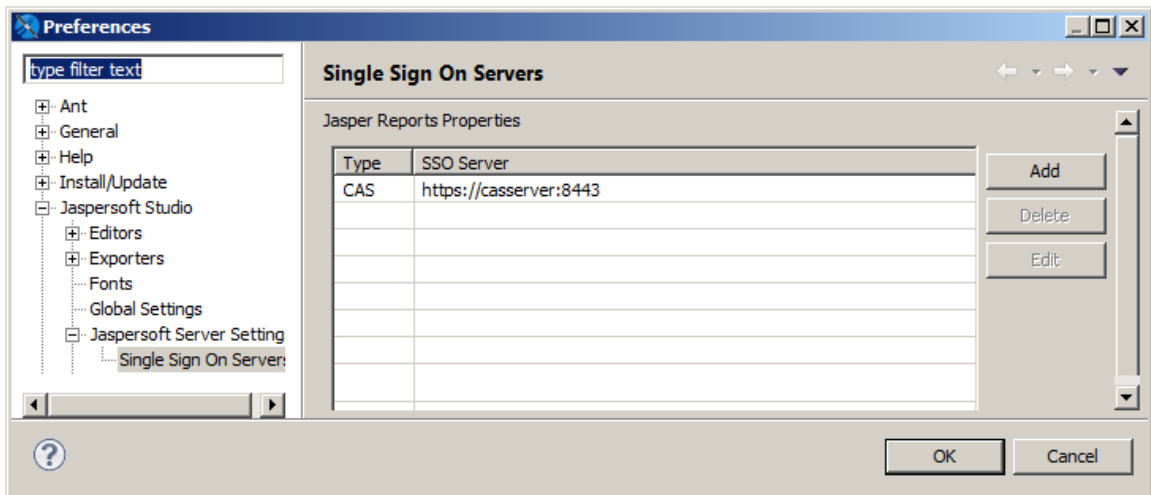


Figure 12-3 Single Sign On Servers in Preferences Dialog

3. In the Single Sign On Servers pane, click Add.
4. Enter the URL of your CAS server along with the username and password you want to use for access.

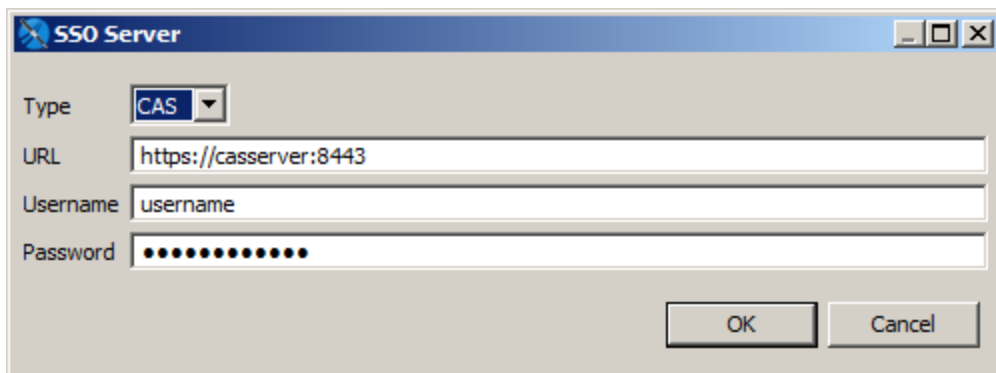

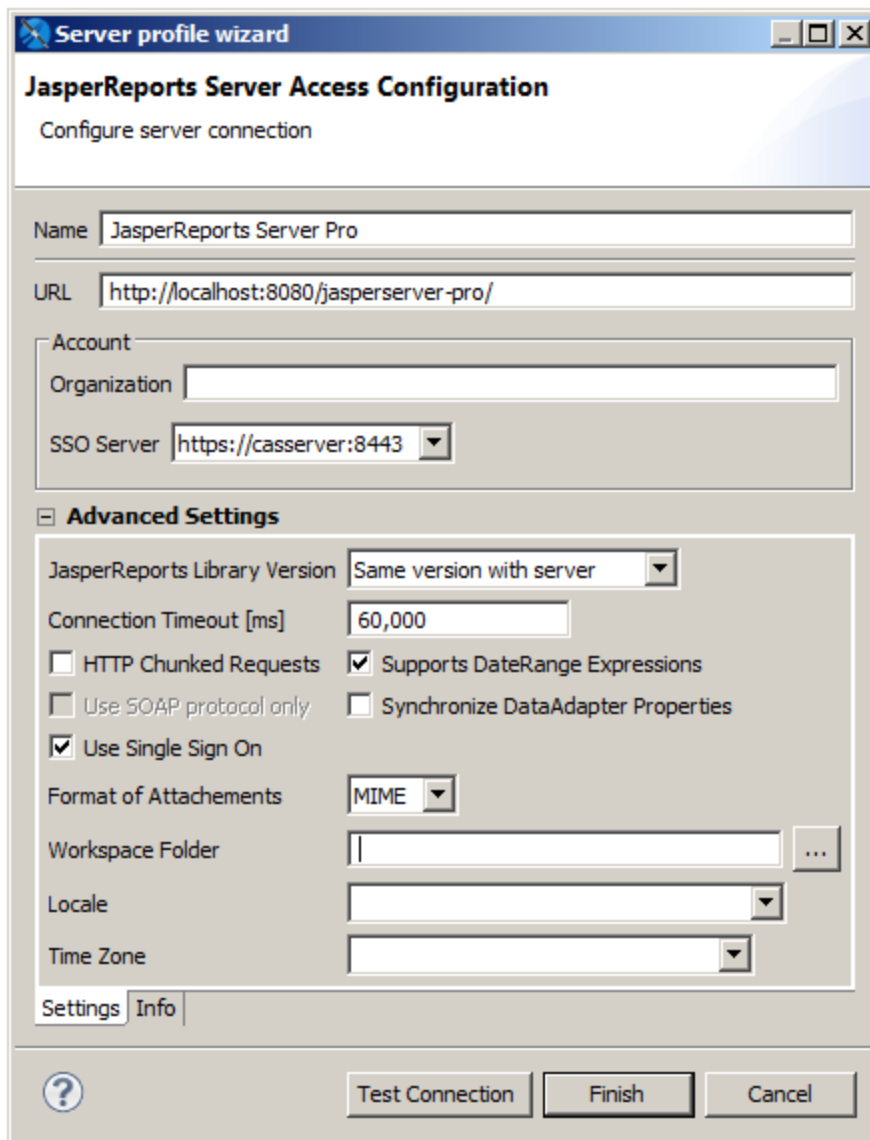


Figure 12-4 SSO Server Settings Dialog

5. Click **OK**.
The CAS server is added to the list of available single sign-on servers.
6. Click **OK**.

Configure your JasperReports Server connection to use CAS:

1. In Jaspersoft Studio, open the **Repository Explorer**, then click the **Create a JasperReports Server Connection** icon .
2. Select **Advanced Settings**.
3. Enable the **Use Single Sign On** option.
The Account section of the Server profile wizard changes.



Server profile wizard

JasperReports Server Access Configuration
Configure server connection

Name: JasperReports Server Pro

URL: http://localhost:8080/jasperserver-pro/

Account

Organization: [Empty text box]

SSO Server: https://casserver:8443

Advanced Settings

JasperReports Library Version: Same version with server

Connection Timeout [ms]: 60,000

HTTP Chunked Requests Supports DateRange Expressions

Use SOAP protocol only Synchronize DataAdapter Properties

Use Single Sign On

Format of Attachements: MIME

Workspace Folder: [Empty text box] ...

Locale: [Empty dropdown menu]

Time Zone: [Empty dropdown menu]

Settings Info

Test Connection Finish Cancel

Figure 12-5 Setting Single Sign-on in the Server Profile Wizard

4. If the server hosts multiple organizations, enter the ID of the organization to which you belong. This can be the root organization (in which case you can leave this entry blank) or another organization.
5. Select the CAS server you want to use for this connection from the **SSO Server** menu.
6. Click **Test Connection** to test the connection. This may take some time, especially the first time you connect.
7. Click **OK** in the confirmation dialog.
8. Click **OK** to create the connection.

12.2 Publishing a Report to JasperReports Server

You can easily publish your reports from Jaspersoft Studio to any JasperReports Server connection. Publishing to the server uploads the JRXML for the report, along with any resources that the report needs such as images and query resources. You must also configure the data source for the report on the server.

12.2.1 Publishing Report Resources

The reports you create in Jaspersoft Studio can have embedded resources, such as images, query resources, and data adapters using `net.sf.jasperreports.data.adapter`. You can use the following columns in Select Resources to publish page of the Report Publishing Wizard to control the resources you upload:

- **Overwrite** – Controls whether or not you overwrite a resource if it already exists. This setting is important when you republish a report that has been published before. Click on the value in this column to display a drop-down menu with the following choices:
 - **Overwrite** – Creates a new resource or overwrites an existing resource with the current version.
 - **Ignore** – Ignores the resource.
 - **Overwrite Only Expression** – Updates the expression for the resource. Enter the new value in the **Expression** column. Does not create or overwrite the resource.
- **Expression** – When **Overwrite** or **Overwrite Only Expression** is selected, lets you choose the expression you want to use. Click on the value in this column and then click ... to open the Expression Editor and edit the expression that specifies the location where the file is saved. By default, resources such as images are published to a repository location, and the uploaded report uses the repo: syntax to refer to the report location.
- **Type** – When **Overwrite** is selected, specifies the existing resource to overwrite. Click on the value in this column to display a drop-down menu with the following choices:
 - **Save to Folder** – Save to a location anywhere on your JasperReports Server instance. When you choose this option, you are prompted to navigate to the location you want.
 - **Link to Resource** – Links to an existing resource on your JasperReports Server instance. When you choose this option, you are prompted to navigate to the resource you want. If you modify a report and the report is set to **Overwrite**, changes overwrite only the path, not the resource.
 - **Use Local Resource** – Save as a resource inside the report unit on JasperReports Server.

12.2.2 Choosing a Data Source for a Published Report

In JasperReports Server, data is usually specified using a JasperReports Server data source. Like a data adapter, a JasperReports Server data source does not contain data; instead it contains the information JasperReports Library needs to construct a `JRDataSource` when the report is run. Choosing a JasperReports Server data source instead

of a data adapter when you publish a report to JasperReports Server lets you take advantage of features specific to JasperReports Server or to use a JNDI connection configured on your application server.

When you upload a report from Jaspersoft Studio to JasperReports Server, you can choose one of the following options:

- **Data Source from Repository** – Use an existing JasperReports Server data source for your report.
- **Local Data Source** – Create a JasperReports Server data source or upload a Jaspersoft Studio data adapter file to use in your report.
- **No Data Source** – Make no changes to the data adapter information in your report.

12.2.2.1 Data Source from Repository


The **Data Source from Repository** option lets you choose an existing data source from the JasperReports Server repository.

To use the **Data Source from Repository** option:

- Make sure the data source is available in JasperReports Server. If you want to create a data source, select **Local Data Source** instead.




If you are using a JDBC or JNDI data source, make sure that your connection uses the same driver as your Jaspersoft Studio data adapter. For example, if you connect to an Oracle database from a commercial edition, you can download and use the native Oracle driver or you can use the TIBCO Oracle JDBC driver that is included with Jaspersoft Studio. If your driver in JasperReports Server does not match the driver used in Jaspersoft Studio, you could see different data in your uploaded report.


- Click  to publish your report and select a JasperReports Server instance and repository location where you want to save the report. Then click OK. If you are prompted to upload resources, select your settings.
- Select **Data Source from Repository** when prompted.
- Click ... and select the correct JasperReports Server data source from the repository.

12.2.2.2 Local Data Source

The **Local Data Source** option lets you create a new data source in JasperReports Server to use with your report, or to upload a data adapter from Jaspersoft Studio to JasperReports Server. To use this option to create a new data source:

- If you want to use a JNDI data source, first set up the JNDI connection for your database in your application server. Make sure to use the same JDBC driver for your Jaspersoft Studio adapter and the JNDI connection in JasperReports Server.
- Click  to publish your report and select a JasperReports Server instance and repository location where you want to save the report. Then click OK. If you are prompted to upload resources, select your settings.
- Select **Local Data Source** when prompted.
- Click ... to open the Add Resource wizard.
- Select the type of data source you want to create, for example, Datasource JDBC, and click **Next**.
- On the Resource Editor page, enter a name and unique ID for the data source. You can also enter an optional description. Then click **Next**.
- On the next page, manually fill in the required information for the data source you want to create.
- Click **Finish** to create the data source and use it in your uploaded report.

For some types of data sources, such as JDBC, JNDI, and bean data sources, you can publish a data adapter from Jaspersoft Studio and set it as the data source for the report:

- Make sure that the data adapter you want to upload is saved locally as an xml file in the same project as your report. See [10.1.2, “Importing and Exporting Data Adapters,” on page 121](#) for information about exporting a global data adapter to your project; see [10.1.3, “Copying a Data Adapter,” on page 122](#) for information about copying a data adapter from one project to another.
- Click  to publish your report. Select a JasperReports Server instance and repository location where you want to save the report. Then click **OK**.
- If you are prompted to upload resources, select your resource settings and click **OK**.
- Select **Local Data Source** when prompted.
- Click ... to open the Add Resource wizard.
- Select the type of data source you want to create and click **Next**. Only some data source types can be imported from data adapters in Jaspersoft Studio, such as Datasource Bean, Datasource JDBC, and Datasource JNDI.
- On the Resource Editor page, click **Import from Jaspersoft Studio**. If this button is not available, you can't import the data source type you selected.
- The **Import** dialog shows a list of local and global adapters. Make sure to select a local adapter, which will include the name of a file. Click **OK** then **Finish** to select the adapter.
- Click **Finish** to publish the report and selected data adapter to the repository.



If you are using a custom data adapter or any other adapter that uses one or more jars that aren't included in JasperReports Server, add the jar(s) to a location on your server classpath.

12.2.2.3 No Data Source


Use **No Data Source** when you have configured `net.sf.jasperreports.data.adapter` in your JRXML, as described in [10.2.3, “Default Data Adapter ,” on page 123](#). Setting `net.sf.jasperreports.data.adapter` lets you use multiple data adapters in the same report, for example, using a different data adapter for a subreport.

To use this option:

- Set the default data adapter(s) for the datasets and subdatasets used in your report, as described in [10.2.3, “Default Data Adapter ,” on page 123](#).
- Follow these guidelines when setting default data adapters in a report that you want to publish:
 - Do not use global data adapters. The default data adapter must reference a local file in the same project as your report, or a data adapter already in your repository.
 - Where possible, use inheritance to reduce the number of times you actually set the default data adapter. If your report uses a specific adapter multiple times, try to structure the report so that the data adapter is set for a single dataset and other datasets inherit it. For example, if you have a crosstab and a table that use the same data adapter, you could create a subreport that contains the table and crosstab. Then if you set your data adapter as the default for the subreport, the table and crosstab inherit this adapter. Reusing the data adapter improves performance when you run the report.
- Publish your report and select **No Data Source** when prompted.
- When you publish the report, the default data adapters are uploaded to the repository.

12.2.3 Example of Publishing a Report

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.

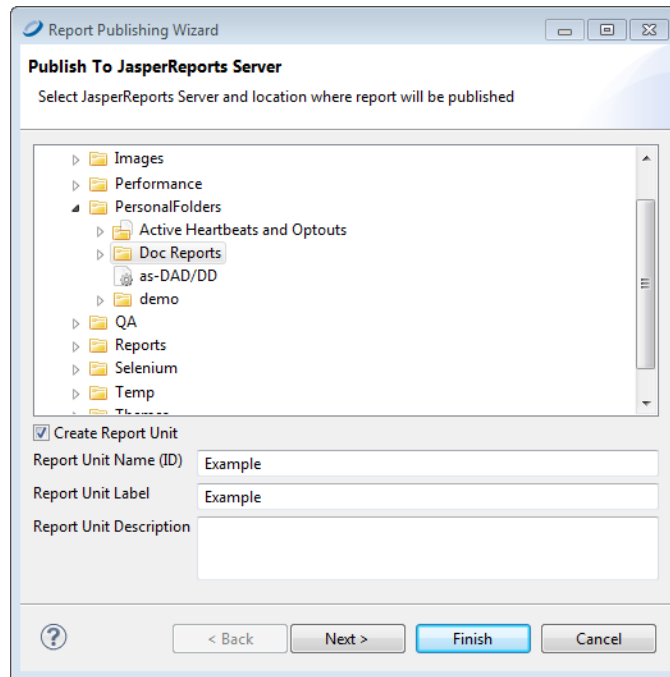


Figure 12-6 Report Publishing Wizard

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. This window displays any resources required by your report, such as images, query resources, and embedded data adapters.

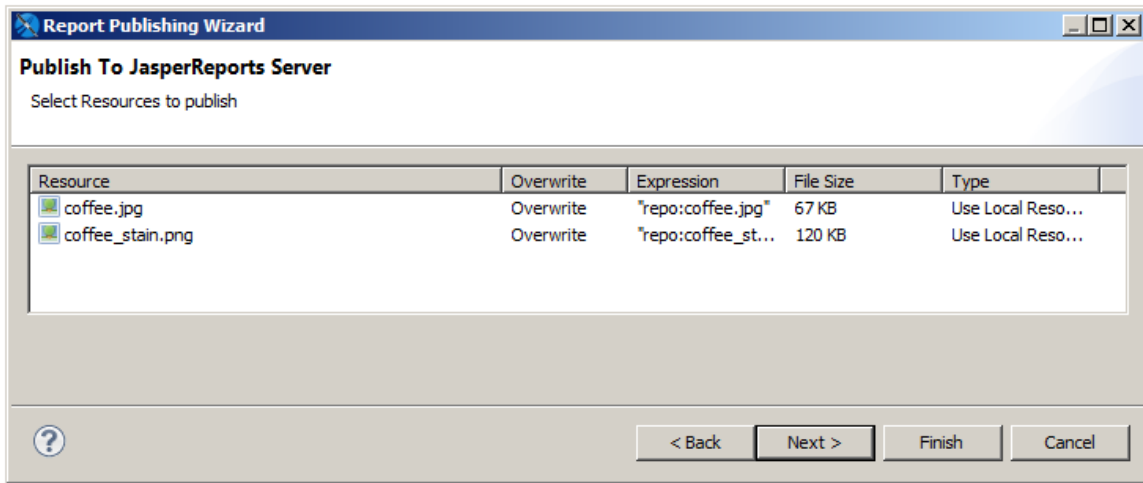


Figure 12-7 Select Resources

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.

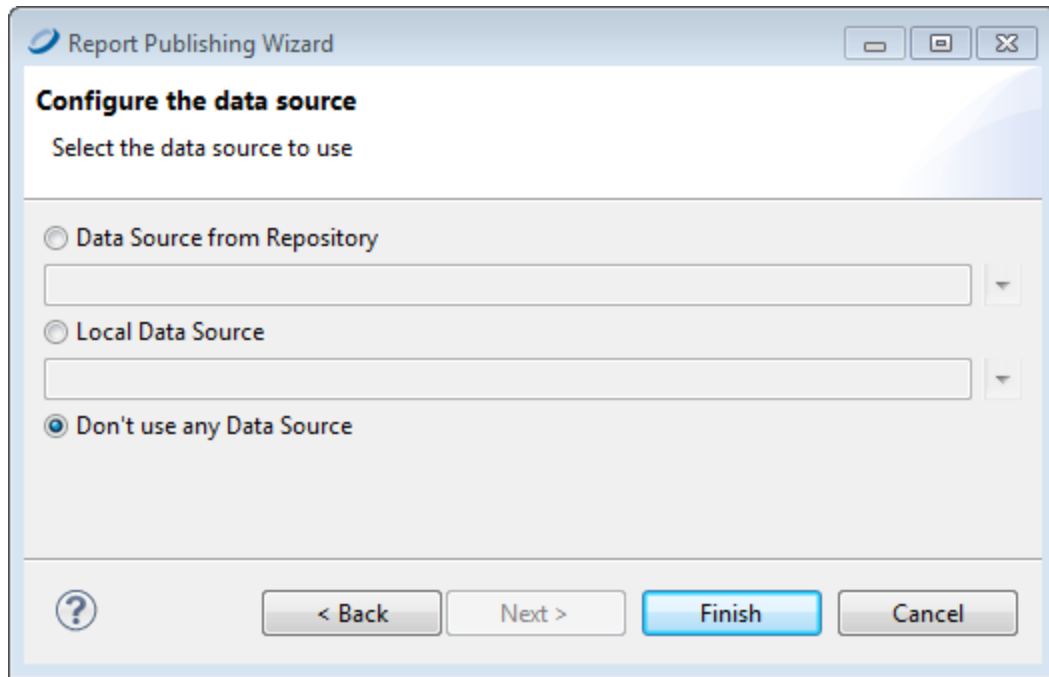


Figure 12-8 Configure Data Source

7. Select a data source configuration. See [12.2.2, “Choosing a Data Source for a Published Report,” on page 174](#) for more information.
8. Click **Finish**. The report is uploaded to the server. If there are no errors, an appropriate message is shown.

12.3 Working with JasperReports Server Templates



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.

JasperReports Server includes several templates that affect the layout of your reports. You can add custom templates to your JasperReports Server instance by uploading a JRXML file to a Templates directory. In addition to font and color choice, templates can contain images such as logos. In a JasperReports Server 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.



JasperReports Server templates are different from report templates in Jaspersoft Studio. See [Chapter 20, "Report Templates,"](#) on page 313 for more information.

12.3.1 Creating a Custom JasperReports Server Template

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.

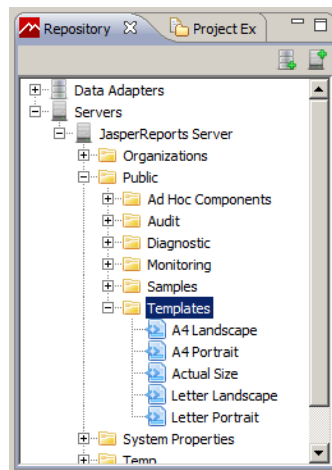


Figure 12-9 Accessing the Templates directory from Jaspersoft Studio

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

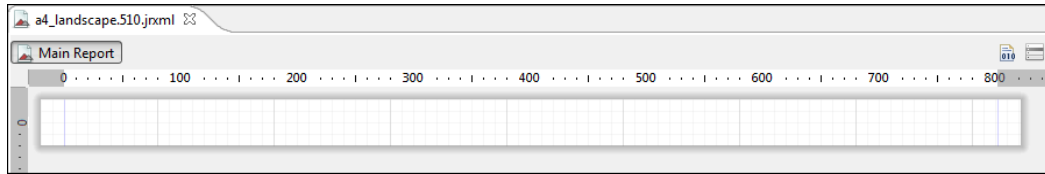


Figure 12-10 Default A4 landscape template

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 view.

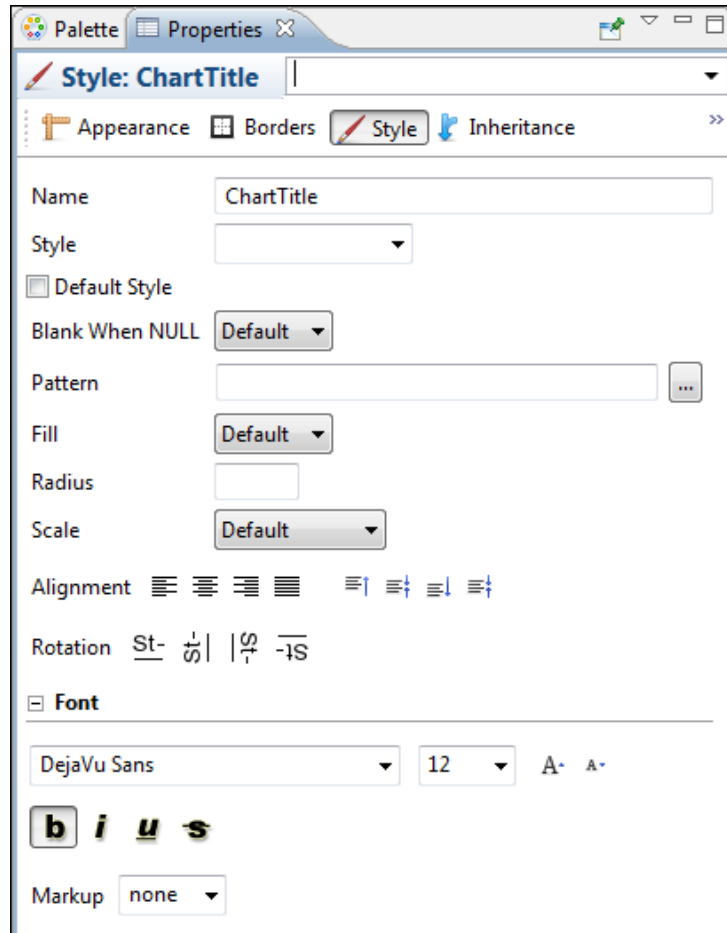


Figure 12-11 Style tab in Properties view

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.

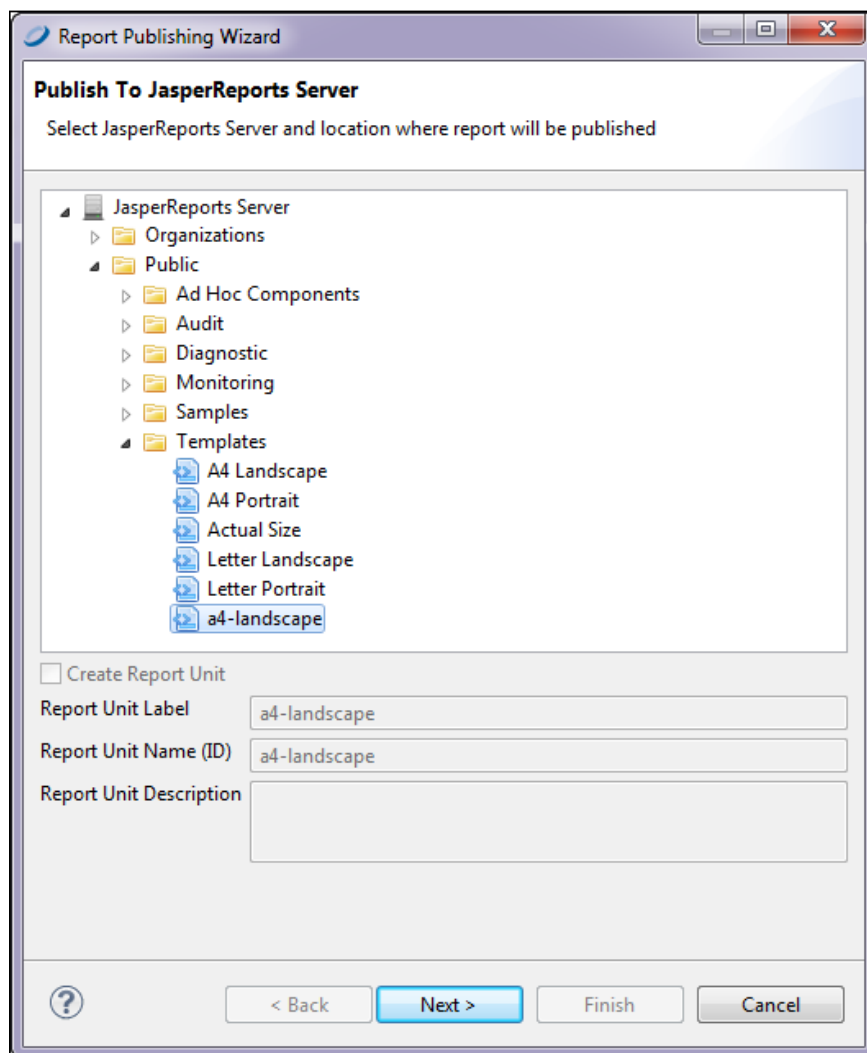


Figure 12-12 Report Publishing wizard

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

12.3.2 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 [14.4, “Chart Themes,” on page 218](#).

The Inheritance tab in the Properties view 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.

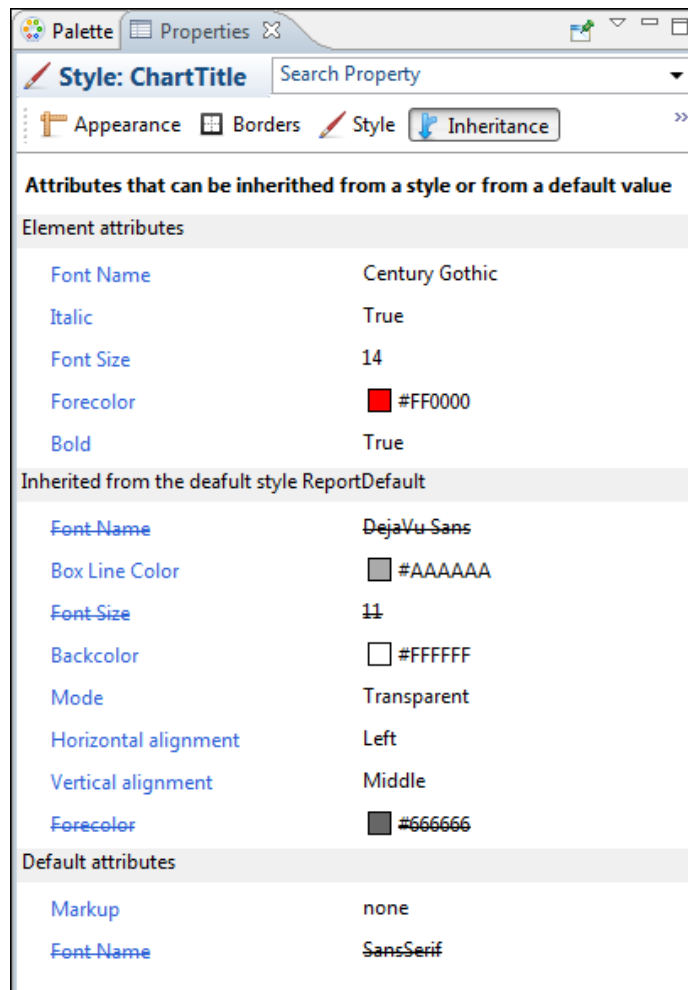


Figure 12-13 Inheritance tab

12.4 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 Server repository.

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.

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 [12.1, "Connecting to JasperReports Server," on page 170](#).
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**.

- 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:

- Log in to JasperReports Server.
- Click **Create > Ad Hoc View**.
- On the topics tab of the Data Chooser, open the Topics folder. For example, navigate to **Organizations > Topics**.
- Click the Topic you created above and click **Table, Chart, or Crosstab**.
- 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 [Chapter 8, "Expressions," on page 93](#).

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

12.5 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](#)

12.5.1 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 (JRXTX), 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 [12.2, “Publishing a Report to JasperReports Server,” on page 174](#)

- 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.

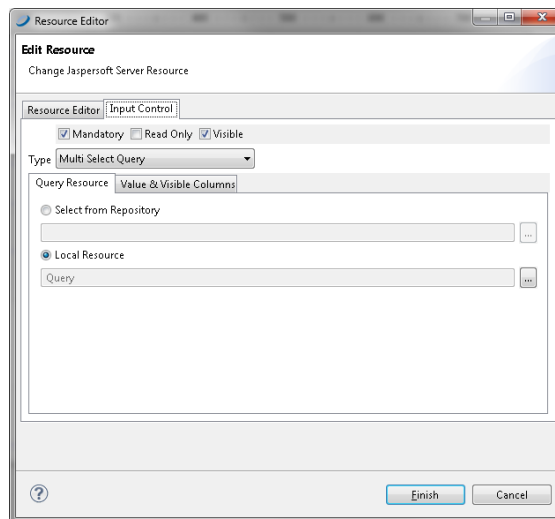




Figure 12-14 Properties of an Input Control Resource

12.5.2 Running a Report

Connect to JasperReports Server to test report changes you make in Jaspersoft Studio. See [12.1, “Connecting to JasperReports Server,” on page 170](#).

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.

12.5.3 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 Studio operates 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**.

12.6 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.

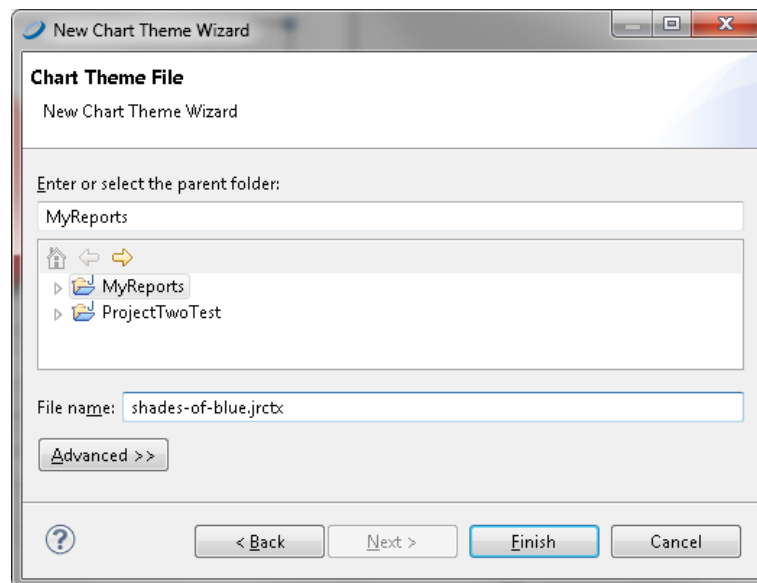


Figure 12-15 New Chart Theme in Jaspersoft Studio

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 view, select each category and review the available options in the Properties view.
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 view and choose Bottom from the Position drop-down to move the title beneath the chart.
6. Click a chart type in the Preview tab to zoom in to examine the effects of your changes more closely. Click again to zoom out.

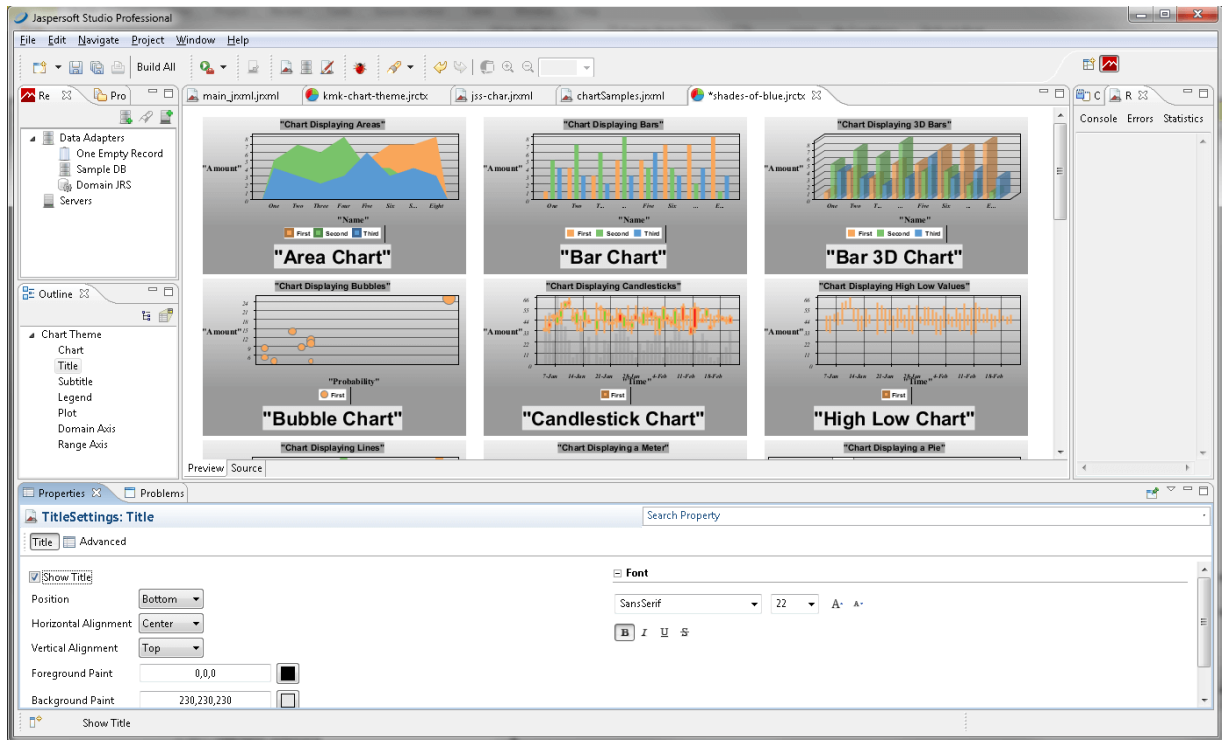


Figure 12-16 A Chart Theme Edited in Jaspersoft Studio

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.

To export your theme as a JAR File:

1. Select your chart theme and click the Export Chart Theme jar icon on the toolbar. A Save As dialog opens.

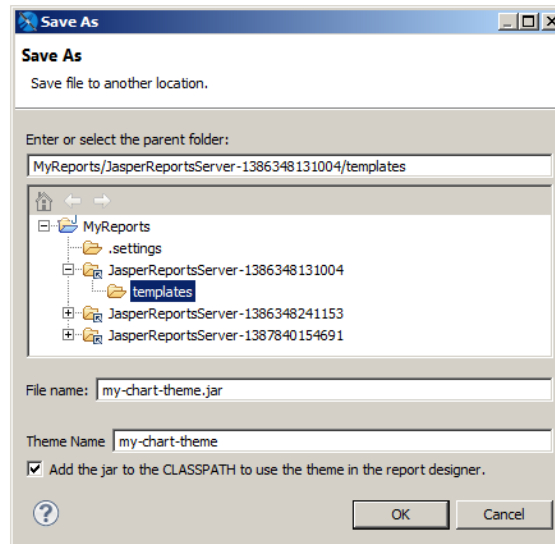


Figure 12-17 Exporting a Chart Theme

2. Choose the location where you want to save your JAR. To upload to an instance of JasperReports Server, select your server instance and then select the Templates directory. To create a jar on your current system, select a location on your hard drive.
3. Enter a file name and theme name for your theme.
4. If you want to use the theme to design reports on your current system, save to a location on your hard drive and select **Add the jar to the CLASSPATH to use the theme in the report designer**.
5. Click **OK**. The chart theme is exported as a JAR.



Once you have uploaded a theme to JasperReports Server, you can use the `repo:` syntax in your reports to specify this JAR as your chart theme. The theme can be used at the report or server level in JasperReports Server. For more information, refer to the *JasperReports Server Administrator Guide*.

12.7 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 [12.1, “Connecting to JasperReports Server,” on page 170](#).

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 Studio compiles 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 [12.2, “Publishing a Report to JasperReports Server,” on page 174](#).

12.8 Understanding the repo: Syntax

In some cases, you may see the `repo:` syntax used to refer to a location in a JasperReports Server repository. The `repo:` syntax can be used to refer to any type of resource, such as reports, images, data sources, and input controls. The `repo:` syntax can be used in two ways:

- `repo:` used without a path – A resource of a report. This syntax is generated when a resource is selected when a report is published to the repository. For example, if you upload an image as a resource of a report, you might see JRXML like this:

```
<imageExpression class="java.lang.String">
  <![CDATA["repo:AllAccounts_Res2"]]>
</imageExpression>
```

When you publish a report to JasperReports Server and upload your resources, Jaspersoft Studio updates the JRXML in the published report to use the `repo:` syntax to refer to the uploaded resources in the repository.

- `repo:` used with a path – A resource saved somewhere in the repository. For example, to refer to an image in the repository, you might see JRXML like this:

```
<imageExpression class="java.lang.String">
  <![CDATA["repo:/Images/myimage.jpg"]]>
</imageExpression>
```

In a multi-organization deployment of JasperReports Server, the path used in a `repo:` expression is relative to the organization of the current user. For example, if User1 in Organization_1 accesses the report,

JasperReports Server looks for myimage.jpg in the Images folder of Organization_1. If User2 in Organization_2 accesses the report, JasperReports Server looks for myimage.jpg in Images folder of Organization_2.

To set the location globally, use /root as the base of your repo: path. For example, to set the path to an image in the Images folder in Organization1, use the following:

```
/root/Organizations/Organization_1/Images/myimage.jpg
```

12.9 Adding a Date/Time Stamp to Scheduled Output in JasperReports Server

When you add a parameter named `_ScheduledTime` to a JRXML report design in Jaspersoft Studio, and then schedule the report to run in JasperReports Server, the output includes a date/time stamp showing when the report ran. The following procedure describes how to set up and use this parameter:

To display the date/time that the report ran:

1. Launch Jaspersoft Studio, and open an existing report.
2. In the Outline view, right-click Parameters, and select **Create Parameter**.
3. Rename the parameter `_ScheduledTime`.

The new parameter appears in the outline view.

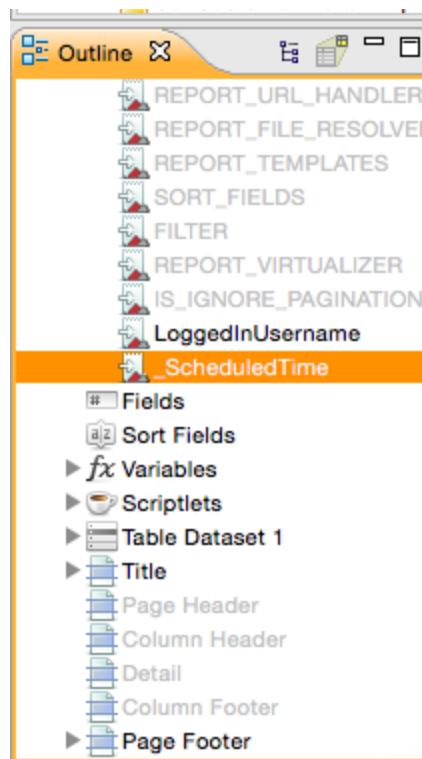


Figure 12-18 `_ScheduledTime` Parameter in Outline View

4. Set the following parameter properties:
 - Class = java.util.Date
 - Is for Prompting = unchecked

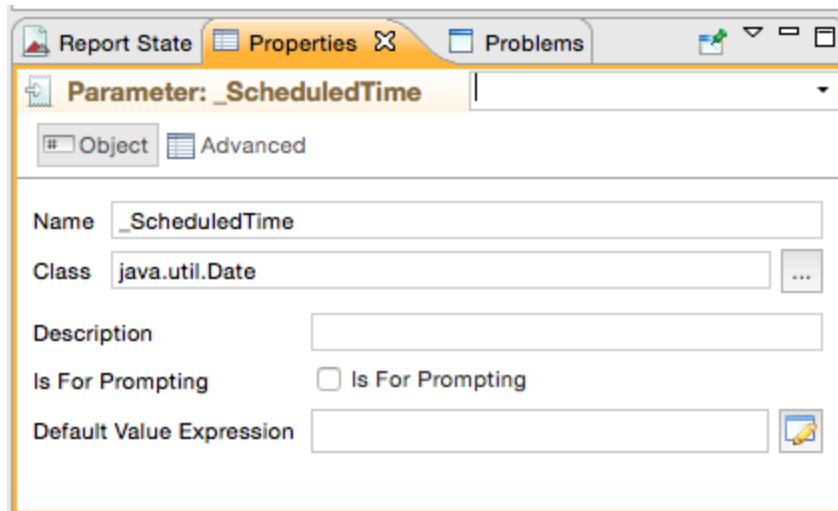


Figure 12-19 `_ScheduledTime` Parameter Properties

5. Drag the `_ScheduledTime` element from the Outline View to a valid location, such as the Title Band, in the Designer:

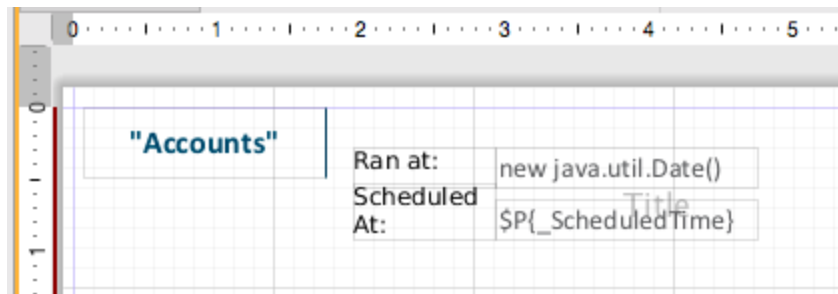



Figure 12-20 Report Design Includes the `_ScheduledTime` Parameter Element

6. Now you can set other properties, such as the text color of the date/time stamp. In Properties, check **Blank when Null** to prevent the word null from appearing on the report when it runs unscheduled.
7. Compile the report, and upload it to JasperReports Server. For more information about uploading reports to JasperReports Server, see [“Accessing JasperReports Server from Jaspersoft Studio” on page 169](#).
8. In the server, schedule the report to run immediately.
9. Open the output file.

Accounts | Ran at: Jun 2, 2015 3:48:13 PM
Scheduled At: Jun 2, 2015 3:48:00 PM



#	Name	Phone	Address
1 Burnaby, Canada			
1	Souza-Quick Construction Group	699-555-8963	3752 Hamilton Ct
2	Suffin-Cleary Engineering, Ltd	367-555-1366	3753 Forest Way
3	L & E Gillmore Communications Partners	293-555-9531	1983 Santa Cruz

Figure 12-21 Output Showing the Scheduled Time the Report Actually Ran

The date and time the report actually ran appears in the output, as well as the scheduled time. In the screenshot above there was a 13-second delay between the scheduled start time and the actual run time.

CHAPTER 13 WORKING WITH 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:

- **Creating a Table**
- **Editing a Table**
- **Table Structure**
- **Working with Columns**

13.1 Creating a Table

To create a new table:

- To have Jaspersoft Studio autosize your table, drag the Table element  from the Elements palette into any band of the report.
- To manually set the size of the table when you insert it, click on the Table element , but do not drag.

The cursor changes  to show that an element is selected. Click and drag in the report editing area to size and place the element. When you size the table when you first insert it, the columns fill the whole table.

Once you have placed your table in your report, use the Table Wizard to choose a new or existing dataset for your table.

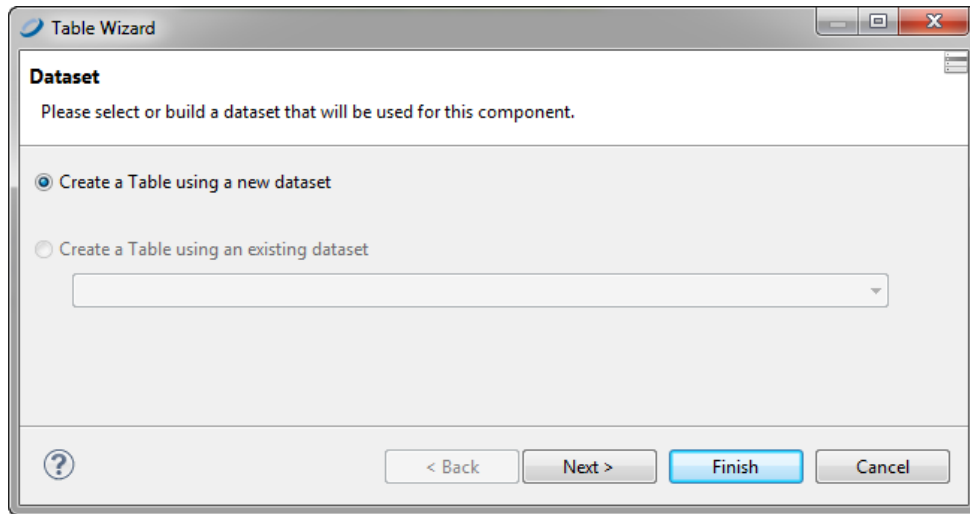


Figure 13-1 Table Wizard - New Table

To create a new dataset for your table:

1. Select **Create a Table from a new dataset** and click **Next**. The Dataset window appears.

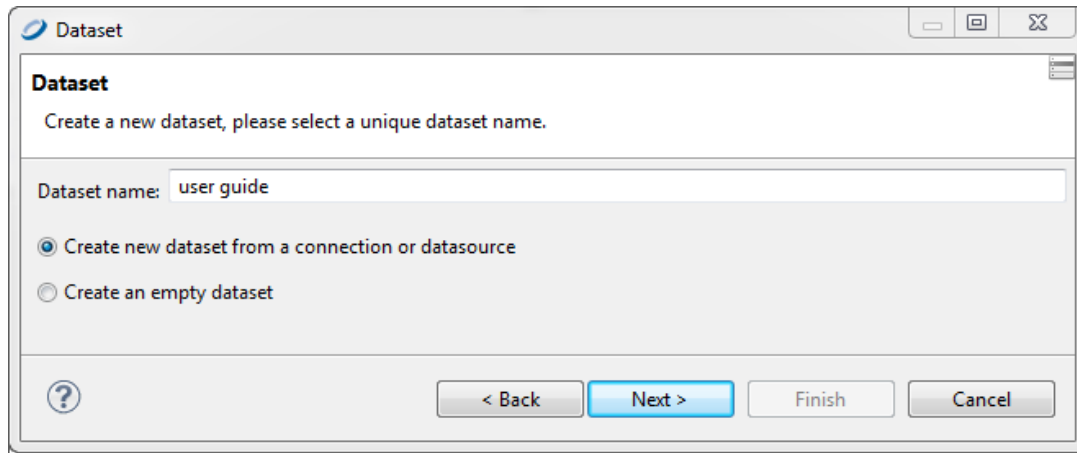


Figure 13-2 Table Wizard - Dataset

2. Name your dataset and select your option: **Create new dataset from a connection or data source** or **Create an empty dataset**. For this example, choose the first option and click **Next**. You'll be prompted to select a data source and design query.

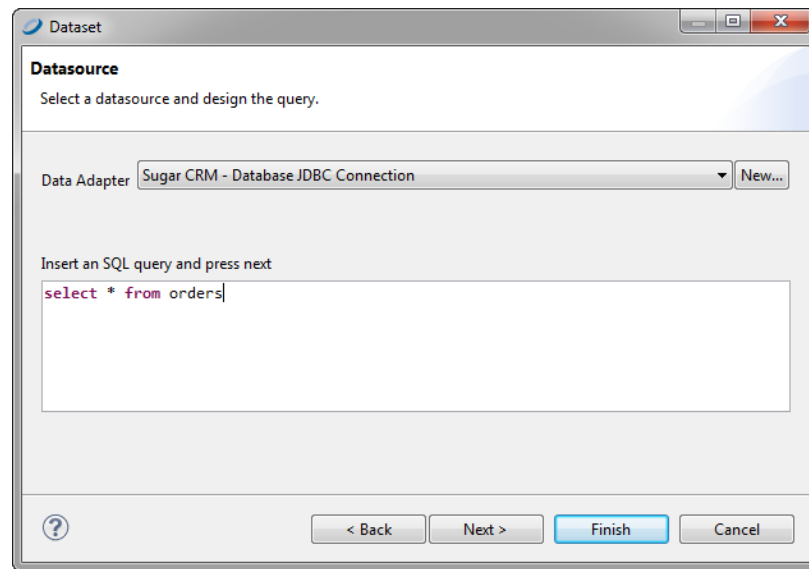


Figure 13-3 Table Wizard - Dataset Datasource

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.

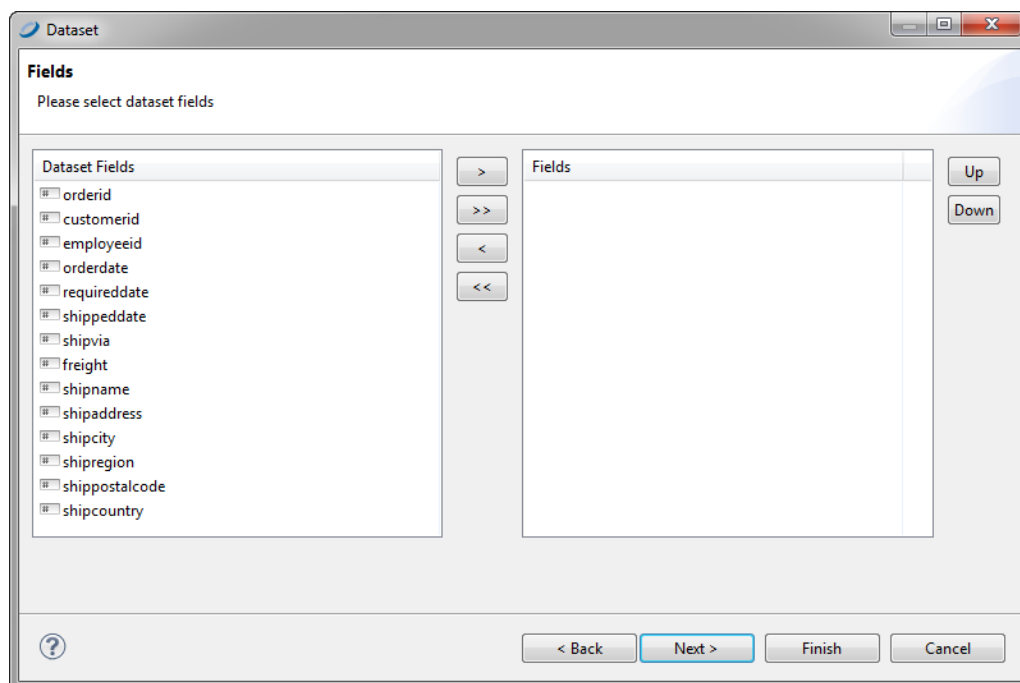


Figure 13-4 Table Wizard - Dataset Fields

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.

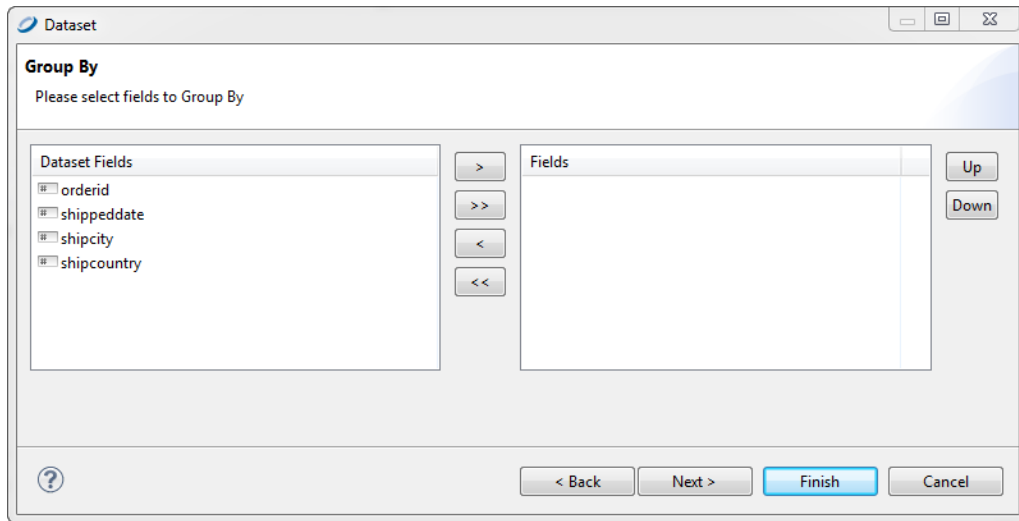


Figure 13-5 Table Wizard - Dataset > Group By

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.

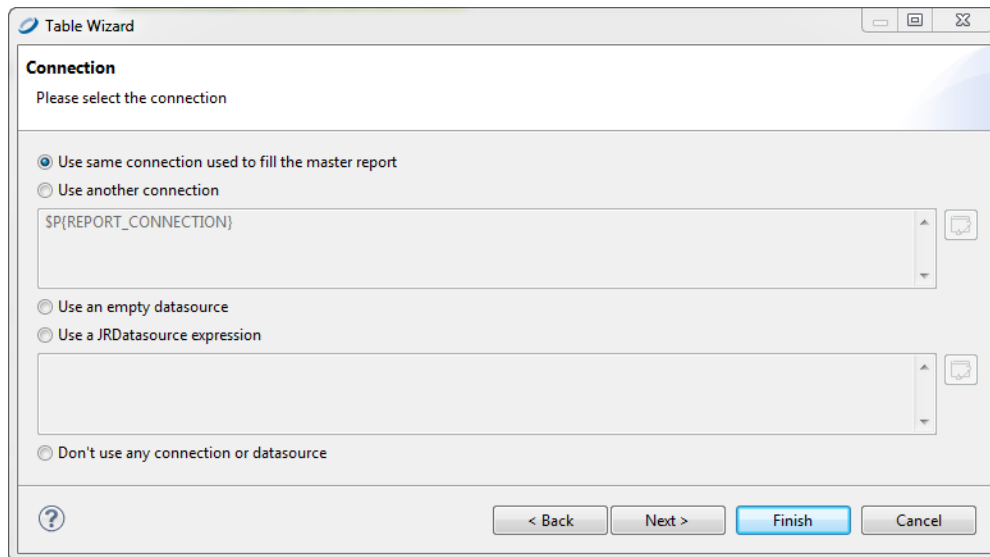


Figure 13-6 Table Wizard - Connection

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 data source
 - Use a JRDataSource expression (you'll enter a JRDataSource expression)
 - Don't use any data source or connection
7. Click **Next**. You'll be prompted to choose the fields for produce table columns.

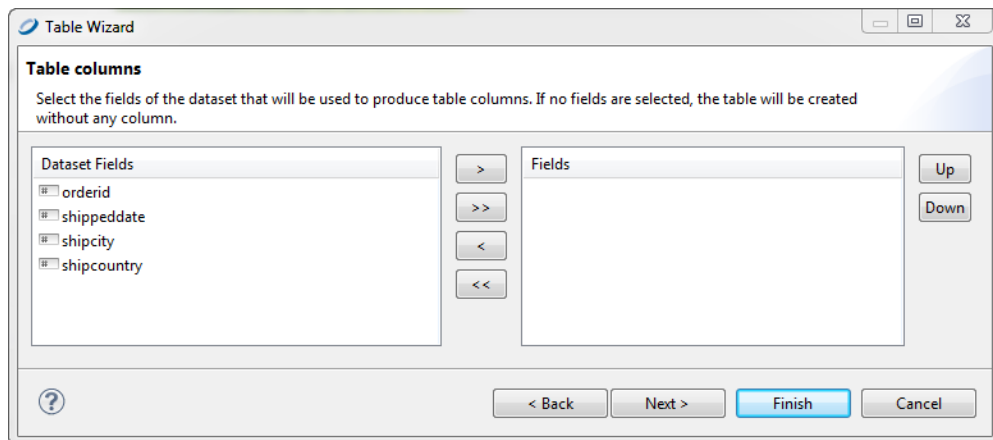


Figure 13-7 Table Wizard - Table Columns

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.

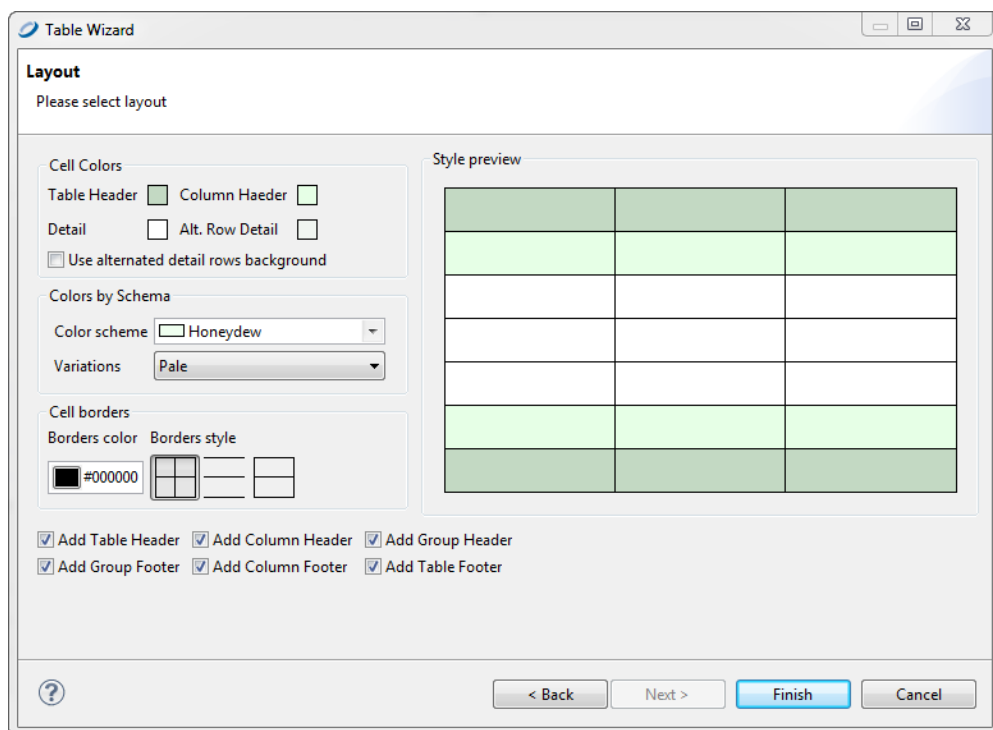


Figure 13-8 Table Wizard - Layout

9. Select the layout for your table, and click **Finish**. The table appears where you dragged the table element in your report.

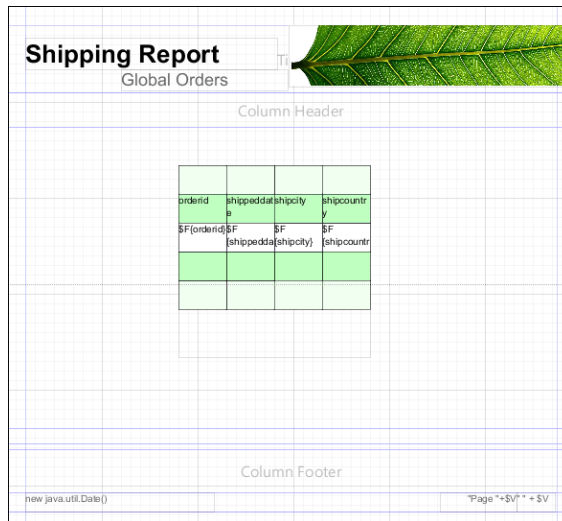


Figure 13-9 Report Containing a Table

To use an existing dataset when creating a table:

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.

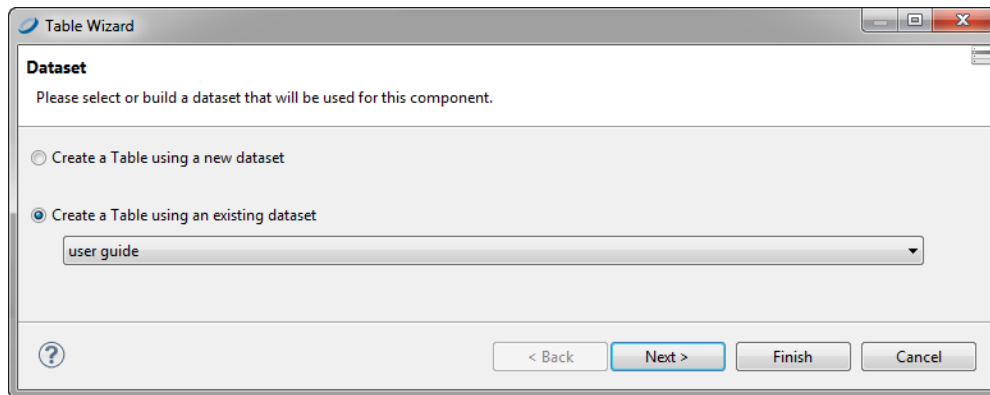


Figure 13-10 Table Wizard - Dataset

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.


13.2 Editing a Table

13.2.1 Editing Table Properties

You can edit the following on the Tables tab in the Properties view:

- **Name** – Enter a name for the table; the name appears in the Outline view.
- **Fit columns to table element** – Select this to have the columns automatically stretch or shrink to fit the table width.
- **Resize the columns taking the space from the next one** – Select this to configure the table so that resizing a column by moving its border means one column grows wider and the other narrower.

13.2.2 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.

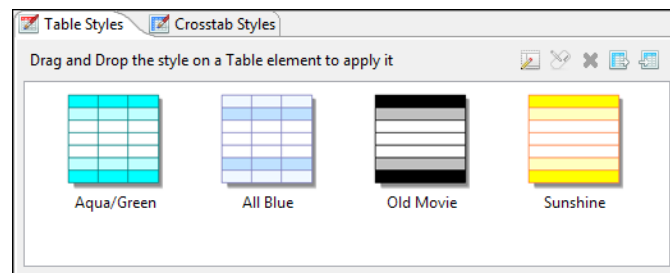


Figure 13-11 Table Styles Tab

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 tab, right-click the table and choose **Change Table Style**.

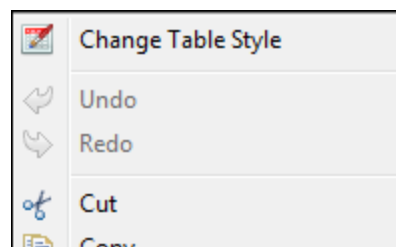


Figure 13-12 Change Table Style

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**.

13.2.3 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 inside the Design tab for your report.

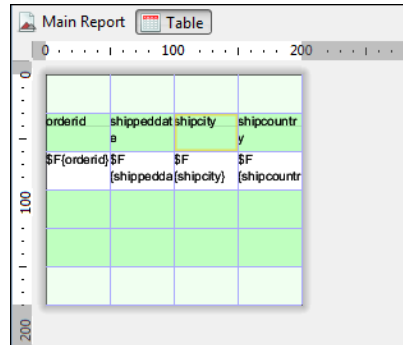


Figure 13-13 Table Editing Tab

- To edit the content and style of a cell, click the cell. Switch to the **Properties** view where you can edit location, size, color, style, and print details for the cell.

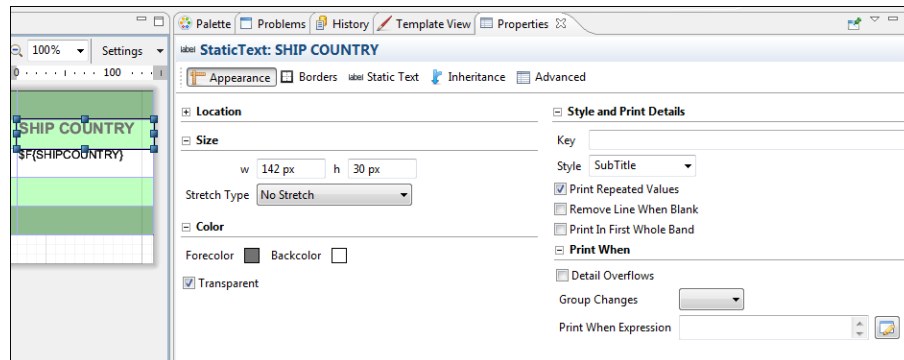


Figure 13-14 Cell with Properties View

- To edit the size and position of a cell, or copy or delete it, right-click on the cell and select an action from the context menu.

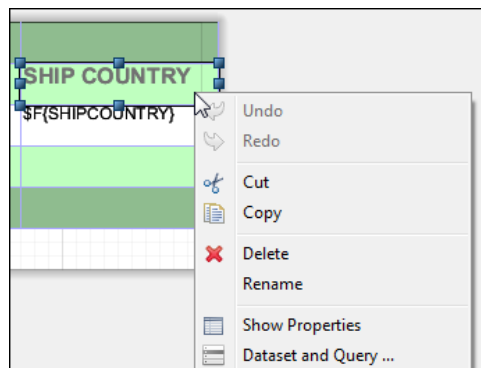


Figure 13-15 Cell Right-Click Menu

- Use Shift-click to select all cells in a row.
- See **“Working with Columns” on page 206** for information about working with columns.

The following figure is the table created in **“Creating a Table” on page 195**, after formatting and editing:

Shipping Report
Global Orders



ORDERID	ORDER DATE	SHIP CITY	SHIP COUNTRY
10248	7/4/96 12:00 AM	Reims	France
10249	7/5/96 12:00 AM	Munster	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 13-16 Formatted Table

13.2.4 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.

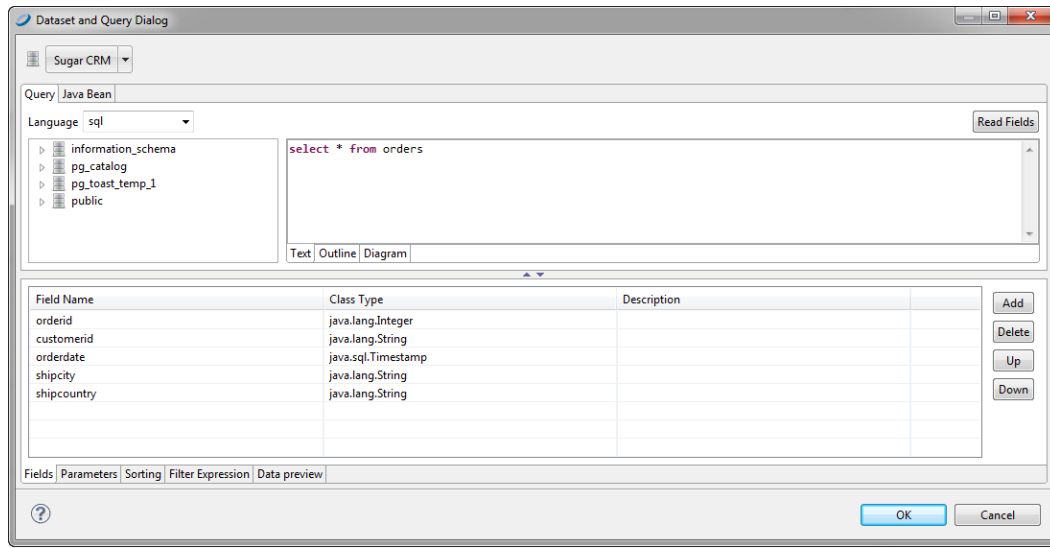


Figure 13-17 Dataset and Query Dialog

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.

13.2.5 Editing Table Source

You can also edit tables using the **Source** tab. In the source the tags are labeled as follows:

- `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.

13.3 Table Structure

Tables consist of cells and columns. This section provides more information about working with each of these elements.

13.3.1 Table Elements

A table must have at least one column, but it can have any number. A set of columns can be collected into a column group with a heading that spans several all those columns.

Each table is divided into sections similar to 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.

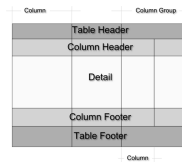


Figure 13-18 Table Structure

In the **Outline** view, table sections are shown as child nodes of the table element node.

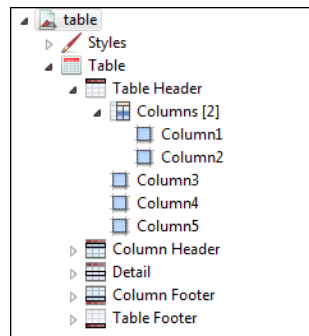


Figure 13-19 Table in Outline View

In the Design tab, 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 tab.

13.3.2 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 206**).

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.

13.4 Working with Columns

There are number of tools that can help you lay out the columns of a table.

13.4.1 Table Properties for Managing Columns

Use the following settings on the Table tab in Properties view to control column behavior:

- To expand the columns to fit the table width, select **Fit columns to table element**.
- To configure the table so that resizing a column by moving the border means one column grows wider and the other narrower, select **Resize the columns taking the space from the next one**.

13.4.2 Working with Individual Columns

To edit individual columns, double-click your table. The table opens in a separate tab inside the Design tab for your report. Here you can do the following:

- To edit cell content, double-click on the column and enter the new content in the editor. See [13.2.3, “Editing Cell Contents,” on page 202](#) for more information.
- To resize a cell, click on a cell with content to display its handles, then click and drag on any handle.
- To resize a column or row, click in an empty cell in the column. The selected row and column are outlined. Drag the outline to resize.
- To add and delete columns, click in the column header or footer. The selected row and column are outlined. Select an option from the action menu. By default, when Jaspersoft Studio adds a column to a table, the new column inherits the properties of the other columns.

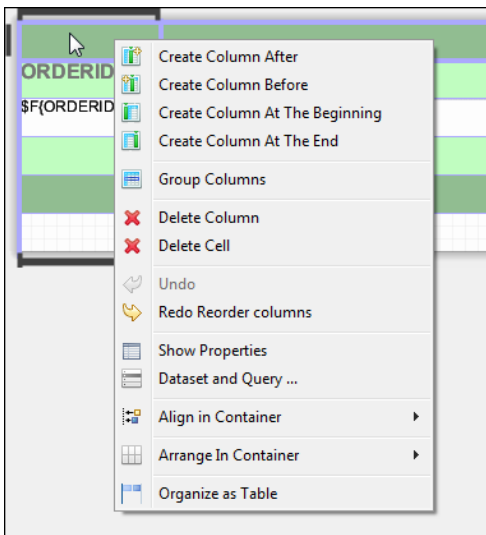


Figure 13-20 Column Context Menu

- Table cells are containers that can include other elements. To set a layout for the cell contents, click in the column header or footer and select Arrange in Container from the action menu, then select a layout option. See 4.2.4, “Positioning Elements in Containers,” on page 41
- 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 outline view.

13.4.3 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.

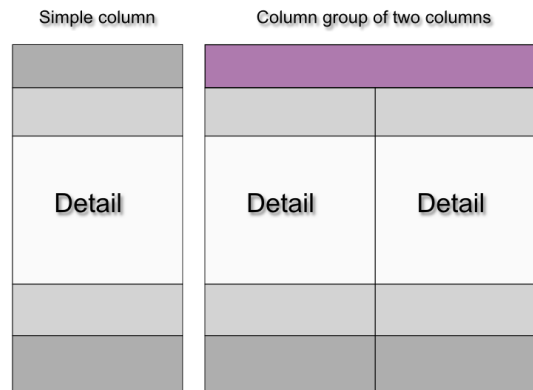


Figure 13-21 Simple Column vs. Column Group

A column group acts as a single unit 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 13-22](#), 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.

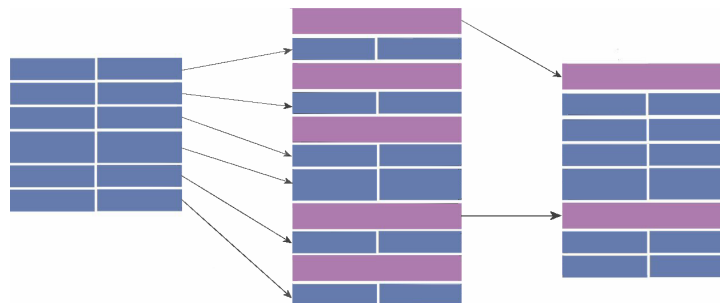


Figure 13-22 Group headings

CHAPTER 14 WORKING WITH CHARTS

The Chart element in the Palette view lets you easily add charts to your reports. In Jaspersoft Studio you can render charts inside a report two different ways. You can use 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.

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

This chapter has the following sections:

- **Creating a Simple Chart**
- **Setting Chart Properties**
- **Spider Charts**
- **Chart Themes**

14.1 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.

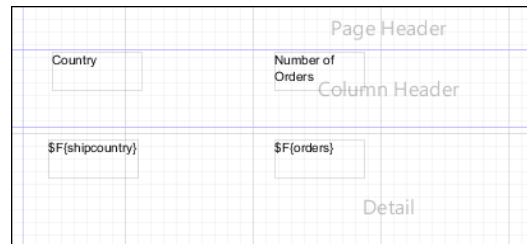


Figure 14-1 Initial Report Design

4. Expand the Summary band to 378 pixels.

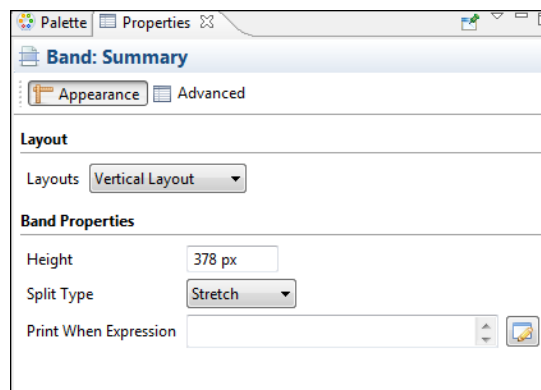


Figure 14-2 Summary Band Properties

5. Drag the Chart tool from the Palette into the Summary band. The Chart Wizard opens.

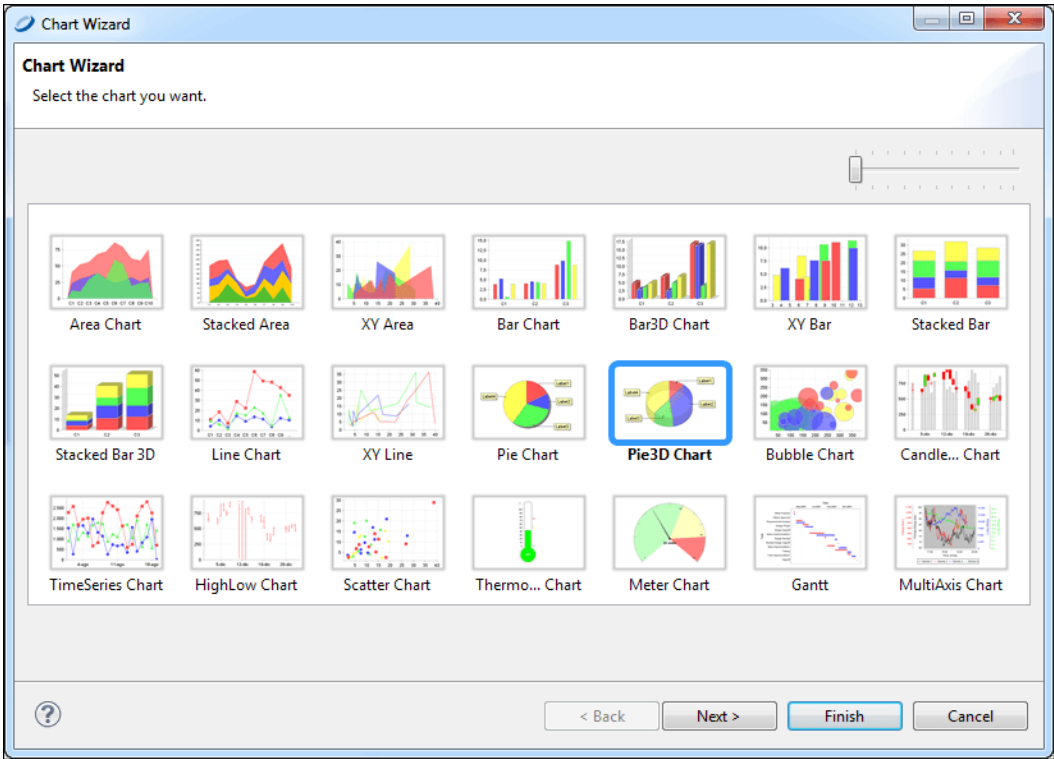


Figure 14-3 Chart Wizard

- 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.

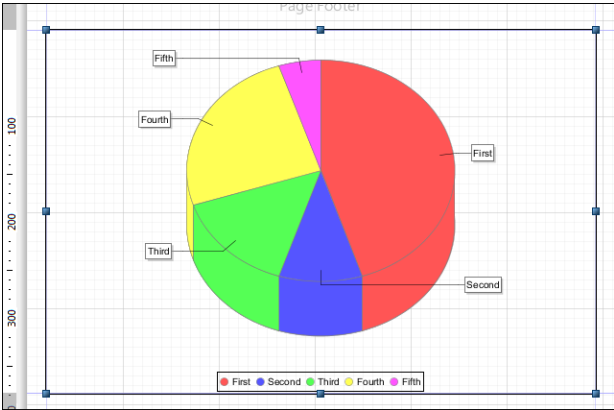



Figure 14-4 Chart in Summary Band

 In the Design tab, 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.

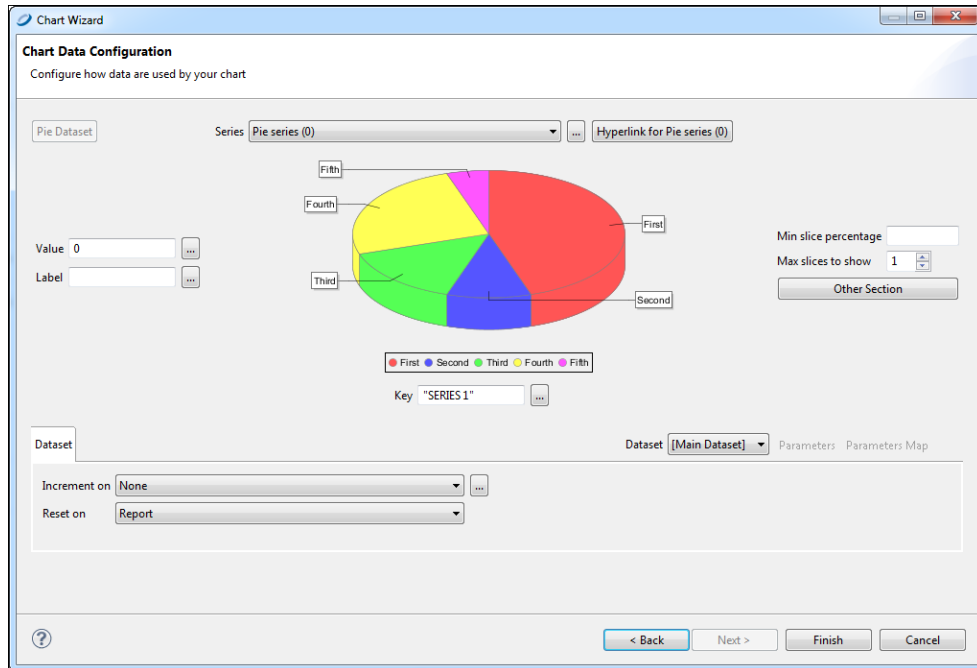


Figure 14-5 Chart Wizard - Chart Data Configuration

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.

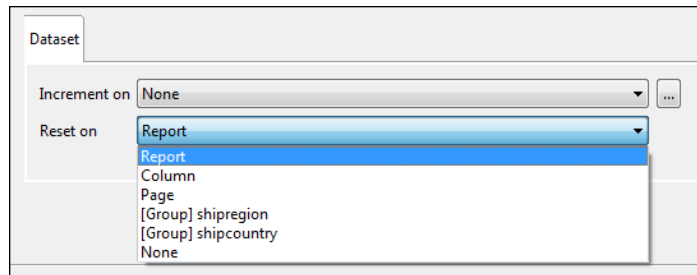


Figure 14-6 Dataset Tab

- 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}`

- Click **Finish**.
- Save your report, and preview it to see the result.

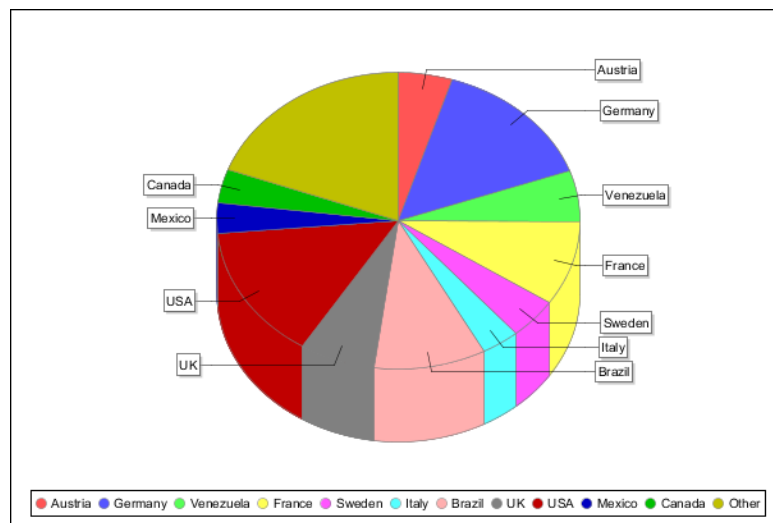


Figure 14-7 Final Chart

In this chart, each slice represents a country and the shipping total for that country.

14.2 Setting Chart Properties

When you select a chart component in the Design tab, the Properties view shows Hyperlinks, Chart, and Chart Plot tabs, in addition to the standard Appearance, Borders, and Advanced tabs.

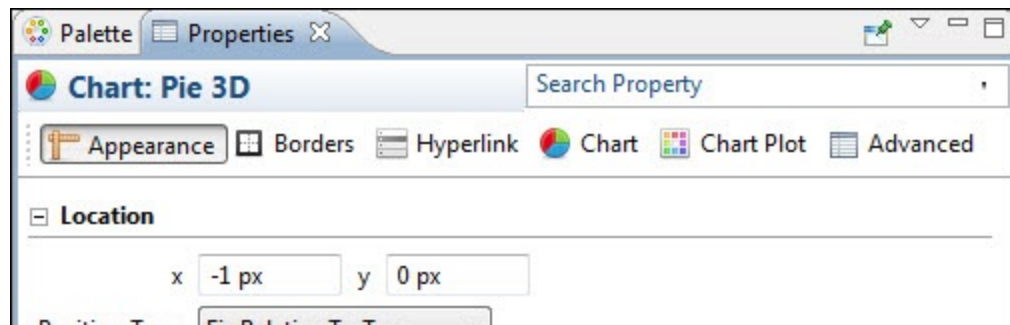


Figure 14-8 Properties View



For more information about setting hyperlinks, see [“Anchors, Bookmarks, and Hyperlinks” on page 55](#).

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.

14.3 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.

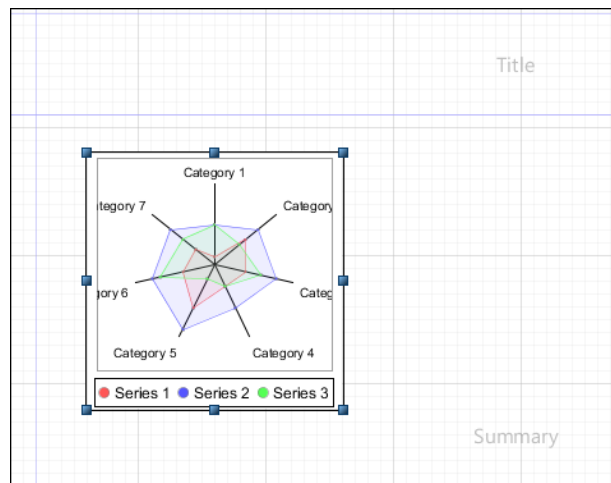


Figure 14-9 Spider Chart

2. Select the report in the Outline view, and in the Properties view, click the **Edit query, filter, and sort options** button. The Dataset and Query dialog opens.

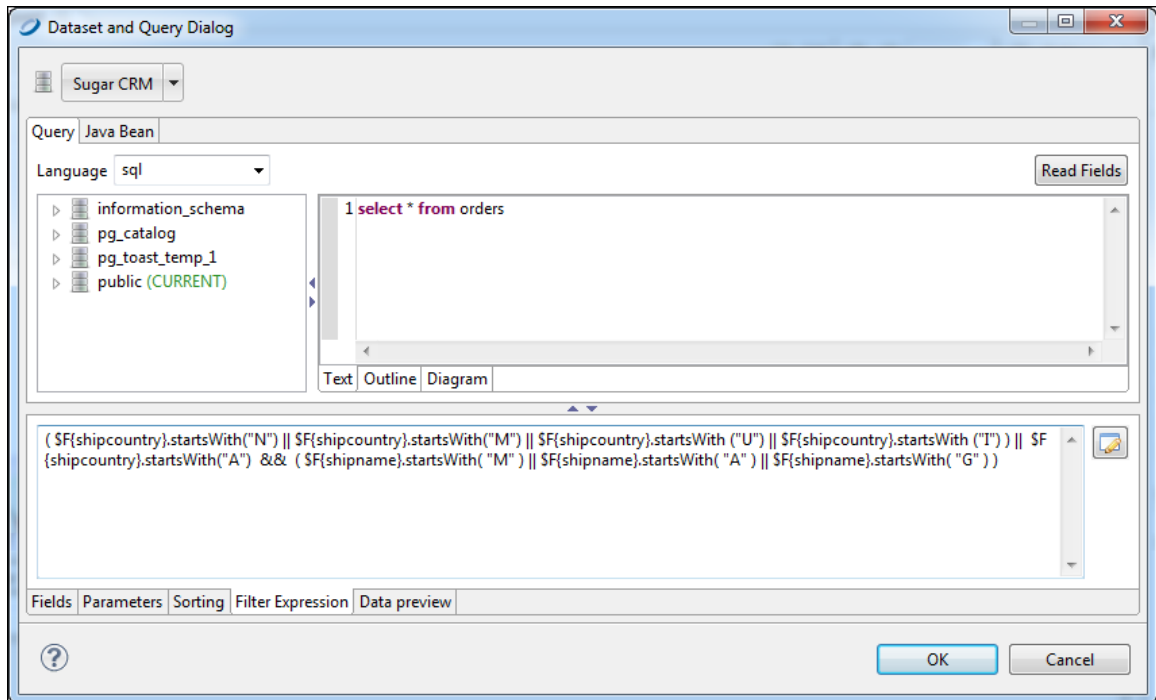


Figure 14-10 Increment Expression for Spider Chart



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 **Increment 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 tab should look like the following figure.

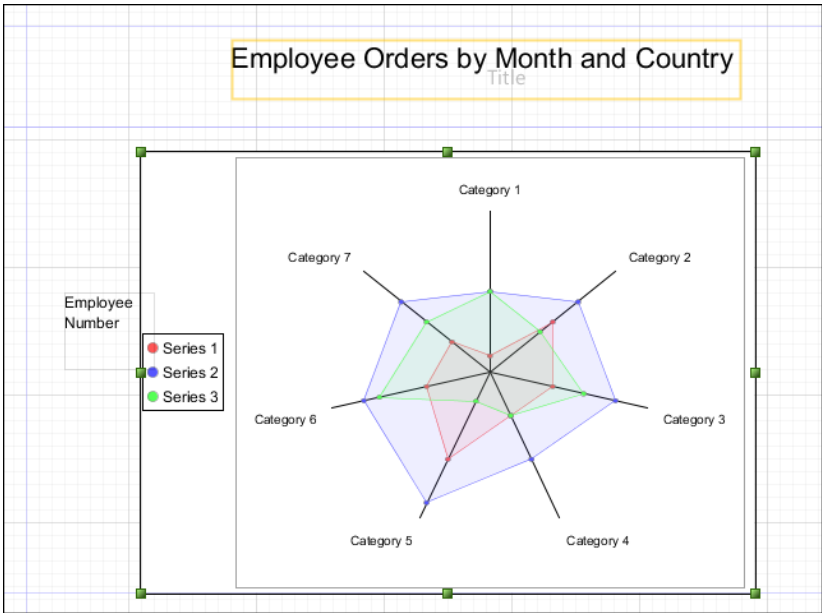


Figure 14-11 Spider Chart Design

6. Preview your report. It should look like the following:



Figure 14-12 Spider Chart Preview

14.4 Chart Themes

Chart themes give you full control over the style of your JFree charts. You can create a chart theme and use it in multiple reports for a uniform look. And you can update that look for all those reports simply by updating the chart theme. JasperSoft Studio is the supported tool for creating JFree 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.
3. Select a folder and enter a file name for your theme. The file extension must be `.jrtcx`.
4. Click **Finish**. The Chart Theme Designer opens.

14.4.1 Using the Chart Theme Designer

In the Outline view, expand the Chart Theme list to view the subsections of a theme design.

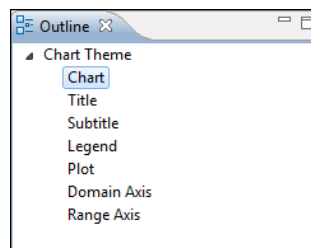


Figure 14-13 Chart Theme Designer Outline View

- **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.

14.4.2 Editing Chart Theme XML

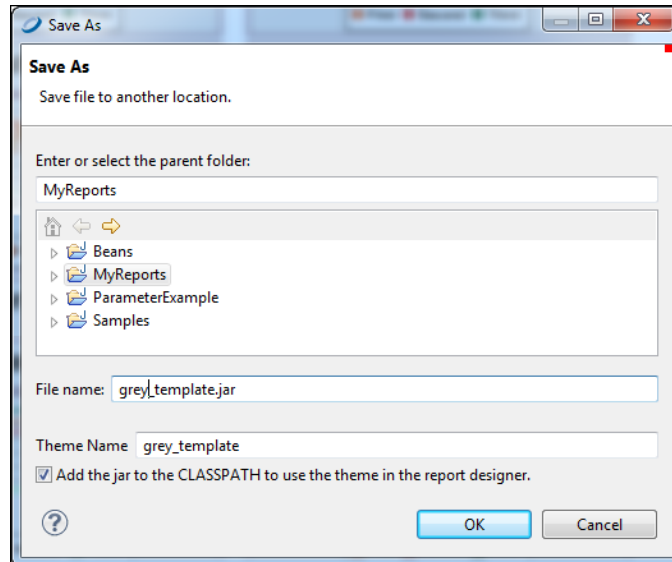
You can see and edit the chart theme XML from the **Source** tab. However, the XML appears as one long line. It is better paste it into a text editor for editing.

14.4.3 Creating a JasperReports Extension for a Chart Theme

To be used in a report, the `.jrtcx` file must be exported to a JasperReports extension JAR file

To export the theme as a JAR:

1. On the **Preview** tab, click . The Save As window opens.

**Figure 14-14 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.

14.4.4 Applying a Chart Theme

Add the theme to your class path to make it available in the list of themes.

CHAPTER 15 HTML5 CHARTS 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](#)
- [Hyperlinks in HTML5 Charts](#)

For details about charts implemented through JFreeCharts, see [“Working with Charts” on page 209](#).

15.1 Overview of HTML5 Charts

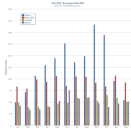
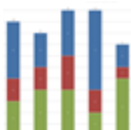
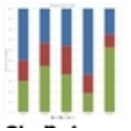
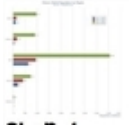


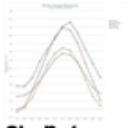
You can create interactive reports with HTML5 charts. And they're more attractive than the basic charts available in Jaspersoft Studio. This section shows how to build a report containing a simple HTML5 chart and how to change chart types and edit charts.


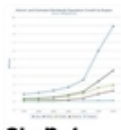
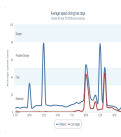
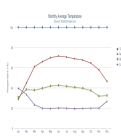
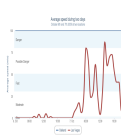

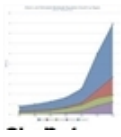
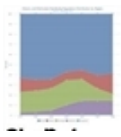
In Jaspersoft Studio HTML5 Charts work like Ad Hoc charts. Levels, for example, are the equivalent of what you would use the Ad Hoc Slider to see. Other explanations include:




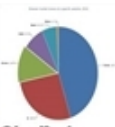


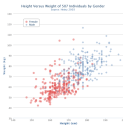
- **Values:** Static properties.
- **Expressions:** Dynamic properties.
- **Categories:** Rows. In a pie chart, the categories are the slices.
- **Measures:** The same as in Ad Hoc. In a pie chart, these are the size of the slice.
- **Series Contributors:** In the Design tab, these are defined at the measure level. In JRXML these are defined as Series.

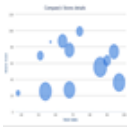
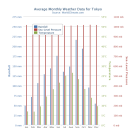
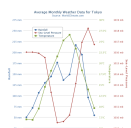
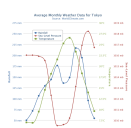

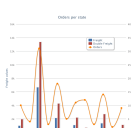

Before you add a chart to your report, consider the best way to display your data. The following table describes the available chart types.

Table 15-1 HTML5 Chart Types


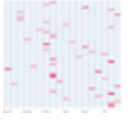
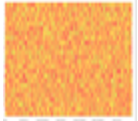



Icon	Description
Column charts - Compare values displayed as columns	
	<p>Column. Multiple measures of a group are depicted as individual columns.</p>
	<p>Stacked Column. Multiple measures of a group are depicted as portions of a single column whose size reflects the aggregate value of the group.</p>
	<p>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.</p>
Bar charts - Compare values displayed as bars	
	<p>Bar. Graphically summarize and display categories of data to let users easily compare amounts or values among categories.</p>
	<p>Stacked Bar. Multiple measures of a group are depicted as portions of a single bar whose size reflects the aggregate value of the group.</p>
	<p>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.</p>
Line charts - Compare values displayed as points connected by lines	
	<p>Line. Displays data points connected with straight lines, typically to show trends.</p>

Icon	Description
	<p>Spline. Displays data points connected with a fitted curve. Allow you to take a limited set of known data points and approximate intervening values.</p>
	<p>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.</p>
	<p>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</p>
	<p>Percent Line. A variation of a line chart in which each series adjoins but does not overlap the preceding series.</p>
	<p>Percent Spline. A variation of a spline chart in which each series adjoins but does not overlap the preceding series.</p>
<p>Area charts - Compare values displayed as shaded areas. Compared to line charts, area charts emphasize quantities rather than trends.</p>	
	<p>Area. Displays data points connected with a straight line and a color below the line; groups are displayed as transparent overlays.</p>
	<p>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.</p>
	<p>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.</p>

Icon	Description
	<p>Area Spline. Displays data points connected with a fitted curve and a color below the line; groups are displayed as transparent overlays.</p>
	<p>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.</p>
	<p>Percent Area Spline. A variation of area spline charts that present values as trends for percentages, totaling 100% for each category.</p>
<p>Pie charts - Compare values displayed as slices of a circular graph</p>	
	<p>Pie. Multiple items of a single group are displayed as sectors of a circle.</p>
	<p>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:</p> <ul style="list-style-type: none"> • 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
	<p>Semi-Pie. Multiple measures of a group are displayed as sectors of a half-circle.</p>
<p>Scatter and Bubble Charts - Show the extent of correlation, if any, between the values of observed quantities.</p>	
	<p>Scatter. Displays a single point for each point in a data series without connecting the points.</p>

Icon	Description
	<p>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</p>
<p>Multi-Axis Charts - Compare trends in two or more data sets whose numeric range differ greatly.</p>	
	<p>Multi-Axis Column. A column chart with two series and two axis ranges.</p>
	<p>Multi-Axis Line. A line chart with two series and two axis ranges.</p>
	<p>Multi-Axis Spline. A spline chart with two series and two axis ranges.</p>
<p>Combination Charts - Display multiple data series in a single chart, combining the features of a area, bar, column, or line charts.</p>	
	<p>Column Line. Combines the features of a column chart with a line chart.</p>
	<p>Column Spline. Combines the features of a column chart with a spline chart.</p>
	<p>Stacked Column Line. Combines the features of a stacked column chart with a line chart.</p>

Icon	Description
	<p>Stacked Column Spline. Combines the features of a stacked column chart with a line chart.</p>
<p>Time Series Charts - Illustrate data points at successive time intervals. Also called Fever Chart.</p>	
	<p>Time Series Area. Displays data points over time connected with a straight line and a color below the line.</p>
	<p>Time Series Area Spline. Displays data points over time connected with a fitted curve and a color below the line.</p>
	<p>Time Series Line. Displays data points over time connected with straight lines.</p>
	<p>Time Series Spline. Displays data points over time connected with a fitted curve.</p>
<p>Spider Charts - Display data in line or data bars arranged on a circular spider web chart. Also called a Radar Chart.</p>	
	<p>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.</p>
	<p>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.</p>

Icon	Description
	<p>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.</p>
<p>Range Charts</p>	
	<p>Heat Map. Represents data in a matrix format, using color coding to show values.</p>
	<p>Time Series Heat Map. Represents data across time in a heat map, using color coding to show values.</p>
	<p>Dual Measure Tree Map. Displays data as color-coded rectangles; the size of each rectangle is proportional to the first measure and the color represents the second measure.</p>
	<p>Tree Map. Displays data as rectangles; the size of each rectangle is proportional to the measure of the data it represents. The tree map displays nested rectangles when you have more than one field; the parent rectangle represents the leftmost measure while the nested rectangles represent the current level of aggregation. Click on a parent rectangle to drill down to the nest rectangles</p>
	<p>One Parent Tree Map. Displays data as nested rectangles; the size of each rectangle is proportional to the measure of the data it represents. The nested rectangles represent the current level of aggregation while the larger rectangle represents the parent level in the hierarchy. Click on a parent rectangle to drill down to the nest rectangles.</p>

15.2 Simple HTML5 Charts

Before you add a chart, consider the best way to display your data. Available chart types are listed in [Table 15-1](#).

15.2.1 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 169](#).

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 15-1 Palette

4. Select a chart type based on the information you want to display. See **Table 15-1** for help. For this example, we'll choose **Pie chart**.

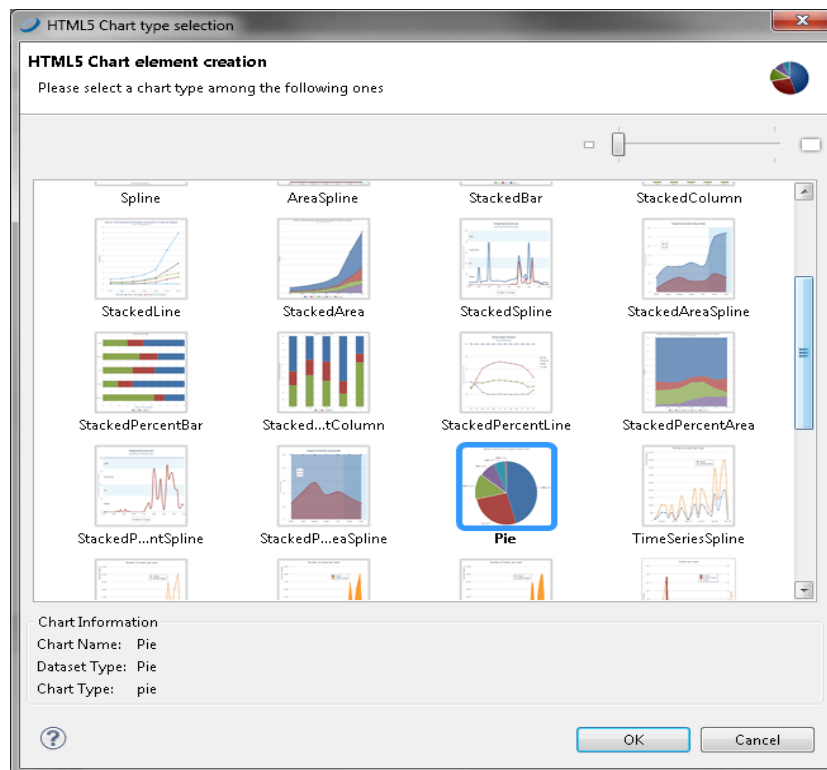


Figure 15-2 Chart Types

Your report now includes a sample chart. The Design tab 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.

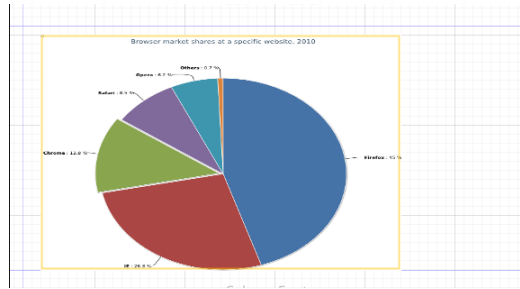


Figure 15-3 Pie Chart Example

- Right-click the chart, and choose **Edit Chart Properties**. The Chart Properties dialog includes tabs for configuring chart properties, chart data, and hyperlinks.

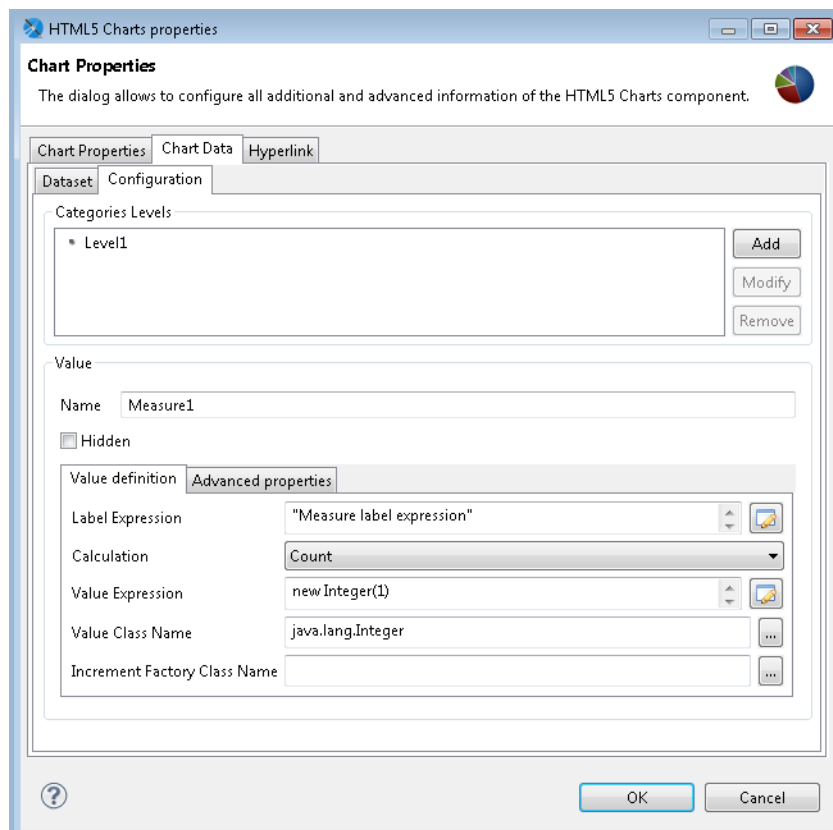



Figure 15-4 HTML5 Charts Properties > Chart Data > Configuration

- Go to the **Chart Data** tab and click the **Configuration** tab.
- Highlight **Level1**, and click **Modify**.
- Name the Expression **Country**.
- Click  to open the Expression Editor and change the default to the `shipcountry` field. Click **OK**.
- 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.


15.2.2 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 **Increment** expression text box, click  to open the Expression Editor.
3. Filter your data set by adding an Increment 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.

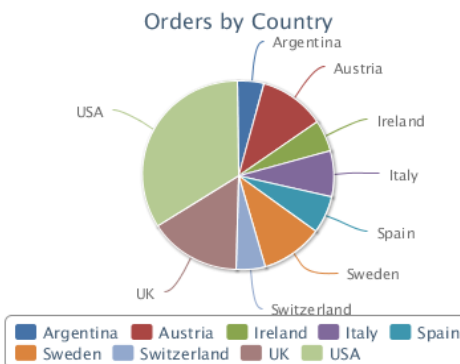


Figure 15-5 Filtered Pie Chart

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 15-6 Bar Chart with Second Measure and Series

15.2.3 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 [15.5.1, “Creating Hyperlinks in HTML5 Charts,” on page 241](#).

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 55**.

15.2.4 Setting Advanced Options for HTML5 Charts

You can add or edit advanced options for HTML5 Charts using the Chart Properties dialog. This section shows how to prevent users from changing chart types in reports you upload to JasperReports Server.

When you create an HTML5 chart in Jaspersoft Studio and upload it to JasperReports Server, users may see a Canvas Options icon . Clicking this icon displays a **Chart Types...** option that allows the user to select a different chart type. You can disable this option at any level – for the chart, for the report, or for this instance of Jaspersoft Studio.

To disable this option for a chart:

1. Select the chart element in the Design or Outline view.
2. In the Properties view for the chart element, click the **Advanced** tab.
3. Expand the **Misc** section and click ... next to **Edit Properties**.
The Properties dialog opens.
4. Click Add to add a new property.
5. Enter the following information:
 - **Name:** `com.jaspersoft.jasperreports.highcharts.interactive`
 - **Value:** `false`
6. Click **OK** and then click **Finish**.

Like many advanced charting options, this option can be set at a higher level. If you set an option at multiple levels, the lowest level is the one that is applied.

- To disable this option for a report, click on the report's root node in Outline view and make sure the Properties view is displayed. In the Properties view, on the tab, select **Advanced > Misc > Properties** and click ... to open the Properties Dialog. Enter the values shown above and click **OK** twice to apply the setting.
- To disable this option Jaspersoft Studio, select **Window > Preferences** from the menu. In the Preferences dialog box, select **Properties** and click **Add** to open the Properties Dialog. Enter the values shown above and click **OK** twice to apply the setting.

You can also set this particular option in JasperReports Server. See the *JasperReports Server Administrator Guide* for more information.

15.3 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 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**.

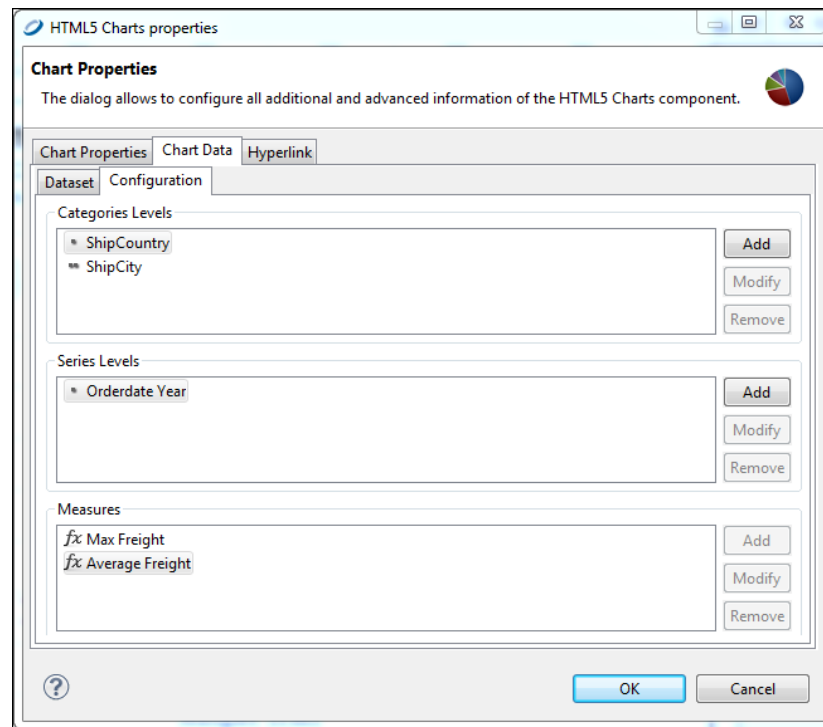


Figure 15-7 Scatter Chart Properties

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**.

To change the position or layout of the legend:

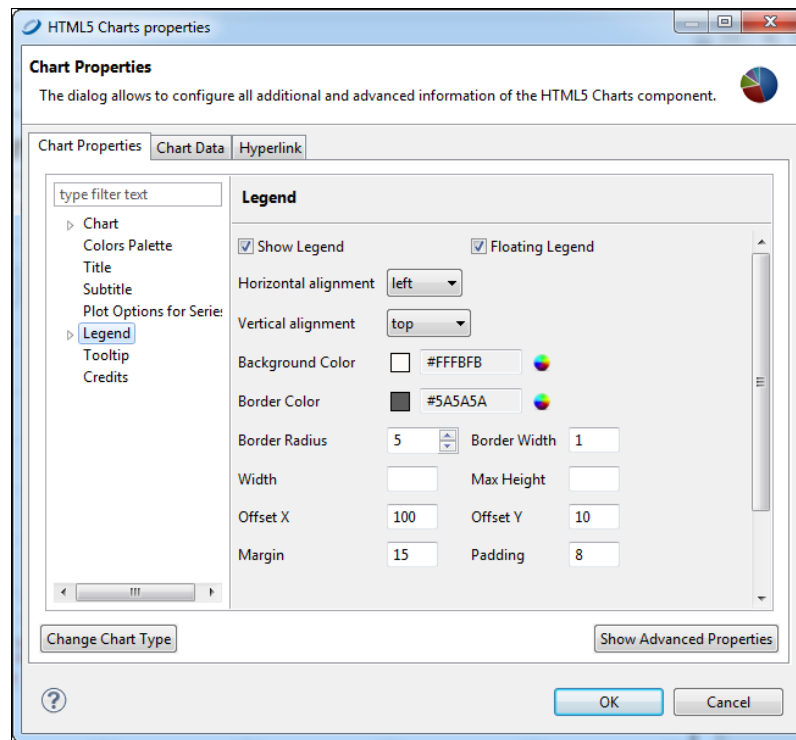
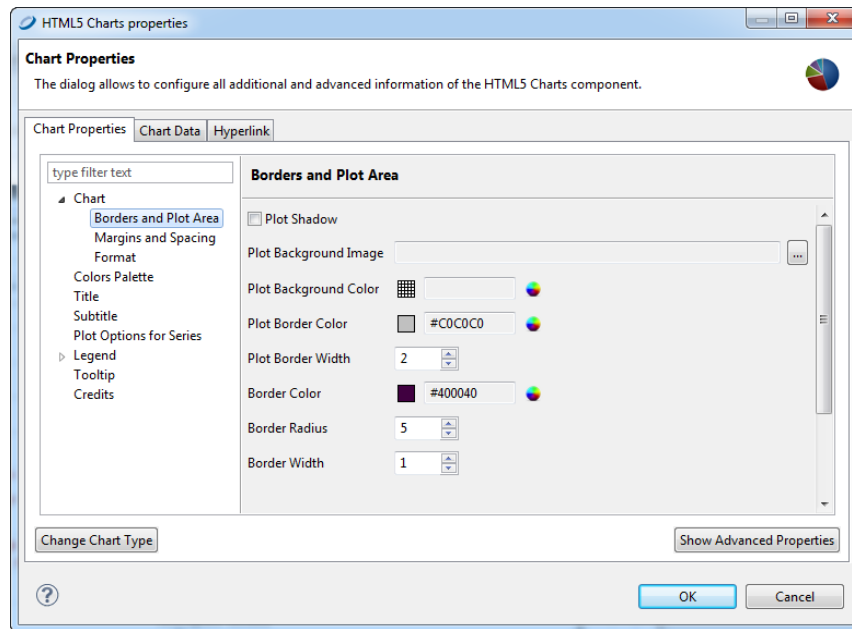


Figure 15-8 Legend Properties

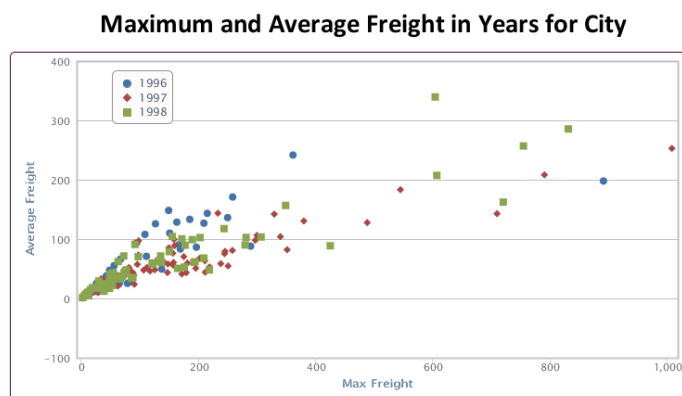
1. Open **HTML5 Chart Properties** and click the **Chart Properties** tab.
 2. Highlight **Legend**.
 3. Check the boxes for **Show Legend** and **Floating Legend**.
 4. Use the drop-down menus to arrange the legend to appear in the top left by setting Offset X and Offset Y to appear within the plot 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** and click **OK**.

**Figure 15-9** Border Properties

5. Save and preview your report.

**Figure 15-10** Scatter Chart

15.4 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. Right-click the chart and select **Chart Properties > Chart Data > Configuration**.

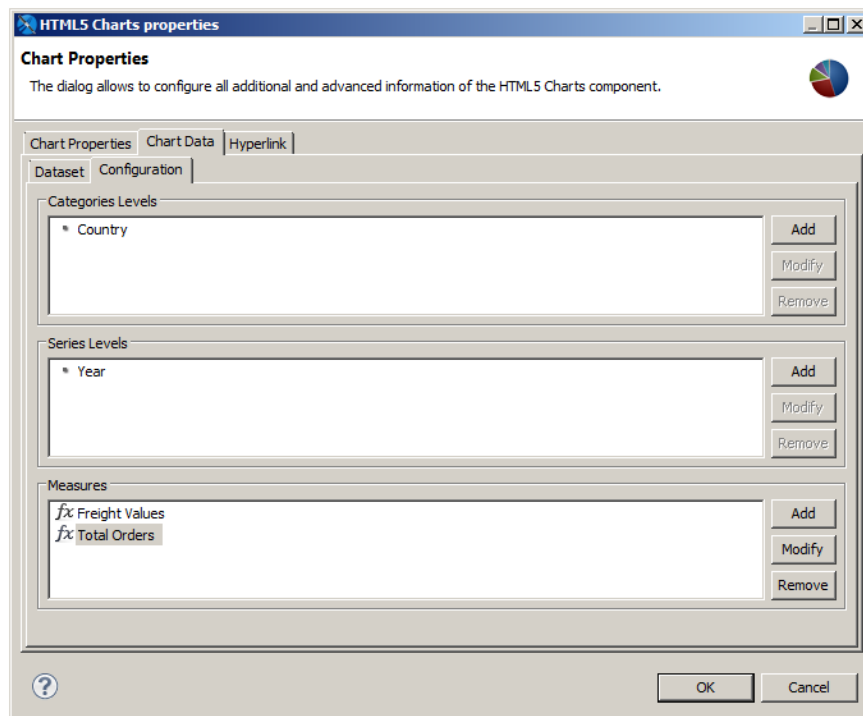


Figure 15-11 Configuration for Chart Data

3. Create a Category with the following:
 - **Name:** Country
 - **Expression:** \$F{SHIPCOUNTRY}

- **Value Class Name:** `java.lang.String`
 - **Order:** None
- Click **OK**.
4. Create a Series with the following:
 - **Name:** Year
 - **Expression:** `new java.text.SimpleDateFormat ("yyyy").format (${ORDERDATE})`
 - **Value Class Name:** `java.lang.Comparable`
 - **Order:** Ascending
 Click **OK**.
 5. Add a Measure for freight values:
 - **Name:** Freight Values
 - **Label Expression:** "Freight"
 - **Calculation:** Count
 - **Value Expression:** `${FREIGHT}`
 - **Value Class Name:** `java.lang.Integer`
 Do not click **OK** yet. The next step shows how to specify that this measure should be used for columns.
 6. Go to Advanced Properties for the measure you just created and click **Add** to specify the series type.

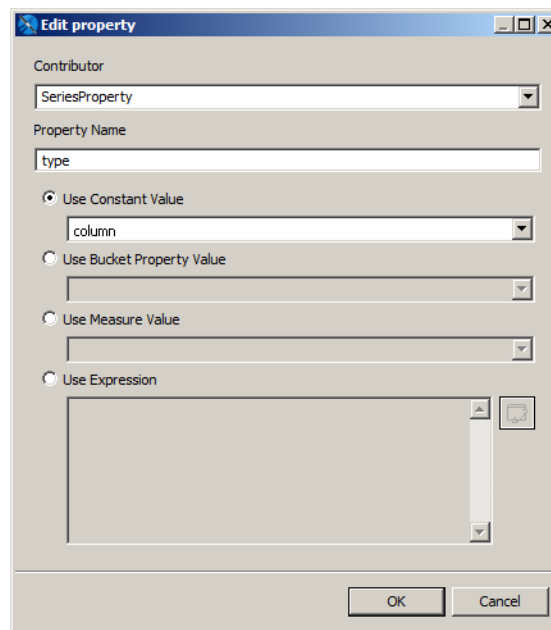


Figure 15-12 Edit Advanced Properties for Freight Values

7. Enter the following values. The support constant values are `column`, `line`, and `spline`:
 - **Contributor:** `SeriesProperty`
 - **Property Name:** `type`
 - **Use Constant Value:** `column`
 Click **OK**, then click **OK** again.
8. 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

Do not click **OK** yet. The next step shows how to specify that this measure should be used for the spline.

9. Go to Advanced Properties for the measure you just created and click **Add** to specify the series type (available options are column, line, and spline):

- **Contributor:** SeriesProperty
- **Property Name:** type
- **Use Constant Value:** spline

Click **OK**, then click **OK** again.

10. To see fewer countries in your chart, select **Edit Chart Properties > Chart Data > Dataset**.

11. Add an Increment expression such as:

```
$F{SHIPCOUNTRY}.startsWith("A") ||
$F{SHIPCOUNTRY}.startsWith("U") ||
$F{SHIPCOUNTRY}.startsWith("I")
```

Click **OK**.

12. To change the position or layout of the legend:

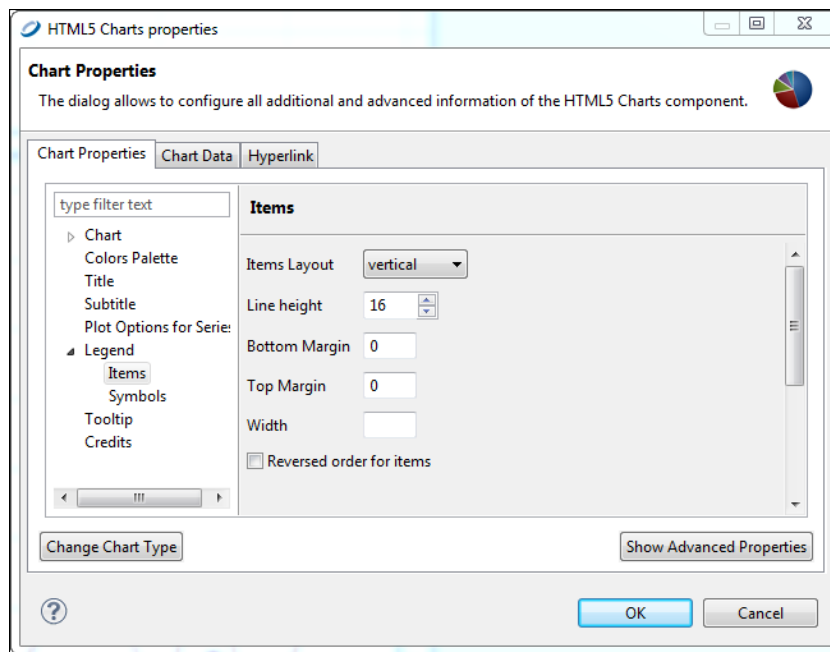


Figure 15-13 Legend Properties

- 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**.

13. Save and preview your report. It should look like the following figure.

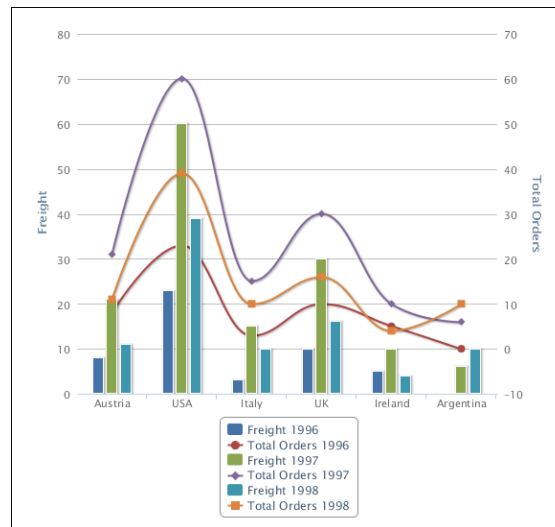


Figure 15-14 Column Spline Chart

15.5 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 4.9, “[Anchors, Bookmarks, and Hyperlinks](#),” on page 55.

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.

15.5.1 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**.

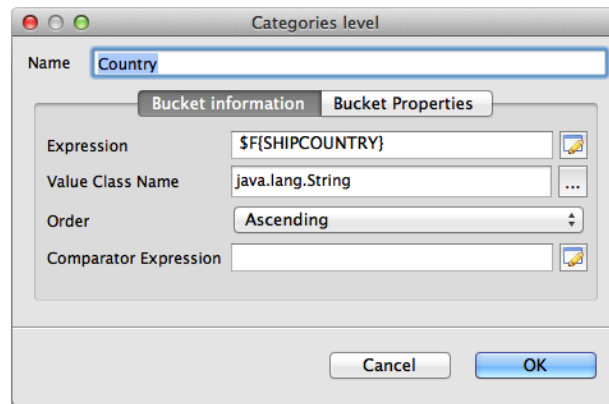


Figure 15-15 Editing Categories in the Chart Data Configuration Window

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:

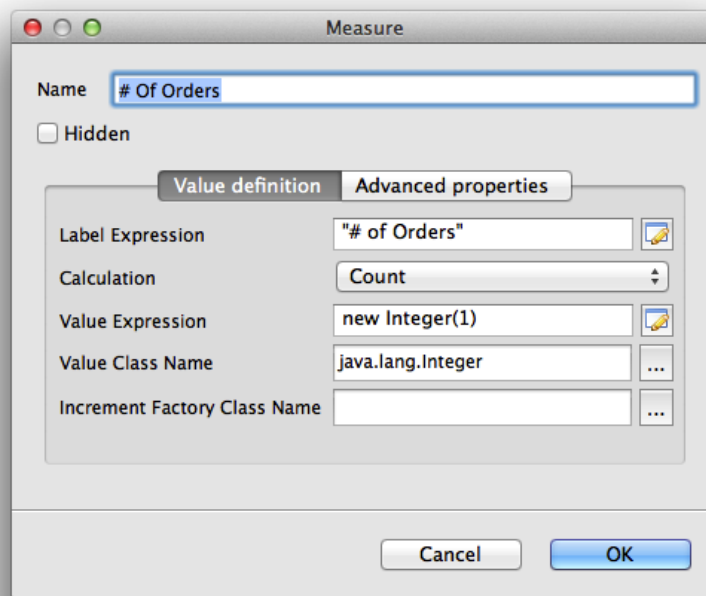


Figure 15-16 Editing the Measure to Count Orders

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

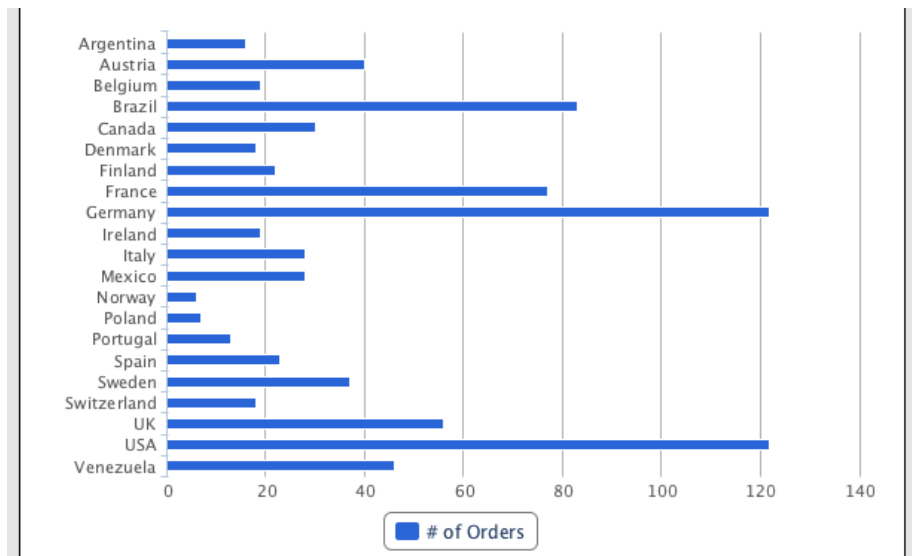


Figure 15-17 Successful Test of the Example Report

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.

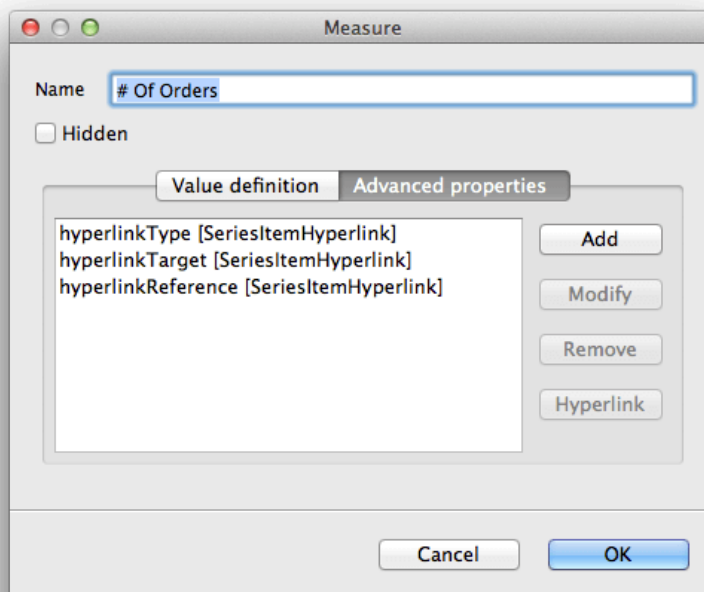


Figure 15-18 Editing the Number of Orders a Second Time

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**.

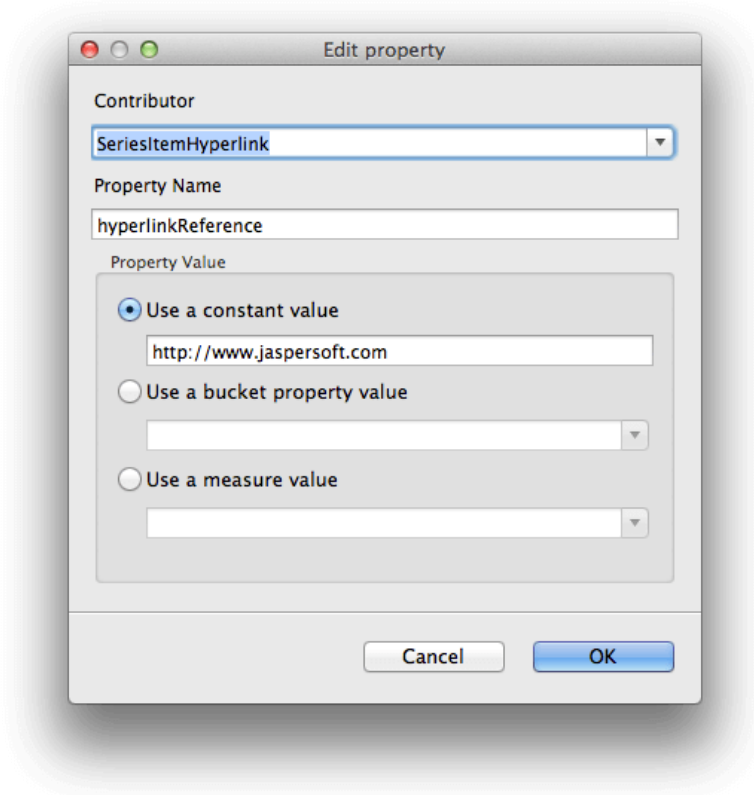


Figure 15-19 Editing the Property Definition for hyperlinkReference

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

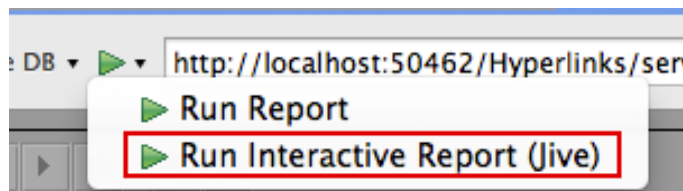


Figure 15-20 Previewing the Report

Property values can come from a static value, a bucket property value (explained in **“Working with Bucket Properties and Hidden Measures” on page 245**) 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).

15.5.2 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.

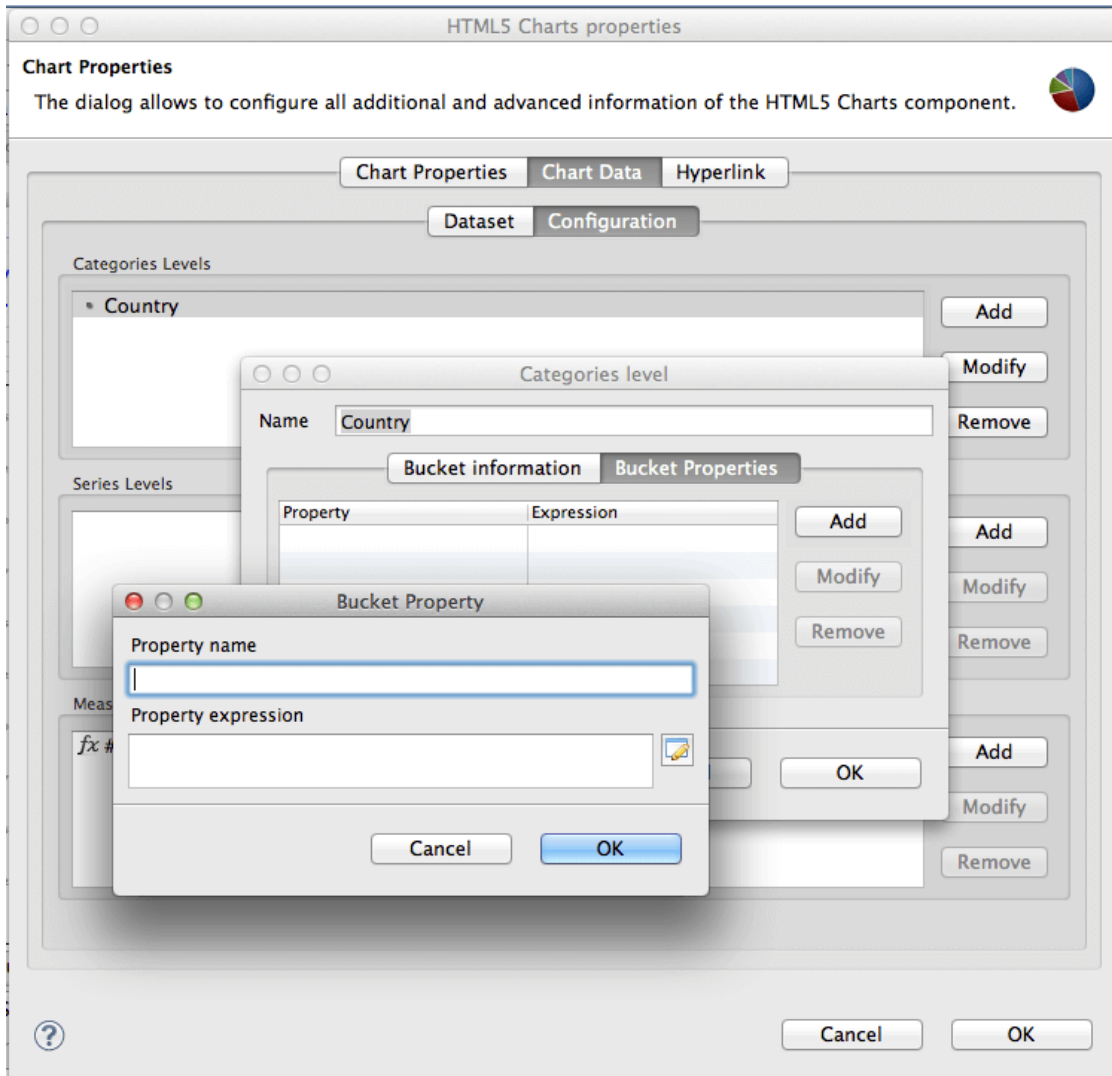


Figure 15-21 Defining Bucket Properties for HTML5 Chart Hyperlinking

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.

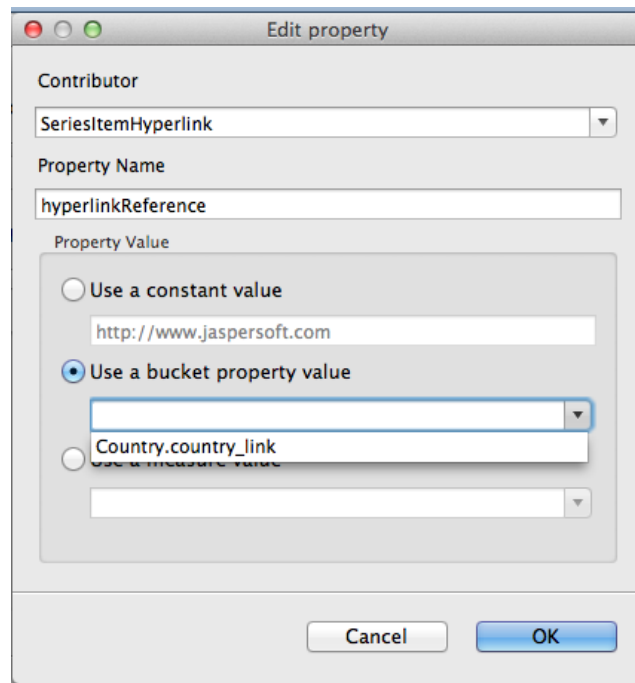


Figure 15-22 Configuring Measure Properties to set the HyperlinkReference Value

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 from 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})

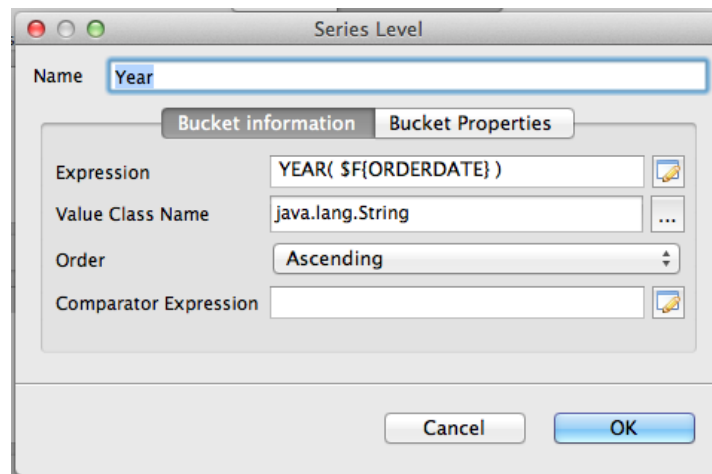


Figure 15-23 Editing Series Level to Define Buckets

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

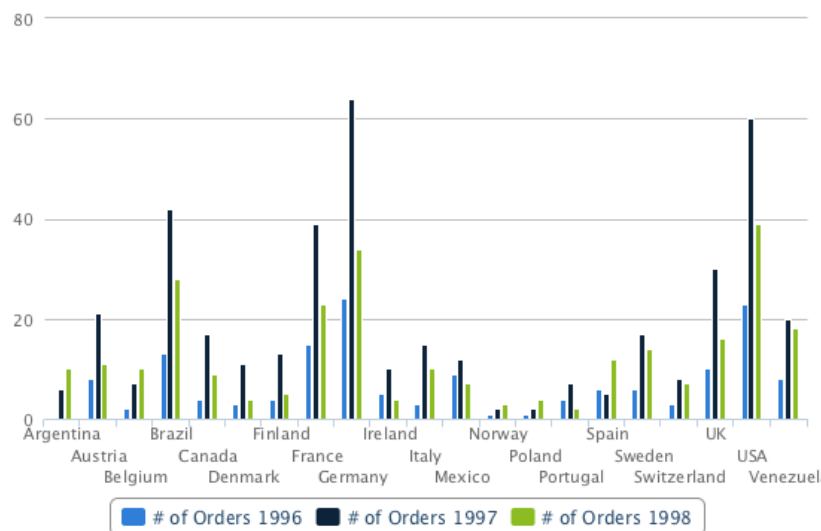


Figure 15-24 Bar Chart with Order Numbers Split by Year

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

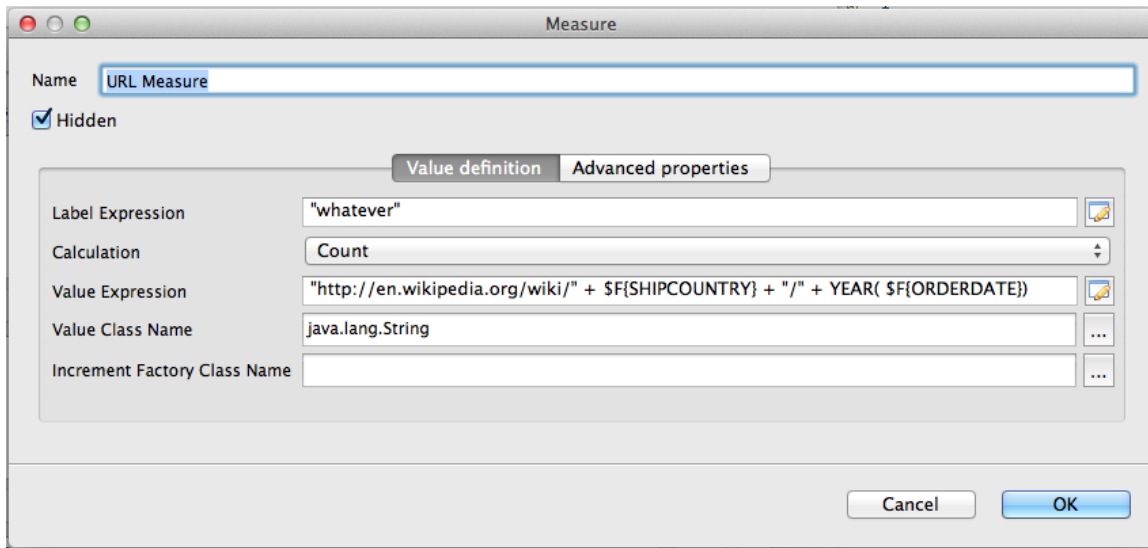


Figure 15-25 Defining the URL to Open when the Measure is Clicked

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.

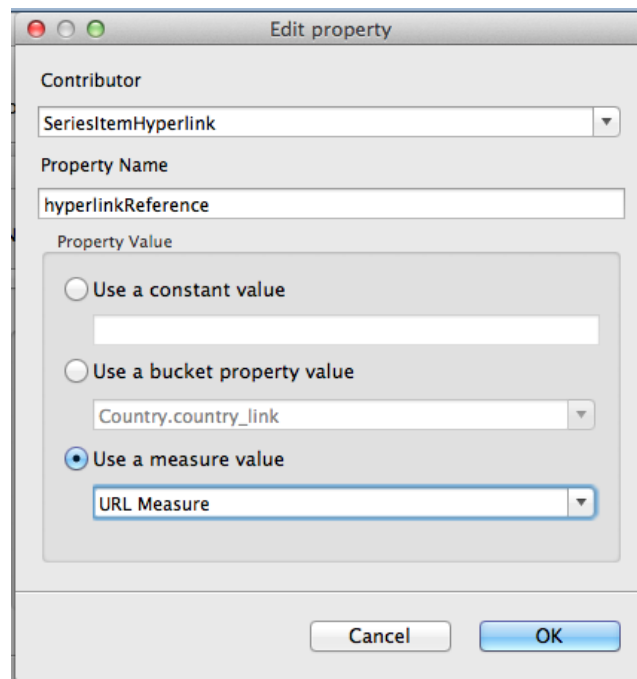


Figure 15-26 Editing the hyperlinkReference to use a Measure Value

3. Save all and preview.

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

15.5.3 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.

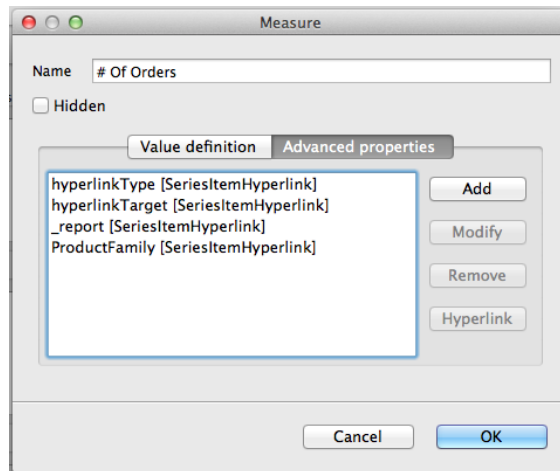


Figure 15-27 Measure Configuration for # of Orders

CHAPTER 16 WORKING WITH CROSTABS

In contrast to a table or tabular report showing individual records, a crosstab shows aggregate data for two or more variables in a tabular matrix.

You do not specify the individual row and column values for a crosstab at design time. Instead you specify fields in your dataset, called row or column groups, and your crosstab displays a row or column for each unique value of the field. This means the exact number of rows and columns remains undefined at design time and the crosstab automatically updates to reflect your dataset. A crosstab must include at least one row and one column group.

The cells in a crosstab show summary data for the corresponding row and column, based on a measure and a summary function. The simplest crosstab is a frequency matrix, such as the following example, which shows the count of pets (measure) by gender (column) and species (row).

	Female	Male	Total
Cats	12	7	19
Dogs	7	8	15
Horses	0	1	1
Snakes	0	1	1
Total Pets	19	17	36

Figure 16-1 Example of a simple crosstab

You can increase the complexity of a crosstab by adding more row or column groups, or by using another summary function, such as sum, average, or percent. For example, the following crosstab shows the sum of the monthly cost of food for each pet (measure) by gender (column) and group and species (rows). Note that when there are multiple row or column groups in a crosstab, they are displayed hierarchically.

		Female	Male	Total
Mammals	Cats	\$480	\$280	\$760
	Dogs	\$133	\$160	\$293
	Horses	\$0	\$275	\$275
	Total	\$613	\$715	\$1,328
Reptiles	Snakes	\$0	\$9	\$9
	Total	\$0	\$9	\$9
Total		\$613	\$724	\$1,337

Figure 16-2 Example of a crosstab with multiple row groups and a sum


Crosstabs in JasperReports support row and column groups, totals and subtotals, and individual cell formatting. Data to fill the crosstab can come from the main report dataset or from a subdataset.

This chapter has the following sections:

- [Example of Creating a Crosstab](#)
- [Working with Crosstab Properties](#)
- [Using the Crosstab Editor](#)
- [Working with Crosstab Parameters](#)

16.1 Example of Creating a Crosstab

When you add a Crosstab element to a report, Jaspersoft Studio displays the Crosstab Wizard automatically.

1. Create a new report:
 - a. Choose a blank template.
 - b. Select the Sample DB data adapter and click **Next**.
 - c. Enter the query `select * from orders`.
 - d. On the Fields page, select all fields and click **Finish**.
2. Because a crosstab summarizes information, you put it in the Summary or Title band. For this example, delete all bands except the Title and Summary bands. This eliminates blank pages in the final report.
3. Drag the Crosstab tool  into the Summary band. The Dataset page of the Crosstab wizard is displayed.

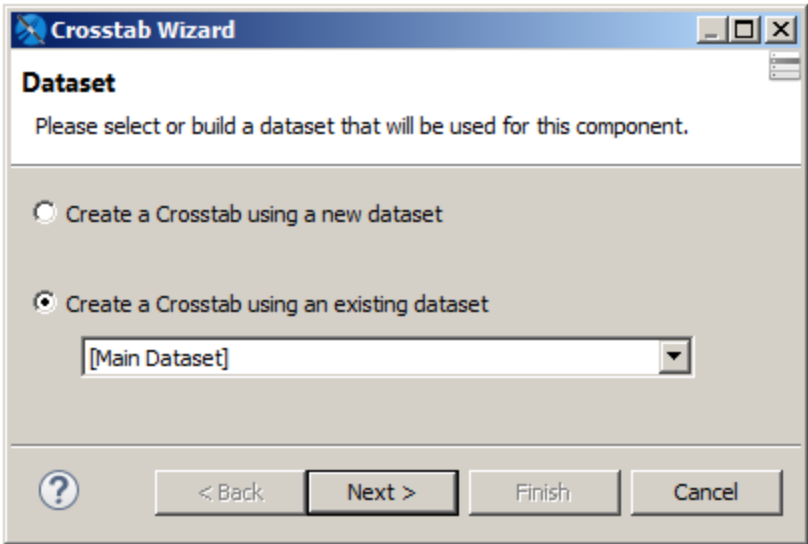


Figure 16-3 Dataset page of the Crosstab wizard

- 4. For this example, make sure that **Create a Crosstab using an existing dataset** is selected, and select **[Main Dataset]** from the drop-down menu.
- 5. Click **Next**. The Columns screen is displayed.
- 6. Enter one or more fields you want as column groups. For this example, choose the ORDERDATE field.

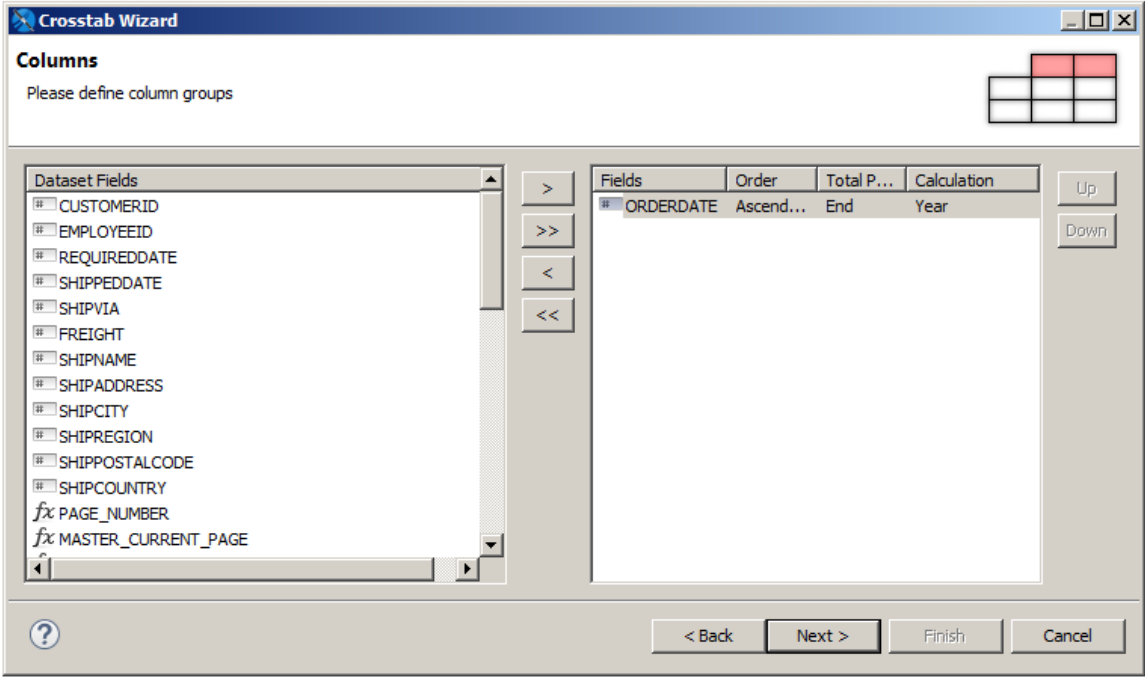


Figure 16-4 Defining column groups

7. Select the `ORDERDATE` field. Then click on the Unique value in the **Calculation** column and select **Year** from the drop-down menu. This aggregates the orders by year.



When you have a time field in your crosstab, you can use the Unique aggregation function to group records having the same value, or you can aggregate it any of the following ways:

- Using a time-based aggregation function (such as `Year`, `Month`, `Week`, or `Day`) when you define the group
 - In this example, this is shown in the previous step.
- Using a dataset query when you create the crosstab.
 - In this example, in the first step of the wizard, you could create a dataset that uses a query that returns the year, such as `select ORDERDATE, SHIPVIA, SHIPPOSTALCODE, SHIPCOUNTRY, SHIPPEDDATE, YEAR(SHIPPEDDATE) as SHIPPEDYEAR from orders`
- Manually editing the element expression in the crosstab editor after the crosstab has been created, as described in , **“Editing the expression of a group,” on page 259**.
 - In this example, you could change the column element expression from `$(SHIPPEDDATE)` to `YEAR($(SHIPPEDDATE))`.

8. Click **Next**. The Rows screen is displayed.
9. Enter one or more fields you want as row groups. For this example, choose `SHIPCOUNTRY` and `SHIPPOSTALCODE`.

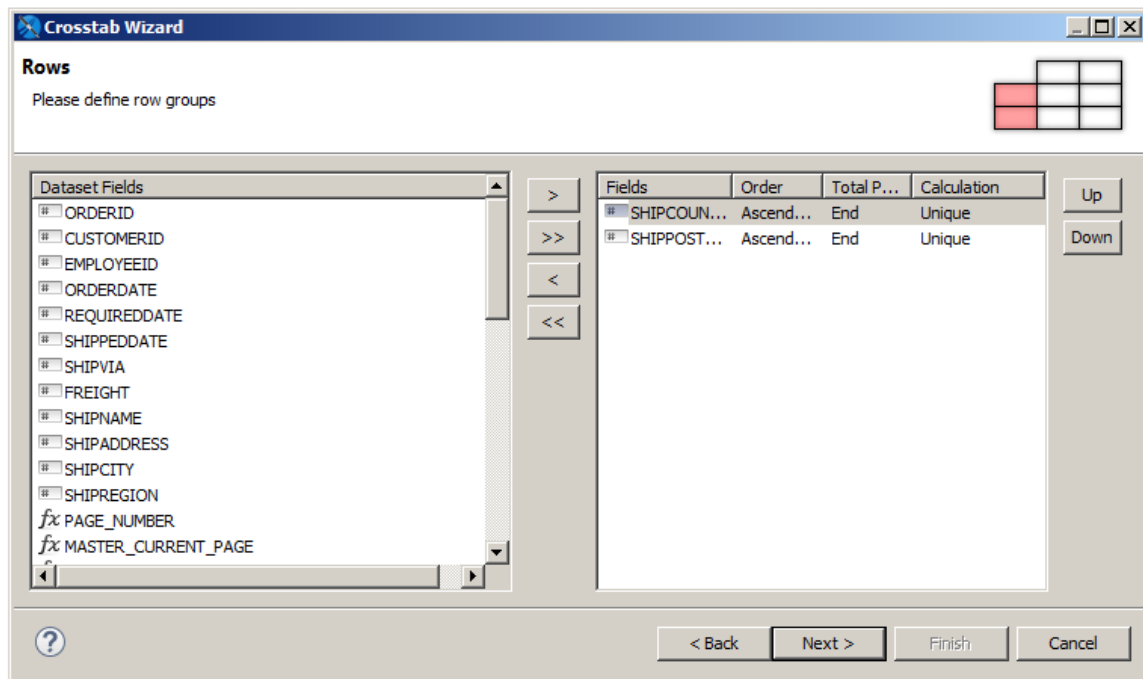


Figure 16-5 Defining row groups

10. Make sure that the fields appear in the order you want them in the crosstab. For this example, ensure that `SHIPCOUNTRY` appears first in the list by selecting it and clicking **Up**.

Grouping by `SHIPCOUNTRY` and then `SHIPPOSTALCODE` results in each row in the crosstab referring to a specific country, with subgroups by postal code within the country. Unlike in the main report,

JasperReports will sort the data for you, although you can disable this function to speed up the fill process if your data is already sorted.

11. Click **Next**. The Measures screen is displayed.
12. Enter one or more fields you want as measures. For this example, choose `ORDERID`.

Measures define the detail data in the crosstab; normally, this is the result of an aggregation function like the count of orders by country by year, or the sum of freight for the same combination (country/ year). By default, the aggregate function is Count, which is what we want for this example. To change the aggregate function, you would select `ORDERID`, click on Count, and select a different value from the drop-down menu.

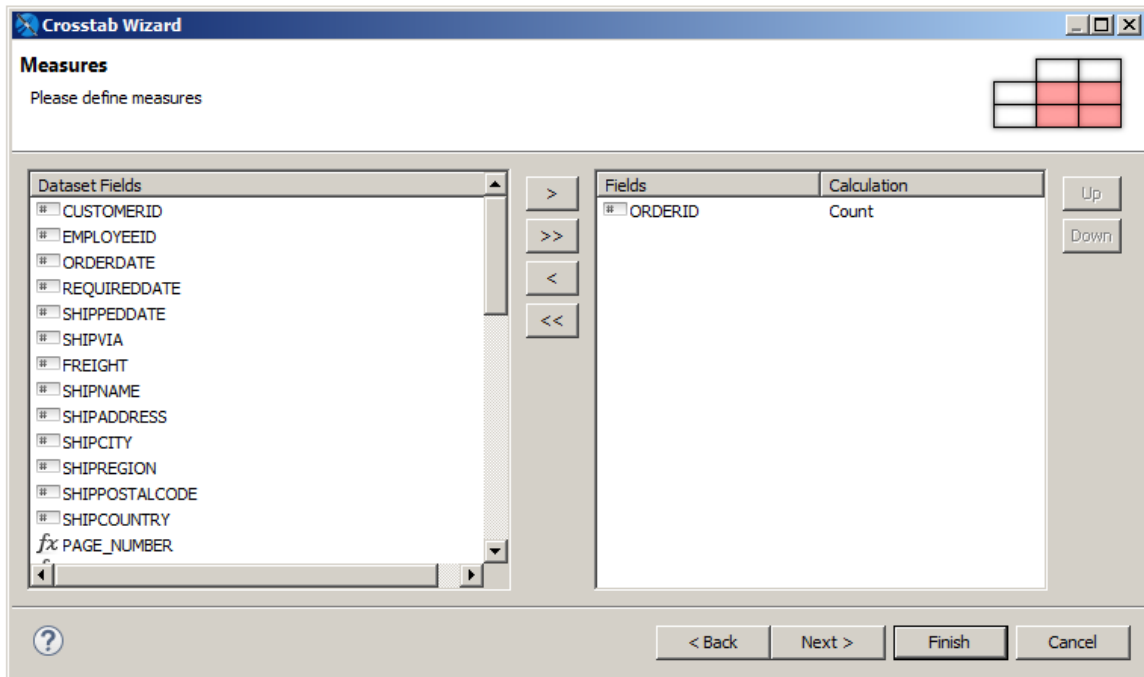


Figure 16-6 Defining measures

13. Click **Next**. The Layout page is displayed.
14. Set options for the crosstab layout. You can indicate whether you want to see grid lines, use color set to distinguish totals, headers, and detail cells, and whether to total the rows and columns.
For this example, select the Burleywood color scheme.

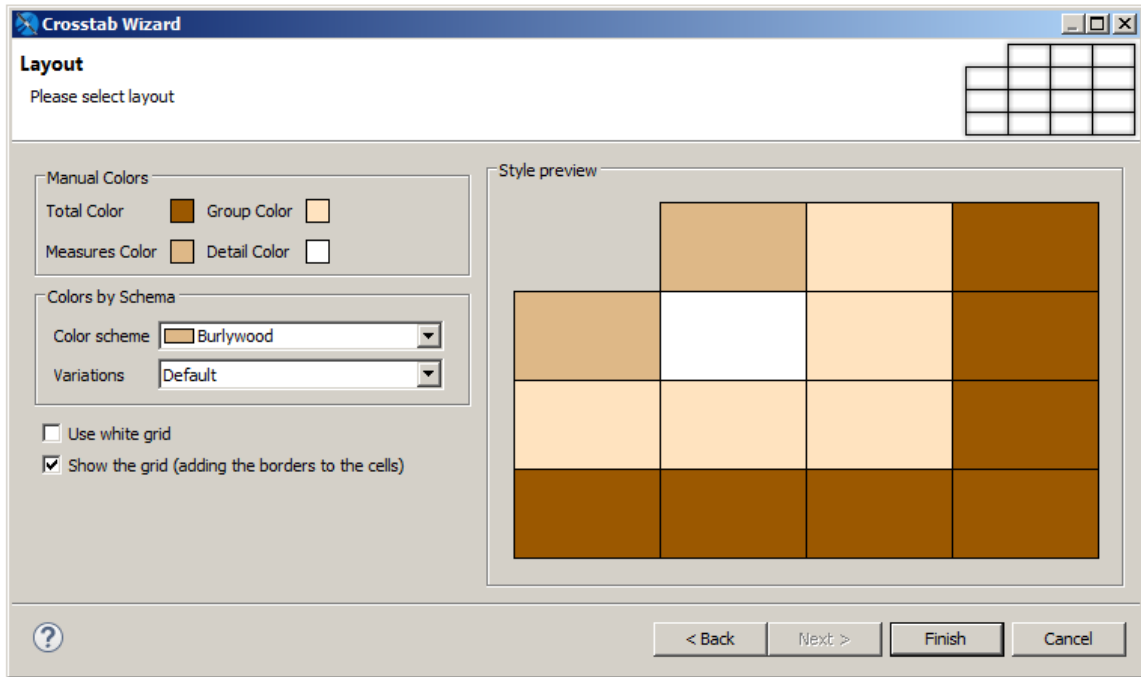


Figure 16-7 Choosing layout options

15. Click **Finish**.

The crosstab is created and added to your report.

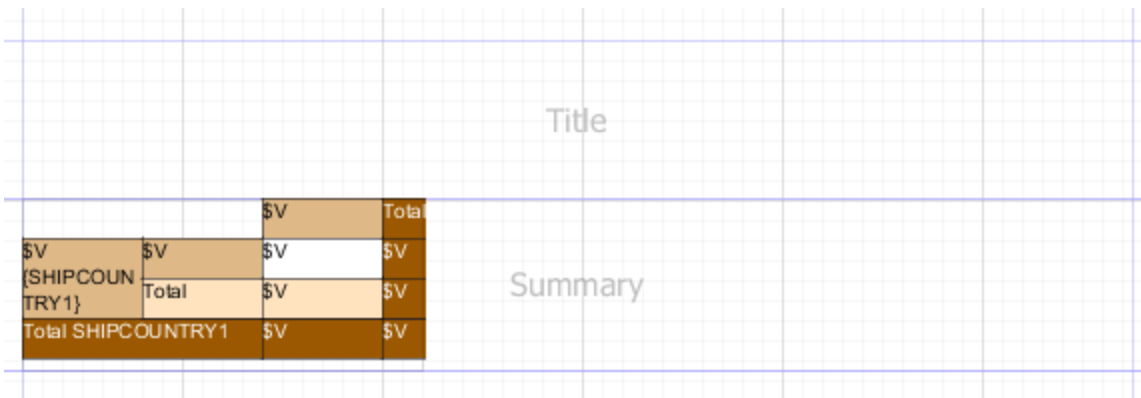


Figure 16-8 Crosstab in Design view

16. Select the crosstab to display a border with handles. Drag the right-hand handle of the crosstab to the right margin of the report.
17. Preview your report. You see a crosstab. The row and column headers are the values of the fields you selected for the rows and columns. For each row and column, the value is the number of orders for that year and postal code.

		1996	1997	1998	Total
Argentina	1010	0	6	10	16
	Total	0	6	10	16
Austria	5020	2	6	2	10
	8010	6	15	9	30
	Total	8	21	11	40
Belgium	B-1180	0	3	4	7
	B-6000	2	4	6	12
	Total	2	7	10	19
Brazil	02389-673	3	5	1	9
	02389-890	2	5	4	11
	04876-786	0	7	2	9
	05422-042	1	2	2	5

Figure 16-9 Preview of the crosstab

16.2 Working with Crosstab Properties

To view or edit crosstab properties, select the crosstab node in the outline view. The properties are displayed in the properties view.

Expressions for elements in a crosstab, such as print-when expressions and text field expressions, can only contain measures. In this context, you cannot use fields, variables, or parameters directly; you always have to use a measure.

You can edit the following crosstab-specific properties on the Crosstab tab in the properties view:

- **Repeat Column Headers** – When selected, the column headers are printed on every page when the crosstab spans additional pages.
- **Repeat Row Headers** – When selected, the row headers are printed on every page when the crosstab spans additional pages.
- **Column Break Offset** – Specifies the vertical space between sections of a crosstab when the crosstab exceeds the page width and two sections appear on the same page.

		1996-07	1996-11	1996-12	1997-01	1997-02	1997-03	1997-04
Argentina	Buenos Aires	0 0.00	0 0.00	0 0.00	1 29.83	1 38.82	0 0.00	0 0.00
	Total in the city	0	0	0	1	1	0	0
Austria	Graz	2 143.38	1 162.33	3 107.70	2 70.84	2 253.36	0 0.00	0 0.00
	Salzburg	0 0.00	1 390.83	0 0.00	1 122.48	0 0.00	1 31.29	1 5.29
	Total in the city	2	2	3	3	2	1	1
	Total	2	2	3	4	3	1	1

COLUMN BREAK OFFSET							
----------------------------	--	--	--	--	--	--	--

		1997-05	1997-09	1997-08	1997-09	1997-10	1997-11	1997-12
Argentina	Buenos Aires	2 12.67	0 0.00	0 0.00	0 0.00	1 22.57	0 0.00	1 1.10
	Total in the city	2	0	0	0	1	0	1
Austria	Graz	1 789.95	2 61.42	1 477.90	1 78.09	1 272.47	0 0.00	3 197.90
	Salzburg	1 339.22	1 35.12	0 0.00	0 0.00	1 96.50	1 117.33	0 0.00
	Total in the city	2	3	1	1	2	1	3
	Total	4	3	1	1	3	1	4

Figure 16-10 Column break offset

16.3 Using the Crosstab Editor

You can edit the fields, expressions, and layout of the crosstab in the crosstab editor. Like the report editor, the crosstab editor has a design view and an outline view. Using the crosstab editor you can:

- Resize rows and columns and format individual cells
- Add and delete row and column groups and edit group properties
- Add, delete, and edit measures
- Edit crosstab totals

To open the crosstab editor:

1. Double-click on the crosstab node in Outline view for the main report.
OR
2. Double-click on the crosstab in Design view for the main report.

When the crosstab editor is selected, a crosstab element appears in outline view. This crosstab element shows the whole crosstab structure, including the crosstab parameters and the row and column groups, measures, and cells.

16.3.1 Formatting Columns, Rows, and Cells

Manually resizing a row or column:

1. Open the crosstab editor.
2. Shift-click in the header of the row or column you want to change.
The row and column you selected are outlined.

	\$V	Total	
\$V	\$V	\$V	\$V
{SHIPCOUNTRY1}	Total	\$V	\$V
Total SHIPCOUNTRY1	\$V	\$V	

Figure 16-11 Row and column selected in crosstab editor

3. Drag an outline to resize the row or column. Make sure that the cells are large enough to completely contain their content when you run the report.

16.3.1.1 Working with Cells

Each intersection between a row and a column defines a cell. Crosstabs have header cells, total cells, detail cells, and (optional) when-no-data cells. Each cell can contain one or more elements that do not use a dataset, such as text fields, static text, rectangles, and images. Cells can't contain subreports, charts, or another crosstab.

Changing cell borders:

1. Open the crosstab editor and select the cell in editor or in outline view.
2. Edit the cell borders on the Borders tab of the property view.

16.3.2 Editing Row or Column Group Properties

You can edit the following properties for a row or column group:

- **Name** – Name of the group. Renaming a group using the Properties dialog renames it everywhere the group is used.
- **Total Position** – Location of the row or column that shows subtotals. Values are None, Start, End (default).
- **Order** – Order of the values in the group (Ascending or Descending).
- **Order By Expression** – Optional expression to use for ordering the values.
- **Comparator Expression** – Optional instance of `java.util.Comparator` to use for ordering the values. If no expression is present, the default ordering for the data type is used (for example, numeric or alphabetic ordering). **Expression** – Bucket expression used to group the rows or columns. The default is to group by field value, for example, `$F{SHIPPOSTALCODE}`.
- **Value Class Name** – Field type.

Editing the expression of a group:

The following example shows how to edit the sample crosstab to group by the first letter or digit of the postal code:

1. Double-click the crosstab to open the crosstab editor.

- In Outline view, select the group you want to edit. For this example, select **Crosstab > Row Groups > SHIPPOSTALCODE1**.

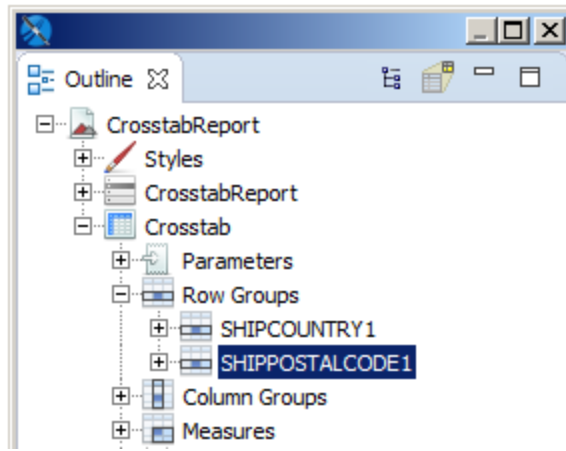


Figure 16-12 Outline tree view - crosstab details in the crosstab editor

- In Properties view, select the Cell tab.
- Change the expression in the **Expression** entry bar to `$(SHIPPOSTALCODE).substring(0,1)`.

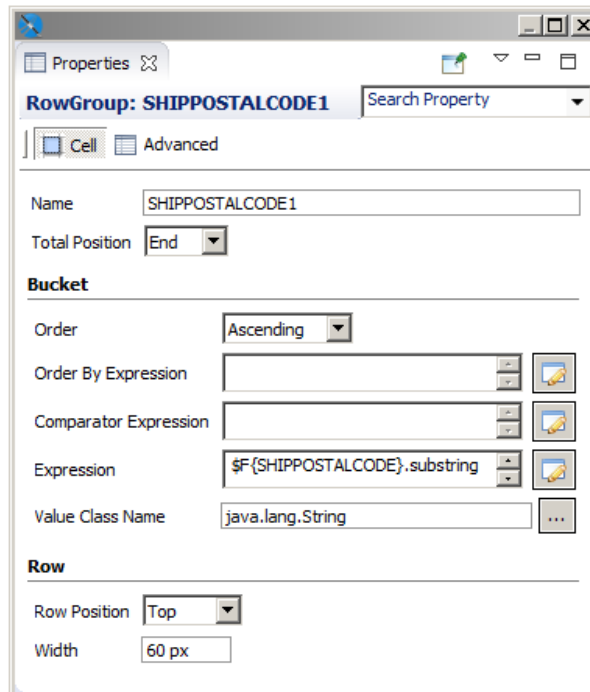


Figure 16-13 Properties for SHIPPOSTALCODE1

When you preview the crosstab, the second row group is now bucketed by the first character of the postal code.

		1996	1997	1998	Total
Argentina	T	0	6	10	16
	Total	0	6	10	16
Austria	S	2	6	2	10
	B	6	15	9	30
	Total	8	21	11	40
Belgium	B	2	7	10	19
	Total	2	7	10	19
Brazil	O	13	42	28	83
	Total	13	42	28	83
Canada	H	3	10	0	13
	T	1	5	8	14
	V	0	2	1	3
	Total	4	17	9	30
Denmark	T	1	5	1	7

Figure 16-14 Crosstab after expression has been edited

16.3.3 Adding and Deleting Row and Column Groups

A crosstab must have at least one row group and one column group. It is easiest to add all the rows and columns you want when you create the crosstab with the Crosstab Wizard. However, if necessary, you can add a row or column group manually.

The following example shows how to add a row group, SHIPREGION, to the example crosstab. Adding a column is similar.

Example of adding a row group:

- 1. Double-click on the crosstab to open the crosstab editor.
- 2. In outline view, double-click the Crosstab node to expand it.
- 3. Right-click the Row Groups node and select **Create Row Group** from the context menu.

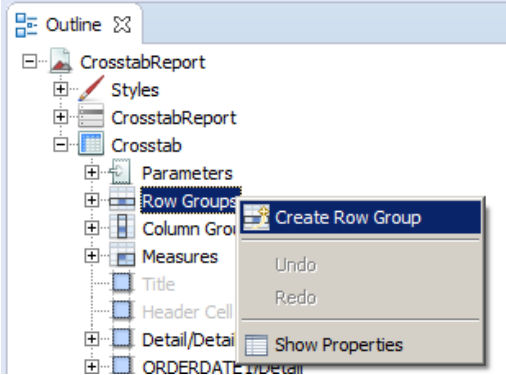


Figure 16-15 Adding a row group

The Group Band dialog is displayed.

4. Enter the information for your group in the Group Band dialog. For this example:
 - a. Enter SHIPREGION1 for the **Group Name**.
 - b. Select **Create Group from a report object** and select SHIPREGION.
 - c. Click **Finish**.

The new group is added to the crosstab as the innermost row group.

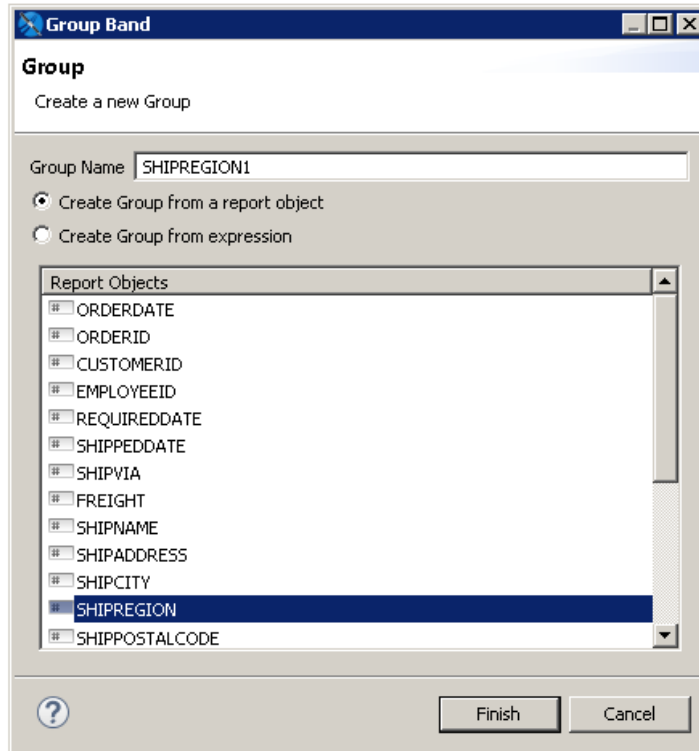


Figure 16-16 Group Band dialog

5. To set the value class of the group, select the top-level node of the new SHIPREGION group in outline view of the crosstab editor. Then, in the Cell tab of the properties view, enter the following value:
 - **Value Class Name** – `java.lang.String`

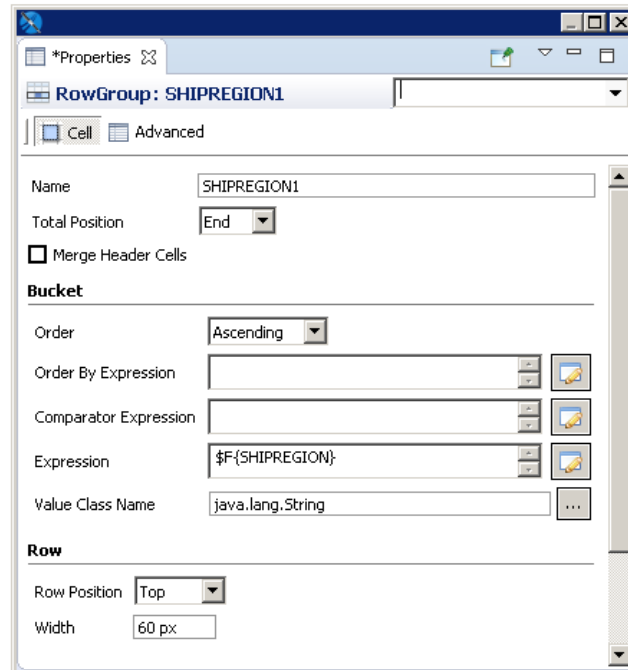


Figure 16-17 Setting Value Class Name of a row group

6. Change the order of the groups by selecting the top-level node of SHIPREGION in outline view and dragging it above SHIPPOSTALCODE.
7. Preview the report.

Deleting a row or column group:

1. Double-click on the crosstab to open the crosstab editor.
2. In outline view, double-click the Crosstab node to expand it.
3. Double-click the Row or Column Groups node to expand it.
4. Right-click on the row or column group you want to delete and select **Delete** from the menu.

16.3.4 Working with Measures

A measure in a crosstab is an object, similar to a variable, that appears in an individual cell. It is the result of a calculation performed on the values for each row and column group that intersect in a cell.


A crosstab can have multiple measures. If you add multiple measures when you first create a crosstab, each measure shows up under the Measure node in the outline view of the crosstab editor. You can also add measures after the crosstab has been created by dragging a text field into a measure cell in your crosstab and setting an expression. In this case, the measure is only visible in the Detail node of the outline view. For an example of adding a measure and setting its expression, see [16.3.4.3, “Adding a Measure as a Text Field,” on page 265](#).



Expressions for elements in a crosstab, such as print-when expressions and text field expressions, can only contain measures. In this context, you cannot use fields, variables, or parameters directly; you always have to use a measure.

16.3.4.1 Measure Properties

The following properties are available for measures that you added when you first created the crosstab:

- **Name** – Name of the measure.
- **Calculation** – Calculation to use for the measure. See 7.3.2, “Calculation Function,” on page 89 for more information.
- **Percentage of Type** – Set this to **Grand Total** to display your measure as a percentage of the grand total.
- **Value Expression** – Expression to use for calculating the measure. To edit this expression, click .
- **Value Class** – Java class to use for the expression.
- **Incrementer Factory Class Name** – Optional custom `Incrementer` class. Use this to implement your own calculation if the available calculation types are not sufficient. Class must be instantiated via a factory that implements the `net.sf.jasperreports.engine.fil.JRIncrementerFactory` interface .
- **Percentage Calculation Class Name** – Optional custom calculator class to perform the percentage calculation. Must use the `net.sf.jasperreports.crosstabs.fill.JRPercentageCalculator` interface.

To display measure properties:

1. Double-click the crosstab to open the crosstab editor.
2. Expand the Crosstab node in outline view.
3. Expand the Measures node.
4. Right-click the measure you want and select **Show Properties** from the menu.

16.3.4.2 Understanding Crosstab Total Variables

When you have multiple row or column groups, you can use crosstab total variables to combine data at different aggregation levels (for example, to calculate a percentage). The following built-in variables are available:

- `<Measure>_<Column Group>_ALL`: The total of all the entries in the specified column group and the current row.
- `<Measure>_<Row Group>_ALL`: The total of all the entries in the specified row group and the current column.
- `<Measure>_<Row Group>_<Column Group>_ALL`: The combined total of all the entries in the specified row and column groups.

You can also select these variables from the expression editor for the Expression field on the Text Field tab of the Properties view for a measure.

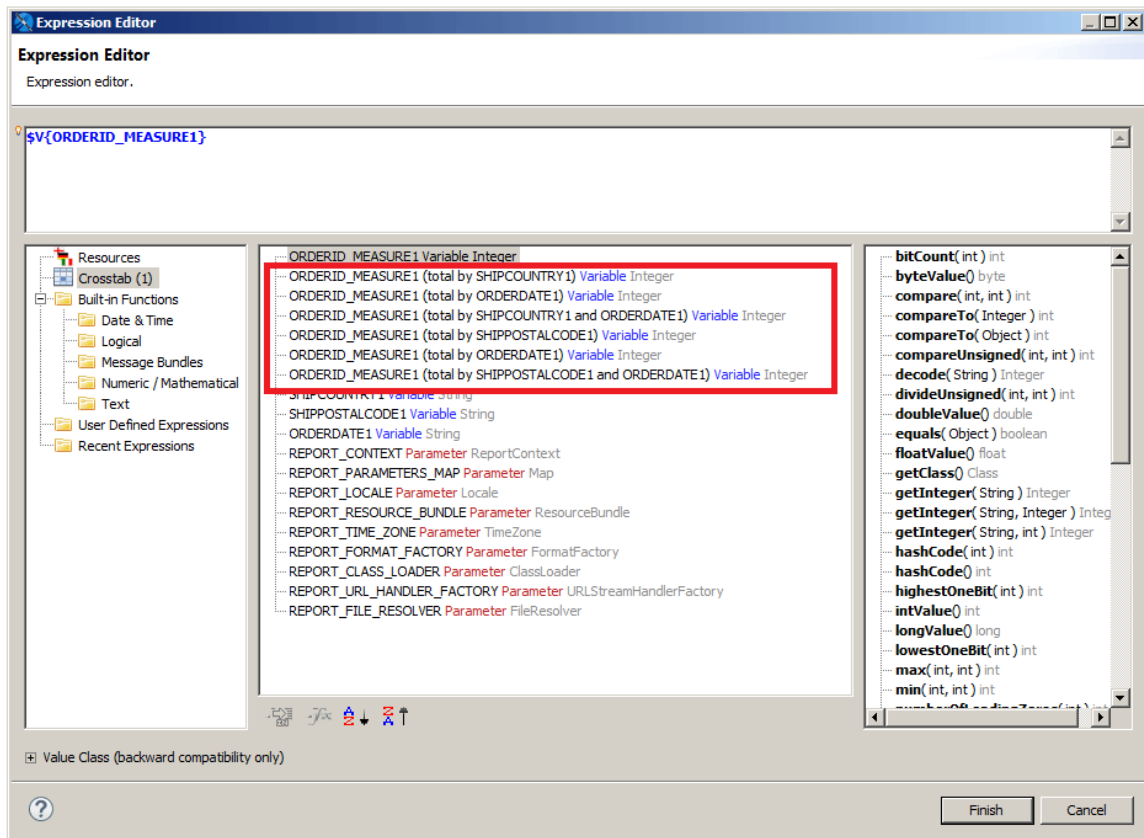




Figure 16-18 Total variables in the expression editor

16.3.4.3 Adding a Measure as a Text Field

This example shows how to add a measure to an existing crosstab using a text field. This example uses crosstab total variables to calculate a percentage. However, measures added as text fields do not have measure properties such as a calculation or an incremter calculation class.

Adding a measure:

1. Create a new report:
 - a. Choose a blank template.
 - b. Select the Sample DB data adapter and click **Next**.
 - c. Enter the query `select * from orders` and click **Next**.
 - d. On the Fields page, select all fields and click **Finish**.
2. Delete all bands except the Summary band. This eliminates blank pages in the final report.
3. Add a crosstab  to the Summary band with the following settings:
 - a. Dataset – **[Main Dataset]**.
 - b. Column group – ORDERDATE; select **Year** from the drop-down menu in the **Calculation** column.
 - c. Row group – SHIPCOUNTRY.
 - d. Measure – ORDERID.
4. In design view for the report, double-click on the crosstab to open the crosstab editor.

5. Shift-click in the second row and drag to expand the row height.
6. Drag a text field  into the intersection of the first row and column. The text field is added to the column.

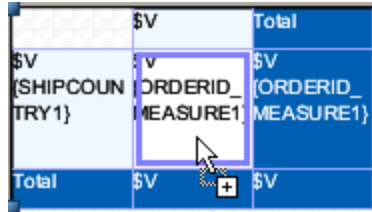


Figure 16-19 Adding a text field to an existing measure

Setting the measure expression:

7. Select the text field you added.
8. Select the Text Field tab in Properties view.

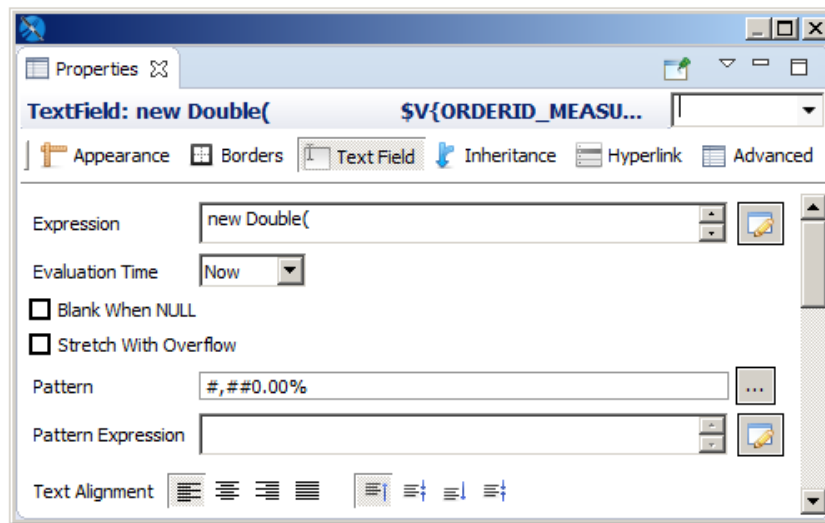



Figure 16-20 Text field properties after setting the expression

9. Click  to the right of the **Expression** field to open the expression editor.
10. Add a formula to calculate the following percentage:
(Number of orders placed in this country and in this year) / (All orders placed in this country)
For Java, use the following expression:

```
new Double(
    ${ORDERID_MEASURE1}.doubleValue()
    /
    ${ORDERID_MEASURE1_ORDERDATE1_ALL}.doubleValue()
)
```

For Groovy, use the following expression:

```
(double) $V{ORDERID_MEASURE1} / (double) $V{ORDERID_MEASURE1_ORDERDATE1_ALL}
```



A percentage must be treated as a floating-point number. For this reason, extract the double-scalar values from `ORDERID_MEASURE1` and `ORDERID_MEASURE1_ORDERDATE1_ALL` objects even if they are objects of class-type `Integer`.

11. Click **Finish** to close the expression editor.
12. Enter #,##0.00% in the **Pattern** field to format the result as a percentage.
13. Click Preview to run the report.

	1996	1997	1998	Total
Argentina	0 0.00%	6 37.50%	10 62.50%	16
Austria	8 20.00%	21 52.50%	11 27.50%	40
Belgium	2 10.53%	7 36.84%	10 52.63%	19
Brazil	13 15.66%	42 50.60%	28 33.73%	83
Canada	4	17	9	30

Figure 16-21 The final report with percentages included

16.4 Working with Crosstab Parameters

Crosstab parameters let you pass dynamic values from the main report to the crosstab as crosstab parameters. They can be used in the expressions of elements displayed in the crosstab. Crosstab parameters are defined and managed using outline view in the crosstab editor.



Crosstab parameters are designed to be used in crosstab elements. They are not the same as the dataset parameters that are used in expressions, in the crosstab context, to filter a query and calculate values.

To add a crosstab parameter:


1. Double-click your crosstab in design view to open the crosstab editor.
2. In outline view, right-click the Parameters node in the Crosstab element and select **Create Parameter**.
3. To set the value of the crosstab parameter, double-click the parameter to open the expression editor. Create an expression for your parameter and click OK.

You can use a map to set the value of the declared crosstab parameters at run time. In this case, you'll need to provide a valid parameters map expression in the crosstab properties.

CHAPTER 17 WORKING WITH THE MAP COMPONENT

The Map element in the Palette view lets you add Google Maps to your reports. You can set the center, zoom, and scale for your map, as well as markers and paths. The Properties view for a map element has tabs to control appearance and map properties, set authentication for Google business license, and create markers and paths.

To add a Map component to your report:

1. Drag the Map component  from the Palette to your report. Usually, you want to add the map to a component that is included only once, such as the Title band or Summary band.

This topic contains the following sections:

- [Working with Map Properties](#)
- [Viewing Authentication Properties](#)
- [Working with Markers](#)
- [Working with Paths](#)
- [Properties for Markers and Paths](#)

17.1 Working with Map Properties

The Map tab in the Properties view lets you set the basic properties for the map:

1. Select a map component in your report and click Map in the Properties view.

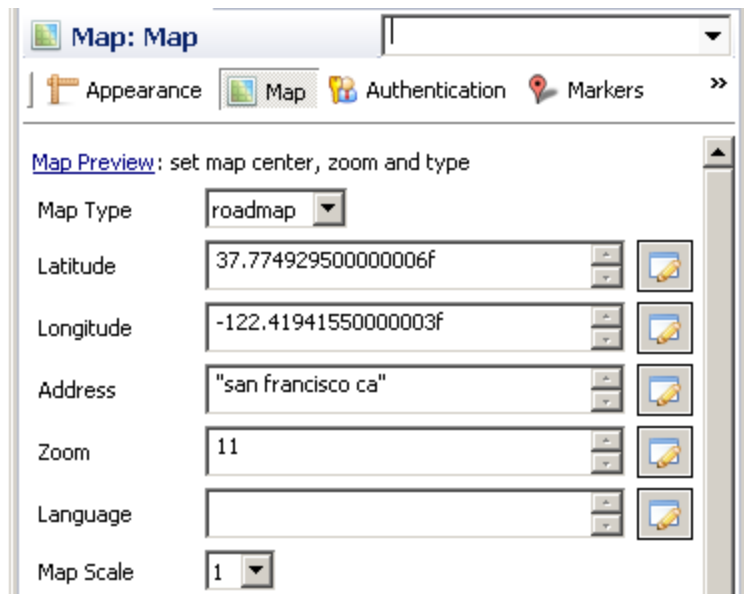


Figure 17-1 Map tab in Map Properties

You can set the following map properties using the Map tab:

- **Map Preview** – Opens a Google Maps window. This window supports standard Google Maps functionality, such as dragging, zooming, and switching between Map and Satellite views.

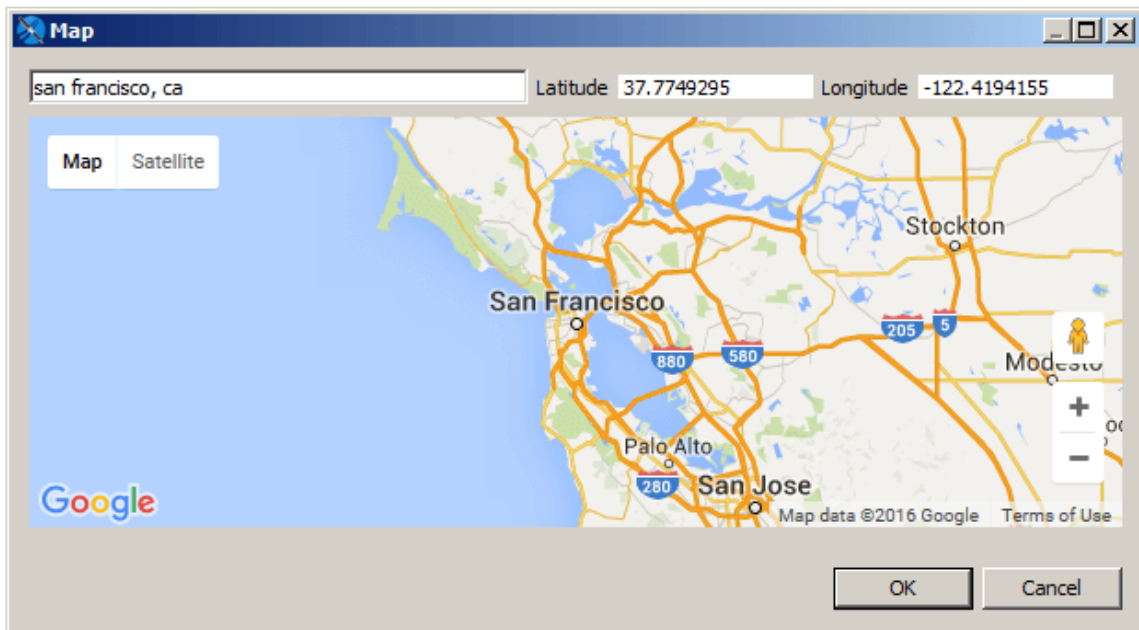







Figure 17-2 Setting a map location

Changes to this window are reflected in the map in your report. In addition, you can change the map's center in any of the following ways. When you close the preview, the map is automatically centered at the selected location:

- **Address** – Enter an address in the entry bar to center the map at that location.
- **Latitude** and **Longitude** – Enter a latitude and longitude to center your map at that location.
- **Double-click** – Double-click anywhere on the map to center it at that location.
- **Map Type** – The Google Maps view. Options are: roadmap, satellite, terrain, and hybrid.
- **Latitude** – The latitude of the map center. You can type directly in the entry bar, or click  to enter an expression.
- **Longitude** – The longitude of the map center. You can type directly in the entry bar, or click  to enter an expression.
- **Address** – A String representing the address of the center. You can type directly in the entry bar, or click  to enter an expression. Must be enclosed in quotes, for example, "350 Rhode Island Ave., San Francisco, CA".
- **Zoom** – Integer representing the Google Maps zoom level. You can type directly in the entry bar, or click  to enter an expression.
- **Language** – String that sets the in-map language. You can type directly in the entry bar, or click  to enter an expression. Must be enclosed in quotes, for example, "ru-RU". See the Google Maps documentation for more information.
- **Map Scale** – Sets the size of the scale bar at the bottom of the map.
- **Evaluation Time** – Drop-down that lets you set the evaluation time of the map. See [7.3.1, “Evaluation Time,” on page 88](#) for more information.
- **Image Type** – Drop-down that lets you set the image type to use when the map is embedded in your report.
- **On Error Type** – Drop-down that lets you set the type of message to display when there is an error with the map.

17.2 Viewing Authentication Properties

If you want to use a Google Maps key or business client license, we recommend that you configure these as global Jaspersoft Studio properties. You can view the status of your Google Maps license information on the Authentication tab.

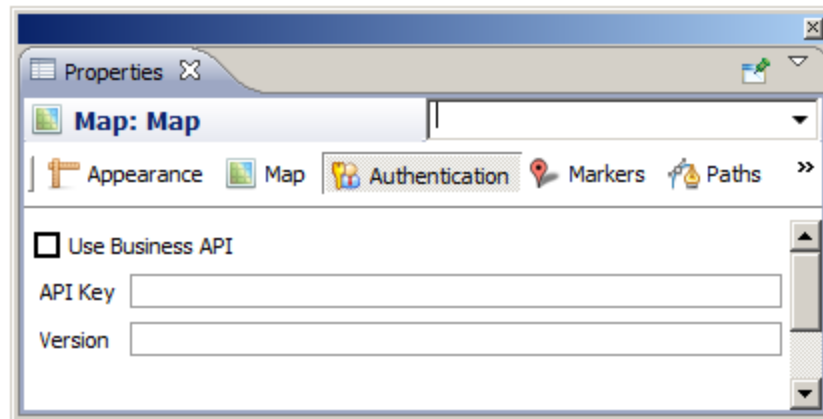


Figure 17-3 Authentication tab in Properties view for a map component

To configure your Google Maps license and/or version information:

1. Select **Window > Preferences** to open the Preferences dialog box.
2. Navigate to Jaspersoft Studio > Properties.
3. To configure a property, click **Add** to open the Properties dialog, enter the name of the property and the property's value, then click **OK**. You can configure the following Google Maps APIs properties. See the JasperReports Library configuration reference for more information on each property:
 - `net.sf.jasperreports.components.map.client.id` – Specifies the client ID for Google Maps API for Business. If set, it takes precedence over the API key property. Usually works along with the signature property for signed URLs.
 - `net.sf.jasperreports.components.map.key` – Specifies the Google Maps API key.
 - `net.sf.jasperreports.components.map.signature` – Specifies the encrypted client signature for signed request URLs.
 - `net.sf.jasperreports.components.map.version` – Indicates which version of the Google Maps API should be loaded.
4. When you have specified all your properties, click **OK** to exit the Preferences dialog box.



Setting the property globally sets the properties when the report is run inside Jaspersoft Studio. If you are publishing your reports to another environment, you must enable these properties in the `jasperreports.properties` file in your environment.

17.3 Working with Markers

A marker identifies a location on a map. You can create markers manually, either using a fixed location that is known when the report is created, or using an expression based on report data. You can also define markers based on a dataset. A single map can include both manual markers and markers from one or more datasets. This section describes:

- **Marker Properties**
- **Adding Markers Manually**
- **Adding Markers Using the Map**
- **Adding Markers Using a Dataset**

- **Modifying Markers**

17.3.1 Marker Properties

You can set properties for each marker. The marker properties available are a subset of Google Maps' properties. See [Table 17-5, “Marker and Path Properties,” on page 285](#) for more information.

17.3.2 Adding Markers Manually

Manually-added markers can be used for a fixed address or location that is known when the report is created. You can also use an expression, for example to set a location based on a parameter value. This method only displays as many markers as you explicitly create.

To manually define a marker:

1. Open or create a report and add a map component. Make sure to set the map's center to a location near your marker. For this example, use the following coordinates:
Latitude – 37.7656842
Longitude – -122.403
2. With the map component selected, click the Markers tab in the Properties view.

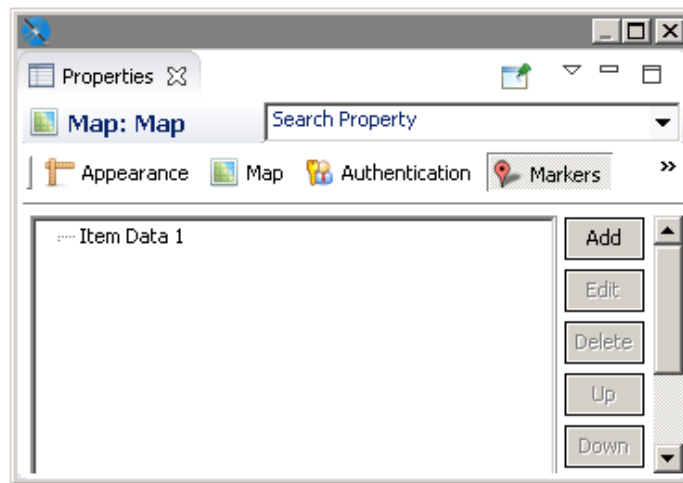


Table 17-1 Markers tab with one marker

3. To specify the marker properties, click **Add** in the Markers tab.
The Markers dialog box opens.
4. To enter an individual marker, select the Markers tab and click **Add** again.
The Marker dialog box opens.

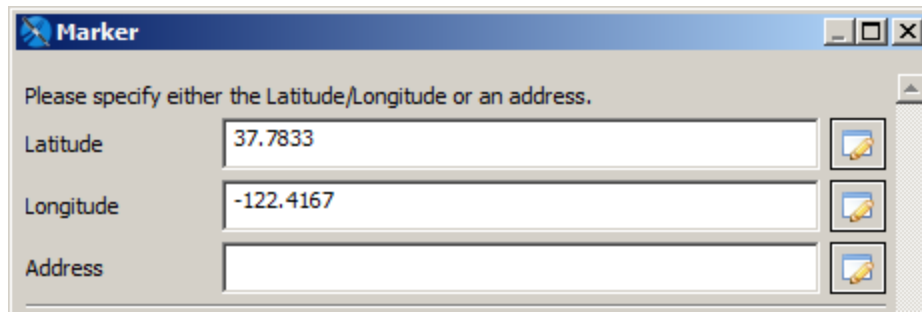




Figure 17-4 Defining a static marker

5. Specify a location for your marker. You can do this by entering latitude and longitude, entering an address, or defining markers on the map preview:
 - **Latitude and Longitude** – Enter the latitude and longitude coordinates for your marker. You can type directly in the entry bar, or click  to enter an expression. For this example, enter the following values:
 - **Latitude** – 37.833
 - **Longitude** – -122.4167
 - **Address** – The address is used only if **Latitude** and **Longitude** are blank. You can type directly in the entry bar, or click  to enter an expression.
6. (Optional) Set the title for your marker, if any.
7. (Optional) To have a new browser window or tab open with related information when a user clicks on the marker, enter the URL and select the Target type.
8. (Optional) Set your icon type (default or custom) and icon properties:
 - If you are using the default marker, you can set additional properties, such as color, label, etc. These properties are not available for a custom icon. This example uses the color 00CCFF and the label J.

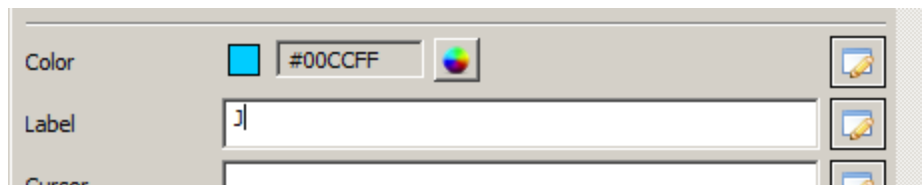


Figure 17-5 Setting color and label for a marker

- To use a marker icon other than the default, click **Custom Icon** to specify a URL that points to the image to use. Currently, we don't support loading an image directly from the repository or as a resource local to the report. Instead, the JavaScript API loads the icon from the URL. Then set additional optional properties for your marker, such as icon height, width, origin, and anchor.
9. Click **OK** to return to the Markers dialog box.
 10. To create additional markers, click **Add**, enter the marker properties, then click **OK** to return to the Markers dialog box.
 11. Click **OK** to create your markers.
 12. Once you have defined your markers, preview your report in HTML. For the example, select the Empty Data Set for your preview.



Figure 17-6 A map with a marker

17.3.3 Adding Markers Using the Map

You can also add markers using the map preview. This only supports a fixed address or location that is known when the report is created.

To manually define a marker using the map:

1. Open or create a report and add a map component. Make sure to set the map's center to a location near your marker. For this example, use the following coordinates:
Latitude – 37.7656842
Longitude – -122.403
2. With the map component selected, click the Markers tab in the Properties view.

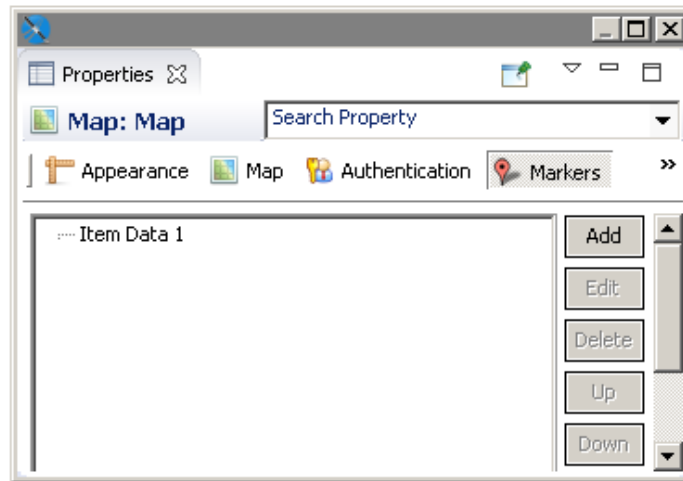


Table 17-2 Markers tab with one marker

3. To specify the marker properties, click **Add** in the Markers tab.
The Markers dialog box opens.
4. To enter an individual marker, select the Map tab. You can perform the following tasks:
 - To create a marker by selecting a location on the map, right-click on the location you want and select **Add marker**.
 - To delete one or more markers, select the marker(s) in the panel at the right and press **Delete**, or right-click on the marker and select **Delete**.
 - To edit a marker's location, double-click on the marker to open the Marker dialog box.

17.3.4 Adding Markers Using a Dataset

The steps above define a fixed number of markers. You can also dynamically define the markers based on locations defined in your report's data or another dataset. A single map can include both manual markers and markers from one or more datasets.


17.3.4.1 Sample Data

In this example, we'll use a CSV file containing the following data for San Francisco landmarks. This file includes data used by markers and paths.

Table 17-3 Sample CSV Data for Markers and Paths

```
landmark, latitude, longitude, path, style
Fiserman's Wharf, 37.8085636, -122.409714, path1, style1
Golden Gate Bridge, , , path1, style1
Cliff House, 37.778485, -122.513963, path1, style1
Stern Grove, 37.7358667, -122.4771518, path1, style1
Stern Grove, 37.7358667, -122.4771518, path2, style2
Golden Gate Park, , , path2, style2
Union Square, 37.788527, -122.407235, path2, style2
"Willie Mays Plaza, San Francisco, CA", 37.778595, -122.38927, path1, style1
Twin Peaks, 37.7544066, -122.4476845, path2, style2
```


Define the San Francisco data adapter:

1. Create a data file with the path data you want. For this example, create a CSV file with the data provided above. Make sure to include a blank line at the end of the file.
2. Click  on the main toolbar. When prompted, navigate to the same folder as your report.
3. Name the file SFDDataAdapter.xml and click **Next**.
4. Select **CSV File** as the data adapter type and click **Next**. The CSV File dialog opens.

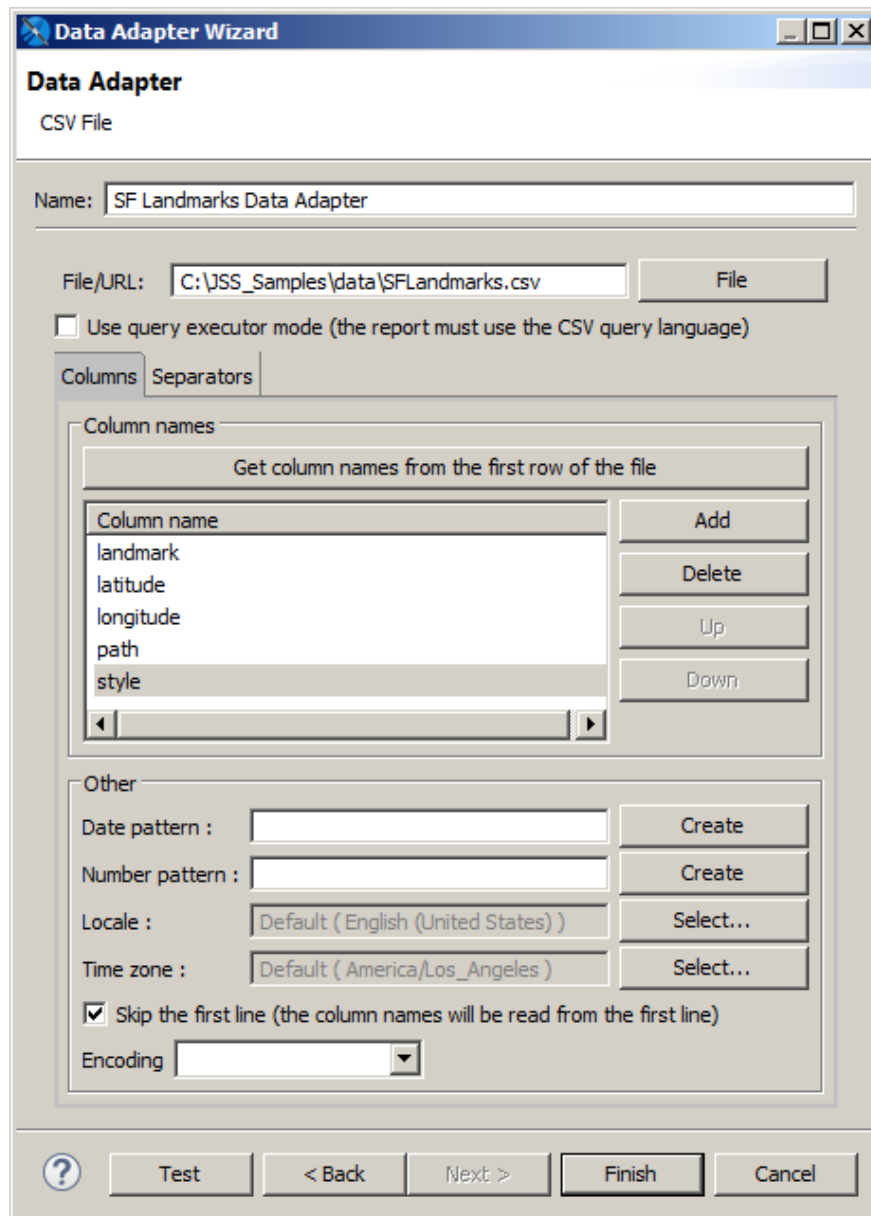


Figure 17-7 Creating a sample data adapter for markers and paths

5. Name your adapter, for example, SF Landmarks Data Adapter.
6. Click File and select the CSV file you created.

7. Click **Get column names from the first row of the file**.
8. Select **Skip the first line**.
9. Click **Finish** to create the adapter.

Create a dataset in your report:

1. Right-click the root node of the report in the outline view and select **Create Dataset**.
2. Name the dataset SFLandmarksDataset, make sure **Create new dataset from a connection or Data Source** is selected, and click **Next**.
3. Select the SFDataAdapter.xml data adapter and click **Next**.
4. Click **>>** to select all fields and click **Finish**.

The dataset is created in your report.

5. Click on the SFLandmarksDataset in outline view.
6. In the Properties view, enter SFDataAdapter.xml in the **Default Data Adapter** entry box. Setting the default data adapter lets you use a different dataset from the one used in the main report. See [10.2.3, "Default Data Adapter ,"](#) on page 123 for more information.

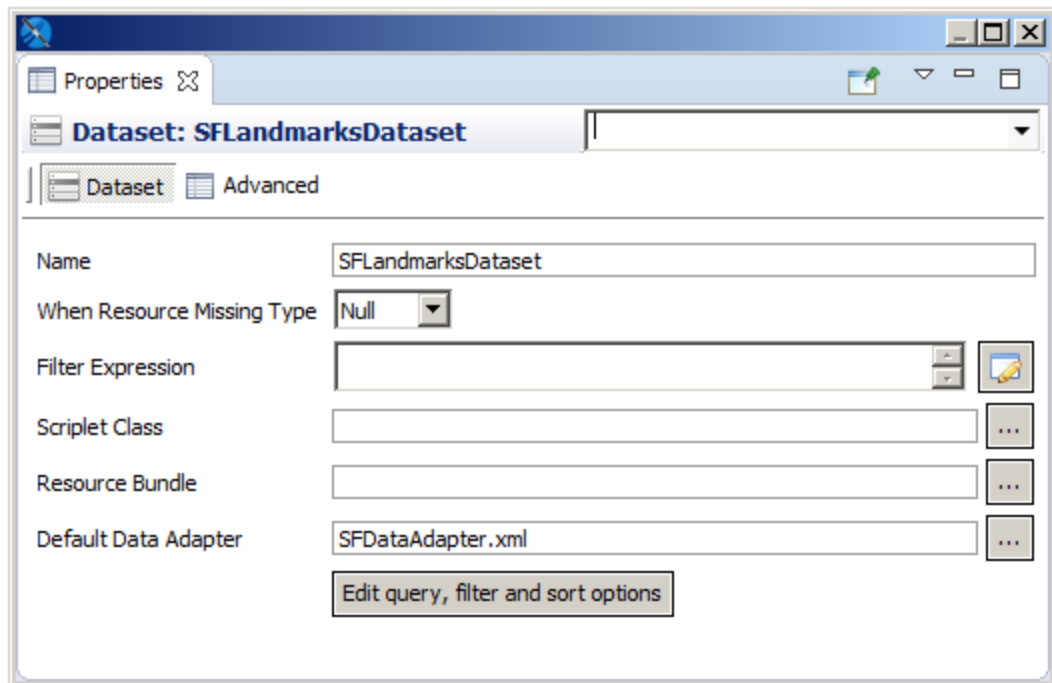


Figure 17-8 Setting the default data adapter for a dataset

17.3.4.2 Using the dataset to set markers

1. Add a map component to the report, or select an existing map in the Design tab.
2. If you have not set the map center or zoom level, do so. For this example, click the Map tab in the Properties view, and use the map preview to select "San Francisco, CA" as the center. Then enter 11 in the **Zoom** field.
3. Click the **Markers** tab in the Properties view.
4. Click **Add** to open the Markers dialog box.

5. Select the **Dataset** tab and select **Use Dataset**.
6. In the Dataset Run section, select your dataset and accept the default settings. For this example, use **SFLandmarksDataset**. You have already set the default data adapter for this dataset.
7. Click the Markers tab and then click **Add**. The Marker dialog box opens.

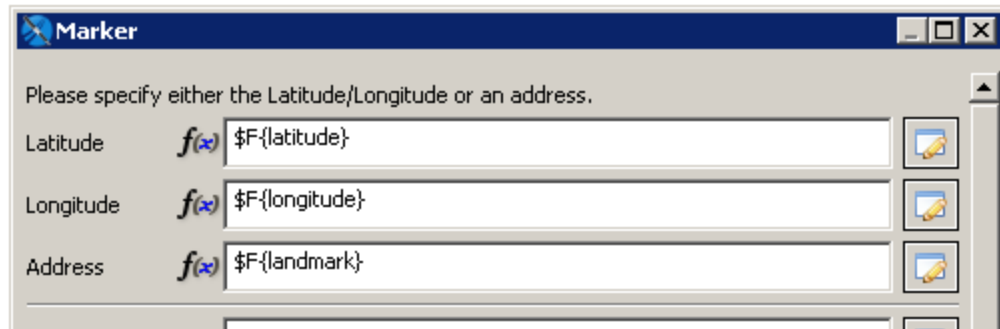



Figure 17-9 Using expressions to set markers from a dataset

8. For a dataset, you typically want to use expressions for your values. For each property you want to read from the dataset, click  on the entry bar, select **Use Expression** and enter the expression to use. For this example, use the following expressions:
 - Latitude – `$F{latitude}`
 - Longitude – `$F{longitude}`
 - Address – `$F{landmark}`



You can use expressions to pass parameters to a map component dynamically. Expressions allow you to evaluate data in your dataset and use the results to populate the map. In the component's properties, properties based on expressions show $f(x)$ next to the field.

9. Click **OK**. The Markers dialog box displays the markers you just created.

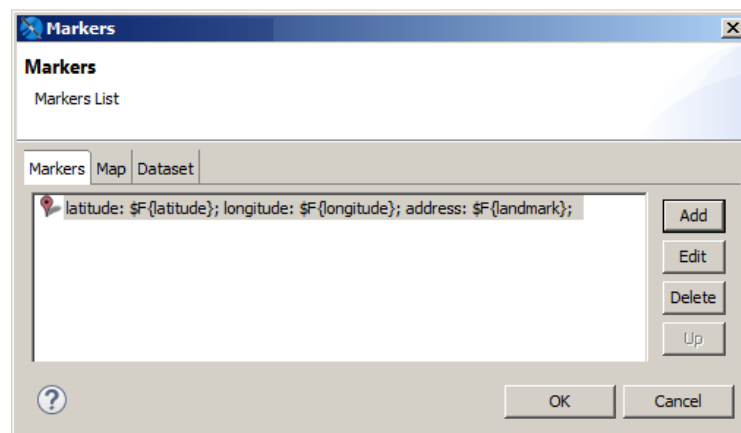


Figure 17-10 Item data for markers created from a dataset

10. Click **OK**. Your markers are displayed on the Marker tab of the Properties view, along with any other markers you have created.

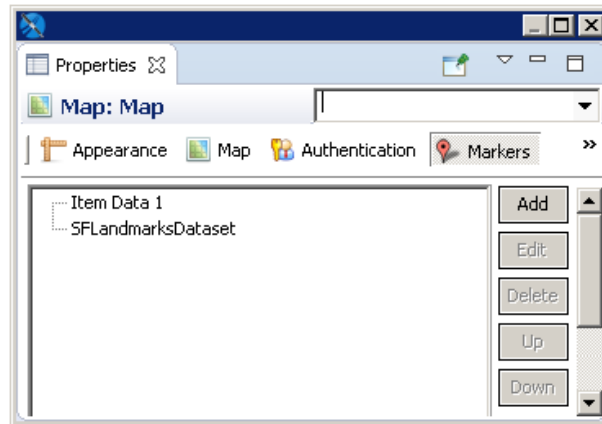


Figure 17-11 Properties view showing markers added manually and markers defined from a dataset

11. Preview your report in HTML. The example below shows the markers from the sample dataset along with a static marker.

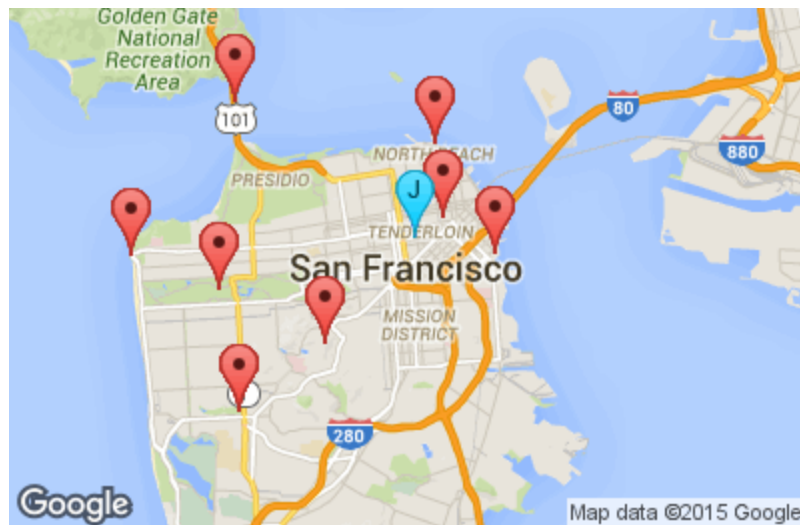


Figure 17-12 San Francisco landmarks shown on a map

17.3.5 Modifying Markers

To edit a marker:

1. Select the map and click on the Markers tab in Properties view.
2. Select the marker you want to change and click **Edit**.

3. To change the dataset, make sure you have set up another dataset in your report before editing the marker. Then you can select the Dataset tab here and select the new dataset from the **Dataset Run** menu.
4. To change marker properties, select the Markers tab, and edit your properties.

To delete a marker:

1. Select the map and click on the Markers tab in Properties view.
2. Select the marker you want to delete and click **Delete**.

17.4 Working with Paths

You can add one or more paths to your maps. A path is defined by:

- A name that serves as a path identifier; the name must be unique in your report
- A collection of places (points) on the map defined by latitude/longitude coordinates or addresses; these are connected to form the path
- (Optional) A style that specifies various style configuration properties, such as line and fill color, line weight, and opacity

17.4.1 Defining Path Styles

A path style specifies the properties (for example, color and weight) of the lines between the points on your path. See [Table 17-5, “Marker and Path Properties,” on page 285](#) for more information. You can create a path style manually, or you can save your path styles as a dataset.

17.4.1.1 Defining Path Styles Manually


1. Edit the map component's properties.
2. On the Paths tab, in the Styles section, click **Add**.
3. In the Style dialog box, enter the properties you want for the path. See [Table 17-5, “Marker and Path Properties,” on page 285](#) for more information about available properties.
4. Click **OK**.

17.4.1.2 Defining Path Styles Using a Dataset

Table 17-4 Sample CSV Data for Path Styles

```
name, strokecolor, strokeopacity, strokeweight, fillcolor, fillopacity, ispolygon
"style1", "#0000FF", 0.6, 1, "#FF33FF", 0.4, true
"style2", "#FF0000", 0.8, 2, , , false
```

Create a data adapter for your path styles:


1. Create a data file with the path data you want. For this example, create a CSV file with the data provided above. Make sure to include a blank line at the end of the file.
2. Click  on the main toolbar.
3. When prompted, navigate to the same folder as your report.
4. For this example, name the file PathStylesDataAdapter.xml and click **Next**.
5. Select **CSV File** as the data adapter type and click **Next**.
6. Name your adapter, for example, Path Styles Adapter.

7. Click **File** and select the CSV file you created.
8. For this example, click **Get column names from the first row of the file** and select **Skip the first line**.
9. Click **Finish** to create the adapter.

Create a dataset in your report:

1. Right-click the root in the outline view and select **Create Dataset**.
2. Name the dataset and click **Next**. For this example, name the dataset PathStyles.
3. Select the data adapter for your path styles (Path Styles Data Adapter) and click **Next**.
4. Click **>>** to select all fields and click **Finish**.
5. Select the dataset (PathStyles) you just created in outline view.
6. In the Properties view, enter the filename of the data adapter (PathStylesDataAdapter.xml) in the Default Data Adapter entry box. Setting the default data adapter lets you use a different dataset from the one used in the main report. See [10.2.3, “Default Data Adapter ,” on page 123](#) for more information.

Define a style using a dataset:

1. Create your data source, a data adapter that points to it, and a dataset that uses the data adapter.
2. Add a map component to the report, or select an existing map in the Design tab.
3. Select the Paths tab in the Properties view.
4. In the Styles section, click **Add** to open the Items dialog box.
5. Click the **Dataset** tab in the Path dialog box and select **Use Dataset**.
6. In the Dataset Run section, select your styles dataset (PathStyles) and accept the default settings. You have already set the default data adapter for this dataset.
7. Select the Items tab and click **Add** to open the Style dialog box.
8. For each property you want to read from the dataset, click  on the entry bar, select **Use Expression** and enter the expression to use. For this example, use the following expressions:
 - Name – `#{name}`
 - Stroke Color – `#{strokecolor}`
 - Stroke Opacity – `#{strokeopacity}`
 - Stroke Weight – `#{strokeweight}`
 - Fill Color – `#{fillcolor}`
 - Fill Opacity – `#{filloppacity}`
 - Is Polygon – `#{ispolygon}`
9. Click **OK** to return to the Items dialog box.
10. Click **OK** to create the style set.

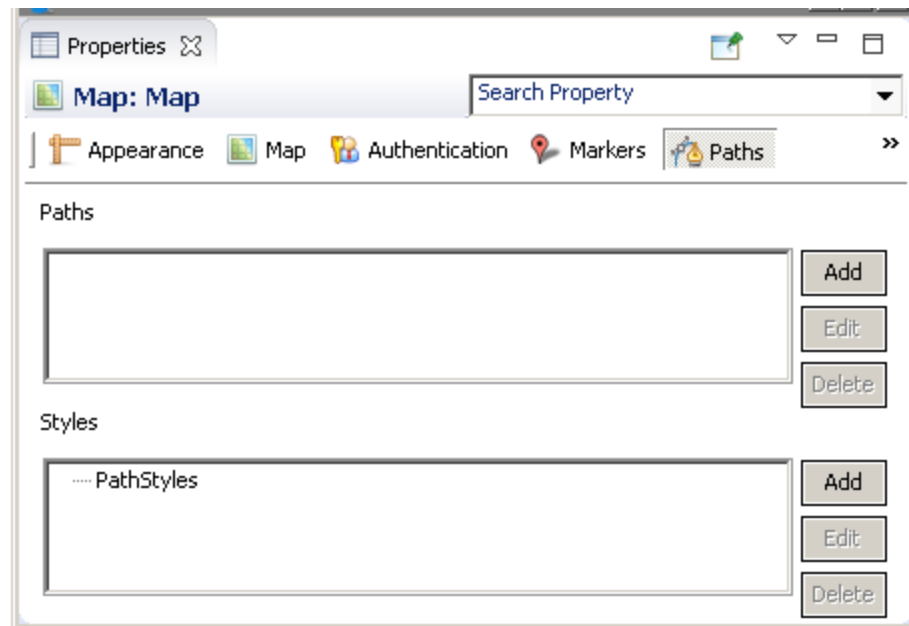


Figure 17-13 Styles on the Path tab of the Properties view for a map

17.4.2 Defining a Path Manually

To define a path using the Add button:

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. In the Paths section, click **Add** to open the Markers dialog.
3. To add a point to the path, click **Add** to open Path dialog box. For each point, specify the following:
 - a. the path name (to identify which path includes the point)
 - b. the latitude/longitude coordinates or the address of the point
 - c. additional optional properties, such as the name of a path style
 Click **OK** to add your point.
4. Use the **Up** and **Down** buttons to change the order in which the points appear.
5. Preview your report in HTML to see your path.

To add points to a path using the map preview:


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. In the Paths section, click **Add** to open the Markers dialog.
3. Select the Maps tab in the Markers dialog box.

4. Select your path from the Paths menu. The Paths menu has the following characteristics:
 - If you already have static paths defined for your map, you can select a path name from the Paths menu. Points you create are added to the currently-selected path. You can switch between paths at any time.
 - If you have not created any static paths, then you can enter a name on this menu. If a static path already exists, you cannot create a new one.
5. To add a point to the current path, right-click on the location you want and select **Add marker**.
6. To delete one or more markers, select the marker(s) in the panel at the right and press **Delete**, or right-click on the marker and select **Delete**.



You can't set formatting or styles using the map preview.

17.4.3 Defining a Path Using a Dataset

1. Create a CSV file, a data adapter that points to it, and a dataset that uses the data adapter. This example uses the same data as in **“Sample Data” on page 276**. Pay close attention when adding points to your data: they are connected on the map in the order that they appear in the data. If they aren't in a sensible order in the data, the path won't make sense, either.
2. Define any styles your paths use. This example uses the styles defined in **17.4.1.2, “Defining Path Styles Using a Dataset,” on page 281**.
3. Add a map component to the report, or select an existing map in the Design tab.
4. If you have not set the center or zoom, do so. For this example, click the Map tab in the Properties view, enter "San Francisco, CA" in the **Address** field, and enter 11 in the **Zoom** field.
5. Select the Paths tab in the Properties view.
6. In the Paths section, click **Add** to open the Markers dialog box.
7. Click the **Dataset** tab in the Markers dialog box and select **Use Dataset**.
8. In the Dataset Run section, select **SFLandmarksDataset** and accept the default settings. You have already set the default data adapter for this dataset.
9. For each property you want to read from the dataset, click  on the entry bar, select **Use Expression**, and enter the expression to use. For this example, use the following expressions:
 - Path Name – `{path}`
 - Latitude – `{latitude}`
 - Longitude – `{longitude}`
 - Address – `{landmark}`
 - Style – `{style}`
10. Click **OK**. The path information is added to the Path section in the Properties view.
11. Preview your report in HTML. The following image shows the example without markers. If you added the markers earlier, they will also be visible.

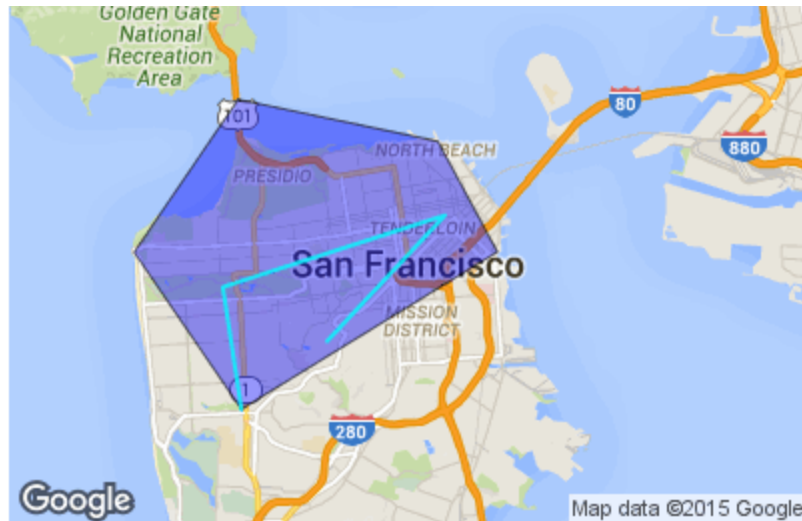


Figure 17-14 Paths on a map

17.4.4 Modifying Paths and Path Styles

To edit a path or path style:

1. Select the map and click on the Paths tab in Properties view.
2. Select the path or style you want to change and click **Edit**.
3. To change the dataset, make sure you have set up another dataset in your report before editing the path or path style. Then you can select the Dataset tab here and select the new dataset from the **Dataset Run** menu.
4. To change path or style properties, select the Items tab, and edit your properties.

To delete a path or path style:

1. Select the map and click on the Paths tab in Properties view.
2. Select the path or style you want to delete and click **Delete**.

17.5 Properties for Markers and Paths

The available properties are a subset of the properties available through the Google Maps APIs. See <https://developers.google.com/maps> for more information.

Table 17-5 Marker and Path Properties

Property	Description
Name	String. Name used to identify the marker or path; must be unique for markers or paths in the report.

Property	Description
Latitude	Number between -90 and 90. The latitude of a location in degrees.
Longitude	Number between -180 and 180. The longitude of a location in degrees.
Address	String. The address or placeID of a location. Only used if Latitude and Longitude are not available.
Color	String. The color of the path or marker. For best results, use hexadecimal representation, as not all Google API implementations will support color strings.
Clickable	Boolean. When <code>true</code> , the marker or path can handle mouse events. Default is <code>true</code> .
Draggable	Boolean. When <code>true</code> , a user can drag the marker or path contour. Default is <code>false</code> .
Visible	Boolean. When <code>true</code> , the marker or path is visible. Defaults to <code>true</code> .
Z Index	Number. Index determining the order in which objects are displayed on the map. Elements with higher values are displayed in front of similar elements with lower values. Markers are always displayed in front of paths.
Properties for Markers Only	
Title	String. Text shown on rollover.
Url	String. Target URL to access when marker is clicked.
Target	String. Target attribute specifying where to open linked document.
Icon	String. URL for the icon.
Custom Icon	Use Custom Icon settings to use a marker icon other than the default. You must specify a URL that points to the image to use. Currently, we don't support loading an image directly from the repository or as a resource local to the report. Instead, the JavaScript API loads the icon from the URL. You can set additional optional properties for your marker, such as icon height, width, origin, and anchor.
Shadow	String. URL for the shadow.
Custom Shadow Icon	Use Custom Shadow Icon settings to use a shadow icon other than the default. You must specify a URL that points to the image to use. Currently, we don't support loading an image directly from the repository or as a resource local to the report. Instead, the JavaScript API loads the icon from the URL. You can set additional optional properties for your shadow, such as height, width, origin, and anchor.
Info Window	Use Info Window settings to add an info window. You can define the window content, pixel offset, and maximum width.

Property	Description
Label	String. Single character that appears on the marker. Not available for custom markers.
Cursor	String. Mouse cursor to show on hover. Not available for custom markers.
Flat	Boolean. Not available for custom markers.
Optimized	Boolean. Not available for custom markers.
Raise on Drag	Boolean. Not available for custom markers.
Size	String. Not available for custom markers.
Properties for Paths and Path Styles Only	
Parent Style	String. Name of path style to use as a parent style. The current style inherits the parent's properties if the parent style is present in the report. Elements set locally in the current style override elements set in the parent.
Stroke Color	String. Color of the stroke; for most consistent results, use hexadecimal format. Default is #000000.
Stroke Opacity	Number. The path's opacity. Number between 0 (transparent) and 1 (opaque). Default is 1.
Stroke Weight	Number. Path weight in pixels. Default
Fill Color	String. Color of the fill for the polygon when Is Polygon is <code>true</code> . Takes values hexadecimal format. Default is null.
Fill Opacity	Number. The opacity for the polygon's fill when Is Polygon is <code>true</code> . Number between 0 (transparent) and 1 (opaque). Default is 1.
Is Polygon	Boolean. When <code>true</code> , creates a polygon (closed path) by connecting the last point on the path to the first point. Default is <code>false</code> (open polyline).
Editable	Boolean. When <code>true</code> , a user can edit the path by dragging the control points on the path line. Default is <code>false</code> .
Geodesic	Boolean. When <code>true</code> , dragged paths follow the great circles on the earth's surface; in this case, since the map is a projection, the lines may not appear straight. When <code>false</code> , paths are straight lines on the map. Defaults to <code>false</code> .

CHAPTER 18 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 service 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.

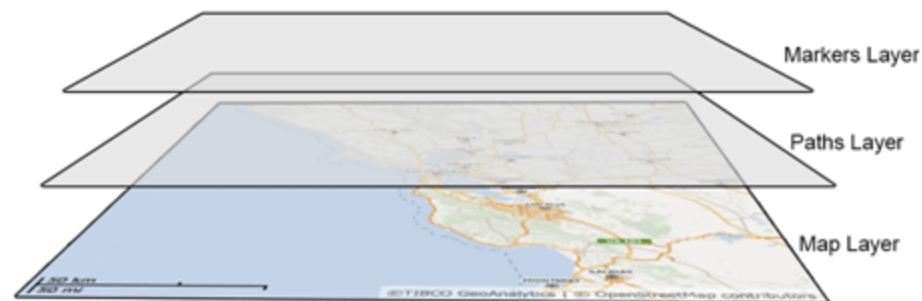



Figure 18-1 Basic structure of the TIBCO GeoAnalytics Map component

18.1 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 be 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 component's location as well as all other TIBCO Map properties.

The screenshot shows a configuration panel for 'Map Attributes'. It includes the following fields and their current values:

- Use Canvas:** false
- Opacity:** (empty)
- Zoom:** 20
- Max Zoom:** (empty)
- Min Zoom:** (empty)
- Repeat X:** (empty)
- Clip Offset:** (empty)

Figure 18-2 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

18.2 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.

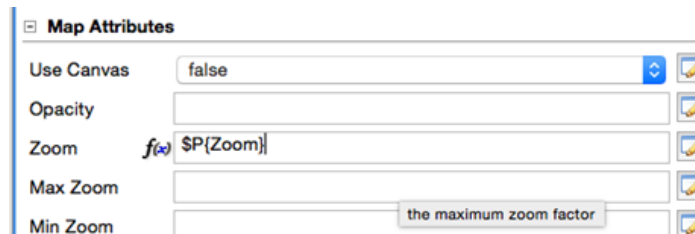


Figure 18-3 Map properties showing Zoom defined by an expression

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.

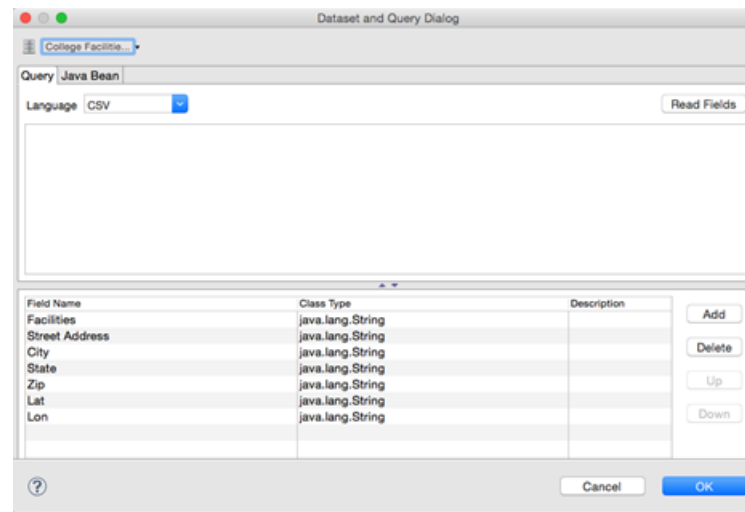


Figure 18-4 Defining the Dataset to use to resolve map attribute expressions

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.

18.3 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 Server interactive report viewer in the Layers drop down; this allows your user to select which layers to drawn.

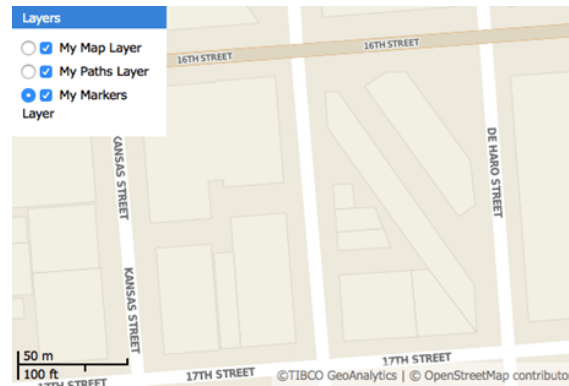


Figure 18-5 Layer names defined in the component can control the layers drawn in the final report

18.4 Working with Markers

Markers are points rendered on the second layer of the TIBCO Map component.

This section describes:

- **Static Markers**
- **Dynamic Markers**

18.4.1 Static Markers

1. Edit the map component's properties.
2. On the Markers tab, click Add.
3. Define your marker by specifying a location and icon. The list of properties for a marker includes:
 - target
 - string
 - optional
 - `_blank`
 - the hyperlink target for the marker

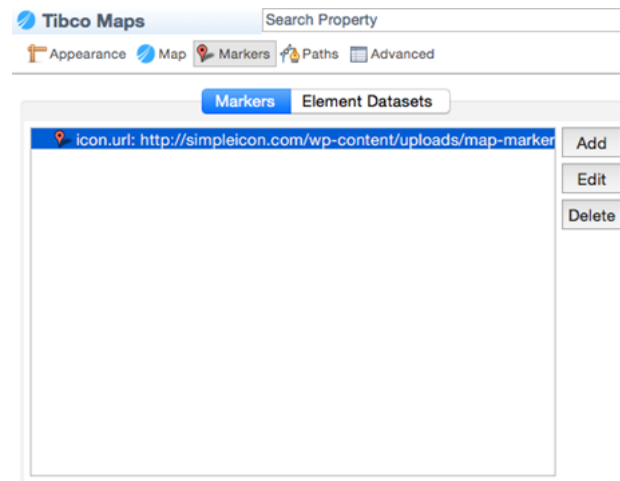


Figure 18-6 Defining a map's markers

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.

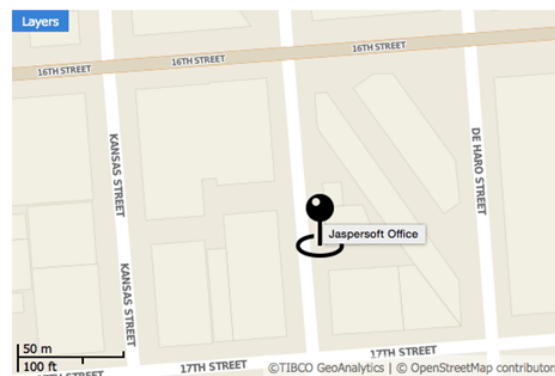


Figure 18-7 A map with a marker

5. For the addresses, set each property to form the address: country, state, zip, city, street.

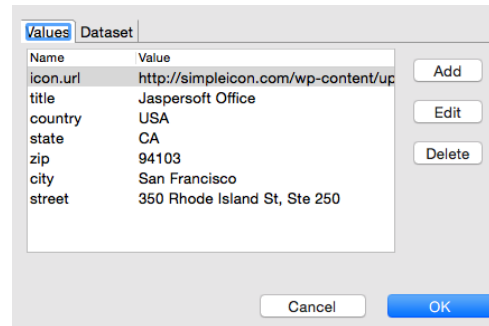


Figure 18-8 Marker properties set to a static location

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
draggable	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.

18.4.2 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 markers, 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. By default, the fields are all set as type String. To change the Latitude field to a Float, double-click in the Class Type column, click the button ellipsis..., and select java.lang.Float from the type menu. Repeat these steps to set the Longitude data type to Float.
6. Click OK.
7. 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.
8. 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.
9. Select the map in the Design tab, click the **Markers** tab, and click **Add**.
10. Click **Dataset**, check the **Use Dataset** check box, and click **Add**.
11. 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.
12. Click **OK**.
The dataset is added to the list of datasets we'll use for markers.
13. Click **Values** and create an expression for each marker property: for example, provide the title, street, city, state, country, and so forth.

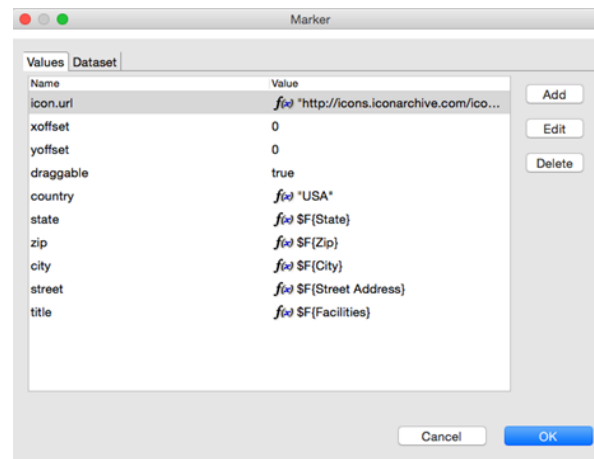


Figure 18-9 Location values defined as expressions

This example uses an icon from the web: [Pink Push Pin](#).

14. Click **OK**.
15. Preview your report in HTML.



Figure 18-10 San Francisco City College facilities marked on a map

18.5 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 19 WORKING WITH SUBREPORTS

The subreport element lets you nest one report (the subreport) inside another (the master report). A subreport can use the same database connection as the parent report or you can specify a different data source in the subreport properties. A master report can contain multiple subreports and subreports can be nested.

Subreports are one of the most advanced features of JasperReports. They allow you to design very complex reports by inserting one or more reports into another report.



Reflecting standard Eclipse design, saving or previewing a report that contains subreports does not update the subreports. When you edit a subreport, you must explicitly save and build the subreport in order for the changes to be visible when you preview the report that contains it. To build a subreport, right-click the project in the Project Explorer and select **Build Project**, or enter **Ctrl-B** to build all projects in the workspace.

Subreports also let you combine two or more child lists of data relating to a single parent element, for example, a report with multiple detail bands of different types. You map parameters between the master report and its subreports to create a blended report where each subreport displays details for each record from the master report. As the master report executes, each time a subreport element is reached, it is executed and its content is embedded into the output of the master report.

Uses for subreports include:

- Modularizing reports – You can create a subreport with your preferred data fields and layout, then use the subreport in multiple master reports.
- Combining multiple queries or data sources in a single report.

19.1 Creating a New Report via the Subreport Wizard

To simplify inserting a subreport, a wizard for creating subreports starts automatically when a Subreport element is added to a report.

You can use the Subreport Wizard to create a brand new report that will be referenced as a subreport or to refer to an existing report. In the latter case, if the report you choose contains one or more parameters, the wizard provides an easy way to define values for them.

To create a new subreport using the wizard:

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.

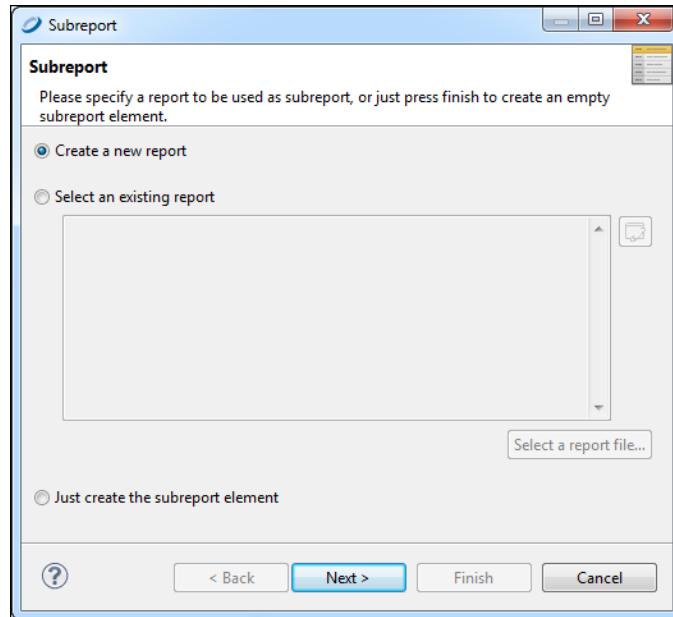


Figure 19-1 Subreport Wizard

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.

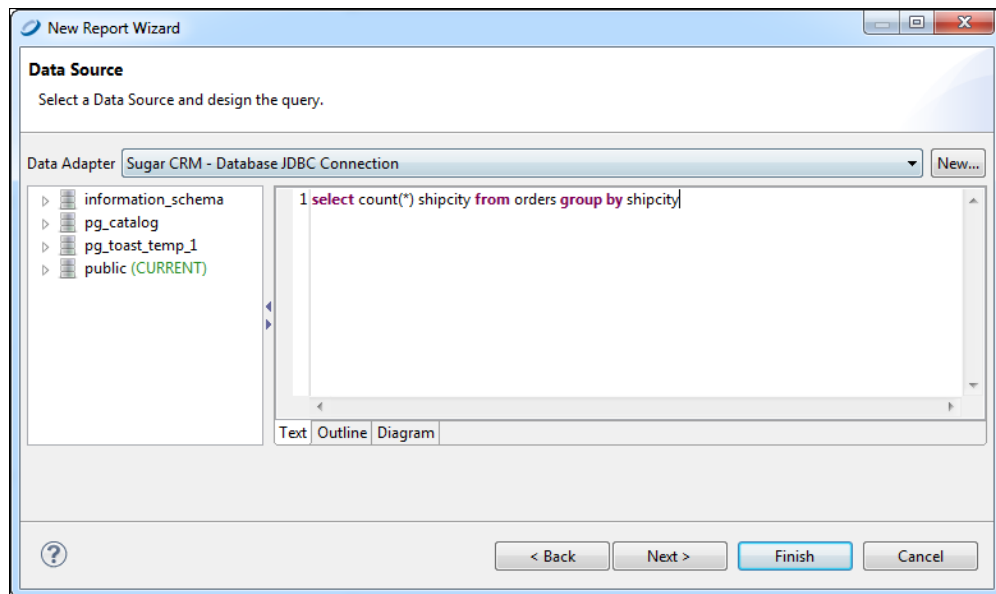


Figure 19-2 Data Source and Query

5. Choose to use the same data adapter as the main report, or a different data adapter.
For this example, choose the same adapter (Sample DB). Enter the following SQL query:

```
select count (*), shipcity from orders group by shipcity
```
6. Click **Next**.
7. Add all the fields to the list on the right. Click **Next**.
8. Click **Next** to skip the **Group By** step. The **Subreport > Connection** window opens.

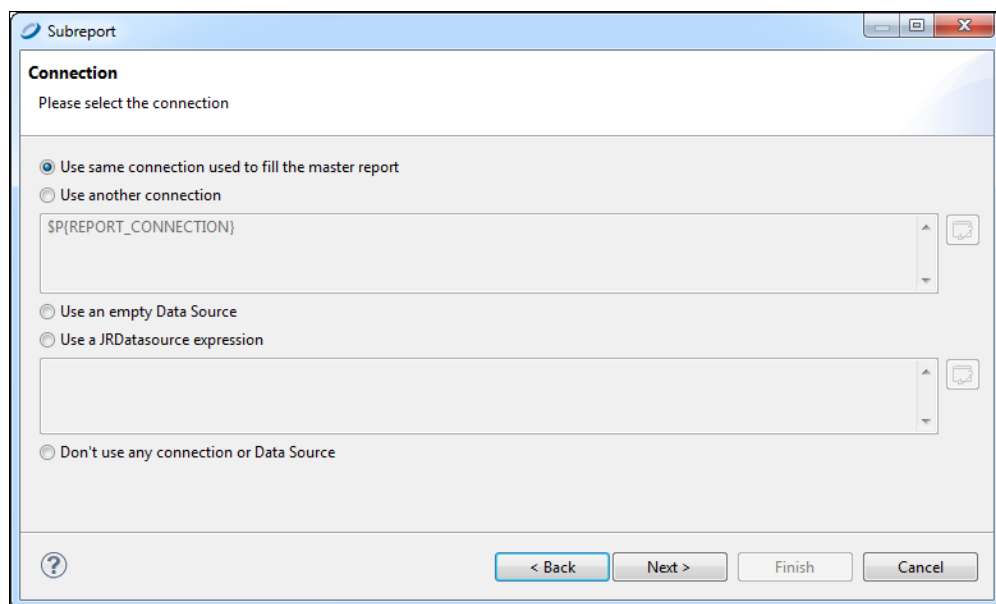


Figure 19-3 Subreport > Connection window

9. 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**.
10. Click **Next**. The **Subreport Parameters** window opens.

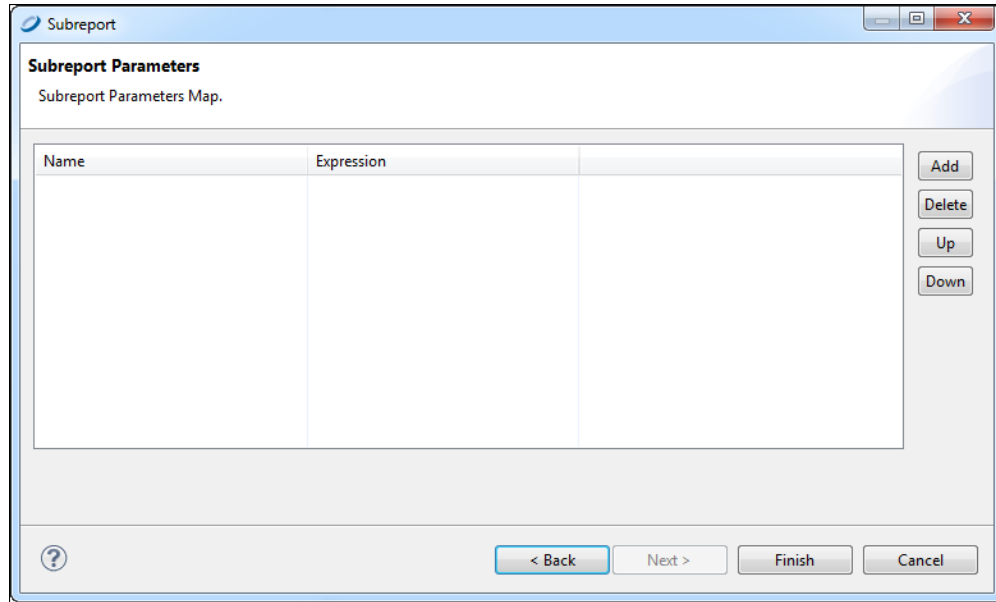


Figure 19-4 Subreport parameters window

11. For this example, skip this window and click **Finish**. A new report opens containing all bands.
12. 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.

19.2 Understanding Subreports

There are three steps to creating and adding a subreport:


1. **Create a report** – Create a parent or master report that will contain the subreport.
2. **Create a subreport** – Create and compile a subreport. Optionally create a dynamic connection to filter the records of the subreport based on the parent's data.
3. **Add the subreport to the parent report** – Insert a subreport element and specify the following:
 - The data adapter or data source for the subreport.
 - The location of the subreport's compiled Jasper file.
 - An optional parameters map (it can be empty) to set the report parameters used in the dynamic connection.

19.2.1 Subreports

A subreport is simply a report composed of its own JRXML source and compiled in a Jasper file. Generally speaking, creating a subreport is very similar to creating any other report. The margins of a subreport are usually

set to zero for subreports because a subreport is meant to be a portion of a page, not an entire document. The horizontal dimension of the subreport should be as large as the element into which it is placed in the parent report.

19.2.2 Subreport Elements

You add a subreport to a report by dragging the Subreport element  from the palette. At design time the element is rendered as a rectangle with the dimensions specified in the subreport.



The Subreport element does not need to be the same size as the subreport. You can think of the Subreport element as a place holder defining the position of the top-left corner to which the subreport is aligned. However, we recommend that you set the dimensions of the Subreport element to the dimensions of the subreport to best visualize the layout of the final report.

19.2.2.1 Properties of a Subreport Element

When a subreport element is selected in the master report, the following properties are available on the Subreport tab of the Properties view:

Property	Description
Run To Bottom	When true, the subreport element will consume the entire vertical space available on the report page.
Overflow Type	
Expression	(Required) Expression that can be used to load the Jasper object to use when filling the subreport portion of the document. Evaluated at run time to retrieve the Jasper object for the subreport. See 19.2.3, “The Expression Property,” on page 308 for more information.
Using Cache	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 <code>String</code> , because that string is used as key for the cache.
Connection Expression or Datasource Expression	At run time, returns a JDBC connection or a <code>JRDataSource</code> used to fill in the subreport. Only one of these expression types can be used. If there is no connection or data source expression, no data is passed to the subreport. This option is useful at times. In this case, the subreport should have the document property <code>When No Data Type</code> set to something like <code>All Sections, No Detail Or No Data Section</code> .
Parameters Map Expression	Optional expression used to produce a <code>java.util.Map</code> object at run time. The expression must contain a set of coupled names/objects that are passed to the subreport to set a value for its parameters.

Property	Description
Edit Return Values	Allows you to define how to store values in local variables calculated or processed in the subreport (such as totals and record count).
Edit Parameters	Allows you to define name/expression pairs used to dynamically set a value for the subreport parameters.

The following properties must be set to link the subreport to the parent report:

- **Expression** – Retrieves the Jasper object that implements the subreport.
- **Connection Expression** or **Datasource Expression** – Defines how to feed the object with data.
- **Parameters** – Sets the values of the subreport parameters.

19.2.3 The Expression Property

The subreport expression specifies the location of the Jasper file used to generate the subreport.

If the expression is a string (`java.lang.String`), JasperReports assumes that the subreport must be loaded from a Jasper file and tries to locate the file in the same way that resources are located, as follows:

1. The string is at first interpreted as a URL.
2. In case of failure (a `MalformedURLException`), the string is interpreted as a physical path to a file. This is the most common case.
3. If the file does not exist, the string is interpreted as a resource located in the classpath.

This means that using an expression of type `String` means you are in some way trying to specify a file path. Optionally, you can put your Jasper file in the classpath and refer to it as a resource, using an expression something like `"subreport.jasper"`.



You can't use a relative path to locate the subreport file; that is, if you have a report in `c:\myreport\main_report.jasper`, you cannot refer to a subreport using an expression like `..\mysubreports\mysubreport.jasper`.

This is because JasperReports does not keep in memory the original location of the Jasper file that it's working with. This makes perfect sense, considering that a Jasper object is not necessarily loaded from a physical file.

To simplify report design when loading a subreport from the file system, do one of the following:

- Place the subreport file in a directory that is in the classpath. This permits you to use very simple subreport expressions, such as a string containing just the name of the subreport file (that is, `"subreport.jasper"`). JasperSoft Studio always includes the classpath of the directory of the report that is running, so all the subreport Jasper files can be found easily if they are located in the same directory.
- Parametrize the Jasper file location and create on-the-fly the real absolute path of the file to load. This can be achieved with a parameter containing the parent directory of the subreport (let's call it `SUBREPORT_DIRECTORY`) and an expression like this:

```
`${SUBREPORT_DIRECTORY} + "subreport.jasper"
```

One advantage of this approach is that you can use the Jasper files' local directory as the default value for the `SUBREPORT_DIRECTORY` parameter. The developer who will integrate JasperReports in his applications can set a different value for that location just by passing a different value for the `SUBREPORT_DIRECTORY` parameter.

19.2.4 Specifying the Data Source

For JasperReports to retrieve data and fill the subreport, you have to set the subreport data source. The following options are available:

- **Use the same connection used to fill the master report** – Select this to use the same JDBC data adapter for the master report and the subreport. The JDBC connection is passed to the subreport to execute it.
- **Use another connection** – Select this to specify a different JDBC data adapter for the subreport.
- **Use a JRDataSource expression** – Select this to use a `JRDataSource` object to fill the subreport.
- **Use an empty datasource** – Select this to set the data source expression to `new JREmptyDataSource()`. That creates a special data source that provides a single record with all the field values set to `null`. This is useful when the subreport is used to display static content such as headers, footers, and backgrounds. In this case, in the subreport, set the report property `When no data type` to `All Data No Details` or `No Data Section` to ensure that at least a portion of the document is actually printed.

JDBC connections make using subreports simple enough. A connection expression must identify a `java.sql.Connection` object (ready to be used, so a connection to the database is already opened). Typically, we'll run the SQL query using the same database connection as the parent report; the connection can be referenced with the `REPORT_CONNECTION` built-in parameter. It must be clear that if we pass a JDBC connection to the subreport, it is because we defined an SQL query in the subreport, a query that will be used to fill it.

Using a different data source is sometimes necessary when a connection like JDBC is not being used; it is more complicated but extremely powerful. It requires writing a data source expression that returns a `JRDataSource` instance that you then use to fill the subreport. Depending on what you want to achieve, you can pass the data source that will feed the subreport through a parameter, or you can define the data source dynamically every time it is required. If the parent report is executed using a data source, this data source is stored in the `REPORT_DATASOURCE` built-in parameter. On the other hand, the `REPORT_DATASOURCE` should never be used to feed a subreport; a data source is a consumable object that is usable for feeding a report only once. Therefore, the parameter technique is not suitable when every record of the master report has its own subreport (unless there is only one record in the master report). When we discuss data sources this will be more clear and you will see how this problem is easily solved with custom data sources. You will also see how to create subreports using different type of connections and data sources.

19.2.5 Subreport Parameters

One of the most common uses of subreport parameters is to pass the key of a record printed in the parent report in order to execute a query in the subreport through which you can extract the records referred to (report headers and lines). For example, let's say you have in the master report a set of customers, and you want to show additional information about them, such as their contact info. The subreports will use the customer ID to get the contact info. The customer ID should be passed to the subreport as a parameter, and its value changes for each record in the master report.

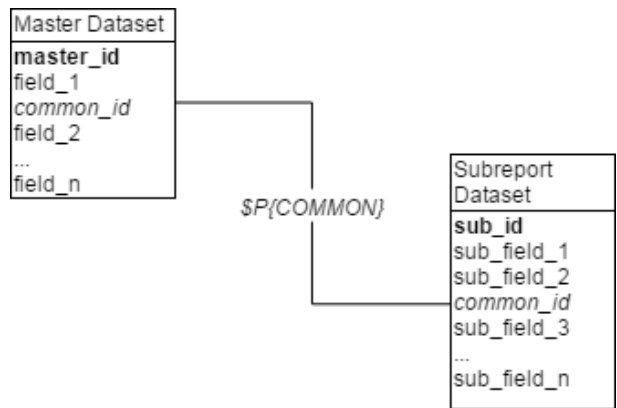


Figure 19-5 Related datasets in master and subreport

To pass parameters from the master report to a subreport, you create a set of parameter name/object pairs that feed the parameters map of the subreport. To do this, click the **Edit Parameters** button on the Subreport tab of the Properties view to open the Subreport Parameters dialog.



When a report is invoked from a program (using one of the `fillReport` methods, for instance), a parameters map is passed to set a value for its parameters. A similar approach is used to set a value for subreport parameters. With subreports you don't have to define a map (even, if possible, specifying a `Parameters Map Expression`). The report engine will take care of that for you.

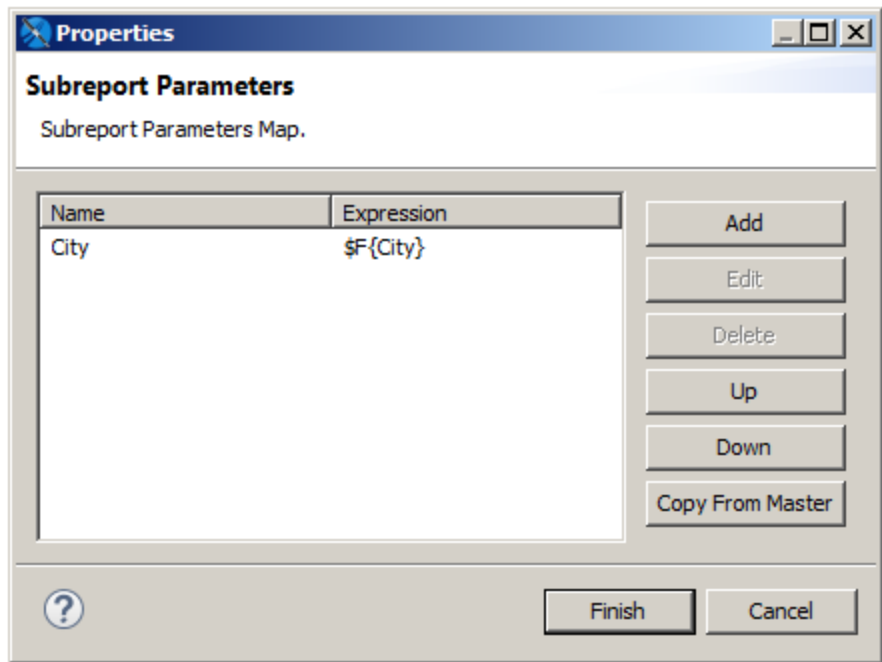



Figure 19-6 Subreport Parameters dialog

To configure a parameter you want to pass to the subreport, click **Add** in the Subreport Parameters dialog to open the Parameter Configuration Dialog box, which lets you set the following:

- **Name** – Name of the parameter. A parameter must have the same name in the master report and the subreport. Parameter names are case-sensitive.



If you make an error typing the name or the inserted parameter has not been defined, no error is thrown. In most cases, the report will fail silently.

- **ValueExpression** – JasperReports expression for the parameter. To create or edit an expression, click  to open the expression editor. You can use fields, parameters, and variables. The return type has to be congruent with the parameter type declared in the subreport; otherwise, an exception of `ClassCastException` will occur at run time.

As cited below, you have the option of directly providing a parameters map to be used with the subreport; the `Parameters Map Expression` allows you to define an expression, the result of which must be a `java.util.Map` object. It is possible, for example, to prepare a map designed for the subreport in your application, pass it to the master report using a parameter, then use that parameter as an expression (for example, `#{myMap}`) to pass the map to the subreport. It is also possible to pass to the subreport the same parameters map that was provided to the parent by using the built-in parameter `REPORT_PARAMETERS_MAP`. In this case the expression looks like this:

```
#{REPORT_PARAMETERS_MAP}
```

Since the subreport parameters can be used in conjunction with this map, you could even use it to pass common parameters, such as the username of the user executing the report.

CHAPTER 20 REPORT TEMPLATES

Templates are one of the most useful tools in 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**



You can also create JasperReports Server templates and upload them to the server. See [12.3, “Working with JasperReports Server Templates,”](#) on page 179.

20.1 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.

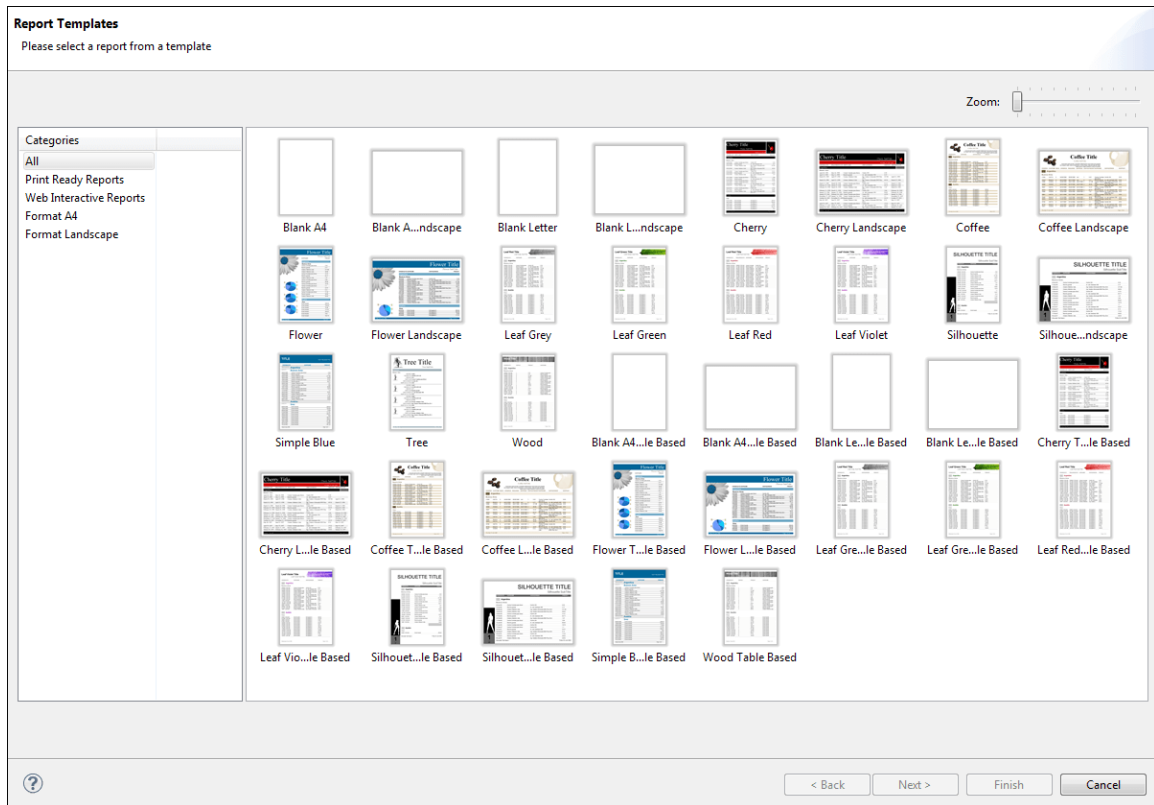


Figure 20-1 Default Report Templates

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).

20.2 Creating and Customizing Templates

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

20.2.1 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**.

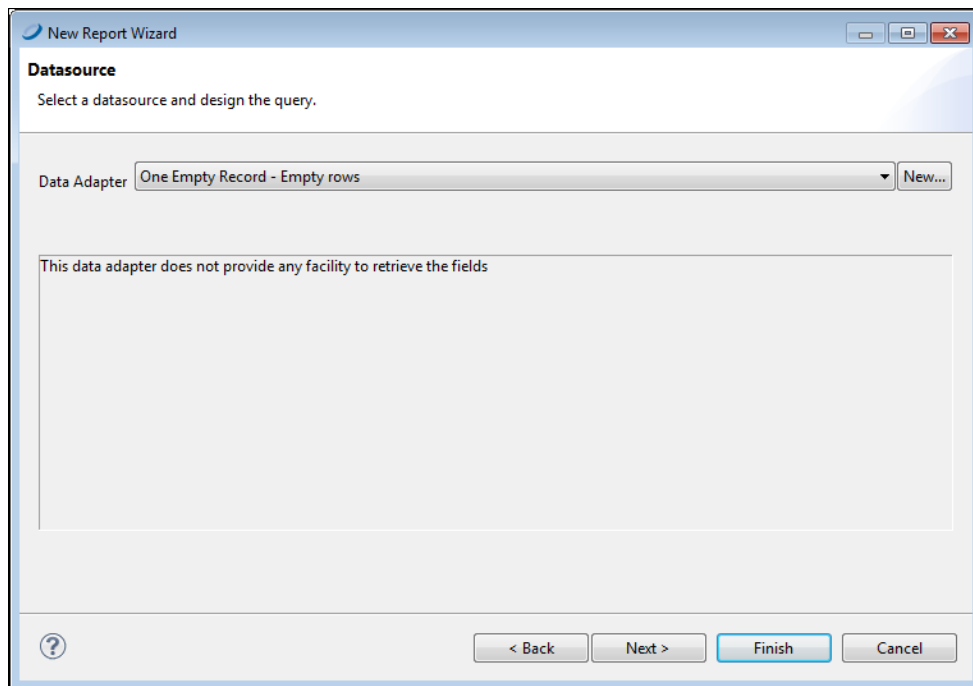


Figure 20-2 One Empty Record Data Source

An empty report opens, containing the following bands:

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

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

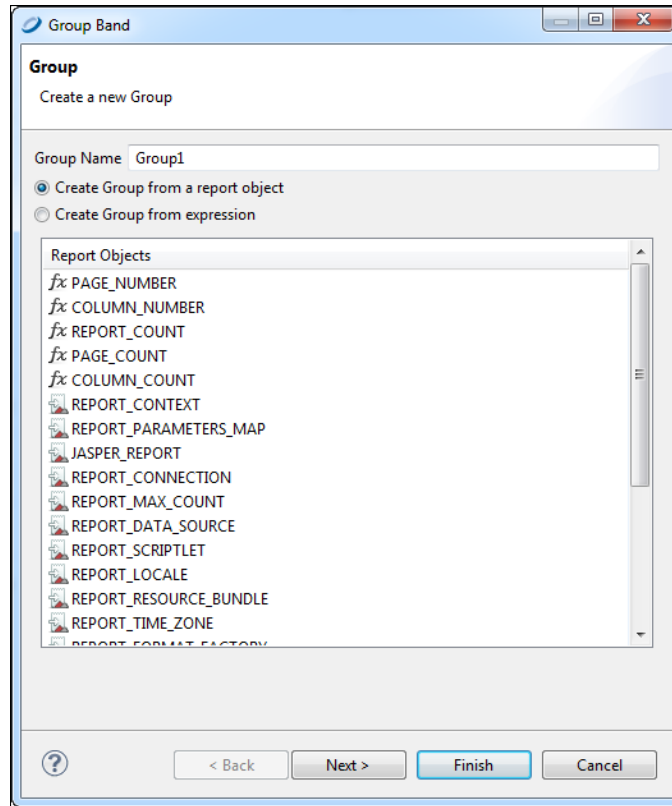


Figure 20-3 Group Band Dialog

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

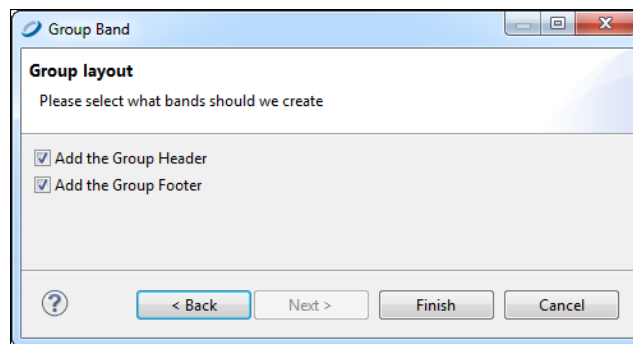


Figure 20-4 Group Layout Dialog

- Leave both **Add the Group Header** and **Add the Group Footer** checked, and click **Finish**. Your report is similar to the one in [Figure 20-5](#).

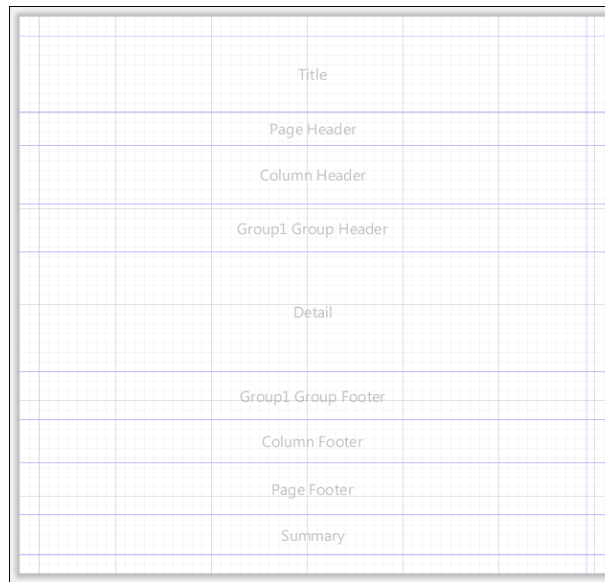


Figure 20-5 Report Containing Group Header and Footer

20.2.2 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 48](#).
2. Add a title: Drag a **Static Text** element to the Title band. Style the text in the Properties view. For more information about Static Text elements, see [“Text Elements” on page 49](#).
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 look something like the one in [Figure 20-6](#).

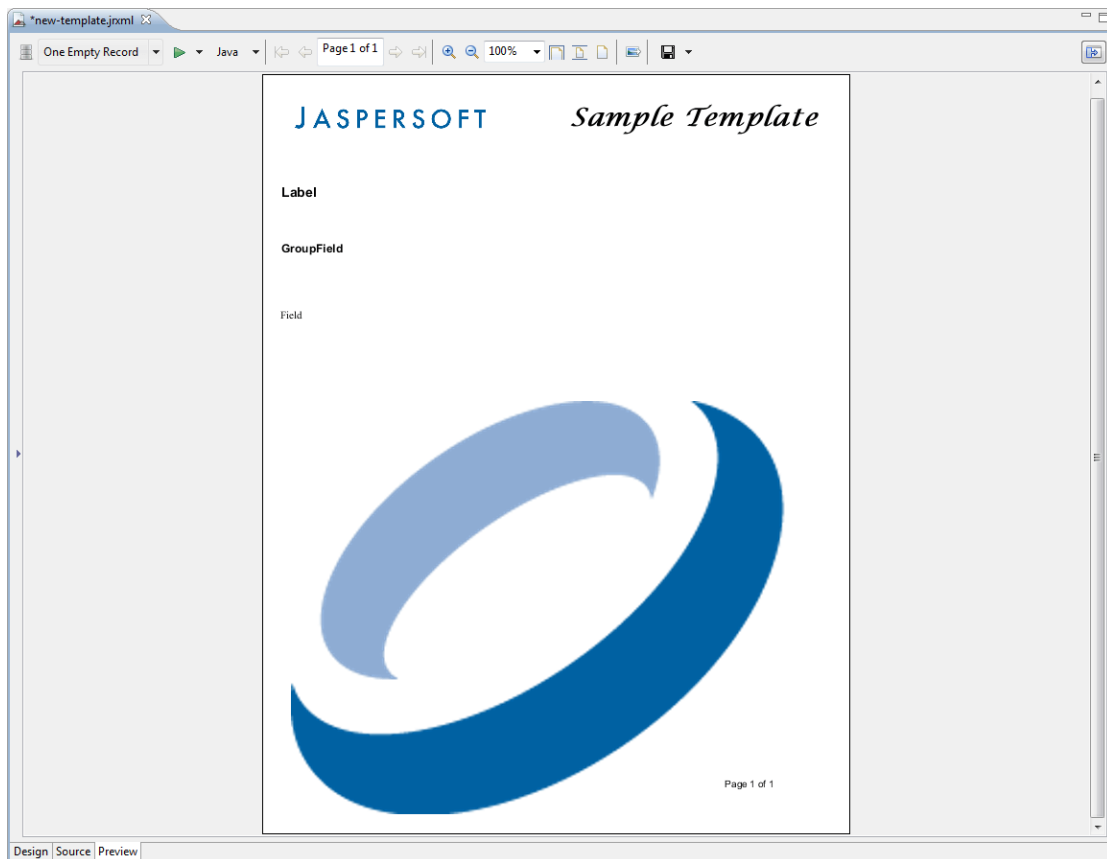


Figure 20-6 Template Preview

20.3 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.

20.3.1 Creating a Template Directory

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

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

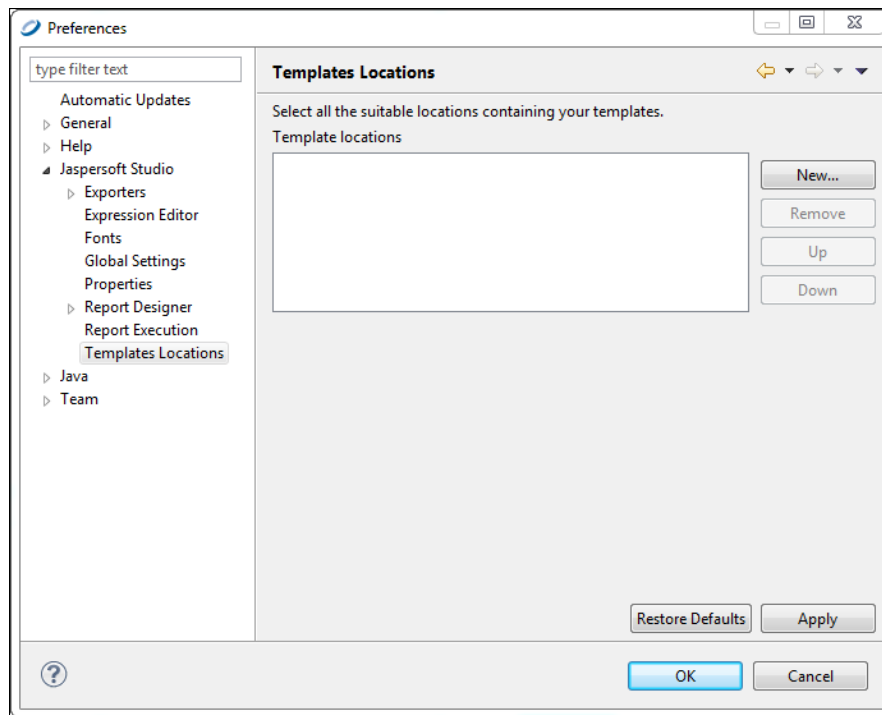


Figure 20-7 Template Location Preferences

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**.

20.3.2 Exporting a Template

Save your template for future use.

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

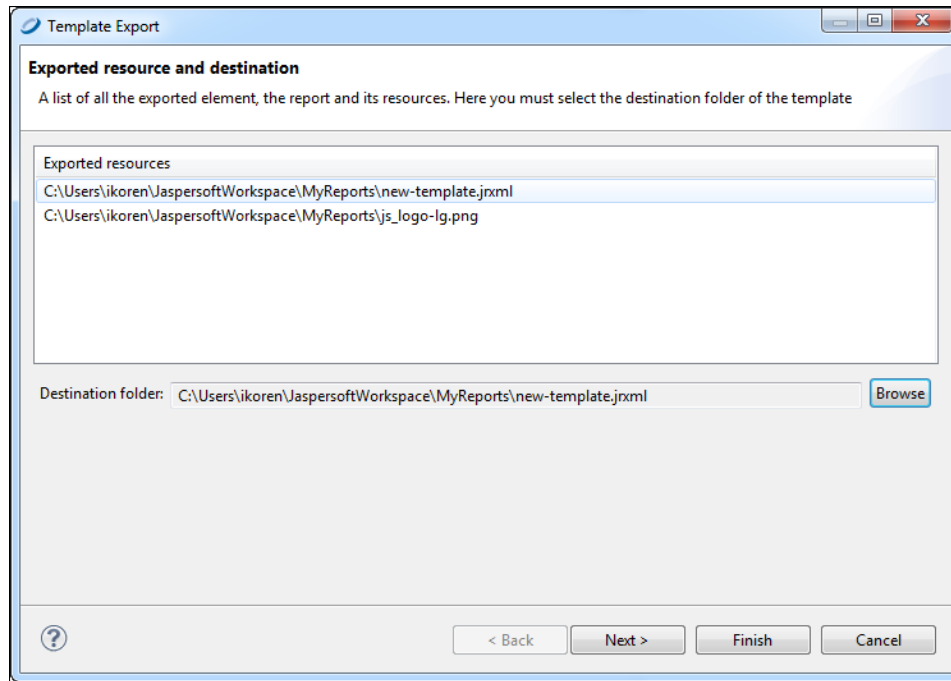


Figure 20-8 Template Export Dialog

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.

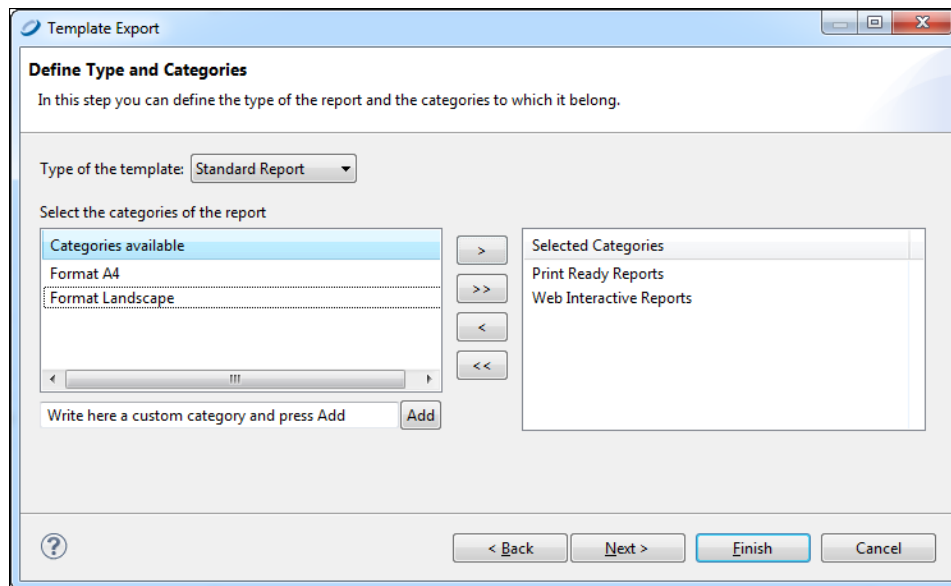


Figure 20-9 Define Type and Categories Dialog


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**.

20.3.3 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.

20.4 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**.

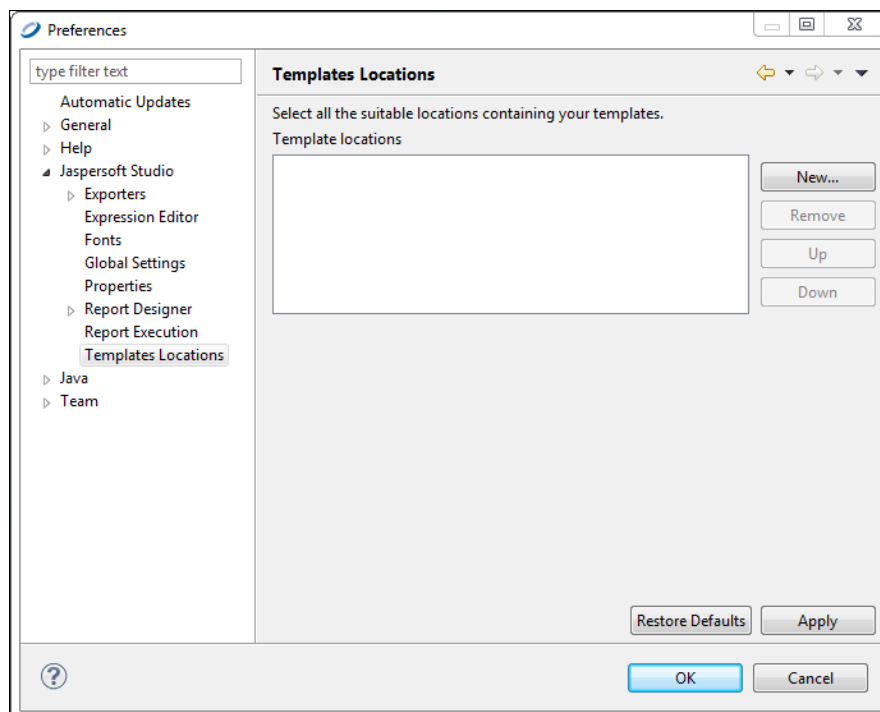


Figure 20-10 Template Location Preferences

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 21 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**

21.1 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 panel, 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.

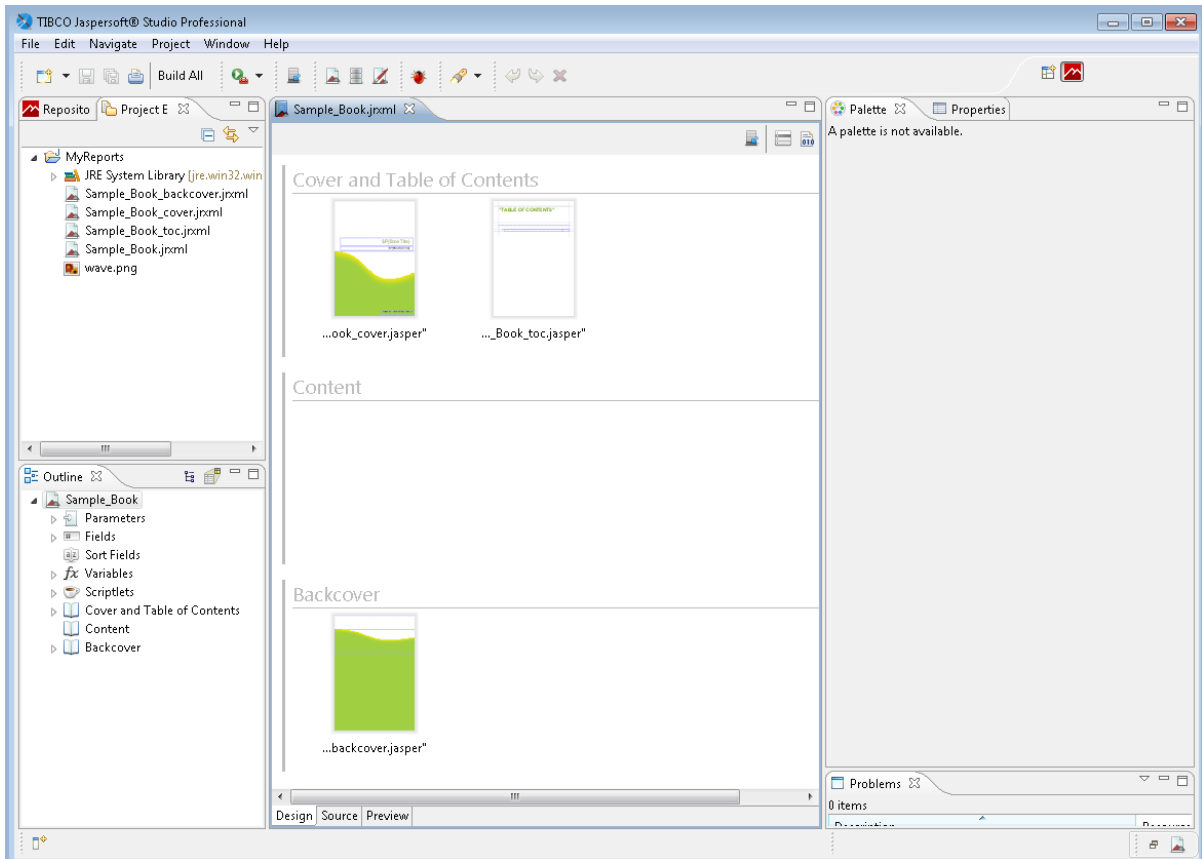


Figure 21-1 Report Book Framework

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.

21.2 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.

21.2.1 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.

21.2.2 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 data source 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 view, 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**.


21.3 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.

21.3.1 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 view, right-click the Parameters folder and select **Create Parameter**.
3. In the Properties view, change the name to `country`.
4. In the Designer, click  and modify the query to say:



```
Select * from orders where shipcountry = ${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 **Data** tab in the Properties view, 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.

21.3.2 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 tab.

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 tab, 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 = ${Country} group by shipcity
```
6. Click **OK**.
7. Save Country_Intro.jrxml.
8. In the Outline view, drag **Country** from the Parameters list into the Title band.
9. Click the Country parameter ($\${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**.

21.4 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`.

21.5 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.

21.6 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

CHAPTER 22 PREFERENCES AND CONFIGURATION

You can set preferences for JasperSoft Studio in the Preferences window.

To open the Preferences window:

1. Select **Window > Preferences** to open the Preferences window.
2. Expand the **JasperSoft Studio** node.

22.1 Properties

You can set JasperReports properties in the **JasperSoft Studio > Properties** page of the Preferences window. Setting a property here sets it as the default for all reports. You can also set many properties at the report or element level. A property set at the element level overrides a property set at the report level; a property set at the report level overrides a property set at the JasperSoft Studio level.

22.2 JasperReports Samples

JasperReports Library provides a number of sample reports that show how to use many of the available features. You can download and install the samples as a project in JasperSoft Studio as follows:

1. Select **File > New > Other** from the main menu.
2. In the New dialog box, expand JasperSoft Studio.
3. Select **JasperReports Samples** and click **Next**.
4. Enter a name for the project folder and click **Finish**.

The sample reports are downloaded to the location you chose. You can now view and work with these reports in JasperSoft Studio.

22.3 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.

22.3.1 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

22.3.2 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 view, and select a supported unit from the pop-up menu:

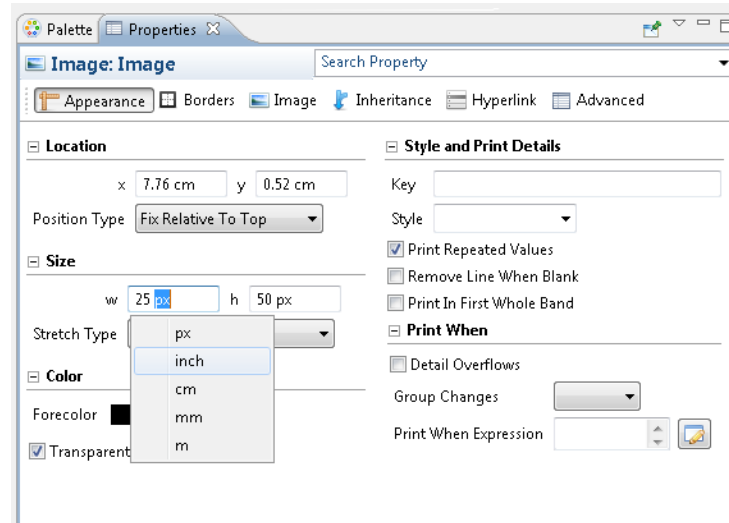


Figure 22-1 Updating a field's measure unit

22.3.3 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.

22.3.4 Approximations

Even though 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).

22.4 Export and Import

Export and import allow you to migrate configuration resources between instances of Jaspersoft Studio. As of Jaspersoft Studio 6.2.2, you can export the following configuration resources:

- global data adapters
- JasperReports Server configurations
- composite elements
- text, table, and crosstab styles
- global JasperReports properties
- Jaspersoft Studio preferences



The Jaspersoft Studio application logger preferences cannot be exported, since they are determined in part by your application INI configuration.

You can choose to export all of these categories or only a subset of them; however, you can't choose individual items inside a category. The result of the export is a single zip file (compressed archive), which can be imported into another Jaspersoft Studio instance. Again, you can choose which of the available categories inside the zip you want to import.

To export configuration resources:

1. Select **File > Export**.
The Export dialog is displayed
2. Select **Jaspersoft Studio > Jaspersoft Studio Configuration** for the destination and click **Next**.
The export wizard shows the resource categories that can be exported, with the number of resources in each category. If there are no resources in a category, the category does not appear on the list.

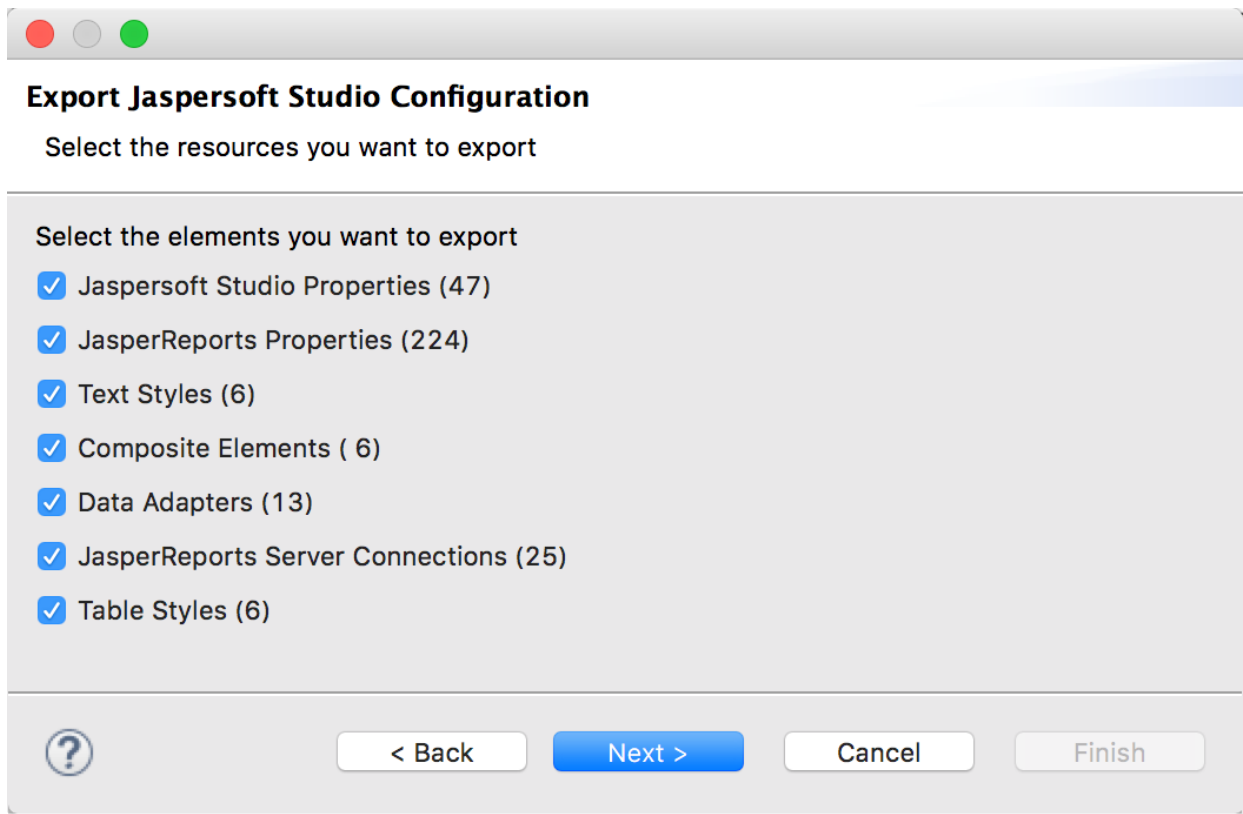


Figure 22-2 Export Jaspersoft Studio Configuration Wizard

3. Select the categories you want to export and click **Next**.
4. Enter the location and name you want for the exported file and click **Finish**.
A zip file is created in the location you chose.

To import configuration resources:

1. Select **File > Import**.
The Import dialog is displayed.
2. Select **Jaspersoft Studio > Jaspersoft Studio Configuration** and click **Next**.
3. Enter the location and file name of the zip file you wish to import and click **Next**.
If the file is a valid configuration file, the wizard shows the resource categories that can be imported, with the number of resources in each category. If the file is not a valid configuration file, you will receive an error message.

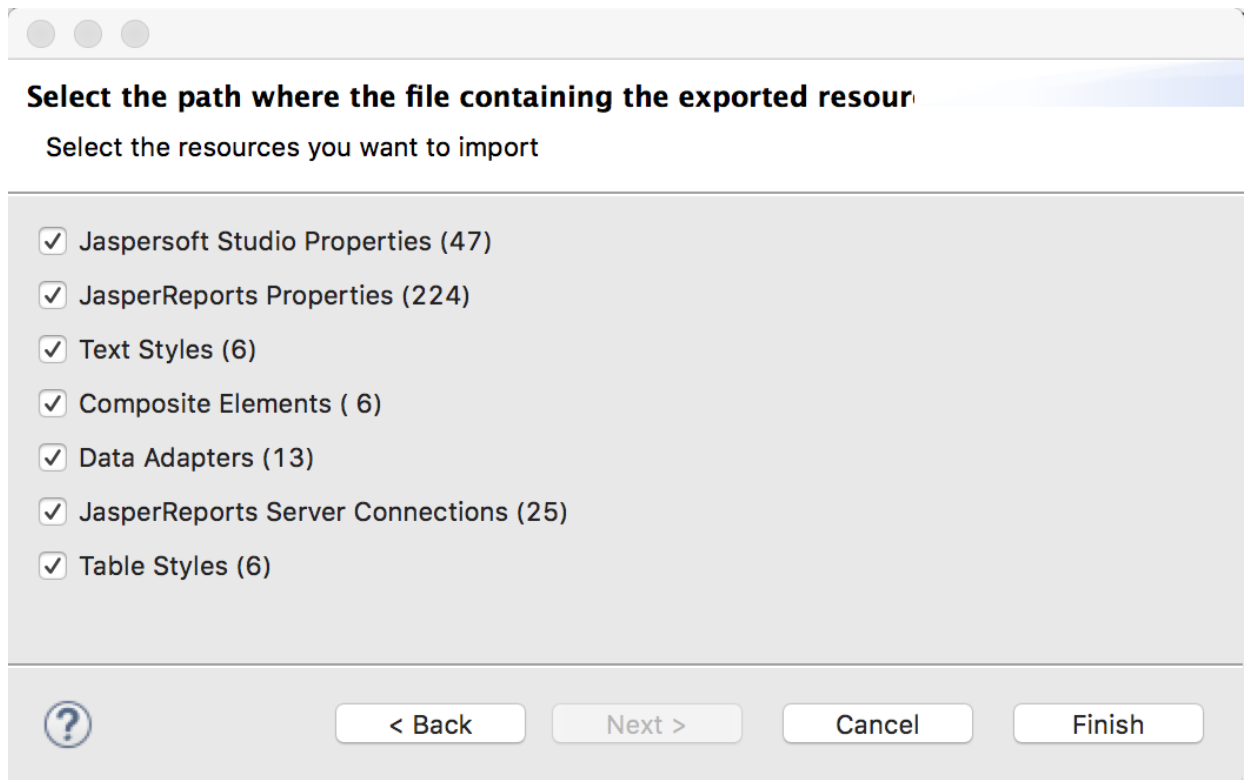


Figure 22-3 Selecting Categories to Import

4. Select the resource categories you want to import and click **Finish**.
5. If there is a naming conflict between an imported resource and an existing resource in your Jaspersoft Studio configuration, choose the action you want in the displayed dialog. For resource categories other than Jaspersoft Studio properties and JasperReports Library properties, you have three choices:
 - **Overwrite** – Overwrites the existing resource(s) with the imported resource(s) of the same name.
 - **Keep both** – Automatically renames the conflicting imported resource(s) with a unique name.
 - **Skip** – Keeps the existing resources and discards the imported resources.

For Jaspersoft Studio properties and JasperReports Library properties, which do not support multiple instances, you are prompted to choose to overwrite or not.

As before, you must choose the same action for all conflicting resources in a category. For example, if you have multiple conflicting global data adapters, you must overwrite, keep both, or skip all global data adapters. A separate dialog is shown for each category where you have conflicting resources. You can choose different actions for different categories.

22.4.1

22.5 Setting Compatibility with Earlier Versions of JasperReports Library

If you are using your reports with an application you have built using JasperReports Library, you can set the version to use for compiling your reports. Normally, when you compile a report, Jaspersoft Studio uses the

corresponding version of JasperReports Library. For example, if you compile a report from Jaspersoft Studio 6.2, it uses JasperReports Library 6.2. For backwards compatibility with your applications, you can configure Jaspersoft Studio to use an earlier version of JasperReports Library to compile your reports. If you do this, any features in your reports that rely on a later version of JasperReports Library will not be available.



If you are exporting your reports to JasperReports Server, you should configure the version in the JasperReports Server connection settings, as described in [12.1.1, “Advanced Connection Settings,” on page 171](#). Use the compatibility setting only if you are using your reports on your own application built from JasperReports Library.

To set the version of JasperReports Library to use for compiling reports:

1. Select **Window > Preferences** from the main menu.
The Preferences dialog is displayed.
2. Select **Jaspersoft Studio > Compatibility**.
The Compatibility window is displayed.

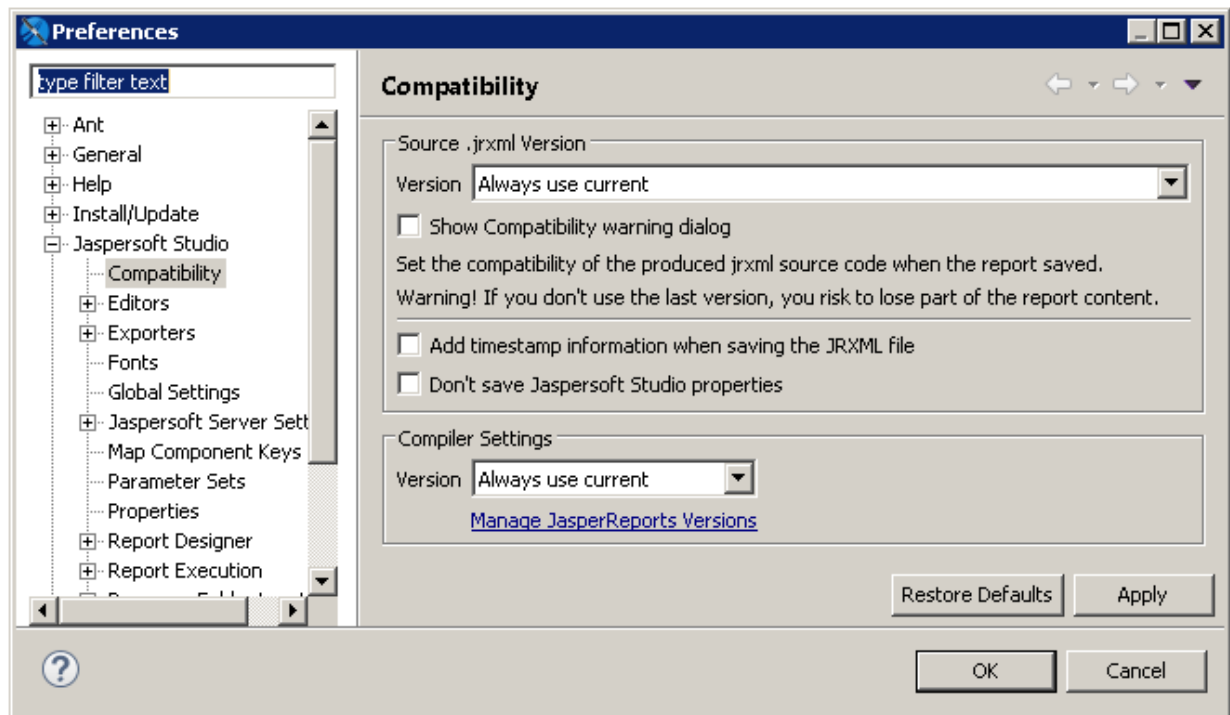


Figure 22-4 Setting JasperReports Library Version

3. To save your reports in an earlier version of JRXML, select the version you want from the **Version** menu in the Source .jrxml Version section of the dialog.
4. To remove Jaspersoft Studio properties from your compiled reports, select **Don't save Jaspersoft Studio properties**. Properties specific to Jaspersoft Studio include some layout information, dimensions in pixels or millimeters, and the data adapter that was most recently used in Jaspersoft Studio.

5. To use an earlier version of JasperReports Library to compile reports, select the version you want from the **Version** menu in the Compiler Settings section of the dialog. If the version you want is not available, set it up as described in the next step.
6. To add a version of JasperReports Library to the **Version** menu in the Compiler Settings section of the dialog click **Manage JasperReports Versions** and select the version you want:
 - a. To use a version you already have installed, click **Add From Path**, then select the directory where the JasperReports Library is located.

Jaspersoft Studio verifies that the path contains JasperReports Library and adds the version to the **Version** menu in the Compiler Settings section of the Compatibility dialog.
 - b. To download and install JasperReports Library from a URL, click **Add From URL** and select the URL from SourceForge (<https://sourceforge.net/projects/jasperreports/>) or from another location.

Jaspersoft Studio downloads and verifies the jar files, copies the files to a Jaspersoft Studio internal directory, and adds the version to the **Version** menu in the Compiler Settings section of the Compatibility dialog.

APPENDIX A 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 runtime executables.

This chapter contains the following sections:

- **JRXML Sources and Jasper Files**
- **Data Sources and Print Formats**
- **A Simple Program**

A.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).

A.1.1 The Report Lifecycle

The life cycle of a JasperReport is divided into two phases:

- Report development – designing and planning the report, creating a JRXML file, and compiling a Jasper file from the JRXML.
- Report execution – loading the Jasper file, filling the report, and exporting the output (a Jasper print object) in a final format.

Jaspersoft Studio is primarily focused on report development, 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, you specify where the data comes from, how it is positioned on the page, and additional functionality, such as parameters for input controls or complex formulas to perform calculations. The result is a template, similar to a form containing blank space, that is filled with data when the report is executed. The template is stored in a JRXML file, which is an XML document that contains the definition of the report layout and design.

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.

The following table shows sample report source code.

Table A-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>
</jasperReport>
```

```

    </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>
  </band>
</columnHeader>

```

```

</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["" +
  ${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 " +
  ${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.

A.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`.

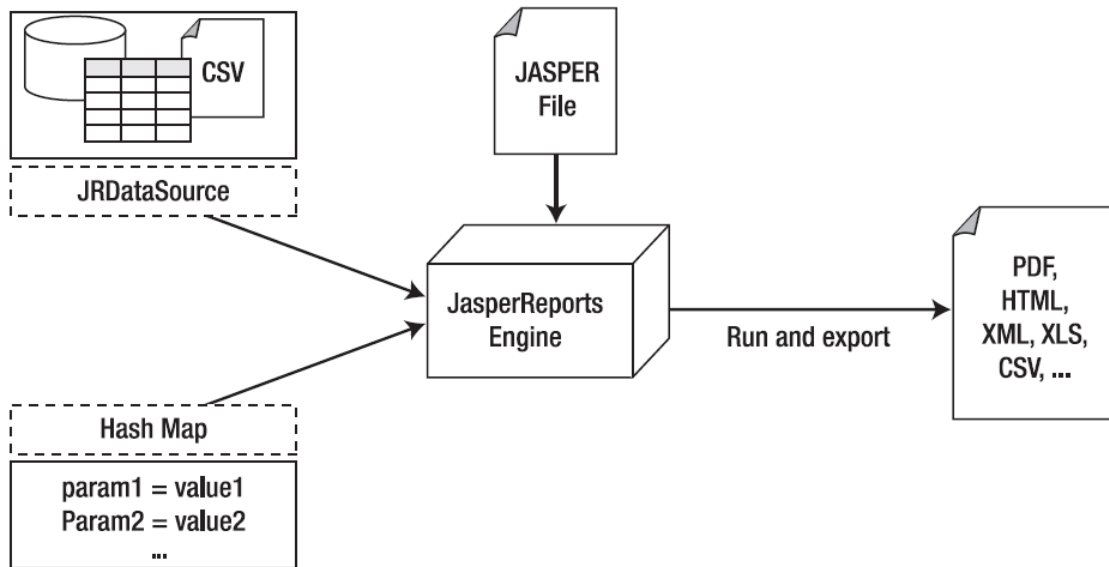


Figure A-1 Data Source and Parameter Flows for Report Creation

`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 preventatively 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 hard copy.

A.3 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.

A.4 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 A-1 on page 341](#).

Table A-2 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);
        }
    }
}
```

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 Jaspersoft Studio, 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 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.

Dashlet

An element in a dashboard. Dashlets are defined by editable properties that vary depending on the dashlet type. Types of dashlet include reports, text elements, filters, and external web content.

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 > Server > Settings Ad Hoc Settings**. By default, this setting is only available to the superuser account.

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

In Jaspersoft Studio, a frame is a rectangular element that can contain other elements and optionally draw a border around them. Elements inside a frame are positioned relative to the frame, not to the band, and when you move a frame, all the elements contained in the frame move together. A frame automatically stretches to fit its contents.

Frame can also refer to an element in a legacy dashboard; it's the equivalent of a dashlet.

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.

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.

Jaspersoft Studio

A commercial 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.

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 Jaspersoft Studio and JasperReports Server. JRXML is an open format that uses the XML standard to define precisely all the structure and configuration of a report.

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.

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 manage 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 help desk 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 JasperSoft Studio. 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

A

- Ad Hoc views
 - localizing 184
 - resources 184
- adding resources to the server 185
- anchors 55

B

- Beans See Java
- bookmarks 55

C

- Cassandra connections 136
- chart reports
 - theme for 187
- charts
 - combination charts 238
 - datasets 209
 - dual-axis charts 238
 - multi-axis charts 238
 - scatter charts 234
 - themes 218
 - three-dimensional pie 209
 - types 209
- ClassNotFoundException error 127
- classpath 127
- columns
 - column groups 204, 206-207
 - in Table component 204, 206

connections

- Cassandra 136
- creating 125
- importing from a workspace 15
- JasperReports Server See JasperReports Server
- JDBC 125
- MongoDB 131

containers

- grid layout 41

creating

- HTML5 charts 227
- report templates for JasperReports Server 179

crosstabs

- adding a measure 265
- adding a row group 261
- creating 252
- crosstab parameters 258, 267
- crosstab total variables 264
- custom calculation 264
- datasets 253
- defined 251
- editing group expressions 259
- measures 263
- properties 257
- resizing columns 258
- row and column groups 259
- time fields and aggregation 254

CSV

- Add node as field 146
- data adapters 149
- PersonBean 141

custom visualization component 61

D

D3 JavaScript library 61

data adapters

- Cassandra 136
- collection of JavaBeans 138
- copying 122
- creating 120
- CSV 149
- default 123
- Domain JRS 189
- exporting 121
- file-based 120
- global 120
- importing 121
- importing from a workspace 16
- in a project 120
- in reports 123
- JavaBeans 138
- MongoDB 131
- net.sf.jasperreports.data.adapter property 123
- overview 119
- publishing to JasperReports Server 175
- Spotfire Information Links 157
- using 123
- XML 141

data sources

- JRDataSource 154, 156, 345
- JREmptyDataSource 152
- JRXmlDataSource 147
- subreports 309

datasets

- for Table component 195, 203
- in charts 209
- in crosstabs 253
- types 209

declaring objects 63

Design tab 28

Domains 189

drivers 126-128

E

elements

- adding and deleting 23-24
- attributes 38

containers 41

in cells 205

in reports 23

in tables 204

layout 41

palette 24

Properties view 41

Subreport 307

extensions 345

F

fields

- adding to reports 23
- CLOB 96
- double 98
- in Groovy expressions 98
- in Java expressions 98
- in SQL queries 64
- Java types vs. SQL types 129
- null 34, 98
- units of measure in 331

folders

- repository 185
- workspace 13

font extensions 101

- creating 107
- using in a report 112

fonts

- excluding scripts in font sets 106
- extensions 101
- font mappings 104
- font sets 106
- for PDF files 101
- Unicode 97
- uploading to JasperReports Server 115
- using font extensions 112

G

GeoAnalytics 289

getFieldValue 156

grid layout 41

Groovy 93

groups

- column groups 204, 206-207
- creating 316
- Group Band 30, 316

- Group Footer 30, 314
- Group Header 30, 314
- H**
- hardware requirements 13
- HTML5 charts
 - advanced settings 233
 - available chart types 221
 - creating 227
 - disabling chart types options 233
 - hyperlinks 232
 - multi-axis charts 238
 - overview 221
 - scatter charts 234
- hyperlinks 55
 - HTML5 charts 232
 - parameters in URL 57
 - URL 57
- hypertext See hyperlinks
- I**
- importing
 - data adapters 16
 - server connections from a workspace 15
 - settings 16
- Incrementer 264
- Information Links 157
- installation 12
- iReport, user interface comparison 29
- J**
- JasperReports
 - and Groovy 98
 - and Java 97
 - and JavaScript 99
 - compatibility between versions 17
 - documentation 339
 - expressions 93
 - extensions 345
 - library 33, 37, 339
 - license 339
 - properties 331
 - samples 331
- JasperReports Server
 - choosing a data source 175
 - connecting to 170
 - creating a Topic in Jaspersoft Studio 183
 - date/time stamp for scheduled reports 191
 - editing a JRXML file 186
 - editing a report unit 186
 - publishing report templates to 181
 - publishing reports to 174
 - repo syntax 190
 - repository 186
 - running a report 186
 - uploading fonts 115
- Jaspersoft OLAP prerequisites 12
- Jaspersoft Studio
 - installing 12
 - setting advanced properties 233
 - source code 18
- Java
 - JavaBeans data adapter 138
 - JavaScript 93
- JavaScript 61
- JDBC connections 125
- JDBC drivers 126-128
- JFreeChart 214
- JRChart 214
- JRDataSource 153-154, 156, 309, 345
- JREmptyDataSource 152
 - subreports 309
- JRExporter 345
- JRFileSystemDataSource 157
- JRViewer 345
- JRXML files
 - about 340
 - editing in the server 186
 - example 341
 - field names in 184
- JRxmlDataSource 147
- L**
- languages
 - Java types vs. SQL types 129
 - MDX 149
 - XPath 141
- layers in TIBCO Maps 293
- layout
 - restoring 28
 - tables 206

- locales
 - in Ad Hoc views 184
 - REPORT_LOCALE parameter 77
- M**
- map component (Google Maps)
 - and markers 272, 276
 - and paths 281
 - using expressions for properties 285
- map component (TIBCO GeoAnalytics)
 - and layers 293
 - and markers 294, 297
 - and paths 300
 - basic structure 290
 - configuring 290
 - overview 289
 - using expressions for properties 292
- markers
 - in Google Maps 272, 276
- markers in TIBCO Maps 294, 297
- measures
 - adding to a crosstab 265
- Microsoft SQL Server Analytic Services 148
- MongoDB
 - specifying queries 133
- N**
- net.sf.jasperreports.data.adapter property 123
- O**
- Outline view 28
- outputs 345
- P**
- page format 31
- palette 38
- parameters
 - adding and deleting 76
 - date/timestamp for scheduled reports 191
 - default 75, 77
 - in subreports 309
 - values 75, 84
- paths
 - in Google Maps 281
 - in TIBCO Maps 300

- PDF
 - font features 101
- perspective 28
- Pie3D 209
- prerequisites for Jaspersoft OLAP 12
- Preview tab 28
- printing 68, 306
- Problems view 29
- Project Explorer 28
- projects
 - data adapters and 120
 - importing 13
 - importing from a workspace 14
 - upgrading 13
- properties
 - advanced 233
 - JasperReports 331
 - Properties view 28, 41
 - report properties 28, 31
 - setting on multiple levels 233
- publishing
 - report templates 181
 - reports 174
- Q**
- queries
 - fields 129
 - MongoDB 133
 - results 130
 - specifying 129, 138
 - SQL 129, 138
- query languages
 - types 66
 - XPath 146
- R**
- records, sorting and filtering 130
- repo syntax 190
- report editor 28
- Report state 29
- report templates
 - and JSS 182
 - in JasperReports Server 179
 - publishing to JasperReports Server 181
- reports
 - charts 209

- crosstabs 251
 - data adapters and 123
 - output files 345
 - page format 31
 - properties 28, 31
 - publishing to JasperReports Server 174, 177
 - setting advanced properties 233
- repository
 - in JasperReports Server 186
- Repository Explorer 28
- resources
 - adding to Jaspersoft Studio from the server 185
 - and Ad Hoc views 184
 - selecting 177
- restoring windows and views 28
- S**
- samples 331
- scatter charts 234
- scripting languages 93
- Select Resources window 177
- settings
 - importing from a workspace 16
- software requirements 12
- source code 18
- Source tab 28
- Spotfire Information Links 157
- SQL queries
 - field types 129
 - fields 64
 - results 130
 - specifying 129, 138
- subdatasets 195, 203
- Subreport element 307
- subreports
 - about 303
 - and XML data sources 146
 - creating 303, 306
 - data sources 309
 - dimensions in master report 307
 - empty datasource in 309
 - expression property 308
 - in bands 30
 - JDBC connections in 131
 - parameters 309
 - printing 306
 - properties 307
 - Subreport Wizard 303
- T**
- table component 195
 - editing 206
 - layout 206
 - table styles 201
- Test button 128
- text See fonts
- themes 218
- themes, for charts 187
- TIBCO Spotfire Information Links 157
- time fields
 - in crosstabs 254
- Topics 183
 - field names in 184
 - localization 184
- U**
- Unicode 97
- W**
- workspace
 - importing projects 13-14
 - upgrading 13-14
- X**
- XML
 - data sources 141
 - report file 339
- XML/A 148-149
- XPath 141, 146

