The Definitive Guide to iReport™

Giulio Toffoli

Apress®

The Definitive Guide to iReport™

ISBN-13 (pbk): 978-1-59059-928-0

ISBN-10 (pbk): 1-59059-928-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Steve Anglin
Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jason Gilmore, Jonathan Hassell, Chris Mills, Matthew Moodie, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Kylie Johnston
Copy Editor: Ami Knox
Assistant Production Director: Kari Brooks-Copony
Production Editor: Laura Esterman
Compositor: Molly Sharp
Proofreader: Elizabeth Berry
Indexer: Brenda Miller
Artist: April Milne
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail `orders-ny@springer-sbm.com`, or visit `http://www.springeronline.com`.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail `info@apress.com`, or visit `http://www.apress.com`.

The source code for this book is available to readers at `http://www.apress.com` in the Source Code/Download section.

To my wonderful wife, Caterina

Contents at a Glance

Contents

Foreword

In the early days of JasperReports™ development, users had to manually edit report template files, and the only visual tool to help them see what they were doing was the built-in report design preview tool.

Of course, JasperReports, as a content-rendering library, was, and still is, mainly a developer tool that provides reporting functionality to Java™ applications that need it. Developers are used to working with source code files and XML, so not having a visual tool at the time did not prevent them from downloading JasperReports and using it right away.

But as things moved forward and JasperReports provided new features to its users, it soon became apparent there was a need for more advanced tools that would simplify report design work and make it more enjoyable.

I started to work on such a tool myself at the time and realized that the complexity of the project and the effort required would probably distract me from what I was doing on JasperReports. I decided to not take up this challenge of creating a visual designer for JasperReports and continued to concentrate on the report engine.

Fortunately, Giulio Toffoli came to the rescue, and within a relatively short time he was able to put together a very useful and intuitive visual designer for creating and testing report templates. This was a major boost to the JasperReports community in general, and the iReport™ designer quickly became the most popular GUI tool for JasperReports.

Since then, things have moved along swiftly, with Giulio making sure that iReport continues to evolve and keeps pace with JasperReports, always being up-to-date and supporting the newest features that were added to the reporting engine.

Things got even better back in 2005, when the iReport project joined the JasperReports project and both became core components in what we now know as the JasperSoft® Business Intelligence Suite. This suite is a comprehensive set of tools for integrated reporting, analysis, and data integration. In addition to JasperReports and iReport, the suite includes JasperServer™, a high-performance business intelligence and report server; JasperAnalysis™, an OLAP engine; and JasperETL™, a ready-to-run data integration platform.

The JasperSoft Business Intelligence Suite began as a complete open source reporting and BI solution. Today, JasperSoft offers both open source and professional versions of the suite. The professional version includes JasperStudio™, the professional edition of iReport.

Nowadays, iReport is not only a simple visual report designer. It is a complete and integrated reporting tool that offers various services including report preview in all JasperReports export formats, data connectivity, wizards, and support for visually designing charts and crosstabs. It is also the report designer for JasperServer and Jasper4Salesforce™, with which it interacts through one of its easy-to-use plug-in modules.

Not only has Giulio done a great job creating iReport and making it the popular and very useful tool it is today, but he has also done an amazing job documenting it. This book is a complete reference to all iReport functionality and features and helps you get a quick start with reporting, even without being familiar with the JasperReports engine. With iReport and its documentation, Giulio helped making JasperReports useful to a wider audience, not only to Java developers. I hope you'll enjoy this book as much as I have enjoyed working with Giulio all these years.

Teodor Danciu

Founder and architect of JasperReports, JasperSoft

About the Author

GIULIO TOFFOLI is a senior software engineer at JasperSoft Corporation, where he serves as the iReport project leader. He has been developing Java applications since 1999 and founded the iReport project in 2001. During this time, Giulio has enjoyed designing complex software architectures and implementing custom software solutions with a focus on desktop and multi-tiered, web-based, client-server applications using Java (J2EE/JEE) and open source technologies. Giulio has a degree in computer science from the University of Bologna and currently resides in Italy.

Acknowledgments

iReport contains code and ideas from many people. Though I run the risk of forgetting somebody, I would like to thank the following people for their contribution to this project: Teodor Danciu, Alexander, Andre Legendre, Craig B. Spengler, David Walters, Egon R. Pereira, ErtanO, G. Raghavan, Heiko Wenzel, Kees Kuip, Octavio Luna, Peter Henderson, Vinod Kumar Singh, Wade Chandler, Erica Pastorello, and all reviewers. A special thanks goes to Kristen Kelleher from JasperSoft and all the Apress guys for the amazing job done for the latest release of this book.

Introduction

iReport is an open source program that can create complex reports which can use every kind of Java application through the JasperReports library. It is written in 100% pure Java and is distributed with source code according to the GNU General Public License. JasperStudio is the professional edition of iReport; it is essentially the same application, but is commercially supported by JasperSoft Corporation and released as part of the JasperSoft Business Intelligence Suite, a comprehensive set of tools for integrated reporting, analysis, and data integration. In addition to JasperStudio, the suite is comprised of JasperServer, a high-performance business intelligence and report server; JasperAnalysis, an OLAP engine to drill down and slice and dice data; and JasperETL, a ready-to-run data integration platform providing data extract-transform-load (ETL) capabilities.

Through an intuitive and rich graphic interface, iReport lets you rapidly create any kind of report very easily. iReport enables engineers who are just learning this technology to access all the functions of JasperReports as well as helping skilled users to save a lot of time during the development of very elaborate reports.

This guide refers to the 2.0.0 version of iReport, but a great portion of this information is directly applicable to earlier versions; my commitment is to keep this guide as up-to-date as possible with future iReport versions.

Features of iReport

The following list describes some of the most important features of iReport:

- 100% support of JasperReports XML tags.
- WYSIWYG editor for the creation of reports. It has complete tools for drawing rectangles, lines, ellipses, text fields, labels, charts, subreports, and bar codes.
- Built-in editor with syntax highlighting for writing expressions.
- Support for Unicode and non-Latin languages (Russian, Chinese, Japanese, Korean, etc.).
- Browser for document structure.
- Integrated report compiler, filler, and exporter.
- Support for all databases accessible by JDBC.
- Virtual support for all kinds of datasources.
- Wizard for creating reports automatically.
- Support for subreports.
- Backup feature of source files.
- Support for document templates.
- TrueType fonts support.
- Support for localization.
- Extensibility through plug-ins.

- Integrated support for scriptlets.
- Support for charts.
- Management of a library of standard objects (e.g., numbers of pages).
- Drag-and-drop functionality.
- Unlimited undo/redo.
- Wizard for creating crosstabs.
- Styles library.
- Docking system.

The iReport team is composed of many skilled and experienced programmers who come from every part of the world. They work daily to add new functionalities and fix bugs.

iReport Community

The iReport web site is at `http://ireport.sourceforge.net`. If you need help with iReport, there is a discussion forum in English: `http://www.jasperforge.org/index.php?option=com_joomlaboard&Itemid=215&func=showcat&catid=9`. This is the place where you can send requests for help and technical questions about the use of the program, as well as post comments and discuss implementation choices or propose new functionalities. There is no guarantee for a prompt reply, but requests are usually satisfied within a few days' time. This service is free. If you need information concerning commercial support, you can write to `sales@jaspersoft.com`.

Please report bugs at the following address: `http://jasperforge.org/sf/tracker/do/listArtifacts/projects.ireport/tracker.bugs`.

In the project site, there is a system to send requests for enhancement (RFE). There is also the ability to suggest patches and integrative code.

All members of the iReport team keep in serious consideration all suggestions, criticism, and advice coming from iReport and JasperStudio users.

Downloading the Code

The source code for this book is available to readers at `http://www.apress.com` in the Downloads section of this book's home page. Please feel free to visit the Apress web site and download all the code there. You can also check for errata and find related titles from Apress.

CHAPTER 1

Getting Started

In this chapter, you'll see what the requirements are for using iReport, the way to obtain the binary distribution and the sources, and how to compile and install it.

Requirements

iReport needs Sun Java 2 SDK 1.5 or newer; in order to compile report scriptlets, it is necessary to install the complete distribution of Java 2 (Java SE Development Kit, or JDK), not only a runtime environment (Java Runtime Environment, or JRE). If you want to compile iReport sources, you should install Jakarta Ant version 1.6 or newer.

As for hardware, like all Java programs, iReport eats a lot of RAM, and so it is necessary to have at least 256MB of memory and about 20MB of free space on disk.

Downloading iReport

It is possible to download iReport from the iReport project page on SourceForge where you can always find the last released iReport distribution (`http://sourceforge.net/projects/ireport`). Four different distributions are available:

`iReport-`*`x.x.x`*`.zip`: This is the official binary distribution in ZIP format.

`iReport-`*`x.x.x`*`.tgz`: This is the official binary distribution in TAR GZ format.

`iReport-`*`x-x-x`*`-src.zip`: This is the official distribution of sources in ZIP format.

`iReport-`*`x.x.x`*`-windows-installer.exe`: This is the official Win32 installer.

`x.x.x` represents the version number of iReport. Every distribution contains all needed libraries from third parties necessary to use the program and additional files, such as templates and base documentation in HTML format.

Accessing Source Code

If you want a more up-to-date version of sources, you can directly access the SVN repository. In this case, it is necessary to have an SVN client (my favorite is Tortoise SVN). If you don't have one, you need to create an account on `http://jasperforge.org/sf/projects/ireport` in order to access the repository.

■ **Caution** iReport source code is no longer available on SourceForge CVS server.

The URL of the SVN repository of iReport is `http://scm.jasperforge.org/svn/repos/ireport/trunk/iReport2`.

Compiling iReport

The distribution with sources contains a `build.xml` file that is used by Jakarta Ant to compile and start iReport and/or to create different distributions of the program.

Download `iReport-x.x.x-src.zip`, unzip it into the directory of your choice, for example, `c:\devel` (or `/usr/devel` on a Unix system). Open a command prompt or a shell, go to the directory where the archive was uncompressed, go to the iReport directory, and enter

```
C:\devel\iReport-2.0.0>ant iReport
```

The sources, which stay in the `src` directory, will be compiled into the `classes` folder, and iReport will start immediately.

Setting Up the Start and Base Configuration

If you preferred downloading the binary version of iReport, uncompress the downloaded archive into the directory of your choice, for example, `c:\devel` (or `/usr/devel` on a Unix system). Open a command prompt or a shell, go to the directory where the archive was uncompressed, go to the iReport directory, and enter

```
C:\devel\iReport-2.0.0>iReport.bat
```

or on Unix:

```
$ ./iReport.sh
```

(In this case, it should be preceded by a `chmod +x` if the file is not executable.)

The Windows Installer and iReport.exe

Starting from version 1.2.3, iReport provides a Windows installer created using NSIS, the popular tool from Nullsoft (`http://nsis.sourceforge.net/Main_Page`).

To install iReport, double-click `iReport-x.x.x-windows-installer.exe` to bring up the screen shown in Figure 1-1.

Click Next, review the license agreement as shown in Figure 1-2, and click I Agree if you accept the terms.

Figure 1-1. *iReport Setup Wizard—Step 1*

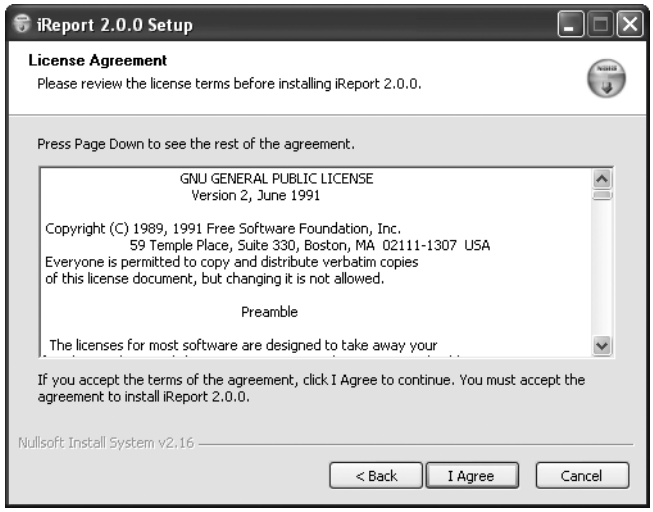

Figure 1-2. *iReport Setup Wizard—Step 2*

iReport can be installed with and without the source code, as shown in Figure 1-3. If you want to install the sources too, you can follow the instructions about how to compile iReport as discussed earlier in the section "Compiling iReport."

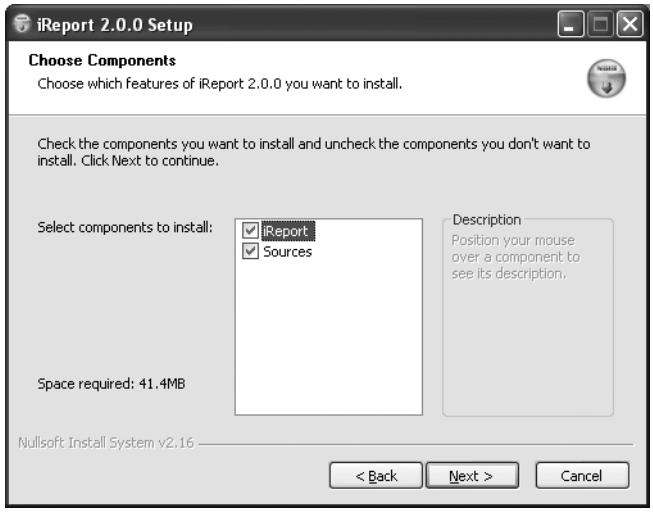

Figure 1-3. *iReport Setup Wizard—Step 3*

At the end of the installation process, you get a new menu item in the program files menu (for instance, on a Windows system, Start ➤ All Programs ➤ JasperSoft ➤ iReport-*x.x.x*).

You can have more than one version of iReport installed on your machine at the same time, but all these versions will share the same configuration files.

The installer creates a shortcut to launch iReport, linking the shortcut to `iReport.exe` (present in the program home directory); this Win32 binary version is really a wrapper created using JSmooth (`http://jsmooth.sourceforge.net/`). From iReport 1.2.5 on, the executable created by JSmooth is able to automatically load all JAR files located in the `lib` directory.

■ **Caution** If you experience some problems like `ClassNotFound` exceptions using `iReport.exe`, try using `iReport.bat` instead.

First iReport Execution

On the first execution, iReport will create a directory named `.ireport` in the user's home directory. Here the personal settings and the configuration of the program are saved. If it is not possible to create this folder, this could cause undesirable effects in the program, and it may not be possible to save the configuration files. In this case, it could be necessary to create the directory manually.

■ **Note** Before proceeding to the program configuration, it is necessary to copy the `tools.jar` file, normally present in the `lib` directory of Sun JDK, into the `iReport lib` directory. The absence of this file can produce some exceptions during the compilation of a report (carried out by using classes contained in this Java library). On Mac OS X, the `tools.jar` file does not exist, but there is a `classes.jar` file, which contains the required classes to compile.

The iReport initial configuration consists of setting up the programs to run for viewing the produced documents according to their file formats, selecting the language to use, and indicating

where to store compiled files. Other configuration settings will be explained subsequently. In order to proceed to the configuration, run iReport and select Options ➤ Settings to bring up the window shown in Figure 1-4.

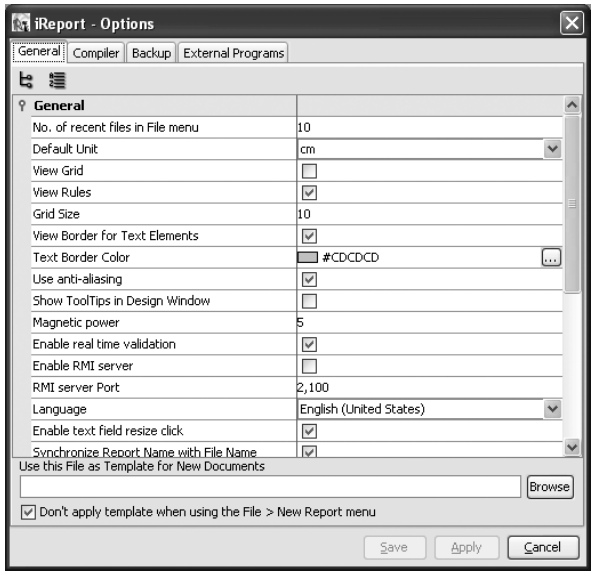

Figure 1-4. *Options window—General options*

Select the language you prefer and go to the Compiler tab, shown in Figure 1-5.

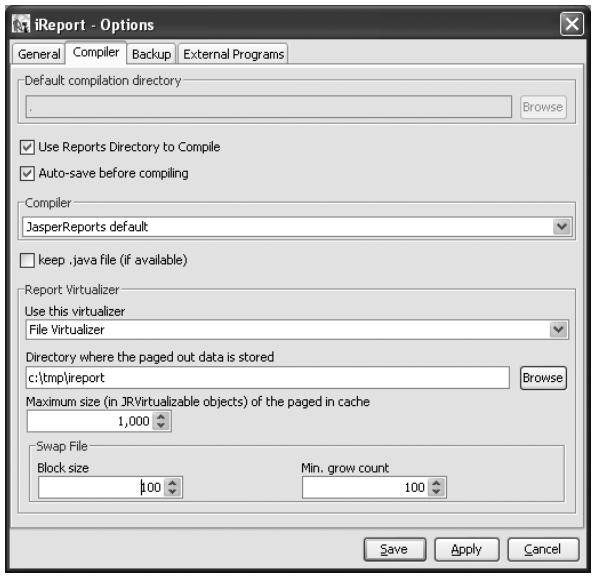

Figure 1-5. *Options window—Compiler options*

In the Compiler tab, you can set where iReport stores JASPER files that are compiled. By default, iReport uses the current directory as the destination for compiled files. Often it is useful to specify a particular directory for saving compiled files; this directory is usually the same one in which the source of the report is located. In this case, check the Use the reports directory for compiles check box.

In this tab, you can also set a specific compiler to use. JasperReports provides different ways to compile your report. The current JasperReports default compiler is the JDT Compiler. If you use the Java compiler (javac), you need to have the `tools.jar` file in your classpath (you can find this file in your JDK).

I suggest you use the default compiler (JDT). Using this compiler, iReport will be able to point you to errors using clickable items in the Problems tab (see Figure 1-6).

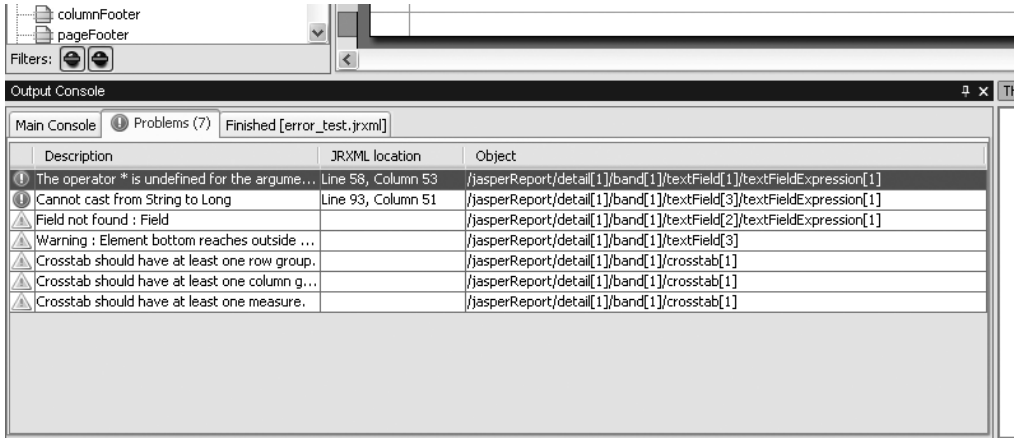

Figure 1-6. *Clickable errors produced using the JDT compiler*

If Groovy is used as the language for expression, a special compiler is used instead of the one specified.

See Chapter 19 for details on the other options present in the Compiler tab.

Complete the configuration by going to the External Programs tab and specifying the external programs to use with different output formats of reports and the editor to use for modifying XML source (see Figure 1-7).

Restart iReport to set all chosen options.

Test the configuration by creating a new blank report (select File ➤ New Document) and confirming all features proposed for the new report. Then click the Run button on the toolbar, shown here:

If everything is OK, you will be prompted to save the report in a JRXML file, and a corresponding JASPER file will be created and a preview of a blank page will appear. This means that iReport has been installed and configured correctly.

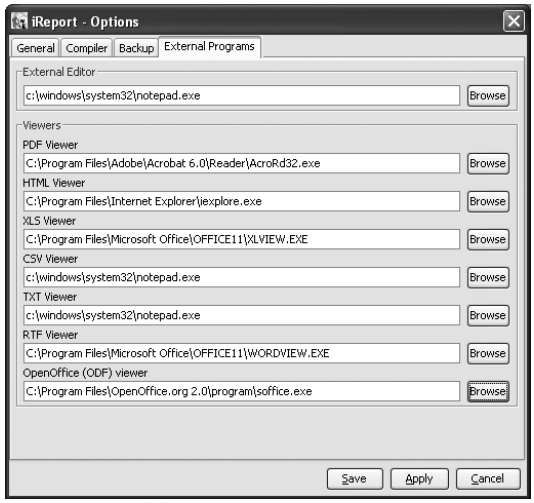

Figure 1-7. *Options window—External Programs options*

Creating a JDBC Connection

The most common datasource for filling a report is typically a relational database. Next, you will see how to set up a JDBC connection in iReport. Select Data ➤ Connections/Datasources and click the New button in the window with the connections list. A new window will appear for the configuration of the new connection (see Figure 1-8). Select Database JDBC connection and click Next. In the new frame, shown in Figure 1-9, enter the connection name (e.g., "My new connection") and select the right JDBC driver. iReport recognizes the URL syntax of many JDBC drivers. You can automatically create the URL by entering the server address and database name in the corresponding boxes and clicking the Wizard button. To complete the connection configuration, enter the username and password for access to the database. If you want to save the password, select the Save password check box.

Figure 1-8. *Creating a JDBC connection*

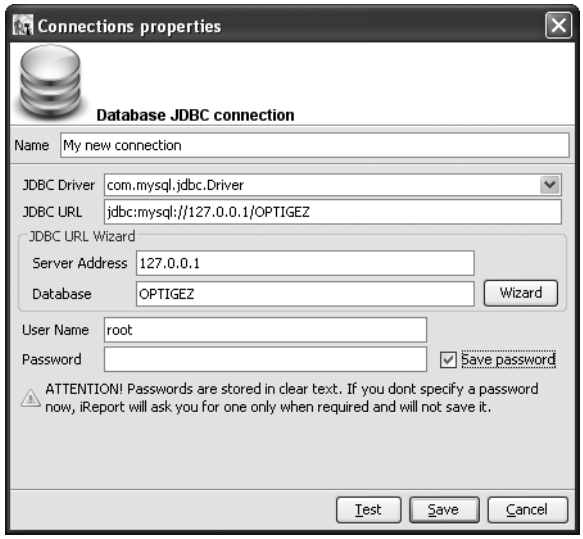

Figure 1-9. *Specifying the properties for a JDBC connection*

Test the connection by clicking the Test button. It is better to test the connections before saving and using them.

iReport is shipped with only the JDBC driver for the MySQL database and the HSQL database engine (HSQLDB). If during the test there is a `ClassNotFound` error, it is possible that there is no JAR archive (or ZIP) in the classpath that contains the selected database driver. Without closing iReport, copy the JDBC driver into the `lib` directory and retry; the new JAR will be automatically located and loaded by iReport. In Chapter 9, I will explain extensively all the configuration modalities of the datasources.

At the end of the test, click the Save button to store the new connection.

In this way, you have created a new datasource, so you have to tell iReport to use it as a predefined datasource. Select Data ➤ Connections/Data Sources and then specify the new datasource. The new connection will be automatically considered the active connection.

In general, to set the active connection, you can use the drop-down list in the main iReport toolbar (see Figure 1-10).

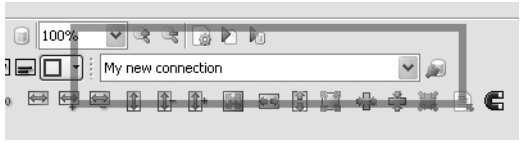

Figure 1-10. *Specifying a JDBC connection*

Another way is by selecting Data ➤ Set Active Connection to bring up the dialog box shown in Figure 1-11.

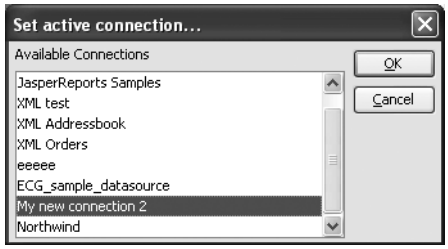

Figure 1-11. *List of the available datasources*

Then select your connection from the list and click the OK button. From now on iReport will use this connection for every operation that needs access to the database (in particular the acquisition of the fields selected through SQL queries and prints creation).

Creating Your First Report

Now that you have installed and configured iReport, and prepared a JDBC connection to the database, you will proceed to create a simple report using the *Report Wizard.*

For it and for many other following examples, you will use HSQLDB, a small relational database written in Java and supplied with a JDBC driver. To be able to use it, copy the `hsqldb.jar` file into the `lib` directory (this file is the database driver, and it is already present in all the iReport distributions from version 0.3.2). In order to know more about this small jewel, please visit the HSQLDB project site at this address: `http://hsqldb.sourceforge.net`.

In order to set up the database connection used in this example, use the parameters listed in Table 1-1.

Table 1-1. *Connection Parameters*

Properties	Value
Name	Northwind
JDBC Driver	`org.hsqldb.jdbcDriver`
JDBC URL	`jdbc:hsqldb:c:/devel/northwind/northwind`
Username	sa
Password	

When the password is blank, as in this case, remember to select the Save password check box when configuring the connection.

Select File ➤ Report Wizard. This loads a tool for the step-by-step creation of a report, starting with a query insertion (see Figure 1-12).

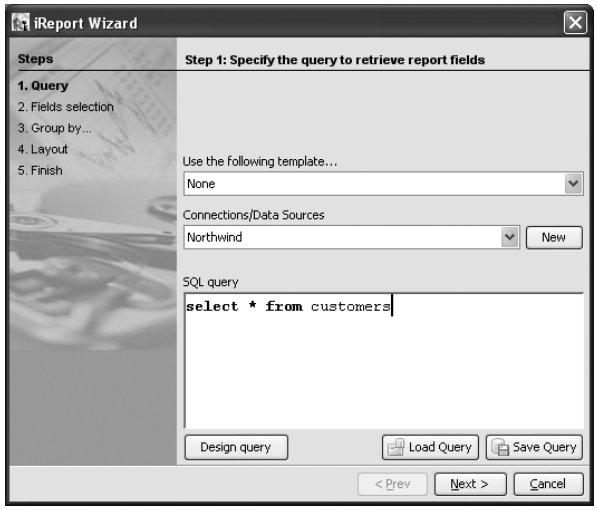

Figure 1-12. *Report Wizard—query insertion*

In the text area, insert a SQL query in order to select data that will go to fill your report, for example:

```
select * from customers order by country
```

and click Next. The clause "order by" is important to the following choice of the grouping (I will discuss the details a little later). iReport will read the fields of the customers table, and it will present them in the next screen of the Report Wizard, as shown in Figure 1-13.

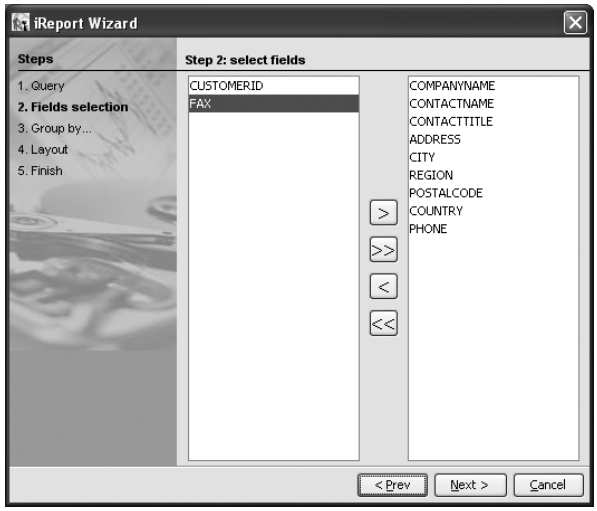

Figure 1-13. *Report Wizard—report fields selection*

Select the fields you wish to include and click Next. Now that you have selected the fields to put in the report, you will be prompted to choose what fields you wish to group by, if any (see Figure 1-14).

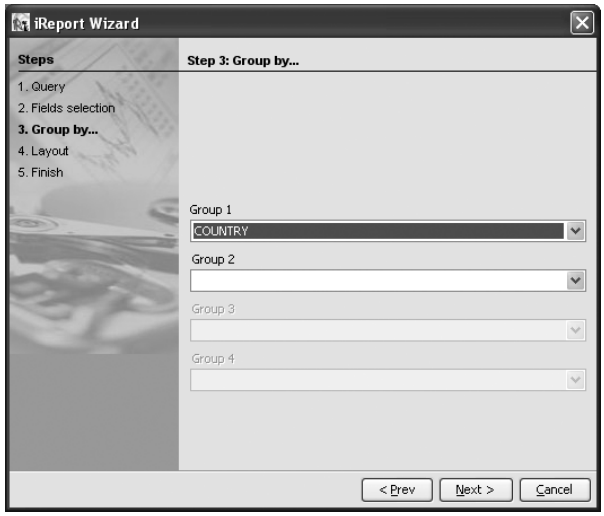

Figure 1-14. *Report Wizard—groupings*

Using the wizard, it is possible to create up to four groups. Others can be defined afterwards. (In fact, it is possible to set up an arbitrary number of groupings.)

For this first report, define a simple grouping on the COUNTRY field, as shown in Figure 1-14.

The next step of the wizard allows you to select the print *template*, which is a model that can be used as the base for the creation of the report (see Figure 1-15). With iReport, some very simple templates are supplied, and you will see how to create some new ones. For the moment, it is enough to know that there are two types of templates: the *tabular* templates, in which every record occupies one line like in a table; and the *columnar* templates, in which the fields of the report are displayed in columns.

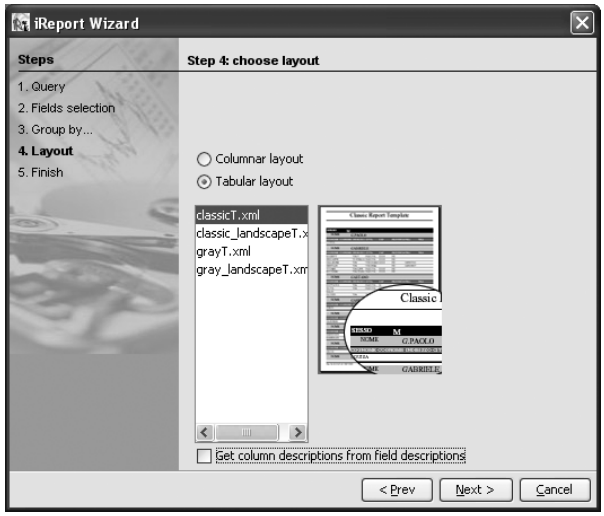

Figure 1-15. *Report Wizard—choosing a template*

For your first report, select a tabular template, preferably the classicT one (here, "T" stands for tabular).

After you have chosen the template, click Next. The last screen of the wizard will appear, and it will tell you the outcome of the operation. Click Finish to create the report, which will appear in the iReport central area, ready for execution, as shown in Figure 1-16.

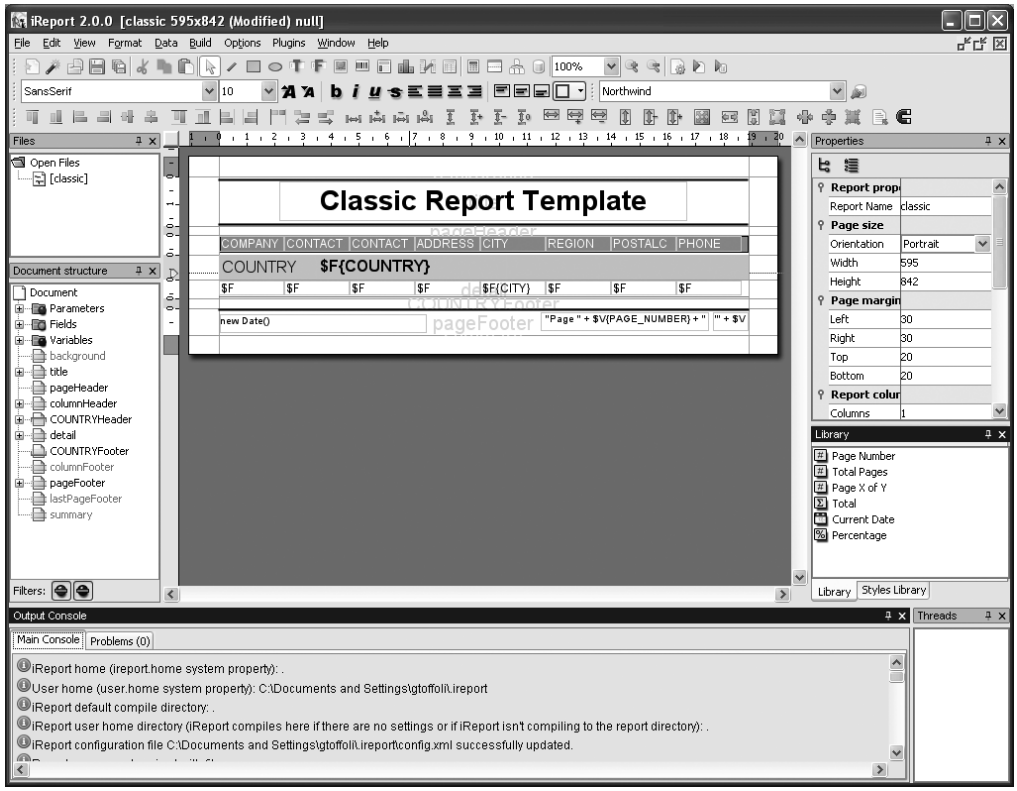

Figure 1-16. *iReport main window*

Before being able to execute the final print, you will have to save the report source created through the wizard and compile it. These operations can be done all at once by clicking the Run report using a connection button, shown here, that is on the toolbar:

After you click the Run report using a connection button, you will be asked for the name under which to save the file. Save the file with the name `report1.jrxml`. In the *console*, which is in the part below the main window, some messages will appear. They will tell you about what is happening: the report will be compiled, created, and finally *exported* (see Figure 1-17).

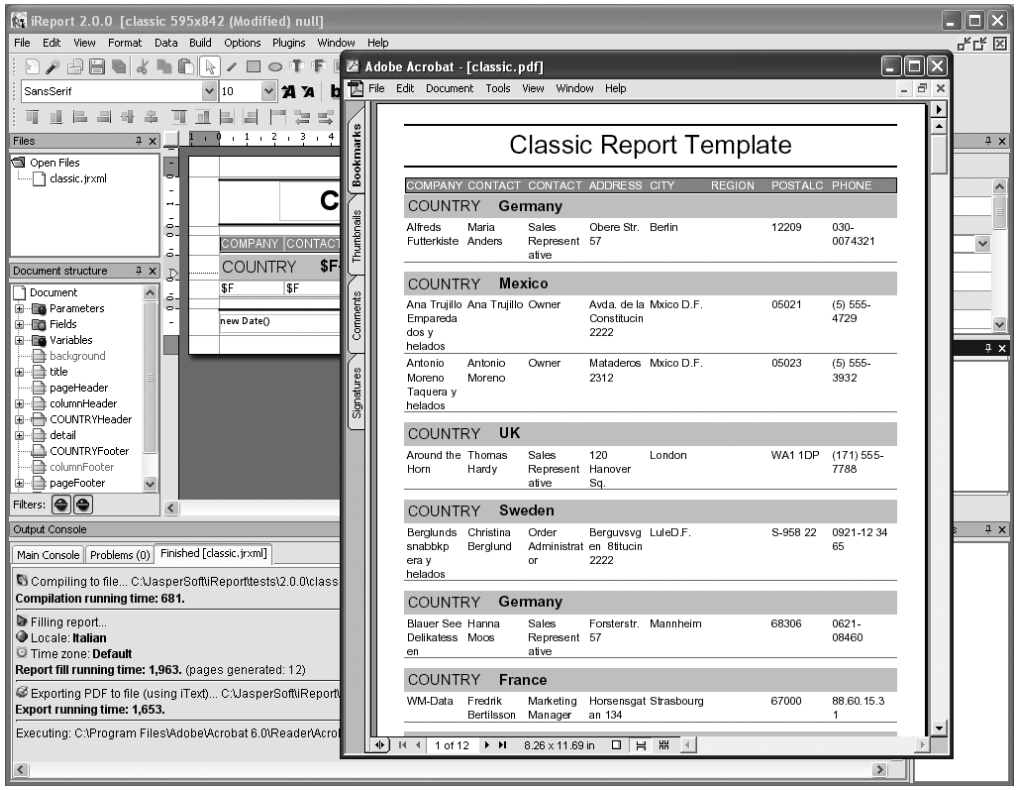

Figure 1-17. *Your first report in PDF format*

At the end of the operation, if everything is OK, the report will be shown in the default program for opening PDF files. The PDF format is the predefined format of export.

Specifying Startup Command-Line Options

It is possible to specify some startup parameters on the command line. These parameters are not case sensitive. They can be truncated so long as they remain unambiguous. For example, the option `-ireport-home` can be specified as `-i` because no other option starts with the letter "i," and therefore the command-line interpreter will successfully interpret `-i` as the truncation of the `-ireport-home` option.

The Boolean options can be specified using both the contracted form `-option` and the extensive form `-option=true` or `-option=false` according to necessity.

It is possible to obtain the options list by entering

```
iReport.bat -?
```

or

```
./iReport.sh -?
```

Table 1-2 explains the different available options. It refers to the iReport 2.0.0 version, and it may not be complete regarding successive versions.

Table 1-2. *Command-Line Options*

Option	Description
`-beanClass <`*`className`*`>`	Shows the specified class in the Bean Datasource tab (in the query editor window).
`-config-file <`*`fileName`*`>`	Specifies the file name for loading an alternate configuration. The file is never changed from iReport, which will save an eventual modified configuration in the canonical directory, that is, the user home `/.ireport`.
`-embedded`	Avoids application exit when the main window is closed.
`-ireport-home <`*`dir`*`>`	Specifies the program directory.
`-no-splash`	Avoids showing the splash window at startup.
`-temp-dir <`*`dir`*`>`	Specifies the directory where temporary files will be saved.
`-user-home <`*`dir`*`>`	Specifies the user home. The predefined directory is the one stored into the `user.home` system property.
`-version`	Outputs the version immediately.
`-webstart`	Specifies that iReport will use a Java Web Start–friendly class loader.

If Ant is used, it is not possible to specify these options directly from the command line, but it will be necessary to modify the `build.xml` file by adding the `<arg>` tags useful to the Java task that runs iReport.