eclipse

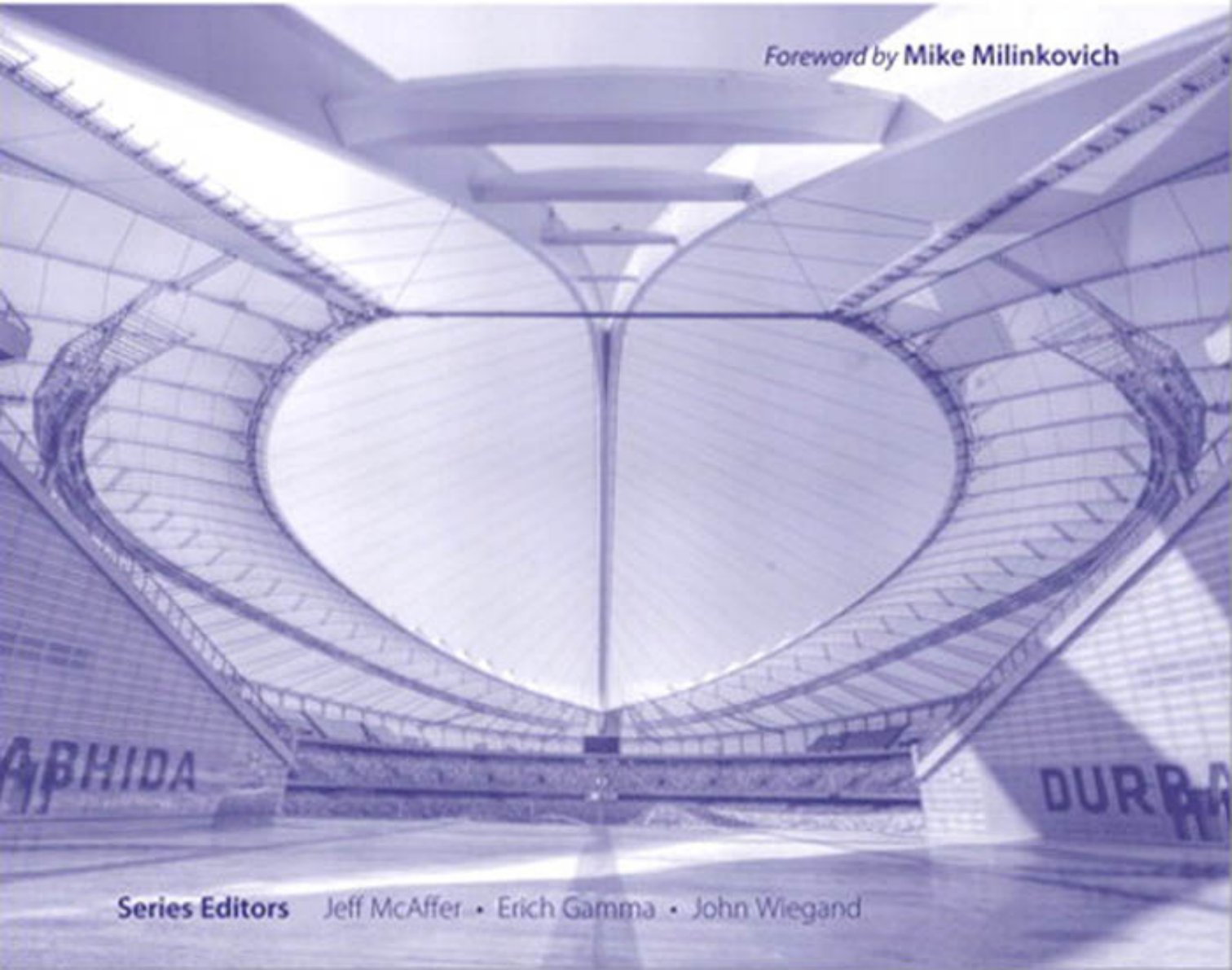
the eclipse series

Integrating and Extending BIRT

Third Edition

Jason Weathersby • Tom Bondur • Iana Chatalbasheva

Foreword by **Mike Milinkovich**



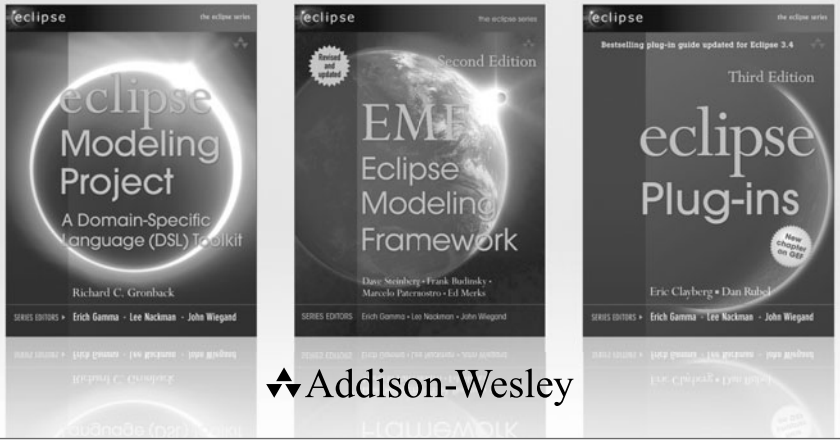
Series Editors Jeff McAffer • Erich Gamma • John Wiegand

Integrating and Extending BIRT

Third Edition

The Eclipse Series

Eric McAffer, Erich Gamma, John Wiegand, Series Editors



Addison-Wesley

Visit **informit.com/series/eclipse** for a complete list of available publications.

Eclipse is a universal, multilanguage software development environment—an open, extensible, integrated development environment (IDE)—that can be used for anything. Eclipse represents one of the most exciting initiatives to come from the world of application development, and it has the support of leading companies and organizations in the technology sector. Eclipse is gaining widespread acceptance in both commercial and academic arenas.

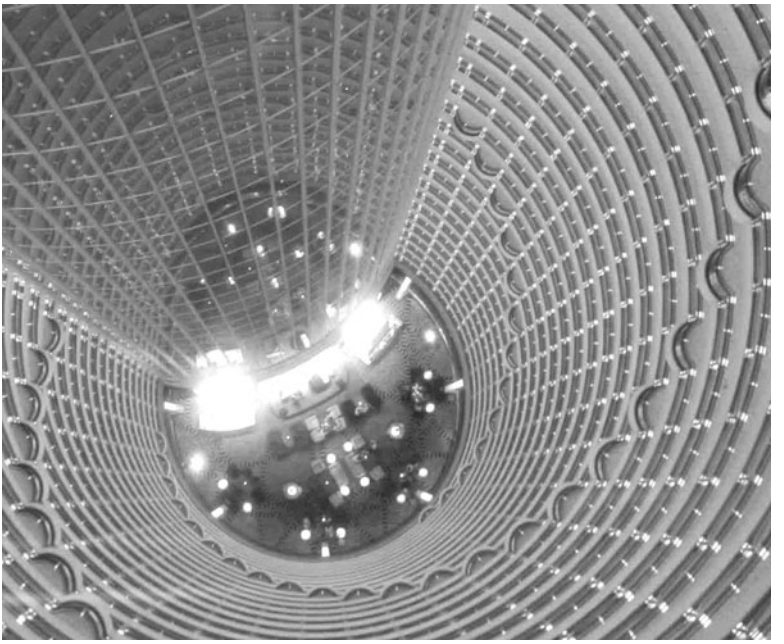
The Eclipse Series is the definitive collection of publications dedicated to the Eclipse platform. Books in this series bring you key technical information, critical insight, and the practical advice you need to build tools to support this revolutionary open-source platform.

PEARSON

Addison-Wesley | Cisco Press | EXAM/CRAM | IBM Press | QUE | PRENTICE HALL | SAMS | Safari Books Online

Integrating and Extending BIRT

Third Edition



Jason Weathersby • Tom Bondur • Iana Chatalbasheva

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Control Number: 2011932838

ISBN-13: 978-0-321-77282-4
ISBN-10: 0-321-77282-2

Text printed on recycled paper in the United States at Courier in Westford, Massachusetts.
First printing, August 2011

Contents

Foreword

It is a common misconception that Eclipse projects are focused on simply providing great tools for developers. Actually, the expectations are far greater. Each Eclipse project is expected to provide both frameworks and extensible, exemplary tools. As anyone who has ever tried to write software with reuse and extensibility in mind knows, that is far more difficult than simply writing a tool.

"Exemplary" is one of those handy English words with two meanings. Both are intended in its use above. Eclipse projects are expected to provide tools that are exemplary in the sense that they provide an example of the use of the underlying frameworks. Eclipse tools are also intended to be exemplary in the sense that they are good and provide immediate utility to the developers who use them.

Since its inception, the BIRT project has worked hard to create both reusable frameworks and extensible tools. This book focuses primarily on how to extend BIRT and how to use BIRT in your own applications and products. As such, it illustrates BIRT's increasing maturity and value as an embedded reporting solution.

As Executive Director of the Eclipse Foundation, I'm pleased with the tremendous progress the BIRT team has made since the project's inception in September of 2004, and I'm equally pleased with the vibrant community that has already grown up around it. As you work with BIRT and the capabilities that are described in this book, I'd encourage you to communicate your successes back to the community, and perhaps consider contributing any interesting extensions you develop. The BIRT web site can be found here:

`http://www.eclipse.org/birt`

It includes pointers to the BIRT newsgroup, where you can communicate and share your results with other BIRT developers, and pointers to the Eclipse installation of Bugzilla, where you can contribute your extensions. If you like BIRT—and I am sure this book will help you learn to love it—please participate and contribute. After all, it is the strength of its community that is the true measure of any open source project's success.

Mike Milinkovich
Executive Director, Eclipse Foundation

This page intentionally left blank

About this book

The second of a two-book series on business intelligence and reporting technology, *Integrating and Extending BIRT*, introduces programmers to BIRT architecture and the reporting framework. Its companion book, *BIRT: A Field Guide*, shows report developers how to create reports using the graphical tools of BIRT Report Designer. Built on the open-source Eclipse platform, BIRT is a powerful reporting system that provides an end-to-end solution, from creating and deploying reports to integrating report capabilities in enterprise applications.

BIRT technology makes it possible for a programmer to build a customized report using scripting and BIRT APIs. This book informs report developers about how to write scripts that:

- Customize the report-generation process
- Incorporate complex business logic in reports

This book also informs application developers about how to:

- Debug and deploy reports
- Integrate reporting capabilities into other applications
- Extend BIRT functionality

A programmer can extend the BIRT framework by creating a new plug-in using the Eclipse Plug-in Development Environment (PDE). This book provides extensive examples that show how to build plug-ins to extend the features of the BIRT framework. The source code for these examples is available for download at http://www.actuate.com/birt/contributions.

The topics discussed in this book include:

- Installing and deploying BIRT
- Deploying a BIRT report to an application server
- Understanding BIRT architecture

- Scripting in a BIRT report design
- Integrating BIRT functionality into applications
- Working with the BIRT extension framework

This revised BIRT 2.6 edition adds the following new content:

- Updated architectural diagrams
- Expanded scripting examples
- Debugging Event Handlers
- Developing an advanced report item with data binding
- Developing a Data Extraction Extension
- Developing a Charting Extension

Who should read this book

This book is intended for people who have a programming background. These readers can be categorized as:

- Embedders and integrators

  These individuals work with the software to integrate it into their current application infrastructure.

- Extenders

  These individuals leverage APIs and other extension points to add capability or to establish new interoperability between currently disparate components or services.

To write scripts in report design, you need knowledge of JavaScript or Java. More advanced tasks, such as extending BIRT's functionality, require Java development experience and familiarity with the Eclipse platform.

Contents of this book

This book is divided into several parts. The following sections describe the contents of each of the parts and chapters.

Part 1, Installing and Deploying BIRT

Part 1 introduces the currently available BIRT reporting packages, the prerequisites for installation, and the steps to install and update the packages. Part 1 includes the following chapters:

- *Chapter 1. Introducing BIRT Report Designers.* BIRT provides a number of separate packages as downloadable archive (.zip) files on the Eclipse web

site. Some of the packages are stand-alone modules, others require an existing Eclipse environment, and still others provide additional functionality to report developers and application developers. This chapter describes the prerequisites for each of the available packages.

- *Chapter 2. Installing a BIRT Report Designer.* BIRT provides two report designers as separate packages, which are downloadable archive (.zip) files on the Eclipse web site. This chapter describes the steps required to install each of the available report designers.
- *Chapter 3. Installing Other BIRT Packages.* This chapter describes the steps required to install and update each of the available packages.
- *Chapter 4. Deploying a BIRT Report to an Application Server.* This chapter introduces the distribution of reports through an application server such as Apache Tomcat, IBM WebSphere, or BEA WebLogic. The instructions in the chapter provide detailed guidance about deploying a BIRT report to Apache Tomcat version 6.0. From those instructions, a developer can infer how to deploy to other versions.

Part 2, Understanding the BIRT Framework

Part 2 introduces the BIRT architecture and the Report Object Model (ROM) and provides background information that helps programmers design or modify reports programmatically, instead of using the graphical tools in BIRT Report Designer. Part 2 includes the following chapters:

- *Chapter 5. Understanding the BIRT Architecture.* This chapter provides an architectural overview of BIRT and its components, including the relationships among the BIRT components and BIRT's relationship to Eclipse and Eclipse frameworks. Architectural diagrams illustrate and clarify the relationships and workflow of the components. The chapter also provides brief overviews of all the major BIRT components.
- *Chapter 6. Understanding the Report Object Model.* This chapter provides an overview of the BIRT ROM. ROM is a specification for a set of XML elements that define both the visual and non-visual elements that comprise a report design. The ROM specification includes the properties and methods of those elements, and the relationships among the elements.

Part 3, Scripting in a Report Design

Part 3 describes how a report developer can customize and enhance a BIRT report by writing event handler scripts in either Java or JavaScript. Part 3 includes the following chapters:

- *Chapter 7. Using Scripting in a Report Design.* This chapter introduces the writing of a BIRT event handler script in either Java or JavaScript, including the advantages and disadvantages of using one language over the other. BIRT event handlers are associated with data sets, data sources, and report items. BIRT fires specific events at specific times in the

processing of a report. This chapter identifies the events that BIRT fires and describes the event firing sequence.

- *Chapter 8. Using JavaScript to Write an Event Handler.* This chapter discusses the coding environment and coding considerations for writing a BIRT event handler in JavaScript. This chapter describes several BIRT JavaScript objects that a developer can use to get and set properties that affect the final report. The BIRT JavaScript coding environment offers a pop-up list of properties and functions available in an event handler. A JavaScript event handler can also use Java classes. This chapter includes a tutorial that describes the process of creating a JavaScript event handler.
- *Chapter 9. Using Java to Write an Event Handler.* This chapter discusses how to write a BIRT event handler in Java. BIRT provides Java adapter classes that assist the developer in the creation of Java event handlers. The report developer uses the property editor of the BIRT Report Designer to associate a Java event handler class with the appropriate report element. This chapter contains a tutorial that steps through the Java event handler development and deployment process. This chapter also describes the event handler methods and their parameters.
- *Chapter 10. Working with Chart Event Handlers.* This chapter describes the BIRT event handler model for the Chart Engine. The model is similar to the model for standard BIRT report elements and supports both the Java and JavaScript environments. This chapter provides details on both environments. The Chart Engine also supports this event model when used outside of BIRT.
- *Chapter 11. Using Scripting to Access Data.* This chapter describes how to access a data source using JavaScript code. A data source that you access using JavaScript is called a scripted data source. With a scripted data source, you can access objects other than an SQL, XML, or text file data source. A scripted data source can be an EJB, an XML stream, a Hibernate object, or any other Java object that retrieves data.
- *Chapter 12. Debugging Event Handlers.* This chapter describes how to use the BIRT report debugger to debug BIRT reports and event handlers written in JavaScript and Java. The BIRT report debugger enables users to run reports step by step, set breakpoints, inspect instance properties, and watch and evaluate variable values in certain contexts. The debugger supports advanced report developers in spotting the errors in report designs quickly, and makes it easy to understand and identify exceptions during run time. The chapter provides a tutorial with step-by-step instructions about debugging Java and JavaScript report event handlers.

Part 4, Integrating BIRT into Applications

Part 4 describes the public APIs that are available to Java developers, except the extension APIs. Part 4 includes the following chapters:

- *Chapter 13. Understanding the BIRT APIs.* This chapter introduces BIRT's public API, which are the classes and interfaces in three package hierarchies:
  - The report engine API, in the org.eclipse.birt.report.engine.api hierarchy, supports developers of custom report generators.
  - The design engine API, in the org.eclipse.birt.report.engine.api hierarchy, supports the development of custom report designs.
  - The chart engine API, in the org.eclipse.birt.chart hierarchy, is used to develop a custom chart generator.
- *Chapter 14. Programming Using the BIRT Reporting APIs.* This chapter describes the fundamental requirements of a reporting application and lists the BIRT API classes and interfaces that are used to create a reporting application. This chapter describes the tasks that are required of a reporting application and provides an overview of how to build a reporting application. The org.eclipse.birt.report.engine.api package supports the process of generating a report from a report design. The org.eclipse.birt.report.model.api package supports creating new report designs and modifying existing report designs.
- *Chapter 15. Programming Using the BIRT Charting API.* This chapter describes the requirements of a charting application, either in a stand-alone environment or as part of a reporting application. The org.eclipse.birt.chart hierarchy of packages provides the charting functionality in BIRT. By describing the fundamental tasks required of charting applications, this chapter introduces the API classes and interfaces that are used to create a chart. This chapter also describes the chart programming examples in the chart examples plug-in.

Part 5, Working with the Extension Framework

Part 5 shows Java programmers how to add new functionality to the BIRT framework. By building on the Eclipse platform, BIRT provides an extension mechanism that is familiar to developers of Eclipse plug-ins. This part also provides information about how to build the BIRT project for developers who need access to the complete BIRT open source code base. Part 5 includes the following chapters:

- *Chapter 16. Building the BIRT Project.* This chapter explains how to download BIRT 2.6 source code and build the BIRT project for development. This chapter describes how to configure an Eclipse workspace, download BIRT source code, and build the BIRT report and web viewers.
- *Chapter 17. Extending BIRT.* This chapter provides an overview of the BIRT extension framework and describes how to use the Eclipse Plug-in Development Environment (PDE) and the BIRT extension points to create, build, and deploy a BIRT extension.

- *Chapter 18. Developing a Report Item Extension.* This chapter describes how to develop a report item extension. The rotated text extension example is a plug-in that renders the text of a report item as an image. The extension rotates the image in the report design to display the text at a specified angle. This chapter describes how to build the rotated text report item plug-in and add the report item to the BIRT Report Designer using the defined extension points.
- *Chapter 19. Developing a Report Rendering Extension.* This chapter describes how to develop a report rendering extension using the Eclipse PDE with sample CSV and XML report rendering extensions as the examples. The chapter describes how to extend the emitter interfaces using the defined extension points to build and deploy a customized report rendering plug-in that runs in the BIRT Report Engine environment.
- *Chapter 20. Developing an ODA Extension.* This chapter describes how to develop several types of DTP ODA extensions. The CSV ODA driver example is a plug-in that reads data from a CSV file. The Hibernate ODA driver example uses Hibernate Query Language (HQL) to provide a SQL-transparent extension that makes the ODA extension portable to all relational databases. This chapter shows how to develop an ODA extension to the BIRT Report Designer 2.6.0 user interface that allows a report designer to select an extended ODA driver.
- *Chapter 21. Developing a Data Extraction Extension.* This chapter describes how to develop a report data extraction extension using the Eclipse PDE. The data extraction example exports report data to XML format. The Eclipse PDE provides the basis for the extension.
- *Chapter 22. Developing a Fragment.* This chapter describes how to build a fragment. The BIRT Report Engine environment supports plug-in fragments. A plug-in fragment is a separately loaded package that adds functionality to an existing plug-in, such as a specific language feature in a National Language Support (NLS) localization application. The example in this chapter creates a Java resource bundle that adds translations to the messages defined in the messages.properties files for the org.eclipse.birt.report.viewer plug-in.
- *Chapter 23. Developing a Charting Extension.* This chapter discusses the process of adding a new chart type to the BIRT chart engine. BIRT provides a radar chart type with the chart engine as an example of how to build new chart types. This chapter describes how to create this radar chart example.

The *Glossary* contains a glossary of terms that are useful to understanding all parts of the book.

Typographical conventions

Table P-1 describes the typographical conventions that are used in this book.

Table P-1 Typographical conventions

Item	Convention	Example
Code examples	Monospace font	`StringName = "M. Barajas";`
File names	Initial capital letter, except where file names are case-sensitive	SimpleReport.rptdesign
Key combination	A + sign between keys means to press both keys at the same time	Ctrl+Shift
Menu items	Capitalized, no bold	File
Submenu items	Separated from the main menu item with a small arrow	File➛New
User input	Monospace font	`2008`
User input in Java code	Monospace font italics	`chkjava.exe` *`cab_name`*`.cab`

Syntax conventions

Table 3-1 describes the symbols that are used to present syntax.

Table 3-1 Syntax conventions

Symbol	Description	Example
[ ]	Optional item	int count [= <value>];
	Array subscript	matrix[ ]
< >	Argument that you must supply	<expression to format>
	Delimiter in XML	<xsd:sequence>
{ }	Groups two or more mutually exclusive options or arguments when used with a pipe	{TEXT_ALIGN_LEFT | TEXT_ALIGN_RIGHT}
	Defines array contents	{0, 1, 2, 3}

(continues)

Table 3-1 Syntax conventions (continued)

Symbol	Description	Example

{ }	Delimiter of code block	if (itemHandle == null)
		{
		  // create a new handle
		}

|	Separates mutually exclusive options or arguments in a group	[public | protected | private] <data type> <variable name>;
	Java bitwise OR operator	int newflags = flags |4

Acknowledgments

John Arthorne and Chris Laffra observed, "It takes a village to write a book on Eclipse." In the case of the BIRT books, it continues to take a virtual village in four countries to create these two books. Our contributors, reviewers, Addison-Wesley editorial, marketing, and production staff, printers, and proofreaders are collaborating by every electronic means currently available to produce the major revisions to these two books. In addition, we want to acknowledge the worldwide community of over one million Java programmers who have completed over ten million downloads of the multiple versions of the software. Their enthusiastic reception to the software creates an opportunity for us to write about it.

We want to thank Greg Doench, our acquisitions editor, who asked us to write a book about BIRT and has been supportive and enthusiastic about our success. Of course, we want to acknowledge the staff at Addison-Wesley who worked on the first and second editions and this third revision. In particular, we would like to acknowledge John Fuller, Michelle Housley, Anne Jones, Mary Kate Murray, Julie Nahil, Stephane Nakib, Elizabeth Ryan, Sandra Schroeder, Beth Wickenhiser, and Lara Wysong. We also want to thank Mike Milinkovich at the Eclipse Foundation and Mark Coggins at Actuate Corporation for providing the forewords for the books.

We particularly want to acknowledge the many, many managers, designers, and programmers too numerous to name who have worked diligently to produce, milestone by milestone, the significant upgrades to BIRT, giving us a reason for these two books. You know who you are and know how much we value your efforts. The following engineers have been of particular assistance to the authors: Linda Chan, Xiaoying Gu, Wenbin He, Petter Ivmark, Rima Kanguri, Nina Li, Wenfeng Li, Yu Li, Jianqiang Luo, Kai Shen, Aniruddha Shevade, Pierre Tessier, Yulin Wang, Mingxia Wu, Gary Xue, Jun Zhai, and Lin Zhu. We want to recognize the important contribution of David Michonneau in the area of charting. We wish to acknowledge Zhiqiang Qian who created a data bound report item extension used to develop the report item extension chapter. Yasuo Doshiro worked closely with the authors on "Developing a Fragment," which provides suggestions for the practical application of BIRT technology to the challenges of translation and localization. Doshiro manages BIRT translation and develops and executes test plans. In addition, we want to acknowledge the support

and significant contribution that was provided by Paul Rogers. Dan Melcher's and Daniel O'Connell's insights into the techniques for building reusable components that can be applied to building internationalized reports. Working examples are to be found at

`http://reusablereporting.blogspot.com/`

Creating this book would not have been possible without the constant support of the members of the Developer Communications team at Actuate Corporation. Many of them and their families sacrificed long personal hours to take on additional tasks so that members of the team of authors could create this material. In particular, we wish to express our appreciation to Terry Ryan for his work on the glossary that accompanies each of the books. In addition, Mary Adler, Minali Balaram, Frances Buran, Bruce Gardner, Shawn Giese, Mike Hovermale, Melia Kenny, James Monaghan, Lois Olson, James Turner, Jeff Wachhorst, C. J. Walter-Hague, and Forest White all contributed to the success of the books.

Actuate's active student intern program under the Executive Sponsorship of Dan Gaudreau, Chief Financial Officer, made it possible for Maziar Jamalian and Rodd Naderzad to support the project in Developer Communications while actively engaged in pursuing undergraduate and graduate degrees in accounting, business, and information science at several different universities.

Part

I

Installing and Deploying BIRT

This page intentionally left blank

Chapter

1

Introducing BIRT Report Designers

There are two designer applications that you can use to create BIRT reports:

- BIRT Report Designer
  A tool that a report developer uses to build a BIRT report design and preview a report. BIRT Report Designer is a set of Eclipse plug-ins that includes BIRT Report Engine, BIRT Chart Engine, and BIRT Demo Database. This tool supports Java and JavaScript customization. BIRT Report Designer requires multiple Eclipse platform components and a Java Development Kit (JDK).
- BIRT RCP Report Designer
  A simplified tool that a novice report developer uses to build a BIRT report design and preview a report. BIRT RCP (Rich Client Platform) Report Designer includes BIRT Report Engine, BIRT Chart Engine, and BIRT Demo Database without the additional overhead of the full Eclipse platform. This tool supports JavaScript customization, but does not support Java customization or debugging.

Understanding BIRT components

BIRT Report Designer 2.6 consists of the following components:

- Eclipse Software Development Kit (SDK) 3.6
  The SDK is a framework that supports the development of plug-ins and extensions to the Eclipse platform. The SDK includes the core platform, the Java Development Tools (JDT), and the Plug-in Developer Environment (PDE).

- Data Tools Platform (DTP) 1.8.0
  The DTP is a set of development tools used to develop plug-ins that access data sources and retrieve data.
- Eclipse Modeling Framework (EMF) 2.6.0
  The EMF supports the development of BIRT charts. The EMF includes the Service Data Objects (SDO), which is a graph-structured data object that supports applying changes to a graph back to the data source.
- Graphical Editing Framework (GEF) 3.6.0
  The GEF is an Eclipse plug-in that the BIRT Report Designer user interface requires. This framework provides a rich, consistent, graphical editing environment for an application running on the Eclipse Platform.
- Eclipse Web Tools Platform (WTP) 3.2
  The WTP is a set of Eclipse plug-ins that support deploying the BIRT report viewer to an application server. The package includes source and graphical editors, tools, wizards, and APIs that support deploying, running, and testing.

Understanding Eclipse BIRT packages

Eclipse BIRT provides the following packages. These packages do not include the required Java 1.5 JDK.

- Report Designer Full Eclipse Install (All-in-One)
  Contains BIRT and the Eclipse Integrated Development Environment (IDE). This all-in-one installation is the easiest way to install BIRT.
- Report Designer
  Contains only BIRT for installing in an existing Eclipse Integrated Development Environment (IDE).
- RCP Report Designer
  Contains a simplified version of BIRT without the Eclipse IDE.
- Software Development Kit (SDK)
  Contains the source code for the BIRT plug-ins, documents, and examples.
- Report Engine
  Contains the run-time version of BIRT for installing in a J2EE application server.
- Chart Engine
  Contains the stand-alone library that supports embedding a chart in a Java application.
- BIRT Web Tools Integration

Contains the plug-ins required to use the BIRT Web Project Wizard in a Web Tools Project, including the source code.

- BIRT Source Code

  Contains the BIRT source code for a specific build. All source code is in a plug-in format ready to import into a workspace to build BIRT. These plug-ins are the required libraries for a standard BIRT installation. Additional libraries may be necessary. For example, this package does not include the Data Tools Platform (DTP) source code.

- BIRT Samples

  Contains sample reports and charts, plus application examples that use the Chart, Report Engine, and Design Engine APIs.

- BIRT Demo Database

  Contains the package for defining and loading the demonstration database into Apache Derby and MySQL, including SQL and data files. The demonstration database package is a convenient way to install the Classic Models database schema and data in the Apache Derby and MySQL systems. The package does not include any BIRT software. The Report Designer and the RCP Report Designer packages include the demonstration database for Apache Derby.

  The demonstration database supports the following Apache and MySQL versions:

  - Apache Derby version 5.1 or higher
  - MySQL Connector/J version 3.1 or MySQL client version 4.x

About types of BIRT builds

The Eclipse BIRT download site makes several types of builds available for BIRT. The following list describes these builds:

- Release build

  A production build that passes the complete test suite for all components and features. Use the release build to develop applications.

- Milestone build

  A development build that provides access to newly completed features. The build is stable, but it is not production quality. Use this type of build to preview new features and develop future reporting applications that depend on those features.

- Stable build

  A development build that is stable, but passes a reduced test suite. New features are in an intermediate stage of development. Use a stable build to preview new features.

- Nightly build

  The Eclipse BIRT development team builds BIRT every night. As BIRT is an open-source project, these builds are available to anyone. These builds are unlikely to be useful to a report developer.

  If a certain feature that you require does not work in a nightly build, you can provide feedback to the development team by filing a bug report. Later, you can download a new build to confirm that the fix solves the problem that you reported.

Chapter

2

Installing a BIRT Report Designer

Installing BIRT Report Designer adds a report design perspective to the Eclipse Integrated Development Environment (IDE). To install BIRT Report Designer, download an archive file from the Eclipse web site and extract it in your existing Eclipse environment. BIRT Report Designer is available for various Linux and Microsoft Windows platforms. The following sections describe how to install BIRT Release 2.6.

Installing BIRT Report Designer Full Eclipse Install

If you are new to Eclipse and BIRT, download and install BIRT Report Designer Full Eclipse Install (All-in-One) package to start developing and designing BIRT reports immediately. This package includes the Eclipse Integrated Development Environment (IDE), BIRT Report Designer, and all other required Eclipse components. You must also download and install Java JDK 1.5.

Complete the following procedure to download this installation package on a Windows or Linux system.

How to install BIRT Report Designer All-in-One

1. Using your browser, navigate to the main BIRT web page at:

   `http://www.eclipse.org/birt/phoenix`

2. From BIRT Project, choose Download BIRT 2.6.
3. From BIRT Report Downloads, choose All-in-One.

4 On BIRT Report Downloads, select the Download Link that meets your requirements, for example, Windows 64-bit.

   Eclipse downloads - mirror selection appears. This page shows all the sites that provide this download file.

5 Choose the download site that is closest to your location.

   The BIRT Report Designer all-in-one archive file downloads to your system.

6 Extract the archive file to a hard drive location that you specify.

   The extraction creates a directory named eclipse at the location that you specify.

To test the BIRT Report Designer installation, start Eclipse, then start BIRT Report Designer as described in the following procedure. BIRT Report Designer is a perspective within Eclipse.

How to test the BIRT Report Designer installation

1 Start Eclipse.

2 Close the welcome window. In the Eclipse Window menu, choose Open Perspective➛Report Design. If Report Design does not appear in the Open Perspective window, choose Other. A list of perspectives appears. Choose Report Design.

   Eclipse displays the BIRT Report Designer perspective.

Installing BIRT RCP Report Designer

BIRT RCP Report Designer is a stand-alone report design application that enables report developers to produce reports in both web and PDF formats. This application uses the Eclipse Rich Client Platform (RCP) to provide a report design environment that is less complex than the full Eclipse platform. If you need the project-based environment that the full Eclipse platform provides, return to the section on installing BIRT Report Designer. BIRT RCP Report Designer runs on Windows only.

To install BIRT RCP Report Designer, download and extract an archive file. The following examples use Release 2.6.

Complete the following procedure to download and install BIRT RCP Report Designer on a Windows system.

How to install BIRT RCP Report Designer

1 Using your browser, navigate to the main BIRT web page at:

   `http://www.eclipse.org/birt/phoenix`

2 From BIRT Home, choose Download 2.6.

3 From BIRT Report Downloads, choose RCP Designer.

   Eclipse downloads - mirror selection appears. This page shows all the sites that provide this download file.

4 Choose the download site that is closest to your location.

   The BIRT RCP Report Designer archive downloads to your system.

5 Extract the archive file to a hard drive location that you specify.

   The extraction creates a directory named birt-rcp-report-designer-2_6_0 at the location that you specify.

To test the installation, start BIRT RCP Report Designer as described in the following procedure.

How to test the BIRT RCP Report Designer installation

1 Navigate to the birt-rcp-report-designer-2_6_0 directory.

2 To run BIRT RCP Report Designer, double-click BIRT.exe. BIRT RCP Report Designer appears.

Troubleshooting installation problems

Installing a BIRT report designer is a straightforward task. If you extract the archive file to the appropriate location and the required supporting files are also available in the expected location, your BIRT report designer will work. One of the first steps in troubleshooting an installation problem is confirming that all files are in the correct location.

Verify that the /eclipse/plugins directory contains JAR files whose names begin with org.eclipse.birt, org.eclipse.emf, and org.eclipse.gef. The following sections describe troubleshooting steps that resolve two common installation errors.

Avoiding cache conflicts after you install a BIRT report designer

Eclipse caches information about plug-ins for faster start-up. After you install or upgrade BIRT Report Designer or BIRT RCP Report Designer, using a cached copy of certain pages can lead to errors or missing functionality. The symptoms of this problem include the following conditions:

- The Report Design perspective does not appear in Eclipse.
- You receive a message that an error occurred when you open a report or use the Report Design perspective.
- JDBC drivers that you installed do not appear in the driver manager.

The solution is to remove the cached information. The recommended practice is to start either Eclipse or BIRT RCP Report Designer from the command line with the -clean option.

To start Eclipse, use the following command:

```
eclipse.exe -clean
```

To start BIRT RCP Report Designer, use the following command:

```
BIRT.exe -clean
```

Specifying a Java Virtual Machine when starting BIRT report designer

You can specify which Java Virtual Machine (JVM) to use when you start a BIRT report designer. This specification is important, particularly for users on Linux, when path and permission problems prevent the report designer from locating an appropriate JVM to use. A quick way to overcome such problems is by specifying explicitly which JVM to use when you start the BIRT report designer.

On Windows and Linux systems, you can either start a BIRT report designer from the command line or create a command file or shell script that calls the appropriate executable file with the JVM path. The example in this section uses BIRT Report Designer on a Windows system.

How to specify which JVM to use when you start a BIRT report designer

On the command line, type a command similar to:

```
eclipse.exe -vm $JAVA_HOME/jdk1.5/bin/java.exe
```

Installing a language pack

All BIRT user interface components and messages are internationalized through the use of properties files. BIRT uses English as the default language, but supports other languages by installing a language pack that contains the required properties files. BIRT 2.6 provides one language pack, NLpack1, which supports the following languages:

- French
- German
- Spanish
- Japanese
- Korean
- Simplified Chinese

The following instructions explain how to download and install the language pack for BIRT 2.6 on Windows.

How to download and install a language pack

To download and install a language pack, perform the following steps:

1. Using your browser, navigate to the BIRT language pack web page at:

   `http://www.eclipse.org/babel/downloads.php`

2. From Babel Language Packs for Galileo, download the language pack for the product that you need.
3. Extract the language pack archive file into the directory above the Eclipse directory.

   For example, if C:/eclipse is your Eclipse directory, extract the language pack into C:/.

4. Start Eclipse and choose Window➛Preferences➛Report Design➛Preview.
5. Select the language of choice from the drop-down list in Choose your locale.
6. Restart Eclipse.

If Windows is not running under the locale you need for BIRT, start Eclipse using the -nl <locale> command line option, where <locale> is a standard Java locale code, such as es_ES for Spanish as spoken in Spain. A list of locale codes is available at the following URL:

```
http://www.oracle.com/technetwork/java/javase/locales-
   137662.html
```

Eclipse remembers the locale you specify on the command line. On subsequent launches of Eclipse, the locale is set to the most recent locale setting. To revert to a previous locale, launch Eclipse using the -nl command line option for the locale to which you want to revert.

Updating a BIRT Report Designer installation

Because BIRT Report Designer is a Java-based application, updating an installation typically requires replacing the relevant files. Eclipse supports the update process for BIRT Report Designer by providing the Update Manager. BIRT RCP Report Designer is a stand-alone product, so you must replace the existing version with a newer version.

This section describes the steps required to update the following BIRT packages:

- Report Designer
- RCP Report Designer

You can use the Eclipse Update Manager to find and install newer major releases of BIRT Report Designer.

How to update a BIRT Report Designer installation using the Update Manager

1. In Eclipse, choose Help➛Check for Updates.
2. In Available Updates, choose Select All then choose Next.
3. In Update Details, choose Next.
4. In Review Licenses, accept the license agreement terms and choose Finish.
5. When the update completes, restart your computer.

How to update BIRT Report Designer manually

1. Back up the workspace directory if it is in the eclipse directory structure.
2. To remove the BIRT files, use one of the following techniques:
   - To prepare for a new all-in-one installation, remove the entire eclipse directory.
   - To prepare for only a BIRT Report Designer installation, remove only the BIRT components.
     1. Navigate to the eclipse\features directory.
     2. Delete all JAR files and subdirectories with birt in their names.
     3. Navigate to the eclipse\plugins directory.
     4. Delete all JAR files and subdirectories with birt in their names.
3. Download and install BIRT Report Designer as described earlier in this book.
4. Restore the workspace directory, if necessary.
5. Restart BIRT Report Designer with the -clean option:

   ```
   eclipse.exe -clean
   ```

Updating BIRT RCP Report Designer installation

Unlike BIRT Report Designer, BIRT RCP Report Designer is a stand-alone application. To update this application, you delete the entire application and reinstall a newer version. If you created your report designs and resources in the birt-rcp-report-designer-<version> directory structure, you must back up your workspace directory and any resources that you want to keep before you delete BIRT RCP Report Designer. After you install a newer version of the application, you can copy your files back to the application directory structure.

As a best practice, do not keep your workspace in the birt-rcp-report-designer-<version> directory structure. Keeping your workspace in a different location enables you to update your installation more easily in the future.

How to update BIRT RCP Report Designer

1. Back up the workspace directory and any other directories that contain report designs, libraries, and other resources, if they are in the birt-rcp-report-designer-<version> directory structure.
2. Delete the birt-rcp-report-designer-<version> directory.
3. Download and install BIRT RCP Report Designer as described earlier in this book.
4. Restore the directories that you backed up in step 1, if necessary.
5. Restart BIRT RCP Report Designer with the -clean option:

   ```
   BIRT.exe -clean
   ```

This page intentionally left blank

Chapter

3

Installing Other BIRT Packages

Beyond the BIRT Report Designer packages, BIRT provides a number of other separate packages as downloadable archive files on the Eclipse web site. Some of these packages are stand-alone modules, others require an existing Eclipse or BIRT environment, and still others provide additional functionality to report developers and application developers. This chapter describes the steps required to install the BIRT packages shown in the following list:

- Chart Engine
- Data Tools Platform (DTP) Integration
- Demo Database
- Report Engine
- Samples
- Source Code
- Web Tools Integration

Installing Chart Engine

Chart Engine supports adding charting capabilities to a Java application. An application can use Chart Engine without using the BIRT reporting functionality or Report Engine. Chart Engine integrates into an existing Eclipse platform on Microsoft Windows, UNIX, or Linux. You can also install Chart Engine on an existing J2EE application server. To use Chart Engine, you use its public API, org.eclipse.birt.chart.

Both BIRT Report Designer and BIRT RCP Report Designer include all the components of Chart Engine. If you are using a BIRT report designer, you do not need to install BIRT Chart Engine separately.

How to install BIRT Chart Engine

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
   For more information about how to navigate to the BIRT web site and BIRT Downloads for build 2.6.0, see Chapter 2, "Installing a BIRT Report Designer."
2. In the Chart Engine section, choose the Chart Engine archive file:
   `birt-charts-2_6_0.zip`
3. In File Downloads, choose Open.
4. Extract the archive file to a location of your choice.
5. Start Eclipse from the command line with the -clean option to remove cached information.

The archive extraction process creates the following subdirectories in the extraction directory:

- ChartRuntime
  This directory contains the plug-ins and libraries that an Eclipse platform requires to run, render, and edit charts.
- ChartSDK
  This directory contains the plug-ins and libraries from the ChartRuntime directory plus the SDK that you need to create your own charting applications. It also includes examples, source code, and a Web Tools Platform (WTP) extension to support charts in web applications.
- DeploymentRuntime
  This directory contains the libraries that you need to run your charting application in a non-Eclipse environment such as on an application server.

The Chart Engine download file also includes extensive Frequently Asked Questions (FAQ) and examples illustrating how to use Chart Engine. After extracting the archive, you can find the FAQ at the following location:

```
<CHART_ENGINE>/DeploymentRuntime/ChartEngine/docs/Charts_FAQ.doc
```

The examples are in a JAR file located at:

```
<CHART_ENGINE>/ChartSDK/eclipse/plugins
   /org.eclipse.birt.chart.examples_<version>.jar
```

Installing BIRT Data Tools Platform Integration

This package includes the minimal set of Data Tools Platform (DTP) plug-ins that BIRT Report Designer requires. If you install the BIRT Report Designer package in an existing Eclipse installation, you can install this BIRT DTP Integration package instead of the full DTP platform.

How to install BIRT DTP Integration

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the BIRT DTP Integration section, choose the BIRT DTP Integration archive file:

   `birt-dtp-integration-2_6_0.zip`
3. On Eclipse downloads, choose your closest download site.
4. Extract the archive file to the directory that contains your Eclipse directory.

   Extracting creates the DTP features and plug-ins in the eclipse\features and eclipse\plugins directories.
5. Start Eclipse from the command line with the -clean option.

To test the BIRT DTP Integration package, open the Report Design perspective in Eclipse, as described in the following procedure.

How to test the BIRT DTP Integration installation

1. Start Eclipse.
2. From the Eclipse main menu, choose Open Perspective➛Report Design. If Report Design does not appear in the Open Perspective window, choose Other. From the list of perspectives, choose Report Design.

   Eclipse displays the BIRT Report Designer perspective.

Installing BIRT Demo Database

The BIRT Demo Database package provides the Classic Models database that this book uses for example procedures. The database is provided in the following formats:

- Apache Derby
- MySQL

BIRT Report Designer and BIRT RCP Report Designer include this database in Apache Derby format, as the Classic Models Inc. sample database data

source. Install BIRT Demo Database if you want to use the native drivers to access this data source.

How to install BIRT Demo Database

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the Demo Database section, choose the Demo Database archive file:

   `birt-database-2_6_0.zip`
3. In File Download, choose Open.
4. Extract the archive file to a location of your choice.

   Extracting creates a directory, ClassicModels, that contains the BIRT Demo Database in Apache Derby and MySQL formats.

To test the BIRT Demo Database, first connect to the database with the native database client tool or a Java application.

How to access BIRT Demo Database using a database client tool

Perform one of the following sets of tasks, based on your preferred database:

- Apache Derby database

  Connect to the database in the derby subdirectory of ClassicModels.
- MySQL
  1. Navigate to the mysql subdirectory of ClassicModels.
  2. Create a database to use or edit create_classicmodels.sql to uncomment the lines that create and select the classicmodels database.
  3. Use the mysql command line interface to run create_classicmodels.sql.
  4. Review load_classicmodels.sql to determine if you can use the script on your platform without editing. Use the mysql command line interface to run load_classicmodels.sql.

Next, connect to the database from BIRT Report Designer or BIRT RCP Report Designer.

How to access BIRT Demo Database from a BIRT report designer

Connect to the database using BIRT Report Designer or BIRT RCP Report Designer.

1. To access the Classic Models database in Apache Derby or MySQL format, first add the driver JAR files to a BIRT report designer installation.
2. In any report design, create a data source on the database. In the same report design, create a data set on the data source.

Installing Report Engine

Report Engine supports adding reporting capabilities to a Java application. BIRT Report Engine integrates into an existing Eclipse platform on Microsoft Windows, UNIX, or Linux. You can also install report engine components on an existing J2EE application server. To support quick deployment of reporting functionality to an application server, Report Engine includes a web archive (.war) file.

How to install BIRT Report Engine

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the Report Engine section, choose the Report Engine archive file:

   `birt-runtime-2_6_0.zip`
3. In File Download, choose Open.
4. Extract the archive file to a suitable directory.
5. Create a system variable, BIRT_HOME.

   Set the value of BIRT_HOME to the BIRT Report Engine installation directory. For example, if you extracted the BIRT Report Engine to C:\, the value of BIRT_HOME is:

   `C:\birt-runtime-2_6_0`

To test the installation, run the Report Engine report generation command line example. This example uses a batch (.bat) file on a Windows system and a shell script (.sh) file on a UNIX or Linux system. This file takes the parameters shown in Table 3-1.

Table 3-1 Parameters for the genReport script

Parameter	Valid for mode	Values
Execution mode -m		Valid values are run, render, and runrender. The default is runrender.
Target encoding -e	render, runrender	A valid encoding. The default is utf-8.
Output format -f	render, runrender	Valid values are HTML and PDF. The default value is HTML.
Report parameters file -F	run, runrender	Path to the parameter file. This file contains lines with the format: <parameter name>=<value>

(continues)

Table 3-1 Parameters for the genReport script (continued)

Parameter	Valid for mode	Values
Locale -l locale	run, runrender	A valid locale string. The default locale is en.
Output file name -o	render, runrender	The full path of the output file. The default value is the name of the report design with an extension based on the output format, .html for an HTML file and .pdf for a PDF file.
Report parameter -p "parameter name=value"	run, runrender	If you provide parameter values with the -p parameter, these values override the values in the report parameters file specified by -F.
HTML format -t	run, runrender	Valid values are HTML and ReportletNoCSS. HTML is the default. This format wraps the HTML output in an <HTML> tag. ReportletNoCSS does not wrap the HTML output in an <HTML> tag.
Report design file	All modes	The full path of the report design file. This parameter must be the last parameter on the command line.

How to test the BIRT Report Engine installation

1. From the command line, navigate to the directory where you installed BIRT Report Engine.
2. Navigate to the ReportEngine subdirectory.
3. To run the genReport script, run the appropriate file for your operating system:
   - On a Windows platform, run genReport.bat.
   - On a UNIX or Linux platform, run genReport.sh.

   Enclose the value for a command line parameter in quotes. For example, the following Windows platform command uses the value, Hello, for the parameter, sample, to generate an HTML file from the report design, test.rptdesign:

   ```
   genReport -p "sample=Hello"
      "C:\birt-runtime-2_6_0\WebViewerExample\test.rptdesign"
   ```

   genReport generates the required output file.
4. Open the output file. In this example, the file is C:\birt-runtime-2_6_0\WebViewerExample\test.html.

For more information about setting up the BIRT Report Engine, see Chapter 4, “Deploying a BIRT Report to an Application Server.”

Installing BIRT Samples

BIRT Samples provides examples of a BIRT report item extension and of charting applications. The report item extension integrates into BIRT Report Designer and BIRT Report Engine.

How to install BIRT Samples

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the Samples section, choose the Samples archive file:

   `birt-samples-plugins-2_6_0.zip`
3. In File Download, choose Open.
4. Extract the archive file to the directory that contains your Eclipse directory.

Installing BIRT Source Code

This package includes the source code for all BIRT plug-ins. You can examine this code to see how BIRT generates reports from designs. You can also import this source code into a workspace to build a custom BIRT installation.

How to install BIRT Source Code

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the BIRT Source Code section, choose the BIRT Source Code archive file:

   `birt-source-2_6_0.zip`
3. In File Download, choose Open.
4. Extract the archive file to a new workspace directory.

   Extracting creates the build files and BIRT features and plugins directories in that workspace directory.

To test the BIRT Source Code package, import the source code projects into your workspace.

How to test the BIRT Source Code installation

1. Start Eclipse.
2. Set the Java preferences for BIRT.

1. From the Eclipse window menu, choose Window➛Preferences.
2. Expand Java, select Compiler. Make the following selections:
   - Set Compiler Compliance Level to 1.6.
   - Deselect Use default compliance settings.
   - Set Generated .class files compatibility to 1.6.
   - Set Source compatibility to 1.6.
3. Choose OK.

3. From the Eclipse window menu, choose File➛Import.
4. In Import—Select, expand General and select Existing Projects into Workspace. Choose Next.
5. In Import—Import Projects, select Select root directory, then type or browse to your workspace directory.

   The BIRT features and plug-ins appear in Projects.
6. Choose Finish.

   Eclipse builds the BIRT projects.

If the projects do not build correctly, check that you installed the prerequisites for BIRT Report Designer, as described in Chapter 1, "Introducing BIRT Report Designers." If you have not installed the BIRT Report Designer Full Eclipse Install, download this package and extract any JAR files that the build requires. Add any libraries that Eclipse does not find to the build paths of specific projects to resolve other build errors.

Installing BIRT Web Tools Integration

This package includes the minimal set of BIRT plug-ins that the Eclipse Web Tools Platform (WTP) requires to build a BIRT web project using the BIRT Web Project Wizard. This package also includes the source code for these plug-ins.

How to install BIRT Web Tools Integration

On the BIRT web site, perform the following operations:

1. Navigate to BIRT Downloads for build 2.6.0.
2. In the BIRT Web Tools Integration section, choose the BIRT Web Tools Integration archive file:

   `birt-wtp-integration-sdk-2_6_0.zip`
3. In File Download, choose Open.
4. Extract the archive file to the directory that contains your Eclipse directory.

Extracting creates the BIRT features and plug-ins in the eclipse\features and eclipse\plugins directories.

To test the BIRT Web Tools Integration package, create a BIRT web project in Eclipse.

How to test the BIRT Web Tools Integration installation

1. Start Eclipse.
2. From the Eclipse window menu, choose File➛New➛Project.
3. In New Project—Select a wizard, expand Web, select Dynamic Web Project. Choose Next.
4. In New Project—Dynamic Web Project, make the choices that you need for your BIRT web project, then choose Finish.

   If you do not have the Java EE perspective open, Eclipse displays the following message:

   > This kind of project is associated with the Java EE perspective. Do you want to open this perspective now?

   Choose Yes.

This page intentionally left blank

Chapter

4

Deploying a BIRT Report to an Application Server

One way to view a BIRT report on the web is to deploy the BIRT report viewer to an application server, such as Apache Tomcat, IBM WebSphere, JBoss, or BEA WebLogic. The BIRT Report Engine includes the BIRT report viewer as a web archive (.war) file and as a set of files and folders. Deploying the BIRT report viewer requires copying files from the BIRT Report Engine, which you must install separately from the BIRT Report Designer. This chapter provides information about deploying the BIRT report viewer using the WAR file or the set of files and folders.

About application servers

The instructions in this chapter specifically address deploying a BIRT report to Apache Tomcat version 6.0. Although the information in this chapter is specific to this version of Tomcat, a BIRT report can also be deployed to other versions of Tomcat and to other application servers.

About deploying to Tomcat

There are only minor differences between the requirements for deploying to Tomcat version 6.0 and deploying to other versions of Apache Tomcat. Apache Tomcat 6.0 runs Java 5 by default, which is also one of the recommended versions to use for BIRT 2.6. If you use an earlier version of Java, you need to install a compatibility package and configure Apache Tomcat to use the Java 1.4 run-time environment. For information about configuring Apache Tomcat to use Java 1.4 run-time, see the Apache Tomcat help pages.

About deploying to other application servers

Most application servers require a WAR file that contains everything that the application requires, including a web.xml file describing the application and various deployment preferences. The BIRT Report Engine includes a WAR file appropriate to Tomcat. Typically, the WAR file does not require modification. In some cases, developers who have experience with other application servers can modify the web.xml file to reflect the requirements of their environments. The section on mapping the report viewer folders, later in this chapter, discusses setting the web.xml parameters.

Deployment to JBoss may require copying axis.jar and axis-ant.jar from WEB-INF/lib to the following directory:

```
jboss/server/default/lib
```

This step is not necessary for all versions of JBoss, but if there are difficulties with a JBoss deployment, copying these files can resolve the problem.

Placing the BIRT report viewer on an application server

You must place the BIRT report viewer in a location where Apache Tomcat can access it. Typically, this location is the $TOMCAT_INSTALL/webapps directory. On restarting Apache Tomcat, the application server automatically recognizes and starts the BIRT report viewer application if the BIRT report viewer is in this folder.

Installing the BIRT report viewer as a web application

The BIRT report viewer files provide core functionality to run, render, and view BIRT reports. To use additional JDBC drivers that are not part of the standard BIRT packages, you must install these drivers as well as the BIRT report viewer itself. If you install the BIRT report viewer as a WAR file, you must include the JDBC drivers in the WAR file.

The following instructions assume that you have installed the BIRT Report Engine from the BIRT web site, that your web application directory is $TOMCAT_INSTALL/webapps, and that your BIRT run-time installation directory is $BIRT_RUNTIME.

How to install the BIRT report viewer from the BIRT Report Engine WAR file

The steps to install the BIRT report viewer from the WAR file differ depending upon whether you need to include additional JDBC drivers for your reports. If there are no additional drivers, install the WAR file from the BIRT Report Engine installation. If you use additional JDBC drivers, you must pack them into the WAR file before you deploy it.

- To install the BIRT report viewer from the BIRT Report Engine WAR file, copy the BIRT Report Engine WAR file, birt.war to the Tomcat applications folder, $TOMCAT_INSTALL/webapps, as illustrated by the following DOS command:

  ```
  copy $BIRT_RUNTIME/birt.war $TOMCAT_INSTALL/webapps
  ```

  Then, restart Apache Tomcat.

- To install the BIRT report viewer with additional JDBC drivers, perform the following steps:

  1. Create a temporary directory and navigate to that directory.
  2. Unpack the BIRT Report Engine WAR file into the temporary directory, using a command similar to the following one:

     ```
     jar -xf $BIRT_RUNTIME/birt.war
     ```

     The BIRT Report Engine application unpacks into the temporary directory.
  3. Copy the JAR files for your JDBC drivers to the following folder in the temporary directory:

     ```
     WEB-INF/platform/plugins/
        org.eclipse.birt.report.data.oda.jdbc_<version>/drivers
     ```
  4. Repack the BIRT Report Engine WAR file from the temporary directory into a new birt.war file, using a command similar to the following one:

     ```
     jar -cf birt.war *
     ```

     This command creates birt.war in the temporary directory.
  5. Copy the new birt.war file to the Tomcat applications folder, $TOMCAT_INSTALL/webapps, as illustrated in the following DOS command:

     ```
     copy birt.war $TOMCAT_INSTALL/webapps
     ```
  6. Restart Apache Tomcat.

How to install the BIRT report viewer from the BIRT Report Engine viewer folder

To install the BIRT report viewer as an application in a file system folder, use the WebViewerExample folder in the BIRT Report Engine installation.

1. Navigate to $TOMCAT_INSTALL/webapps.
2. Create a subdirectory named birt.
3. Copy the web viewer example directory and all its subdirectories to this new folder, as illustrated by the following DOS command:

   ```
   xcopy /E "$BIRT_RUNTIME/WebViewerExample"
      $TOMCAT_INSTALL/webapps/birt
   ```

4 If the BIRT reports need additional JDBC drivers, add the JAR files for the JDBC drivers to the following directory:

```
$TOMCAT_INSTALL/birt/WEB-INF/platform/plugins/
   org.eclipse.birt.report.data.oda.jdbc_<version>/drivers
```

5 Restart Apache Tomcat.

Testing the BIRT report viewer installation

To test the installation of the BIRT report viewer described in earlier sections, type the following URL in a web browser address field:

```
<server_name>:<port>
```

<server_name> is the name of the application server and <port> is the port that the application server uses.

Tomcat opens the JavaServer Page (JSP), index.jsp. This file exists in both the WAR file and in the BIRT report viewer root directory. A link on this page runs the simple BIRT report design file, test.rptdesign. If the BIRT report viewer is installed correctly, Tomcat uses index.jsp to process the report design and generate and render the report that it describes. The first time you run the report, Tomcat compiles the JSP files that comprise the viewer, so there is a delay before the report appears in the web browser.

Changing the BIRT report viewer context root

By default, the context root of the URL for a web application is the path to the application directory or the WAR file. The default WAR file for the BIRT report viewer is birt.war, so the default URL to access a BIRT report from Apache Tomcat is similar to the following one:

```
http://localhost:8080/birt/run?__report=myReport.rptdesign
```

To change the BIRT context root, change the name of the /birt directory or the WAR file in $TOMCAT_INSTALL/webapps. Next, restart Apache Tomcat. In the URL to access your BIRT report, specify the name that you chose. For example, if you chose reports, the URL to access a BIRT report becomes:

```
http://localhost:8080/reports/run?__report=myReport.rptdesign
```

The URL examples in this section use a relative path to access the report design. The BIRT_VIEWER_WORKING_FOLDER parameter sets the path to access a report design as a relative path.

Changing the BIRT report viewer location

To place the BIRT report viewer in a location other than $TOMCAT_INSTALL/webapps, add a context mapping entry to the server.xml file in $TOMCAT_INSTALL/conf.

To add a context mapping entry, add the following line to server.xml just above the </host> tag near the end of the file:

```
<Context path="/birt_context" docBase="BIRT_Path"/>
```

where *birt_context* is the context root for the BIRT report viewer application and *BIRT_Path* is the absolute path to the directory containing the BIRT report viewer.

Save the changes to server.xml and restart Apache Tomcat to make the changes active.

Understanding the BIRT report viewer context parameters

To determine the locations for report designs, images in reports, and log files, the BIRT report viewer uses context parameters defined in the web.xml file. The path provided as the value for any of these parameters can be relative or absolute. A relative path is relative to the root folder of the BIRT report viewer application. A path to a writable location for a BIRT report viewer that is deployed as a WAR file must be an absolute path.

By default, the relative path for report designs is relative to the BIRT report viewer's root folder. Place all report designs in this folder or use the full path to the report design in the URL. Using a relative path is not convenient for deployment of the BIRT report viewer in a WAR file as changes to report designs would require repackaging the WAR file. To set a different location for report designs, change the BIRT_VIEWER_WORKING_FOLDER parameter in the BIRT report viewer application's web.xml file.

Other context parameters determine other aspects of the behavior of the BIRT report viewer, such as the default locale and the level of detail in the viewer's log files.

How to set the location for report designs

1. Navigate to $TOMCAT_INSTALL/webapps.
2. Open web.xml in a code editor by performing one of the following steps, based on your deployment configuration:
   - If you use a WAR file to deploy the BIRT report viewer, extract WEB-INF/web.xml from birt.war into a temporary location.
   - If you use a folder to deploy the BIRT report viewer, navigate to <context root>/WEB-INF.
3. Locate the following element:

   ```
   <context-param>
      <param-name>BIRT_VIEWER_WORKING_FOLDER</param-name>
      <param-value></param-value>
   </context-param>
   ```

4. Change the param-value element, so that it includes the absolute path to the folder for the report designs, similar to the following code where *Report_Folder* is the absolute path to the folder for the report designs:

```
<context-param>
   <param-name>BIRT_VIEWER_WORKING_FOLDER</param-name>
   <param-value>Report_Folder</param-value>
</context-param>
```

5 Save web.xml and close the editor.

6 If you use a WAR file to deploy the BIRT report viewer, replace WEB-INF/web.xml in birt.war with the file just modified.

7 Copy the report designs into the folder specified in the param-value element for BIRT_VIEWER_WORKING_FOLDER.

8 Restart Apache Tomcat.

Verifying that Apache Tomcat is running BIRT report viewer

If there are problems accessing the BIRT report viewer, use the Tomcat manager to verify that the BIRT report viewer is running on Apache Tomcat. Running the Tomcat manager requires a manager's account. If a Tomcat manager account does not exist, create one by adding the following line to $TOMCAT_INSTALL/conf/tomcat-users.xml:

```
<user name="admin" password="tomcat" roles="manager" />
```

Having a manager's account available, first open the Tomcat main page, which for a typical Apache Tomcat installation is http://localhost:8080, as shown in Figure 4-1.

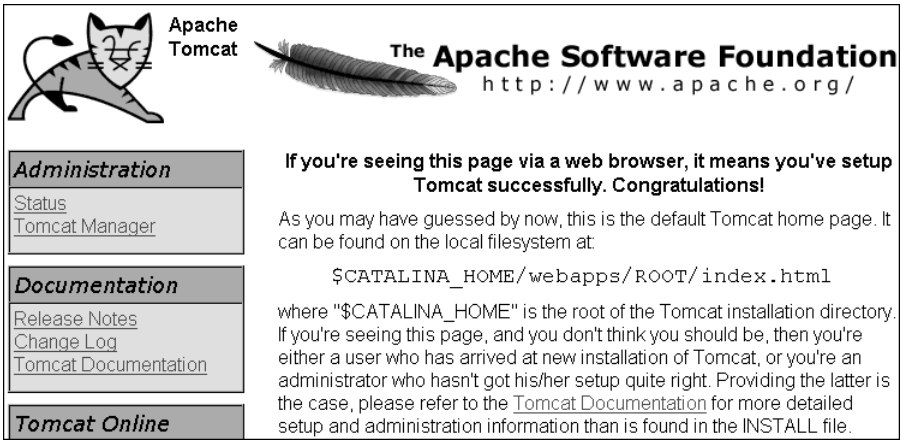


Figure 4-1 Apache Tomcat home page

On the Tomcat main page, in the Administration panel, choose Tomcat Manager. In the manager login window, type the user name and password of the manager account defined in the tomcat-users.xml file. When the BIRT report viewer application is running, the Running status for Eclipse BIRT Report Viewer is true, as shown in Figure 4-2.

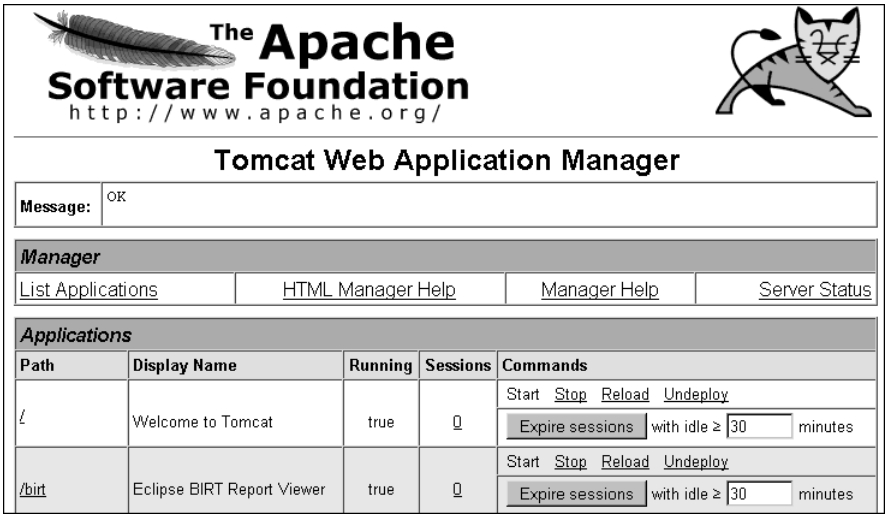


Figure 4-2 Running status for the BIRT report viewer

Placing fonts on the application server

BIRT Report Engine requires certain TrueType fonts to display a PDF report. BIRT searches for fonts in the common font directories for Windows and Linux. The directories that BIRT searches on a Windows system include:

- /windows/fonts for drives A through G
- /WINNT/fonts for drives A through G

and on a Linux system include:

- /usr/share/fonts/default/TrueType
- /usr/share/fonts/truetype

If PDF reports appear to be missing content, place the necessary fonts in any of the directories in the preceding list. Alternatively, specify your own font search path in the environment variable BIRT_FONT_PATH.

Viewing a report using a browser

After deploying the BIRT report viewer to your J2EE container, you can use the two available BIRT report viewer servlets to access your BIRT reports using a web browser. To view a BIRT report using a browser, use a URL of one of the following formats, where *parameter_list* is a list of URL parameters:

```
http://localhost:8080/birt/run?parameter_list
http://localhost:8080/birt/frameset?parameter_list
```

The run and frameset servlets display reports in two different ways. The run servlet displays the report as a stand-alone web page or a PDF file. If the report requires parameters, specify them in the URL. The frameset servlet displays a page in the browser with a toolbar containing four buttons to do the following tasks:

- Print the report.
- Display a table of contents.
- Display a parameters dialog.
- Display a dialog for exporting data.

Using connection pooling on Tomcat

BIRT provides support for connection pooling. For a Tomcat application server with a connection pool configured, BIRT reports can be set up to use a connection from the connection pool when connecting to a JDBC database. A BIRT JDBC data source uses the JNDI URL property to access the connection pool service on the web application server to get a connection from the pool.

Setting up a report to use connection pooling

Use BIRT Report Designer to configure reports to use connection pooling. The BIRT JDBC data source wizard requires configuring a direct-access connection as well as the JNDI URL. The reason for this requirement is that some JNDI service providers do not support client-side access. During design time, such JDBC drivers use the direct-access JDBC connection. The JDBC data-set query builder uses the direct JDBC connection to obtain its metadata.

In BIRT Report Designer, only the design functions directly related to a data-source design, such as Test Connection and Preview Results of a data set, attempt to use a JNDI name path. If the JNDI connection fails for any reason, the data source reverts to using the JDBC driver direct-access URL.

Similarly, at report run time, such as during report preview, the JDBC run-time driver attempts to look up its JNDI data source name service to get a pooled JDBC connection. If such look-up is not successful for any reason, the JDBC driver uses the direct-access URL to create a JDBC connection.

Using a jndi.properties file

Each individual JNDI application on the web application server uses its own environment settings stored in the JVM system properties.

The JNDI reads the following standard JNDI properties from the system

properties:

```
java.naming.factory.initial
java.naming.factory.object
java.naming.factory.state
```

```
java.naming.factory.control
java.naming.factory.url.pkgs
java.naming.provider.url
java.naming.dns.url
```

To simplify the task of setting up the JNDI initial context environment for an individual JNDI application, the JNDI feature supports the use of a jndi.properties resource file. Install this file in the drivers subfolder of the oda.jdbc plugin located at the following path:

```
WEB-INF\platform\plugins\
   org.eclipse.birt.report.data.oda.jdbc_<version>\drivers
```

This file contains a list of key-value pairs in the properties file format, key=value. The key is the name of the property, and the value is a string, for example, java.naming.factory.object=jnp://localhost:1099.

Here is an example of a JNDI resource file used with JBoss application server:

```
java.naming.factory.initial=
   org.jnp.interfaces.NamingContextFactory
java.naming.provider.url=jnp://localhost:1099
java.naming.factory.url.pkgs=org.jboss.naming:org.jnp.interfaces
```

The JDBC run-time driver looks for the jndi.properties file in the web application's folder tree. If the driver does not find the file or has a problem reading from it, the initial context uses the default behavior, as defined by javax.naming.Context, to locate any JNDI resource files. Configuring the classpath for classes referenced by the environment properties is necessary.

Configuring a JNDI connection object on Tomcat

The JNDI URL property for the JDBC data source supports retrieving a JDBC connection from a pool when BIRT reports are deployed to a web application server. More information about configuring connection pooling on Tomcat is available at:

```
http://tomcat.apache.org/tomcat-6.0-doc/jndi-resources-howto.html
```

How to configure a JNDI connection object on Tomcat

The following example assumes you already have deployed the BIRT report viewer to a Tomcat 6.0 application server in the folder, $TOMCAT_INSTALL /webapps/birt, as described earlier in this chapter.

1. Install the JDBC Driver. Make an appropriate JDBC driver available to both Tomcat internal classes and the web application, for example, by

installing the driver's JAR files into the following library directory in the Tomcat application server home folder:

```
$CATALINA_HOME/common/lib
```

2 Declare the resource requirements in the BIRT report viewer's WEB-INF/web.xml file. For example, add the following entry to set up a JNDI service for a MySQL format database with the name, MySqlDB:

```
<resource-ref>
   <description>Resource reference to a factory for
      java.sql.Connection</description>
   <res-ref-name>jdbc/MySqlDB</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
</resource-ref>
```

3 Configure Tomcat's resource factory as a Resource element in the BIRT report viewer's META-INF/context.xml file, similar to the following lines:

```
<Context>
   <Resource name="jdbc/MySqlDB" auth="Container"
      type="javax.sql.DataSource" maxActive="5" maxIdle="-1"
      maxWait="10000" username="root" password=""
      driverClassName="com.mysql.jdbc.Driver"
      url="jdbc:mysql://localhost:3306/classicmodels"
      description="MySQL DB"/>
</Context>
```

4 Make the JNDI URL in your report design match Tomcat's resource factory, similar to the following line:

```
java:comp/env/jdbc/MySqlDB
```

5 Open the report design using BIRT Report Designer. Edit the data source. In Edit Data Source, in JNDI URL, type the URL, as shown in Figure 4-3.

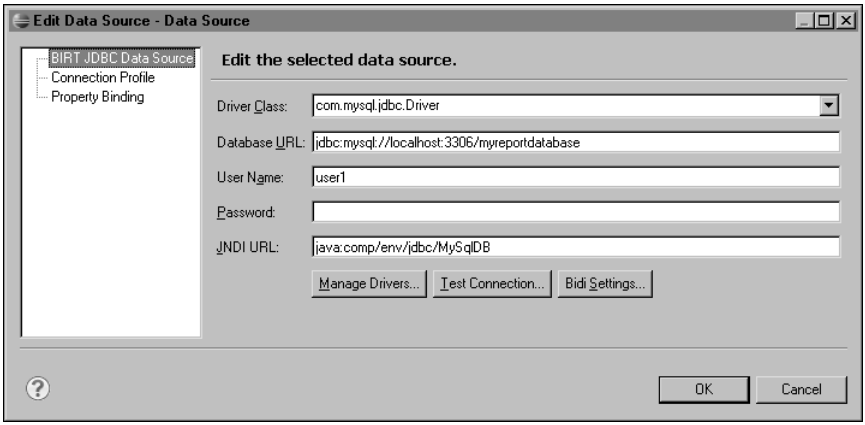


Figure 4-3 Setting the JNDI URL for a JDBC data source

6. Copy the report design to the BIRT report viewer root folder.
7. Restart the Tomcat service.
8. Run the report using a URL similar to the following one:

   `http://localhost:8080/birt/run?__report=myJNDIReport.rptdesign`

   The report uses a connection from the connection pool to connect to the database on a MySQL server.

This page intentionally left blank

Part

II

Understanding the BIRT Framework

This page intentionally left blank

Chapter

5

Understanding the BIRT Architecture

BIRT consists of many related components. This chapter provides an overview of the BIRT architecture, the BIRT components, the Eclipse components upon which BIRT relies, and the relationships that tie them all together.

Understanding the BIRT integration

As an Eclipse project, BIRT is tightly integrated with Eclipse frameworks and platforms. Like all Eclipse projects, BIRT is implemented as a set of Eclipse plug-ins. The BIRT plug-ins provide the functionality for all BIRT components, including BIRT applications, the engines that drive the applications, and supporting application programming interfaces (APIs). The BIRT plug-ins also provide the interface mechanism for communicating with several Eclipse frameworks and platforms.

The relationships between BIRT and the Eclipse components are most easily viewed as a stack. Each tier in the stack depends upon, uses, and integrates with the tier below it, as shown in Figure 5-1.

Figure 5-2 presents the various BIRT components and how they relate to one another. In this diagram, a component in a solid box is a standard BIRT component. A component in a dashed box is a custom component that a Java developer can provide. Some custom components are extensions of BIRT and others are applications that use the BIRT APIs. A component in a dotted box is a standard BIRT component that the containing component uses. For example, because BIRT Report Designer uses the design engine, the design engine appears in a dotted box within the box for BIRT Report Designer.

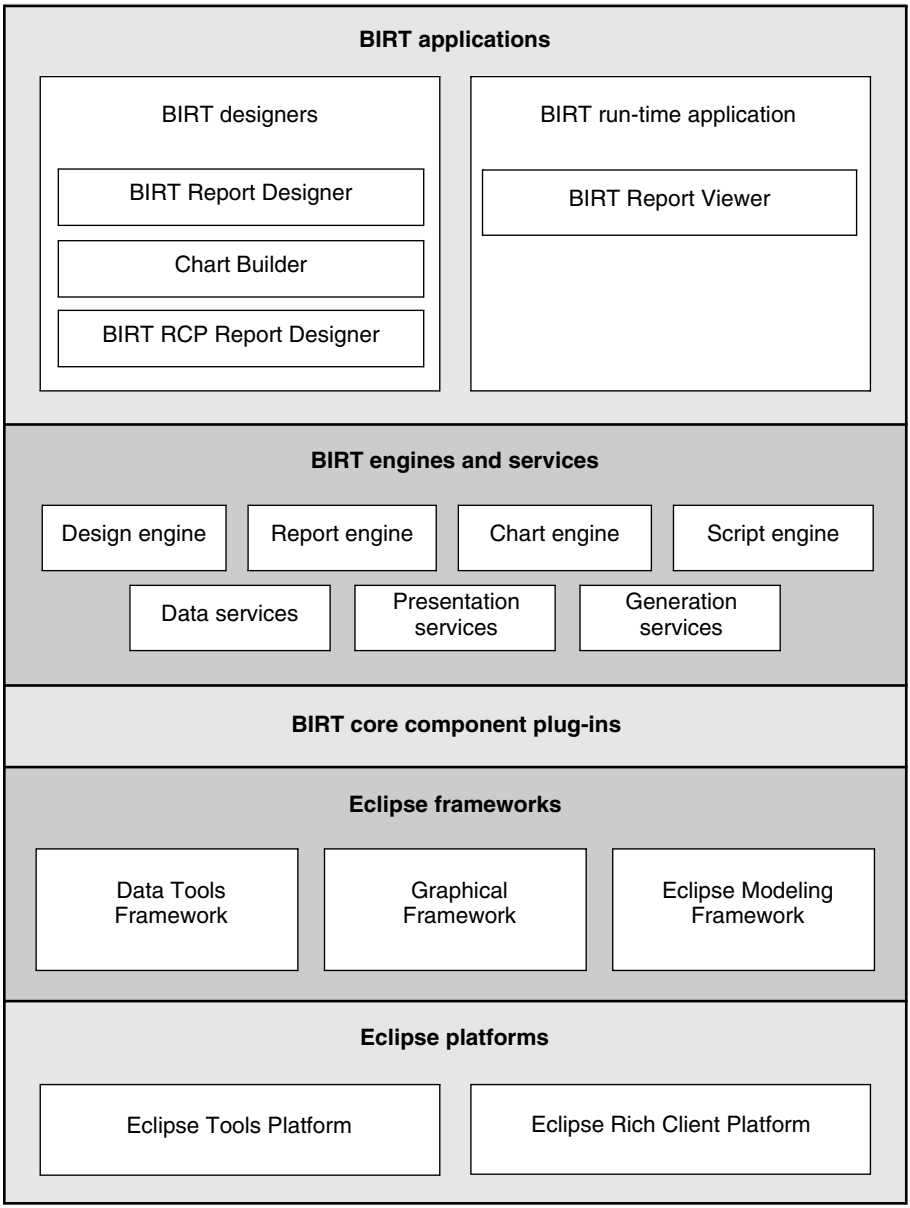


Figure 5-1 BIRT components as plug-ins to the Eclipse platform

Figure 5-3 shows the relationships among BIRT components as they generate a formatted report from a design.

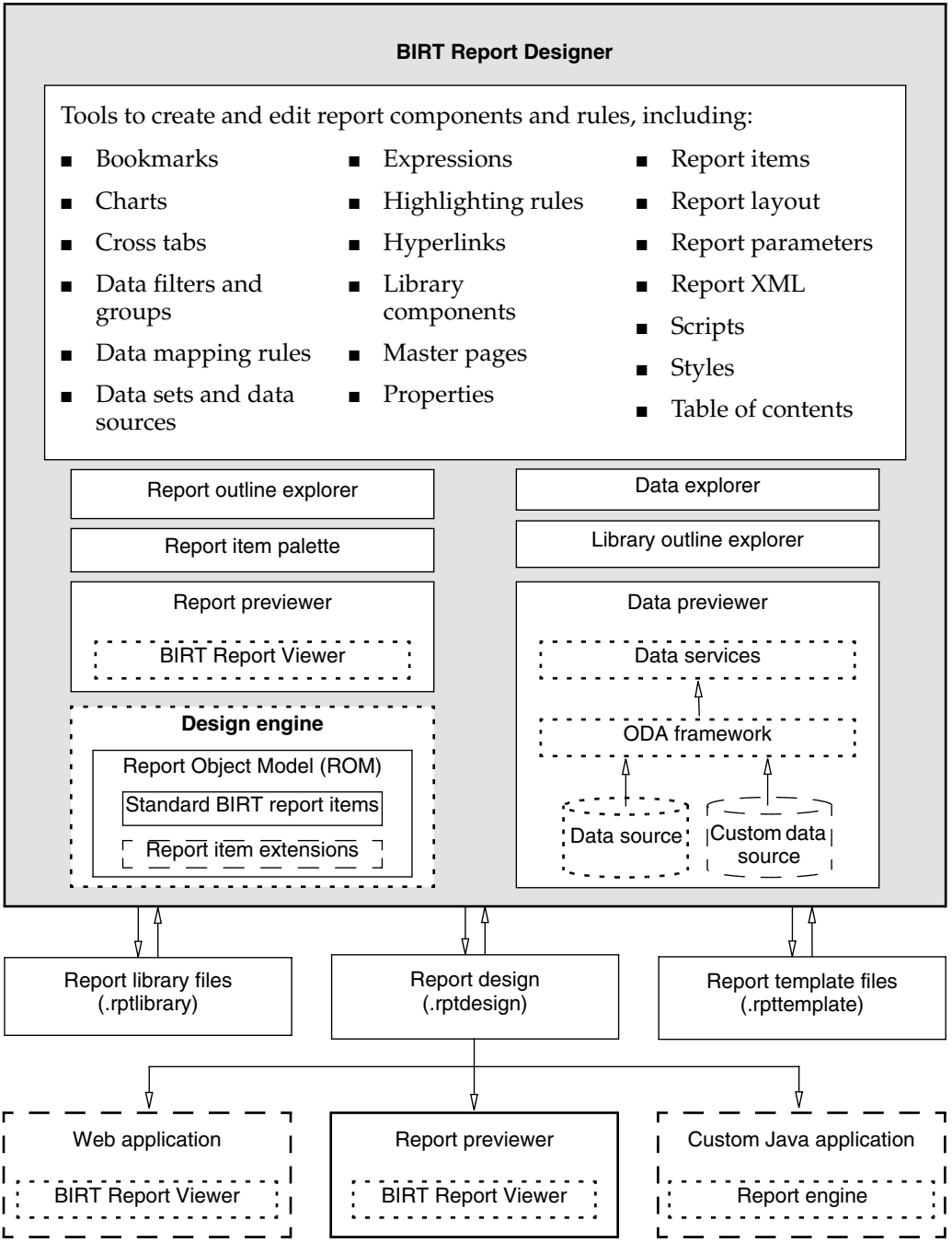


Figure 5-2 Task flow among standard BIRT components and custom components

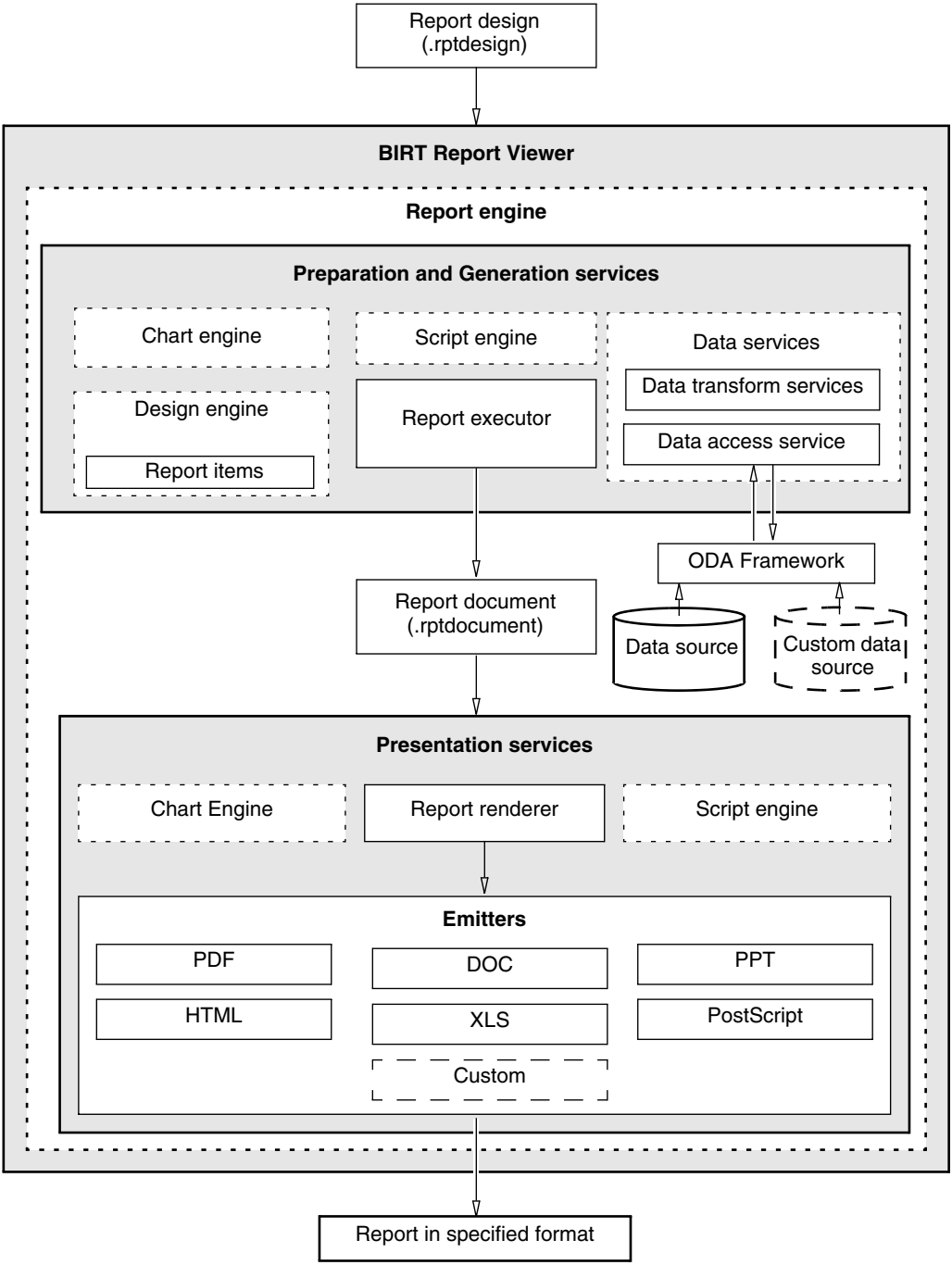


Figure 5-3 Task flow among BIRT and custom components to generate a report

BIRT Report Designer provides drag-and-drop capabilities to design reports quickly. The report designer uses the design engine to produce XML report design files. The BIRT Report Engine consumes these report design files, then, at run-time, fetches the appropriate data using queries defined at design-time. A BIRT report for immediate viewing is generated in memory and emitted in the desired output format, which can be a Microsoft Excel, non-paginated HTML, paginated HTML, PDF, PostScript, Microsoft PowerPoint, or Word file. To create a persistent report, the report engine transforms and summarizes the data and caches the generated report in an intermediate binary file, the report document file. This caching mechanism enables BIRT to scale to handle large quantities of data. BIRT also provides a tightly integrated charting component, which supports including a variety of chart types in reports. The charts are rendered at runtime using the charting engine. This charting engine also supports creating chart output separate from a report. Figure 5-4 illustrates this architecture and process flow.

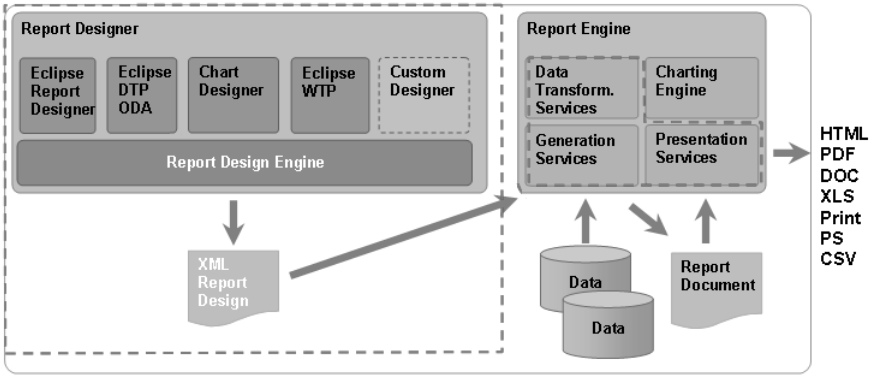


Figure 5-4 BIRT architecture and process flow

About the BIRT applications

There are three BIRT applications: BIRT Report Designer, BIRT RCP Report Designer, and BIRT Report Viewer. The two report designers are very similar. BIRT Report Designer runs as a set of Eclipse plug-ins and lets you build reports within the Eclipse workbench. BIRT RCP Report Designer has a simplified report design interface based on Eclipse Rich Client Platform.

About BIRT Report Designer and BIRT RCP Report Designer

BIRT Report Designer is a graphical report design tool. BIRT Report Designer uses the design engine to generate a report design file based on the report object model (ROM). ROM supports the standard set of BIRT report items and custom report items. BIRT Report Designer also supports the reuse of a

report design by saving it as a template. You can also save individual report components in a component library, making those components accessible to other report designs.

The primary functional differences between the BIRT RCP Report Designer and BIRT Report Designer are:

- BIRT RCP Report Designer has no integrated debugger.
- BIRT RCP Report Designer does not support Java event handlers.

Other than these differences, the functionality of the two report designers is identical and all further mentions of BIRT Report Designer in this chapter apply equally to BIRT RCP Report Designer.

About the BIRT Viewer

BIRT provides the BIRT Viewer web application that includes all the necessary files to deploy to most J2EE application servers. This web application supports running reports and viewing paginated HTML, with a table of contents and bookmarks, and extracting data to a values file. URLs provide this level of integration. The host application forwards the reporting request to the BIRT Viewer and allows the user to interact directly with the report.

The BIRT Viewer is also available as an Eclipse plug-in. This plug-in is used within the report designer to preview and display a report while it is being developed. Within BIRT Report Designer, it is deployed to and works with the Eclipse Tomcat application server plug-in. As report requests are made, the Eclipse workbench starts the Tomcat application and launches the BIRT Viewer.

About the BIRT engines and services

An engine is a set of Java APIs that provide basic functionality in a specific domain. BIRT contains several engines, for example, the report engine, design engine, and chart engine. These engines provide several different types of services. A service is a set of Java classes that provide functionality using the API provided from different engines. For example, the generation services use the design engine API and report engine API to generate reports and produce report documents respectively.

About the design engine

The design engine contains the APIs used to validate and generate a report design file. BIRT Report Designer and any custom Java application that generates a BIRT report design use the design engine. The generation services also use the design engine when building the report document. The design

engine contains APIs that validate the elements and structure of the design file against the ROM specification.

About the report engine

The BIRT report engine enables XML report designs created by the BIRT Report Designer to be used by a J2EE/Java application. To support this functionality, the report engine provides two core services, generation and presentation.

The report engine provides extensions to support custom report items and custom output formats. The report engine also supports Java application developers who want to integrate powerful report generation and viewing capabilities into their applications without having to build the infrastructure from lower-level Java components.

The BIRT report engine API supports integrating the run-time part of BIRT into Java applications. The report engine provides the ability to specify parameters for a report, run a report to produce HTML, PDF, DOC, PS, or PPT output and fetch an image or chart.

About the generation services

The generation service within the report engine connects to the data sources specified in a report design, uses the data engine to retrieve and process the data, creates the report layout, and generates the report document. Report content can be either viewed immediately using the presentation services, or saved for later use. The saved report documents containing snapshot views of data can be retained for use and comparison over time.

About the presentation services

The presentation services process the report document created by the generation services and render the report to the requested format and the layout specified in the design. The presentation services use the data engine to retrieve and process data from the report document. The presentation services use whichever report emitter they require to generate a report in the requested format. BIRT has several standard emitters, HTML PDF, DOC, PPT, PS, and XLS. BIRT also supports custom emitters that modify these default formats or deliver new formats.

Extensions to the presentation engine and services provide display capability for chart report items and custom report items.

About the chart engine

The chart engine contains APIs to generate charts and associate them with data from a data source. The BIRT Report Viewer interprets any chart information in a report design and uses the chart engine to generate the chart specified by the design. Use of the chart engine is not restricted to a BIRT

application. Any Java application can use chart engine APIs to create and display a chart.

About the data engine and services

The data engine contains the APIs and provides services to retrieve and transform data. The data services retrieve data from its source and process the data as specified by the report design. When used by the generation engine, the data services retrieve data from the data source specified in the design. When used by the presentation engine, the data services retrieve data from the report document. The data engine extension of ODA provides the connection method and the drivers for data sources.

About data services

The data engine provides two key service types: data access services and data transformation services. The data access services communicate with the ODA framework to retrieve data. The data transformation services perform such operations as sorting, grouping, aggregating, and filtering the data returned by the data access services.

About the ODA framework

BIRT uses the ODA framework provided by the Eclipse Data Tools Platform project to manage ODA and native drivers, load drivers, open connections, and manage data requests. The ODA framework contains extension points that support adding a custom ODA driver. Write a custom ODA driver if you have a data source that BIRT does not support and a scripted data source is not desired. Use of a custom ODA driver may require extending not only the data engine but also BIRT Report Designer. A BIRT Report Designer extension is necessary if the data source requires a user interface component to specify the data set and data source properties.

About the types of BIRT report items

A report item is a visual component of a report, such as a label, a list, or a chart. There are three categories of report items in BIRT: standard report items, custom report items, and the chart report item.

About standard report items

A report item is a visual component of a report. A report item can be as simple as a label or as complex as a cross tab. Every report item has an icon on the Palette view in BIRT Report Designer.

About custom report items

Custom report items are either new report items or extensions of existing report items. An example of an extension to a report item is adding a property, such as color. An example of a new report item is the rotated text report item, which is a reference implementation of a report item extension.

Creating a new report item and extending an existing report item both involve extending BIRT through the Eclipse plug-in mechanism. Custom items require an extension to one or more of the following components to support the new item:

- BIRT Report Designer
  Extending BIRT Report Designer provides user interface components for a report developer to specify properties and other settings for the report item.
- The design engine
  Extending the design engine validates the report item settings provided by a report developer.
- The report engine
  Extending the report engine supports generating and presenting report output for the report item.

About the chart report item

A chart report item is a standard BIRT component implemented as a BIRT extension. The user interface for creating a chart report item is a chart builder that steps the report developer through the process of designing the chart and associating it with the appropriate data columns.

About the Report Object Model (ROM)

ROM is the model upon which BIRT is based. ROM is a specification for the structure, syntax, and semantics of the report design. The formal expression of ROM is through an XML schema and a semantic definition file. The ROM specification appears in the following plug-in JAR file:

```
$INSTALL_DIR\eclipse\plugins\
   org.eclipse.birt.report.model_<version>.jar
```

About the types of BIRT files

BIRT Report Designer uses four types of files:

- Report design files

- Report document files
- Report library files
- Report template files

The following sections provide a brief overview of each of these file types.

About report design files

A report design file is an XML file that contains the report design, the complete description of a BIRT report. The report design describes every aspect of a report, including its structure, format, data sources, data sets, JavaScript event handler code, and the names of Java event handlers. BIRT Report Designer creates the report design file and BIRT report engine processes it.

The file extension of a report design file is rptdesign.

About report document files

A report document file is a binary file that encapsulates the report design, incorporates the data, and contains additional information, such as data rows, pagination information, and table of contents information.

The file extension of a report document file is rptdocument.

About report library files

A report library file is an XML file that contains reusable and shareable BIRT report components. A report developer uses Resource Explorer in BIRT Report Designer to provide shared access to a library, update a library, and use report elements from a library.

A BIRT report library can contain any report element, such as:

- Data sets and data sources
- Embedded images
- Event handler code
- Styles
- Visual report items

The file extension of a report library file is rptlibrary.

About report template files

A report template is an XML file that contains a reusable design. A report developer can use a template as a basis for developing a new report. A report developer uses a report template to maintain a consistent style across a set of

report designs and for streamlining the report design process. A report template can specify many different elements of a report, including:

- One or more data sources
- One or more data sets
- Part or all of the layout of a report design, including grids, tables, lists, and other report items
- Grouping, filtering, and data binding definitions
- Styles
- Library components
- Master pages
- Cheat sheets

Report templates act as a starting point for report development. They speed up report development by capturing the layout of common types of reports. They also make it easy to create reports with a consistent look. Building BIRT templates is similar to building BIRT reports. The difference lies in converting report items into template report items which act as placeholders.

The file extension of a report template file is rpttemplate.

About custom Java applications

Java developers can use the BIRT APIs to create a custom report designer or a custom report generator.

About a custom report designer

A custom report designer is a Java application that a Java developer creates to generate a well-formed report design file based on specific requirements. A custom report designer does not necessarily include a user interface. A typical example of a custom report designer is a Java application that dynamically determines the content, structure, or data source for a report, based on business logic. A custom report designer uses the same design engine API as BIRT Report Designer.

About a custom Java report generator

A custom Java report generator performs the same function as the BIRT report generator and is typically integrated into either a web application or a stand-alone Java application. A custom Java report generator uses the report engine API to read a report design file and generate a report. A custom Java report generator can use business logic to implement security requirements, control content, and determine the output format.

About extensions to BIRT

Through its public APIs and the BIRT extension framework, BIRT enables a Java developer to expand the capabilities of BIRT. BIRT uses Eclipse extensions to enable extending the functionally of the framework. The extension points provided by BIRT support the creation of new graph types, additional data sources, report controls, and emitters for rendering to additional outputs. These extension points appeal to users who have specialized data access and formatting needs. The following list shows some of the possible custom extensions:

- A custom report item

  A custom report item is a report item extension. This report item can be an extension, an existing BIRT report item, or a new report item.

- A custom ODA data source driver

  A custom ODA data source driver is a custom ODA extension that connects to a data source type other than those that BIRT directly supports.

- A custom report emitter

  A custom report emitter generates a report in a format other than HTML or PDF.

Later chapters in this book provide fully worked examples of all these types of extensions.

Chapter

6

Understanding the Report Object Model

This chapter provides an overview of the BIRT Report Object Model (ROM) and the primary elements that comprise the model. ROM defines the rules for constructing a valid report design file in much the same way that HTML defines the rules for constructing a valid web page. ROM, therefore, is the model for the BIRT report design, template, or library file in the same way that HTML is the model for the web page. For information about every component of ROM, see the online help entry Report Object Model (ROM) Definitions Reference in BIRT Programmer Reference.

About the ROM specification

The ROM specification defines a set of XML elements that describe the visual and non-visual components of a report. The XML file that BIRT Report Designer generates to describe a report design, library, or template consists entirely of ROM elements. Visual components, known as report items, include data items, labels, and tables. ROM also provides the framework for extended report items such as charts and cross tabs. Non-visual components, for example, data cubes, data sets, data sources, report parameters, and styles, support report items, but do not appear in a report. The ROM specification defines elements, their properties, and an element's relationship to other elements. The ROM specification describes elements by their methods, properties, slots, and styles. ROM elements describe:

- The data source and query with which to populate a report
- The placement, size, style, and structure of report items
- The report page layout

The report design, template, or library file contains XML elements that describe the ROM elements in the file. The BIRT design engine interprets the ROM elements using the ROM specification and the ROM schema. The ROM specification describes the content of each report element type. The ROM schema describes the supported structure of the XML in a file. Each BIRT file type appears in the ROM schema. Examining the XML in a BIRT file and the ROM schema shows that a template's structure is identical to a report design.

BIRT Report Designer displays the elements that the design engine interprets. Visual report items appear in the layout window. Data-related items such as cubes, data sets, and report parameters appear in the data explorer. All elements in the report design appear in the Outline view.

ROM methods

A ROM element can have one or more methods, called event handlers. BIRT fires many different events during the course of executing a report. When BIRT fires an event, the appropriate event handler executes to handle the event. By default, event handlers are empty methods that do nothing. By supplying code for an event handler, a report developer customizes and extends the functionality of BIRT. Supplying code for an event handler is called scripting. An event handler can be scripted in either JavaScript or Java.

Report items can have four events: onPrepare, onCreate, onPageBreak, and onRender. Each event fires in a specific phase of report creation. onPrepare fires in the preparation phase. onCreate fires during the generation phase. onRender and onPageBreak fire during the presentation phase.

ROM properties

ROM element properties are typed. Property types are similar to variable types in programming or data types in database terminology. Like variables and data types, ROM property types can be simple or complex. Design.xsd defines these types. Simple types include color, dimension, number, and string. Complex types include lists and structures. A complex type contains more than one component. For example, a text type contains both the text and a resource key used to internationalize the text.

ROM slots

A ROM slot is a container for elements of defined types. For example, a report element has a Body slot that contains any number of any type of report item. The Styles slot in the report element contains only Style items, which are the styles available to the report.

ROM styles

The ROM style system is based on cascading style sheets (CSS), where a style set in a container cascades to its contents. The Report element contains all

other elements, so the style property of the Report element defines the default style for the entire report. An element within the report can override the default style. A report developer can either choose a style from a defined set of styles or create a new style. Typical style attributes include alignment, background image, color, and text size.

About the ROM schema

The ROM schema, written in the XML Schema language, encapsulates the ROM specification. XML Schema provides a standard way of defining the structure, content, and semantics of an XML file. XML Schema is similar to Document Type Definition (DTD). The ROM schema, therefore, contains the formal expression of the content, structure, and semantics of the ROM report design. The ROM schema, design.xsd, is located at:

```
http://www.eclipse.org/birt/2005/design
```

Design.xsd is also in the plug-in, org.eclipse.birt.report.model.

A statement similar to the following one appears at the top of every report design, library, or template file:

```
<report xmlns="http://www.eclipse.org/birt/2005/design"
   version="3.2.21" id="1">
```

BIRT uses this statement, which identifies the version of the schema, to interpret the file structure. A file is not valid if it contains elements that are not defined in the schema or that violate the rules in the schema.

Opening a file using a schema-aware tool such as XMLSpy supports verifying the file against the schema. Using a schema-aware tool also enables a developer of a custom report designer to verify the output of the custom report designer.

The ROM schema defines syntax that supports extensions to BIRT without making changes to the actual schema. For example, an extended item uses the following tag:

```
<extended-item name="extension">
```

The ROM schema defines properties using the following syntax:

```
<property name="propertyName">value</property>
```

The ROM schema does not define any actual properties. ROM element properties are defined in another file, rom.def.

About the rom.def file

The rom.def file contains metadata defining the specific ROM elements, their properties, their slots, and their methods. You can find rom.def in:

```
$INSTALL_DIR\eclipse\plugins
   \org.eclipse.birt.report.model_<version>.jar
```

The rom.def file is an internal file that the design engine uses to present a property sheet for a ROM element. The property sheet for an element contains the element's properties and their types, the element's methods, and valid choice selections for each of the element's properties. When the BIRT development team changes the structure of a ROM element, the changed item includes an attribute that shows the BIRT version in which that change occurred, for example:

```
<Structure displayNameID="Structure.FormatValue"
   name="FormatValue" since="2.6">
```

The rom.def file specifies the following kinds of metadata:

- Choice

  A choice definition specifies all the allowable values that an attribute can have. Most choice definitions relate to style attributes. The following example from rom.def defines all the allowable font families available to a fontFamily style specification:

  ```
  <ChoiceType name="fontFamily">
     <Choice displayNameID="Choices.fontFamily.serif"
        name="serif" />
     <Choice displayNameID="Choices.fontFamily.sans-serif"
        name="sans-serif" />
     <Choice displayNameID="Choices.fontFamily.cursive"
        name="cursive" />
     <Choice displayNameID="Choices.fontFamily.fantasy"
        name="fantasy" />
     <Choice displayNameID="Choices.fontFamily.monospace"
        name="monospace" />
  </ChoiceType>
  ```

- Class

  A class definition defines a Java class that a report designer application can access using the BIRT model API. Class definitions describe the following component types:

  - Data types, such as Array, Number, and String
  - Functional classes, such as Global, Math, and RegExp

  A class definition defines attributes, constructors, localization, members, and methods identifiers. The following example from rom.def shows part of the definition of the Number class:

  ```
  <Class displayNameID="Class.Number" name="Number"
     native="true" toolTipID="Class.Number.toolTip">
     <Constructor displayNameID="Class.Number.Number"
        name="Number" returnType="Number"
        toolTipID="Class.Number.Number.toolTip">
  ```

```
      <Argument name="value" tagID="Class.Number.Number.value"
      type="Object"/>
   </Constructor>
   <Member dataType="number"
      displayNameID="Class.Number.MAX_VALUE" isStatic="true"
      name="MAX_VALUE"
      toolTipID="Class.Number.MAX_VALUE.toolTip"/>
   <Member dataType="number"
      displayNameID="Class.Number.MIN_VALUE" isStatic="true"
      name="MIN_VALUE"
      toolTipID="Class.Number.MIN_VALUE.toolTip"/>
   ...
   <Method displayNameID="Class.Number.toExponential"
      name="toExponential" returnType="String"
      toolTipID="Class.Number.toExponential.toolTip" >
      <Argument name="digits"
         tagID="Class.Number.toExpoential.digits" type="number"
      />
   </Method>
   <Method displayNameID="Class.Number.toFixed" name="toFixed"
      returnType="String"
      toolTipID="Class.Number.toFixed.toolTip">
      <Argument name="digits"
         tagID="Class.Number.toFixed.digits" type="number"/>
      </Method>
   ...
</Class>
```

- Element

  The rom.def file contains an element definition for every ROM element. Every element definition includes attributes, such as the element's name, display name, and the element type that it extends. ROM supports methods, properties, property visibility, slot, and style properties in an element definition. The following example from the rom.def file illustrates an element definition:

```
<Element allowsUserProperties="true" canExtend="true"
   displayNameID="Element.Parameter" extends="ReportElement"
   hasStyle="false" isAbstract="true" isNameRequired="true"
   name="Parameter" nameSpace="parameter" since="1.0">
   <Property displayNameID="Element.Parameter.helpText"
      name="helpText" runtimeSettable="false" since="1.0"
      type="string"/>
   <Property displayNameID="Element.Parameter.helpTextID"
      name="helpTextID" type="resourceKey"/>
   <Property displayNameID="Element.Parameter.promptText"
      name="promptText" runtimeSettable="false" since="2.0"
      type="string"/>
   <Property displayNameID="Element.Parameter.promptTextID"
      name="promptTextID" since="2.1" type="resourceKey"/>
```

```
   <Property displayNameID="Element.Parameter.hidden"
      name="hidden" runtimeSettable="false" since="1.0"
      type="boolean">
      <Default>false</Default>
   </Property>
   <Method context="factory"
      displayNameID="Element.Parameter.validate"
      name="validate" returnType="boolean" since="2.5"
      toolTipID="Element.Parameter.validate.toolTip">
      <Argument name="reportContext"
         tagID="Element.Parameter.validate.reportContext"
         type="org.eclipse.birt.report.engine.api.script
         .IReportContext"/>
   </Method>
</Element>
```

The property visibility property specifies whether BIRT exposes an inherited property to the user interface. For example, the label element extends ReportItem, which has dataSet and dataBindingRef properties. Because a label does not bind data, a property inherited by a label element must not appear visible to a user. The following example shows the use of property visibility to hide these properties:

```
<Element canExtend="true" displayNameID="Element.Label"
   extends="ReportItem" isAbstract="false"
   javaClass="org.eclipse.birt.report.model.elements.Label"
   name="Label" selector="label" since="1.0" xmlName="label">
   ...
   <PropertyVisibility name="dataSet" visibility="hide"/>
   <PropertyVisibility name="dataBindingRef" visibility="hide"
   />
   ...
</Element>
```

A Slot property defines the element as a container and specifies the types of items that the slot contains. Slots appear in the user interface in BIRT views such as Outline. The following example illustrates a slot definition:

```
<Element allowsUserProperties="true" canExtend="true"
   displayNameID="Element.CascadingParameterGroup"
   extends="ParameterGroup" hasStyle="false" isAbstract="false"
   isNameRequired="true" javaClass="org.eclipse.birt.report
   .model.elements.CascadingParameterGroup"
   name="CascadingParameterGroup" since="2.0"
   xmlName="cascading-parameter-group">
   ...
   <Slot displayNameID="Element.CascadingParameterGroup.slot
   .parameters" multipleCardinality="true" name="parameters"
   since="2.0" xmlName="parameters">
      <Type name="ScalarParameter"/>
      <Trigger validator="CascadingParameterTypeValidator"/>
   </Slot>
</Element>
```

A style property defines the style components supported by the ROM element. The following example illustrates style property definitions for the data element. The label element definition does not have these style properties because it contains only string values, not numbers.

```
<Element allowsUserProperties="true" canExtend="true"
   displayNameID="Element.Data" extends="ReportItem"
   hasStyle="true" isAbstract="false" isNameRequired="false"
   javaClass="org.eclipse.birt.report.model.elements.DataItem"
   name="Data" selector="data" since="1.0" xmlName="data">
   ...
   <StyleProperty name="numberFormat"/>
   <StyleProperty name="numberAlign"/>
   ...
</Element>
```

- Structure

  A structure is a complex data type that typically consists of two or more members. A few structures that are candidates for future expansion have only a single member. The following example from the rom.def file illustrates the definition of a structure.

```
<Structure displayNameID="Structure.TimeInterval"
   name="TimeInterval" since="2.5.2">
   <Member displayNameID="Structure.TimeInterval.measure"
      isIntrinsic="true" name="measure" since="2.5.2"
      type="integer"/>
   <Member detailType="interval"
      displayNameID="Structure.TimeInterval.unit"
      isIntrinsic="true" name="unit" since="2.5.2"
      type="choice">
      <Allowed>hour,minute,second</Allowed>
   </Member>
</Structure>
```

- Style

  A style definition contains the least information of any type of metadata described in rom.def. A style definition defines the name of the style, its display name, and a reference value, as shown in the following example.

```
<Style displayNameID="Style.Report" name="report"
   reference="Overall default" />
```

- Validator

  A validator definition specifies a Java class with which to do validation. Two of the validator classes are for validating values. The rest are semantic validators. The following example from rom.def illustrates a semantic validator definition.

```
<SemanticValidator
   class="org.eclipse.birt.report.model.api.validators
   .CascadingParameterTypeValidator" modules="design, library"
   name="CascadingParameterTypeValidator" />
```

Understanding ROM elements

ROM elements are defined in a set of hierarchies. Abstract elements, which cannot be used directly in a BIRT file, are at the top of each hierarchy. These elements define key characteristics of concrete elements in the same way that abstract classes in Java define methods and variables that a concrete class implements. Report designs are made of concrete elements that derive from the abstract elements.

About the primary ROM elements

The primary ROM elements consist of abstract elements from which other elements derive and concrete elements that provide the overall file definition.

The following elements are the abstract components that form the basis for understanding ROM:

- DataSet
  DataSet defines the fundamental properties of a data set relating to columns and parameters.
- DataSource
  DataSource defines the methods that a concrete data source element must support.
- DesignElement
  DesignElement defines basic features of ROM elements. DesignElement represents any component of a report design that has properties.
- Listing
  Listing is the abstract base element for list and table items. Both items support a data set, filtering, sorting, and methods.
- MasterPage
  MasterPage defines the basic properties of a page.
- ReportElement
  ReportElement represents any item that can be named and customized. Most components in ROM derive from ReportElement, including the elements that are visible in the user interface, such as data sets, styles, master pages, and report items.
- ReportItem
  ReportItem is the base element for the visual elements. A report item includes a style. The style provides visual characteristics for any element that appears in a report, such as a section or report item.

The following concrete elements describe the root element of a BIRT file: