Oracle® GlassFish Server 3.0.1 Scripting Framework Guide

Part No: 821–1760–10
June 2010

100616@24378

Contents

Preface

Oracle GlassFish Server 3.0.1 Scripting Framework Guide explains how to develop scripting applications in languages such as Ruby on Rails and Groovy on Grails for deployment to GlassFish Server.

This preface contains information about and conventions for the entire Oracle GlassFish Server (GlassFish Server) documentation set.

GlassFish Server 3.0.1 is developed through the GlassFish project open-source community at `https://glassfish.dev.java.net/`. The GlassFish project provides a structured process for developing the GlassFish Server platform that makes the new features of the Java EE platform available faster, while maintaining the most important feature of Java EE: compatibility. It enables Java developers to access the GlassFish Server source code and to contribute to the development of the GlassFish Server. The GlassFish project is designed to encourage communication between Oracle engineers and the community.

The following topics are addressed here:

- "GlassFish Server Documentation Set" on page 7
- "Related Documentation" on page 9
- "Typographic Conventions" on page 10
- "Symbol Conventions" on page 10
- "Default Paths and File Names" on page 11
- "Documentation, Support, and Training" on page 12
- "Searching Oracle Product Documentation" on page 12
- "Third-Party Web Site References" on page 12

GlassFish Server Documentation Set

The GlassFish Server documentation set describes deployment planning and system installation. The Uniform Resource Locator (URL) for GlassFish Server documentation is `http://docs.sun.com/coll/1343.13`. For an introduction to GlassFish Server, refer to the books in the order in which they are listed in the following table.

TABLE P–1 Books in the GlassFish Server Documentation Set

Book Title	Description
Release Notes	Provides late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of the supported hardware, operating system, Java Development Kit (JDK), and database drivers.
Quick Start Guide	Explains how to get started with the GlassFish Server product.
Installation Guide	Explains how to install the software and its components.
Upgrade Guide	Explains how to upgrade to the latest version of GlassFish Server. This guide also describes differences between adjacent product releases and configuration options that can result in incompatibility with the product specifications.
Administration Guide	Explains how to configure, monitor, and manage GlassFish Server subsystems and components from the command line by using the `asadmin`(1M) utility. Instructions for performing these tasks from the Administration Console are provided in the Administration Console online help.
Application Deployment Guide	Explains how to assemble and deploy applications to the GlassFish Server and provides information about deployment descriptors.
Your First Cup: An Introduction to the Java EE Platform	Provides a short tutorial for beginning Java EE programmers that explains the entire process for developing a simple enterprise application. The sample application is a web application that consists of a component that is based on the Enterprise JavaBeans specification, a JAX-RS web service, and a JavaServer Faces component for the web front end.
Application Development Guide	Explains how to create and implement Java Platform, Enterprise Edition (Java EE platform) applications that are intended to run on the GlassFish Server. These applications follow the open Java standards model for Java EE components and APIs. This guide provides information about developer tools, security, and debugging.
Add-On Component Development Guide	Explains how to use published interfaces of GlassFish Server to develop add-on components for GlassFish Server. This document explains how to perform *only* those tasks that ensure that the add-on component is suitable for GlassFish Server.
Embedded Server Guide	Explains how to run applications in embedded GlassFish Server and to develop applications in which GlassFish Server is embedded.
Scripting Framework Guide	Explains how to develop scripting applications in languages such as Ruby on Rails and Groovy on Grails for deployment to GlassFish Server.
Troubleshooting Guide	Describes common problems that you might encounter when using GlassFish Server and how to solve them.

TABLE P–1 Books in the GlassFish Server Documentation Set *(Continued)*

Book Title	Description
Error Message Reference	Describes error messages that you might encounter when using GlassFish Server.
Reference Manual	Provides reference information in man page format for GlassFish Server administration commands, utility commands, and related concepts.
Domain File Format Reference	Describes the format of the GlassFish Server configuration file, `domain.xml`.
Java EE 6 Tutorial	Explains how to use Java EE 6 platform technologies and APIs to develop Java EE applications.
Message Queue Release Notes	Describes new features, compatibility issues, and existing bugs for GlassFish Message Queue.
Message Queue Administration Guide	Explains how to set up and manage a Message Queue messaging system.
Message Queue Developer's Guide for JMX Clients	Describes the application programming interface in Message Queue for programmatically configuring and monitoring Message Queue resources in conformance with the Java Management Extensions (JMX).

Related Documentation

Javadoc tool reference documentation for packages that are provided with GlassFish Server is available as follows:

- The API specification for version 6 of Java EE is located at `http://download.oracle.com/docs/cd/E17410_01/javaee/6/api/`.
- The API specification for GlassFish Server 3.0.1, including Java EE 6 platform packages and nonplatform packages that are specific to the GlassFish Server product, is located at: `https://glassfish.dev.java.net/nonav/docs/v3/api/`.

Additionally, the following resources might be useful:

- The Java EE Specifications (`http://java.sun.com/javaee/technologies/index.jsp`)
- The Java EE Blueprints (`http://java.sun.com/reference/blueprints/`)

For information about creating enterprise applications in the NetBeans Integrated Development Environment (IDE), see `http://www.netbeans.org/kb/`.

For information about the Java DB for use with the GlassFish Server, see `http://developers.sun.com/javadb/`.

The GlassFish Samples project is a collection of sample applications that demonstrate a broad range of Java EE technologies. The GlassFish Samples are bundled with the Java EE Software Development Kit (SDK), and are also available from the GlassFish Samples project page at `https://glassfish-samples.dev.java.net/`.

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–2 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`**
		`Password:`

AaBbCc123	A placeholder to be replaced with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized (note that some emphasized items appear bold online)	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file.

Symbol Conventions

The following table explains symbols that might be used in this book.

TABLE P–3 Symbol Conventions

Symbol	Description	Example	Meaning
`[ ]`	Contains optional arguments and command options.	`ls [-l]`	The `-l` option is not required.
`{ | }`	Contains a set of choices for a required command option.	`-d {y|n}`	The `-d` option requires that you use either the `y` argument or the `n` argument.
`${ }`	Indicates a variable reference.	`${com.sun.javaRoot}`	References the value of the `com.sun.javaRoot` variable.
-	Joins simultaneous multiple keystrokes.	Control-A	Press the Control key while you press the A key.
+	Joins consecutive multiple keystrokes.	Ctrl+A+N	Press the Control key, release it, and then press the subsequent keys.

TABLE P–3 Symbol Conventions *(Continued)*

Symbol	Description	Example	Meaning
→	Indicates menu item selection in a graphical user interface.	File → New → Templates	From the File menu, choose New. From the New submenu, choose Templates.

Default Paths and File Names

The following table describes the default paths and file names that are used in this book.

TABLE P–4 Default Paths and File Names

Placeholder	Description	Default Value
as-install	Represents the base installation directory for GlassFish Server. In configuration files, *as-install* is represented as follows: `${com.sun.aas.installRoot}`	Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: *user's-home-directory*`/glassfishv3/glassfish` Windows, all installations: *SystemDrive*`:\glassfishv3\glassfish`
as-install-parent	Represents the parent of the base installation directory for GlassFish Server.	Installations on the Oracle Solaris operating system, Linux operating system, and Mac operating system: *user's-home-directory*`/glassfishv3` Windows, all installations: *SystemDrive*`:\glassfishv3`
domain-root-dir	Represents the directory in which a domain is created by default.	*as-install*`/domains/`
domain-dir	Represents the directory in which a domain's configuration is stored. In configuration files, *domain-dir* is represented as follows: `${com.sun.aas.instanceRoot}`	*domain-root-dir*`/`*domain-name*

Documentation, Support, and Training

The Oracle web site provides information about the following additional resources:

- Documentation (`http://docs.sun.com/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://education.oracle.com/`)

Searching Oracle Product Documentation

Besides searching Oracle product documentation from the docs.sun.com web site, you can use a search engine by typing the following syntax in the search field:

search-term `site:docs.sun.com`

For example, to search for "broker," type the following:

```
broker site:docs.sun.com
```

To include other Oracle web sites in your search (for example, java.sun.com, www.sun.com, and developers.sun.com), use `sun.com` in place of `docs.sun.com` in the search field.

Third-Party Web Site References

Third-party URLs are referenced in this document and provide additional, related information.

Note – Oracle is not responsible for the availability of third-party web sites mentioned in this document. Oracle does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Oracle will not be responsible or liable for any actual or alleged damage or loss caused or alleged to be caused by or in connection with use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

◆ ◆ ◆ CHAPTER 1

Using JRuby on Rails With Oracle GlassFish Server

This chapter explains how to get started using JRuby on Rails with Oracle GlassFish Server 3.0.1.

Introduction to JRuby and Rails on Oracle GlassFish Server

This section provides an overview of JRuby and Rails in the context of Oracle GlassFish Server.

What Is Ruby on Rails ?

Ruby is an interpreted, dynamically-typed, object-oriented programming language. It has a simple, natural syntax that enables developers to create applications quickly and easily. It also includes the easy-to-use RubyGems packaging utility for customizing a Ruby installation with additional plug-ins.

Rails is a web application framework that leverages the simplicity of Ruby and eliminates much of the repetition and configuration required in other programming environments. With Rails, you can create database-backed web applications, complete with models and tables, by running a few one-line commands.

To learn more about Ruby on Rails, see Ruby on Rails.

What Is JRuby ?

JRuby is a Java implementation of the Ruby interpreter. While retaining many of the popular characteristics of Ruby, such as dynamic-typing, JRuby is integrated with the Java platform. With JRuby on Rails, you get the simplicity and productivity offered by Ruby and Rails and the power of the Java platform offered by JRuby, thereby giving you many benefits as a Rails developer, including the following:

- Access the rich set of Java libraries from Rails applications
- Use the powerful and secure support of Java Unicode strings with Rails applications
- JRuby on Rails applications can spin off and pool multiple threads because JRuby uses Java threads, which map to native Ruby threads

To learn more about JRuby, see the JRuby Home page.

JRuby on Rails, GlassFish Server, and GlassFish v3 Gem

Developing and deploying Rails applications on Oracle GlassFish Server offers several advantages over using a typical web server for running Rails applications:

- Provides a simple, integrated deployment environment; develop and deploy from a single toolkit
- Enables deployment of multiple Rails applications to a single GlassFish Server instance
- Makes it possible to configure a single Rails application to handle multiple requests

Rails applications can be deployed on GlassFish Server in either of two ways:

- Deploy the application to a directory in the GlassFish Server domain.
- Run the application using GlassFish v3 Gem.

Note – A *Gem* is a Ruby package that contains a library or an application. Rails itself is a Gem that is installed on JRuby. See "GlassFish v3 Gem" on page 33 for more information.

Installing JRuby and Rails

Installing JRuby and Rails on GlassFish Server comprises three general steps:

1. Download and install GlassFish Server.
2. Download and install JRuby.
3. Install Rails on top of your JRuby installation.

The instructions in this guide assume that GlassFish Server software is already installed and running. If GlassFish Server is not already installed, refer to the *Oracle GlassFish Server 3.0.1 Installation Guide* for installation instructions.

JRuby and Rails can be installed on GlassFish Server in either of two ways:

- **Using the GlassFish Server Update Tool**

  GlassFish Server Update Tool is a GUI-based application for selecting and installing GlassFish Server add-ons and updates. It is the easiest means for installing JRuby and Rails for use GlassFish Server.

  Note – Update Tool also includes a command-line (CLI) Image Packaging System (IPS) utility, called `pkg`, that provides the same core functionality as its GUI-based counterpart. The `pkg` tool is started with the *as-install*`/bin/pkg` command. Using the `pkg` utility to install JRuby and Rails is beyond the scope of this document. See the Update Center wiki for complete information about Update Tool and the `pkg` command.

- **In Standalone Mode**

  You can download and install JRuby and Rails separately from GlassFish Server, in standalone mode. When installed in standalone mode, integrating JRuby and Rails with GlassFish Server involves a few additional configuration steps.

The remainder of this section explains how to install JRuby and Rails using these two different installation methods.

- "To Install JRuby and Rails Using Update Tool" on page 15
- "To Install JRuby in Standalone Mode" on page 17

▼ To Install JRuby and Rails Using Update Tool

This procedure explains how to install JRuby and JRuby Gems (including Rails) directly on top of an existing GlassFish Server installation using the GUI-based Update Tool.

Before You Begin GlassFish Server must be **installed and running** before proceeding with these instructions.

In these instructions, the root GlassFish Server installation directory is referred to as *as-install*.

1 **In a command shell for your operating system, change to the *as-install*`/bin` directory and run the `updatetool` command.**

Tip – If you are running Update Tool from behind a firewall that implements a proxy server, you may need to set the `HTTP_PROXY` environment variable before running the `updatetool` command. For example, in a Solaris/Linux Bash shell:

```
export HTTP_PROXY=http://proxy-server:port
```

If this is the first time you have launched `updatetool`, the full Update Tool product will not yet be installed, and you are prompted to allow installation to proceed.

a. **Type y when prompted to install Update Tool.**

The installer downloads and installs the full Update Tool product and then exits.

b. **Type the `updatetool` command again to launch Update Tool.**

The Update Tool main window is displayed, with Available Updates highlighted.

2 **Select the JRuby on GlassFish and JRuby Gems packages from the list of Available Add-Ons, and then click Install to install the packages.**

The contents of the JRuby on GlassFish and JRuby Gems packages are as follows:

- **JRuby on GlassFish** – Contains JRuby 1.3.1
- **JRuby Gems** – Contains Rails 2.3.2, Warbler, `jdbc-mysql`, and `activerecord-jdbcmysql-adapter` packages.

3 **Set the `JRUBY_HOME` environment variable to the location of your JRuby installation.**

By default, when installed using Update Tool, JRuby is installed in a subdirectory below *as-install*`/glassfish`. The commands to set `JRUBY_HOME` will typically be as follows:

- **Solaris/Linux:**

```
export JRUBY_HOME=as-install/glassfish/jruby
```

- **Windows:**

```
set JRUBY_HOME=C:\as-install\glassfish\jruby
```

4 **Add the `JRUBY_HOME/bin` directory to your system path.**

For example:

- **Solaris/Linux:**

```
export PATH=$PATH:$JRUBY_HOME/bin
```

- **Windows:**

  ```
  set PATH=%JRUBY_HOME%\bin;%PATH%
  ```

5 **Proceed with the JRuby container configuration instructions in "Configuring JRuby for GlassFish Server" on page 18.**

▼ To Install JRuby in Standalone Mode

This procedure explains how to download and install JRuby and then configure it to work with GlassFish Server. If you prefer to use the GlassFish Server Update Tool to install JRuby directly on GlassFish Server, use the procedure in "To Install JRuby and Rails Using Update Tool" on page 15 instead.

1 **Download the desired JRuby package from the JRuby Downloads page and save it in the location of your choice.**

Packages are available in a variety of formats, including tar.gz, ZIP, Windows executable, and JAR.

2 **Unpack the JRuby package in the directory of your choice.**

For the purposes of this procedure, the directory in which you unpack the JRuby package is referred to as *jruby-install*.

3 **Set the `JRUBY_HOME` environment variable to the location of your JRuby installation.**

For example:

```
export JRUBY_HOME=jruby-install
```

4 **Add `JRUBY_HOME/bin` directory to your system path.**

For example:

```
export PATH=$PATH:$JRUBY_HOME/bin
```

5 **Perform the following additional steps to configure the JRuby installation to work with GlassFish Server.**

a. **Change to the GlassFish Server `bin` directory and start the GlassFish Server domain.**

   For example:

   ```
   cd as-install/glassfish/bin
   ./asadmin start-domain domain-name
   ```

b. **Specify the JRuby home directory:**

   ```
   ./asadmin configure-jruby-container --jruby-home=jruby-install
   ```

6 Install the required Rails Gems for standalone JRuby operation.

```
jruby -S gem install rails
```

The `-S` parameter tells JRuby to look for the `install rails` script anywhere in the *JRUBY_HOME* path.

Configuring JRuby for GlassFish Server

After performing the basic installation, you can further configure several aspects of JRuby to work with GlassFish Server:

- "Configuring the JRuby Container" on page 18
- "Configuring JRuby Deployment Options" on page 19
- "Configuring the JRuby Runtime Pool" on page 20
- "Configuring the JRuby Container Through GlassFish Server Administration Console" on page 20

Configuring the JRuby Container

The GlassFish Server `asadmin configure-jruby-container` subcommand provides several options for configuring a JRuby container. Changes made to a JRuby container using the `configure-jruby-container` subcommand are written to the GlassFish Server domain's `domain.xml` file, which makes the changes persistent.

The general syntax for configuring the JRuby container is:

```
asadmin configure-jruby-container --property=value
```

For example, the following command sets the `jruby-home` property:

```
asadmin configure-jruby-container --jruby-home=jruby-install
```

Table 1–1 lists the options provided by the `asdmin configure-jruby-container` subcommand.

TABLE 1–1 `asdmin configure-jruby-container` Options

Option	Description
`--help | -?`	Displays the help text for the subcommand.
`--monitoring`	If set to `true`, enables monitoring for the JRuby container. The default is `false`.

TABLE 1–1 `asdmin` `configure-jruby-container` Options *(Continued)*

Option	Description
`--jruby-home`	The directory in which JRuby itself (not the GlassFish Server JRuby container) is installed. The specified directory must exist or an error will occur. Note, however, that the `configure-jruby-container` subcommand does not check whether JRuby is installed in the directory. The default directory is *as-install*`/jruby`, which is the directory in which Update Tool installs JRuby. Therefore, if you obtained JRuby from Update Tool, this option is not required.
`--jruby-runtime`	The initial number of JRuby runtime instances in the pool. This number must be greater than `0`, greater than or equal to `--jruby-runtime-min`, and less than or equal to `--jruby-runtime-max`. The default is `1`.
`--jruby-runtime-min`	The minimum number of JRuby runtime instances in the pool. This number must be greater than `0`, and less than or equal to `--jruby-runtime` and `--jruby-runtime-max`. The pool will always be at least this large, but can also be larger. The default is `1`.
`--jruby-runtime-max`	The maximum number of JRuby runtime instances in the pool. This number must be greater than `0`, and greater than or equal to `--jruby-runtime` and `--ruby-runtime-min`. The default is `1`. Setting this value too high can cause `OutOfMemory` errors, either in the heap or `PermGen`.
`--show`	If set to `true`, displays the current settings of the Enterprise Server JRuby container. The default is `true`.

For additional information about the `asadmin` JRuby configuration options, see the `configure-jruby-container(1)` man page.

Configuring JRuby Deployment Options

You can change deployment-specific options for the JRuby application using the following command syntax:

```
asadmin deploy --property property=value[:property=value]
```

For example, the following command deploys an application using a JRuby instance that is different from the one configured for the GlassFish Server instance:

```
asadmin deploy --property jruby-home=latest-jruby-install application
```

See the Appendix A, "The asadmin Deployment Subcommands," in *Oracle GlassFish Server 3.0.1 Application Deployment Guide* for detailed instructions on using the extensive list of `asadmin deploy` options to deploy applications, including JRuby, on GlassFish Server.

Configuring the JRuby Runtime Pool

GlassFish Server 3.0.1 provides a JRuby runtime pool to enable the servicing of multiple concurrent requests. It should be noted, however, that Rails is not currently thread-safe, and while JRuby is able to take advantage of Java's native threading, Rails cannot benefit from it. Each JRuby runtime runs a single instance of Rails, and requests are handed off to whichever instance happens to be available at the time of the request.

See Table 1–1 for descriptions of the JRuby runtime pool options.

The following are some important points to remember when configuring the JRuby runtime pool:

- To allow consistent and fast runtime access for the requesting applications, the dynamic runtime pool maintains itself with the minimum number of runtimes possible. The pool may take an initial runtime value, but that value is not used after pool creation.
- The JRuby runtime pool values can be set either at the container level or at deploy time:
  - To set the runtime pool values at the container level, use the `asadmin configure-jruby-container` subcommand, as described in "Configuring the JRuby Container" on page 18.
  - To set the runtime pool values at deploy time, use the `asadmin deploy` subcommand, as described in "Configuring JRuby Deployment Options" on page 19.
- Runtime pool properties can be set simultaneously at both the container level and at deploy time. If both settings are used, the deploy runtime settings take precedence over the container runtime settings.

Configuring the JRuby Container Through GlassFish Server Administration Console

You can use the GlassFish Server Administration Console to configure a subset of JRuby container options:

- JRuby Home
- Initial Pool Size
- Minimum Pool Size
- Maximum Pool Size

These options are the same as those described in Table 1–1.

Note – When configured through the Administration Console, these settings apply to the currently running GlassFish Server instance only. When the instance is shut down, the changes will be lost. To make your changes persistent from instance to instance, use the `asadmin configure-jruby-container` command instead.

For complete information about using the GlassFish Server Administration Console, see "Administration Console" in *Oracle GlassFish Server 3.0.1 Administration Guide*.

▼ To Configure JRuby Container from the GlassFish ServerAdministration Console

The following procedure explains how to access a JRuby Container from the GlassFish Server Administration Console.

1. **Open the GlassFish Server Administration Console from a Web browser.**

   For example:

   ```
   http://localhost:4848
   ```

2. **Log in to the Administration Console, if configured for secure login.**

   Depending on your GlassFish Server configuration, your Administration Console may not require login credentials.

3. **Select Common Tasks→Configuration→JRuby Container from the tree in the panel on the left side of the Administration Console.**

4. **Change JRuby Container properties, as desired, in the Details panel.**

   See "Configuring the JRuby Container" on page 18 for explanations of JRuby container properties. For detailed descriptions of the JRuby configuration options, see the `configure-jruby-container(1)` man page.

Creating a Simple Rails Application

After installing and configuring JRuby on GlassFish Server, you are ready to start coding. This section explains how to create a simple Rails application, named `hello`, that displays the following message:

```
Welcome to JRuby on Rails on the Oracle GlassFish Server!
```

The process of creating and deploying the `hello` application can be broken down into four subtasks:

- "To Create the `hello` Application" on page 22

- "To Create the Controller and the View" on page 22
- "To Pass Data From the Controller to the View" on page 23
- "To Use Rails Without a Database" on page 24

After completing the procedures in this section, continue to "Deploying and Running a Rails Application" on page 25 for information about deploying applications to GlassFish Server.

▼ To Create the `hello` Application

This procedure explains how to create a simple Rails application, named `hello`, and deploy it on GlassFish Server 3.0.1.

Before You Begin JRuby and Rails must be installed as described in "Installing JRuby and Rails" on page 15 before proceeding with the instructions in this section.

1. **Select or create a directory in which you want to create the sample application.**

   In this example, a directory named `/apps/jruby-apps` is used.

2. **Change to the directory you are using for your applications, and then create a Rails application named `hello`.**

   ```
   cd /apps/jruby-apps
   jruby -S rails hello
   ```

   This command creates the `hello` subdirectory, which contains a set of automatically generated files and subdirectories.

   The directories containing the files that you will likely use most are:

   - `app`: Application code (controllers, helpers, models, views, layouts)
   - `config`: Configuration files (environments, initializers, locales), including database, boot, and route files
   - `public`: Files and resources that need to be accessed directly rather than accessed through the Rails call stack; includes images, style sheets, and HTML files

3. **Proceed with the instructions in "To Create the Controller and the View" on page 22.**

▼ To Create the Controller and the View

The next step in creating and deploying the `hello` application is to create an application *controller* and a default application *view*.

- The **controller** handles requests, dispatches them to other parts of the application as necessary, and determines which view to render.

- The **view** is the file that generates the output to the browser. In Rails, views are typically written with ErB, a templating mechanism.

1 **Change to the directory in which you created the `hello` application in "To Create the `hello` Application" on page 22.**

In this example:

```
cd /apps/jruby-apps/hello
```

2 **Create a controller and default view for your application:**

```
jruby script/generate controller home index
```

You should see a controller file called `home_controller.rb` in the `hello/app/controllers` directory, and a view file called `index.html.erb` in the `hello/app/views` directory.

3 **Proceed with the instructions in "To Pass Data From the Controller to the View" on page 23.**

▼ To Pass Data From the Controller to the View

After creating a controller and a view for the `hello` application, the next step is to pass data from the controller to the view. This procedure explains how to set an instance variable in the controller and then access its value from the view.

1 **Open the `hello/app/controllers/home_controller.rb` file in a text editor.**

2 **Add an instance variable named `@hello_message` to the action named `index` and then save the file.**

The file should now contain the following:

```
class HomeController < ApplicationController
def index
@hello_message = "Welcome to JRuby on Rails on Oracle GlassFish Server"
end
end
```

In Rails, actions map to views. In this case, when you access the `index.html.erb` file, the `index` action executes, making the `@hello_message` variable available to `index.html.erb`.

3 **Open the `hello/app/views/home/index.html.erb` file in a text editor.**

4 **Add the following output block at the end of the file and then save the file:**

```
<p><%= @hello_message %></p>
```

The file should now contain the following:

```
<h1>Home#index</h1>
<p>Find me in app/views/home/index.html.erb</p>
<p><%= @hello_message %></p>
```

This JRuby code, embedded into the view, inserts the value of `@hello_message` into the page. When you deploy run the `hello` application, “`Welcome to JRuby on Rails on the GlassFish Server`" is displayed in your browser.

5 **At this point, you can either:**

- **Proceed with the instructions in “To Use Rails Without a Database” on page 24, which are relevant to the `hello` sample, but will not be relevant for most applications used in a production environment.**
- **Proceed directly to “Deploying and Running a Rails Application” on page 25 for instructions on deploying the application to GlassFish Server.**

▼ To Use Rails Without a Database

Although Rails is intended for creating database-backed web applications, this example is simple enough that it does not require one. In this case, you can edit the `enviroment.rb` configuration file to indicate that your application does not use a database.

1 **Open `hello/config/environment.rb` file in a text editor.**

2 **Look for the following commented property in the file:**

```
#config.frameworks -= [ :active_record, :active_resource, :action_mailer ]
```

3 **Uncomment the property by removing the pound (#) character from the beginning of the line.**

The line should now read as follows:

```
config.frameworks -= [ :active_record, :active_resource, :action_mailer ]
```

4 **Exit and save the file.**

5 **Proceed to “Deploying and Running a Rails Application” on page 25 for instructions on deploying the application to GlassFish Server.**

Deploying and Running a Rails Application

There are two general ways to deploy a Rails application on GlassFish Server:

- Deploy it as a directory to the GlassFish Server domain, using either the `asadmin deploy` command or the GlassFish Server Administration Console.
- Run it using GlassFish v3 Gem.

The reminder of this section explains how to deploy a Rails application to GlassFish Server as a directory. For more information about running Rails applications using GlassFish v3 Gem, see "To Run a Rails Application on GlassFish v3 Gem" on page 34.

- "To Deploy a Rails Application as a Directory Using the `asadmin` Command" on page 25
- "To Run a Rails Application From the GlassFish Server `autodeploy` Directory" on page 26

Note – These procedures provide only the most basic deployment instructions. For comprehensive information about deploying applications to GlassFish Server, refer to the *Oracle GlassFish Server 3.0.1 Application Deployment Guide*.

▼ To Deploy a Rails Application as a Directory Using the `asadmin` Command

The following procedure explains how to use the `asadmin` command to deploy the sample `hello` application as a directory within a GlassFish Server domain, and how to access the application from your web browser. You can also use these same instructions to deploy any Rails application to GlassFish Server.

Before You Begin These instructions build upon the example `hello` application described in "Creating a Simple Rails Application" on page 21.

1 **Start the GlassFish Server domain to which you want to deploy the application:**

```
asadmin start-domain domain-name
```

2 **Change to the directory where you created the sample application.**

For example:

```
cd /apps/jruby-apps/hello
```

3 **Deploy the `hello` application with `asadmin` command:**

as-install`/bin/asadmin deploy hello`

- **If you want to use a different JRuby instance than the update installation or the instance you configured with the `configure-jruby-container` command, you can use the following deploy-time option:**

  `asadmin deploy --property jruby-home=`*jruby-install*

4 **Run the `hello` application using the following URL in your browser:**

`http://localhost:8080/hello/`

▼ To Run a Rails Application From the GlassFish Server `autodeploy` Directory

As with the instructions in "To Deploy a Rails Application as a Directory Using the `asadmin` Command" on page 25, this procedure builds upon the example `hello` application described in "Creating a Simple Rails Application" on page 21.

1 **Copy the entire `hello` application directory to the** *as-install*`/glassfish/domains/`*domain-dir*`/autodeploy` **directory.**

For example:

```
cp -r /apps/jruby-apps/hello glassfishv3/glassfish/domains/domain1/autodeploy
```

2 **Run the `hello` application using the following URL in your browser:**

`http://localhost:8080/hello/`

Accessing a Database From a Rails Application

One of the main strengths of Rails is that it makes it easy to create applications that access databases. This section explains how to create a simple application that accesses a book database using MySQL.

▼ To Set Up the MySQL Database Server

Before You Begin JRuby, Rails, and the required Gems should already be installed, as described in "Installing JRuby and Rails" on page 15.

1 **Download and install MySQL database server:**

MySQL Server is available from the MySQL Downloads page. MySQL installation instructions are available from the MySQL Documentation page.

2 **Configure the server according to the MySQL documentation, including entering a `root` password.**

3 **Start the MySQL server.**

4 **Install the JRuby `activerecord-jdbcmysql-adapter` gem, if necessary.**

For example:

```
gem install activerecord-jdbcmysql-adapter
```

5 **Modify the `database.rake` script for your Rails installation so it uses `jdbcmysql` rather than `mysql`.**

For example, if using the version of JRuby and Rails installed through GlassFish Server Update Center on Solaris or Linux, the `database.rake` file is located in:

```
$JRUBY_HOME/lib/ruby/gems/1.8/gems/rails-2.3.5/lib/tasks
```

Change all instances of `mysql` with `jdbcmysql`, and then save the file.

▼ To Create a Database-Backed Rails Application

1 **Create or select a directory for creating a database-backed Rails application, and change to that directory.**

In this example, a directory named `/apps/jruby-apps` is used.

2 **Create and configure a application template to use the MySQL database:**

```
jruby -S rails books -d mysql
```

This creates a `books` directory.

3 **Change to the `books/config` directory and open the `config/database.yml` file in a text editor.**

a. **Replace all instances of `adapter: mysql` with `adapter: jdbcmysql`.**

b. **Enter your MySQL root password under the development heading in the `database.yml` file.**

4 **Save the file and change back to the `books` directory, if you are not already there.**

5 **Create the database by running the following command:**

```
jruby -S rake db:create
```

The `rake` command invokes the Rake tool. The Rake tool builds applications by running Rake files, which are written in Ruby and provide instructions for building applications.

6. **(Optional) If desired, verify that a database named `books_development` was successfully created.**

   For example:

   ```
   mysql -u root -p -e 'show databases'
   ```

7. **Still in the `books` directory, create the scaffold and the `book` model for the application:**

   ```
   jruby script/generate scaffold book title:string \
   author:string isbn:string description:text
   ```

   When you run the `script/generate` command, you specify the name of the model, the names of the columns, and the types for the data contained in the columns.

   A scaffold is the set of code that Rails generates to handle database operations for a model object, which is `Book` in this case. The scaffold consists of a controller and some views that allow users to perform the basic operations on a database, such as viewing the data, adding new records, and editing records. Rails also creates the model object when generating the scaffold.

8. **Create the database tables:**

   ```
   jruby -S rake db:migrate
   ```

   When Rails is finished creating the tables, you should see output similar to the following:

   ```
   ==  CreateBooks: migrating ====================================================
   -- create_table(:books)
      -> 0.1470s
      -> 0 rows
   ==  CreateBooks: migrated (0.1470s) ===========================================
   ```

   If you need to reset the database later, you can run the following command:

   ```
   jruby –S rake db:reset
   ```

9. **(Optional) If desired, verify that two tables, named `books` and `schema_migrations`, were successfully created in the `books_development` database.**

   For example:

   ```
   mysql -u root -p -e 'show tables from books_development'
   ```

Accessing Java Libraries From a Rails Application

The primary advantage of developing with JRuby is that you have access to Java libraries from a Rails application. For example, say you want to create an image database and a web application that allows processing of the images. You can use Rails to set up the database-backed web application and use the powerful Java 2D API for processing the images on the server-side.

This section shows you how to get started using Java libraries in a Rails application while stepping you through building a simple Rails application that does basic image processing with the Java 2D API.

This application demonstrates the following concepts involved in using Java libraries in a Rails application:

- Giving your controller access to Java libraries
- Creating constants to refer to Java classes
- Performing file input and output using the `java.io` and `javax.imageio` packages
- Assigning Java objects to Ruby objects
- Calling Java methods and using variables
- Converting arrays from Java language arrays to Ruby arrays
- Streaming files to the client

For simplicity's sake, this application does not use a database. You will need a `JPEG` file to run this application.

▼ To Create the Rails Application That Accesses Java Libraries

1. **Change to the directory you want to create an application.**

   ```
   /apps/jruby-apps/
   ```

2. **Create an application by running this command:**

   ```
   jruby -S rails imageprocess
   ```

3. **Open the `imageprocess/config/environment.rb` file in a text editor, and modify the file as described in "To Use Rails Without a Database" on page 24.**

4. **Change to the `imageprocess` directory you just created.**

5. **Create a controller and default view for the application by running this command:**

   ```
   jruby script/generate controller home index
   ```

6. **Change to the `imageprocess/app/views/home` directory.**

7. **Create a second view by copying the default view to `seeimage.html.erb`:**

   ```
   cp index.html.erb seeimage.html.erb
   ```

▼ To Create the Views That Display the Images Generated by Java2D Code

In this task, you will perform the following actions:

- Load an image on which you want to perform image processing with Java2D.
- Make the initial view show the original image and provide a link that the user clicks to perform the `ColorConvertOp` image processing operation.
- Make the other view display the processed image.

1. **Find a JPEG image that you can use with this application.**

2. **Add the image to** `imageprocess/public/image` **file.**

3. **Change to** `imageprocess/app/views/home` **directory.**

4. **Open the** `index.html.erb` **file in a text editor.**

5. **Replace the contents of this file with the following HTML markup:**

   ```
   <html>
       <body>
           <img src="../../images/kids.jpg"/><p>
           <%= link_to "Perform a ColorConvertOp on this image", :action => "seeimage" %>
       </body>
   </html>
   ```

   This page loads an image from `imageprocess/public/images` and provides a link that references the `seeimage` action. The `seeimage` action maps to the `seeimage` view, which shows the processed image.

6. **Replace** `kids.jpg` **in the** `index.html.erb` **with the name of your image that you saved earlier in this procedure.**

7. **Save the** `index.html.erb` **file.**

8. **Open the** `seeimage.html.erb` **file in a text editor.**

9. **Replace the contents of this file with the following HTML markup:**

   ```
   <html>
       <body>
           <img src="/home/processimage"/><p>
           <%= link_to "Back", :action => "index" %>
       </body>
   </html>
   ```

The `img` tag on this page accesses the `processimage` action in `HomeController`. The `processimage` action is where you will put the Java2D code to process the image that you loaded into `index.html.erb`.

▼ To Add Java2D Code to a Rails Controller

With this task, you will add the code to process your JPEG image.

1. **Add the following line to `HomeController`, right after the class declaration:**

   ```
   include Java
   ```

   This line is necessary for you to access any Java libraries from your controller.

2. **Create a constant for the `BufferedImage` class so that you can refer to it by the shorter name:**

   ```
   BI = java.awt.image.BufferedImage
   ```

3. **Add an empty action, called `seeimage`, at the end of the controller:**

   ```
   def seeimage
   end
   ```

   This action is mapped to the `seeimage.html.erb` view.

4. **Give controller access to your image file using `java.io.File`, making sure to use the name of your image in the path to the image file. Place the following line inside the `seeimage` action:**

   ```
   filename = "#{RAILS_ROOT}/public/images/kids.jpg"
   imagefile = java.io.File.new(filename)
   ```

   Notice that you do not need to declare the types of the variables, `filename` or `imagefile`. JRuby can tell that `filename` is a `String` and `imagefile` is a `java.io.File` instance because that is what you assigned them to be.

5. **Read the file into a `BufferedImage` object and create a `Graphics2D` object from it so that you can perform the image processing on it. Add these lines directly after the previous two lines:**

   ```
   bi = javax.imageio.ImageIO.read(imagefile)
   w = bi.getWidth()
   h = bi.getHeight()
   bi2 = BI.new(w, h, BI::TYPE_INT_RGB)
   big = bi2.getGraphics()
   big.drawImage(bi, 0, 0, nil)
   bi = bi2
   biFiltered = bi
   ```

   Refer to The Java Tutorial for more information on the Java 2D API.

The important points are :

- You can call Java methods in pretty much the same way in JRuby as you do in Java code.
- You do not have to initialize any variables.
- You can just create a variable and assign anything to it. You do not need to give it a type.

6. **Add the following code to convert the image to grayscale:**

```
colorSpace = java.awt.color.ColorSpace.getInstance(
java.awt.color.ColorSpace::CS_GRAY)
op = java.awt.image.ColorConvertOp.new(colorSpace, nil)
dest = op.filter(biFiltered, nil)
big.drawImage(dest, 0, 0, nil);
```

7. **Stream the file to the browser:**

```
os = java.io.ByteArrayOutputStream.new
javax.imageio.ImageIO.write(biFiltered, "jpeg", os)
string = String.from_java_bytes(os.toByteArray)
send_data string, :type => "image/jpeg", :disposition => "inline",
    :filename => "newkids.jpg"
```

Sometimes you need to convert arrays from Ruby to Java code or from Java code to Ruby. In such cases, you need to use the `from_java_bytes` routine to convert the bytes in the output stream to a Ruby string so that you can use it with `send_data` to stream the image to the browser. JRuby provides some other routines for converting types, such as `to_java` to convert from a Ruby Array to a Java String. See Conversion of Types.

▼ To Run a Rails Application That Uses Java 2D Code

1. **Deploy the application on the GlassFish v3 Gem:**

```
asadmin deploy imageprocess
```

2. **Run the application by entering the following URL into your browser:**

`http://localhost:3000/home/index`

You should now see an image and a link that says, “`Perform a ColorConvertOp on this image.`”

3. **Click the link.**

You should now see a grayscale version of the image from the previous page.

Monitor Rails Applications on GlassFish Server 3.0.1

The GlassFish Server 3.0.1 offers monitoring services for various objects.

Monitoring is the process of reviewing the statistics of a system to improve performance or solve problems. The monitoring service can track and display operational statistics, such as the number of requests per second, the average response time, and the throughput. For more information on monitoring, see Chapter 8, "Administering the Monitoring Service," in *Oracle GlassFish Server 3.0.1 Administration Guide*.

Monitoring for JRuby Container

The JRuby container installed on the GlassFish Server can be configured to be monitored. By default the monitoring services are disabled.

Monitoring can be configured for the container using the following `asadmin` CLI command:

```
configure-jruby-container --monitoring=true
```

The monitoring service can be enabled or disabled. Enable monitoring for GlassFish Server JRuby container with the following runtime command:

```
enable-monitoring --modules jruby-container=HIGH
```

The other possible values are `OFF` or `LOW`. Disable the monitoring for JRuby container with the following command:

```
disable-monitoring --modules jruby-container=OFF
```

Viewing JRuby Container Statistics

JRuby statistics are available through both the `asadmin` CLI and the Administration Console. To learn more about different types of JRuby statistics available from GlassFish Server, see "JRuby Statistics" in *Oracle GlassFish Server 3.0.1 Administration Guide*.

GlassFish v3 Gem

Another way to work with JRuby is to install the GlassFish v3 Gem on a JRuby standalone instance. The GlassFish v3 Gem is just a lightweight version of the Oracle GlassFish Server 3.0.1 that runs as part of JRuby instance.

When you install the GlassFish v3 Gem on JRuby, you have a Oracle instance embedded in the JRuby virtual machine. This gives you a more complete development environment because you have everything you need for JRuby on Rails applications running inside the JRuby virtual machine in addition to everything you need from the GlassFish Server to create web applications.

▼ To Install the GlassFish v3 Gem

To install the GlassFish v3 Gem on your JRuby standalone instance, use the following procedure:

1. **Download and install JRuby and Rails according to the instructions in “To Install JRuby in Standalone Mode” on page 17.**
2. **Run the Gem installer to install the GlassFish v3 Gem:**

   ```
   jruby -S gem install glassfish
   ```
3. **Start your server with the following command:**

   ```
   jruby -S glassfish
   ```

▼ To Run a Rails Application on GlassFish v3 Gem

You can also run your JRuby on Rails application on the embedded GlassFish v3 Gem. Create a simple Rails application as described in “Creating a Simple Rails Application” on page 21. Instead of deploying it to the GlassFish Server, use the GlassFish v3 Gem to run that application as described in the following task.

1. **Change to the directory where a sample application has been created. For example:**

   ```
   cd /jruby-apps
   ```
2. **Deploy the previously created `hello` application:**

   ```
   jruby -S glassfish helloV3
   ```
3. **Run the application using the following URL in your web browser:**

   `http://localhost:3000/home/index`

   You should now see the following message in your browser window:

   ```
   Welcome to JRuby on Rails on the Oracle GlassFish Server!
   ```

   Note that the GlassFish v3 Gem runs on port `3000` and not the default port `8080`.

▼ To Deploy and Run the Database-Backed Web Application

Creating database-backed web applications was described in "To Create a Database-Backed Rails Application" on page 27. The following task describes running the created `books` application on the GlassFish v3 Gem. You can alternatively deploy it to your regular GlassFish Server using directory-based deployment, as described in "To Deploy a Rails Application as a Directory Using the `asadmin` Command" on page 25.

1. **Change to `/books` directory.**
2. **Deploy the application to the GlassFish v3 Gem by running the following command:**

   ```
   jruby -S glassfish books
   ```

3. **Run the application in your web browser using the following URL:**

   ```
   http://localhost:3000/books
   ```

   The opening page says "Listing books" and has an empty table, meaning that there are no book records in the database yet. To add book records to the table, do the next step.

4. **Add records to the table by clicking the New book link on the `index.html` page.**
5. **Enter the data for book on the `new.html` page and click Create.**

Using GlassFish v3 Gem CLI

The GlassFish v3 Gem offers several commands to configure the GlassFish Server server. You can change the configuration options either by using GlassFish Server command line or by modifying the `glassfish.yml` file. You view these options by using the following command:

```
glassfish -h
```

The options include `runtimes`, `runtimes-min`, `runtimes-max`, and also a configuration option to create a configuration file for GlassFish Server.

You can create a `glassfish.yml` file which is a default GlassFish Server configuration file generated by the following command:

```
jruby -S gfrake config
```

Introduction to Warbler

In "Deploying and Running a Rails Application" on page 25, direct deployment of a rails application to GlassFish Server has been described. Warbler provides an easier way to deploy a rails application to a Java application server.

What Is Warbler ?

Warbler is a Gem that makes WAR file out of a Rails, Merb, or Rack-based application. Warbler provides a minimal, flexible, Ruby-like way to bundle application files for deployment to a Java application server.

Warbler provides a set of out-of-the box defaults to allow most Rails applications to assemble and work without external Gem dependencies.

Warbler bundles JRuby and the JRuby-Rack Servlet adapter for dispatching requests to the application inside the Java application server, and assembles all jar files in the *WARBLER_HOME*`/lib/` directory into the application.

To learn more about Warbler, see Warbler.

Creating and Deploying a Simple Rails Application with Warbler

The procedure for creating a simple Rails application for Warbler, is similar to the procedure described in "Creating a Simple Rails Application" on page 21.

▼ To Create a Rails Application

1. **Create a new directory for the Warbler application. For example:**

   ```
   mkdir rails-warbler
   ```

2. **Change to `rails-warbler` directory and create a sample application called `hello`:**

   ```
   jruby -S rails hello
   ```

3. **Edit the `enviroment.rb` file to indicate that your application does not use a database:**

   Open `rails-warbler/hello/config/environment.rb` in a text editor.

4. **Remove the pound character (#) in front of line 21 to uncomment it so that it reads as follows and save:**

   ```
   config.frameworks -= [ :active_record, :active_resource, :action_mailer ]
   ```

5 **Exit and save the file.**

6 **Use Warbler to create a war file in `rails-warbler/hello` application directory:**

```
jruby -S warble
```

This creates a `hello.war` file in the directory.

▼ To Deploy the WAR File

1 **Change to the `rails-warbler/hello` application directory.**

2 **Deploy the application WAR file to the GlassFish Server by running the `asadmin` command:**

```
asadmin deploy hello.war
```

3 **Run the `hello` application by entering the following URL in your browser:**

`http://`*host-name*`:`*port*`/hello`

For Further Information

For more information on Ruby-on-Rails, JRuby, JRuby on GlassFish Server and Warbler, see the following resources:

- Ruby-on-Rails
- JRuby
- Everything on Scripting in Glassfish
- Warbler

◆ ◆ ◆

CHAPTER 2

Developing Grails Applications

This chapter introduces Groovy and Grails, a Java technology-based alternative to scripting.

- "Introduction to Groovy and Grails" on page 39
- "Installing Grails" on page 39
- "Creating a Simple Grails Application" on page 40

Introduction to Groovy and Grails

Groovy is a dynamic, object-oriented language for the Java Virtual Machine, which builds on the strengths of Java but has additional features inspired by languages such as Python, Ruby, and Smalltalk. For more information about Groovy, see the Groovy Project page.

Grails is an open source web application framework that leverages the Groovy language and complements Java web development. Grails is a standalone development environment that can hide all configuration details or allow integration of Java business logic. It provides easy-to-use tools to build web applications in Groovy. For more information about Grails, see the Grails Project page.

Installing Grails

Grails is available as an IPS package from GlassFish Server Update Tool. To develop and deploy Grails applications on GlassFish Server, you must first install the Grails module.

▼ To Install the Grails Module

1 **Install the Grails add-on component that is available from Update Tool.**

For information about Update Tool, see the *Oracle GlassFish Server 3.0.1 Installation Guide*.

2 **Create a `GRAILS_HOME` environment variable that points to the Grails directory,** *as-install*`/grails`**.**

3 **Add the** *as-install*`/grails/bin` **directory to the** `PATH` **environment variable.**

Example 2–1 Setting UNIX Environment Variables

On Solaris, Linux, and other operating systems related to UNIX, use the following commands to set the `GRAILS_HOME` and `PATH` environment variables:

```
export GRAILS_HOME=as-install/glassfish/grails
export PATH=$GRAILS_HOME/bin:$PATH
chmod a+x $GRAILS_HOME/bin/*
```

Example 2–2 Setting Windows Environment Variables

On the Windows operating system, use the following commands to set the `GRAILS_HOME` and `PATH` environment variables:

```
set GRAILS_HOME=C:\GlassFish\grails
set PATH=%GRAILS_HOME%\bin;%PATH%
```

Creating a Simple Grails Application

To create and run a simple `helloworld` application, perform the following tasks:

- "To Create the `helloworld` Application" on page 40
- "To Create the `hello` Controller" on page 41
- "To Run a Grails Application Using Standard Deployment" on page 41

For more information on creating Grails applications, see the Grails Quick Start guide.

▼ To Create the `helloworld` Application

1 **Change to the** *as-install*`/grails/samples` **directory.**

2 **Run the `grails create-app helloworld` command.**

The `grails create-app` command creates a `helloworld` application that you can modify.

▼ To Create the `hello` Controller

1. **Change to the *as-install*`/grails/samples/helloworld` directory.**
2. **Run the `grails create-controller hello` command.**

   The `grails create-controller` command creates a controller file that you can modify in the `/grails/samples/helloworld/grails-app/controllers` directory:
3. **Edit the generated `HelloController.groovy` file so that it looks as follows:**

```
class HelloController {

       def world = {
                render "Hello World!"
        }
    //def index = { }
}
```

▼ To Run a Grails Application Using Standard Deployment

1. **Change to the application directory. For example:**

   `cd` *as-install***`/grails/samples/helloworld`**
2. **Create the WAR file using the following command:**

```
grails war
```

   This command creates a WAR file, `helloworld-0.1.war` in the `helloworld` application directory. The WAR file contains all the application's dependencies, and various jar files.
3. **Deploy the WAR file in one of the following ways:**

   - **In the Administration Console, open the Applications component, go to the Web Applications page, select the Deploy button, and type the path to the WAR file.**

     For example:

     as-install`/grails/samples/helloworld/helloworld-0.1.war`
   - **Use the `asadmin deploy` command from command line to deploy the WAR file. For example:**

```
asadmin deploy helloworld-0.1.war
```

Note – Depending on the configuration, you might be prompted for the `asadmin` password at this time.

---

4. **To test your application, open `http://`*host*`:`*port*`/`*war-file-name* in your browser.**

   Do not include the `.war` extension. For example:

   ```
   http://localhost:8080/helloworld-0.1
   ```

   You should see a screen that shows the following message:

   ```
   Welcome to Grails
   ```

   Clicking the `HelloController` link should change the display to the following message:

   ```
   Hello World!
   ```

See Also For details about the Administration Console, see the GlassFish Server online help.

For details about the `asadmin deploy` command, see the *Oracle GlassFish Server 3.0.1 Reference Manual*.

For details about the Grails commands, see the Grails Quick Start guide.

For general information about deployment, see the *Oracle GlassFish Server 3.0.1 Application Deployment Guide*.

◆ ◆ ◆ CHAPTER 3

Jython on Django

This chapter provides an overview of Jython and Django and how to get started using them on Oracle GlassFish Server.

The following topics are addressed here:

Overview

Jython is a Java implementation of the Python language. Jython is integrated with the Java platform and generates the code that runs on Java. Jython implements almost all modules of Python, except those written in C. Jython programs can import and use Java classes effortlessly. Jython provides the following advantages:

- Provides the advantages of easy and powerful Python syntax
- Allows the import of Java classes and extending those classes
- Provides the ability to compile programs to Java bytecode

To learn more about Jython, see the Jython Project pages.

Django is a web framework for Python and implementations of Python such as Jython. Django allows quick and easy creation of high-performance web applications. Django provides the following advantages:

- Provides an automatic administrative interface to web applications
- Provides an extensible and powerful templating system
- Allows building of data models that can access the databases quickly

To learn more about Django, see the Django Project pages.

You have the advantages of both Jython and Django when you build web applications using Jython on Django for the GlassFish Server.

Installing Jython and Django

Jython and Django installation on GlassFish Server comprises several substeps:

- "To Install Jython as Standalone" on page 44
- "To Install Jython from Update Tool" on page 45
- "To Install Django" on page 45
- "To Install the Jython Container for Oracle GlassFish Server" on page 46
- "To Install Jython Support Libraries for Django" on page 46

Jython can be installed in either of the following ways:

- As a standalone product
- From GlassFish Server Update Tool

The following sections explain these tasks in more detail.

▼ To Install Jython as Standalone

You can download Jython and install it as a standalone product. If Jython is installed as standalone, you need to perform additional configuration steps so GlassFish Server can use the Jython installation.

1. **Download Jython from the following location:**

   `http://downloads.sourceforge.net/project/jython/jython/2.5.1/jython_installer-2.5.1.jar`

2. **Run the installer as follows:**

   ```
   java -jar jython_installer-2.5.1.jar
   ```

3. **Set the following environmental variables:**

   Set the `JYTHON_HOME` variable to the Jython install location:

   `export JYTHON_HOME=`*jython-install-location*

   Add the `JYTHON_HOME`/bin directory to the path:

   ```
   export PATH=$JYTHON_HOME/bin:$PATH
   ```

   You should now be able to invoke Jython from command line as follows:

   ```
   jython
   ```

4 **Configure GlassFish Server to use the Jython installation with the following command:**

```
asadmin deploy --property jython.home=jython-install-location
```

▼ To Install Jython from Update Tool

The GlassFish Server Update Tool provides a Jython package. This package installs a Jython instance that enables creation of Jython applications for GlassFish Server. If you installed Jython from Update Tool, there is no need for further configuration of the `jython.home` property.

1 **Start Update Tool with the following command:**

```
as-install/bin/updatetool
```

2 **Choose the following option from Available Add-ons and click Install:**

```
Jython Runtime IPS package for GlassFish v3
```

Update center automatically completes the installation of the container and configures it for use with GlassFish Server.

3 **Set the following environmental variables:**

Set the `JYTHON_HOME` variable to the Jython install location:

```
export JYTHON_HOME=as-install/glassfish/jython
```

Add the `JYTHON_HOME`/bin directory to the path:

```
export PATH=$JYTHON_HOME/bin:$PATH
```

You should now be able to invoke Jython from command line as follows:

```
jython
```

▼ To Install Django

1 **Download Django from the following location:**

http://media.djangoproject.com/releases/1.1.1/Django-1.1.1.tar.gz.

2 **Extract the tar file:**

```
gunzip Django-1.1.1.tar.gz
tar -xvf Django-1.1.1.tar
```

3 **Change to the extracted directory:**

```
cd Django-1.1.1
```

4 **Install Django with the following command:**

```
jython setup.py install
```

▼ To Install the Jython Container for Oracle GlassFish Server

The GlassFish Server Update Tool provides a Jython container package. The following procedure explains how to install the Jython Container module and Grizzly adapter JAR files in the *as-install*/`glassfish/modules` directory, and enable deployment of Jython/Django applications on GlassFish Server.

1 **Start Update Tool with the following command:**

```
as-install/bin/updatetool
```

2 **Choose the following option from Available Add-ons and click Install:**

```
GlassFish V3 Jython Container
```

Update Tool automatically completes the installation of the container and configures it for use with GlassFish Server.

3 **Start the GlassFish Server:**

```
as-install/bin/asadmin start-domain -v
```

4 **Test the configuration:**

a. **Change to the Django** `samples` **directory:**

```
cd Django-1.1.1/samples
```

b. **Deploy the example applications on the server:**

```
as-install/bin/asadmin deploy .
```

c. **Access the deployed example applications from a web browser:**

```
http://localhost:8080/examples
```

▼ To Install Jython Support Libraries for Django

The `django-jython` project created database backends and management commands for Django and Jython application development. With this task you can install the `django-jython` packages and enable database support for Django.

1. **Download the `django-jython` packages from the following location:**

   http://django-jython.googlecode.com/files/django-jython-1.0.0.tar.gz

2. **Extract the tar file:**

   ```
   gunzip django-jython-1.0.0.tar.gz
   tar -xvf django-jython-1.0.0.tar
   ```

3. **Change to the extracted directory:**

   ```
   cd django-jython-1.0.0
   ```

4. **Install the package:**

   ```
   jython setup.py install
   ```

Creating and Deploying a Simple Jython Application using Django

After completing the software installation procedures described in "Installing Jython and Django" on page 44, you are ready to create Jython applications using Django. This section explains how to create a simple application. GlassFish Server users can deploy Django applications with the directory deployment method.

To Create a Simple Django Project

Django comes with a built-in administration utility. You can enable it to make the process of creating projects easier.

1. **Use the following command to enable the Django administration utility:**

   ```
   alias django–admin-jy="jython jython-install/bin/django-admin.py"
   ```

2. **Change to Django install directory:**

   ```
   cd django-install
   ```

3. **Use the following command to create a project:**

   ```
   django-admin-jy startproject myproject
   ```

To Deploy a Django Application From the Command Line

To deploy a Django application from the command line using the `asadmin` command, do the following:

1. **Make sure `JYTHON_HOME` and `PATH` environmental variables are set, as described in "Installing Jython and Django" on page 44.**

2. **Change to the directory containing the project. For example:**

   ```
   cd /tools/jython/projects
   ```

3. **Use the following command to deploy the application:**

   ```
   as-install/bin/asadmin deploy myproject/
   ```

`asadmin` CLI for Jython

The `asadmin deploy` command enables you to set several deployment-specific properties for Jython applications. Table 3–1 lists these properties.

TABLE 3–1 Jython Deployment Properties

Property	Default Value	Possible Value	Description
`jython.home`	None	Path to a directory	Path to a Jython installation. Required.
`jython.mediaRoot`	None	Path to a directory	Optional parameter containing the path to the location, for server to serve the static files.
`jython.frameworkRoot`	None	Path to a directory	Optional parameter containing the path of the framework being used. Currently supports Django.
`jython.applicationType`	None	String representing application type such as Django	Optional parameter to specify the framework, including non-Django applications.

Use the following command syntax to set these properties:

```
asadmin deploy --property property=value
```

For example, you can set the `jython.frameworkRoot` property to the Django directory as follows:

```
asadmin deploy --property jython.frameworkRoot=django-install
```

These values are persistent in the `domain.xml` file.

Using the Django Administration Utility

"Installing Jython and Django" on page 44 explained Jython on Django installation and configuration for Oracle GlassFish Server. To use the Django administration utility for creating applications based on a database, you need a database and the JDBC drivers for Jython to connect to that database. The following steps briefly describe the tasks involved:

1. Install a database such as PostgreSQL.
2. Create a database instance.
3. Install the `django-jython` packages for the database connectors, if not already installed.
4. Edit the `settings.py` file and configure the database and administration utility.
5. Add the database drivers to the class path to allow Jython to access the database.
6. Synchronize the database with the following command:

   ```
   jython manage.py syncdb
   ```
7. Edit the `urls.py` file and uncomment the lines pertaining to the administration utility.
8. Make the stylesheets available to the Jython container:

```
asadmin deploy --property jython.mediaRoot=jython-install/Lib/site-packages/django/contrib/admin/
```

You can find more information on how to install and use databases with Django administration in the following tutorial:

http://weblogs.java.net/blog/vivekp/archive/2009/06/run_django_appl_1.html

Further Information

The following links contains more details on the information provided in this chapter:

- http://docs.djangoproject.com/en/dev/howto/jython/#howto-jython
- http://wiki.python.org/jython/DjangoOnJython

◆ ◆ ◆ CHAPTER 4

Scala and Lift

Scala is a general purpose programming language designed to express common programming patterns in a concise, elegant, and type-safe way. It smoothly integrates features of object-oriented and functional languages. It is also fully interoperable with Java. For details, see `http://www.scala-lang.org/`.

Lift is an expressive and elegant framework for writing web applications using Scala. Lift stresses the importance of security, maintainability, scalability, and performance, while allowing for high levels of developer productivity. For details, see `http://liftweb.net/`.

Using Scala and Lift

It is common practice to start a Lift web application using Maven. Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting, and documentation from a central information source. For details, see `http://maven.apache.org/`.

To create a new Lift project, use Maven interactively in one of the following three ways:

```
mvn archetype:generate -DarchetypeCatalog=http://scala-tools.org/
```

Or:

```
mvn org.apache.maven.plugins:maven-archetype-plugin:1.0-alpha-7:create \
 -DarchetypeGroupId=net.liftweb                               \
 -DarchetypeArtifactId=lift-archetype-blank                   \
 -DarchetypeVersion=0.7.1                                     \
 -DremoteRepositories=http://scala-tools.org/repo-releases   \
 -DgroupId=__my.liftapp__ -DartifactId=__liftapp__
```

Or:

```
mvn org.apache.maven.plugins:maven-archetype-plugin:1.0-alpha-7:create \
 -DarchetypeGroupId=net.liftweb                              \
 -DarchetypeArtifactId=lift-archetype-basic                  \
 -DarchetypeVersion=0.7.1                                    \
 -DremoteRepositories=http://scala-tools.org/repo-releases  \
 -DgroupId=__my.liftapp__ -DartifactId=__liftapp__
```

After coding your application, build the WAR file using the `mvn package` command. Then deploy the WAR file to the Oracle GlassFish Server as you would any other web application.

◆ ◆ ◆ CHAPTER 5

PHP

PHP is a popular scripting language that is used mainly for generating dynamic web pages. The PHP interpreter takes PHP code as input and produces web pages as output. It can be used in standalone mode, but is typically deployed on a server.

Enabling PHP on OracleGlassFish Server

To enable PHP, deploy the Quercus PHP interpreter to the GlassFish Server as a web module.

▼ To Deploy the Quercus PHP Interpreter to the GlassFish Server

1. **Download the Quercus PHP interpreter from `http://quercus.caucho.com/`.**

2. **Deploy the downloaded WAR file to the GlassFish Server.**

3. **To verify that your PHP engine is working, enter the following URL in your browser:**

   `http://localhost:8080/quercus-4.0.1/`

   This is the default PHP script that is available in the Quercus interpreter.

4. **Move your PHP application to a subdirectory of the Quercus directory.**

   The Quercus application directory is located at *domain-dir*`/applications/quercus-4.0.1/`. For example:

   as-install/*domain-dir*`/applications/quercus-4.0.1/myapp/`

5 **To verify your PHP application is working, access your application from a browser.**

For example, enter the following URL in your browser:

```
http://localhost:8080/quercus-4.0.1/myapp/
```

See Also For more PHP information, documentation, and examples, see the Quercus PHP interpreter.