

JBoss Tools 4.1 User Guide

**Information about
using the plug-ins
comprising JBoss Tools**

by Red Hat Documentation Team

Preface	vii
1. Document Conventions	vii
1.1. Typographic Conventions	vii
1.2. Pull-quote Conventions	viii
1.3. Notes and Warnings	ix
2. Getting Help and Giving Feedback	x
I. Tools for Every Project	1
1. JBoss Central and the JBoss Perspective	3
1.1. Overview of JBoss Central and the JBoss Perspective	3
1.1.1. About JBoss Central	3
1.1.2. About the JBoss Perspective	3
1.2. Features of JBoss Central and the JBoss Perspective	4
1.2.1. Features Overview	4
1.2.2. View JBoss Central	4
1.2.3. Access Project Wizards in JBoss Central	4
1.2.4. Access Information in JBoss Central	5
1.2.5. Install Software in JBoss Central	6
1.2.6. Open the JBoss Perspective	8
1.2.7. Manage the JBoss Perspective	8
1.2.8. View Cheat Sheets	8
1.3. Customizing JBoss Central and the JBoss Perspective	9
1.3.1. Customizing Overview	9
1.3.2. Change the Behavior of JBoss Central on IDE Start	9
1.3.3. Customize the Default Views, Menus and Toolbars of the JBoss Perspective	10
1.3.4. Use Project Examples when Working Offline	11
1.3.5. Install Software when Working Offline	12
1.3.6. Set Cheat Sheet Behavior	13
2. JBoss Server Tools	15
2.1. Overview of JBoss Server Tools	15
2.1.1. About JBoss Server Tools	15
2.1.2. About IDE Server Definitions	15
2.2. Features of JBoss Server Tools	16
2.2.1. Features Overview	16
2.2.2. Download a JBoss Community Application Server	16
2.2.3. Create a Default Local Server Definition with Runtime Detection	17
2.2.4. Create an Additional Server Runtime Environment	19
2.2.5. Define an Additional Local Server	20
2.2.6. Define a Remote Server	22
2.2.7. Manage Server Settings	25
2.3. Customizing JBoss Server Tools	30
2.3.1. Customizing Overview	30
2.3.2. Enable Runtime Detection on IDE Start	30
2.3.3. Set a Default Server	30

2.3.4. Default File Sets	31
2.3.5. Default Classpath Entries	31
3. Forge Tools	33
3.1. Overview of Forge Tools	33
3.1.1. About Forge	33
3.1.2. About Forge Tools	33
3.2. Features of Forge Tools	33
3.2.1. Features Overview	33
3.2.2. Manage the Forge Console	34
3.2.3. Manage the Forge Runtime Server	34
3.2.4. Navigate to Project Resources on the Forge Command Line	35
3.2.5. Background Actions Invoked by Forge Commands	35
3.2.6. Access a List of Forge Commands	36
3.2.7. Use Forge Wizards	36
3.3. Customizing Forge Tools	37
3.3.1. Customizing Overview	37
3.3.2. Customize the Forge Start	37
3.3.3. Manage Forge Runtime Servers	38
II. Tools for Creating Web Interfaces	39
4. LiveReload Tools	41
4.1. Overview of LiveReload Tools	41
4.1.1. About LiveReload	41
4.1.2. About LiveReload Tools	41
4.2. Features of LiveReload Tools	41
4.2.1. Features Overview	41
4.2.2. Create a LiveReload Server	42
4.2.3. Configure the LiveReload Server	43
4.2.4. View Resources in LiveReload-enabled Browsers	43
4.2.5. View Resources in LiveReload-enabled BrowserSim	45
III. Tools for Creating Mobile Applications	47
5. Mobile Web Tools	49
5.1. Overview of Mobile Web Tools	49
5.1.1. About Mobile Web Tools	49
5.2. Features of Mobile Web Tools	49
5.2.1. Features Overview	49
5.2.2. Create a Mobile Web Project	49
5.2.3. Use a HTML5 jQuery Mobile File Template	51
5.2.4. Access the jQuery Mobile Palette	52
5.2.5. Insert a jQuery Mobile Palette Widget into a HTML5 File	53
5.2.6. Get Assistance with jQuery Mobile Programming	53
5.2.7. View jQuery Mobile Pages in a Browser	54
5.3. Customizing Mobile Web Tools	54
5.3.1. Customizing Overview	54
5.3.2. Customize jQuery Mobile File Templates	54

6. BrowserSim	57
6.1. Overview of BrowserSim	57
6.1.1. About BrowserSim	57
6.1.2. System Requirements for BrowserSim	57
6.2. Features of BrowserSim	57
6.2.1. Features Overview	57
6.2.2. View a Web Application on BrowserSim	58
6.2.3. Manage Web Applications on BrowserSim	58
6.2.4. Change the Appearance of a Simulated Device	59
6.2.5. View a Web Page in Different Browsers and Simulated Devices	60
6.2.6. Generate a Screen Capture of a Simulated Device	60
6.2.7. Activate LiveReload for BrowserSim	60
6.2.8. View the Source of a Web Page	62
6.3. Customizing BrowserSim	62
6.3.1. Customizing Overview	62
6.3.2. Make BrowserSim the Default Browser	63
6.3.3. Add BrowserSim to the Global Toolbar	63
6.3.4. Set a Shortcut for the Run BrowserSim Action	64
6.3.5. Add or Modify Devices in BrowserSim	64
6.3.6. Change the Default Behavior when a Device does not Fit the Display	65
6.3.7. Change the Default LiveReload Port	65
6.3.8. Set the Location for Saved Screen Captures	66
6.3.9. Change the Default Settings for Weinre	66
7. Hybrid Mobile Tools and CordovaSim	67
7.1. Overview of Hybrid Mobile Tools and CordovaSim	67
7.1.1. About Apache Cordova	67
7.1.2. About Hybrid Mobile Tools	67
7.1.3. About CordovaSim	68
7.1.4. System Requirements for Hybrid Mobile Tools	69
7.1.5. System Requirements for CordovaSim	70
7.1.6. Install Hybrid Mobile Tools and CordovaSim	70
7.2. Features of Hybrid Mobile Tools and CordovaSim	71
7.2.1. Features Overview	71
7.2.2. Create a Hybrid Mobile Project	72
7.2.3. Enable Cordova Plug-ins for an Application	73
7.2.4. Manage Cordova Settings of a Hybrid Mobile Project	74
7.2.5. Run a Hybrid Mobile Application on Devices and Simulators	76
7.2.6. Manage Hybrid Mobile Project Run Configurations	78
7.2.7. Export a Hybrid Mobile Project	79
7.3. Customizing Hybrid Mobile Tools and CordovaSim	80
7.3.1. Customizing Overview	80
7.3.2. Set the Android SDK Location	80
IV. Tools for Deployment and Maintenance	81
8. OpenShift Tools	83

8.1. Overview of OpenShift Tools	83
8.1.1. About OpenShift	83
8.1.2. About OpenShift Tools	83
8.2. Features of OpenShift Tools	84
8.2.1. Features Overview	84
8.2.2. Create an OpenShift Online User Account	84
8.2.3. Connect to OpenShift	85
8.2.4. Manage a Connection	86
8.2.5. Generate and Upload SSH Keys to OpenShift	87
8.2.6. Manage SSH Keys	88
8.2.7. Create a Domain	88
8.2.8. Manage a Domain	89
8.2.9. Deploy a New or Existing Application on OpenShift	90
8.2.10. Import a Deployed OpenShift Application into the IDE	94
8.2.11. Generate a Server Adapter for an Application	96
8.2.12. View a Deployed Application and Associated Information	97
8.2.13. Manage a Deployed Application	98
8.3. Customizing OpenShift Tools	100
8.3.1. Customizing Overview	100
8.3.2. Change the Timeout Behavior of OpenShift Requests	100
A. Revision History	103

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the [Liberation Fonts](https://fedorahosted.org/liberation-fonts/) [https://fedorahosted.org/liberation-fonts/] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later include the Liberation Fonts set by default.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

Mono-spaced Bold

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

To see the contents of the file `my_next_bestselling_novel` in your current working directory, enter the `cat my_next_bestselling_novel` command at the shell prompt and press **Enter** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

Press **Enter** to execute the command.

Press **Ctrl+Alt+F2** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in `mono-spaced bold`. For example:

File-related classes include `filesystem` for file systems, `file` for files, and `dir` for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).

To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find...** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

Mono-spaced Bold Italic Of Proportional Bold Italic

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

To connect to a remote machine using ssh, type `ssh username@domain.name` at a shell prompt. If the remote machine is `example.com` and your username on that machine is john, type `ssh john@example.com`.

The `mount -o remount file-system` command remounts the named file system. For example, to remount the `/home` file system, the command is `mount -o remount /home`.

To see the version of a currently installed package, use the `rpm -q package` command. It will return a result as follows: `package-version-release`.

Note the words in bold italics above: `username`, `domain.name`, `file-system`, `package`, `version` and `release`. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in `mono-spaced roman` and presented thus:

books	Desktop	documentation	drafts	mss	photos	stuff	svn
books_tests	Desktop1	downloads	images	notes	scripts	svgs	

Source-code listings are also set in mono-spaced roman but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
    public static void main(String args[])
        throws Exception
    {
        InitialContext iniCtx = new InitialContext();
        Object          ref    = iniCtx.lookup("EchoBean");
        EchoHome        home   = (EchoHome) ref;
        Echo            echo    = home.create();

        System.out.println("Created Echo");

        System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
    }
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.



Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.



Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.



Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

Do You Need Help? If you experience difficulty with a procedure described in this documentation, visit the JBoss Tools website at <http://www.jboss.org/tools>. Through the JBoss Tools website, you can:

- search or browse through a knowledgebase of technical support articles in the wiki.
- view video tutorials on how to use the tools.
- participate in discussions in the user forum.
- keep up to date with changes via the blog, mailing list and News and Noteworthy web page.

Give us Feedback. If you find a typographical error or have a suggestion for improving this documentation, we would love to hear from you. Submit a report in JIRA against the project **Documentation for JBoss Tools and Developer Studio (TOOLSDOC)**. The following link will take you to a pre-filled bug report for this product: <https://issues.jboss.org/secure/CreateIssueDetails!init.jspx?issuetype=1&pid=12310980>.

Fill out the following template in JIRA's **Description** field. Be as specific as possible when describing the issue; this will help ensure that we can fix it quickly.

Document URL:

Section Number and Name:

Describe the issue:

Suggestions for improvement:

Additional information:

Be sure to give us your name so that you can receive full credit for reporting the issue.

Part I. Tools for Every Project

JBoss Central and the JBoss Perspective

1.1. Overview of JBoss Central and the JBoss Perspective

1.1.1. About JBoss Central

JBoss Central is a core IDE view, providing access to JBoss information and assistance in one centralized location. It offers features for first time users of JBoss Tools, as well as those with more experience.

JBoss Central comprises resources and actions for creating new projects, learning about the tools and installing software updates:

- Project wizards create new projects based on sample applications and different technologies.
- The TicketMonster tutorial, part of JBoss Java Developer Framework, demonstrates how to use the IDE to best advantage in developing a complex web application that utilizes JBoss technologies.
- Links to JBoss blog posts give timely insight into JBoss developments and the links to JBoss resources, such as documentation and forums, provide easy access to key information sources.
- Software installation and update information assist in the easy management of IDE plug-ins.

1.1.2. About the JBoss Perspective

JBoss is a key IDE perspective, giving easy access to useful views and actions when developing applications with JBoss technologies. The perspective consists of a default set of views, menus and toolbars.

Views

The default views of the JBoss perspective are JBoss Central, OpenShift Explorer, Outline, Package Explorer, Palette, Problems, Project Explorer, Properties, and Servers. These views are vital when developing with JBoss technologies.

Menus

The default menus of the JBoss perspective are File, Edit, Navigate, Search, Project, Run, Window, Help. These are shared with other perspectives but contain unique menu items for actions such as starting project wizards, converting line delimiters, showing items in JBoss

perspective views, generating Javadoc, building packages, accessing JBoss Central and cheat sheets, and reporting problems with tools.

Toolbars

The default toolbars that compose the global toolbar of the JBoss perspective are BrowserSim, Debug, File, Help, Java EE, Java Element Creation, JBoss Tools WTP Server Actions, Launch, Navigate, Search, Select Maven Profiles, and Web Browser. The global toolbar provides access to frequently used actions.

1.2. Features of JBoss Central and the JBoss Perspective

1.2.1. Features Overview

The aim of this section is to guide you in using JBoss Central and the JBoss perspective:

- View JBoss Central to access project wizards and information resources
- Install and update software with JBoss Central
- Open the JBoss perspective for easy access to key views, menus and toolbars
- Access cheat sheets that assist in project development

1.2.2. View JBoss Central

JBoss Central provides resources for getting started with application development and installing auxiliary IDE software.

To open JBoss Central, click the **JBoss Central** icon

Alternatively, click **Help** → **JBoss Central**.

To maximize JBoss Central so that it fills the IDE window, double-click the **JBoss Central** view label. To reduce JBoss Central to its smaller size, double-click the **JBoss Central** view label again. These maximizing and minimizing actions work for all IDE views.

1.2.3. Access Project Wizards in JBoss Central

JBoss Central provides access to wizards for generating projects. There are two types of project wizards available in JBoss Central: **Start from scratch** and **Start from a sample**.

Start from scratch

These wizards create the same project using different underlying technology, as stated in the project name. To view a project summary, hover the cursor over a project wizard. If you do

not have the necessary plug-in installed to use a wizard, you are prompted to install it when you click the wizard.

Figure 1.1. Start from Scratch Wizards

Start from a sample

These wizards create sample web, mobile, back-end and portal applications. To view a project summary, hover the cursor over a project wizard.

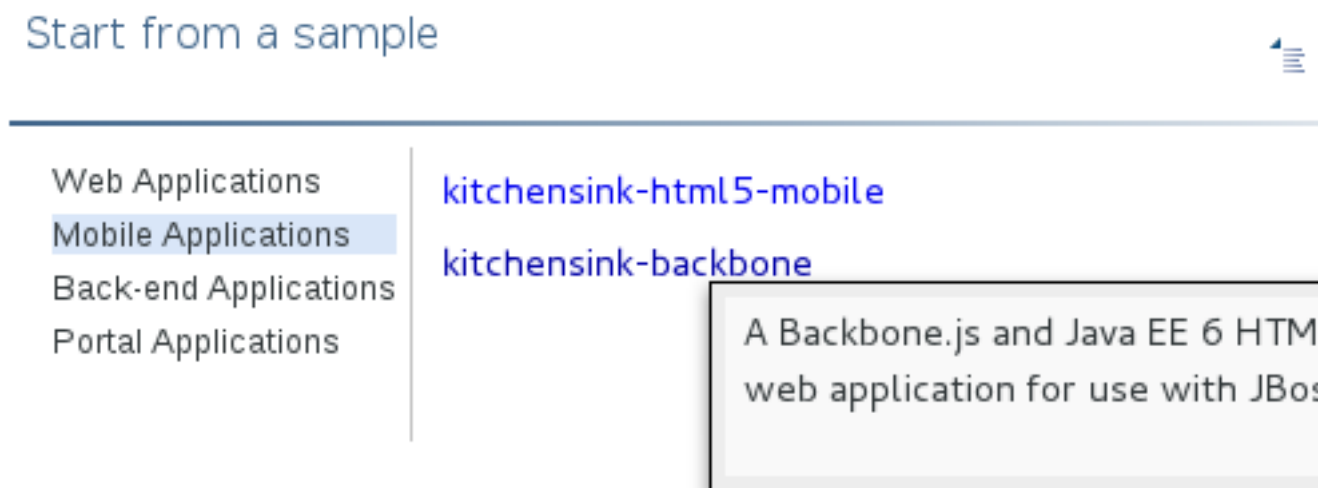


Figure 1.2. Start from Sample Wizards

To create a project from one of the wizards, in JBoss Central select the **Getting Started** tab. Click one of the links below **Start from scratch** or **Start from a sample**. Alternatively, click **File** → **New** and select a project from the list. A wizard opens to guide you through the process of creating the project.

Additionally, access is provided to the comprehensive TicketMonster tutorial, part of JBoss Developer Framework. The tutorial guides you through developing a complex web application utilizing JBoss technologies from within the IDE. To view the tutorial, in JBoss Central select the **Getting Started** tab and click the **TicketMonster** link. The JBoss Developer Framework website opens in the default IDE browser.

Figure 1.3. Access the TicketMonster Tutorial

1.2.4. Access Information in JBoss Central

JBoss Central provides easy access to JBoss and developer information. Each of the information sources listed below are web-based and, depending on the IDE default browser setting, clicking on links results in an internal or external browser window opening.

View the JBoss Tools website

In JBoss Central, click the **JBoss Tools Home** icon

View the latest JBoss developments

In JBoss Central, select the **Getting Started** tab and click the links under **JBoss Buzz**. To see previews of the blog posts, hover the cursor over these links. To view a complete list of JBoss blog posts, click the **JBoss Buzz** button

JBoss blog posts are available to follow as RSS feeds or with Twitter. To review these options, click the **News** or **Twitter** icons.

Figure 1.4. JBoss Buzz

View JBoss developer information sources

In JBoss Central, select the **Getting Started** tab and click the links under **Other resources**. These links provide access to videos, documentation and forums.

Figure 1.5. Other Resources

Search the JBoss Community website

In JBoss Central, click the arrow next to the search box and select **Search JBoss Community**. In the search field, enter the search terms.

Figure 1.6. Search the JBoss Community Website



Note

To change the default IDE browser, click **Window** → **Web Browser** and select a browser from the listed options.

1.2.5. Install Software in JBoss Central

JBoss Central enables you to install and update a range of IDE plug-ins. The available plug-ins comprise JBoss and third-party plug-ins that have been specifically tested for use with the IDE. These plug-ins include ones for mobile and web development, source control management, utilities and Maven.

For all actions listed below, open JBoss Central and select the **Software/Update** tab.

View available software

The available software is listed in the table. To refresh the list of available plug-ins, click the **Refresh** icon

View installed plug-ins

Select the **Show Installed** check box. The installed plug-ins are listed in the table as disabled.

Figure 1.7. Show Installed Check Box Selected and Installed Plug-ins Listed as Disabled

Install available software

In the **Find** field, type the name of the software or scroll through the list to locate it. Select the check box corresponding to the software you want to install and click **Install** or click the **Install** icon

Figure 1.8. Find and Install Software

In the **Install** wizard, ensure the check boxes are selected for the software you want to install and click **Next**.

Figure 1.9. Install Window

Review the details of the items listed for installing and click **Next**. After reading and agreeing to the license(s), click **I accept the terms of the license agreement(s)** and click **Finish**. The **Installing Software** window opens and reports the installation progress.

During the installation process you may receive warnings about installing unsigned content. If this is the case, check the details of the content and if satisfied click **OK** to continue with the installation.

Figure 1.10. Warning Prompt for Installing Unsigned Content

Once installing is complete, you are prompted to restart the IDE. Click **Yes** to restart now and **No** if you need to save any unsaved changes to open projects. Note that changes do not take effect until the IDE is restarted.

Check for software updates

Click the **Check for Updates** icon

The **Contacting Software Sites** window opens and reports the progress of checking. Once checking is complete, a prompt informs you of any new software found. Click **OK** to close the prompt.

Figure 1.11. Information Window Showing No Updates Found

1.2.6. Open the JBoss Perspective

The JBoss perspective provides a default set of views, menus and toolbars to assist with common tasks associated with developing applications that use JBoss technologies.

To open the JBoss perspective, click **Window**→**Open Perspective**→**Other**. From the list of available perspectives, select **JBoss** and click **OK**. The views associated with the JBoss perspective open and menus and toolbars change as appropriate.

Figure 1.12. Select JBoss in the Open Perspective Window

1.2.7. Manage the JBoss Perspective

There are a number of actions provided by the IDE for managing perspectives, including the JBoss perspective.

Reset the JBoss perspective

Click **Window**→**Reset Perspective**. At the prompt asking if you want to reset the current JBoss perspective to its default settings, click **Yes**. This action resets the views to their original size and position and reverts the contents of menus and toolbars to their original listings.

Switch to the JBoss perspective

Click the **JBoss** icon

Alternatively, to view a list of open perspectives, click **Window**→**Navigation**→**Next Perspective** or press and hold **Ctrl** and press **F8**. From the list of perspectives, select the JBoss perspective or press **F8** repeatedly until the JBoss perspective is selected. This action is useful if you are using multiple perspectives.

Close the JBoss perspective

Click **Window**→**Close Perspective**.

1.2.8. View Cheat Sheets

Typically, cheat sheets contain detailed information about projects, with step by step guidance and explanations for how to create and deploy applications. JBoss Central and the JBoss perspective provide actions to make the cheat sheets that accompany projects easier to access and view.

A cheat sheet contained in a project is automatically opened in the **Cheat Sheets** view when the project is imported into the workspace with **File**→**Import**.

Figure 1.13. Cheat Sheet Open in Cheat Sheets View

To open a cheat sheet manually, in the **Project Explorer** view right-click the project name or a cheat sheet file and click **Open In Cheat Sheets View**. Alternatively, click **Help**→**Cheat Sheets**, click **Select a cheat sheet from a file** and type the location of the file in the field or click **Browse** to navigate to the file. Click **OK** to close the window. The cheat sheet opens in the **Cheat Sheets** view.



Note

If the cheat sheet file name begins with dot, it may not be automatically visible in the **Project Explorer** view. To change the viewing preferences of the **Project Explorer** view, click the **View Menu** icon and click **Customize View**. In the **Filters** tab, clear the **. * resources** check box and click **OK**.

1.3. Customizing JBoss Central and the JBoss Perspective

1.3.1. Customizing Overview

The aim of this section is to guide you in customizing JBoss Central and the JBoss perspective:

- Make JBoss Central visible when the IDE starts
- Customize the views, menus and toolbars of the JBoss perspective
- Enable offline availability of JBoss Central elements, such as project examples and auxiliary plug-ins
- Specify the default IDE behavior for cheat sheets

1.3.2. Change the Behavior of JBoss Central on IDE Start

JBoss Central is set by default to show when the IDE starts but you can customize this behavior.

To change the behavior, in JBoss Central select or clear the **Show on Startup** check box as appropriate.

Figure 1.14. Show on Startup Check Box

Alternatively, in JBoss Central click the **Preferences** icon

or click **Window→Preferences**. In both cases, expand **JBoss Tools** and select **JBoss Central**. Select or clear the **Show JBoss Central on Startup** check box as appropriate. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 1.15. Set JBoss Central Behavior in JBoss Central Pane of Preferences Window

1.3.3. Customize the Default Views, Menus and Toolbars of the JBoss Perspective

You can customize the views, menus and toolbars displayed by the JBoss perspective, as with any perspective.

For all actions listed below, ensure the JBoss perspective is the current perspective.

Customize views

Open or close views as desired and click **Window→Save Perspective As**. From the **Existing Perspectives** list, select **JBoss** and click **OK**. At the prompt asking if you want to overwrite the existing JBoss perspective, click **Yes**.

Figure 1.16. Save Perspective As Window

Customize menu and toolbar elements

Click **Window→Customize Perspective**. In the **Tool Bar Visibility** tab, select or clear the check boxes corresponding to the individual toolbars and icons visible in the global toolbar of the JBoss perspective as appropriate. In the **Menu Visibility** tab, select or clear the check boxes corresponding to the menus and menu items visible in the JBoss perspective as appropriate. Disabled items in the **Tool Bar Visibility** and **Menu Visibility** tabs can be activated in the **Command Groups Availability** tab. Click **OK** to save any changes and close the window.

Figure 1.17. Tool Bar Visibility Tab in Customize Perspective Window

Figure 1.18. Menu Visibility Tab in Customize Perspective Window

Reorder the individual toolbars comprising the global toolbar

Click the vertical dashed line indicating the beginning of an individual toolbar and drag the toolbar to its new location in the global toolbar.

Figure 1.19. Select and Drag a Toolbar to Reorder the Global Toolbar

Revert all perspective customizations

Click **Window** → **Reset Perspective**. At the prompt asking if you want to reset the JBoss perspective to its saved state, select the **Also discard perspective's customization** check box and click **Yes**.

Figure 1.20. Revert Perspective Window

1.3.4. Use Project Examples when Working Offline

When a project wizard in JBoss Central is used, the IDE searches online repositories for the most recent versions of project dependencies. JBoss Central enables you to create a cache of the necessary project dependencies when you are online so that you can still use the project wizards offline. As detailed below, this is achieved by first creating the cache and then informing the IDE to use that cache.



Important

Groovy must be installed and configured as stated in the Groovy documentation. For more information, see <http://groovy.codehaus.org/Installing+Groovy> at the Groovy website.



Important

Maven must be correctly configured for all of the project wizards before the cache script is run. The JBoss Public Maven repository must be specified in `settings.xml`.

To create the cache, click **Window** → **Preferences**. Expand **JBoss Tools** → **Project Examples** and select **Offline Support**. To run the cache generating script, click **Copy to Clipboard** and paste the selected text on a command line outside the IDE. The script downloads and builds all of the project examples in a new directory `offline`. The script may take some time to complete.

Once the cache is generated, copy `offline/.jbosstools/cache` to the directory where you want to keep the project examples cache. Copy the contents of `offline/.m2/repository` to your local maven repository.

To inform the IDE to use the generated project examples cache, click **Window** → **Preferences**. Expand **JBoss Tools** → **Project Examples** and select **Offline Support**. Select the **Enable offline**

mode for project examples check box. Ensure the correct cache location is specified in the **Offline directory** field. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 1.21. Offline Mode Enabled for JBoss Central Project Examples

1.3.5. Install Software when Working Offline

You can install the JBoss and third-party plug-ins listed in JBoss Central in offline mode using the JBoss Central `.zip` file, as detailed below. This feature is useful if you regularly work offline or if you are installing these auxiliary plug-ins in a number of JBoss Developer Studio installations as it removes the need to repeatedly download the same plug-ins.

Note that the JBoss Central `.zip` file is a snapshot of the plug-ins at the time of the given JBoss Developer Studio release and updated versions of the plug-ins must be installed from JBoss Central in online mode.

Procedure 1.1. Install from JBoss Central `.zip` File



Note

To install JBoss and third-party plug-ins in offline mode, you must first download the JBoss Central `.zip` file. You can download the JBoss Central `.zip` file from <http://download.jboss.org/jbosstools/targetplatforms/jbtcentraltarget/> on the JBoss Tools website.

1. Click **Help**→**Install New Software**.
2. In the **Work with** field, enter the path of the JBoss Central `.zip` file. Alternatively, click **Add** and click **Archive** to locate the file.
3. Clear the **Group items by category** check box. This action makes the contents of the JBoss Central `.zip` file visible in the table of components.
4. From the table of components, select the software to be installed and click **Next**.
5. Review the details of the items listed for install and click **Next**.
6. After reading and agreeing to the license(s), click **I accept the terms of the license agreement(s)** and click **Finish**. The **Installing Software** window opens and reports the progress of the installation.
7. During the installation process you may receive warnings about installing unsigned content. If this is the case, review the details of the content and if satisfied click **OK** to continue with the installation.

8. Once installing is complete, you are prompted to restart the IDE. Click **Yes** to restart now and **No** if you need to save any unsaved changes to open projects. Note that changes do not take effect until the IDE is restarted.

1.3.6. Set Cheat Sheet Behavior

You can specify the default action the IDE is to take when finding cheat sheets in projects that it is importing. The available options include always or never showing cheat sheets or the IDE prompting for which action to take for each import.

To customize the action, click **Window** → **Preferences**. Expand **JBoss Tools** and select **Project Examples**. From the **Show included cheat sheet(s) when importing a project** list, click one of the options. Click **Apply** and click **OK** to close the window.

Figure 1.22. Set Cheat Sheet Behavior in Project Examples Pane of Preferences Window

JBoss Server Tools

2.1. Overview of JBoss Server Tools

2.1.1. About JBoss Server Tools

JBoss Server Tools is tooling for working with application servers in the IDE. It extends and enhances the existing server functionality of the IDE.

JBoss Server Tools consists of wizards, editors and actions that enable you to define, configure and manage application servers.

Runtime detection is available to assist you in locating installed application servers and integrating them into the IDE ready for use. Additionally, you can define remote servers for JBoss community application servers in the enhanced New Server wizard, which integrates with the IDE Remote System Explorer (RSE) tool. JBoss Server Tools also allows you to download JBoss community application servers from within the IDE.

Extensions to the standard IDE New Server wizard and Server Editor allow you to specify and customize more management and behavior settings for servers. Settings include IDE interaction with servers such as launching, communicating and publishing. The JBoss Server Editor supports you to manage server settings for global and individual application deployment.

JBoss Server Tools actions enable you to make customizations based on application server type, such as default filesets added to new servers and classpath filesets. You can also customize the default server used for IDE actions such as 'Run on Server'.

2.1.2. About IDE Server Definitions

A server definition is a description of how the IDE is to use an application server. It informs the IDE of configuration information with which to start, communicate and manage an application server, including application deployment. As such, a server definition is an essential element for working with an application server in the IDE.

A complete server definition is composed of two items, as listed here.

Server runtime environment

This informs the IDE about available local application servers. Each server runtime environment defines a specific application server configuration in terms of the application server itself, a configuration file and a Java developer kit. By varying the configuration details, one application server can be associated with several server runtime environments.

Server adapter

This informs the IDE about management settings for server runtime environments. Each server adapter, also referred to as simply a server, completes the definition for a specific server by detailing settings such as access parameters, launch arguments, and publishing options.

By varying the configuration details, one server runtime environment can have several server adapters associated with it.

You must generate at least one complete server definition for an application server before you can use it in the IDE. To assist you with this, JBoss Server Tools provides runtime detection that automatically generates a complete default server definition for any installed application servers found in a given local system search path.

See Also:

- [Section 2.2.3, “Create a Default Local Server Definition with Runtime Detection”](#)

2.2. Features of JBoss Server Tools

2.2.1. Features Overview

The aim of this section is to guide you in using JBoss Server Tools:

- Download JBoss community application servers from with the IDE
- Use runtime detection to locate installed application servers and generate complete server definitions, making them ready for IDE use
- Define custom configured servers, both local and remote, by creating additional runtime environments and server adapters
- Manage server configuration and specify IDE-server interaction with the JBoss Server Editor, including settings such as launch options, publishing frequency and communication ports

2.2.2. Download a JBoss Community Application Server

Application servers must be installed on your system in order to deploy applications to local and remote instances of them. Many application servers must be downloaded from outside the IDE. But JBoss Server Tools can assist you to download JBoss community application servers from within the IDE, as detailed in the procedure below.

Procedure 2.1. Download a JBoss Community Application Server

1. Click **Window**→**Preferences**, expand **JBoss Tools** and select **JBoss Runtime Detection**.

Figure 2.1. JBoss Runtime Detection Pane

2. Click **Download**.
3. From the table, select an application server and click **Next**.

Figure 2.2. Application Server Options in Download Runtimes Wizard

4. After reading and agreeing to the terms of the license, click **I accept the terms of this license agreement** and click **Next**.
5. In the **Install folder** field, type the path in which the downloaded application server should be installed or click **Browse** to navigate to the location.

Figure 2.3. Location Options in Download Runtimes Wizard

6. In the **Download folder** field, type the path to which the application server should be downloaded or click **Browse** to navigate to the location.
7. To automatically remove the downloaded archive after installing, select the **Delete archive after installing** check box.
8. Click **Finish** to commence downloading and installing. The IDE progress bar and **Download** window show the progress of the download process. You can click **Run in Background** to close the download window and continue the download process.
9. Click **OK** to close the **Preferences** window.

Once downloading is complete, the application server archive is extracted and several automated actions occur:

- The path of the application server is added to the runtime detection **Paths** table.
- A default server runtime environment is generated for the application server.
- A default server adapter is created for the server runtime environment.



Note

Alternatively, you can download and install JBoss community application servers when adding server runtime environments within **Preferences** under **Server** → **Runtime Environments**, with the wizard for creating new servers, or with the **Start from scratch** and **Start from a sample** wizards in JBoss Central.

2.2.3. Create a Default Local Server Definition with Runtime Detection

Before the IDE can use an application sever, you must create a server definition for it. JBoss Server Tools provides runtime detection that automatically generates a complete default server definition

for any installed application servers found in a given local system search path, as detailed in the procedure below. Note that complete server definitions are also automatically generated for JBoss community servers downloaded with JBoss Server Tools.

Procedure 2.2. Define a Local Server with Runtime Detection

1. Click **Window**→**Preferences**, expand **JBoss Tools** and select **JBoss Runtime Detection**.

Figure 2.4. JBoss Runtime Detection Pane of Preferences Window

2. Click **Add**.
3. Select a path from which recursive scanning for application servers is to commence. To detect a specific application server, select the install directory for that application server. To detect multiple application servers, select a directory higher up the directory tree.
4. Depending on the outcome of the scan, follow the appropriate step:
 - If no new application servers are found or if new application servers are found but you do not want to create any server runtime environments for them, click **Cancel**.
 - If new application servers are found and you want to generate server runtime environments for them, select the check boxes of the appropriate application servers and click **OK**.

Figure 2.5. Application Servers Found by Runtime Detection

In all cases, the path is added to the **Paths** table.

5. Click **Apply** and click **OK** to close the **Preferences** window.

Generated server runtime environments are listed in **Preferences** under **Server**→**Runtime Environments**. A default server adapter is automatically created for each generated server runtime environment to complete the server definition. Server adapters are listed in the **Servers** view.

Figure 2.6. Server Adapters Listed in the Servers View



Note

Alternatively, you can manually define servers by using the IDE server functions to create a server runtime environment and then to create a server adapter.

2.2.4. Create an Additional Server Runtime Environment

Runtime detection creates a server runtime environment as part of the default server definition for each application server it discovers in given search paths on your system. But you may want to create additional server runtime environments that specify a variety of JREs or configuration files for a given installed application server. The procedure below details the process for creating new server runtime environments. For older versions of application servers, you can also create a new runtime server by cloning an existing one as explained below.

Procedure 2.3. Create a Server Runtime Environment

1. Click **Window** → **Preferences**, expand **Server** and select **Runtime Environments**.

Figure 2.7. Runtime Environments Pane

2. Click **Add**.
3. Complete the fields and options as detailed:
 - From the **Select the type of runtime environment** list, select a JBoss community application server.
 - To create a complete local server definition, select the **Create a new local server** check box.

Figure 2.8. Application Server Options in the New Server Runtime Environment Wizard

4. Click **Next**.
5. Complete the fields and options as detailed:
 - In the **Name** field, type a name by which to identify the server runtime environment within the IDE.
 - In the **Home Directory** field, type the path of the installed application server or click **Browse** to navigate to the location. Alternatively, to use a JBoss community application server that is not already installed on the system, click **Download and install runtime** and follow the instructions.
 - From the **JRE** list, select the JRE to use with the application server.

- In the **Configuration file** field, type the path of the application server configuration file or click **Browse** to navigate to the location. Note that the path of the application server configuration file is relative to `Home Directory/standalone/configuration/`, where `Home Directory` is specified in the **Home Directory** field.

Figure 2.9. Server Runtime Environment Options in the New Server Runtime Environment Wizard



Note

For older application servers, the **Configuration file** field is replaced with the **Directory** field. In this field, type the path where the application server configurations are installed and then select a listed configuration. To clone from an existing server runtime environment, after selecting a listed configuration click **Copy**. Complete the name for the new configuration and the location where the configuration should be stored and click **OK**.

6. Click **Next** if the button is enabled, otherwise click **Finish**. The **Next** button is only enabled if you selected the **Create a new local server** check box earlier. On the presented page, complete the appropriate information and click **Finish**.

Figure 2.10. Server Adapter Behavior Options in the New Server Runtime Environment Wizard

The new server runtime environment is listed in the **Server runtime environments** table of the **Runtime Environments** pane in the **Preferences** window.



Note

Alternatively, you can create server runtime environments with the wizard for creating new servers in the **Servers** view or with the **Start from scratch** and **Start from a sample** wizards in **JBoss Central**.

2.2.5. Define an Additional Local Server

Runtime detection defines a local server as part of the default server definition for each application server it discovers in given search paths on your system. But you may want to create additional server adapters that have different configurations for a given server runtime environment in order

to define multiple servers. To create a new server adapter to define a local server, you must use the new server wizard as detailed in the procedure below.

Procedure 2.4. Define a Local Server

1. Click the **Servers** view. If the **Servers** view is not visible, click **Window**→**Show View**→**Servers**.
2. Depending on the number of existing servers, follow the appropriate step:
 - If there are no existing servers, click **Click this link to create a new server**.
 - If there are one or more existing servers, right-click an existing server and click **New**→**Server**.
3. Complete the fields and options as detailed:
 - From the **Select the server type** list, select a JBoss community application server.
 - The **Server's host name** and **Server name** fields are completed by default. In the **Server name** field, you can type a custom name by which to identify the server in the **Servers** view.
 - From the **Server runtime environment** list, select an existing server runtime environment for the application server type. Alternatively, to create a new runtime environment click **Add** and complete the fields and options as appropriate.

Figure 2.11. Server Runtime Environment Options in the New Server Runtime Environment Wizard



Note

If the **Server runtime environment** field is not shown, no server runtime environments exist for the selected application server type. A server runtime environment must be selected before you can successfully create a server adapter and complete the server definition. To create a new server runtime environment without canceling the wizard, click **Next** and complete the fields and options as appropriate.

4. Click **Next**.
5. The server behavior options displayed vary depending on the selected application server type. Complete the fields and options as detailed:

- To specify that the server life cycle will be managed from outside the IDE, select the **Server is externally managed** check box.
- To specify that the server should be launched to respond to requests on all hostnames, select the **Listen on all interfaces to allow remote web connections** check box. This option adds the `-b 0.0.0.0` argument to the server launch command.
- From the location list, select **Local**.



Note

The **Expose your management port as the server's hostname** option, which enables management commands sent by the IDE to be successfully received by the server, is bypassed for local servers regardless of whether the check box is selected.

Figure 2.12. Server Adapter Behavior Options in the New Server Wizard

6. Click **Next**.
7. To select applications to deploy with this server, from the **Available** list select the applications and click **Add**. Applications to be deployed are detailed in the **Configured** list.

Figure 2.13. Add or Remove Server Resources in the New Server Wizard

8. Click **Finish** to create the server. The server is listed in the **Servers** view, with the information in brackets detailing the server status.



Important

You can create multiple servers that use the same application server. But a warning is displayed if you try to simultaneously run more than one server on the same host. This is because multiple running servers on the same host can result in port conflicts.

2.2.6. Define a Remote Server

You can define remote servers for JBoss community application servers. To complete a server definition, you must create a server adapter, or server, that informs the IDE how to communicate and manage the remote server, as detailed in the procedure below.



Important

A complete server definition requires a server runtime environment and a server adapter. Ideally the server runtime environment would be created by specifying the remote application server and remote Java developer kit but server runtime environments can only be created using local components. To work around this issue, you must have a version of the remote application server and remote Java developer kit installed locally and create a server runtime environment based on these.

Procedure 2.5. Define a Remote Server

1. Click the **Servers** view. If the **Servers** view is not visible, click **Window**→**Show View**→**Servers**.
2. Depending on the number of existing servers, follow the appropriate step:
 - If there are no existing servers, click **Click this link to create a new server**.
 - If there are one or more existing servers, right-click an existing server and click **New**→**Server**.
3. Complete the fields and options as detailed:
 - From the **Select the server type** list, select a JBoss community application server.
 - The **Server's host name** and **Server name** fields are completed by default. In the **Server name** field, you can type a custom name by which to identify the server in the **Servers** view.
 - From the **Server runtime environment** list, select an existing server runtime environment for the application server type. Alternatively, to create a new runtime environment click **Add** and complete the fields and options as appropriate.

Figure 2.14. Server Runtime Environment Options in the New Server Runtime Environment Wizard



Note

If the **Server runtime environment** field is not shown, no server runtime environments exist for the selected application server type. A server runtime environment must be selected before you can successfully create a server adapter and complete the server definition. To create a new server runtime

environment without canceling the wizard, click **Next** and complete the fields and options as appropriate.

4. Click **Next**.
5. The server behavior options displayed vary depending on the selected application server type. Complete the options as detailed:
 - To specify that the server life cycle will be managed from outside the IDE, select the **Server is externally managed** check box.
 - To specify that the server should be launched to respond to requests on all hostnames, select the **Listen on all interfaces to allow remote web connections** check box. This option adds the `-b 0.0.0.0` argument to the server launch command.
 - To enable management commands sent by the IDE to be successfully received by the server, select the **Expose your management port as the server's hostname** check box. This option is useful for remote servers.



Note

To make use of this facility, a management user must exist for the remote server and you must provide the management user credentials to the IDE.

- From the location list, select **Remote System Deployment**.

Figure 2.15. Remote System Deployment Options in the New Server Wizard

6. Complete the additional fields and options for the remote server as detailed:
 - From the **Host** list, select the host. Alternatively, to specify a new host, click **New Host** and follow the instructions.
 - In the **Remote Server Home** field, type the path of the application server or click **Browse** to navigate to the location.
 - In the **Remote Server Configuration File** field, type the path of the configuration file or click **Browse** to navigate to the location.
7. Click **Next**.
8. To select applications to deploy with this server, from the **Available** list select the applications and click **Add**. Applications to be deployed are detailed in the **Configured** list.

Figure 2.16. Add or Remove Server Resources in the New Server Wizard

9. Click **Finish** to create the server. The server is listed in the **Servers** view, with the information in brackets detailing the server status.

2.2.7. Manage Server Settings

JBoss Server Tools provides the JBoss Server Editor for managing the settings of servers. This editor has two tabs: Overview and Deployment. As described below, each tab enables you to configure fundamental server settings.

The **Overview** tab details the settings for the server. Within this tab you can provide management information, specify application publishing and reload behavior, and customize port settings.

Figure 2.17. Overview Tab of the JBoss Server Editor

The **Deployment** tab lists applications deployed to the server. Within this tab you can specify the general publishing behavior for applications and provide deployment settings for individual applications.

Figure 2.18. Deployment Tab of the JBoss Server Editor

To open the JBoss Server Editor for a specific server, in the **Servers** view double-click the server. All changes to the settings of a server must be saved before the results will take effect. To save changes made to server settings in the JBoss Server Editor, press **Ctrl+S**. You may be required to enter the server management password when making changes to certain settings.

2.2.7.1. Manage Server Settings in the Overview Tab

The Overview tab of the JBoss Server Editor enables you to vary the management and behavior settings of an individual server. Each section of the Overview tab is outlined below. All changes to server settings must be saved before the results will take effect. To save, press **Ctrl+S**.

General information

This section details essential information comprising the server definition: the name by which the server is identified in the IDE, the hostname of the server and the server runtime environment.

Figure 2.19. General Information Section

To change the server runtime environment, from the **Runtime Environment** list select a server runtime environment. Alternatively, to create and assign a new server runtime environment click **Runtime Environment** and follow the instructions.

To view or edit the server launch configuration, click **Open launch configuration**.

Management login credentials

This section holds credentials, specifically username and password, necessary for the IDE to successfully communicate management commands with the server. The password is obscured and stored in Eclipse Secure Storage for security. Incorrect management credentials can cause the IDE to not detect when a server is started.

Figure 2.20. Management Login Credentials Section

Server behavior

This section enables you to customize server behavior that encompasses how the IDE communicates with the server.

Figure 2.21. Server Behavior Section

To specify that the server life cycle will be managed from outside the IDE, select the **Server is externally managed** check box.

To specify that the server should be launched to respond to requests on all hostnames, select the **Listen on all interfaces to allow remote web connections** check box. This option is most useful for remote servers and adds the `-b 0.0.0.0` argument to the server launch command.

To enable management commands sent by the IDE to be successfully received by the server, select the **Expose your management port as the server's hostname** check box. This option is useful for remote servers and unnecessary for local servers.



Warning

The **Expose your management port as the server's hostname** feature should be used carefully for servers on production as it leaves the server open for anyone to access.

To specify the location of the server, from the list select **Local** or **Remote System Deployment**. For remote systems, there are further details that must be specified: the host, the path of the remote application server, and the remote application server configuration file.

Figure 2.22. Additional Remote Server Behavior Options in the Server Behavior Section

Publishing

This section details the publishing action the IDE should take in response to modifications to local resources of deployed applications. Publishing involves replacing changed project resources in the dedicated deployment location of a server and the IDE action options are **Never publish automatically**, **Automatically publish when resources change**, and **Automatically publish after a build event**. Additionally, you can specify a minimum time interval that must occur between consecutive automated publish actions by the IDE to control the frequency of publishing.

Figure 2.23. Publishing Section

Timeouts

This section specifies the maximum length of time, in seconds, the IDE should wait for server actions to complete before aborting. The server actions are specifically starting and stopping.

Figure 2.24. Timeouts Section

Deployment scanner

This section enables you to customize the behavior of deployment scanners, which detect the applications deployed to a server. You can manage deployment scanners or allow the IDE to do it for you. The management options available are **Add missing deployment scanners after server startup** and **Remove added deployment scanners before shutdown**.

Figure 2.25. Deployment Scanners Section

Application reload behavior

This section details the application reload action the IDE should take in response to changed published resources of deployed applications. Application reload involves undeploying and redeploying an application and this action is necessary when you make changes to project resources that will not be detected by the server. By default, the application reload behavior is set to invoke application redeployment when `.jar` files are changed.

Figure 2.26. Application Reload Behavior Section

To customize which changes invoke application redeployment, select the **Customize application reload behavior on changes to project resources** check box. In the **Force module restart on following regex pattern** field, type a regex pattern indicating the changed resources that you want to trigger redeployment.

To disable application reload, select the **Customize application reload behavior on changes to project resources** check box and ensure the **Force module restart on following regex pattern** field is empty.

Server state detectors

This section specifies which method the IDE should use to verify the started and stopped status of the server.

Figure 2.27. Server State Detectors Section

There are four methods from which to choose:

- **Web Port**, which pings the web port on the host to see if the server responds
- **Timeout**, which waits for a specified time duration and then declares the start or stop operation a success without any actual verification
- **Process Terminated** (available for Shutdown Poller of local servers only), which checks if a server process is still alive and sets the server status to stopped when it is terminated
- **JMX**, which polls JMX, the JBoss Management service

Note that server state detection options are disabled if the **Server is externally managed** check box under **Server Behavior** is selected.

Server ports

This section details the ports and port offset that the IDE should use for communication with the server.

Figure 2.28. Server Ports Section

Port offset is typically offered by newer application servers and it enables multiple servers to run on the same system without port conflicts. JBoss Server Tools uses information in the server configuration file, typically XPath values, to automatically detect the correct ports and port offset for communicating with the server but you can perform further customization.

To view the configuration file information used by JBoss Server Tools for automatic port detection, click **Configure** corresponding to the appropriate tool. The information used is displayed in the **Current Value** field. To change this value, click **Edit XPath**. Click **OK** to close the window.

Figure 2.29. Edit Port Window

To manually specify the server ports or port offset, clear the **Detect from Local Runtime** check box corresponding to the appropriate tool and edit the port value.

2.2.7.2. Manage Server Settings in the Deployment Tab

The Deployment tab of the JBoss Server Editor enables you to vary the deployment settings of an individual server. Each section of the Deployment tab is outlined below. All changes to server settings must be saved before the results will take effect. To save, press **Ctrl+S**.



Important

Changing deployment settings when modules are already deployed can adversely result in multiple deployed copies of an application. For this reason, many of the functions of the Deployment tab are only enabled when a server is fully synchronized and it has no modules deployed.

Default settings for the server

This section specifies where deployments are kept and how they are packaged.

Figure 2.30. Default Settings Section

You can customize the deployment location and packaging type:

- To select the workspace deployment folder for the server, click **Use workspace metadata**.
- To select the deployment folder of the application server, click **Use the JBoss deploy folder**.
- To select a folder of your choice, click **Use a custom deploy folder**. With this option, complete the **Deploy directories** and **Temporary Deploy Directory** fields. The temporary folder must be on the same file system as the final deploy location otherwise publishing often fails.
- For all modules to be archived for deployment, select the **Deploy projects as compressed archives** check box. This avoids exploded deployments and reduces the amount of memory deployments occupy but may result in slower deployment.

Settings per module

This section shows deployment settings for all modules in the workspace regardless of whether they are deployed on the server under consideration.

Figure 2.31. Module Settings Section

To filter the module list in the case that you have many modules, from the **Filter by** list select the criteria for the filter. If you select the **By Module Name** filter option, in the text field enter the part or whole name of the module.

To change the **Deployment Location** and **Temporary Deploy Directory** on a per module basis, in the table click the value to be changed and enter an alternative value. Ensure the values for these variables are specified use absolute paths or paths relative to the default deploy directory.

2.3. Customizing JBoss Server Tools

2.3.1. Customizing Overview

The aim of this section is to guide you in customizing JBoss Server Tools:

- Enable runtime detection to search paths for application servers on IDE start
- Select a default server for IDE actions
- Specify default file sets that are listed in the **Servers** view for ease of access
- Customize classpath entries for your projects based on application server type

2.3.2. Enable Runtime Detection on IDE Start

You can customize runtime detection to automatically search paths for installed application servers when the IDE starts. If any application servers are found, you are prompted about creating corresponding complete server definitions.

To enable automated searching on IDE start, click **Window** → **Preferences**. Expand **JBoss Tools** and select **JBoss Runtime Detection**. In the **Paths** table, select the **Every start** check box for all of the paths that you want to be automatically searched on IDE start. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 2.32. Every Start Check Box Selected for JBossAS Path

2.3.3. Set a Default Server

JBoss Server Tools enables you to select a default server on which to carry out actions such as **Run on server**. This is useful when you have multiple server instances but use one predominately.

To set a default server, in the global toolbar of the JBoss perspective click the **Select a default server** icon

From the list of existing servers, click the server that you want to set as the default. Alternatively, to create a new default server click **New Server** and follow the instructions.

Figure 2.33. Default Server Menu Option

2.3.4. Default File Sets

File sets are collections of files that are listed under the server in the **Servers** view for ease of access. JBoss Server Tools generates a default file set for new JBoss community application servers that includes the server configuration file. But JBoss Server Tools also provides the ability for you to customize default file sets for individual servers and application server types.

To customize the file set for an individual server, in the **Servers** view expand the server. Right-click **Filesets** and click **Create File Filter**. In the **Name** field, type a name for the filter. In the **Root Directory** field, type the path of the directory in which the filter is to be applied or click **Browse** to navigate to the location. In the **Includes** and **Excludes** fields, type the regex patterns for filtering. Click **OK** to create the filter. The new filter is listed under the server in the **Servers** view and expanding the filter shows all of the matching files.

Figure 2.34. Create File Filter Menu Option

Figure 2.35. New File Filter Window

To customize the default file set for an application server type, click **Window** → **Preferences**. Expand **Server** and select **Default Filesets**. From the list, select a JBoss community application server type. Click **Add** or click **Remove** to customize the default file sets. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 2.36. Default Filesets Pane

2.3.5. Default Classpath Entries

Classpath entries specify the availability of `.jar` files for your projects. JBoss Server Tools generates a default classpath file set for new JBoss community application servers based on the `.jar` files that accompany each. But JBoss Server Tools also provides the ability for you to customize classpaths for individual servers and application server types.

To customize the classpath file set for an application server type, click **Window**→**Preferences**. Expand **Server**→**Runtime Environments** and select **Default Classpath Entries**. From the **Select classpath filesets for this runtime type** list, select a JBoss community application server type. Click **Add** or click **Remove** to customize the classpath file sets. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 2.37. Default Classpath Entries Pane

Forge Tools

3.1. Overview of Forge Tools

3.1.1. About Forge

Forge is an application for developing Java EE applications, promoting ease in getting started, working efficiently, and integrating technologies. It simplifies development workflow by providing automation at a command line level. Such automation is useful if you are working with new or complex technology or completing repetitive tasks.

Forge consists of a command line interface and sets of commands provided through plug-ins. The Forge command line accepts standard UNIX commands such as `cd` for change directory, `mkdir` to create a new directory and `touch` to create a new file. But the real power of Forge lies in its unique command set, which is extendable through plug-ins. For example, the `scaffold` command generates a user interface for an application and the `persistence` command adds persistence to a project. Furthermore, Forge comprehends a range of file types, with actions for working with their contents.

3.1.2. About Forge Tools

Forge Tools is tooling that integrates Forge into the IDE. It enhances your workflow by providing and integrating Forge functionality within the IDE.

Forge Tools consists of the Forge Console and background actions. The Forge Console provides a Forge command line on which to execute all of the Forge commands. The background actions result in the outcome of the Forge commands being immediately reflected in the IDE. For example, files are opened in editors and projects and directories expanded in explorer views as actions creating and modifying them are executed on the Forge command line.

3.2. Features of Forge Tools

3.2.1. Features Overview

The aim of this section is to guide you in using Forge Tools:

- View and manage the Forge Console view, from which the Forge command line is accessible
- Start and stop the Forge runtime server in order to access the command line
- Navigate to project resources in the Forge command line
- Understand how Forge commands affect the IDE through background actions, such as expanding directories in explorer views and opening files in editors

- Use wizards for accomplishing common Forge commands as an alternative to the Forge command line

3.2.2. Manage the Forge Console

A key element for working with Forge is the Forge command line. In order to see this, you need to open the **Forge Console** view.

Figure 3.1. Stopped Console in Forge Console View

To open the **Forge Console** view, click **Window** → **Show View** → **Other**. Expand **Forge**, select **Forge Console** and click **OK**. Alternatively, press **Ctrl+4**, at which you are prompted whether you also want to start the Forge runtime server. Irrespective of your response to starting Forge, the **Forge Console** view opens.

3.2.3. Manage the Forge Runtime Server

You must start a Forge runtime server in order to access the Forge command line. Forge Tools provides actions for starting and managing the Forge runtime server.

Start Forge

In the **Forge Console** view, click the **Start the default Forge runtime** icon

Alternatively, press **Ctrl+4** and at the prompt asking if you want to start the Forge runtime server, click **Yes**. A progress bar in the IDE window and text in the **Forge Console** view indicate the starting status of Forge.

Figure 3.2. Started Console in Forge Console View

Run Forge in the background

With a running instance of Forge, in the **Forge Console** view click the **Close** icon

Despite the **Forge Console** view not being open, Forge continues to run in the background. Reopen the **Forge Console** view to access the same Forge command line, identifiable by the existing command line output.

Stop Forge

In the **Forge Console** view, click the **Stop the running Forge runtime** icon

Alternatively, on the Forge command line, enter `exit`.

Figure 3.3. `exit` Command on Forge Command Line

3.2.4. Navigate to Project Resources on the Forge Command Line

Forge Tools enables you to efficiently navigate to project resources, such as directories and files, on the Forge command line.

To navigate to a project resource on the Forge command line, in the **Project Explorer** view right-click any project resource and click **Show In → Forge Console**. If the Forge runtime server is not started, you are prompted to start it. At the prompt, click **OK**.

Figure 3.4. Show In → Forge Console Menu Option

Alternatively, in the **Project Explorer** view, select a project resource and click the **Go to Selection** icon

in the **Forge Console** view.

Both of these actions result in the Forge command line automatically executing the `pick-up` command for the project resource. Following this, the command line navigates to the project resource, the command line indicates the selected resource, and the resource opens in an editor if it is a file.

Figure 3.5. `pick-up` Command Executed on Forge Command Line

3.2.5. Background Actions Invoked by Forge Commands

Forge Tools invokes background actions in the IDE in response to commands issued on the command line in the **Forge Console** view. For example, if you create a project file or directory on the Forge command line, the **Project Explorer** view is automatically refreshed to show the newly created project resource. A list of Forge commands that result in additional background actions is given below.

`cd`

The command navigates to the project directory and selects it in the **Project Explorer** and **Package Explorer** views.

`pick-up`

The command navigates to the project resource and in the case that the resource is a file it is opened in an editor within the IDE. The project resource is selected and expanded in the **Project Explorer** and **Package Explorer** views. If the resource resides outside the workspace, it is selected and expanded in the **Remote Systems Explorer** view, providing this is installed.

`open`

The command opens a file in an editor within the IDE. The project resource is selected and expanded in the **Project Explorer** and **Package Explorer** views. If the resource resides outside the workspace, it is selected and expanded in the **Remote Systems Explorer** view, providing this is installed.

`new-project`

The command creates a new project in a specified location. The project is automatically imported into the workspace and it is visible in the **Project Explorer** and **Package Explorer** views.

`persistence setup`

The command creates a `persistence.xml` file. This file is selected in the **Project Explorer** and **Package Explorer** views and it is automatically opened in an editor within the IDE.

`entity`

The command creates a new entity and associated Java file. This file is selected in the **Project Explorer** and **Package Explorer** views and it is automatically opened in the Java editor within the IDE.

`field`

The command creates a new field for an entity. The Java file associated with the entity is selected in the **Project Explorer** and **Package Explorer** views and it is automatically opened in an editor within the IDE and the field selected. The field is also selected in the **Outline** view.

3.2.6. Access a List of Forge Commands

Forge Tools provides access to a readily available list of Forge commands. Additionally, the commands can be easily inserted in to the Forge command line, as detailed below.

To view the list of Forge commands, with a running instance of Forge, press **Ctrl+4**. To insert one of the commands in to the Forge command line, in the pop-up window expand the command groups and double-click a command.

Figure 3.6. Forge Commands Window

3.2.7. Use Forge Wizards

You may prefer to work with wizards rather than the command line. Forge Tools provides wizards for some of the most used Forge commands, in addition to supporting command line functionality. There are three wizards currently available relating to entities, as detailed below.

Entities from Tables

This wizard generates entities from an existing database. There are options for creating a new project if one does not already exist and browsing for the `driver.jar` and driver classes.

REST Endpoints from Entities

This wizard generates REST endpoint for entities.

Scaffold UI from Entities

This wizard generates the necessary scaffolding for you to use JPA entities in your project. There are options for JavaServer Faces and AngularJS implementations, with the wizard creating the associated pages and Java backing beans.

To open a Forge Tools wizard, click **File**→**New**→**Other** and expand **JBoss Tools**→**Forge**. Select one of the listed wizards, click **Next** and follow the instructions. In the case that Forge is not already started, the wizard automatically starts it.

Figure 3.7. Forge Wizards Listed in New Window

3.3. Customizing Forge Tools

3.3.1. Customizing Overview

The aim of this section is to guide you in customizing Forge Tools:

- Customize when and how Forge starts
- Manage available Forge runtime servers

3.3.2. Customize the Forge Start

Forge Tools provides a number of options for customized Forge starts.

Figure 3.8. Forge Pane of Preferences Window

Start Forge on IDE start

Click **Window**→**Preferences** and select **Forge**. Select the **Start Forge when workbench starts** check box. Click **OK** to close the **Preferences** window.

Start Forge in debug mode

Click **Window**→**Preferences** and select **Forge**. Select the **Start Forge in Debug Mode** check box. Click **OK** to close the **Preferences** window. The debug mode enables you to view the progress of processes executed on the Forge command line in the **Debug** view. This mode is most useful if you are developing and testing plug-ins to extend the functionality of Forge.

Specify arguments for Forge start

Click **Window**→**Preferences** and select **Forge**. In the **Forge Startup VM Arguments** field, type the arguments you want Forge to use when it starts. Click **OK** to close the **Preferences** window.



Note

The standard Java VM arguments can be used when starting Forge. Additionally, Forge specific commands include `--debug` to start Forge in debug mode and `-pluginDir` to specify the directory where Forge is to look for plugins to install rather than the default `.forge/plugins` directory.

3.3.3. Manage Forge Runtime Servers

Forge Tools is distributed with a Forge runtime server but you may want to use different versions of Forge runtime servers. Forge Tools provides the ability to manage the Forge runtime servers that are available in the IDE, as detailed below.

To manage the available Forge runtime servers, click **Window** → **Preferences**, expand **Forge** and select **Installed Forge Runtimes**.

Figure 3.9. Installed Forge Runtimes Pane of Preferences Window

- To add a Forge runtime server, click **Add**. In the **Name** field, type a name to distinguish the Forge runtime server in the IDE. In the **Location** field, type the location of the runtime server or click **Browse** to navigate to the location. Click **OK** to close the window.
- To change the name or the location of a Forge runtime server, from the **Installed Forge Runtimes** list select a runtime and click **Edit**. Modify the **Name** and **Location** fields as appropriate. Click **OK** to close the window.
- To delete a Forge runtime server, from the **Installed Forge Runtimes** list select the runtime and click **Remove**.
- To set a runtime server as the default, select the check box corresponding to the Forge runtime server. This runtime server is used when Forge starts.

Click **OK** to close the **Preferences** window.



Important

It is not possible to edit or delete the Forge runtime server that is distributed with Forge Tools. This server is named **embedded** in the **Installed Forge Runtimes** list. Additionally, it is not possible to delete a Forge runtime server that is selected as the default. To delete a default runtime server, you must first select a different runtime server as the default.

Part II. Tools for Creating Web Interfaces

LiveReload Tools

4.1. Overview of LiveReload Tools

4.1.1. About LiveReload

LiveReload is an open source tool that refreshes web pages open in browsers as their source is edited. Immediate and automatic web page refreshing, without the need to manually refresh, assists to simplify the workflow of web developers.

LiveReload consists of server-side and client-side components. The server-side application monitors the source and sends notifications as the source is changed. On the client-side, a JavaScript snippet receives the notifications and invokes the browser to refresh the web page. The server-side and client-side components communicate using the WebSocket protocol, which allows the server-side application to send notifications to the client-side JavaScript snippet.

The JavaScript snippet can be installed and activated using a LiveReload browser extension. Alternatively, it can be inserted manually in the HTML pages. Depending upon the changes to the source, the web page is either reloaded or the browser updated instantly without reloading.

The LiveReload application is available for OS X and Microsoft Window operating systems and through a third-party provider for Linux-based operating systems. LiveReload browser extensions are available for Safari, Chrome and Firefox.

4.1.2. About LiveReload Tools

LiveReload Tools is tooling that implements LiveReload in the IDE. It enables you to take advantage of LiveReload browser refreshing while editing files within the IDE. The capabilities of LiveReload Tools also remove the need to install the LiveReload standalone application or browser extensions.

LiveReload Tools is the implementation of the LiveReload server-side component, which listens to the IDE for resource changes and sends notifications for refreshes as appropriate. The server can also be configured to inject the `livereload.js` JavaScript snippet into the source code, which removes the need to install LiveReload browser extensions.

Unique features of LiveReload tools include LiveReload for both workspace and deployed resources due to the use of HTTP protocol for all resources and LiveReload on external devices with access and ease of use for remote connections to LiveReload servers. The LiveReload server can also be used in conjunction with BrowserSim, extending LiveReload to simulated devices.

4.2. Features of LiveReload Tools

4.2.1. Features Overview

The aim of this section is to guide you in using LiveReload Tools:

- Create a LiveReload server to receive information from the IDE on resource changes
- Configure LiveReload for use with system web browsers and external devices
- View resources in LiveReload-enabled web browsers, including system and external device browsers and BrowserSim

4.2.2. Create a LiveReload Server

The LiveReload server is integral to the LiveReload integration with the IDE. In its default configuration, the server receives information from the IDE when resources change and sends notifications for refresh actions to be taken. The server may also be further configured to inject the necessary JavaScript code snippets into HTML files or enable remote connections. The procedure below guides you through creating a default LiveReload server.

Procedure 4.1. Create a LiveReload Server

1. Click the **Servers** view. If the **Servers** view is not visible, click **Window**→**Show View**→**Servers**.
2. Depending on the number of existing servers, follow the appropriate step:
 - If there are no existing servers, click **Click this link to create a new server**.
 - If there are one or more existing servers, right-click an existing server and click **New**→**Server**.
3. From the list of server types, expand **Basic** and select **LiveReload Server**.

Figure 4.1. LiveReload Server Selected in New Server Wizard

4. The **Server's host name** and **Server name** fields are automatically populated. The `localhost` value in the **Server's host name** field indicates that the server is to be run on the local system and the value in the **Server name** field is the name by which the LiveReload server is identified in the **Servers** view. You can edit these values as appropriate by typing in the fields.
5. Click **Finish** to close the window. The LiveReload server is listed in the **Servers** view.

Figure 4.2. LiveReload Server Listed in the Servers View

**Note**

Alternatively, LiveReload servers can be automatically created when you select to view workspace and deployed resources in web browsers using the actions provided by LiveReload Tools.

4.2.3. Configure the LiveReload Server

A LiveReload client must be inserted into HTML files in order for it to receive the LiveReload server notifications about changed resources. LiveReload Tools provides additional configuration options so that the LiveReload server can inject the necessary `<script>` element into the HTML resources. Furthermore, the LiveReload server can be configured to allow access from external devices, enabling LiveReload to function in browsers of remote devices.

The details below outline how to set the configuration options using the Server Editor. To open the Server Editor for a LiveReload server, in the **Servers** view double-click the LiveReload server. Alternatively, right-click the LiveReload server and click **Open** or press **F3**.

Figure 4.3. LiveReload Server Options in the Server Editor

Insert JavaScript code snippet to resources

In the Server Editor under **LiveReload Server Configuration**, select the **Inject the livereload.js script in HTML pages** check box.

Enable LiveReload in external device browsers

In the Server Editor under **LiveReload Server Configuration**, select the **Allow Remote Connections** check box. This option is disabled by default when a LiveReload server is created as it exposes your workspace files, which may not always be desirable.

All changes to the settings of a LiveReload server must be saved and the server restarted before the results will take effect. To save setting changes, press **Ctrl+S** or click **File** → **Save** or click the **Save** icon. To restart the server, in the **Servers** view right-click the LiveReload server and click **Restart**.

4.2.4. View Resources in LiveReload-enabled Browsers

You can use the actions of LiveReload Tools to open HTML, XHTML and AsciiDoc resources in LiveReload-enabled browsers, as detailed below. If the requisite LiveReload server does not exist or is not correctly configured, you are prompted by LiveReload Tools, which can complete the necessary requirements for you. Note that LiveReload Tools functionality may only work with deployed XHTML resources due to their dynamic nature.



Note

To use LiveReload Tools with AsciiDoc files, you must install the AsciiDoctor.js browser extension from <http://asciidoctor.org/news/2013/09/18/introducing-asciidoctor-js-live-preview/> on the AsciiDoctor website. The browser extension renders AsciiDoc files as HTML and it is available for Chrome and FireFox.

View workspace resources in a web browser

In the **Project Explorer** view, right-click the resource file and click **Open With** → **Web Browser via LiveReload Server**.

This requires the server to be configured to **Inject the liveload.js script in HTML pages** and, if the server is not correctly configured, you are prompted to enable this option.

Figure 4.4. Live Reload Settings Prompt

The IDE-specified external web browser opens, with LiveReload activated, and displays the workspace resource.

View deployed resources in a web browser

Ensure the server and application of the deployed resources are started. In the **Servers** view, right-click the application and click **Show In** → **Web Browser via LiveReload Server**.

Figure 4.5. Show In → Web Browser via LiveReload Server Menu Option

This requires the server to be configured to **Inject the liveload.js script in HTML pages** and, if the server is not correctly configured, you are prompted to enable this option.

Figure 4.6. Live Reload Settings Prompt

The IDE-specified external web browser opens, with LiveReload activated, and displays the deployed resource.



Note

To change the IDE-specified external web browser, click **Window** → **Preferences** and expand **General** → **Web Browser**. From the **External web browsers** list, select the browser to use for actions involving

external web browsers. Click **Apply** and click **OK** to close the **Preferences** window.

View deployed resources on an external device

Ensure the server and application of the deployed resources are started. In the **Servers** view, right-click the application and click **Show In** → **Web Browser on External Device**.

Figure 4.7. Show In → Web Browser on External Device Menu Option

This requires the server to be configured to **Inject the livereload.js script in HTML pages** and **Allow Remote Connections** and, if the server is not correctly configured, you are prompted to enable these options.

Figure 4.8. Live Reload Settings Prompt

A QR code and LiveReload server port URL corresponding to the deployed application are displayed and these can be input into external device browsers.

Figure 4.9. URL and QR Code for LiveReload in External Devices



Note

The configuration of a LiveReload server can be viewed and manually set in the Server Editor.

4.2.5. View Resources in LiveReload-enabled BrowserSim

The LiveReload server can be used in conjunction with BrowserSim. In this case, the server sends notifications about changed resources and BrowserSim inserts the JavaScript code, which invokes the simulated device browser window to refresh. The procedure below outlines how to enable LiveReload in BrowserSim for workspace and deployed resources.

Procedure 4.2. View Resources in LiveReload-enabled BrowserSim

1. Ensure the LiveReload server is started. If it is not started, in the **Servers** view right-click the LiveReload server and click **Start**.
2. Complete the appropriate step depending on the location of your resources:

- For workspace resources, in the **Project Explorer** view right-click the resource file and click **Open With**→**BrowserSim**.
- For deployed resources, in the **Servers** view right-click the application and click **Show In**→**BrowserSim**.

Figure 4.10. Show In→BrowserSim Menu Option



Important

Ensure the server and application of the deployed resources are started before attempting to view the resources in LiveReload-enabled BrowserSim. To start the server and the application, in the **Servers** view right-click each and click **Start**.

3. Right-click the simulated device and ensure the **Enable LiveReload** check box is selected.

Figure 4.11. Enable LiveReload Menu Option for BrowserSim



Important

The **Enable LiveReload** check box has no effect when the LiveReload server is set to insert the JavaScript code and the web resource is viewed in BrowserSim via the LiveReload server port URL. LiveReload is always enabled in this case.

Part III. Tools for Creating Mobile Applications

Mobile Web Tools

5.1. Overview of Mobile Web Tools

5.1.1. About Mobile Web Tools

Mobile Web Tools provides tooling for developing mobile web applications within the IDE. It simplifies the process of getting started with HTML5 and jQuery Mobile technologies that are used for these types of applications and provides efficient workflows for developing with them on a daily basis.

Mobile Web Tools consists of project wizards, file templates, the jQuery Mobile palette, and actions for developing with HTML5 and jQuery Mobile:

- The HTML5 project wizards and HTML5 jQuery Mobile file template enable you to quickly generate the foundations of mobile web applications and to view mobile web application programming in action.
- The jQuery Mobile palette offers a visual reference of the available HTML5 and jQuery Mobile user interface widgets, which can be effortlessly inserted into your project files with drag-and-drop functionality. In conjunction with the jQuery Mobile palette, widget wizards allow you to easily customize HTML5 and jQuery Mobile widgets when inserting them into your project.
- Code assistance and one-click action for viewing jQuery Mobile pages in browsers enable you to work more efficiently when developing mobile web applications in the IDE.

5.2. Features of Mobile Web Tools

5.2.1. Features Overview

The aim of this section is to guide you in using Mobile Web Tools:

- Generate the foundations of web applications with the project wizards and file templates
- Access the jQuery Mobile palette and add customized widgets to your mobile web application with the aid of the widget wizards
- Use code completion to assist in programming
- View pages of the application in various browsers

5.2.2. Create a Mobile Web Project

Project wizards are available to assist you in generating new mobile web applications. The project wizards are listed in JBoss Central:

- **HTML5 Project**, under **Start from scratch**
- **kitchensink-html5-mobile**, under **Start from a sample**



Important

In order to deploy the applications created by these wizards, JBoss Application Server 7.1 must be defined for use in the IDE. You can define a server using runtime detection from within a project wizard. From the **Requirements** table on the first page of the project wizard, select **server/runtime** and click **Install** or **Download and Install** and follow the instructions. Alternatively, before starting a wizard click **Window** → **Preferences**, expand **JBoss Tools** and select **JBoss Runtime Detection**. Click **Add** and follow the instructions.

Procedure 5.1. Create a HTML5 Project with the HTML5 Project Wizard

1. In JBoss Central under **Start from scratch**, click **HTML5 Project**.

Figure 5.1. HTML5 Project Wizard in JBoss Central

2. From the **Target Runtime** list, select an existing JBoss Application Server 7.1 server runtime environment or select **<none>**. When **<none>** is selected, a non-enterprise server runtime environment is assumed.
3. Click **Next**.
4. Complete the fields and options as detailed:
 - In the **Project name** field, type a name for the project.
 - In the **Package** field, type a package name for the project resources.
 - To create the project in a different location to the workspace default, clear the **Use default Workspace location** check box and type the path in the field or click **Browse** to navigate to the location.

Figure 5.2. HTML5 Project Wizard

5. Click **Next**.
6. Check the Maven details for the project to be created and click **Finish**.

7. During project creation, the wizard imports project dependencies. When the **HTML5 Project** wizard displays **'HTML5 Project' Project is now ready**, click **Finish** to close the wizard. A `README.md` file for the project automatically opens for viewing.

Procedure 5.2. Create a HTML5 Project with the kitchensink-html5-mobile Wizard

1. In JBoss Central under **Start from a sample**, hover the mouse over **Mobile Applications** and click **kitchensink-html5-mobile**.

Figure 5.3. kitchensink-html5-mobile Wizard in JBoss Central

2. Click **Next**.
3. To create the project in a different location to the workspace default, clear the **Use default location** check box and type the path in the field or click **Browse** to navigate to the location.

Figure 5.4. kitchensink-html5-mobile Wizard

4. Click **Finish**.
5. During project creation, the wizard imports project dependencies. When the **kitchensink-html5-mobile** wizard displays **'kitchensink-html5-mobile' Project is now ready**, click **Finish** to close the wizard. A `README.md` file for the project automatically opens for viewing.

5.2.3. Use a HTML5 jQuery Mobile File Template

Mobile Web Tools provides a HTML5 jQuery Mobile file template to assist in creating mobile web applications. The template inserts the necessary JavaScript and CSS library references into the HTML header and includes skeleton jQuery Mobile page and listview widgets in the HTML body. The procedure below details how to insert the template into your project.

Procedure 5.3. Use a HTML5 jQuery Mobile File Template

1. In the **Project Explorer** view, right-click a project and click **New** → **HTML File**.
2. Complete the fields and options as detailed:
 - In the **Enter or select the parent folder** field, type a project path or expand the project folder tree and select the parent folder for the new file.

- In the **File name** field, type the name for the new file. It is not essential to include the file extension in the name as this is automatically appended if it is found to be missing.

Figure 5.5. New HTML File Wizard

3. Click **Next**.
4. Complete the fields and options as detailed:
 - Ensure the **Use HTML Template** check box is selected.
 - From the **Templates** table, select **HTML5 jQuery Mobile Page**.

Figure 5.6. HTML5 jQuery Mobile Page Template for New HTML Files

5. Click **Finish**. The new HTML5 file is listed in the **Project Explorer** view and automatically opened in the JBoss Tools HTML Editor.

5.2.4. Access the jQuery Mobile Palette

Mobile Web Tools offers a jQuery Mobile palette, with wizards for adding jQuery Mobile and HTML5 widgets to your project. The jQuery Mobile palette, part of the **Palette** view, is available for use when working with HTML5 files in the JBoss Tools HTML Editor.

Figure 5.7. jQuery Mobile Palette in the Palette View

The jQuery Mobile palette is automatically displayed in the **Palette** view when a HTML5 file is opened in the JBoss Tools HTML Editor. To open a file in this editor, in the **Project Explorer** view right-click a HTML5 file and click **Open With** → **JBoss Tools HTML Editor**. Alternatively, if **JBoss Tools HTML Editor** is the default option for **Open With**, double-click the HTML5 file to open it in the editor. The file opens in the editor and the jQuery Mobile palette is displayed in the **Palette** view.



Note

The **Palette** view must be visible in order to see the jQuery Mobile palette. To open the view, click **Window** → **Show View** → **Other**, expand **General** and double-click **Palette**.

To show or hide an individual palette in the **Palette** view, click the name of the individual palette.

To search for a palette element within the jQuery Mobile palette, in the search field type a search term or phrase. The elements displayed in the jQuery Mobile palette are filtered as you type in the search field.

5.2.5. Insert a jQuery Mobile Palette Widget into a HTML5 File

The jQuery Mobile palette contains wizards for the HTML5 and jQuery Mobile user interface widgets commonly used in mobile web applications. The widgets are grouped in the palette by functionality, with tooltips providing widget descriptions.

To insert a palette widget in a file open in the JBoss Tools HTML Editor, drag the widget icon to the appropriate place in the file. Alternatively, ensure the text cursor is located at the desired insertion point in the file and click the widget icon. For widgets with no attributes that can be customized, such as **JS/CSS** and **Field Container**, the code snippets are immediately inserted into the file. For widgets with attributes that can be customized, a widget wizard opens allowing you to input attribute information. Once you have completed the customizable fields, click **Finish** and the code snippet is inserted into the file.

Figure 5.8. Page Widget Wizard

The widget wizards have three common aspects:

Design fields

These fields are unique to each widget. They allow you to customize the attributes of the widget by providing names, actions, numbers of elements, and styling themes. All widget wizards assign automatically generated values to the **ID** attribute in the case that you do not specify a value. Content assist is available for the **URL (href)** field by placing the text cursor in the field and pressing **Ctrl+Space**.

Add references to JS/CSS

This check box provides the ability to automatically add any missing library references to the HTML5 file that are required by the widget.

Preview Panes

These panes show previews of the code snippet for the widget and of the rendered widget. The preview panes can be shown and hidden by clicking **Show Preview** and **Hide Preview**, respectively.

5.2.6. Get Assistance with jQuery Mobile Programming

Mobile Web Tools offers code assist to help you when working with jQuery Mobile. Code assist lists available options for attributes and attribute values. Code assist is available for use in files and in the **URL (href)** field of widget wizards.

To view code assist in a file, ensure the text cursor is located at the desired insertion point in the file and press **Ctrl+Space**. Repeatedly press **Ctrl+Space** to cycle through HTML and JSF EL

completion options. To view more information about a listed item, select the item. To insert a listed item into the code, double-click the item.

Figure 5.9. Code Assist for File Contents

To view code assist in a widget wizard, ensure the text cursor is located in the **URL (href)** field and press **Ctrl+Space**. To view more information about a listed item, select the item. To insert a listed item into the code, double-click the item.

Figure 5.10. Code Assist for Widget Wizards

5.2.7. View jQuery Mobile Pages in a Browser

Mobile Web Tools provides an action to easily and quickly open jQuery Mobile pages in web browsers for viewing.

To open a jQuery Mobile page from a file open in the **JBoss Tools HTML Editor**, press **Ctrl** and move the mouse over the `<div>` tag corresponding to the page widget. Continue to press **Ctrl** and from the menu select one of the options:

- **Open With Browser**, which shows the page in the default browser of the IDE
- **Open With BrowserSim**, which shows the page in BrowserSim

Figure 5.11. Open With Menu Option for a jQuery Mobile Page Widget

5.3. Customizing Mobile Web Tools

5.3.1. Customizing Overview

The aim of this section is to guide you in customizing Mobile Web Tools:

- Customize the HTML5 jQuery Mobile templates available in the IDE

5.3.2. Customize jQuery Mobile File Templates

The **HTML5 jQuery Mobile Page** template for new HTML files is provided by Mobile Web Tools. You can customize this template and add more jQuery Mobile templates to the IDE.

To customize the jQuery Mobile templates available in the IDE, click **Window**→**Preferences**. Expand **Web**→**HTML Files**→**Editor** and select **Templates**.

Figure 5.12. Templates Pane of Preferences Window

There are a number of available actions:

Add a template

There are several options for adding templates:

- To create a new template, click **New**. In the **Name** and **Description** fields, type a name and description of the template, respectively. In the **Pattern** field, type the code for the template. From the **Context** list, select the instance in which the IDE should make the template available. Click **OK** to close the window.
- To restore all templates that have been deleted, click **Restore Removed**.
- To load an existing template into the IDE, click **Import** and select the file. The file must be an XML file, with appropriate file headers and the HTML5 and jQuery Mobile content written in XML syntax and contained between XML `template` and `templates` tags.

Edit a template

From the table, select a template and click **Edit**. You can modify the name, description, code content and context in which the IDE makes the template available. After making changes, click **OK** to close the window.

Remove a template

From the table, select a template and click **Remove**.

BrowserSim

6.1. Overview of BrowserSim

6.1.1. About BrowserSim

BrowserSim is a mobile web browser simulator. It enables you to view and interact with web pages as they would appear on mobile devices.

BrowserSim consists of an external web browser window. This browser window is different from standard browsers in that it can be transformed to simulate browsers on a variety of mobile devices. The browser window can also return touch events in response to interaction and be rotated to demonstrate the changes that would occur to a web application as a result of rotating a mobile device. Additionally, BrowserSim provides access to tools for inspecting and editing the source of web pages.

6.1.2. System Requirements for BrowserSim

The overall system requirements of JBoss Tools are applicable for BrowserSim and can be viewed at <https://community.jboss.org/wiki/MatrixOfSupportedPlatformsRuntimesAndTechnologiesInJBossToolsJBDS> on the JBoss Tools website. But, as reiterated below, there are additional system requirements and restrictions when using BrowserSim on Microsoft Windows operating systems.

BrowserSim depends on WebKit and, consequently, requires Apple Safari to be installed on Microsoft Windows operating systems. Only a 32-bit version of Apple Safari is available for Microsoft Windows operating systems. To work around this restriction for 64-bit Microsoft Windows operating systems, you must set BrowserSim to use a 32-bit JVM when running in 64-bit versions of Eclipse. Note that 32-bit JVM choice is limited to Oracle 32-bit JRE 1.6, JDK 1.6, or JDK 1.7 on Microsoft Windows operating systems because Oracle 32-bit JRE 1.7 is incompatible with Apple Safari.

To set BrowserSim to use a 32-bit JVM, click **Window** → **Preferences**. Expand **JBoss Tools** and select **BrowserSim/CordovaSim**. Under **Select JRE to run BrowserSim**, click **Select** and from the list select a 32-bit JRE or Java developer kit. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 6.1. BrowserSim/Cordova Pane of Preferences Window

6.2. Features of BrowserSim

6.2.1. Features Overview

The aim of this section is to guide you in using BrowserSim:

- View local and remote web applications in BrowserSim
- Change the appearance of simulated devices, by rotating them and changing or removing their skins
- Check the look and functionality of web pages in different browsers, such as the default browser or additional synchronized simulated devices
- Generate screen captures of simulated devices and their web pages
- View the page source of web pages with editors, Firebug Lite and Weinre

6.2.2. View a Web Application on BrowserSim

You can use BrowserSim to view web applications on local file systems or deployed on runtime servers.

To view web applications in local file systems

In the **Project Explorer** view, select a `.html` file and click the **Run BrowserSim** icon

Alternatively, right-click a `.html` file and click **Open With** → **BrowserSim**.

To view deployed web applications

In the **Servers** view, expand the server on which the application is deployed. Right-click the application and click **Show In** → **BrowserSim**. Alternatively, select the application and click the **Run BrowserSim** icon. If BrowserSim is the default IDE browser, you can also right-click the application and click **Show In** → **Web Browser**.



Important

Ensure the server and application of the deployed resources are started before attempting to view the resources in BrowserSim. To start the server and the application, in the **Servers** view right-click each and click **Start**.

BrowserSim employs intelligent loading when selecting which web page to open in the browser window. If a web page is already open in a standard browser window, this is the web page that is loaded in BrowserSim.

Figure 6.2. Web Page Open on Simulated Device

Runtime errors for simulated devices can be viewed in the **Error Log** view. To open the **Error Log** view, click **Window** → **Show View** → **Other**. Expand **General** and double-click **Error Log**.

6.2.3. Manage Web Applications on BrowserSim

BrowserSim provides actions to assist with viewing web applications on its simulated devices.

Open a web link

In the browser window, click the web link. The link opens in the simulated device browser window.

Show or hide the address bar

Click the notification bar, where the connectivity, time and battery are displayed, to show or hide the address bar.

Reload a web page

Click the **Reload** icon. The icon differs in appearance and location across the simulated devices.

Stop loading a web page

Click the **Stop** icon at the end of the address bar.

Use simulated touch

Right-click the simulated device and click **Enable Touch Events**. This option changes the cursor to a circle when it is over the simulated device browser window and returns mouse events in the browser window as touch events.

Close BrowserSim

Right-click the simulated device and click **Close**. Open BrowserSim windows automatically close when the IDE closes.

6.2.4. Change the Appearance of a Simulated Device

BrowserSim provides a number of ways to change the appearance of simulated devices. Changes made to the appearance of simulated devices are retained by BrowserSim and they are automatically applied when it starts in future.

Rotate the view between portrait and landscape modes

Click any corner of the simulated device. Alternatively, right-click the simulated device and click **Rotate Left** or **Rotate Right**.

Figure 6.3. Rotate Mouse Pointer in Upper-right Corner of Simulated Device

Change the simulated device

Right-click the simulated device, click **Skins** and select from the listed devices.

Figure 6.4. Skins Menu Option

Remove or use skins

To view a plain browser window without the mobile device skin, right-click the simulated device and click **Use Skins**. To reapply the skin, click **Device**→**Use Skins**.

6.2.5. View a Web Page in Different Browsers and Simulated Devices

From within BrowserSim, web pages can be viewed in different browsers and simulated devices.

View in the default browser of the system

Right-click the simulated device and click **Open in default browser**. An external browser window opens and displays the web page.

View simultaneously on synchronized simulated devices

Right-click the simulated device and click **Open Synchronized Window**. Select from the list of available skins for the additional simulated device. An additional simulated device opens and displays the same web page as that of the synchronized simulated device. Opening a web page in one synchronized simulated device results in the web page opening in all the synchronized simulated devices.

Figure 6.5. Synchronized Simulated Devices

6.2.6. Generate a Screen Capture of a Simulated Device

BrowserSim provides the ability to generate screen captures of a simulated device and the web pages it shows.

To generate a screen capture of a simulated device, right-click the simulated device and click **Screenshot**. Select the output for the screen capture from the list of options:

- **Save** to save as a `.png` file in the default location. The default location is a customizable setting and if it is not set you are prompted to select a location to which to save the file each time.
- **Save As** to save as a `.png` file in a location you specify.
- **Copy to Clipboard** to copy the graphic for immediate use.

6.2.7. Activate LiveReload for BrowserSim

LiveReload for BrowserSim refreshes web pages open in simulated device browser windows as the source is edited in the IDE. A LiveReload server sends notifications as resources are changed in the IDE and BrowserSim inserts the JavaScript code, which invokes the simulated device browser window to refresh. The procedures below outline how to create a LiveReload server and how to enable LiveReload in BrowserSim for workspace and deployed resources.

Procedure 6.1. Create a LiveReload Server

1. Click the **Servers** view. If the **Servers** view is not visible, click **Window→Show View→Servers**.
2. Depending on the number of existing servers, follow the appropriate step:
 - If there are no existing servers, click **Click this link to create a new server**.
 - If there are one or more existing servers, right-click an existing server and click **New→Server**.
3. From the list of server types, expand **Basic** and select **LiveReload Server**.

Figure 6.6. LiveReload Server Selected in New Server Wizard

4. The **Server's host name** and **Server name** fields are automatically populated. The `localhost` value in the **Server's host name** field indicates that the server is to be run on the local system and the value in the **Server name** field is the name by which the LiveReload server is identified in the **Servers** view. You can edit these values as appropriate by typing in the fields.
5. Click **Finish** to close the window. The LiveReload server is listed in the **Servers** view.

Figure 6.7. LiveReload Server Listed in the Servers View

Procedure 6.2. View Resources in LiveReload-enabled BrowserSim

1. Ensure the LiveReload server is started. If it is not started, in the **Servers** view right-click the LiveReload server and click **Start**.
2. Complete the appropriate step depending on the location of your resources:
 - For workspace resources, in the **Project Explorer** view right-click the resource file and click **Open With→BrowserSim**.
 - For deployed resources, in the **Servers** view right-click the application and click **Show In→BrowserSim**.

Figure 6.8. Show In→BrowserSim Menu Option



Important

Ensure the server and application of the deployed resources are started before attempting to view the resources in LiveReload-enabled BrowserSim. To start the server and the application, in the **Servers** view right-click each and click **Start**.

3. Right-click the simulated device and ensure the **Enable LiveReload** check box is selected.

Figure 6.9. Enable LiveReload Menu Option for BrowserSim



Important

The **Enable LiveReload** check box has no effect when the LiveReload server is set to insert the JavaScript code and the web resource is viewed in BrowserSim via the LiveReload server port URL. LiveReload is always enabled in this case.

6.2.8. View the Source of a Web Page

The source of web pages displayed in simulated device browser windows can be viewed with a variety of applications that can be initiated from within BrowserSim.

Open the page source in an editor

Right-click the simulated device and click **View Page Source**. The file containing the page source opens in an IDE editor.

Inspect the page source with Firebug Lite

Right-click the simulated device and click **Debug** → **Firebug Lite**. The Firebug Lite application is displayed in an external window.

Inspect and edit the page source with Weinre

Right-click the simulated device and click **Debug** → **Weinre**. The Weinre Inspector is displayed in an external window. Weinre supports remote debugging, enabling you to debug an application running on a mobile device from your desktop browser.

6.3. Customizing BrowserSim

6.3.1. Customizing Overview

The aim of this section is to guide you in customizing BrowserSim:

- Make BrowserSim more prominent to use by making it the default browser, by adding its icon to the global toolbar and creating a shortcut key for launching it
- Extend the functionality of BrowserSim by adding or modify the simulated devices it provides
- Customize the default settings of BrowserSim for large simulated devices, for LiveReload, for screen captures and for Weinre

6.3.2. Make BrowserSim the Default Browser

You can set BrowserSim to be the default browser used in actions such as **Show In→Web Browser** and **Run on Server**.

To set BrowserSim as the default browser, click **Window→Web Browser→BrowserSim**.

Alternatively, click **Window→Preferences**, expand **General** and select **Web Browser**. Click **User external web browser** and from the **External web browsers** list select the **BrowserSim** check box. Click **Apply** and click **OK** to close the **Preferences** window.

6.3.3. Add BrowserSim to the Global Toolbar

The BrowserSim icon is part of the BrowserSim toolbar and, by default, this toolbar is included in the global toolbar of the JBoss perspective. But the BrowserSim icon might not be visible in other perspectives because the icons in the global toolbar change depending on the perspective you are using. As detailed in the procedure below, you can add the BrowserSim toolbar to other perspectives.

Procedure 6.3. Add BrowserSim to the Global Toolbar in the Current Perspective

1. Ensure you are using the perspective in which you would like to add the BrowserSim toolbar. To open the desired perspective, click **Window→Open Perspective→Other** and double-click the perspective.
2. Click **Window→Customize Perspective**.
3. In the **Command Groups Availability** tab, select the **BrowserSim** check box. This option makes the BrowserSim toolbar available for adding to the current perspective.
4. In the **Tool Bar Visibility** tab, ensure the **BrowserSim** check box is selected. This option adds the BrowserSim toolbar to the global toolbar of the current perspective.

Figure 6.10. BrowserSim Check Box Selected in Tool Bar Visibility tab of Customize Perspective Window

5. Click **OK** to close the window. The **Run BrowserSim** icon is now visible in the global toolbar of the perspective.

6.3.4. Set a Shortcut for the Run BrowserSim Action

If you use BrowserSim frequently but do not want to set it as the default browser, you can set a shortcut for the **Run BrowserSim** action, as described in the procedure below.

Procedure 6.4. Set a Shortcut for BrowserSim

1. Click **Window** → **Preferences**, expand **General** and select **Keys**.
2. To find the **Run BrowserSim** action, in the **type filter text** field enter `BrowserSim`.
3. From the table, select **Run BrowserSim**.
4. In the **Binding** field, type the key combination you want to use as a shortcut. Check the **Conflicts** table to ensure the key binding you have chosen does not conflict with existing shortcuts.
5. Once a unique key binding is selected, click **Apply** and click **OK** to close the **Preferences** window.

Figure 6.11. Keys Pane of Preferences Window

6.3.5. Add or Modify Devices in BrowserSim

You may wish to preview a web application on a simulated mobile device that is not predefined in BrowserSim. You can add more devices to BrowserSim and modify the existing devices, as detailed below.

Procedure 6.5. Add Devices to BrowserSim

1. Right-click the simulated device and click **Preferences**.
2. In the **Devices** section of the **Devices** tab, click **Add**.
3. Complete the fields and options as detailed:
 - In the **Name** field, type the name you want to give the device.
 - In the **Width** and **Height** fields, type the dimensions of the device window in pixels.
 - In the **Pixel Ratio** field, type a value for the ratio of CSS pixels to device pixels.

- In the **User Agent** field, type the User Agent string of your device. Clearing the **User Agent** check box results in the default User Agent for the BrowserSim browser being used.



Note

User Agent is a string denoting the device, operating system and browser combination. This string may be used by websites to provide content tailored for devices, operating systems and browsers. Information is widely available on the Internet to assist you in identifying the User Agent associated with a particular device.

- From the **Skin** list, select the skin to be used or select **None**.

Figure 6.12. Add Device Window

4. Click **OK** to add the new device. It is listed in the **Devices** table.
5. Click **OK** to close the **Preferences** window.

To modify existing devices in BrowserSim, right-click the simulated device and click **Preferences**. In the **Devices** table, select a device and click **Edit**. Once you have finished editing the fields, click **OK**. Click **OK** to close the **Preferences** window.

6.3.6. Change the Default Behavior when a Device does not Fit the Display

When a device window is too large to fit the display of the system you are prompted about which action the IDE is to taken. This default IDE behavior can be modified, with alternative options of always truncate or never truncate.

To change the default behavior, right-click the simulated device and click **Preferences**. In the **Truncate the device window when it does not fit display** section of the **Devices** tab, click **Always truncate** or **Never truncate** to change the behavior as appropriate and click **OK** to close the **Preferences** window.

Figure 6.13. Truncate Preferences

6.3.7. Change the Default LiveReload Port

The LiveReload server uses a port to communicate resource changes to BrowserSim. The default port can be changed.

To change the default LiveReload port, right-click the simulated device and click **Preferences**. Click the **Settings** tab and view the **LiveReload options** section. Note that if LiveReload is not already enabled for BrowserSim you must select the **Enable LiveReload** check box. In the **LiveReload Port** field type the port number you want to use. Click **OK** to close the **Preferences** window.

Figure 6.14. LiveReload Port Preferences

6.3.8. Set the Location for Saved Screen Captures

The **Save** option for screen captures saves graphics files to a set location when that location has been predefined.

To set the location, right-click the simulated device and click **Preferences**. In the **Screenshots** section of the **Settings** tab, in the **Location** field type the location where you want graphics files to be saved or click **Browse** to navigate to the location. Click **OK** to close the **Preferences** window.

Figure 6.15. Screen Capture Preferences

6.3.9. Change the Default Settings for Weinre

By default, the **Weinre** option for viewing the source of a web page uses the Weiner server provided by PhoneGap. If you have a different Weiner server available, the default Weinre settings can be changed.

To change the default settings for Weinre, right-click the simulated device and click **Preferences**. In the **Weinre** section of the **Settings** tab, in the **Script URL** field type the address of the `.js` file provided by the Weinre server and in the **Client URL** field type the address of the web page showing the Weinre Inspector interface. Click **OK** to close the **Preferences** window.

Figure 6.16. Weinre Preferences

Hybrid Mobile Tools and CordovaSim

7.1. Overview of Hybrid Mobile Tools and CordovaSim

7.1.1. About Apache Cordova

Apache Cordova is a framework for hybrid mobile application development. It simplifies mobile application development by enabling developers to use device-independent APIs, in place of native code, to access device functionality.

Cordova consists of plug-ins, each providing a wrapper API to gain access to corresponding native device APIs. Cordova APIs are available for a range of device functionality, including cameras, accelerometers, and filesystems. All of the Cordova APIs have a JavaScript interface and native backing code to support a variety of different device operating systems, including Android and iOS.

The JavaScript interface of Cordova APIs provides consistent methods for accessing native device functionality regardless of operating system. Applications using Cordova are written once and packaged using native software development kits (SDKs) to produce hybrid mobile applications.

Cordova applications have a distinctive project structure:

- An essential `config.xml` file holds information about the application. It specifies features, preferences and access details that affect how the application works, for example whether the application responds to orientation changes.
- An application is implemented as a web page. The default starting page is `index.html` and device-independent resources are stored in a `www` directory.
- An application must reference the `cordova.js` file, which holds necessary API bindings that enable the application to interact with device features.

Apache Cordova is an open source project, of which there are various distributions. PhoneGap, in addition to being the originator of Cordova, and IBM Worklight are such distributions.

7.1.2. About Hybrid Mobile Tools

Hybrid Mobile Tools provides tooling for developing Cordova-based hybrid mobile applications within the IDE. It simplifies the process of getting started with the Cordova technology that can be used for these types of applications and provides workflows for developing Cordova-based hybrid mobile applications.

Hybrid Mobile Tools consists of the Apache Cordova API and wizards, a dedicated editor and actions for developing Cordova-based hybrid mobile applications for Android and iOS operating systems:

- The Hybrid Mobile Application project wizard creates a sample project to demonstrate the Cordova project structure and provides you with a template from which to create your own applications.
- The Cordova Configuration Editor offers simplified management of `config.xml` files, with dedicated tabs for adding and editing project descriptions, features, preferences and access. An additional tab enables direct editing of `config.xml` files.
- The Cordova Plug-in Discovery wizard assists you to add Cordova plug-ins to your projects. Plug-ins can be installed from the Cordova registry, Git locations and system directories. Information displayed in the **Project Explorer** view is extended to show installed plug-ins and enable their management.
- Actions are provided for calling external Android and iOS SDKs to package Cordova-based hybrid mobile projects into native applications and subsequently run them on their associated simulators or connected Android devices.
- Export wizards are available for exporting Cordova-based hybrid mobile projects from the IDE workspace as Cordova-enabled Android or iOS projects and ready-to-sign applications.

Hybrid Mobile Tools, together with CordovaSim, provides a rounded hybrid mobile development and testing environment.

7.1.3. About CordovaSim

CordovaSim is a mobile application simulator. It enables you to view and test Cordova-based hybrid mobile applications through a mobile device simulator.

CordovaSim consists of a device input panel based on an extended version of Apache Ripple, teamed with BrowserSim. The input panel provides the ability to give sample input for device features to the application, such as camera, geolocation and accelerometer data. BrowserSim displays and enables user interaction with the application.

Extensions to the device input panel for CordovaSim include the following:

- CordovaSim supports a range of Cordova API implementations: the core Apache Cordova plug-ins, several PhoneGap plug-ins and custom plug-ins. Supported PhoneGap plug-ins include `BarcodeScanner`, `InAppBrowser` and `ChildBrowser`. (Note that exceptions to the supported core Apache Cordova plug-ins are `File` and the `captureAudio` and `captureVideo` methods of `Capture`.)
- CordovaSim automatically injects an up-to-date version of the `cordova.js` library into your project code at simulation time in the case that it is missing. Typically this file is added when a project is built with a native SDK. This response by CordovaSim enables it to simulate applications of Cordova-based hybrid mobile projects and Cordova-enabled Android projects without requiring native SDKs.

- The use of BrowserSim for displaying the application extends the functionality of the mobile application simulator to encompass all of the BrowserSim functionality. This functionality includes skins, screen captures and LiveReload.

7.1.4. System Requirements for Hybrid Mobile Tools

The overall system requirements of JBoss Tools are applicable for Hybrid Mobile Tools and can be viewed at <https://community.jboss.org/wiki/MatrixOfSupportedPlatformsRuntimesAndTechnologiesInJBossToolsJBDS> on the JBoss Tools website. But, as reiterated below, there are additional system requirements and restrictions when using Hybrid Mobile Tools.

Hybrid Mobile Tools actions involving Android and iOS require the associated SDKs to be installed on your system.



Note

CordovaSim is a standalone simulator for mobile device operating systems, including Android and iOS. It does not require native SDKs to be installed in order to simulate successfully.

Android SDK (including emulator)

This is available as part of Android Development Tools. For further information see <http://developer.android.com/sdk/index.html> on the Android Developers website. Note that for Android actions in Hybrid Mobile Tools it is only necessary to install the Android SDK, which includes the Android Emulator.

Once downloaded, you must use the included Android SDK Manager to further download necessary APIs and set up Android virtual devices (AVDs).



Important

You must have Android API 17 or later installed on your system to use the Hybrid Mobile Tools **Run on Android Emulator** action. You are prompted by an error message when only Android API 16 or earlier is available. In the event that multiple Android API of 17 and later are available, Hybrid Mobile Tools uses the latest version.

Further, once Hybrid Mobile Tools is installed, you must set the Android SDK location in the IDE before you can use Hybrid Mobile Tools actions involving Android.

iOS SDK (including simulator)

This is available as part of Apple XCode. For further information see <https://developer.apple.com/xcode/> on the Apple website.

See Also:

- [Section 7.3.2, “Set the Android SDK Location”](#)

7.1.5. System Requirements for CordovaSim

The overall system requirements of JBoss Tools are applicable for CordovaSim and can be viewed at <https://community.jboss.org/wiki/MatrixOfSupportedPlatformsRuntimesAndTechnologiesInJBossToolsJBDS> on the JBoss Tools website. But CordovaSim uses BrowserSim and, as reiterated below, there are additional system requirements and restrictions when using BrowserSim on Microsoft Windows operating systems.

BrowserSim depends on WebKit and, consequently, requires Apple Safari to be installed on Microsoft Windows operating systems. Only a 32-bit version of Apple Safari is available for Microsoft Windows operating systems. To work around this restriction for 64-bit Microsoft Windows operating systems, you must set BrowserSim to use a 32-bit JVM when running in 64-bit versions of Eclipse. Note that 32-bit JVM choice is limited to Oracle 32-bit JRE 1.6, JDK 1.6, or JDK 1.7 on Microsoft Windows operating systems because Oracle 32-bit JRE 1.7 is incompatible with Apple Safari.

If BrowserSim is already installed, it can be set to use a 32-bit JVM either before or after installing CordovaSim. To set BrowserSim to use a 32-bit JVM, click **Window** → **Preferences**. Expand **JBoss Tools** and select **BrowserSim/CordovaSim**. Under **Select JRE to run BrowserSim**, click **Select** and from the list select a 32-bit JRE or Java developer kit. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 7.1. BrowserSim/Cordova Pane of Preferences Window

7.1.6. Install Hybrid Mobile Tools and CordovaSim

Hybrid Mobile Tools and CordovaSim are not packaged as part of JBoss Tools installations. These plug-ins must be installed independently through JBoss Central, as detailed in the procedure below.

Procedure 7.1. Install Hybrid Mobile Tools and CordovaSim

1. To install these plug-ins, drag the following link into JBoss Central: <https://devstudio.jboss.com/central/install?connectors=org.jboss.tools.aerogear.hybrid>. Alternatively, in JBoss Central select the **Software/Update** tab. In the **Find** field, type **JBoss Hybrid Mobile Tools** or scroll through the list to locate **JBoss Hybrid Mobile Tools + CordovaSim**. Select the corresponding check box and click **Install**.

Figure 7.2. Start the Hybrid Mobile Tools and CordovaSim Installation Process with the Link

Figure 7.3. Find Hybrid Mobile Tools and CordovaSim in JBoss Central Software/Update Tab

2. In the **Install** wizard, ensure the check boxes are selected for the software you want to install and click **Next**. It is recommended that you install all of the selected components.
3. Review the details of the items listed for install and click **Next**. After reading and agreeing to the license(s), click **I accept the terms of the license agreement(s)** and click **Finish**. The **Installing Software** window opens and reports the progress of the installation.
4. During the installation process you may receive warnings about installing unsigned content. If this is the case, check the details of the content and if satisfied click **OK** to continue with the installation.

Figure 7.4. Warning Prompt for Installing Unsigned Content

5. Once installing is complete, you are prompted to restart the IDE. Click **Yes** to restart now and **No** if you need to save any unsaved changes to open projects. Note that changes do not take effect until the IDE is restarted.

Once installed, you must inform Hybrid Mobile Tools of the Android SDK location before you can use Hybrid Mobile Tools actions involving Android.

See Also:

- [Section 7.3.2, “Set the Android SDK Location”](#)

7.2. Features of Hybrid Mobile Tools and CordovaSim

7.2.1. Features Overview

The aim of this section is to guide you in using Hybrid Mobile Tools and CordovaSim:

- Create the basis of new hybrid mobile projects using the project wizard
- Add and remove Cordova plug-ins from your applications
- Manage the Cordova functionality of applications using the Cordova Configuration Editor

- Run and test hybrid mobile applications with CordovaSim or call external Android and iOS SDKs to run applications on their associated simulators and, in the case of Android, attached devices
- Customize the settings used by CordovaSim, Android and iOS simulators for running hybrid mobile applications
- Export workspace applications as Cordova-enabled native projects or ready-to-sign applications

7.2.2. Create a Hybrid Mobile Project

A project wizard is available to assist you in generating new hybrid mobile applications, as demonstrated in the procedure below. It creates a Cordova project with structure compatible with projects generated by the Cordova command-line interface (CLI).

Procedure 7.2. Create a Hybrid Mobile Project

1. Click **File**→**New**→**Project**.
2. Expand **Mobile**, select **Hybrid Mobile (Cordova) Application Project** and click **Next**.

Figure 7.5. Select Hybrid Mobile Application Project in New Project Wizard

3. Complete the following fields:
 - In the **Project name** field, type a name for the project. This value is the name of the directory to be created and in which the source files for the application are stored, for example `My_App`.
 - In the **Name** field, type a name by which the hybrid mobile application is to be known. This value is the display text used to represent the application in listings and device home screens, for example `My Application`.
 - In the **ID** field, type an ID for the hybrid mobile application. The value is typically a reverse domain-style identifier, for example `com.example.myapplication`, and for applications that are to be distributed through device platform application stores the ID value will be provided by the store.



Note

There are restrictions on the ID you can use for an application. IDs must consist only of alphanumeric characters and dots. IDs must begin with an alpha character and contain at least one dot.

Figure 7.6. Hybrid Mobile Application Project Wizard

4. By default, the project is created in a subdirectory of the workspace that is named according to the project name. To change the default location, clear the **Use default location** check box. From the **Choose file system** list, select the **default** or **RSE** (Remote System Explorer) as appropriate. In the **Location** field, type the path where the project is to be created or click **Browse** to navigate to the location.
5. To create the project, click **Finish**.

During project creation, the wizard imports project dependencies and populates a `config.xml` file. Once created, the project is listed in the **Project Explorer** view and the `config.xml` file is automatically opened in the **Cordova Configuration Editor**.

7.2.3. Enable Cordova Plug-ins for an Application

Plug-ins, or features, provide the application with access to the necessary Cordova APIs at runtime. Hybrid Mobile Tools provides actions for installing and removing plug-ins associated with applications, as detailed here.

Add a plug-in

In the **Project Explorer** view, right-click the `plugins` folder of the project and click **Install Cordova Plug-in**.

The **Cordova Plug-in Discovery** wizard opens. The **Cordova Plug-in Discovery** wizard can install Cordova plug-ins from Cordova registries, Git locations and system directories:

- In the **Registry** tab, in the **Find** field enter the name of the feature or scroll through the list to find the plug-in. Select the check box of the plug-in and click **Next**. Check the details of the selected plug-in and use the drop-down list next to the plug-in name to select the version to be installed. Click **Finish**.
- In the **Git** tab, in the **URL** field type the URL that specifies the plug-in location. Click **Finish**.
- In the **Directory** tab, in the **Directory** field type the path of the plug-in or click **Browse** to navigate to the location. Click **Finish**.

Figure 7.7. Example of a Cordova Plug-in Selected in the Registry Tab of Cordova Plug-in Discovery Wizard

After installing the plug-in, configuration files are automatically updated with relevant settings for the plug-in. Note that the `config.xml` file is only updated with **features** and **param** entries if an installed plug-in has native parts.



Important

Some plug-ins require you to define preference values. At the time of installing such a plug-in, Hybrid Mobile Tools creates an item in the **Preference** table with the appropriate preference name but with a value of **PLEASE_DEFINE**. You must edit the preference and provide the required value.

Remove a plug-in

In the **Project Explorer** view, in the plugins folder right-click the plug-in and click **Remove Cordova Plug-in**.



Note

Alternatively, you can add and remove plug-ins by using the **Platform Properties** tab of the Cordova Configuration Editor.

See Also:

- [Section 7.2.4.2, “Manage Cordova Settings in the Platform Properties Tab”](#)

7.2.4. Manage Cordova Settings of a Hybrid Mobile Project

The Cordova Configuration Editor is available for managing the settings of Cordova projects that are specified in the `config.xml` file. This editor has three tabs: Overview, Platform Properties, and `config.xml`. As described below, the first two tabs provide interfaces for configuring the settings specified in the `config.xml` file and the third tab enables direct editing of the file.

The **Overview** tab details explanatory application information. Within this tab you can specify the name and description of the project, the content source of the application, and author details.

Figure 7.8. Overview Tab of the Cordova Configuration Editor

The **Platform Properties** tab specifies Cordova project functionality, such as features (plug-ins and parameters), preferences and access.

Figure 7.9. Platform Properties Tab of the Cordova Configuration Editor

The **config.xml** tab provides an editor in which to view and modify the `config.xml` file directly.

Figure 7.10. config.xml Tab of the Cordova Configuration Editor

To open the Cordova Configuration Editor for a specific hybrid mobile project, in the **Project Explorer** view right-click the `config.xml` file. Click **Open With** → **Cordova Configuration Editor**. All changes to the Cordova settings of a project must be saved before the results take effect. To save, press **Ctrl+S**.

7.2.4.1. Manage Cordova Settings in the Overview Tab

The Overview tab of the Cordova Configuration Editor enables you to edit the application information of a hybrid mobile project. Information pertains to the name, description and author of the application. More specifically, the **Name and Description** section details the application ID, name, version, description and content source or home page. The **Author** section holds the author name, email and URL. All field values can be edited as detailed below.

Change the value of a variable

Click the appropriate field and edit the content.

All changes to `config.xml` must be saved before the results take effect. To save, press **Ctrl+S**.

7.2.4.2. Manage Cordova Settings in the Platform Properties Tab

The Platform Properties tab of the Cordova Configuration Editor enables you to specify the Cordova settings in your hybrid mobile project. Features, parameters, preferences and access can be added and removed as detailed below.

Add a feature

Features are the Cordova API plug-ins required by the application in order to access native APIs at runtime. Examples include `Camera`, `Contacts` and `Geolocation`.

To add a feature, click **Add** for the **Features** table. The **Cordova Plug-in Discovery** wizard opens. Follow the instructions as appropriate for the plug-in source:

- For the Cordova registry, click the **Registry** tab. In the **Find** field, enter the name of the feature or scroll through the list to find the plug-in. Select the check box for the plug-in and click **Next**. Check the details of the selected plug-in and use the drop-down list next to the plug-in to select the version to be installed.
- For a Git location, click the **Git** tab. In the **URL** field, type the URL that specifies the plug-in location.
- For a system directory, click the **Directory** tab. In the **Directory** field, type the path of the plug-in or click **Browse** to navigate to the location.

To add the feature, click **Finish**.

Add a parameter

All parameters are associated with a feature and provide information about the specific mapping of Cordova and native APIs.

To add a parameter, from the **Features** table select an item for which to create a parameter. For the **Params** table, click **Add**. In the **name** and **value** fields, type the service name and Java class full name (including namespace), respectively. To add the parameter, click **OK**.

Add a preference

Preferences details the global, cross-platform and platform-specific behaviors for the web view of the hybrid mobile application.

To add a preference, click **Add** for the **Preference** table. Complete the **name** and **value** fields as appropriate. To add the parameter, click **OK**.

By default for an application created with the Hybrid Mobile Tools project wizard, the **Preferences** table has two entries. The `fullscreen` and `webviewbounce` elements specify whether the application is fullscreen and bounces when pulled down in iOS devices, respectively. For a full list of available preferences see http://cordova.apache.org/docs/en/latest/config_ref_index.md.html#The%20config.xml%20File on the Apache Cordova website.

Add access

Access entries specify the external network resources to which the application has access, also referred to as whitelisting.

To add an access entry, click **Add** for the **Access** table. In the required **Origin** field, type the URL to which access is granted, using `*` as a wildcard character. Select the **Allow Subdomains** and **Browser Only** check boxes as appropriate. These items enable access to subdomains and cause links to open in browsers rather than the application window, respectively. To add the access entry, click **OK**.

By default for an application created with the Hybrid Mobile Tools project wizard, the **Access** table has an entry allowing access to all networks, `<access origin="*" />`. You are advised to declare access to specific network resources.

Remove a feature, parameter, preference or access

In the appropriate table, select the item to be removed and click **Remove**. Note that removing a feature also removes the associated parameters.

All changes to `config.xml` must be saved before the results take effect. To save, press **Ctrl+S**.

7.2.5. Run a Hybrid Mobile Application on Devices and Simulators

You can use the actions of Hybrid Mobile Tools to run applications on devices and simulators, as detailed below.

Run on an Android device

In the **Project Explorer** view, right-click the project name and click **Run As**→**Run on Android Device**. This option calls the external Android SDK to package the workspace project and run it on an Android device if one is attached. Note that Android APIs and AVDs must be installed and the IDE correctly configured to use the Android SDK for this option to execute successfully.

Run on an Android emulator

In the **Project Explorer** view, right-click the project name and click **Run As**→**Run on Android Emulator**. This option calls the external Android SDK to package the workspace project and run it on the Android emulator. Note that Android APIs and AVDs must be installed and the IDE correctly configured to use the Android SDK for this option to execute successfully.



Important

You must have Android API 17 or later installed on your system to use the **Run on Android Emulator** action. You are prompted by an error message when only Android API 16 or earlier is available. In the event that multiple Android API of 17 and later are available, Hybrid Mobile Tools uses the latest version.

Run on iOS Simulator



Important

This option is only displayed when using OS X operating systems, for which iOS Simulator is available. For information about iOS Simulator see <https://developer.apple.com/xcode/index.php> on the Apple Developer website.

In the **Project Explorer** view, right-click the project name and click **Run As**→**Run on iOS Emulator**. This option calls the external iOS SDK to package the workspace project into an XCode project and run it on the iOS Simulator.

Run with CordovaSim

In the **Project Explorer** view, right-click the project name and click **Run As**→**Run with CordovaSim**. This opens the application in CordovaSim, which is composed of a BrowserSim simulated device and a device input panel.

Figure 7.11. CordovaSim for Samsung Galaxy Nexus Simulated Device

See Also:

- [Section 7.1.4, “System Requirements for Hybrid Mobile Tools”](#)

7.2.6. Manage Hybrid Mobile Project Run Configurations

Run configurations inform simulators how to run the application associated with a project. Hybrid Mobile Tools generates a default run configuration for a project the first time it is run by a specific simulator. This default run configuration is simulator-specific and named according to the project name. You can create and customize multiple run configurations for your projects using the Run Configurations manager.

The information below details how to manage run configurations using the **Run Configurations** manager. To open the **Run Configurations** manager for a project, in the **Project Explorer** view right-click the project name and click **Run As** → **Run Configurations**. Note that run configurations are organized by simulator within the Run Configurations manager, namely CordovaSim, Android and iOS Simulator.

Figure 7.12. A CordovaSim Run Configuration Selected in Run Configurations Manager

Create a run configuration

From the list of run environments, right-click the simulator and click **New**. Complete the fields as appropriate. To save the new run configuration, click **Apply**.

View and edit a run configuration

From the list of run environments, expand the simulator. This shows a list of the run configurations associated with the simulator.

Details for a run configuration are organized in tabs. All simulators have the same **Common** tab. These options include where to save the run configuration information and how standard input and output are managed. Additional customizable options vary according to simulator:

- For Android, you can specify details about the virtual device to be used by the emulator and the values of environment variables. Additionally, you can customize which of the information returned by the Android emulator is shown in the IDE console.
- For CordovaSim, you can customize default values including the location of the root folder containing key device-independent files, the application start page opened when CordovaSim starts, and the server port used by CordovaSim to host the application.

To change the value of any variables listed in the tabs, click the appropriate field and edit the content. To save changes, click **Apply**.

Run an application using a run configuration

From the list of run environments, expand the simulator and select a run configuration. Click **Run**. This starts the simulator, which runs the application associated with the project using the specified configuration settings.

7.2.7. Export a Hybrid Mobile Project

Hybrid Mobile Tools provides actions for exporting workspace projects from the IDE. Projects can be exported as native projects and ready-to-sign applications, as detailed in the procedure below.



Important

Android and iOS APIs must be installed and the IDE correctly configured to use the Android SDK for this procedure to execute successfully.

Procedure 7.3. Export a Hybrid Mobile Project

1. In the **Project Explorer** view, right-click the project name and click **Export**.
2. Expand **Mobile**, select the export type as appropriate and click **Next**:
 - To export as an application, select **Export Mobile Application**.
 - To export as a native project, select **Export Native Platform Project**.

Figure 7.13. Select from the Mobile Export Types in the Export Wizard

3. Complete the following fields:
 - From the **Select Projects** list, select the check boxes of one or more workspace projects to be exported.
 - From the **Select Platforms** list, select the check boxes of one or more operating systems for which you want to export the selected project. Only operating systems with installed SDKs are listed.
 - In the **Directory** field, type the path to which the projects are to be exported or click **Browse** to navigate to the location.

Figure 7.14. Provide Export Settings in the Export Wizard

4. Click **Finish**. Projects are exported to the specified location. Exported native projects are organized with subdirectories for each selected operating system.

See Also:

- [Section 7.1.4, "System Requirements for Hybrid Mobile Tools"](#)

7.3. Customizing Hybrid Mobile Tools and CordovaSim

7.3.1. Customizing Overview

The aim of this section is to guide you in customizing Hybrid Mobile Tools and CordovaSim:

- Specify an Android SDK location

7.3.2. Set the Android SDK Location

You must inform Hybrid Mobile Tools of the Android SDK location before you can use Hybrid Mobile Tools actions involving Android.

To set the Android SDK location, click **Window**→**Preferences** and select **Hybrid Mobile**. In the **Android SDK Directory** field, type the path of the installed SDK or click **Browse** to navigate to the location. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 7.15. Hybrid Mobile Pane of Preferences Window

Part IV. Tools for Deployment and Maintenance

OpenShift Tools

8.1. Overview of OpenShift Tools

8.1.1. About OpenShift

OpenShift is Red Hat's Platform as a Service (PaaS) for applications. It consists of an application platform in the cloud, enabling you to build, test and run applications in a cloud architecture. OpenShift provides disk space, CPU resources, network connectivity, and a runtime environment.

OpenShift is available in three versions: OpenShift Online, Enterprise and Origin. OpenShift Online is the public cloud offering, with free and paid plans, hosted at <https://openshift.redhat.com>. OpenShift Enterprise is the private cloud offering, obtained through a Red Hat OpenShift subscription and hosted in a private data center. OpenShift Origin is the community and local cloud offering, available to download and install locally for development and testing purposes.

OpenShift has a number of key features to assist you in developing and deploying applications:

- Unique domain names, or namespaces, support the hosting of your applications. A user account provides you with access to domains, the latter having the potential to be associated with multiple applications.
- Numerous cartridges give you access to predefined build and runtime environments with popular languages, database and management frameworks. OpenShift can also be extensively customized with the Do-It-Yourself (DIY) cartridge.
- Different sized gears provide RAM and disk space for your applications and cartridges. You can use a set number of small gears as part of OpenShift Online with Free Plan, a free OpenShift user account, and extend to more gears and bigger gears with OpenShift Online with Silver Plan or OpenShift Enterprise.
- Built-in administrative and stack management frees you up to focus on code development. OpenShift manages the intricate details of deploying your application to the stack and interfacing with middleware technologies for you.
- Automatic or manual scaling of the resources supporting your applications ensures that application performance does not suffer as usage increases. OpenShift can create additional instances of your application across more gears and enable clustering.

OpenShift can be accessed via the web interface at <https://www.openshift.com/> on the OpenShift website or via the OpenShift command line interface.

8.1.2. About OpenShift Tools

OpenShift Tools is tooling available within the IDE for OpenShift. It provides an alternative way of accessing OpenShift and managing the development of applications deployed on OpenShift servers.

OpenShift Tools consists of a set of wizards and actions, which together provide core functionality for developing OpenShift applications:

- The tools prepare you for working with OpenShift, by assisting you to create OpenShift user accounts and domains.
- OpenShift Tools assists you with the essential tasks of setting up your system and the IDE for OpenShift interaction, such as creating connections and generating and uploading SSH keys.
- When creating and developing OpenShift applications, OpenShift Tools provides wizards for creating new and importing existing OpenShift applications.
- A variety of actions are available for managing deployed applications, for tasks such as restarting applications, uploading changes to applications, viewing OpenShift server output, editing application cartridges and environment variables, and deleting applications.

8.2. Features of OpenShift Tools

8.2.1. Features Overview

The aim of this section is to guide you in using OpenShift Tools:

- Create OpenShift Online user accounts
- Connect to OpenShift servers
- Generate SSH keys and upload them to an OpenShift user account
- Create and manage domains
- Create new OpenShift applications from within the IDE
- Deploy existing workspace applications to OpenShift and import existing OpenShift applications into a workspace
- Manage deployed OpenShift applications and view information about them

8.2.2. Create an OpenShift Online User Account

To begin using OpenShift Online, you need to create a user account. OpenShift Tools provides the ability to create an OpenShift Online user account from within the IDE.

To create a user account, click the **OpenShift Explorer** view. If the **OpenShift Explorer** view is not visible, click **Window** → **Show View** → **Other**, expand **JBoss Tools** and double-click **OpenShift Explorer**.

Figure 8.1. OpenShift Explorer View

Click the **Connect to OpenShift** icon

and click the link to sign up for an account. This opens <https://openshift.redhat.com/app/account/new> in a browser window. Follow the instructions on the OpenShift web page to create an account. Once created, you can close the browser window.

Figure 8.2. Sign Up Link in Sign in to OpenShift Wizard

Management of your OpenShift Online user account, such as changing or resetting your password, must be carried out through the OpenShift management console at <https://openshift.redhat.com/app/login?redirectUrl=%2Fapp%2Fconsole>.

8.2.3. Connect to OpenShift

Once you have an OpenShift user account, you can connect to OpenShift and then create domains and applications. The procedure below guides you through connecting to OpenShift for the first time in the IDE.

Procedure 8.1. Connect to OpenShift

1. In the **OpenShift Explorer** view, click the **Connect to OpenShift** icon
2. Complete the fields and options as detailed:
 - From the **Connection** list, select **New Connection**.
 - If you want to use a server other than the default at <https://openshift.redhat.com>, clear the **Use default server** check box and in the **Server** field type the address of the server. This option is most relevant when you are using OpenShift Enterprise or Origin servers.
 - In the **Username** and **Password** fields, type your OpenShift user account authentication information.
 - If you want the **Password** field to automatically populate for this connection in future, select the **Save password** check box.



Note

The password is retained in secure storage provided by the IDE. To manage the settings for secure storage, click **Window** → **Preferences**, expand **General** → **Security** and select **Secure Storage**.

Figure 8.3. Sign in to OpenShift Wizard

3. Click **Finish** for OpenShift Tools to connect to OpenShift.
 - a. If your credentials are incorrect, the **Sign in to OpenShift** wizard remains open for you to change your authentication information.
 - b. If you selected for your password to be saved, you are prompted to enter your secure storage password or, if this is your first use of secure storage, you are prompted to set a secure storage password.

Once your credentials are verified as correct, the wizard closes and a live OpenShift connection is listed in the **OpenShift Explorer** view.



Figure 8.4. OpenShift Connection Listed in OpenShift Explorer View

When you close the IDE, any live OpenShift connections will be disconnected but they can be easily reestablished. OpenShift Tools lists previous connections in the **OpenShift Explorer** view until cleared by you. In the **OpenShift Explorer** view, double-click or expand the appropriate connection to open an automatically completed connection wizard. Type your password or, if using the saved password facility, the master password and click **Finish**.

8.2.4. Manage a Connection

Using OpenShift Tools, you can view and manage live OpenShift connections.

View information about a connection

In the **OpenShift Explorer** view, right-click the connection and click **Properties**. The **Properties** view opens and shows information about the associated domains, key and user account. The **Key** parameter is unique to the connection and it is used by the IDE for identification purposes.

Figure 8.5. Properties View

Refresh information about a connection

In the **OpenShift Explorer** view, right-click the connection and click **Refresh**. Information is retrieved from OpenShift and the **OpenShift Explorer** view updated as appropriate. This action is useful if you are simultaneously making changes to your domains and applications in the IDE and the OpenShift web interface or command line interface. Additionally, it may be used to recover from errors.

Delete a connection

In the **OpenShift Explorer** view, right-click the connection and click **Remove Connection**.

8.2.5. Generate and Upload SSH Keys to OpenShift

SSH keys are essential when working with OpenShift. They enable you to develop and access deployed applications. SSH keys are also used to control access of other contributors to your OpenShift applications. SSH keys must be uploaded to the OpenShift server and, as detailed in the procedure below, OpenShift Tools can assist with both the generation and uploading of SSH keys to OpenShift.

Procedure 8.2. Generate and Upload SSH Keys to OpenShift

1. In the **OpenShift Explorer** view, right-click the connection and click **Manage SSH Keys**.
2. To create a new SSH private-public key pair, click **New**.
3. Complete the fields and options as detailed:
 - In the **Name** field, type a name for the key pair that will be used by OpenShift to distinguish this key pair from others associated with your account.
 - From the **Key Type** list, select **SSH_RSA**.
 - Ensure the **SSH2 Home** field contains the location where you want to create the files associated with the key pair. To change the location, clear the **Default** check box and click **Browse** to navigate to the desired location.



Note

The default location for creating SSH key files is determined by the SSH information for the IDE. The default location can be altered by clicking **Windows**→**Preferences**, expanding **General**→**Network Connections**, selecting **SSH2** and changing the location in the **SSH2 home** field of the **General** tab.

- In the **Private Key File Name** field, type a name for the private key file.
- In the **Private Key Passphrase** field, type a passphrase for use in accessing the private key. This field is not mandatory and can be left empty if you want.
- In the **Public Key File Name** field, type a name for the public key file. Typically the file name of the public key is that of the private key with `.pub` appended.

Figure 8.6. New SSH Key Wizard

4. Click **Finish**. The SSH key pair is generated and the public key automatically uploaded to OpenShift.
5. Click **OK** to close the **Manage SSH Keys** window.

8.2.6. Manage SSH Keys

OpenShift Tools provides actions for managing the SSH keys of your OpenShift account.

Upload an existing public SSH key to OpenShift

In the **OpenShift Explorer** view, right-click the connection and click **Manage SSH Keys**. Click **Add Existing**. In the **Name** field, type a name for the key that will be used by OpenShift to distinguish the key from others associated with your account. Click **Browse** to navigate to and select the public key file. Click **Finish** and click **OK** to close the **Manage SSH Keys** window.

You must also inform the IDE of the location of the private key file. Click **Window** → **Preferences**, expand **General** → **Network Connections** and selecting **SSH2**. Click **Add Private Key** and locate the private key file. Click **Apply** and click **OK** to close the **Preferences** window.

Remove a public SSH key from OpenShift

In the **OpenShift Explorer** view, right-click the connection and click **Manage SSH Keys**. From the **SSH Public Keys** table select the key you want to remove from your OpenShift account and click **Remove**. At the prompt asking if you are sure you want to remove the key, click **OK**. Click **OK** to close the **Manage SSH Keys** window.



Note

Remove only disassociates keys with your OpenShift account. The files associated with a 'removed' SSH public-private key pair still exist in the local location where they were generated and can be uploaded again to OpenShift using the **Add Existing** action.

Refresh the SSH key information associated with OpenShift

In the **OpenShift Explorer** view, right-click the connection and click **Manage SSH Keys**. Click **Refresh** and click **OK** to close the **Manage SSH Keys** window. It may be necessary to use this action if you make changes to your OpenShift SSH key settings through the OpenShift web interface while the IDE is open with a live OpenShift connection.

8.2.7. Create a Domain

Once you have an OpenShift user account, you need to create domains in which to host your applications. Note that user accounts for OpenShift Online with Free plan can be associated with one domain only. The procedure below guides you through creating a new domain but you first need a live connection. If you already have a domain associated with your user account then domain information is automatically passed to the IDE when a live connection is started.

Procedure 8.3. Create a Domain

1. In the **OpenShift Explorer** view, right-click the connection and click **New→Domain**. Alternatively, right-click the connection, click **Manage Domains** and click **New**.
2. In the **Domain Name** field, type the name of the domain you would like to use. When the domain is created, the name you provide is appended with the cloud address, for example `.rhcloud.com` for OpenShift Online.
3. Click **Finish**. Domain names must be unique so if the name you have chosen is already in use you will see a warning. In this case, choose another name and try again until you have a unique one.

Figure 8.7. Create Domain Wizard



Note

There are restrictions on the name you can use for a domain. Names must consist only of alphanumeric characters and can have a maximum length of 16 characters.

8.2.8. Manage a Domain

OpenShift Tools provides actions for managing the domains of your OpenShift account.

View the domains associated with a connection

In the **OpenShift Explorer** view, right-click the connection and click **Manage Domains**. Alternatively, right-click the connection and click **Properties**. The **Properties** view opens, where the first row of the table contains the names of the domains associated with the connection.

Rename a domain

In the **OpenShift Explorer** view, right-click the domain and click **Edit Domain**. Alternatively, right-click the connection and click **Manage Domains**. From the **Domains** table, select the domain and click **Edit**. In the **Domain Name** field, type the new name of the domain and click **Finish**. You cannot change the name of a domain which has associated applications.



Important

Renaming your domain changes the public URLs of applications you later create.

Delete a domain

In the **OpenShift Explorer** view, right-click the domain and click **Delete Domain**. Alternatively, right-click the connection and click **Manage Domains**. From the **Domains** table, select the domain and click **Remove**. You cannot delete a domain that has any applications associated with it unless, at the prompt, you select the **Force applications deletion** check box. Click **OK** to complete the deleting action.



Note

Forcing the deletion of applications results in the applications being deleted from the OpenShift server. The projects of applications will still be visible in the **Project Explorer** and **Git Repositories** view as the local clone of the Git repository for projects is not deleted.

8.2.9. Deploy a New or Existing Application on OpenShift

OpenShift Tools provides the **OpenShift Application** wizard to assist you in creating and deploying OpenShift applications.

As detailed in the procedure below, OpenShift applications can be created using three sources: an existing workspace project, a Git source or a default project template. For an existing workspace project, the wizard merges the existing project contents with the key metadata files from a new OpenShift application so that the application can be deployed on OpenShift. For a Git source, the wizard uses the source as the new OpenShift application so the source must be OpenShift-enabled, namely have a `.openshift` directory and have the openshift profile specified in the `pom.xml`. For a project template, the templates are provided by OpenShift.

In addition to deploying your OpenShift applications, the wizard assists you in setting up linked remote (OpenShift server) and local Git repositories containing the original and clone of your project, respectively. You can then push project changes to OpenShift via Git or allow the OpenShift server adapter to do it for you.



Important

You must have SSH keys set up first in order to successfully proceed with the **OpenShift Application** wizard.

Procedure 8.4. Create and Deploy an Application on OpenShift

1. In the **OpenShift Explorer** view, right-click the connection or domain and click **New → Application**. Alternatively, in JBoss Central click **OpenShift Application**, after which you are prompted to select an OpenShift connection and provide your user authentication information.

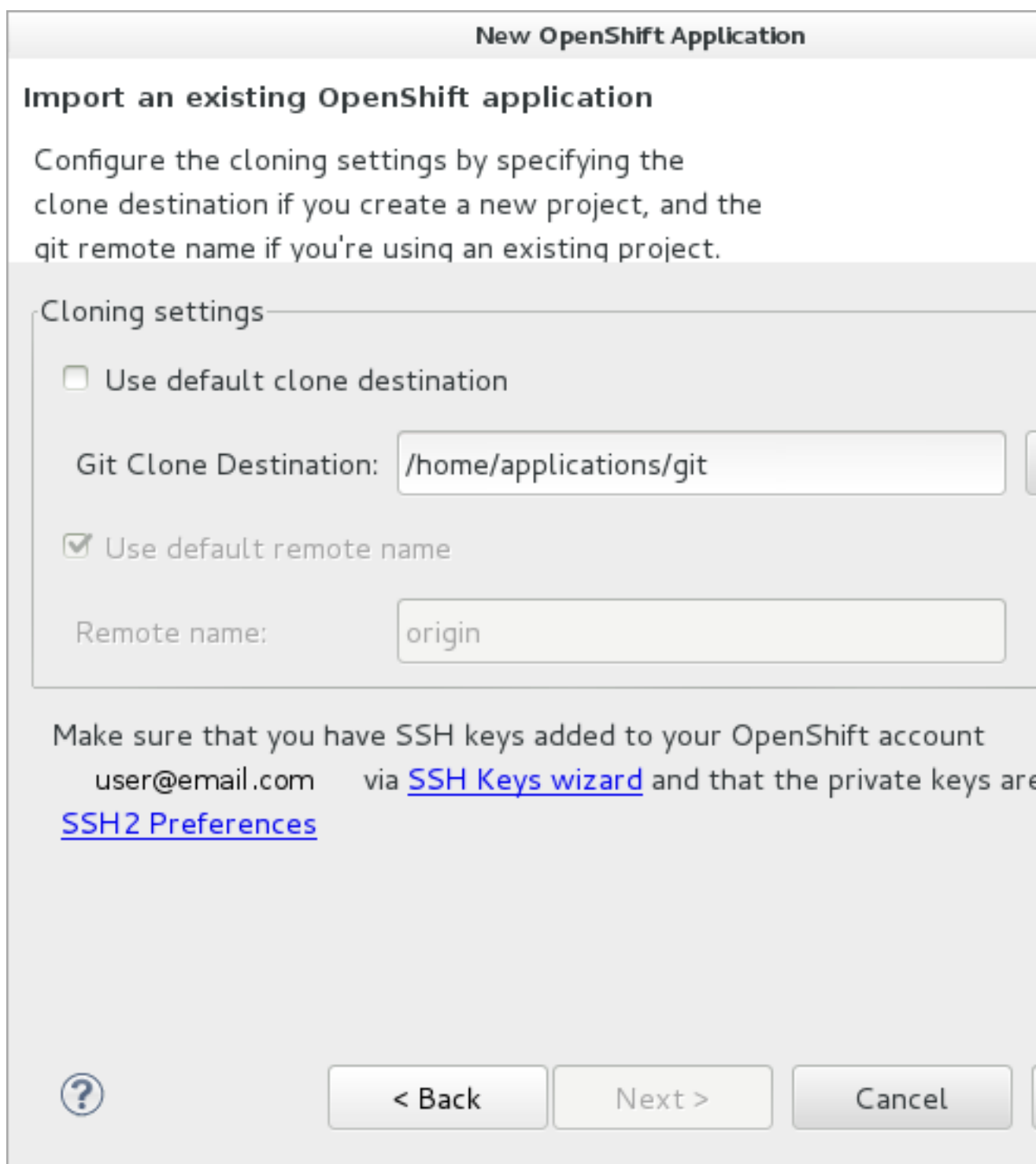
Figure 8.8. OpenShift Application Wizard in JBoss Central

2. If you do not have a domain associated with the connection, the wizard prompts you to create one. In the **Domain Name** field, type the name of the domain you would like to use and click **Finish**. Domain names must be unique so if the name you have chosen is already in use you will see a warning. In this case, choose another name and try again until you have a unique one.
3. Complete the fields and options about the OpenShift application as detailed:
 - From the **Domain** list, select the domain to which to assign the application.
 - In the **Name** field, type a name for the new OpenShift application. There are restrictions on the name you can use for an application. Names must consist only of alphanumeric characters. In the case of an existing workspace project, for simplicity you may choose the OpenShift application name to be the same as the name of the workspace project.
 - From the **Type** list, select a runtime server. This will ensure the necessary core programming or technology cartridge is added to your application.
 - From the **Gear profile** list, select the gear size. This is RAM and disk space required by your applications and its cartridges. If you are using OpenShift Online with Free Plan, you have access to small gears only.
 - If you want OpenShift to automatically increase the instances of your application and enable clustering as usage increases, select the **Enable scaling** check box.
 - From the **Embedded Cartridges** list, select the functionality you want to enable in your application. This will add associated capabilities and support to your application.
 - To specify that the new application is to be based on source code from an existing Git repository, click **Advanced** and clear the **Use default source code** check box. In the **Source code** field, type the URL of the source code location.
 - To declare environment variables to be used when the application is run, click **Advanced** and click **Environment Variables**. Click **Add** to declare an environment variable. In the **Name** and **Value** fields, type a name and value for the environment variable respectively. Click **OK** to save the information and click **OK** to close the **Environment Variables** window.

Figure 8.9. Specify Application Details in the New OpenShift Application Wizard

4. Click **Next**.

5. Complete the fields and options as detailed:
 - To specify that the new application is to be based on an existing workspace project, clear the **Create a new project** check box and in the **Use existing project** field type the name of the project or click **Browse** to locate the project. Otherwise, ensure the **Create a new project** check box is selected.
 - Ensure the **Create and set up a server for easy publishing** check box is selected. This option automatically creates an OpenShift server adapter for the application, enabling you to easily upload changes made in the IDE to the OpenShift server.
 - To disable Maven builds, check the **Disable automatic Maven builds when pushing to OpenShift** check box. This informs OpenShift not to launch the Maven build process when the Maven project is pushed to OpenShift but to put the deployment straight into the deployments folder. It is useful when you want to push applications already built for deployment rather than source code.
6. Click **Next**.
7. The **Git Clone Destination** field shows the location to be used for the local Git repository. The location must already exist to be able to proceed with the wizard. To change the location, clear the **Use default clone location** check box and type the location in the **Git Clone Destination** field or click **Browse** to navigate to the location.



New OpenShift Application

Import an existing OpenShift application

Configure the cloning settings by specifying the clone destination if you create a new project, and the git remote name if you're using an existing project.

Cloning settings

☐ Use default clone destination

Git Clone Destination:

☒ Use default remote name

Remote name:

Make sure that you have SSH keys added to your OpenShift account `user@email.com` via [SSH Keys wizard](#) and that the private keys are [SSH2 Preferences](#)




Figure 8.10. Specify Repository Details in the New OpenShift Application Wizard

8. Click **Finish**. If you are prompted that the authenticity of the host cannot be established and asked whether you want to continue connecting, check that the host name matches that of your application and domain and click **Yes**.
9. At the prompt asking if you want to publish committed changes to OpenShift, click **Yes**. The progress of the application creation process is visible in the **Console** view.

Once created, the application is listed under the connection and domain in the **OpenShift Explorer** view. The application type proceeds the application name. The project is also listed in the **Project Explorer** and **Git Repositories** views, where the details proceeding the application name indicate the current Git branch and status compared to the remote repository. Additionally, the server adapter for the application is visible in the **Servers** view.



Note

To view the project in the **Git Repositories** view, in the **Project Explorer** view right-click the project name and click **Team** → **Show in Repositories View**. Alternatively, click **Window** → **Show View** → **Other**, expand **Git** and double-click **Git Repositories**.

8.2.10. Import a Deployed OpenShift Application into the IDE

All applications deployed on OpenShift are listed under live connections in the **OpenShift Explorer** view. But only the project files of OpenShift applications created through the IDE are immediately available in the **Project Explorer** and **Git Repositories** views. If you want to work on the project files associated with an application, you must first import the application. OpenShift Tools can assist you to import your deployed OpenShift applications into the IDE, as detailed in the procedure below.

Procedure 8.5. Import an OpenShift Application

1. Click **File** → **Import**, expand **OpenShift** and double-click **Existing OpenShift Application**. Alternatively, in the **OpenShift Explorer** view, right-click the application and click **Import Application**.
2. Complete the fields and options as detailed:
 - From the **Domain** list, select the domain of the application.
 - Ensure the **Use existing application** check box is selected and type the name of the application in the text field. This field has an automatic completion feature to assist you in typing the application name or click **Browse** to see a list of all of your applications associated with the domain.



Important

Project names in the IDE workspace must be unique. If the name of the application you want to import is identical to an existing project in the workspace, the OpenShift Tools will not complete the import. To work around this constraint, you can import the OpenShift application to another workspace or change the name of either the conflicting project or application.

Figure 8.11. Import OpenShift Application Wizard

3. Click **Next**.
4. Complete the fields and options as detailed:
 - Ensure the **Create a new project** check box is selected. This option creates a new project in your IDE workspace for the existing OpenShift application.
 - Ensure the **Create and set up a server for easy publishing** check box is selected. This option automatically creates an OpenShift server adapter for the application, enabling you to easily upload changes made in the IDE to the OpenShift server.
 - To disable Maven builds, check the **Disable automatic Maven builds when pushing to OpenShift** check box. This informs OpenShift not to launch the Maven build process when the Maven project is pushed to OpenShift but to put the deployment straight into the deployments folder. It is useful when you want to push applications already built for deployment rather than source code.
5. Click **Next**.
6. The **Git Clone Destination** field shows the location to be used for the local Git repository. The location must already exist to be able to proceed with the wizard. To change the location, clear the **Use default clone location** check box and type the location in the **Git Clone Destination** field or click **Browse** to navigate to the location.
7. Click **Finish**. If you are prompted that the authenticity of the host cannot be established and asked whether you want to continue connecting, check that the host name matches that of your application and domain and click **Yes**.
8. OpenShift Tools modifies the .gitignore file on importing the application. At the prompt asking if you want to publish committed changes to OpenShift, click **Yes**. The progress of the import process is visible in the **Console** view.

Once imported, the project is listed in the **Project Explorer** and **Git Repositories** views, where the details proceeding the application name indicate the current Git branch and status compared to the remote repository. Additionally, the server adapter for the application is visible in the **Servers** view.

8.2.11. Generate a Server Adapter for an Application

In order to easily publish changes to a deployed OpenShift application, each application needs a server adapter. The **OpenShift Application** wizard can automatically generate server adapters for new or imported OpenShift applications if you select the **Create and set up a server for easy publishing** check box. But OpenShift also provides an action to assist you in generating server adapters for OpenShift application that already exist in the IDE, as detailed in the procedure below. You can use this action if you need to regenerate a deleted server adapter for an OpenShift application or if you create or import an OpenShift application and do not select the **Create and set up a server for easy publishing** check box.

Procedure 8.6. Generate a server adapter for an application

1. In the **OpenShift Explorer** view, right-click the application and click **Create a Server Adapter**.
2. Complete the fields and options as detailed:
 - From the list of server types, expand **OpenShift** and select **OpenShift Server**.
 - The **Server's host name** and **Server name** field are automatically completed. The **Server's host name** field contains the host name of the server and the **Server name** field contains the name by which the server adapter is known in the **Servers** view. You can edit these values as appropriate by typing in the fields.

Figure 8.12. OpenShift Server Selected in New Server Wizard

3. Click **Next**.
4. Complete the fields and options as detailed:
 - Ensure the **Connection**, **Domain Name**, **Application Name** and **Deploy Project** fields contain the correct information relating to the application for which you want to generate the server adapter.
 - In the **Remote** field, type the alias for the remote Git repository. For OpenShift Online applications this is `origin`.
 - In the **Output Directory** field, type the location where archived projects for deployment are to be stored or click **Browse** to navigate to the location.

Figure 8.13. Server Details in New Server Wizard

5. Click **Next**.
6. From the **Available** list, select the project for which the server adapter is being generated and click **Add**. The application is now listed under **Configured**.
7. Click **Finish** for OpenShift Tools to generate the server adapter. Once generated, the server adapter is listed in the **Servers** view.

Figure 8.14. OpenShift Server Adapter Listed in Servers View

8.2.12. View a Deployed Application and Associated Information

OpenShift Tools provides actions for viewing deployed OpenShift applications and information about them.

View a deployed application

In the **OpenShift Explorer** view, right-click the application and click **Web Browser**. A browser tab opens displaying your deployed application. Alternatively, in the **Servers** view, right-click the server adapter for the application and click **Show In → Web Browser**.

View information about an application

In the **OpenShift Explorer** view, right-click the application and click **Details**. The displayed information includes the public URL of the application, application type, and remote Git repository location. Click **OK** to close the **Details** window.

Figure 8.15. Application Details Window

View output from the OpenShift server

In the **OpenShift Explorer** view, right-click the application and click **Tail files**. Alternatively, in the **Servers** view right-click the server adapter of the application and click **OpenShift → Tail files**. The **Tail Log Files** window opens, with either the default retrieval syntax or last used syntax for this application in the **Tail options** field.

To change the retrieval command, in the **Tail options** field type the appropriate syntax. To specify the gears for which to show the server logs, from the table select the check boxes of the appropriate gears. Click **Finish** for OpenShift to retrieve the output, which is displayed in a distinct **Console** view for each gear.

Figure 8.16. Default Retrieval Syntax in Tail Options Field

View values of variables associated with an application

In the **OpenShift Explorer** view, right-click the application and click **All Environment Variables**. Variable names and values are listed in the **Console** view. Alternatively, in the **Servers** view, right-click the server adapter of the application and click **OpenShift → All Environment Variables**.

Figure 8.17. Environment Variables Listed in Console View

View properties of cartridges associated with an application

In the **OpenShift Explorer** view, right-click the cartridge and click **properties**. The **Properties** view opens and lists information about the cartridge.

View information about the server of an application

In the **Servers** view, double-click the server adapter for the application. A Server Editor opens, enabling viewing and editing of server details. To save any changes, press **Ctrl+S** or click **File → Save** or click the **Save** icon.

Refresh information about an application

In the **OpenShift Explorer** view, right-click the connection, domain, application or cartridge and click **Refresh**. Information is retrieved from OpenShift and the **OpenShift Explorer** view is updated as appropriate. This action is useful if you are simultaneously making changes in the IDE and the OpenShift web interface or command line interface to your domain and applications. Additionally, it may be used to recover from errors.

8.2.13. Manage a Deployed Application

OpenShift Tools provides actions for developing and managing deployed OpenShift applications.

Upload modifications to a deployed application

In the **Severs** view, right-click the server adapter for the application and click **Publish**. At the prompt asking if you want to publish to OpenShift by committing changes to Git, you can customize the default commit message `Commit from JBoss Tools`. Click **Yes** and changes, together with the commit message, are pushed to the remote Git repository. Additionally, the application is automatically updated on the OpenShift server and the **Console** view displays OpenShift server output.



Note

To view a log of changes to the local git repository, in the **Git Repositories** view, right-click a repository and click **Show In → History**. The **History** view opens, showing a log of commits for the local Git repository.

Edit environment variables associated with an application

In the **OpenShift Explorer** view, right-click the application and click **Edit Environment Variables**. Click **Add**, **Edit** or **Remove** to customize the environment variables. Click **Finish** to close the window.

Add or remove markers associated with an application

In the **Project Explorer** view, right-click the application and click **OpenShift→Configure Markers**. Select or clear the check boxes of markers as desired. Information about markers is given in the **Marker Description** section of the **Configure OpenShift Markers Window**. Click **OK** for your marker choice to be applied to the application.

Figure 8.18. Configure OpenShift Markers Window

Add or remove cartridges associated with an application

In the **OpenShift Explorer** view, right-click the application and click **Edit Embedded Cartridges**. Select or clear the check boxes of cartridges as desired. Click **Finish** for your cartridge choice to be applied to the application. You are prompted if the cartridges you have chosen to add or remove require further action, such as the addition of prerequisite cartridges or removal of conflicting cartridges. You can choose to ignore or apply the suggestions of the prompt.

Figure 8.19. Edit Embedded Cartridges Window

Restart an application

In the **OpenShift Explorer** view, right-click the application and click **Restart Application**. Alternatively, in the **Servers** tab right-click the server adapter of the application and click **OpenShift→Restart Application**.

Forward remote ports

You can forward the remote ports of the OpenShift server to your system to enable access to various services, such as MySQL. Port forwarding is available for all OpenShift applications, including scalable ones.



Important

Your application must be running before attempting to configure port forwarding.

In the **OpenShift Explorer** view, right-click the application and click **Port forwarding**. Alternatively, in the **Servers** view right-click the server adapter of the application and click **OpenShift→Port forwarding**.

Figure 8.20. Application Port Forward Window

After checking the authenticity of SSH keys, the **Application port forward** window opens. Before commencing port forwarding, there are a number of options you can set:

- By default, the local address is 127.0.0.1. If this is unavailable, a random available address is allocated. To set the local address to be the same as the remote address, clear the **Use '127.0.0.1' as the local address for all Services** check box.
 - By default, the local port numbers are the same as the remote port numbers. To set independent local port numbers, select the **Find free ports for all Services** check box.
- To commence port forwarding, click **Start All**. Click **OK** to close the **Application port forward** window.

Delete a server adapter for an OpenShift application

In the **Servers** view, right-click the server adapter for the application and click **Delete**. At the prompt asking if you are sure you want to delete the server adapter, click **OK**.

Delete an application

In the **OpenShift Explorer** view, right-click the application and click **Delete Application**. At the prompt asking if you are sure you want to destroy the application, select **OK**. The progress of the deleting process is shown in the activity bar in the lower right of the IDE window. To open the **Progress** view and see more detailed progress information or cancel the deleting process, double-click on the activity bar.



Note

Deleting applications results in the applications being deleted from the OpenShift server. The projects of applications are still be visible in the **Project Explorer** and **Git Repositories** view as the local Git repository copies of projects are not deleted. Additionally, any server adapters for deleted OpenShift applications are still listed in the **Servers** view but they are invalid.

8.3. Customizing OpenShift Tools

8.3.1. Customizing Overview

The aim of this section is to guide you in customizing OpenShift Tools:

- Specify the timeout behavior for OpenShift requests

8.3.2. Change the Timeout Behavior of OpenShift Requests

You may find that some requests made to OpenShift require a long time to complete and do not finish within the IDE default timeout limit of 120 seconds. For example, some of the OpenShift

quickstarts take a long time to checkout the associated large source code. To resolve the timeout restriction, you can modify the default timeout limit to meet your requirements.

To modify the timeout limit, click **Window**→**Preferences**, expand **JBoss Tools** and select **OpenShift**. In the **Remote requests timeout** field, type the required timeout limit in seconds. Click **Apply** and click **OK** to close the **Preferences** window.

Figure 8.21. Set Timeout Behavior in OpenShift Pane of Preferences Window

Appendix A. Revision History

Revision History

Revision 1.0.0-1

Thu Jan 09 2014

MichelleMurray<robot@dev.null.com>

Built from Content Specification: 22477, Revision: 574674 by mmurray

