



Red Hat CodeReady Studio 12.15

Getting Started with CodeReady Studio Tools

Introduction to using Red Hat CodeReady Studio tools

Red Hat CodeReady Studio 12.15 Getting Started with CodeReady Studio Tools

Introduction to using Red Hat CodeReady Studio tools

Levi Valeeva
lvaleeva@redhat.com

Supriya Takkhi
sbharadw@redhat.com

Yana Hontyk
yhontyk@redhat.com

Legal Notice

Copyright © 2020 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

This compilation of topics contains information on how to start using Red Hat CodeReady Studio Tools for efficient development.

Table of Contents

CHAPTER 1. GIT BASICS IN CODEREADY STUDIO	4
1.1. SETTING UP GIT PERSPECTIVE	4
1.2. MANAGING REPOSITORIES IN GIT PERSPECTIVE	4
1.2.1. Creating a new Git repository	4
1.2.2. Adding an existing local Git repository	5
1.2.3. Cloning an existing Git repository	6
1.2.4. Adding a remote for the repository	6
1.3. MANAGING BRANCHES IN GIT PERSPECTIVE	7
1.3.1. Creating a new branch	7
1.3.2. Working in the branch	8
1.3.3. Submitting a pull request	8
1.4. COMMITTING AND PUSHING THE CHANGES	8
CHAPTER 2. MAVEN BASICS IN CODEREADY STUDIO	10
2.1. CREATING A NEW MAVEN PROJECT	10
2.2. IMPORTING EXISTING MAVEN PROJECTS	11
2.2.1. Importing the existing locally stored Maven project	11
2.2.2. Importing the existing remotely stored Maven project	12
2.3. CREATING A NEW MAVEN MODULE	12
2.4. ADDING A MAVEN DEPENDENCY TO THE MAVEN PROJECT	14
2.5. ADDING MAVEN SUPPORT TO AN EXISTING NON-MAVEN PROJECT	14
2.6. ADDITIONAL RESOURCES	14
CHAPTER 3. APPLICATION DEPLOYMENT IN CODEREADY STUDIO	15
3.1. CONFIGURING A LOCAL SERVER	15
3.2. CONFIGURING A REMOTE SERVER	16
3.3. DEPLOYING AN APPLICATION	17
CHAPTER 4. JBOSS EAP AND JBOSS WFK BASICS IN CODEREADY STUDIO	18
4.1. CONFIGURING MAVEN REPOSITORIES	18
4.2. SETTING UP JBOSS EAP	19
4.3. VERIFYING THAT JBOSS EAP AND CODEREADY STUDIO ARE WORKING CORRECTLY	20
CHAPTER 5. OPENSIFT BASICS IN CODEREADY STUDIO	22
5.1. SETTING UP OPENSIFT APPLICATION EXPLORER VIEW	22
5.2. CONNECTING TO THE OPENSIFT CLUSTER USING OPENSIFT APPLICATION EXPLORER	22
5.3. CREATING A NEW LAUNCHER PROJECT	23
5.4. CREATING A NEW PROJECT USING OPENSIFT APPLICATION EXPLORER	24
5.5. CREATING A NEW COMPONENT USING OPENSIFT APPLICATION EXPLORER	24
5.6. DEPLOYING A COMPONENT ON CLUSTER USING OPENSIFT APPLICATION EXPLORER	25
5.7. DEFINING AN EXTERNAL ACCESS URL USING OPENSIFT APPLICATION EXPLORER	25
5.8. DEBUGGING AN APPLICATION ON THE CLUSTER USING OPENSIFT APPLICATION EXPLORER	26
CHAPTER 6. QUARKUS TOOLS BASICS IN CODEREADY STUDIO	27
6.1. CREATING A NEW QUARKUS PROJECT	27
6.2. RUNNING A QUARKUS APPLICATION	28
6.3. DEBUGGING A QUARKUS APPLICATION	28
6.4. USING LANGUAGE SUPPORT IN CODEREADY STUDIO	29
6.4.1. Language support	29
6.4.2. Using Quarkus code completion	29
6.4.3. Enabling language support for MicroProfile	30
CHAPTER 7. HIBERNATE TOOLS BASICS IN CODEREADY STUDIO	32

7.1. CREATING A NEW JPA PROJECT	32
7.2. ADDING LIBRARIES	34
7.3. GENERATING ENTITIES	35
7.4. CREATING A HIBERNATE MAPPING FILE	35
7.5. CREATING A HIBERNATE CONFIGURATION FILE	36
7.6. CREATING A HIBERNATE CONSOLE CONFIGURATION FILE	36
7.7. EDITING HIBERNATE PROJECT CONFIGURATIONS	37
CHAPTER 8. NODE.JS BASICS IN CODEREADY STUDIO	39
8.1. CREATING A NEW JAVASCRIPT PROJECT	39
8.2. CREATING A PACKAGE.JS FILE	40
8.3. CREATING A NEW .JS FILE	40
8.4. IMPORTING AN EXISTING NODE.JS APPLICATION	41
8.5. RUNNING A NODE.JS APPLICATION	41
8.6. DEBUGGING A NODE.JS APPLICATION	41
CHAPTER 9. MOBILE WEB TOOLS BASICS IN CODEREADY STUDIO	43
9.1. CREATING AN HTML5 PROJECT	43
9.2. ADDING A NEW HTML5 JQUERY MOBILE FILE	44
9.3. ADDING A NEW MOBILE PAGE	45

CHAPTER 1. GIT BASICS IN CODEREADY STUDIO

CodeReady Studio includes Git Perspective that allows developers to manage their Git repositories from a graphical interface. The following section outlines the basic workflow of a Git project in Git Perspective and how to accomplish the most common Git-related tasks.

The following section describes how to:

- Set up Git Perspective.
- Manage repositories in Git Perspective.
- Manage branches in Git Perspective.
- Manage commits in Git Perspective.

1.1. SETTING UP GIT PERSPECTIVE

The following section describes how to open Git Perspective in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Perspective** → **Open Perspective** → **Other**.
The **Open Perspective** window appears.
3. Select **Git**.
4. Click the **Open** button.
The **Git Repositories** view appears.

1.2. MANAGING REPOSITORIES IN GIT PERSPECTIVE

The following section describes how to use Git Perspective to:

- Create a new Git repository.
- Add an existing local Git repository.
- Clone an existing Git repository.
- Add a remote for the repository.

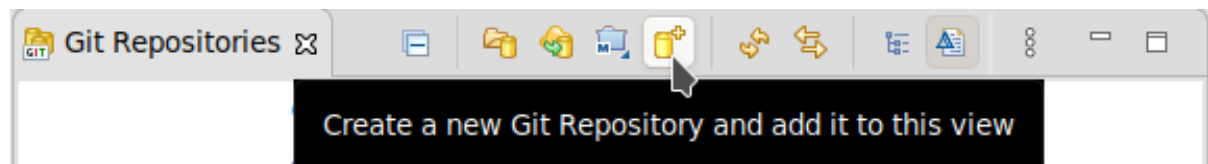
1.2.1. Creating a new Git repository

The following section describes how to use Git Perspective to create a new Git repository.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Click the **Create a new Git Repository and add it to this view** icon.

Figure 1.1. Click the Create a New Git Repository button



The **Create a Git Repository** window appears.

A path to the default **Repository directory** is generated. If you are satisfied with the default path, continue with the repository creation. Alternatively, you can select the option to **Create as bare repository**.



NOTE

Bare repositories are recommended for central repositories, not for development environments. Bare repositories do not contain a working or checked out copy of any source files. This prevents editing files and committing changes. Additionally, they store the Git revision history for your repository in the root folder instead of a **.git** sub-folder.

4. Click the **Create** button.

A new Git repository is created on your local machine and is now listed in the **Git Repositories** view.

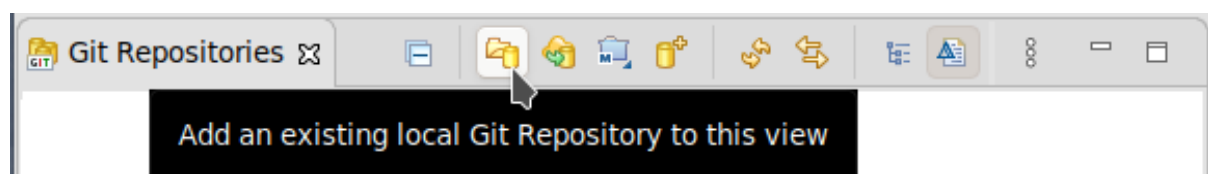
1.2.2. Adding an existing local Git repository

The following section describes how to use Git Perspective to add a local Git repository to the IDE.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Click the **Add an existing local Git Repository to this view** icon.

Figure 1.2. Click the Add an existing local Git Repository icon



The **Add Git Repositories** window appears.

4. Click the **Browse** button to locate your local Git repository.
Optionally, select the **Look for nested repositories** check box to search for nested repositories.
5. Check the path to the appropriate **.git** file in the **Search results** field.
6. Click the **Add** button

The local repository is now listed in the **Git Repositories** view.

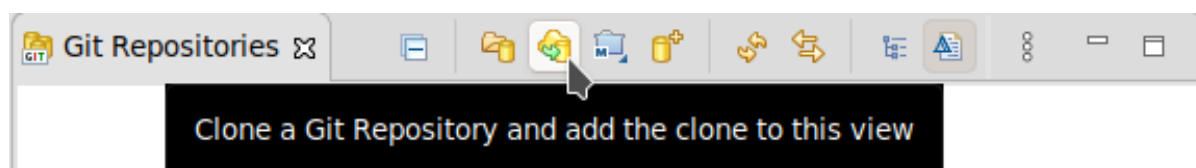
1.2.3. Cloning an existing Git repository

The following section describes how to use Git Perspective to create a local clone of the repository that already exists online (GitHub, GitLab).

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Clone the source repository with HTTPS or SSH.
4. Click the **Clone a Git Repository and add the clone to this view** icon.

Figure 1.3. Click the Clone a Git Repository icon



The **Clone Git Repository** window appears.

5. Add the address for the source repository to the **URI** field.
The **Host** and **Repository path** fields are populated automatically.
6. Click the **Next** button.
7. Select the branches you want to clone.
8. Click the **Next** button
9. Ensure that the **Directory** path and **Initial branch** are set correctly.
Optionally, you can customize the **Remote name** and choose the appropriate working sets by selecting the **Add project to working sets** check box.
10. Click the **Finish** button.

The new cloned repository is now listed in the **Git Repositories** view.

1.2.4. Adding a remote for the repository

After setting up your repository in Git Perspective for the first time, set up a remote for the repository. This is a one-time set up step for newly created or added repository.

The following section describes how to use Git Perspective to set up the remote for your repository.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Expand the target repository in the **Git Repositories** view.

4. Right-click **Remotes** → **Create Remote**.
The **New Remote** window appears.
5. Name your remote.
6. Ensure that the **Configure push** is selected.
7. Click the **Create** button.
The **Configure Push** window appears.
8. Click the **Change** button.
The **Select a URI** window appears.
9. Add the URI, username and password for the source repository.
The **Host** and **Repository** path fields are populated automatically.
10. Click **Finish** → **Save**.

Verification steps

- Expand the target **repository** → **Remotes** in the **Git Repositories** view. Your newly added remote is now listed in there.

1.3. MANAGING BRANCHES IN GIT PERSPECTIVE

The following section describes how to use Git Perspective to:

- Create a new branch.
- Work in the new branch.
- Submit a pull request.

1.3.1. Creating a new branch

The following section describes how to use Git Perspective to create a new branch.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Expand the target **repository** → **Branches** → **Remote Tracking** → **origin** to view all the remote branches.
4. Right-click **master** → **Create Branch...**
The **Create Branch** window appears.
5. Select the source of the new branch.
6. Name the branch.
7. Select the **Configure upstream for push and pull** and **Checkout new branch** check boxes.
8. Select the appropriate option in the **When pulling** field.

9. Click the **Finish** button.

Verification

- Expand the target **repository** → **Branches** → **Local** in the **Git Repositories** view. Your newly added branch is now listed there.

1.3.2. Working in the branch

The following section describes how to open a built-in terminal in Git Perspective.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Press **Shift+Ctrl+Alt+T**.
The **Launch Terminal** window appears.
4. Choose **Local Terminal**.
5. Set **Encoding** to **Default (ISO-8859-1)**.
6. Click the **OK** button.
The **Terminal** window now displays the command-line terminal.

Note that as default, the current working directory is the home directory of your current user.

1.3.3. Submitting a pull request

It is strongly recommended to update your local repository before merging your changes, especially when working in a shared repository.

The following section describes how to use Git Perspective to submit a pull request (PR).

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Right-click the target **repository** → **Pull**.
The **Pull Request** window appears.
4. Click the **Close** button.

Now the changes from the remote repository are merged into your local repository.

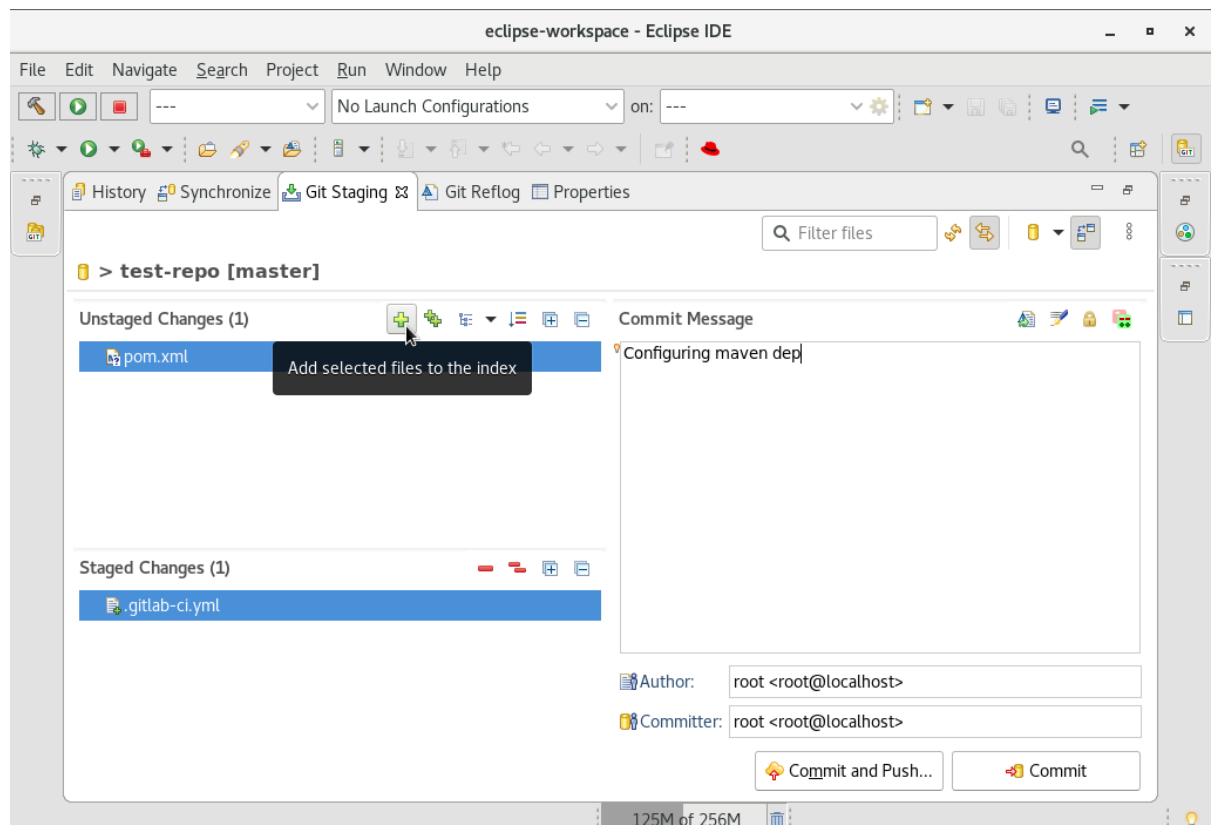
1.4. COMMITTING AND PUSHING THE CHANGES

The following section describes how to commit and push the changes in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Right-click on the target **repository** → **Commit**.
The **Git Staging** view appears.
4. Select the changes you want to stage.
5. Click the **Add selected files to the index** button to stage the changes you want to commit.

Figure 1.4. Stage files to commit



6. Add a commit message to the **Commit Message** field.
Author and **Committer** fields are populated automatically.
7. Click the **Commit** button to commit your changes, or the **Commit and Push** button to commit your changes and push them to the remote repository.
Note that when selecting the **Commit and Push** option you are prompted to enter the repository address, and your access username and password for that repository.

CHAPTER 2. MAVEN BASICS IN CODEREADY STUDIO

In the context of application development, Maven provides a standardized build system for projects, and facilitates fetching dependencies from one or more repositories.

Root Maven projects can serve as aggregators for multiple Maven modules (sub-projects). For each module that is part of a maven project, a `<module>` entry is added to the project's `pom.xml` file. A `pom.xml` contains `<module>` entries is often referred to as an **aggregator pom**.

When modules are included into a project it is possible to execute Maven goals across all of the modules by a single command issued from the parent project directory.

The following section describes how to:

- Create a new Maven project.
- Import existing Maven projects.
- Create a new Maven module.
- Add a Maven dependency to the Maven project.
- Add Maven support to a non-maven project.

Prerequisites

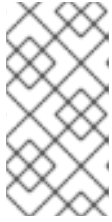
- Set up Git Perspective.
For more information on how to set up Git Perspective, see [Section 1.1, "Setting up Git Perspective"](#).

2.1. CREATING A NEW MAVEN PROJECT

The following section describes how to create a new Maven project in CodeReady Studio. The instructions provided ensure that the packaging option is set to **pom**, which is a requirement for multi-module Maven projects. Alternately, to create a standalone Maven project instead, set the packaging option to an option other than **pom**.

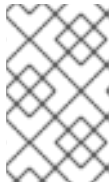
Procedure

1. Start CodeReady Studio.
2. Open Git Perspective.
3. Click **File** → **New** → **Other**.
The **New** window appears.
4. Enter **Maven Projects** in the **Wizards** field.
5. Select **Maven Projects**.
6. Click the **Next** button.
The **New Maven Project** window appears.
7. Select the **Create a simple project** checkbox.

**NOTE**

By selecting the **Create a simple project** check box you are skipping the archetype selection and the project type is automatically set to Project Object Model (POM). To create a standalone project, clear the **Create a simple project** check box and follow the onscreen instructions.

8. Click the **Browse** button to select the workspace location.
9. Click the **Next** button.
10. Enter the group ID and the artifact ID.

**NOTE**

The values cannot include spaces or special characters. The only special characters allowed are periods (.), underscores (_), and dashes (-). An example of a typical group ID or artifact ID is **org.company-name_project-name**.

Optionally, you can name your project and give it a description.

11. Set **Packaging** to **pom**.
12. Click the **Finish** button.

Verification steps

1. Click **Window** → **Show View** → **Other**.
The **Show view** window appears.
2. Enter **Project Explorer** in the search field.
3. Select **Project Explorer**.
4. Click the **Open** button.
The **Project Explorer** view appears.

Your newly created Maven is now listed in the **Project Explorer** view.

2.2. IMPORTING EXISTING MAVEN PROJECTS

The following section describes how to import existing Maven projects into CodeReady Studio.

2.2.1. Importing the existing locally stored Maven project

The following section describes how to import the existing locally stored Maven project to CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open Git Perspective.

3. Open Project Explorer.
4. Click **File** → **Import**.
The **Import** window appears.
5. Enter **Existing Maven Projects** in the **Select an import wizards** field.
6. Select **Existing Maven Projects**
7. Click the **Next** button.
8. Click the **Browse** button to locate your Maven project.
9. Click the **Finish** button.

Your local Maven project is now listed in the **System Explorer** view.

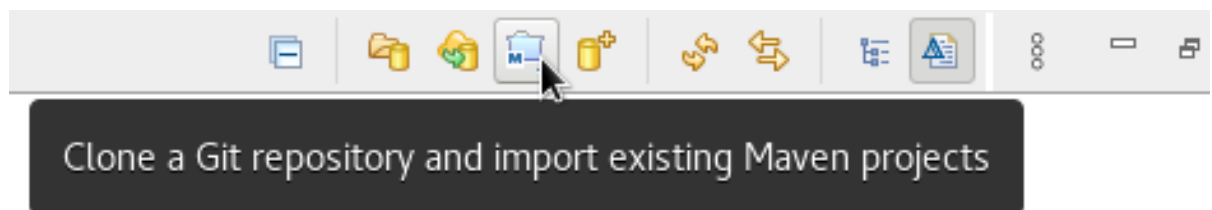
2.2.2. Importing the existing remotely stored Maven project

The following section describes how to import the existing remotely stored Maven project to CodeReady Studio.

Procedure

1. Clone the Clone the source repository with HTTPS or SSH.
2. Start CodeReady Studio.
3. Open Git Perspective.
4. Click the **Clone a Git repository and import existing Maven projects** icon.

Figure 2.1. Importing the existing Maven project



The **Check out as Maven project from SCM** window appears.

5. Add the address for the source repository you cloned during step one to the **SCM URL** field.
6. Click the **Next** button.
7. Click the **Browse** button to select the workspace location.
8. Click the **Finish** button.

Your remote Maven project is now listed in the **System Explorer** view and the **Git Perspective** view.

2.3. CREATING A NEW MAVEN MODULE

The following section describes how to create a new Maven module.

Prerequisites

- An existing Maven project.
For more information on how to create a Maven project, see [Section 2.1, “Creating a new Maven project”](#).

Procedure

1. Start CodeReady Studio.
2. Open Git Perspective.
3. Click **File** → **New** → **Other**.
The **New** window appears.
4. Enter **Maven Module** in the **Wizards** field.
5. Select **Maven Module**.
6. Click the **Next** button.
The **New Maven Module** window appears.
7. Select the **Create a simple project** check box.



NOTE

By selecting the **Create a simple project** check box you are skipping the archetype selection and the project type is automatically set to Project Object Model (POM). To create a standalone module, clear the **Create a simple project** check box and follow the onscreen instructions.

8. Name your module.
9. Click the **Browse** button to select the parent project.
10. Click the **Next** button.
11. Set **Packaging** to **pom**.
Optionally, you can name your module and give it a description.
12. Click the **Finish** button.

Verification steps

1. Click **Window** → **Show View** → **Other**.
The **Show view** window appears.
2. Enter **Project Explorer** in the search field.
3. Select **Project Explorer**.
4. Click the **Open** button.
The **Project Explorer** view appears.

Your newly created Maven module is now listed in the **System Explorer** view.

2.4. ADDING A MAVEN DEPENDENCY TO THE MAVEN PROJECT

The following section describes how to add a Maven dependency to the Maven project.

Prerequisites

- An existing Maven project.
For more information on how to create a Maven project, see [Section 2.1, “Creating a new Maven project”](#).

Procedure

1. Start CodeReady Studio.
2. Open Git Perspective.
3. Open Project Explorer.
4. Right-click the target **project** → **Maven** → **Add Dependency**.
5. Enter the group ID or the artificial ID in the **Enter groupId, artificialId or sha1 prefix or pattern** field.
The fields above are populated automatically.
6. Click the **OK** button.

The dependency is now added to the **pom.xml** file of your target project.

2.5. ADDING MAVEN SUPPORT TO AN EXISTING NON-MAVEN PROJECT

The following section describes how to add Maven support to an application created without Maven support.

1. Start CodeReady Studio.
2. Open Git Perspective.
3. Open Project Explorer.
4. Right-click the target **project** → **Configure** → **Convert to Maven Project**
The **Create a new POM** window appears.

All fields are populated automatically. If you want to change the group ID or the Artificial ID, note that the values cannot include spaces or special characters. The only special characters allowed are periods (.), underscores (_), and dashes (-).

5. Click the **Finish** button.
Your newly generated **pom.xml** file appears in the **System Explorer** view.

2.6. ADDITIONAL RESOURCES

- For more information on how to use Maven, see the [JBoss Community Archive](#) .

CHAPTER 3. APPLICATION DEPLOYMENT IN CODEREADY STUDIO

In order to deploy applications to a server from within the IDE, you must configure the IDE with information about the server. For a local server this information includes the following:

- A server runtime environment with details about the server location, runtime JRE, and configuration files
- A server adapter with management settings for the server runtime environment, including access parameters, launch arguments, and publishing options

JBoss Server Tools enables you to efficiently configure a local server ready for use with the IDE using Runtime Detection. As demonstrated here, this feature is useful for quickly configuring a server for deploying and testing an application.

The following section describes how to:

- Configure a local server.
- Configure a remote server.
- Deploy your application.

3.1. CONFIGURING A LOCAL SERVER

Runtime Detection searches a given local system path to locate certain types of runtime servers. For any servers found, Runtime Detection automatically generates both a default server runtime environment and a default server adapter. These items can be used for immediate application deployment as is or they can be customized to meet your requirements.

The following section describes how to configure a local server in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.
The **Preferences** window appears.
3. Enter **JBoss Tools** in the search field.
4. Select **JBoss Runtime Detection**
The **JBoss Runtime Detection** window appears.
5. Click the **Add** button.
6. Locate the directory containing the runtime server.
7. Click the **OK** button.
8. Click the **Apply and Close** button.

Verification steps

1. Click **Window** → **Show View** → **Other**.
The **Show View** window appears.
2. Enter **Servers** in the search field.
3. Select **Servers**.
4. Click the **Open** button.
The **Servers** view appears.

Your newly added local server is now listed in the **Servers** view.

3.2. CONFIGURING A REMOTE SERVER

The following section describes how to configure a remote server.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **New** window appears.
3. Enter **Server** in the search field.
4. Select **Server**.
5. Click the **Next** button.
The **Define a New Server** window appears.
6. Select the server type.
7. Click the **Next** button.
8. Select the **Remote** check box.
9. Select one of the **Controlled by** options.
10. Select the **Server lifecycle externally managed** check box.
11. Assign runtimes to the server.
12. Click the **Next** button.
The **Remote System Integration** window appears.
13. Click the **Browse** button in the **Remote Server Home** field.
14. Specify the path to the directory that contains the remote server.
15. Click the **Finish** button.

Verification steps

1. Click the **Window** → **Show View** → **Other**.
The **Show View** window appears.

2. Type **Servers** in the search field.
3. Select **Servers**.
4. Click the **Open** button.
The **Servers** view appears.

Your newly added remote server is now listed in the **Servers** view.

3.3. DEPLOYING AN APPLICATION

After configuring the local server, you can deploy applications to the server from the IDE using the server adapter. The server adapter enables runtime communication between the server and IDE for easy deployment of applications and server management.

The following section describes how to deploy an application to the server in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Right-click the target **project** → **Run as** → **Run on Server**.
The **Run on Server** window appears.
3. Select the **Choose an existing server** check box.
4. Expand **localhost** to select the server for deployment.
5. Click the **Finish** button.
The **Console** view appears.

Your application opens in the internal CodeReady Studio web browser.

CHAPTER 4. JBOSS EAP AND JBOSS WFK BASICS IN CODEREADY STUDIO

Eclipse IDE supports application development and deployment with Red Hat JBoss Enterprise Application Platform (JBoss EAP) and Red Hat JBoss Web Framework Kit (JBoss WFK). However, you need to configure Maven repositories first. This configuration is essential for using the enterprise versions of the example Maven projects provided in Red Hat Central. These projects are intended for deployment to JBoss EAP and require IDE access to JBoss EAP and JBoss WFK repositories.

The following section describes how to use CodeReady Studio to:

- Configure Maven repositories.
- Set up JBoss EAP.
- Verify that JBoss EAP and CodeReady Studio are working correctly.

4.1. CONFIGURING MAVEN REPOSITORIES

The following section describes how to configure Maven repositories.

Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.
The **Preferences** window appears.
3. Enter **JBoss Tools** in the search field.
4. Select **JBoss Maven Integration**.
The **JBoss Maven Integration** window appears.
5. Click the **Configure Maven Repositories** button.
The **Maven Repositories** window appears.
6. Click the **Add Repository** button.
The **Add Maven Repository** window appears.
7. Click the down-arrow in the **Profile ID** field.
8. Select the **redhat-ga-repository**.
Other fields are populated automatically.
9. Click the **OK** button.
10. Click the **Finish** button.
The **Confirm File Update** window appears.
11. Click the **Yes**.
12. Click the **Apply and Close** button.

Additional resources

- For more information on Maven repositories, see [Maven: Getting Started - Developers](#).

4.2. SETTING UP JBOSS EAP

To set up JBoss EAP in Eclipse IDE, you must direct the IDE to the local or remote runtime servers. This establishes a communication channel between the IDE and the JBoss EAP server for efficient deployment and server management workflows.

The following section describes how to install JBoss EAP in CodeReady Studio.

Prerequisites

- Configured Maven repositories.
For more information on how to configure Maven repositories, see [Section 4.1, “Configuring Maven repositories”](#)

Procedure

1. Start CodeReady Studio.
2. Click **Window** → **Preferences**.
The **Preferences** window appears.
3. Enter **JBoss Tools** in the search field.
4. Select **JBoss Runtime Detection**
The **JBoss Runtime Detection** window appears.
5. Click the **Download** button.
The **Download Runtimes** window appears.
6. Select one of the JBoss EAP versions.



NOTE

If you select the JBoss EAP version 6.0.x or earlier, follow the on-screen instructions. If you select a later version, follow the instructions below.

7. Click the **Next** button.
8. Click the **Add** button.
9. Enter your **access.redhat.com** username and password.
10. Click the **OK** button.
The **Secure Storage Password** window appears.
11. Enter the new master password.
Optionally, you can provide a password hint.
12. Click the **Next** button.
Review the license agreement, if satisfied, accept the license and click the **Next** button to continue with the installation.
13. Click the **Browse** button to select the **Install folder**.

14. Click the **Browse** button to select the **Download folder**.
15. Click the **Finish** button.
The **Download Red Hat JBoss EAP** window appears.

Note that downloading and installing the Runtime might take a while.
16. Ensure the path to the JBoss EAP installation file is checked.
17. Click the **Apply and Close** button.

Verification steps

1. Click the **Window → Show View → Other**.
The **Show View** window appears.
2. Type **Servers** in the search field.
3. Select **Servers**.
4. Click the **Open** button.
The **Servers** view appears.

Your newly added JBoss EAP is now listed in the **Servers** view.

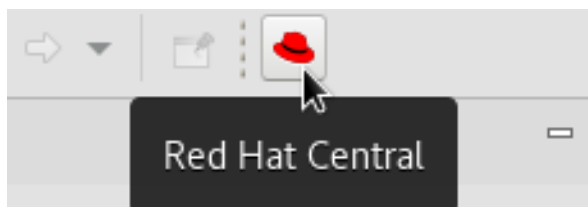
4.3. VERIFYING THAT JBOSS EAP AND CODEREADY STUDIO ARE WORKING CORRECTLY

The following section describes how to verify that JBoss EAP and CodeReady Studio are working correctly by building, deploying, and running the imported **helloworld** application in the JBoss EAP server.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Click the **Red Hat Central** icon.

Figure 4.1. Click the Red Hat Central icon

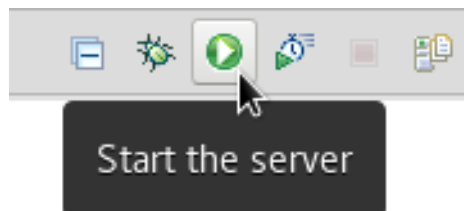


4. Enter **helloworld** into the search field.
5. Click the **helloworld-rs** quickstart.
The **New Project Example** window appears.
6. Click the **Finish** button.

Note that it might take some time for the **helloworld** application and its Maven dependencies to get imported. Once all the dependencies are imported, the **New Project Example** window appears.

7. Click the **Finish** button.
8. In the **Server** view select the **Red hat JBoss EAP**.
9. Click the **Start the Server** icon.

Figure 4.2. Click the Start the Server icon



10. In the **Project Explorer** view right-click the **jboss-helloworld-rs** → **Run Ad** → **Run on Server**. The **Run on Server** window appears.
11. Click the **Finish** button. The **helloworld** application opens in the internal CodeReady Studio web browser.
12. Choose between the JSON or XML format.

The CodeReady Studio web browser prints the **Hello World!** message.

CHAPTER 5. OPENSIFT BASICS IN CODEREADY STUDIO

CodeReady Studio includes OpenShift Application Explorer view, which provides a different and simplified user experience allowing easy and rapid feedback through the inner loop and debugging.

The following section describes how to:

- Set up OpenShift Application Explorer.
- Connect to the OpenShift cluster using OpenShift Application Explorer.
- Create a new launcher project using OpenShift Application Explorer.
- Create a new project using OpenShift Application Explorer.
- Create a new component using OpenShift Application Explorer.
- Deploy a component on cluster using OpenShift Application Explorer.
- Define a URL to externally access the component.
- Debug an application on the cluster using OpenShift Application Explorer.

5.1. SETTING UP OPENSIFT APPLICATION EXPLORER VIEW

The following section describes how to open OpenShift Application Explorer in CodeReady Studio.

Procedure

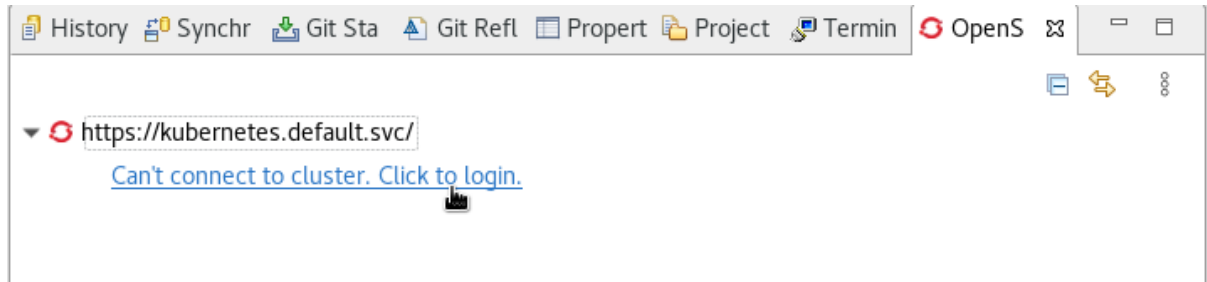
1. Start CodeReady Studio.
2. Click **Window** → **Show View** → **Other**.
The **Show View** window appears.
3. Enter **OpenShift** in the search field.
4. Select **OpenShift Application Explorer**.
5. Click the **Open** button.
The **OpenShift Application Explorer** view appears.

5.2. CONNECTING TO THE OPENSIFT CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to login to the OpenShift cluster in CodeReady Studio using OpenShift Application Explorer.

Procedure

1. Start CodeReady Studio.
2. Open **OpenShift Application Explorer**.
3. Click the **Can't connect to cluster. Click to login** button.



The **Login** window appears.

4. Enter your login credentials.
5. Click the **Finish** button.
Your projects appear in the **OpenShift Application Explorer** view.

5.3. CREATING A NEW LAUNCHER PROJECT

The following section describes how to create a new launcher project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **New** window appears.
3. Enter **Launcher** in the search field.
4. Select **Launcher project**.
5. Click the **Next** button.
The **New Launcher project** window appears.
6. Set **Mission** to **rest-http**.
7. Set **Runtime** to **vert.x community**.
8. Name your project.
9. Select the location for your project.
10. Click the **Finish** button.
Note that the process of resolving dependencies might take some time to complete.

Your newly created launcher project is now listed in the **OpenShift Application Explorer** view.

Verification steps

1. Click **Window** → **Show View** → **Other**.
The **Show View** window appears.
2. Enter **Project Explorer** in the search field.
3. Select **Project Explorer**.

4. Click the **Open** button.
The **Project Explorer** view appears.

Your newly created launcher project is now listed in the **Project Explorer** view.

5.4. CREATING A NEW PROJECT USING OPENSIFT APPLICATION EXPLORER

The following section describes how to create a new project using OpenShift Application Explorer in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Right-click any place in **OpenShift Application Explorer** → **New** → **Project**.
The **New project** window appears.
4. Name your project.
5. Click the **OK** button.

Your newly created project is now listed in the **OpenShift Application Explorer** view.

5.5. CREATING A NEW COMPONENT USING OPENSIFT APPLICATION EXPLORER

The following section describes how to create a new component using OpenShift Application Explorer in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Right-click the target **Project** → **New** → **Component**.
The **New component** window appears.
4. Name your project.
5. Click the **Browse** button to select the **Eclipse Project**.
6. Set **Component type** to **java**.
7. Set **Component version** to **8**.
8. Name your application.
9. Clear the **Push after create** check box.
10. Click the **Finish** button.
The **Console** view appears, displaying the validation process.

Your newly created component is now listed in the **OpenShift Application Explorer** view under your project.

Additional resources

- For more information on how to create an Eclipse project, see [Section 5.3, “Creating a new launcher project”](#).

5.6. DEPLOYING A COMPONENT ON CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to deploy a component on cluster using OpenShift Application Explorer in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Expand your project.
4. Expand your application.
5. Right-click the target **component** → **Push**.
The **Console** view appears, displaying the process of file synchronization.

5.7. DEFINING AN EXTERNAL ACCESS URL USING OPENSIFT APPLICATION EXPLORER

The following section describes how to define an external access URL using OpenShift Application Explorer in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Expand your project.
4. Expand your application.
5. Right-click the target **component** → **New** → **URL**.
6. Name your URL.
7. Set **Port** value to **8080**.
8. Click the **Finish** button.
The **Console** view appears, displaying the process of URL creation.
9. Right-click the target **component** → **Push**.

Your newly created URL is now listed in the **OpenShift Application Explorer** view under your component.

5.8. DEBUGGING AN APPLICATION ON THE CLUSTER USING OPENSIFT APPLICATION EXPLORER

The following section describes how to debug a component using OpenShift Application Explorer in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Start **OpenShift Application Explorer**.
3. Expand your project.
4. Expand your application.
5. Right-click the target **component** → **Debug**.
The **Console** view appears.
6. Expand your component.
7. Right-click the target **url** → **Open in Browser**.
The **Confirm Perspective Switch** window appears.
8. Click the **Switch** button.
The **Debug Perspective** window appears.

CHAPTER 6. QUARKUS TOOLS BASICS IN CODEREADY STUDIO

Quarkus is a Kubernetes-Native full-stack Java framework aimed to optimize work with Java virtual machines. Quarkus provides tools for Quarkus applications developers, helping them reduce the size of Java application and container image footprint, eliminate programming baggage, and reduce the amount of memory required.

The following section describes how to:

- Create a new Quarkus project.
- Run a Quarkus application.
- Debug a Quarkus application.
- Use Quarkus code completion.

Prerequisites

- Ensure you have the latest version of JBoss Tools installed. For more information, see <https://tools.jboss.org/downloads/jbosstools/2020-03/4.15.0.Final.html>.

6.1. CREATING A NEW QUARKUS PROJECT

The following section describes how to create a new Quarkus project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **New** window appears.
3. Enter **Quarkus** in the search field.
4. Select **Quarkus Project**.
5. Click the **Next** button.
The **New Quarkus project** window appears.
6. Select the project type.
7. Name your project.
8. Select the location for your project.
9. Click the **Next** button.
10. Ensure that the default values are correct.
11. Click the **Next** button.
The **Quarkus extensions** window appears.
12. Select the appropriate **Categories** for your projects.

The available extensions for the selected category are displayed in the **Extensions** column.

13. Double-click on the extension to select or deselect it.
The selected extensions appear in the **Selected** column.
14. Click the **Finish** button.

Verification steps

1. Click **Window** → **Show view** → **Other**.
The **Show View** window appears.
2. Enter **Project Explorer** in the search field.
3. Select **Project Explorer**.
4. Click the **Open** button.
The **Project Explorer** view appears.

Your newly created Quarkus project is now listed in the **Project Explorer** view.

6.2. RUNNING A QUARKUS APPLICATION

The following section describes how to run a Quarkus application in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **Run** → **Run Configurations**.
The **Run Configurations** window appears.
3. Scroll down to **Quarkus Application**.
4. Right-click **Quarkus Application** → **New Configuration**.
5. Click the **Browse** button to locate your target project.
6. Click the **OK** button.
7. Click the **Run** button.
The **Console** view appears.

Your application starts after the built process.

6.3. DEBUGGING A QUARKUS APPLICATION

The following section describes how to debug a Quarkus application in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **Run** → **Debug Configurations**.

3. Scroll down to **Quarkus Application**.
4. Click the **Debug** button.
The **Console** view appears.

Your Quarkus application starts and connects to a remote JVM debug configuration to your running Quarkus application. If you set breakpoints in your application source files, the execution automatically stops after reaching the breakpoint.

6.4. USING LANGUAGE SUPPORT IN CODEREADY STUDIO

6.4.1. Language support

Every Quarkus application is configured through an **application.properties** configuration file. The content of this configuration file is dependent of the set of Quarkus extensions that your application is using.

Quarkus Tools includes content assist which provides code completion, validation, and documentation. Code completion allows you to quickly complete statements in your code. Multiple choices are available to you via popups.

This language support is now available for Kubernetes, OpenShift, S2i, Docker properties, MicroProfile REST Client properties, and MicroProfile Health artifacts.

6.4.2. Using Quarkus code completion

The following section describes how to use Quarkus **application.properties** content assist in CodeReady Studio.

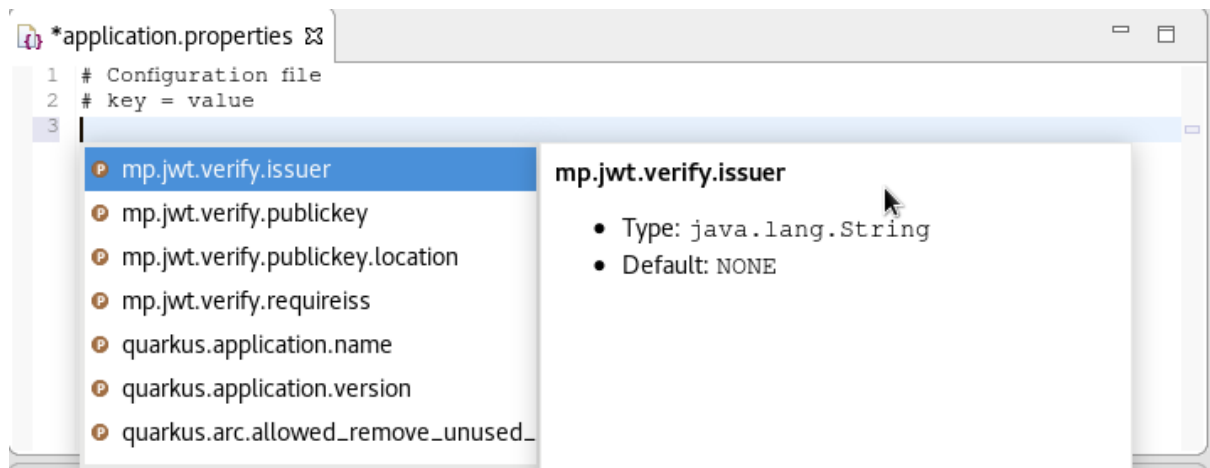
Prerequisites

- An existing Quarkus project.
For more information on how to create a Quarkus project, see [Section 6.1, "Creating a new Quarkus project"](#)

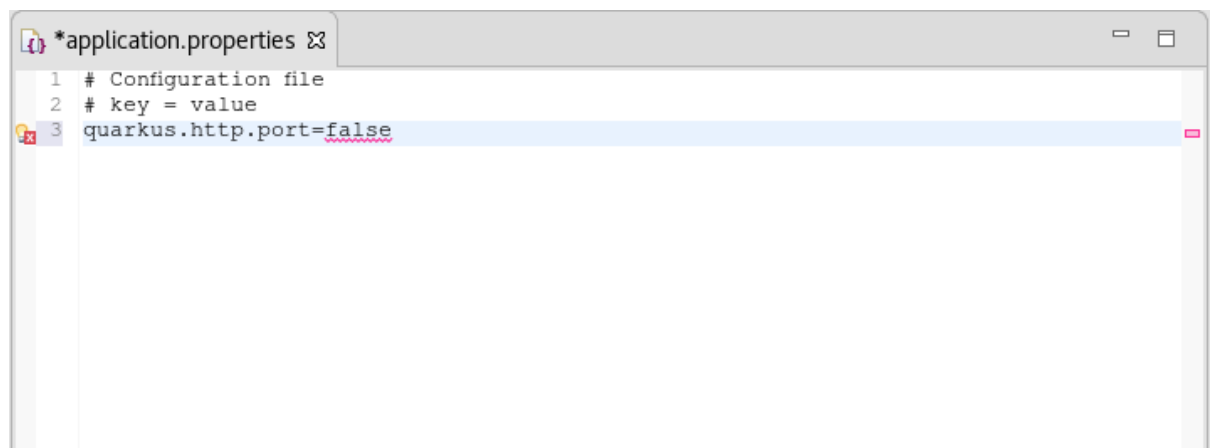
Procedure

1. Start CodeReady Studio.
2. Start **Project Explorer**.
3. Expand your **Quarkus project** → **src** → **main** → **resources**.
4. Right-click **application.properties** → **Open With** → **Generic Text Editor**.
The **Text Editor** window appears.
5. Navigate to an empty line.
6. Press **Ctrl+Space** to invoke code completion.
The code compilation suggestions appear. Hover the mouse over the suggestions to display documentation.

Figure 6.1. CodeReady Studio code compilation invoked



If you enter a wrong value, the editor underlines the error with a red wavy line.



Additional resources

- Language support for MicroProfile REST Client properties and MicroProfile Health artifact needs to be enabled separately. For more information, see [Section 6.4.3, “Enabling language support for MicroProfile”](#).

6.4.3. Enabling language support for MicroProfile

The following section describes how to enable language support for MicroProfile REST Client properties.

Prerequisites

- An existing Quarkus project.
For more information on how to create a Quarkus project, see [Section 6.1, “Creating a new Quarkus project”](#)

Procedure

1. Start CodeReady Studio.
2. Start **Project Explorer**.
3. Expand the target **Quarkus project** → `src/main/java`.

4. Right-click **org.acme** → **New** → **Other**.
The **New** window appears.
5. Enter **file** in the search field.
6. Select **File**.
7. Click the **Next** button.
8. Name the file **MyServiceClient.java**.
9. Click the **Finish** button.
10. Paste the following content into the **MyServiceClient.java** file:

```
package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

@RegisterRestClient
public interface MyServiceClient {
    @GET
    @Path("/greet")
    Response greet();
}
```

11. Press **Ctrl+S** to save the changes.

Note that you can adjust the language support by changing the configuration key for the client from **@RegisterRestClient** to **@RegisterRestClient(configKey = "myclient")**. The language support will be adjusted accordingly.

Additional resources

- For more information on how to use language support, see [Section 6.4.2, "Using Quarkus code completion"](#).

CHAPTER 7. HIBERNATE TOOLS BASICS IN CODEREADY STUDIO

Hibernate Tools is a collection of tools for projects related to Hibernate version 5 and earlier. The tools provide Eclipse plugins for reverse engineering, code generation, visualization and interaction with Hibernate.

The following section describes how to:

- Create a new JPA project.
- Generate entities.
- Create a Hibernate mapping file.
- Create a Hibernate configuration file.
- Create a Hibernate console configuration file.
- Edit Hibernate project configurations.

Prerequisites

1. Download the [h2 version of the Sakila database](#).
2. Navigate to the directory that contains the **runh2.sh** file.
3. Execute the **runh2.sh** file:

```
┆ $ ./runh2.sh
```

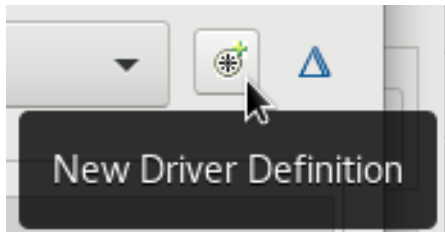
7.1. CREATING A NEW JPA PROJECT

The following section describes how to create a new JPA project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **New** window appears.
3. Enter **JPA** in the search field.
4. Select **JPA Project**
5. Click the **Next** button.
The **New JPA Project** window appears.
6. Name your project.
Note that the name of the project cannot include spaces or special characters. The only special characters allowed are periods (.), underscores (_), and dashes (-).
7. Click the down-arrow in the **Target runtime** field.

8. Select the runtime server.
9. Ensure that the JPA version is set to 2.1.
10. Click the **Next** button.
The **Java** window appears.
11. Select the source folder.
12. Click the **Next** button.
The **JPA Facet** window appears.
13. Click the down-arrow in the **Platform** field.
14. Select **Hibernate (JPA 2.1)**.
15. Add user libraries or set the **JPA Implementation Type** to **Disable Library Configuration**.
For more information on how to set up user libraries, see [Section 7.2, "Adding libraries"](#).
16. Click **Add connection**.
The **New Connection Profile** window appears.
17. Enter **Generic** in the search field.
18. Select **Generic JDBC**.
19. Enter **sakila** in the **Name** field
20. Click the **Next** button.
The **Specify a Driver and Connection Details** window appears.
21. Click the **New Driver Definition** icon.



Click the New Driver Definition icon.

The **New Driver Definition** window appears.

22. Select the **Generic JDBC Driver**.
23. Click the **JAR List** tab.
24. Click the **Add JAR/Zip** button.
25. Select the **.jar** file for the Sakila database.
26. Click the **Properties** tab.
27. Add **jdbc:h2:tcp://localhost/./sakila** to the **Connection URL** field.
28. Click the **Driver Class** field.

29. Click the three dots icon at the end of the **Driver Class** field.
The **Available Classes from Jar List** window appears.
30. Select the **Browse the Class** option.
The **org.h2.Driver** appears.
31. Click the **OK** button.
32. Enter **sa** to the **User ID** field.
33. Click the **OK** button.
34. Click the **Finish** button.
35. Click the **Finish** button.
The **Open Associated Perspective?** window appears.
36. Click the **Open Perspective** button.

Your newly created JPA project is now listed in the **Project Explorer**.

7.2. ADDING LIBRARIES

The following section describes how to add libraries to your Hibernate project in CodeReady Studio.

Procedure

1. Download **Hibernate ORM** from <http://hibernate.org/orm/>.
2. Extract the files from the **.zip** file.
3. Start CodeReady Studio.
4. Click **Window → Preferences**.
The **Preferences** window appears.
5. Enter **Libraries** in the search field.
6. Select **User Libraries** under **Java**.
7. Click the **New** button.
The **New User Library** window appears.
8. Name your user library.
9. Click the **OK** button.
10. Select your new user library.
11. Click the **Add External JARs** button.
12. Locate the directory you extracted the **Hibernate ORM .zip** file into.
13. Navigate to the **/lib/required/** directory.
14. Select the **.jar** files.

15. Click the **Apply and Close** button.

7.3. GENERATING ENTITIES

The following section describes how to generate entities for your hibernate project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Right-click the target **JPA project** → **JPA Tools** → **Generate Tables from Entities**.
The **Generate Tables from Entities** window appears.
4. Select the **Use Console Configuration** check box.
5. Click the **Finish** button.

7.4. CREATING A HIBERNATE MAPPING FILE

Hibernate mapping files specify how your objects relate to the database tables.

The following section describes how to create a Hibernate mapping file in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **Show View** window appears.
3. Enter **Hibernate** in the search field.
4. Select **Hibernate XML Mapping file (hbm.xml)**
5. Click the **Next** button.
The **Create Hibernate XML Mapping file (hbm.xml)** window appears.
6. Click the **Add Class** button to add classes.
7. Click the **Add Package** button to add packages.
Alternatively, you can create an empty **hbm.xml** file by not selecting any packages or classes.
8. Select the **depth control** check box to define the dependency depth used when choosing classes.
9. Click the **Next** button.
10. Select the parent directory.
11. Name your **hbm.xml** file .
12. Click the **Finish** button.

Your newly created **hbm.xml** file is now displayed in CodeReady Studio.

7.5. CREATING A HIBERNATE CONFIGURATION FILE

For reverse engineering, prototype queries, or to Hibernate Core usage, a **hibernate.properties** or a **hibernate.cfg.xml** file is required. CodeReady Studio provides a wizard to generate the **hibernate.cfg.xml**.

The following section describes how to create a Hibernate mapping file in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **Show View** window appears.
3. Enter **Hibernate** in the search field.
4. Select **Hibernate Configuration file (cfg.xml)**.
5. Click the **Next** button.
The **Create Hibernate Configuration file (cfg.xml)** window appears.
6. Select the parent directory.
7. Click the **Next** button.
The **Hibernate Configuration file (cfg.xml)** window appears.
8. Name your **cfg.xml** file.
9. Click the down-arrow in the **Database dialect** field.
10. Select the relevant database.
11. Click the down-arrow in the **Driver class** field.
12. Select the relevant driver.
13. Click the down-arrow in the **Connection URL** field.
14. Select the relevant URL.
15. Click the **Finish** button.

Your newly created **cfg.xml** file is now displayed in CodeReady Studio.

7.6. CREATING A HIBERNATE CONSOLE CONFIGURATION FILE

A Console configuration file describes how the Hibernate plugin configures Hibernate. It also describes the configuration files and classpaths needed to load the POJOs, JDBC drivers, and so on. It is required to make use of query prototyping, reverse engineering and code generation. You can have multiple console configurations per project, however, one configuration is sufficient.

The following section describes how to create a Hibernate console configuration file in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **Show View** window appears.
3. Enter **Hibernate** in the search field.
4. Select **Hibernate Console Configuration**.
5. Click the **Next** button.
The **Create Hibernate Console Configuration** window appears.
6. Name your configuration file .
7. Ensure that the **Type** is set to **Core**.
8. Select the relevant **Hibernate version**.
9. Click the **Browse** button to locate the target project.
10. Click the **New** button to configure a new **Database connection**.
11. Click the **Setup** button to setup the **Property file**.
The **Setup property file** window appears.
12. Click the **Create new** button.
The **Create Hibernate Properties file (.properties)** window appears.
13. Select the parent directory.
14. Name your **.properties** file.
15. Click the **Finish** button.
16. Click the **Setup** button setup the **Configuration file**.
17. Select the path to the target **.cfg.xml** file.
For more information on how to create a new **.cfg.xml** file, see [Section 7.5, "Creating a Hibernate configuration file"](#).
18. Click the **OK** button.
19. Click the **Finish** button.

7.7. EDITING HIBERNATE PROJECT CONFIGURATIONS

The following section describes how to edit configurations for the Hibernate project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.

- The **Show View** window appears.
3. Enter **Hibernate** in the search field.
 4. Select **Hibernate Configuration**.
 5. Click the **OK** button.
The **Hibernate Configurations** view appears.
 6. Right-click the target **project** → **Edit Configuration**.
The **Edit Configuration** window appears.
 7. Edit the scenery configurations.
 8. Click the **Apply** button.
 9. Click the **OK** button.

CHAPTER 8. NODE.JS BASICS IN CODEREADY STUDIO

Node.js is an event-based, asynchronous I/O framework and is used to develop applications that run JavaScript on the client and server side. This allows the application to reuse parts of the code and to avoid switching contexts. Node.js is commonly used to create applications such as static file servers, messaging middleware, HTML5 game servers, web application framework, and others.

CodeReady Studio supports Node.js application development using the **npm** package installer and offers a built-in debugging tool to identify and fix issues with applications.

The following section describes how to:

- Create a new JavaScript project.
- Create a new **package.js** file.
- Create a new JavaScript file.
- Import an existing JavaScript project.
- Run a Node.js application.

Prerequisites

- Installed **npm**.
- Installed Node.js.
To install the prerequisites on RHEL, use:

```
# yum install npm nodejs
```

8.1. CREATING A NEW JAVASCRIPT PROJECT

The following section describes how to create a new JavaScript project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **File** → **New** → **Other**.
The **New** window appears.
3. Enter **JavaScript** in the search field.
4. Select **JavaScript Project**.
5. Click the **Next** button.
The **Create a JavaScript project** window appears.
6. Name your project.
All other fields are populated automatically.
7. Click the **Finish** button.
The **Open Associated Perspective** window appears.

8. Click the **Open Perspective** button.

Your newly created JavaScript project is now listed in the **Project Explorer** view.

8.2. CREATING A PACKAGE.JSON FILE

The following section describes how to create a new **package.json** file in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **File** → **New** → **Other**.
The **New** window appears.
3. Enter **npm** in the search field.
4. Select **npm Init**.
5. Click the **Next** button.
The **npm Initialization Wizard** window appears.
6. Click the **Browse Workspace** button to select the base directory for the **package.json** file.
7. Click the **Finish** button.
The **Open Associated Perspective** window appears.
8. Click the **Open Perspective** button.

Your newly created **package.js** file is now displayed in the CodeReady Studio editor.

Additional resources

- For more information on how to handle npm **package.json** files, see <https://docs.npmjs.com/files/package.json#dependencies>.

8.3. CREATING A NEW .JS FILE

The following section describes how to create a new **.js** file in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Click **File** → **New** → **File**.
The **Create New File** window appears.
3. Select a parent folder for the file.
4. Name the file.
For example **index.js**.
5. Click the **Finish** button.

Your newly created **.js** file is now displayed in the CodeReady Studio editor.

8.4. IMPORTING AN EXISTING NODE.JS APPLICATION

The following section describes how to import an existing Node.js application.

Procedure

1. Start CodeReady Studio.
2. Click **File** → **Import**.
The **Import** window appears.
3. Expand the **General** folder.
4. Select **Existing Project into Workspace**
5. Click the **Next** button.
6. Click the **Browse** button to locate your existing project.
7. Select the path to your project in the **Projects** field.
8. Click the **Finish** button.

Verification steps

1. Click **Window** → **Perspective** → **Open Perspective** → **Other**.
The **Open Perspective** window appears.
2. Select **JavaScript**.
3. Click the **Open** button.
The **Project Explorer** view appears.

Your newly imported Node.js application is now listed in the **Project Explorer** view.

8.5. RUNNING A NODE.JS APPLICATION

The following section describes how to run a Node.js application in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Right-click target **project** → **Run as** → **Node.js Application**.

The **Console** view appears running your Node.js application.

8.6. DEBUGGING A NODE.JS APPLICATION

CodeReady Studio includes a debugger to help identify and resolve the issues.

The following section describes how to debug Node.js applications in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Expand the target project.
4. Right-click the **index.js** → **Debug as** → **Node Program**.
The **Console** view appears, displaying the error log.

CHAPTER 9. MOBILE WEB TOOLS BASICS IN CODEREADY STUDIO

Mobile Web Tools provide an **HTML5 Project** wizard that enables you to create web applications optimized for mobile devices. The **HTML5 Project** wizard is a useful starting point for creating all new HTML5 web applications in the IDE. The wizard generates a sample ready-to-deploy HTML5 mobile application with REST resources from a Maven archetype.

You can customize the application using the built-in editor, deploy and view the application with the built-in browser.

The IDE provides the **Mobile Web** palette that allows the user to make interactive web applications. This palette offers a wide range of features including drag-and-drop widgets for adding common web interface framework features such as HTML5, jQuery Mobile, and Ionic tags to html files. It also contains widgets like **Panels**, **Pages**, **Lists**, **Buttons** to make the applications more user friendly and efficient.

The following section describes how to:

- Create an HTML5 project.
- Add a new jQuery mobile file to a project.
- Add a new jQuery mobile page to a project.

Prerequisites

- Configured server.
For information on configuring a local runtime server and deploying applications to it, see [Section 3.1, "Configuring a local server"](#).

The IDE must be configured for any servers to which you want to deploy your application, including the location and type of the application server and any custom configuration or management settings. The following sections assume you completed the configuration in advance, but that step can also be completed at deployment.

9.1. CREATING AN HTML5 PROJECT

The **HTML5 Project** wizard generates a sample project based on a Maven archetype and the project and application identifiers provided by you. The Maven archetype version is indicated in the **Description** field in the first page of the wizard and you can change the version, and therefore the project look and dependencies, by selecting either an enterprise or non-enterprise target runtime within the wizard.

The following section describes how to create an HTML5 project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Press **Ctrl+N**.
The **New** window appears.
3. Enter **HTML5** in the search field.
4. Select **HTML5 Project**.

5. Click the **Next** button.
The **New Project Example** window appears.
6. Click the down-arrow in the **Target Runtime** field.
7. Select the server.
8. Click the **Next** button.
9. Name your project.
10. Name your package.
11. Click the **Finish** button.
Note that it might take some time for the project to generate.

The **New Project Example** window appears.
12. Click the **Finish** button.

Verification steps

1. Click the **Window → Show View → Other**.
The **Show View** window appears.
2. Select **Project Explorer**.
3. Click the **Open** button.
The **Project Explorer** view appears.

Your newly created project is now listed in the **Project Explorer** view.

9.2. ADDING A NEW HTML5 JQUERY MOBILE FILE

The HTML5 **jQuery Mobile** file template consists of JavaScript and CSS library references that are inserted in the file's HTML header. The template also inserts a skeleton of the **jQuery Mobile** page and **listview** widgets in the file's HTML body. The following procedure details the steps to insert the template into your project.

The following section describes how to add a new HTML5 jQuery Mobile file to an existing project.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Expand the target **project → src → main**.
4. Right-click the **webapp → New → HTML File**.
The **New HTML File** window appears.
5. Ensure the **webapp** directory is selected.
6. Name your file.

7. Click the **Next** button.
8. Select **HTML5 jQuery Mobile Page (1.4)**.
9. Click the **Finish** button.

The newly added HTML5 jQuery mobile file is now displayed in the CodeReady Studio editor.

9.3. ADDING A NEW MOBILE PAGE

The following section describes how to add a new jQuery Mobile Page to an existing web application.

Procedure

1. Start CodeReady Studio.
2. Open **Project Explorer**.
3. Expand the target **project** → **src** → **main** → **webapp**.
4. Right-click the **.html** file.
5. Click **Open With** → **JBoss Tools HTML Editor**.
The **Pallet** view opens.
6. Click the **Pallet** view.
7. Expand **jQuery Mobile**.
8. Click the **Page** icon.
The **Insert Tag** window appears.
9. Name the **Header**.
10. Name the **Footer**.
11. Click the **Finish** button.

Your newly added page is now displayed in the CodeReady Studio editor.

Note that the same workflow can be used to customize the pages of your web application with widget selection from the **Pallet** view.