Red Hat CodeReady Studio 12.14

Getting Started with CodeReady Studio Tools

Introduction to using Red Hat CodeReady Studio tools

Last Updated: 2020-04-08

Red Hat CodeReady Studio 12.14 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

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

Eclipse IDE includes Git Perspective that allows developers to manage their Git repositories from a graphical interface. The following section describes 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 locate Git Perspective in the IDE.

Procedure

1. Start Eclipse IDE.
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.

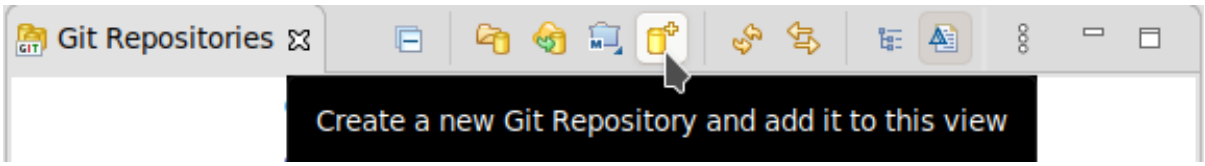
1.2.1. Creating a new Git repository

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.

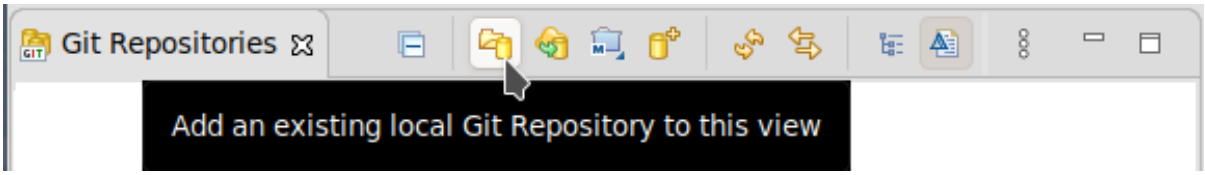
1.2.2. Adding an existing local Git repository

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, check the **Look for nested repositories** 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.

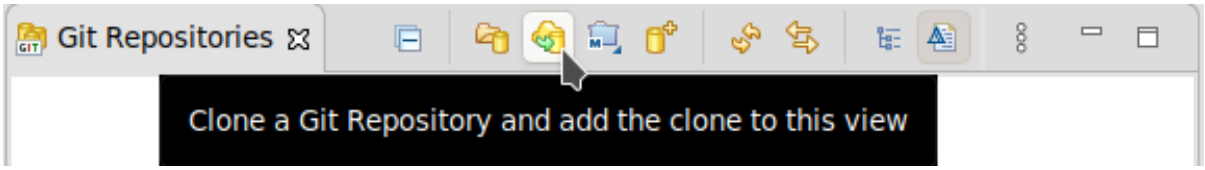
1.2.3. Cloning an existing Git repository

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

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 you cloned during step one 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 the **Directory** path and **Initial branch** are set correctly.
   Optionally, you can customize the **Remote name** and select the appropriate working sets by checking the **Add project to working sets** 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 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. Ensure the **Configure upstream for push and pull**and **Checkout new branch** boxes are checked.

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.

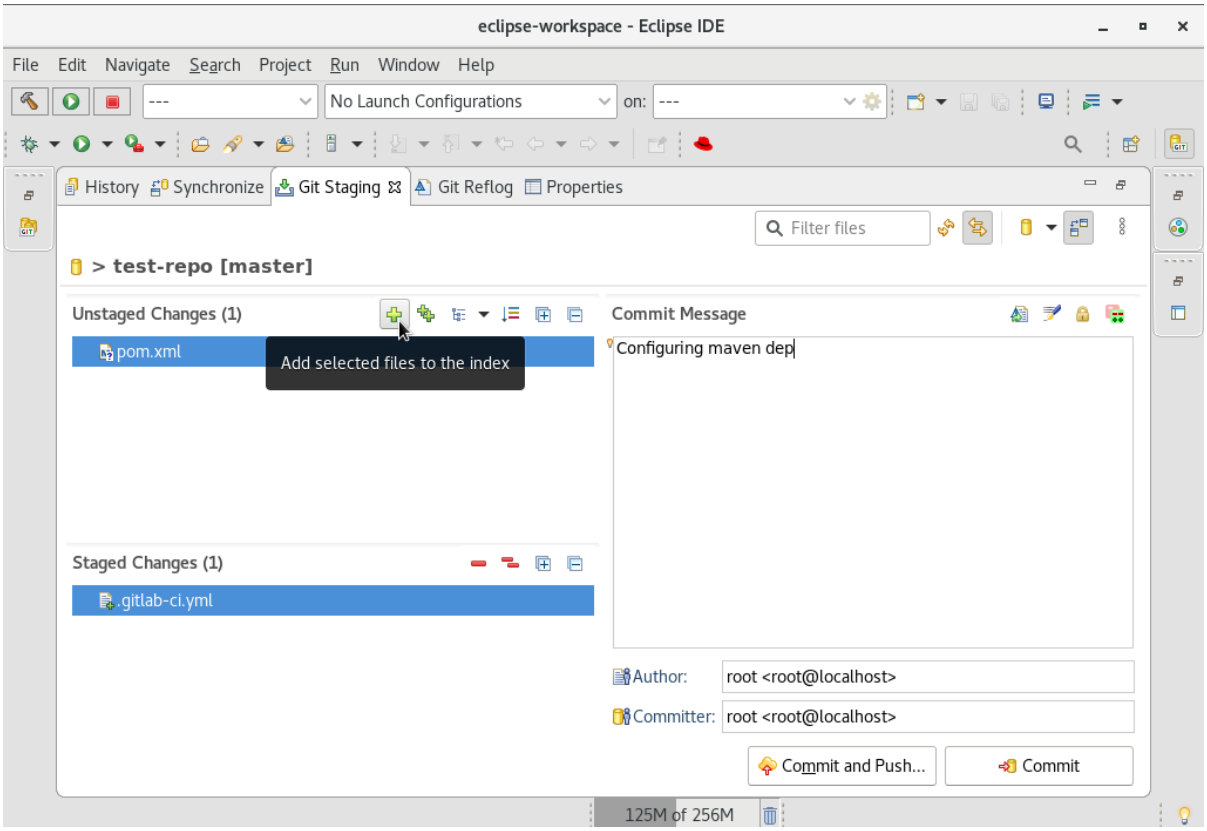
1.4. COMMITTING AND PUSHING THE CHANGES

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

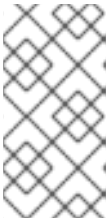
- Set up Git Perspective.
  For more information on how to set up Git Perspective, see Section 1.1, "Setting up Git Perspective".

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

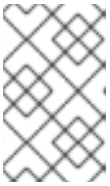
1. Start CodeReady Studio.
2. Open Git Perspective.
3. Click **File → New → Other**.
   The **New** window appears.
4. Enter **Maven Projects** in the **Wizards** field.
5. Select **Maven Projects**.
6. Click the **Next** button.
   The **New Maven Project** window appears.
7. Check the **Create a simple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



NOTE

By checking the **Create a simple project**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 artificial 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 artificial 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 **System 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.

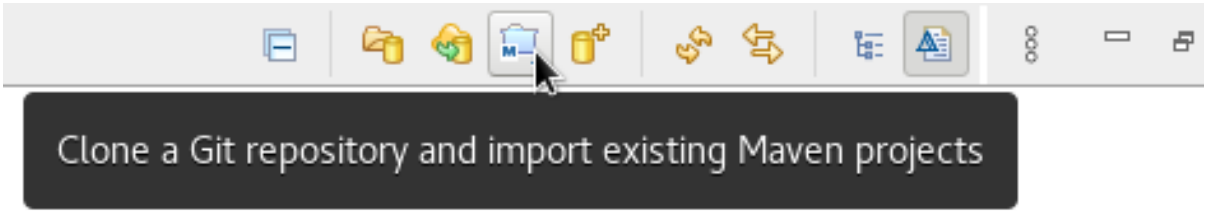
2.2.2. Importing the existing remotely stored Maven project

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. Check the **Create a simple project**box.

NOTE

By checking the **Create a simple project**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. 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.

3.1. 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.

3.2. 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.

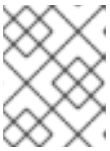
The following section describes how to install JBoss EAP in CodeReady Studio.

Prerequisites

- Configured Maven repositories.
  For more information on how to configure Maven repositories, see Section 3.1, "Configuring Maven repositories"

Procedure

1. Start CodeReady Studio.
2. Start **Git Perspective**.
3. Click **Window → Preferences**.
   The **Preferences** window appears.
4. Enter **JBoss Tools** in the search field.
5. Select **JBoss Runtime Detection**.
   The **JBoss Runtime Detection** window appears.
6. Click the **Download** button.
   The **Download Runtimes** window appears.
7. 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.

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

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

    Note that downloading and installing the Runtime might take a while.
17. Ensure the path to the JBoss EAP installation file is checked.
18. 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.

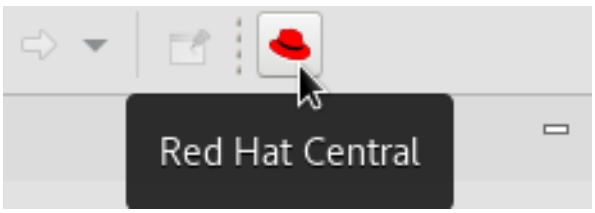
3.3. VERIFYING THAT JBOSS EAP AND CODEREADY STUDIO ARE WORKING CORRECTLY

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 **Git Perspective**.
3. Open **Project Explorer**.
4. Click the **Red Hat Central** icon.

   Figure 3.1. Click the Red Hat Central icon

5. Enter **helloworld** into the search field.
6. Click the **helloworld-rs** quickstart.

The **New Project Example** window appears.

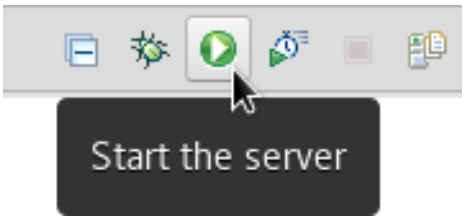
7. Click the **Finish** button.
   Note that it might take some time for the **helloworld** application and its Maven dependencies to get imported. Once all the dependencies are imported, the **New Project Example** window appears.

8. Click the **Finish** button.

9. In the **Server** view select the **Red hat JBoss EAP**.

10. Click the **Start the Server** icon.

    Figure 3.2. Click the Start the Server icon

11. In the **Project Explorer** view right-click the **jboss-helloworld-rs → Run Ad → Run on Server**.
    The **Run on Server** window appears.

12. Click the **Finish** button.
    The **helloworld** application opens in the internal CodeReady Studio web browser.

13. Choose between the JSON or XML format.

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

CHAPTER 4. 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 7, use:

```
# yum install npm nodejs
```

  To install the prerequisites, on RHEL 8, use:

```
# dnf install npm nodejs
```

4.1. 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. Open **Git Perspective**.
3. Click **File → New → Other**.
   The **New** window appears.
4. Enter **JavaScript** in the search field.
5. Select **JavaScript Project**.
6. Click the **Next** button.
   The **Create a JavaScript project**window appears.

7. Name your project.
   All other fields are populated automatically.

8. Click the **Finish** button.
   The **Open Associated Perspective** window appears.

9. Click the **Open Perspective** button.

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

4.2. CREATING A PACKAGE.JS FILE

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

Procedure

1. Start CodeReady Studio.

2. Open **Git Perspective**.

3. Click **File → New → Other**.
   The **New** window appears.

4. Enter **npm** in the search field.

5. Select **npm Init**.

6. Click the **Next** button.
   The **npm Initialization Wizard** window appears.

7. Click the **Browse Workspace** button to select the base directory for the **package.json** file.

8. Click the **Finish** button.
   The **Open Associated Perspective** window appears.

9. 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.

4.3. CREATING A NEW .JS FILE

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

Procedure

1. Start CodeReady Studio.

2. Open **Git Perspective**.

3. Click **File → New → File**.

The **Create New File** window appears.

4. Select a parent folder for the file.
5. Name the file.
   For example **index.js**.
6. Click the **Finish** button.

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

4.4. 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.

4.5. 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 **Git Perspective**.

3. Open **Project Explorer**.
4. Right-click target **project → Run as → Node.js Application**.

The **Console** opens running your Node.js application.

4.6. 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** appears, displaying the error log.

CHAPTER 5. FORGE TOOLS BASICS IN CODEREADY STUDIO

CodeReady Studio offers Forge Tools for developing Java EE applications and to extend the IDE functionality in Eclipse.

The following section describes how to:

- Launch the **Forge Console**.
- Create a new Forge project.
- Create a **persistence.xml** file.
- Add a new field to the entity.
- Generate a scaffold.
- Manage addons.
- Run Forge-based applications.

IMPORTANT

Forge Tools are no longer fully supported by Red Hat.

5.1. LAUNCHING THE FORGE CONSOLE

The following section describes how to launch the **Forge Console** view.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Click **Window: → Show view → Other**.
   The **Show View** window appears.
4. Enter **Forge Console** in the search Field.
5. Select **Forge Console**.
6. Click the **Open** button.

The **Forge Console** view appears.

5.2. CREATING A NEW FORGE PROJECT

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

Procedure

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

3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
4. Enter **Project** in the search field.
5. Select **Project: New**.
   The **Project: New** window appears.
6. Name your project. Note that the project name cannot include spaces or special characters. The only special characters allowed are periods (**.**), underscores (_), and dashes (**-**).
7. Click the **Browse** button to select the project location.
8. Click the down-arrow in the **Stack** field.
9. Select **Java EE 7**.
10. Click the **Finish** button.

Verification steps

1. Click **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 Forge project is now listed in the Project Explorer view.

5.3. CREATING A PERSISTENCE.XML FILE

The following section describes how to create a **persistence.xml** file for your Forge project in CodeReady Studio using JPA.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
4. Enter **JPA** in the search field.
5. Select **JPA: New Entity**.
   The **JPA: Setup** window appears.

   Not that the Forge Console automatically detects any prerequisites that must be set up and prompts you to create those at runtime.
6. Click the **Next** button.
   The **JPA: Connection Settings** window appears.

Ensure that the automatically generated values are correct.

7. Click the **Next** button.
   The **JPA: New Entity** window appears.

8. Name your entity.
   Note that the entity name cannot include spaces or special characters. The only special character allowed is underscore (_).

9. Click the **Finish** button.

Your newly created Forge entity is now displayed in the CodeReady Studio editor.

Verification steps

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

2. Select **Project Explorer**.

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

4. Expand your Forge project.

Your newly created **.java** Forge entity and the corresponding **persistence.xml** file are now listed in the **Project Explorer**.

5.4. ADDING A NEW FIELD TO THE ENTITY

The following section describes how to add a new field to the entity for your Forge project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.

2. Open **Git Perspective**.

3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.

4. Enter **Field** in the search field.

5. Select **JPA: New Field**.
   The **JPA: New Field** window appears.

6. Click the down-arrow in the **Target Entity** field.

7. Select the entity for your project.

8. Name your new field.

9. Click the **Finish** button.

Your newly created field is now added to the **.java** file of your Forge project. The **.java** file is displayed in the CodeReady Studio editor.

5.5. GENERATING A SCAFFOLD

Scaffolding is an automatic code generation technique. A program generates a basic CRUD (create, read, update, delete) admin interface by using available information (usually a database).

The following section describes how to generate a scaffold for your Forge project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
4. Enter **Scaffold** in the search field.
5. Select **Scaffold: Generate**.
   The **Scaffold: Generate** window appears.
6. Click the down-arrow in the **Scaffold Type** field.
7. Select **Angular JS**.
8. Click the **Next** button.
   The **Setup Facets** window appears.
9. Ensure that the automatically generated values are correct.
10. Click the **Next** button.
    The **Select JPA entities** window appears.
11. Select the JPA entities for scaffolding.
12. Click the **Finish** button.

Your newly generated entries are now displayed in the CodeReady Studio editor.

5.6. ADDON MANAGEMENT

The following section describes how to:

- Create a new addon.
- Add a custom Forge command to the addon.
- Installing a custom Forge command.
- Executing a custom Forge command.

5.6.1. Creating addons

The following section describes how to create addons for your Forge project in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
4. Enter **Project** in the search field.
5. Select **Project: New**.
   The **Project: New** window appears.
6. Name your addon. Note that the name of the addon cannot include spaces or special characters. The only special characters allowed are periods (**.**), underscores (_), and dashes (**-**).
7. Click the **Browse** button to select the project location.
8. Click the down-arrow in the **Project Type** field.
9. Select **Forge Addon (JAR)**.
10. Click the **Next** button.
    The **Furnace Addon Setup** window appears.
11. Ensure that the automatically selected dependencies are correct.
12. Click the **Finish** button.
    Note that the process of setting up the dependencies might take some time to complete.

Your newly created addon is now displayed in the CodeReady Studio editor.

Verification steps

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

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

5.6.2. Adding a custom Forge commands to the addon

The following section describes how to create a simple Java class and turn it into a Forge command in CodeReady Studio.

Prerequisites

- Existing addon.
  For more details on how to create an addon, see Section 5.6.1, “Creating addons”.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
4. Enter **New Class** in the search field.
5. Select **Java: New Class**.
   The **Create a new Java Class**window appears.
6. Name your new class. Note that the class name cannot include spaces or special characters. The only special character allowed is underscore (_).
7. Click the **Finish** button.
8. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
9. Enter **UI** in the search field.
10. Select **Addon: New Annotated UI Command**.
    The **Addon: New Annotated UI Command** window appears.
11. Enter the **Command name**.
12. Click the **Finish** button.

Your custom Forge command is now created.

5.6.3. Installing a custom Forge command

The following section describes how to install a custom Forge command into Eclipse IDE using CodeReady Studio.

Procedure

1. Open the **Project Explorer** view.
2. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
3. Enter **Build** in the search field.
4. Select **Build and Install an Addon**
   The **Build and Install an Addon**window appears.
5. Ensure that the path to your addon is correct.
6. Click the **Finish** button.

Your custom command is now installed into Eclipse IDE.

5.6.4. Executing a custom Forge command

The following section describes how to execute a custom Forge command in CodeReady Studio.

Procedure

1. Press **Ctrl**+**4**.
   The **Current Selection** window appears.
2. Enter the name of your command in the search field.
3. Select your command.

5.7. RUNNING FORGE-BASED APPLICATION

The following section describes how to run a Forge-based application in CodeReady Studio.

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Click **Window → Show View → Other**.
   The **Show View** window appears.
4. Select **Project Explorer**.
5. Click the **Open** button.
   The **Project Explorer** view appears.
6. Right-click your Forge **project → Runs as → Run on Server**.
   The **Run On Server** window appears.
7. Select the server to use.
   For example Red Hat JBoss Enterprise Application Platform 7.0.
8. Click the **Finish** button.
   The **Console** view appears.

Your Forge-based application opens in the CodeReady Studio browser.

CHAPTER 6. 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
```

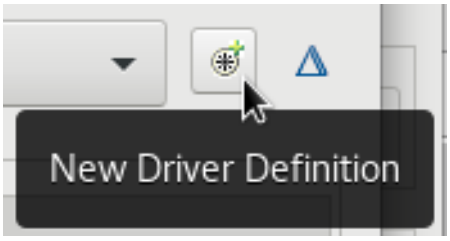
6.1. CREATING A NEW JPA PROJECT

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

Procedure

1. Start CodeReady Studio.
2. Open **Git Perspective**.
3. Press **Ctrl**+**N**.
   The **New** window appears.
4. Enter **JPA** in the search field.
5. Select **JPA Project**
6. Click the **Next** button.
   The **New JPA Project** window appears.
7. 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 (**-**).

8. Click the down-arrow in the **Target runtime** field.
9. Select the runtime server.
10. Ensure that the JPA version is set to 2.1.
11. Click the **Next** button.
    The **Java** window appears.
12. Select the source folder.
13. Click the **Next** button.
    The **JPA Facet** window appears.
14. Click the down-arrow in the **Platform** field.
15. Select **Hibernate (JPA 2.1)**.
16. 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 6.2, "Adding libraries".
17. Click **Add connection**.
    The **New Connection Profile** window appears.
18. Enter **Generic** in the search field.
19. Select **Generic JDBC**.
20. Enter **sakila** in the **Name** field
21. Click the **Next** button.
    The **Specify a Driver and Connection Details** window appears.
22. Click the **New Driver Definition** icon.

    Click the New Driver Definition icon.

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

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

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

6.2. 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. Open **Git Perspective**.
5. Click **Window → Preferences**.
   The **Preferences** window appears.
6. Enter **Libraries** in the search field.
7. Select **User Libraries** under **Java**.
8. Click the **New** button.
   The **New User Library** window appears.
9. Name your user library.
10. Click the **OK** button.
11. Select your new user library.
12. Click the **Add External JARs** button.
13. Locate the directory you extracted the **Hibernate ORM .zip** file into.

14. Navigate to the /**lib**/**required**/ directory.
15. Select the **.jar** files.
16. Click the **Apply and Close** button.

6.3. GENERATING ENTITIES

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

Procedure

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

6.4. 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. Open **Git Perspective**.
3. Click **Ctr**+**N**.
   The **Show View** window appears.
4. Enter **Hibernate** in the search field.
5. Select **Hibernate XML Mapping file (hbm.xml)**.
6. Click the **Next** button.
   The **Create Hibernate XML Mapping file (hbm.xml)** window appears.
7. Click the **Add Class** button to add classes.
8. Click the **Add Package** button to add packages.
   Alternatively, you can create an empty **hbm.xml** file by not selecting any packages or classes.
9. Check the **depth control** box to define the dependency depth used when choosing classes.

10. Click the **Next** button.
11. Select the parent directory.
12. Name your **hbm.xml** file .
13. Click the **Finish** button.

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

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

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

6.6. 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. Open **Git Perspective**.
3. Click **Ctr**+**N**.
   The **Show View** window appears.
4. Enter **Hibernate** in the search field.
5. Select **Hibernate Console Configuration**.
6. Click the **Next** button.
   The **Create Hibernate Console Configuration** window appears.
7. Name your configuration file .
8. Ensure that the **Type** is set to **Core**.
9. Select the relevant **Hibernate version**.
10. Click the **Browse** button to locate the target project.
11. Click the **New** button to configure a new **Database connection**.
12. Click the **Setup** button to setup the **Property file**.
    The **Setup property file** window appears.
13. Click the **Create new** button.
    The **Create Hibernate Properties file (.properties)** window appears.
14. Select the parent directory.
15. Name your **.properties** file.
16. Click the **Finish** button.
17. Click the **Setup** button setup the **Configuration file**.
18. Select the path to the target **.cfg.xml** file.
    For more information on how to create a new **.cfg.xml** file, see Section 6.5, “Creating a Hibernate configuration file”.
19. Click the **OK** button.
20. Click the **Finish** button.

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

CHAPTER 7. 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 8.1, “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.

7.1. 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. Open **Git Perspective**.
3. Press **Ctr**+**N**.
   The **New** window appears.
4. Enter **HTML5** in the search field.

5. Select **HTML5 Project**.
6. Click the **Next** button.
   The **New Project Example** window appears.
7. Click the down-arrow in the **Target Runtime** field.
8. Select the server.
9. Click the **Next** button.
10. Name your project.
11. Name your package.
12. Click the **Finish** button.
    Note that it might take some time for the project to generate.

    The **New Project Example** window appears.
13. 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.

7.2. 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.

7.3. 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.

CHAPTER 8. 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.

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

Verification steps

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 local server is now listed in the **Servers** view.

8.2. 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. Check the **Remote** box.
9. Select one of the **Controlled by** options.
10. Check the **Server lifecycle externally managed** 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.

8.3. 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. Start **Project Explorer**.
3. Right-click the target **project → Run as → Run on Server**.
   The **Run on Server** window appears.
4. Ensure that the **Choose an existing server** box is checked.
5. Expand **localhost** to select the server on which to deploy
6. Click the **Finish** button.
7. The **Console** view appears

Your application opens in the internal CodeReady Studio web browser.