Adobe, 345 Park Avenue, San Jose, California 95110, USA.

02/22/2021

Contents

Develop using Eclipse

Introduction

Eclipse can be used to configure the development environment for Adobe Experience Manager (AEM). You can use the Eclipse IDE to develop AEM projects with the help of Maven, Git, and other plugins from the Eclipse marketplace. Along with these plugins, the Eclipse AEM plugin can be used for synchronizing code to a local AEM repository.

Objectives

After completing this course, you will be able to:

- Install and configure Eclipse
- Install and configure the Eclipse AEM plugin
- Import an AEM project in Eclipse
- Synchronize AEM content

Installing and Configuring Eclipse

Eclipse is an open source Integrated Development Environment (IDE) used to edit your project source locally on your file system. For AEM projects, you must ensure Eclipse has the following plugins:

- Maven Integration for Eclipse (M2E)
- AEM plug-in for Eclipse

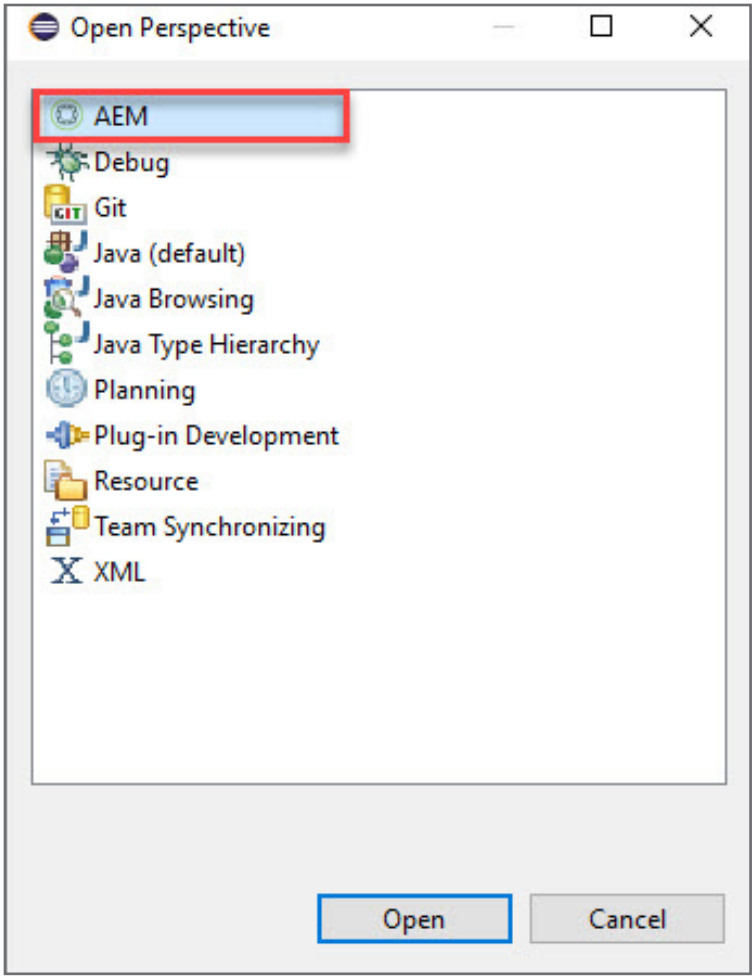
Benefits of AEM Plug-in for Eclipse

The AEM plug-in has the following benefits:

- Synchronizes both content and OSGi bundles
- Supports debugging and code hot swapping
- Includes a project creation wizard to simplify bootstrapping of AEM projects
- Integrates with AEM Services seamlessly through the Eclipse Server Connector
- Easy JCR properties edition

AEM Perspective

The AEM perspective offers complete control over all your AEM projects and Services.



Using AEM Perspective, you can configure an AEM Server to which Eclipse will connect.
The AEM perspective enables you to add and modify nodes and properties in your AEM project through the AEM Console and the JCR properties view.

Exercise 1: Install and configure Eclipse (Local only)

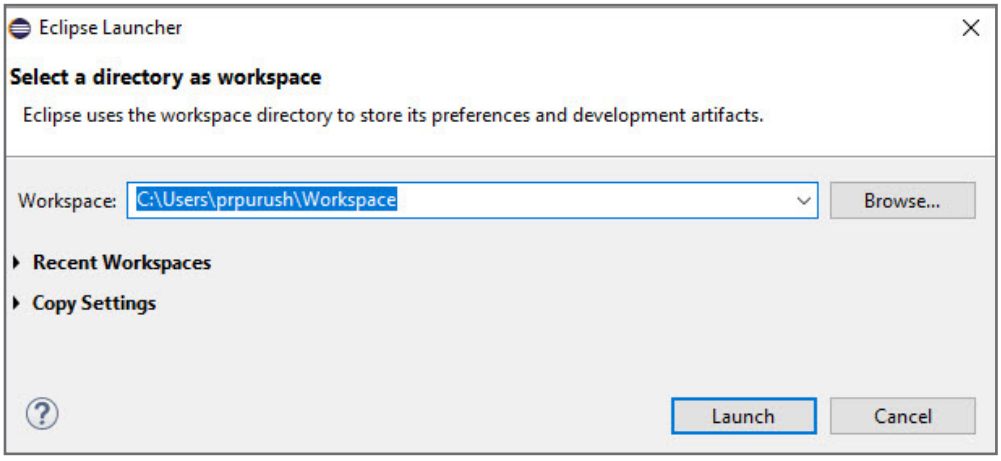
In this exercise, you will install and configure Eclipse.
This exercise includes the following tasks:

1. Install Eclipse
2. Configure Eclipse

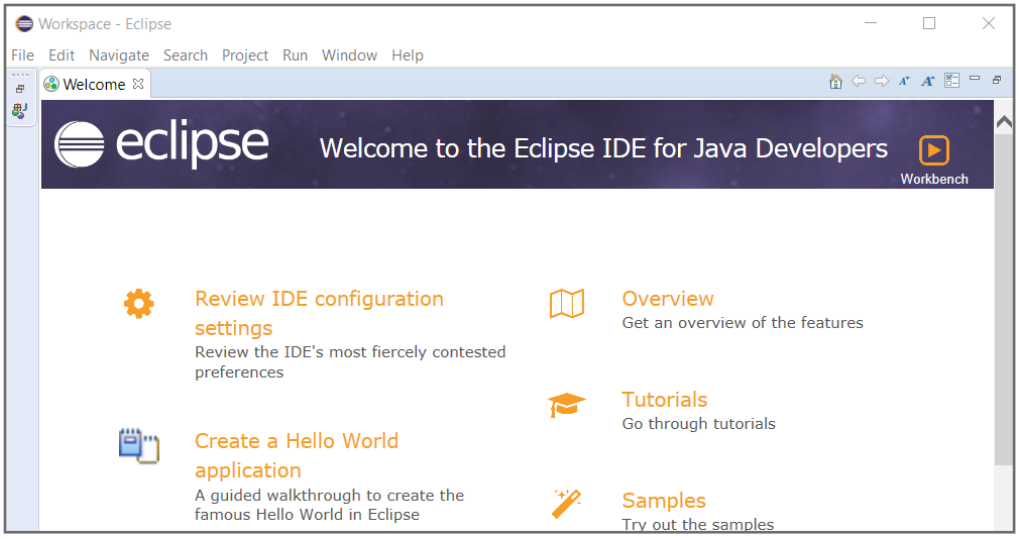
Note: You can skip this exercise if you use a ReadyTech environment because AEM is already installed as part of the image.

Task 1: Install Eclipse

1. You can download Eclipse directly from their website: https://www.eclipse.org/downloads/.
2. Navigate to the directory where you extracted the contents of the Eclipse installation zip file. For example, navigate to **C:\Program Files\Eclipse** on Windows or **Applications/Eclipse** on Mac.
3. Double-click **eclipse.exe** (or **eclipse.app**) to start Eclipse. The Eclipse Launcher opens, as shown:



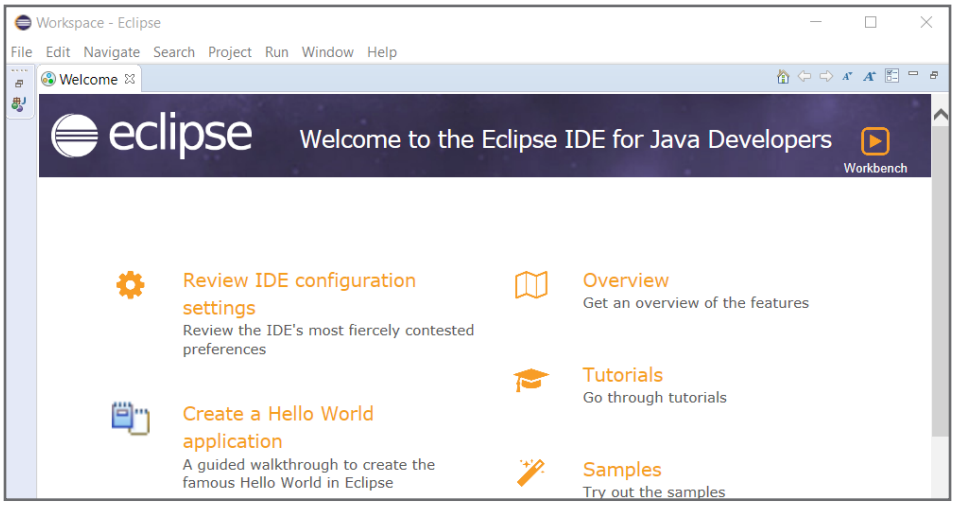
4. Accept the default workspace and click **Launch**. The Eclipse Development Environment opens, as shown:



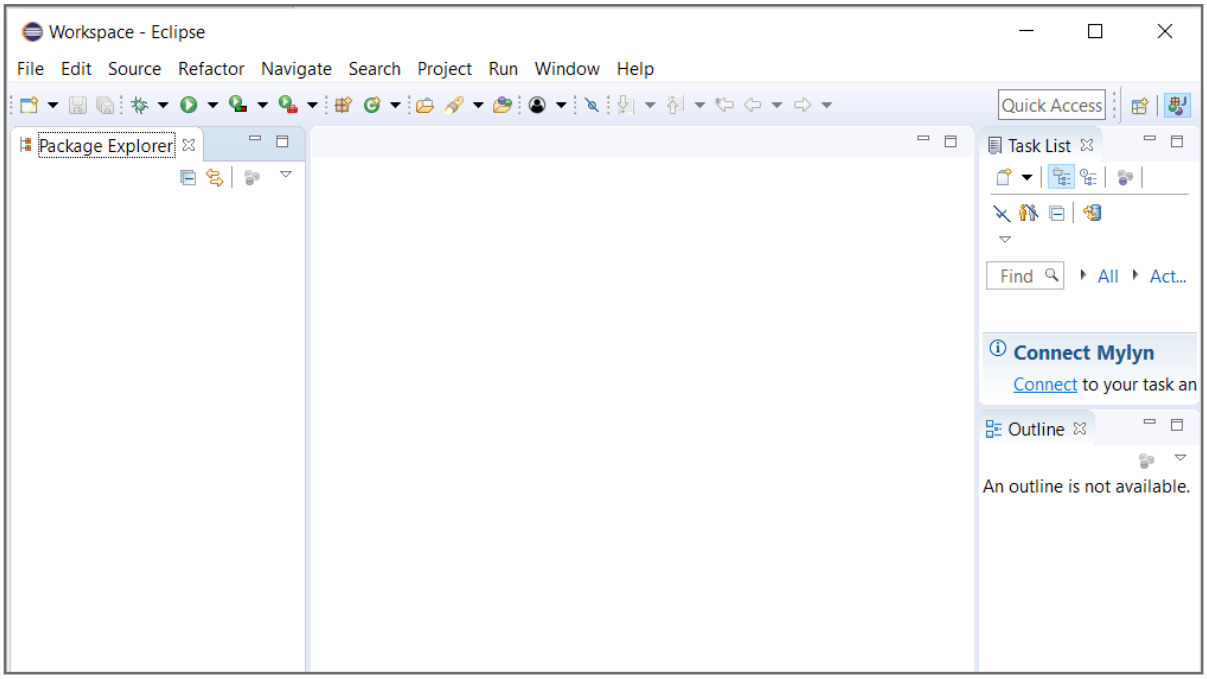
Task 2: Configure Eclipse

In this task, you will configure Eclipse.

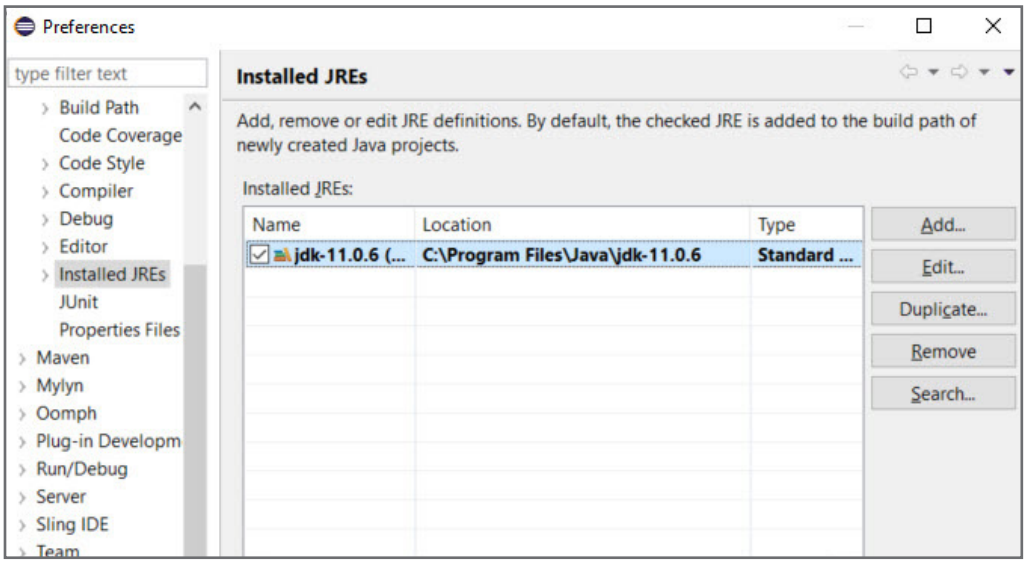
1. Click the **Workbench** logo in the upper-right corner, as shown, to close the Welcome screen.



The Workbench opens, as shown:



2. Verify Eclipse's JRE is set to a JDK by clicking **Window** > **Preferences** > **Java** > **Installed JREs**. You should see a path similar to the one shown in the following screenshot. Otherwise, you must provide the correct directory path to your JDK.



3. Click X in the upper right or click **Cancel** in the lower right to close the Preferences window.

Exercise 2: Install the Eclipse AEM plug-in (Local only)

In this exercise, you will install and configure Eclipse AEM plug-in.

Note: You can skip this exercise if you use a ReadyTech environment because AEM is already installed as part of the image.

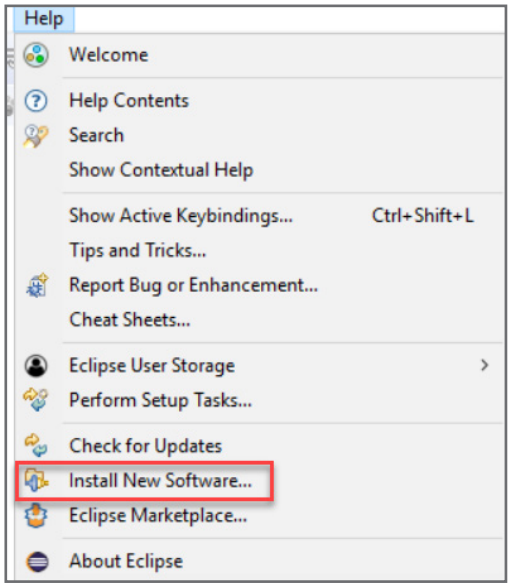
1. Open the URL https://eclipse.adobe.com/aem/dev-tools/ or use the file provided in the **Exercise_Files_TB**.

Note: There are two ways to install the plug-in:Online:
1. You will provide the link to install the plug-in in Eclipse.Offline.
2. You will provide the downloaded plug-in in Eclipse.

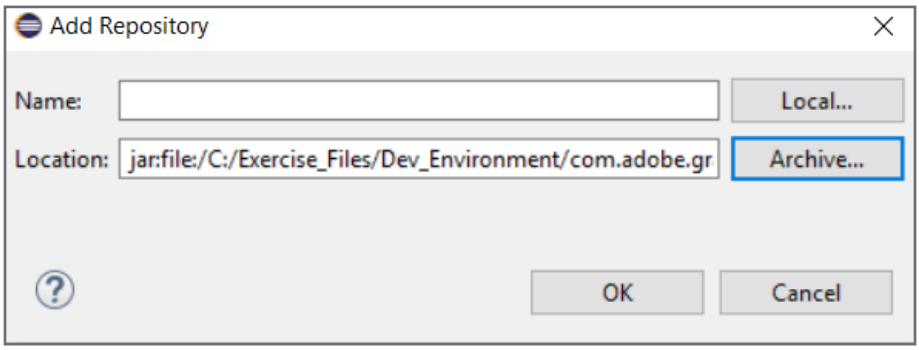
You will perform the offline method in this exercise.

To install your package:

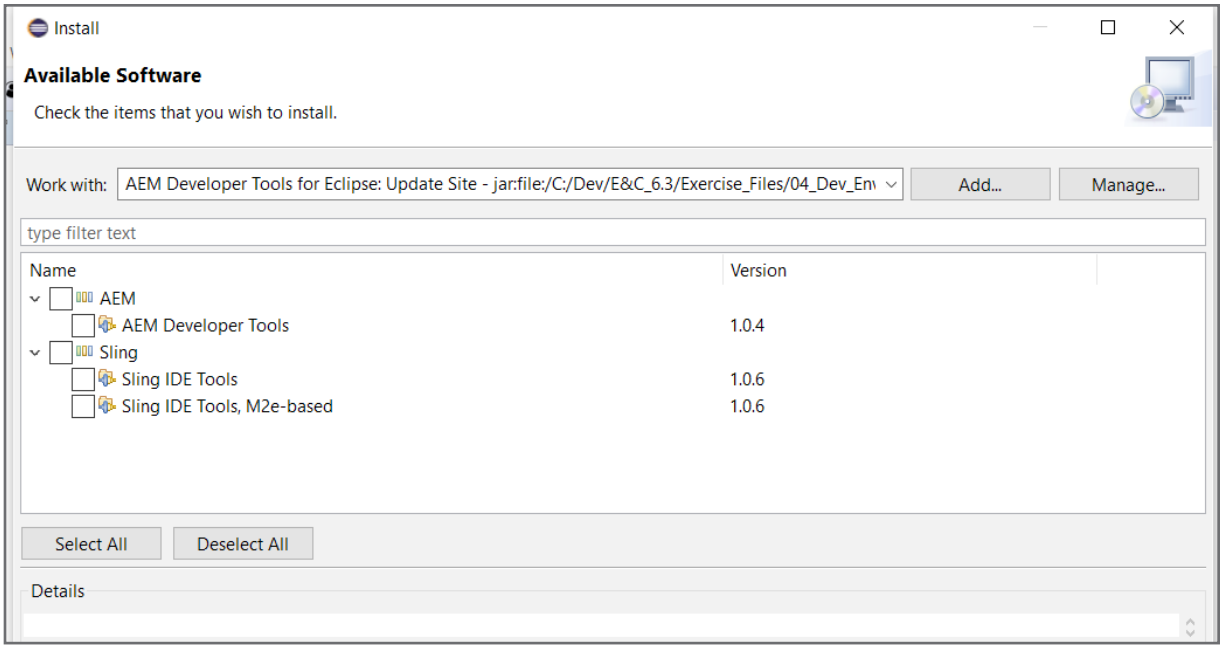
2. Double-click **eclipse.exe** (or eclipse.app) to start Eclipse. The Eclipse Development Environment opens:
3. Select **Help** > **Install New Software,** as shown. The Install window opens.



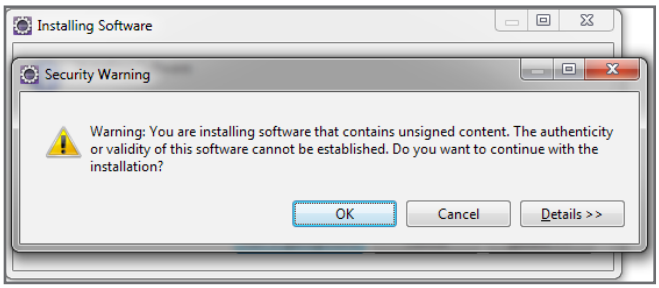
4. Click **Add**. The Add Repository dialog box opens.
5. Click **Archive**.
6. Navigate to the repository archive and select the zip file (**com.adobe.granite.ide.p2update-1.3.0.zip**) provided for the plug-in from **Exercise_Files_TB/Dev_Environment/**.
7. Click **Open** to add the location to the **Location** field in the **Add Repository** window.
8. Click **Add**. The location of the repository is added, as shown:



9. The **Available Software** window opens displaying the items you want to install, as shown:



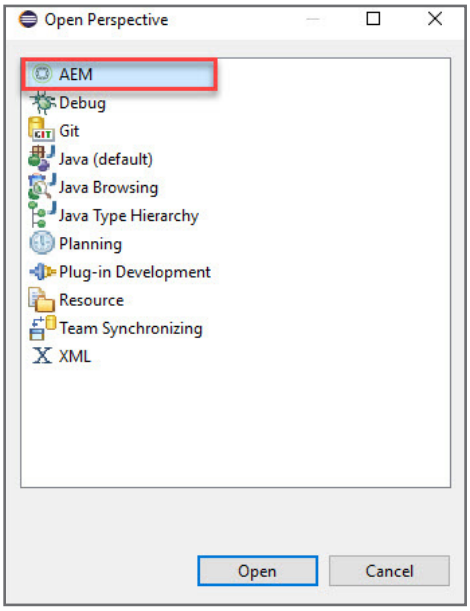
10. Click **Select All** to select AEM and Sling.
11. Click **Next**. The Install Details screen opens.
12. Click **Next**. The Review Licenses screen opens.
13. Click the **I accept the terms of the license agreements** option and then click **Finish**. The **Installing Software** dialog box with a progress bar opens. The installation may take a couple of minutes. You can see the progress of the installation in the lower-right corner of the Eclipse workspace.
14. If a **Security Warning** window pops up, as shown, click **Install anyway** to continue the installation. The **Software Updates** dialog box will re-open until the installation is completed.



15. Click **Restart Now** to restart Eclipse to load the newly installed tools.

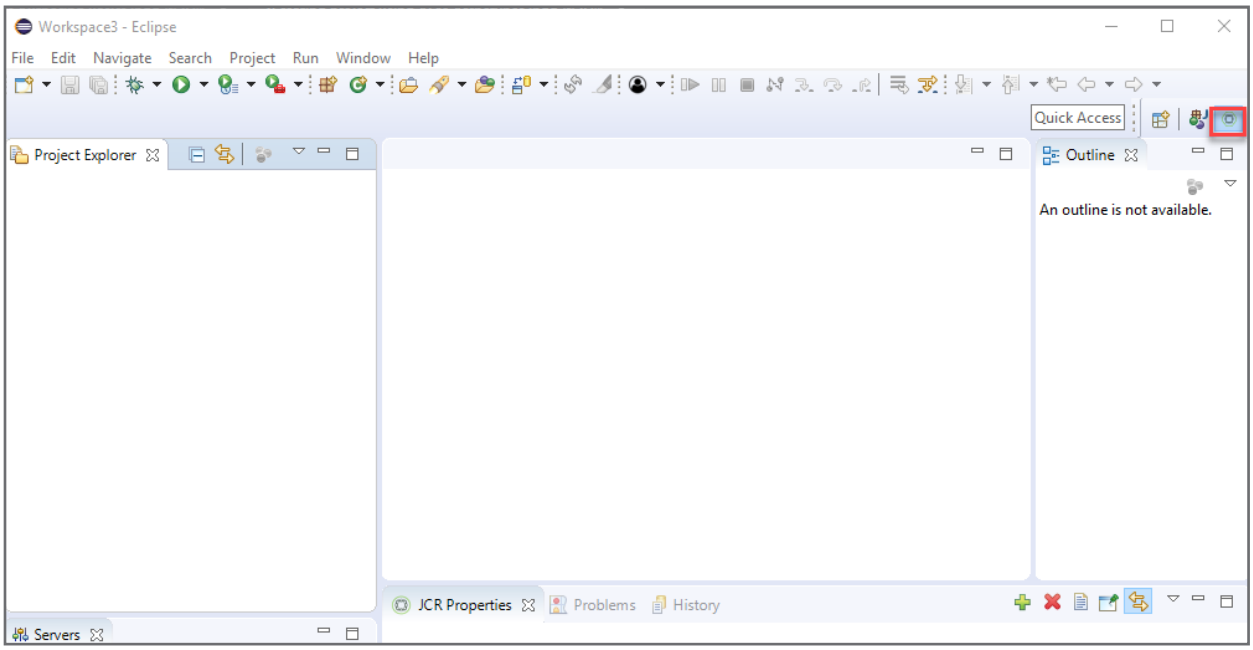
Note: This process takes about a minute. If you see a dialog box that asks if you want to keep the current location of your Eclipse install, click **OK**. Eclipse opens.

16. Click **Workbench** in the upper-right corner. The Workspace opens.
17. A new AEM perspective becomes available in Eclipse. To verify the AEM perspective was added, click **Window** > **Perspective** > **Open Perspective** > **Other...**, which opens the **Open Perspective** window, as shown:



18. Click **AEM** and then click **Open**. This closes the **Open Perspective** window.

19. Notice the AEM perspective icon is visible at the top, as shown:



Note: Keep Eclipse open for the next exercise.

Exercise 3: Import an AEM project

In this exercise, you will import an AEM project using the AEM Archetype.

This exercise includes the following task:

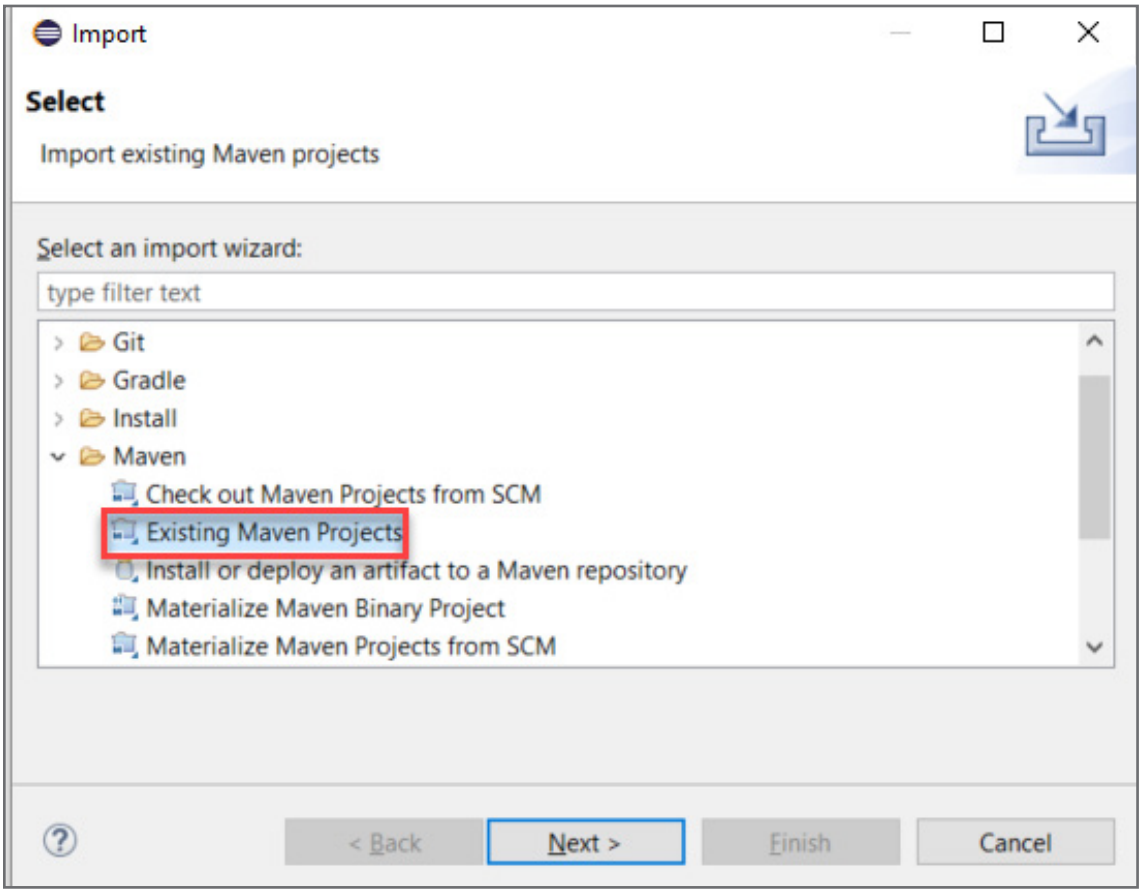
1. Import an AEM project
2. Configure jcr_root folders

Task 1: Import an AEM project

In this task, you will import an AEM project.

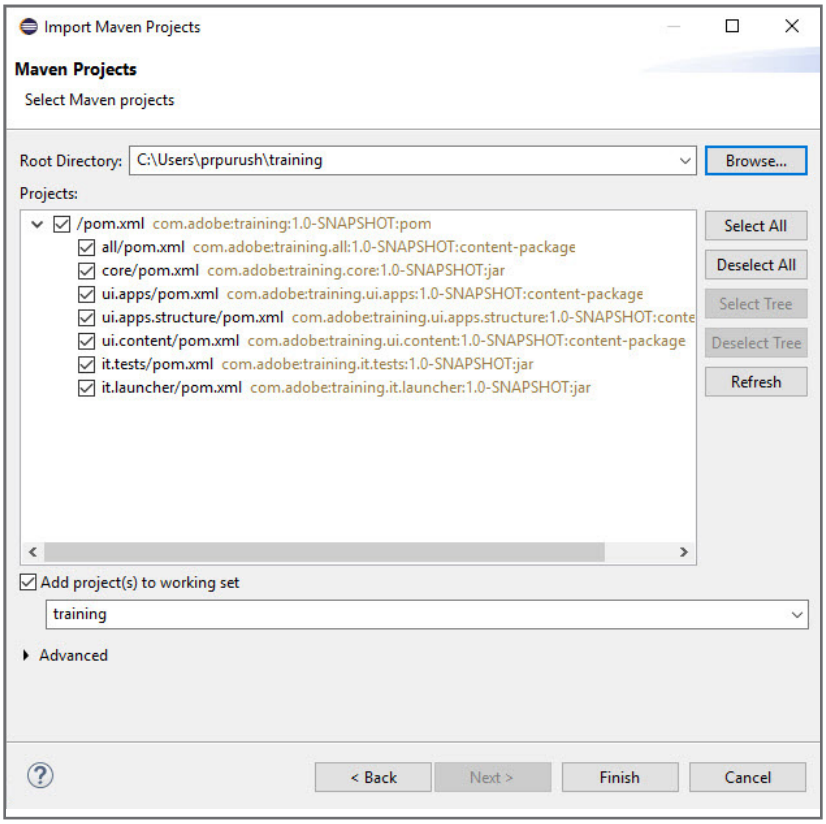
1. Right-click the **Eclipse** icon, and select **Open** from the list. The **Eclipse Launcher** opens .
2. Click **Launch**. The eclipse-workspace – Eclipse IDE window opens.
3. Click **File** > **Import**. The **Import** window opens.

4. Click the down arrow symbol beside **Maven** to expand it.
5. Select **Existing Maven Projects**, and click **Next**, as shown. The **Import Maven Projects** window opens.



6. Click the **Browse** button beside the **Root Directory** field. The **Select Root Folder** window opens.
7. In the **Select Root Folder** window, navigate to the path to your Maven project **/<AEM project>** and click **Ok**.

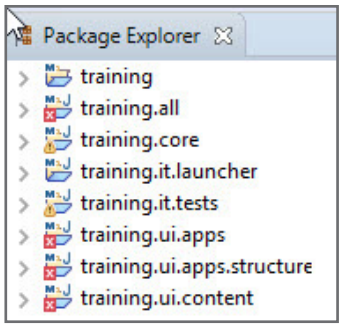
8. Select all the POM files, as shown: :



Note: Your AEM project might have a different name.

9. Ensure that the Add project(s) to working set checkbox is selected.

10. Click **Finish**. The AEM project is imported in the Project Explorer, as shown :



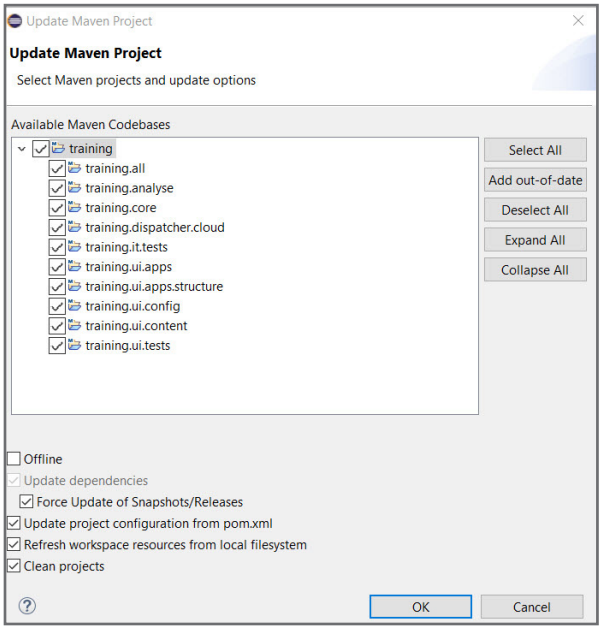
Note: Just like the screenshot above, your project will probably contain errors. These errors can be from a variety of things and you can optionally ignore these in the Eclipse preferences.

11. A **Setup maven plugin connectors** window pops up. Click **Resolve All Later** and then **Finish**. Click **OK** on the **Incomplete Maven Goal Execution** popup.
12. On the **Project Explorer** tab, right-click the parent folder and select **Maven** > **Update Project.** The **Update Maven Project** window opens.
13. Verify your codebase is selected as **training**.

Note: Your Maven codebase might have a different name.

---

14. Select the **Force Update of Snapshots/Releases** checkbox, and click **OK**, as shown. Your project is now updated.

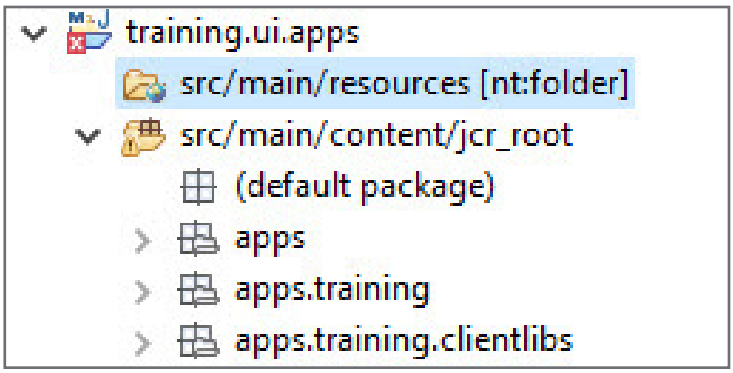


15. Ignore any errors. You will fix them later.

You have successfully imported an AEM project.

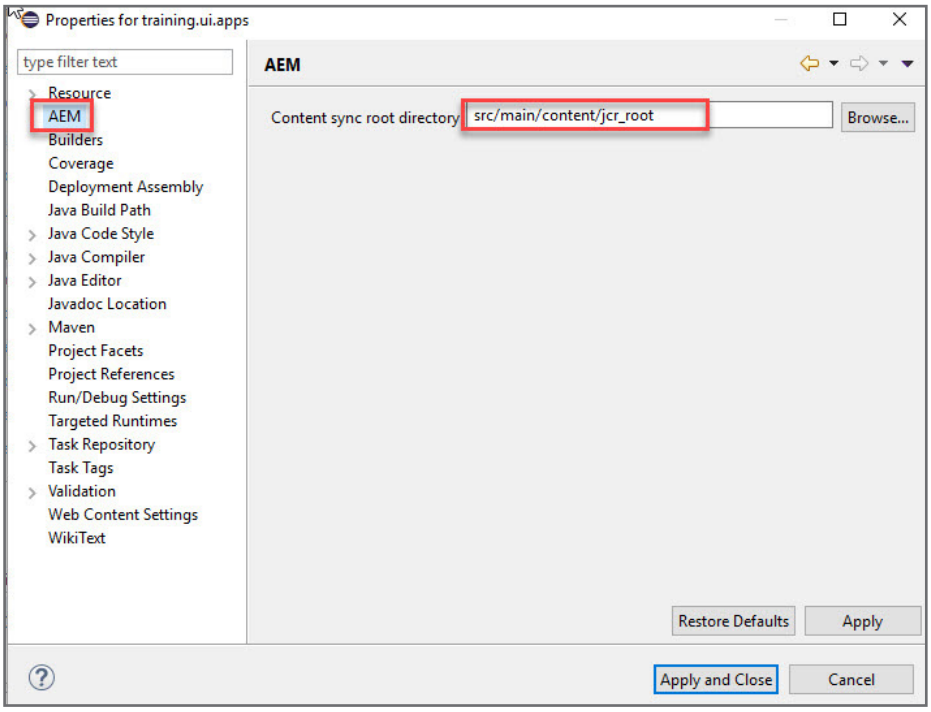
Task 2: Configure jcr_root folders

1. In Eclipse Project Explorer, navigate to: **<AEM project>.ui.apps** and notice **src/main/resources** is synchronized to AEM, as shown:

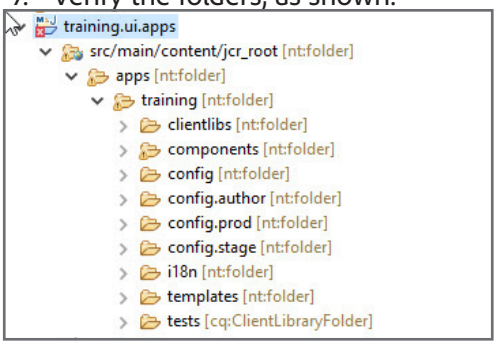


Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

2. To Synchronize **src/main/content/jcr_root** folder, right-click **<AEM project>.ui.apps** folder and choose **Properties** from the menu.
3. Select **AEM**.
4. Click **Browse** and select the content sync root directory: **src/main/content/jcr_root**, as shown:



5. Click **Ok**.
6. Click **Apply** and **Apply and Close.**
7. Verify the folders, as shown:



8. Similarly, right-click **<AEM project>.ui.content** folder, choose **Properties** from the menu and repeat steps 3-6.
9. Similarly, right-click **<AEM project>.ui.config** folder, choose **Properties** from the menu and repeat steps 3-6.

You have successfully configured your jcr_root folders.

Exercise 4: Synchronization tools for Eclipse

The AEM plugin allows you to connect to an AEM server to auto-push changes made in the project into the JCR. This can be done to synchronize content in the ui.apps and ui.content modules with the JCR, but also for hot code swap (such as updating the bundle without Maven builds) on changes made in the core module. The synchronization happens when saving a file in those modules, but can also be manually triggered (as well as changes done in the repository can be manually imported into the project).

The AEM server connection allows you to synchronize modules individually, by using the add/remove resources function.
This exercise includes the following tasks:

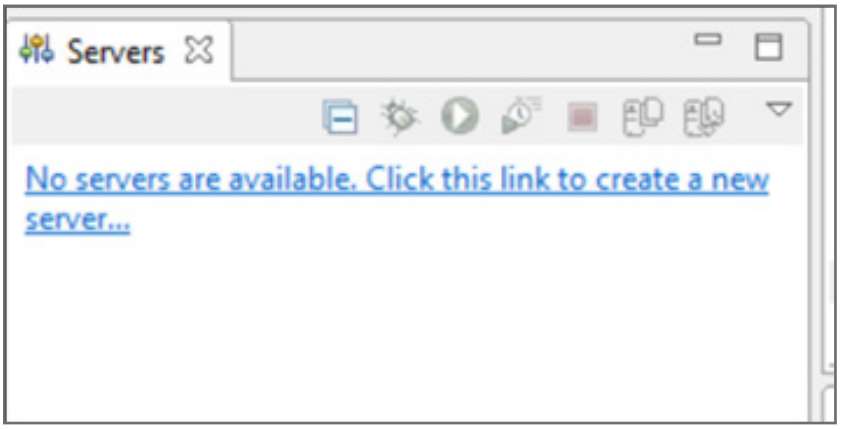
1. Configure the AEM server
2. Sync AEM content

Task 1:Configure the AEM server

In this task, you will configure a connection to the AEM server to enable content synchronization for the ui.apps and ui.content modules, but NOT the core module, since you are using Maven to build and install the code (preferred method).

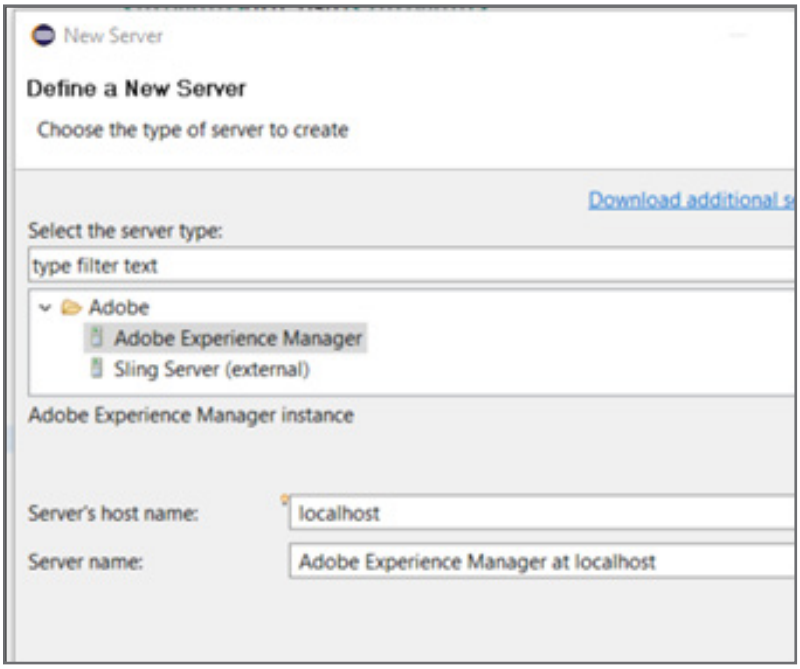
1. In Eclipse, verify **AEM Perspective** is selected in the upper-right corner.

2. On the left-hand side of the Eclipse Workspace below the **Project Explorer tab**, notice the **Servers** tab. Click the **No Servers are available. Click this link to create a new server...** link, as shown:



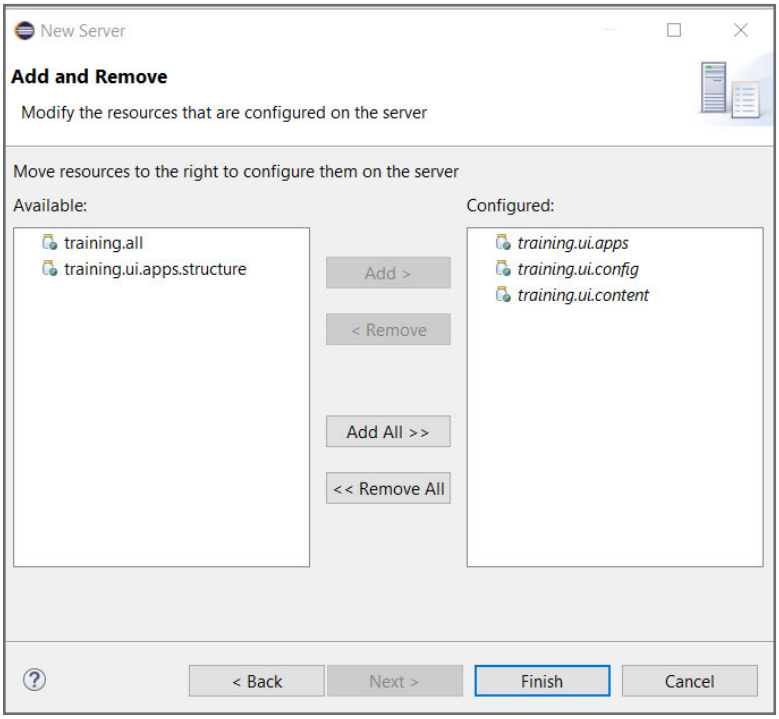
The **Define a New Server** dialog box opens.

3. Select **Adobe** > **Adobe Experience Manager**, as shown:



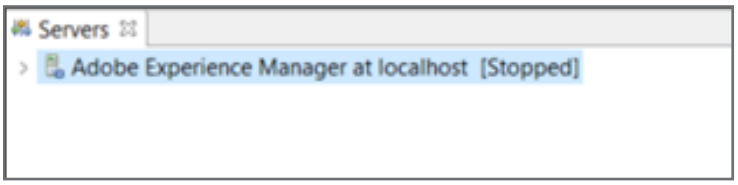
4. Accept the default settings for the **Server's host name** and **Server name** fields.
5. Click **Next**. The **Add and Remove resources** dialog box opens.

6. Select **<AEM project>.ui.apps, <AEM project>.ui.config** and **<AEM project>.ui.content** one by one, and click the **Add >** button to move the specified resources from the **Available** section to the **Configured** column, as shown:



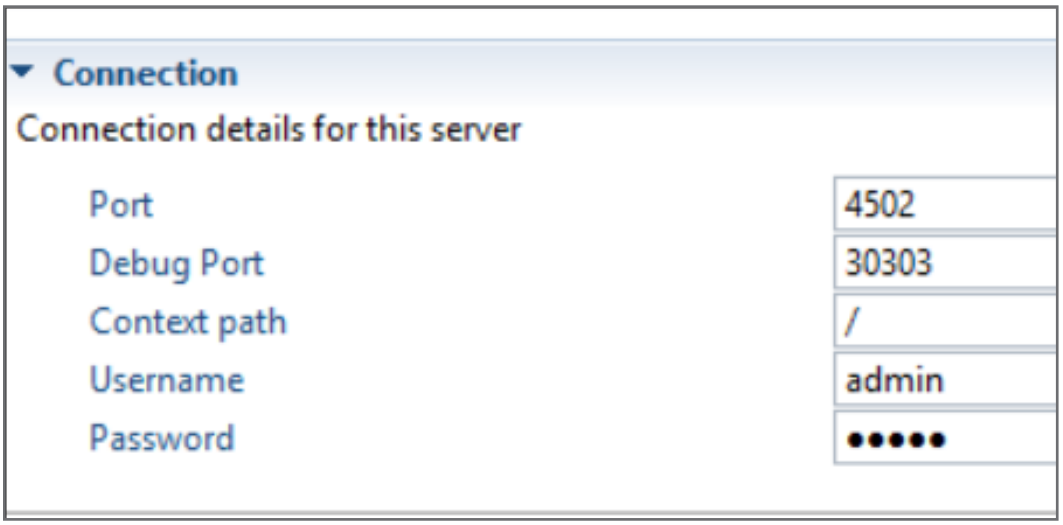
Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

7. Click **Finish** on the **New Server** screen. The AEM Server is now defined.
8. On the left-hand side of the workspace (below the **Project Explorer** tab), click the **Servers** tab and note how **Adobe Experience Manager at localhost [Stopped]** is now available, as shown:



9. To modify the configuration for the AEM Server, double-click the **Adobe Experience Manager at localhost [Stopped]** to open it in the editor.

10. In the **Connection** area, change the current port (listed in the **Port** field, in the **Connection** area) to **4502**, as shown:

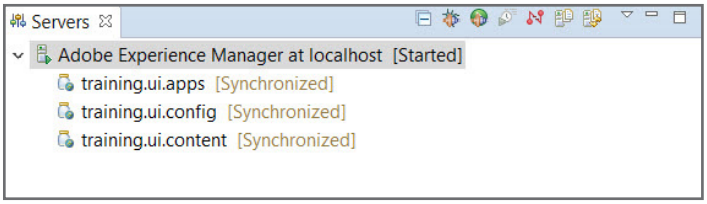


11. Save the changes (**File** > **Save** OR **Ctrl+S**).

You have now created a connection from Eclipse to the AEM server on your computer. You will now start the AEM server.

12. Right-click **Adobe Experience Manager at localhost** on the **Servers** tab, and click **Start**. The server is started.

13. Verify the Server has started, as shown:



The contents are now in synch with the AEM server.

Task 2: Sync AEM content

If your project doesn't have the Helloworld component, select a component of your choice to complete this exercise.

1. In Eclipse Project Explorer, navigate to: **<AEM project>.ui.apps** > **src / main /content/jcr_root** > **apps> training** > **components** > **helloworld**.
2. Double-click **helloworld.html**. The HTML page opens.
3. Add the following line at the end of the **<pre>** tag in the code: "**This is a change in Eclipse**" .

```
17      <h2 class="cmp-helloworld__title">Hello World Component</h2>
18      <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
19          <p class="cmp-helloworld__item-label">Text property:</p>
20          <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
21      </div>
22      <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.mess
23          <p class="cmp-helloworld__item-label">Model message:</p>
24          <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message}"This is a change in Eclipse"</pre>
25      </div>
26  </div>
27
```

4. Save the changes.

Note: When you save helloworld.html, the AEM Server connection in Eclipse exports helloworld.html to the JCR.

5. In **CRXDE Lite**, browse to **apps** > **training** > **components** > **content** > **helloworld** > **helloworld. html** and double-click **helloworld.html.** The HTML page opens.
6. Verify the change in CRXDE Lite:

```
10      Unless required by applicable law or agreed to in writing, software
11      distributed under the License is distributed on an "AS IS" BASIS,
12      WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13      See the License for the specific language governing permissions and
14      limitations under the License.
15  */-->
16  <div class="cmp-helloworld" data-cmp-is="helloworld">
17      <h2 class="cmp-helloworld__title">Hello World Component</h2>
18      <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
19          <p class="cmp-helloworld__item-label">Text property:</p>
20          <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
21      </div>
22      <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.message}">
23          <p class="cmp-helloworld__item-label">Model message:</p>
24          <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message}"This is a change in Eclipse"</pre>
25      </div>
26  </div>
```

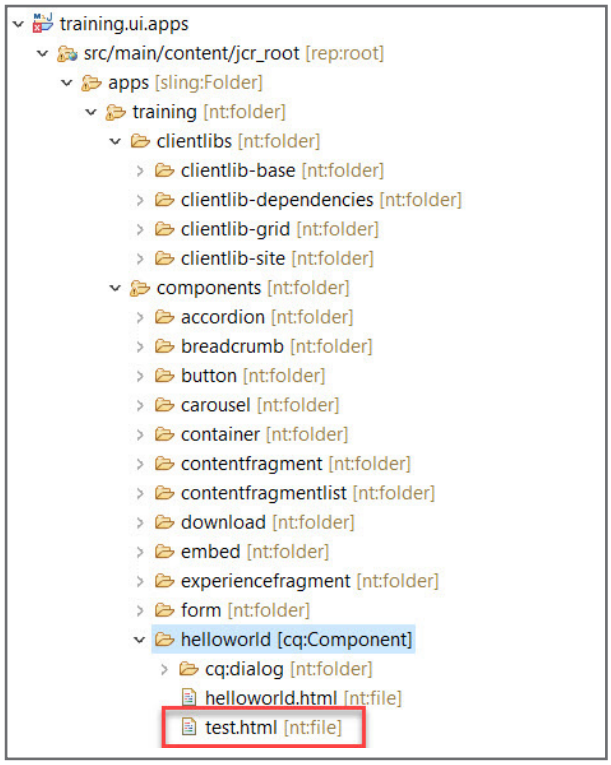
Note: Any files saved in your project will auto sync with AEM based on the filter.xml file in each module that is configured with the AEM connection.

7. You will now sync changes made in AEM back into your Maven project. In **CRXDE Lite**, navigate to **apps** > **training** > **components** > **helloworld**.
8. Right-click the **helloworld** component node, and select **Create** > **Create File**.
9. Enter the name as **test.html.**

10. Click **OK**. The file is created. Click **Save All** in the upper left to save the changes.

Note: Ensure to click Save All after every change in CRXDE lite.

11. In **Eclipse**, under **<AEM project>.ui.apps**, select **/src/main/content/jcr_root/apps/training/components/helloworld**, right-click and select **Import from Server.**
12. Accept the default settings and click **Finish**. The selected node and its children are imported from the server.
13. Under **<AEM project>.ui.apps**, navigate to **/apps/training/components/helloworld** and verify **test.html** appears in your project, as shown:



Note: Step 11 is a very typical process in AEM Java development to sync new content from the JCR to Eclipse. Remember that Eclipse is our master repository locally and anything created in the JCR that is a part of our project must be pulled back down into Eclipse. This is a very common process with config nodes, dialog structures, components, and clientlibs.

Tip: If you create something in CRXDE Lite that you want to keep, you must sync it back to Eclipse using the process above.

References

You can use the following links for more information on:

- Development Tools for AEM Projects:

https://docs.adobe.com/content/help/en/experience-manager-learn/cloud-service/local-development-environment-set-up/development-tools.html