

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe. Adobe assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Acrobat, the Creative Cloud logo, and the Adobe Marketing Cloud logo are either registered trademarks or trademarks of Adobe in the United States and/or other countries.

All other trademarks are the property of their respective owners.

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

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

# Contents

<b>Configuring the Adobe Experience Manager Platform</b>	<b>3</b>
OSGi configurations	4
Exercise 1: Create an OSGi config	6
Task 1: Examine the ui.config module	6
Task 2: Create OSGi configs	7
Task 3: Install via command line	8
Supported Runmodes	10
Exercise 2: Create an environment specific config	12
Task 1: Verify AEM run mode is dev author	12
Task 2: Create an environment specific config	15
Task 3: Install via command line	16
Creating system users	17
Exercise 3: Use Repointit to create a service user	20
Task 1: Create an OSGi configuration file in JSON format	20
Task 2: Install via command line	21
Exercise 4: Create a service user Mapper	23
Task 1: Create an OSGi configuration file in JSON format	23
Task 2: Install and verify the configuration	24
References	26

# Configuring the Adobe Experience Manager Platform

---

## Introduction

OSGi configurations are used by developers and administrators to manage Adobe Experience Manager (AEM) Service settings in conjunction with supported runmodes. In this module you will learn how to create OSGi configurations and how runmodes affect them. You will also learn about special repository initialization configurations for a project.

## Objectives

After completing this course, you will be able to:

- Explain OSGi configurations
- Explain the supported runmodes
- Explain how OSGi configurations are organized to match runmodes
- Create an OSGi configuration
- Start AEM with an environment run mode
- Explain repository initialization
- Create a system user
- Create a system user mapper

## OSGi configurations

---

OSGi is a fundamental element in the technology stack of AEM. OSGi also supports the modular deployment of bundles. You can stop, install, and start these bundles individually. The interdependencies are handled automatically. Each OSGi component is contained in one of the deployed bundles, which helps manage applications easily.

OSGi configurations are system parameters and settings contained in bundles that you can manage and configure.

OSGi configuration should be committed to source control rather than through the Web Console. Techniques include:

- Making the necessary changes on the developer's local AEM environment with the AEM Web Console's configuration manager and then exporting the results to the AEM project on the local file system
- Creating the OSGi configuration manually in the AEM project on the local file system, referencing the AEM console's configuration manager for the property names.

## Create OSGi configurations with JSON files

The recommended best practice is to define OSGi configurations by using JSON configuration files. You should define the configurations in the AEM Project's code packages (ui.config) as configuration files (.cfg.json).

OSGi configurations target OSGi components through their Persistent Identity (PID), which by default takes the OSGi component's Java class name. The OSGi configurations are deployed as JSON files with the following file naming convention:

```
<PID><~unique qualifier for factory services>.cfg.json
```


For example, to provide the configuration for the Day CQ Root Mapping Servlet, as implemented by **com.day.cq.commons.servlets.RootMappingServlet** (a singleton service), you can define an OSGi configuration at: **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig > config > com.day.cq.commons.servlets.RootMappingServlet.cfg.json**.

The contents of the configuration file will be in json where the key is the OSGi property name and the value is what should be set.

```
{  
    "rootmapping.target" : "/sites.html"  
}
```

OSGi configuration changes are applied immediately to the relevant OSGi component.

---

 **Note:** The prior versions of AEM supported OSGi configuration files using different file formats such as .cfg, .config and as XML sling:OsgiConfig resource definitions. These formats are superseded by the cfg.json OSGi configuration format.

---

## Exercise 1: Create an OSGi config

---

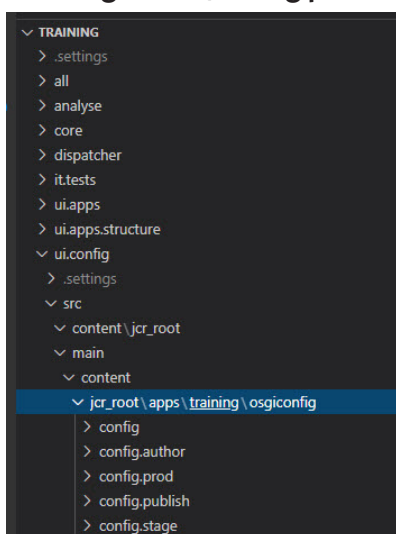
**Scenario:** As a developer, you need to create OSGi configurations in the JCR using /apps. In a typical development process, there are multiple servers designated for specific tasks in your infrastructure. Each server could have different configurations depending on the service and deployment type.

This exercise includes the following tasks:

1. Examine runmode-specific config folders
2. Create OSGi configs
3. Install via command line

Task 1: Examine the ui.config module

1. Open the IDE containing your maven project for this course if not already opened.
2. In the IDE navigation on the left, navigate to **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig**.
3. You will notice several config.\* folders that hold OSGi configurations. For example: **config**, **config.author**, **config.publish**, **config.prod**, **config.stage** folders, as shown:



4. Open a few of the folders and investigate the OSGi configuration definitions, as shown:



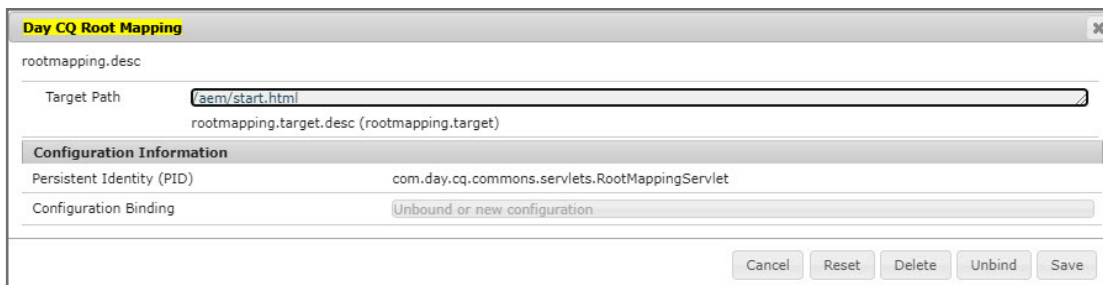
```

1 {
2   "mobile.resourceTypes": [
3     "training/components/page"
4   ],
5   "README": "Indicate which page resource types should display the mobile emulators."
6 }
7

```

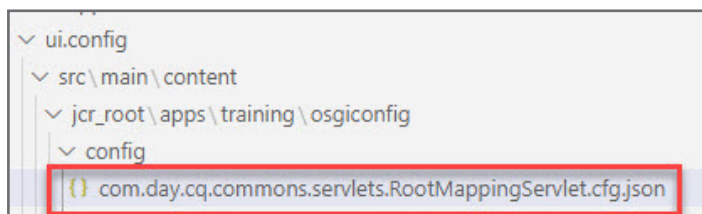
## Task 2: Create OSGi configs

1. Open a browser and go to <http://localhost:4502/system/console/configMgr>. The Adobe Experience Manager Web Console Configuration page opens
2. Search on the web page for **Day CQ Root Mapping**. The search string's corresponding words will become orange, indicating a match for your search string.
3. Click the **Day CQ Root Mapping** configuration to open it in a dialog window, as shown:



This configuration shows what is currently set for this OSGi configuration. Although you could update the values directly in this window, it will not be a part of your maven project. If you would like the configuration to be a part of your project, you need to create a corresponding json file in the ui.config module.

4. In your IDE, navigate to **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig**.
5. Right-click the folder **config** and choose **New file**.
6. Enter the name as **com.day.cq.commons.servlets.RootMappingServlet.cfg.json**, as shown:



7. Double-click **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** to open it in the editor.

- To add the json properties to your file, open up the **Exercise\_Files-EC** folder for this course and navigate to **ui.config/src/main/content/jcr\_root/apps/training/osgiconfig/config**.
- Open the file **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** using a text editor. Copy the contents and paste it to the file **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** in your IDE, as shown:


```
ui.config > src > main > content > jcr_root > apps > training > osgiconfig > config > {} com.day.cq.commons.servlets.RootMappingServlet.cfg.json > ...
1 {
2   "rootmapping.target":"/sites.html"
3 }
```

- Select **File > Save** from the menu to save the file.

Now that the new json configuration file is created, we need to use **Maven** to build and install it into the local AEM Server.

### Task 3: Install via command line

- Open a command prompt to the location of your Maven project. For example: **C:/adobe/<myproject >**

 **Note:** If you are using an IDE with an integrated terminal such as Visual Studio Code, you can run your Maven commands there instead since it is already open to your project.

- In the command prompt run the command:

```
$ mvn clean install -Padobe-public -PautoInstallSinglePackage
```

**TIP:** If you would like the install process to occur faster, you can go into the desired module and run a more specific profile. This allows for only that module to be installed rather than the entire project.

For example: Core - use `autoInstallBundle` , ui.apps - use `autoInstallPackage`, ui.content - use `autoInstallPackage`

- Your project has now successfully installed into your local AEM Server.

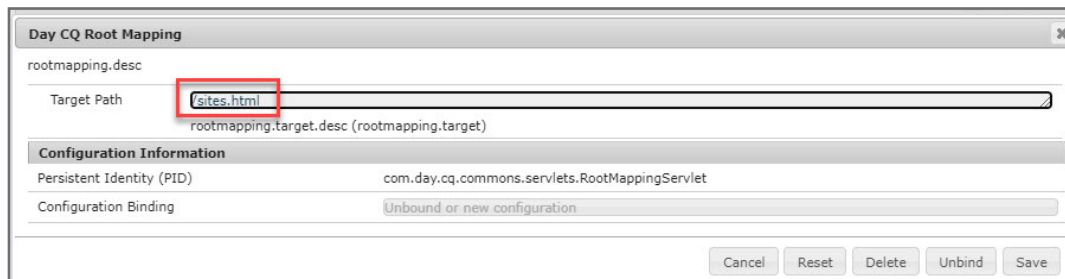
```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

[INFO] Reactor Summary for training 1.0-SNAPSHOT:
[INFO]
[INFO] training ..... SUCCESS [ 0.662 s]
[INFO] TrainingProject - Core ..... SUCCESS [ 13.150 s]
[INFO] TrainingProject - Repository Structure Package ..... SUCCESS [ 1.740 s]
[INFO] TrainingProject - UI apps ..... SUCCESS [ 23.782 s]
[INFO] TrainingProject - UI content ..... SUCCESS [ 20.758 s]
[INFO] TrainingProject - UI config ..... SUCCESS [ 0.575 s]
[INFO] TrainingProject - All ..... SUCCESS [ 1.511 s]
[INFO] TrainingProject - Integration Tests ..... SUCCESS [ 13.059 s]
[INFO] TrainingProject - Dispatcher ..... SUCCESS [ 0.664 s]
[INFO] TrainingProject - UI Tests ..... SUCCESS [ 0.629 s]
[INFO] TrainingProject - Project Analyser ..... SUCCESS [ 36.316 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:00 min
[INFO] Finished at: 2021-03-19T11:40:26-07:00
[INFO] -----
```



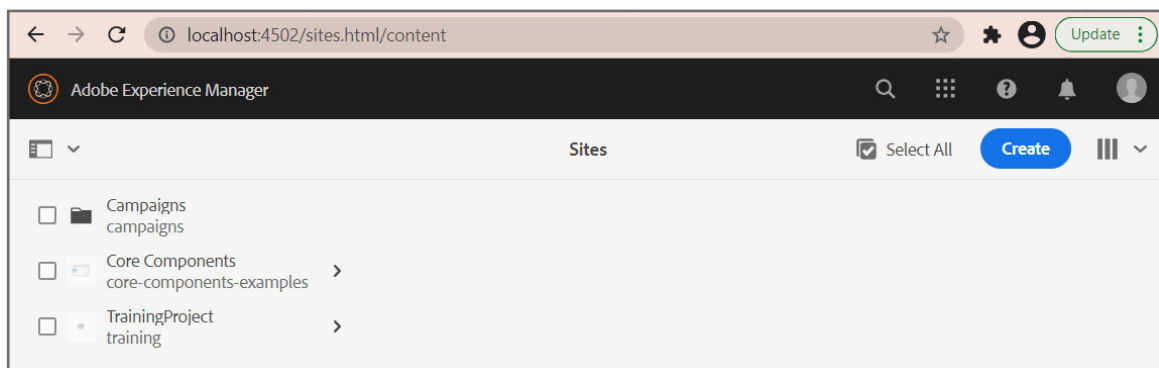
Now that the configurations have been installed, we can go to the Web Console to verify the configurations were set correctly.

4. Open a browser and go to <http://localhost:4502/system/console/configMgr>. The AEM Web Console Configuration page opens
5. Search on the web page for **Day CQ Root Mapping**. The search string's corresponding words will become orange, indicating a match for your search string.
6. Click the **Day CQ Root Mapping** configuration to open it in a dialog window, as shown:



The Target Path value should be **/sites.html**. This value matches the Day CQ Root Mapping configuration stored in the config folder.

7. On your browser and navigate to <http://localhost:4502>  
The root access should redirect to <http://localhost:4502/sites.html/content>, as shown:



You now know how to create different OSGi configurations for specific runmodes.

## Supported Runmodes

---

In existing AEM solutions, customers have the option of running services with arbitrary run modes and apply OSGI configuration or install OSGI bundles to those specific services. Run modes that are defined typically include the "service" (author and publish) and the environment (dev, stage, and prod), but there could be more (for example, local-dev, QA, and so on).

AEM as a Cloud Service on the other hand is more opinionated about which run modes are available and how OSGI bundles and OSGI configuration can be mapped to them:

- OSGI configuration run modes must reference dev, stage, and prod for the environment or author, publish for the service. A combination of <service>.<environment\_type> is supported and must be used in this particular order (for example, author.dev or publish.prod). The OSGI tokens should be referenced directly from code rather than using the getRunModes method, which will no longer include the environment\_type at runtime.
- OSGI bundles run modes are limited to the service (author, publish). You should install per-run mode OSGI bundles in the content package under install/author OR install/publish.

Dev changes to Development to make all items below consistent (i.e Staging, Production, Publish, etc.):  
The supported runmode configurations are:

- config (The default, applies to all AEM services)
- config.author (Applies to all AEM Author services)
- config.author.dev (Applies to AEM Dev Author services)
- config.author.stage (Applies to AEM Staging Author services)
- config.author.prod (Applies to AEM Production Author services)
- config.publish (Applies to AEM Publish services)
- config.publish.dev (Applies to AEM Dev Publish services)
- config.publish.stage (Applies to AEM Staging Publish services)
- config.publish.prod (Applies to AEM Production Publish services)
- config.dev (Applies to AEM Dev services)
- config.stage (Applies to AEM Staging services)
- config.prod (Applies to AEM Production services)

The OSGI configuration that has the most matching runmodes is used.

When developing locally, a runmode startup parameter can be passed in to specify which runmode OSGI configuration will be used. For example:

```
java -jar aem-author-4502.jar -r author,dev -gui
```

## Exercise 2: Create an environment specific config

**Scenario:** As a developer you need to:

- Be able to start AEM with an environment run mode using the command line
- Create multiple sets of custom OSGi configurations to match supported run modes in order to specify different groups of settings for each Service or environment you are deploying

This exercise includes the following tasks:

1. Start AEM with an environment run mode
2. Create an environment specific config
3. Install via command line

**Task 1:** Verify AEM run mode is dev author

In this task, you will verify if your AEM author Service is using an environment run mode called **dev author**.

1. Verify your AEM author service is running.
2. Open a browser and go to, <http://localhost:4502/system/console>. The AEM Web Console Configuration page opens.
3. Navigate to **Status > Sling Settings**.
4. In the Apache Sling Settings, observe the current run modes your author Service is using (**dev**, **s7connect**, **crx3**, **author**, **samplecontent**, and **crx3tar**):

**Adobe Experience Manager Web Console**  
**Sling Settings**

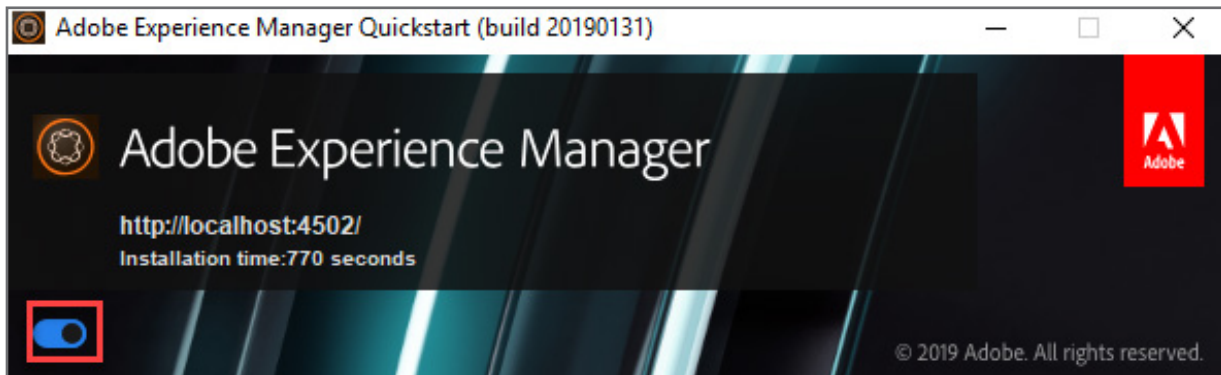
Main OSGi Sling Status Web Console

Date: March 22, 2021 at 2:38:49 PM PDT

Apache Sling Settings

Sling ID = 0b1d19f0-b88a-441d-b668-d85253d5b04e  
Sling Name = Instance 0b1d19f0-b88a-441d-b668-d85253d5b04e  
Sling Description = Instance with id 0b1d19f0-b88a-441d-b668-d85253d5b04e and run modes [dev, s7connect, crx3, author, samplecontent, crx3tar]  
Sling Home = C:\adobe\aeem-sdk\crx-quickstart  
Sling Home URI = file:///C:/adobe/aeem-sdk/crx-quickstart/  
**Run Modes = [dev, s7connect, crx3, author, samplecontent, crx3tar]**


5. If not, shut down your running AEM author Service by clicking the **ON / OFF** toggle button in the GUI window, as shown:



If you are running AEM using the command line, use **CTRL+C** (in Windows) in your command window to shut down AEM.

6. Start AEM using the command line in order to use an environment run mode. This is because the environment run modes can be generated using command line parameters. Navigate to the directory on your machine where your author Service quickstart file resides.

---

 **Note:** If you are using a ReadyTech Service, this directory should be **C:/adobe/aem-sdk/**.


---

7. Start AEM again using the following command that specifies the author and dev run modes:

```
java -jar aem-sdk-quickstart.jar -r author,dev -gui
```

8. Verify your AEM author service started, this time with a command window available to view details of the startup. In addition, the GUI window will be available.

---

 **Tip:** Be patient as it may take up to two minutes for your Service to start.

---

9. Sign in to AEM again.
10. Go to, <http://localhost:4502/system/console> and navigate to **Status > Sling Settings**.

11. In the Apache Sling Settings, observe the current run modes your author Service is using (**dev**, **s7connect**, **crx3**, **author**, **samplecontent**, and **crx3tar**):

## Adobe Experience Manager Web Console

### Sling Settings

[Main](#) [OSGi](#) [Sling](#) [Status](#) [Web Console](#)

Date: March 22, 2021 at 2:38:49 PM PDT

Apache Sling Settings

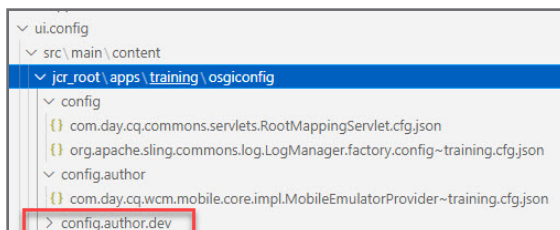
Sling ID = 0b1d19f0-b88a-441d-b668-d85253d5b04e  
Sling Name = Instance 0b1d19f0-b88a-441d-b668-d85253d5b04e  
Sling Description = Instance with id 0b1d19f0-b88a-441d-b668-d85253d5b04e and run modes [dev, s7connect, crx3, author, samplecontent, crx3tar]  
Sling Home = C:\adobe\cem-sdk\crx-quickstart  
Sling Home URI = file:/C:/adobe/aem-sdk/crx-quickstart/  
**Run Modes = [dev, s7connect, crx3, author, samplecontent, crx3tar]**

## Task 2: Create an environment specific config

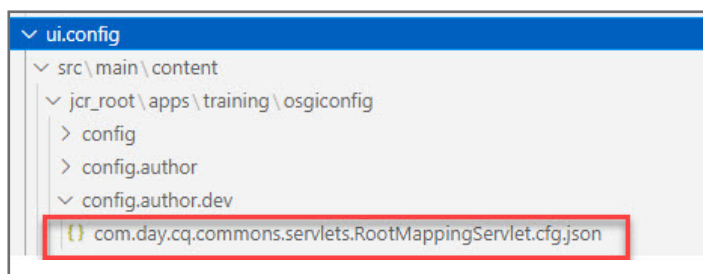
In this task, you will change the default “root” page for dev environments to be CRXDE Lite.

To create another root mapping OSGi configuration node:

1. Open **CRXDE Lite** using <http://localhost:4502/crx/de> or from AEM by navigating to **Tools > CRXDE Lite**.
2. In your IDE, navigate to **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig**.
3. Right-click the folder **osgiconfig** and choose **New Folder**.
4. Enter the name as **config.author.dev**.



5. Right-click the folder **config.author.dev** and choose **New File**.
6. Enter the name as **com.day.cq.commons.servlets.RootMappingServlet.cfg.json**, as shown:



7. Double-click **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** to open it in the editor.
8. To add the json properties to your file, open up the **Exercise\_Files-EC** folder for this course and navigate to **ui.config/src/main/content/jcr\_root/apps/training/osgiconfig/config.author.dev**.
9. Open the file **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** using a text editor. Copy the contents and paste it to the file **com.day.cq.commons.servlets.RootMappingServlet.cfg.json** in your IDE, as shown:



10. Select **File > Save** from the menu to save the file.

Now that the new json file and the configuration file are created, we need to use **Maven** to build and install them into the local AEM Server

### Task 3: Install via command line

1. Open a command prompt to the location of your Maven project.

For example: **C:/adobe/< myproject >**

---

**Note:** If you are using an IDE with an integrated terminal such as Visual Studio Code, you can run your Maven commands there instead since it is already open to your project.

---

2. In the command prompt run the command:

```
$ mvn clean install -Padobe-public -PautoInstallSinglePackage
```

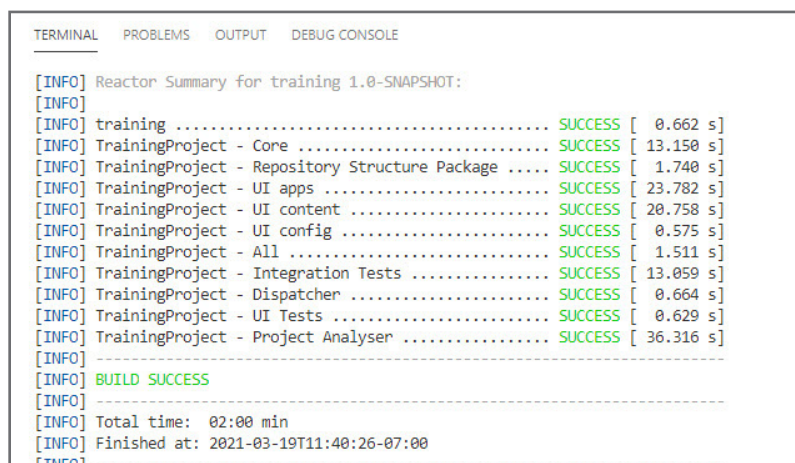
---

**TIP:** If you would like the install process to occur faster, you can go into the desired module and run a more specific profile. This allows for only that module to be installed rather than the entire project.

For example: Core - use `autoInstallBundle` , ui.apps - use `autoInstallPackage`, ui.content - use `autoInstallPackage`

---

3. Your project has now successfully installed into your local AEM Server.



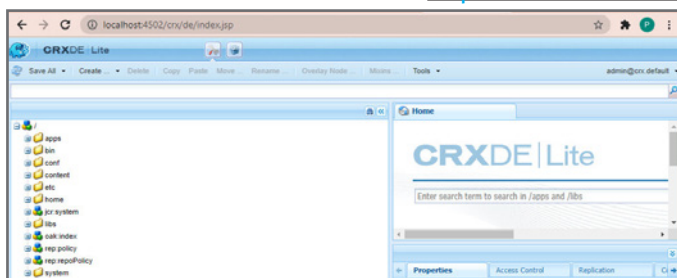
```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

[INFO] Reactor Summary for training 1.0-SNAPSHOT:
[INFO]
[INFO] training ..... SUCCESS [ 0.662 s]
[INFO] TrainingProject - Core ..... SUCCESS [ 13.150 s]
[INFO] TrainingProject - Repository Structure Package ..... SUCCESS [ 1.740 s]
[INFO] TrainingProject - UI apps ..... SUCCESS [ 23.782 s]
[INFO] TrainingProject - UI content ..... SUCCESS [ 20.758 s]
[INFO] TrainingProject - UI config ..... SUCCESS [ 0.575 s]
[INFO] TrainingProject - All ..... SUCCESS [ 1.511 s]
[INFO] TrainingProject - Integration Tests ..... SUCCESS [ 13.059 s]
[INFO] TrainingProject - Dispatcher ..... SUCCESS [ 0.664 s]
[INFO] TrainingProject - UI Tests ..... SUCCESS [ 0.629 s]
[INFO] TrainingProject - Project Analyser ..... SUCCESS [ 36.316 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 02:00 min
[INFO] Finished at: 2021-03-19T11:40:26-07:00
[INFO] -----
```

Now that the OSGi component and OSGi configurations have been installed, we can verify the changes.

4. On your browser and navigate to <http://localhost:4502>

The root access should redirect to <http://localhost:4502/crx/de/index.jsp>, as shown:





## Creating system users

---

A service is a piece or collection of functionality. Examples of services include the Sling queuing system, Tenant Administration, Event Handlers, custom Workflow processes, and a Message Transfer System. Each service is identified by a unique service name. Because a service is implemented as a component in an OSGi bundle, services are named by the bundles providing them.

Adobe recommends running a service within the calling user's permission context when possible. However, background tasks are often not associated with a specific user. As a result, Apache Sling provides the following three part mechanism for service authentication and authorization.

- Service user
- Service ID
- ServiceUserMapper

### SlingRepositoryInitializer

The Apache Sling SlingRepositoryInitializer enables you to create resources at startup before the Sling Repository Service is registered as an OSGi service. As a result, the application logic can take the existence of those resources for granted. Repointit statements should be under change control and therefore are defined in OSGi configurations.

The **repointit** Repository Initialization language:

- Creates paths, service users and Access Control Lists in a content repository
- Registers JCR namespaces and node types

An example of repoint statements:

```
create service user training-user-3

set ACL on /content

    allow jcr:read for training-user-3

end

create path /content/example3.com(sling:Folder)

create path /content/example3.com(sling:Folder mixin
mix:referenceable, mix:sharable)
```

---

The repoint statements are defined in an OSGi configuration (immutable). However, repoint statements often refer to or modify mutable content

---

## Service Users

A service user is a JCR user with no password set and a minimal set of privileges that are necessary to perform a specific task. Having no password set means that it will not be possible to log in with a service user, except as a known service.

## Service ID

A service may be comprised of multiple parts, so each part of the service may be further identified by a subservice name. A subservice name is optional. The examples of subservice name are the names for the subsystems in a Message Transfer System, such as accepting messages, queueing messages, and delivering messages.

The combination of the Service Name and Subservice Name defines the Service ID. The Service ID is finally mapped to a Resource Resolver and/or JCR user ID for authentication.

Therefore, the actual service identification (service ID) is defined as:

```
service-id = service-name [ ":" subservice-name ]
```

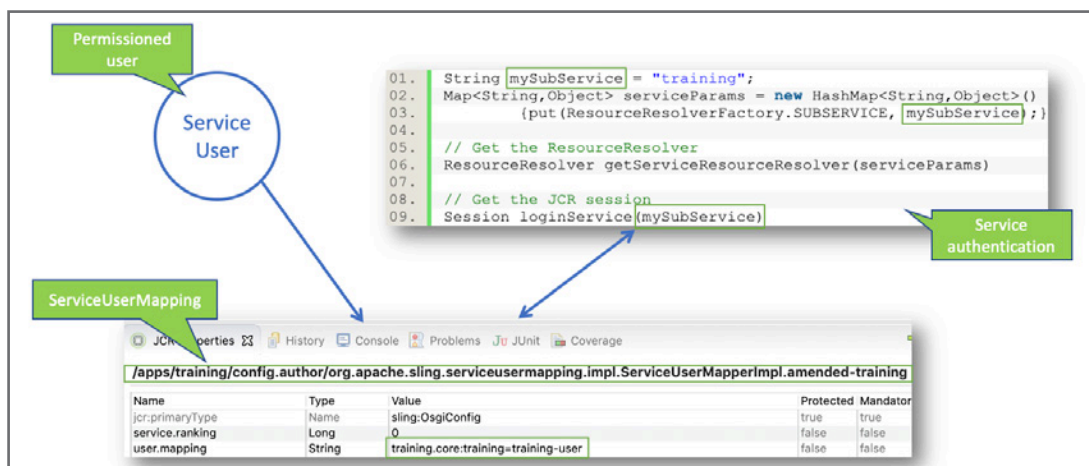
## ServiceUserMapper

The ServiceUserMapper service enables you to map the Service IDs comprised of the Service Names defined by the providing bundles and optional Subservice Name to ResourceResolver and/or JCR Repository user IDs. This mapping is configurable such that system administrators are in full control of assigning users to services.

## ResourceResolverFactory

The Sling ResourceResolverFactory provides a factory method to allow the service to authenticate to the repository, as shown:

```
ResourceResolver getServiceResourceResolver(Map<String, Object>
authenticationInfo) throws LoginException;
```



## Exercise 3: Use Repointit to create a service user

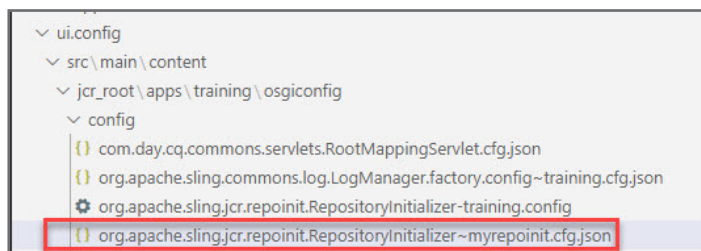
**Scenario:** A developer is implementing a background service that requires authentication and authorization to write to the repository. A service user is necessary to authenticate the service to the repository and to create a permission context within which the service will run. In addition, the service user needs to be granted the appropriate permissions.

This exercise includes the following tasks:

1. Create an OSGi configuration file in JSON format
2. Install and verify the configuration

Task 1: Create an OSGi configuration file in JSON format

1. In your IDE, navigate to **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig**.
2. Right-click the folder **config** and choose **New File**.
3. Enter the name as **org.apache.sling.jcr.repointit.RepositoryInitializer~myrepointit.cfg.json**, as shown:



4. To add the json properties to your file, open up the **Exercise\_Files-EC** folder for this course and navigate to **ui.config/src/main/content/jcr\_root/apps/training/osgiconfig/config**.
5. Open the file **org.apache.sling.jcr.repointit.RepositoryInitializer~myrepointit.cfg.json** using a text editor. Copy the contents and paste it to the file **org.apache.sling.jcr.repointit.RepositoryInitializer~myrepointit.cfg.json** in your IDE, as shown:

```
ui.config > src > main > content > jcr_root > apps > training > osgiconfig > config > {} org.apache.sling.jcr.repointit.RepositoryInitializer~myrepointit.cfg.json > ...
1 {
2   "scripts":[
3     "create service user training-user\n set ACL on /content\n allow jcr:all for training-user\n end\n"
4   ]
5 }
6 |
```

6. Select **File > Save** from the menu to save the file.

Now that the configuration file is created, we need to use **Maven** to build and install it into the local AEM Server

#### Task 2: Install via command line

1. Open a command prompt to the location of your Maven project.  
For example: **C:/adobe/< myproject >**

---

**Note:** If you are using an IDE with an integrated terminal such as Visual Studio Code, you can run your Maven commands there instead since it is already open to your project.

---

2. In the command prompt run the command:

```
$ mvn clean install -Padobe-public -PautoInstallSinglePackage
```

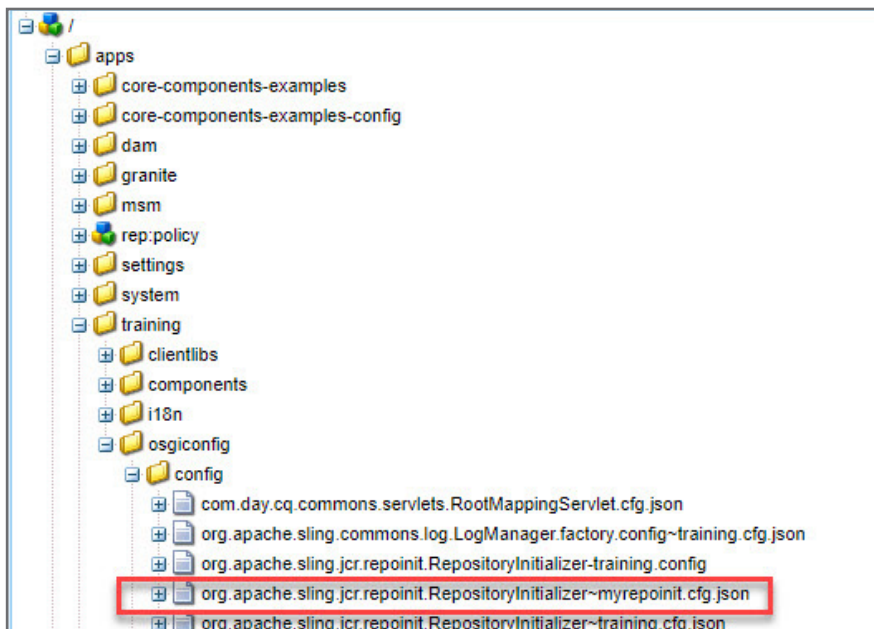
---

**TIP:** If you would like the install process to occur faster, you can go into the desired module and run a more specific profile. This allows for only that module to be installed rather than the entire project.

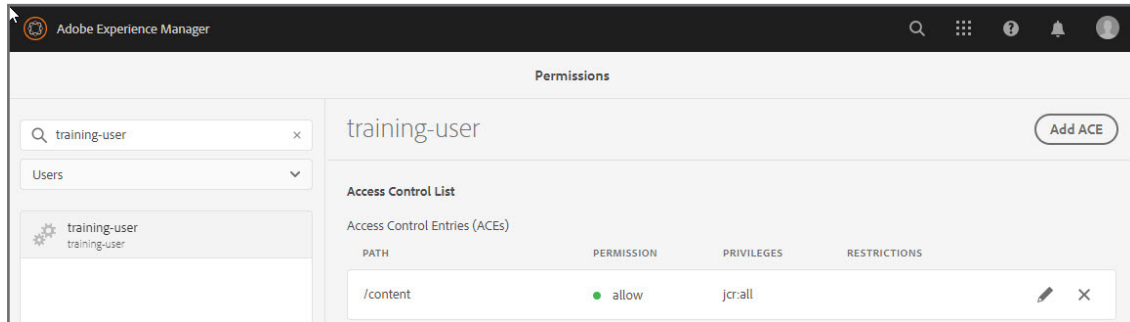
For example: Core - use `autoInstallBundle` , ui.apps - use `autoInstallPackage`, ui.content - use `autoInstallPackage`

---

3. Your project has now been successfully updated on your local AEM Server.
4. Open **CRXDE Lite** and navigate to **apps/training/config**.
5. Verify that the **org.apache.sling.jcr.repoinit.RepositoryInitializer~myrepoinit.cfg.json** OSGi configuration file is deployed, as shown:



6. Open the file and verify the contents.
7. On a browser tab, from the **Navigation** page, navigate to **Tools > Security > Permissions**. The **Permissions** window opens.
8. Select Users from the drop-down menu and search for the **training-user** to verify that the user is created and an ACE is created, as shown:



You now know how to initialize configuration information for an AEM instance. You have defined a set of repoint statements in an OSGi configuration, deployed the configuration to AEM and verified the results.

## Exercise 4: Create a service user Mapper

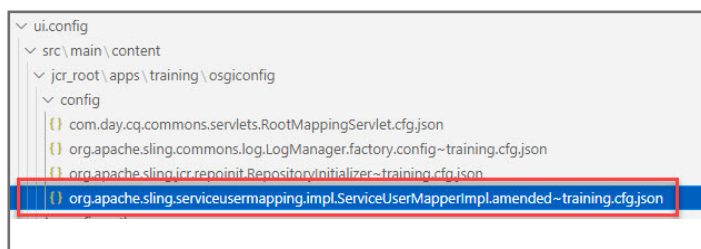
**Scenario:** In this exercise, you will create the mapping from our bundle to the user with an OSGi configuration file. The Bundles tab is the mechanism for installing the OSGi bundles required for AEM. The configuration we are going to setup is the `ServiceUserMapperImpl` factory.

This exercise includes the following tasks:

1. Create an OSGi configuration file in JSON format
2. Install and verify the configuration

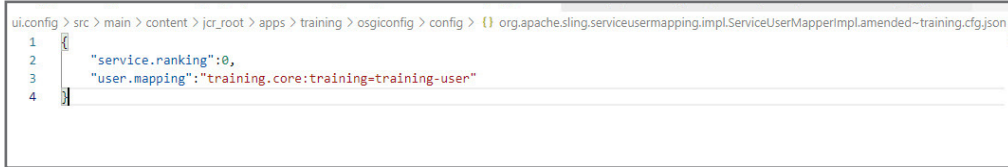
Task 1: Create an OSGi configuration file in JSON format

1. In your IDE, navigate to **ui.config > src/main/content/jcr\_root > apps > training > osgiconfig**.
2. Right-click the folder **config** and choose **New File**.
3. Enter the name as **org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended~training.cfg.json**, as shown:



4. To add the json properties to your file, open up the **Exercise\_Files-EC** folder for this course and navigate to **ui.config/src/main/content/jcr\_root/apps/training/osgiconfig/config**.

5. Open the file **org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended~training.cfg.json** using a text editor. Copy the contents and paste it to the file **org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended~training.cfg.json** in your IDE, as shown:



```
1 {  
2   "service.ranking":0,  
3   "user.mapping":"training.core:training=training-user"  
4 }
```

6. Select **File > Save** from the menu to save the file.

Now that the new json file and the configuration file are created, we need to use **Maven** to build and install them into the local AEM Server

## Task 2: Install and verify the configuration

1. Open a command prompt to the location of your Maven project.  
For example: **C:/adobe/< myproject >**

---

**Note:** If you are using an IDE with an integrated terminal such as Visual Studio Code, you can run your Maven commands there instead since it is already open to your project.

---

2. In the command prompt run the command:

```
$ mvn clean install -Padobe-public -PautoInstallSinglePackage
```

---

**TIP:** If you would like the install process to occur faster, you can go into the desired module and run a more specific profile. This allows for only that module to be installed rather than the entire project.

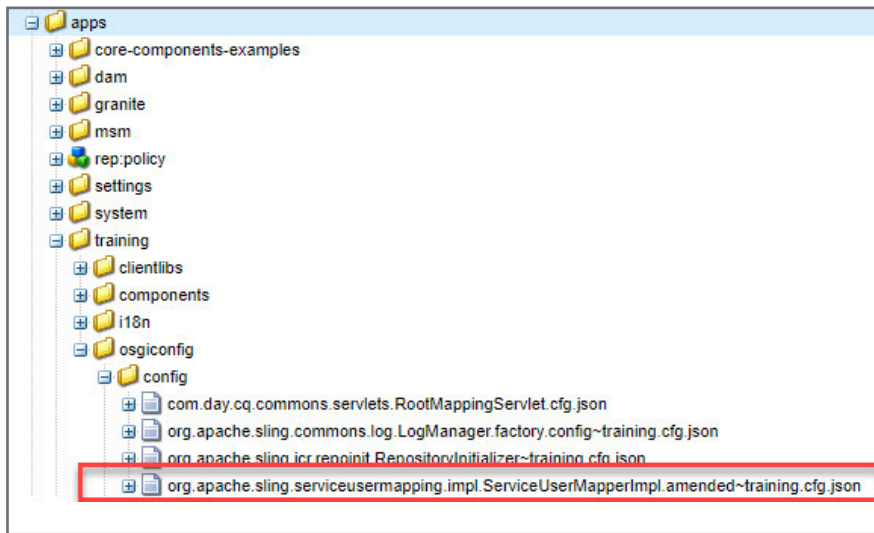
For example: Core - use `autoInstallBundle` , ui.apps - use `autoInstallPackage`, ui.content - use `autoInstallPackage`

---

3. Your project has now successfully installed into your local AEM Server.
4. Open **CRXDE Lite** and navigate to **apps/training/config**.



5. Verify that the `org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended~training.cfg.json` OSGi configuration file is deployed, as shown:



6. Open the file and verify the contents.
7. Navigate to the Web Console (<http://localhost:4502/system/console/configMgr>) and choose **Status** > **Sling Service User Mappings**. The user mappings are displayed.
8. Use the browser's search to look for and find **training** and observe the new user mapping, as shown:

```
Date: March 22, 2021 at 4:31:03 PM PDT

*** Mappings by user (1 users): (format: service name / sub service name / user)
training-user
training.core / training / training-user
*** Mappings by principals (129 principals): (format: service name / sub service name / principal names)
account-manager
com.adobe.cq.cq-account / account-management-service / [account-manager, content-reader-service, gr
activity-service
com.adobe.granite.activitystreams / activity-service / [activity-service]
activitypurgesrv
```

You have successfully created a system user mapper using json configuration.

## References

---

For further information on OSGi Configurations and Run Modes, refer:

- [Configuring Run Modes](#)
- [OSGi configurations](#)
- [Configuring OSGi for AEM as a Cloud Service](#)
- [repointit](#)
- [Service Authentication](#)