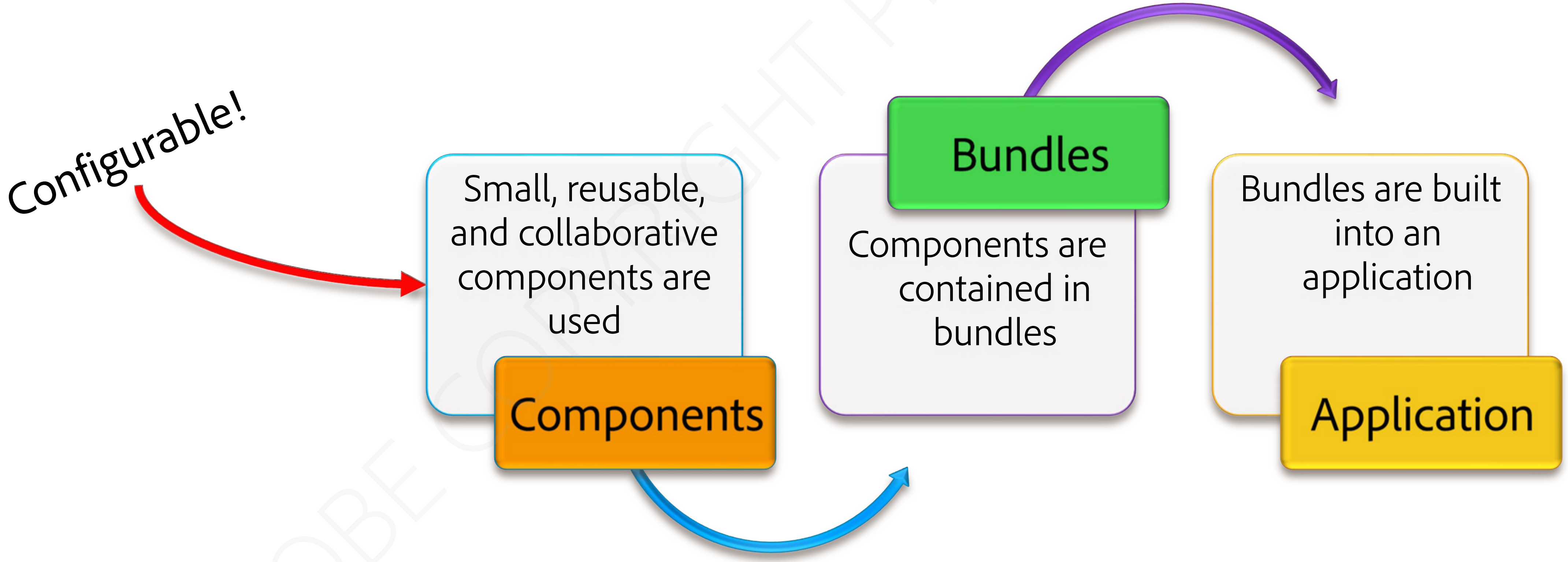
Configure the AEM Platform

Agenda:

- OSGi configurations overview
- JSON config files
- Runmodes
- Service Users
- Repository Initializer

OSGi

Fundamental element in the technology stack of Adobe Experience Manager (AEM)



Configurable!
Small, reusable, and collaborative components are used
Components
Bundles
Components are contained in bundles
Bundles are built into an application
Application

Creating OSGi configurations

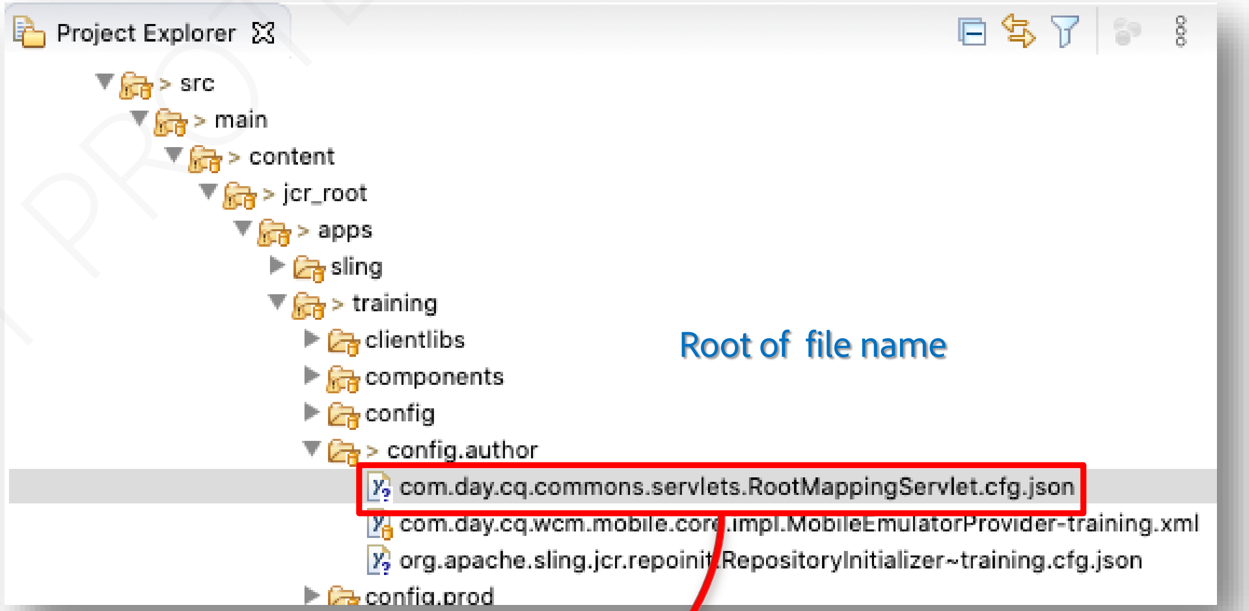
Stored in a Maven Project

Configurations are considered code (/apps)

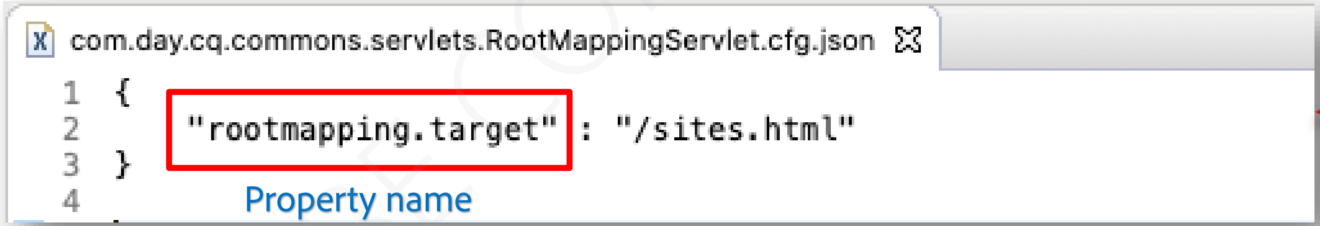
JSON file name identifies the configuration

- <PID>.cfg.json

Key/value pairs are configurations



```
▼ > src
  ▼ > main
    ▼ > content
      ▼ > jcr_root
        ▼ > apps
          ▶ sling
          ▼ > training
            ▶ clientlibs
            ▶ components
            ▶ config
            ▼ > config.author
                com.day.cq.commons.servlets.RootMappingServlet.cfg.json
                com.day.cq.wcm.mobile.core.impl.MobileEmulatorProvider-training.xml
                org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json
            ▶ config.prod
```

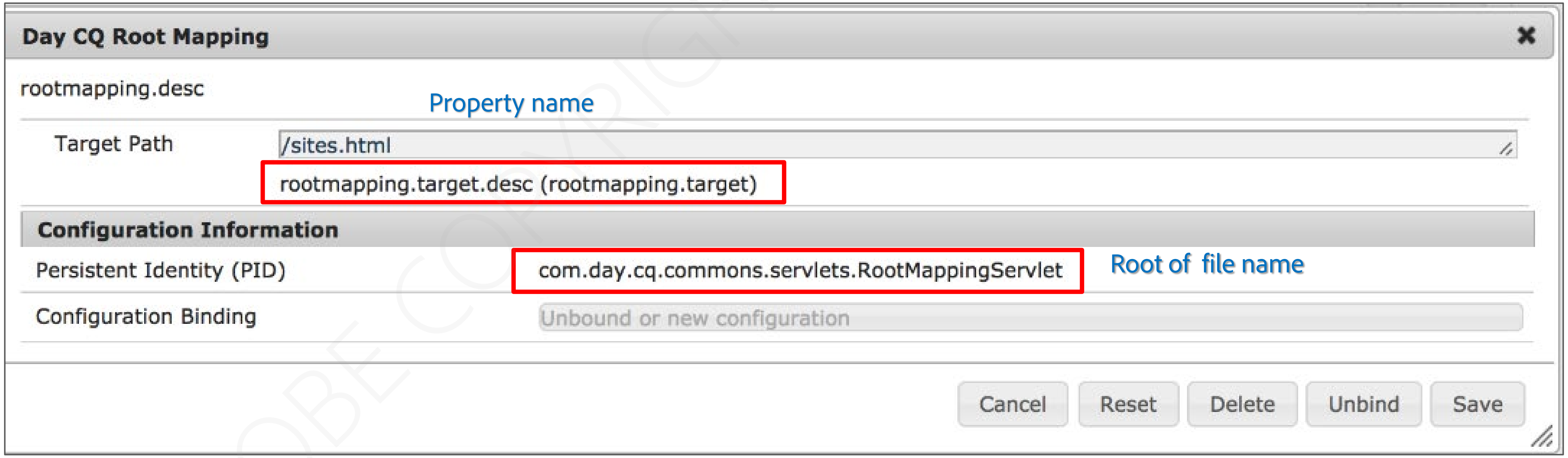


```
{
    "rootmapping.target" : "/sites.html"
}
```

Determining property name and file name to use

From the Web Console > OSGi > Configurations

- Find the desired configuration
- Obtain the Property name and Node name-PID to use in your configuration node.



rootmapping.desc

Target Path: /sites.html

rootmapping.target.desc (rootmapping.target)

Persistent Identity (PID): com.day.cq.commons.servlets.RootMappingServlet

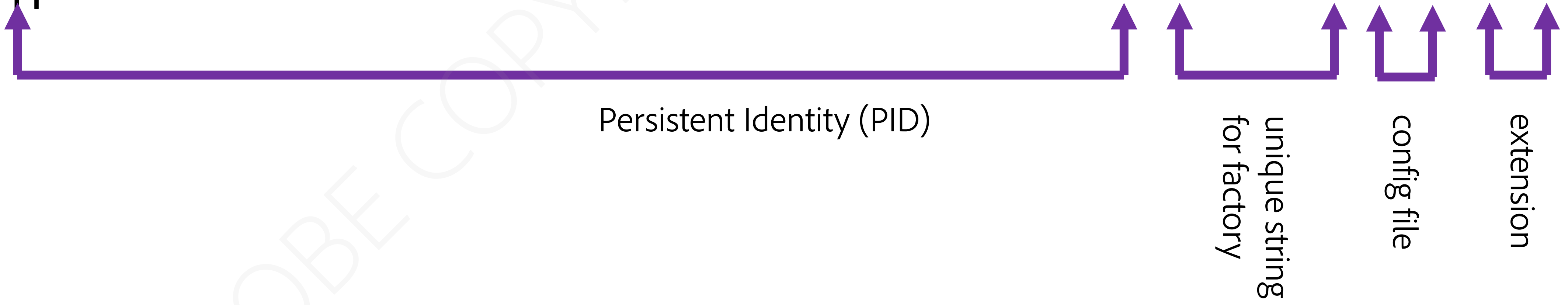
Configuration Binding: Unbound or new configuration

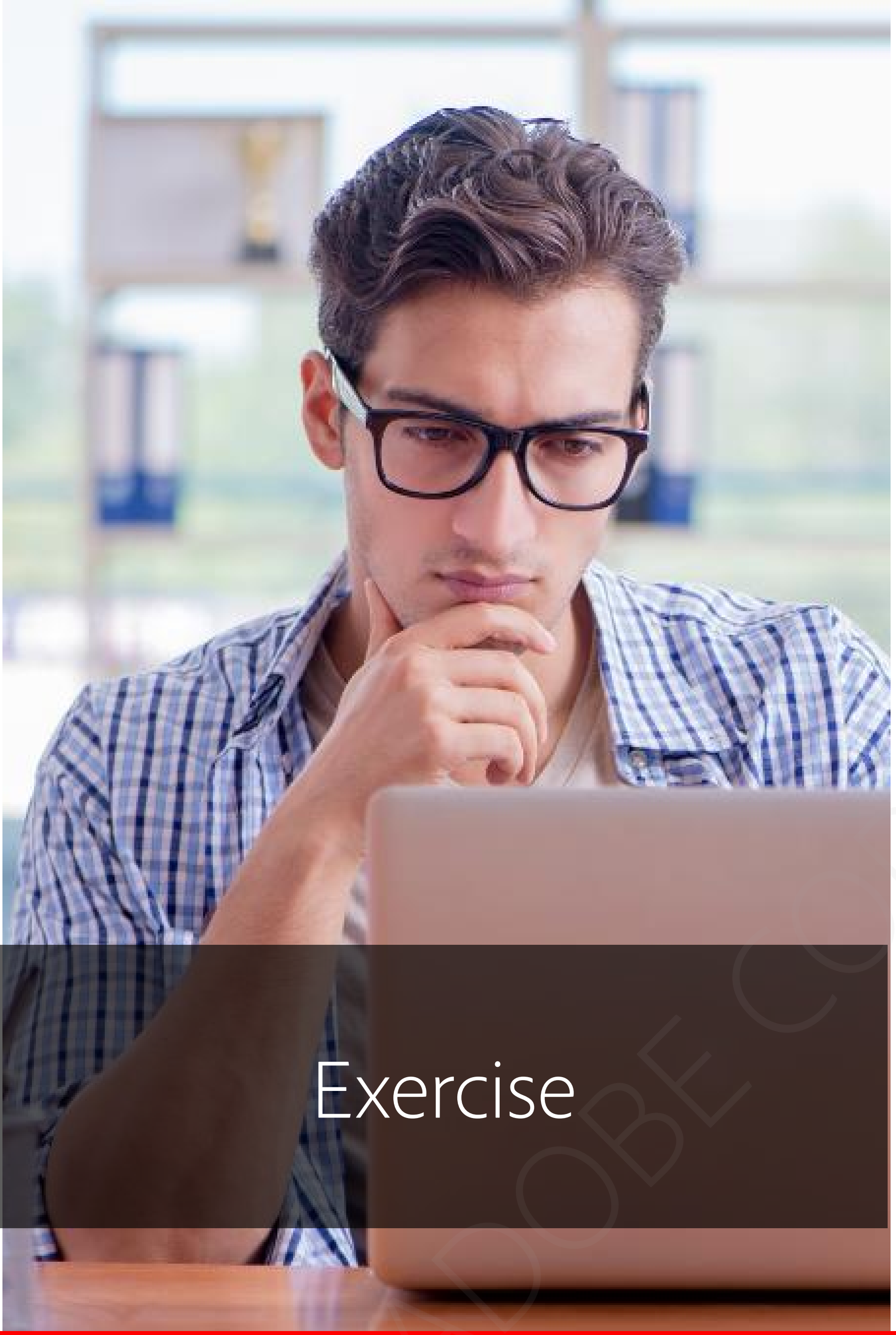
OSGi configuration files for factory services

Factory services allow for multiple configurations for multiple instances of a service

Defined by a unique string after the PID

com.adobe.granite.workflow.purge.Scheduler~training.cfg.json





Exercise 1: Create an OSGi Configuration

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.

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

Run modes overview

- Identifies an AEM service
  - To target specific environment needs
  - Based on service tier and environment type
- Tier run modes
  - Author and publish run modes are mutually exclusive
  - Tier run modes are set at installation time and cannot be changed
- Environment run modes
  - Dev, stage, prod

Available Run modes

/apps/*/config.[author | publish].[dev | stage | prod]

Examples:

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

Multiple configurations for the same PID: Example

An instance was started with:

- -r author,dev

The following configurations are defined:

- /apps/*/config.author

  com.day.cq.wcm.core.impl.VersionManagerImpl

- /apps/*/config.author.dev

  com.day.cq.wcm.core.impl.VersionManagerImpl

The config in /apps/*/config.author.dev will be applied.

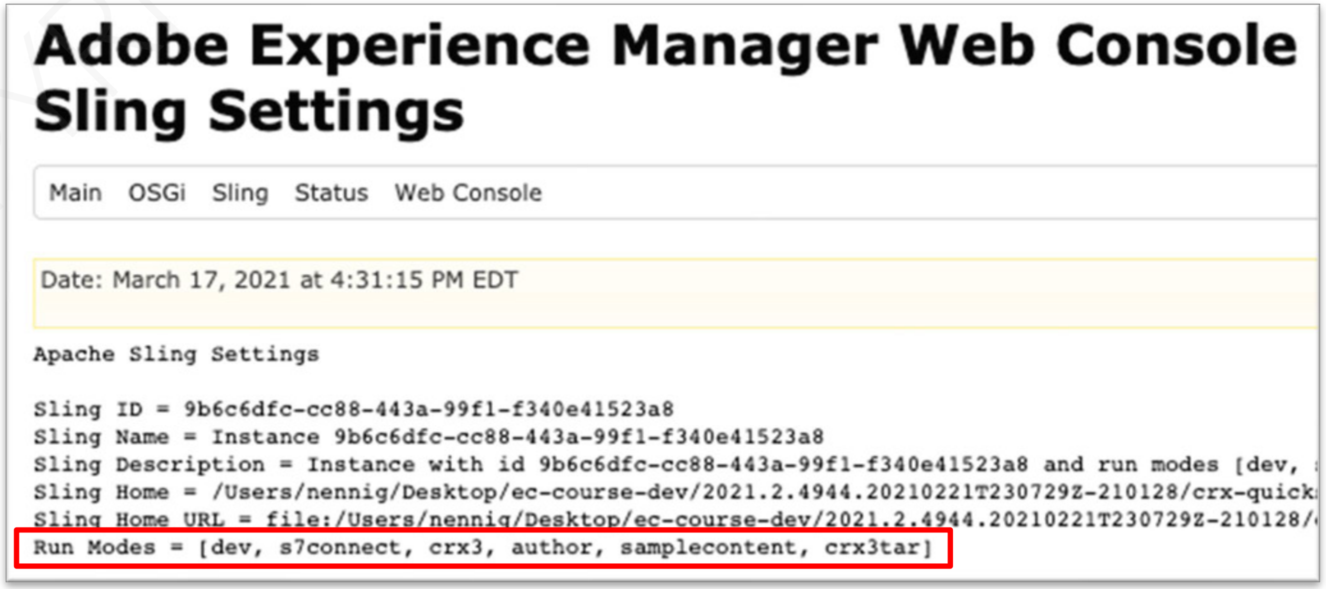
This is because the config with the highest number of matching run modes is applied.

Using and Viewing runmodes locally

Start AEM with a runmode:

- `$ java –jar aem-quickstart.jar –r dev,author`

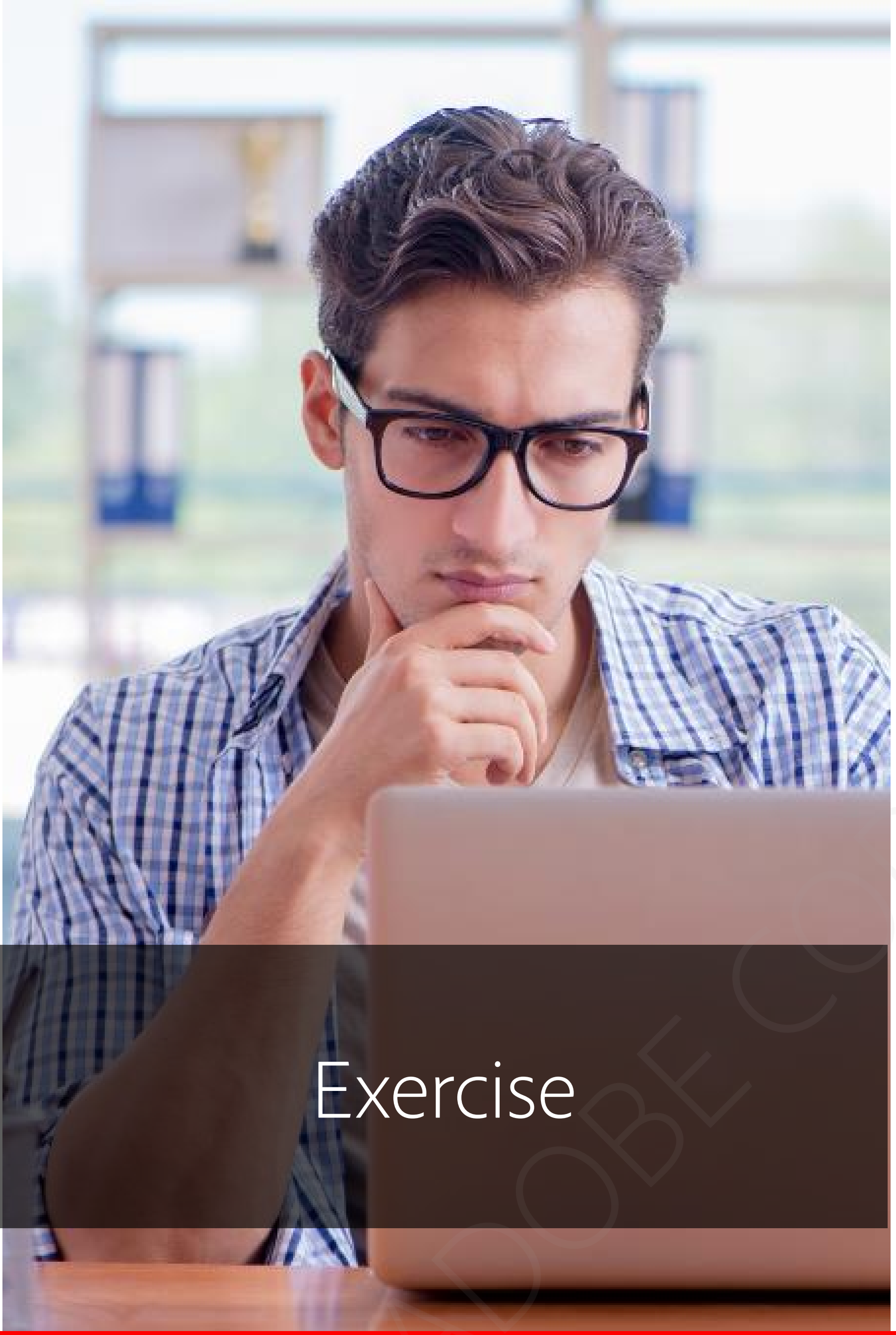
Tools > Web Console > Status > Sling Settings



Adobe Experience Manager Web Console
Sling Settings

Date: March 17, 2021 at 4:31:15 PM EDT

```
Apache Sling Settings

Sling ID = 9b6c6dfc-cc88-443a-99f1-f340e41523a8
Sling Name = Instance 9b6c6dfc-cc88-443a-99f1-f340e41523a8
Sling Description = Instance with id 9b6c6dfc-cc88-443a-99f1-f340e41523a8 and run modes [dev,
Sling Home = /Users/nennig/Desktop/ec-course-dev/2021.2.4944.20210221T230729Z-210128/crx-quick
Sling Home URL = file:/Users/nennig/Desktop/ec-course-dev/2021.2.4944.20210221T230729Z-210128/
Run Modes = [dev, s7connect, crx3, author, samplecontent, crx3tar]
```



Exercise 2: Create a configuration for an Environment

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

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

Repository initialization

Some elements of an application need to be loaded first to install the application successfully

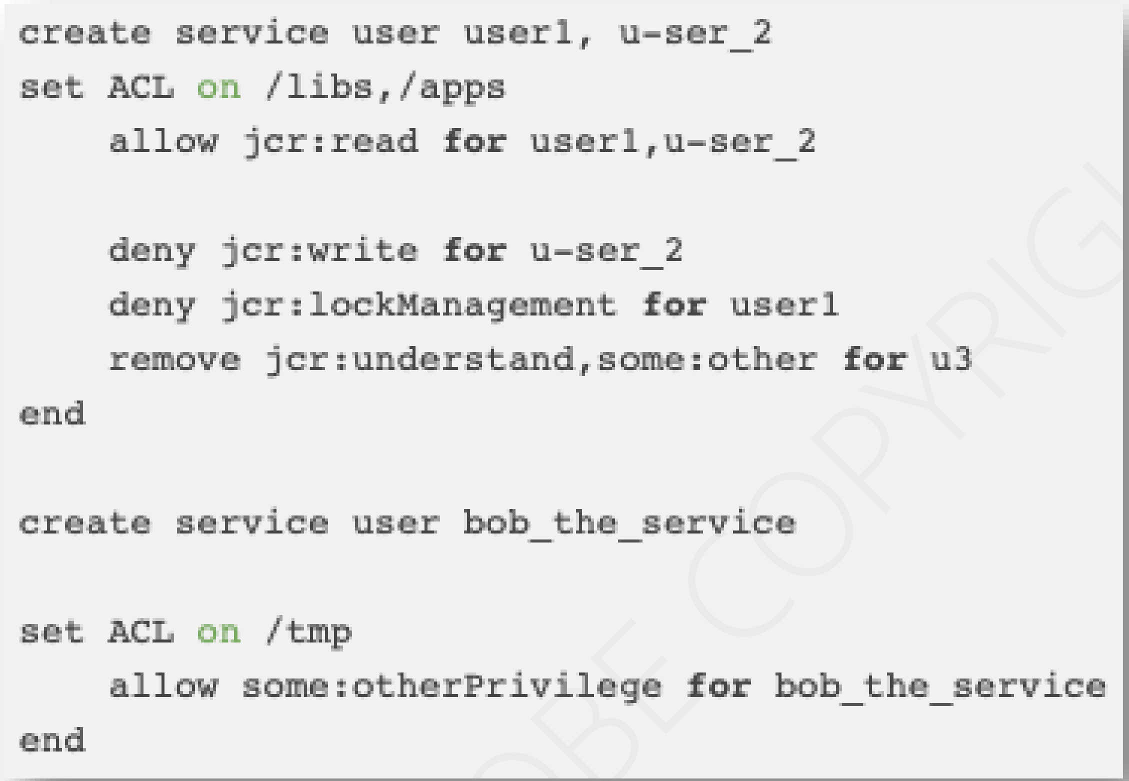
- Create/delete/disable service users
- Create/delete users and groups
- Add access control lists (ACLs)
- Add path (for example, for root folder structures)
- Add CNDs (nodetype definitions)
- Add repository namespaces

Sling RepositoryInitializer runs the code before the SlingRepository service is registered

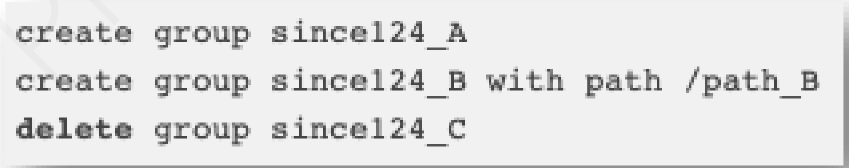
- Enables initialization
- Controlled by use of repoinit script
- Available for AEM 6.5.4+ and Cloud Service

Repository Initialization Language (repoinit)

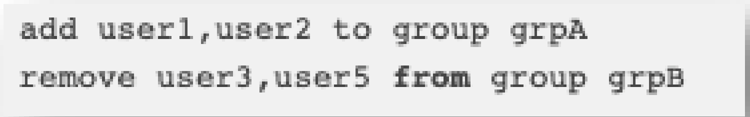
Example language



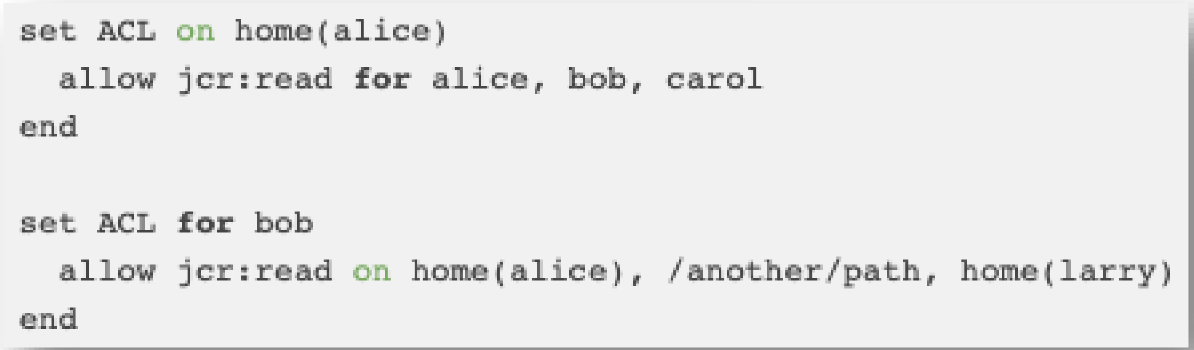
```
create service user user1, u-ser_2
set ACL on /libs,/apps
    allow jcr:read for user1,u-ser_2

    deny jcr:write for u-ser_2
    deny jcr:lockManagement for user1
    remove jcr:understand,some:other for u3
end

create service user bob_the_service

set ACL on /tmp
    allow some:otherPrivilege for bob_the_service
end
```



```
create group since124_A
create group since124_B with path /path_B
delete group since124_C
```

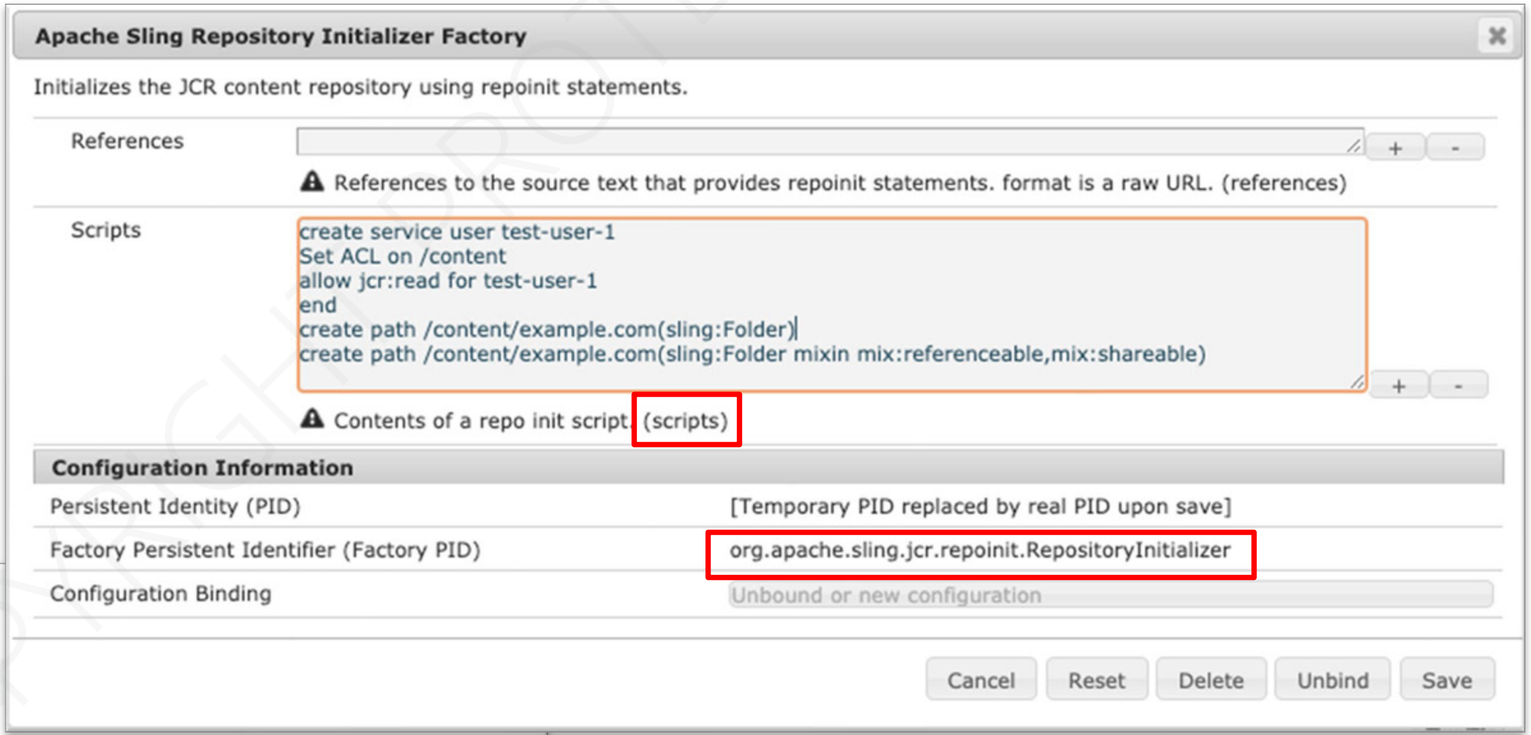


```
add user1,user2 to group grpA
remove user3,user5 from group grpB
```



```
set ACL on home(alice)
  allow jcr:read for alice, bob, carol
end

set ACL for bob
  allow jcr:read on home(alice), /another/path, home(larry)
end
```

RepositoryInitializer Configuration Factory



Apache Sling Repository Initializer Factory

Initializes the JCR content repository using repoinit statements.

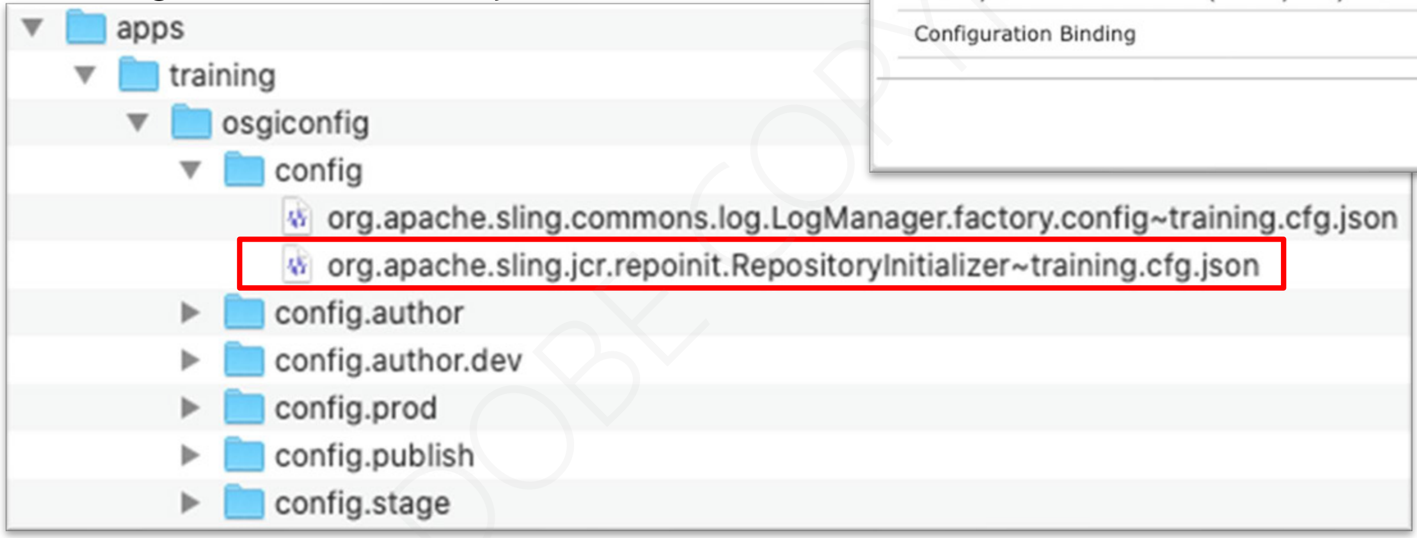
References	
	References to the source text that provides repoinit statements. format is a raw URL. (references)
Scripts	see below
	Contents of a repo init script. (scripts)

```
create service user test-user-1
Set ACL on /content
allow jcr:read for test-user-1
end
create path /content/example.com(sling:Folder)
create path /content/example.com(sling:Folder mixin mix:referenceable,mix:shareable)
```

Configuration Information

Persistent Identity (PID)	[Temporary PID replaced by real PID upon save]
Factory Persistent Identifier (Factory PID)	org.apache.sling.jcr.repoinit.RepositoryInitializer
Configuration Binding	Unbound or new configuration

ui.config/src/main/content/jcr_content/



```
apps
  training
    osgiconfig
      config
        org.apache.sling.commons.log.LogManager.factory.config~training.cfg.json
        org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json
      config.author
      config.author.dev
      config.prod
      config.publish
      config.stage
```

Repoinit example | Service Users

Special users attached to services for JCR related tasks

Need to be created before OSGi bundles are started

Useful for Security:

- Forces developers to create service users and map them OSGi components
- To prevent excessive use of administrative JCR Sessions and ResourceResolvers

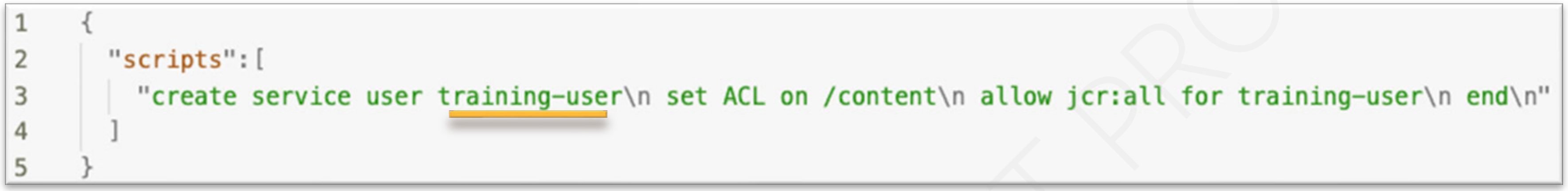
Useful for Flexibility

- Access ResourceResolvers or JCR Sessions without requiring hard-coding or configuring passwords
- Service users can be reused for multiple components with similar access requirements

Users are bound to an OSGi bundle through a ServiceUserMapper configuration

Using Service Users

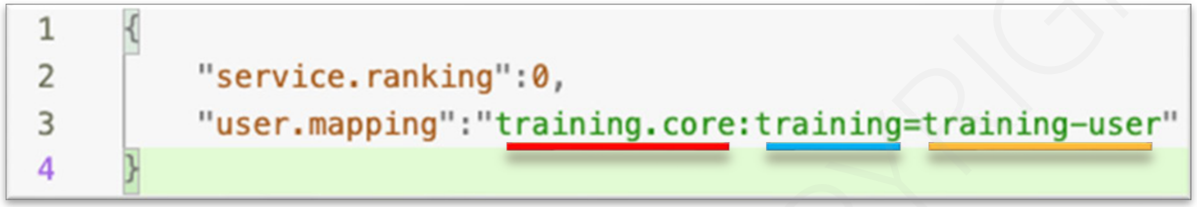
ui.config/apps/…/config/org.apache.sling.jcr.repoinit.RepositoryInitializer~training.cfg.json



```
{
  "scripts":[
    "create service user training-user\n set ACL on /content\n allow jcr:all for training-user\n end\n"
  ]
}
```

Repoinit script to create the service user and give needed permissions

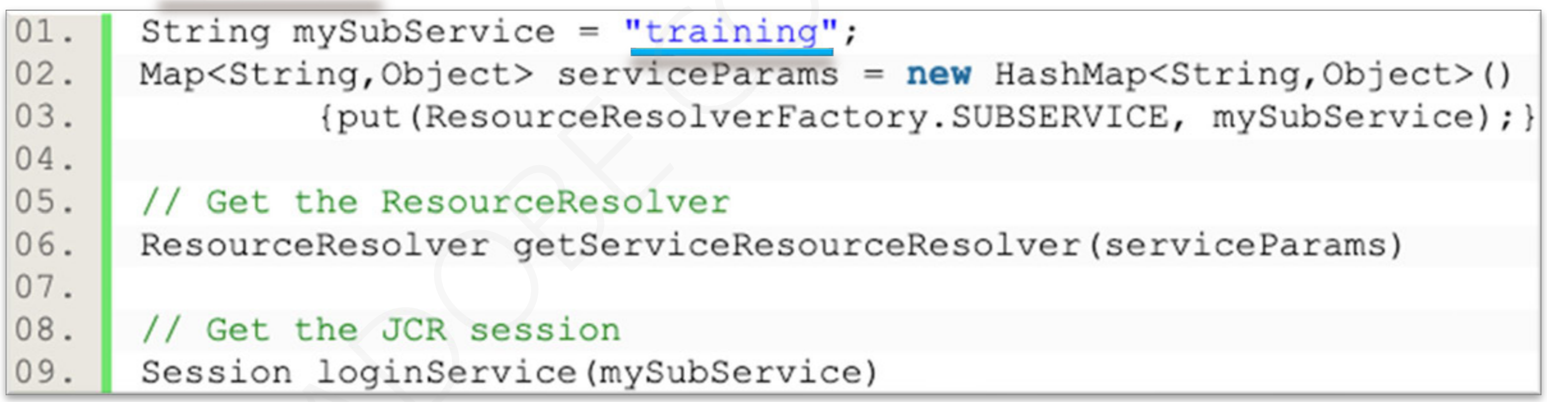
ui.config/apps/…/config/org.apache.sling.serviceusermapping.impl.ServiceUserMapperImpl.amended~training.cfg.json



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

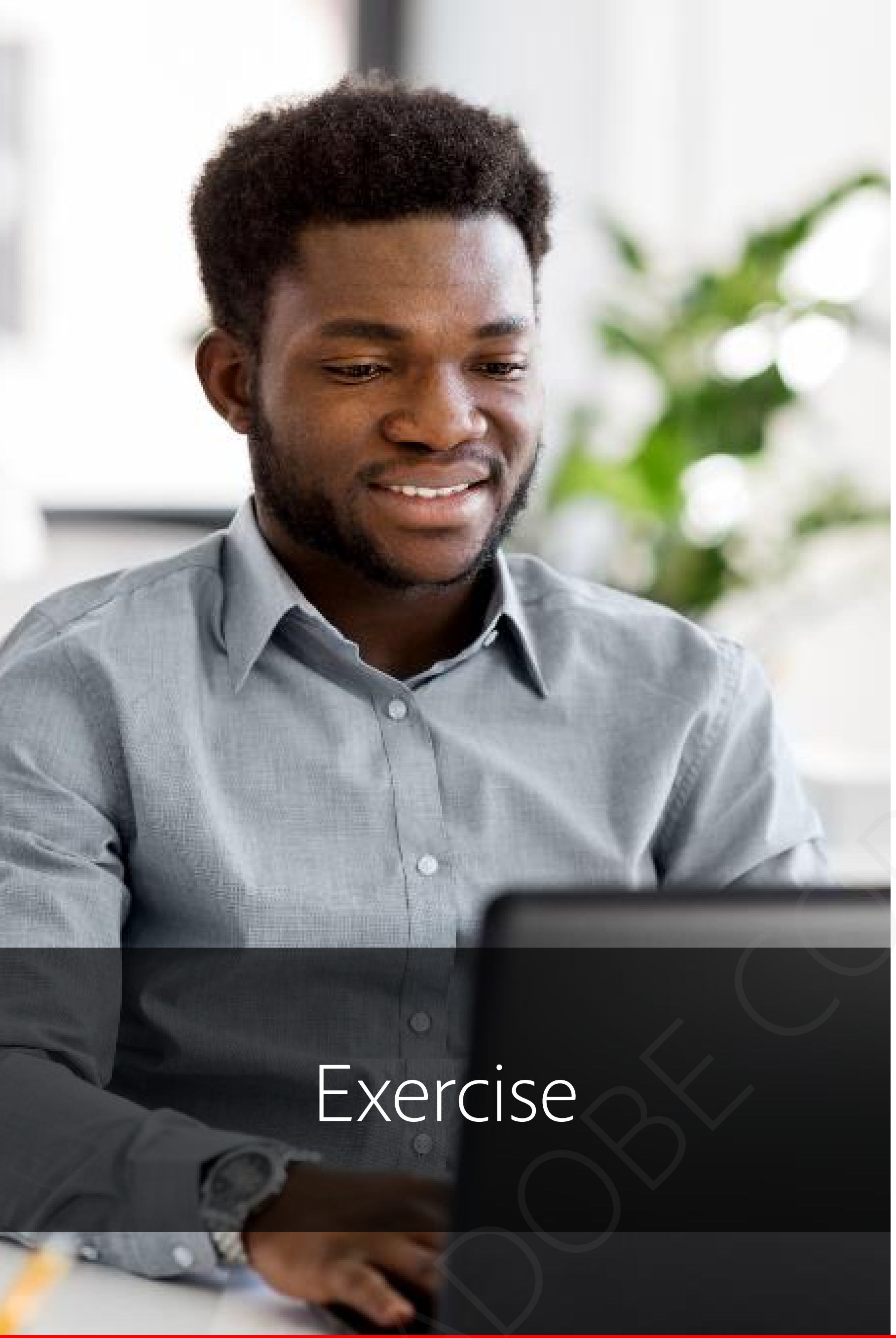
Mapper config to bind the service user to the bundle

core/…/training/core/MyService.java



```
String mySubService = "training";
Map<String,Object> serviceParams = new HashMap<String,Object>()
        {put(ResourceResolverFactory.SUBSERVICE, mySubService);}

// Get the ResourceResolver
ResourceResolver getServiceResourceResolver(serviceParams)

// Get the JCR session
Session loginService(mySubService)
```

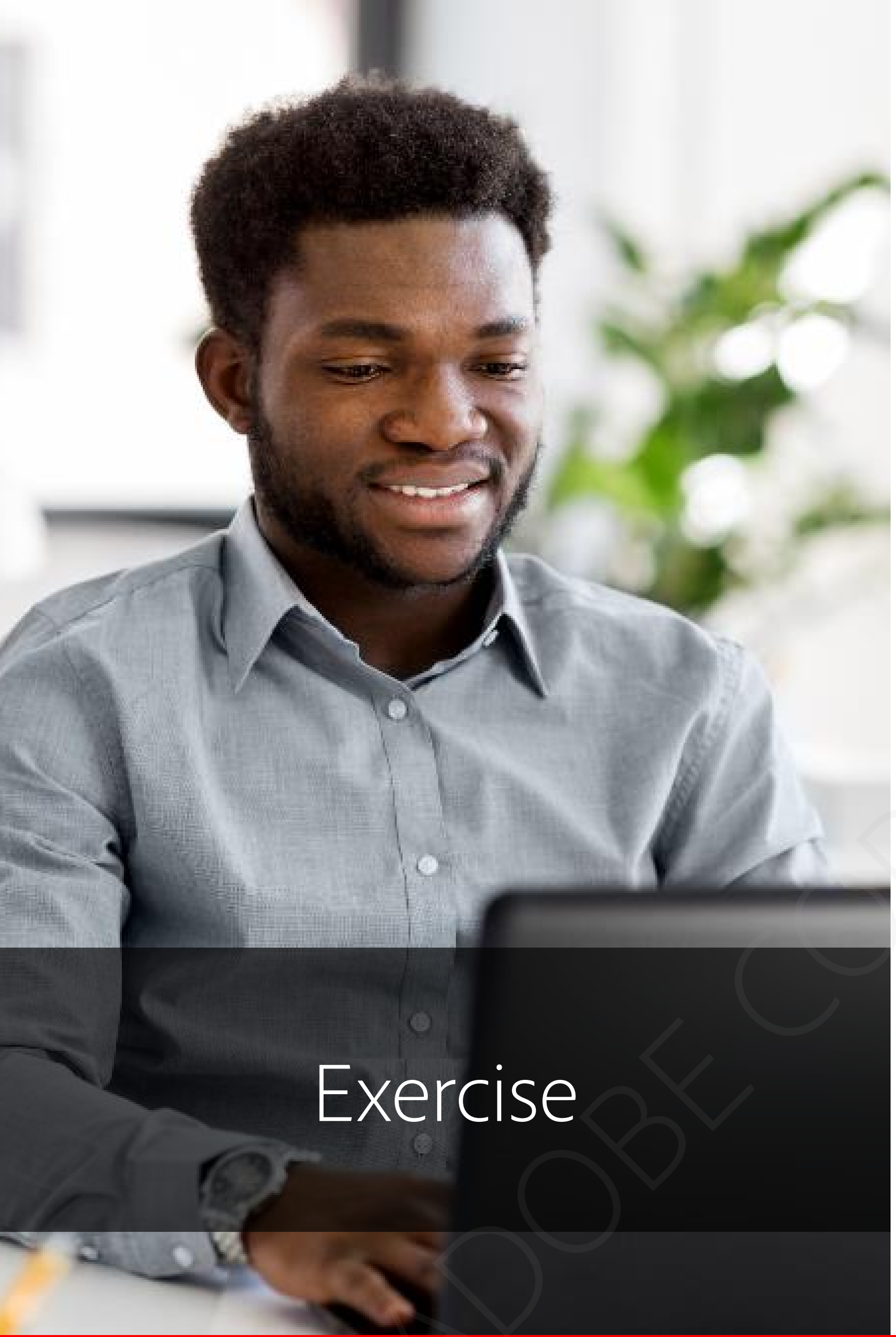
OSGi Service using the service user



Exercise 3: Use repoinit statements to define service user ‘training-user’

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.

- Task 1: Create an OSGi configuration file in JSON format
- Task 2: Install and verify the configuration



Exercise 4: Create a system user mapper

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.

- Task 1: Create an OSGi configuration file in JSON format
- Task 2: Install and verify the configuration

Key Takeaways

- Run modes enable you to define specific AEM behaviors for different types of AEM instances
- OSGi Configuration definitions are stored in config folders matching the desired run mode
- All configuration definitions are stored in the repository and activated by setting the appropriate run mode.
- You can check the run modes of any local instance using the link http://localhost:4502/system/console and navigating to **Status > Sling Settings**.

Adobe