

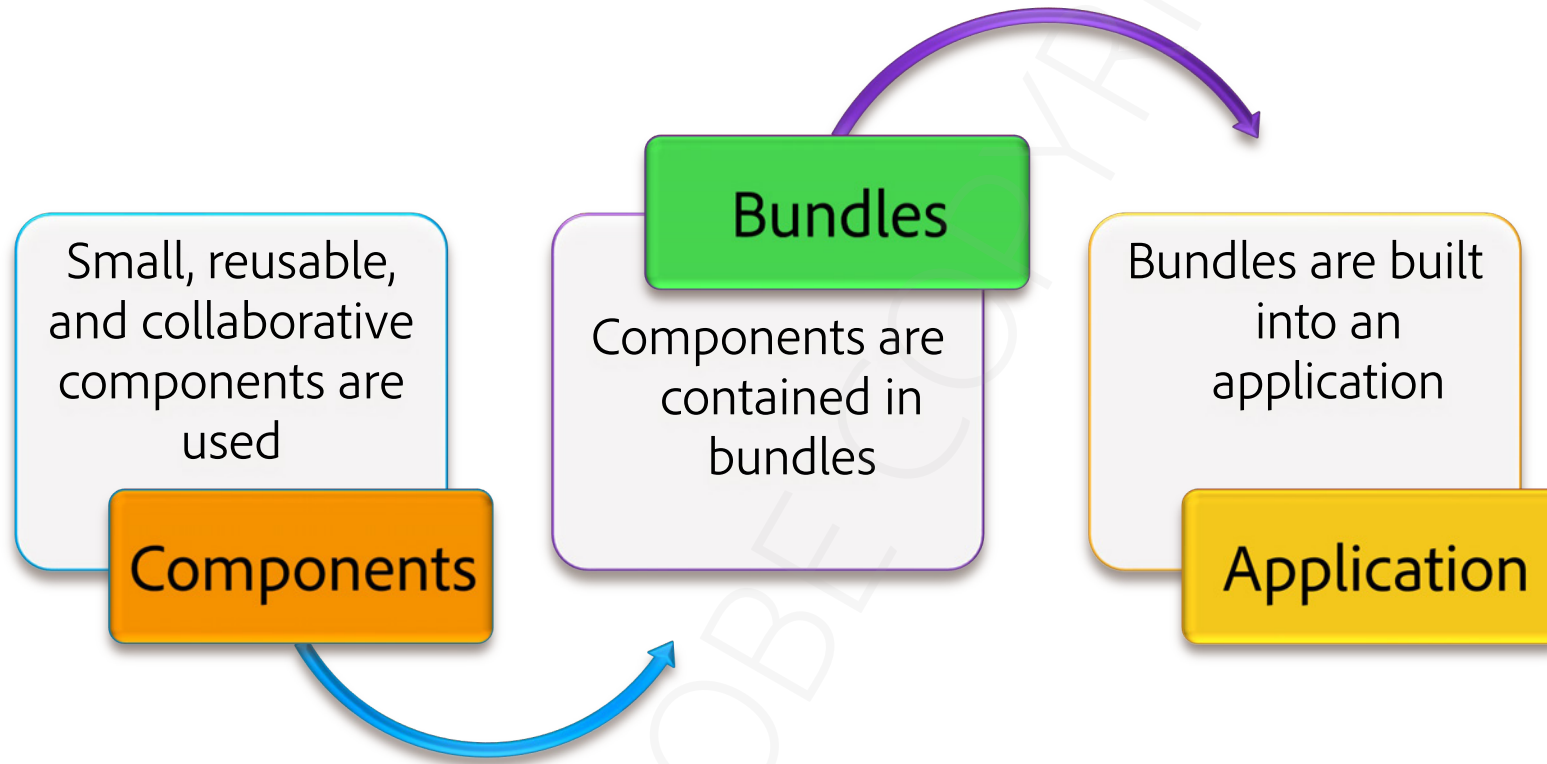
# Configure AEM Cloud Service

## Agenda:

- OSGi configurations
- Most commonly used OSGi configurations
- Creating an OSGi configuration
- Role of System Users
- Repository Initializer Service and its language

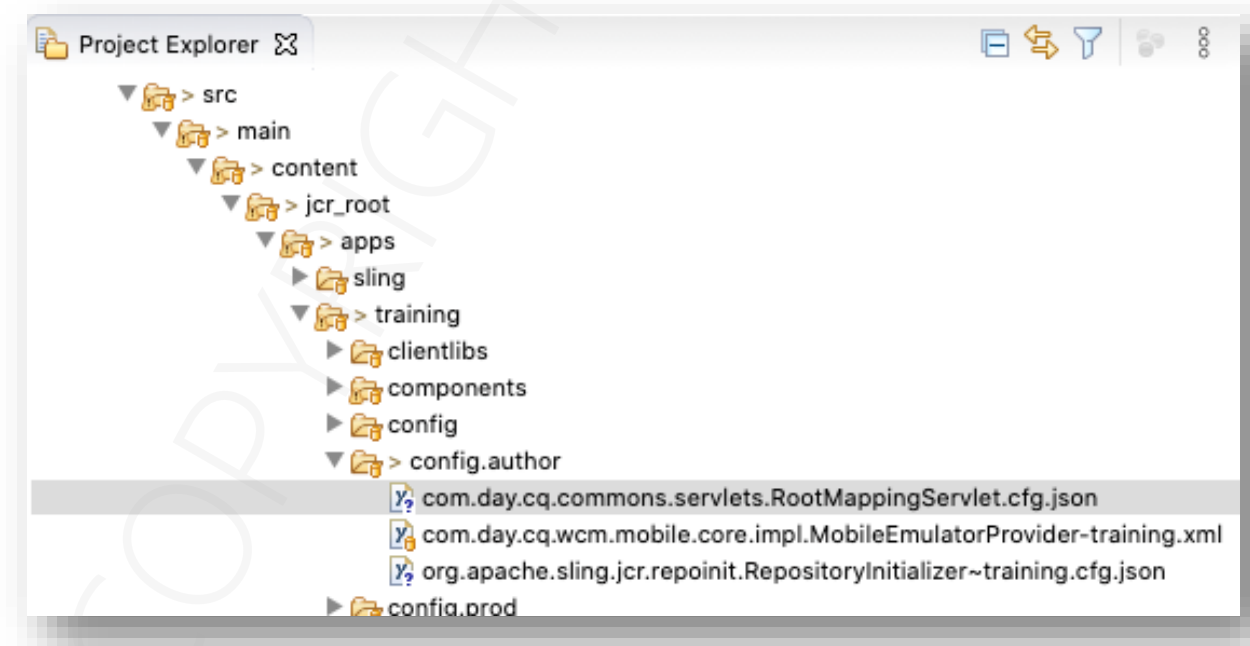


Fundamental element in the technology stack of Adobe Experience Manager (AEM)



# Creating OSGi configurations

- Configuration files are created in AEM Maven Project
- Configurations are considered code
  - Enables configurations to be pushed into Cloud Manager
  - All AEM services will contain these configurations
  - Like code, OSGi configs can only be updated through a CI/CD pipeline



# 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.

**Day CQ Root Mapping**

rootmapping.desc

Property name

Target Path: /sites.html

rootmapping.target.desc (rootmapping.target)

**Configuration Information**

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

Root of file name

Configuration Binding: Unbound or new configuration

Cancel Reset Delete Unbind Save

## OSGi configuration file contents

/apps/<my-app>/config/com.day.cq.commons.servlets.RootMappingServlet.cfg.json

```
1  {  
2  
3    "rootmapping.target" : "/sites.html"  
4  
5  }
```

# OSGi configuration values

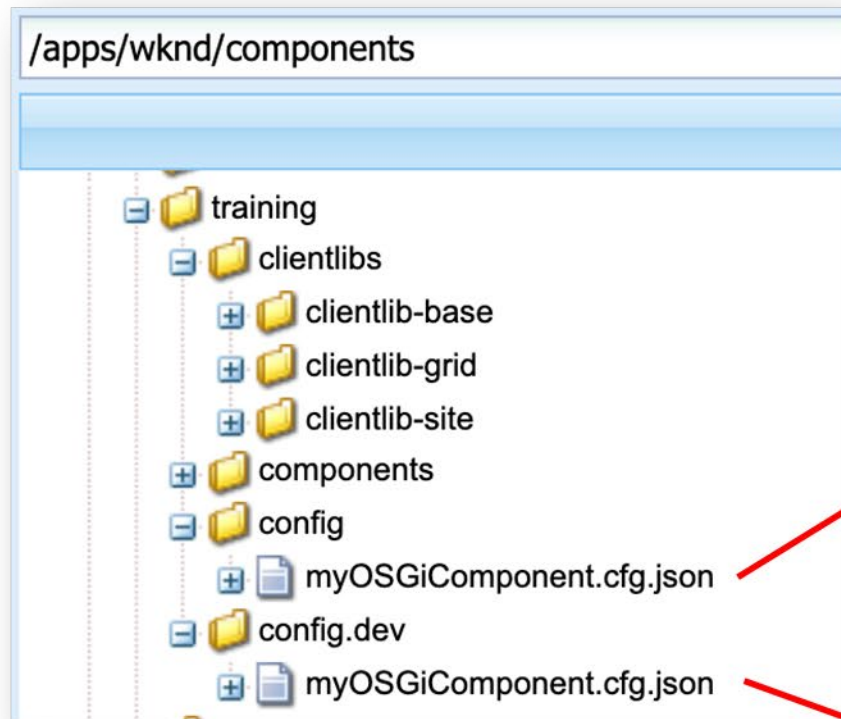
**Inline values**

```
{  
  "scheduledpurge.workflowStatus" : "COMPLETED",  
  "scheduledpurge.modelIds" : ["publish_example",  
"request_for_deactivation"],  
  "api-key" : "$[secret:server-api-key]",  
  "url" : "$[env:server-url]"  
}
```

**Secret values**

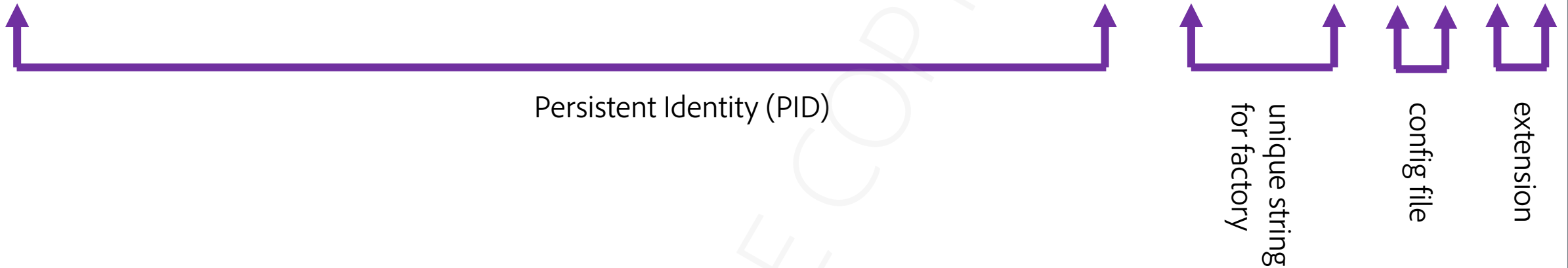
**Environment-specific values**

# Environment-specific configurations



# OSGi configuration files for factory services

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





# Workflow purge maintenance task

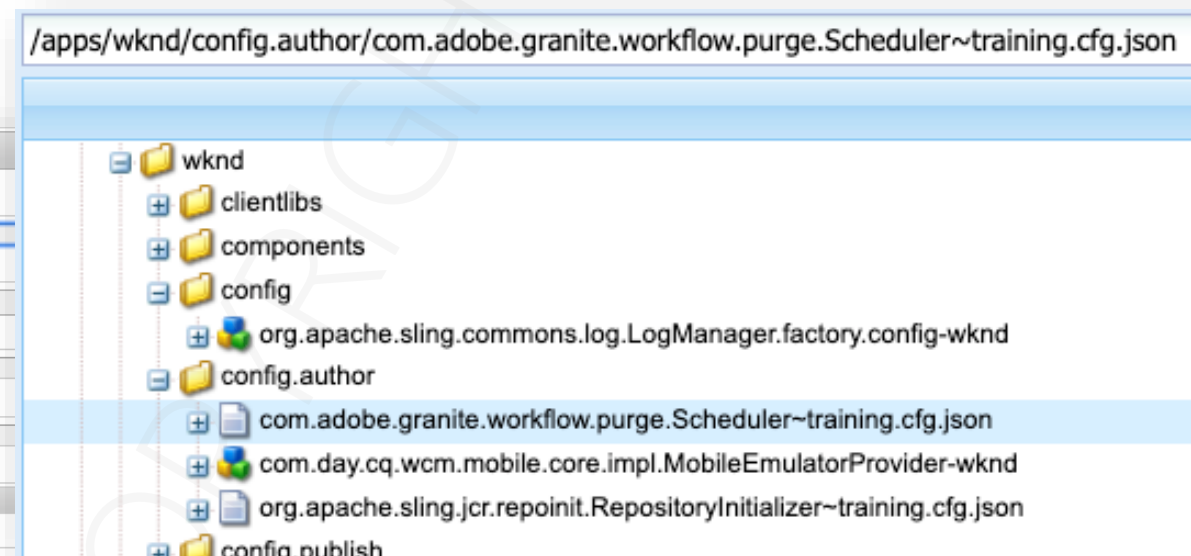
**Adobe Granite Workflow Purge Configuration**

Add configurations for different purges to execute during maintenance

Job Name	Purge Completed workflow instances for the Publish Example <small>Enter a descriptive name for this scheduled purge (scheduledpurge.name)</small>
Workflow Status	COMPLETED <small>Choose which status of WorkFlows to purge (scheduledpurge.workflowStatus)</small>
Models To Purge	publish_example <small>Enter model id for models to purge. Leave blank for all models (scheduledpurge.modelIds)</small>
Workflow Age	30 <small>Enter how old workflows are before they are purged (scheduledpurge.daysold)</small>

**Configuration Information**

Persistent Identity (PID)	com.adobe.granite.workflow.purge.Scheduler~training
Factory Persistent Identifier (Factory PID)	com.adobe.granite.workflow.purge.Scheduler
Configuration Binding	Unbound or new configuration



```
1 {
2   "scheduledpurge.workflowStatus": "COMPLETED",
3   "scheduledpurge.name": "Purge Completed workflow instances for the Publish Example",
4   "scheduledpurge.daysold:Integer": 30,
5   "scheduledpurge.modelIds:String[]": [
6     "/var/workflow/models/publish_example"
7   ]
8 }
9
```

# Common OSGi configurations

- Day CQ WCM Undo
- Day CQ Root Mapping
- Maintenance Tasks
  - Workflow Purge
  - Ad-hoc Task Purge
  - Project Purge

# Generating configurations

OSGi > OSGi installer Configuration Printer

serialization  
format

## Adobe Granite Workflow Purge Configuration

Add configurations for different purges to execute during maintenance

Job Name	Purge Completed workflow instances for the Publish Example Enter a descriptive name for this scheduled purge (scheduledpurge.name)
Workflow Status	COMPLETED Choose which status of WorkFlows to purge (scheduledpurge.workflowStatus)
Models To Purge	publish_example Enter model id for models to purge. Leave blank for all models (scheduledpurge.modelIds)
Workflow Age	30 Enter how old workflows are before they are purged (scheduledpurge.daysold)

## Configuration Information

Persistent Identity (PID)	com.adobe.granite.workflow.purge.Scheduler.2d5e0718-df58-4ee4-8074-5afc02f250aa
Factory Persistent Identifier (Factory PID)	com.adobe.granite.workflow.purge.Scheduler
Configuration Binding	Unbound or new configuration

## Adobe Experience Manager Web Console OSGi Installer Configuration Printer

Main OSGi Sling Status Web Console

Log out

### OSGi Installer Configuration Printer

To emit the configuration properties just enter the configuration PID, select a [serialization format](#) and click 'Print'

PID com.adobe.granite.workflow.purge.Scheduler.2d5e0718-df58-4ee4-8074-5afc02f250aa

Serialization Format OSGi Configurator JSON Print

```
{
  "scheduledpurge.workflowStatus": "COMPLETED",
  "scheduledpurge.name": "Purge Completed workflow instances for the Publish Example",
  "scheduledpurge.daysold": 30,
  "scheduledpurge.modelIds": [
    "publish_example"
  ]
}
```

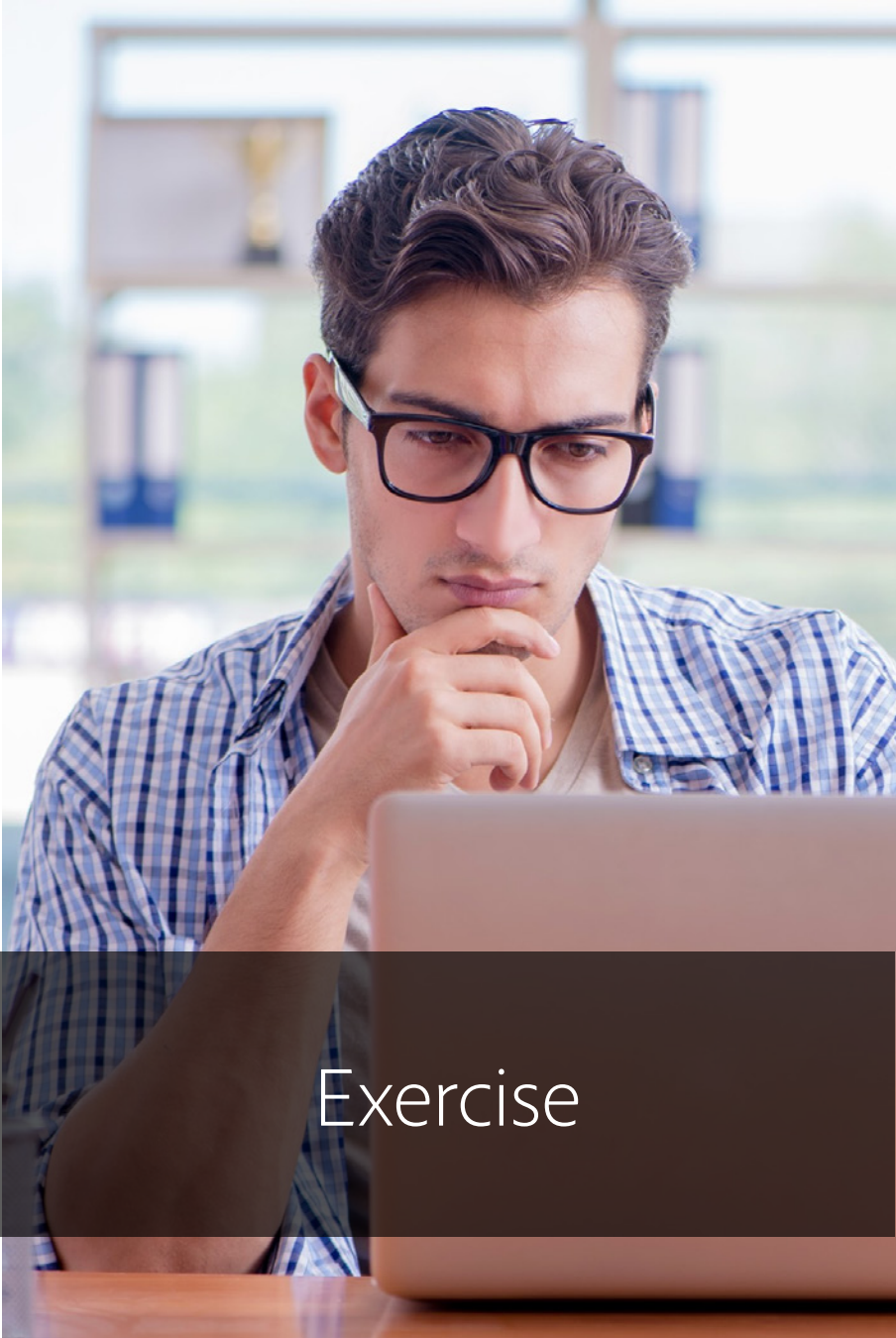
Serialized Configuration  
Properties

Copy to Clipboard



## Exercise

# Exercise 1: Use the OSGi Installer Configuration Printer



## Exercise

## Exercise 2: Configure AEM

# Permission service | Sling service

- Piece or collection of functionality
  - Examples of services might include
    - ❑ Sling queuing system
    - ❑ Tenant Administration
    - ❑ Event Handlers
    - ❑ Custom Workflow processes
- Identified by a unique service name
- Implemented as a component in an OSGi bundle
- Named by the bundles providing them

# Permission service | Service permission context

Adobe recommends:

- Run a service within the calling user's permission context when possible

Background tasks

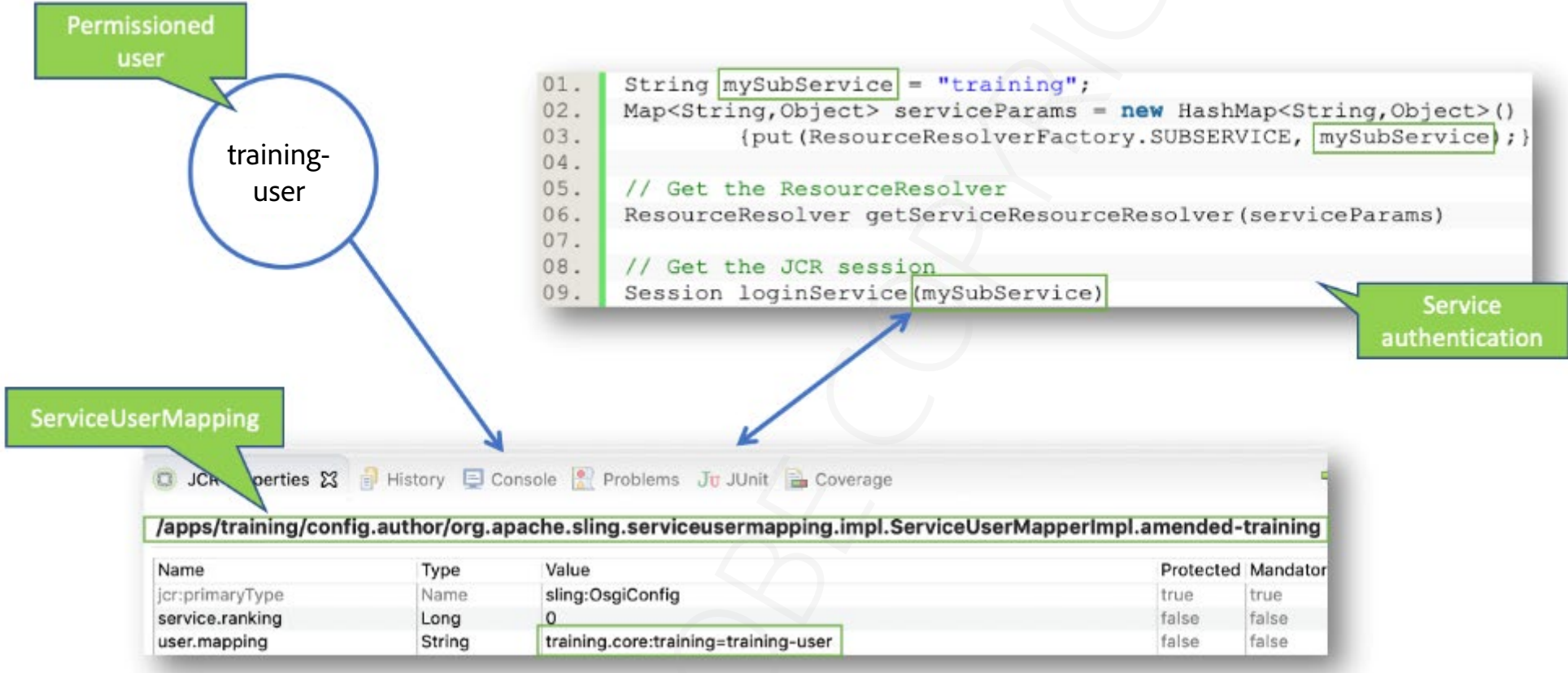
- Typically not called.
- Not associated with a specific user

Apache Sling provides a three-part mechanism for service authentication and authorization:

- Service user
- Service ID
- ServiceUserMapper



# Permission service | Service to user mapping





# 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

**SlingRepositoryInitializer** runs the code before the **SlingRepository** service is registered

- Enables initialization
- Controlled by use of repoinit script
- Useful for initialization and content migration purposes

# Repository Initialization Language (repointit)

## 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
```

# Repointit versus mutable content package

repointit	Mutable Content Package
Creates resources at startup	Resources are created after startup
Safe instruction set: with few exceptions, supported operations are additive	Removal of something is explicit
Fast and atomic operations	Performance highly dependent on the structures covered by a filter

During application deployment, *repointit* statements are executed independent of the installation of any content packages

# Repointit configuration

The screenshot displays the Adobe Experience Manager (AEM) interface for configuring the Repointit service. The top left pane shows the project structure with the file `org.apache.sling.jcr.repointit.RepositoryInitializer~training.cfg.json` selected. The top right pane shows the same file in the `config` folder of the `training` app. The bottom pane shows the content of the file, which is a JSON object with a `scripts` array containing a command to create a service user and set ACLs.

```
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
```

# Caution: Syntax

### Apache Sling Repository\_INITIALIZER Factory

Initializes the JCR content repository using repoint statements.

References

⚠ References to the source text that provides repoint statements. format is a raw URL. (references)

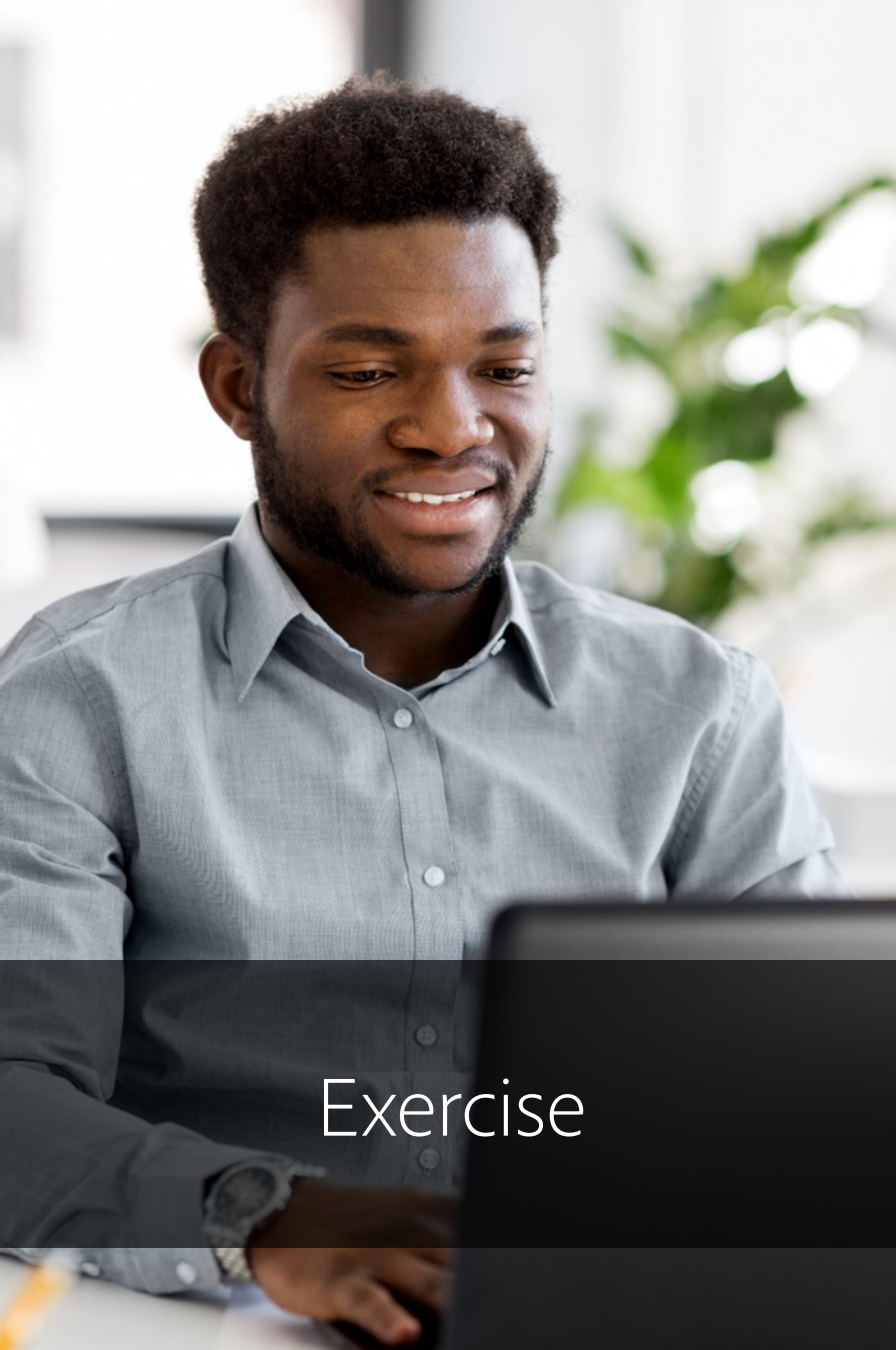
Scripts

```
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)
```

⚠ Contents of a repo init script. (scripts)

#### Configuration Information

Persistent Identity (PID)	[Temporary PID replaced by real PID upon save]
Factory Persistent Identifier (Factory PID)	org.apache.sling.jcr.repoint.RepositoryInitializer
Configuration Binding	<pre>{   "scripts": [     "create service user test-user-1\r\nSet ACL on /content\r\nallow jcr:read for test-user-1\r\nend\r\ncreate path /content/example.com(sling:Folder)\r\ncreate path /content/example.com(sling:Folder mixin mix:referenceable,mix:shareable)"   ],   "references": [     ""   ] }</pre>



## Exercise

### **Exercise 3: Use repoint statements to define service user 'training-user'**





## Key takeaways

- OSGi
  - Fundamental element in the technology stack of AEM
- Common OSGi configurations
  - Day CQ WCM Undo
  - Day CQ Root Mapping
  - Maintenance Tasks
- Sling Service
  - Piece or collection of functionality
- repointit
  - Creates resources at startup
  - performs fast and atomic operations