OSGI

What is OSGI

STANDS FOR “OPEN SERVICE GATEWAY INITIATIVE”.

JAVA FRAMEWORK FOR THE DEVELOPMENT AND DEPLOYMENT OF MODULAR SOFTWARE PROGRAMS AND LIBRARIES BY BREAKING THE APPLICATION INTO INDIVIDUAL MODULES CALLED BUNDLES SO THAT THESE BUNDLES CAN BE INDEPENDENTLY STARTED AND STOPPED.

ITS CORE SPECIFICATION DEFINES A COMPONENT AND SERVICE MODEL FOR JAVA.

THE COMPONENTS AND SERVICES CAN BE DYNAMICALLY INSTALLED, ACTIVATED, DE-ACTIVATED, UPDATED AND UNINSTALLED.

OSGI SPECIFICATION HAS SEVERAL IMPLEMENTATIONS: ECLIPSE EQUINOX, KNOPFLERFISH OSGI OR APACHE FELIX.

AEM USES APACHE FELIX.

“is the dynamic module system for Java™.”

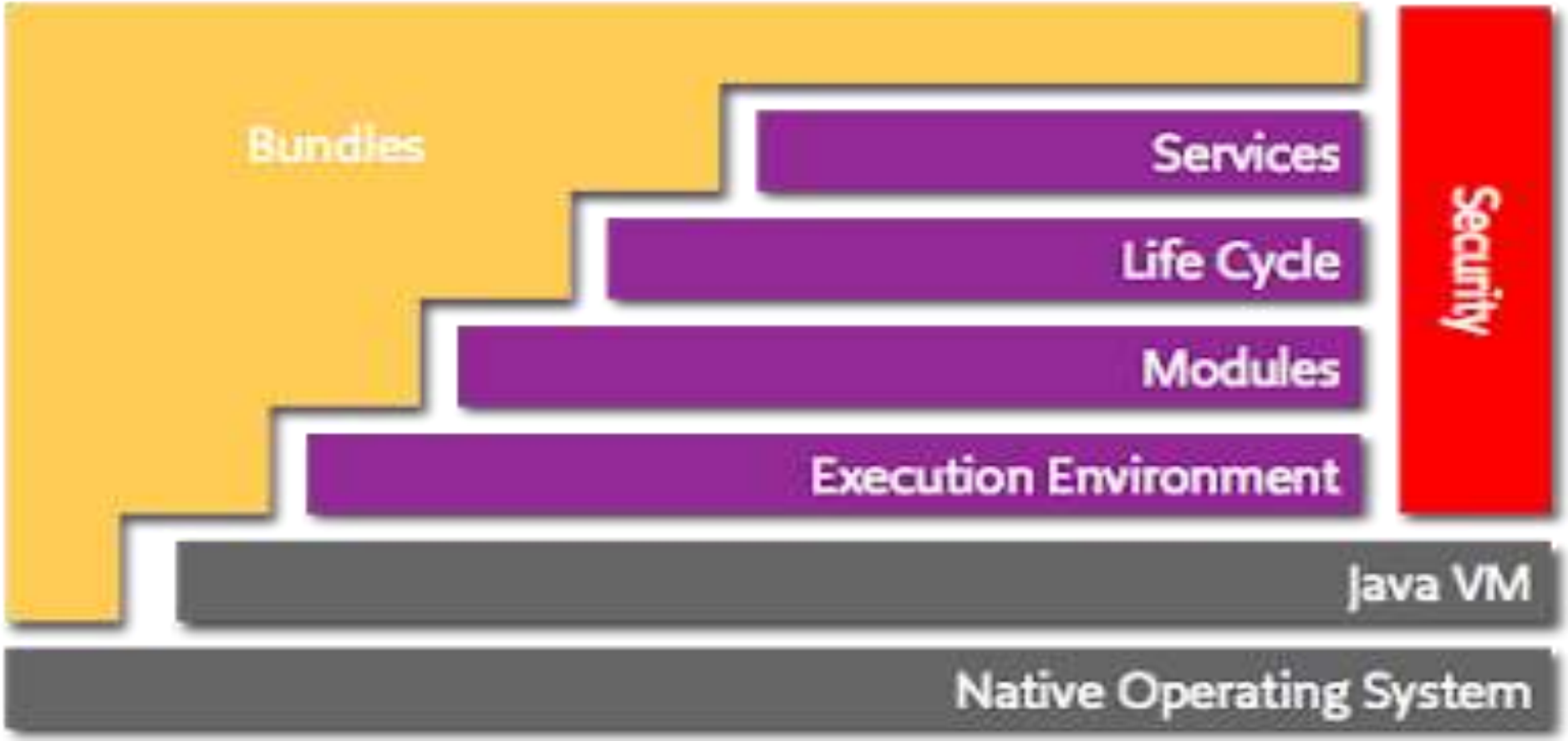
comes under the classification Universal Middleware.

“provides the standardized primitives that allow applications to be constructed from small, reusable and collaborative components. These components can be composed into an application and deployed.”

OSGi bundles can contain compiled Java code, scripts, content that is to be loaded in the repository, and configuration or additional files, as needed.

allows the bundles to be loaded, and installed, during normal operations. In the case of CQ5, this is managed by the Sling Management Console.

OSGI


Bundles
Services
Life Cycle
Modules
Execution Environment
Security
Java VM
Native Operating System

OSGi has a layered model:

- **Bundles** – Bundles are normal jar components with extra manifest headers.
- **Services** – The service layer, which hold the service-side of the framework, keeps the service registry and manages it
- **Life-Cycle** – The lifecycle layer manages and keeps track of the frameworks and bundles lifecycle state. It is used to install or uninstall framework objects and start or stop them.
- **Modules** – The module layer, which is the bundle space, holds the bundles that are installed on the framework and are managed through the lifecycle layer.
- **Security** – The security layer, which extends the jave 2 security architecture, is optional. When active, it validate the bundle signatures and controls the component access rights
- **Execution Environment** – The execution environment layer, which is the bottom layer on which the bundles live, is selected to fit the underlying hardware or operating system.

Bundle in AEM

A **bundle** is essentially a Jar file.

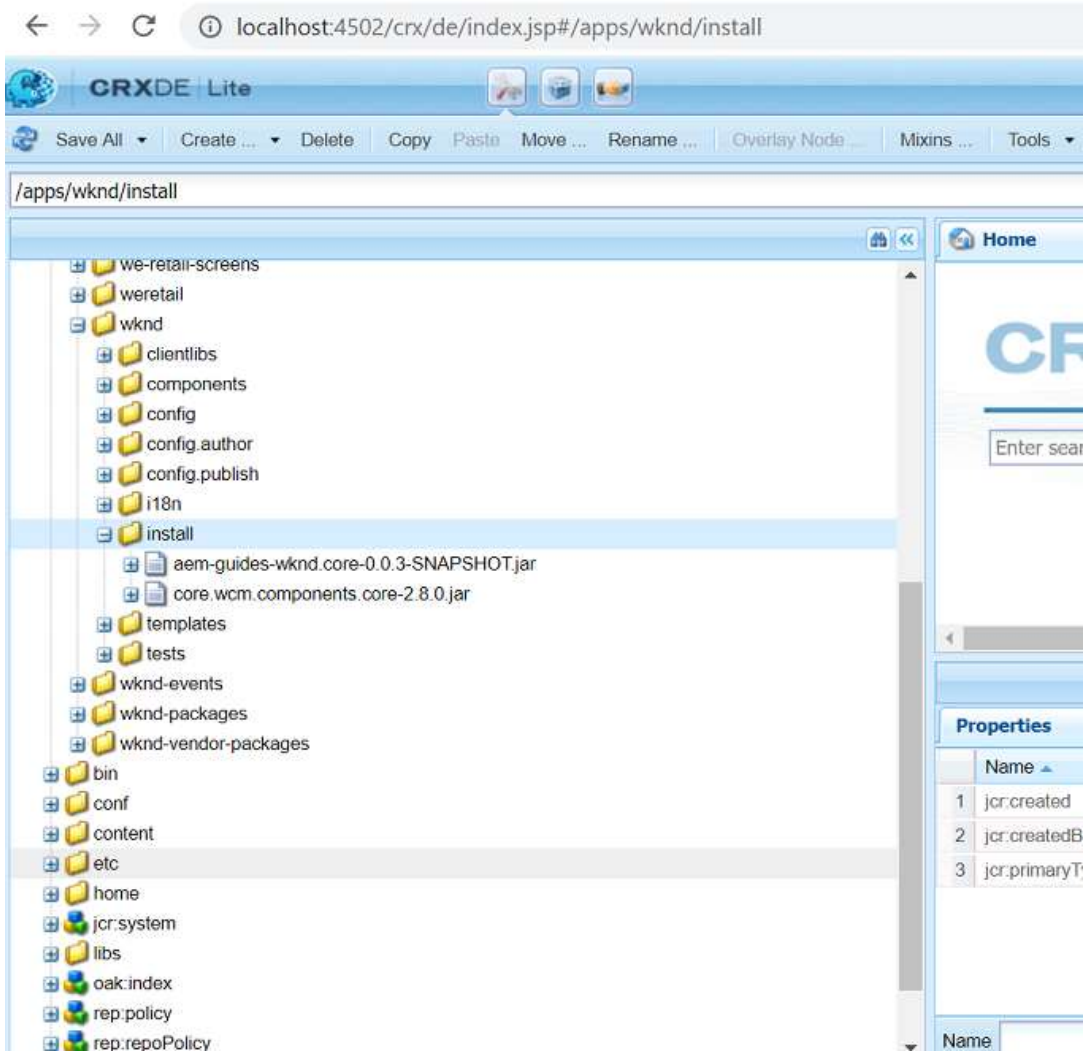
You deploy in the Apache Felix console.

Felix console is the OSGi container. To make it simpler, to accomplish a complex task, you can deploy a **bundle** in Felix console that runs with **AEM**, and then avail the services offered by the **bundle**.

Bundle(jar) = Classes + Jars + configuration Files + Manifest headers

Where we can see OSGI bundle in crx?

- login to <host>:<port>/crx/de/index.jsp
- Click apps
- Go to your project, In my case it is wknd(refer screenshot)
- Click install
- OSGI bundle is
- /apps/wknd/install/aem-guides-wknd.core-0.0.3-SNAPSHOT.jar



OSGI bundle in web console

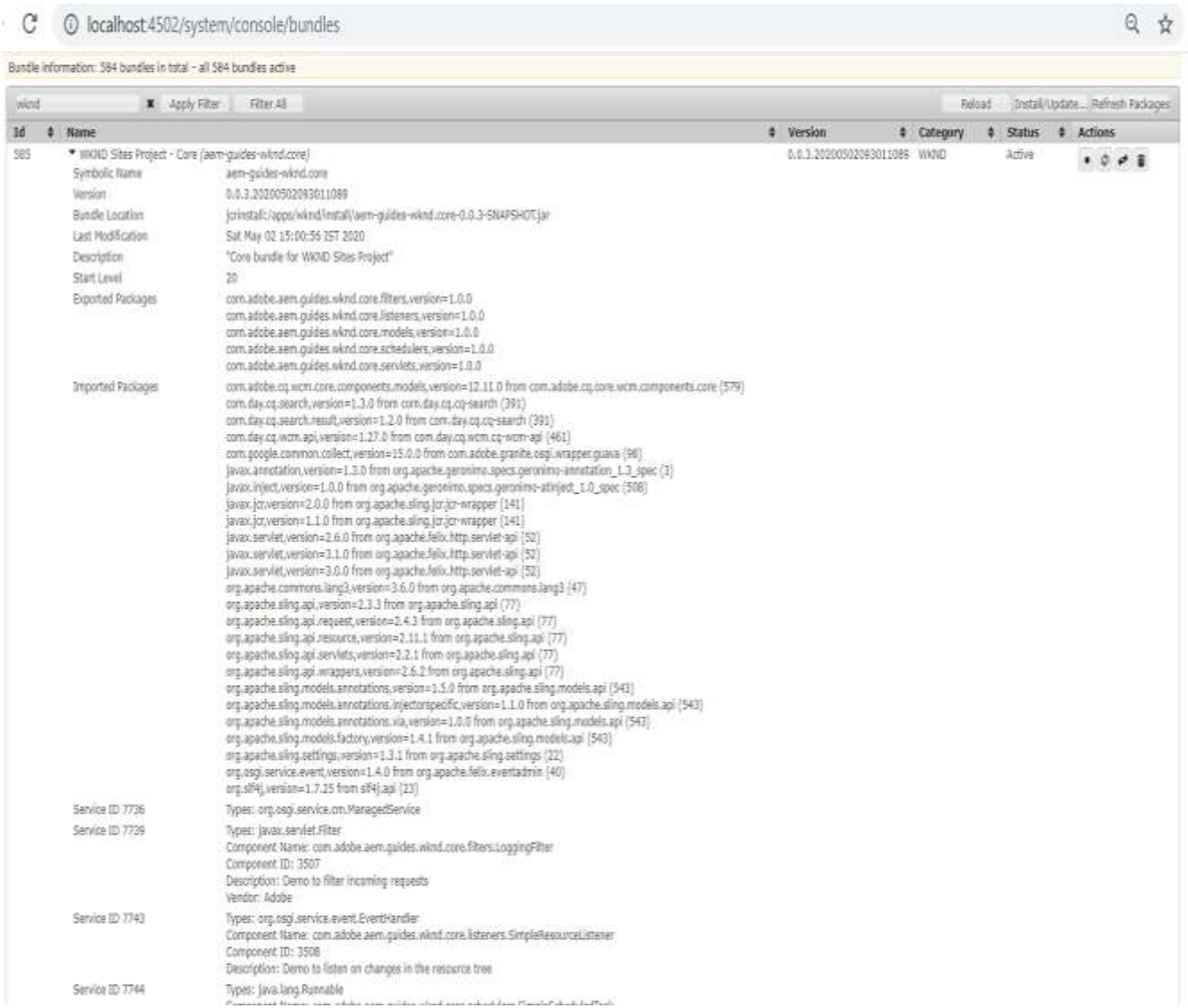
- login to <host>:<port>/crx/de/index.jsp for eg: http://localhost:4502/crx/de/index.jsp
- Go to Web console: <host>:<port>/system/console/bundles for eg:

http://localhost:4502/system/console/bundles

Find your project name:

Bundles

- A bundle must have a unique identity, a long, chosen by the Framework.
- This identity must not change during the lifecycle of a bundle, even when the bundle is updated.
- Uninstalling and then reinstalling the bundle must create a new unique identity.



Modules

Modularity is at the core of the OSGi specifications and embodied in the *bundle*concept. In Java terms, a bundle is a plain old JAR file.

OSGi hides everything in that JAR

Package vs Bundle

Package

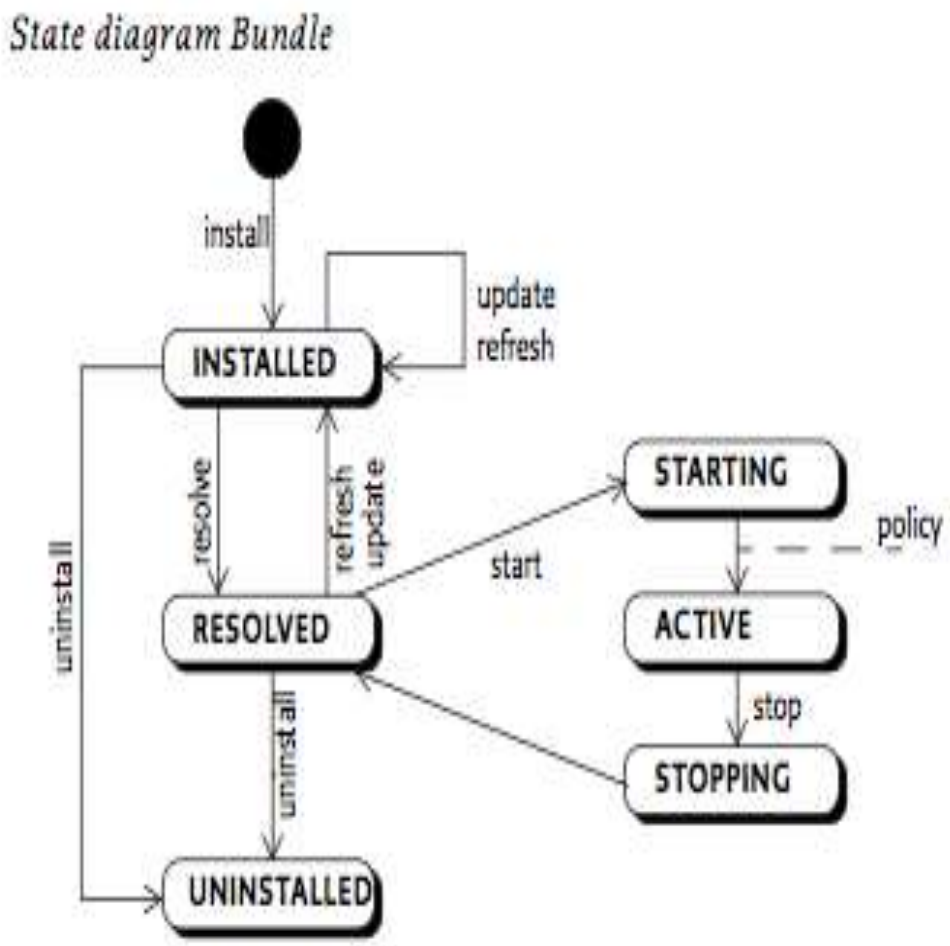
- A Package is a zip file that contains the content in the form of a file-system serialization (called “vault” serialization) that displays the content from the repository as an easy-to-use-and-edit representation of files and folders. Packages can include content and project-related data.

Bundle

- Bundle is a tightly coupled, dynamically loadable collection of classes, jars, and configuration files that explicitly declare their external dependencies (if any).

OSGI Lifecycle: Lifecycle states

- INSTALLED
- RESOLVED
- UNINSTALLED
- STOPPING
- ACTIVE
- STARTING



OSGI Lifecycle ctd..

INSTALLED: The OSGi runtime knows the bundle is there.

RESOLVED: The bundle is there and all it's prerequisites (dependencies) are available. The bundle can be started (or has been stopped).

STARTING: The bundle is being started. If it has a BundleActivator class, the BundleActivator.start() method is being executed. When done, the bundle will become ACTIVE. Note: Bundles that can be activated lazily (Bundle-ActivationPolicy: lazy) stay in this state until one of their class files is loaded.

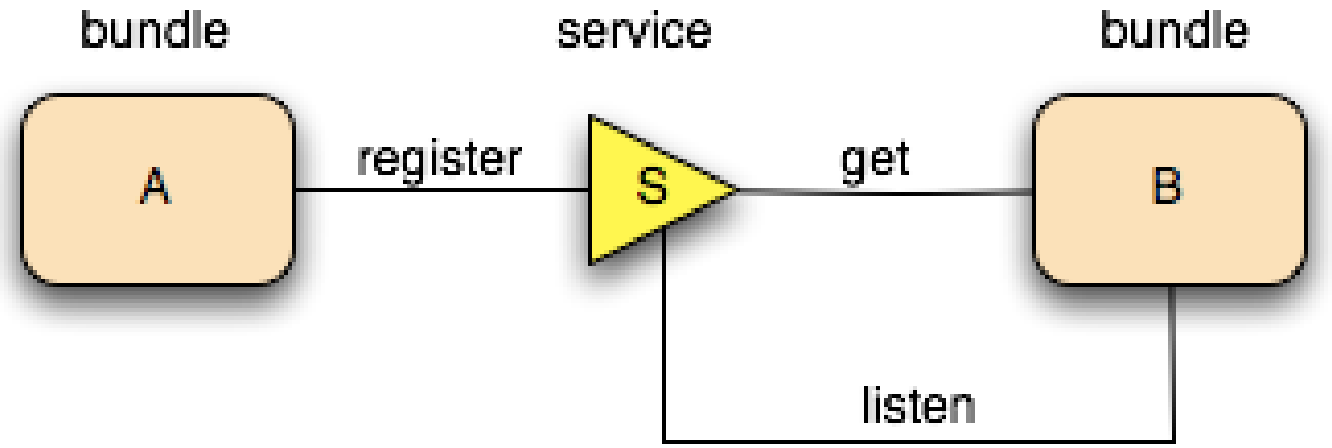
ACTIVE: The bundle is running.

STOPPING: The bundle is being stopped. If it has a BundleActivator class, the BundleActivator.stop() method is being executed. When done, the bundle will become RESOLVED

UNINSTALLED: The bundle has been removed from the OSGi runtime.

Services

- We need the service model is because Java shows how hard it is to write collaborative model with only class sharing.
- OSGi *service registry*. A bundle can create an object and register it with the OSGi service registry under one or more interfaces. Other bundles can go to the registry and list all objects that are registered under a specific interfaces or class.



Basic SCR Annotation used for developing a component or service in osgi are:-

@COMPONENT – DEFINES THE CLASS AS A COMPONENT.

@SERVICE – DEFINES THE SERVICE INTERFACE THAT IS PROVIDED BY THE COMPONENT.

@REFERENCE – INJECTS A SERVICE INTO THE COMPONENT.

@PROPERTY – DEFINES A PROPERTY THAT CAN BE USED IN THE CLASS.

@component

- The @Component annotation is the only required annotation. If this annotation is not declared for a Java class, the class is not declared as a component.

```
@Component(
    service = Servlet.class,
    property = {
        "sling.servlet.extensions=html",
        "sling.servlet.selectors=training",
        "sling.servlet.paths=/bin/trainingservlet",
                "sling.servlet.paths=/bin/trainingservlet2",
        "sling.servlet.methods=get",
        "sling.servlet.resourceTypes=my-aem-project/components/page/page"
    }
)
```

@Reference

The @Reference annotation defines references to other services made available to the component by the Service Component Runtime.

- This annotation may be declared on a Class level or any Java field to which it might apply. Depending on where the annotation is declared, the parameters may have different default values.

Sling servlets in OSGI- AEM 6.3+

```
@Component(service=Servlet.class,
        property={
                Constants.SERVICE_DESCRIPTION + "=Simple Demo Servlet",
                "sling.servlet.methods=" + HttpConstants.METHOD_GET,
                "sling.servlet.resourceTypes="+
"com.poc.osgiannotation/components/structure/page",
               "sling.servlet.paths="+ "/bin/servlet",
                "sling.servlet.extensions=" + "txt"
        })
public class SimpleServlet extends SlingSafeMethodsServlet {
{
 @Override
  protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response)throws IOException
  {
     response.getWriter().print(" I am in doGet Method");
  }
}
```

Declarative Services Component **annotation**

- Declarative services is a compile time process.
- Annotations:
  - @Designate(ocd="T.class")
  - @ObjectClassDefinition
  - @AttributeDefinition

@Designate

- Generate a Designate element in the Meta Type Resource for an ObjectClassDefinition(ocd) using the annotated Declarative Services component.

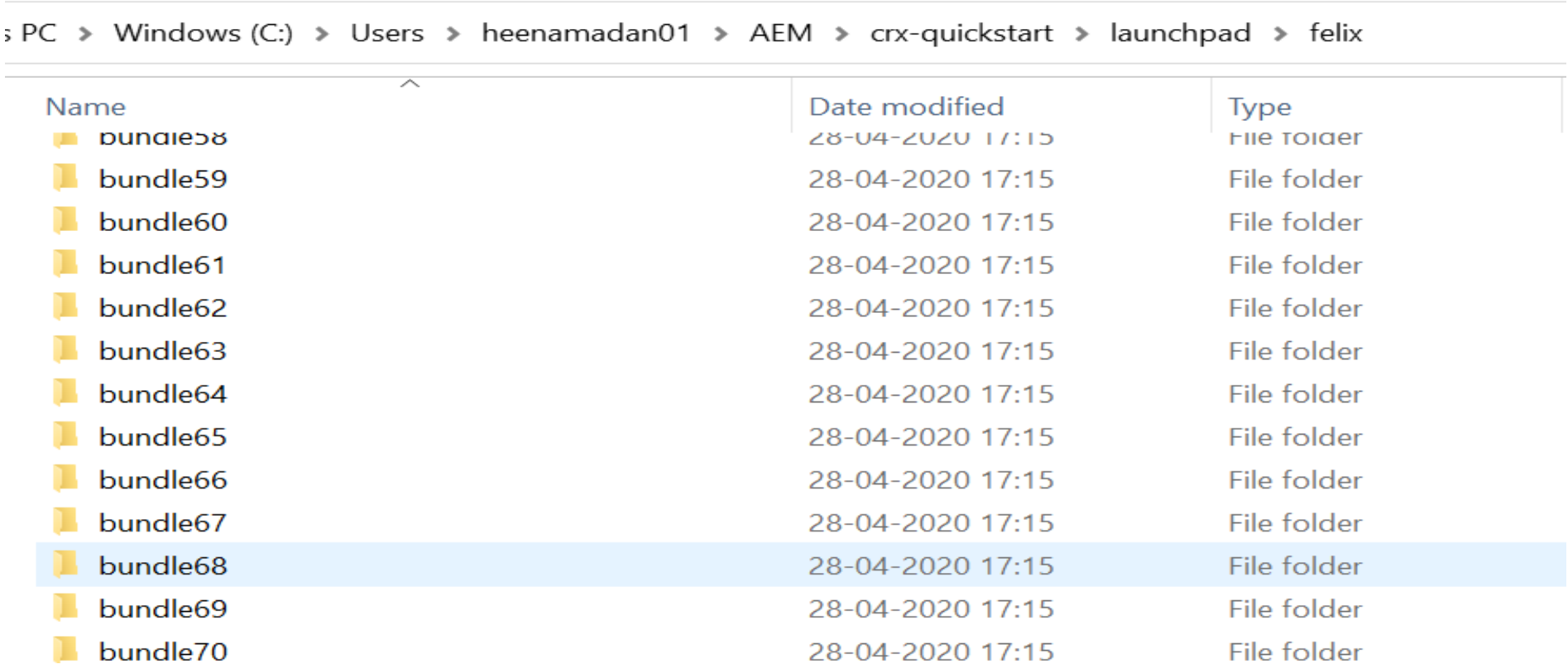
- This annotation must be used on a type that is also annotated with the Declarative Services Component annotation. The component must only have a single PID which is used for the generated Designate element.

```
@Component(
   immediate = true,
   service = Servlet.class,
   property = {
      "sling.servlet.resourceTypes=project/components/component"
   }
)
@Designate(ocd = SampleOsgiServlet.Configuration.class)
public class SampleOsgiServlet extends SlingSafeMethodsServlet {

   @Activate
   protected void Activate(Configuration config) {
      boolean enabled = config.servletname_enabled();
   }

   @ObjectClassDefinition(name="OSGi Annotation Demo Servlet")
   public @interface Configuration {
      @AttributeDefinition(
         name = "Enable",
         description = "Enable the servlet"
      )
      boolean servletname_enabled() default false;
   }
}
```

OSGI BASICS

- OSGI is a modular programming approach. Application can be divided into modules or bundles. Bundle will be JAR file + Metadata.
- OSGI uses Apache Felix Implementation.
- All Bundles are deployed on Felix container.
- Every bundle has its own life cycle. i.e., it’s independent. Can be redeploy without affecting other bundle.
- Each Bundle has its own class loader. Which allows developers to start and stop each bundle separately.
- OSGI supports multiple version of bundle.
- AEM works with the inbuilt bundles for separate functionalities.

- Bundles are stored under crx-quickstart/launchpad/felix.



s PC > Windows (C:) > Users > heenamadan01 > AEM > crx-quickstart > launchpad > felix

Name	Date modified	Type
bundle58	28-04-2020 17:15	File folder
bundle59	28-04-2020 17:15	File folder
bundle60	28-04-2020 17:15	File folder
bundle61	28-04-2020 17:15	File folder
bundle62	28-04-2020 17:15	File folder
bundle63	28-04-2020 17:15	File folder
bundle64	28-04-2020 17:15	File folder
bundle65	28-04-2020 17:15	File folder
bundle66	28-04-2020 17:15	File folder
bundle67	28-04-2020 17:15	File folder
bundle68	28-04-2020 17:15	File folder
bundle69	28-04-2020 17:15	File folder
bundle70	28-04-2020 17:15	File folder

OSGi Configuration with the Web Console

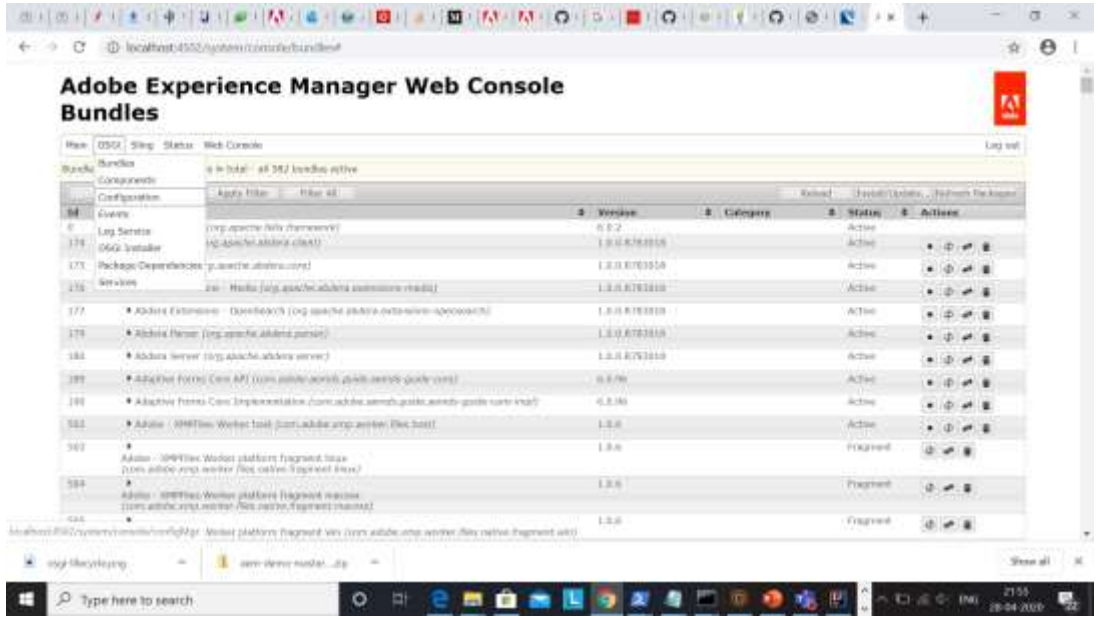
- Go to web console:
- http://localhost:4502/system/console/bundles

- Under OSGI-> Click Configuration

OR

Directly go to url:

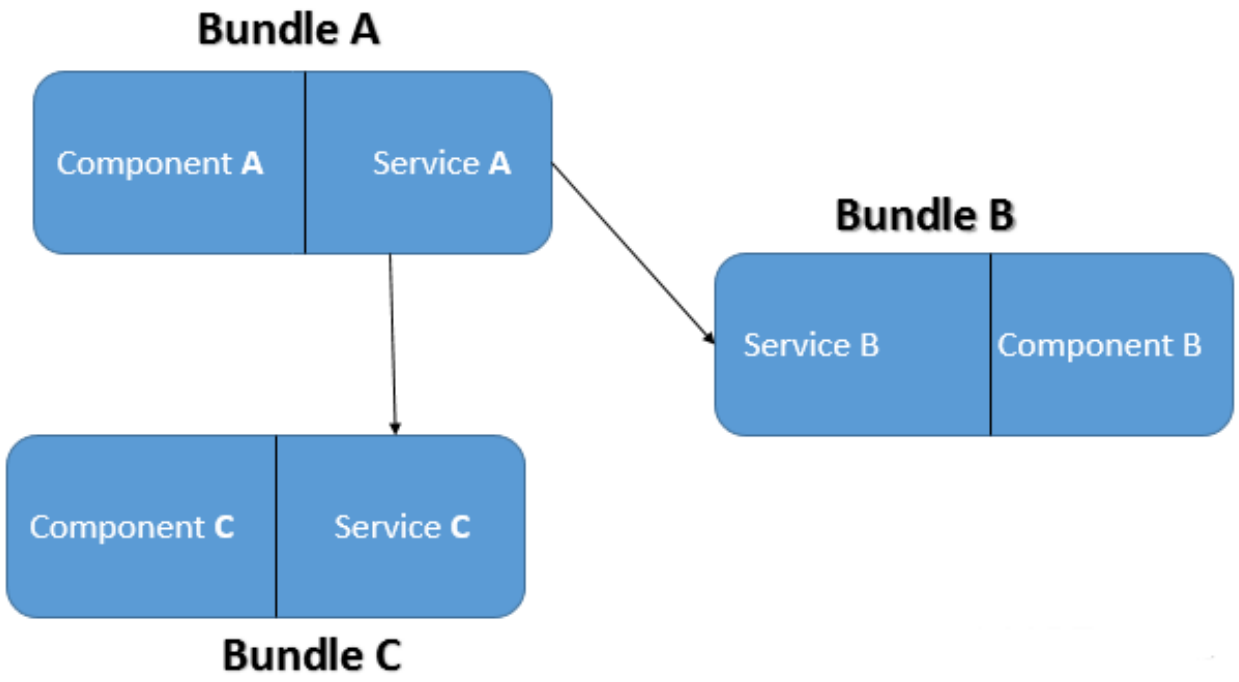
http://localhost:4502/system/console/configMgr



Basic SCR Annotation used for developing a component or service in osgi are:-

- @Component – defines the class as a component.
- @Service – defines the service interface that is provided by the component.
- @Reference – injects a service into the component.
- @Property – defines a property that can be used in the class.

OSGI DI



- A of Bundle A has dependency on Class B & C of Bundle B & C , Now OSGI will export Class B & Class C and import them into Bundle A to resolve dependency.

Run Modes

Run modes allow you to tune your AEM instance for a specific purpose.

for example

author	Publish	Test	Development	Production etc.

Starting CQ with a specific run mode

There are many ways to set run modes of AEM instances:

1) Using the sling.properties file.

- The sling.properties file can be used to define the required run mode:
- Edit the configuration file:
- <cq-installation-dir>/crx-quickstart/conf/sling.properties
- Add the following properties; the following example is for author:

sling.run.modes=author

Using jar file

- The jar file must use the naming convention:

  cq5-<run-mode>-p<port-number>
- For example, set the publish run mode by naming the jar file:

cq5-publish-p4503

It sets as publish run mode

•

Using the -r option

- A custom run mode can be activated by using the -r option when launching the quickstart.
- Use below command to start your Aem instance with “author” as run mode

```
java -jar cq-56-p4502.jar -r author
```

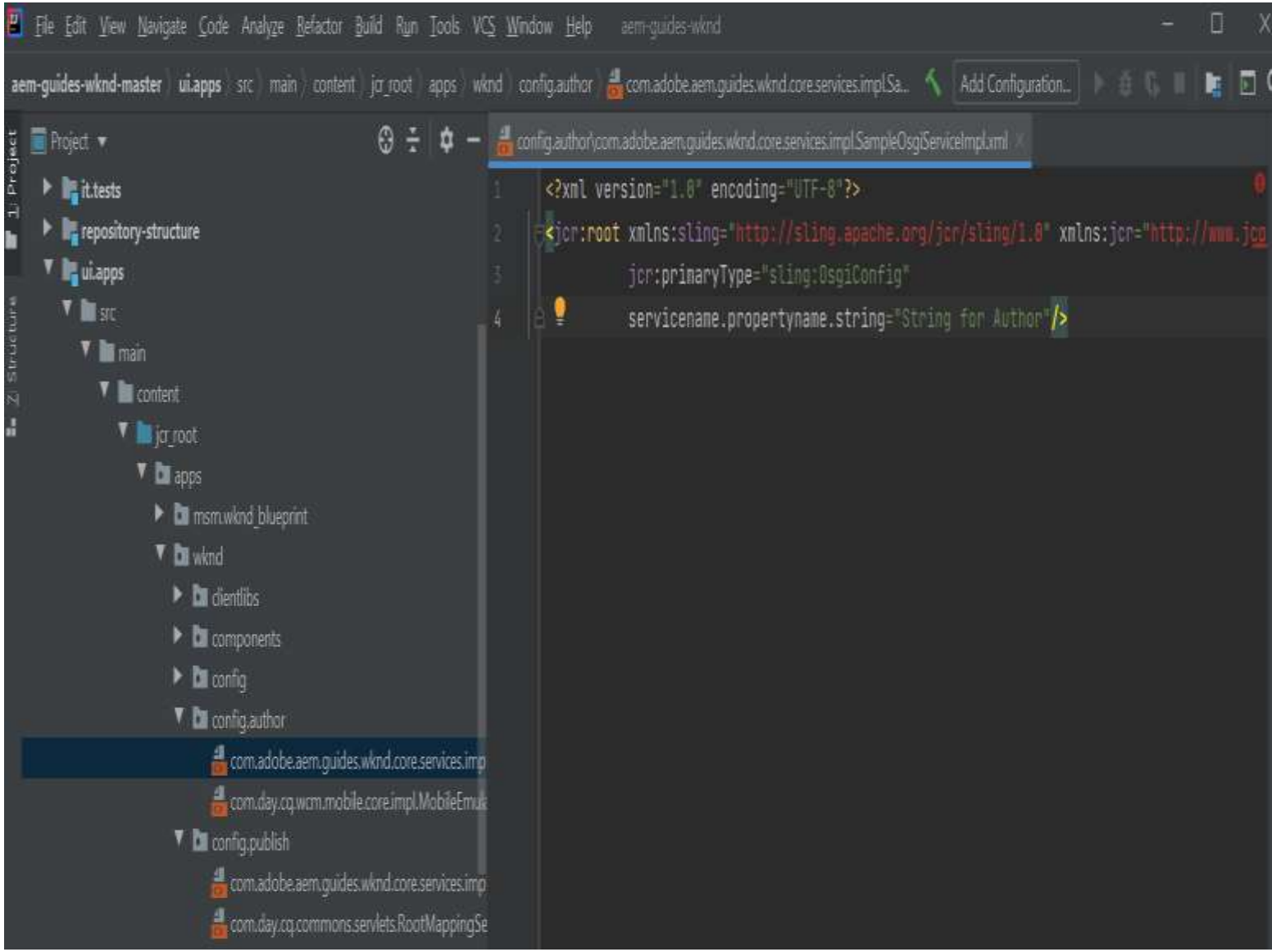
Defining configuration properties for a run mode

- A collection of values for configuration properties, used for a particular run mode, can be saved in the repository.
- The run mode is indicated by a suffix on the folder name.

  For example:
- Config - Applicable for all run modes
- config.author - Used for author run mode
- config.publish - Used for publish run mode

Configuring OSGi

By Configuring files:



To update a configuration with the web console:

- Access the **Configuration** tab of the Web Console.
- Go to http://localhost:4502/system/console/bundles
- Click Osgi -> Configuration

Or directly go to

http://localhost:4502/system/console/configMgr

- Select the bundle that you want to configure. Open configuration

edit value-> click save

localhost:4502/system/console/configMgr/com.adobe.aem.guides.wknd.core.servlets.SampleOsgiServlet

AEM Communities User Generated Content Contribution Limits Service -

AEM Communities User Sync Listener -

AEM Communities Utilities -

Annotation Demo Service - OSGi

Boolean Property ☑

Sample boolean value (servicename.propertyname.boolean)

String Property: String for Author

Sample String property (servicename.propertyname.string)

Dropdown property: DAYS

Sample dropdown property (servicename.propertyname.dropdown)

String Array Property: foo, bar

Sample String array property (servicename.propertyname.string.array)

Password Property: ••••••••••

Sample password property (servicename.propertyname.password)

Long Property: 0

Sample long property (servicename.propertyname.long)

Configuration Information

Persistent Identity (PID)	com.adobe.aem.guides.wknd.core.services.impl.SampleOsgiServiceImpl
Configuration Binding	Unbound or new configuration

Cancel Reset Delete Unbind Save

✔ 200 : org.apache.jackrabbit.oak.security.authentication.token.TokenLoginModule (sufficient) Apache Felix JAAS Support

Ways to Create AEM projects

- 1) Normal way
  - create maven/gradle project.
  - Inside that parent project create modules
  - Add dependencies in pom.xml(in maven case)
- 2) Using Eclipse Plugin.

Installing AEM Eclipse Plugin

- 1. Goto Help, Install New Software….
  2. Click Add and
  enter **http://eclipse.adobe.com/aem/dev-tools/** in Location and click OK.

Project Creation in Eclipse:

- **Creating a new project**
- You can create a new project by performing these steps:
- 1. Open the Eclipse IDE.
- 2. Switch to the AEM perspective, to have the panels arranged in a convenient way: Menu Window –> Open Perspective –> Other… –> AEM –> OK.
- 3. Click on the new project icon.
- 4. Select AEM –> AEM Sample Multi-Module Project.
- 5. Select version 10 of the Maven Archetype, which is a blueprint used for the project that is going to be created.

Steps for creating an OSGI Bundle in AEM:-

- Run the below Maven command:

mvn archetype:generate -DarchetypeRepository=http://repo.adobe.com/nexus/content/groups/public/ -DarchetypeGroupId=com.day.jcr.vault -DarchetypeArtifactId=multimodule-content-package-archetype -DarchetypeVersion=1.0.2 -DgroupId=com.aem -DartifactId=MyFirstOSGIBundle -Dversion=1.0-SNAPSHOT -Dpackage=com.aem -DappsFolderName=aemcq5tutorials -DartifactName="MY First OSGI Service" -DcqVersion=“5.6.1" -DpackageGroup="Aem Cq5 Tutorials“

Follow #39,40

When prompted for confirmation, Specify **Y.**

C:\windows\system32\cmd.exe - mvn archetype:generate -DarchetypeRepository=http://repo.ado...

```
e-1.0.2.jar (12 KB at 1.9 KB/sec)
Downloading: http://repo.adobe.com/nexus/content/groups/public/com/day/jcr/vault
/multimodule-content-package-archetype/1.0.2/multimodule-content-package-archety
pe-1.0.2.pom
Downloaded: http://repo.adobe.com/nexus/content/groups/public/com/day/jcr/vault/
multimodule-content-package-archetype/1.0.2/multimodule-content-package-archetyp
e-1.0.2.pom (5 KB at 4.1 KB/sec)
[INFO] Using property: groupId = com.aem
[INFO] Using property: artifactId = MyFirstOSGIBundle
[INFO] Using property: version = 1.0-SNAPSHOT
[INFO] Using property: package = com.aem
[INFO] Using property: appsFolderName = aemcq5tutorials
[INFO] Using property: artifactName = MY First OSGI Service
[INFO] Using property: cqVersion = 5.6.1
[INFO] Using property: packageGroup = Aem Cq5 Tutorials
Confirm properties configuration:
groupId: com.aem
artifactId: MyFirstOSGIBundle
version: 1.0-SNAPSHOT
package: com.aem
appsFolderName: aemcq5tutorials
artifactName: MY First OSGI Service
cqVersion: 5.6.1
packageGroup: Aem Cq5 Tutorials
 Y: : y
```

- Then go to project directory and run:

  mvn eclipse:eclipse

```
C:\Project\practice>cd MyFirstOSGIBundle

C:\Project\practice\MyFirstOSGIBundle>mvn eclipse:eclipse
[INFO] Scanning for projects...
```

Benefits Of OSGI

Reduced Complexity

Reuse

Real World – The OSGi framework is dynamic.

Easy Deployment

Dynamic Updates

Versioning

Fast

Secure

DEMO

References

- https://docs.osgi.org/javadoc/r6/cmpn/org/osgi/service/metatype/annotations/Designate.html
- https://docs.adobe.com/content/help/en/experience-manager-65/deploying/configuring/configure-runmodes.html
- http://www.aemcq5tutorials.com/tutorials/create-osgi-bundle-in-aem/

- Note: Follow adobe documents for AEM relevant stuff.