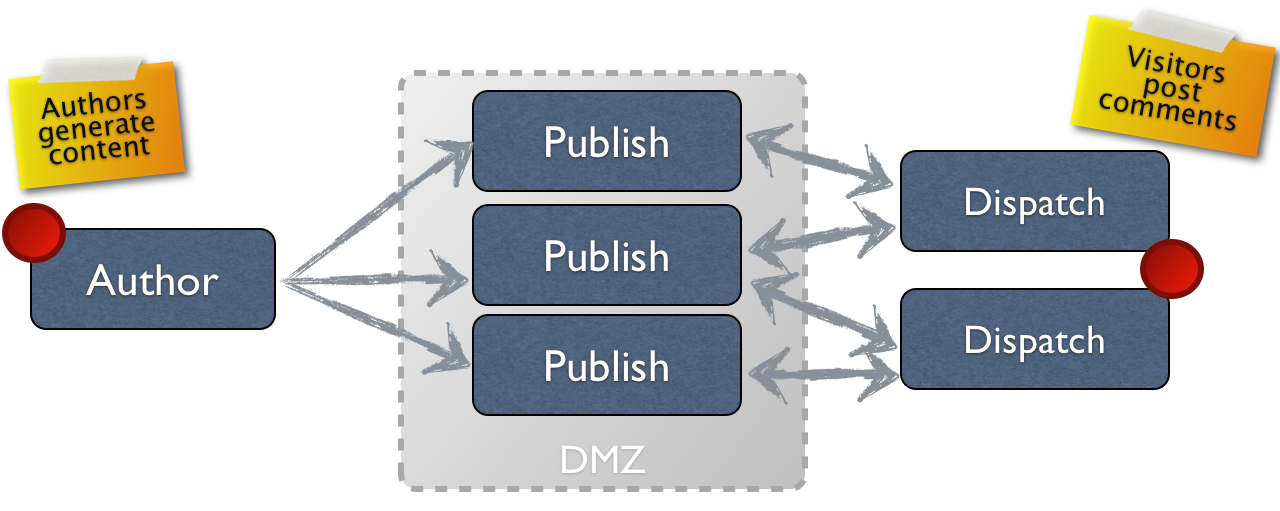
what is the use of aem tools ?

AEM is a content management system for building websites, mobile apps, and forms



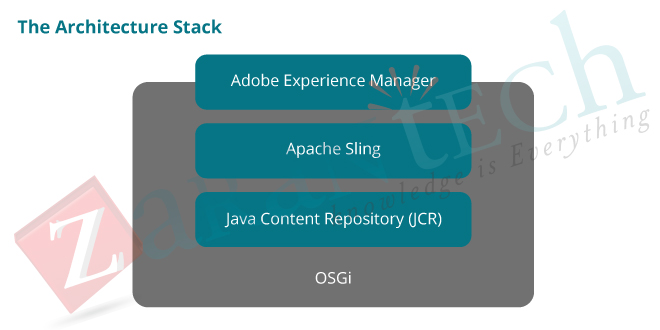
🡪 The author environment  for creating, updating, and reviewing this content before actually publishing it

what is dispatcher in aem ?

Dispatcher in aem used for caching and load balancing . Dispatcher also helps to protect your AEM server from attack.

what is component in aem ?

Component is a building block of aem.AEM components are used to hold the data. We can insert the data in the component by using dialog box



AEM UI---SLING FRAMEWORK—REPO--JRE

AEM is built on top of the Apache Sling framework and under that we have a jcr.

Aem related data like structure data or unstructured data , we can not access directly for that apache sling provide the sling framework through this we can access the jcr

* Apache Sling Request Processing
* Apache Sling is a RESTful framework to access a java content repository over HTTP protocol.
* Apache Sling is responsible for mapping the URL to the appropriate resource in the content repository

**Folder Structure of the AEM Repository**

## /apps

Here we do the development

/conf

All configurations of your site. Stored Dynamic templates and Policies

## /content

All contents of your site is stored here.

## /libs

All libraries and definition that belongs to Adobe Experience Manager code

## /tmp

A temporary working area.

**Core Components**:

AEM provides a set of out-of-the-box components known as "Core Components

**Custom Components**:

While Core Components cover common use cases, you can also create custom components tailored to your project's unique needs. Custom components are built using HTML, CSS, and JavaScript, and you can define their behavior using Sightly (HTL) or JSP scripting languages.



Page Rendering Component:s

Page Rendering Component is useful for rendering process. It is responsible for assembling content, applying the design, and generating the final HTML output for web pages. It plays a central role in delivering a consistent and dynamic web experience to users

Foundation Components :

Foundation Componentsare a set of fundamental building blocks and out-of-the-box components provided by Adobe for creating web pages and experiences.

Foundation Components and Core Components are sets of pre-built components that help developers and content authors create web pages and manage content.

PROXY COMPONENT :

WHAT IS POLICIES IN AEM ?

"policies" refer to the configuration settings and rules that can be applied to components

Core Components TYPES:

Adobe has developed most commonly used components across the applications

• Available of Core components

Template Components - Page, Navigation, language Navigation, Breadcrumb &

Quick search

Page authoring components - Title, Text, Image, Button, Teaser, Download, List, Experience Fragment, Content Fragment List, Embed, Social Media Sharing, Separator, Progress Bar & PDF Viewer

Container components - Container, Carousel, Tabs & Accordion

Form components - Form Container, Form Text, Form Option, Form Hidden & Form

Submit

Dialog :

• Dialogs are graphical interfaces which provide input to components

Authors use dialog boxes to customize the component behavior or content

There are two types:

Touch UI Dialogs

Classic Ul Dialog

Design Dialog :

In summary, Dialogs are for content authors to configure specific instances of components on pages, while Design Dialogs are for template developers or administrators to define default settings for components within templates

Overriding/Extend:

You can inherit component properties by using sling:resourceSuperType property.

Overlaying:

In AEM the default functionality is available in "/libs". Overlaying means copying the functionality from "/libs" to "/apps" at exact same path. You can do the desired changes in newly copied functionality under "/apps"

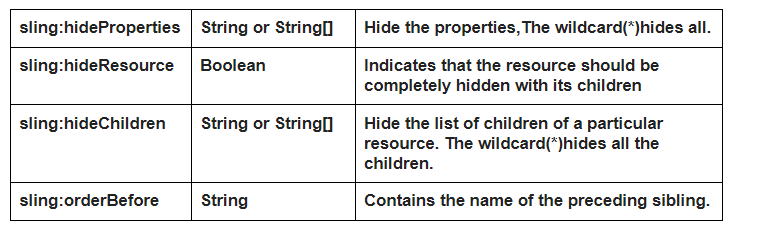
• The resource merger provides following properties

sling:hide Properties

sling:hideResource

sling:hideChildren

sling:orderBefore



what is template in aem ?

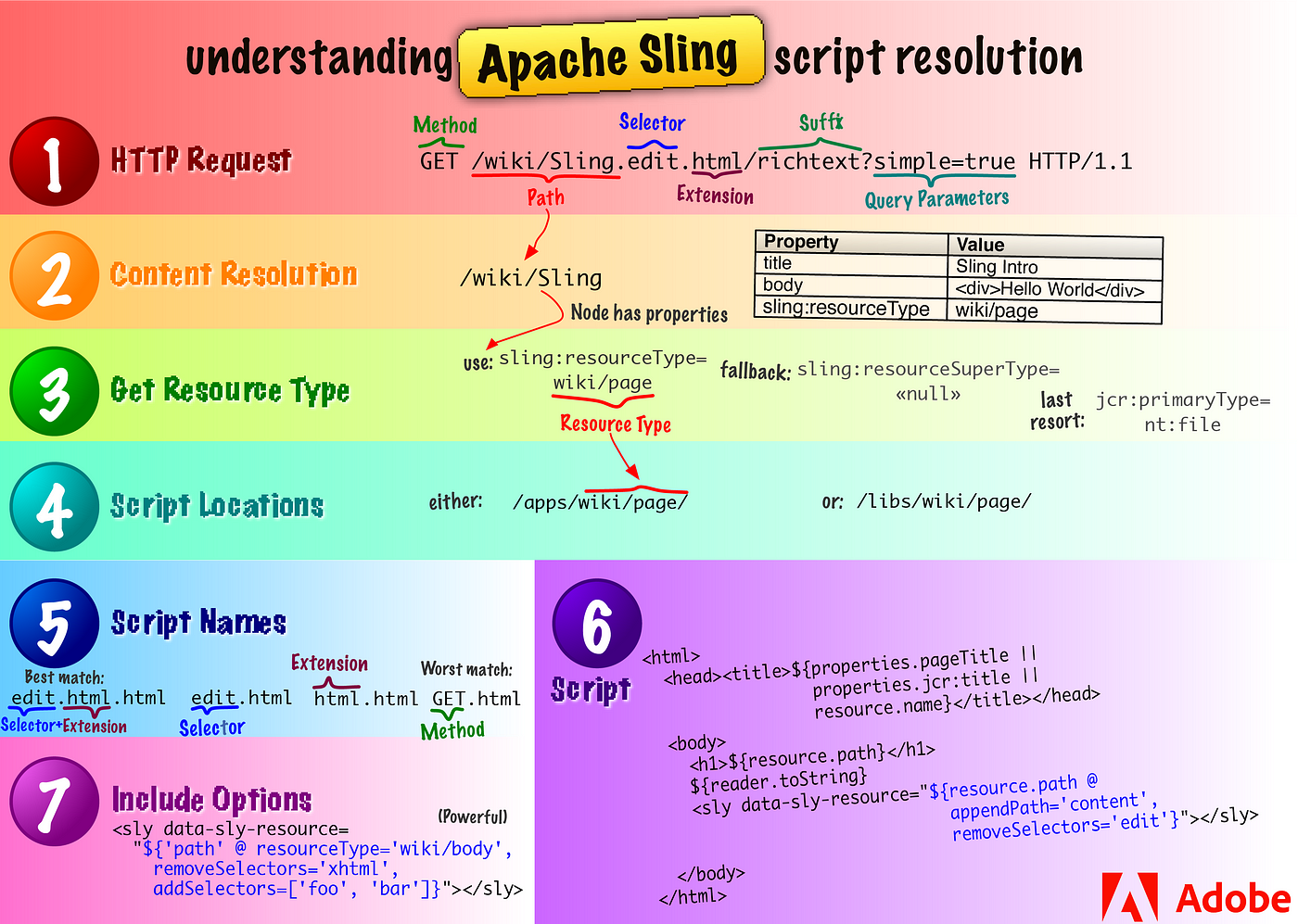
template used to define the structure and layout of web pages

template is a base for creating pages

#### ****Sling Resource Resolution – Mapping URL to Respective JCR Node :****

**If the component contains .html and .jsp files , then the first preference goes to html or jsp?**





Whenever request comes from the end user , first it checks the resource, If no resource is found , search without file extension , If no resource found , return status code 404 , If it there it check the script location under /apps folder if not found it checks in the /libs floder , If it found in /apps folder , it will check the script name based on that page rendering will be happen

#### ****Lets decompose above analysis in simple point, Below is the priority of resolution in Descending order(Highest to lowest):-****

Between the path and extension is called selector.

1. Selector +Extension .jsp –> **test.myComponent.html.jsp**
2. Selector.jsp –> **test.jsp**
3. Extension.jsp –>**html.jsp**
4. Node Name.jsp –> **myComponent.jsp**
5. method name.jsp(Based on type of request we make Either GET or POST) –> **GET.jsp** or **POST.jsp**

**page.selector.html—page.html.html—page.html.jsp-- html.jsp--page.html ---page.jsp**

#### ****What is the use of EditConfig node in creating a component?****

cq:EditConfig node is used to define the behavior of the component.

# Quick Publish vs Manage Publish AEM

https://www.aemcq5tutorials.com/tutorials/manage-publish-aem/

#### ****What are the differences between package and 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)

#### ****Explain life cycle of OSGI[Open Systems Gateway initiative] bundle?****

OSGi is a framework which allows modular development of applications using java.  
A large application can be constructed using small reusable components(called bundles in terms of OSGi) each of which can be independently started, stopped, and also can be configured dynamically while running without requiring a restart.

Following are the states of OSGI life cycle:

* **Installed –**The bundle has been successfully installed.
* **Resolved –**All Java classes that the bundle needs are available. This state indicates that the bundle is either ready to be started or has stopped.
* **Starting –**The bundle is being started, the BundleActivator.start method will be called, and this method has not yet returned. When the bundle has an activation policy, the bundle will remain in the STARTING state until the bundle is activated according to its activation policy.
* **Active –**The bundle has been successfully activated and is running; its Bundle Activator start method has been called and returned.
* **Stopping –**The bundle is being stopped. The BundleActivator.stop method has been called but the stop method has not yet returned.
* **Uninstalled –**The bundle has been uninstalled. It cannot move into another state.

jcr:primaryType means in aem ?

In Adobe Experience Manager (AEM), the **jcr:primaryType** property is a mandatory property that specifies the primary node type of a JCR (Java Content Repository) node. The JCR is the underlying technology that AEM uses to store and manage its content.

The **jcr:primaryType** property indicates the node type of a particular node in the content repository. Nodes in AEM are instances of node types, and the primary type defines the fundamental characteristics and properties of that node.

HOW TO MANUALLLY BUILD THE CODE WITHOUT USING ANY TOOLS LIKE MAVEN ?

### ****Compile Java Code:****

* If your AEM project includes Java code, you'll need to compile it. You can use the **javac** command to compile Java source files

javac -cp path/to/dependencies:. -d classes src/main/java/your/package/YourClass.java

### ****Create JAR File:****

* Once your Java code is compiled, you may need to package it into a JAR file. You can use the **jar** command for this.

jar cvf your-project.jar -C classes/ .

### ****Deploy to AEM:****

* Copy the JAR file and any other necessary files to the appropriate directories in your AEM instance.
* For example, if your code includes OSGi bundles, you might copy the JAR file to the **/apps** or **/libs** directory in the AEM repository.

WHICH MAVEN VERSION YOU ARE USING ?

Find the version of maven=mvn –v

Apache maven 3.8.8

Java version 11.0.15

AEM VERSION—6.5.5

HOW TO CREATE THE CUSTOM COMPONENT PROCESSES ?

When we create the component by default it creates the jsp file ,we need to change the jsp to html.

We have to give the component group after that we have to copy the cq:dialog and paste it in

Our component

HOW TO CREATE A TEMPLATE ?

In the aem tool we have a two options one is navigation and another one is tools option , for creating the template we have to go to the tools option ,in the tools we can find the template option after clicking that template option we can see the create button ,we have to click the create button after clicking we have to select template type , based on that template will be created.

HOW TO CREATE A STATIC TEMPLATE ?

WHAT ARE THE MODES ARE AVAILABLE IN PAGE ?

EDIT

LAYOUT

DEVELOPER

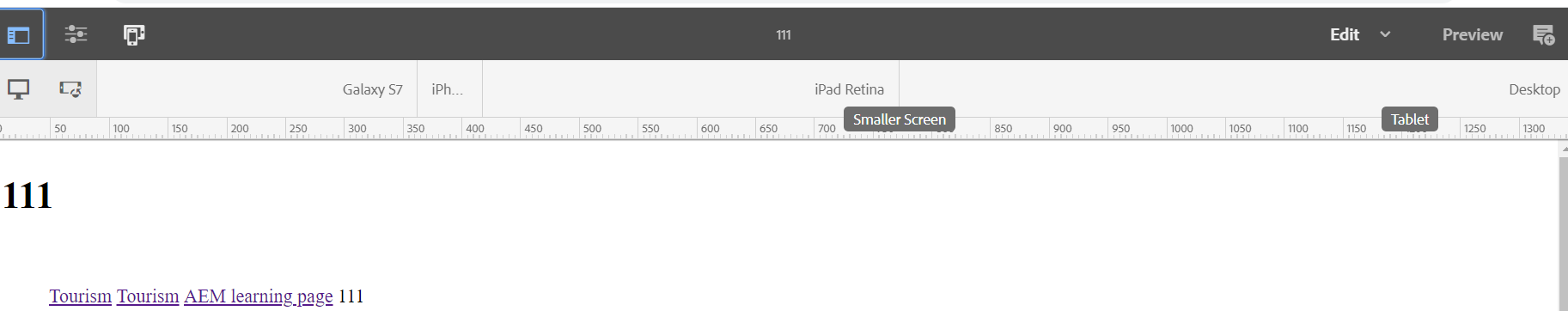
TIMEWRAP

WHAT ARE THE MODES ARE AVAILABLE IN TEMPLATE ?

STRUCTURE

INITIAL CONTENT

LAYOUT



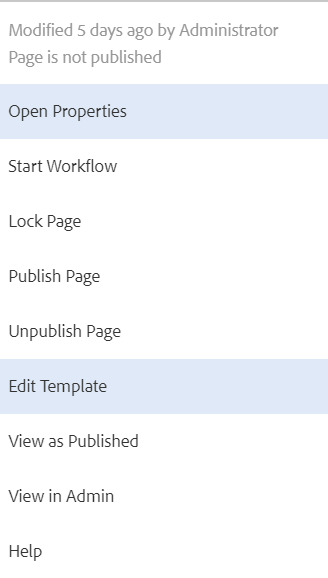
TOGGLE SIDE PANNEL:

Assests

Components

Content tree

PAGE INFORMATION:



EMULATOR:

Emulator nothing but layout

How to create the dynamic image and title using slightly ?

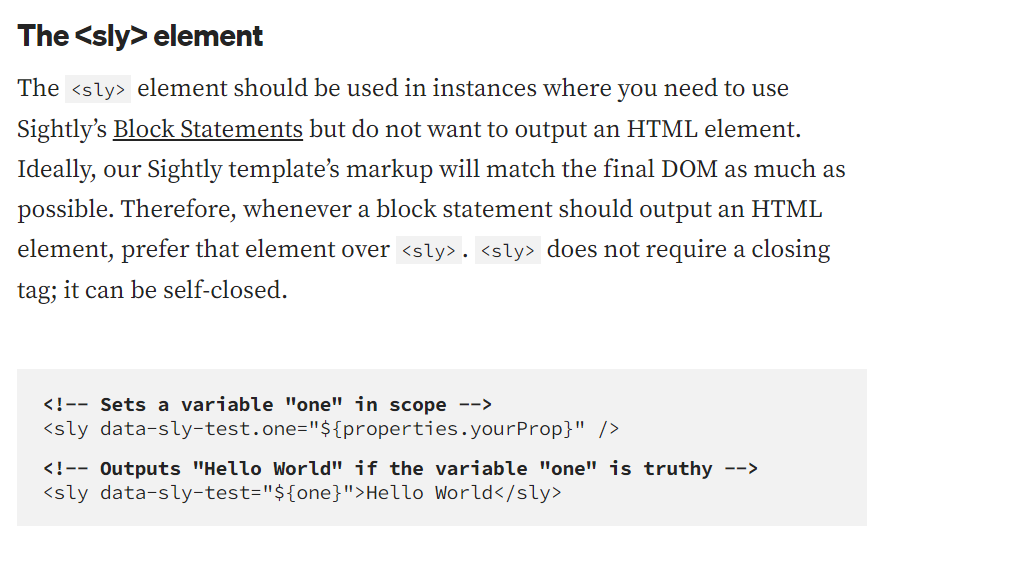
<h1>${properties.numberfield @context ='html'}</h1>

<h1>${properties.text @context ='html'}</h1>

<img src="${properties.fileReference}" alt="Image Name" width="300" height="200">

What is a POM?

Project Object Model. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects.



TO GIVE THE COMMENTS TO THE SIGHTLY CODE🡺 <!--/\*sightly/\*-->

<!------------------------------------

1>Data-Sly-Test:  
 Used to test the conditions of given expression

<div data-sly-test="${wcmmode.edit}">

gopal

</div>

------------------------------------------------------------------------------------

<div data-sly-test ="${properties.title =='Gopal'}">

<h1>${properties.title }</h1>

</div>  
-------------------------------------->

2>Data-Sly-Use:

<!-- Used to initial the java model object and ready to access the public   
 getter methods from that model  
-->  
<div   
 data-sly-use.javaSlingModelObject="${'com.project.sightly.JAVACLASS'}">  
 ${javaSlingModelObject.title}  
</div>

<!-- Used to access the clientlib markup template -->  
<sly data-sly-use.clientlib="/libs/granite/sightly/templates/clientlib.html" />

🡨---------------------------------------

3>Data-Sly-Include:

<!-- Used to replace the content with included .html or .jsp  
-->  
*<div data-sly-include="path/to/any.html">  
 Element to be replaced  
</div>*

*<div data-sly-include="${'/apps/mysite/components/demoBreadcrumb/malaibred.html'}"></div>*

*----------------------------------------🡪*

*4>Data-Sly-Resource:*

<!-- Used to include the component resource in other component and   
 resolves the component using sling resolution  
-->  
<sly data-sly-resource="${'path' @   
resourceType='project/components/examplecomponent'   
cssClassName='ANY\_ADDITIONAL\_CLASSS' decoration=true}" />

🡨---------------------------------------

*4>Data-Sly-List:*

<!-- Used to print the list of child elements wrapped around the parent  
 // <span> is displayed based on the size of the children  
-->  
<div data-sly-list="${page.listChildren}">  
 <span>Index Value: ${itemList.index}</span>  
</dl>

*----------------------------------------🡪*

5>*Data-Sly-Repeat:*

<!-- Used to print the parent and children elements  
 // <div> and <span> is displayed based on the size of the children  
  
<div data-sly-repeat="${page.listChildren}"

<span>Index Name: ${item.name}</span>  
</div>

🡨---------------------------------------

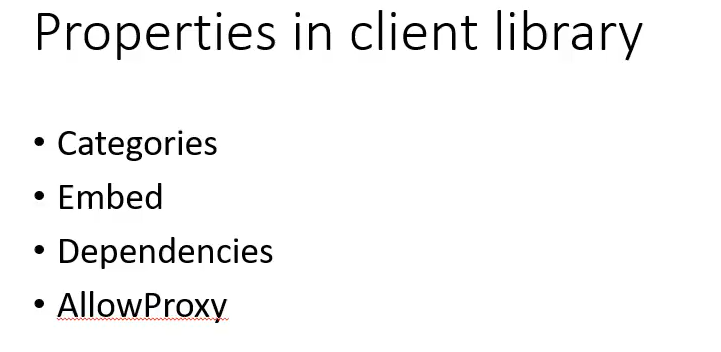
6>Data-Sly-Call:

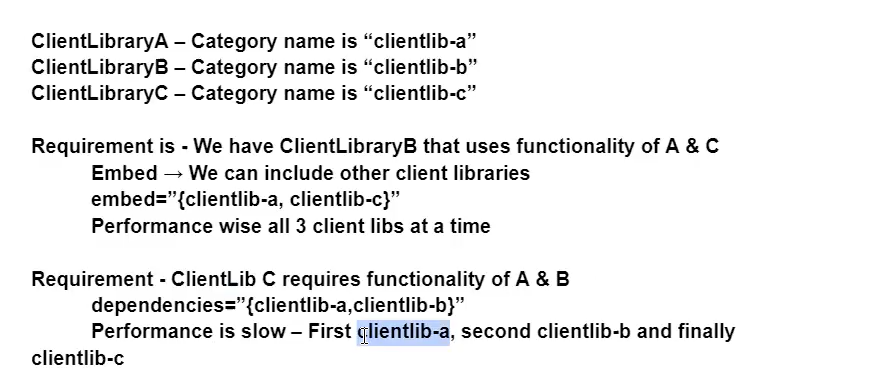
<!-- Used to call the template .  
-->  
<div data-sly-call="${templateExample @ heading=properties.jcr:title}">  
 Div that gets replaced  
</div>  
*----------------------------------------🡪*

**Client Libraries:** clientLibraries are collections of client-side resources such as CSS, JavaScript.

Modern websites rely heavily on client-side processing driven by complex JavaScript and CSS code. Organizing and optimizing the serving of this code can be a complicated issue.

To help deal with this issue, AEM provides **Client-side Library Folders** .

**

**

*🡪in embed all clientlib all started at a time*

*🡪in dependencies all clientlibs started individually one after the other*

*🡪in realtime mostly used embed*

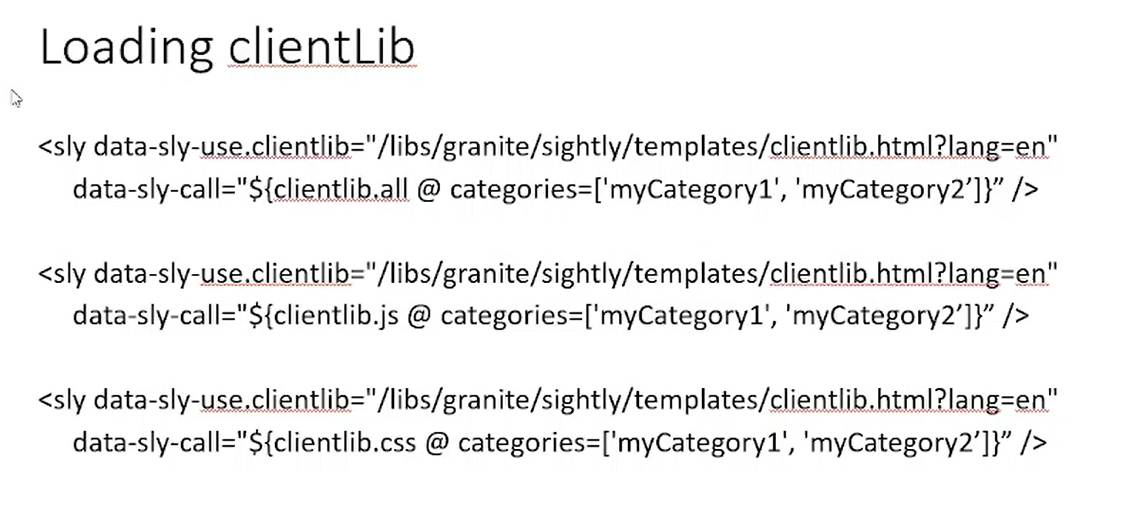
### Using HTL

In HTL, client libraries are loaded through a helper template provided by AEM, which can be accessed through [data-sly-use](https://helpx.adobe.com/experience-manager/htl/using/block-statements.html#use). Three templates are available in this file, which can be called through [data-sly-call](https://helpx.adobe.com/experience-manager/htl/using/block-statements.html#template-call):

* **css** - Loads only the CSS files of the referenced client libraries.
* **js** - Loads only the JavaScript files of the referenced client libraries.
* **all** - Loads all the files of the referenced client libraries (both CSS and JavaScript).
* 

the target directory is used to house all output of the build.

The target directory is created by Maven. It contains all the compiled classes, JAR files etc. When executing the mvn clean command, Maven would clean the target directory. The webapp directory contains Java web application, if the project is a web application.

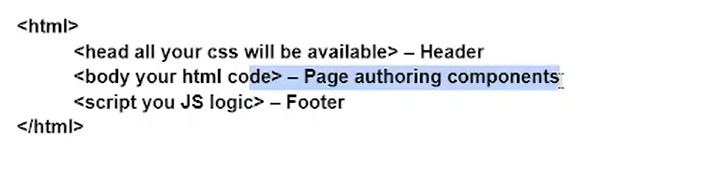


🡪LAST METHOD IS USED CALL THE CSS FILES IF WE HAVE HTML AND JS FILES ALSO,IT CALLS ONLY CSS FILES

🡪SECOND METHOD USED TO CALL THE JS FILES

🡪FIRST METHOD USED TO CALL ALL THE FILES IN PAGE



**

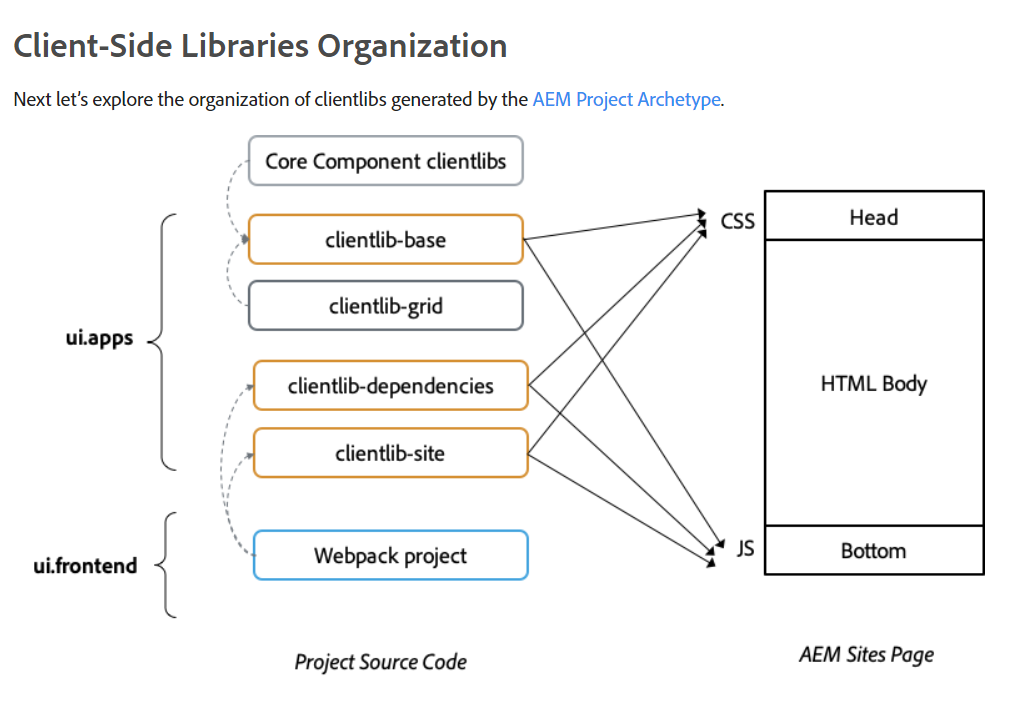
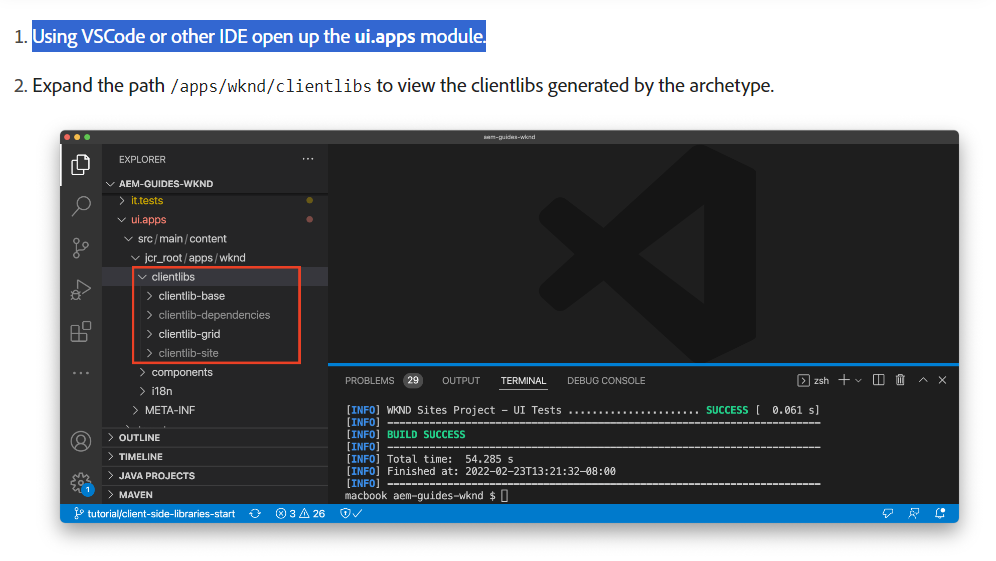
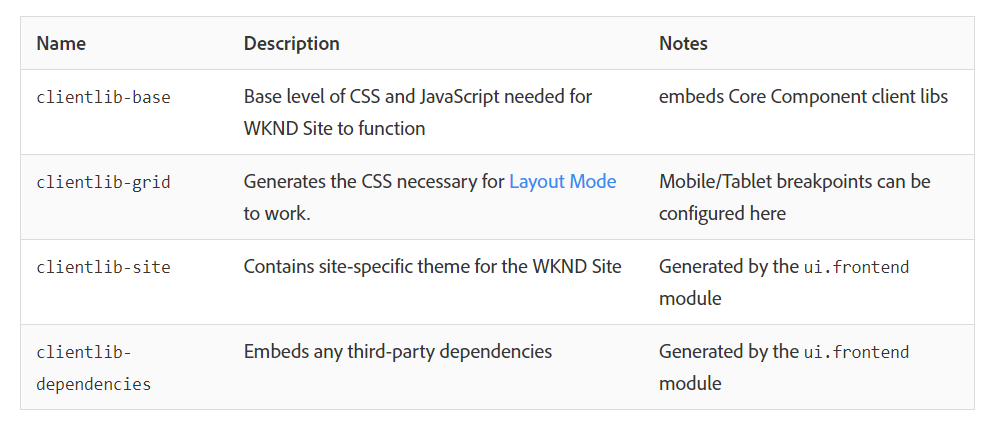
**Client-side scripting**

Source code is visible to the user.

HTML, CSS, and javascript are used.

**Server-side scripting**

Source code is not visible to the user because its output   
of server-sideside is an HTML page.   
 PHP, Python, Java, Ruby are used.

1. 
2. **Maven Archetype:**
3. In Maven, an archetype is a project template or a predefined project structure that serves as a starting point for creating new projects. Archetypes help standardize the project layout, dependencies, and configuration. They provide a quick way to bootstrap a new project with a certain structure and pre-configured settings.
4. When you create a new Maven project, you can use an archetype to generate the initial project structure. Maven archetypes come with predefined configurations for different types of projects, such as web applications, JAR libraries, and more.
5. * *
6. Observe that clientlib-site and clientlib-dependencies are ignored from source control. This is by design, since these are generated at build time by the ui.frontend module.

**categories** – This is the identifier into which categories a clientlib belongs. A clientlib can have one or more categories.

WHAT ARE BENEFITS OF USING TAGS ?

Tags are used for organizing our aem pages and assets. As an author begins typing a keyword in the content finder ,suggested tags will be appered on the drap down based on that author select pages or assests.

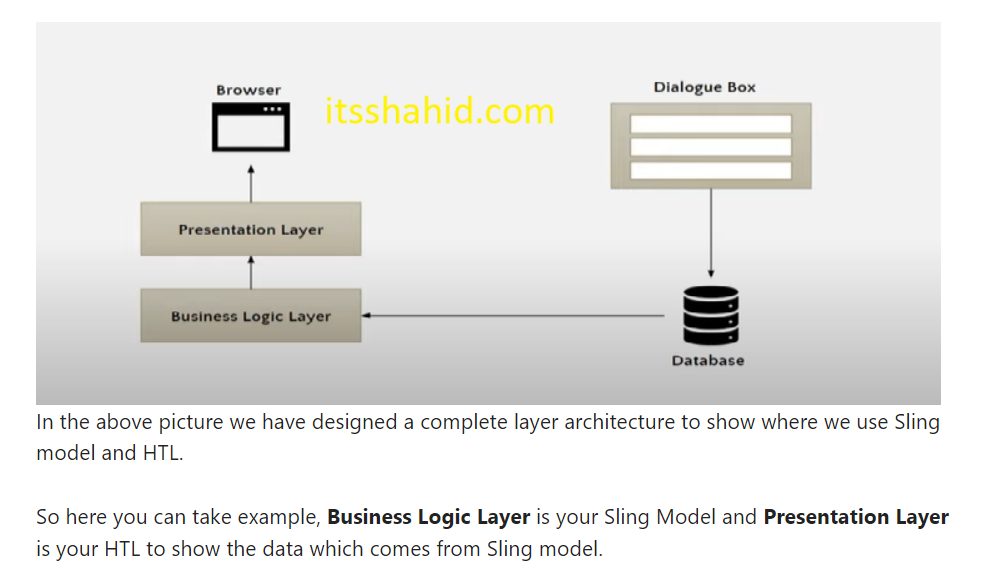
🡪with the namespace option we can categorize the tags based on the departmen

what is style system in aem ?

The Style System allows a template author to define style classes in the policy of a component so that a content author is able to select them when editing the component on a page

**embed**helps us in combining all clientlibraries and load it as single CSS and JS file. If **A** category embeds two different categories as B and C. The resulting CSS and JS will have one single JS and CSS file.

**allowProxy**  access client libraries via proxy servlet.



Sling Models:

Sling Models are annotation driven Java™ “POJOs” (Plain Old Java™ Objects) .

sling model mapping the data from the JCR to Java™ variables.

1. **Sling Models (Java):**
   * If you need to fetch data from an external API, process it, and present it in a specific format, using Sling Models in Java allows you to encapsulate this logic in a Java class.
   * For complex forms that require server-side validation or processing, you might use Sling Models to handle the form submission logic.
2. **HTL (CRXDE):**
   * For simple components like a text or image component where the rendering logic is straightforward, HTL might be sufficient without the need for Java.
   * When building a component that primarily focuses on presentation and doesn't involve complex business logic, HTL can provide a clean and simple solution.

In practice, a combination of both Sling Models and HTL is often used. Sling Models handle the complex business logic, data processing, and integration aspects, while HTL focuses on rendering and presenting the content in a clean and readable way. The choice between the two depends on the specific requirements of the component and the development team's preferences and expertise.

Top of Form

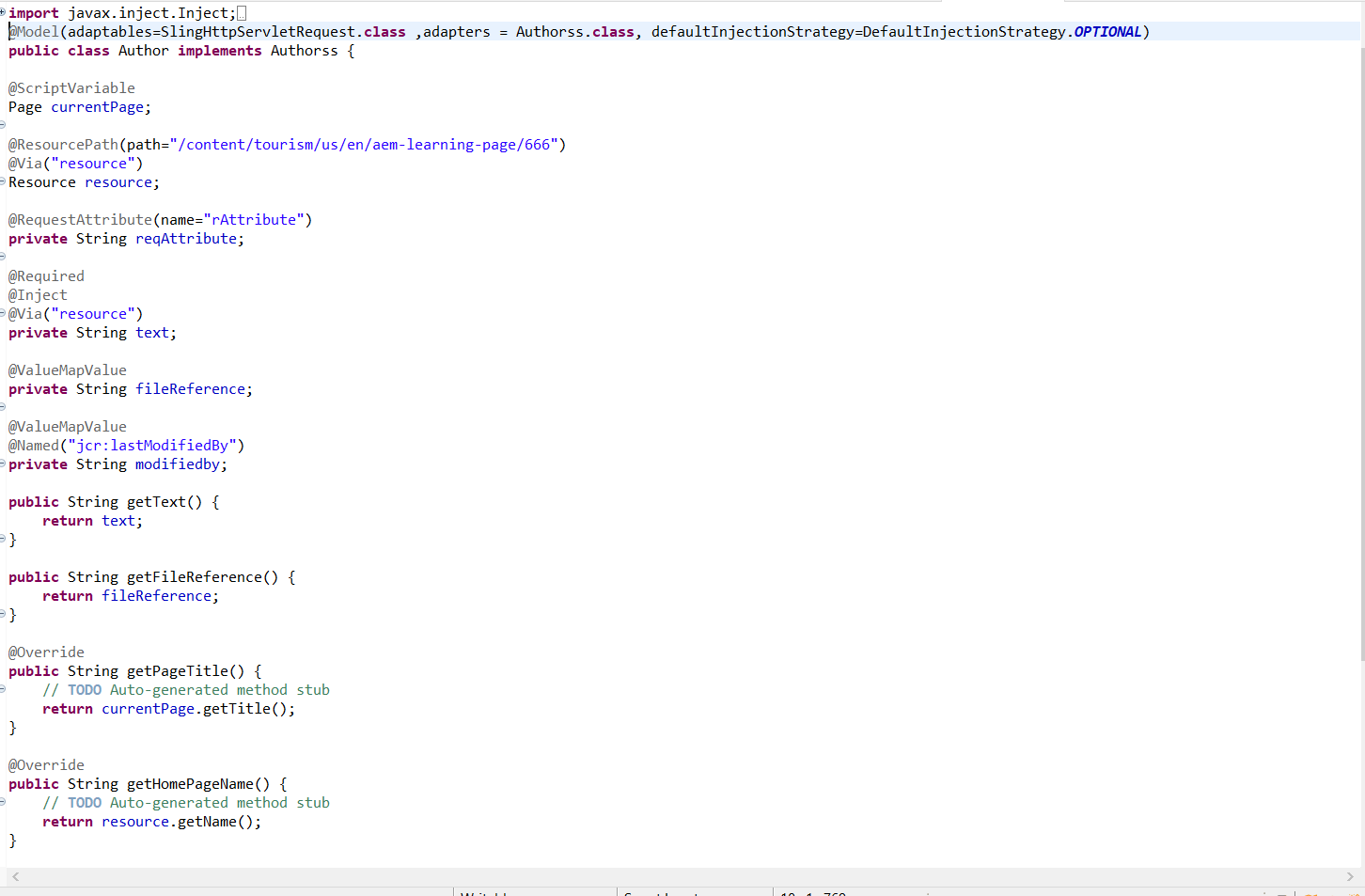
Let's consider a task where you need to implement a component that displays a list of products with discounted prices based on certain business rules.

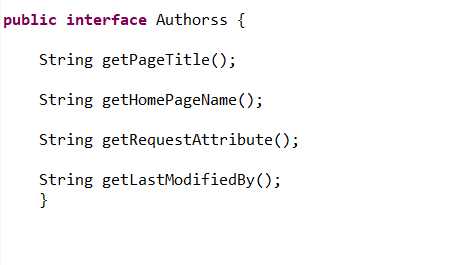
**Task: Implementing a Product List with Discounted Prices**

1. **Sling Models (Java):**
   * **Business Logic:** Implementing complex business rules for calculating discounted prices based on various factors such as user roles, product categories, or promotions.
   * **Integration with External Systems:** Fetching dynamic pricing information from an external service or API and incorporating it into the product display.
   * **Data Transformation:** Performing complex data transformations on product data before rendering it in the HTML.

**HTL (CRXDE):**

* **Rendering Markup:** HTL is excellent for rendering the HTML markup based on the data provided by the Sling Model.
* **Simple Display Logic:** HTL is suitable for basic display logic and conditional rendering based on simple criteria.
* In this example, the complex business logic, data manipulation, and integration with external services are implemented in the Java class (**DiscountedProductListModel**). HTL is then used to iterate through the list of discounted products and render the HTML markup.
* This demonstrates the clear separation of concerns between business logic (Sling Models) and presentation (HTL). While HTL is great for rendering HTML, it is less suitable for handling complex business rules and data manipulation, which are better suited for Sling Models in Java.





🡪We can convert pojo class to Sling model with the help of **@Model** annotation declare on top of the class.

🡪@Inject — This annotation is used to inject OSGi services and other dependencies into the Sling Model.

**We need to avoid using @Inject as value injection happens at runtime. If we use @Inject then you are making the framework do extra analysis.**

🡪@Optional — This annotation is used with @Inject to indicate that the injected dependency is optional.

🡪@Via — This annotation is used to specify the path to the property in the content tree that should be mapped to the annotated field.

🡪@Named — This annotation is used to specify the name of the property in the content tree that should be mapped to the annotated field.

🡪@Default — This annotation is used to specify a default value for the annotated field.

🡪@PostConstruct — This annotation is used to mark a method that should be called after the Sling Model has been instantiated and all of its dependencies have been injected.

**🡪@Named**: If there is a need to change the getter of any attribute like (sling:resourceType, jcr:primaryType) @Named annotation helps to achieve this.

**🡪@Default:** A default value can be provided for Strings or primitive data types.If there is no value of that property, default value takes place.

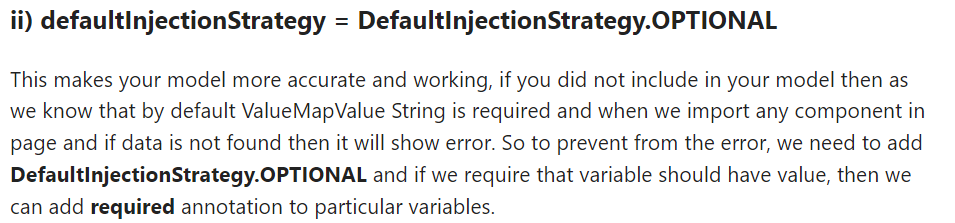
|  |
| --- |
| @Model(adaptables = Resource.class)  public class Test {      @Inject @Default(values = "/content/test")    String path;    public String getPath() {        return path;    }  } |

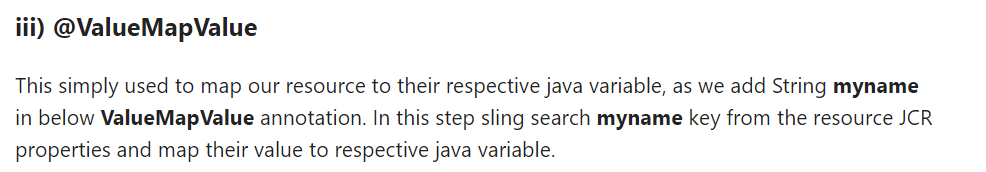
🡪**@Inject:** This annotation is used to inject a property, resource, request anything.This is a generic annotation, which traverses all the sling model injectors based on service ranking.

|  |
| --- |
| @Model(adaptables = Resource.class)  public classTest {    @Inject    String path;    public String getPath() {        return path;    }  } |

The `adaptables` attribute in the `@Model` annotation specifies the adaptable types that the AEM Sling Model can be adapted from. When the Sling Model is requested for a specific adaptable type, AEM searches for a Sling Model that matches that type

The **@ChildResource** annotation is used to inject child resources of the current resource into the Sling Model.





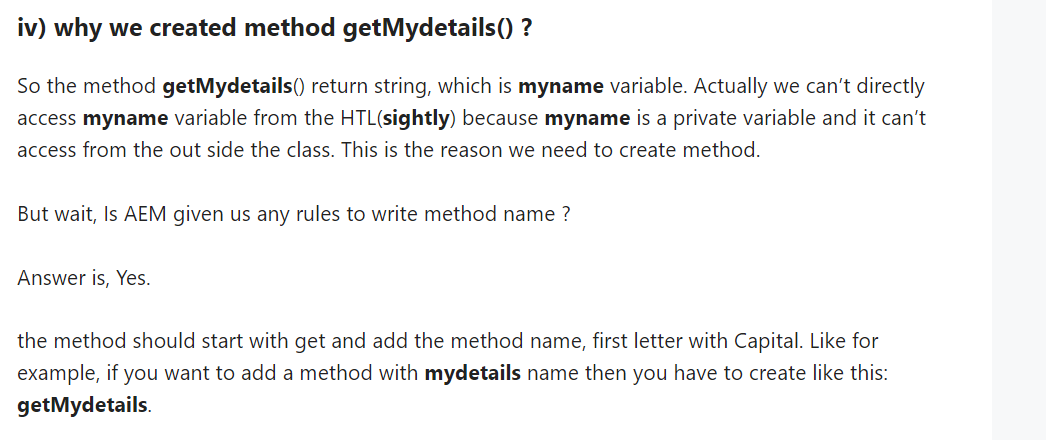


Image always be rendered without adapting to the device it is being viewed in. This means the image will suffer from styling constraints, might appear big or small to a user, ultimately disrupting website performance and user experiences.

when you update the image, any user that has viewed the image before will see the old version only and not the new one.

Therefore, the only solution is to code all values in the Adobe Experience Manager sling models to prevent bugs and prevent inconsistency in how links work across different components

**Why Run Modes?**

* Uniquely identify an environment and instances
* Unique configurations based on environment
* OSGI Component Creation for a specific environment
* Bundle Creation for a specific environment

**There are two types of run modes:**

* Primary Run Mode
* Secondary Run  Mode

. **Primary Run Modes are:**

* **Author**: This instance is used for the complete development and authoring purpose.
* **Publish**: This is the actual environment which can be accessed by end users.

**Combination of primary run modes also happen:**

* Author+samplecontent
* Author+nosamplecontent
* Publish+samplecontent
* Publish+nosamplecontent

**Secondary Run Modes are:**

* Dev Server
* QA Server
* UAT Server
* Prod Server

We can create our own customized run modes on the basis of location( i.e., us,uk), language or any other basis.

### Difference between Design Dialog and Edit Dialog?

**Ans:**

* design dialog and it is stored under /etc/designs folder.
* information entered in the dialog will be stored under /content folder.

### What is the purpose of clientlibs?

**Ans:**It is used for adding site-specific js and CSS files to the page

### What are Dependency/Dependencies in the client library?

**Ans:**It is a list of other client library categories on which this library folder depends. For example, given two cq: ClientLibraryFolder nodes F and G, if a file in F requires another file in G to function properly, then at least one of the categories of G should be among the dependencies of F.

### What is the difference between parsys and iparsys?

**Ans:**

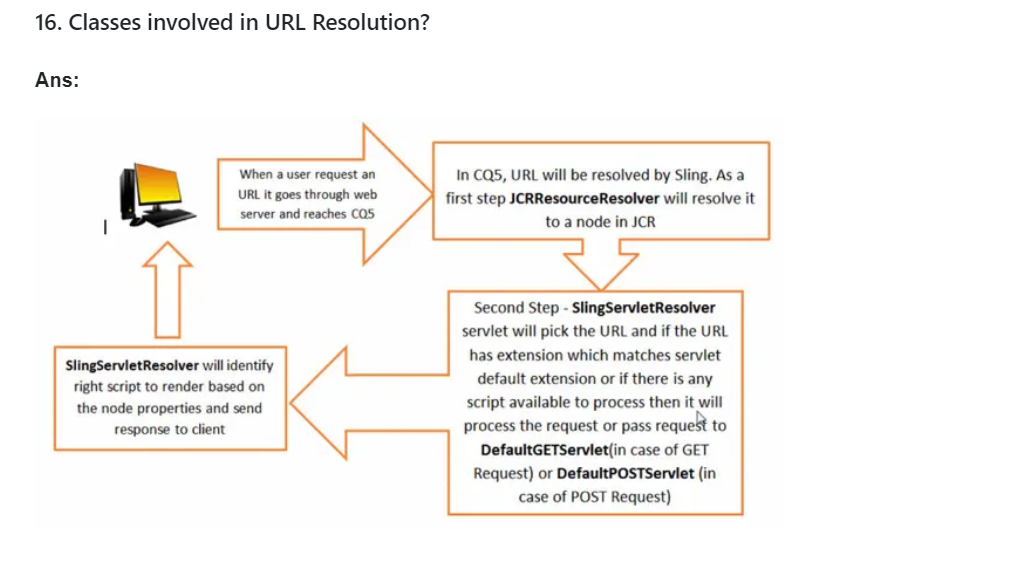
**parsys:** It is called the “Paragraph System” component in which you can add other components at the page level.

**iparsys:** It is an inherited paragraph system, which inherits the parent page “paragraph system”.

### What is sightly in AEM?

 In AEM 6.0, Sightly was introduced as an HTML templating language. It is intended to replace JSP files and is the primary HTML templating solution. The word "sightly" implies "pleasant to the eye," and its goal is to make your markup look nice and tidy. The following benefits of using Sightly:

* **Lightweight** — There are no dependencies, thus it is quick and lean.



### What is sightly in AEM?

 Sightly was introduced as an HTML templating language. It is intended to replace JSP files and is the primary HTML templating solution. The word "sightly" implies "pleasant to the eye," and its goal is to make your markup look nice and tidy. The following benefits of using Sightly:

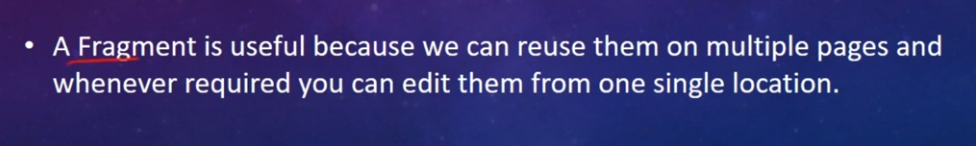
* **Lightweight** — There are no dependencies, thus it is quick and lean.

Experience Fragment:

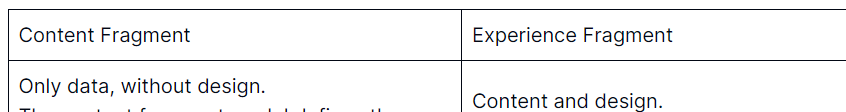
-- Experience Fragment is a group of components

--experience fragment allows content authors to create reuable content fragment that will be shared across all pages

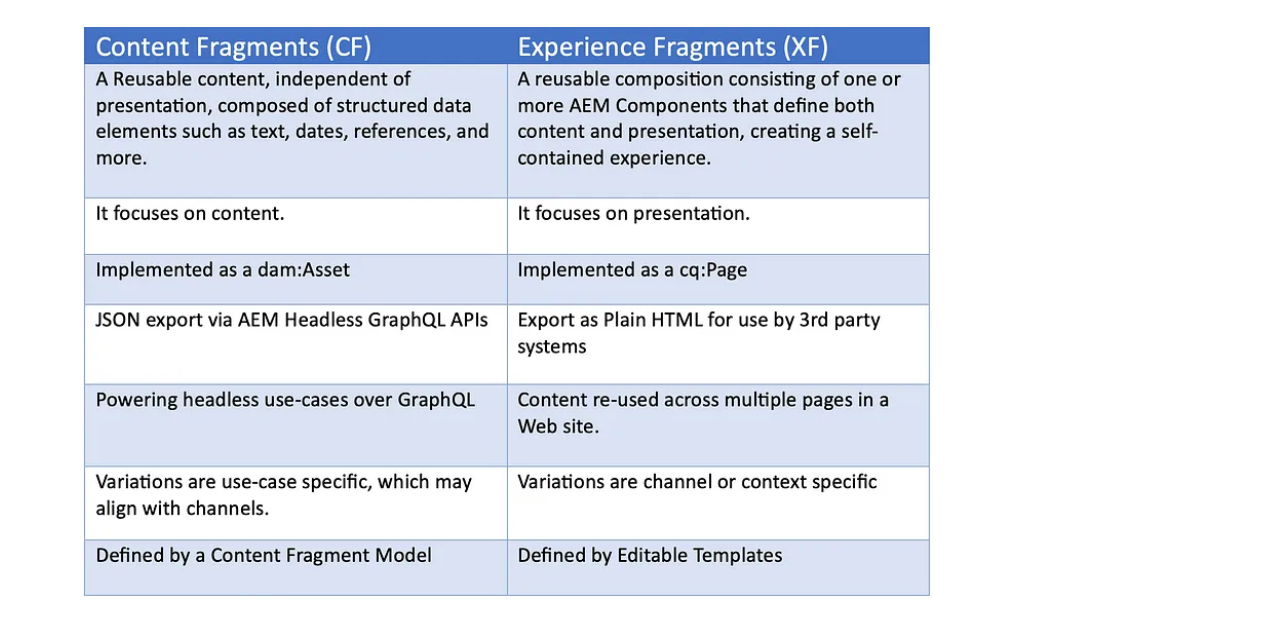
Content Fragment:



Content Fragments VS Experience Fragments :



**Layout Flexibility:** Experience Fragments provide more control over layout and design compared to Content Fragments. Authors can define the structure, positioning, and styling of components within the fragment



OSGI service:

//For creating osgi service we create one interface ,one model class,one service class.//

Why we need model class here ?

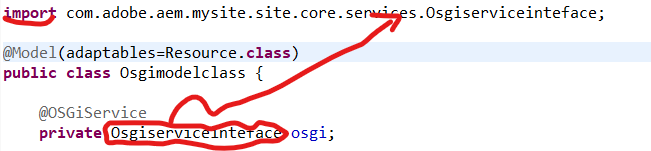
To call backend code to frontend we use the model class.

Why we need Interface here ?

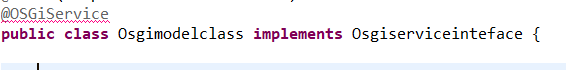
we need to call the osgi services to model class for that we store the inteface in the model class insted of implementing the interface , if we implement in the model class we can not insert the osgi services into the interface .

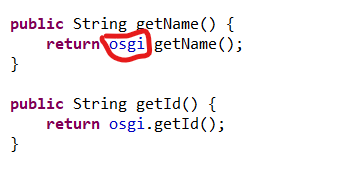


If you want to import osgi service directly insted of taking the interface but one problem occur , here actually we can not import the class but we can import the interface



If we implement the interface in model class we can not insert the osgi services,outside the method java not allowed to use the @OSGiService annotation,one more problem is there we can not use the reference name in the return type this is a another problem





Why we need Service class here ?

for creating the osgi component we need the osgi service class , it creates osgi service in system console we can reuse this service wherever we want

@Component: This is an annotation used in OSGi environments to declare a class as an OSGi component.

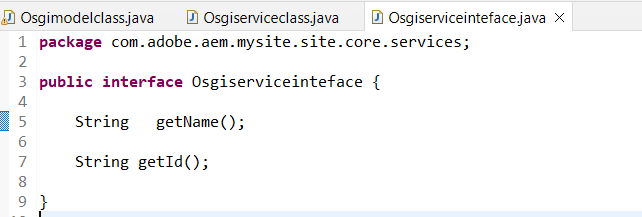
We can create OSGI service with the help of **@Component** annotation from org.osgi.service.component.annotations package and register it as an OSGI service.

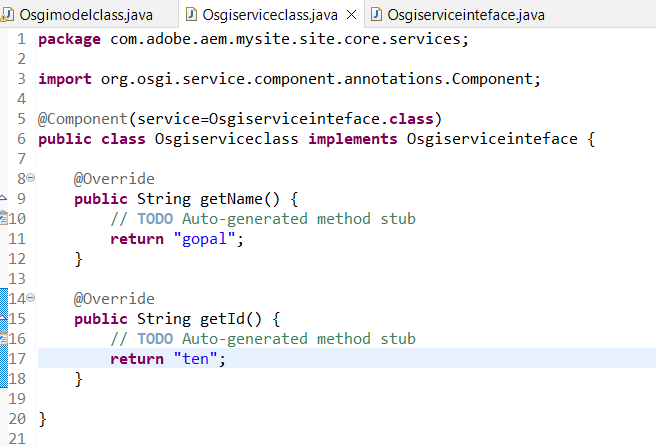
Services can be consume using **@Reference** annotation in Servlet and other services. User **@OSGiService** annotation n to inject service in sling model.

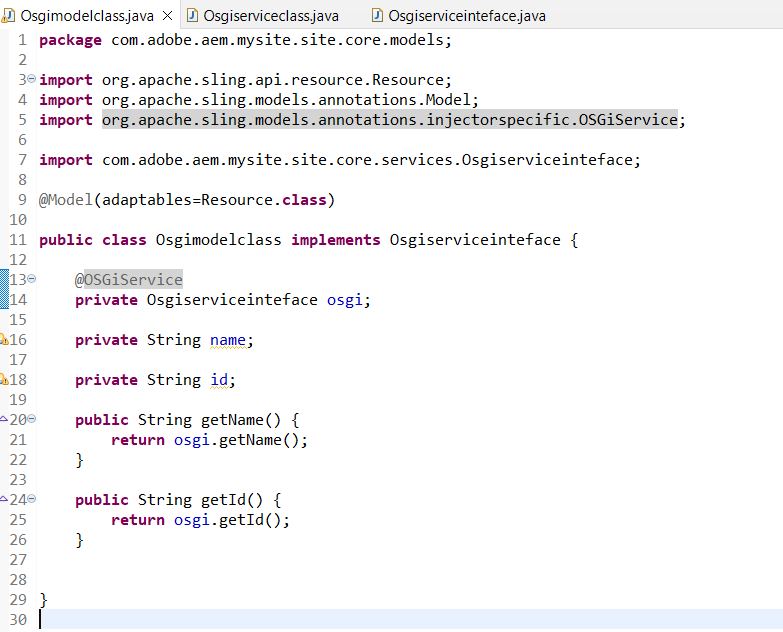
Certainly! In OSGi, when a component is defined with the immediate = true attribute within the @Component annotation, it means that this specific component will be activated and instantiated as soon as the OSGi framework starts, disregarding the default behavior of waiting for dependencies to be fulfilled.

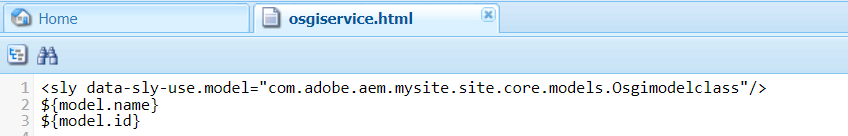
By setting immediate = true, the component doesn’t wait for its required dependencies to become available before being activated. Instead, it's immediately instantiated once the OSGi framework initializes, assuming all its dependencies are resolved during this initialization phase.

This attribute is particularly useful for components that are critical or standalone, meaning they don't rely heavily on other services or components to function properly. It ensures that such components are ready and available right from the start of the application, potentially reducing any delay or waiting time for their activation until all dependencies are fulfilled.









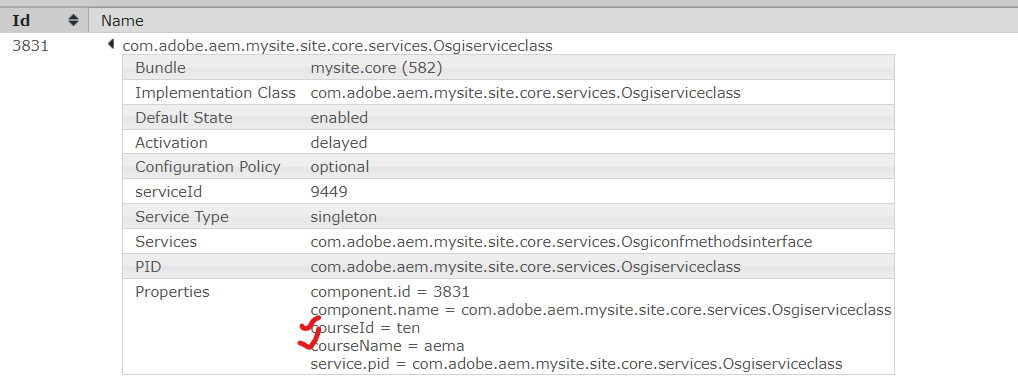
OSGI CONFIGURATION:

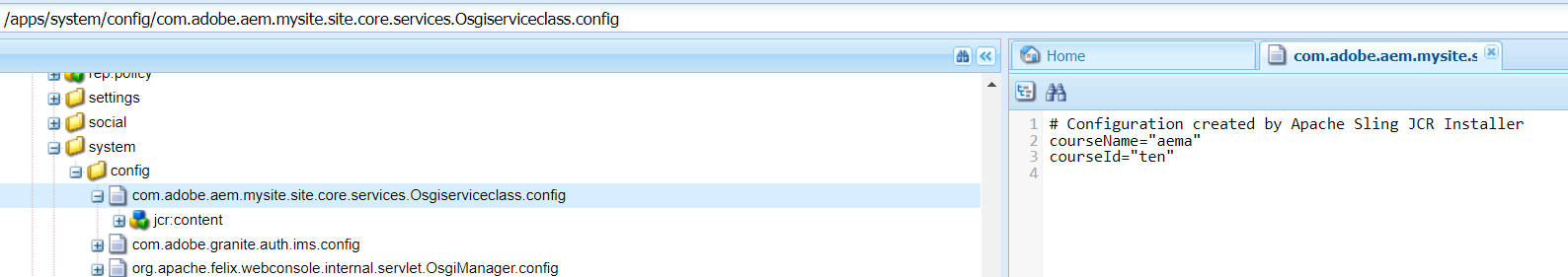
What is the use of osgi configuration rather than osgi service ?

Osgi configurations creates the bundle here we can give the input ,for suppose if we give the data in osgi service we have to run the java code to render the output on page ( by the way it not create the bundle ) but in osgi configuration we can set the data in bundle it will render the data on page without running the java code,this is the use of osgi configuration.

Why we need to give the @Designate annotation in osgiservice class rather than model class or other classes ?

If we give in osgiserviceclass it will add the data in osgicomponent because of this we can give the input to the bundle in crxde directly



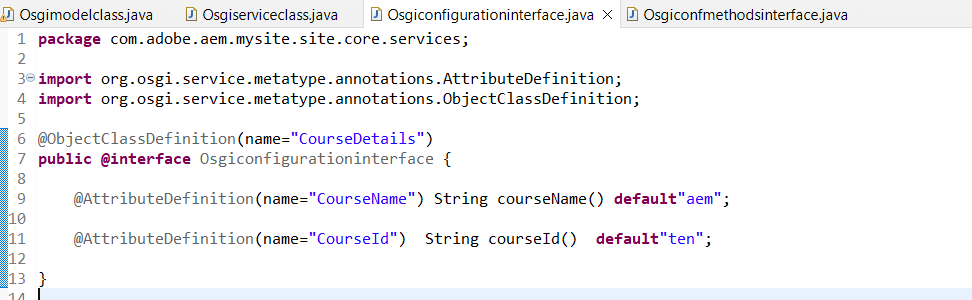


@Designate:add configurations to OSGI Services with the help of @Designate annotation. @Designate is must to have ocd(object class definition) property to declare service name as mentioned in below syntax.

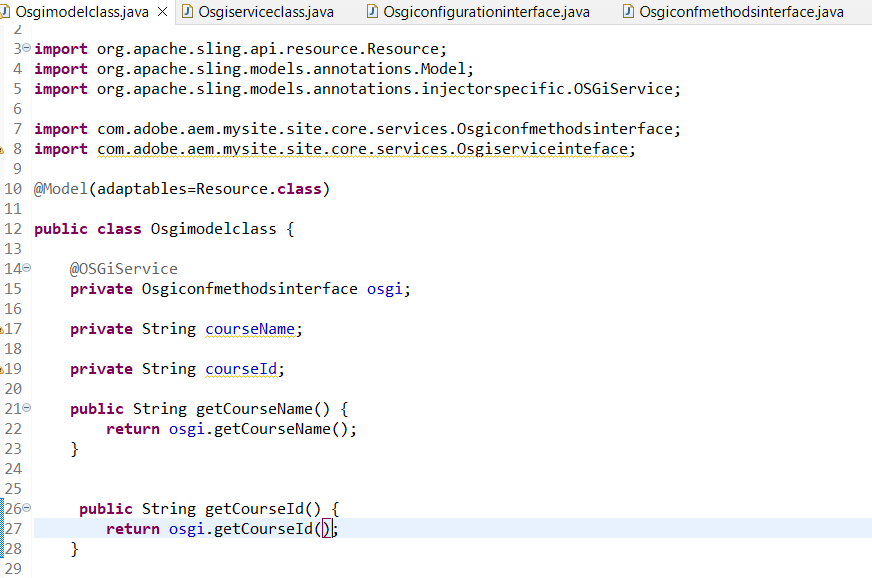
@Designate(ocd = PracticeServiceOCDConfiguration.class)

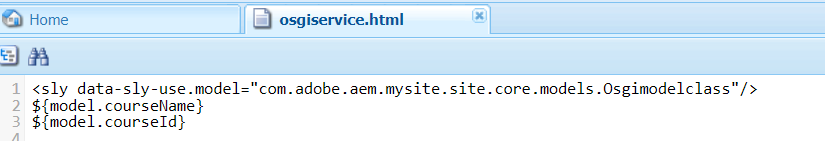
@interface :Generally in normal Interface we will give the return type(data) in java itself only but in the osgi configuration we give the data in the osgi bundle ,it is outside of the data that is why we are using @Interface

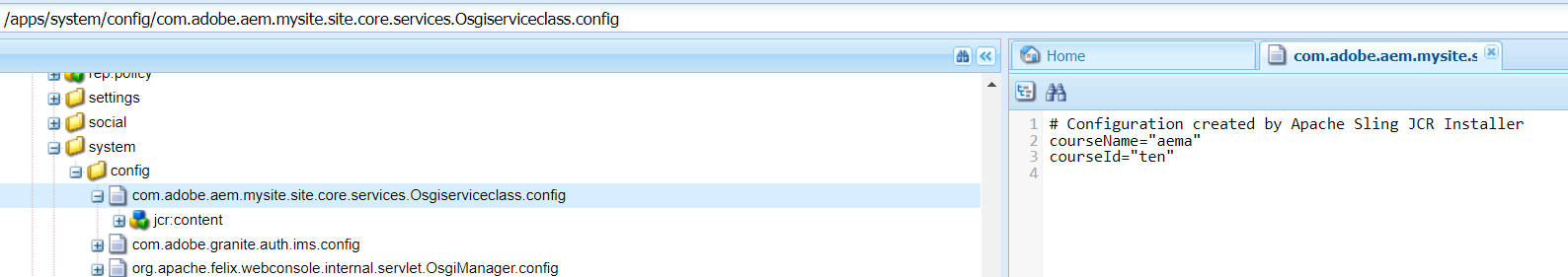
@PostConstruct: This annotation can be used to run the logic once all the field level injections are done.

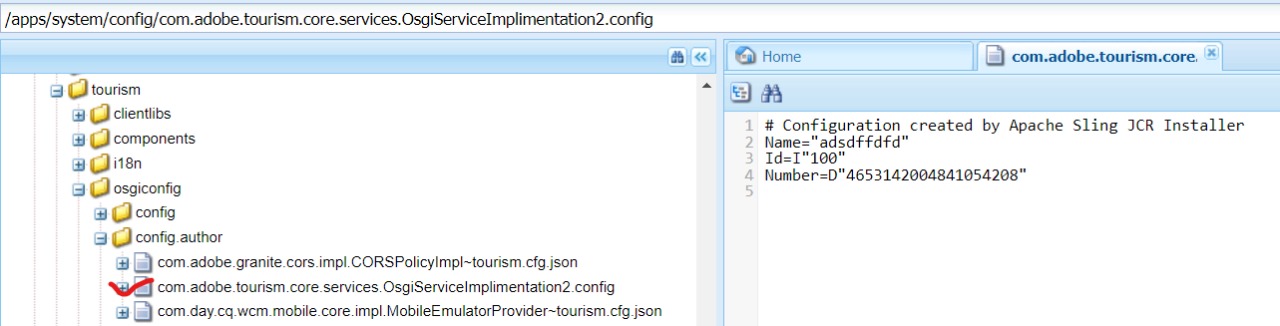












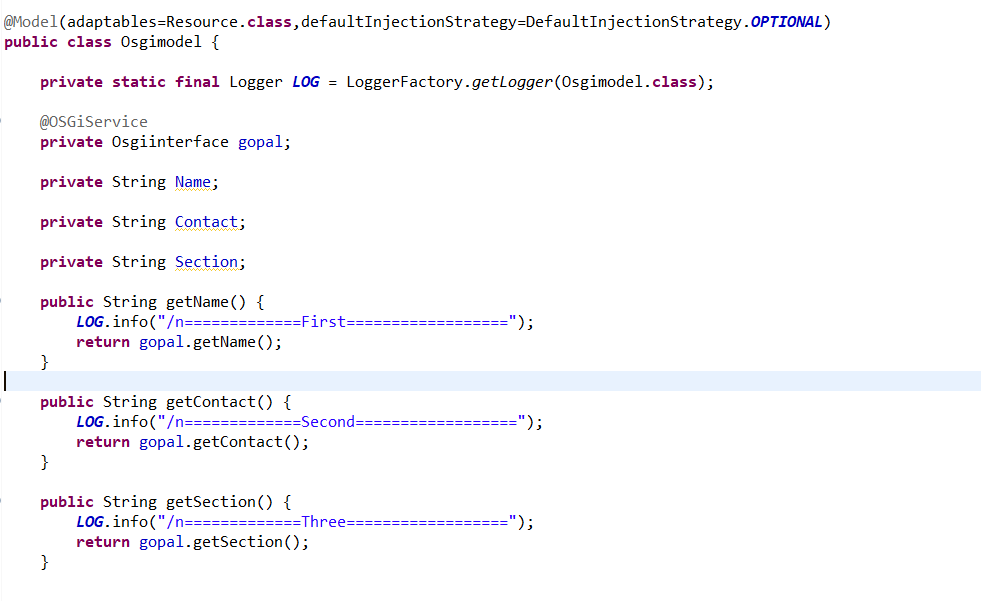
HOW TO CREATE A LOGGERS

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**private** **static** **final** Logger ***LOG*** = LoggerFactory.*getLogger*(Osgimodel.**class**);

***LOG***.info("/n=============First==================");







<http://localhost:4502/system/console/slinglog>

>run the javacore now

OSGI BUNDLE:

As AEM applications grow in complexity, handling third-party OSGi bundles becomes critical for maintaining application stability and flexibility. Without proper dependency management, conflicts in versions and tightly coupled components can lead to unpredictable behavior and hinder upgrades.

To address these challenges, we rely on the OSGi Bundle Repository (OBR) to effectively manage third-party dependencies in AEM projects.

Using OSGi Bundle Repository (OBR) for Dependency Management: The OSGi Bundle Repository (OBR) serves as a centralized repository for OSGi bundles, providing a convenient way to manage and resolve dependencies. It allows developers to declare external bundles as dependencies, making them easily accessible across AEM applications.

**Integrating a Payment Gateway:** Consider a custom AEM e-commerce application that needs to integrate a third-party payment gateway for secure online transactions. Without OBR, directly embedding the payment gateway bundle could lead to version conflicts with other bundles or hinder upgrades.

By using OBR, the AEM project can declare the payment gateway bundle as a dependency, ensuring that it is independently managed and upgradable.

**Benefits of Independent Upgrades with OBR:** One of the significant advantages of using OBR for managing third-party dependencies is the ability to upgrade specific bundles independently. This reduces the risk of regressions and allows developers to stay up-to-date with the latest features and bug fixes without affecting other parts of the application

-------------------------------------------

# Practical Use Case Of Multifield :

With the use of multifield we can add n number of items in the component , where the author can add or remove items on the page.

🡪 Multifield can contain single or multiple fields

🡪Multifield is used for repeat the instances of the same fields

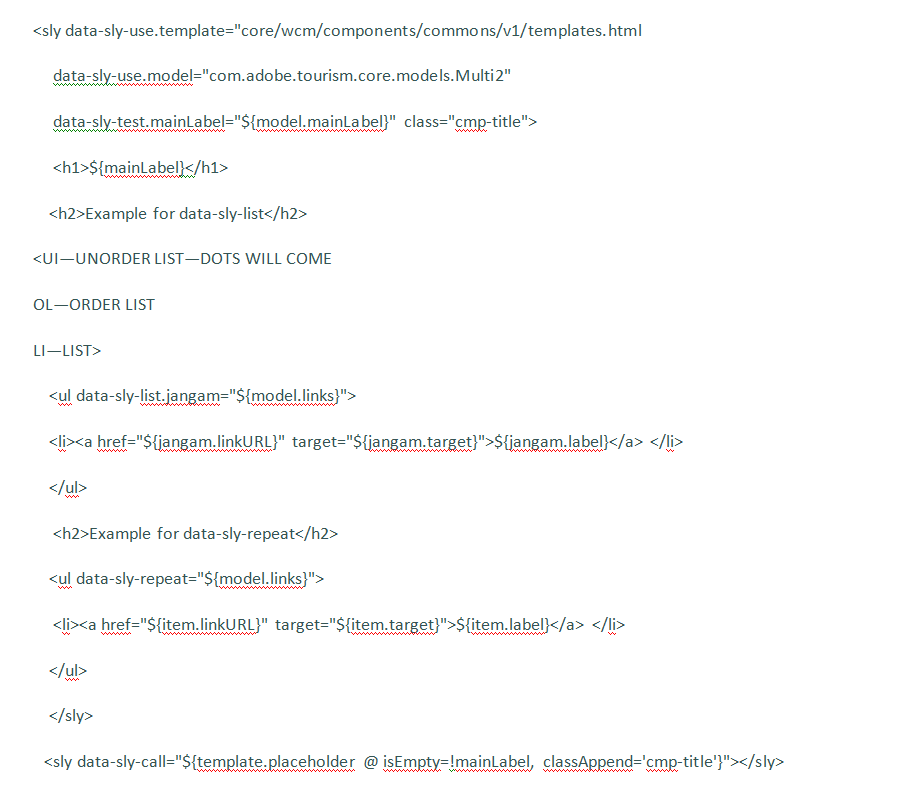
MULTIFIELD CODE:



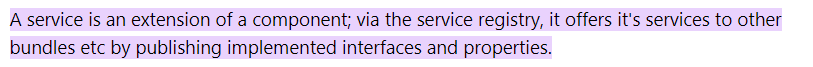
 

### isNoneBlank()

**isNoneBlank()** is a static method of the StringUtils class that accepts a list of strings and checks that there are no blank strings in the list. The method returns false if at least one of the strings in the given list of strings is considered to be blank.



------------------------------------------OSGI Services in AEM



OSGI (Open Service Gateway Initiative) is a Java framework for modularizing and managing the life cycle of applications and services.

In AEM, OSGI services are used to provide modularity and flexibility to the platform. They allow developers to create small, independent units of functionality that can be plugged into AEM and used in a variety of different ways.

One of the key benefits of using OSGI services in AEM is that they can be easily extended and customized to meet specific business requirements. This means that organizations can tailor their AEM implementation to fit their unique needs, without having to build everything from scratch.

Another advantage of OSGI services is that they allow for easier collaboration and code sharing between developers. By breaking down complex functionality into smaller, reusable pieces, developers can work on different parts of the codebase without stepping on each other's toes.

In summary, OSGI services are an essential part of AEM's architecture and provide a lot of value to developers. If you're working with AEM, I highly recommend learning more about OSGI and how it can be used to create modular and flexible solutions.

An OSGi service in AEM is a component that follows the OSGi framework and can be deployed in AEM as a bundle. To write an OSGi service in AEM, you need to create a Java class that implements the **org.osgi.service.component.Component** interface. This interface defines the **activate()** and **deactivate()** methods, which are called when the service is activated and deactivated, respectively.

What is OSGi and its purpose in simple words?

OSGi stands for "Open Services Gateway initiative." It's a framework for developing and deploying modular Java applications. The main purpose of OSGi is to make it easy to create, manage, and use Java components (called "bundles") that can be easily installed, started, stopped, updated, or uninstalled, without affecting the rest of the system.

1. **JCR (Java Content Repository) API:**
   * AEM is built on top of the JCR, and you can interact with the repository using the JCR API. This allows you to perform operations like creating, updating, and deleting nodes and properties in the content repository programmatically.
2. **Sling API:**
   * Sling is a web framework for building web applications on top of the JCR. It provides a set of APIs that allow you to work with resources in a RESTful way. You can use the Sling API to manipulate content, query the repository, and handle requests and responses

**1. Java Content Repository (JCR) APIs**

**Scenario**: You need to perform low-level content manipulation and interact directly with the repository.

Example: Updating Node Properties

import javax.jcr.Node;  
  
import javax.jcr.Session;  
  
  
// Get the current session  
  
Session session = resourceResolver.adaptTo(Session.class);  
  
// Access a node by path  
  
Node node = session.getNode(“/content/mywebsite/article”);  
  
// Update a property  
  
node.setProperty(“title”, “Updated Title”);  
  
// Save changes  
  
session.save();

**Sling APIs:**

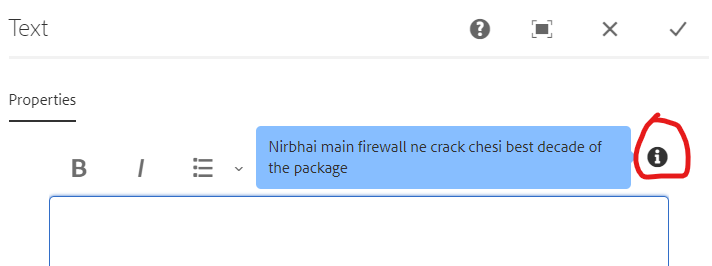
**Scenario:** You want to build RESTful services and handle requests efficiently.

Example: Creating a Servlet

import org.apache.sling.api.SlingHttpServletRequest;  
  
import org.apache.sling.api.SlingHttpServletResponse;  
  
import org.apache.sling.api.servlets.SlingAllMethodsServlet;  
  
// Extend SlingAllMethodsServlet  
  
public class MyServlet extends SlingAllMethodsServlet {  
   
 @Override  
 protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) {  
 // Handle GET request  
 }  
  
  
 @Override  
 protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response) {  
 // Handle POST request  
 }









ResourceResolverFactory is an important interface in Adobe Experience Manager (AEM) used to obtain a ResourceResolver instance. A ResourceResolver is responsible for providing access to the JCR (Java Content Repository) and resolving resources within the AEM repository.

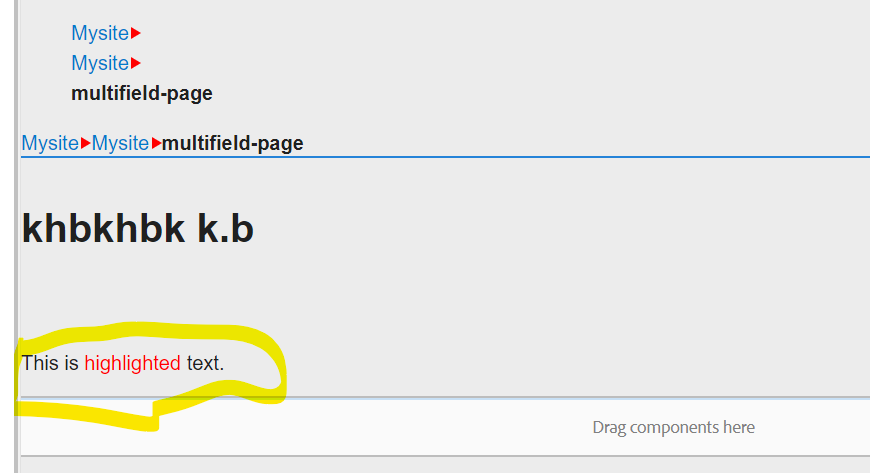
Here's why and where ResourceResolverFactory is commonly used:

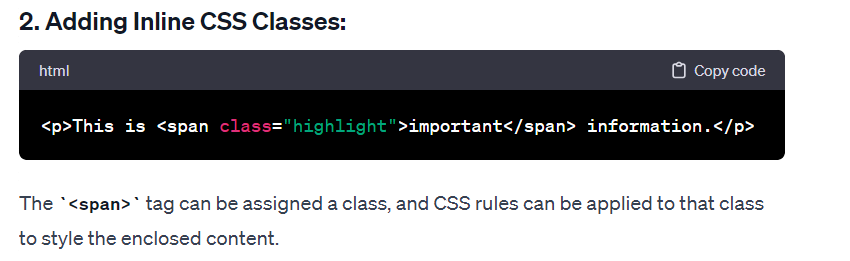
1. **Accessing the JCR:**It allows developers to access the JCR content within AEM programmatically. With a ResourceResolver, you can perform various operations like reading, writing, querying, and manipulating content in the repository.
2. **Handling Resource Resolution:**A ResourceResolver obtained from the ResourceResolverFactory is used to resolve resources by their path within the repository. This enables fetching resources like pages, components, assets, etc., and performing operations on them.
3. **Permissions and Security:**ResourceResolverFactory assists in obtaining a ResourceResolver with specific permissions based on the user's credentials. Different users may have different levels of access to AEM resources, and the ResourceResolver obtained can respect these permissions.

SPAN:

The **<span>** tag in HTML is a generic inline container that is used to group and apply styles to inline elements. It doesn't add any visual formatting on its own, but it can be styled using CSS. It is commonly used when you want to apply styles or scripting to a specific portion of text or content within a larger block of text.

<p>This is <span style="color: red;">highlighted</span> text.</p>

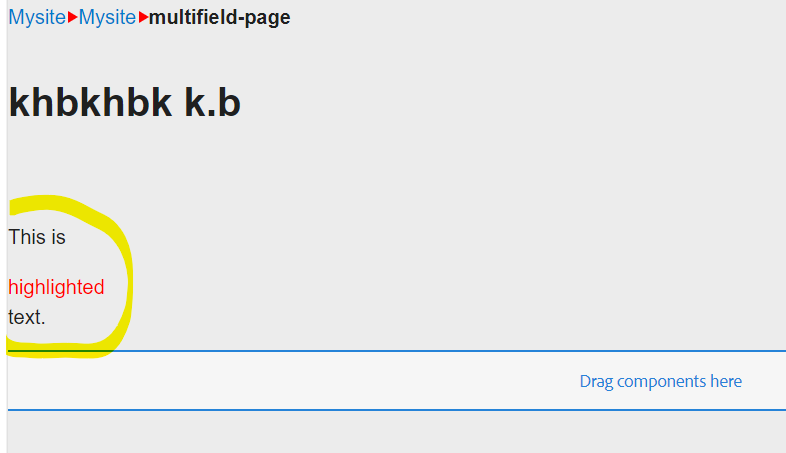




In summary, the **<span>** tag is a versatile and lightweight inline container used for various purposes, such as styling, scripting, and grouping inline content within HTML documents. It is commonly employed in situations where more specific inline elements, like **<div>** or **<p>**, are not suitable.Top of Form

<p>This is <div style="color: red;">highlighted</div> text.</p>

OUTPUT:





# AEM Handler and listener

Event handler is responsible for Sling level events and Event listener is responsible for JCR level events.

Both handler and lister is responsible for executing an event.

Imagine you want to perform a specific action whenever a new page is created or an existing page is modified in AEM. In such cases, you can use an event listener to respond to the content changes.

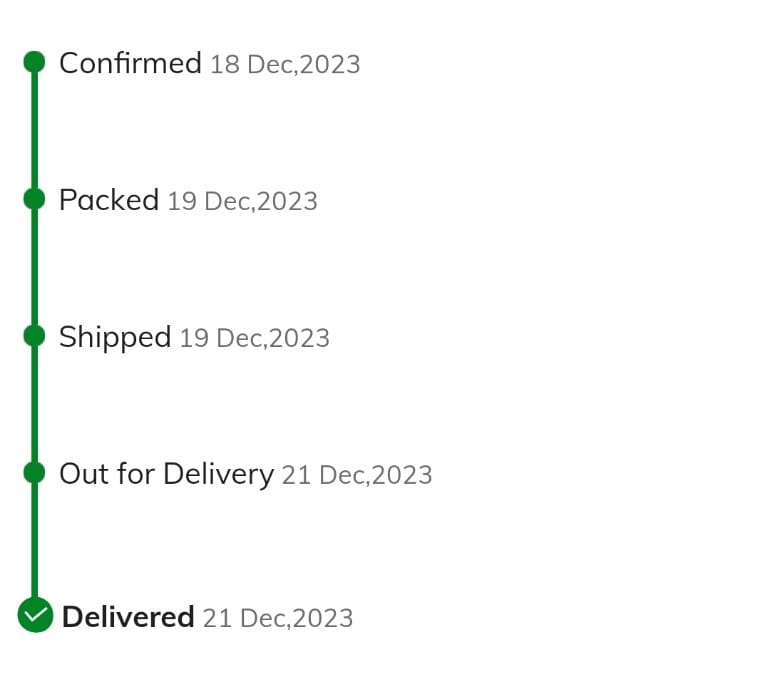


“Order service🡪 **OrderStatusChangeEvent🡪 OrderStatusChangeListener(event handler)”**

**Scenario: E-Commerce Order Status Notifications**

1. **Order Service:**
   * The application has an **OrderService** responsible for managing orders.
   * Whenever there is a change in the order status (e.g., from "Processing" to "Shipped"), the **OrderService** updates the status of the order.
2. **Order Status Change Event:**
   * To notify other parts of the system about the order status change, the **OrderService** triggers an event, let's call it **OrderStatusChangeEvent**.
   * This event carries information about the specific order that underwent a status change.
3. **Event Handler:**
   * There is an event handler, known as the **OrderStatusChangeListener**, configured to listen for **OrderStatusChangeEvent** events.
   * When an order status change event is detected, the handler executes specific actions, such as sending a notification to the customer.
4. **Customer Notification:**
   * The notification to the customer can take various forms, such as an email, an in-app message, or a push notification.
   * The content of the notification includes relevant information about the order status change (e.g., "Your order has been shipped!").

🡪It doesn't need to know how notifications are sent; it only triggers an event.

Order service:

In the above example, the term "OrderService" represents a hypothetical service class that would be responsible for managing orders within the e-commerce platform. The **OrderService** is a conceptual class that encapsulates the logic related to orders, such as creating orders, updating order details, and managing the order lifecycle.

Here's a breakdown of what the **OrderService** might do:

1. **Order Creation:**
   * The **OrderService** would likely have methods for creating new orders when customers place an order on the e-commerce platform. This involves capturing details such as the items ordered, customer information, and initial order status.
2. **Order Status Update:**
   * One of the key functionalities of the **OrderService** in the given example is to update the status of an existing order. For instance, when an order is processed and ready to be shipped, the **OrderService** would be responsible for updating the order status accordingly.
3. **Event Triggering:**
   * In addition to updating the order status, the **OrderService** triggers an event when a status change occurs. This event, referred to as **OrderStatusChangeEvent** in the example, carries information about the specific order that underwent the status change.
4. **Communication with Event Listeners:**
   * The **OrderService** communicates with event listeners (such as **OrderStatusChangeListener**) to notify them of the order status change event. This enables other parts of the system to respond to the event, such as sending notifications to customers.

Servlet:

***package*** *com.adobe.aem.mysite.site.core.servlets;*

***import*** *java.io.IOException;*

***import*** *javax.servlet.Servlet;*

***import*** *javax.servlet.ServletException;*

***import*** *org.apache.sling.api.SlingHttpServletRequest;*

***import*** *org.apache.sling.api.SlingHttpServletResponse;*

***import*** *org.apache.sling.api.servlets.SlingSafeMethodsServlet;*

***import*** *org.osgi.service.component.annotations.Component;*

*@Component(service=Servlet.****class****,*

*property= {*

*"sling.servlet.resourceTypes=mysite/components/HeaderNavigation"*

*})*

***public******class*** *Practiseservlet* ***extends*** *SlingSafeMethodsServlet{*

***public******void*** *doGet(SlingHttpServletRequest req , SlingHttpServletResponse res)*

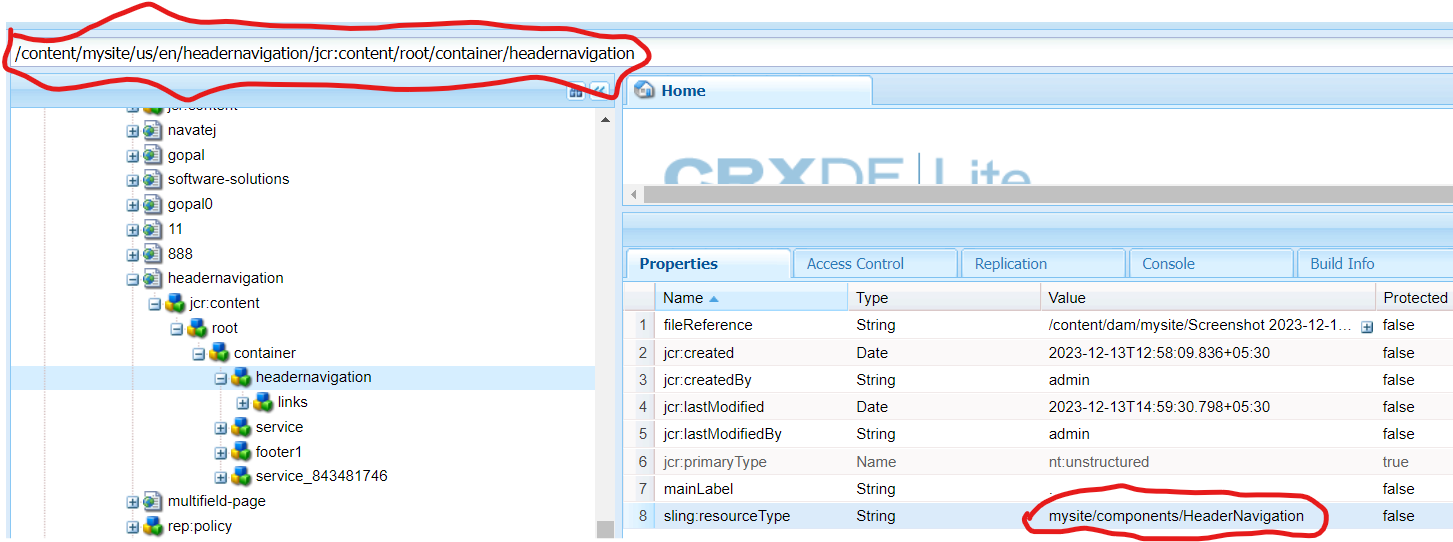
***throws*** *IOException{*

**

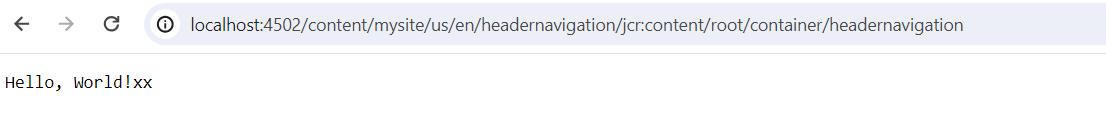
*// Write "Hello, World!" to the response*

*res.getWriter().write("Hello, World!xx");*

*}}*



OUTPUT:



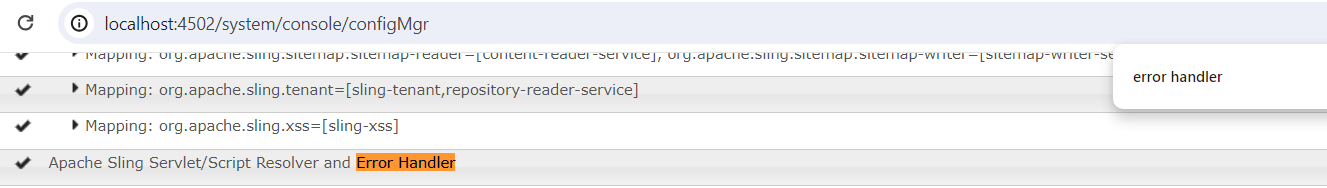
### SlingSafeMethodsServlet:

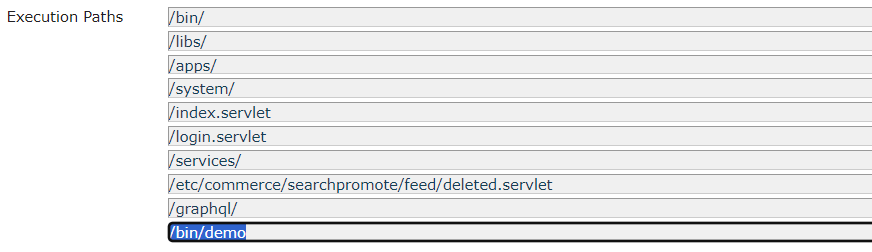
**SlingSafeMethodsServlet** is a base class in Apache Sling designed for servlets that handle HTTP methods considered "safe" or "read-only." These safe methods include HTTP GET, HEAD, and OPTIONS. In the context of web applications, "safe" methods are those that should not have the significance of taking an action other than retrieval.

Here's a clearer explanation:

* **Safe Methods:**
  + **GET:** Used for retrieving information from the server.
  + **HEAD:** Similar to GET but without the response body. It's often used to check for the existence of a resource.
  + **OPTIONS:** Used to describe the communication options for the target resource.

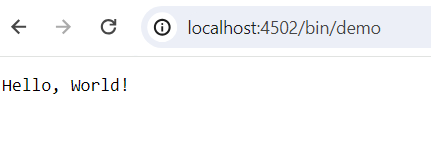
Servlet Registration:



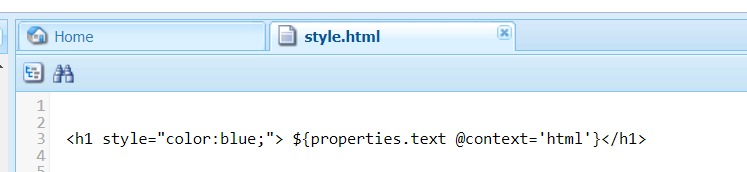


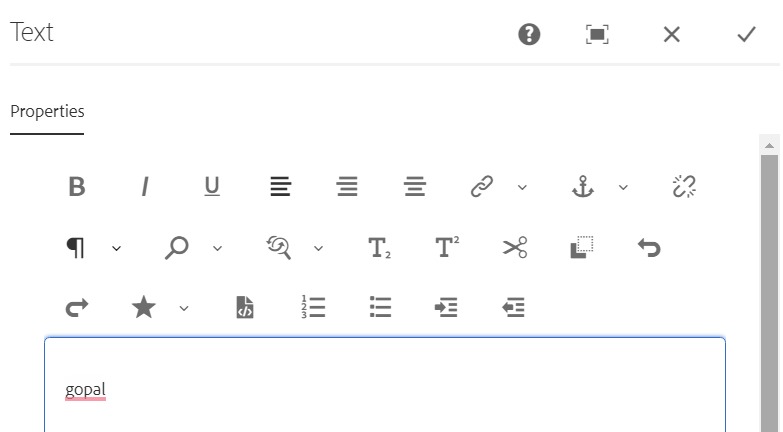


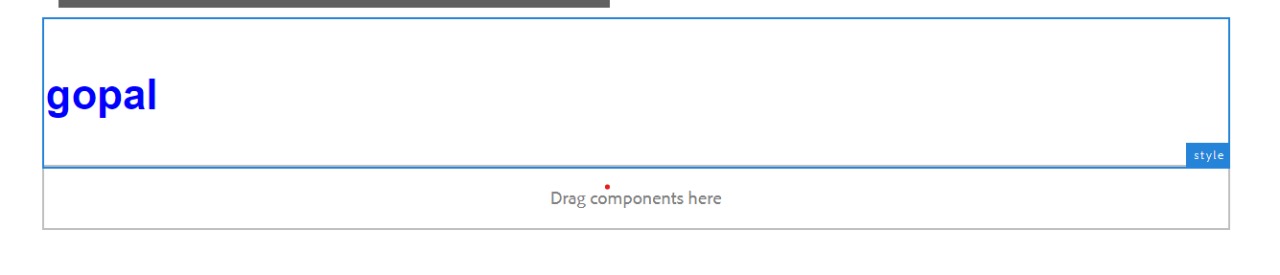
Output:



Style System without clientlibs







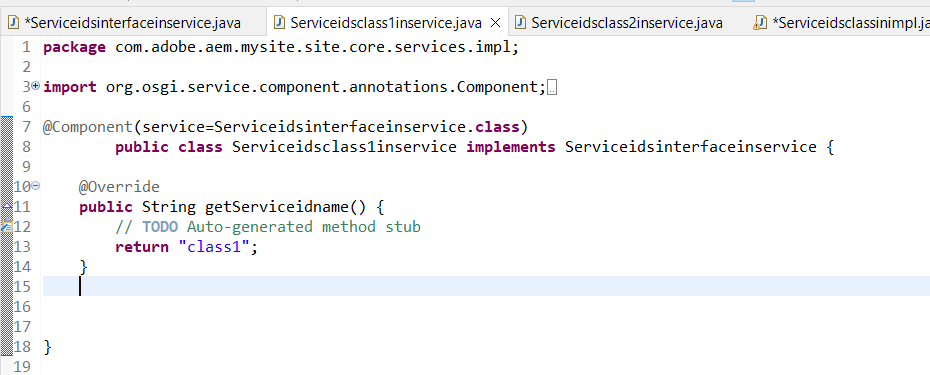
sling:resourceSuperType (vs) sling:resourceType

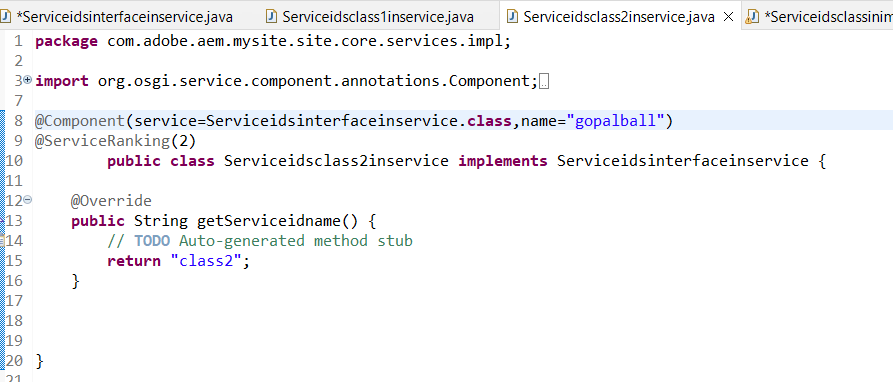
* The resource type is used by AEM to determine which script (JSP, HTL, etc.) should be used to render the component.
* For example, a **sling:resourceType** value might be something like **components/page** or **components/text**.
* If a component has a **sling:resourceSuperType** property, it means that it inherits from another component

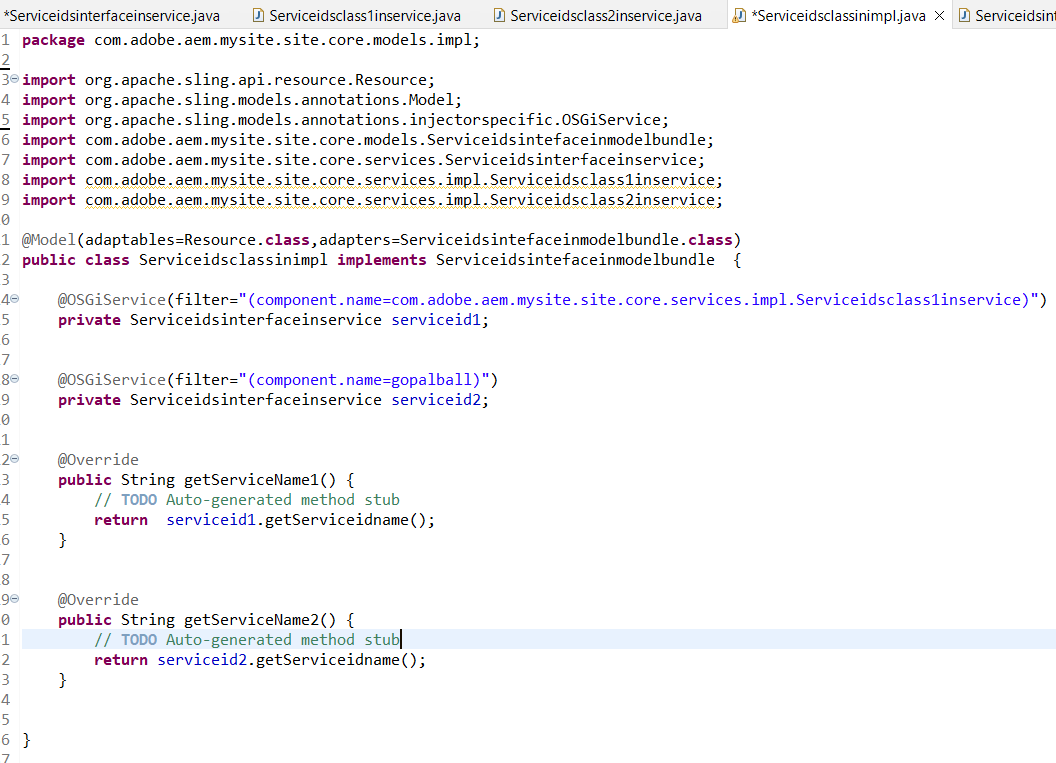
SERVICE ID:

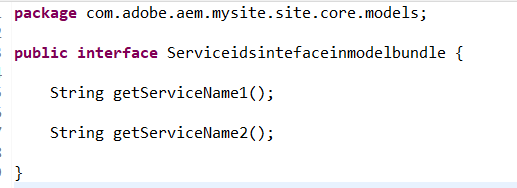
HERE THE REQUIREMENT IS IF WE HAVE A TWO SERVICE ID’S CREATED AT A TIME WHICH ONE LOAD FIRST,WHAT IS THE USE OF RANKING,WHAT IS THE USE OF FILTER.









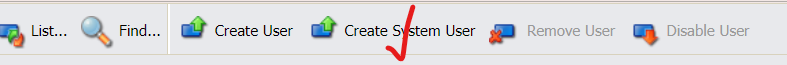


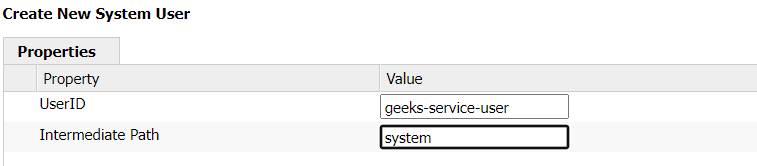
Creating System User

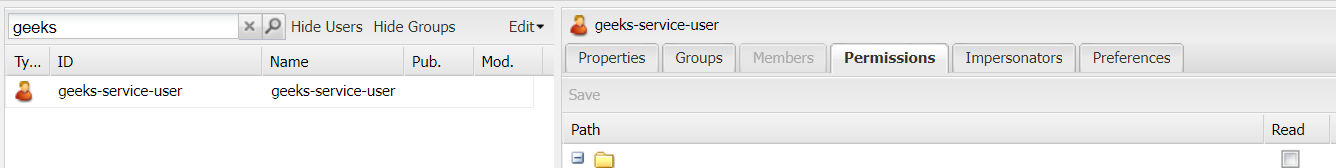
<http://localhost:4502/crx/explorer/index.jsp>

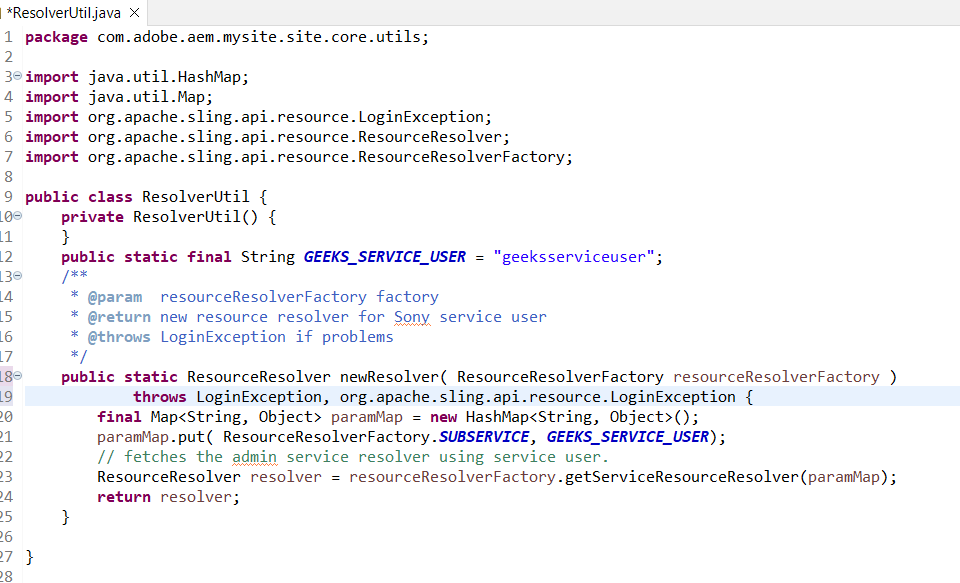
<http://localhost:4502/useradmin>

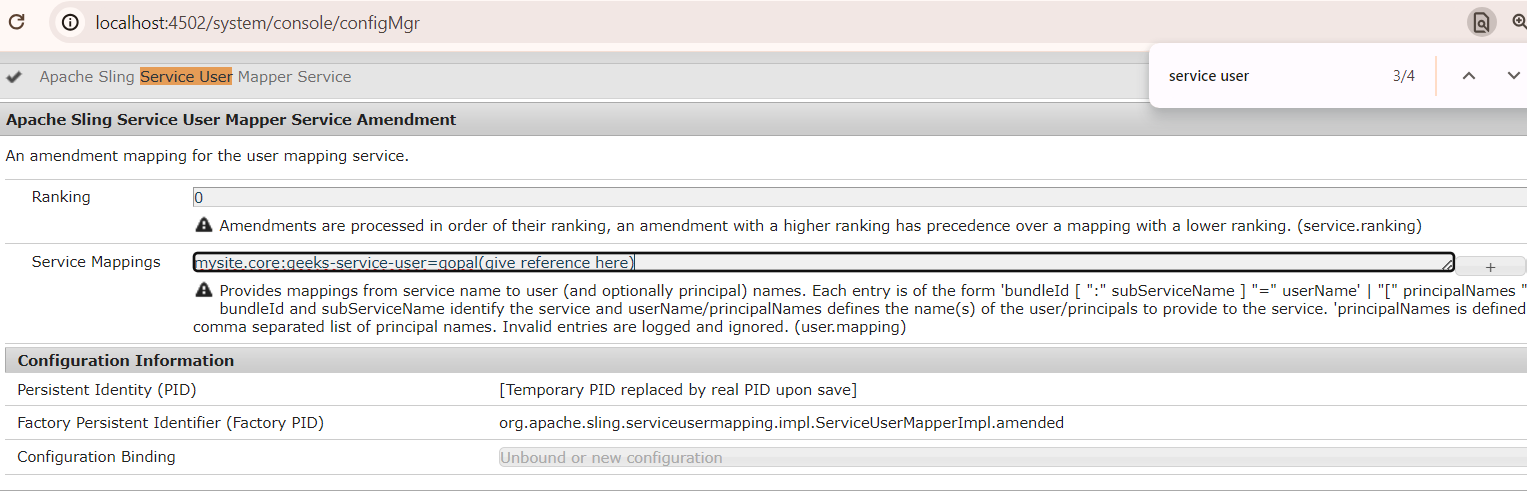






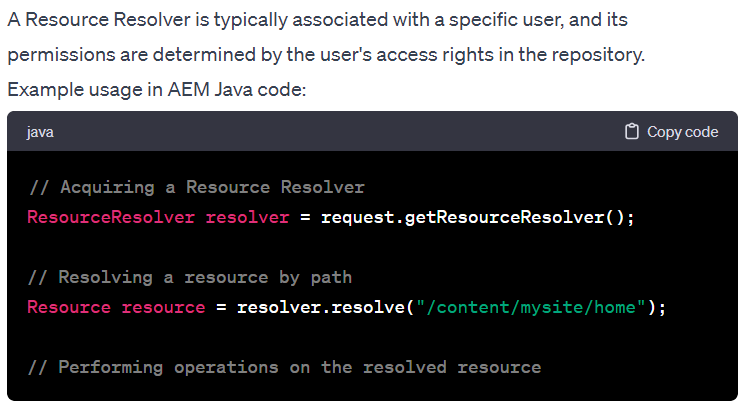


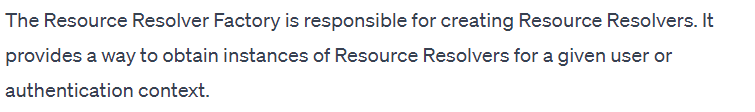




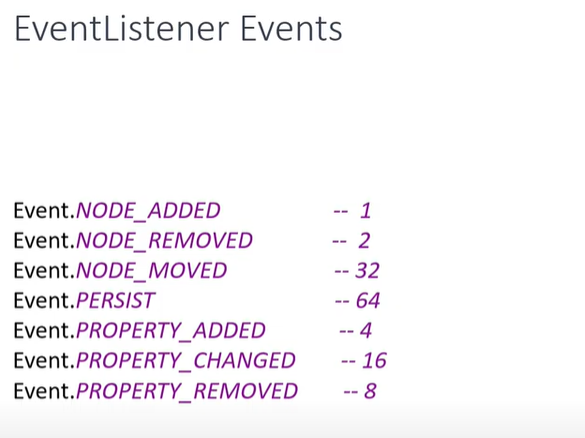
Resource Resolver and Resource Resolver Factory:



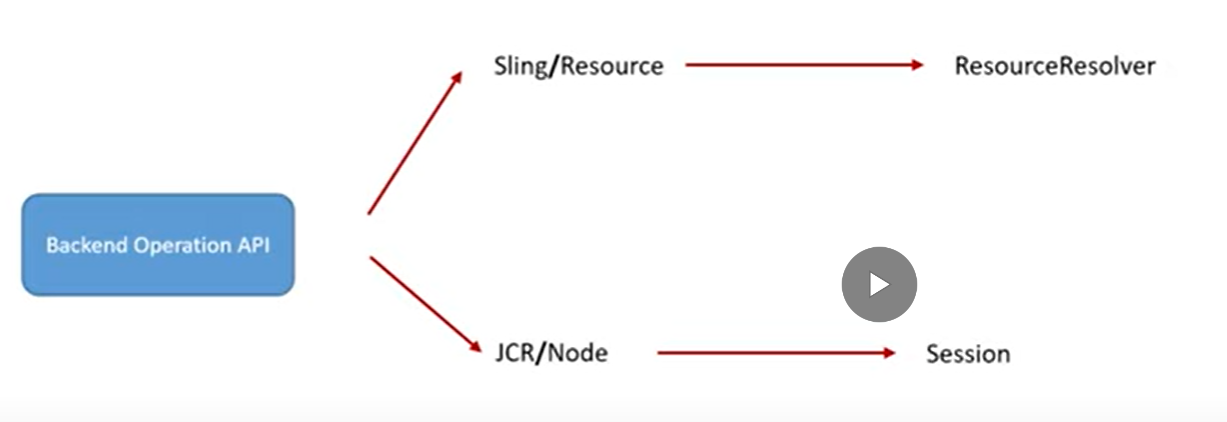


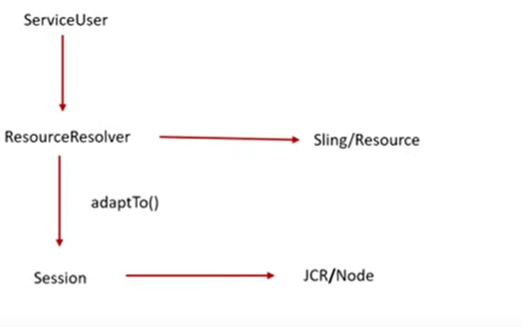


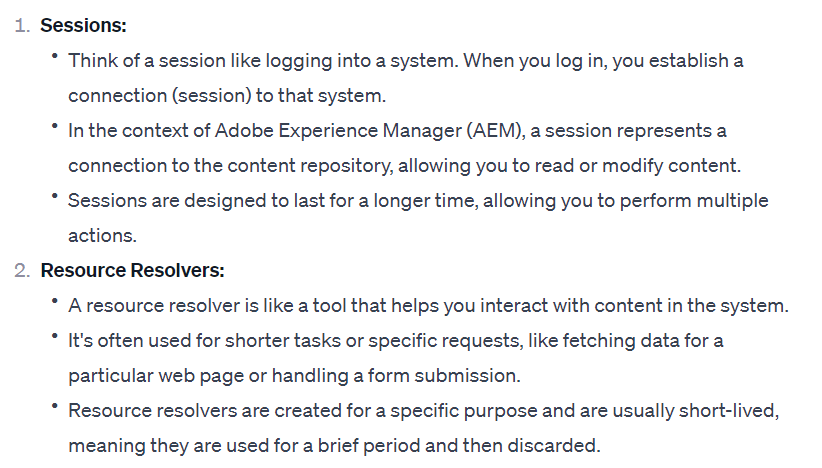
EVENT LISTNERS:

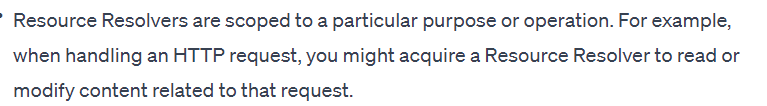


.









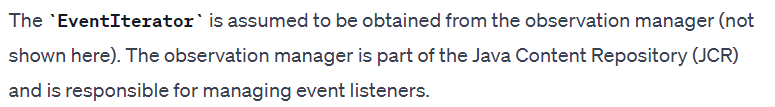
🡪From getworkspace() you will get getobservation() from getobservation you will add the addEventListener()

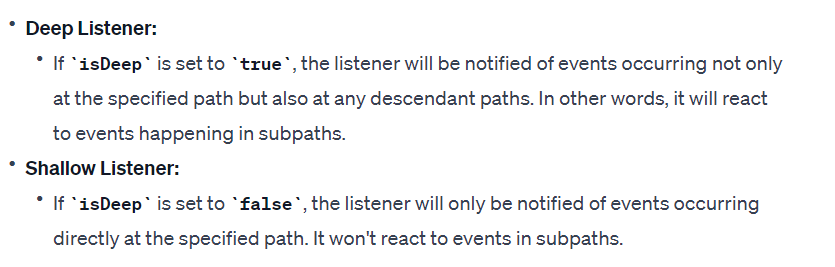
🡪Iterator contain event of all nodes,if one page is created multiple nodes created like cq:page , jcr:primarytype

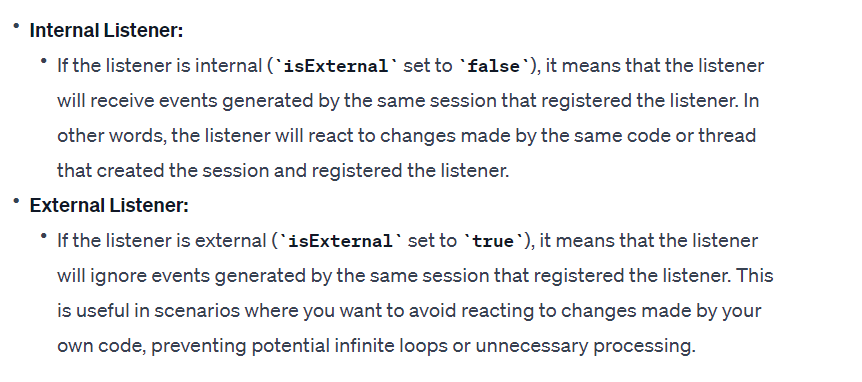
🡪**Session** represents a connection to the content repository (JCR - Java Content Repository), and it is used to perform read and write operations on the repository

🡪The **onEvent** method is called by the ‘’observation manager’’ when events of interest occur.

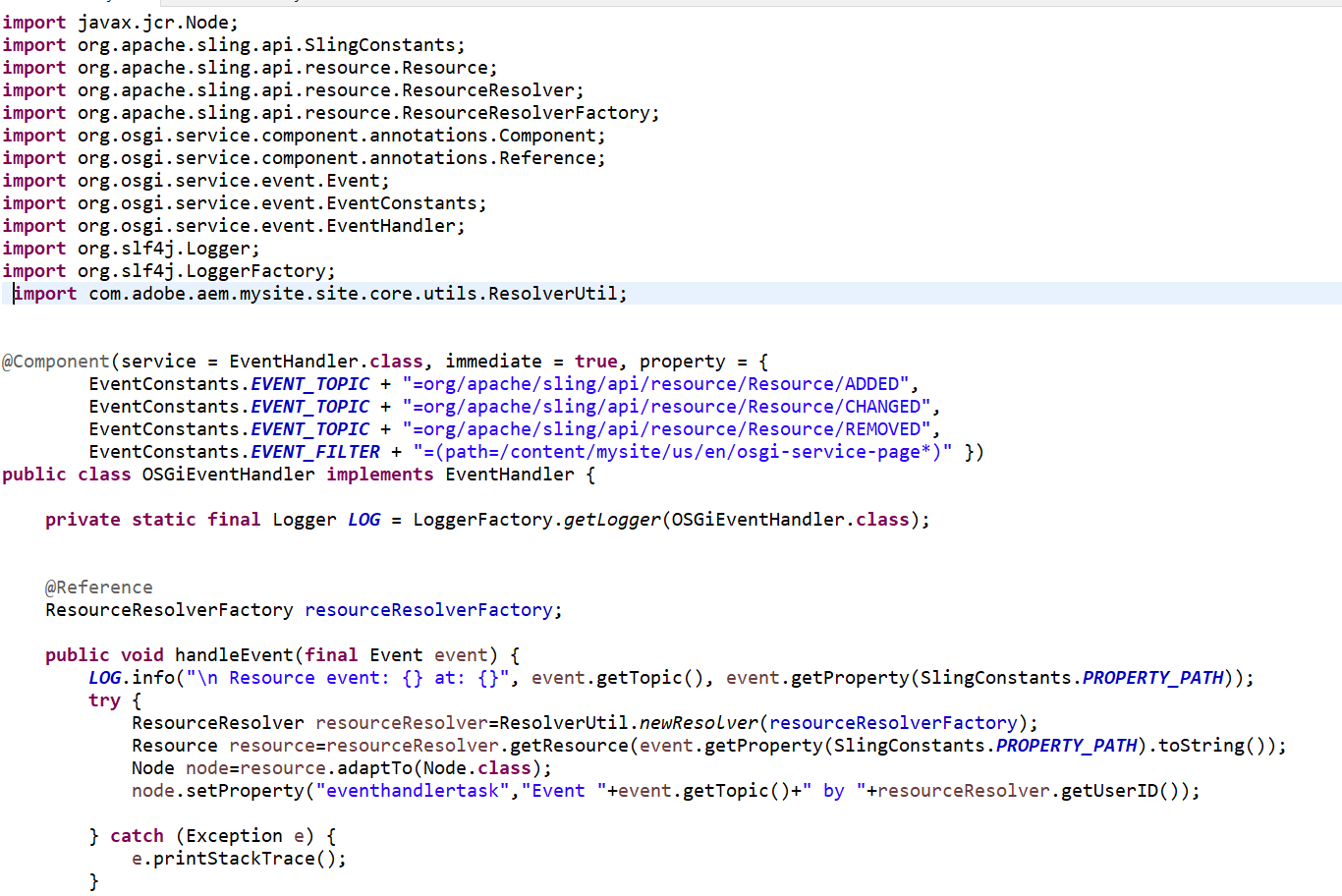






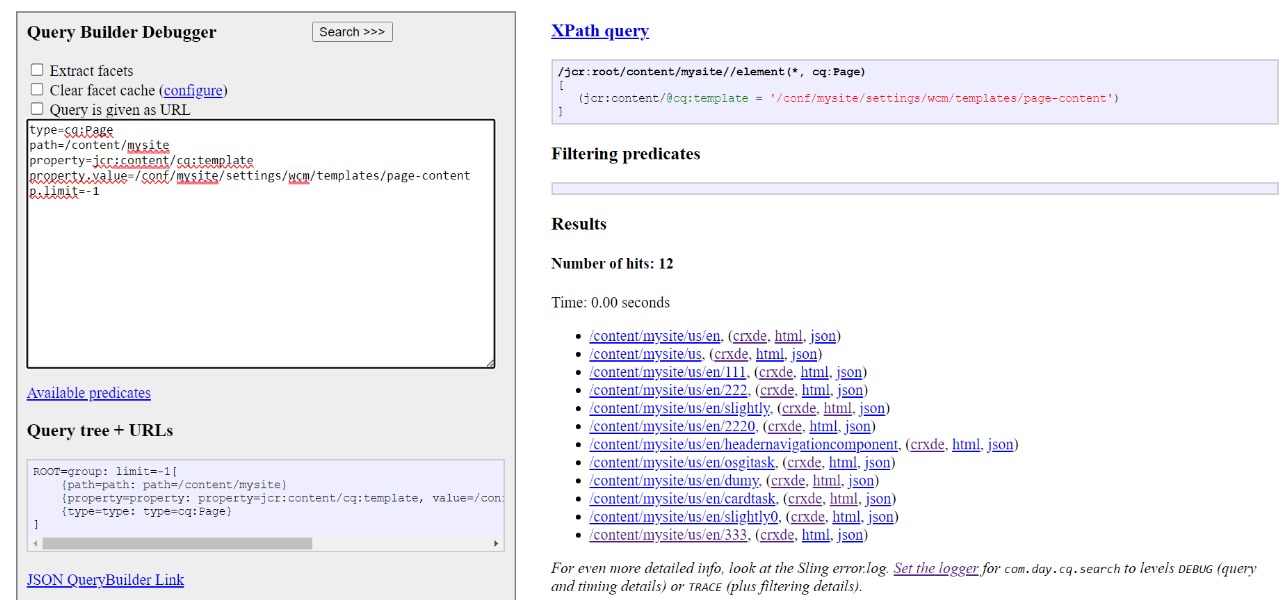


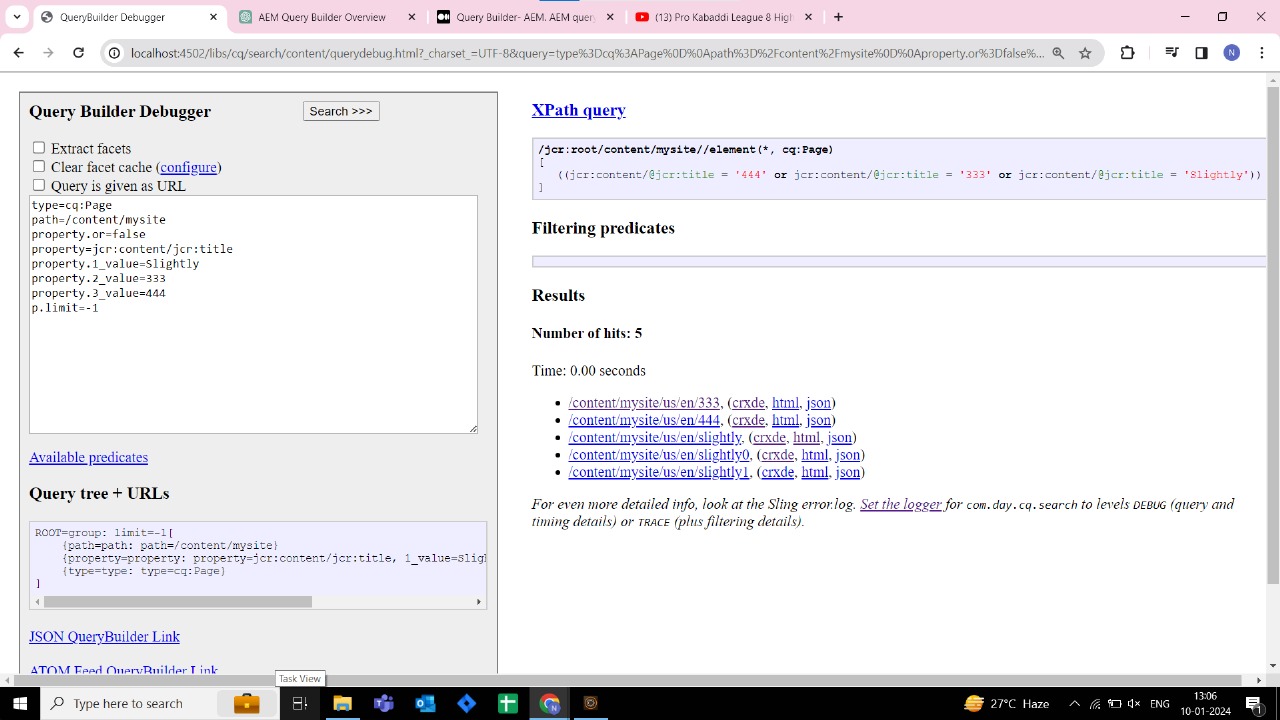
EVENT HANDLER:

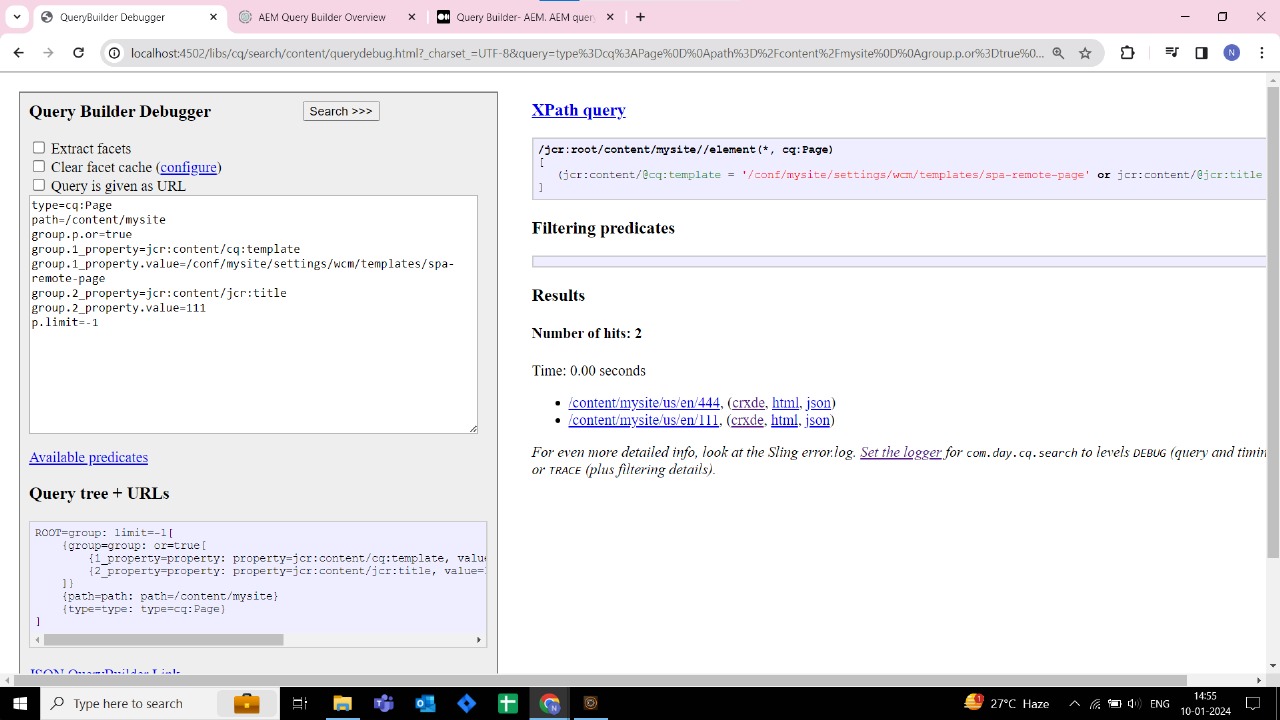


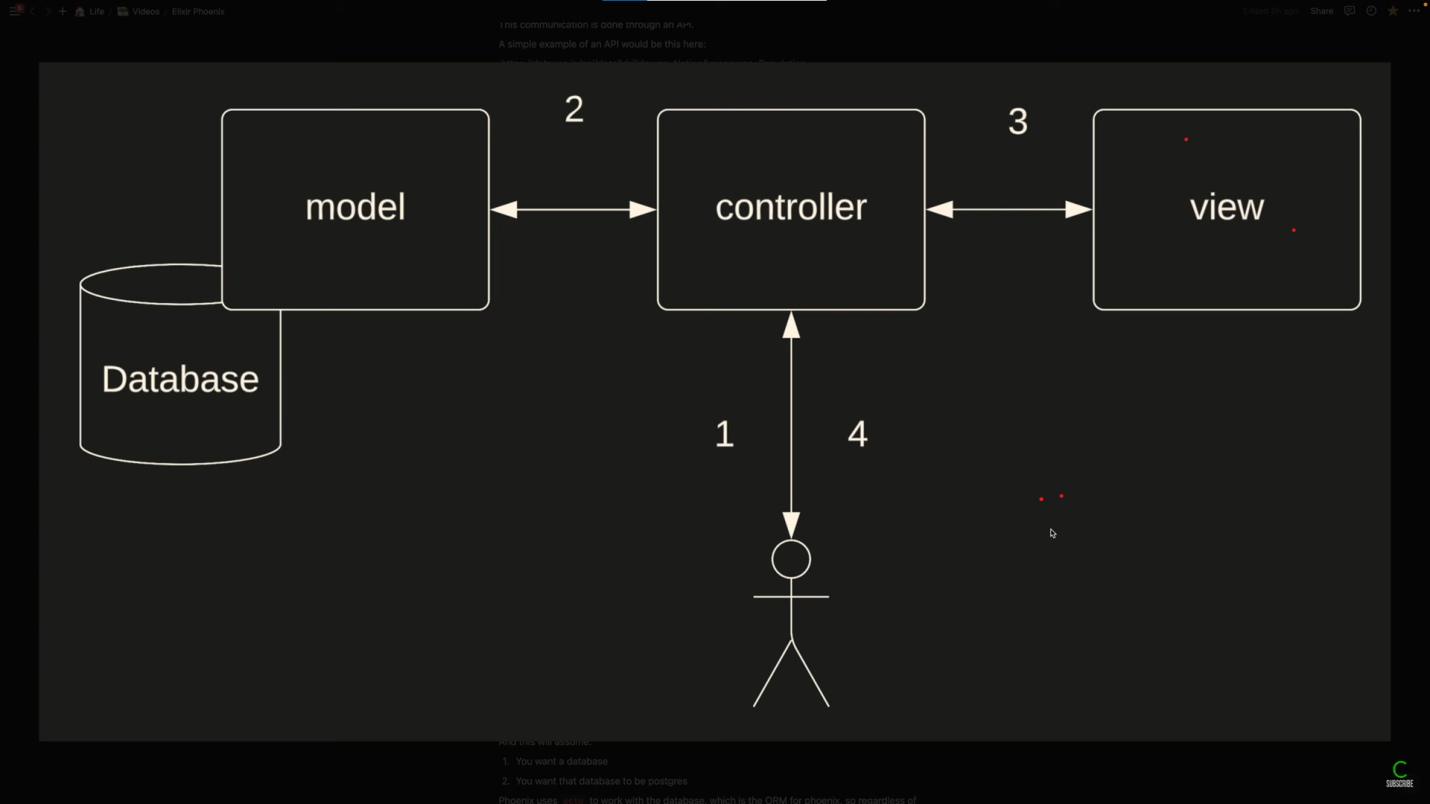
QUERY BUILDER:

<http://localhost:4502/libs/cq/search/content/querydebug.html?>

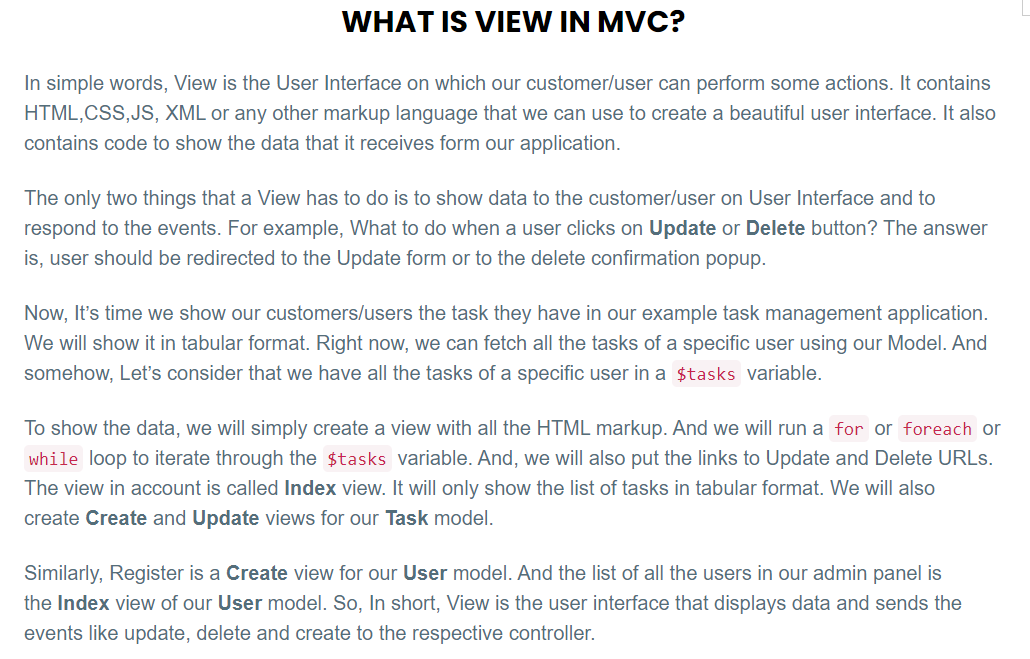


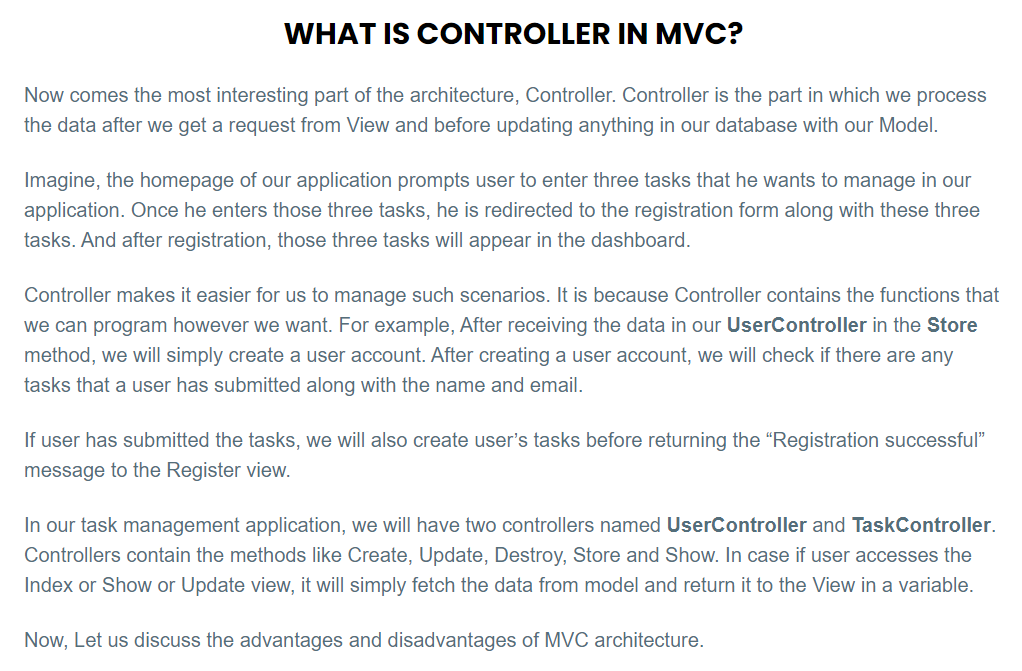




MVC: 

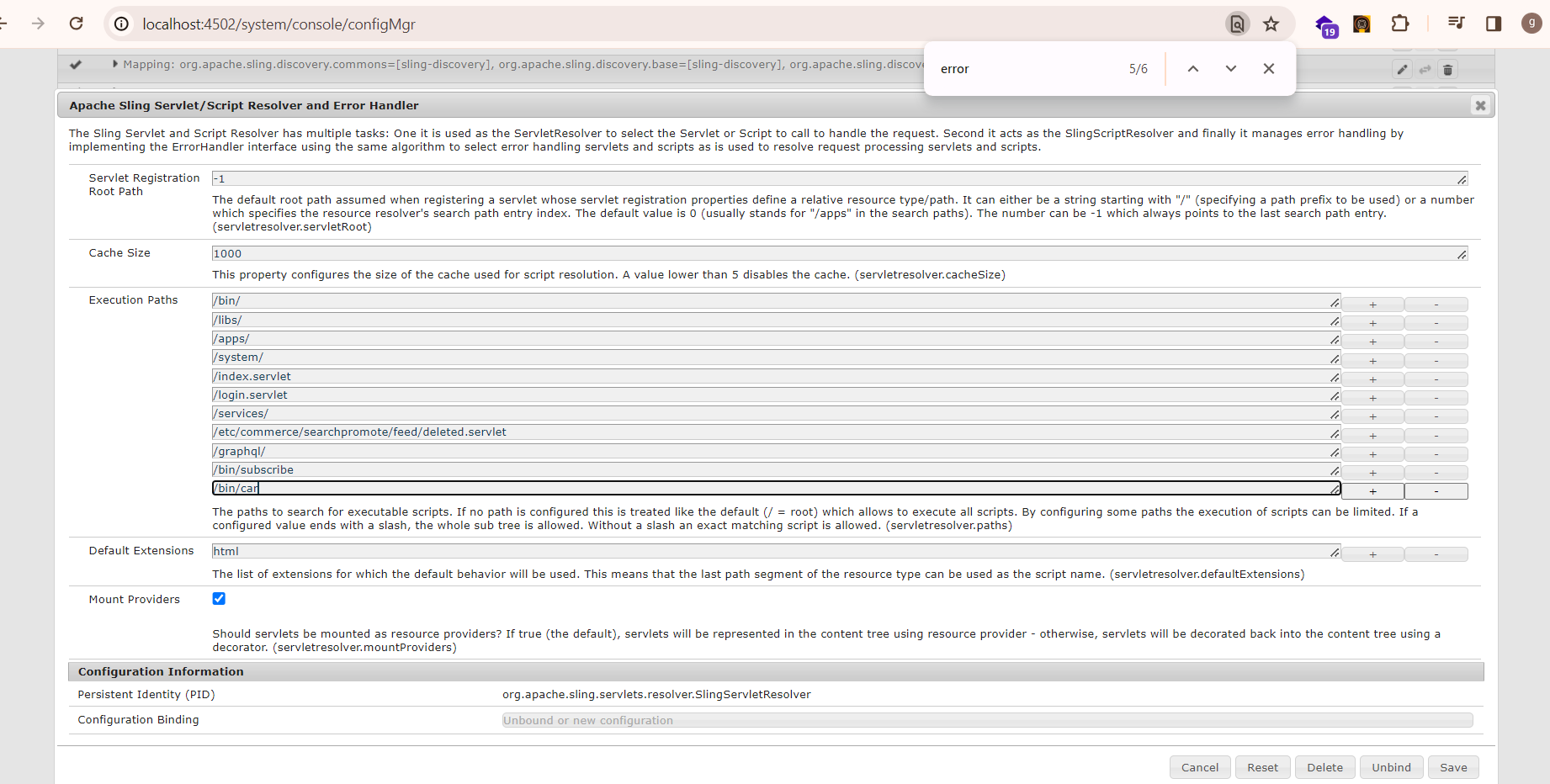
### 





Servlet Registeration

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



-