**Daily - AEM Task**

**1. Editable/Dynamic Template:**

There are two types of content, template level content and page level content.

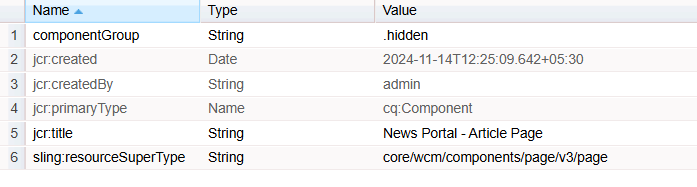
Global/common level content will be declared in Template.

Local/page level content will be declared in Page.

Page Component > Template Types > Template > Webpages

**Step 1: Page component creation**

In the newsportal website, already page component available under apps/component, just copy paste and rename.

  
**Step 2: Template Type creation**

In the newsportal website, already page template available under **/conf/newsportal/settings/wcm/template-types/page**, Just copy paste the node.

A screenshot of a computer

Description automatically generated

3 things need to be done:

Change the jcr:title of the article-page > jcr:content

A screenshot of a computer

Description automatically generated

Change the **sling:resourceType** to article-page component path on both Initial and Structure node.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**EVERY PAGE SHOULD have jcr:content and root mandatory node.**

**EVERY PAGE SHOULD CONTAIN ATLEAST ONE CONTAINER COMPONENT THAT IS ROOT. Root is pointing to container component.**

**Remove all the nodes under root on initial and structure node.**

A screenshot of a computer program

Description automatically generated

**Step 3: Template Creation**

Go to AEM Start > Tools > General > Templates > Newsportal (click on Create button at top right corner) > Select Article Template Type > Give name “Article Template”

A screenshot of a web page

Description automatically generated

A screenshot of a computer

Description automatically generated

Then Create button.

Template created under > /conf/newsportal/settings/wcm/templates/article-template

A screenshot of a computer

Description automatically generated

Whatever nodes are available under template type will be available under article template as well. So, template type is a blue print of template with predefined structure.

**Step 4: Content creation in template**

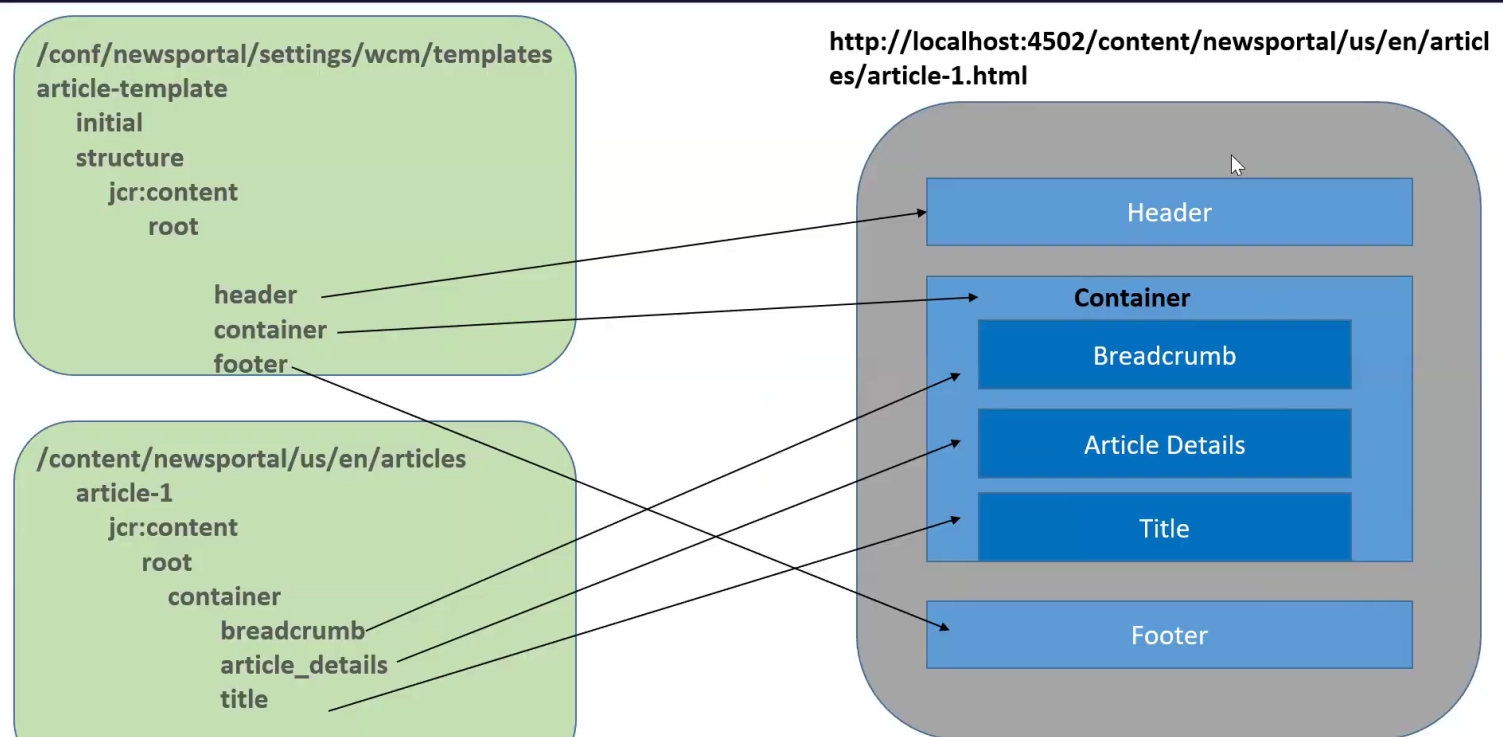
Without Enable the template, we can’t use this for page creation.

Page execution always starts from template. And structure node will be executed

Select the Article Template > Edit

A close-up of a computer screen

Description automatically generated



Default component will be authored in the Initial mode/Default mode. The connection will be executed only at the time of page creation. After that, no relationship between the initial content and template.

**Initial content is like a DISCOUNT. Discounts are avail only for new customers.**

If you made any changes in initial mode component/content, I will impact only newly created pages not existing pages.

A screenshot of a computer

Description automatically generated

Change the componentGroup in the component and make the policy update on the template.

**2. Experience Fragment (XF): Similar to Dynamic/Editable template creation.**

**1. XF page component creation**

**2. XF template type creation**

**3. XF template creation > Go to template and create XF template using XF template type and enable it.**

**4. Go to Nagivation> Experience Fragment > Create > Select XF template > Provide a name of the new XF template (Global/common component authoring)**

If there is re-usable content, In page level or template level, We can add it as a XF.

**We can use the XF for third party applications using the “plain” sector.**

destaco.com/content/experience-fragments/newsportal/us/en/site/header/product-header.plain.html

A screenshot of a computer

Description automatically generated

**3. Content Fragment (CF):**

**Step 1: Create a Content Fragment Model**

**Tools > General > Content Fragment Model > Go to Newsportal folder > Click on “Create” and provide the model name and click on “Create” button**

A screenshot of a computer

Description automatically generated

**Step 2:**

**Go to Assets console > Files > Newsportal > Create a folder “Content Fragment”**

**Go to “content-fragment” folder > click on “create” button > Contact Fragment**

A screenshot of a computer

Description automatically generated

**Select the new created content fragment model & click Next > provide a name of the content fragment > “Blog Author” > then click Create button.**

A screenshot of a computer

Description automatically generated

**Select the content fragment > click on Edit**

A screenshot of a computer

Description automatically generated

**Archiving Headless, using below 2 implementations in AEM:**

1. Asset HTTP API

localhost:4502/content/dam/newsportal/content-fragments/author-profiles

http://localhost:4502/api/assets/newsportal/content-fragments/author-profiles.json

2. GraphQL

Tools > General > GraphQL > Click on create and select newsportal.  
 **CF vs XF:**

1. We can use CF & XF for global content (common)

2. CF & XF are having variations

3. CF & XF support to export data (cf- .**json**, xf - .**html**)

**Difference:**

CF – Render structured/plain content without design

XF – Render content with design (styles, ex: header,footer)

CF – used to headless approach

XF – used to personalization (target), marketing (mobile, email social,web) global content (ex: header and footer)

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**4. Project Set-up**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Using below link, create a sample maven project with latest archetype:**

[**https://github.com/adobe/aem-project-archetype**](https://github.com/adobe/aem-project-archetype)

**Terminal:**

**mvn -B org.apache.maven.plugins:maven-archetype-plugin:3.2.1:generate \**

**-D archetypeGroupId=com.adobe.aem \**

**-D archetypeArtifactId=aem-project-archetype \**

**-D archetypeVersion=51\**

**-D appTitle="My Site" \**

**-D appId="mysite" \**

**-D groupId="com.mysite"**

* **mvn -B org.apache.maven.plugins:maven-archetype-plugin:3.2.1:generate -D archetypeGroupId=com.adobe.aem -D archetypeArtifactId=aem-project-archetype -D archetypeVersion=51 -D appTitle="Myportal" -D appId="myportal" -D groupId="com.myportal”**

**Sample AEM project template**

This is a project template for AEM-based applications. It is intended as a best-practice set of examples as well as a potential starting point to develop your own functionality.

**Modules**

The main parts of the template are:

* [core:](file:///D:\AEM\Cloud\Code\mysite\core\README.md) Java bundle containing all core functionality like OSGi services, listeners or schedulers, as well as component-related Java code such as servlets or request filters.
* [it.tests:](file:///D:\AEM\Cloud\Code\mysite\it.tests\README.md) Java based integration tests
* [ui.apps:](file:///D:\AEM\Cloud\Code\mysite\ui.apps\README.md) contains the /apps (and /etc) parts of the project, ie JS&CSS clientlibs, components, and templates
* [ui.content:](file:///D:\AEM\Cloud\Code\mysite\ui.content\README.md) contains sample content using the components from the ui.apps
* ui.config: contains runmode specific OSGi configs for the project
* [ui.frontend:](file:///D:\AEM\Cloud\Code\mysite\ui.frontend.general\README.md) an optional dedicated front-end build mechanism (Angular, React or general Webpack project)
* [ui.tests.cypress:](file:///D:\AEM\Cloud\Code\mysite\ui.tests.cypress\README.md) Cypress based UI tests
* [ui.tests.wdio:](file:///D:\AEM\Cloud\Code\mysite\ui.tests.wdio\README.md) Selenium based UI tests
* all: a single content package that embeds all of the compiled modules (bundles and content packages) including any vendor dependencies
* analyse: this module runs analysis on the project which provides additional validation for deploying into AEMaaCS

**How to build**

To build all the modules run in the project root directory the following command with Maven 3:

mvn clean install

To build all the modules and deploy the all package to a local instance of AEM, run in the project root directory the following command:

**mvn clean install -PautoInstallSinglePackage**

Or to deploy it to a publish instance, run

**mvn clean install -PautoInstallSinglePackagePublish**

Or alternatively

**mvn clean install -PautoInstallSinglePackage -Daem.port=4503**

Or to deploy only the bundle to the author, run

**mvn clean install -PautoInstallBundle**

Or to deploy only a single content package, run in the sub-module directory (i.e ui.apps)

**mvn clean install -PautoInstallPackage**

**Newsportal**

-header

- top-bar

- brand-logo

- navigation

- footer

-footer

-footer-list

**Web-page**

**AEM Components**

1. index.html (divided into multiple components)

**AEM Clientlibs (**

**In the clientlib-desktop > 2 fields are important**

**categories > String[] > newsportal.desktop**

**allowproxy > Boolean > true**

2. css

3. js

4. fonts and icons

**AEM DAM**

5. assets

6. content (text+asset reference) 🡪Content Nodes (pages, template)

**Three Important Steps:**

1. upload all assets into AEM DAM (localhost:4502/assets)

2. create clientlib and upload all required js, css and fonts, resources

3. convert index.html into multiple components (.html, dialog)

**<field>** is a mandatory one child node for Multifield.

Multifield node shouldn’t have “**name**” property. It should be in “**field**” node.

**5. OSGI Components & Services:**

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

OSGI Component:

- It is a Java Class

- OSGI Component can listen lifecycle events (activate, deactivate, modified)

OSGI Annotations:

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

**Class Level:**

1. @Component  🡪 It will register a normal java class into OSGi component.

**Method Level:**

2) @Activate

3) @Deactivate

4) @Modified

A screen shot of a computer program

Description automatically generated

5) @Reference – We call one component into another

@Reference will assign a dynamic object that is created by osgi container to

ArticleService articleObj;

A screen shot of a computer program

Description automatically generated

If you want to use once OSGi component into another OSGi component, we have to convert that component into OSGi service component.

Eg.

@Component(service=ArticleService.class)

We can’t define OSGI service, without OSGI component.

OSGi service is an OSGi component. But OSGi component may or may not be an OSGi service.

ArticleService

@Activate

- activate()

@Deactivate

- deactivate()

@Modified

- update()

PressReleaseService

- activate

- deactivate

- update

newsportal.core.vid.jar

- activate

- Container Identify all OSGI Component java classes based on XML files inside jar file

- Register OSGI component java classes with OSGI container

- Creates Object for all OSGI Component Java classes

- Triggers activate event and execute activate method.

**IMPORTANT:**

1. All OSGI components are singleton () java classes.

2. We can't create object for OSGI components.

Browser - Client

  Request - <https://gorest.co.in/public/v2/posts>

Java

- Client

- Request

- Response

- convert response into string

**OSGi Dynamic Configuration: (4 Steps)**

OSGi annotation:

**Class level:**

@ObjectClassDefinition > used to create a dynamic osgi configuration class

@Designate > Link the configuration class file with osgi component “ArticleService.class” file

**Property level:**

@AttributeDefinition

**Step 1:** Create a configuration class and define 3 properties that are make it dynamic.

A screenshot of a computer

Description automatically generated

A screen shot of a computer program

Description automatically generated

Step 2: Link the configuration class file with osgi ArticleService.class file using **@Designate**

A screen shot of a computer

Description automatically generated

Step 3: We have to pass the **dynamic configuration class** as an argument to all lifecycle method (Activate, Deactivate, Modified)

Using the dynamic object **config**, call the method from ArticleConfiguration class. And assign it to instance variable.

A screen shot of a computer program

Description automatically generated

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated

Step 4: **We have to create a dynamic configuration file (JSON) under ui.config module.**

ui.config

/apps

/newsportal

/osgiconfig

/config

com.bhasaka.newsportal.core.services.ArticleService.cfg.json (//cfg is a selector and json is an extension)

{ // These key : value pair taken from ArticleConfiguration.class file. Key is a dynamic **property**, go to configuration class file take it from there. Value we can define.

“articleRestApi”:”https://www.google.com”,

“enable”: “true”,

“clientid”:”989898”

}

Terminal:

>cd ui.config

ui.config> cd mvn clean install -PautoInstallPackage

cd ../core

core> mvn clean install -PautoInstallBundle

**Update Dynamic Configuratin:**

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

1. CRX De

/apps/newsportal/osgiconfig/config/com.bhasaka.newsportal.core.services.ArticleService.cfg.json

2. System Console Configuration

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

ui.config

/apps

/newsportal

/osgicongis

/config

com.bhasaka.newsportal.core.services.ArticleService.cfg.json

{

"articleRestAPI":"https://gorest.co.in/public/v2/posts",

"enable":"true",

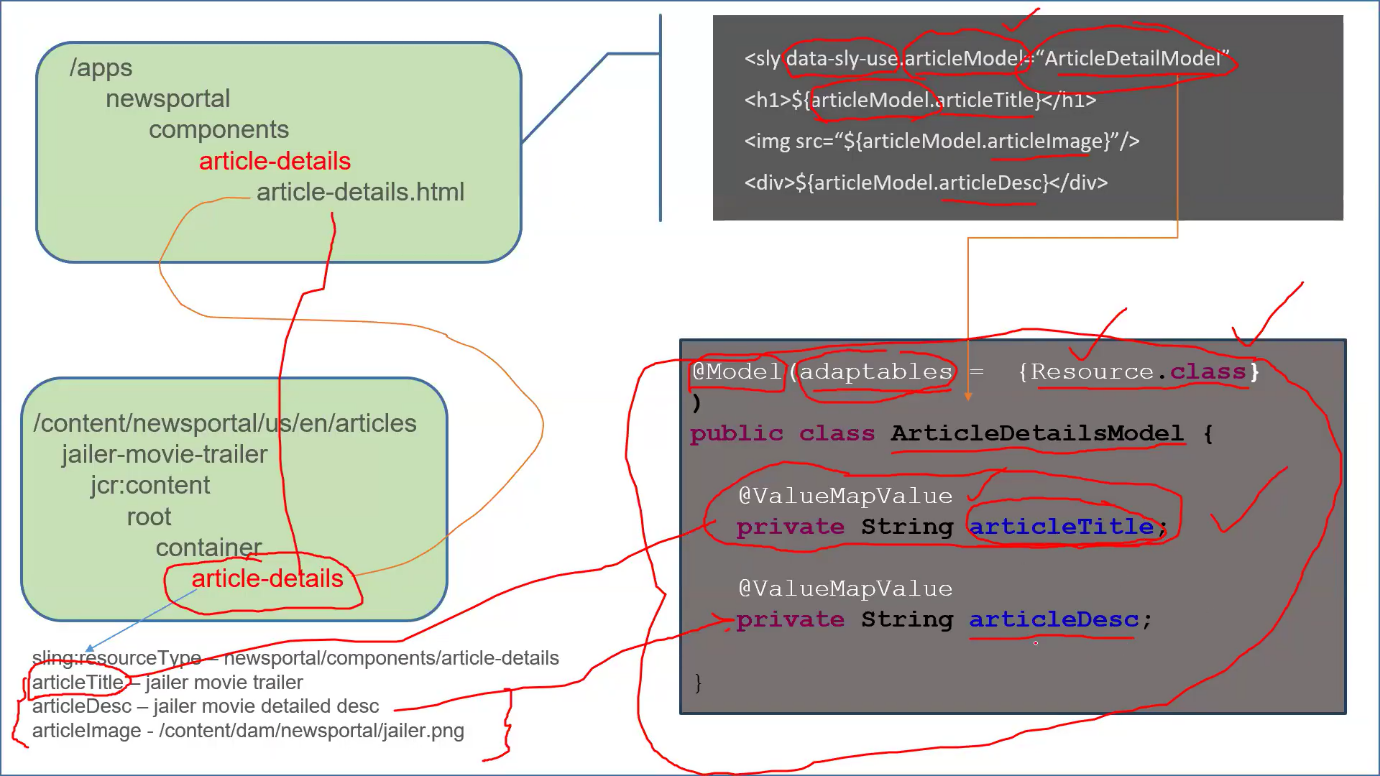
"clientId":"8908978"

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**6. Sling Model**

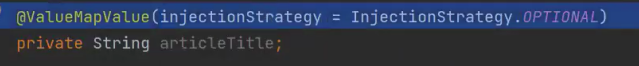
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



**@Model > used to register the java class into OSGi Model**

**@ValueMapValue** > used to read content from the content node in our java class

Private String articleTitle; // by default all the variables are mandatory. It should be available in the content node. However, We can make it optional, by adding the below

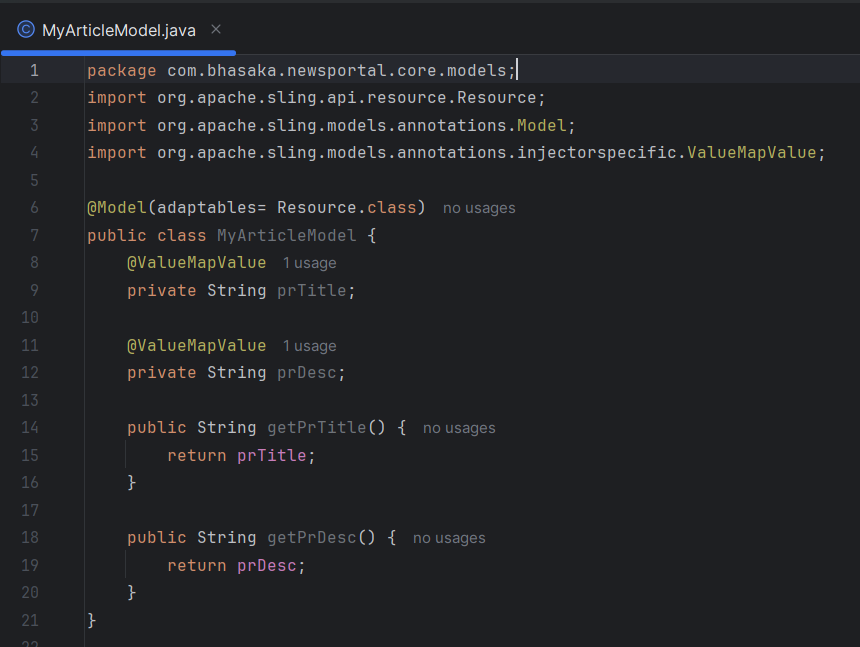


Also, We can make the optional, in the class level. **adaptables = Resource.class is mandatory attribute for @Model, Ensure CORRECT package has been imported for each annotations.**

**Issue Faced:**

The error message "com.bhasaka.newsportal.core.models.MyArticle cannot be resolved to a type" typically occurs in Java or in a Java-based project (such as an AEM project) when the compiler or IDE cannot find the class or type MyArticle.

The error "Identifier com.bhasaka.newsportal.core.models.MyArticleModel cannot be correctly instantiated by the Use API" typically occurs when AEM (Adobe Experience Manager) is unable to instantiate the model class using the Use API. This issue usuallyarises in a component where the model is being used in a Sightly (HTL) file, but there’s a problem with how the model is defined or instantiated.



A screen shot of a computer

Description automatically generated

* **getter() method is mandatory for every instance variable.**

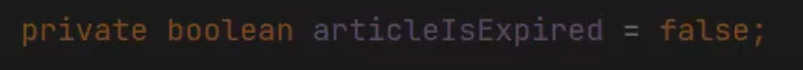
**If you want to access “content node” property in java class, use ‘name’ attribute. Also able to assign default value.**

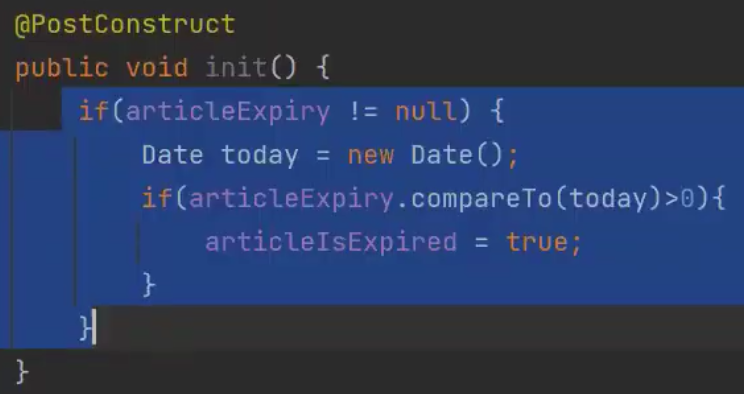
A screen shot of a computer

Description automatically generated

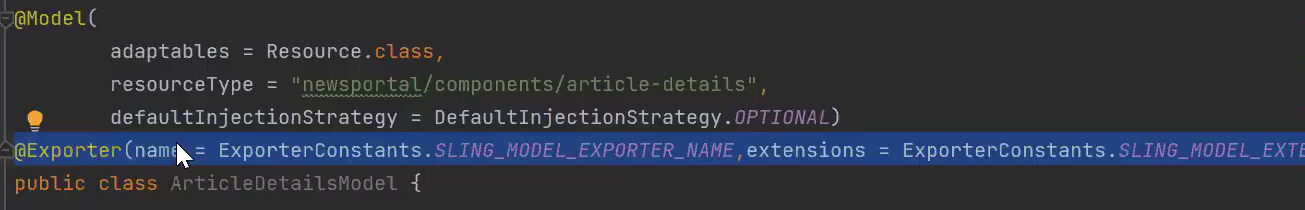
**@PostConstruct**

**public void init(){}** > The method will execute after injecting data into the variables.





Exporter 🡪 Used to export it to json



We can find out sling model > <http://localhost:4502/system/console/bundles>

**Manifest Headers >**

Sling-Model-Classes: com.bhasaka.newsportal.core.models.HelloWorldModel, com.bhasaka.newsportal.core.models.TopBarModel

Also, go to Sling > Sling Adapters (<http://localhost:4502/system/console/adapters>) (Search the model name)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**7. Sling Servlet**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* It is a OSGi service. We can access this from the browser.
* It’s Rest API.

Eg.

We have created OSGi service ie. “ArticleService” under core> module>services (*We can’t access this through browser directly*)

1. Using Sling Model, we can access the OSGi service OR
2. Access this in another OSGi service using @Reference OR
3. We call this service directly in the HTL (**component.html**) using object.

**HTTP Request method types:**

1. **GET** > Default > Read > perform read operation to fetch data from server.

2. **POST** > Create > perform write operation to post data from client to server. Eg. Contact Form data. When we entering the firstname, lastname in the contact form, It will go from client to server.

3. **PUT** > Create/Update > To create a new data/updating existing data. Eg. Contact address in Amazon online purchase.

4, **DELETE** > Delete > delete the data from the server.

Client (browser/postman) > Server (AEM)

**RecentArticleServlet extends SlingAllMethodServlet/SlingSafeMethodServlet**

**RecentArticleServlet is an Implementation class**

**SlingSafeMethodServlet**

* Read - GET

**SlingAllMethodServlet**

* Read – GET
* Write - POST
* Write/Update - PUT
* DELETE

**🡪Every servlet is an OSGi Service. In order to register it as a service, It must be a @Component.**

**Service class should be attribute of the @Component**

Eg.

**@Component(service=Servlet.class)** // Servlet is an interface

public class RecentArticleServlet extends **SlingSafeMethodServlet**{

}

**To register the OSGi service class into Servlet using below two types:**

**Two types of Servlet:**

**1.Path Based Servlet**

@**SlingServletPaths**(“/bin/newsportal/recent-articles”) // **This is important to register the class into Path based servlet.**

**HTTP Request Method**

―---------------------------------

GET

POST

PUT

DELETE

--------------------------------------―

**RecentArticleServlet**

doGet

do Post

do Put

doDelete

**Disadvantage:**

1. paths are hardcoded in java codes.

2. can't apply permissions.

A screenshot of a computer program

Description automatically generated

A computer screen shot of a program code

Description automatically generated

Terminal:

> cd core

core>mvn clean install -PautoInstallBundle

Go to browser > [http://localhost:4502//bin/newsportal/newsarticle //](http://localhost:4502//bin/newsportal/newsarticle%20//) doGet will executed

A screenshot of a computer

Description automatically generated

**We can test doPost, doPut, doDelete in 2 ways:**

1. Via javascript Ajax call
2. HTTP client > such as Postman

A screenshot of a computer

Description automatically generated

If we want to use our own servlet path instead of this **> /bin/newsportal/recent-articles** (this is an identifier in the servlet configuration and used to identify the servlet. And we have to allow this URL in the dispatcher filter section.)

Go to >[**http://localhost:4502/system/console/configMgr**](http://localhost:4502/system/console/configMgr)

Search term :” **ServletResolver**”

**Click on it:**



**Click on + icon and add “/newsportal/” and click SAVE.**

A screenshot of a computer

Description automatically generated

**Go to servlet file and remove bin and update path.**

A black screen with white text

Description automatically generated

**2. Resource Based Servlet:**

A computer screen shot of a blue and white text

Description automatically generated

**We can’t access “Resource Based Servlet” using the above URL. Because, resource means node. Through node only we can access the “Resource Based Servlet”**

A screenshot of a computer

Description automatically generated

**Go to crx/de > We have to create a node under content and add sling:resourceType and mention the resourceTypes URL given on the servlet class.**

A screenshot of a computer

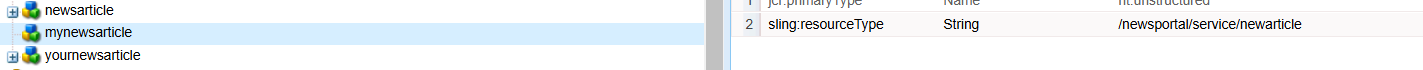
Description automatically generated

Now, we can access the servlet using the content node path.

<http://localhost:4502/content/newsarticle/>

* It physically exists
* Able to give permission

Using the node path only, we are accessing the servlet. So, we can create many nodes with different names to access the servlet.



Also, we can mention the extensions, allowed methods and selectors.

A screen shot of a computer

Description automatically generated

Selectors > It is a part of the URL and It is available in between the content node path and extensions.

A screenshot of a computer

Description automatically generated

If you want to know the class implementation of the servlet, there is one more console,

* System/console > Sling > Sling Servlet Resolver

A screenshot of a web browser

Description automatically generated

**If we have more than one servlet, have a create a new node and add sling:resourceType. Each node will pointing to one servlet or component.**

**Collections (Assets console)**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In **Adobe Experience Manager (AEM)**, **Collections** are a feature designed to organize and manage digital assets efficiently. They allow users to group assets together in a logical or functional way without affecting their actual storage location in the DAM (Digital Asset Manager). Collections can be created manually or dynamically based on search criteria.

**What are Collections in AEM?**

1. **Static Collections**:
   * Manually curated groups of assets.
   * Assets are selected and added individually to the collection.
   * Suitable when you need complete control over the assets in the collection.
2. **Smart (Dynamic) Collections**:
   * Automatically generated collections based on saved search criteria or filters.
   * Assets are added or removed dynamically as they meet or fail the specified criteria.
   * Ideal for large asset repositories where assets need to be organized based on metadata or rules.

**Key Features of Collections in AEM:**

* **Shared Accessibility**: Collections can be shared among team members for collaborative workflows.
* **Metadata Usage**: Metadata plays a key role in organizing assets dynamically in Smart Collections.
* **Ease of Use**: Collections provide a user-friendly interface to view and manage grouped assets.
* **Access Permissions**: Access to collections can be controlled through AEM’s permission system.
* **Versioning and Renditions**: Assets in collections maintain their original properties, renditions, and versions.

**Advantages of Using Collections in AEM:**

1. **Improved Asset Organization**:
   * Helps in logically grouping assets without altering their physical location in the DAM.
   * Reduces clutter and confusion in asset repositories.
2. **Faster Content Discovery**:
   * With Smart Collections, assets are automatically grouped based on predefined rules, making it easier to find relevant assets quickly.
3. **Enhanced Collaboration**:
   * Teams can share collections, making it simple to work together on a set of assets for projects or campaigns.
4. **Time-Saving**:
   * Smart Collections dynamically update based on criteria, reducing the need for manual reorganization.
5. **Flexibility**:
   * Collections can cater to various use cases such as campaign-specific assets, localization, or user-specific preferences.
6. **Scalability**:
   * As asset libraries grow, dynamic collections ensure that the organization remains efficient and manageable.
7. **Centralized Access Control**:
   * Permissions can be set at the collection level, ensuring secure access to grouped assets.
8. **Marketing Campaign Management**:
   * Collections can group assets relevant to specific campaigns, regions, or product launches, streamlining asset delivery.

**Use Cases for Collections:**

* **Marketing Teams**: Organize assets for specific campaigns or regions.
* promotions.
* **Localization**: Dynamically group assets based on language metadata.
* **Branding Guidelines**: Centralize brand-related assets in a collection for easy distribution.

By utilizing Collections in AEM, teams can work more effectively, ensuring that the right assets are easily accessible and organized according to business needs.

**SQL 2**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<http://localhost:4502/crx/de>

Tools > Query

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated