Tasks - 20/03/2025

1. Creating a Custom News Component

AEM enables the creation of custom components for dynamic content display. The News component will allow authors to input articles with a title, details, and a publication date.

Steps:

- Open CRXDE Lite in AEM (http://localhost:4502/crx/de).
- Navigate to /apps/myTraining/components and create a folder named news.
- Inside the news folder, create the necessary component files, including news.html.
- Structure the component to dynamically retrieve properties from the content repository.
- Save and activate the component.

Code Implementation:
news.html

html

CopyEdit

```
<sly data-sly-use.clientLib="/libs/granite/sightly/templates/clientlib.html"/>

<sly data-sly-call="${clientLib.css @ categories='myTraining.news'}"/>

<div class="cop-news-component">

    <h2 class="news-title">${properties.title}</h2>

    <p class="news-detail">${properties.detail}</p>

    <p class="news-date">${properties.publishedDate}</p>

    <p class="news-source">Source: ${properties.source}</p>

</div>

<sly data-sly-call="${clientLib.js @ categories='myTraining.news'}"/>

<sly data-sly-use.newsModel="com.myTraining.core.models.NewsComponentModel"/>

<p>${newsModel.helloMessage}</p>
```

NewsModel.java

java

CopyEdit

```
package com.myTraining.core.models;
```

```
import com.myTraining.core.services.HelloWorldService;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.DefaultInjectionStrategy;
import org.apache.sling.models.annotations.Model;
import org.apache.sling.models.annotations.Default;
import javax.inject.Inject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.apache.sling.models.annotations.injectorspecific.OSGiService;

@Model(adaptables = Resource.class)
public class NewsModel {

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

    @Inject
    @Default(values = "No Title")
    private String title;

    @Inject
    @Default(values = "No Details Available")
    private String detail;

    @Inject
    @Default(values = "Unknown Date")
    private String publishedDate;

    @Inject
```

```
    @Default(values = "Unknown Source")
    private String source;

    @OSGiService
    private HelloWorldService myTrainingService;

    public String getTitle() {
        return title;
    }

    public String getDetail() {
        return detail;
    }

    public String getPublishedDate() {
        return publishedDate;
    }

    public String getSource() {
        return source;
    }

    public String getServiceMessage() {
        String message = myTrainingService.getMessage();
        LOG.info("NewsModel - Received message from service: {}", message);
        return message;
    }
}
```

2. Creating a Multifield News Component

A multifield component allows authors to input multiple news items within a single instance.

Steps:

- Create a component called newsMultifield under /apps/myTraining/components.
- Define a cq:dialog with a multifield widget for entering multiple articles.
- Ensure the structure supports dynamic content handling.
- Save and activate the component.

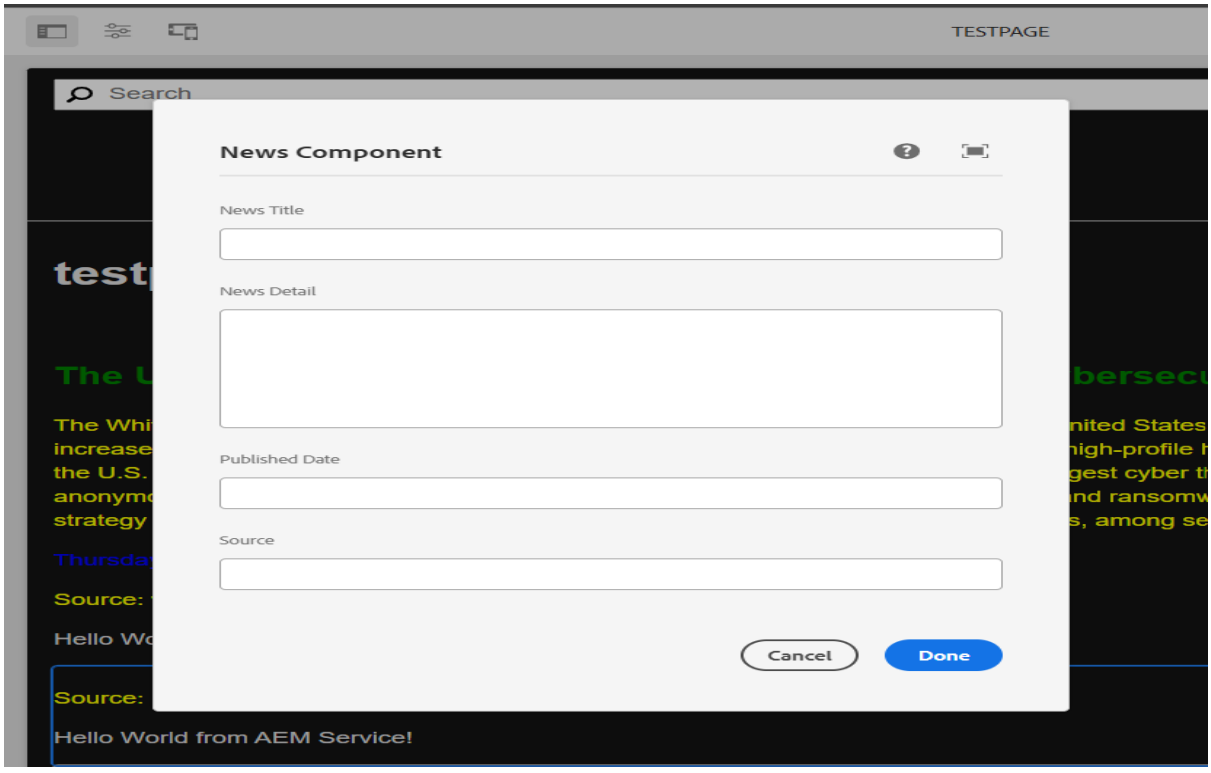
Code Implementation:
NewsListModel.java

java

CopyEdit

```
package com.myTraining.core.models;


import org.apache.sling.api.resource.Resource;

import javax.inject.Inject;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.apache.sling.models.annotations.DefaultInjectionStrategy;

import org.apache.sling.models.annotations.Model;


@Model(adaptables = Resource.class, defaultInjectionStrategy =
DefaultInjectionStrategy.OPTIONAL)

public class NewsListModel {


   @Inject

   private List<NewsItem> newsItems;


   public List<NewsItem> getNewsItems() {

      return Optional.ofNullable(newsItems).orElse(new ArrayList<>());

   }
```

```
    public static class NewsItem {
        @Inject
        private String title;

        @Inject
        private String source;

        public String getTitle() {
            return title;
        }

        public String getSource() {
            return source;
        }
    }
}
```



3. Adding Clientlibs and Styling the News Component

Client-side libraries in AEM manage styles and scripts efficiently.

Steps:

- Navigate to /apps/myTraining/clientlibs and create a folder for clientlib-news.
- Define styles to apply the following colors:
    - Green for headings
    - Yellow for news details
    - Black for the date
- Link the clientlib to the News component.
- Save and activate the styles.

CSS Styling:

css

CopyEdit

```
.cop-news-component h2 {
   color: green;
}

.cop-news-component p {
   color: yellow;
}

.cop-news-component .news-date {
   color: black;
}
```

4. Developing a Base Page Component

A Base Page Component provides a reusable structure for pages with predefined functionalities.

Steps:

- Navigate to /apps/myTraining/components and create a component named basepage.

- Define a cq:dialog for configurable page properties.
- Structure the basepage template to include essential sections such as head, body, and footer.
- Save and activate the component.

Code Implementation:

xml

CopyEdit

```
<?xml version="1.0" encoding="UTF-8"?>
<jcr:root xmlns:sling="http://sling.apache.org/jcr/sling/1.0"
    xmlns:cq="http://www.day.com/jcr/cq/1.0"
    xmlns:jcr="http://www.jcp.org/jcr/1.0"
    jcr:primaryType="cq:Component"
    jcr:title="Base Page"
    componentGroup="myTraining">
</jcr:root>
```

5. Setting Up Global Page Properties

Global Page Properties enable site-wide configurations such as titles, logos, and branding.

Steps:

- Create a template-type component for managing global settings.
- Define a cq:dialog to allow authors to configure site-wide properties.
- Ensure accessibility of these properties within page templates.
- Save and activate the component.

Code Implementation:

xml

CopyEdit

```
<?xml version="1.0" encoding="UTF-8"?>
<jcr:root xmlns:sling="http://sling.apache.org/jcr/sling/1.0"
    xmlns:cq="http://www.day.com/jcr/cq/1.0"
    xmlns:jcr="http://www.jcp.org/jcr/1.0"
```

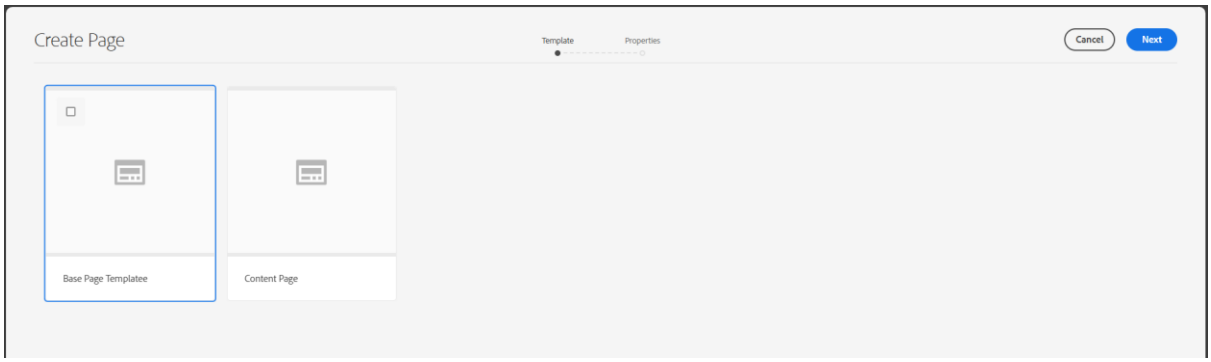
```
        jcr:primaryType="cq:Component"
        jcr:title="Global Page Properties"
        componentGroup="myTraining">
</jcr:root>
```

cq:dialog Structure:

xml

CopyEdit

```
<jcr:root xmlns:sling="http://sling.apache.org/jcr/sling/1.0"
        xmlns:cq="http://www.day.com/jcr/cq/1.0"
        xmlns:jcr="http://www.jcp.org/jcr/1.0"
        jcr:primaryType="nt:unstructured">
    <items jcr:primaryType="nt:unstructured">
        <ogTitle jcr:primaryType="nt:unstructured"
sling:resourceType="granite/ui/components/coral/foundation/form/textfield" fieldLabel="OG
Title" name="./ogTitle"/>
        <ogDescription jcr:primaryType="nt:unstructured"
sling:resourceType="granite/ui/components/coral/foundation/form/textfield" fieldLabel="OG
Description" name="./ogDescription"/>
        <ogImage jcr:primaryType="nt:unstructured"
sling:resourceType="granite/ui/components/coral/foundation/form/pathfield"
fieldLabel="OG Image Path" name="./ogImage"/>
    </items>
</jcr:root>
```



6. Understanding extraClientLibs in AEM

The extraClientLibs property in AEM allows additional client libraries to be loaded dynamically for specific templates or components.

Use Cases:

- When specific styles or scripts need to be applied to selected pages.
- Extending existing client libraries without modifying global configurations.

Implementation:

- Define additional client libraries within the component or template.
- Use extraClientLibs to selectively load JavaScript or CSS.
- Ensure proper dependency management to prevent conflicts.
- Save and verify integration.