20/03/2025 - TASKS

1. Creating a New Component for News

AEM allows users to create custom components to display content dynamically. The News component will enable authors to enter news articles with a title, details, and a date.

Steps:

1. **Open CRXDE Lite in AEM (http://localhost:4502/crx/de).**
2. **Navigate to /apps/myTraining/components and create a new folder named news.**
3. **Inside the news folder, create the necessary component files, including the news.html file.**
4. **Structure the component to retrieve properties dynamically from the content repository.**
5. **Save and activate the component for use.**

Code Snippet:

News.html

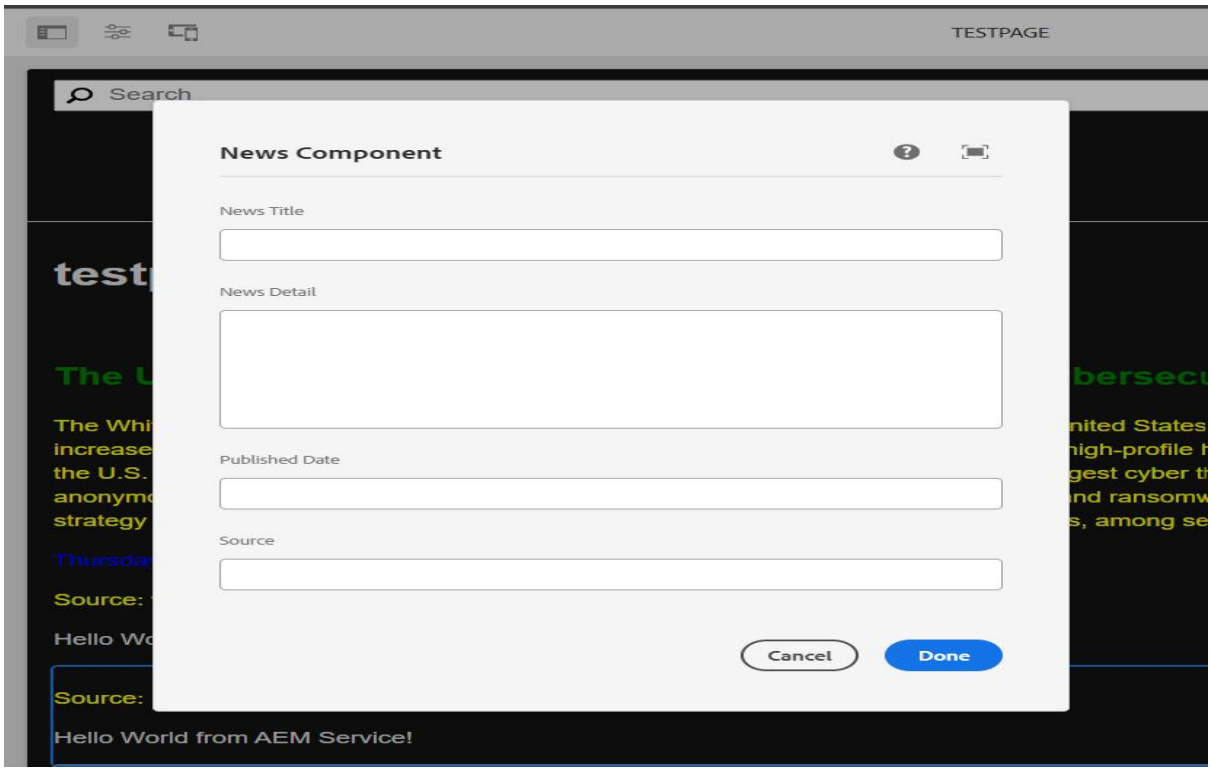
```
<sly data-sly-use.clientLib="/libs/granite/sightly/templates/clientlib.html"/>
<sly data-sly-call="${clientLib.css @ categories='myTraining.news'}"/>
<div class="cop-news-component">
  <h2 class="news-title">${properties.title}</h2>
  <p class="news-detail">${properties.detail}</p>
  <p class="news-date">${properties.publishedDate}</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

```
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 org.apache.sling.models.annotations.injectorspecific.ValueMapValue;
import javax.inject.Inject;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.apache.sling.models.annotations.injectorspecific.OSGiService;

@Model(adaptables = org.apache.sling.api.resource.Resource.class)
public class NewsModel {

    private static final Logger LOG = LoggerFactory.getLogger(NewsModel.class);  // ☑ Fixed Logger
    @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. Create a Multifield Component

A multifield component allows authors to add multiple news entries dynamically within a single component instance.

Steps:

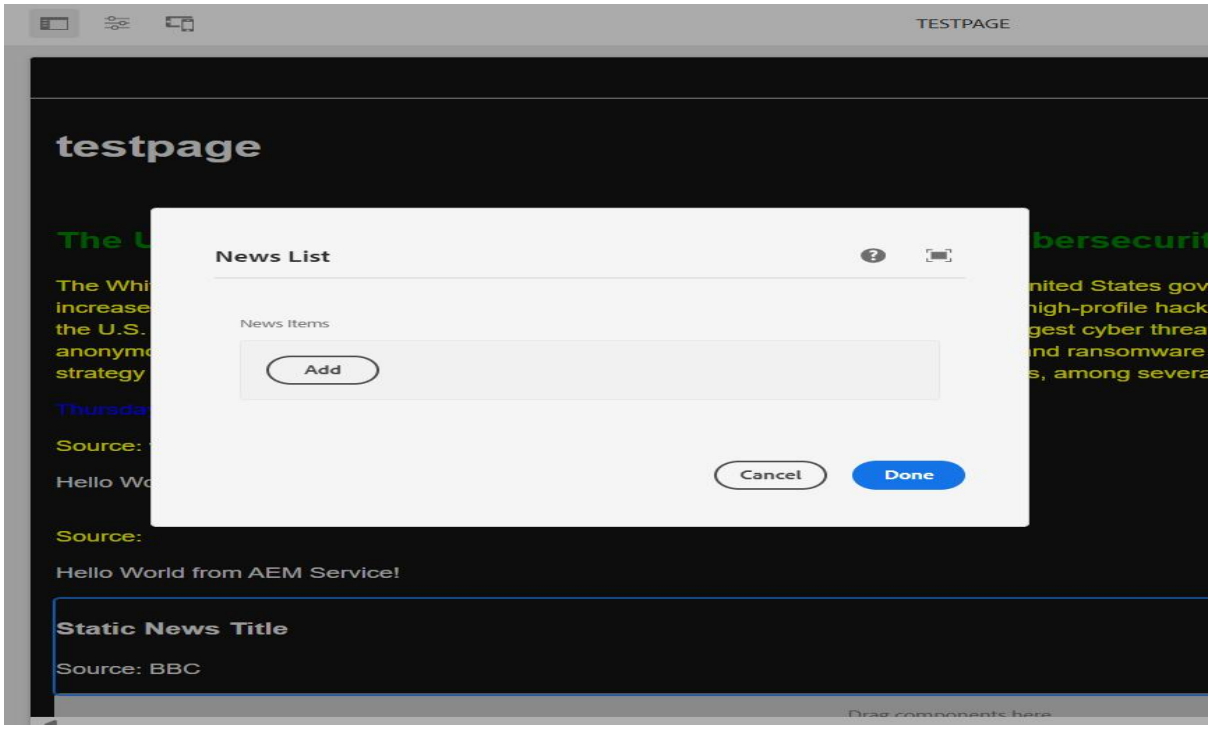
1. **Create a new component called newsMultifield under /apps/myTraining/components.**
2. **Define a cq:dialog to include a multifield widget.**
3. **Ensure the multifield structure can handle dynamic content inputs.**
4. **Save and activate the component for use in AEM pages.**

Code Snippet:

NewsListCompontnt.java

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

import com.adobe.cq.sightly.WCMUsePojo;
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;
      }
   }
}
```

Output:



3. Create Clientlibs and Apply Styling

Client-side libraries (Clientlibs) in AEM help manage styles and scripts efficiently. We will define styles for the News component.

Steps:

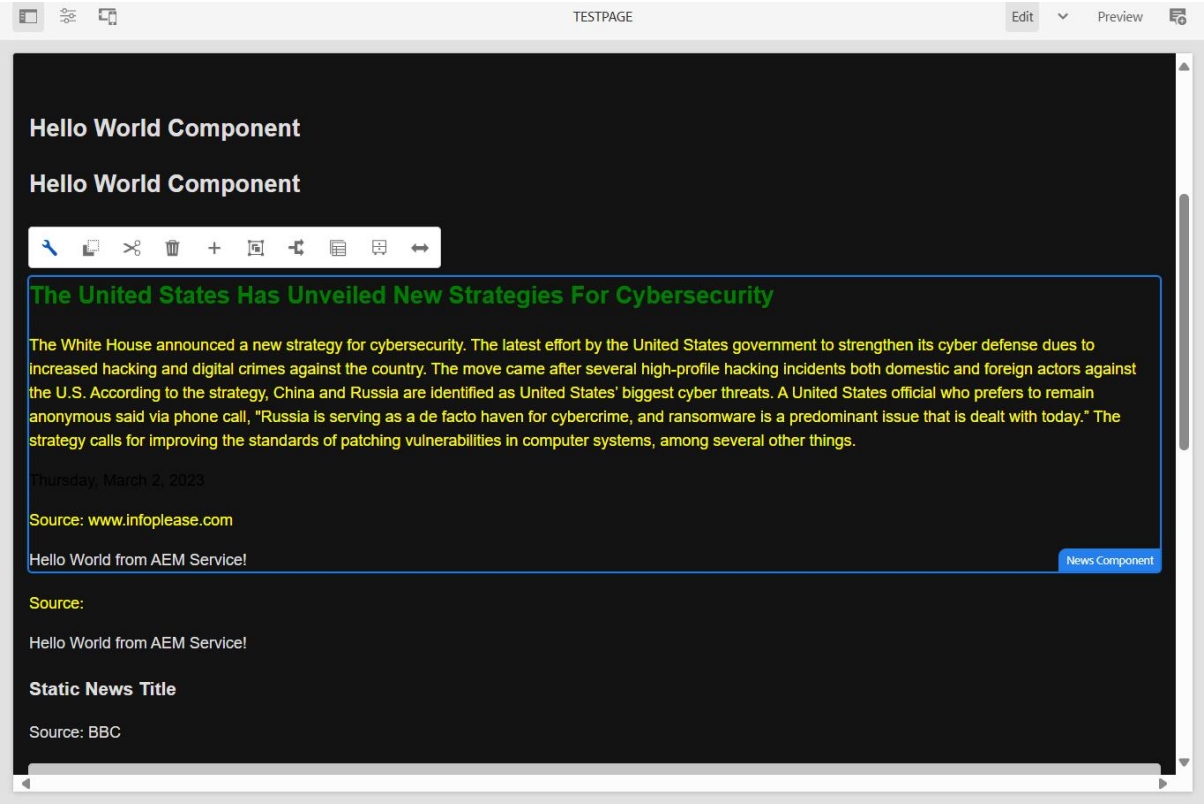
1. **Navigate to /apps/myTraining/clientlibs and create a new folder for clientlib-news.**
2. **Define CSS styles to apply colors:**
    - **Green for heading**
    - **Yellow for news details**
    - **Black for date**
3. **Ensure the clientlib is properly linked to the News component.**
4. **Save and activate the styles.**

Code Snippet:

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

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

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

Output:



4. Create a Base Page Component

A Base Page Component provides a reusable structure for pages, enabling common functionalities and templates.

Steps:

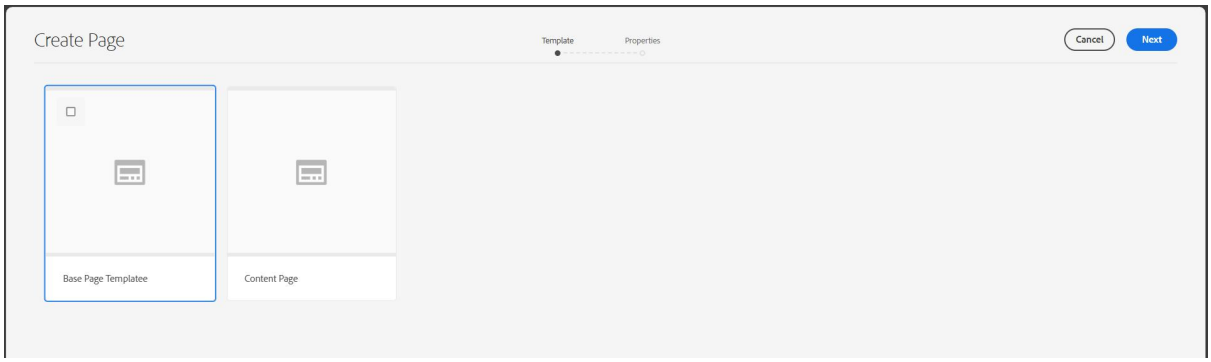
1. **Navigate to /apps/myTraining/components and create a new component called basepage.**
2. **Define a cq:dialog for configurable page properties.**

3. **Structure the basepage template to include key sections such as head, body, and footer.**
4. **Save and activate the base page component for use across multiple templates.**

Code Snippet:

```
<?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>
```

Output



5. Create Global Page Properties

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

Steps:

1. **Create a new template-type component to manage global properties.**
2. **Define a cq:dialog to allow authors to configure global settings.**
3. **Ensure that the global properties can be accessed dynamically within page templates.**
4. **Save and activate the component for usage.**

Code Snippet:

```
<?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

<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. What is extraClientLibs?

extraClientLibs is a property in AEM that allows loading additional client libraries dynamically for specific templates or components.

Use Case:

- **When certain scripts or styles need to be loaded only for specific pages or components.**
- **When you want to extend existing client libraries without modifying global settings.**

Implementation Steps:

1. **Define the additional client libraries within the component or template.**
2. **Configure extraClientLibs to load required JavaScript or CSS selectively.**
3. **Ensure proper dependency management to avoid conflicts.**
4. **Save and test the integration.**