1. Create a new component which has news title (<h2>), news detail (<p>), published date, and source. Create this component using Sling Model.

In AEM, components can use **Sling Models** to bind backend data to frontend views. Here, we create a simple **News Item component** with the following fields:

- Title
- Detail
- Published Date
- Source

These fields are defined in the dialog and accessed using a Sling Model in the HTL (HTML Template Language).

Sling Model: NewsItemModel.java

```
package com.example.core.models;


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

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

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


import javax.inject.Inject;


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

public class NewsItemModel {


    @Inject

    private String title;
```

```
    @Inject
    private String detail;

    @Inject
    private String publishedDate;

    @Inject
    private String source;

    public String getTitle() {
        return title;
    }

    public String getDetail() {
        return detail;
    }

    public String getPublishedDate() {
        return publishedDate;
    }

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

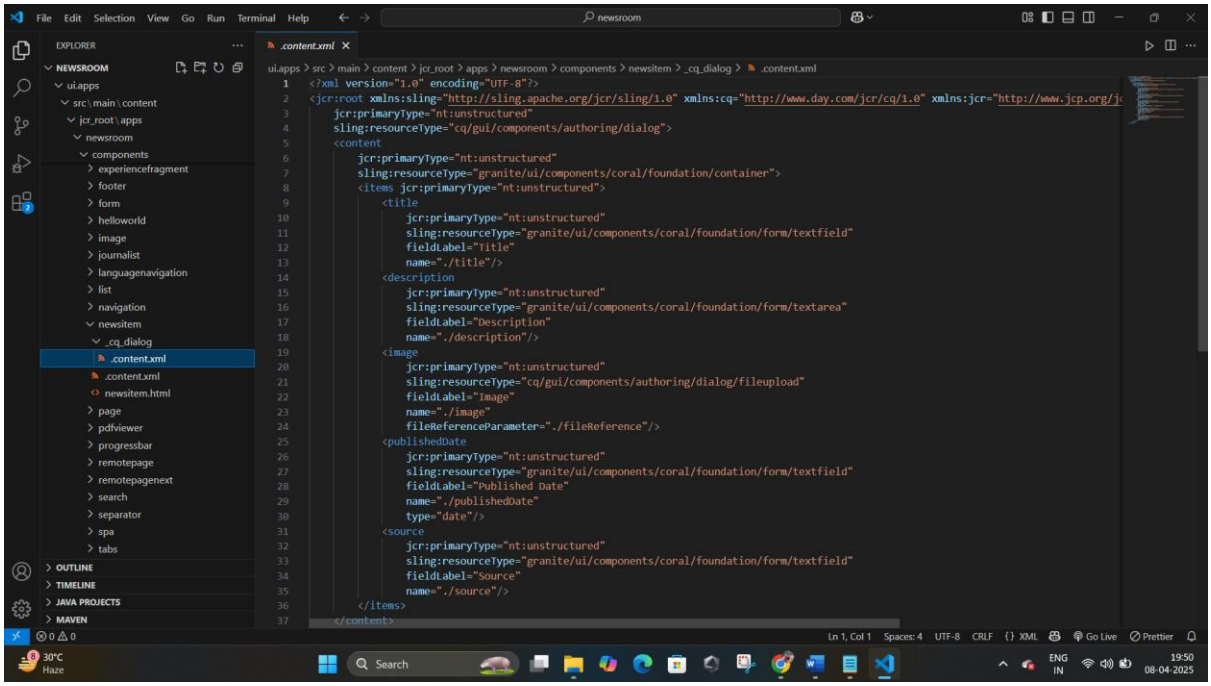
HTL File: news-item.html

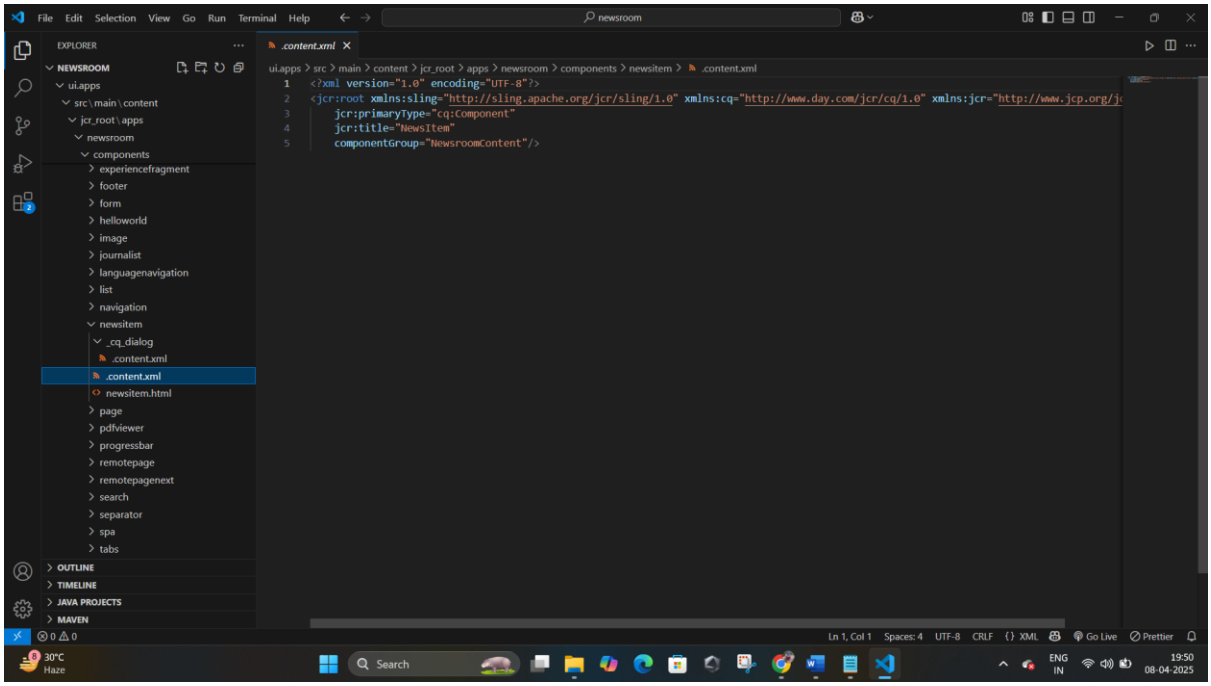
```
<data-sly-use.news="com.example.core.models.NewsItemModel" />
```

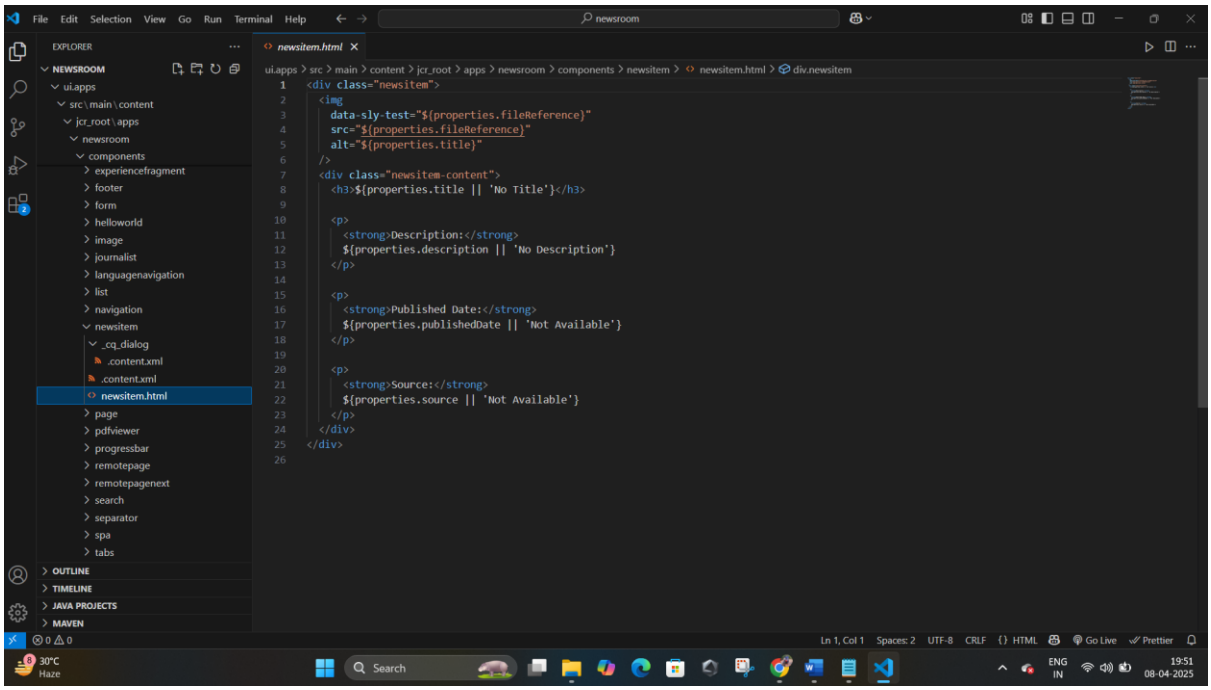
```
<div class="cop-news-component">
    <h2 class="news-title">${news.title}</h2>
    <p class="news-detail">${news.detail}</p>
    <p class="news-date">${news.publishedDate}</p>
    <p class="news-source">Source: ${news.source}</p>
</div>
```

What's Happening?

- The Sling Model fetches the fields from the component dialog.
- HTL template uses this model and renders the data in proper HTML tags.
- CSS class cop-news-component helps with styling.




File Edit Selection View Go Run Terminal Help
newsroom
EXPLORER
NEWSROOM
ui.apps
src\main\content
jcr_root\apps
newsroom
components
experiencefragment
footer
form
helloworld
image
journalist
languagenavigation
list
navigation
newsitem
_cq_dialog
.content.xml
.content.xml
newsitem.html
page
pdfviewer
progressbar
remotepage
remotepagenext
search
separator
spa
tabs
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
.content.xml
ui.apps > src > main > content > jcr_root > apps > newsroom > components > newsitem > .content.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <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/jc
3 jcr:primaryType="cq:Component"
4 jcr:title="NewsItem"
5 componentGroup="NewsroomContent"/>
Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} XML Go Live Prettier
30°C Haze
Search
ENG IN
19:50 08-04-2025


File Edit Selection View Go Run Terminal Help
newsroom
EXPLORER
NEWSROOM
ui.apps
src\main\content
jcr_root\apps
newsroom
components
experiencefragment
footer
form
helloworld
image
journalist
languagenavigation
list
navigation
newsitem
_cq_dialog
.content.xml
.content.xml
newsitem.html
page
pdfviewer
progressbar
remotepage
remotepagenext
search
separator
spa
tabs
OUTLINE
TIMELINE
JAVA PROJECTS
MAVEN
newsitem.html
ui.apps > src > main > content > jcr_root > apps > newsroom > components > newsitem > newsitem.html > div.newsitem
1
2 <img
3 data-sly-test="${properties.fileReference}"
4 src="${properties.fileReference}"
5 alt="${properties.title}"
6 />
7
8 <h3>${properties.title || 'No Title'}</h3>
9
10
11 Description:
12 ${properties.description || 'No Description'}
13
14
15
16 Published Date:
17 ${properties.publishedDate || 'Not Available'}
18
19
20
21 Source:
22 ${properties.source || 'Not Available'}
23
24
25
26
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} HTML Go Live Prettier
30°C Haze
Search
ENG IN
19:51 08-04-2025

2. Create multifield component (multiple news) using Sling Model. Fields: title of news and source of news

Explanation:

A **multifield component** allows authors to input multiple sets of similar data (e.g., multiple news items). Here, each news item includes:

- Title
- Source

We use:

- A Coral UI Multifield dialog
- A Sling Model to map a list of items
- HTL to loop over and render each item

cq:dialog XML

```
<newsItems
   jcr:primaryType="nt:unstructured"
   sling:resourceType="granite/ui/components/coral/foundation/form/multifield"
   fieldLabel="News Items"
   name="./newsItems"
   composite="{Boolean}true">
   <field
      jcr:primaryType="nt:unstructured"
      sling:resourceType="granite/ui/components/coral/foundation/form/fieldset">
      <layout
         jcr:primaryType="nt:unstructured"

sling:resourceType="granite/ui/components/coral/foundation/layouts/fixedcolumns"/>
      <items jcr:primaryType="nt:unstructured">
```

```
        <title
            jcr:primaryType="nt:unstructured"
            sling:resourceType="granite/ui/components/coral/foundation/form/textfield"
            fieldLabel="Title"
            name="./title"/>
        <source
            jcr:primaryType="nt:unstructured"
            sling:resourceType="granite/ui/components/coral/foundation/form/textfield"
            fieldLabel="Source"
            name="./source"/>
      </items>
   </field>
</newsItems>
```

NewsMultifieldModel.java

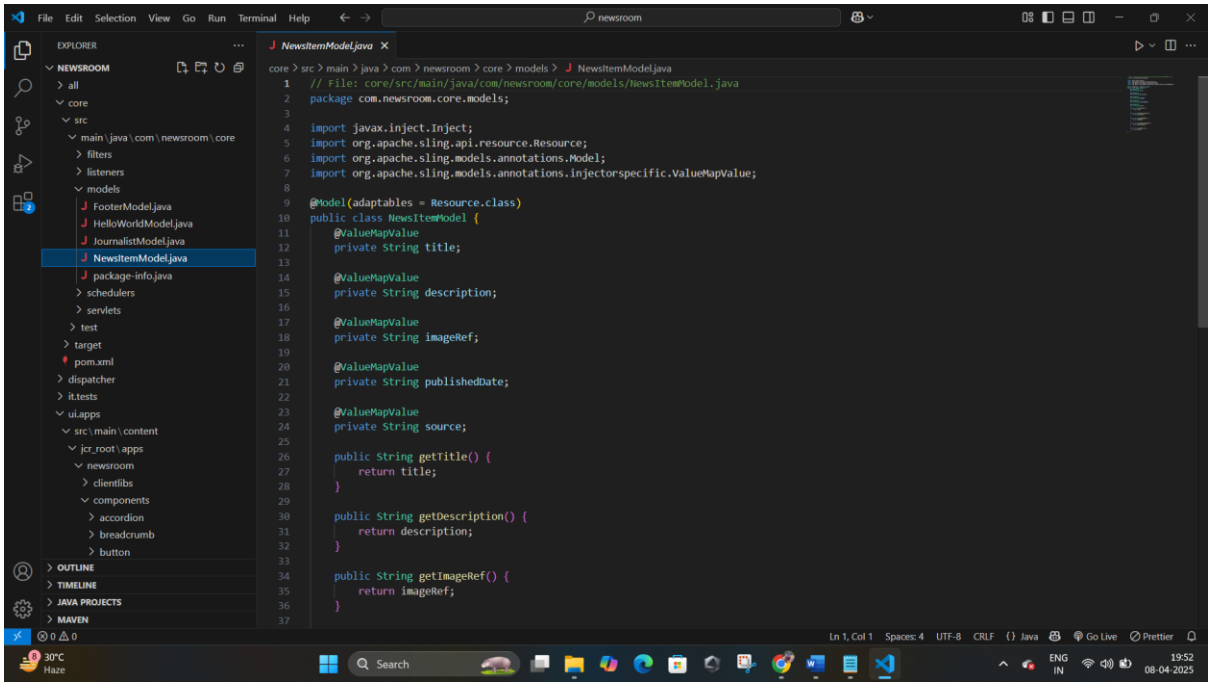
```
package com.example.core.models;

import org.apache.sling.api.resource.Resource;
import org.apache.sling.models.annotations.DefaultInjectionStrategy;
import org.apache.sling.models.annotations.Model;

import javax.inject.Inject;
import java.util.List;

@Model(adaptables = Resource.class, defaultInjectionStrategy =
DefaultInjectionStrategy.OPTIONAL)
public class NewsMultifieldModel {
```

```
    @Inject
    private List<NewsItem> newsItems;

    public List<NewsItem> getNewsItems() {
        return newsItems;
    }

    @Model(adaptables = Resource.class)
    public static class NewsItem {
        @Inject
        private String title;

        @Inject
        private String source;

        public String getTitle() { return title; }

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

news-multifield.html

```
<data-sly-use.newsList="com.example.core.models.NewsMultifieldModel" />

<ul class="cop-news-component">
    <data-sly-list.newsItem="${newsList.newsItems}">
        <li>
```

```
        <h2>${newsItem.title}</h2>
        <p>Source: ${newsItem.source}</p>
      </li>
    </data-sly-list>
</ul>
```



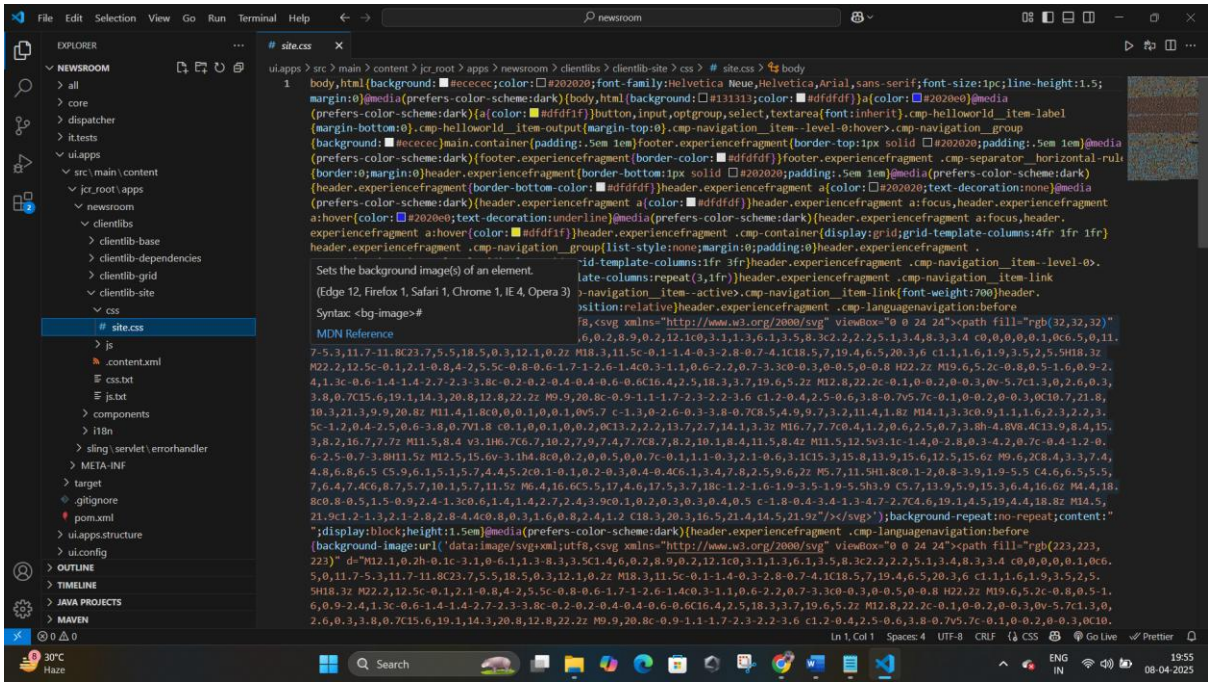
3. Create clientlibs for news component

Explanation:

clientlibs in AEM allow us to add custom CSS or JS to components. We'll create a clientlib to style the news component.

/apps/news/clientlibs/newscomponent

- css.txt
- news.css



4. Apply color styles: Green for heading, Yellow for detail <p>, Black for date

Already handled in news.css:

.news-title { color: green; }

.news-detail { color: yellow; }

.news-date { color: black; }

5. Add Component Style: Add custom style class cop-news-component

Already used in both HTL files:

6. Create base page component and add metadata (og:title, og:description, og:image). Link metadata file in base page

Explanation:

We want to support Open Graph tags (used by Facebook, LinkedIn, etc.) for better sharing. We'll use a `head-meta.html` file and include it in our `basepage.html`.

basepage.html

```
<head>
  <sly data-sly-include="head-meta.html" />
</head>
```

head-meta.html

```
<meta property="og:title" content="${currentPage.properties['og:title']}" />
<meta property="og:description" content="${currentPage.properties['og:description']}" />
<meta property="og:image" content="${currentPage.properties['og:image']}" />
```

7. Create custom page properties tab: Global Properties with fields og:title, og:description, og:image

Dialog XML:

Add this under `/apps/your-project/components/page/cq:dialog/content/items/tabs/items/basic/items`

```
<globalProperties
  jcr:primaryType="nt:unstructured"
  sling:resourceType="granite/ui/components/coral/foundation/container"
```

```
    jcr:title="Global Properties">
    <items jcr:primaryType="nt:unstructured">
        <ogTitle
            jcr:primaryType="nt:unstructured"
            sling:resourceType="granite/ui/components/coral/foundation/form/textfield"
            fieldLabel="OG Title"
            name="./og:title"/>
        <ogDescription
            jcr:primaryType="nt:unstructured"
            sling:resourceType="granite/ui/components/coral/foundation/form/textarea"
            fieldLabel="OG Description"
            name="./og:description"/>
        <ogImage
            jcr:primaryType="nt:unstructured"
            sling:resourceType="granite/ui/components/coral/foundation/form/pathfield"
            fieldLabel="OG Image"
            name="./og:image"/>
    </items>
</globalProperties>
```

8. What is extraClientlibs? Add it to the multifield component

Explanation:

`extraClientlibs` is used to load additional client-side resources (JS/CSS) **in the author dialog**, not the page.

How to Add:

In your cq:dialog root node:

extraClientlibs = ["newsroom.dialog"]

Create ClientLib:

Path: /apps/newsroom/clientlibs/dialog

categories = newsroom.dialog

Then add any JS/CSS needed to support dialog functionality.