AEM – Components and Clientlibs

1. Create a New Component with News Title, Details, Date, and Source Using Sling Model

Step 1: Create the Component

- Open CRXDE Lite (http://localhost:4502/crx/de).
- Navigate to /apps/myTraining/components/ and create a new component named newsComponent.
- Inside newsComponent, create a newsComponent.html file and cq:dialog.

Step 2: Create Sling Model

- Create a Java class under com.myTraining.core.models.

Java Code

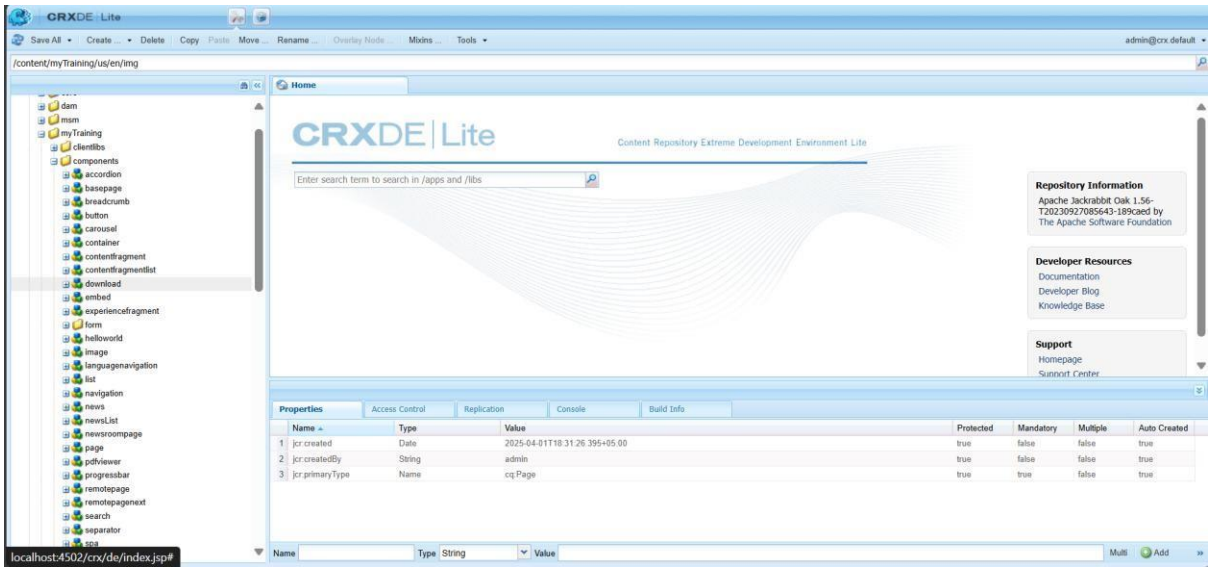
```
package com.myTraining.core.models;


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.injectorspecific.ChildResource;
import java.util.ArrayList;
import java.util.List;


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

```
    private List<NewsItem> newsList = new ArrayList<>();
    public List<NewsItem> getNewsList() {
        return newsList;
    }
    public static class NewsItem {
        @ValueMapValue
        private String newsTitle;

        @ValueMapValue
        private String newsSource;

        public String getNewsTitle() { return newsTitle; }
        public String getNewsSource() { return newsSource; }
    }
}
```

Step 3: Update newsComponent.html

Html code

```
<div class="cop-news-component">
    <h2>${news.newsTitle}</h2>
    <p>${news.newsDetail}</p>
    <p class="date">Published Date: ${news.publishedDate}</p>
    <p>Source: ${news.newsSource}</p>
</div>
```



2. Create a Multi-Field Component (Multiple News) Using Sling Model

Step 1: Create Multi-Field Component

- Create a component named multiNewsComponent under /apps/myTraining/components/.
- Inside cq:dialog, define a multi-field.

Xml code

```
<content jcr:primaryType="nt:unstructured"
sling:resourceType="granite/ui/components/coral/foundation/form/multifield"
fieldLabel="News List"
 name="./newsList">
<field jcr:primaryType="nt:unstructured"
sling:resourceType="granite/ui/components/coral/foundation/form/fieldset">
      <items jcr:primaryType="nt:unstructured">
       <newsTitle jcr:primaryType="nt:unstructured"
          sling:resourceType="granite/ui/components/coral/foundation/form/textfield"
            fieldLabel="News Title"
            name="./newsTitle"/>
```

```
            <newsSource jcr:primaryType="nt:unstructured"
                        sling:resourceType="granite/ui/components/coral/foundation/form/textfield"
                        fieldLabel="News Source"
                        name="./newsSource"/>
        </items>
      </field>
    </content>
```

Step 2: Create Sling Model for Multi-Field Component

Java Code

```
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);

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

    @Inject
    @Default(values = "No Details Available")
```

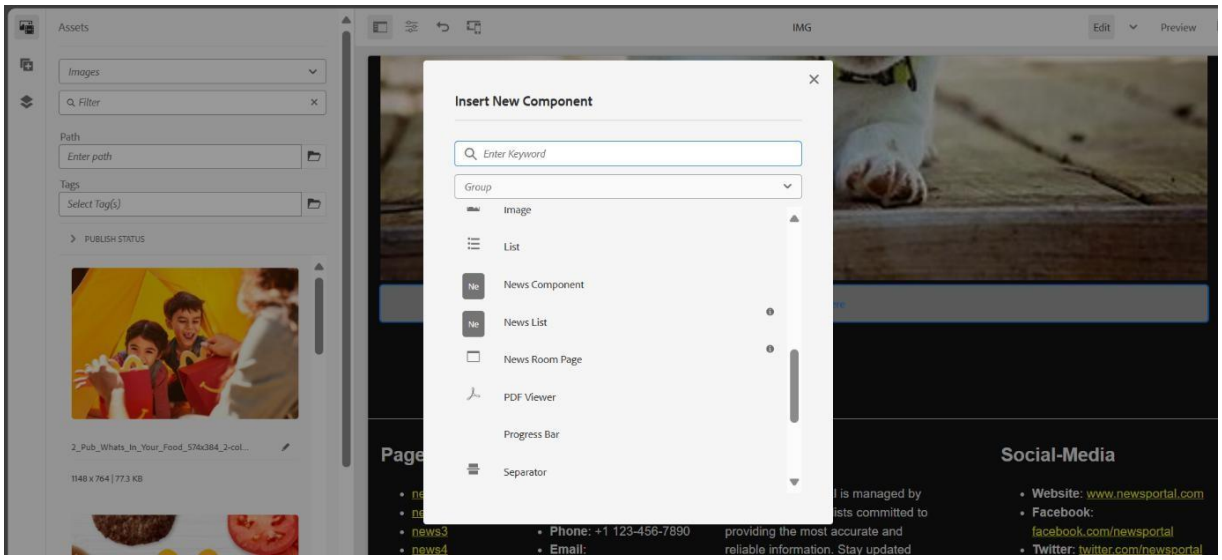
```
    private String detail;

    @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;
    }
}
```

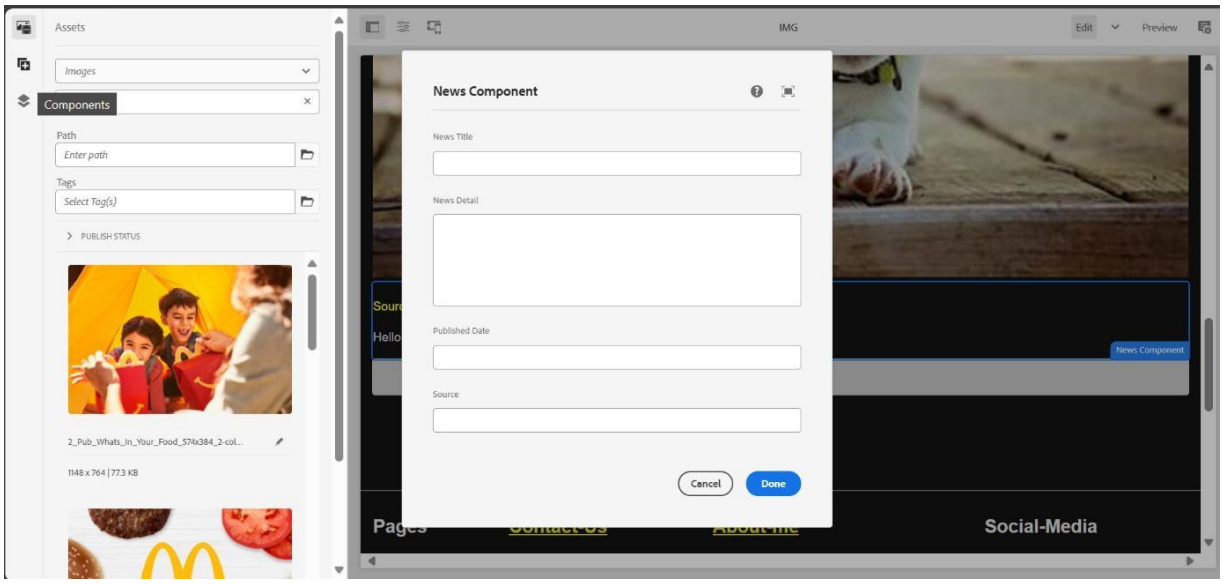
Step 3: Update NewsComponent.html

Html code

```
<div>

    <h2>News List</h2>

    <ul>
```

```
    <sly data-sly-use.multiNews="com.myTraining.core.models.MultiNewsModel">
      <sly data-sly-list.newsItem="${multiNews.newsList}">
        <li>
          <h3>${newsItem.newsTitle}</h3>
          <p>Source: ${newsItem.newsSource}</p>
        </li>
      </sly>
    </sly>
  </ul>
</div>
```





3,4. Create Clientlibs for News Component

- Navigate to /apps/myTraining/clientlibs/ and create a new folder newsClientLib.
- Add the following properties in newsClientLib:
    - jcr:primaryType: cq:ClientLibraryFolder
    - categories: [newsComponent]
- Inside newsClientLib, create css.txt and news.css.
- Add styles in news.css.

Css Code

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

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

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

- Link clientlibs in newsComponent.html.

code

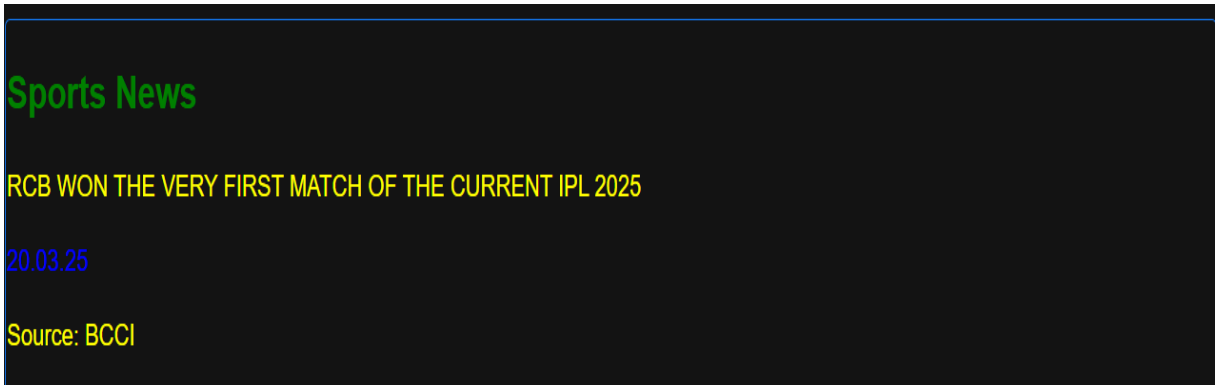
```
<sly data-sly-use.clientlib="/libs/granite/sightly/templates/clientlib.html"/>
<sly data-sly-call="${clientlib.css @ categories='newsComponent'}"/>
```

5. Add Custom Style Name to News Component

- Navigate to newsComponent → cq:editConfig.
- Add the following property.

code

```
<cq:styleGroups
  jcr:primaryType="nt:unstructured">
  <newsStyle
    jcr:primaryType="nt:unstructured"
    cq:styleGroupLabel="News Styles">
    <cop-news-component
      jcr:primaryType="nt:unstructured"
      cq:styleLabel="News Component Style"/>
  </newsStyle>
</cq:styleGroups>
```



6. Create Base Page Component and Add Metadata

- Create basePage under /apps/myTraining/components/.
- Create metadata.html.

code

```
<meta property="og:title" content="${properties.ogTitle}"/>
<meta property="og:description" content="${properties.ogDescription}"/>
<meta property="og:image" content="${properties.ogImage}"/>
```

- Link this in basePage.html.

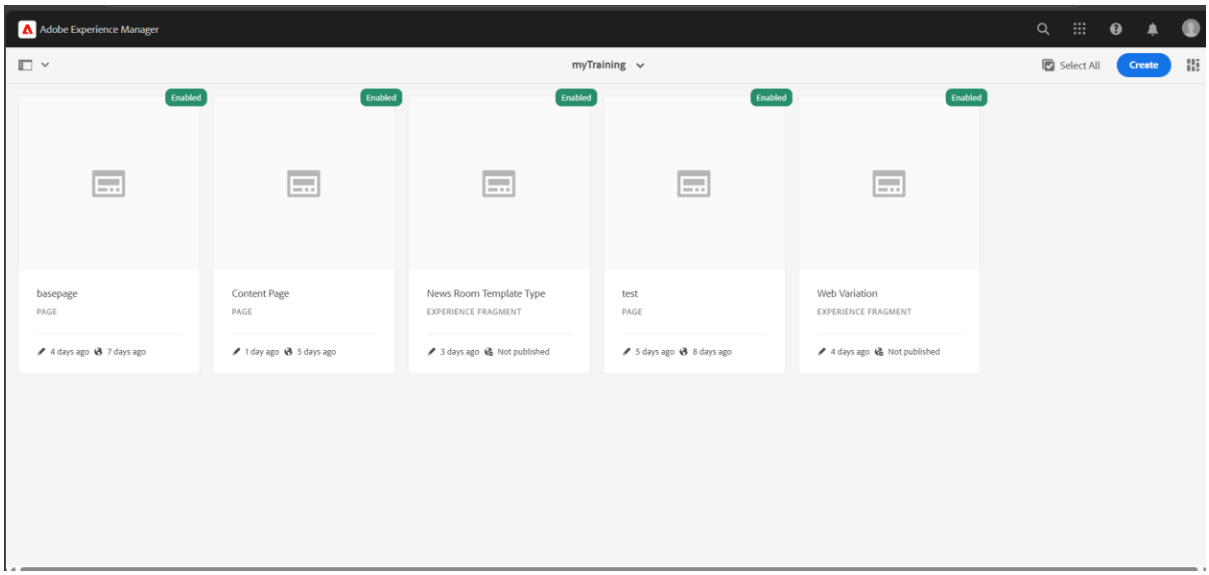
```
<sly data-sly-include="metadata.html"></sly>
```

7. Create Custom Page Properties (Global Properties with og:title, og:description, and og:image path)

- Navigate to basePage → cq:dialog.
- Add the following fields inside cq:dialog

code

```
<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"
  name="./ogImage"/>
```



8. What is extraClientLib and How to Add It to Multi-Field Component?

- extraClientLib is used to include additional client-side libraries dynamically.
- To add it in multiNewsComponent, update cq:editConfig.

code

```
<cq:editConfig
  jcr:primaryType="cq:EditConfig"
  extraClientLibs="[myTraining.extraClientLib]"/>
```