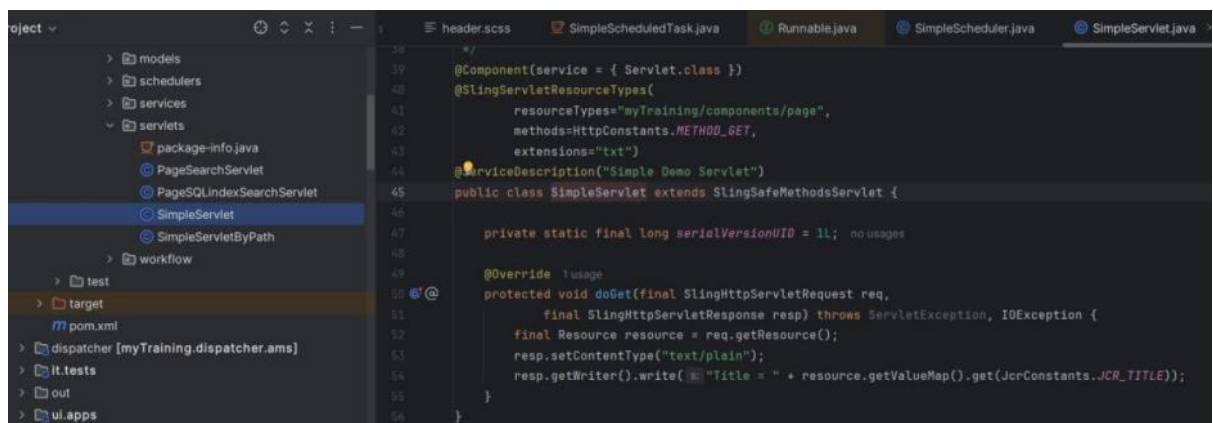


AEM Task – 7: AEM Servlets and Page Management

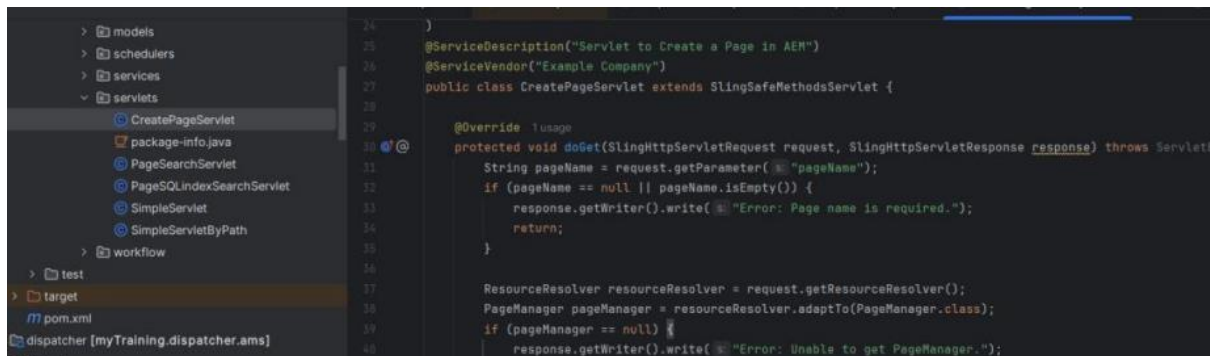
Created SampleServlet (Using resourceType)

- Developed a servlet using **SlingAllMethodsServlet** and registered it with **resourceType**.
- The servlet was designed to handle both **GET** and **POST** requests.
- It processed user inputs and returned appropriate responses based on the requested resource.
- Deployment was completed, and the servlet's registration was verified in the **AEM Web Console**.



Created CreatePageServlet (Using Path)

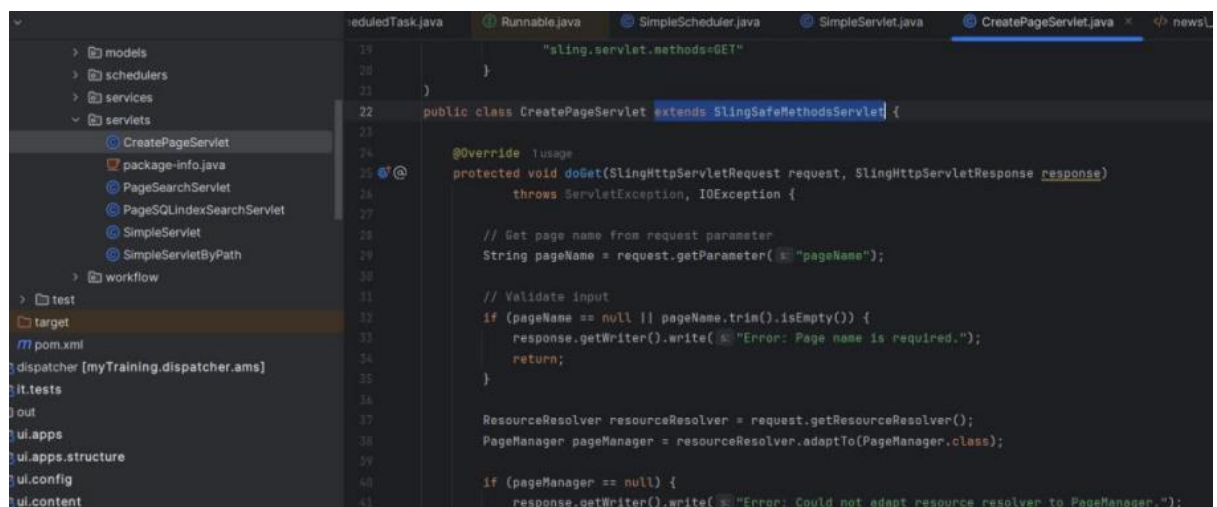
- Developed a servlet using **SlingSafeMethodsServlet** and registered it with a specific path (**/bin/createPage**).
- It handled requests to dynamically create pages in AEM based on user input.
- The servlet received the page name, validated its uniqueness, and created the page within a predefined content structure.
- The **PageManager API** was utilized to perform the page creation.
- The servlet was successfully deployed, tested, and verified.



```
24 )
25 @ServiceDescription("Servlet to Create a Page in AEM")
26 @ServiceVendor("Example Company")
27 public class CreatePageServlet extends SlingSafeMethodsServlet {
28
29     @Override @usage
30     protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException {
31         String pageName = request.getParameter("pageName");
32         if (pageName == null || pageName.isEmpty()) {
33             response.getWriter().write("Error: Page name is required.");
34             return;
35         }
36
37         ResourceResolver resourceResolver = request.getResourceResolver();
38         PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
39         if (pageManager == null) {
40             response.getWriter().write("Error: Unable to get PageManager.");
41         }
42     }
43 }
```

Accepted User Input for Page Creation

- The servlet was configured to accept a **page name** from the user.
- A validation process was implemented to ensure **duplicate pages** were not created.
- Once validated, the **PageManager API** was used to dynamically create the new page.
- After execution, a **confirmation message** was logged and displayed.

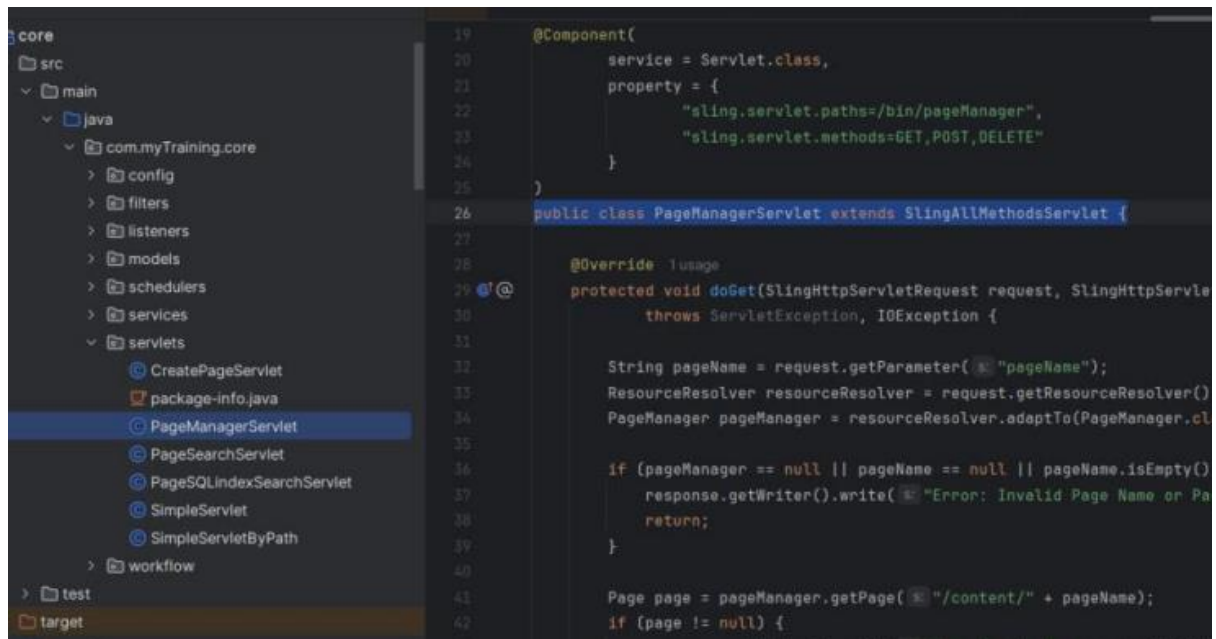


```
19 "sling.servlet.methods=GET"
20 }
21 }
22 public class CreatePageServlet extends SlingSafeMethodsServlet {
23
24     @Override @usage
25     protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
26         throws ServletException, IOException {
27
28         // Get page name from request parameter
29         String pageName = request.getParameter("pageName");
30
31         // Validate input
32         if (pageName == null || pageName.trim().isEmpty()) {
33             response.getWriter().write("Error: Page name is required.");
34             return;
35         }
36
37         ResourceResolver resourceResolver = request.getResourceResolver();
38         PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
39
40         if (pageManager == null) {
41             response.getWriter().write("Error: Could not adapt resource resolver to PageManager.");
42         }
43     }
44 }
```

Managed Pages Using PageManager API

- The **PageManager API** was used to create, retrieve, modify, and delete AEM pages.
- Appropriate permissions were configured to enable the servlet to perform these operations.

- Functional testing was carried out to verify the successful execution of page-related tasks.



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes a 'core' module with 'src' and 'test' directories. Under 'src/main/java', there is a package 'com.myTraining.core' containing several servlets, including 'PageManagerServlet'. The code editor displays the implementation of 'PageManagerServlet', which is a SlingAllMethodsServlet. It has a @Component annotation with service = Servlet.class and property = { "sling.servlet.paths=/bin/pageManager", "sling.servlet.methods=GET,POST,DELETE" }. The doGet method is overridden to handle GET requests, retrieving a page name from the request and returning the page content.

```

19 @Component(
20     service = Servlet.class,
21     property = {
22         "sling.servlet.paths=/bin/pageManager",
23         "sling.servlet.methods=GET,POST,DELETE"
24     }
25 )
26 public class PageManagerServlet extends SlingAllMethodsServlet {
27
28     @Override @Usage
29     protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
30         throws ServletException, IOException {
31
32         String pageName = request.getParameter("pageName");
33         ResourceResolver resourceResolver = request.getResourceResolver();
34         PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
35
36         if (pageManager == null || pageName == null || pageName.isEmpty()) {
37             response.getWriter().write("Error: Invalid Page Name or Page Not Found");
38             return;
39         }
40
41         Page page = pageManager.getPage("/content/" + pageName);
42         if (page != null) {

```

Created SearchServlet for Content Search

- Developed and registered a servlet at **/bin/search** to enable content search functionality.
- Integrated the **Query Builder API** using **PredicateMap** to execute structured queries.
- Configured the search functionality with parameters such as:
 - Path filtering** to restrict searches to **/content/my-site**.
 - Page type filtering** to search only within **cq:Page** nodes.
 - Full-text search** to retrieve relevant content based on user input.
- The servlet was deployed, tested, and verified to return accurate search results.

