AEM Training - Day 7 Assignment (26-03-2025)

1. Create SampleServlet Extending SlingAllMethodsServlet:

Create an AEM servlet that is registered using resourceType. This servlet will handle HTTP GET requests and log the execution.

Code Snippet (SampleServlet.java):

```
@Component(service = Servlet.class, property = {

   "sling.servlet.resourceTypes=myapp/components/page",

   "sling.servlet.methods=GET"

})

public class SampleServlet extends SlingAllMethodsServlet {

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

   @Override

   protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
throws ServletException, IOException {

response.getWriter().write("Sample Servlet Executed");

      LOG.info("Sample Servlet Executed");

   }

}
```

2. Create CreatePageServlet Extending SlingSafeMethodsServlet:

This servlet is registered using a specific path and is designed to create pages based on user input.

Code Snippet (CreatePageServlet.java):

```
@Component(service = Servlet.class, property = {

   "sling.servlet.paths=/bin/createPage",

   "sling.servlet.methods=POST"

})

public class CreatePageServlet extends SlingSafeMethodsServlet {
  @Reference     private ResourceResolverFactory
resolverFactory;
  @Override
  protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response)
throws ServletException, IOException {
      String pageName = request.getParameter("pageName");        try (ResourceResolver
resolver = resolverFactory.getServiceResourceResolver(null)) {           PageManager
pageManager = resolver.adaptTo(PageManager.class);           if (pageManager != null) {
            Page newPage = pageManager.create("/content/mysite", pageName,
"/apps/myapp/templates/page", pageName);
            resolver.commit();              response.getWriter().write("Page created
successfully: " + newPage.getPath());
          } else {
            response.getWriter().write("PageManager is null");
          }
      } catch (PersistenceException e) {
response.getWriter().write("Page creation failed: " + e.getMessage());
      }
  }
}
```

3.Use PageManager API to Create Pages:

This servlet demonstrates how to dynamically create pages using the PageManager API.

Steps:

1. Retrieve the PageManager instance from the ResourceResolver.
2. Call the create() method, passing the parent path, page name, template path, and title.
3. Commit the changes using resolver.commit().

Code Snippet:

```
PageManager pageManager = resolver.adaptTo(PageManager.class); if
(pageManager != null) {
   Page newPage = pageManager.create("/content/mysite", pageName,
"/apps/myapp/templates/page", pageName);
   resolver.commit();
}
```

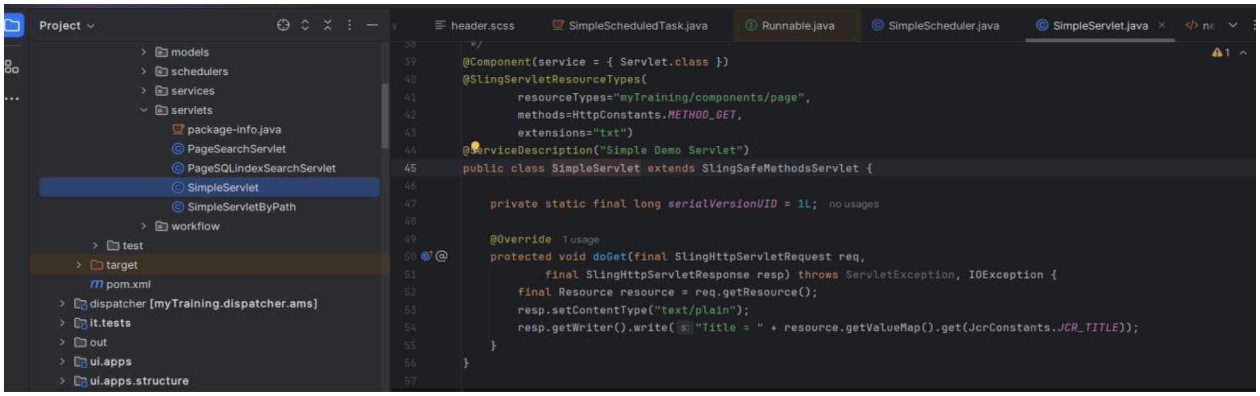
4. Create SearchServlet Using PredicateMap:

This AEM servlet allows you to search for content using the QueryBuilder API and a PredicateMap.

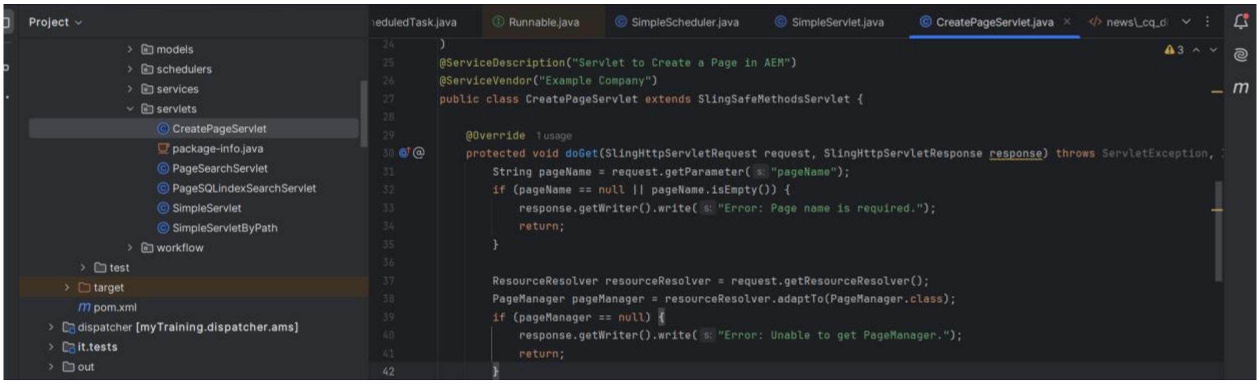
Code Snippet (SearchServlet.java):

```
@Component(service = Servlet.class, property = {
   "sling.servlet.paths=/bin/searchContent",
   "sling.servlet.methods=GET"
})
public class SearchServlet extends SlingSafeMethodsServlet {
   @Reference    private
QueryBuilder queryBuilder;
@Override

protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
throws ServletException, IOException {
      Map<String, String> map = new HashMap<>();
map.put("path", "/content/mysite");
map.put("type", "cq:Page");       map.put("p.limit",
"-1");
      Query query = queryBuilder.createQuery(PredicateGroup.create(map),
request.getResourceResolver().adaptTo(Session.class));       SearchResult
result = query.getResult();        for (Hit hit : result.getHits()) {
response.getWriter().write(hit.getPath() + "\n");
        }
    }
}
```

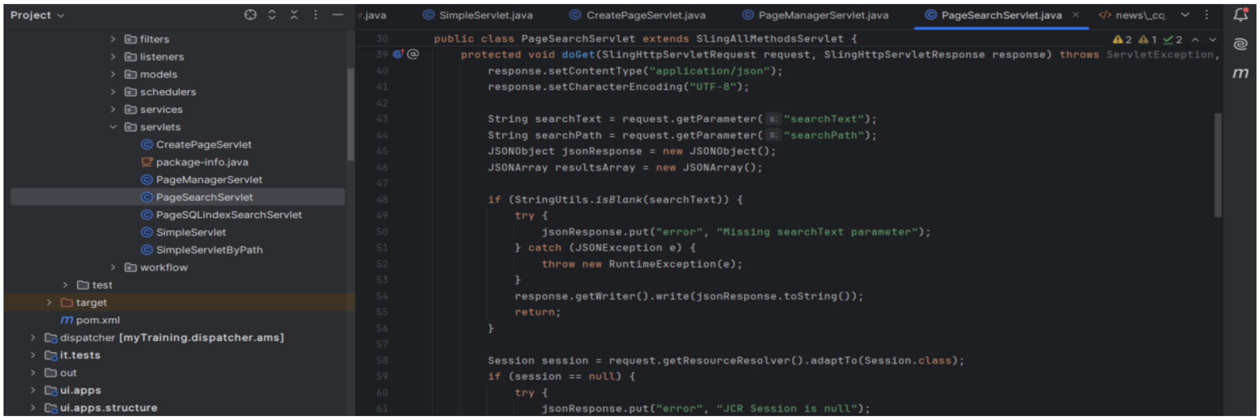
Screenshots:



```java
 */
@Component(service = { Servlet.class })
@SlingServletResourceTypes(
        resourceTypes="myTraining/components/page",
        methods=HttpConstants.METHOD_GET,
        extensions="txt")
@ServiceDescription("Simple Demo Servlet")
public class SimpleServlet extends SlingSafeMethodsServlet {

    private static final long serialVersionUID = 1L;  no usages

    @Override  1 usage
    protected void doGet(final SlingHttpServletRequest req,
            final SlingHttpServletResponse resp) throws ServletException, IOException {
        final Resource resource = req.getResource();
        resp.setContentType("text/plain");
        resp.getWriter().write("Title = " + resource.getValueMap().get(JcrConstants.JCR_TITLE));
    }
}
```



```java
)
@ServiceDescription("Servlet to Create a Page in AEM")
@ServiceVendor("Example Company")
public class CreatePageServlet extends SlingSafeMethodsServlet {

    @Override  1 usage
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException, 
        String pageName = request.getParameter("pageName");
        if (pageName == null || pageName.isEmpty()) {
            response.getWriter().write("Error: Page name is required.");
            return;
        }

        ResourceResolver resourceResolver = request.getResourceResolver();
        PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
        if (pageManager == null) {
            response.getWriter().write("Error: Unable to get PageManager.");
            return;
        }
```



```java
public class PageSearchServlet extends SlingAllMethodsServlet {
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException,
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");

        String searchText = request.getParameter("searchText");
        String searchPath = request.getParameter("searchPath");
        JSONObject jsonResponse = new JSONObject();
        JSONArray resultsArray = new JSONArray();

        if (StringUtils.isBlank(searchText)) {
            try {
                jsonResponse.put("error", "Missing searchText parameter");
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
            response.getWriter().write(jsonResponse.toString());
            return;
        }

        Session session = request.getResourceResolver().adaptTo(Session.class);
        if (session == null) {
            try {
                jsonResponse.put("error", "JCR Session is null");
```