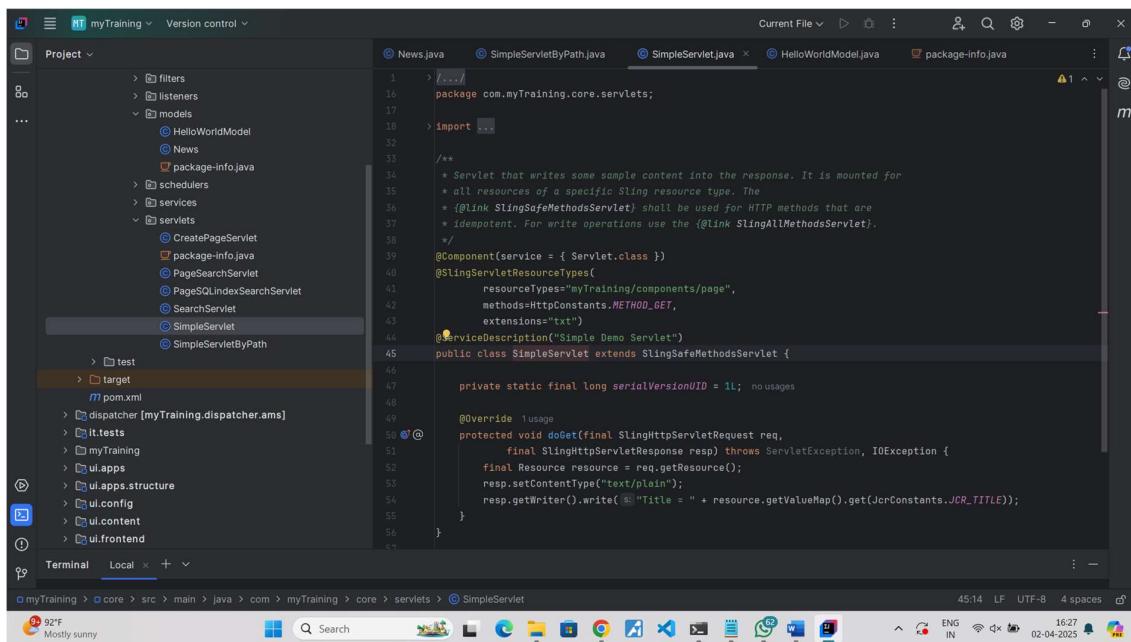


# **AEM ASSIGNMENT DAY-7**

## **1. Create SampleServlet (Extending SlingAllMethodsServlet and Registering Using ResourceType)**

### **📌 Steps:**

1. Create SampleServlet.java in /apps/myTraining/core/servlets/
2. Extend SlingAllMethodsServlet
3. Register it using @SlingServletResourceTypes
4. Implement doGet and doPost methods



The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure under "myTraining". Key components include "filters", "listeners", "models" (with "HelloWorldModel" and "News" selected), "services", "schedulers", and "servlets" (containing "CreatePageServlet", "PageSearchServlet", "PageSQLIndexSearchServlet", "SearchServlet", "SimpleServlet", and "SimpleServletByPath").
- Code Editor:** Displays the content of "SimpleServlet.java". The code defines a class "SimpleServlet" that extends "SlingSafeMethodsServlet". It includes annotations for component registration and service description, and implements the "doGet" method.
- Terminal:** Shows the command line path: myTraining > core > src > main > java > com > myTraining > core > servlets > SimpleServlet.
- System Tray:** Shows system status including battery level (92%), weather (Mostly sunny), and system time (16:27).

## **2. Create CreatePageServlet (Extending SlingSafeMethodsServlet and Registering Using Path)**

### **📌 Steps:**

1. Create CreatePageServlet.java in /apps/myTraining/core/servlets/
2. Extend SlingSafeMethodsServlet
3. Register it using @SlingServletPaths
4. Use PageManager API to create a page

```

1 package com.myTraining.core.servlets;
2 import ...
3
4 @Component(service = Servlet.class, property = {
5     "sling.servlet.paths=/bin/createPage",
6     "sling.servlet.methods=GET"
7 })
8 public class CreatePageServlet extends SlingSafeMethodsServlet {
9
10    @Override void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException {
11        ResourceResolver resourceResolver = request.getResourceResolver();
12        Session session = resourceResolver.adaptTo(Session.class);
13
14        if (session == null) {
15            response.getWriter().write("Session not available");
16            return;
17        }
18
19        PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
20        if (pageManager == null) {
21            response.getWriter().write("PageManager not available");
22            return;
23        }
24
25        String parentPath = "/content/mySite"; // Parent path where the page will be created
26        String newPageName = "new-page"; // Name of the new page
27        String templatePath = "/conf/mySite/settings/wcm/templates/basic-template"; // Template for the new page
28        String title = "New Page Title"; // Title of the new page
29
30        ...
31    }
32
33    ...
34
35    ...
36
37    ...
38
39    ...
40
41    ...
42
43    ...
44
45    ...
46
47

```

### 3. Create SearchServlet (Using Query Builder API and PredicateMap)

#### 📌 Steps:

1. Create SearchServlet.java in /apps/myTraining/core/servlets/
2. Extend SlingSafeMethodsServlet
3. Use QueryBuilder API to fetch pages
4. Return results in JSON format

```

1 package com.myTraining.core.servlets;
2 import ...
3
4 @Component(service = Servlet.class, property = {
5     "sling.servlet.paths=/bin/searchContent",
6     "sling.servlet.methods=GET"
7 })
8 public class SearchServlet extends SlingSafeMethodsServlet {
9
10    @Reference usage
11    private QueryBuilder queryBuilder;
12
13    @Override void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws ServletException {
14        ResourceResolver resourceResolver = request.getResourceResolver();
15        Session session = resourceResolver.adaptTo(Session.class);
16
17        if (session == null) {
18            response.getWriter().write("Session not available");
19            return;
20        }
21
22        Map<String, String> map = new HashMap<>();
23        map.put("path", "/content/mySite"); // Define search root
24        map.put("type", "cq:Page"); // Look for pages
25        map.put("p.limit", "1"); // Fetch all results
26
27        Query query = queryBuilder.createQuery(PredicateGroup.create(map), session);
28        SearchResult result = query.getResult();
29
30        response.getWriter().write(result.toString());
31
32    }
33
34    ...
35
36    ...
37
38    ...
39
39    ...
40
41
42
43
44
45
46
47

```

