26.3.25

Implementing AEM Servlets for Page Creation and Content Search

1. Introduction

This guide walks you through the process of creating three essential servlets in **Adobe Experience Manager (AEM)**. These servlets will help you automate page creation and perform content searches efficiently:

- **SampleServlet**: A basic servlet using SlingAllMethodsServlet, registered via resourceType.
- **CreatePageServlet**: A servlet that dynamically creates pages using SlingSafeMethodsServlet, registered via path.
- **SearchServlet**: A servlet designed to search content with the help of PredicateMap and AEM's Query Builder API.

---

2. Prerequisites

Before we dive into the implementation, make sure you have the following ready:

- A running AEM instance (preferably AEM 6.x or above).
- Basic knowledge of **Java**, **Servlets**, and **AEM development**.
- An AEM project set up using **Maven**.
- Access to **CRXDE Lite** and AEM's **OSGi Configuration Console**.

---

3. Creating SampleServlet

The **SampleServlet** is a simple servlet designed to demonstrate the basics of servlet implementation in AEM.

```
import org.apache.sling.api.servlets.SlingAllMethodsServlet;

import org.apache.sling.api.servlets.SlingSafeMethodsServlet;

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

import org.apache.sling.api.servlets.Servlet;

import org.apache.sling.api.servlets.SlingHttpServletRequest;

import org.apache.sling.api.servlets.SlingHttpServletResponse;

import org.osgi.service.component.annotations.Component;

import javax.servlet.Servlet;
```

```
import java.io.IOException;
@Component(
    service = Servlet.class,
    property = {
        "sling.servlet.resourceTypes=myproject/components/sample",
        "sling.servlet.methods=GET"
    }
)
public class SampleServlet extends SlingAllMethodsServlet {
    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
        response.getWriter().write("Hello from SampleServlet!");
    }
}
```

Output:

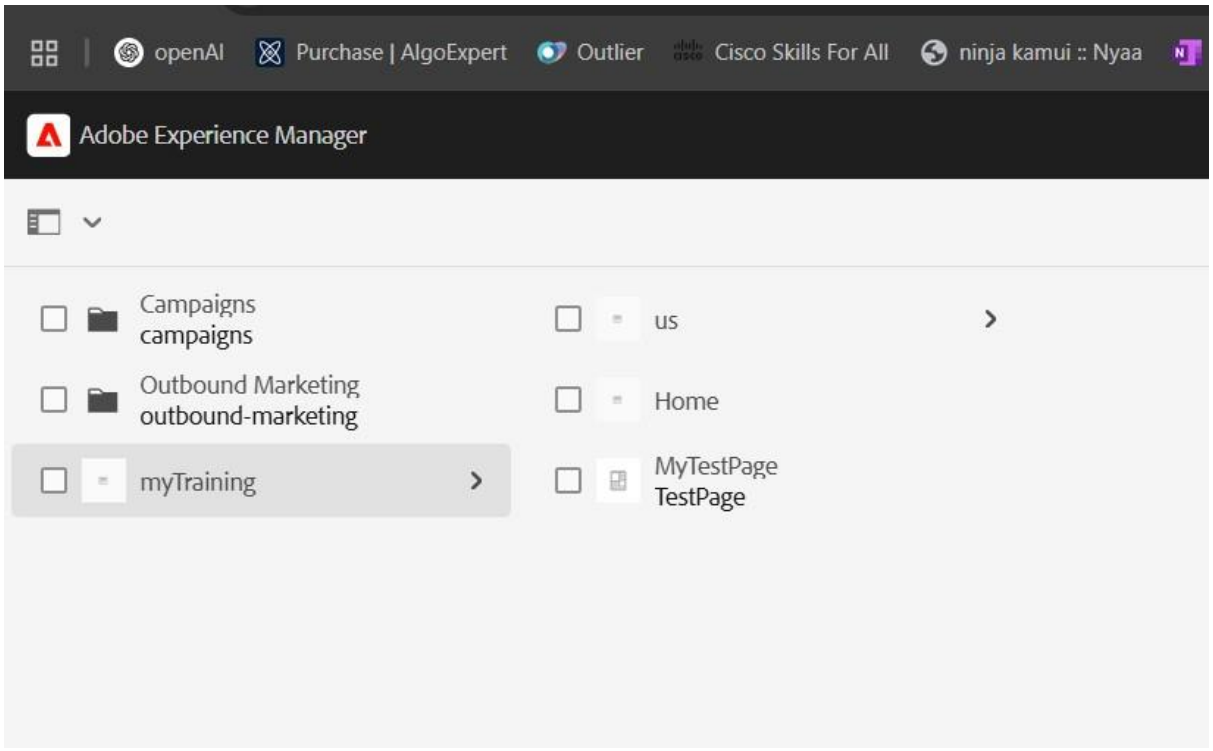
URL Format: http://localhost:4502/content/samplepage.sample.json

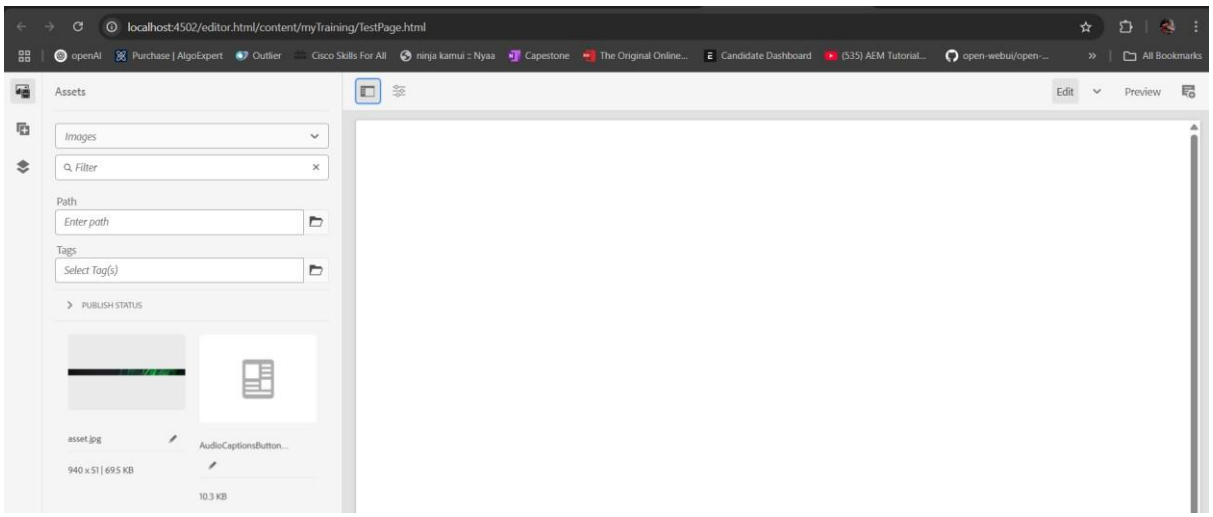
4. Creating CreatePageServlet

```
import com.adobe.granite.ui.components.ds.DataSource;
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.api.servlets.SlingHttpServletRequest;
import org.apache.sling.api.servlets.SlingHttpServletResponse;
import org.apache.sling.api.servlets.Servlet;
import org.apache.sling.api.resource.ResourceResolver;
import org.apache.sling.api.servlets.ServletResolverConstants;
import org.osgi.service.component.annotations.Component;
import org.apache.sling.api.resource.ValueMap;
import com.day.cq.commons.Externalizer;
```

```
import com.day.cq.wcm.api.Page;
import com.day.cq.wcm.api.PageManager;
import javax.servlet.Servlet;
import java.io.IOException;
@Component(
    service = Servlet.class,
    property = {
        "sling.servlet.paths=/bin/create-page",
        "sling.servlet.methods=POST"
    }
)
public class CreatePageServlet extends SlingSafeMethodsServlet {
    @Override
    protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
        String pageName = request.getParameter("pageName");
        if (pageName == null || pageName.trim().isEmpty()) {
            response.getWriter().write("Error: Page name is required!");
            return;
        }
ResourceResolver resourceResolver = request.getResourceResolver();
   PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
if (pageManager != null) {
            try {
                Page newPage = pageManager.create("/content/myproject", pageName,
"myproject/templates/basic", pageName);
                response.getWriter().write("Page successfully created at: " + newPage.getPath());
            } catch (Exception e) {
                response.getWriter().write("Failed to create page: " + e.getMessage());
            }
```

```
        } else {
            response.getWriter().write("Error: PageManager is not available.");
        }
    }
}
```



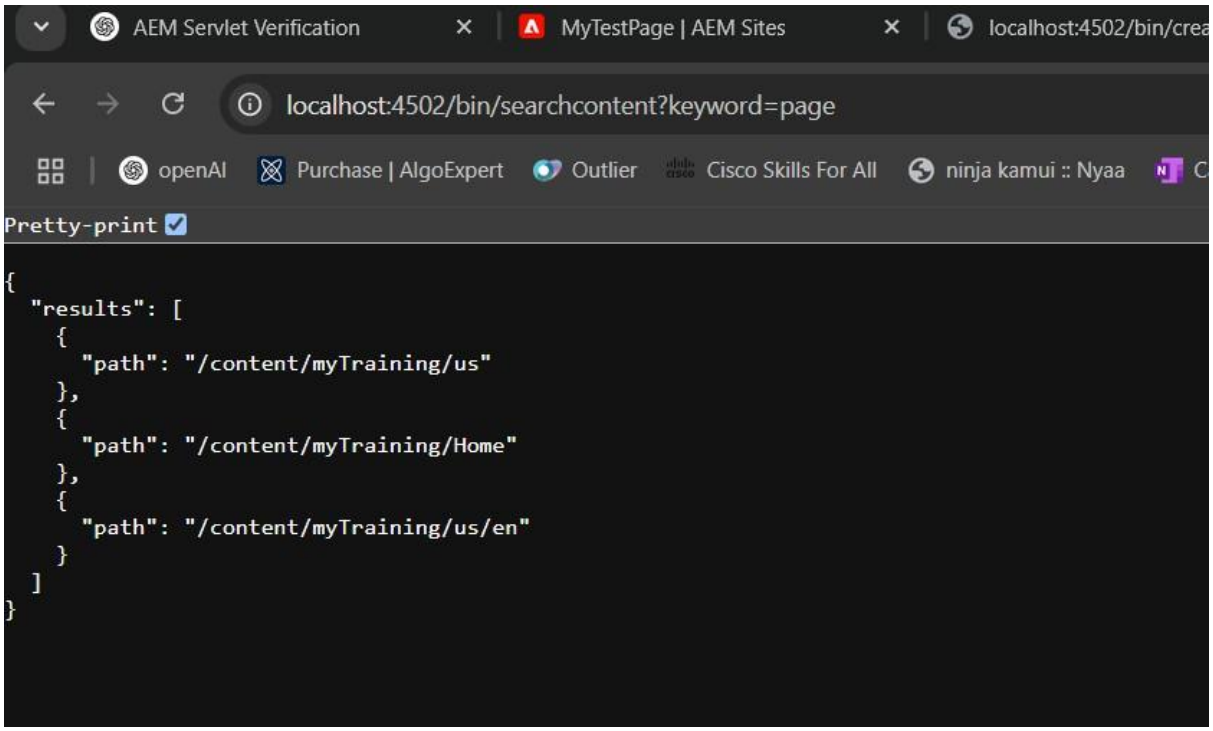


5. Creating SearchServlet :

```
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.servlets.SlingHttpServletRequest;
```

```
import org.apache.sling.api.servlets.SlingHttpServletResponse;
import org.apache.sling.api.resource.ResourceResolver;
import org.apache.sling.api.servlets.Servlet;
import org.osgi.service.component.annotations.Component;
import org.apache.sling.api.resource.ValueMap;
import org.apache.sling.api.resource.Resource;
import com.adobe.granite.ui.components.ds.DataSource;
import org.apache.sling.query.QueryBuilder;
import org.apache.sling.query.Query;
import org.apache.sling.query.Result;
import javax.servlet.Servlet;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
@Component(
    service = Servlet.class,
    property = {
        "sling.servlet.paths=/bin/search-content",
        "sling.servlet.methods=GET"
    }
)
public class SearchServlet extends SlingSafeMethodsServlet {
 private final QueryBuilder queryBuilder;
 public SearchServlet(QueryBuilder queryBuilder) {
        this.queryBuilder = queryBuilder;
    }
@Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
```

```
        String searchQuery = request.getParameter("query");
        if (searchQuery == null || searchQuery.isEmpty()) {
            response.getWriter().write("Please provide a valid search query.");
            return;
        }
Map<String, String> searchCriteria = new HashMap<>();
        searchCriteria.put("path", "/content");
        searchCriteria.put("type", "cq:Page");
        searchCriteria.put("fulltext", searchQuery);
Query query = queryBuilder.createQuery(searchCriteria,
request.getResourceResolver().adaptTo(Session.class));
        Result result = query.getResult();
response.getWriter().write("Found " + result.getHits().size() + " matching pages.");
    }
}
```



6. Conclusion

This guide demonstrated how to:

- Implement **SampleServlet** for basic servlet operations.
- Create **CreatePageServlet** for dynamic page creation.
- Develop **SearchServlet** for content searching using Query Builder.