26/03/2025 - TASKS

Table of Contents

---

1. Create SampleServlet using SlingAllMethodServlet

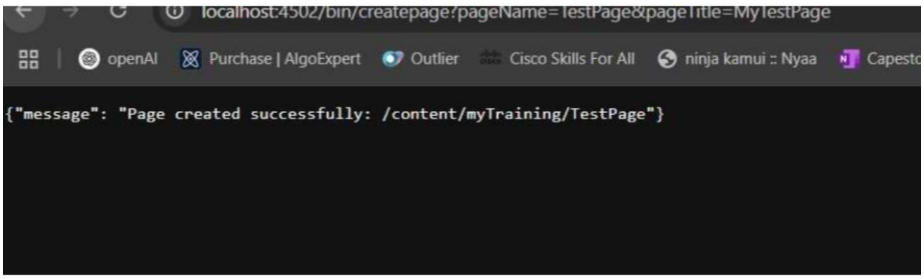
Steps:

- **Extend SlingAllMethodsServlet.**
- **Annotate with @SlingServletResourceTypes.**
- **Implement doGet and doPost methods.**
- **Deploy and test the servlet.**

Example Code:

```
package com.myTraining.core.servlets;

import com.day.cq.commons.jcr.JcrConstants;
import org.apache.sling.api.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
import org.apache.sling.api.resource.Resource;
import org.apache.sling.api.servlets.HttpConstants;
import org.apache.sling.api.servlets.SlingAllMethodsServlet;
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.servlets.annotations.SlingServletResourceTypes;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.propertytypes.ServiceDescription;

import javax.servlet.Servlet;
import javax.servlet.ServletException;
import java.io.IOException;
```

```
/**
 * Servlet that writes some sample content into the response. It is mounted for
 * all resources of a specific Sling resource type. The
 * {@link SlingSafeMethodsServlet} shall be used for HTTP methods that are
 * idempotent. For write operations use the {@link SlingAllMethodsServlet}.
 */
@Component(service = { Servlet.class })
@SlingServletResourceTypes(
      resourceTypes="myTraining/components/page",
      methods=HttpConstants.METHOD_GET,
      extensions="txt")
@ServiceDescription("Simple Demo Servlet")
public class SimpleServlet extends SlingAllMethodsServlet {

   private static final long serialVersionUID = 1L;

   @Override
   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));
   }
}
```



---

2. Create CreatePageServlet using SlingSafeMethodServlet

Steps:

- **Extend SlingSafeMethodsServlet.**
- **Use @SlingServletPaths annotation.**
- **Implement logic for creating pages dynamically.**
- **Deploy and test servlet functionality.**

Example Code:

```
@Component(service = Servlet.class, property = {"sling.servlet.paths=/bin/createPage"})
public class CreatePageServlet extends SlingSafeMethodsServlet {
   @Override
   protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws IOException {
      response.getWriter().write("Page creation logic");
   }
}
```

---

3. Create AEM pages dynamically using servlet

Steps:

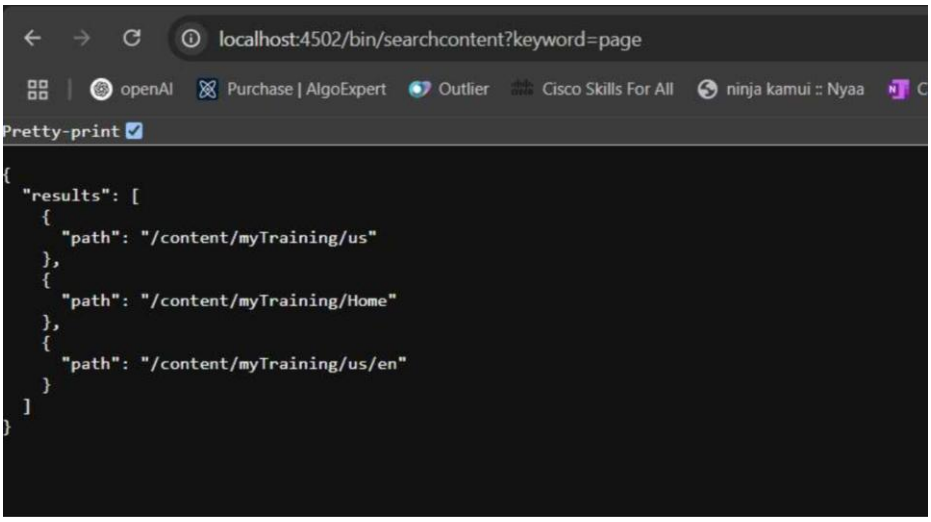
- **Accept page name via request parameters.**
- **Invoke CreatePageServlet to generate a new page.**
- **Validate that the page appears under the specified content hierarchy.**

---

4. Use Page Manager APIs for page creation

Steps:

- **Obtain PageManager from ResourceResolver.**
- **Ensure correct permissions with system users.**
- **Assign a suitable template and components to the page.**
- **Verify the page creation process through logs and UI.**

- 

---

5. Create SearchServlet using PredicateMap for content search

Steps:

- **Extend SlingSafeMethodsServlet for search functionality.**
- **Register servlet at /bin/searchServlet.**
- **Implement query logic using PredicateMap and QueryBuilder.**
- **Process and return search results in JSON format.**

Example Code:

```
package com.myTraining.core.servlets;

import com.day.cq.search.Query;
import com.day.cq.search.QueryBuilder;
import com.day.cq.search.result.Hit;
import com.day.cq.search.result.SearchResult;
import com.day.cq.search.PredicateGroup;
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
import org.apache.sling.api.resource.ResourceResolver;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Reference;
```

```
import org.osgi.framework.Constants;

import javax.jcr.Session;
import javax.servlet.Servlet;
import javax.servlet.ServletException;
import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Component(
        service = {Servlet.class},
        property = {
                Constants.SERVICE_DESCRIPTION + "= Search Content Servlet",
                "sling.servlet.paths=/bin/searchContent",
                "sling.servlet.methods=GET"
        }
)
public class SearchServlet extends SlingSafeMethodsServlet {

    @Reference
    private QueryBuilder queryBuilder; // Inject QueryBuilder service

    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)
throws ServletException, IOException {
        response.setContentType("text/plain");
        String searchText = request.getParameter("query");

        if (searchText == null || searchText.isEmpty()) {
            response.getWriter().write("No search query provided");
            return;
        }

        ResourceResolver resourceResolver = request.getResourceResolver();
        Session session = resourceResolver.adaptTo(Session.class); // Get JCR Session

        if (session == null) {
```

```
        response.getWriter().write("JCR Session is null");
        return;
    }

    if (queryBuilder == null) {
        response.getWriter().write("QueryBuilder service is unavailable");
        return;
    }

    // Build search predicates
    Map<String, String> predicates = new HashMap<>();
    predicates.put("path", "/content/myTraining");  // Change this to your content root
    predicates.put("type", "cq:Page");
    predicates.put("fulltext", searchText);
    predicates.put("p.limit", "10"); // Limit search results

    try {
        Query query = queryBuilder.createQuery(PredicateGroup.create(predicates), session);
        SearchResult result = query.getResult();

        List<Hit> hits = result.getHits();
        if (hits.isEmpty()) {
            response.getWriter().write("No results found for: " + searchText);
        } else {
            for (Hit hit : hits) {
                response.getWriter().write("Found page: " + hit.getPath() + "\n");
            }
        }
    } catch (Exception e) {
        response.getWriter().write("Error executing search query: " + e.getMessage());
    }
}
```