Date: 26-03-25

Creating SampleServlet (Extending SlingAllMethodsServlet)

This servlet handles both GET and POST requests and is registered using a resourceType.

Implementation Steps:

1. Extend the SlingAllMethodsServlet class.
2. Register it using @SlingServletResourceTypes.
3. Implement request handling logic.

SampleServlet.java

```
package com.example.core.servlets;

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

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

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

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

import javax.servlet.Servlet;

import java.io.IOException;

@Component(service = Servlet.class)

@SlingServletResourceTypes(

        resourceTypes = "example/components/page",

        methods = {"GET", "POST"},

        extensions = "json"

)

public class SampleServlet extends SlingAllMethodsServlet {

    @Override

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

        response.setContentType("application/json");

        response.getWriter().write("{\"message\":\"GET request processed successfully\"}");
    }

    @Override

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

        response.setContentType("application/json");

        response.getWriter().write("{\"message\":\"POST request processed successfully\"}");

    }

}
```

Creating CreatePageServlet (Extending SlingSafeMethodsServlet)

This servlet is responsible for creating pages and is registered using a path.

Implementation Steps:

1. Extend SlingSafeMethodsServlet.
2. Register using @SlingServletPaths.
3. Implement page creation logic.

CreatePageServlet.java

```
package com.example.core.servlets;

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

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

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

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

import com.day.cq.wcm.api.PageManager;

import com.day.cq.wcm.api.Page;

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

import javax.servlet.Servlet;

import java.io.IOException;

@Component(service = Servlet.class)
```

```
@SlingServletPaths("/bin/createPage")
public class CreatePageServlet extends SlingSafeMethodsServlet {
    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
        response.setContentType("application/json");
        String pageName = request.getParameter("name");
        String parentPath = "/content/example"; // Modify as needed
        String templatePath = "/conf/example/settings/wcm/templates/default"; // Modify
based on your setup
        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(parentPath, pageName, templatePath,
pageName, true);
                if (newPage != null) {
                    response.getWriter().write("{\"success\":\"Page created successfully at: " +
newPage.getPath() + "\"}");
                } else {
                    response.getWriter().write("{\"error\":\"Failed to create page\"}");
                }
            } catch (Exception e) {
                response.getWriter().write("{\"error\":\"Exception occurred: " + e.getMessage() +
"\"}");
```

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

Creating SearchServlet Using PredicateMap for Content Search

This servlet will search for content pages using PredicateMap and QueryBuilder.

Implementation Steps:

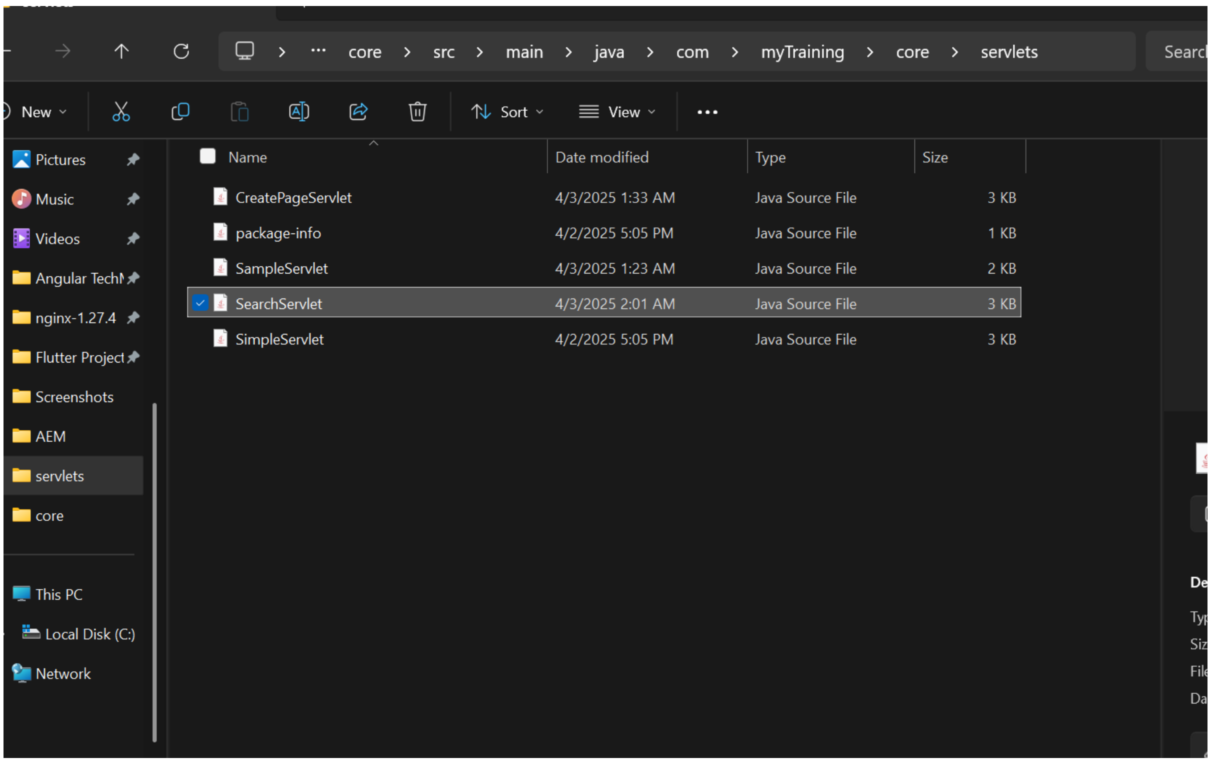
1. Utilize QueryBuilder API.
2. Define search parameters in PredicateMap.
3. Execute query and return search results.

SearchServlet.java

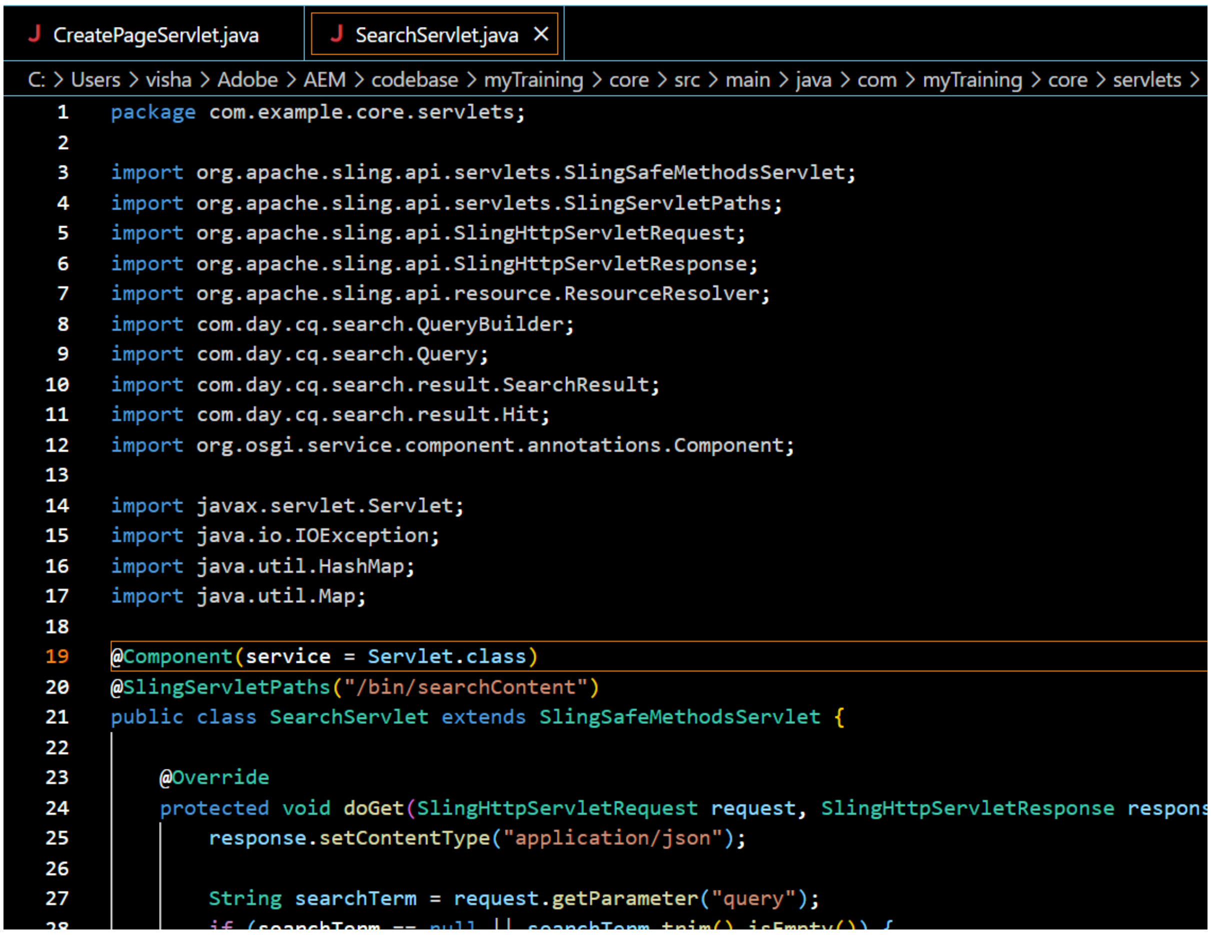
```
package com.example.core.servlets;

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

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

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

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

import com.day.cq.search.QueryBuilder;

import com.day.cq.search.Query;

import com.day.cq.search.result.SearchResult;

import com.day.cq.search.result.Hit;

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

import javax.servlet.Servlet;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;
```

```
@Component(service = Servlet.class)
@SlingServletPaths("/bin/searchContent")
public class SearchServlet extends SlingSafeMethodsServlet {
  @Override
  protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
    response.setContentType("application/json");
    String searchTerm = request.getParameter("query");
    if (searchTerm == null || searchTerm.trim().isEmpty()) {
      response.getWriter().write("{\"error\":\"Search term is required\"}");
      return;
    }
    ResourceResolver resourceResolver = request.getResourceResolver();
    QueryBuilder queryBuilder = resourceResolver.adaptTo(QueryBuilder.class);
    if (queryBuilder != null) {
      Map<String, String> predicateMap = new HashMap<>();
      predicateMap.put("type", "cq:Page"); // Search only pages
      predicateMap.put("fulltext", searchTerm);
      predicateMap.put("path", "/content/example"); // Modify path as needed
      predicateMap.put("p.limit", "10"); // Limit results

      Query query = queryBuilder.createQuery(predicateMap, resourceResolver);
      SearchResult result = query.getResult();
      StringBuilder jsonResponse = new StringBuilder("{\"results\":[");
      boolean first = true;
      for (Hit hit : result.getHits()) {
        if (!first) {
          jsonResponse.append(",");
```

```
                }
                jsonResponse.append("{\"path\":\"").append(hit.getPath()).append("\"}");
                first = false;
            }
            jsonResponse.append("]}");
            response.getWriter().write(jsonResponse.toString());
        } else {
            response.getWriter().write("{\"error\":\"QueryBuilder not available\"}");
        }
    }
}
```



Name	Date modified	Type	Size
CreatePageServlet	4/3/2025 1:33 AM	Java Source File	3 KB
package-info	4/2/2025 5:05 PM	Java Source File	1 KB
SampleServlet	4/3/2025 1:23 AM	Java Source File	2 KB
SearchServlet	4/3/2025 2:01 AM	Java Source File	3 KB
SimpleServlet	4/2/2025 5:05 PM	Java Source File	3 KB


CreatePageServlet.java
SearchServlet.java
C: > Users > visha > Adobe > AEM > codebase > myTraining > core > src > main > java > com > myTraining > core > servlets >

```java
package com.example.core.servlets;

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

import javax.servlet.Servlet;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

@Component(service = Servlet.class)
@SlingServletPaths("/bin/searchContent")
public class SearchServlet extends SlingSafeMethodsServlet {

    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse respons
        response.setContentType("application/json");

        String searchTerm = request.getParameter("query");
        if (searchTerm == null || searchTerm.trim().isEmpty()) {
```