AEM – Task 7

Task 1: Create SampleServlet Using SlingAllMethodsServlet

1. Extend SlingAllMethodsServlet.
2. Register the servlet using resourceType.
3. Implement the doGet() method to return a sample JSON response.
4. Log the request processing for debugging.

Java Code

SampleServlet.java

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


import org.apache.sling.api.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
import org.apache.sling.api.servlets.SlingAllMethodsServlet;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.propertytypes.ServiceDescription;
import org.osgi.service.component.propertytypes.ServiceVendor;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import javax.servlet.Servlet;
import java.io.IOException;


@Component(
        service = Servlet.class,
        property = {
                "sling.servlet.resourceTypes=myTraining/components/sample",
                "sling.servlet.methods=GET",
                "sling.servlet.extensions=json"
```

```
    }
)
@ServiceDescription("Sample Servlet using resourceType")
@ServiceVendor("myTraining")
public class SampleServlet extends SlingAllMethodsServlet {

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

    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws IOException {
        LOGGER.info("[SampleServlet] Processing GET request...");
        response.setContentType("application/json");
        response.getWriter().write("{\"message\": \"Hello from SampleServlet!\"}");
    }
}
```

Screenshot

Task 2: Create CreatePageServlet Using SlingSafeMethodsServlet

1. Extend SlingSafeMethodsServlet.
2. Register the servlet using a specific path.
3. Accept a page name from the user via a query parameter.

4. Use PageManager APIs to create the page dynamically.
5. java

JavaCode

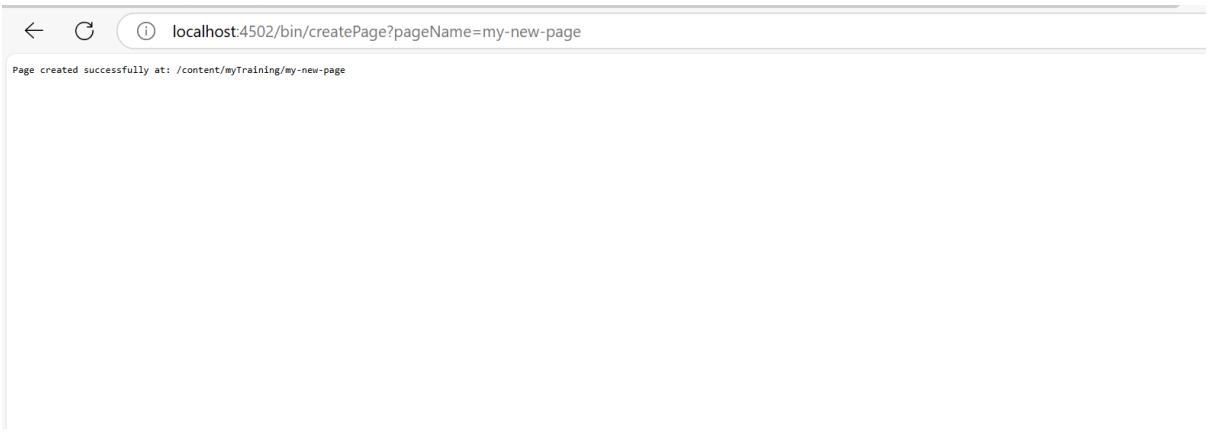
CreatePageServlet.java

```
package com.servlet.core.servlets;


import com.day.cq.wcm.api.Page;
import com.day.cq.wcm.api.PageManager;
import com.day.cq.wcm.api.WCMException;
import org.apache.sling.api.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.resource.ResourceResolver;
import org.osgi.service.component.annotations.Component;

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

@Component(
        service = Servlet.class,
        property = {
                "sling.servlet.paths=/bin/createNewPage",
                "sling.servlet.methods=GET"
        }
)
public class CreateNewPageServlet extends SlingSafeMethodsServlet {

    private static final String PARENT_PATH = "/content/servlet"; // Change as needed
```

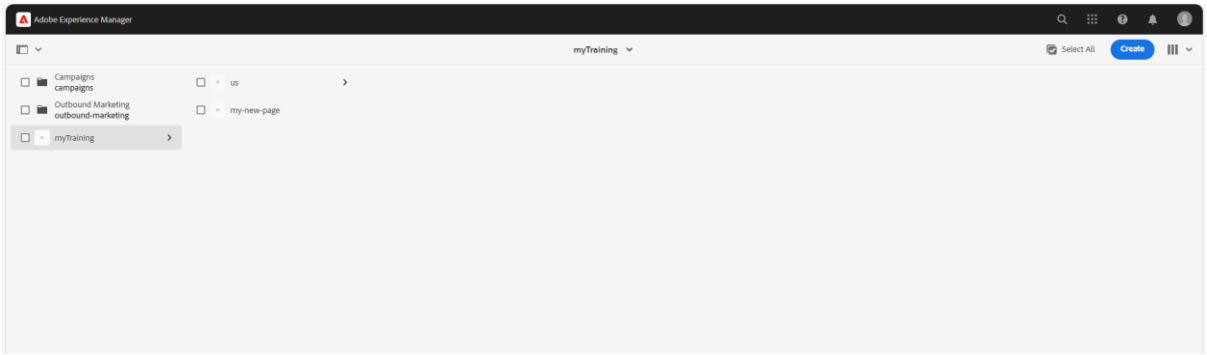
```
    private static final String TEMPLATE_PATH = "/conf/servlet/settings/wcm/templates/page-content"; // Change as needed


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


        String pageName = request.getParameter("pageName");


        if (pageName == null || pageName.trim().isEmpty()) {
            response.getWriter().write("Error: Page name is required.");
            return;
        }


        String formattedPageName = "custom-" + pageName; // Append "custom-" prefix to page name
        ResourceResolver resolver = request.getResourceResolver();
        PageManager pageManager = resolver.adaptTo(PageManager.class);


        if (pageManager != null) {
            try {
                Page newPage = pageManager.create(PARENT_PATH, formattedPageName, TEMPLATE_PATH, formattedPageName);
                response.getWriter().write("Page created successfully at: " + newPage.getPath());
            } catch (WCMException e) {
                response.getWriter().write("Error creating page: " + e.getMessage());
            }
        } else {
            response.getWriter().write("Error: PageManager is null. Unable to create page.");
        }
```

```
    }
}
```

Screenshot



New page created



Task 3: Use PageManager API to Create Pages

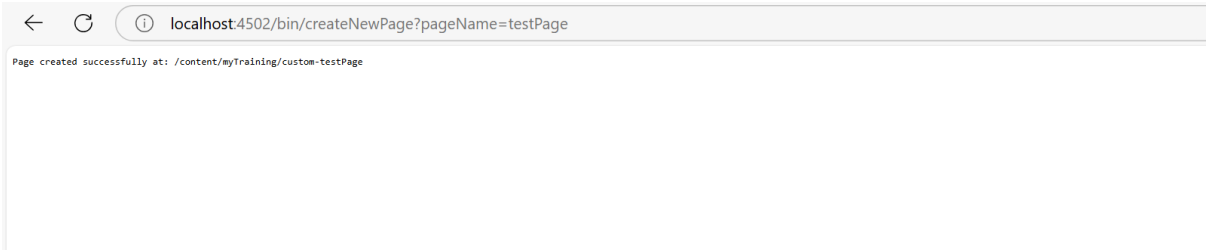
Javacode

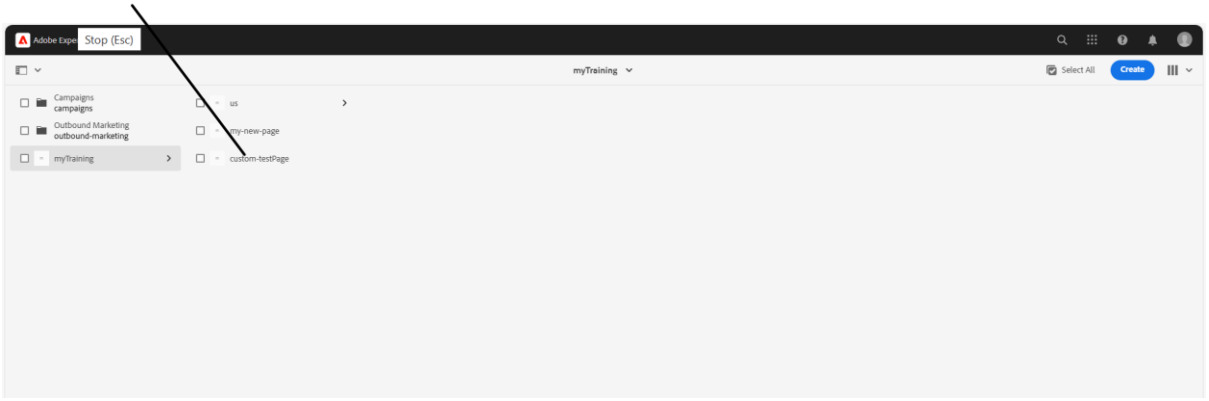
```
package com.myTraining.core.servlets;



import com.day.cq.wcm.api.Page;
import com.day.cq.wcm.api.PageManager;
import com.day.cq.wcm.api.WCMException;
import org.apache.sling.api.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
```

```
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.resource.ResourceResolver;
import org.osgi.service.component.annotations.Component;
import javax.servlet.Servlet;
import javax.servlet.ServletException;
import java.io.IOException;

@Component(
        service = Servlet.class,
        property = {
                "sling.servlet.paths=/bin/createNewPage",
                "sling.servlet.methods=GET"
        }
)
public class CreateNewPageServlet extends SlingSafeMethodsServlet {

    private static final String PARENT_PATH = "/content/myTraining"; // Change as needed
    private static final String TEMPLATE_PATH =
"/conf/myTraining/settings/wcm/templates/page-content"; // Change as needed

    @Override
    protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse
response) throws ServletException, IOException {
        String pageName = request.getParameter("pageName");

        if (pageName == null || pageName.trim().isEmpty()) {
            response.getWriter().write("Error: Page name is required.");
            return;
        }

        String formattedPageName = "custom-" + pageName; // Append "custom-" prefix to
page name

        ResourceResolver resolver = request.getResourceResolver();
        PageManager pageManager = resolver.adaptTo(PageManager.class);

        if (pageManager != null) {
            try {
                Page newPage = pageManager.create(PARENT_PATH, formattedPageName,
TEMPLATE_PATH, formattedPageName);
                response.getWriter().write("Page created successfully at: " + newPage.getPath());
            } catch (WCMException e) {
                response.getWriter().write("Error creating page: " + e.getMessage());
            }
```

```
        } else {
            response.getWriter().write("Error: PageManager is null. Unable to create page.");
        }
    }
}
```

Screenshots





Task 4: Create SearchServlet Using PredicateMap

1. Extend SlingSafeMethodsServlet.
2. Register the servlet using a path.
3. Use PredicateMap to search for pages based on user input.
4. Return the results in JSON format.

Java Code

SearchServlet.java :

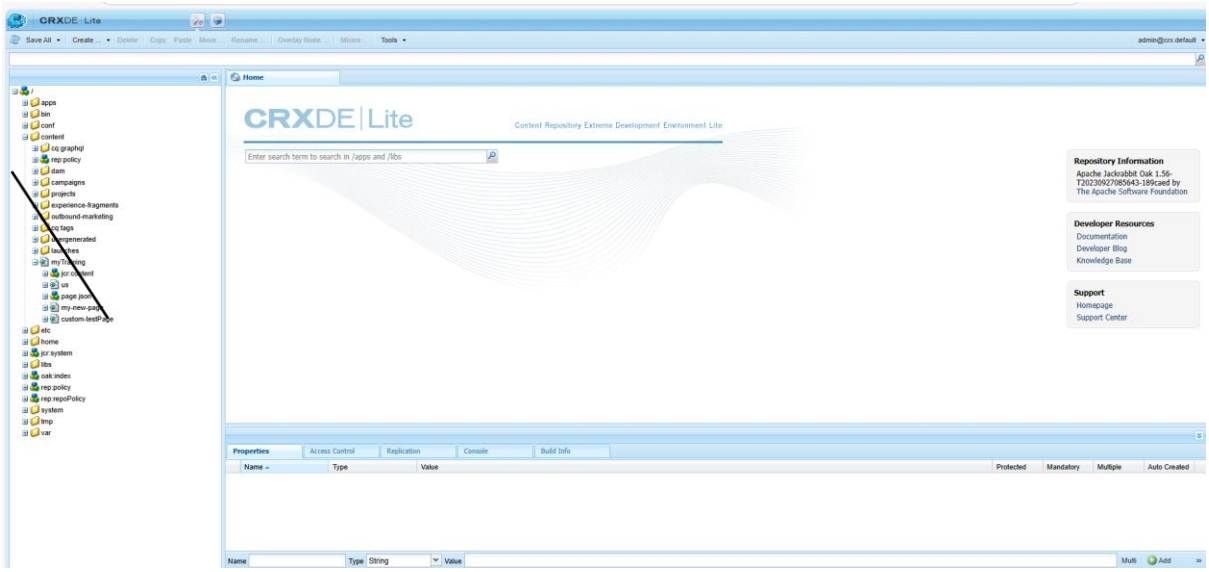
package com.servlet.core.servlets;

```
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.SlingHttpServletRequest;
import org.apache.sling.api.SlingHttpServletResponse;
import org.apache.sling.api.servlets.SlingSafeMethodsServlet;
import org.apache.sling.api.resource.ResourceResolver;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Reference;
import javax.jcr.Session;
import javax.servlet.Servlet;
import javax.servlet.ServletException;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.List;

@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 {

        String searchTerm = request.getParameter("query");

        if (searchTerm == null || searchTerm.trim().isEmpty()) {
            response.setContentType("application/json");
            response.getWriter().write("{\"error\": \"Search query is required.\"}");
            return;
        }

        ResourceResolver resolver = request.getResourceResolver();
        Session session = resolver.adaptTo(Session.class);

        if (session == null) {
            response.setContentType("application/json");
            response.getWriter().write("{\"error\": \"Unable to get JCR session.\"}");
            return;
        }

        try {
            Map<String, String> predicateMap = new HashMap<>();
            predicateMap.put("type", "cq:Page");
            predicateMap.put("fulltext", searchTerm);
            predicateMap.put("path", "/content/servlet"); // Adjust this as per your AEM content
structure
```

```
        predicateMap.put("p.limit", "10"); // Limit to 10 results

        Query query = queryBuilder.createQuery(PredicateGroup.create(predicateMap),
session);
        SearchResult searchResult = query.getResult();
        List<Hit> hits = searchResult.getHits();

        StringBuilder jsonResponse = new StringBuilder();
        jsonResponse.append("{ \"results\": [");

        for (int i = 0; i < hits.size(); i++) {
            String pagePath = hits.get(i).getPath();
            jsonResponse.append("{\"path\": \"").append(pagePath).append("\"}");
            if (i < hits.size() - 1) {
                jsonResponse.append(",");
            }
        }

        jsonResponse.append("] }");
        response.setContentType("application/json");
        response.getWriter().write(jsonResponse.toString());

      } catch (Exception e) {
        response.setContentType("application/json");
        response.getWriter().write("{\"error\": \"Error executing search: " + e.getMessage() +
"\"}");
      }
   }
}
```

Screenshot



Custom test page search