1. Create SampleServlet Using SlingAllMethodsServlet

Steps:

1. Create a servlet by extending SlingAllMethodsServlet.
2. Register it using resourceType.
3. Override doGet and doPost methods to handle requests.
4. Deploy and test using CURL or AEM URL.

Java Code:

SampleServlet.java

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

import org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

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

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

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

import java.io.IOException;

@Component(service = javax.servlet.Servlet.class)

@SlingServletResourceTypes(

    resourceTypes = "servlet/components/page", // Must match sling:resourceType in JCR

    methods = { "GET" },

    extensions = "json"

)

public class SampleServlet extends SlingAllMethodsServlet {

  private static final long serialVersionUID = 1L;

  @Override

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

    response.setContentType("application/json");

    response.getWriter().write("{ \"message\": \"Hello from SampleServlet!\" }");

  }

}
```

2. Create CreatePageServlet Using SlingSafeMethodsServlet

Steps:

1. Extend SlingSafeMethodsServlet.
2. Register it using sling.servlet.paths.
3. Accept page name from the user and create an AEM page.
4. Use PageManager API to create the page.

Java Code:

CreatePageServlet.java

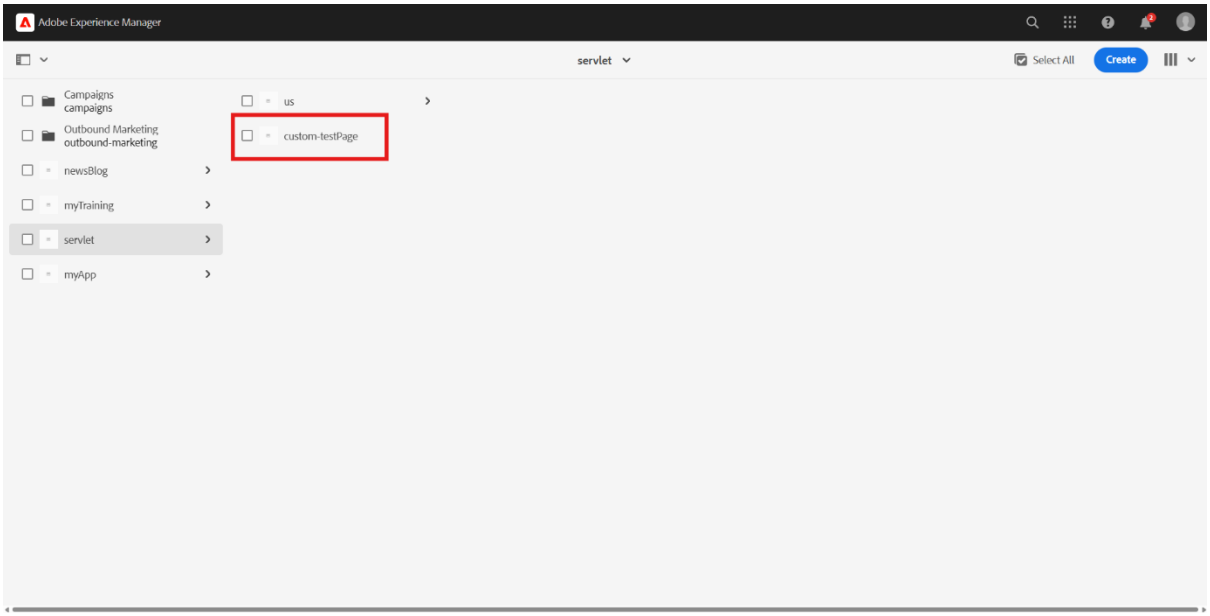
```
package com.servlet.core.servlets;

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:

- Servlet execution creating an AEM page.

- Newly created page in AEM.



3. Search Content Using SearchServlet & PredicateMap

Steps:

1. Create a servlet extending SlingSafeMethodsServlet.
2. Use QueryBuilder and PredicateMap to search content.
3. Retrieve search results and display them in JSON format.
4. Deploy and test using query parameters.

Java Code:

SearchServlet.java

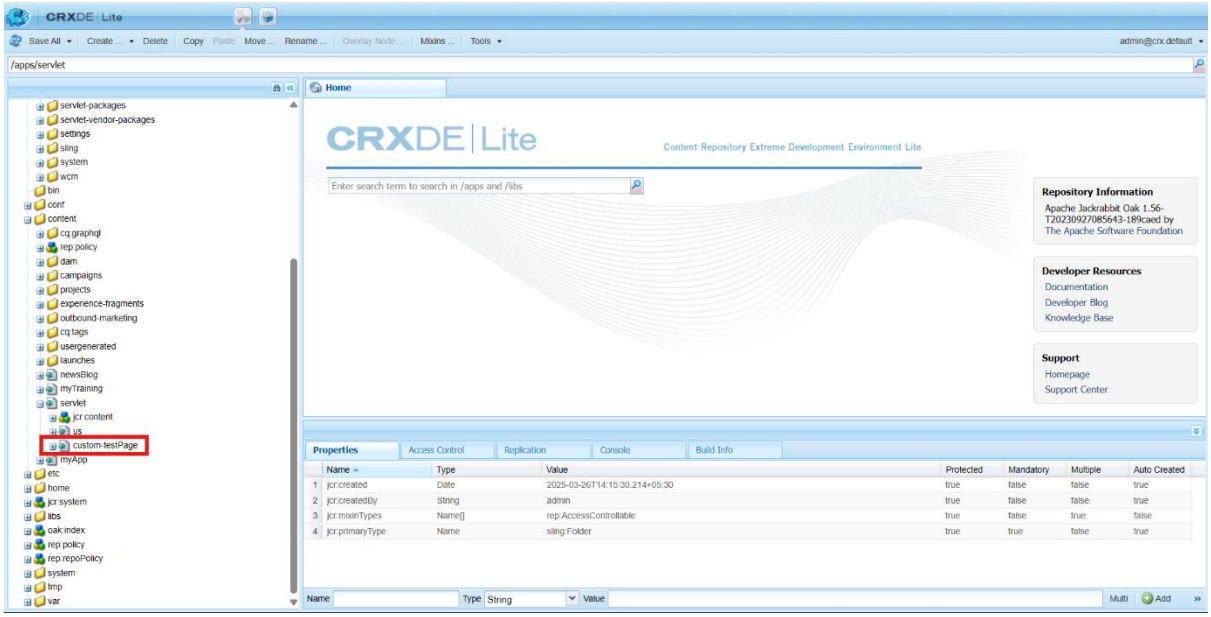
```
package com.servlet.core.servlets;

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:

- CRXDE Lite showing created content page.



- Servlet execution results in the browser.