# 1️. Create SampleServlet Extending SlingAllMethodsServlet Overview

* This servlet will handle **GET** and **POST** requests.
* It will be registered using sling:resourceType.

# Java Class: SampleServlet.java

•¡˙ **Location:** core/src/main/java/com/assignment/servlets/SampleServlet.java package com.assignment.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.apache.sling.servlets.annotations.SlingServletResourceTypes; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.servlet.Servlet;

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

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

resourceTypes = "assignment/components/sample", methods = {"GET", "POST"},

extensions = "json"

)

public class SampleServlet extends SlingAllMethodsServlet {

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

@Override

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

LOG.info("Sample Servlet GET Request"); response.setContentType("application/json"); response.getWriter().write("{\"message\": \"GET request processed\"}");

}

@Override

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

LOG.info("Sample Servlet POST Request"); response.setContentType("application/json"); response.getWriter().write("{\"message\": \"POST request processed\"}");

}

}

# 2️. Create CreatePageServlet Extending SlingSafeMethodsServlet Overview

* This servlet will handle **GET** requests.
* It will be registered using **path-based** registration.

# Java Class: CreatePageServlet.java

**Location:** core/src/main/java/com/assignment/servlets/CreatePageServlet.java package com.assignment.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.osgi.service.component.annotations.Component; import org.apache.sling.servlets.annotations.SlingServletPaths; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.jcr.Session; import javax.servlet.Servlet;

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

@Component(service = Servlet.class) @SlingServletPaths("/bin/createPage")

public class CreatePageServlet extends SlingSafeMethodsServlet {

private static final Logger LOG = LoggerFactory.getLogger(CreatePageServlet.class);

@Override

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

String pageName = request.getParameter("pageName"); if (pageName == null || pageName.isEmpty()) {

response.getWriter().write("Page name is required."); return;

}

try {

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

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

if (pageManager != null) {

Page newPage = pageManager.create("/content/us/en", pageName, "/conf/wknd/settings/wcm/templates/content-page", pageName);

LOG.info("Page created successfully at: {}", newPage.getPath()); response.getWriter().write("Page created successfully at: " + newPage.getPath());

} else {

LOG.error("PageManager is null"); response.getWriter().write("Error: PageManager is null.");

}

} catch (WCMException e) { LOG.error("Error creating page: ", e);

response.getWriter().write("Error creating page.");

}

}

}

# 3️. Take Page Name from User and Create Pages in AEM using Servlet

* **Steps to Test**:
  1. Open a browser.
  2. Visit: <http://localhost:4502/bin/createPage?pageName=newsPage1>
  3. It will create a page under /content/us/en/newsPage1.

# 4️. Use PageManager APIs for Page Creation

* **PageManager API Used in CreatePageServlet**:

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

Page newPage = pageManager.create("/content/us/en", pageName, "/conf/wknd/settings/wcm/templates/content-page", pageName);

* It creates a new page under /content/us/en using an existing AEM template.

# 5️. Create SearchServlet to Search Content Using PredicateMap Overview

* This servlet will **search AEM content** using **Query Builder API**. **Java Class: SearchServlet.java**

**Location:** core/src/main/java/com/assignment/servlets/SearchServlet.java

package com.assignment.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 org.apache.sling.api.SlingHttpServletRequest;

import org.apache.sling.api.SlingHttpServletResponse;

import org.apache.sling.api.servlets.SlingSafeMethodsServlet; import org.osgi.service.component.annotations.Component; import org.apache.sling.servlets.annotations.SlingServletPaths; import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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) @SlingServletPaths("/bin/searchContent")

public class SearchServlet extends SlingSafeMethodsServlet {

private static final Logger LOG = LoggerFactory.getLogger(SearchServlet.class);

@Override

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

String searchText = request.getParameter("query"); if (searchText == null || searchText.isEmpty()) {

response.getWriter().write("Query parameter is required."); return;

}

try {

QueryBuilder queryBuilder = request.getResourceResolver().adaptTo(QueryBuilder.class); Session session = request.getResourceResolver().adaptTo(Session.class);

Map<String, String> queryMap = new HashMap<>(); queryMap.put("path", "/content/us/en"); queryMap.put("type", "cq:Page"); queryMap.put("fulltext", searchText); queryMap.put("p.limit", "-1");

Query query = queryBuilder.createQuery(queryBuilder.createQuery(queryMap), session); SearchResult result = query.getResult();

List<Hit> hits = result.getHits();

response.setContentType("application/json"); response.getWriter().write("{ \"results\": ["); for (Hit hit : hits) {

response.getWriter().write("{ \"path\": \"" + hit.getPath() + "\" },");

}

response.getWriter().write("] }");

} catch (Exception e) { LOG.error("Search error: ", e);

response.getWriter().write("Error occurred during search.");

}

}

}