**1. Create SampleServlet extend the SlingAllMethod Servlet and register it using resourceType.**

Path: /apps/sample/core/src/main/java/com/sample/core/servlets

**SampleServlet.java**

package com.sample.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.framework.Constants;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import java.io.IOException;

@Component(

service = {Servlet.class},

property = {

Constants.SERVICE\_DESCRIPTION + "=Sample Servlet",

"sling.servlet.methods=GET",

"sling.servlet.resourceTypes=sample/components/sample"

}

)

public class SampleServlet extends SlingAllMethodsServlet {

@Override

protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)

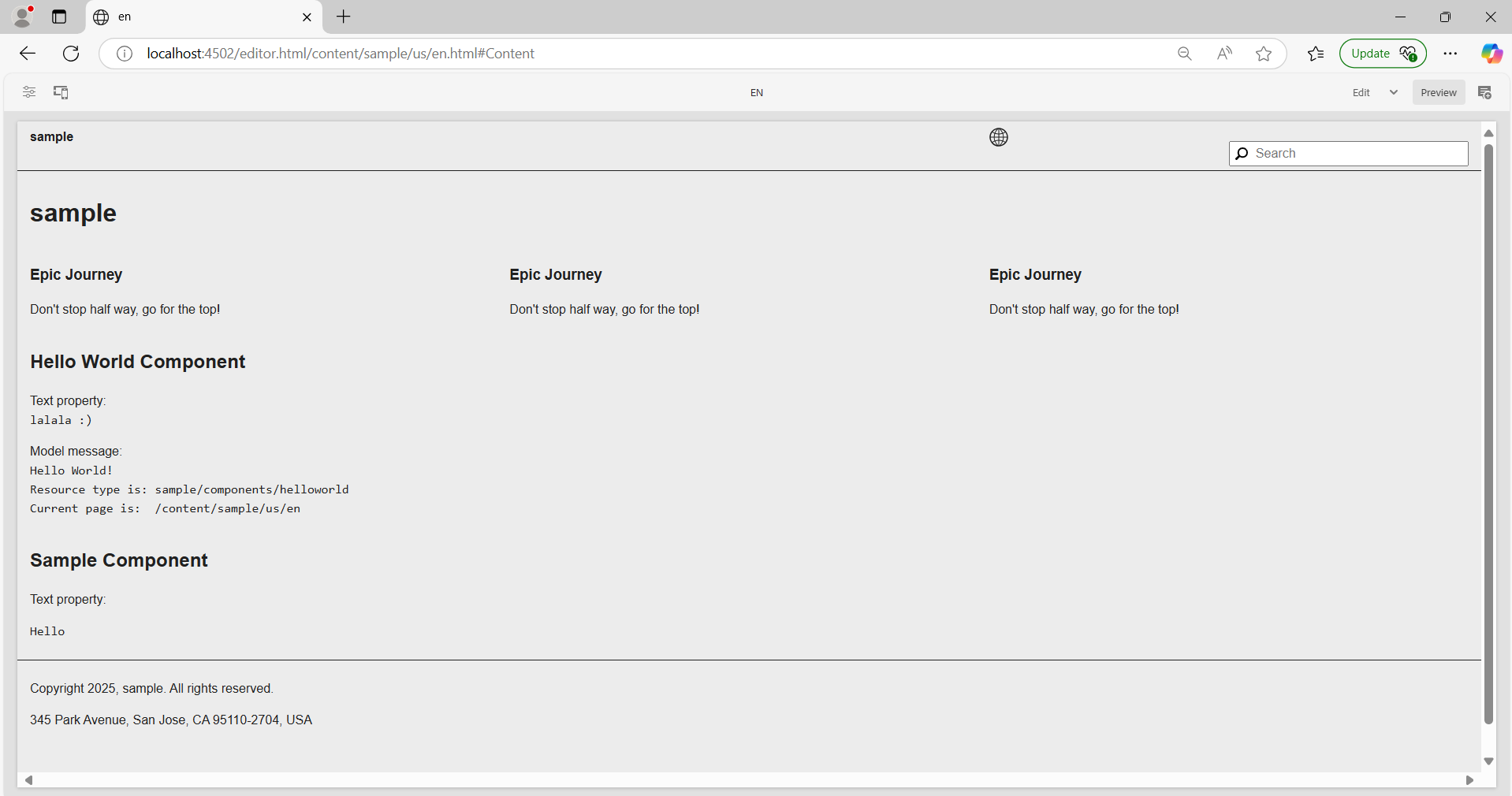
throws ServletException, IOException {

response.setContentType("text/plain");

response.getWriter().write("Hello from SampleServlet!");

}

}

****

**2. Create CreatePageServlet extends the SlingSafeMethod Servlet and registers it using path.**

Path: /apps/sample/core/src/main/java/com/sample/core/servlets

**CreatePageServlet.java**

package com.sample.core.servlets;

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

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

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.osgi.framework.Constants;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import java.io.IOException;

@Component(

service = {Servlet.class},

property = {

Constants.SERVICE\_DESCRIPTION + "= Create Page Servlet",

"sling.servlet.paths=" + "/bin/createpage",

"sling.servlet.methods=GET"

}

)

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 missing!");

return;

}

try {

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

if (pageManager != null) {

Page newPage = pageManager.create("/content/sample", pageName, "cq:Page", pageName);

if (newPage != null) {

response.getWriter().write("Page created: " + newPage.getPath());

LOG.info("Page created at: {}", newPage.getPath());

} else {

response.getWriter().write("Page creation failed.");

}

} else {

response.getWriter().write("Failed to adapt to PageManager.");

}

} catch (Exception e) {

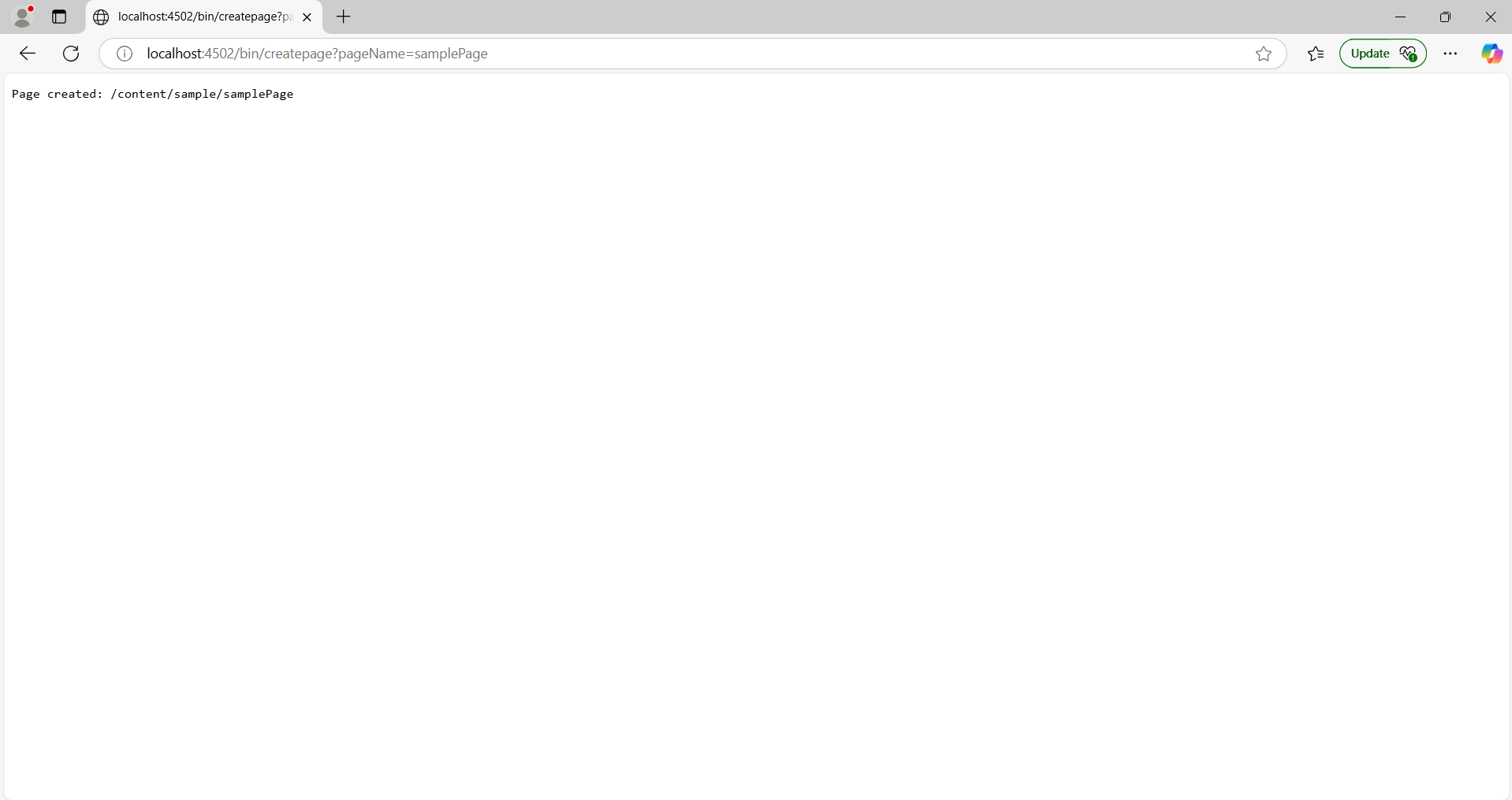
LOG.error("Error creating page", e);

response.getWriter().write("Error creating page: " + e.getMessage());

}

}

}

****

**3. Take the page name from the user and create pages in aem using the above servlet.**

Path: /apps/sample/core/src/main/java/com/sample/core/servlets

**CreatePageServlet.java**

package com.sample.core.servlets;

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

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

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.osgi.framework.Constants;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import javax.servlet.Servlet;

import javax.servlet.ServletException;

import java.io.IOException;

@Component(

service = {Servlet.class},

property = {

Constants.SERVICE\_DESCRIPTION + "= Create Page Servlet",

"sling.servlet.paths=" + "/bin/createpage",

"sling.servlet.methods=GET"

}

)

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 missing!");

return;

}

try {

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

if (pageManager != null) {

Page newPage = pageManager.create("/content/sample", pageName, "cq:Page", pageName);

if (newPage != null) {

response.getWriter().write("Page created: " + newPage.getPath());

LOG.info("Page created at: {}", newPage.getPath());

} else {

response.getWriter().write("Page creation failed.");

}

} else {

response.getWriter().write("Failed to adapt to PageManager.");

}

} catch (Exception e) {

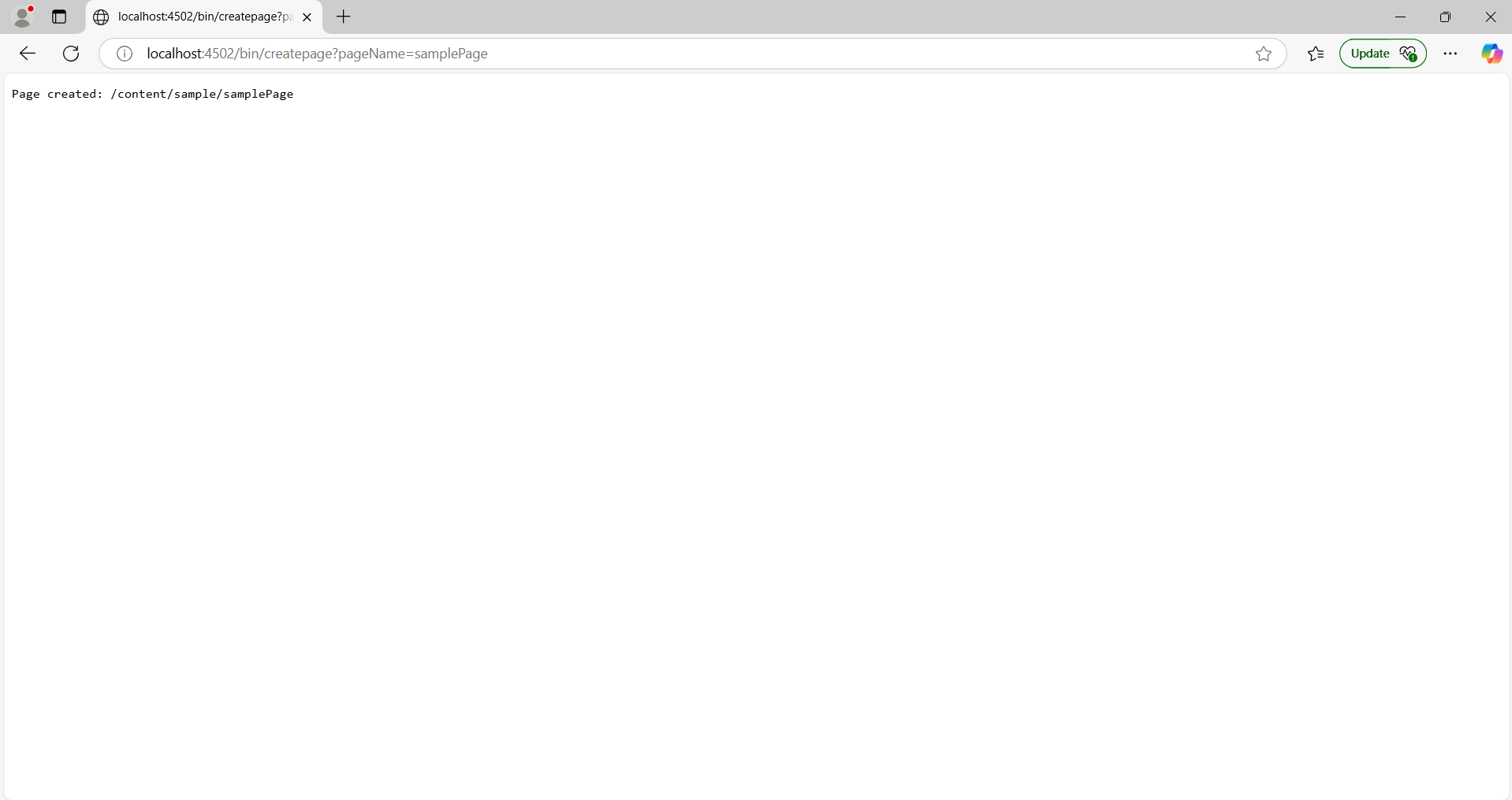
LOG.error("Error creating page", e);

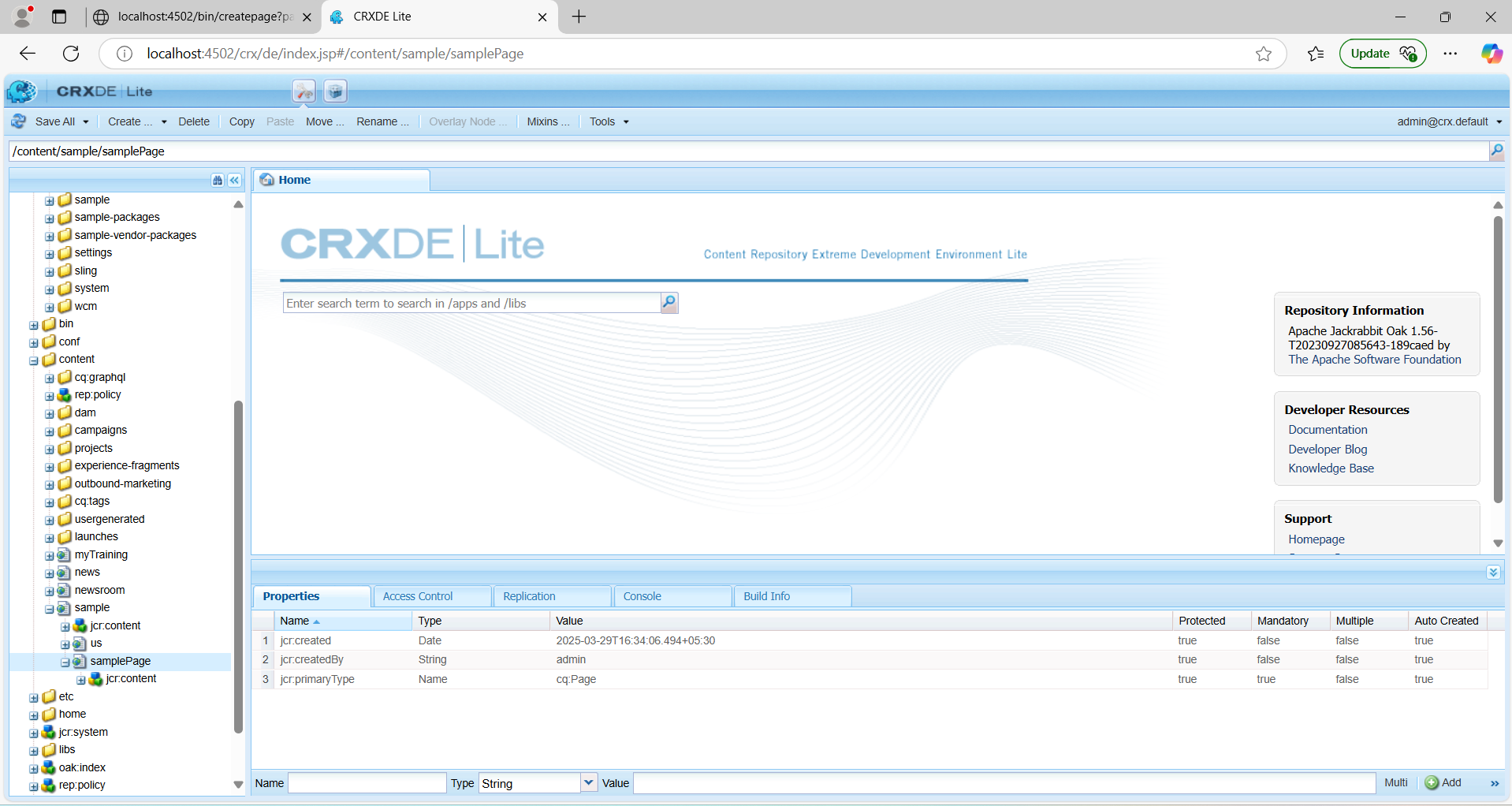
response.getWriter().write("Error creating page: " + e.getMessage());

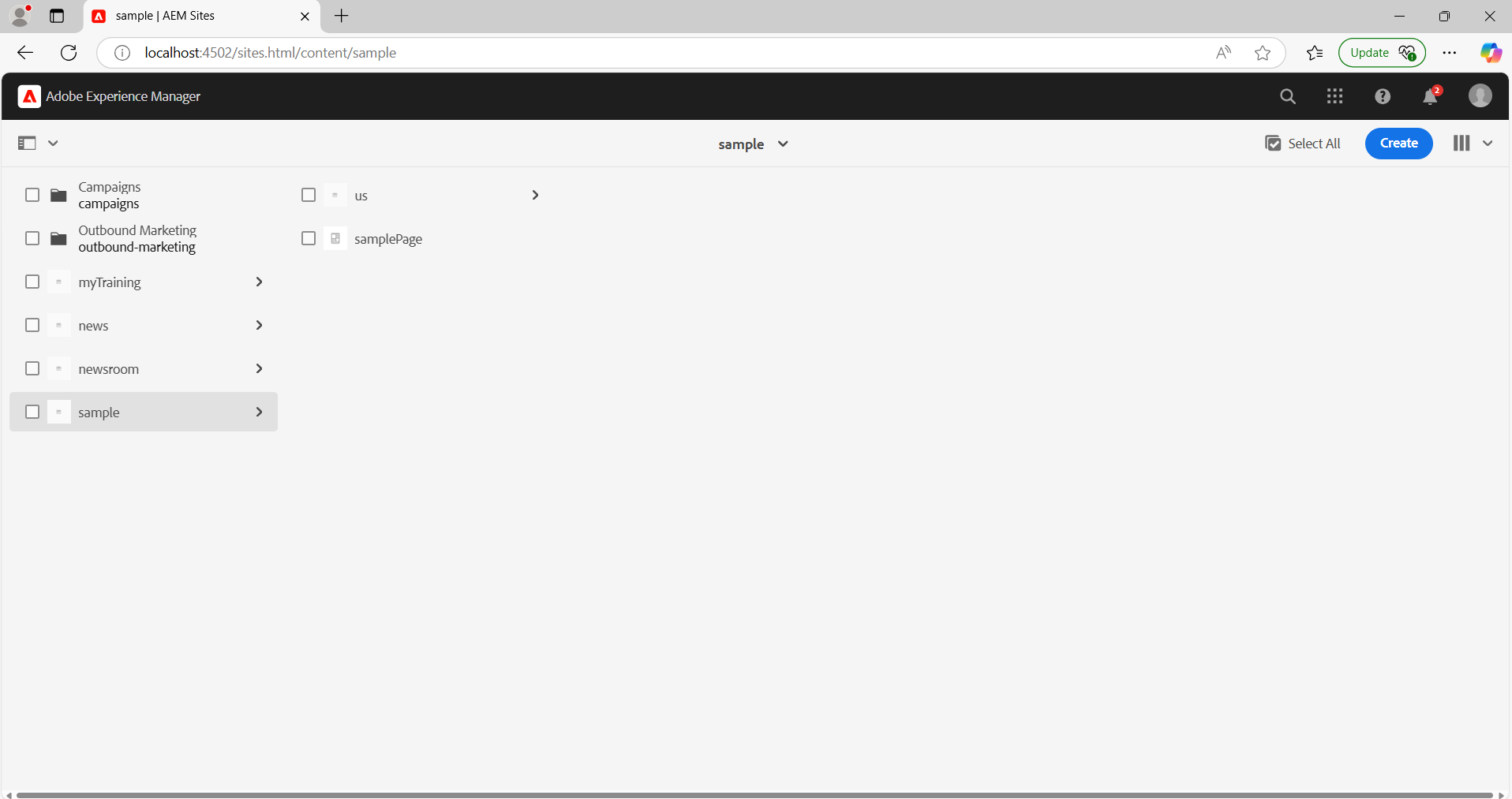
}

}

}

****

****

****

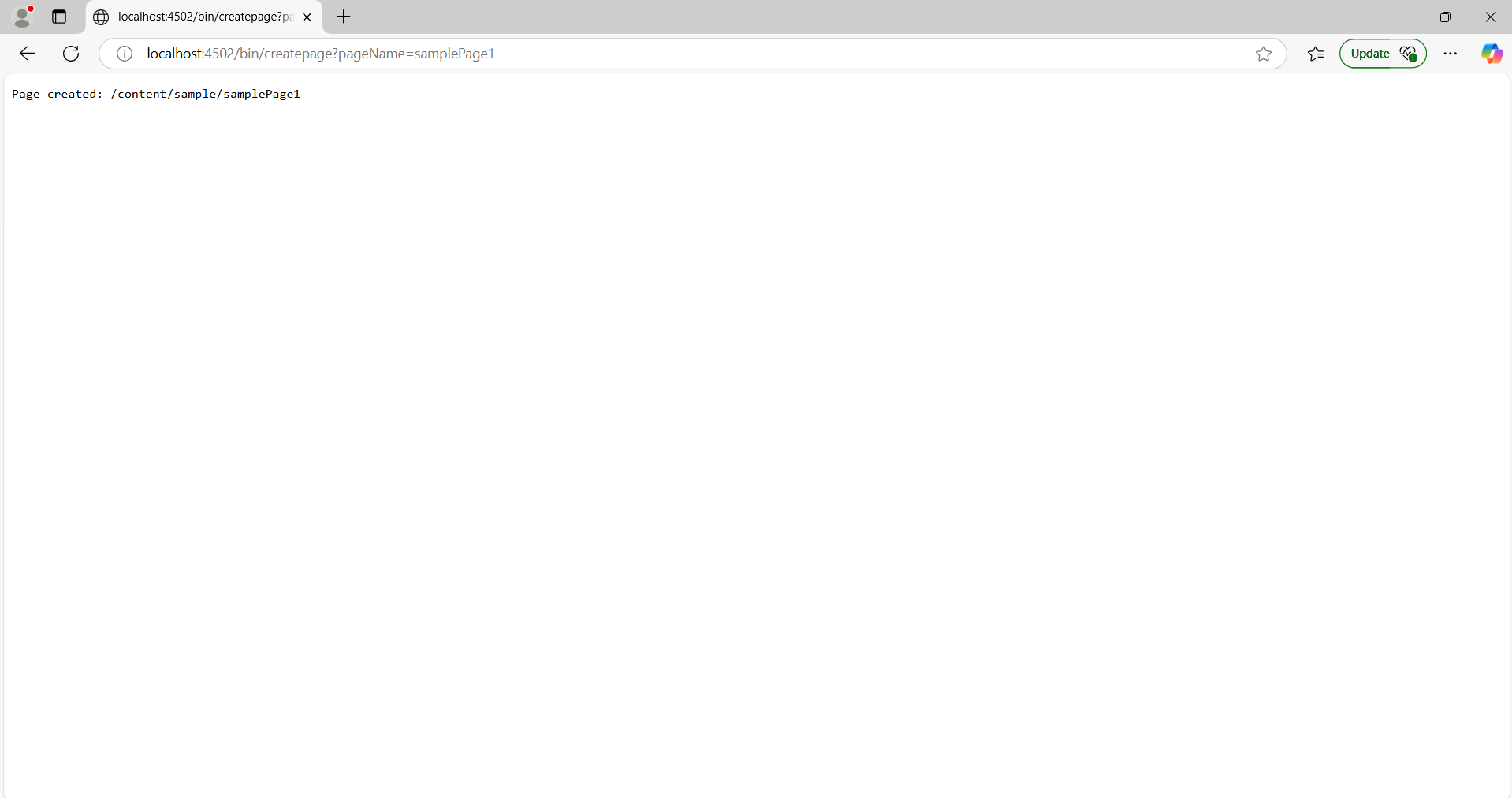
**4. Use page Manager Apis for the above task.**

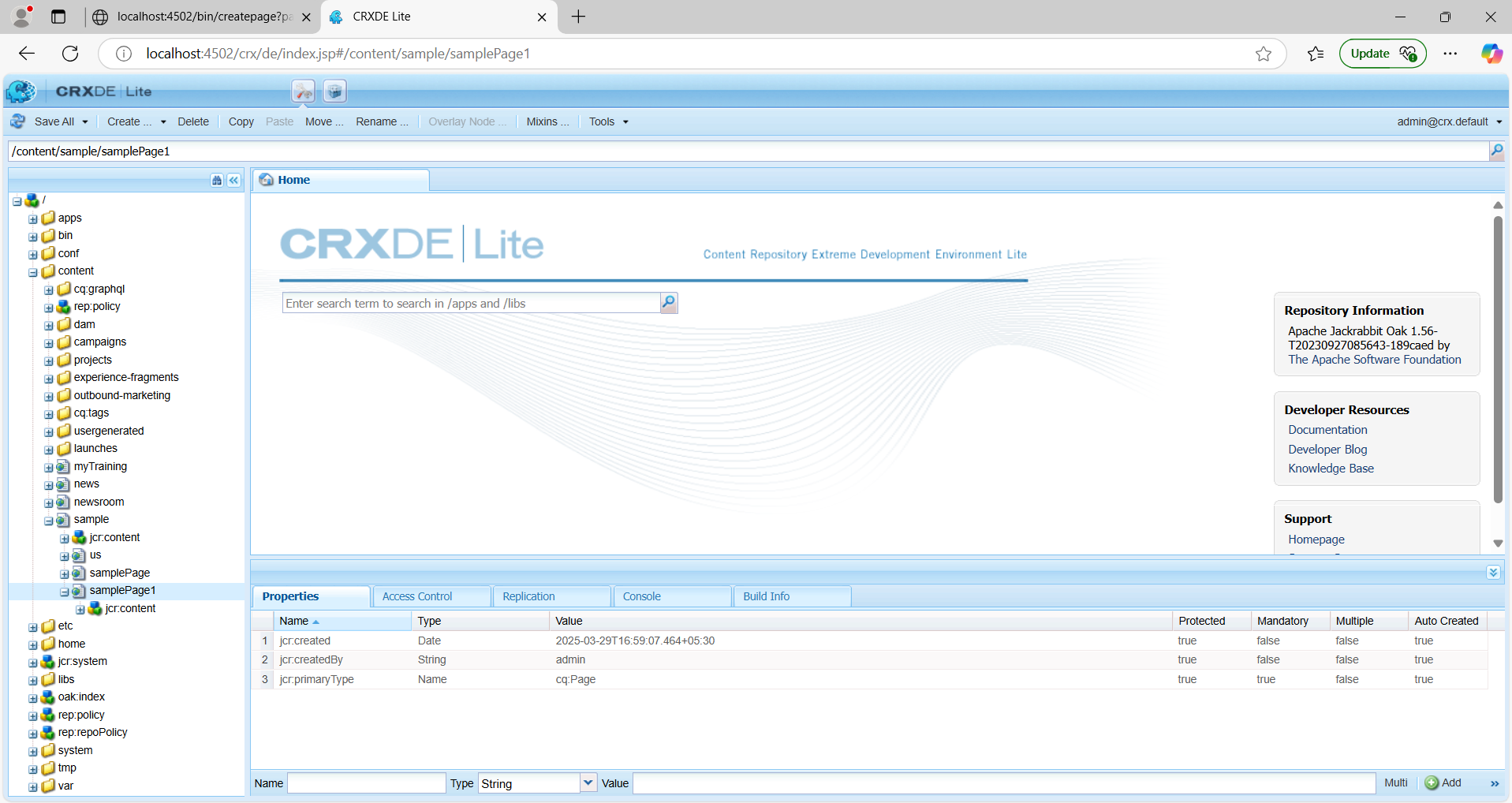
In the CreatePageServlet.java file, the PageManager API is used to create the page:

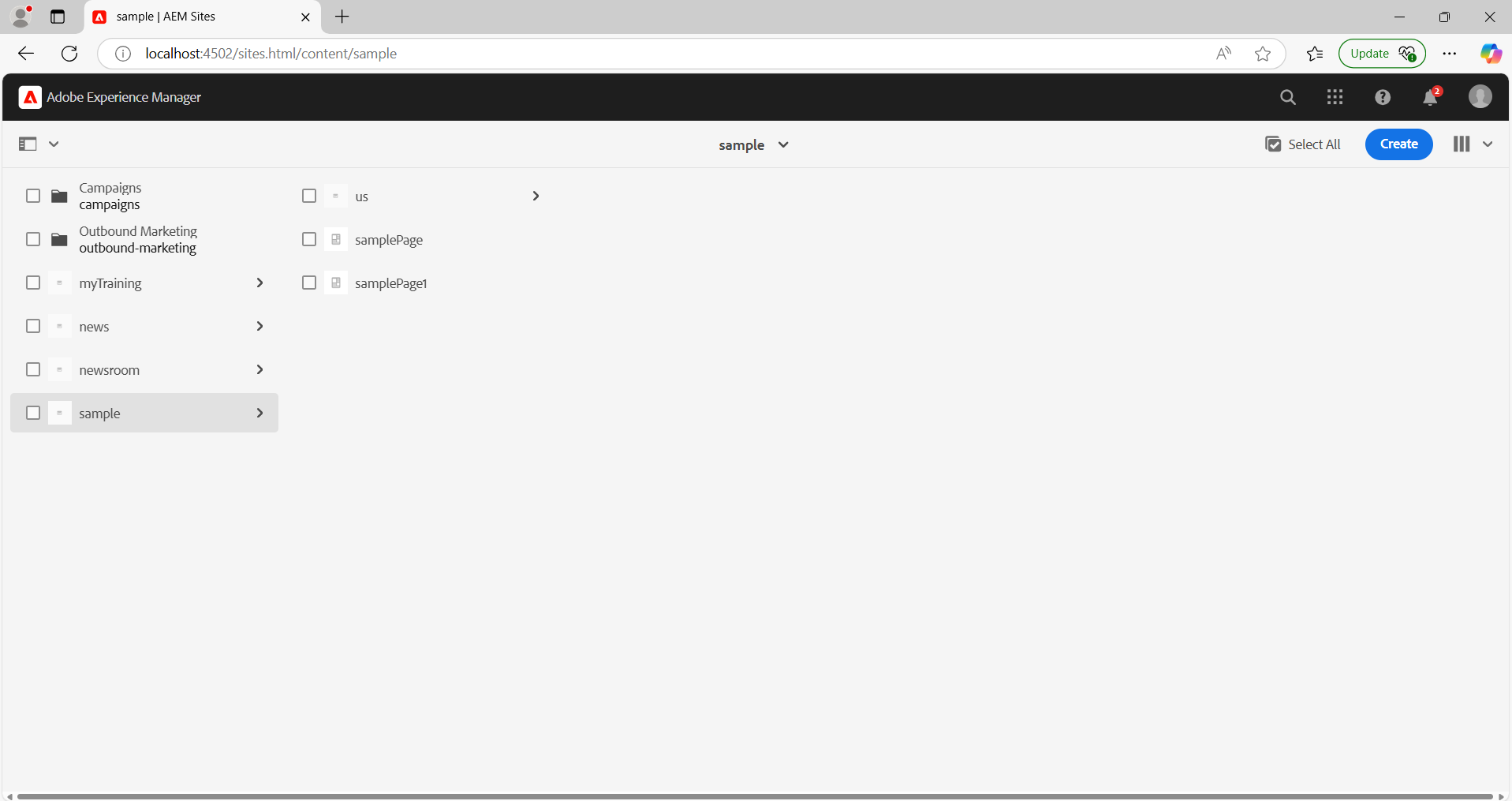
PageManager pageManager = request.getResourceResolver().adaptTo(PageManager.class);

Page newPage = pageManager.create("/content/sample", pageName, "/apps/sample/templates/page", pageName);

* pageManager.create() creates the page.
* The parameters specify the parent path, page name, template, and title.
* The session is saved to persist the changes.

****

****

****

**5. Create a SearchServlet to search for the content using PredicateMap to search the content from pages. Use this reference for the query builder: https://medium.com/@manumathew28.94/query-builder-aem-5869a1850c85**

Path: /apps/sample/core/src/main/java/com/sample/core/servlets

**SearchServlet.java**

package com.sample.core.servlets;

import com.day.cq.search.Query;

import com.day.cq.search.QueryBuilder;

import com.day.cq.search.PredicateGroup;

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.framework.Constants;

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

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

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},

property = {

Constants.SERVICE\_DESCRIPTION + "= Search Servlet Using PredicateMap and QueryBuilder",

"sling.servlet.paths=" + "/bin/search",

"sling.servlet.methods=GET"

}

)

public class SearchServlet extends SlingSafeMethodsServlet {

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

@Reference

private QueryBuilder queryBuilder;

@Override

protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response)

throws ServletException, IOException {

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

if (searchTerm == null || searchTerm.trim().isEmpty()) {

response.getWriter().write("Search term is missing!");

return;

}

try {

Map<String, String> map = new HashMap<>();

map.put("path", "/content/sample");

map.put("type", "cq:Page");

map.put("fulltext", searchTerm);

map.put("p.limit", "-1");

map.put("p.nodedepth", "4");

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

if (session != null) {

Query query = queryBuilder.createQuery(

PredicateGroup.create(map),

session

);

SearchResult result = query.getResult();

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

response.getWriter().write("Search Results:\n");

if (hits.isEmpty()) {

response.getWriter().write("No results found.");

} else {

for (Hit hit : hits) {

response.getWriter().write(hit.getPath() + "\n");

}

}

} else {

response.getWriter().write("Failed to adapt to JCR Session.");

}

} catch (Exception e) {

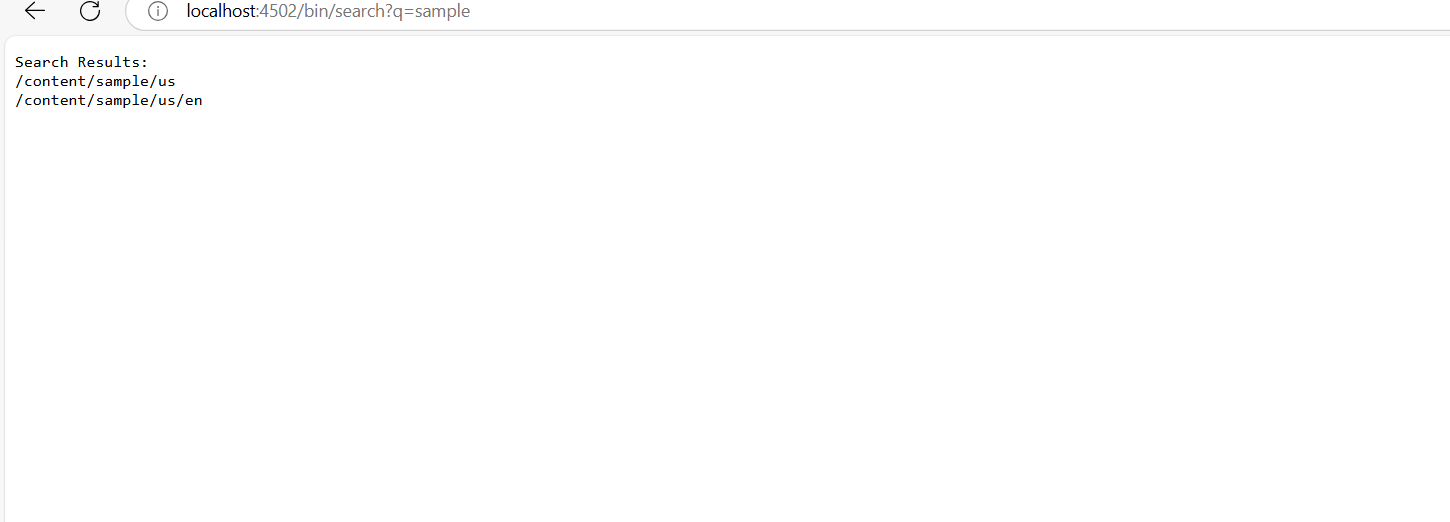
LOG.error("Error while executing query", e);

response.getWriter().write("Error: " + e.getMessage());

}

}

}

****