

1.Create SampleServlet extend the SlingAllMethod Servlet and register it using resourceType --->{http://localhost:4502/content/samplePage.html}

@Component(service = Servlet.class)

**@SlingServletResourceTypes(
 resourceTypes = "myTraining/components/page",
 methods = "GET"**

)

public class SampleServlet extends SlingAllMethodsServlet {

@Override

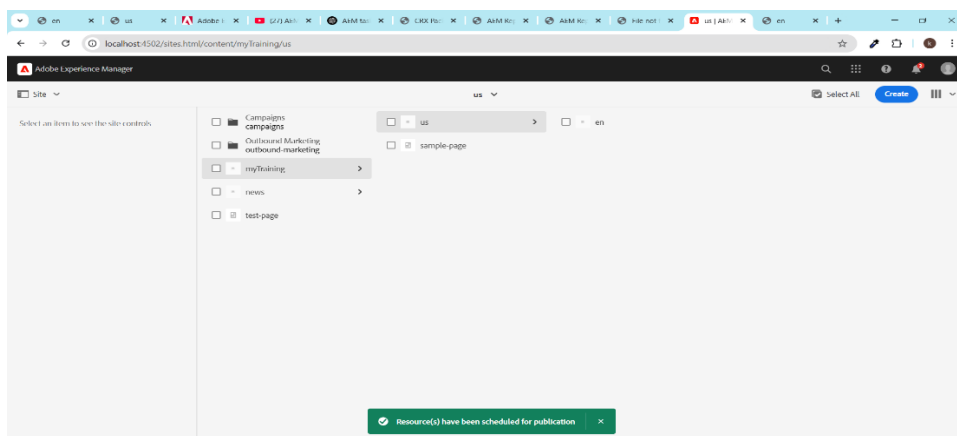
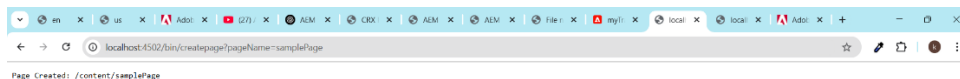
**protected void doGet(SlingHttpServletRequest request,
SlingHttpServletResponse response)**

throws ServletException, IOException {

response.getWriter().write("SampleServlet Accessed!");

}

}



Page created successfully and displayed in aem .

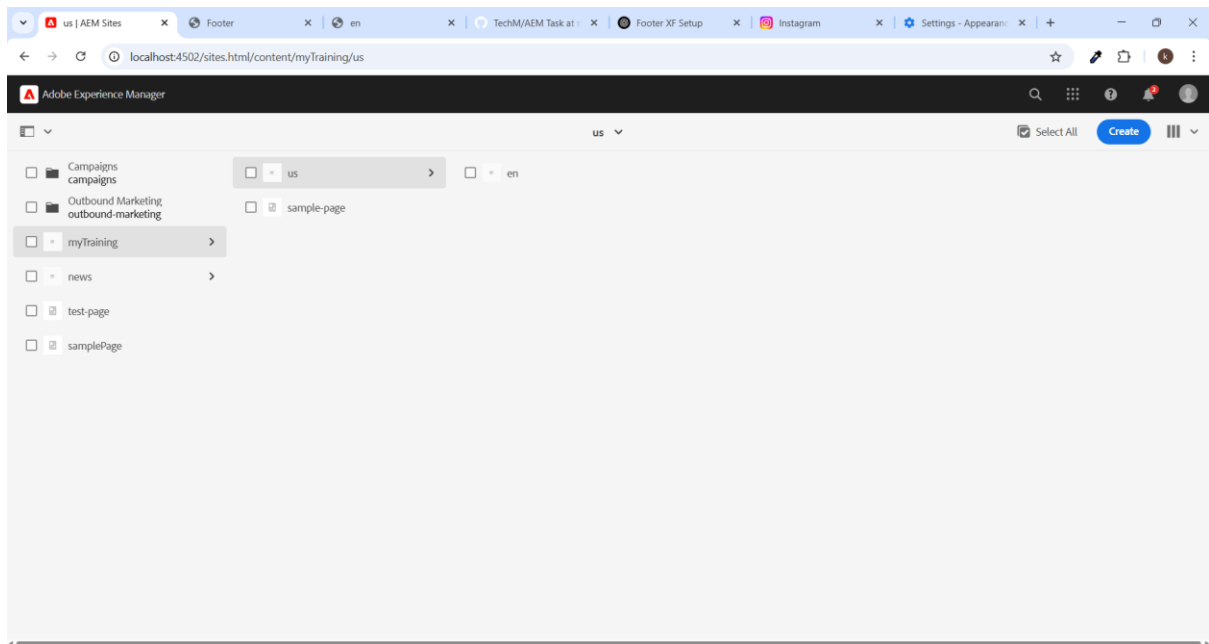
2.Create CreatePageServlet extend the SlingSafeMethod Servlet and register it using path and 4.Use page Manager Apis for above task.

```
@Component(service = Servlet.class)
@SlingServletPaths("/bin/createpage")
public class CreatePageServlet extends SlingSafeMethodsServlet {

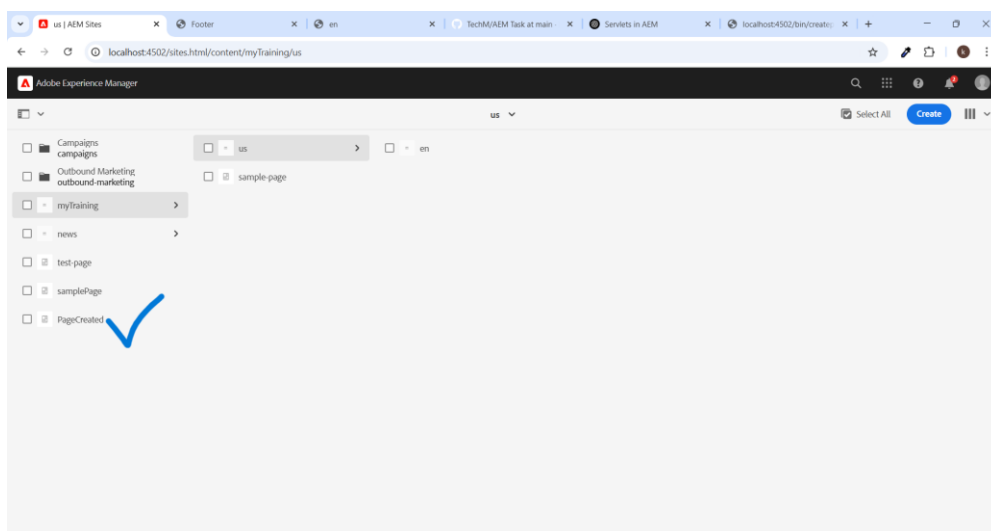
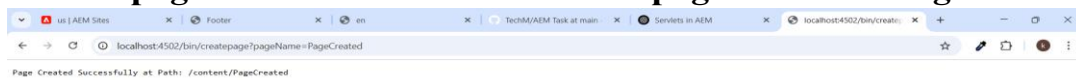
    @Override
    protected void doGet(SlingHttpServletRequest request,
        SlingHttpServletResponse response)
        throws ServletException, IOException {
        String pageName = request.getParameter("pageName");
        if (pageName == null || pageName.isEmpty()) {
            response.getWriter().write("Page Name is required!");
            return;
        }
        ResourceResolver resourceResolver = request.getResourceResolver();
        PageManager pageManager =
            resourceResolver.adaptTo(PageManager.class);

        if (pageManager != null) {
            try {
                Page newPage = pageManager.create("/content", pageName,
                    "myTraining/components/page", pageName);
                response.getWriter().write("Page Created Successfully at Path: " +
                    newPage.getPath());
            } catch (Exception e) {
                response.getWriter().write("Error creating page: " + e.getMessage());
            }
        } else {
            response.getWriter().write("Failed to Adapt to PageManager!");
        }
    }
}
```

Before Adding Page :



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



Page created successfully and displayed in aem .

5. Create one SearchServlet to search the content using PredicateMap to search the content from pages

```
@Component(
    service = Servlet.class,
    property = {
        "sling.servlet.paths=/bin/custom/searchpages",
        "sling.servlet.methods=GET"
    }
)
public class SearchServlet extends SlingAllMethodsServlet {
    @Reference
    private QueryBuilder queryBuilder;
    @Override
    protected void doGet(SlingHttpServletRequest request,
        SlingHttpServletResponse response) throws ServletException, IOException {
        response.setContentType("application/json");
        response.setCharacterEncoding("UTF-8");
        String searchText = request.getParameter("searchText");
        String searchPath = request.getParameter("searchPath");
        JSONObject jsonResponse = new JSONObject();
        JSONArray resultsArray = new JSONArray();
        // Validate input
        if (StringUtils.isBlank(searchText)) {
            try {
                jsonResponse.put("error", "Missing searchText parameter");
                response.getWriter().write(jsonResponse.toString());
            } catch (JSONException e) {
                throw new RuntimeException(e);
            }
        }
        return;
    }
    // Default search path if none provided
    if (StringUtils.isBlank(searchPath)) {
        searchPath = "/content"; // Default to AEM content root
    }
    Session session = request.getResourceResolver().adaptTo(Session.class);
    if (session == null) {
        try {
            jsonResponse.put("error", "JCR Session is null");
        }
```

```

        response.getWriter().write(jsonResponse.toString());
    } catch (JSONException e) {
        throw new RuntimeException(e);
    }
    return;
}
try {
    // Build search query using PredicateMap
    Map<String, String> predicateMap = new HashMap<>();
    predicateMap.put("path", searchPath);
    predicateMap.put("type", "cq:Page"); // Search AEM pages only
    predicateMap.put("fulltext", searchText);
    predicateMap.put("fulltext.relPath", "jcr:content"); // Search inside
jcr:content
    predicateMap.put("p.limit", "10"); // Limit to 10 results
    predicateMap.put("orderby", "@jcr:content/jcr:title"); // Sort by title

    Query query =
queryBuilder.createQuery(PredicateGroup.create(predicateMap), session);
    SearchResult searchResult = query.getResult();

    List<Hit> hits = searchResult.getHits();
    for (Hit hit : hits) {
        JSONObject pageObject = new JSONObject();
        pageObject.put("path", hit.getPath());
        pageObject.put("title", hit.getProperties().get("jcr:title", "Untitled"));
        pageObject.put("description", hit.getProperties().get("jcr:description",
"No Description"));
        pageObject.put("tags", hit.getProperties().get("cq:tags", new
String[]{}));
        resultsArray.put(pageObject);
    }
    jsonResponse.put("results", resultsArray);
} catch (Exception e) {
    try {
        jsonResponse.put("error", "Search failed: " + e.getMessage());
    } catch (JSONException ex) {
        throw new RuntimeException(ex);
    }
}
}

```

```
response.getWriter().write(jsonResponse.toString());  
}  
}
```

