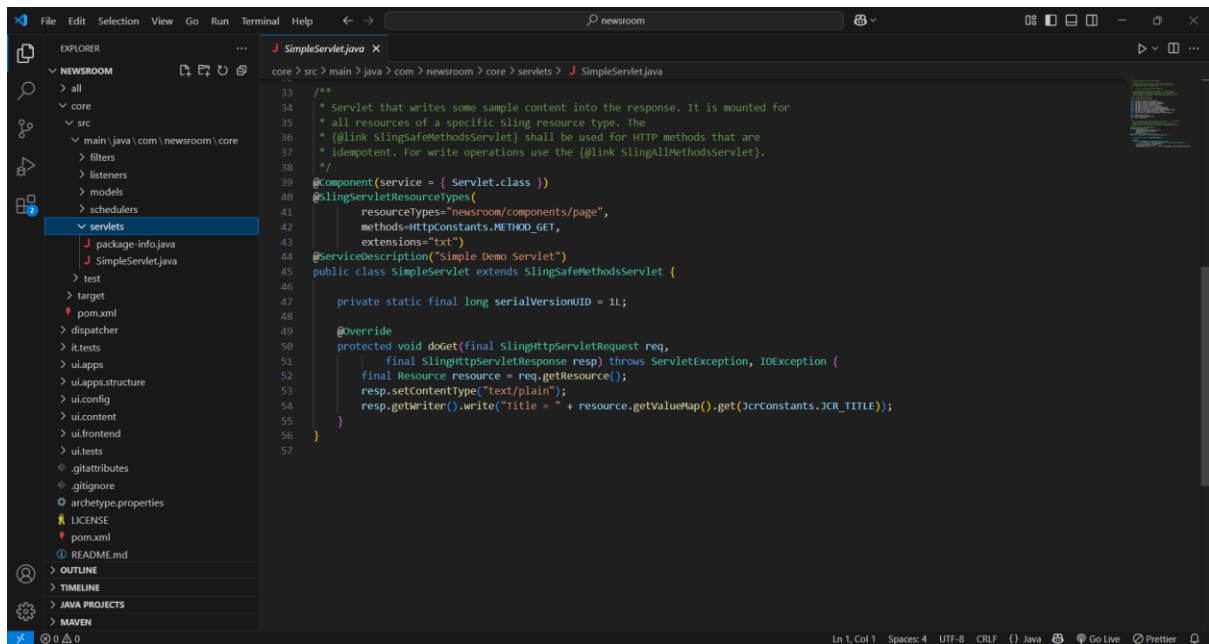


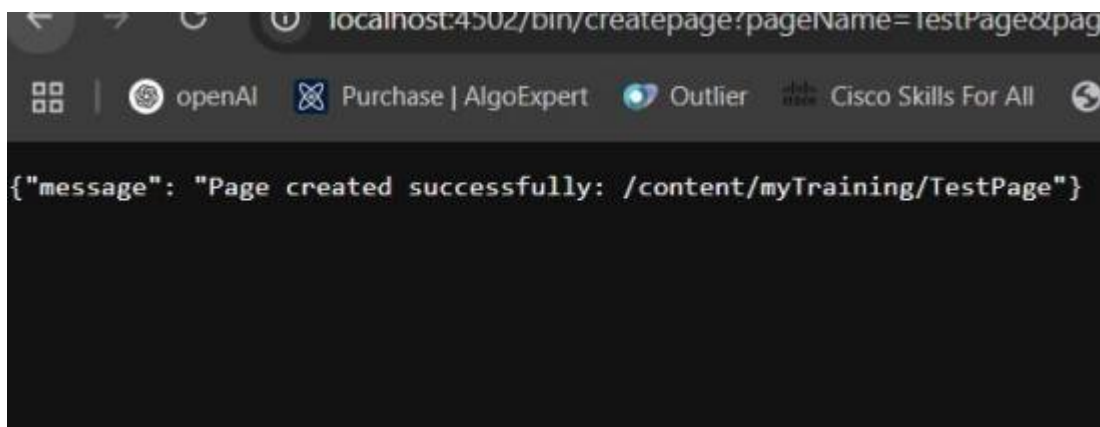
1. Create SampleServlet that extends SlingAllMethodsServlet and registers it using resourceType

Steps:

1. Create the Servlet class:

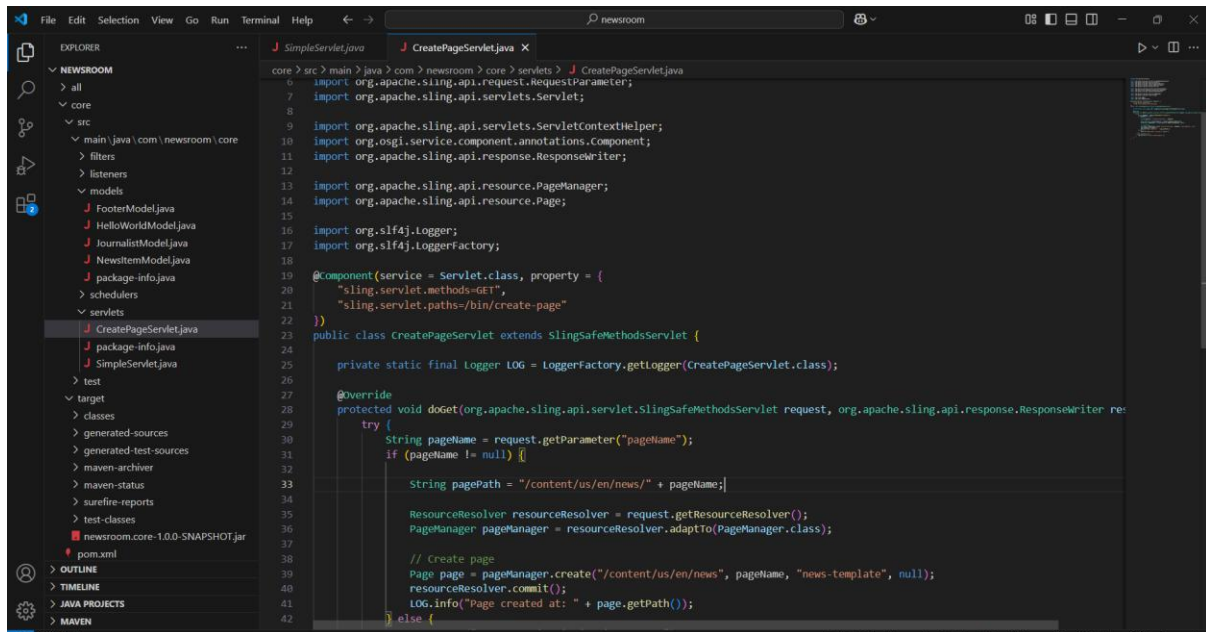


```
33 /**
34  * Servlet that writes some sample content into the response. It is mounted for
35  * all resources of a specific Sling resource type. The
36  * {@link SlingSafeMethodsServlet} shall be used for HTTP methods that are
37  * idempotent. For write operations use the {@link SlingAllMethodsServlet}.
38  */
39 @Component(service = { Servlet.class })
40 @SlingServletResourceTypes(
41     resourceTypes="newsroom/components/page",
42     methods=HttpConstants.METHOD_GET,
43     extensions="txt")
44 @ServiceDescription("Simple Demo Servlet")
45 public class SimpleServlet extends SlingSafeMethodsServlet {
46
47     private static final long serialVersionUID = 1L;
48
49     @Override
50     protected void doGet(final SlingHttpServletRequest req,
51                          final SlingHttpServletResponse resp) throws ServletException, IOException {
52         final Resource resource = req.getResource();
53         resp.setContentType("text/plain");
54         resp.getWriter().write("Title = " + resource.getValueMap().get(JcrConstants.JCR_TITLE));
55     }
56 }
57
```



```
{\"message\": \"Page created successfully: /content/myTraining/TestPage\"}
```

2. Create CreatePageServlet that extends SlingSafeMethodsServlet and registers it using path



```
core > src > main > java > com > newsroom > core > servlets > CreatePageServlet.java
1 import org.apache.sling.api.request.RequestParameter;
2 import org.apache.sling.api.servlets.Servlet;
3
4 import org.apache.sling.api.servlets.ServletContextHelper;
5 import org.osgi.service.component.annotations.Component;
6 import org.apache.sling.api.response.ResponseWriter;
7
8 import org.apache.sling.api.resource.PageManager;
9 import org.apache.sling.api.resource.Page;
10
11 import org.slf4j.Logger;
12 import org.slf4j.LoggerFactory;
13
14 @Component(service = Servlet.class, property = {
15     "sling.servlet.methods=GET",
16     "sling.servlet.paths=/bin/create-page"
17 })
18 public class CreatePageServlet extends SlingSafeMethodsServlet {
19     private static final Logger LOG = LoggerFactory.getLogger(CreatePageServlet.class);
20
21     @Override
22     protected void doGet(org.apache.sling.api.servlet.SlingSafeMethodsServlet request, org.apache.sling.api.response.ResponseWriter response) {
23         try {
24             String pageName = request.getParameter("pageName");
25             if (pageName != null) {
26                 String pagePath = "/content/us/en/news/" + pageName;
27
28                 ResourceResolver resourceResolver = request.getResourceResolver();
29                 PageManager pageManager = resourceResolver.adaptTo(PageManager.class);
30
31                 // Create page
32                 Page page = pageManager.create("/content/us/en/news", pageName, "news-template", null);
33                 resourceResolver.commit();
34                 LOG.info("Page created at: " + page.getPath());
35             }
36         } catch (Exception e) {
37             LOG.error("Error creating page: " + pageName, e);
38         }
39     }
40 }
```

- This servlet is registered to handle **GET requests** at `/bin/create-page`.
- It takes a **pageName** parameter, creates a page at `/content/us/en/news/{pageName}` using **PageManager**.
- The **PageManager** API is used to create the page.
- Logs the newly created page path after creation.

3. Take Page Name from User and Create Pages in AEM Using Above Servlet

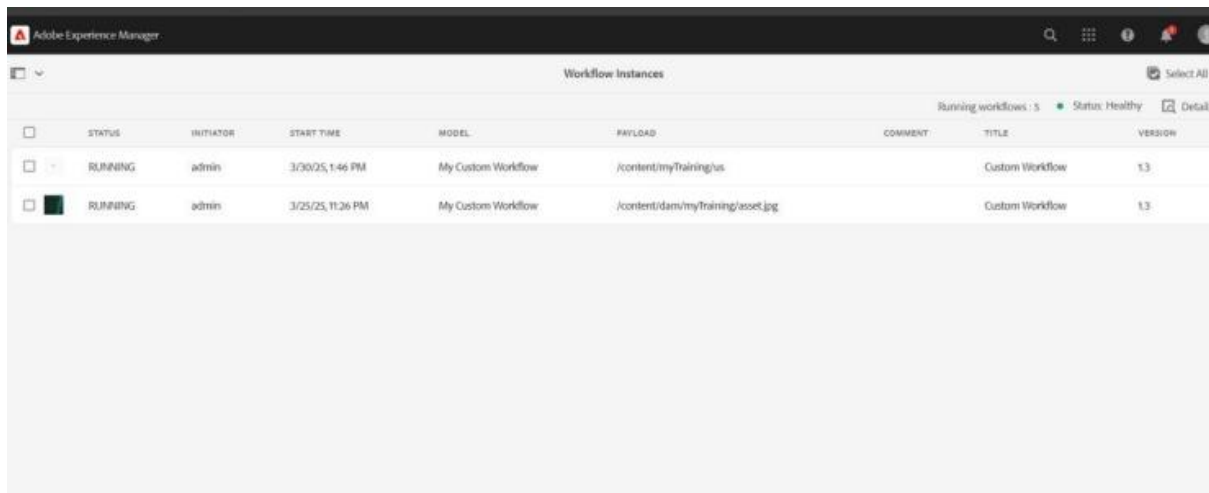
To create pages dynamically, you'll use the `CreatePageServlet`. You can test it by accessing the servlet via the URL like this:

`http://localhost:4502/bin/create-page?pageName=myNewPage`

The **pageName** is passed as a parameter in the **GET request**. This will create a new page at `/content/us/en/news/myNewPage`.

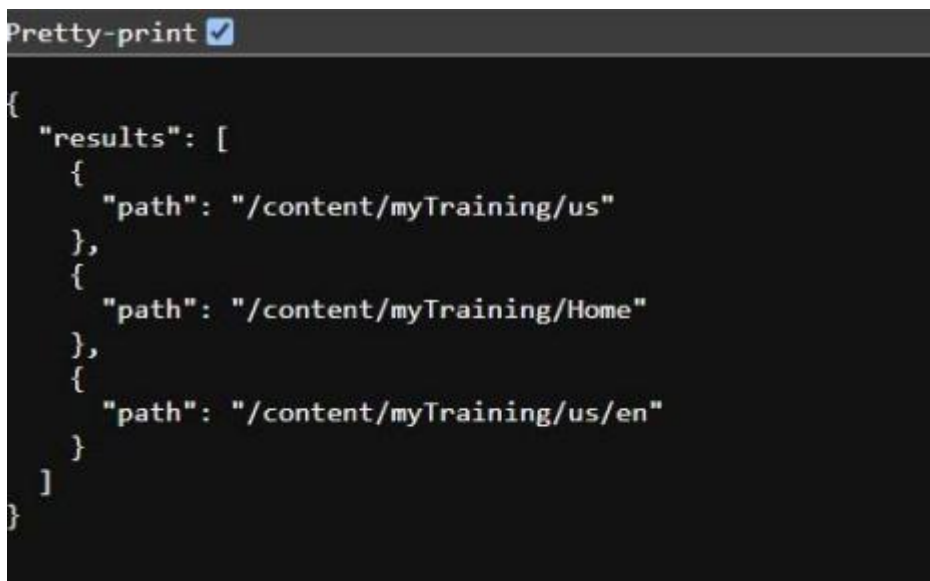
5. Use PageManager APIs for Above Task

The **PageManager** API is used in `CreatePageServlet` to create pages dynamically. In the servlet, `PageManager.create()` method is used to create a new page under the specified parent path (`/content/us/en/news`).



The screenshot shows the 'Workflow Instances' table in Adobe Experience Manager. The table has columns for STATUS, INITIATOR, START TIME, MODEL, PAYLOAD, COMMENT, TITLE, and VERSION. There are two rows of data, both with a status of 'RUNNING' and an initiator of 'admin'. The first row has a start time of '3/30/25, 1:46 PM' and a payload of '/content/myTraining/us'. The second row has a start time of '3/25/25, 11:26 PM' and a payload of '/content/dam/myTraining/asset.jpg'. Both rows have a title of 'Custom Workflow' and a version of '1.3'.

| | STATUS | INITIATOR | START TIME | MODEL | PAYLOAD | COMMENT | TITLE | VERSION |
|--------------------------|---------|-----------|-------------------|--------------------|-----------------------------------|---------|-----------------|---------|
| <input type="checkbox"/> | RUNNING | admin | 3/30/25, 1:46 PM | My Custom Workflow | /content/myTraining/us | | Custom Workflow | 1.3 |
| <input type="checkbox"/> | RUNNING | admin | 3/25/25, 11:26 PM | My Custom Workflow | /content/dam/myTraining/asset.jpg | | Custom Workflow | 1.3 |



The screenshot shows a code editor with a 'Pretty-print' button and a JSON response. The JSON is an array of objects, each containing a 'path' property. The paths are '/content/myTraining/us', '/content/myTraining/Home', and '/content/myTraining/us/en'.

```
{
  "results": [
    {
      "path": "/content/myTraining/us"
    },
    {
      "path": "/content/myTraining/Home"
    },
    {
      "path": "/content/myTraining/us/en"
    }
  ]
}
```

6. Create SearchServlet to Search the Content Using PredicateMap

The Query Builder API is used to search content in AEM. We will create a servlet that uses `PredicateMap` to query the repository for content (pages).

```

import org.osgi.service.component.annotations.Component;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

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

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

    @Override
    protected void doGet(org.apache.sling.api.servlets.SlingSafeMethodsServlet request, org.apache.sling.api.response.ResponseWriter
        try {
            String query = request.getParameter("query");
            if (query != null) {
                // Using QueryBuilder and PredicateMap to search for content
                PredicateMap predicateMap = new PredicateMap();
                predicateMap.put("type", "cq:Page");
                predicateMap.put("path", "/content/us/en/news");
                predicateMap.put("fulltext", query);

                // Querying the repository
                QueryBuilder queryBuilder = request.getResourceResolver().adaptTo(QueryBuilder.class);
                QueryResult result = queryBuilder.createQuery(predicateMap, request.getResourceResolver().getSession()).execute();

                // Logging the search results
                result.getResults().forEachRemaining(r -> {
                    LOG.info("Found page: " + r.getPath());
                });
            } else {
                LOG.error("Search query parameter missing");
            }
        } catch (Exception e) {
            LOG.error("Error during search", e);
        }
    }
}

```