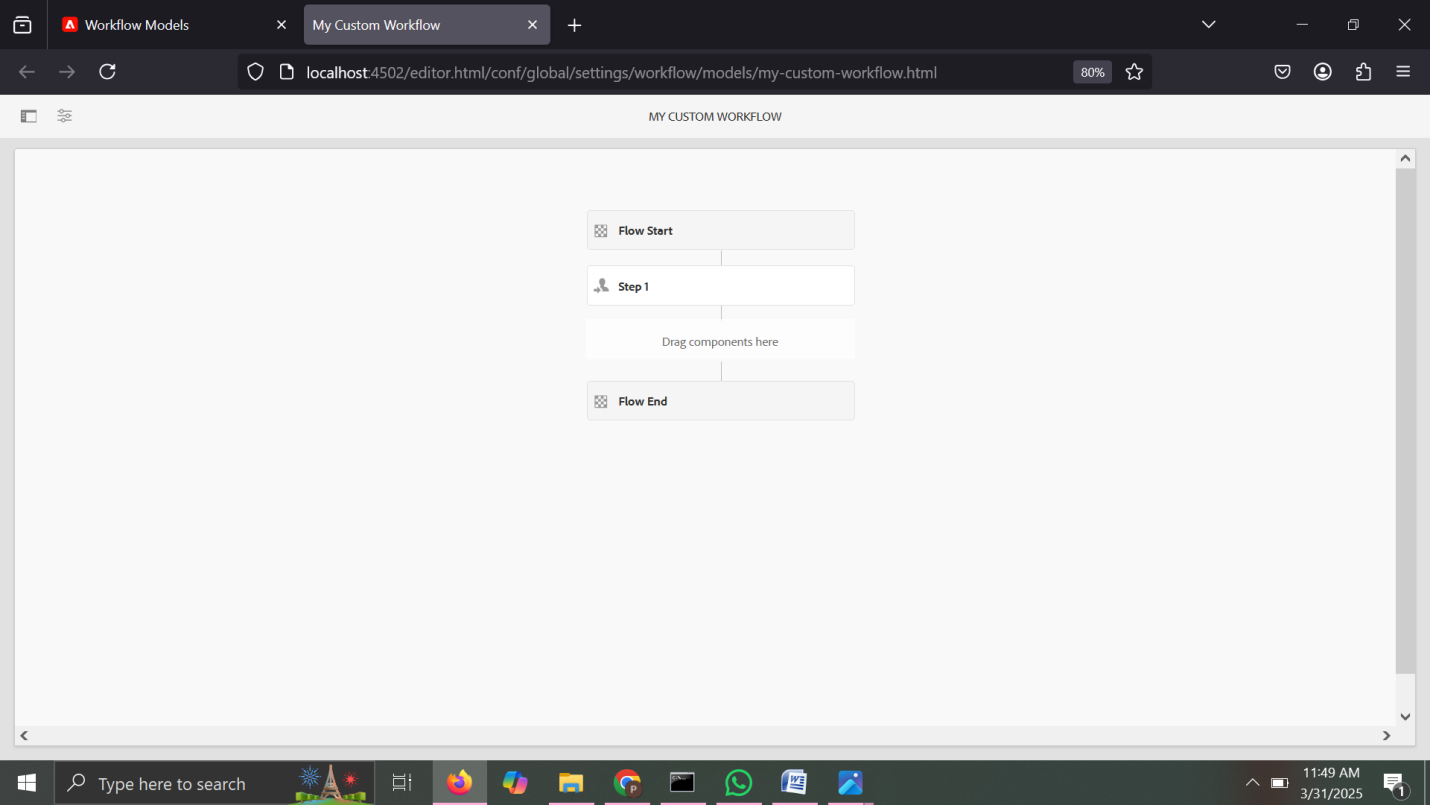
**1.Create Custom Workflow (my custom workflow)**

****

**2.Create custom workflow process and print the page title in logs and run this workflow in page so that it can give some metadata in logs**

package com.myTraining.core.workflow;  
  
import com.adobe.granite.workflow.WorkflowSession;  
import com.adobe.granite.workflow.exec.WorkItem;  
import com.adobe.granite.workflow.exec.WorkflowProcess;  
import com.adobe.granite.workflow.metadata.MetaDataMap;  
import org.apache.sling.api.resource.Resource;  
import org.apache.sling.api.resource.ResourceResolver;  
import org.apache.sling.api.resource.ValueMap;  
import org.osgi.service.component.annotations.Component;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@Component(service = WorkflowProcess.class, property = {"process.label=Log Page Title and Metadata"})  
public class LogPageTitleWorkflowProcess implements WorkflowProcess {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(LogPageTitleWorkflowProcess.class);  
  
 @Override  
 public void execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap) {  
 String pagePath = workItem.getWorkflowData().getPayload().toString();  
  
 try (ResourceResolver resourceResolver = workflowSession.adaptTo(ResourceResolver.class)) {  
 if (resourceResolver != null) {  
 Resource pageResource = resourceResolver.getResource(pagePath + "/jcr:content");  
  
 if (pageResource != null) {  
 ValueMap properties = pageResource.getValueMap();  
 String pageTitle = properties.get("jcr:title", "No Title");  
 String pageDescription = properties.get("jcr:description", "No Description");  
  
 *LOG*.info("Executing workflow for page: {}", pagePath);  
 *LOG*.info("Page Title: {}", pageTitle);  
 *LOG*.info("Page Description: {}", pageDescription);  
 } else {  
 *LOG*.warn("Page content node not found for path: {}", pagePath);  
 }  
 } else {  
 *LOG*.error("ResourceResolver is null.");  
 }  
 } catch (Exception e) {  
 *LOG*.error("Error while logging page metadata", e);  
 }  
 }  
}

package com.myTraining.core.listeners;  
  
import org.apache.sling.api.resource.Resource;  
import org.apache.sling.api.resource.ResourceResolver;  
import org.apache.sling.api.resource.ResourceResolverFactory;  
import org.osgi.service.component.annotations.Activate;  
import org.osgi.service.component.annotations.Component;  
import org.osgi.service.component.annotations.Reference;  
import org.osgi.service.event.Event;  
import org.osgi.service.event.EventConstants;  
import org.osgi.service.event.EventHandler;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@Component(  
 service = EventHandler.class,  
 immediate = true,  
 property = {  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceAdded",  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceChanged",  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceRemoved"  
 }  
)  
public class PageEventHandler implements EventHandler {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(PageEventHandler.class);  
  
 @Reference  
 private ResourceResolverFactory resourceResolverFactory;  
  
 @Activate  
 protected void activate() {  
 *LOG*.info("✅ AEM Page Event Handler Activated");  
 }  
  
 @Override  
 public void handleEvent(Event event) {  
 String resourcePath = (String) event.getProperty("path");  
 String eventType = event.getTopic();  
  
 *LOG*.info("🔹 Event Type: {} | Resource Path: {}", eventType, resourcePath);  
  
 // Use ResourceResolver if needed  
 try (ResourceResolver resolver = getResourceResolver()) {  
 if (resolver != null) {  
 Resource resource = resolver.getResource(resourcePath);  
 if (resource != null) {  
 *LOG*.info("✅ Resource Found: {}", resource.getPath());  
 } else {  
 *LOG*.warn("⚠️ Resource Not Found at Path: {}", resourcePath);  
 }  
 }  
 } catch (Exception e) {  
 *LOG*.error("❌ Error in Event Handler", e);  
 }  
 }  
  
 private ResourceResolver getResourceResolver() {  
 try {  
 Map<String, Object> param = new HashMap<>();  
 param.put(ResourceResolverFactory.*SUBSERVICE*, "readService");  
 return resourceResolverFactory.getServiceResourceResolver(param);  
 } catch (Exception e) {  
 *LOG*.error("❌ Failed to Get Resource Resolver", e);  
 return null;  
 }  
 }  
}

**3.Create Event handler in aem and print the resource path in logs.**

package com.myTraining.core.listeners;  
  
import org.apache.sling.api.resource.Resource;  
import org.apache.sling.api.resource.ResourceResolver;  
import org.apache.sling.api.resource.ResourceResolverFactory;  
import org.osgi.service.component.annotations.Activate;  
import org.osgi.service.component.annotations.Component;  
import org.osgi.service.component.annotations.Reference;  
import org.osgi.service.event.Event;  
import org.osgi.service.event.EventConstants;  
import org.osgi.service.event.EventHandler;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@Component(  
 service = EventHandler.class,  
 immediate = true,  
 property = {  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceAdded",  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceChanged",  
 EventConstants.*EVENT\_TOPIC* + "=org/apache/sling/api/resource/ResourceRemoved"  
 }  
)  
public class PageEventHandler implements EventHandler {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(PageEventHandler.class);  
  
 @Reference  
 private ResourceResolverFactory resourceResolverFactory;  
  
 @Activate  
 protected void activate() {  
 *LOG*.info("✅ AEM Page Event Handler Activated");  
 }  
  
 @Override  
 public void handleEvent(Event event) {  
 String resourcePath = (String) event.getProperty("path");  
 String eventType = event.getTopic();  
  
 *LOG*.info("🔹 Event Type: {} | Resource Path: {}", eventType, resourcePath);  
  
 // Use ResourceResolver if needed  
 try (ResourceResolver resolver = getResourceResolver()) {  
 if (resolver != null) {  
 Resource resource = resolver.getResource(resourcePath);  
 if (resource != null) {  
 *LOG*.info("✅ Resource Found: {}", resource.getPath());  
 } else {  
 *LOG*.warn("⚠️ Resource Not Found at Path: {}", resourcePath);  
 }  
 }  
 } catch (Exception e) {  
 *LOG*.error("❌ Error in Event Handler", e);  
 }  
 }  
  
 private ResourceResolver getResourceResolver() {  
 try {  
 Map<String, Object> param = new HashMap<>();  
 param.put(ResourceResolverFactory.*SUBSERVICE*, "readService");  
 return resourceResolverFactory.getServiceResourceResolver(param);  
 } catch (Exception e) {  
 *LOG*.error("❌ Failed to Get Resource Resolver", e);  
 return null;  
 }  
 }  
}

**4.create sling job to print hello world message in logs**

package com.myTraining.core.jobs;  
  
import org.apache.sling.event.jobs.Job;  
import org.apache.sling.event.jobs.consumer.JobConsumer;  
import org.osgi.service.component.annotations.Component;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@Component(  
 service = JobConsumer.class,  
 property = {  
 JobConsumer.*PROPERTY\_TOPICS* + "=com/myTraining/slingjob/hello"  
 }  
)  
public class HelloWorldSlingJob implements JobConsumer {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(HelloWorldSlingJob.class);  
  
 @Override  
 public JobResult process(Job job) {  
 *LOG*.info("✅ Sling Job Executed: Hello, World!");  
 return JobResult.*OK*;  
 }  
}

**5. Create one schedular to print the yellow world in logs in every 5 mins through custom configuration using cron expression.**

package com.myTraining.core.jobs;  
  
import org.apache.sling.event.jobs.Job;  
import org.apache.sling.event.jobs.consumer.JobConsumer;  
import org.osgi.service.component.annotations.Component;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@Component(  
 service = JobConsumer.class,  
 property = {  
 JobConsumer.*PROPERTY\_TOPICS* + "=com/myTraining/slingjob/hello"  
 }  
)  
public class HelloWorldSlingJob implements JobConsumer {  
  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(HelloWorldSlingJob.class);  
  
 @Override  
 public JobResult process(Job job) {  
 *LOG*.info("✅ Sling Job Executed: Hello, World!");  
 return JobResult.*OK*;  
 }  
}