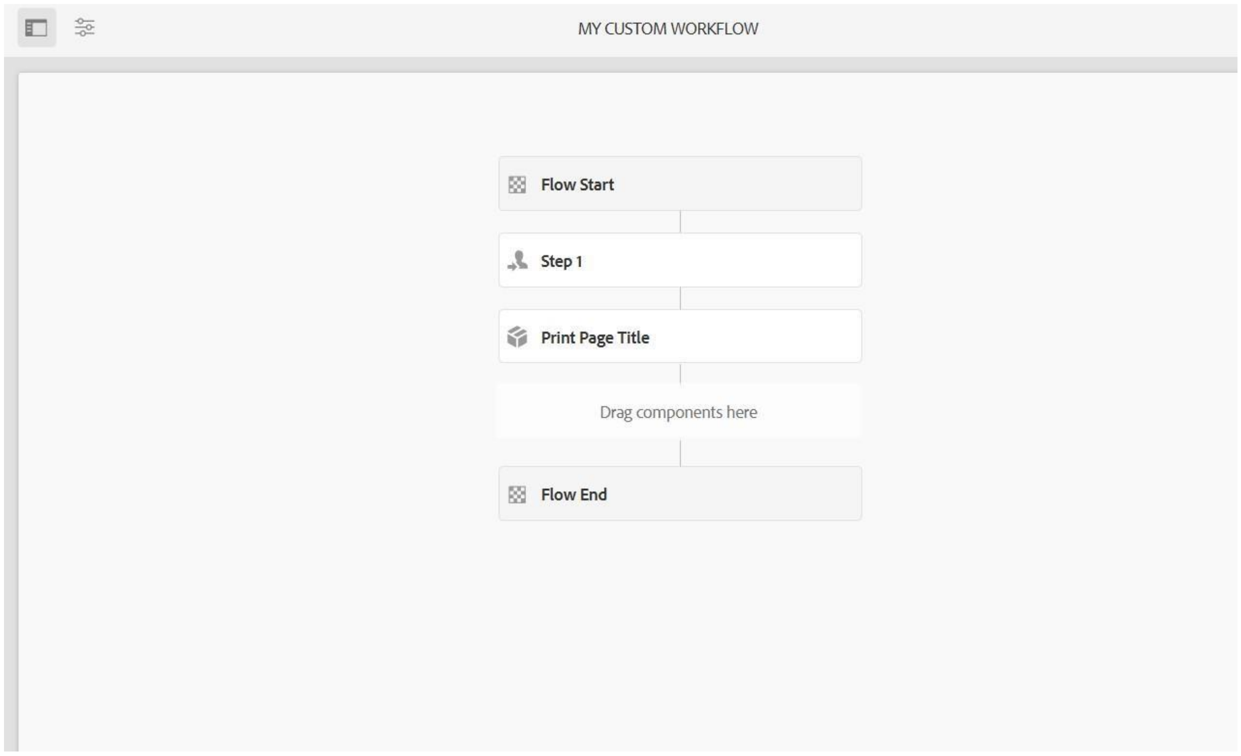
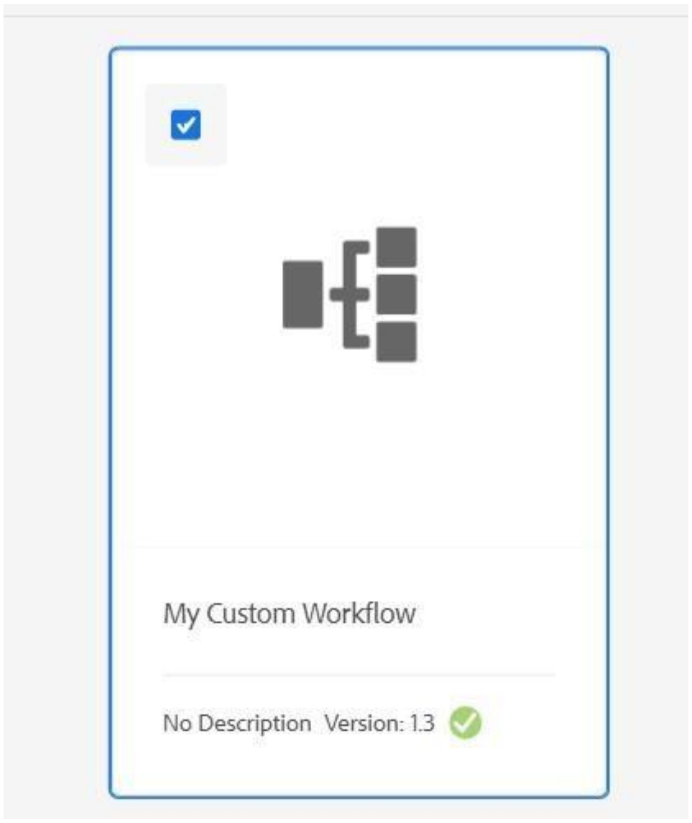
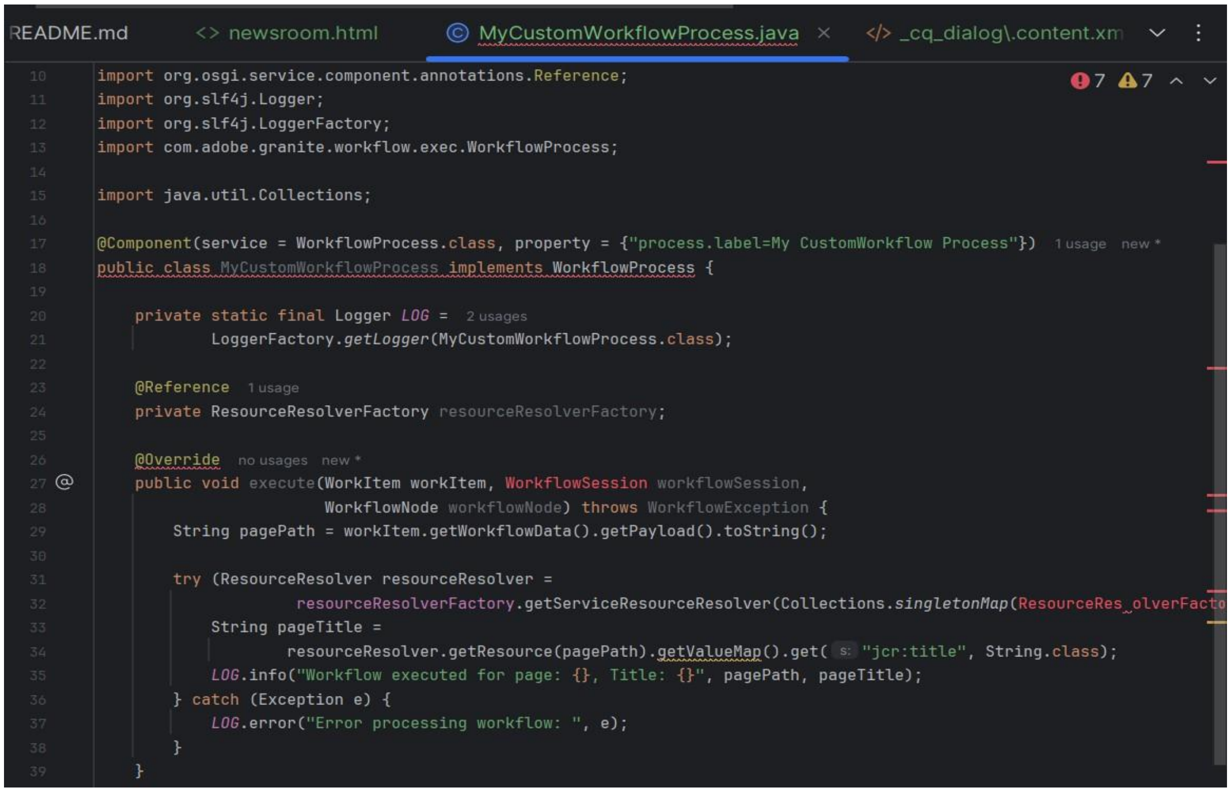
1.Create Custom Workflow (my custom workflow)



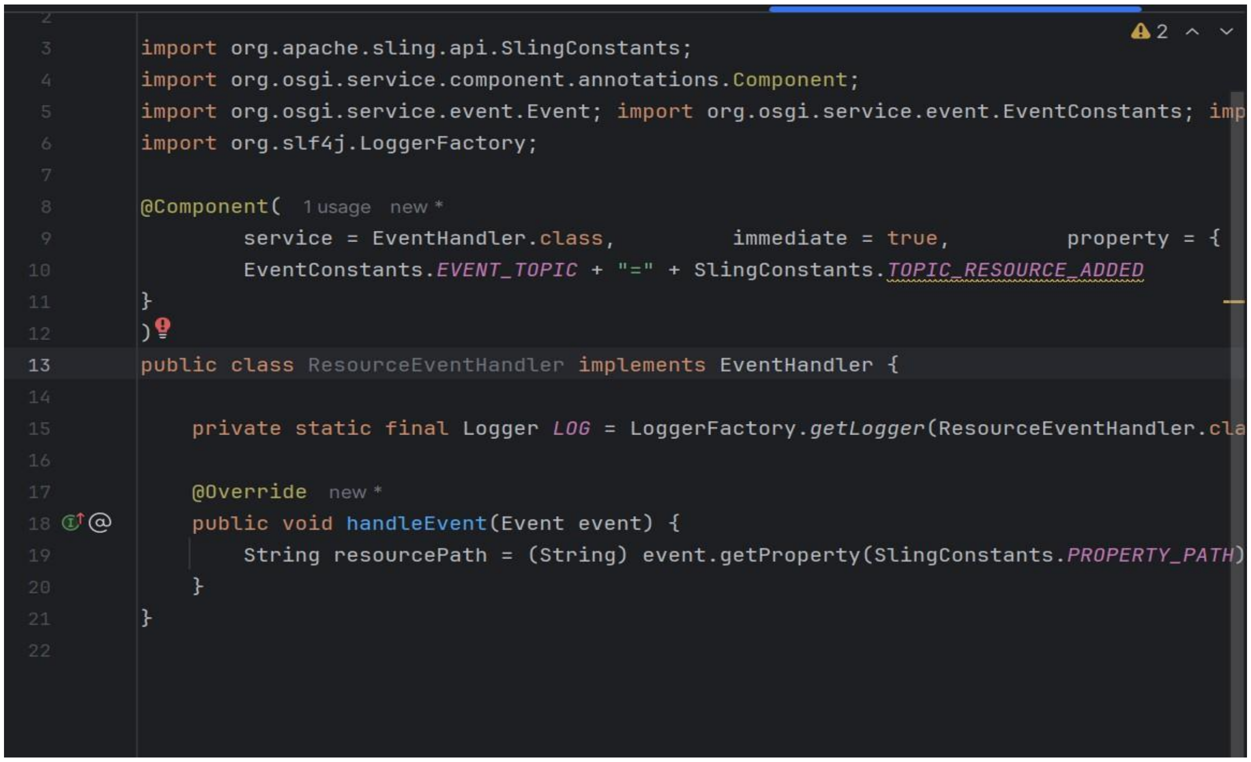


2.Create custom workflow process and print the page title in logs and run this workflow in page so that it can give some metadata in logs



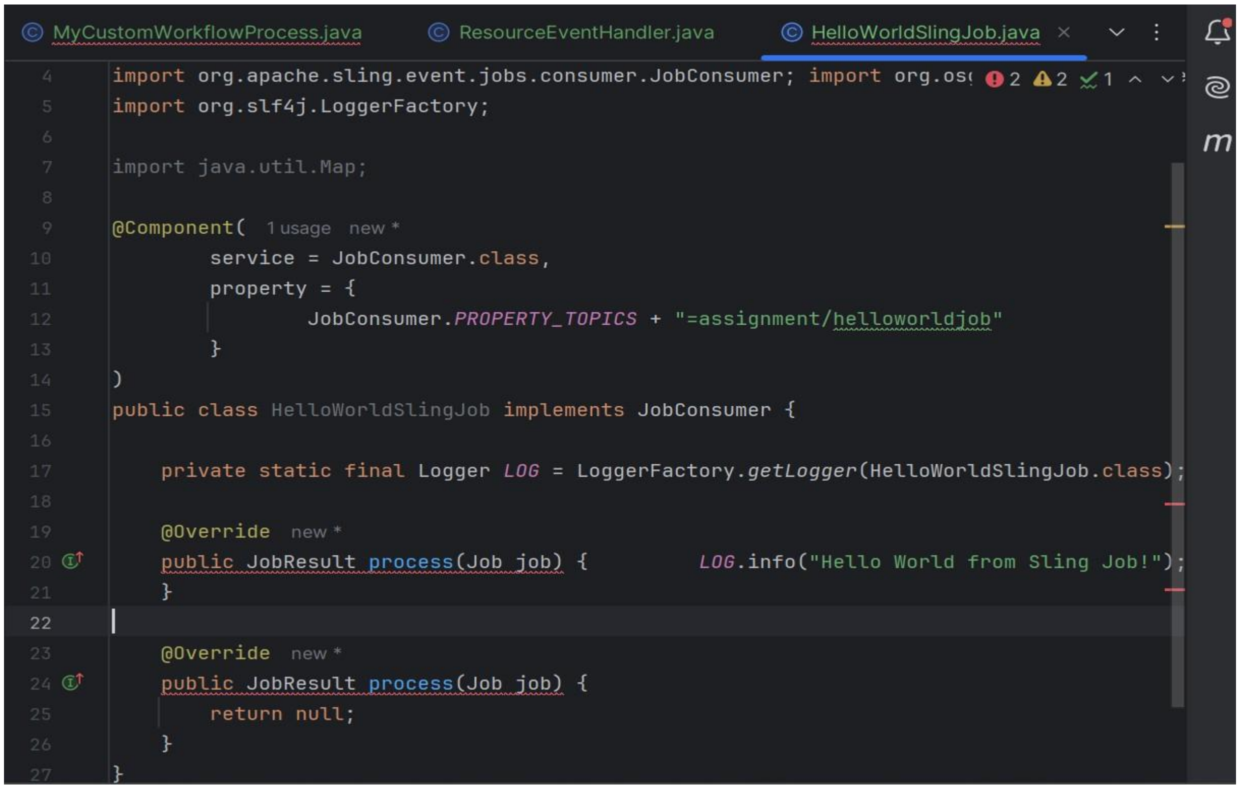
```java
import org.osgi.service.component.annotations.Reference;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import com.adobe.granite.workflow.exec.WorkflowProcess;

import java.util.Collections;

@Component(service = WorkflowProcess.class, property = {"process.label=My CustomWorkflow Process"})  1 usage  new *
public class MyCustomWorkflowProcess implements WorkflowProcess {

    private static final Logger LOG =  2 usages
            LoggerFactory.getLogger(MyCustomWorkflowProcess.class);

    @Reference  1 usage
    private ResourceResolverFactory resourceResolverFactory;

    @Override  no usages  new *
    public void execute(WorkItem workItem, WorkflowSession workflowSession,
                        WorkflowNode workflowNode) throws WorkflowException {
        String pagePath = workItem.getWorkflowData().getPayload().toString();

        try (ResourceResolver resourceResolver =
                     resourceResolverFactory.getServiceResourceResolver(Collections.singletonMap(ResourceRes olverFacto
            String pageTitle =
                    resourceResolver.getResource(pagePath).getValueMap().get( s: "jcr:title", String.class);
            LOG.info("Workflow executed for page: {}, Title: {}", pagePath, pageTitle);
        } catch (Exception e) {
            LOG.error("Error processing workflow: ", e);
        }
    }
```

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



```java
import org.apache.sling.api.SlingConstants;
import org.osgi.service.component.annotations.Component;
import org.osgi.service.event.Event; import org.osgi.service.event.EventConstants; imp
import org.slf4j.LoggerFactory;

@Component(  1 usage  new *
        service = EventHandler.class,          immediate = true,          property = {
        EventConstants.EVENT_TOPIC + "=" + SlingConstants.TOPIC_RESOURCE_ADDED
}
)
public class ResourceEventHandler implements EventHandler {

    private static final Logger LOG = LoggerFactory.getLogger(ResourceEventHandler.cla

    @Override  new *
    public void handleEvent(Event event) {
        String resourcePath = (String) event.getProperty(SlingConstants.PROPERTY_PATH)
    }
}
```

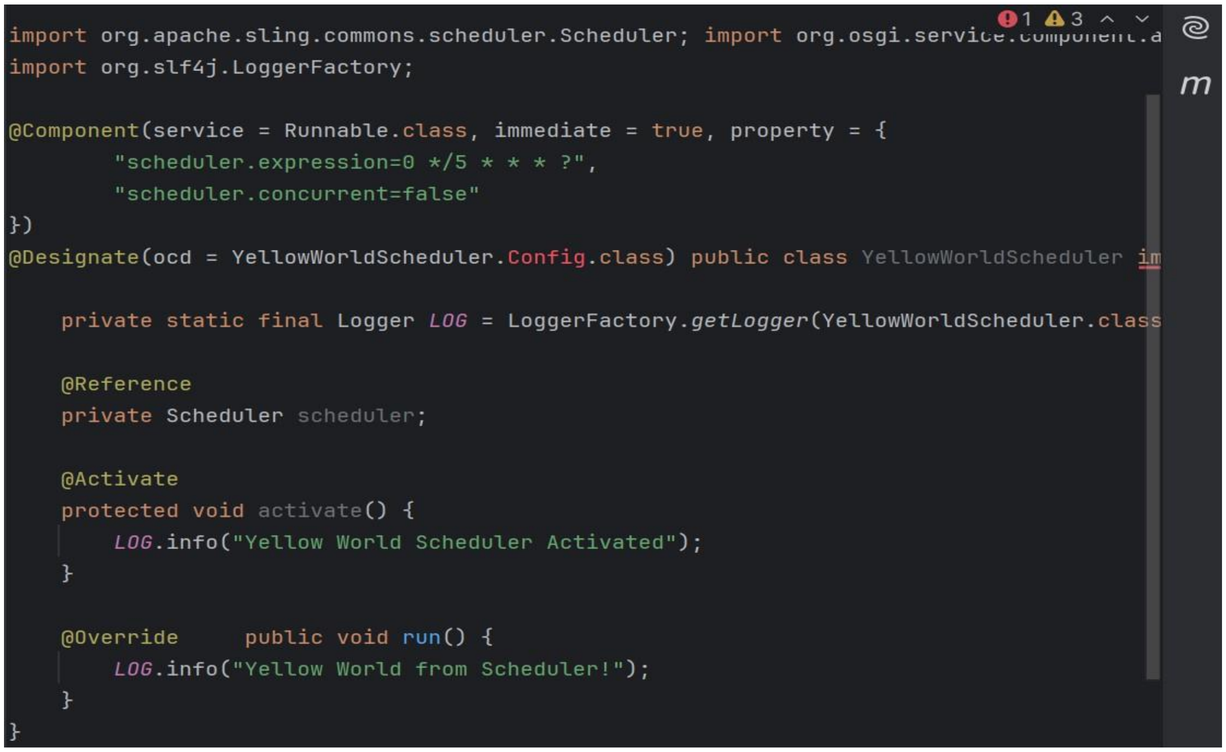
4.create sling job to print hello world message in logs



MyCustomWorkflowProcess.java   ResourceEventHandler.java   HelloWorldSlingJob.java

```
import org.apache.sling.event.jobs.consumer.JobConsumer; import org.os(
import org.slf4j.LoggerFactory;

import java.util.Map;

@Component(  1 usage  new *
        service = JobConsumer.class,
        property = {
                JobConsumer.PROPERTY_TOPICS + "=assignment/helloworldjob"
        }
)
public class HelloWorldSlingJob implements JobConsumer {

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

    @Override  new *
    public JobResult process(Job job) {          LOG.info("Hello World from Sling Job!");
    }

    @Override  new *
    public JobResult process(Job job) {
        return null;
    }
}
```

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



```
import org.apache.sling.commons.scheduler.Scheduler; import org.osgi.service.component.a
import org.slf4j.LoggerFactory;

@Component(service = Runnable.class, immediate = true, property = {
        "scheduler.expression=0 */5 * * * ?",
        "scheduler.concurrent=false"
})
@Designate(ocd = YellowWorldScheduler.Config.class) public class YellowWorldScheduler im

    private static final Logger LOG = LoggerFactory.getLogger(YellowWorldScheduler.class

    @Reference
    private Scheduler scheduler;

    @Activate
    protected void activate() {
        LOG.info("Yellow World Scheduler Activated");
    }

    @Override      public void run() {
        LOG.info("Yellow World from Scheduler!");
    }
}
```

5. Create 3 users and add them in a group (Dev author create this new group) and give permission to read only for /content and /dam folder only and they should have replication access as well.

Go to http://localhost:4502/useradmin

1. Click "Create User" ○ User 1: user1 ○ User 2: user2 ○ User 3: user3
2. Click "Create Group" → Name: "Dev Author"
3. Add user1, user2, and user3 to "Dev Author" group

Assign Permissions

1. Select "Dev Author" Group
2. Go to "Permissions" Tab
3. Set Read-Only Access to /content and /dam ○ Path: /content → Permission: Read ○ Path: /dam → Permission: Read
4. Enable Replication Access ○ Path: /etc/replication → Allow Replication