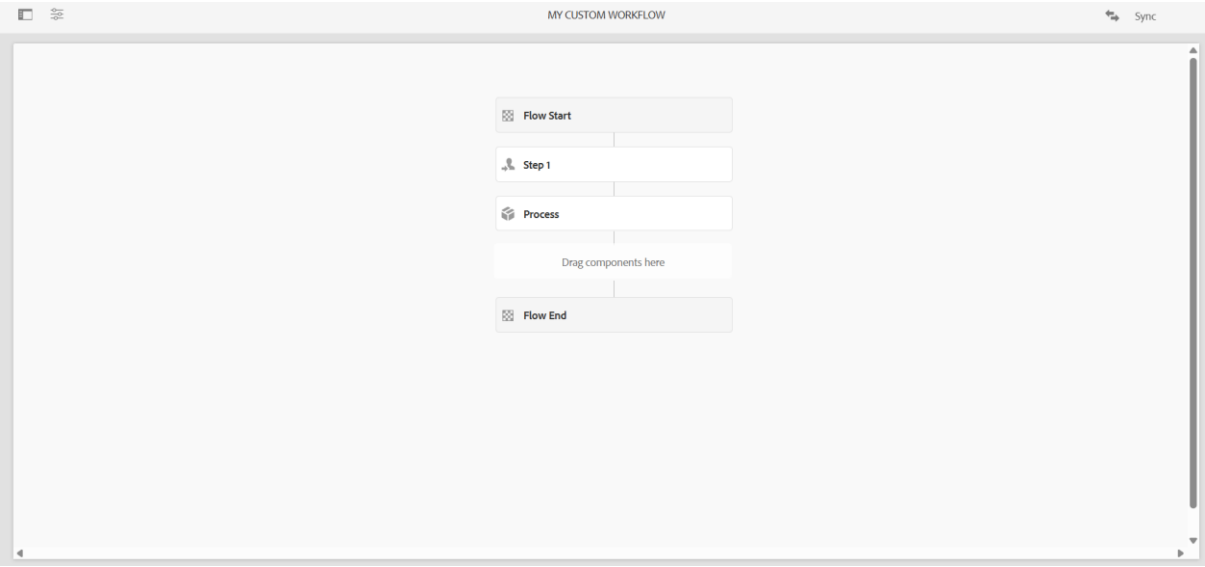
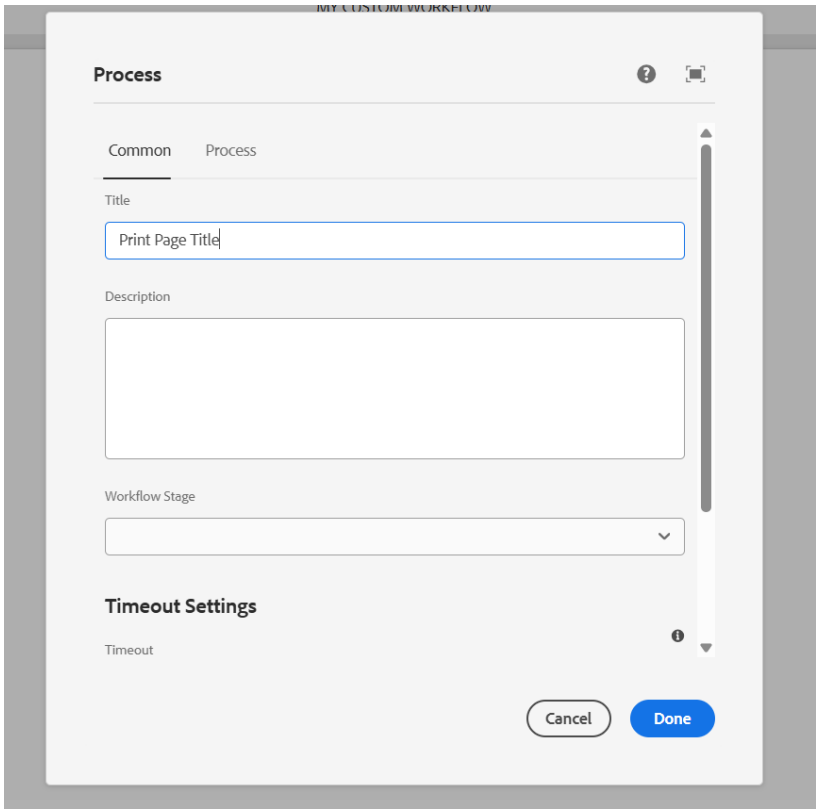
1. Create Custom Workflow (my custom workflow).
Go to Tools > Workflow > Models.
Click on "Create".
Enter the title: my custom workflow.
Click on "Create & Open".
Add a Process Step to the workflow model.
Configure the Process Step:

- Title: Custom Process Step
- Process: CustomWorkflowProcess
  Save the 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

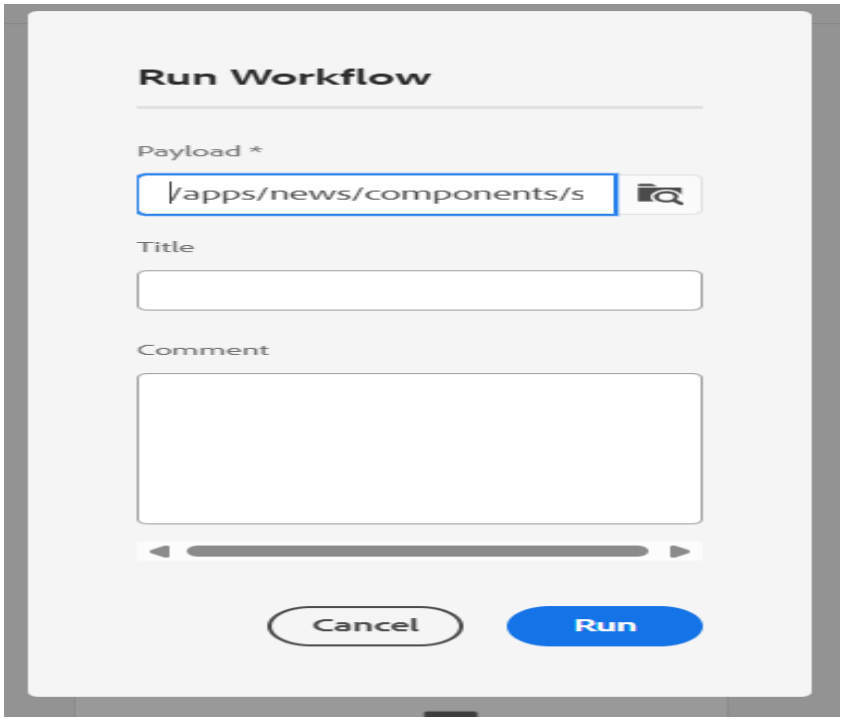


Configure the Process Step :
In Process Tab : Provide a custom class name like com.example.core.workflows.PrintPageTitleProcess
Title: "Print Page Title Step

Create a Workflow Process Java Class Go to AEM project: /core/src/main/java/com/myTraining/core/workflows/PrintPageTitleProcess.java

```
@Component(service = WorkflowProcess.class, property = {"process.label=Print Page Title"})
 public class PrintPageTitleProcess implements WorkflowProcess {
private static final Logger LOG = LoggerFactory.getLogger(PrintPageTitleProcess.class);
@Override
public void execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap) {
try {
String path = workItem.getWorkflowData().getPayload().toString();
ResourceResolver resolver = workflowSession.adaptTo(ResourceResolver.class);
if (resolver != null) {
 Resource resource = resolver.getResource(path + "/jcr:content");
 if (resource != null) {
 String title = resource.getValueMap().get("jcr:title", String.class);
LOG.info("Page Title: {}", title);
}
else {
LOG.warn("Resource not found at: {}", path);
} } }
catch (Exception e)
 {
 LOG.error("Error executing workflow process: ", e);
} } }
```

Run the workflow:



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

Create a java file named CustomeventHandler.java

C:\Users\LENOVO\training\aem\codebase\myTraining\core\src\main\java\com\myTraining\core\li steners\CustomeventHandler.java

```
@Component(
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.class);
@Override
public void handleEvent(Event event) {
String resourcePath = (String) event.getProperty(SlingConstants.PROPERTY_PATH);
LOG.info("Resource added at path: {}", resourcePath);
} }
```

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

Create HelloWorldSlingJob.java file inside the following path core/src/main/java/myTraining /jobs/HelloWorldSlingJob.java

```
Package com.muTraining.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 = {"job.topics=custom/job/helloworld"})
 public class CustomSlingJob implements JobConsumer {
 private static final Logger LOG = LoggerFactory.getLogger(CustomSlingJob.class);
@Override
public JobResult process (Job job) {
LOG.info("Hello World from Sling Job!");
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.commons.scheduler.Scheduler;
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.metatype.annotations.Designate;
import org.slf4j.Logger;
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 implements Runnable {
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!");
 }}
```

6.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 AEM as an Administrator (http://localhost:4502). Navigate to

Tools → Security → Groups

Click Create → Create Group.

Enter Group Name: eg:dev-authors

Click Save

Assign Permissions to the Group:

Navigate to Tools → Security → Permissions.

Click on the group and set Read-Only Access.