25.3.25

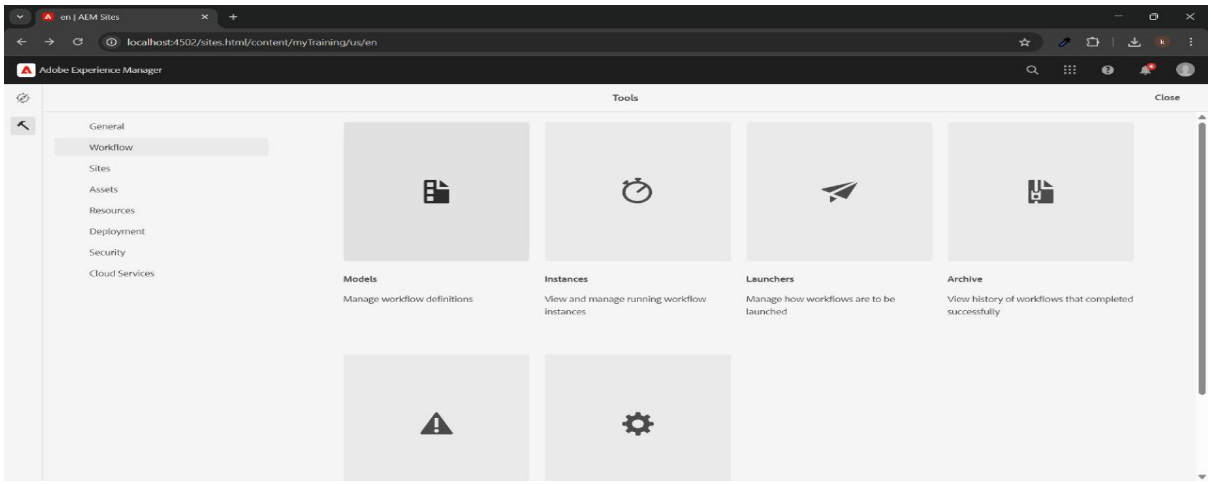
1. Create Custom Workflow

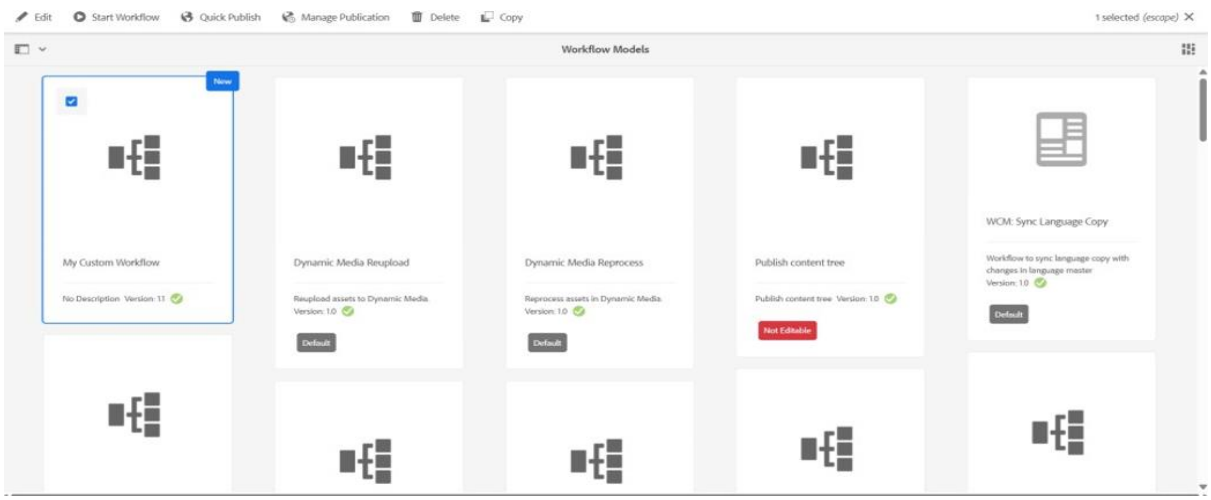
Step 1:
Navigate to Tools → Workflow → Models in AEM.

Step 2:
Click on the "Create" button to start a new workflow.

Step 3:
In the pop-up window, enter the title as "My Custom Workflow".

Step 4:
Click on "Create & Open" to start designing your custom workflow.

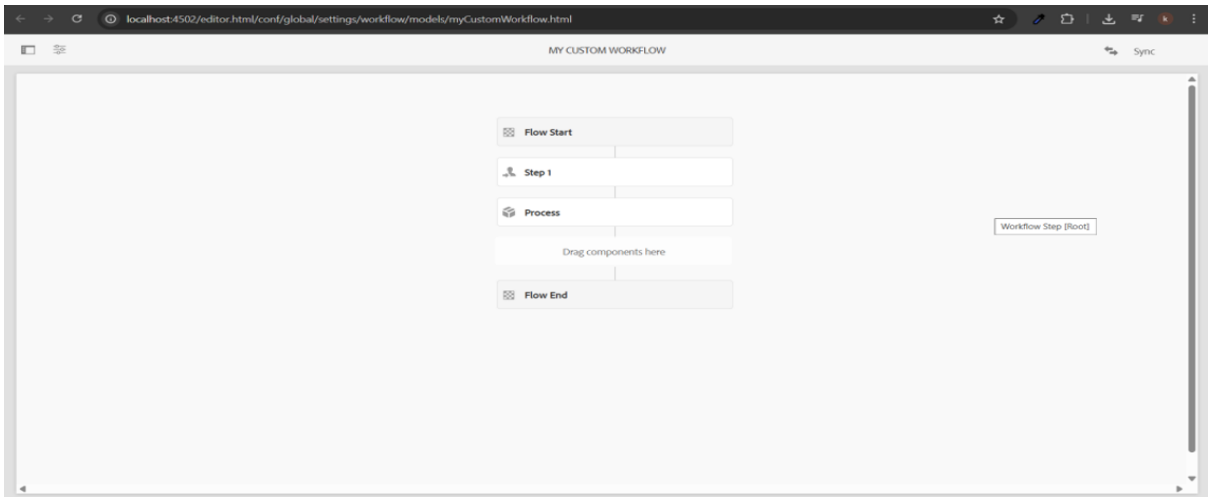




2. Create Custom Workflow Process

- Go to Tools → Workflow → Models
- Click "Create" → Enter the title: "Custom Workflow Process"
- Click "Create & Open" to start designing the workflow.
- Add a "Step" and name it, for example, "Custom Processing Step".

- In the step properties, choose "Custom Workflow Process".



3. Creating Event Handler in AEM

An **Event Handler** in AEM is used to listen for changes in the JCR repository, such as **node creation, modification, or deletion**. This allows you to trigger custom logic automatically when specific events occur.

```
package com.example.core.listeners;

import org.osgi.service.component.annotations.Component;

import org.osgi.service.event.Event;

import org.osgi.service.event.EventHandler;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

@Component(
    service = EventHandler.class,
    property = {
        "event.topics=org/apache/sling/api/resource/Resource/ADDED"
    }
)

public class ResourceAdditionListener implements EventHandler {

private static final Logger LOGGER =
LoggerFactory.getLogger(ResourceAdditionListener.class);

@Override
    public void handleEvent(Event event) {
```

```
        String addedResourcePath = (String) event.getProperty("path");
  LOGGER.info(" A new resource was added at the following path: {}", addedResourcePath);
    }
}
```

4. Create sling job to print hello world messages in logs

```
package com.example.core.jobs;
import org.apache.sling.event.jobs.Job;
import org.apache.sling.event.jobs.JobConsumer;
import org.apache.sling.event.jobs.JobResult;
import org.osgi.service.component.annotations.Component;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
@Component(
    service = JobConsumer.class,
    property = {
        JobConsumer.PROPERTY_TOPICS + "=my/custom/job"
    }
)
public class HelloWorldJob implements JobConsumer {
private static final Logger LOGGER = LoggerFactory.getLogger(HelloWorldJob.class);
@Override
    public JobResult process(Job job) {
        LOGGER.info("Hello World from the Sling Job Processor!");
        return JobResult.OK;
    }
}
```

5. Set up a scheduler to log "Yellow World" every 5 minutes using a custom configuration with a cron expression.

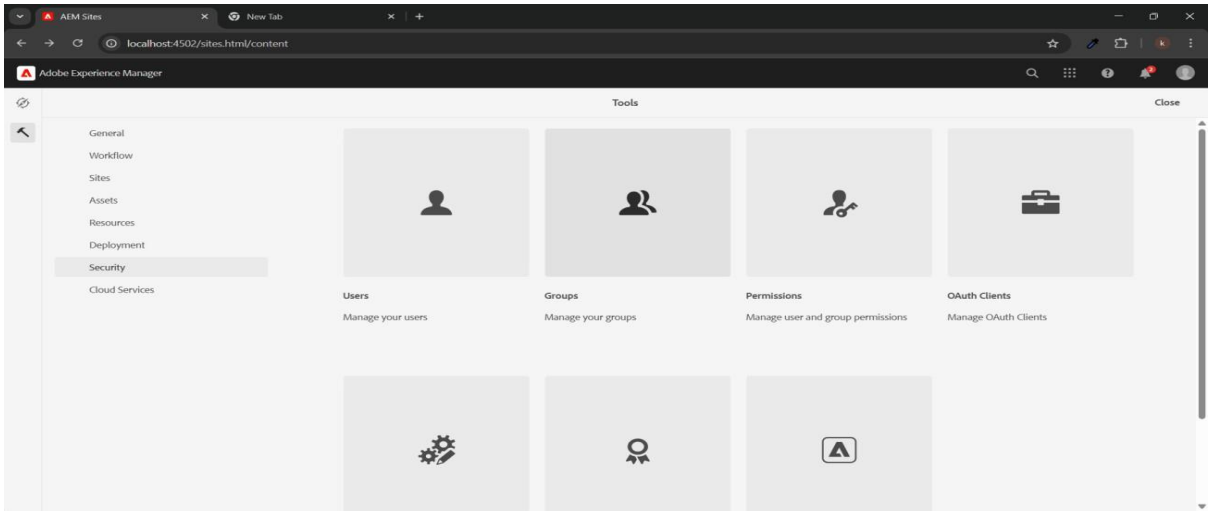
```
package com.example.core.schedulers;
```

```
import org.osgi.service.component.annotations.Component;
import org.osgi.service.component.annotations.Activate;
import org.osgi.service.component.annotations.Modified;
import org.osgi.service.metatype.annotations.Designate;
import org.osgi.service.metatype.annotations.ObjectClassDefinition;
import org.osgi.service.metatype.annotations.AttributeDefinition;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
@Component(
    service = Runnable.class,
    property = {
        "scheduler.expression=0 0/5 * * * ?",
        "scheduler.concurrent=false"
    }
)
@Designate(ocd = YellowWorldScheduler.SchedulerConfig.class)
public class YellowWorldScheduler implements Runnable {
private static final Logger LOGGER =
LoggerFactory.getLogger(YellowWorldScheduler.class);
@ObjectClassDefinition(name = " Yellow World Scheduler Settings")
    public @interface SchedulerConfig {
 @AttributeDefinition(name = "Cron Expression")
        String expression() default "0 0/5 * * * ?";
    }
 @Activate
    @Modified
    protected void initialize(SchedulerConfig config) {
        LOGGER.info("Yellow World Scheduler has started!");
    }
```

```
@Override
    public void run() {
        LOGGER.info(" Yellow World is shining in the logs!");
    }
}
```

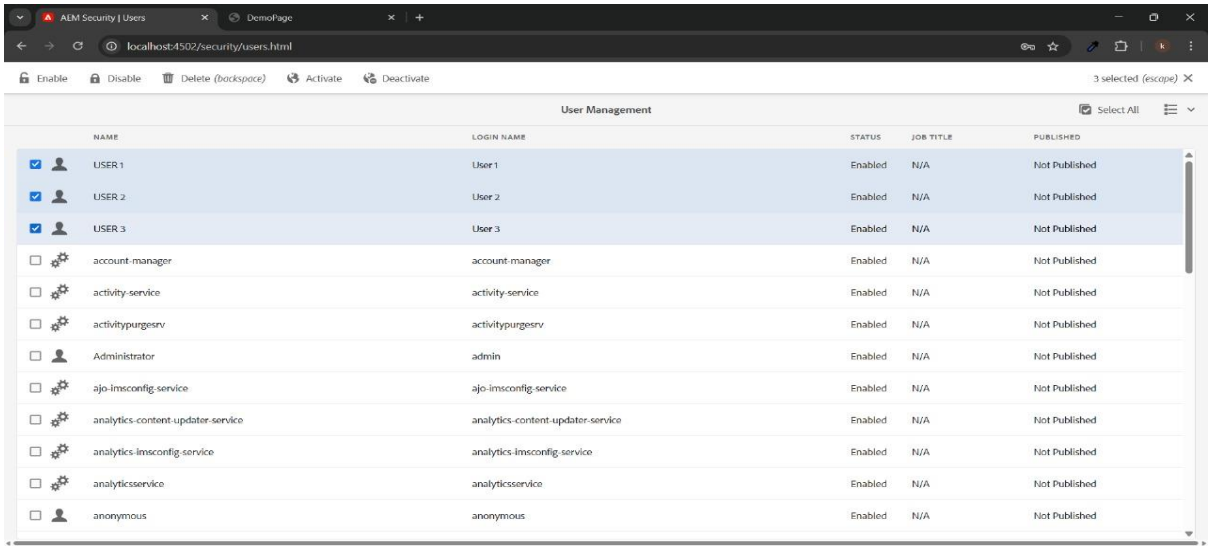
6. Create 3 users and add them to a group

Steps: Navigate to AEM → **Tools** → **Security** → **Groups**. Click on **"Create"**, and name the new group **"Dev Author"**.



Go to AEM → **Tools** → **Security** → **Users**. Create three users:

- **User 1:** USER 1
- **User 2:** USER 2
- **User 3:** USER 3



In CRXDE, navigate to the following paths:

- **Path:** /content
    - Add **"read"** permission for the **"Dev Author"** group.
- **Path:** /content/dam
    - Add **"read"** permission for the **"Dev Author"** group.

Deployment & Testing Steps:

1. **Deploy and Test the Custom Workflow:**
    - Apply the custom workflow to different pages and verify its functionality.
2. **Validate Workflow Process Logs:**
    - Check if the workflow process correctly logs the page titles as expected.
3. **Trigger the Event Handler:**
    - Manually trigger the event handler and confirm that the resource path appears in the logs.
4. **Run the Sling Job Manually:**
    - Execute the Sling Job and verify that it logs the message "Hello World" properly.
5. **Confirm Scheduler Logs:**
    - Ensure that the scheduler logs the message "Yellow World" every 5 minutes consistently.
6. **Verify User Permissions:**
    - Double-check that all users in the **"Dev Author"** group have the correct access rights.