**Table of Contents**

1. Define a Custom Workflow Model
2. Implement a Custom Workflow Process
3. Set Up an Event Handler
4. Build a Sling Job
5. Create a Scheduler
6. Configure Users & Groups with Permissions
7. Perform Testing and Debugging

**🔹 1. Define a Custom Workflow Model**

**Steps**:

* Open the AEM Workflow Console.
* Click "Create" to add a new workflow model.
* Drag and arrange workflow steps from the side panel.
* Configure transitions and conditions between steps.
* Save, activate, and verify that the model behaves as expected.

**🔹 2. Implement a Custom Workflow Process**

**Steps**:

* Create a Java class that implements the WorkflowProcess interface.
* Implement logic in the execute() method to define processing behavior.
* Register the class as an OSGi component with proper properties.
* Deploy the component and integrate it into the workflow model.

**Sample Code**:

java

CopyEdit

@Component(service = WorkflowProcess.class, property = {"process.label=Custom Workflow"})

public class CustomWorkflowProcess implements WorkflowProcess {

@Override

public void execute(WorkItem item, WorkflowSession session, MetaDataMap metaData) throws WorkflowException {

String payload = (String) item.getWorkflowData().getPayload();

System.out.println("Processing: " + payload);

}

}

**🔹 3. Set Up an Event Handler**

**Steps**:

* Create a Java class implementing the EventHandler interface.
* Use OSGi annotations to register for specific event topics.
* Define the event handling logic in handleEvent().
* Deploy and validate by triggering the configured events.

**Sample Code**:

java

CopyEdit

@Component(immediate = true, service = EventHandler.class, property = {

"event.topics=org/apache/sling/api/resource/Resource/ADDED"

})

public class CustomEventHandler implements EventHandler {

@Override

public void handleEvent(Event event) {

System.out.println("Event received: " + event.getTopic());

}

}

**🔹 4. Build a Sling Job**

**Steps**:

* Implement the JobConsumer interface.
* Register the component using appropriate job topic properties.
* Deploy and observe job execution in logs or the Job Manager.

**Sample Code**:

java

CopyEdit

@Component(service = JobConsumer.class, property = {

JobConsumer.PROPERTY\_TOPICS + "=custom/job"

})

public class CustomJobConsumer implements JobConsumer {

@Override

public JobResult process(Job job) {

System.out.println("Processing job: " + job.getTopic());

return JobResult.OK;

}

}

**🔹 5. Create a Scheduler**

**Steps**:

* Implement a Runnable OSGi component.
* Use configuration interfaces to define scheduling intervals.
* Deploy and monitor scheduled executions through logs or runtime.

**Sample Code**:

java

CopyEdit

@Designate(ocd = ScheduledTask.Config.class)

@Component(service = Runnable.class)

public class ScheduledTask implements Runnable {

@Override

public void run() {

System.out.println("Scheduled task executed");

}

}

**🔹 6. Configure Users & Groups with Permissions**

**Steps**:

* Go to AEM’s User Management interface.
* Create new user accounts and user groups.
* Assign roles and permissions to define access levels.
* Test permissions to ensure proper access control.

**🔹 7. Perform Testing and Debugging**

**Steps**:

* Manually initiate workflows for testing.
* Monitor logs for workflow, job, and event handler outputs.
* Use debugging tools and log tracing to fix any issues.
* Confirm each component behaves as expected in different scenarios.



