**Task-12 CODE**

**EVENT LISTENER FOR CREATING PAGE WORKFLOW STARTED**

**package** com.adobe.aem.guides.demo.core.listeners;

**import** java.util.HashMap;

**import** java.util.Map;

**import** javax.jcr.Session;

**import** org.apache.sling.api.resource.LoginException;

**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.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** com.day.cq.workflow.WorkflowException;

**import** com.day.cq.workflow.WorkflowService;

**import** com.day.cq.workflow.WorkflowSession;

**import** com.day.cq.workflow.exec.WorkflowData;

**import** com.day.cq.workflow.model.WorkflowModel;

@Component(service = EventHandler.**class**, immediate = **true**,

property = {

EventConstants.***EVENT\_TOPIC*** + "=org/apache/sling/api/resource/Resource/ADDED"

}

)

**public** **class** PageCreation **implements** EventHandler {

**private** **static** **final** Logger ***log*** = LoggerFactory.*getLogger*(PageCreation.**class**);

@Reference

**private** WorkflowService workflowService;

@Reference

**private** ResourceResolverFactory resolverFactory;

@Override

**public** **void** handleEvent(Event event) {

***log***.info("Page creation event triggered");

handleResourceAddedEvent(event);

}

**private** **void** handleResourceAddedEvent(Event event) {

String path = (String) event.getProperty("path");

// log.info("Resource added at path: {}", path);

ResourceResolver resourceResolver = **null**;

**try** {

Map<String, Object> param = **new** HashMap<>();

param.put(ResourceResolverFactory.***SUBSERVICE***, "hemanth");

resourceResolver = resolverFactory.getServiceResourceResolver(param);

Resource resource = resourceResolver.getResource(path);

**if** (resource != **null** && resource.isResourceType("cq:Page")) {

***log***.info("Page created at path: {}", path);

startWorkflow(path);

}

} **catch** (LoginException e) {

***log***.error("Error obtaining resource resolver", e);

} **finally** {

**if** (resourceResolver != **null**) {

resourceResolver.close();

}

}

}

**private** **void** startWorkflow(String payloadPath) {

Map<String, Object> param = **new** HashMap<>();

param.put(ResourceResolverFactory.***SUBSERVICE***, "hemanth");

**try** (ResourceResolver resolver = resolverFactory.getServiceResourceResolver(param)) {

Session session = resolver.adaptTo(Session.**class**);

WorkflowSession workflowSession = workflowService.getWorkflowSession(session);

String workflowModelPath = "/var/workflow/models/HemanthModel"; // Replace with your workflow model path

WorkflowModel workflowModel = workflowSession.getModel(workflowModelPath);

WorkflowData workflowData = workflowSession.newWorkflowData("JCR\_PATH", payloadPath);

workflowSession.startWorkflow(workflowModel, workflowData);

***log***.info("Workflow started for payload: {}", payloadPath);

} **catch** (LoginException e) {

***log***.error("LoginException while starting workflow", e);

} **catch** (WorkflowException e) {

***log***.error("WorkflowException while starting workflow", e);

} **catch** (Exception e) {

***log***.error("Exception while starting workflow", e);

}

}

}

**ADD PROPERTY THROUGH WORKFLOW**

**package workflow;**

**import java.util.Calendar;**

**import java.util.HashMap;**

**import java.util.Map;**

**import javax.jcr.Node;**

**import javax.jcr.Session;**

**import org.apache.sling.api.resource.LoginException;**

**import org.apache.sling.api.resource.PersistenceException;**

**import org.apache.sling.api.resource.ResourceResolver;**

**import org.apache.sling.api.resource.ResourceResolverFactory;**

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

**import org.osgi.service.component.annotations.Reference;**

**import org.slf4j.Logger;**

**import org.slf4j.LoggerFactory;**

**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;**

**@Component(service = WorkflowProcess.class, property = {"process.label=Page Created"})**

**public class PagecreatedWorkflow implements WorkflowProcess {**

**private static final Logger log = LoggerFactory.getLogger(PagecreatedWorkflow.class);**

**@Reference**

**private ResourceResolverFactory resolverFactory;**

**@Override**

**public void execute(WorkItem workItem, WorkflowSession workflowSession, MetaDataMap metaDataMap) {**

**String payloadPath = workItem.getWorkflowData().getPayload().toString();**

**ResourceResolver resourceResolver = null;**

**try {**

**// Using service user to obtain resource resolver**

**Map<String, Object> param = new HashMap<>();**

**param.put(ResourceResolverFactory.SUBSERVICE, "hemanth");**

**resourceResolver = resolverFactory.getServiceResourceResolver(param);**

**Session session = resourceResolver.adaptTo(Session.class);**

**Node pageNode = session.getNode(payloadPath);**

**if (pageNode != null && pageNode.hasNode("jcr:content")) {**

**Node contentNode = pageNode.getNode("jcr:content");**

**Calendar currentDate = Calendar.getInstance();**

**contentNode.setProperty("expired", currentDate);**

**session.save();**

**log.info("Property 'expired' added to jcr:content of page at {}", payloadPath);**

**} else {**

**log.warn("jcr:content node not found for page at {}", payloadPath);**

**}**

**} catch (Exception e) {**

**log.error("Error adding property to jcr:content of page at {}", payloadPath, e);**

**throw new RuntimeException(e);**

**} finally {**

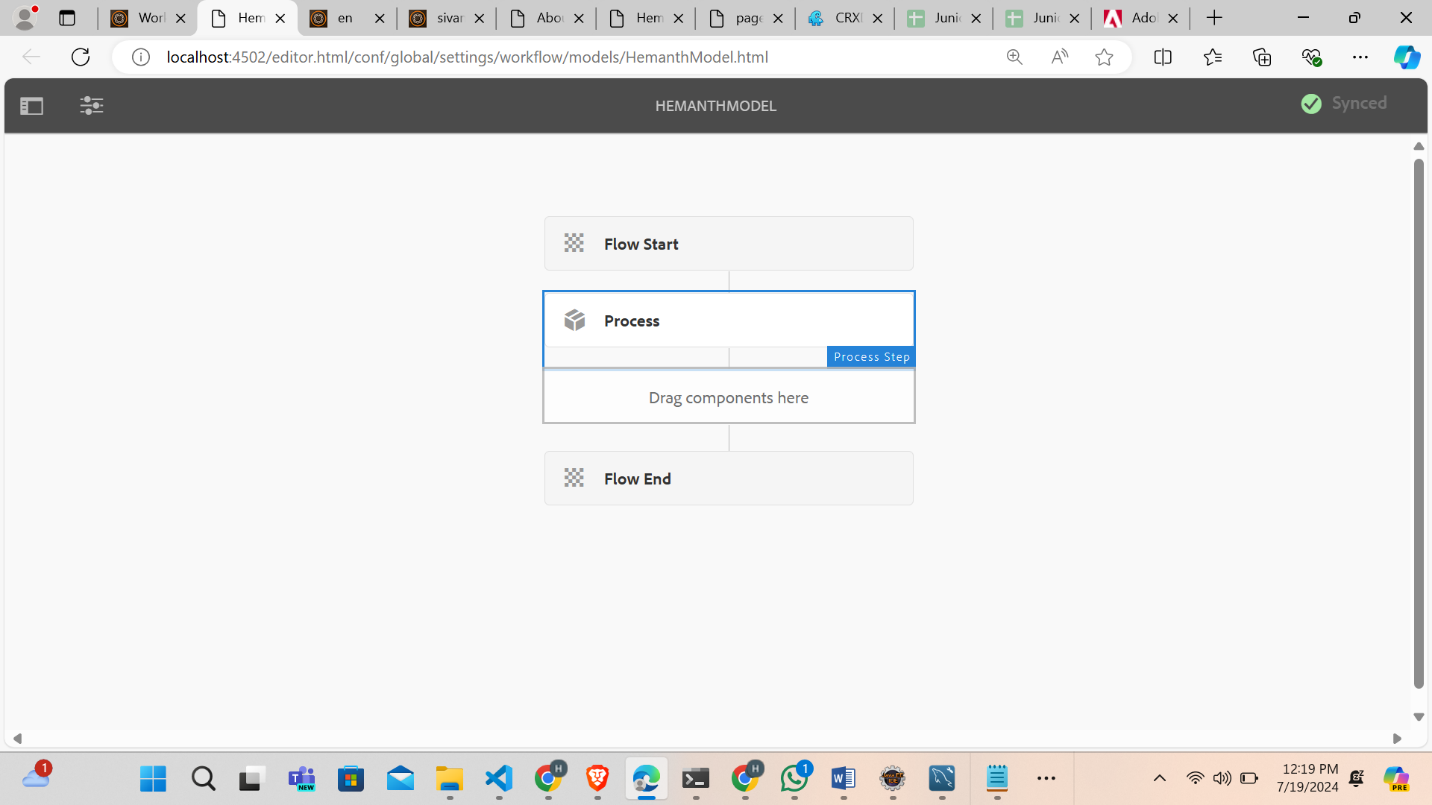
**if (resourceResolver != null) {**

**resourceResolver.close();**

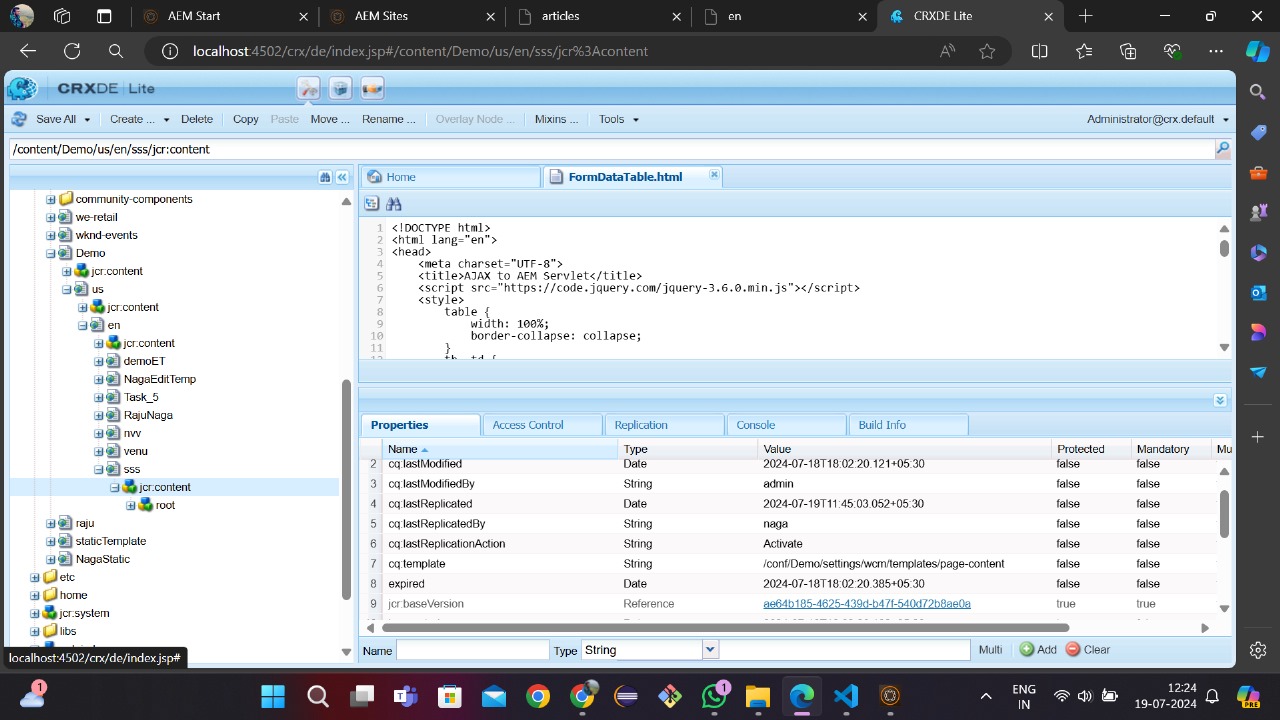
**}**

**}}}**

**WORKFLOW PROCESS STEP**

****

**ADD PROPERTY ON CRXDE**

****