**Task-13 CODE**

**Question:** 1)When you create any page in project you should have trigger the **Eventlistener** after that you have to trigger the workflow and it should add the **Expirydate property** to that page **jcr:content** and the value should be current date and time.

2) **Continuation:** And when you create any page in other project.it should add the expirydate property to that page jcr:content and the value should be **Previous date and time.**

**Answer:**

**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** javax.jcr.observation.Event;

**import** javax.jcr.observation.EventIterator;

**import** javax.jcr.observation.EventListener;

**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.apache.sling.jcr.api.SlingRepository;

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

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

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

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

**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(immediate = **true**)

**public** **class** PageCreatedTask **implements** EventListener {

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

@Reference

**private** WorkflowService workflowService;

@Reference

**private** ResourceResolverFactory resolverFactory;

@Reference

**private** SlingRepository repository;

**private** Session observationSession;

@Activate

**protected** **void** activate() {

**try** {

observationSession = repository.loginService("hemanth", **null**);

observationSession.getWorkspace().getObservationManager().addEventListener(

**this**,

Event.***NODE\_ADDED***,

"/content", // or any specific path you want to observe

**true**,

**null**,

**null**,

**false**

);

***log***.info("PageCreationEventListener activated");

} **catch** (Exception e) {

***log***.error("Error activating PageCreationEventListener", e);

}

}

@Deactivate

**protected** **void** deactivate() {

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

**try** {

observationSession.getWorkspace().getObservationManager().removeEventListener(**this**);

observationSession.logout();

} **catch** (Exception e) {

***log***.error("Error deactivating PageCreationEventListener", e);

}

}

}

@Override

**public** **void** onEvent(EventIterator events) {

**while** (events.hasNext()) {

Event event = events.nextEvent();

**try** {

String path = event.getPath();

handleResourceAddedEvent(path);

} **catch** (Exception e) {

***log***.error("Error handling event", e);

}

}

}

**private** **void** handleResourceAddedEvent(String 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";

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.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");

String templatePath = contentNode.getProperty("cq:template").getString();

Calendar currentDate = Calendar.*getInstance*();

**if** (!"/conf/Demo/settings/wcm/templates/page-content".equals(templatePath)) {

currentDate.add(Calendar.***DATE***, -1);

}

contentNode.setProperty("expirydate", 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();

}

}

}

}











