Task-9 CODE

QUESTION

When you publish the page? And it will trigger the event handler and display the logs, after which you should trigger the workflow. then start the workflow automatically.

Create a workflow that includes both a process and a participant step.

Description:

When a user publishes a page, the administrator should receive an email notification.

ANSWER:

HANDLER CODE

```
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.ModifiableValueMap;
import org.apache.sling.api.resource.PersistenceException;
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.replication.ReplicationAction;
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 + "=" + ReplicationAction.EVENT_TOPIC
    }
)
public class HemanthHandlerJuly implements EventHandler {
    private static final Logger log = LoggerFactory.getLogger(HemanthHandlerJuly.class);

    @Reference
    private WorkflowService workflowService;

    @Reference
    private ResourceResolverFactory resolverFactory;

    @Override
    public void handleEvent(Event event) {
        log.info("Handle my trigger");
        handleReplicationEvent(event);
    }

    private void handleReplicationEvent(Event event) {
        ReplicationAction action = ReplicationAction.fromEvent(event);
        String path = action.getPath();
        log.info("Content activated at path: {}", path);
        startWorkflow(path);
//        addPropertyToPage(path);
    }

    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/sivamodel"; // 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);
        }
    }

    private void addPropertyToPage(String pagePath) {
        Map<String, Object> param = new HashMap<>();
        param.put(ResourceResolverFactory.SUBSERVICE, "hemanth");
        try (ResourceResolver resolver = resolverFactory.getServiceResourceResolver(param)) {
            String jcrContentPath = pagePath + "/jcr:content";
            Resource resource = resolver.getResource(jcrContentPath);
            if (resource != null) {
                ModifiableValueMap properties = resource.adaptTo(ModifiableValueMap.class);
                if (properties != null) {
                    properties.put("changed", true);
                    resolver.commit();
                    log.info("Property 'changed' set to true for: {}", jcrContentPath);
                }
            } else {
                log.warn("Resource not found at path: {}", jcrContentPath);
            }
        } catch (LoginException e) {
            log.error("LoginException while adding property", e);
        } catch (PersistenceException e) {
            log.error("PersistenceException while adding property", e);
        } catch (Exception e) {
            log.error("Exception while adding property", e);
        }
    }
}
```

WORKFLOW PROCESS CODE

```
package workflow;
```

```
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.aem.guides.demo.core.service.EmailService;
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=Send Email
Process"})
public class Workflow implements WorkflowProcess {
        private static final Logger log = LoggerFactory.getLogger(Workflow.class);
         @Reference
          private EmailService emailService;

          @Override
          public void execute(WorkItem workItem, WorkflowSession workflowSession,
MetaDataMap args) {
              try {
                  log.info("this is process step by hemanth");
                  String pagePath = workItem.getWorkflowData().getPayload().toString();
                  emailService.sendEmail("hemanth.nellori@gmail", "Page Published", "A page
has been published: " + pagePath);
              } catch (Exception e) {
                  e.printStackTrace();
              }
        }
}
```

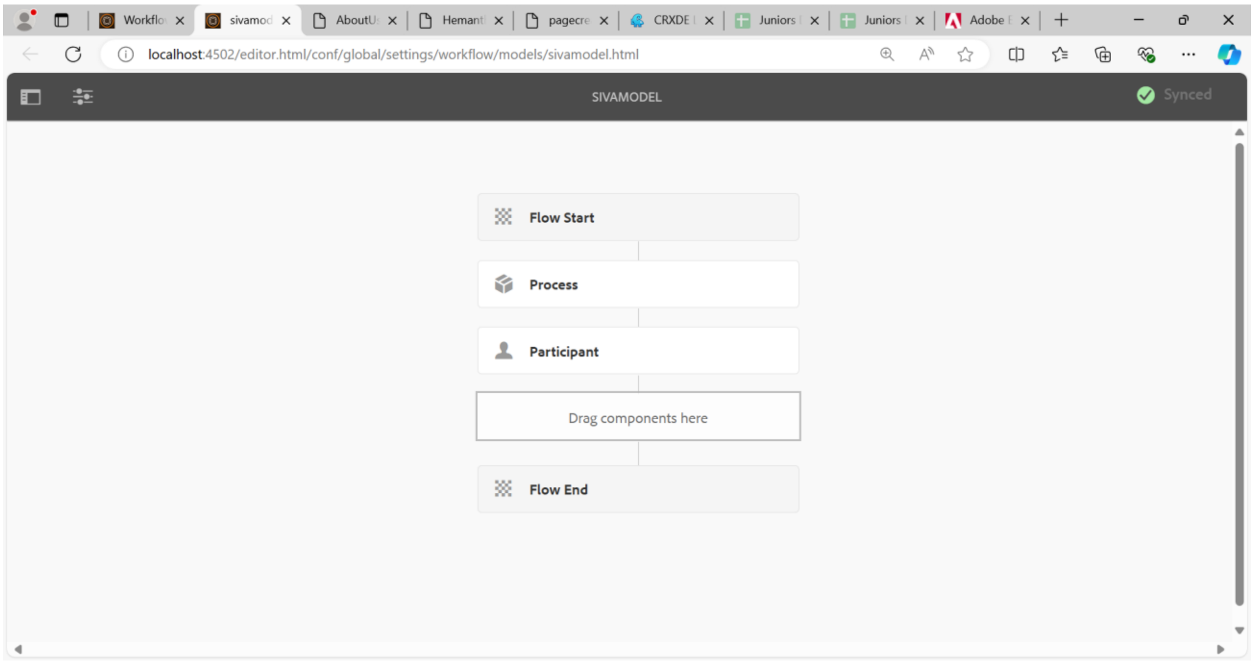
SERVICE CODE

```
package com.adobe.aem.guides.demo.core.service;

import org.osgi.service.component.annotations.Component;
import javax.mail.*;
import javax.mail.internet.*;
import java.util.Properties;

@Component(immediate = true, service = EmailService.class)
public class EmailService {

  public void sendEmail(String to, String subject, String body) throws MessagingException {
    String from = "hnellori@gmail.com";
    final String username = "hemanth";
    final String password = "hemanth$123456";
    String host = "smtp.example.com";

    Properties props = new Properties();
    props.put("mail.smtp.auth", "true");
    props.put("mail.smtp.starttls.enable", "true");
    props.put("mail.smtp.host", host);
    props.put("mail.smtp.port", "465");

    Session session = Session.getInstance(props, new javax.mail.Authenticator() {
      protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(username, password);
      }
    });

    Message message = new MimeMessage(session);
    message.setFrom(new InternetAddress(from));
    message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(to));
    message.setSubject(subject);
    message.setText(body);

    Transport.send(message);
  }
}
```

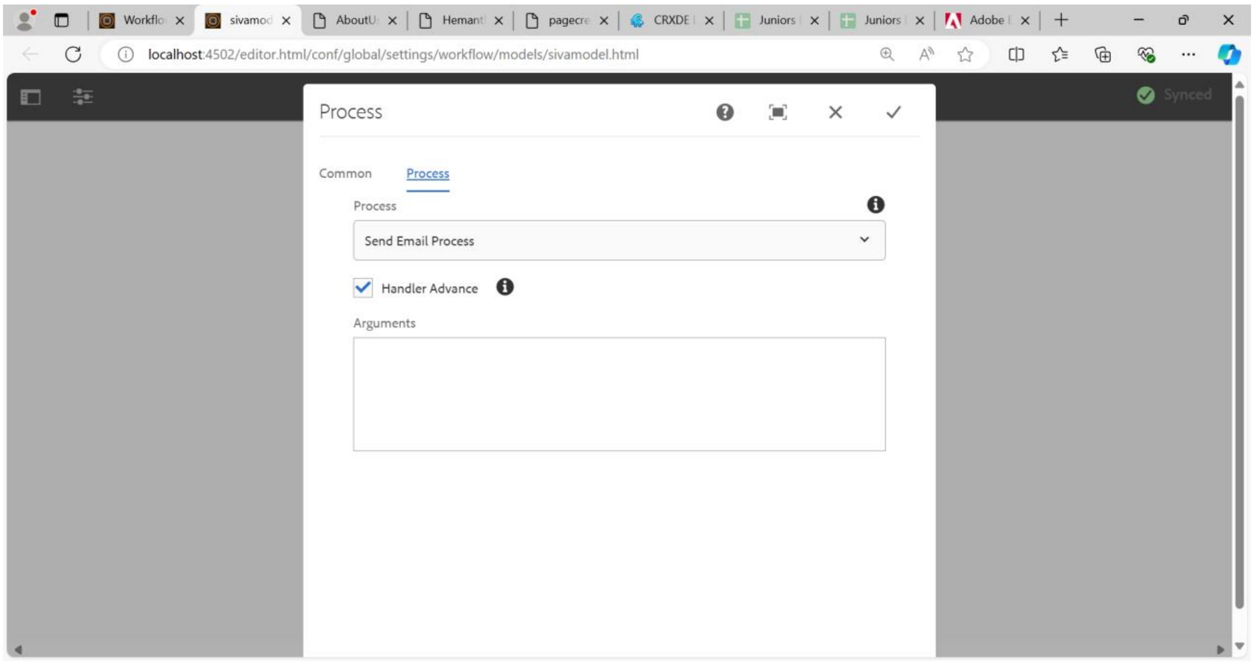
EMAIL TEST SERVLET CODE

```
package com.adobe.aem.guides.demo.core.servlets;

import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class Email {
    public static void main(String[] args) {
        // SMTP server details
        String smtpHost = "smtp.gmail.com"; // Replace with your SMTP server
        int smtpPort = 587; // Replace with the port of your SMTP server
        final String username = "hemanthsainellori@gmail.com"; // Replace with your email
address
        final String password = "yofmzgflukuqojtr"; // Replace with your email password

        // Sender's email address
        String fromEmail = "hemanthsainellori@gmail.com"; // Replace with your email address

        // Recipient's email address
        String toEmail = "hnellori@gmail.com"; // Replace with recipient's email address

        // Set properties
        Properties props = new Properties();
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", smtpHost);
        props.put("mail.smtp.port", smtpPort);

        // Create session
        Session session = Session.getInstance(props, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        });

        try {
            // Create a default MimeMessage object
            Message message = new MimeMessage(session);
            // Set From: header field of the header
            message.setFrom(new InternetAddress(fromEmail));

            // Set To: header field of the header
            message.setRecipients(Message.RecipientType.TO, InternetAddress.parse(toEmail));

            // Set Subject: header field
            message.setSubject("Testing JavaMail");

            // Now set the actual message
            message.setText("Hello, this is a test message from JavaMail.");

            // Send message
            Transport.send(message);

            System.out.println("Email sent successfully!");

        } catch (MessagingException e) {
            throw new RuntimeException(e);
        }
    }
}
```

WORKFLOWMODEL


localhost:4502/editor.html/conf/global/settings/workflow/models/sivamodel.html
SIVAMODEL
Synced
Flow Start
Process
Participant
Drag components here
Flow End

WORKFLOWMODEL PROCESS STEP


localhost:4502/editor.html/conf/global/settings/workflow/models/sivamodel.html
Synced
Common
Process

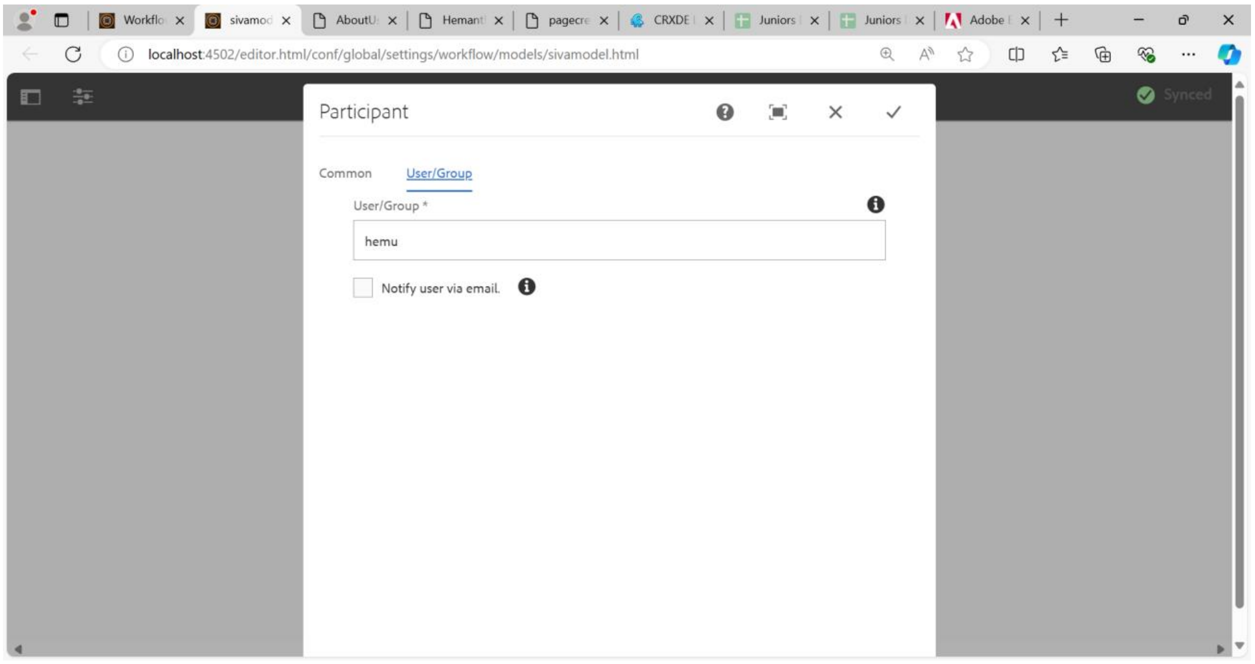
Process

Process: Send Email Process

✓ Handler Advance

Arguments

WORKFLOW PARTICIPANT STEP

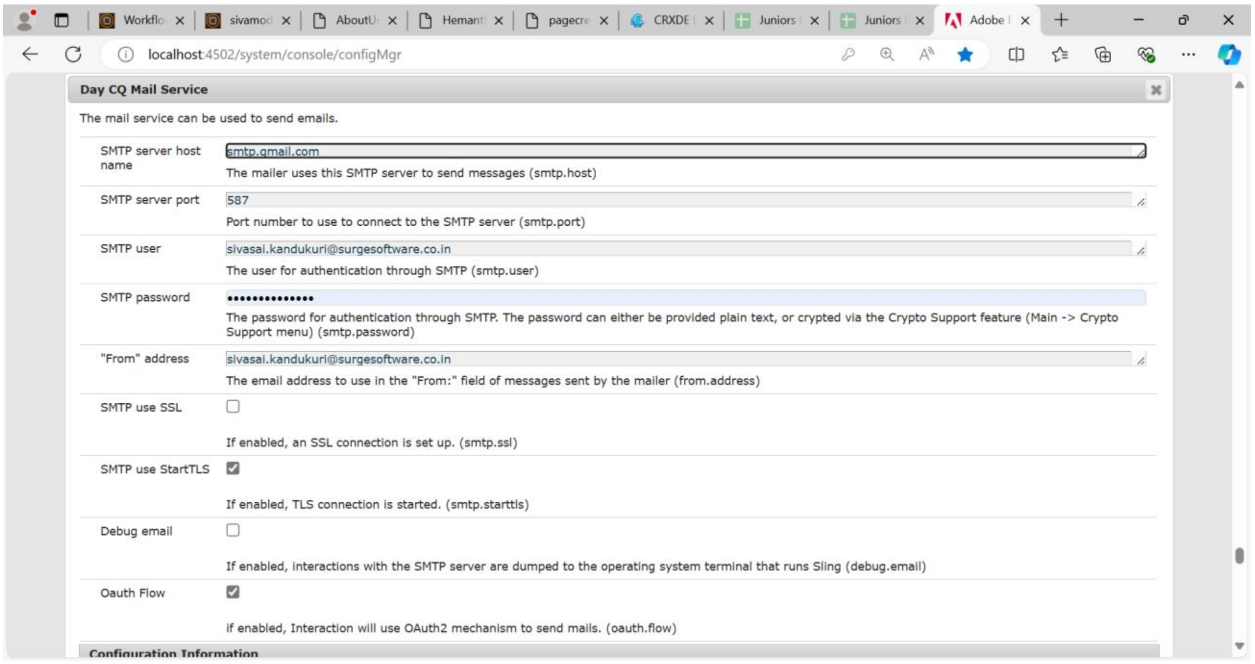

localhost:4502/editor.html/conf/global/settings/workflow/models/sivamodel.html
Participant
Common
User/Group
Synced

User/Group *

hemu

Notify user via email.

DQ CQ MAIL SERVICE



Day CQ Mail Service

The mail service can be used to send emails.

SMTP server host name	smtp.gmail.com The mailer uses this SMTP server to send messages (smtp.host)
SMTP server port	587 Port number to use to connect to the SMTP server (smtp.port)
SMTP user	sivasai.kandukuri@surgesoftware.co.in The user for authentication through SMTP (smtp.user)
SMTP password	•••••••••••••• The password for authentication through SMTP. The password can either be provided plain text, or crypted via the Crypto Support feature (Main -> Crypto Support menu) (smtp.password)
"From" address	sivasai.kandukuri@surgesoftware.co.in The email address to use in the "From:" field of messages sent by the mailer (from.address)
SMTP use SSL	☐ If enabled, an SSL connection is set up. (smtp.ssl)
SMTP use StartTLS	☑ If enabled, TLS connection is started. (smtp.starttls)
Debug email	☐ If enabled, interactions with the SMTP server are dumped to the operating system terminal that runs Sling (debug.email)
Oauth Flow	☑ if enabled, Interaction will use OAuth2 mechanism to send mails. (oauth.flow)

Configuration Information