**Task-11 CODE**

**FRONT-END CODE FOR CRUD OPERATION**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<title>AJAX to AEM Servlet</title>**

**<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>**

**<style>**

**table {**

**width: 100%;**

**border-collapse: collapse;**

**}**

**th, td {**

**border: 1px solid orange;**

**padding: 8px;**

**text-align: left;**

**}**

**th {**

**background-color: black;**

**color: white;**

**}**

**td {**

**background-color: black;**

**color: white;**

**}**

**.edit, .delete {**

**color: orange;**

**cursor: pointer;**

**}**

**</style>**

**</head>**

**<body>**

**<form id="myForm">**

**<input type="hidden" id="rowIndex" value="">**

**<input type="text" name="param1" id="param1" placeholder="Name"><br></br>**

**<input type="text" name="param2" id="param2" placeholder="Email"><br></br>**

**<input type="text" name="param3" id="param3" placeholder="Subject"><br></br>**

**<input type="text" name="param4" id="param4" placeholder="Message"><br></br>**

**<div class="your-component" data-path="${resource.path @ context='unsafe'}"></div>**

**<button type="button" onclick="sendData()">Submit</button>**

**</form>**

**<div id="formDataTable"></div>**

**<script>**

**function sendData() {**

**const rowIndex = $("#rowIndex").val();**

**const param1 = $("#param1").val();**

**const param2 = $("#param2").val();**

**const param3 = $("#param3").val();**

**const param4 = $("#param4").val();**

**const componentPath = $(".your-component").attr("data-path");**

**console.log("Component Path:", componentPath);**

**$.ajax({**

**url: '/bin/myServlet',**

**type: 'POST',**

**data: {**

**rowIndex: rowIndex,**

**param1: param1,**

**param2: param2,**

**param3: param3,**

**param4: param4,**

**componentPath: componentPath**

**},**

**success: function(response) {**

**console.log('Response from POST:', response);**

**alert('Response: ' + response.message);**

**fetchTableData();**

**clearForm();**

**},**

**error: function(xhr, status, error) {**

**console.error('Error: ' + error);**

**}**

**});**

**}**

**function fetchTableData() {**

**const componentPath = $(".your-component").attr("data-path");**

**console.log("Fetching data from path:", componentPath);**

**$.ajax({**

**url: '/bin/myServlet',**

**type: 'GET',**

**data: { path: componentPath },**

**success: function(response) {**

**console.log('Response from GET:', response);**

**if (response.data) {**

**createTable(response.data);**

**} else {**

**console.error('No data received');**

**}**

**},**

**error: function(xhr, status, error) {**

**console.error('Error: ' + error);**

**}**

**});**

**}**

**function createTable(data) {**

**const tableContainer = document.getElementById('formDataTable');**

**tableContainer.innerHTML = '';**

**const table = document.createElement('table');**

**const headerRow = document.createElement('tr');**

**const headers = ['Name', 'Email', 'Subject', 'Message', 'Edit', 'Delete'];**

**headers.forEach(headerText => {**

**const th = document.createElement('th');**

**th.textContent = headerText;**

**headerRow.appendChild(th);**

**});**

**table.appendChild(headerRow);**

**data.forEach((item, index) => {**

**const dataRow = document.createElement('tr');**

**const nameCell = document.createElement('td');**

**nameCell.textContent = item.name;**

**dataRow.appendChild(nameCell);**

**const emailCell = document.createElement('td');**

**emailCell.textContent = item.email;**

**dataRow.appendChild(emailCell);**

**const subjectCell = document.createElement('td');**

**subjectCell.textContent = item.subject;**

**dataRow.appendChild(subjectCell);**

**const messageCell = document.createElement('td');**

**messageCell.textContent = item.message;**

**dataRow.appendChild(messageCell);**

**const editCell = document.createElement('td');**

**editCell.textContent = 'Edit';**

**editCell.className = 'edit';**

**editCell.dataset.index = index;**

**editCell.onclick = function() {**

**editRow(item, index);**

**};**

**dataRow.appendChild(editCell);**

**const deleteCell = document.createElement('td');**

**deleteCell.textContent = 'Delete';**

**deleteCell.className = 'delete';**

**deleteCell.dataset.index = index;**

**deleteCell.onclick = function() {**

**deleteRow(index);**

**};**

**dataRow.appendChild(deleteCell);**

**table.appendChild(dataRow);**

**});**

**tableContainer.appendChild(table);**

**}**

**function editRow(item, index) {**

**$("#rowIndex").val(index);**

**$("#param1").val(item.name);**

**$("#param2").val(item.email);**

**$("#param3").val(item.subject);**

**$("#param4").val(item.message);**

**}**

**function deleteRow(index) {**

**const componentPath = $(".your-component").attr("data-path");**

**$.ajax({**

**url: '/bin/myServlet',**

**type: 'POST',**

**data: {**

**rowIndex: index,**

**delete: 'true',**

**componentPath: componentPath**

**},**

**success: function(response) {**

**console.log('Response from DELETE:', response);**

**alert('Row deleted successfully');**

**fetchTableData();**

**},**

**error: function(xhr, status, error) {**

**console.error('Error: ' + error);**

**}**

**});**

**}**

**function clearForm() {**

**$("#rowIndex").val('');**

**$("#param1").val('');**

**$("#param2").val('');**

**$("#param3").val('');**

**$("#param4").val('');**

**}**

**document.addEventListener('DOMContentLoaded', function() {**

**fetchTableData();**

**});**

**</script>**

**</body>**

**</html>**

**SERVLET CODE**

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

**import org.apache.sling.api.SlingHttpServletRequest;**

**import org.apache.sling.api.SlingHttpServletResponse;**

**import org.apache.sling.api.servlets.SlingAllMethodsServlet;**

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

**import org.osgi.framework.Constants;**

**import com.google.gson.JsonArray;**

**import com.google.gson.JsonObject;**

**import javax.jcr.Node;**

**import javax.jcr.RepositoryException;**

**import javax.jcr.Session;**

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

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

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

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

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

**import javax.servlet.Servlet;**

**import java.io.IOException;**

**import java.util.HashMap;**

**import java.util.Iterator;**

**import java.util.Map;**

**@Component(**

**service = { Servlet.class },**

**property = {**

**Constants.SERVICE\_DESCRIPTION + "=Example Servlet",**

**"sling.servlet.paths=/bin/myServlet",**

**"sling.servlet.methods={GET, POST, DELETE}"**

**}**

**)**

**public class MyServlet extends SlingAllMethodsServlet {**

**@org.osgi.service.component.annotations.Reference**

**private SlingRepository repository;**

**@org.osgi.service.component.annotations.Reference**

**private ResourceResolverFactory resolverFactory;**

**@Override**

**protected void doGet(SlingHttpServletRequest request, SlingHttpServletResponse response) throws IOException {**

**String componentPath = request.getParameter("path");**

**ResourceResolver resourceResolver = null;**

**try {**

**// Get the resource resolver**

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

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

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

**// Get the resource at the specified path**

**Resource resource = resourceResolver.getResource(componentPath + "/data");**

**JsonArray dataArray = new JsonArray();**

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

**Iterator<Resource> children = resource.listChildren();**

**while (children.hasNext()) {**

**Resource child = children.next();**

**Node childNode = child.adaptTo(Node.class);**

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

**JsonObject dataObject = new JsonObject();**

**dataObject.addProperty("name", childNode.getProperty("name").getString());**

**dataObject.addProperty("email", childNode.getProperty("email").getString());**

**dataObject.addProperty("subject", childNode.getProperty("subject").getString());**

**dataObject.addProperty("message", childNode.getProperty("message").getString());**

**dataArray.add(dataObject);**

**}**

**}**

**}**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.add("data", dataArray);**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**} catch (LoginException | RepositoryException e) {**

**e.printStackTrace();**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.addProperty("status", "error");**

**jsonResponse.addProperty("message", e.getMessage());**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**} finally {**

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

**resourceResolver.close();**

**}**

**}**

**}**

**@Override**

**protected void doPost(SlingHttpServletRequest request, SlingHttpServletResponse response) throws IOException {**

**String rowIndex = request.getParameter("rowIndex");**

**String delete = request.getParameter("delete");**

**String componentPath = request.getParameter("componentPath");**

**ResourceResolver resourceResolver = null;**

**Session session = null;**

**try {**

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

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

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

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

**Resource componentResource = resourceResolver.getResource(componentPath);**

**Node componentNode = componentResource.adaptTo(Node.class);**

**if (!componentNode.hasNode("data")) {**

**componentNode.addNode("data", "nt:unstructured");**

**}**

**Node dataNode = componentNode.getNode("data");**

**if (delete != null && delete.equals("true")) {**

**// Handle delete**

**if (dataNode.hasNode("item" + rowIndex)) {**

**Node itemNode = dataNode.getNode("item" + rowIndex);**

**itemNode.remove();**

**session.save();**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.addProperty("status", "success");**

**jsonResponse.addProperty("message", "Data deleted successfully.");**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**} else {**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.addProperty("status", "error");**

**jsonResponse.addProperty("message", "Node not found.");**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**}**

**} else {**

**// Handle create/update**

**String param1 = request.getParameter("param1");**

**String param2 = request.getParameter("param2");**

**String param3 = request.getParameter("param3");**

**String param4 = request.getParameter("param4");**

**Node itemNode;**

**if (rowIndex == null || rowIndex.isEmpty()) {**

**int itemIndex = 0;**

**while (dataNode.hasNode("item" + itemIndex)) {**

**itemIndex++;**

**}**

**itemNode = dataNode.addNode("item" + itemIndex, "nt:unstructured");**

**} else {**

**itemNode = dataNode.getNode("item" + rowIndex);**

**}**

**itemNode.setProperty("name", param1);**

**itemNode.setProperty("email", param2);**

**itemNode.setProperty("subject", param3);**

**itemNode.setProperty("message", param4);**

**session.save();**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.addProperty("status", "success");**

**jsonResponse.addProperty("message", "Data stored successfully.");**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**}**

**} catch (LoginException | RepositoryException e) {**

**e.printStackTrace();**

**JsonObject jsonResponse = new JsonObject();**

**jsonResponse.addProperty("status", "error");**

**jsonResponse.addProperty("message", e.getMessage());**

**response.setContentType("application/json");**

**response.setCharacterEncoding("UTF-8");**

**response.getWriter().write(jsonResponse.toString());**

**} finally {**

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

**session.logout();**

**}**

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

**resourceResolver.close();**

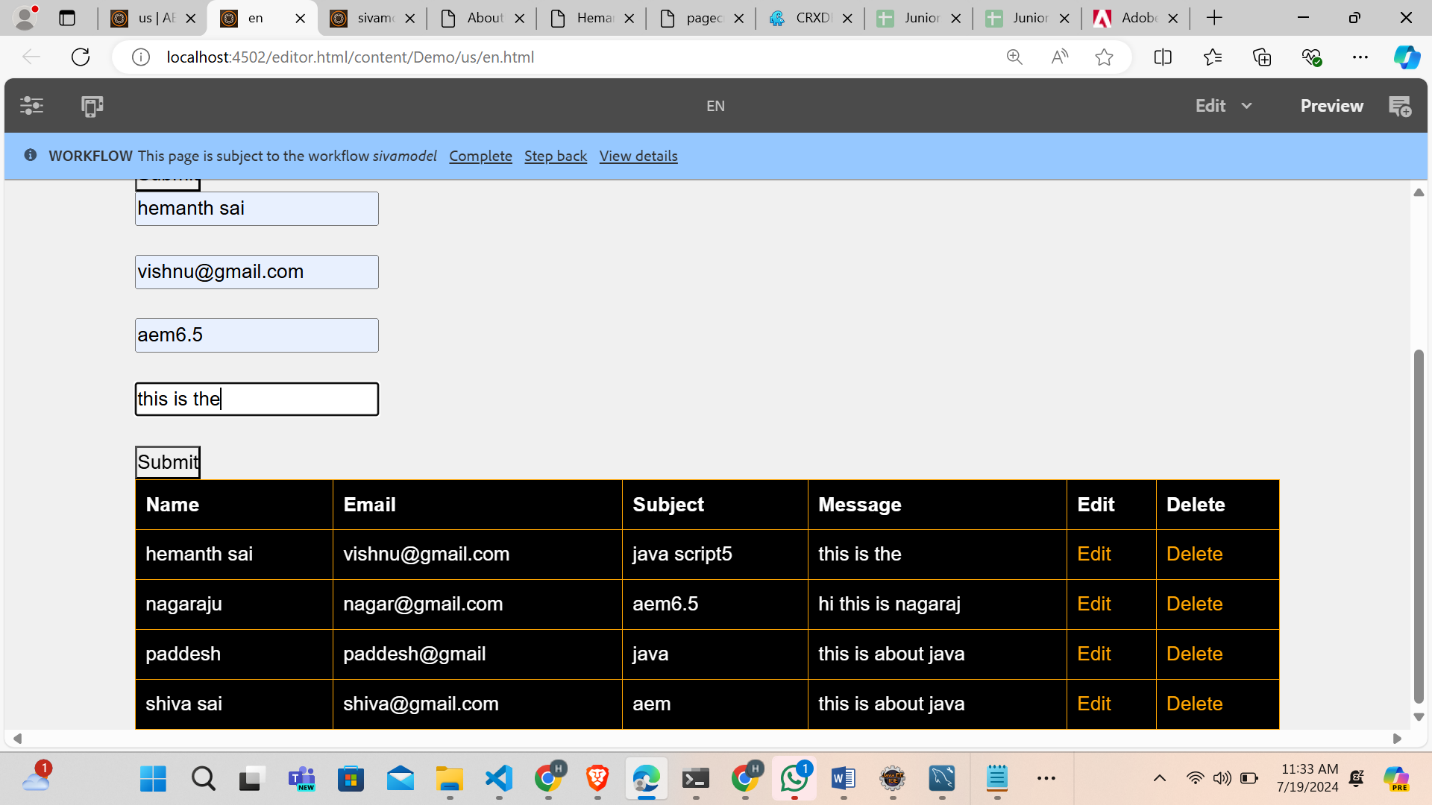
**}**

**}**

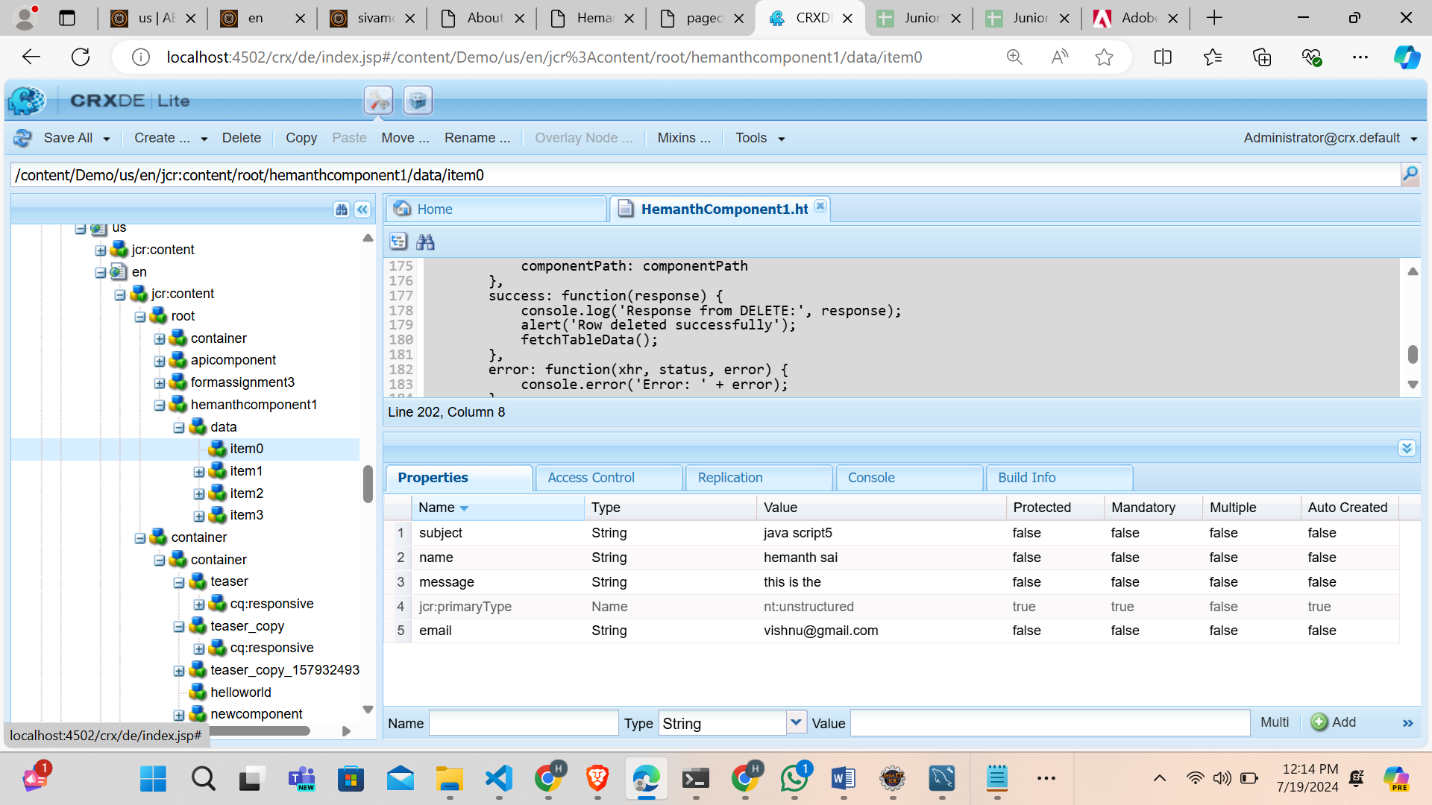
**}**

**}**

**FORM AND TABLE CREATION**

****

**NODE CREATION AND STORED FORM DATA**

****