package com.example.demo.service;

import io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient;

import io.confluent.kafka.schemaregistry.client.SchemaRegistryClient;

import io.confluent.kafka.schemaregistry.client.rest.exceptions.RestClientException;

import org.apache.avro.Schema;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import java.io.IOException;

@Service

public class SchemaService {

private final SchemaRegistryClient schemaRegistryClient;

public SchemaService(@Value("${schema.registry.url}") String schemaRegistryUrl) {

this.schemaRegistryClient = new CachedSchemaRegistryClient(schemaRegistryUrl, 100);

}

public String getSchema(String subject) {

try {

Schema schema = schemaRegistryClient.getLatestSchemaMetadata(subject).getSchema();

return schema.toString(true);

} catch (IOException | RestClientException e) {

e.printStackTrace();

return null;

}

}

}

package com.example.demo.controller;

import com.example.demo.service.SchemaService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class SchemaController {

private final SchemaService schemaService;

@Autowired

public SchemaController(SchemaService schemaService) {

this.schemaService = schemaService;

}

@GetMapping("/getSchema")

public String getSchema(@RequestParam String subject) {

return schemaService.getSchema(subject);

}

}

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fetch Schema</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

#schema {

white-space: pre-wrap; /\* Preserve whitespace and line breaks \*/

background-color: #f4f4f4;

padding: 10px;

border: 1px solid #ccc;

margin-top: 20px;

overflow-x: auto; /\* Handle long lines \*/

}

input {

margin-bottom: 10px;

display: block;

}

</style>

</head>

<body>

<h1>Fetch Schema from Schema Registry</h1>

<form id="schemaForm">

<label for="event">Event:</label>

<input type="text" id="event" name="event" required>

<label for="topic">Topic:</label>

<input type="text" id="topic" name="topic" required>

<button type="submit">Fetch Schema</button>

</form>

<div id="schema"></div>

<script>

document.getElementById('schemaForm').addEventListener('submit', async function(event) {

event.preventDefault();

const eventInput = document.getElementById('event').value;

const topicInput = document.getElementById('topic').value;

const subject = `${eventInput}-${topicInput}`;

const url = `/getSchema?subject=${encodeURIComponent(subject)}`;

try {

const response = await fetch(url);

if (response.ok) {

const schema = await response.json();

document.getElementById('schema').textContent = JSON.stringify(schema, null, 2);

} else {

document.getElementById('schema').textContent = 'Error fetching schema';

}

} catch (error) {

document.getElementById('schema').textContent = 'Error: ' + error.message;

}

});

</script>

</body>

</html>  
  
##################################################################################  
  
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>  
  
  
<!DOCTYPE html>

<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Fetch Schema</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

#schema {

white-space: pre-wrap;

background-color: #f4f4f4;

padding: 10px;

border: 1px solid #ccc;

margin-top: 20px;

overflow-x: auto;

}

input {

margin-bottom: 10px;

display: block;

}

</style>

</head>

<body>

<h1>Fetch Schema from Schema Registry</h1>

<form id="schemaForm" th:action="@{/getSchema}" th:object="${schemaRequest}" method="get">

<label for="event">Event:</label>

<input type="text" id="event" name="event" th:field="\*{event}" required>

<label for="topic">Topic:</label>

<input type="text" id="topic" name="topic" th:field="\*{topic}" required>

<button type="submit">Fetch Schema</button>

</form>

<div id="schema">

<pre th:text="${schemaJson}"></pre>

</div>

</body>

</html>  
  
  
package com.example.demo.model;

public class SchemaRequest {

private String event;

private String topic;

// getters and setters

public String getEvent() {

return event;

}

public void setEvent(String event) {

this.event = event;

}

public String getTopic() {

return topic;

}

public void setTopic(String topic) {

this.topic = topic;

}

}  
  
package com.example.demo.controller;

import com.example.demo.model.SchemaRequest;

import com.example.demo.service.SchemaService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

@Controller

public class SchemaController {

private final SchemaService schemaService;

@Autowired

public SchemaController(SchemaService schemaService) {

this.schemaService = schemaService;

}

@GetMapping("/")

public String showForm(Model model) {

model.addAttribute("schemaRequest", new SchemaRequest());

return "index"; // This corresponds to index.html in src/main/resources/templates

}

@GetMapping("/getSchema")

public String getSchema(@ModelAttribute SchemaRequest schemaRequest, Model model) {

String subject = schemaRequest.getEvent() + "-" + schemaRequest.getTopic();

String schemaJson = schemaService.getSchema(subject);

model.addAttribute("schemaJson", schemaJson);

return "index"; // Return to the same page with updated data

}

}