<!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;

}

.dropdown-suggestions {

border: 1px solid #ccc;

border-top: none;

max-height: 150px;

overflow-y: auto;

background-color: white;

position: absolute;

z-index: 1000;

width: calc(100% - 22px); /\* Adjust according to padding \*/

}

.dropdown-suggestions div {

padding: 8px;

cursor: pointer;

}

.dropdown-suggestions div:hover {

background-color: #e9e9e9;

}

</style>

</head>

<body>

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

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

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

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

<div id="eventSuggestions" class="dropdown-suggestions"></div>

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

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

<div id="topicSuggestions" class="dropdown-suggestions"></div>

<label for="version">Version:</label>

<input type="text" id="version" name="version" autocomplete="off" required>

<div id="versionSuggestions" class="dropdown-suggestions"></div>

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

</form>

<div id="schema">

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

</div>

<!-- JavaScript to fetch subjects and handle auto-suggest -->

<script th:inline="javascript">

document.addEventListener("DOMContentLoaded", async function() {

// Fetch all subjects

const response = await fetch('/getAllSubjects');

if (response.ok) {

const subjects = await response.json();

initializeAutoSuggest(subjects);

}

});

function initializeAutoSuggest(subjects) {

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

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

const versionInput = document.getElementById('version');

const eventSuggestions = document.getElementById('eventSuggestions');

const topicSuggestions = document.getElementById('topicSuggestions');

const versionSuggestions = document.getElementById('versionSuggestions');

function showSuggestions(input, suggestions, dropdown) {

dropdown.innerHTML = '';

if (suggestions.length === 0) {

return;

}

suggestions.forEach(function(suggestion) {

var div = document.createElement('div');

div.textContent = suggestion;

div.onclick = function() {

input.value = suggestion;

dropdown.innerHTML = '';

};

dropdown.appendChild(div);

});

}

function filterSuggestions(value, type) {

const query = value.toLowerCase();

return subjects.filter(subject => {

const parts = subject.split('-');

const topic = parts[0].toLowerCase();

const event = parts[1].toLowerCase();

const version = parts[2] ? parts[2].toLowerCase() : '';

switch(type) {

case 'event':

return event.includes(query);

case 'topic':

return topic.includes(query);

case 'version':

return version.includes(query);

default:

return false;

}

}).map(subject => {

const parts = subject.split('-');

switch(type) {

case 'event':

return parts[1];

case 'topic':

return parts[0];

case 'version':

return parts[2] || '';

default:

return '';

}

});

}

eventInput.addEventListener('input', function() {

const suggestions = filterSuggestions(this.value, 'event');

showSuggestions(this, suggestions, eventSuggestions);

});

topicInput.addEventListener('input', function() {

const suggestions = filterSuggestions(this.value, 'topic');

showSuggestions(this, suggestions, topicSuggestions);

});

versionInput.addEventListener('input', function() {

const suggestions = filterSuggestions(this.value, 'version');

showSuggestions(this, suggestions, versionSuggestions);

});

document.addEventListener('click', function(e) {

if (!eventInput.contains(e.target)) {

eventSuggestions.innerHTML = '';

}

if (!topicInput.contains(e.target)) {

topicSuggestions.innerHTML = '';

}

if (!versionInput.contains(e.target)) {

versionSuggestions.innerHTML = '';

}

});

}

</script>

</body>

</html>  
  
  
@RestController

public class SchemaController {

@Autowired

private RestTemplate restTemplate;

@Value("${schema.registry.url}")

private String schemaRegistryUrl;

@GetMapping("/getAllSubjects")

public List<String> getAllSubjects() {

String url = schemaRegistryUrl + "/subjects";

ResponseEntity<List<String>> response = restTemplate.exchange(url, HttpMethod.GET, null,

new ParameterizedTypeReference<List<String>>() {});

return response.getBody();

}

@GetMapping("/getSchema")

public String getSchema(@RequestParam("event") String event,

@RequestParam("topic") String topic,

@RequestParam("version") String version) {

String subject = topic + "-" + event + "-" + version;

return fetchSchemaFromRegistry(subject);

}

private String fetchSchemaFromRegistry(String subject) {

String url = schemaRegistryUrl + "/subjects/" + subject + "/versions/latest";

ResponseEntity<String> response = restTemplate.getForEntity(url, String.class);

return response.getBody();

}

}