Project Implementation Steps

Lien GitHub : https://github.com/alaeddine-hash/ServiceWebWeatherAndCrytoCurrency.git

Step 1: Setting Up the Spring Boot Project

- Use Spring Initializer (https://start.spring.io/) to set up your project.

- Choose necessary options, including Web, JAXB, and SOAP Web Services.

Step 2: Creating the Data Model

- Define two classes, 'WeatherInfo' and 'CurrencyInfo', to store information from both services.

Step 3: Creating the Service to Consume the REST Service

- Use 'RestTemplate' to consume the REST service.

- Register on OpenWeatherMap and obtain an API key

(6e2dba07979ec3f4d9cba0fdb5612a6d).

- Use this key to get weather data from the API.

Step 4: Creating the Service to Consume the SOAP Service

- Use JAXB to generate classes from the WSDL.

- Use 'WebServiceTemplate' to make the SOAP calls.

- Replace ” com.xignite.services ” with the path of your generated classes.

- Define <https://www.xignite.com/xCurrencies.asmx> in my property file with the URL of the SOAP service.

Step 5: Creating the Controller to Expose our API

- Create a controller to expose your API with '@RestController'.

- Set up a GET method at '/exchange-info/{location}' to return the exchange and weather info for the given location.

Step 6: Error Handling

- Create a '@ControllerAdvice' to handle global exceptions.

- Define custom exceptions (like 'WeatherServiceException', 'CurrencyServiceException') to handle any errors that occur while calling the REST and SOAP services.

Step 7: Testing the Service

- Use Postman to test your service.

- Run your Spring Boot application, create a new GET request in Postman with the URL 'http://localhost:8082/exchange-info/{location}'.

- Send the request and check the response in the lower panel of Postman.

- Test your service with different locations and currency parameters.

Step 8: Documenting the Service

- Document your service by creating a specification document.

- Include a description of your service, context, technical specifications, implementation details, and test results.

- Use a tool like Swagger to generate interactive API documentation.

Remember to handle any potential errors and thoroughly test your service for robustness and reliability.

Today, we worked on integrating the OpenWeatherMap API into the web service. We successfully retrieved weather information for a specific location, which included the temperature in Kelvin. We discussed the conversion of Kelvin to Celsius and Fahrenheit using the appropriate formulas.

To convert Kelvin to Celsius, we subtracted 273.15 from the temperature value. To convert Kelvin to Fahrenheit, we used the formula (Kelvin - 273.15) \* 9/5 + 32.

We applied these conversion formulas to the provided response, and the temperature was approximately 8.6°C in Celsius and 47.5°F in Fahrenheit.

Tomorrow, we can continue validating the web service by performing further testing and ensuring the functionality and reliability of the implemented features. We can also focus on any additional tasks or requirements that need to be addressed in the service