MyWeather App Documentation

Daylight Hours Comparison

The compareDaylightHours feature allows comparing the length of daylight hours between two cities and returning the city with the longest day. It is implemented in the WeatherController class.

Endpoint

GET /daylight/{city1}/{city2}

Parameters

city1: The name of the first city.

city2: The name of the second city.

Response

The response is a String indicating the city with the longest daylight hours.

Implementation

The WeatherController invokes the compareDaylightHours method in the WeatherService, passing the two city names as parameters.

In the WeatherService class, the compareDaylightHours method retrieves the CityInfo for both cities from the VisualcrossingRepository.

The sunrise and sunset times are extracted from the CityInfo objects.

The duration between sunrise and sunset is calculated for both cities using LocalTime and Duration classes.

The daylight hour durations are compared, and a result indicating the city with the longest daylight hours is returned.

Rain Check

The checkRain feature allows checking which city is currently experiencing rain. It is implemented in the WeatherController class.

Endpoint

GET /raincheck/{city1}/{city2}

Parameters

city1: The name of the first city.

city2: The name of the second city.

Response

The response is a String indicating the city where it is currently raining.

Implementation

The WeatherController invokes the checkRain method in the WeatherService, passing the two city names as parameters.

In the WeatherService class, the checkRain method retrieves the CityInfo for both cities from the VisualcrossingRepository.

The current weather conditions are extracted from the CityInfo objects.

The conditions are checked to determine if it includes the keyword "rain" (case-insensitive).

Based on the weather conditions in each city, a result indicating the city where it is currently raining is returned.

Exception Handling

The provided code does not include explicit exception handling. However, it is recommended to add appropriate exception handling to handle scenarios such as invalid city names, API errors, or network issues. Exception handling can be implemented using try-catch blocks or by utilizing Spring's exception handling mechanisms.

Testing

Writing comprehensive unit tests is essential to ensure the correctness and reliability of the implemented features. However, due to certain challenges and time constraints, unit tests have not been provided as part of this submission.

Despite this limitation, it is highly recommended to create unit tests to validate the behavior of the compareDaylightHours and checkRain methods in different scenarios.

To test these methods, you can consider the following test cases:

Valid city names with different daylight durations.

Valid city names where the daylight durations are the same.

Valid city names where it is currently raining in one or both cities.

Invalid city names to ensure appropriate exception handling.

Conclusion

In conclusion, the implementation of the two new features, "Daylight Hours Comparison" and "Rain Check," was successful within the given time constraints. However, a significant portion of the available time was dedicated to setting up the development environment, including installing the necessary software and prerequisites.