Weather Application

Overview

The Weather Application is a Spring Boot-based REST API that fetches real-time weather data and forecasts for a given city using external APIs. It uses caching to optimize performance and reduce API calls.

Features

- Fetch current weather information for a specified city.
- Retrieve weather forecasts for multiple days.
- Caching to reduce redundant API calls.
- Centralized logging for monitoring application activity.

Technologies Used

- **Backend**: Spring Boot
- **Caching**: Spring Cache
- **HTTP Client**: RestTemplate
- **Logging**: Logback with SLF4J
- **APIs**:

  OpenWeatherMap API (for current weather)

  WeatherAPI (for forecasts)

Endpoints

1. Get Current Weather

- **URL**: `/api/weather`
- **Method**: `GET`
- **Query Parameters**:
  - `city` (String): Name of the city.
- **Example Request**:
  bash
  Copy code

Request
GET http://localhost:8080/api/weather?city=Thiruvarur

Response
```
{
   "weather": "Sunny",
   "temperature": 32
}
```

. Get Weather Forecast

- **URL**: `/api/forecast`
- **Method**: `GET`
- **Query Parameters**:
  - `city` (String): Name of the city.
  - `days` (int): Number of days to forecast.

Request
GET http://localhost:8080/api/forecast?city=Chennai&days=3

Response
```
[   {       "date": "2024-12-05",       "avgTemp": 30,       "maxTemp": 34,       "minTemp": 27,
"condition": "Cloudy"    },    …
{       "date": "2024-12-08",       "avgTemp": 28.05,       "maxTemp":32,       "minTemp": 25,
"condition": "Cloudy"    }
]
```
Setup and Installation

Prerequisites

- Java 17 or later
- Maven 3.6 or later

Steps

1. Clone the repository:

```
git clone https://github.com/your-repo/weather-app.git cd weather-app
```

2. Update API keys:

- Replace `API_KEY` in `WeatherService` with your OpenWeatherMap API key.
- Replace `API_KEY1` in `WeatherService` with your WeatherAPI key.

3. Build the application:

mvn clean install

4. Run the application:

mvn spring-boot:run

5. Access the application at http://localhost:8080.

Logging

- Logs are stored in `src/weather-app.log`.

Log format:
ruby
Copy code
```
yyyy-MM-dd HH:mm:ss [thread] LEVEL Logger - Message
```

-
- Logging configuration can be adjusted in `logback-spring.xml`.

Caching

- **Cache Names**:
    - `weatherCache` for current weather data.
    - `forecastCache` for weather forecast data.
- Cache hits and misses are logged.

Error Handling

Invalid city name or unavailable data returns:
json
Copy code
```
{
    "error": "City not found. Please check the city name and try
again."
}
```

-

General API or server errors return:
json
Copy code
```json
{
    "error": "Failed to fetch weather data. Please try again later."
}
```

-

Monitoring

The application tracks:

- Total API requests.
- Cache hits and misses.

Improvements

- Integrate a frontend for visualization.
- Add support for more weather parameters.
- Expand caching mechanisms.