Web Project Report

Author

SUN Yudi (sunyudi7@gmail.com)

PAN Wenchong (wenchong.pan@etu.cyu.fr)

1. Introduction

This project aims to develop a web service using Java Dynamic Web Project for managing country and city information for a travel agency, along with integration with external weather services. It includes functionalities for managing country and city information, querying weather information, and implementing unified access and data modification through a client.

1.1. Web Services:

Package: `travel.management.web.data`

- **City.java, Country.java**
  - Attributes: Name, destination type, country ID, longitude, latitude, etc.

Package: `travel.management.web.resource`

- **CityService.java, CountryService.java**
  - Functions: Provide services related to countries and cities, such as adding, deleting, retrieving, and updating city information.

Package: `travel.management.web.service`

- **CityResource.java, CountryResource.java**
  - Functions: Define RESTful APIs for country and city resources, including operations like adding, retrieving, updating, and deleting city information.

1.2. Client:

The client interacts with the travel management system's RESTful API to manage and query country and city information.

Package: `travel.management.client`

- **API.java, TravelAgentClient.java**
  - Functions: Add country, add city, get list of cities of specific type, and display weather information for cities.

1.3. Code Management:

We use GitHub's distributed version control system for collaboration and code version management.

2. Service

Package: `travel.management.web`

Classes:

1. **CityResource**:
   - Handles HTTP requests related to cities.
   - Important functions:
     - `addCity(City city)`: Adds a new city.
     - `getCitiesByCountry(Long countryId)`: Retrieves cities by country ID.
     - `updateCity(Long cityId, City updatedCity)`: Updates city information.
     - `deleteCity(Long cityId)`: Deletes a city.
2. **CountryResource**:
   - Manages HTTP requests related to countries.
   - Important functions:
     - `addCountry(Country country)`: Adds a new country.
     - `getAllCountries()`: Retrieves all countries.
     - `getCountryById(Long countryId)`: Retrieves a country by ID.
     - `updateCountry(Long countryId, Country updatedCountry)`: Updates country information.
     - `deleteCountry(Long countryId)`: Deletes a country.

Package: `travel.management.web.data`

Classes:

1. **City**:
   - Represents a city with attributes such as name, destination type, longitude, and latitude.
   - Important functions:
     - `City(String name, String destinationType, Long countryId, double longitude, double latitude)`: Constructor for creating a city.
     - Accessor and mutator methods for accessing and modifying city attributes.
2. **Country**:
   - Represents a country with attributes such as name and a set of associated cities.
   - Important functions:
     - `Country(String name)`: Constructor for creating a country.
     - Accessor and mutator methods for accessing and modifying country attributes.

Package: `travel.management.web.service`

This package contains service classes responsible for managing city and country data.

Classes:

1. **CityService**:
    - Provides CRUD operations for managing cities.
    - Important functions:
        - `addCity(City city)`: Adds a new city.
        - `deleteCity(long id)`: Deletes a city by ID.
        - `getCity(long id)`: Retrieves a city by ID.
        - `getAllCities()`: Retrieves all cities.
        - `getCitiesByCountry(long countryId)`: Retrieves cities by country ID.
        - `updateCity(long id, City city)`: Updates city information.

2. **CountryService**:
    - Provides CRUD operations for managing countries.
    - Important functions:
        - `addCountry(Country country)`: Adds a new country.
        - `deleteCountry(Long id)`: Deletes a country by ID.
        - `getCountryById(Long id)`: Retrieves a country by ID.
        - `getAllCountries()`: Retrieves all countries.
        - `updateCountry(Long id, Country updatedCountry)`: Updates country information.

3. Client

Package: `travel.management.client`

Classes:

1. **API**:
    - Provides methods for making HTTP requests to external APIs, specifically used for retrieving weather data.
    - Important functions:
        - `getWeather(double lat, double lng, String params)`: Sends a request to the Stormglass API to fetch weather data based on latitude, longitude, and specified parameters.

2. **TravelAgentClient**:
    - Acts as the main client application for interacting with the travel management system.
    - Important functions:
        - `main(String[] args)`: Entry point of the client application. It demonstrates various functionalities such as adding a country, adding a city to a country, retrieving cities by type, and displaying weather information for cities.
        - `addCountry(String name)`: Sends a request to add a new country to the server.
        - `addCity(String name, String destinationType, double latitude, double longitude, Long countryId)`: Sends a request to add a new city to the server associated with a country.
        - `getCitiesByType(Long countryId, String type)`: Sends a request to retrieve cities of a specific type for a given country from the server.

    - `displayWeather(String cityName, double latitude, double longitude)`: Displays weather information for a city by calling the `getWeather` method.

4. Demonstration

Use Case: Adding Countries and Cities, Querying Cities of Specific Type

Steps:

1. First, the client adds a country named "France".
2. Then, the client adds a city named "Marseille" with its type set to "Beach", and associates it with the country "France".
3. Next, the client retrieves a list of all "Beach" cities within the country "France".
4. Finally, the client displays weather information for each beach city.

Results:

- The country "France" and the city "Marseille" are successfully added to the system.
- The retrieved list of beach cities contains "Marseille".
- Weather information for each beach city is successfully displayed.