Location Selection Spring Boot Project Documentation

Table of Contents

- 1. Project Overview
- 2. Technology Stack
- 3. Project Structure
- 4. Entities and Relationships
- 5. Repository Interfaces
- 6. Controller Explanation
- 7. API Endpoints
- 8. Front-end Integration
- 9. How to Run the Project
- 10. Future Improvements

1. Project Overview

This Spring Boot project provides a web interface and REST APIs to select locations through cascading dropdowns:

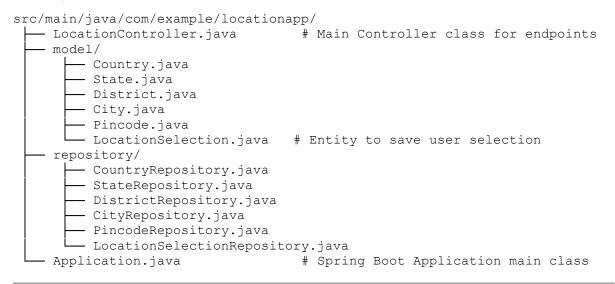
Country → State → District → City → Pincode

Users select a location step-by-step. After submitting the form, the chosen location is saved to the database.

2. Technology Stack

- Java JDK 17 or higher
- Apache Maven 3.8+
- Spring Boot 3.x
- Apache NetBeans IDE 12.5 (for full Java support)
- Visual Studio Code (for frontend or hybrid development)
- MySQL Server & MySQL Workbench 8.0 CE
- Web browser (Chrome, Firefox, Edge)

3. Project Structure



4. Entities and Relationships

Country

- o id, name
- o One-to-many relationship with State

State

- o id, name
- Many-to-one relationship with Country
- o One-to-many with District

District

- o id, name
- Many-to-one with State
- One-to-many with City

City

- o id, name
- o Many-to-one with District
- o One-to-many with Pincode

Pincode

- o id, code
- Many-to-one with City

• LocationSelection

 Stores the full selected location (Country, State, District, City, Pincode) as foreign keys.

5. Repository Interfaces

Each entity has a Spring Data JPA repository:

```
public interface CountryRepository extends JpaRepository<Country, Long> {}
public interface StateRepository extends JpaRepository<State, Long> {
    List<State> findByCountryId(Long countryId);
}

public interface DistrictRepository extends JpaRepository<District, Long> {
    List<District> findByStateId(Long stateId);
}

public interface CityRepository extends JpaRepository<City, Long> {
    List<City> findByDistrictId(Long districtId);
}

public interface PincodeRepository extends JpaRepository<Pincode, Long> {
    List<Pincode> findByCityId(Long cityId);
}

public interface LocationSelectionRepository extends
JpaRepository<LocationSelection, Long> {}
```

These repositories allow querying child entities by their parent IDs to support cascading dropdowns.

6. Controller Explanation

LocationController manages HTTP requests:

- GET /
 - Loads the form and passes the list of countries to start selection.
- GET /states?countryId=X
 - Returns states for a given country (JSON).
- GET /districts?stateId=X
 - Returns districts for a given state (JSON).
- GET /cities?districtId=X
 - Returns cities for a given district (JSON).
- GET /pincodes?cityId=X
 - Returns pincodes for a given city (JSON).
- POST /submit-location
 - Saves the full location selection into the database and redirects to / with success flag.

7. API Endpoints

Method	URL	Description	Parameters	Response Type
GET	/	Loads the	Optional: success flag	Thymeleaf
		selection form		HTML page
GET	/states	Get states for a	countryId (Long)	JSON List <state></state>
		country		
GET	/districts	Get districts for	stateId (Long)	JSON
		a state		List <district></district>
GET	/cities	Get cities for a	districtId (Long)	JSON List <city></city>
		district		
GET	/pincodes	Get pincodes	cityId (Long)	JSON
		for a city		List <pincode></pincode>
POST	/submit-	Save selected	country, state,	Redirect
	location	location	district, city,	
			pincode (Long)	

8. Front-end Integration

- Thymeleaf is used for rendering the form page.
- Cascading dropdowns are populated by AJAX calls to the JSON endpoints.
- On selection of a higher-level dropdown (e.g., Country), an AJAX call fetches the next level (States).
- On final submission, the selected IDs are sent in a POST request.
- The form redirects back with a success indicator on successful save.

9. How to Run the Project

Prerequisites

- Java 17 installed
- MySQL database running
- Maven installed
- Database schema created with tables corresponding to entities

Steps

- 1. Clone the project.
- 2. Configure application.properties with DB credentials:

```
spring.datasource.url=jdbc:mysql://localhost:3306/locationdb
spring.datasource.username=root
spring.datasource.password=your_password
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

3. Build the project:

mvn clean install

4. Run the project:

mvn spring-boot:run

5. Access the app at:

http://localhost:8080/

10. Future Improvements

- Add validation and error handling for the form.
- Add security (e.g., login/authentication).
- Add pagination if datasets are large.
- Support editing and deleting saved locations.