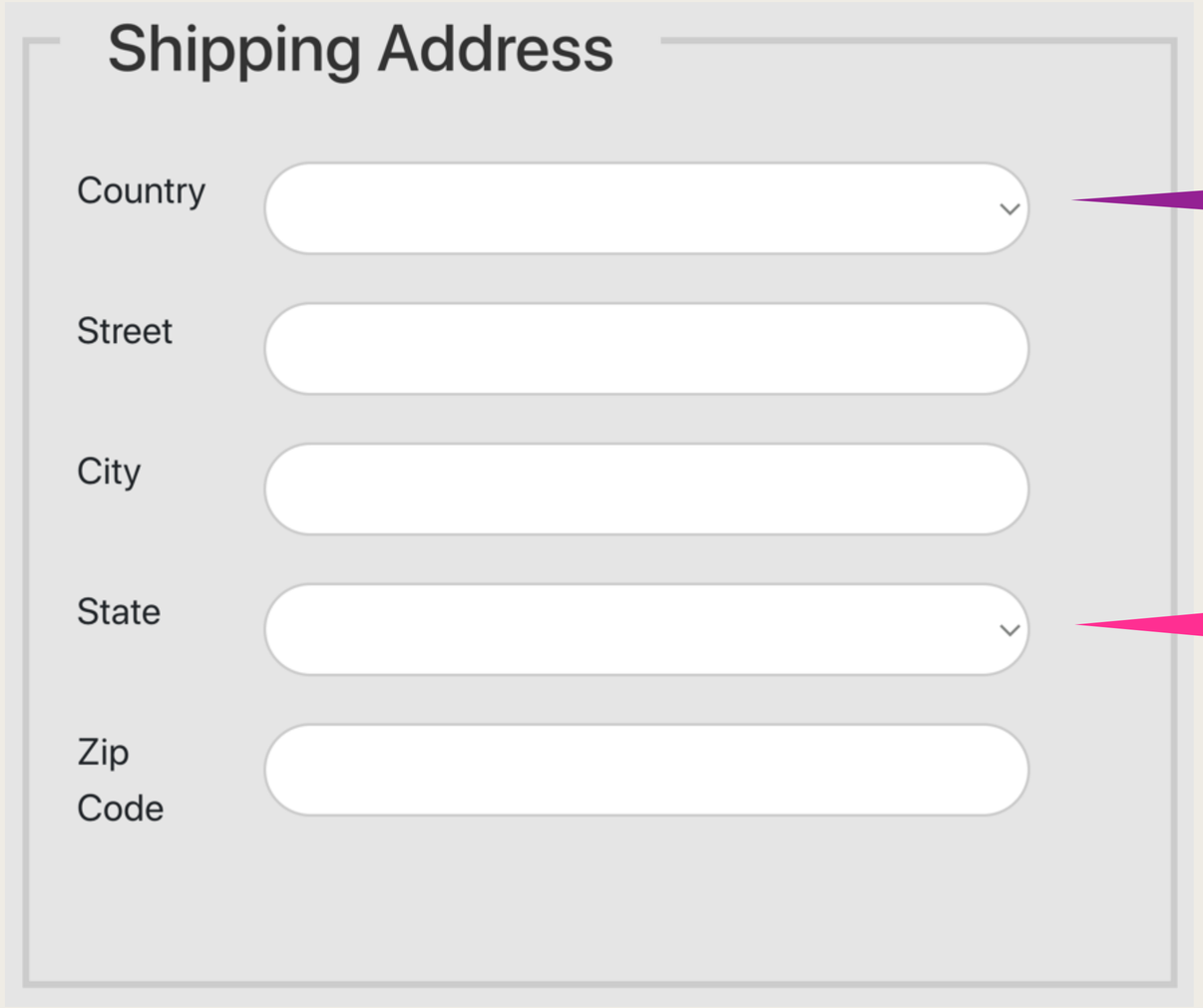
luv2code

Checkout Form
Populate Country-State Drop-Down Lists

Address - Country and State



Populating Drop-Down Lists

- Populate countries and states from the backend REST API (database)
- The user will select a country
- Depending on country selected, populate list of states
- Term "State" is different in each country
  - Similar to "Province", "District", "Region", "Préfecture" etc ...

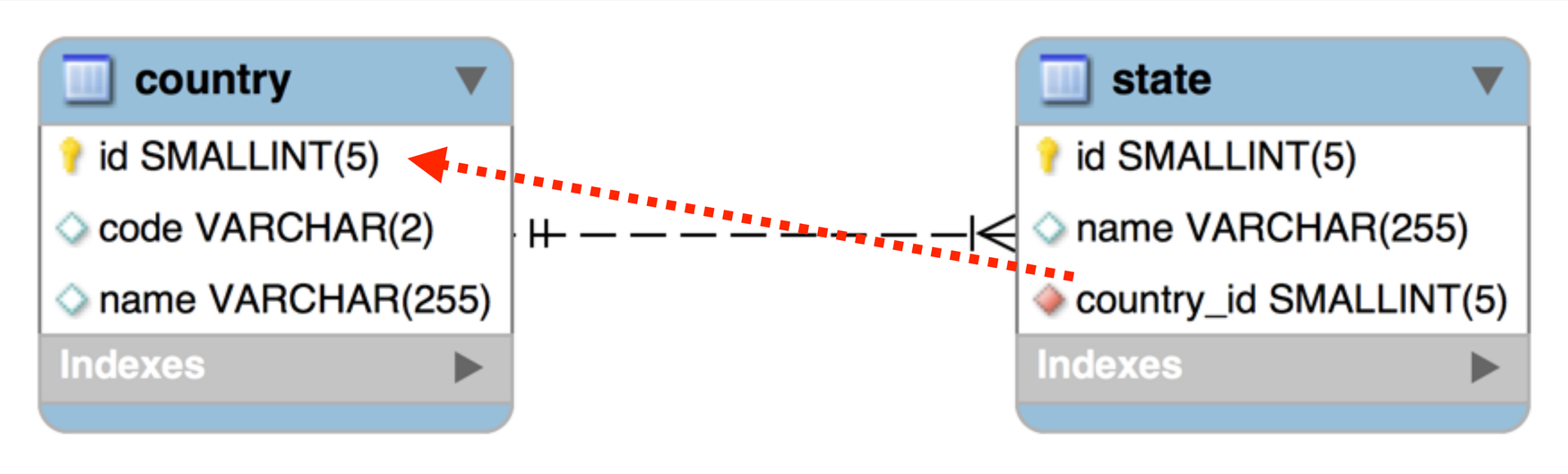
Development Process - Backend

1. Create database tables
2. Develop JPA Entities (Country, State)
3. Create Spring Data Repositories
4. Update Spring Data REST Configs

Step 1: Create Database Tables

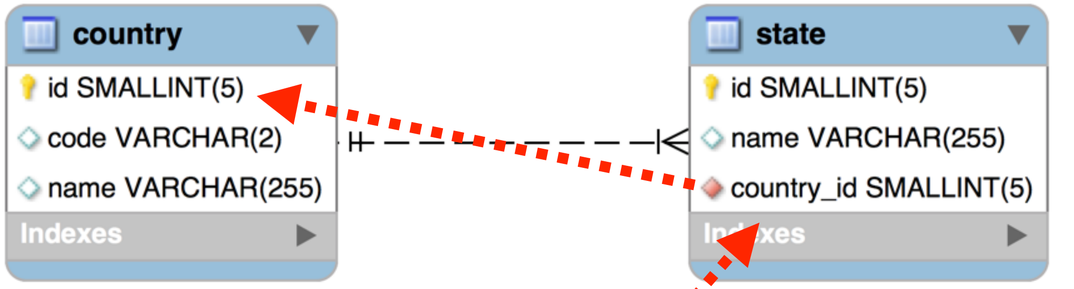
One country has many states

Many states belong to one country



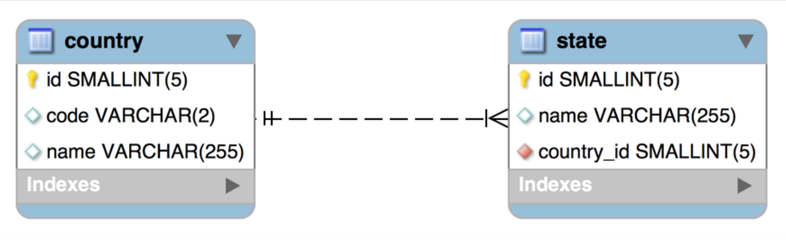
Step 1: Create Database Tables

```
CREATE TABLE country (
  id smallint unsigned NOT NULL,
  code varchar(2) DEFAULT NULL,
  name varchar(255) DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=InnoDB;
```




```
CREATE TABLE state (
  id smallint unsigned NOT NULL AUTO_INCREMENT,
  name varchar(255) DEFAULT NULL,
  country_id smallint unsigned NOT NULL,
  PRIMARY KEY (id),
  KEY fk_country (country_id),
  CONSTRAINT fk_country FOREIGN KEY (country_id) REFERENCES country (id)
) ENGINE=InnoDB AUTO_INCREMENT=1;
```

Step 1: Create Database Tables




```
INSERT INTO `country` VALUES
(1,'BR','Brazil'),
(2,'CA','Canada'),
(3,'DE','Germany'),
(4,'IN','India'),
(5,'TR','Turkey'),
(6,'US','United States');
```

id, code, name

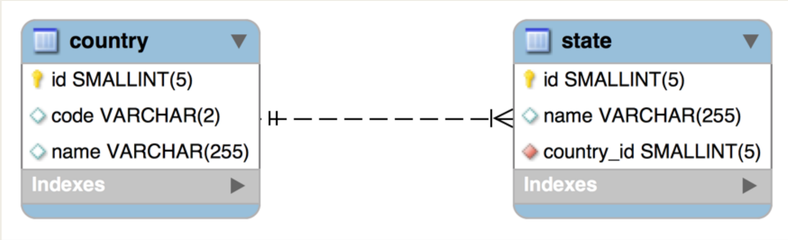
```
INSERT INTO `state` VALUES
(1,'Acre',1),
(2,'Alagoas',1),
(3,'Amapá',1),
...
(57,'Andhra Pradesh',4),
(58,'Arunachal Pradesh',4),
(59,'Assam',4),
(60,'Bihar',4),
(61,'Chhattisgarh',4),
(62,'Goa',4),
...
```

Brazil: 1

India: 4

id, name, country

Step 1: Create Database Tables




```
SELECT * FROM country;
```

id	code	name
1	BR	Brazil
2	CA	Canada
3	DE	Germany
4	IN	India
5	TR	Turkey
6	US	United States

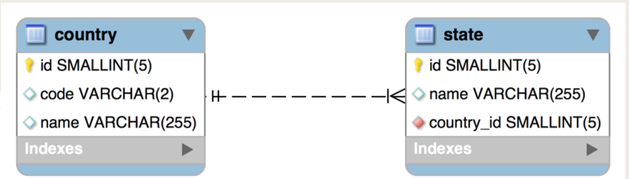
```
SELECT * FROM state where country_id=4;
```

id	name	country_id
57	Andhra Pradesh	4
58	Arunachal Pradesh	4
59	Assam	4
60	Bihar	4
61	Chhattisgarh	4
62	Goa	4
63	Gujarat	4
64	Haryana	4

Step 2: Develop JPA Entities: Country and State

```java
@Entity
@Table(name="country")
@Getter
@Setter
public class Country {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="code")
    private String code;

    @Column(name="name")
    private String name;

    @OneToMany(mappedBy="country")
    private List<State> states;

}
```

@OneToMany
One country has many states




```java
@Entity
@Table(name="state")
@Data
public class State {

    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    @Column(name="id")
    private int id;

    @Column(name="name")
    private String name;

    @ManyToOne
    @JoinColumn(name="country_id")
    private Country country;

}
```

@ManyToOne
Many states belong to one country

Step 3: Create Spring Data REST Repositories

```
@CrossOrigin("http://localhost:4200")
@RepositoryRestResource(collectionResourceRel = "countries", path = "countries")
public interface CountryRepository extends JpaRepository<Country, Integer> {

}
```

Expose /countries endpoint

To retrieve all countries

http://localhost:8080/api/countries

To retrieve country id=4

http://localhost:8080/api/countries/4

Step 3: Create Spring Data REST Repositories

To retrieve all countries

`http://localhost:8080/api/countries`


```
{
  "_embedded" : {
    "countries" : [ {
      "id" : 1,
      "code" : "BR",
      "name" : "Brazil",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/1"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/1"
        }
      }
    }, {
      "id" : 2,
      "code" : "CA",
      "name" : "Canada",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/countries/2"
        },
        "country" : {
          "href" : "http://localhost:8080/api/countries/2"
        }
      }
    }, {
```


To retrieve country id=4

`http://localhost:8080/api/countries/4`


```
{
  "id" : 4,
  "code" : "IN",
  "name" : "India",
  "_links" : {
    "self" : {
      "href" : "http://localhost:8080/api/countries/4"
    },
    "country" : {
      "href" : "http://localhost:8080/api/countries/4"
    }
  }
}
```

Step 3: Create Spring Data REST Repositories

```
@CrossOrigin("http://localhost:4200")
@RepositoryRestResource
public interface StateRepository extends JpaRepository<State, Integer> {

}
```

Expose `/states` endpoint

To retrieve all states

`http://localhost:8080/api/states`

Step 3: Create Spring Data REST Repositories

```
@CrossOrigin("http://localhost:4200")
@RepositoryRestResource
public interface StateRepository extends JpaRepository<State, Integer> {

    List<State> findByCountryCode(@Param("code") String code);

}
```

India (IN)

To retrieve states for a given country code

http://localhost:8080/api/states/search/findByCountryCode?code=IN

http://localhost:8080/api/states/search/findByCountryCode?code=US

United States (US)

Step 3: Create Spring Data REST Repositories

To retrieve states for a given country code

`http://localhost:8080/api/states/search/findByCountryCode?code=IN`

`http://localhost:8080/api/states/search/findByCountryCode?code=US`

code=IN


```
{
  "_embedded" : {
    "states" : [ {
      "id" : 57,
      "name" : "Andhra Pradesh",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/57"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/57"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/57/country"
        }
      }
    }, {
      "id" : 58,
      "name" : "Arunachal Pradesh",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/58"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/58"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/58/country"
        }
      }
```


code=US


```
{
  "_embedded" : {
    "states" : [ {
      "id" : 93,
      "name" : "Alabama",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/93"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/93"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/93/country"
        }
      }
    }, {
      "id" : 94,
      "name" : "Alaska",
      "_links" : {
        "self" : {
          "href" : "http://localhost:8080/api/states/94"
        },
        "state" : {
          "href" : "http://localhost:8080/api/states/94"
        },
        "country" : {
          "href" : "http://localhost:8080/api/states/94/country"
        }
      }
```

Step 4: Update Spring Data REST Configs

- Update our coding for MyDataRestConfig.java

- Make the APIs for `/country` and `/state` read-only
  - This is reference data, no need to change it via REST API
  - Disable HTTP: PUT, POST, etc ...

- Coding details covered in next video