# Spring Data Exam

# Explore volcano

## Functionality Overview

The application should be able to easily **import** hard-formatted data and **support functionalities** for also **exporting** the imported data. The application is called – **Explore volcano**.

Look at the pictures below to see what must happen:

* The home page before importing anything:

The import JSON page before importing anything:

* Import the **countries** first:
* Import the **volcanoes** second:
* The import JSON page after importing both files
* The import XML page before importing the given data
* Import the **volcanologist's** data:
* The import XML page after importing the data:
* The home page after the data is imported:
* Export all active volcanoes higher than 3000m and info about the last eruption:

To help you ensure the correctness of your solutions and **gain immediate feedback**, we have included a set of **commented-out tests** within the provided skeleton project. These tests are designed to verify the functionality of the tasks you are required to implement.

## Model Definition

There are 3 main models that the **Explore volcano database** application should contain in its functionality.

We have provided you with an Entity-Relationship (ER) diagram that represents the database schema. The ER diagram displays the tables and their relationships, along with the field types for each table.

Ensure the **Java** code **validates** data against the specified constraints before persisting it into the database. Handle any validation errors gracefully and provide meaningful feedback to the users.

Design them most appropriately, considering the following **data constraints**:

### Country

* id – accepts **integer** values, a **primary identification field, and an auto incremented field**.
* name – accepts **char sequence** (between **3** to **30** both inclusive). The values are **unique in the database**. It **cannot be nullable**.
* **capital** - accepts **char** **sequence** (between **3** to **30** both inclusive). It can be nullable.
* Constraint: The countries table has a relation with the volcanoes table. It can be nullable.

### Volcano

* id – accepts **integer** values, a **primary identification field, an auto incremented field**.
* name - accepts **char** **sequence** (between **2** to **30** both inclusive). The values are **unique in the database**. It **cannot be nullable**.
* elevation - The highest point of the volcano. Accepts only positive numbers. It **cannot be nullable**.
* volcano type - categorization of the volcanoes. String enumeration, one of the following – **CINDER\_CONE, STRATOVOLCANO, SHIELD\_VOLCANO, LAVA\_DOME, CALDERA**. It can be nullable.
* **is active** – accepts a true or false, representing whether the volcano is active or not. It **cannot be nullable**.
* **last eruption** – indicates when the last eruption occurred. It can be nullable.
* Constraint: The volcanoes table has a relation with the countries table. It can be nullable.

### Volcanologist

* id - accepts **integer** values, a **primary identification field, an auto incremented field**.
* first name - accepts **char** **sequence** (between **2** to **30** both inclusive). **Unique,** it **can not be nullable**.
* last name - accepts **char** **sequence** (between **2** to **30** both inclusive). **Unique,** it **can not be nullable**.
* salary - accepts positive number values. It **can not be nullable**.
* age - accepts number values that are between **18** and **80** both inclusive. It **can not be nullable**.
* exploring from - a date in the "**yyyy-MM-dd**" format. It can be nullable.
* Constraint: The volcanologists table has a relation with volcanoes table. It can be nullable.

### Relationships

Your partners gave you a little hint about the more complex relationships in the database so that you can implement it correctly.

One **Volcanologist** may explore only one **Volcano**, but one **Volcano** may be explored by many **Volcanologists**.

One **Country** can have many **Volcanoes**, but one **Volcano** is located in only one **Country**.

#### Constraint

* Name the entities and their class members **exactly** in the **format stated** above.
* All fields are **NOT NULL** unless explicitly stated to be nullable.

## Data Import

Use the provided files to populate the database with data. Import all the information from those files into the database.

**You are not allowed to modify the provided files.**

**ANY INCORRECT** data should be **ignored** and a message should be printed:

**"Invalid {country/volcano/volcanologist}"**

**When the import is finished:**

**"Successfully imported {country/volcano/volcanologist} {countryName – capitalName/volcanoName – of type volcanoType/full name}"**

**Judge** will only accept **file paths** in a specific format. When dealing with file paths for files, please adhere to the following format: "src/main/resources/files/xml/format-example.xml"

### JSON Import

Your new colleagues have prepared some JSON data for you to import.

#### Countries (countries.json)

##### Constraint

**If a country with the same name already exists in the DB return "Invalid country".**

**When the import is finished:**

**"Successfully imported country {countryName – capitalName}"**

#### Volcanoes (volcanoes.json)

##### Constraint

* **If a volcano with the same name already exists in the DB return "Invalid volcano".**
* **When the import is finished:**

**"Successfully imported volcano {volcanoName} of type {volcanoType}"**

* **The provided country ids will always be valid.**

### XML Import

Your new colleagues have prepared some XML data for you to import.

#### Volcanologists (volcanologists.xml)

##### Constraint

* **If an volcanologist with the same full name (first name and last name) already exists in the DB return "Invalid volcanologist".**
* **If a volcanologist is exploring volcano that doesn't exist in the DB return "Invalid volcanologist".**
* **When the import is finished:**

**"Successfully imported volcanologist {full name}"**

Please be aware that due to variations in local settings on different computers, the representation of decimal numbers (Double) may differ. In some regions, the decimal separator is a comma (,), while in others, it is a dot (.).

Judge local settings represent all decimal numbers using a dot (.) as the decimal separator. *Locale.US*

## Data Export

Get ready to export the data you have imported in the previous task. Here you will have some complex database querying. Export the data in the formats specified below.

### Export all active volcanoes with elevation above 3000m. and information about the last eruption from the Database

* Extract from the database, the **volcano name, location (country name) and the date of last eruption**.
* **Filter only volcanoes that are active with an elevation of more than 3000m. and have information about the last eruption. Order the results descending by elevation.**
* Return the information in this format:

**"Volcano: {volcano name}**

**\*Located in: {country name}**

**\*\*Elevation: {elevation}**

**\*\*\*Last eruption on: {lastEruption}**

**. . ."**