Database Frameworks – Spring Data Retake Exam

Hiberspring Inc.

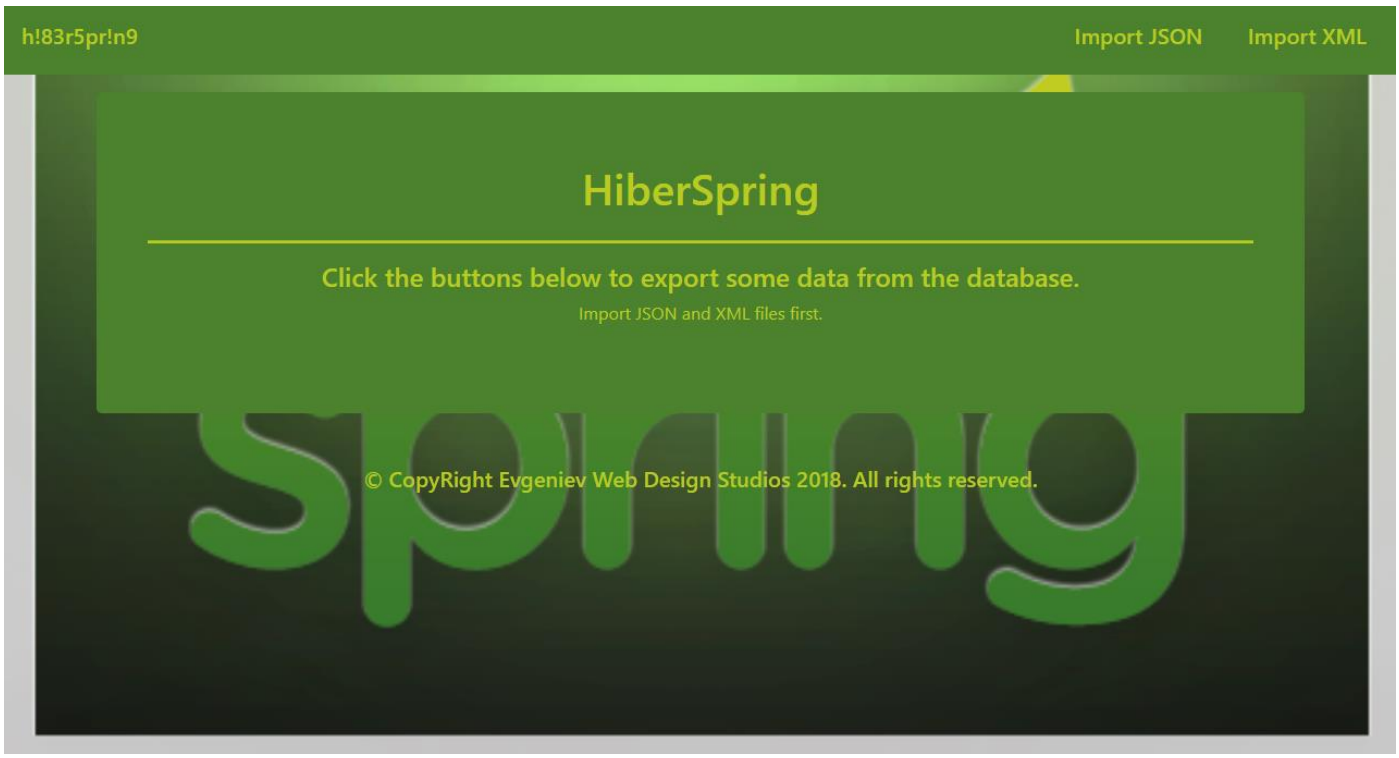
Hiberspring Incorporation is a new non-profit organization which has products of almost any type.
By some rare circumstances, though, it became the monopoly of the garbage bag business part of the market.
Anyways … Since they are a non-profit organization, they had to find a non-profit charitable database specialist for their database application. Guess what, you fit the description perfectly.

1. Functionality Overview

The application should be able to easily **import** hard-formatted data from **XML** and **JSON** and **support functionality** for also **exporting** the imported data. The application is called – **Hiberspring Inc**.

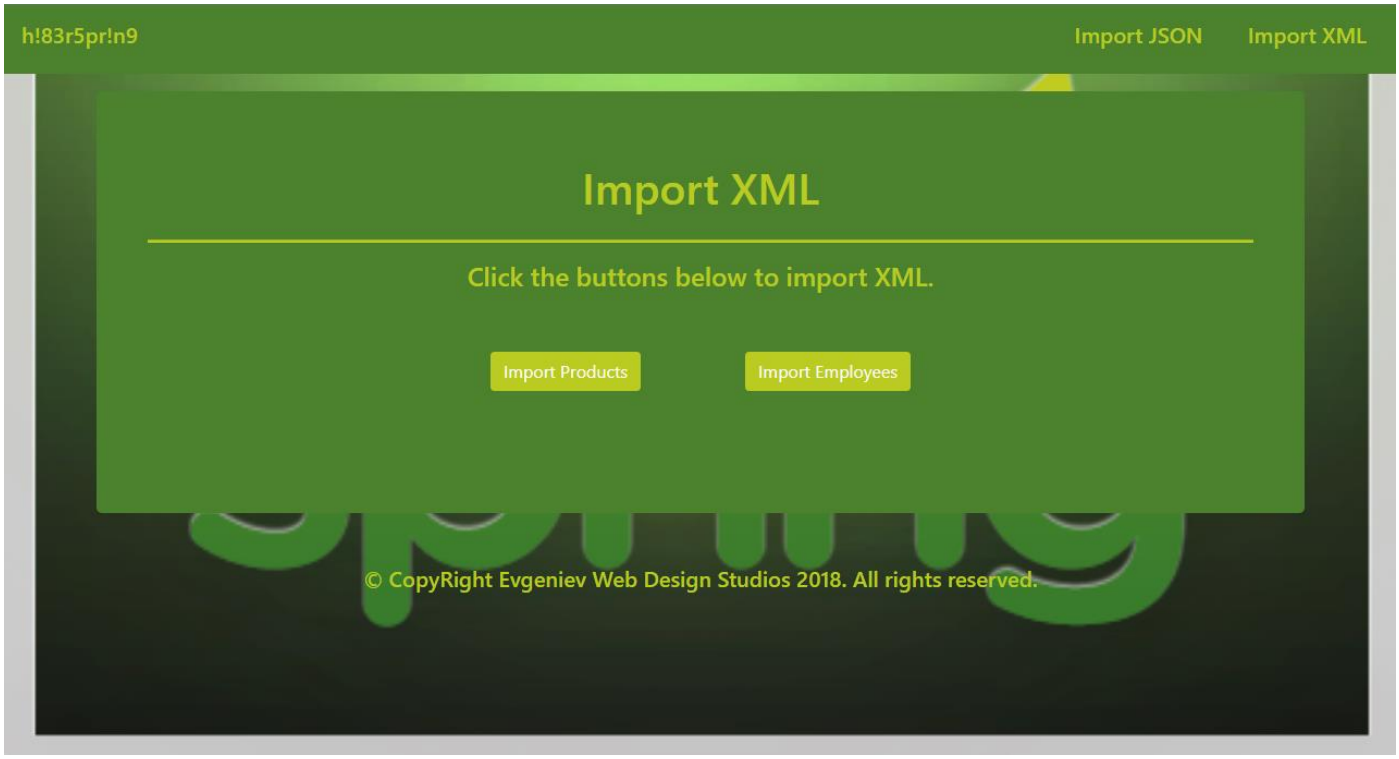
Look at the pictures below to see what must happen:

- Home page before importing anything:

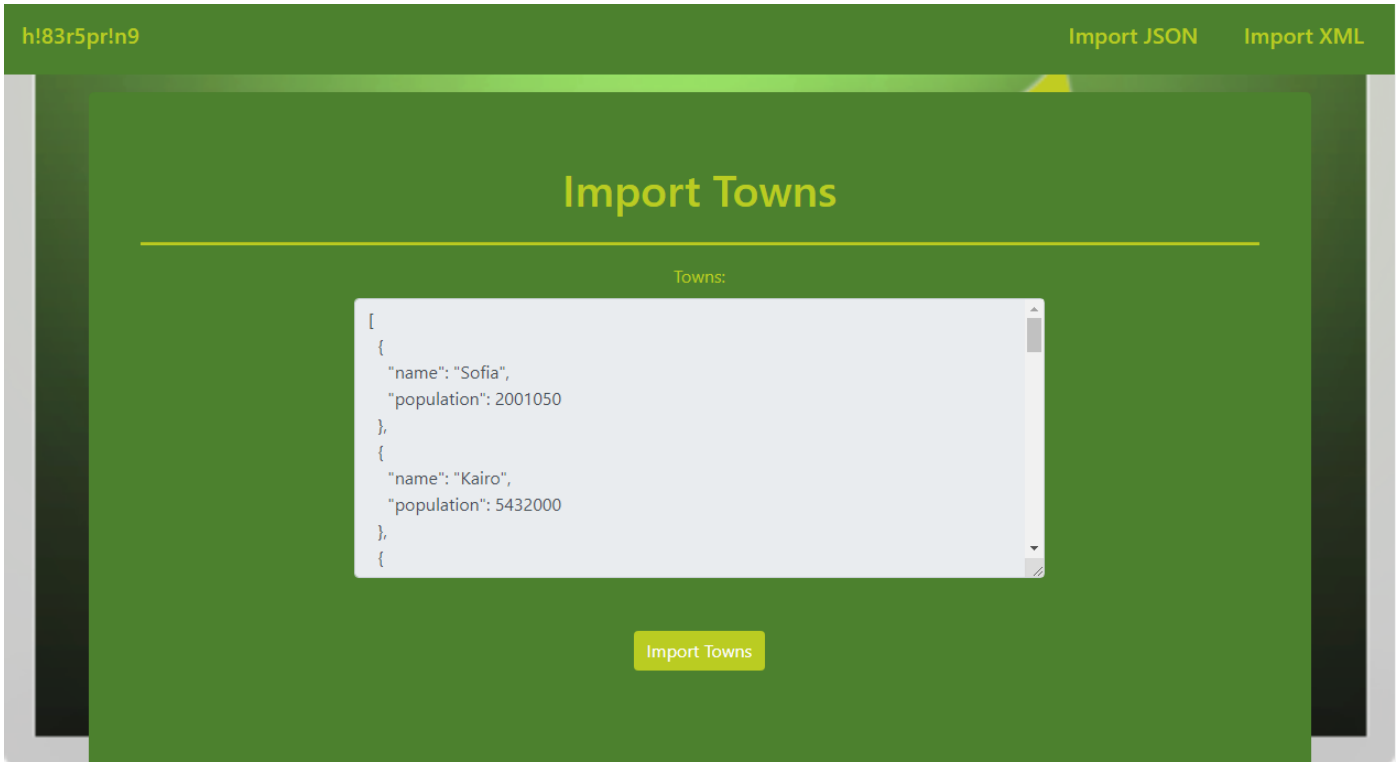


- Import JSON page before importing anything:

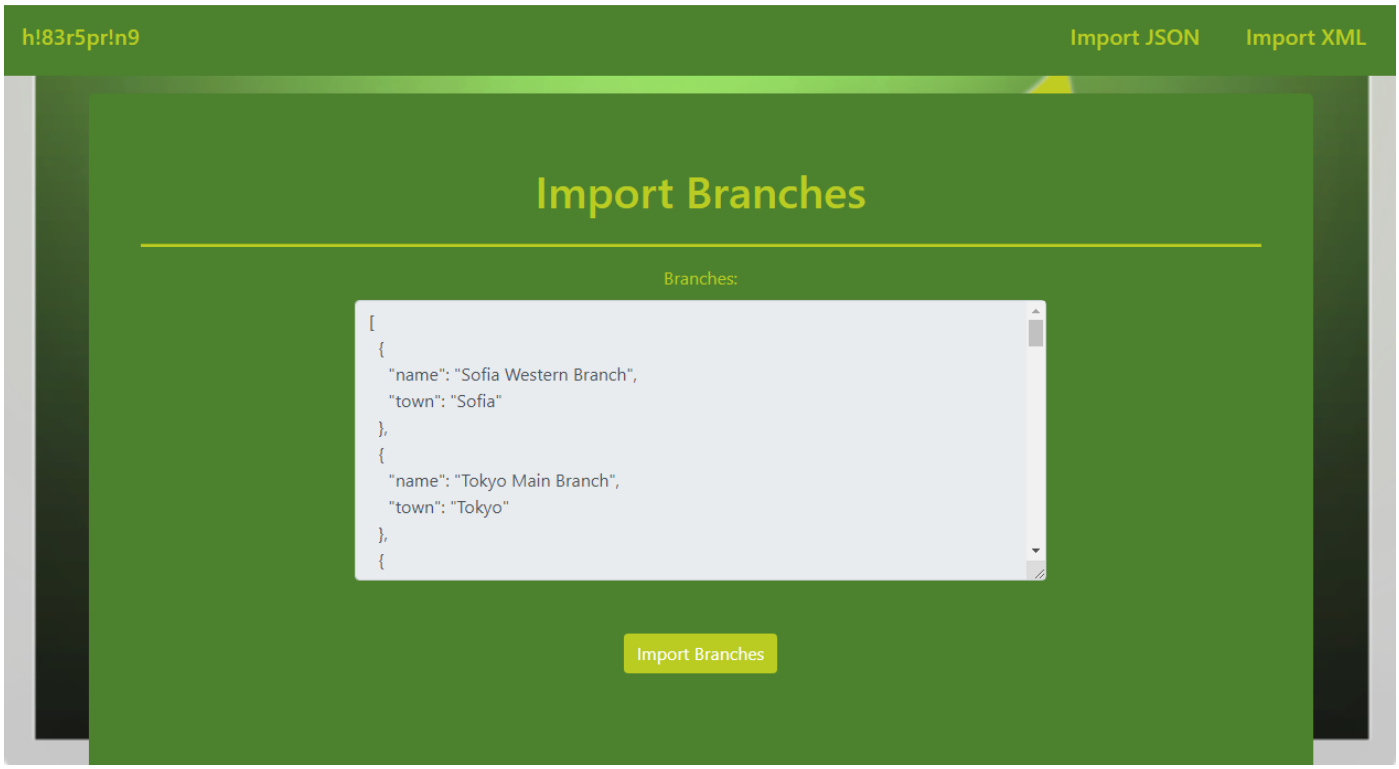
- Import XML page before importing anything:



- Import Towns page after reading the **towns.json** file:

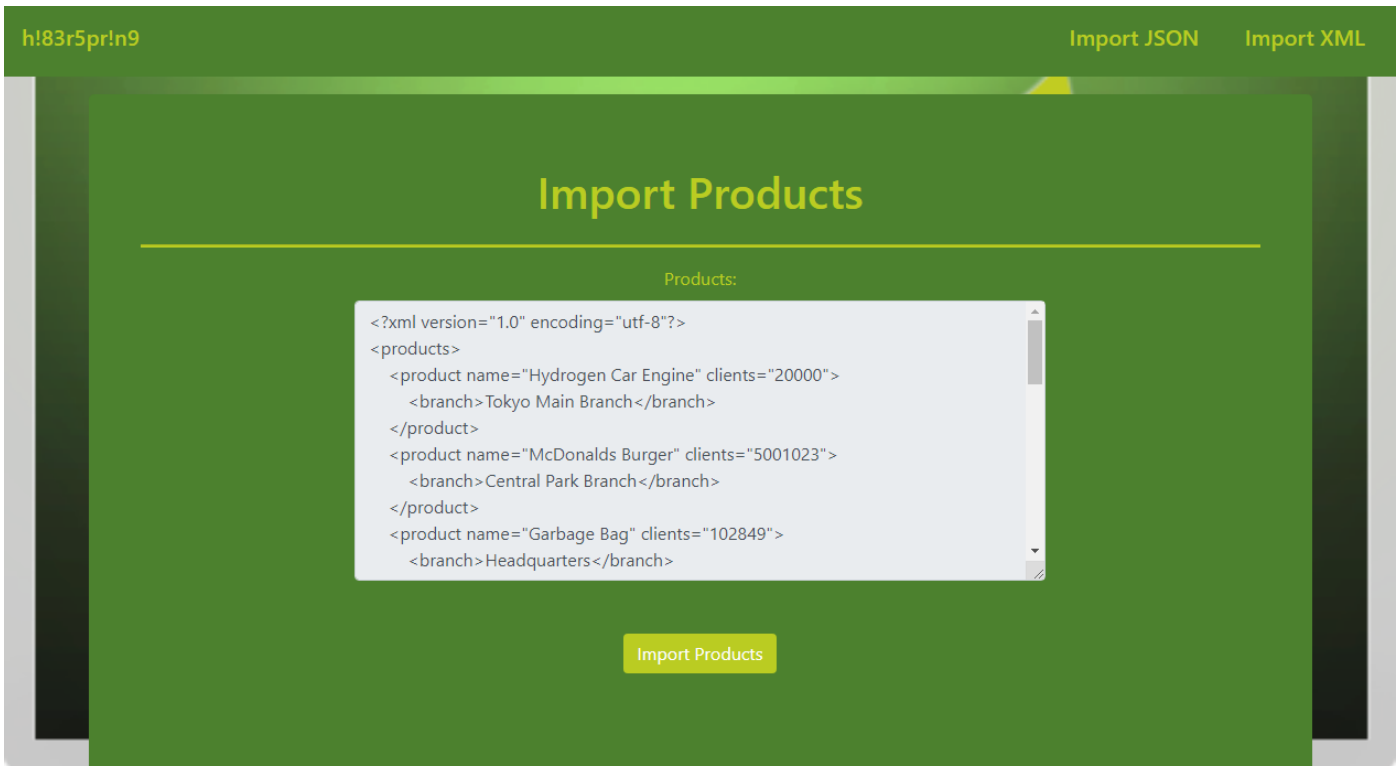


- Import Branches page after reading the **branches.json** file:



- Import Employee Cards page after reading the **employee_cards.json** file:

- Import Products page after reading **products.xml** file:

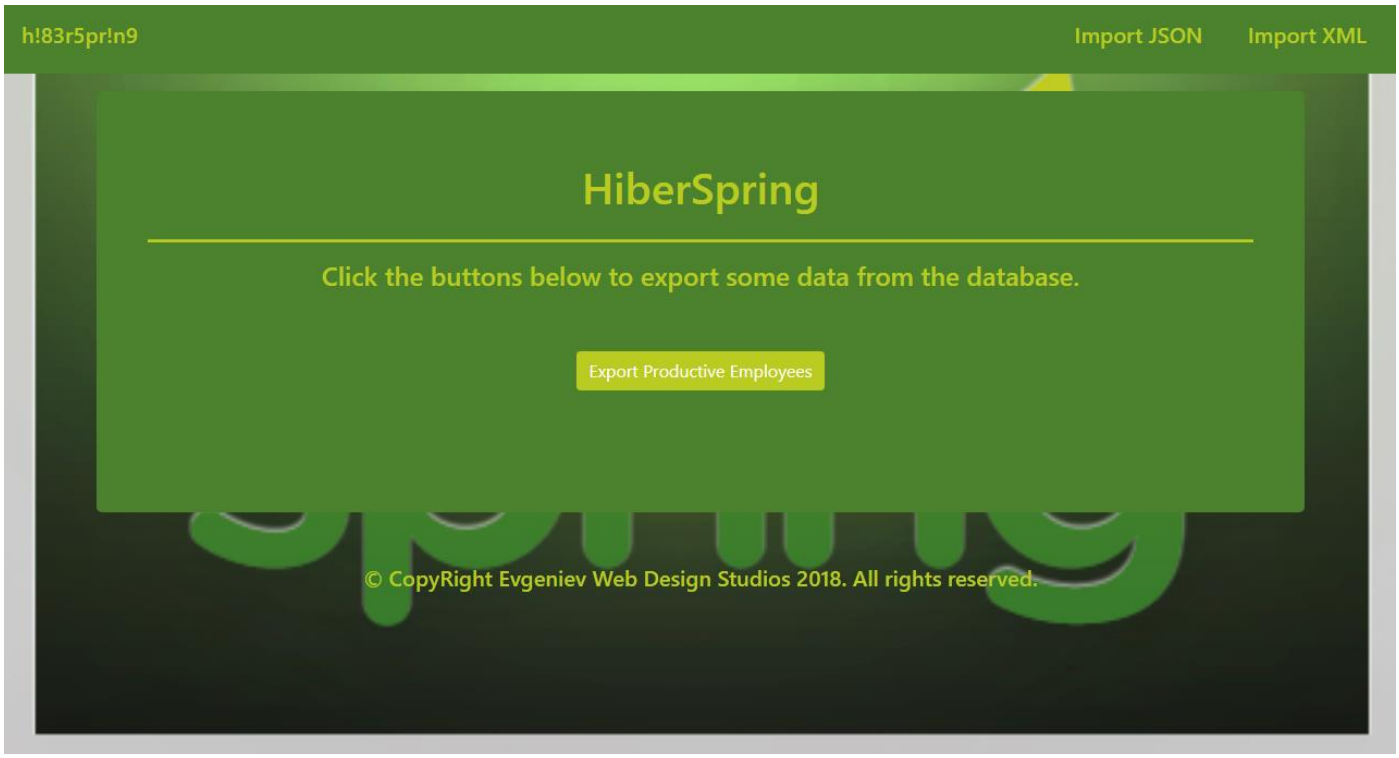


- Import Employees page after reading **employees.xml** file:

- Import JSON page after importing the given data:

- Import XML page after importing the given data:

- Home page after importing the given data:



- Export Productive Employees page:

2. Project Skeleton Overview

You will be given a **Skeleton**, containing a **certain architecture(MVC)** with **several classes**, some of which – completely empty. The **Skeleton** will include the **files** with which you will **seed** the **database**.

3. Model Definition

You can see here what properties each model has:

Employee

- **`Id`** – an **integer**.
- **`First Name`** – a **string**.
- **`Last Name`** – a **string**.
- **`Position`** – a **string**.
- **`Card`** – an **`EmployeeCard`**, could be any **`EmployeeCard`**. Must be **UNIQUE though**.
- **`Branch`** – a **`Branch`**, could be any **`Branch`**.

EmployeeCard

- **`Id`** – an **integer**.
- **`Number`** – a **string**. Should be **UNIQUE**.

Branch

- **`Id`** – an **integer**.
- **`Name`** – a **string**.
- **`Town`** – a **`Town`**, could be any **`Town`**.

Product

- **Id** – an **integer**.
- **Name** – a **string**.
- **Clients** – an **integer**.
- **Branch** – a **Branch**, could be any **Branch**.

Town

- **Id** – an **integer**
- **Name** – a **string**.
- **Population** – an **integer**.

All data is **REQUIRED**, unless it is explicitly said that it **null** is **allowed**.

4. Data Import

So here comes the **Importing** of **data** and the **populating** of the **database**. You have to **import** data from **JSON** and **XML** files.

Implement the needed **DTOs** for the imports.

Make sure all fields have been entered, otherwise the import **entity** data **should NOT be considered valid**.

You will also have to print a simple message indicating if the data has been imported successfully or there was an error.

In case of **SUCCESS** the message format is "**`Successfully imported {entityClassName} {entityField}.`**". The **`entityField`** may vary, from **`card number`** to **`employee full name`**, to just **`name`**.

In case of **ERROR** you always print "**`Error: Invalid data.`**".

Importing from JSON format

Towns

Input

towns.json

```
[
      { "name" : "Sofia", "population" : 2001050 },
      { "name" : "Kairo", "population" : 5432000 },
      { "name" : "New York", "population" : 11563790 },
      { "name" : "Tokyo", "population" : 27634593 },
      { "name" : "Moscow", "population" : 4523120 },
      { "name" : "Rome", "population" : 3021333 },
      { "name" : "Madrid", "population" : 7403213 },
      { "name" : "Paris", "population" : 8900043 },
      { "name" : "Zanzibar" },
      { "name" : "Rio de Janeiro", "population" : 6345231 },
```

```
      . . .
]
```

Output

```
Successfully imported Town Sofia.
Successfully imported Town Kairo.
Successfully imported Town New York.
Successfully imported Town Tokyo.
Successfully imported Town Moscow.
Successfully imported Town Rome.
Successfully imported Town Madrid.
Successfully imported Town Paris.
Error: Invalid data.
Successfully imported Town Rio de Janeiro.
. . .
```

Branches

Input

branches.json

```
[
      { "name" : "Sofia Western Branch", "town" : "Sofia" },
      { "name" : "Tokyo Main Branch", "town" : "Tokyo" },
      { "name" : "Headquarters", "town" : "Sofia" },
      { "town" : "New York" },
      { "name" : "Kairo Central Branch", "town" : "Kairo" },
      { "name" : "Tokyo Underground Branch", "town" : "Tokyo" },
      { "name" : "USA Main Branch", "town" : "Washington DC" },
      { "name" : "Sofia Eastern Branch", "town" : "Sofia" },
      { "name" : "Central Branch of New York" },
      { "name" : "Central Park Branch", "town" : "New York" },
      . . .
]
```

Output

```
Successfully imported Branch Sofia Western Branch.
```

```
Successfully imported Branch Tokyo Main Branch.
Successfully imported Branch Headquarters.
Error: Invalid data.
Successfully imported Branch Kairo Central Branch.
Successfully imported Branch Tokyo Underground Branch.
Successfully imported Branch USA Main Branch.
Successfully imported Branch Sofia Eastern Branch.
Error: Invalid data.
Successfully imported Branch Central Park Branch.
. . .
```

EmployeeCards

Input

employee_cards.json

```
[
      { "number" : "zi4n5-y41Pq-ugz5v-3vrNH-Dv21y" },
      { "number" : "UAIP0-0UVao-3axBt-vWF8c-45paZ" },
      { "number" : "65RrK-NRzLZ-pJLZN-Chp3q-tovmA" },
      { "number" : "DXKwE-pprkA-dLT9g-bGnbp-1304U" },
      { "number" : "3mQuf-dGsVC-v5RhD-esuzu-0XcXp" },
      . . .
]
```

Output

```
Successfully imported Employee Card zi4n5-y41Pq-ugz5v-3vrNH-Dv21y.
Successfully imported Employee Card UAIP0-0UVao-3axBt-vWF8c-45paZ.
Successfully imported Employee Card 65RrK-NRzLZ-pJLZN-Chp3q-tovmA.
Successfully imported Employee Card DXKwE-pprkA-dLT9g-bGnbp-1304U.
Successfully imported Employee Card 3mQuf-dGsVC-v5RhD-esuzu-0XcXp.
. . .
```

Importing from XML format

The other **2 tables** must be populated with data in **XML** format.

Products

Input

products.xml

```
<?xml version="1.0" encoding="utf-8"?>
<products>
  <product name="Hydrogen Car Engine" clients="20000">
    <branch>Tokyo Main Branch</branch>
  </product>
  <product name="McDonalds Burger" clients="5001023">
    <branch>Central Park Branch</branch>
  </product>
  <product name="Garbage Bag" clients="102849">
    <branch>Headquarters</branch>
  </product>
  ...
</products>
```

Output

```
Successfully imported Product Hydrogen Car Engine.
Successfully imported Product McDonalds Burger.
Successfully imported Product Garbage Bag.
. . .
```

Employees

Input

employees.xml

```
<?xml version="1.0" encoding="utf-8"?>
<employees>
  <employee first-name="John" last-name="Winchester" position="Security Manager">
    <card>zi4n5-y41Pq-ugz5v-3vrNH-Dv21y</card>
    <branch>USA Main Branch</branch>
  </employee>
  <employee first-name="Leeroy" last-name="Gips" position="Security Manager">
    <card>3mQuf-dGsVC-v5RhD-esuzu-0XcXp</card>
    <branch>Kairo Central Branch</branch>
```

```
  </employee>
  <employee first-name="Rick" last-name="Sanchez" position="Head Scientist">
    <card>65RrK-NRzLZ-pJLZN-Chp3q-tovmA</card>
    <branch>Headquarters</branch>
  </employee>
  <employee first-name="Tony" last-name="Dolfin" position="Cleaner">
    <card>a45xz-dkgw1-zadv1-aXXXc-491Az</card>
  </employee>
  . . .
</employees>
```

Output

```
Successfully imported Employee John Winchester.
Successfully imported Employee Leeroy Gips.
Successfully imported Employee Rick Sanchez.
Error: Invalid data.
. . .
```

5. Data Export

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

Productive Employees

Extract all Employees, who are **working** in a **Branch**, which has **at least one product**.

- Extract the **Employee's full name** (**first name** + **' '** + **last name**), the **Employee's position**, and the **Employee's Card's Number**.
- Order the data by **full name** in **alphabetical order**, and then by **length** of **position** in **descending order**.
- The format is described below:

  Name: {employee1Name}

  Position: {employee1Position}

  Card Number: {employee1CardNumber}

  Name: {employee2Name}

  Position: {employee2Position}

  Card Number: {employee2CardNumber}

  …

```
Name: Alex Mercer
Position: Head Security
Card Number: Vy2un-LBvJU-31FsV-GbD6B-WUkBT
----------------------------------------
Name: Jake Drinkwater
Position: Head Biologist
Card Number: M3Y0n-A10Ev-lCk8M-8BtnZ-25Rxv
----------------------------------------
Name: Jefrey Hunnington
Position: Office Manager
```