Table of Contents

[**Purpose of this document:** 2](#_Toc14617147)

[**Problem Statement Question 3:** 2](#_Toc14617148)

[**Expected Result:** 2](#_Toc14617149)

[**Overview of the System:** 2](#_Toc14617150)

[**Features:** 2](#_Toc14617151)

[**Assumptions:** 2](#_Toc14617152)

[**Improvements:** 2](#_Toc14617153)

[**Tools Used:** 3](#_Toc14617154)

[**Architecture and Flow:** 3](#_Toc14617155)

[**Steps followed for Building Application:** 4](#_Toc14617156)

[**Source Code:** 7](#_Toc14617157)

[**Steps to run the application:** 8](#_Toc14617158)

[**Testing the application:** 8](#_Toc14617159)

## **Purpose of this document:**

This document contains the high level design details of Bank Information Management Service. It includes the architecture of the system and use cases.

## **Problem Statement Question 3:**

Given the following content of a csv file:

name;bank\_identifier

Postbank;10010010

Eurocity;10030700

Commerzbank;10040000

Raiffeisenbank;22163114

1. Write a program which imports the entries of the csv file into a database of your choice

(preferably PostgreSQL)

2. Fetch the record with the bank identifier 10040000 from DB and print the name of the related bank to system out.

## **Expected Result:**

Scala play framework or Scala vert.x project with implemented algorithm. Step by step explanation of how to run the project.

## **Overview of the System:**

Bank Information Management system maintains the banks information. It maintains the name and code of each Bank. Each bank will have the unique code to identify it. We can find the find out the bank name based on this unique code from database.

## **Features:**

This is a simple Bank information management system. Which has two end points

1. <http://localhost:9000/processCSV> - GET - Responsible for reading the CSV file and delete all rows from table and insert new records from CSV.
2. [http://localhost:9000/bank/{code}](http://localhost:9000/bank/%7bcode%7d) – GET – Responsible for finding the bank information based on given code.

## **Assumptions:**

1. Security for the actions (end points) is not provided.
2. CSV file is already existed in the application/public folder with the data.

## **Improvements:**

1. Security can be provided with any one of the authentication mechanisms.
2. CSV file can get from Frontend form and process it.

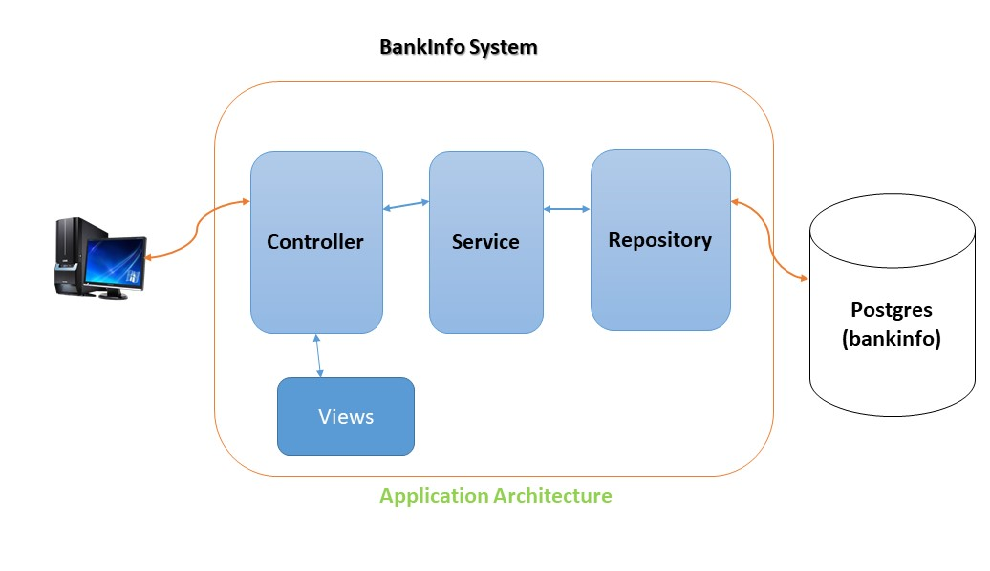
## **Tools Used:**

1. Java 1.8
2. Scala 2.12.8
3. IntelliJ IDEA with Scala plugin
4. Postgres 11.x Database
5. Play Framework 2.7.3 (MVC framework) + Anorm (to do database operations)
6. SBT build tool.

## **Architecture and Flow:**

Bank Information management system built as multi layered application for clear separation of responsibilities. **Followed MVC architecture** to differentiate the backend logic and front end logic.

I have chosen **Play Framework 2.7.3** for MVC web framework and **Anorm (plugin to Play)** as a layer to do database operations. Play is a high-productivity Java and Scala web application framework that integrates the components and APIs you need for modern web application development.



Bank Info system contains below layers as show in the above architecture diagram.

1. Controller (Web layer)
2. Service (For complete business logic)
3. Repository (Database related logic)

Play supports the Dependency Injection, and injected all the dependent objects using @Inject annotation. This avoids the creation of objects on our own using new operator instead server will create all the objects and injected.

**Postgres** database used to store the bank information. Created bank table using below query

**create table bank (name varchar(250), bank\_identifier int);**

Developed two end points and two views accordingly to fulfil the two expected result.

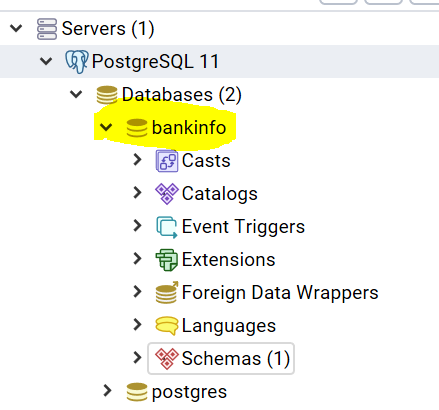
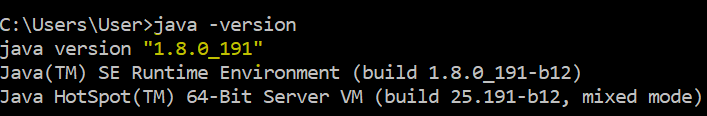
1. <http://localhost:9000/processCSV> - GET - Responsible for reading the CSV file and delete all rows from table and insert new records from CSV.
2. [http://localhost:9000/bank/{code}](http://localhost:9000/bank/%7bcode%7d) – GET – Responsible for finding the bank information based on given code.

## **Steps followed for Building Application:**

**Step 1:**

Checked/installed for all the prerequisite software’s available on system. Prerequisite software’s are.

1. Java 1.8
2. SBT Build tool
3. Postgres 11.4.2 with **bankinfo** schema

**Step 2:**

Create a bank table with two columns name, bank\_identifier in bankinfo database. Now bank table will be empty with two columns.

**create table bank (name varchar(250), bank\_identifier int);**

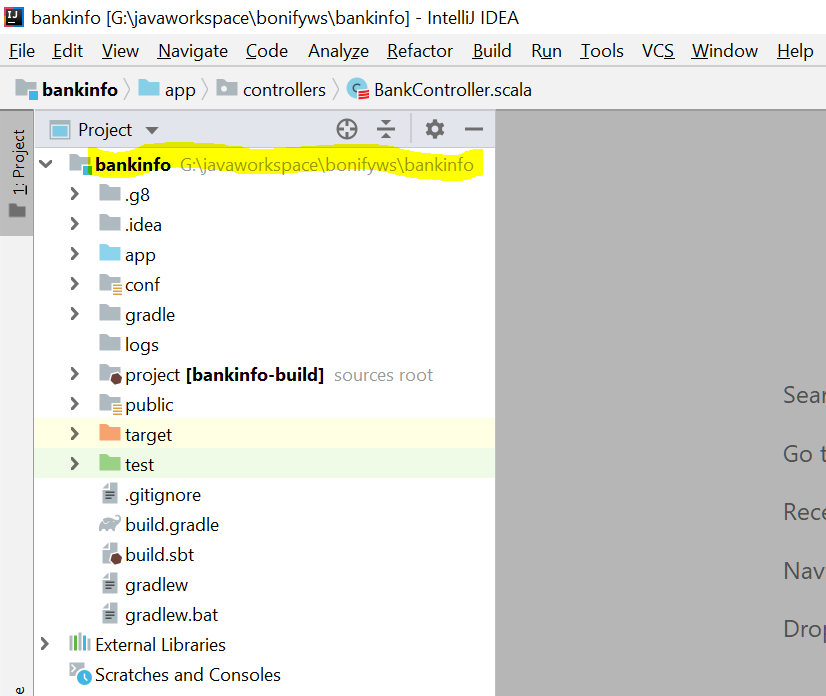
**Step 3:**

Create a play application using below command from console.

sbt new playframework/play-scala-seed.g8

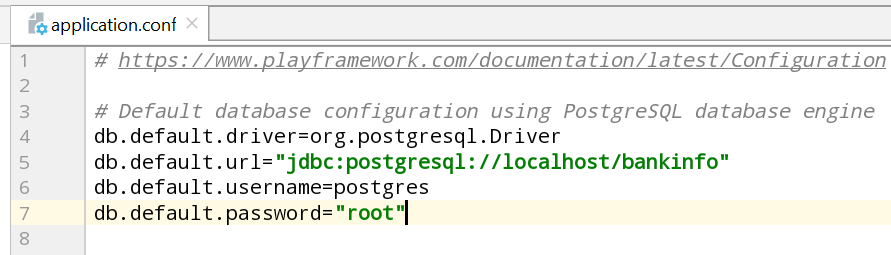
**Step 4:**

Import the created application into the IntelliJ IDE using import options while IntelliJ startup.

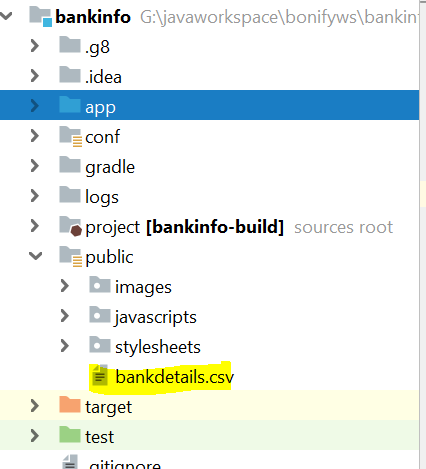


**Step 5:**

Open the **application.conf** file and add the database configuration properties in **application.conf** file. It will give the flexible to change the database urls, usernames and passwords with ease.



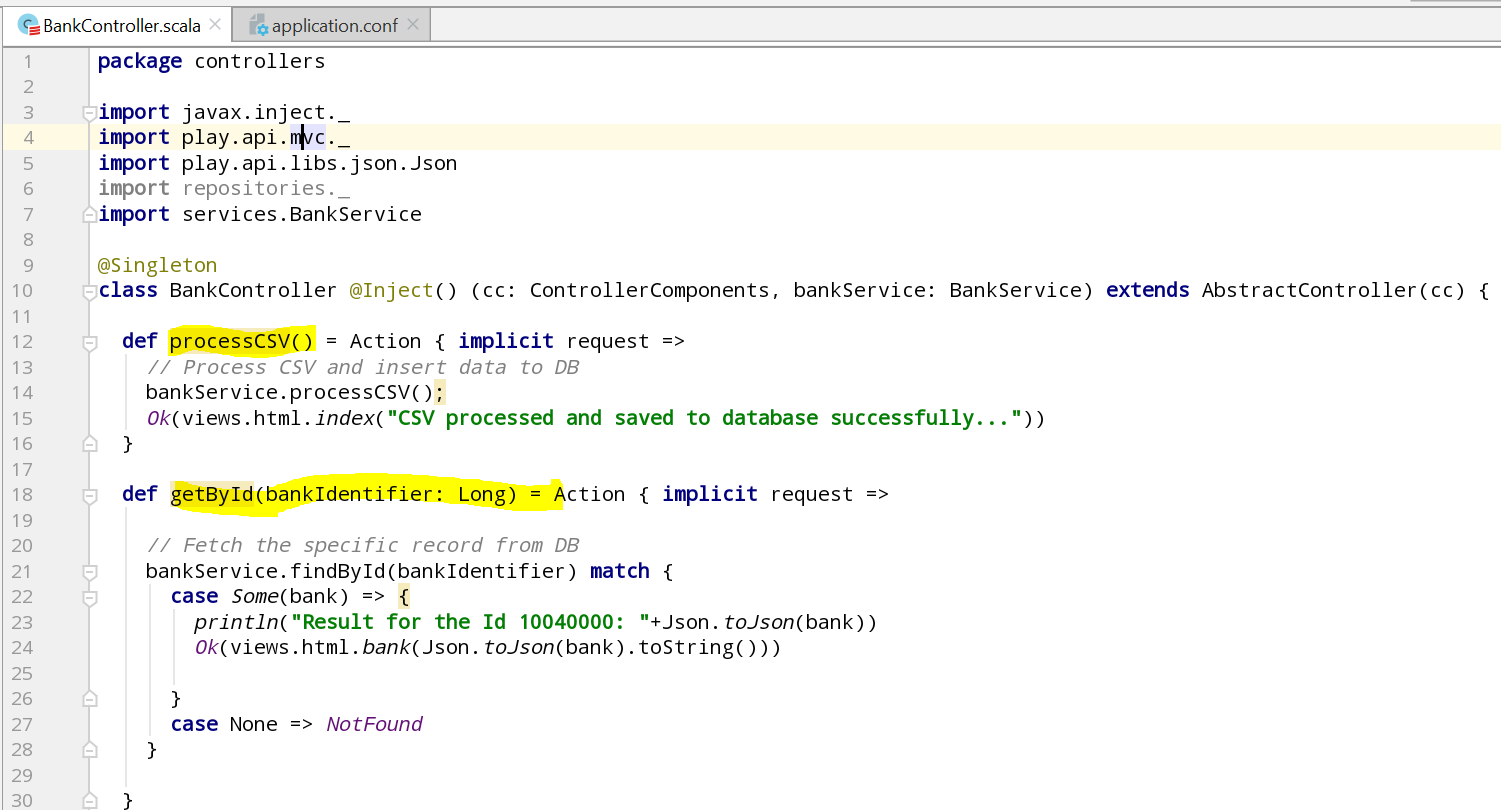
Create the bankdetails.csv with delimiter as semicolon and place in public folder.



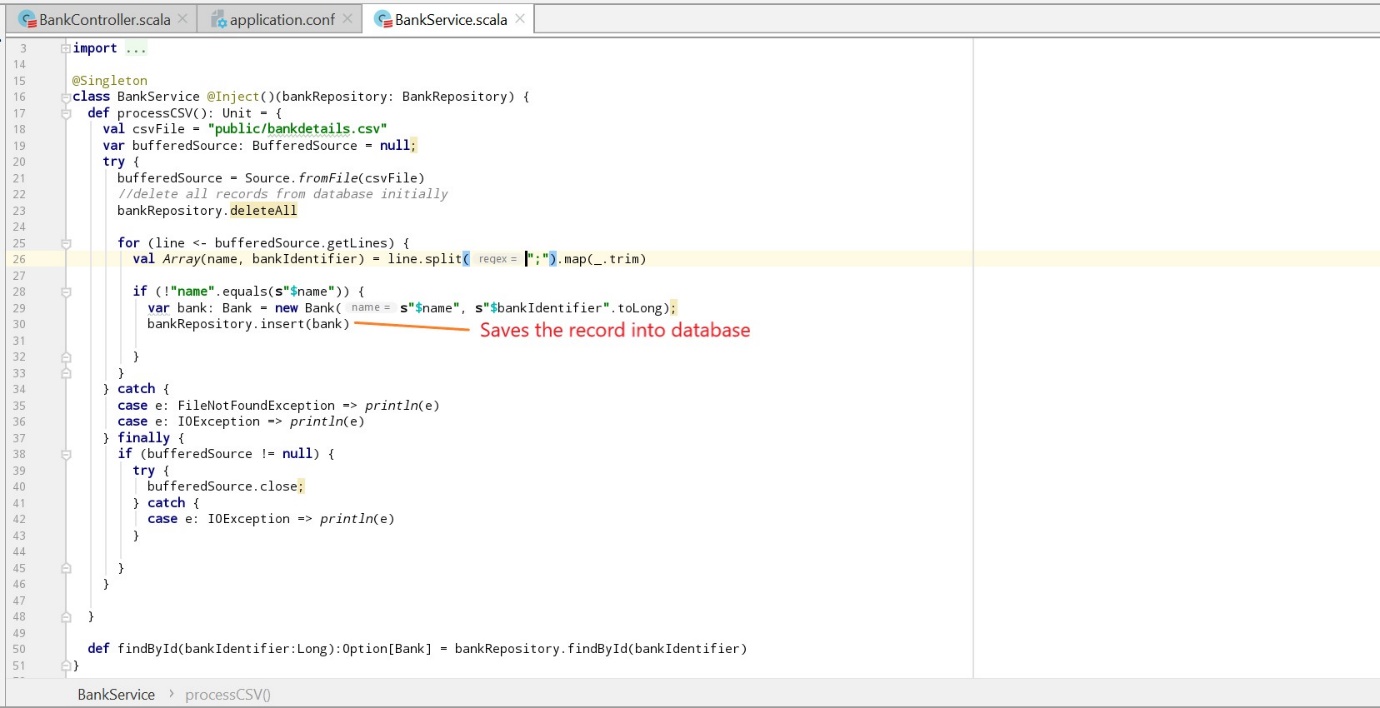
**Step 6:**

Created BankController, BankService and BankRepository along with Bank model class.

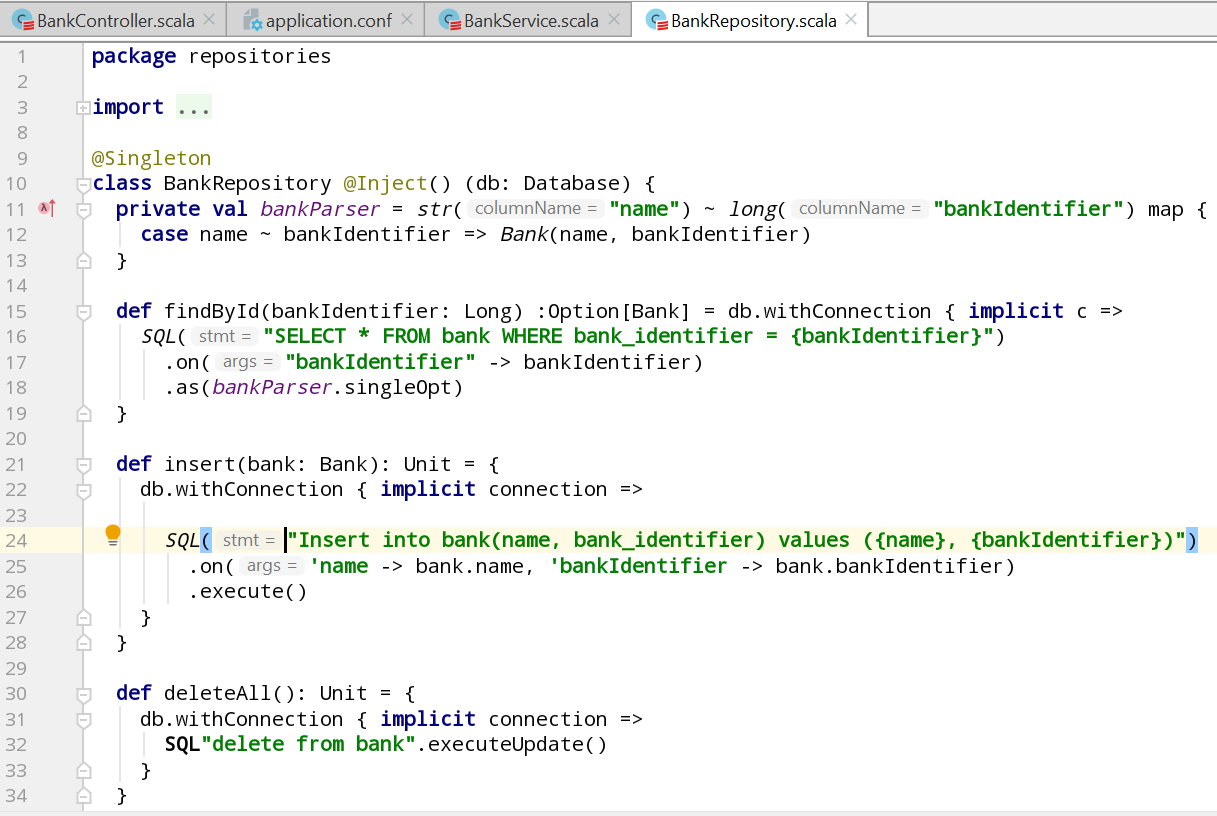
**BankController** – Receives the request from UI and delegates the request to service and result will be sent to the UI. BankService is injected into the BankController to do the business logic. Bankcontroller contains two methods one is for processing of CSV and another is for finding the bank information for given identifier. It is in **app/controller** package.



**BankService –** Receives the request from Controller and do the necessary business logic like read csv and build the model object to pass it to repository. From service class we call repository to do the database operations. It is in **app/services** package.

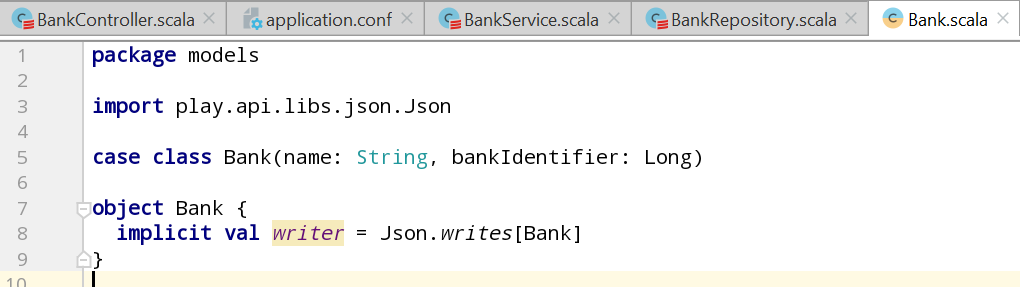


**BankRepository –** Receives the request from Service class and perform the database operations and send the result back based on the request. It is in **app/repositories** package.



**Bank Model:**

It is in **app/models** package.



**Step 7:**

Created the two views corresponding to two end points. Both will be in **app/views** package.

1. index.scala.html – to show the successful message on successful processing of csv.
2. bank.scala.html – to show the bank info based on identifier.

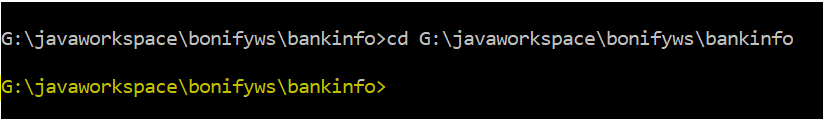
## **Source Code:**



## **Steps to run the application:**

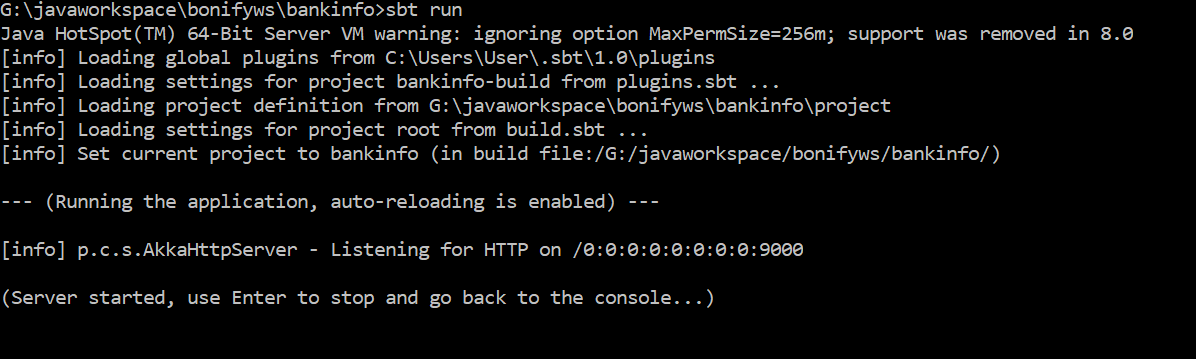
**Step 1:**

Make sure that all the prerequisites **(Java 1.8, SBT latest version, Postgress11.4.3 with bankinfo dabtase)** are on system. Unzip the source code and move to that directory from command prompt using cd command.



**Step 2:**

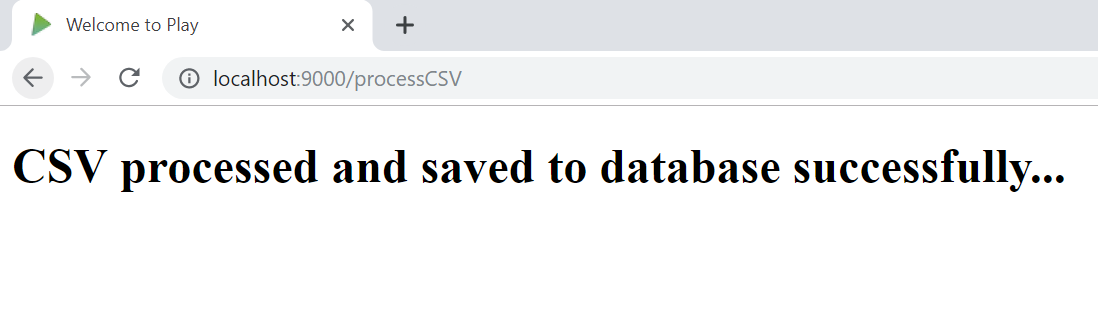
Run the **sbt run** command and wait for application to be started.

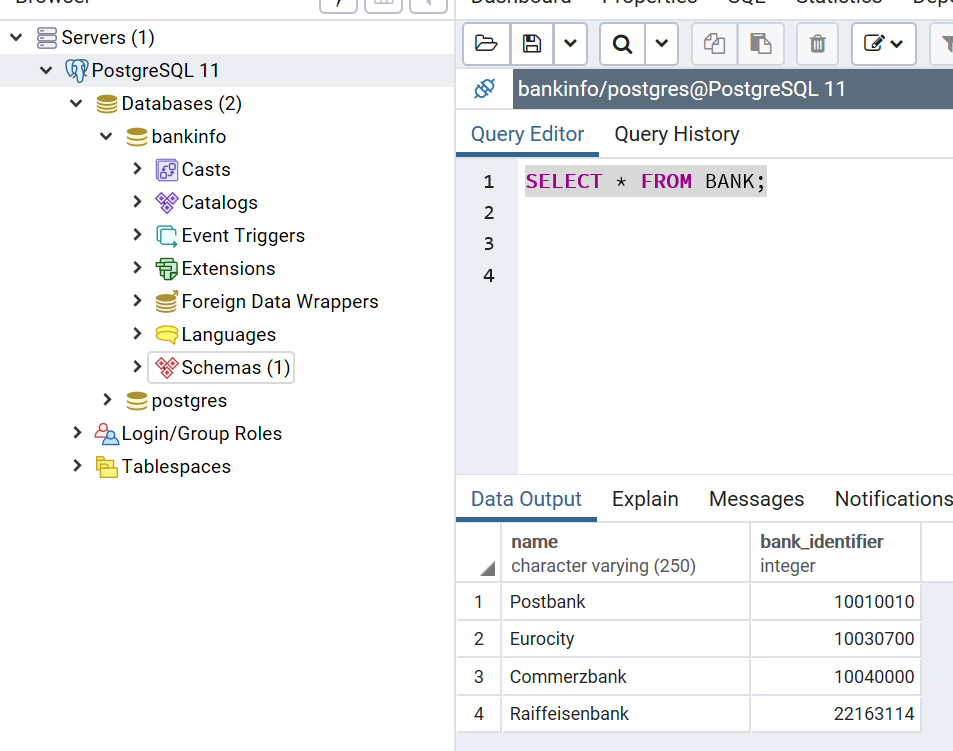


## **Testing the application:**

**Test for Processing of CSV:**

Hit the <http://localhost:9000/processCSV> from browser. We should be able to see the message on the web page saying **“CSV processed and saved to database successfully...”** And all records should be saved into the database.





**Test for Getting Bank Info based on Identifier:**

Hit the <http://localhost:9000/bank/10040000> from browser. We should be able to get the corresponding bank info from database in the json format. And bank information will be printed on the server console.

