Lab 13. Spring Boot

> Author: Yida Tao
>
> Part of the materials is based on https://amigoscode.com/p/spring-boot

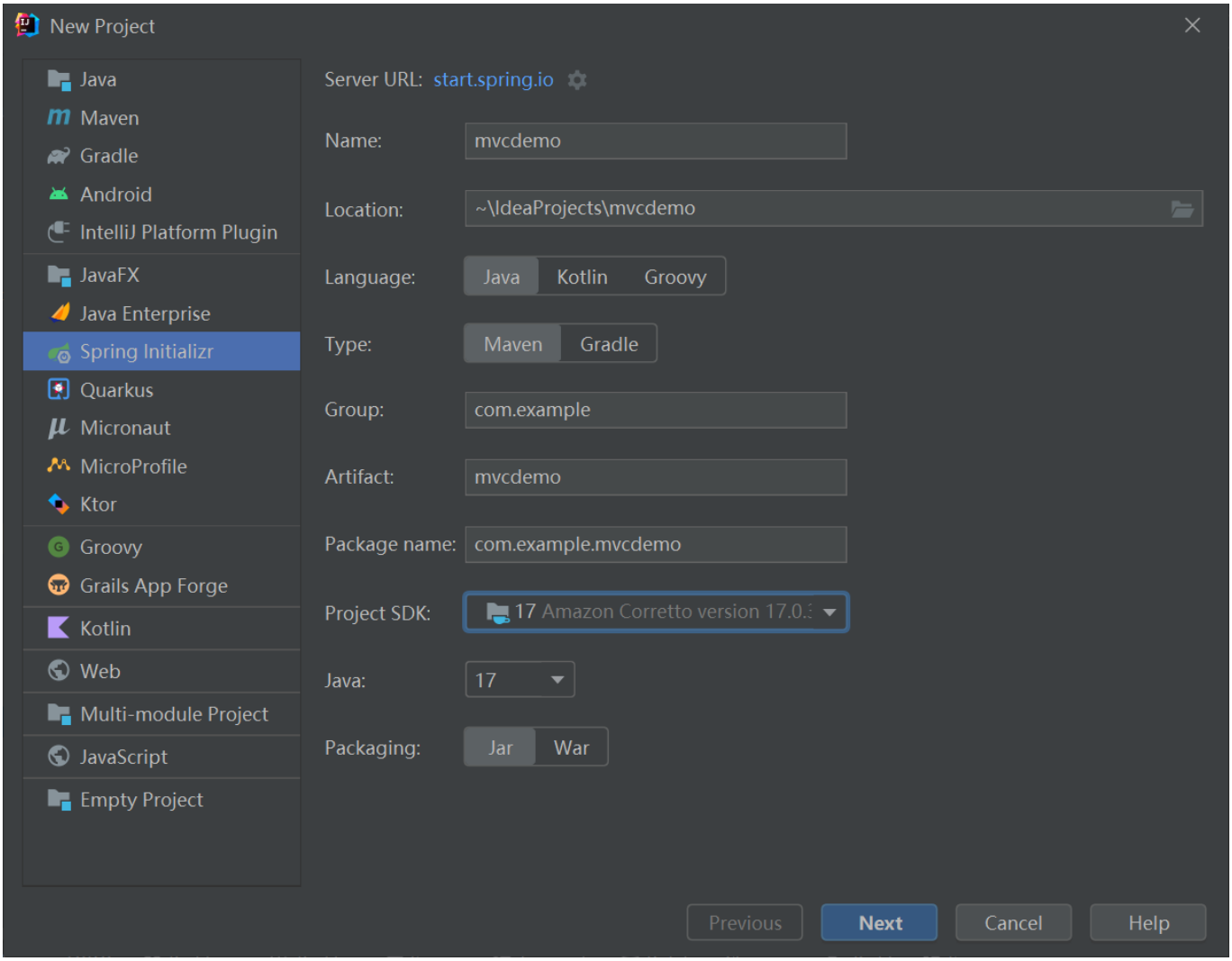
In this tutorial, we'll create a simple Spring Boot application, which could either be accessed by a browser or as a RESTful API.

Downloading IntelliJ Ultimate

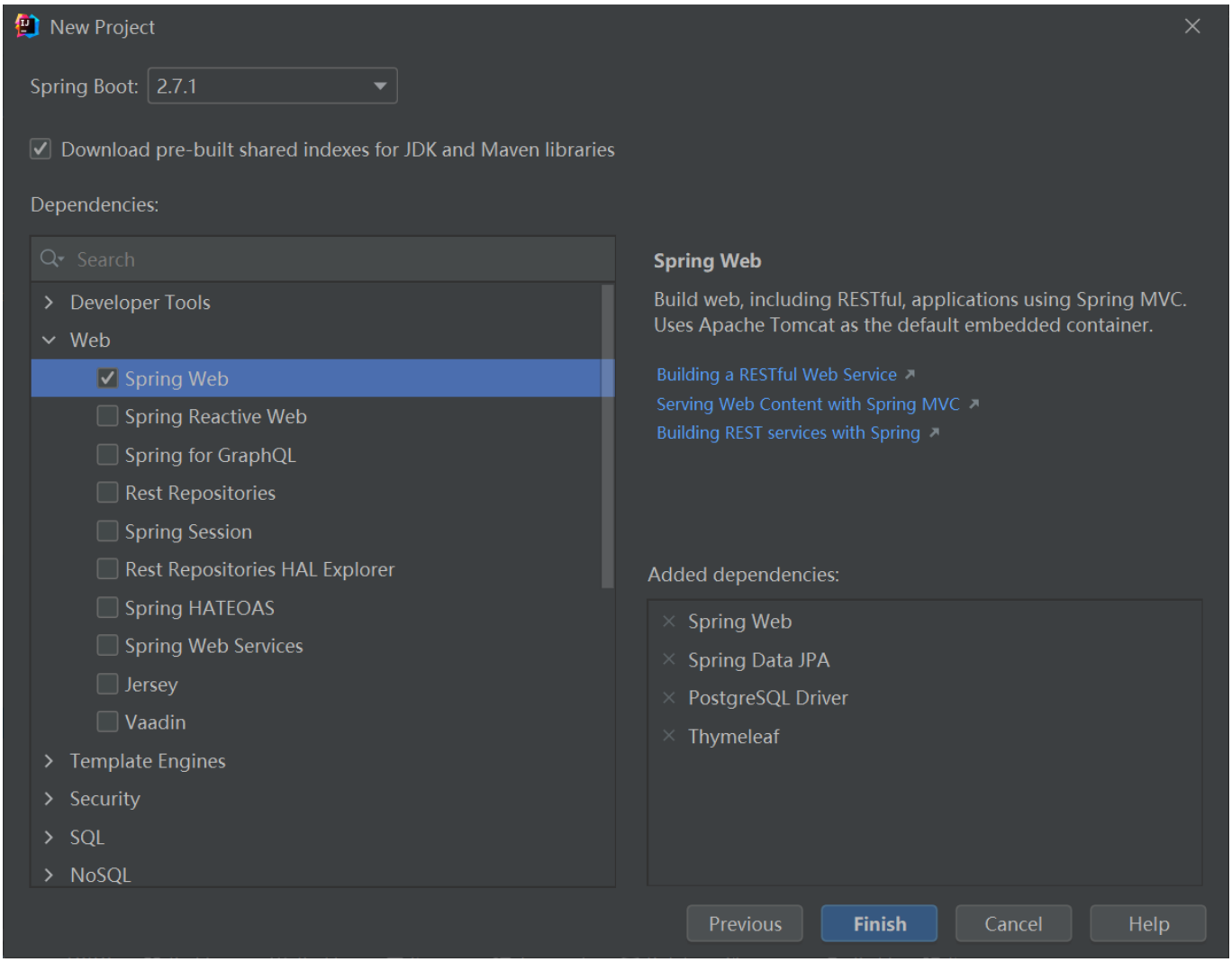
IntelliJ Ultimate makes developing Spring Boot applications easier. You may register a free educational license using your SUSTech email. See here for registration instructions.

Creating a New Spring Boot Project

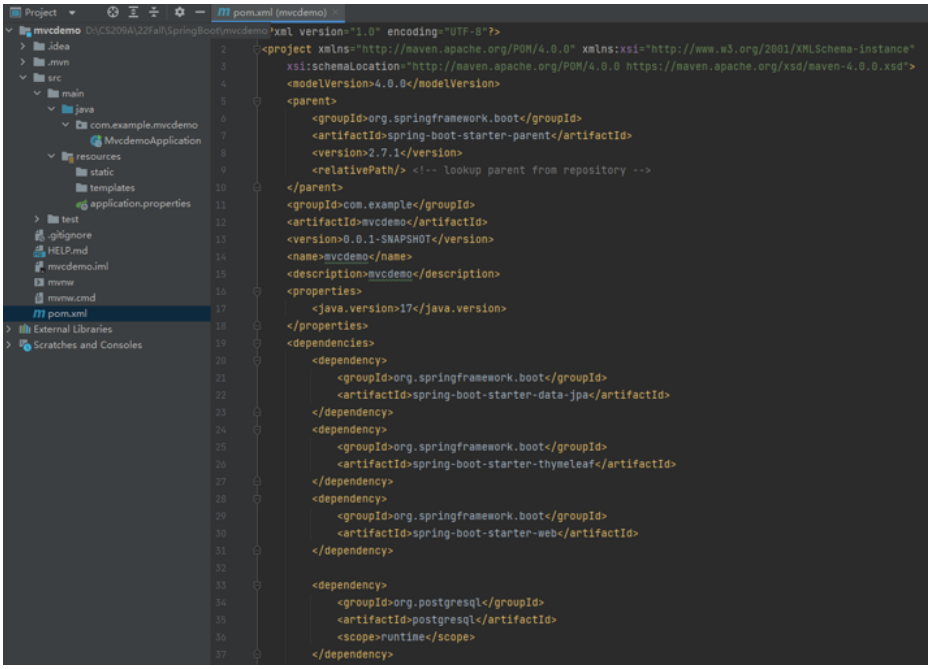
File -> New -> Project -> Spring Initializr, then specify project information like name and Java version and click "Next".



Then, we'll select necessary dependencies for this project. In this tutorial, we'll use Spring Web (for MVC and REST service), Spring Data JPA (for data persistence), PostgreSQL Driver (for connecting to a PostgreSQL database), and Thymeleaf (for working with HTML).

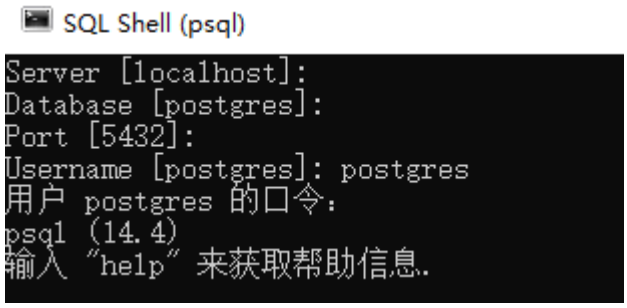


Click "Finish", and you'll see a project structure as follows. Maven will automatically download all the dependencies, which may take a while.



Since we're going to use the PostgreSQL database, we need to install this database as instructed here. During installation, you could set the username and password.

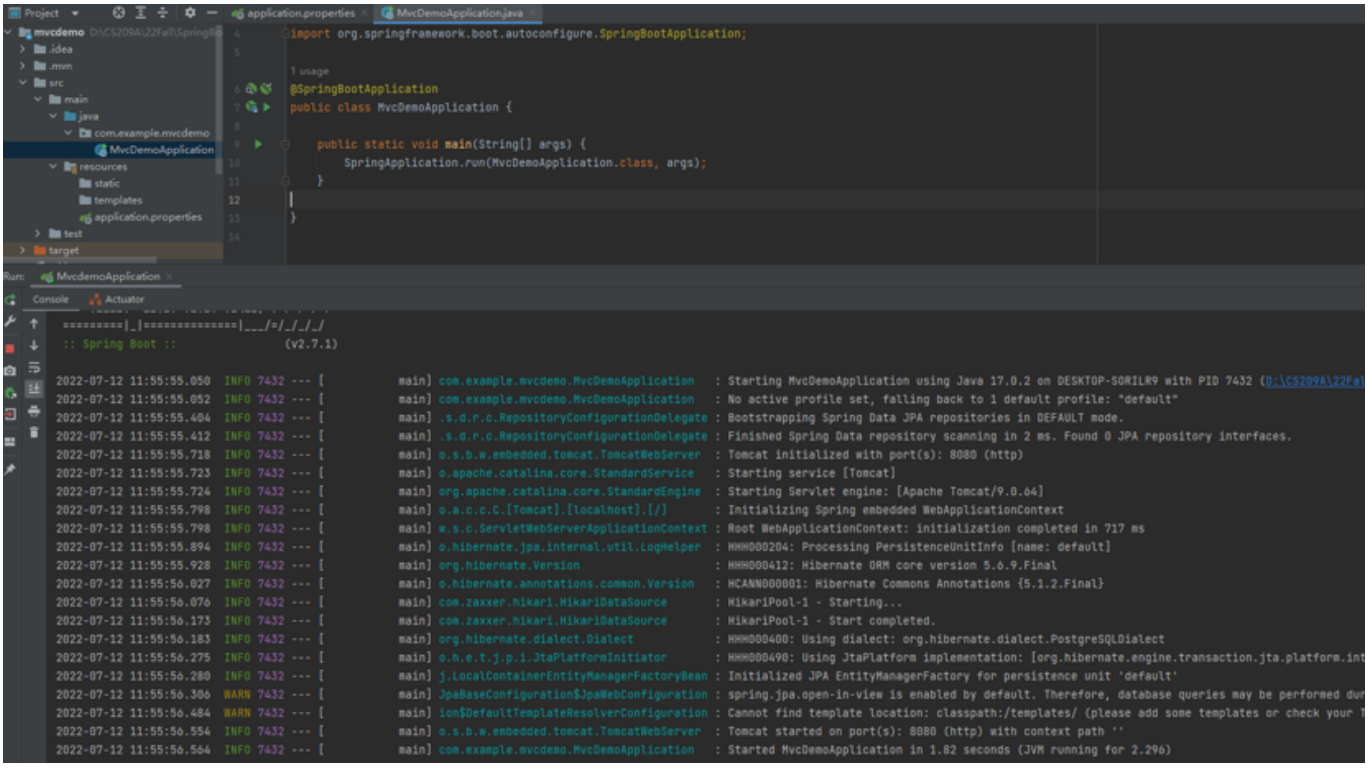
After installing PostgreSQL, you could open `SQL Shell (psql)` (windows). Enter all the necessary information such as the server, database, port, username, and password. To accept the default, you can simply press `Enter`. Note that you should provide the password that you entered during installing the PostgreSQL.



In SQL shell, type `CREATE DATABASE cs209a;` to create a database called "cs209a". Finally, add the following properties to `src/main/resources/application.properties` to configure the database to our Spring Boot project.

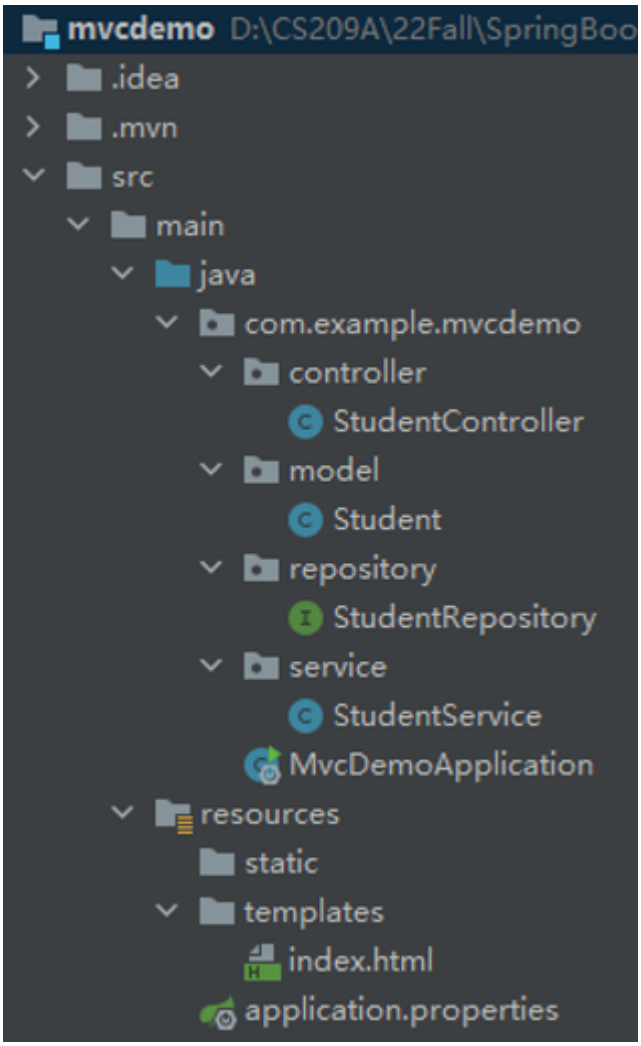
```
spring.datasource.url=jdbc:postgresql://localhost:5432/cs209a
spring.datasource.username=postgres
spring.datasource.password=123456
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.format_sql=true

server.error.include-message=always
```

Now, run `MvcDemoApplication.java`, and if you see the following content in your console, congratulations! You've successfully executed your first Spring Boot application.

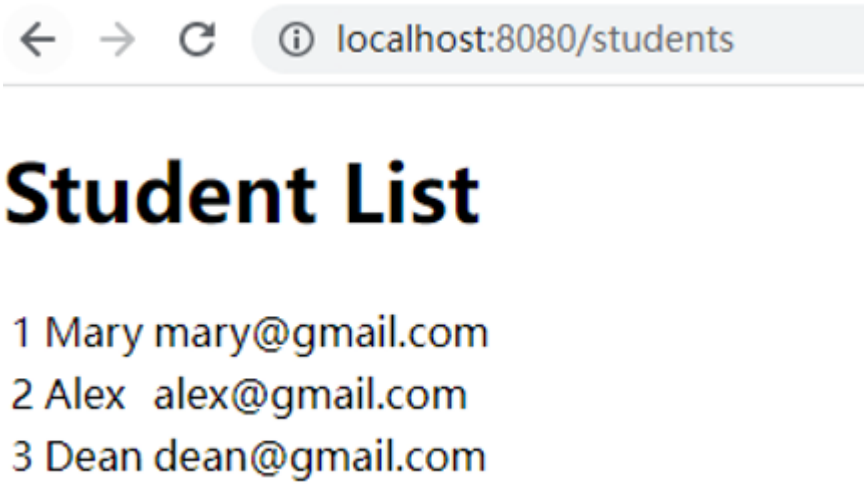


Creating a Simple MVC Application

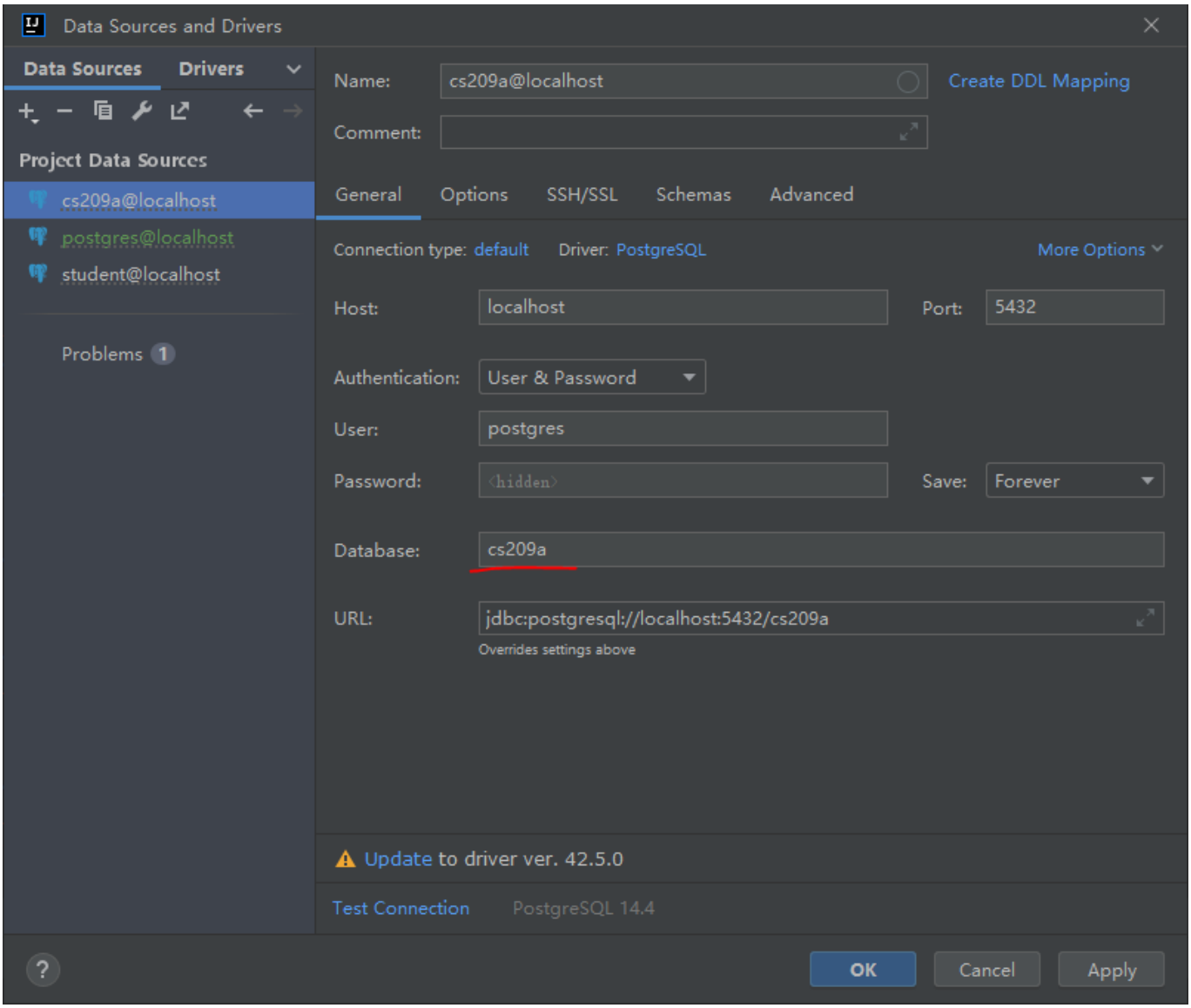
Let's create a simple MVC application, which has also been explained in our lectures. Please download `Student.java`, `StudentController.java`, `StudentRepository.java`, `StudentService.java` from Sakai and put them into proper packages.

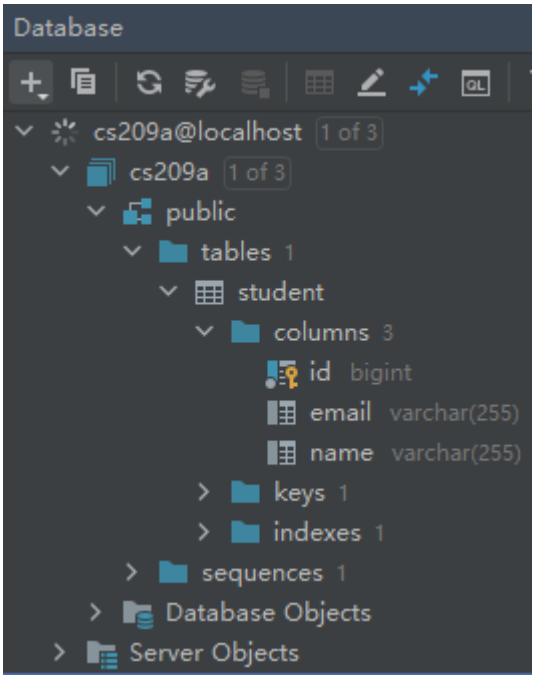


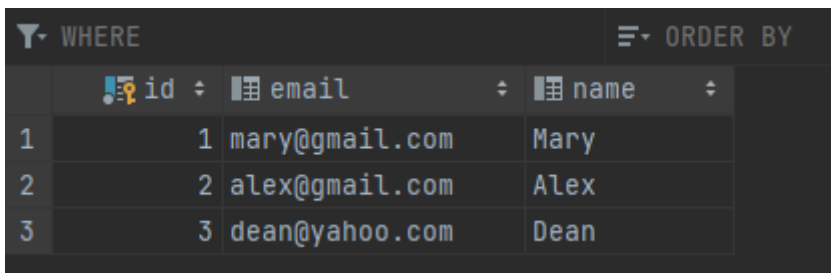
Then, start Spring Boot by executing `MvcDemoApplication.java`. Now open your browser and type `localhost:8080/students`. If you've done everything correctly, you should see the following table:



Using JPA features as well as the proper annotations such as `@Entity` and `@Table`, Spring Boot automatically creates and updates a `Student` table in `cs209a` database. You might also view the table inside of IntelliJ IDEA by clicking `Database` -> `+` -> `Data source` -> `PostgreSQL` and type in `cs209a` to connect to the database.

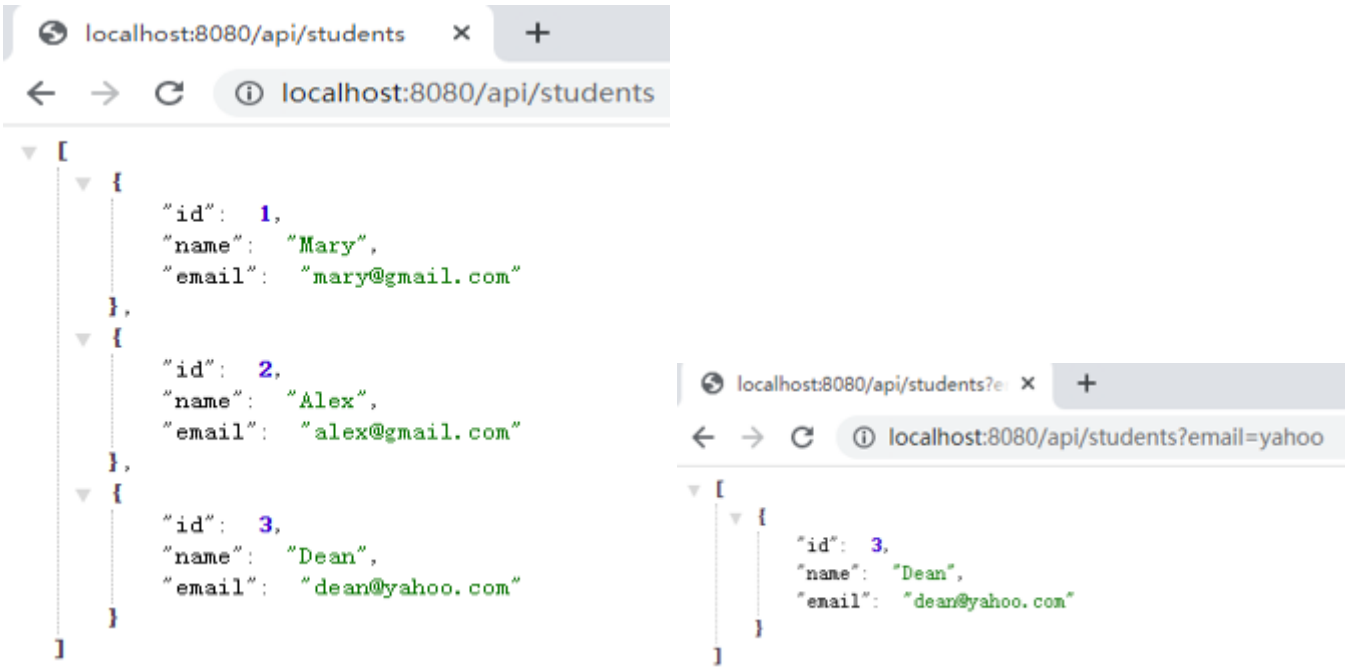




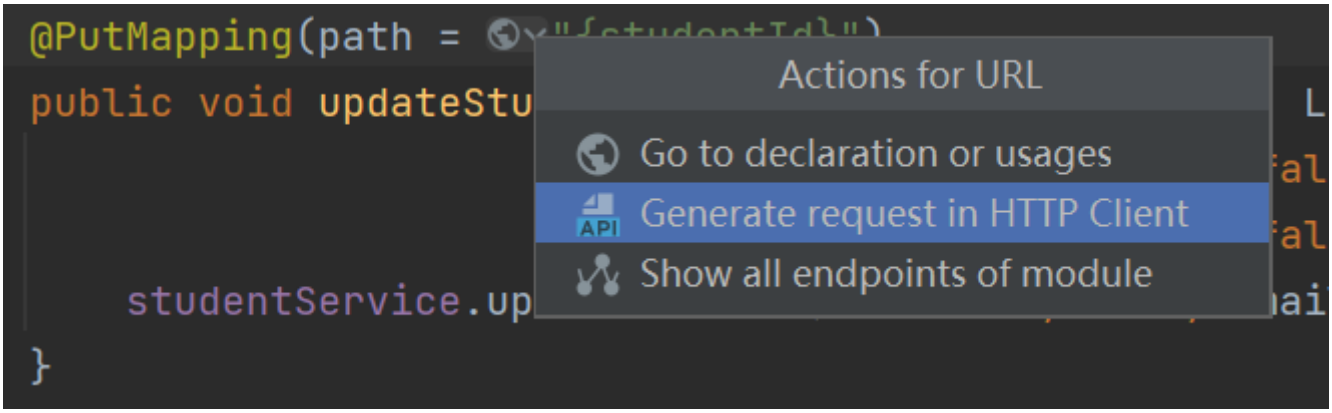


Creating a Simple RESTful Service

Let's also create a `StudentRestController.java` based on our lecture notes and put it in the `controller` package. Specifically, by implementing the `getStudentsByEmail` method, which maps to `/api/students`, you'll get the following `json` responses back.



By implementing the `updateStudent` method, you'll be able to execute a `PUT` request like `PUT http://localhost:8080/api/students/1?name=may&email=may@gmail.com` to update the information of a certain student. You could open the HTTP client (as follows) to execute the REST request.



Click the green button to execute the PUT method, and refresh the page in your browser to check the updated table.