JOHN ABBOTT
CEGEP/COLLEGE

# Spring Boot with MySQL Demo

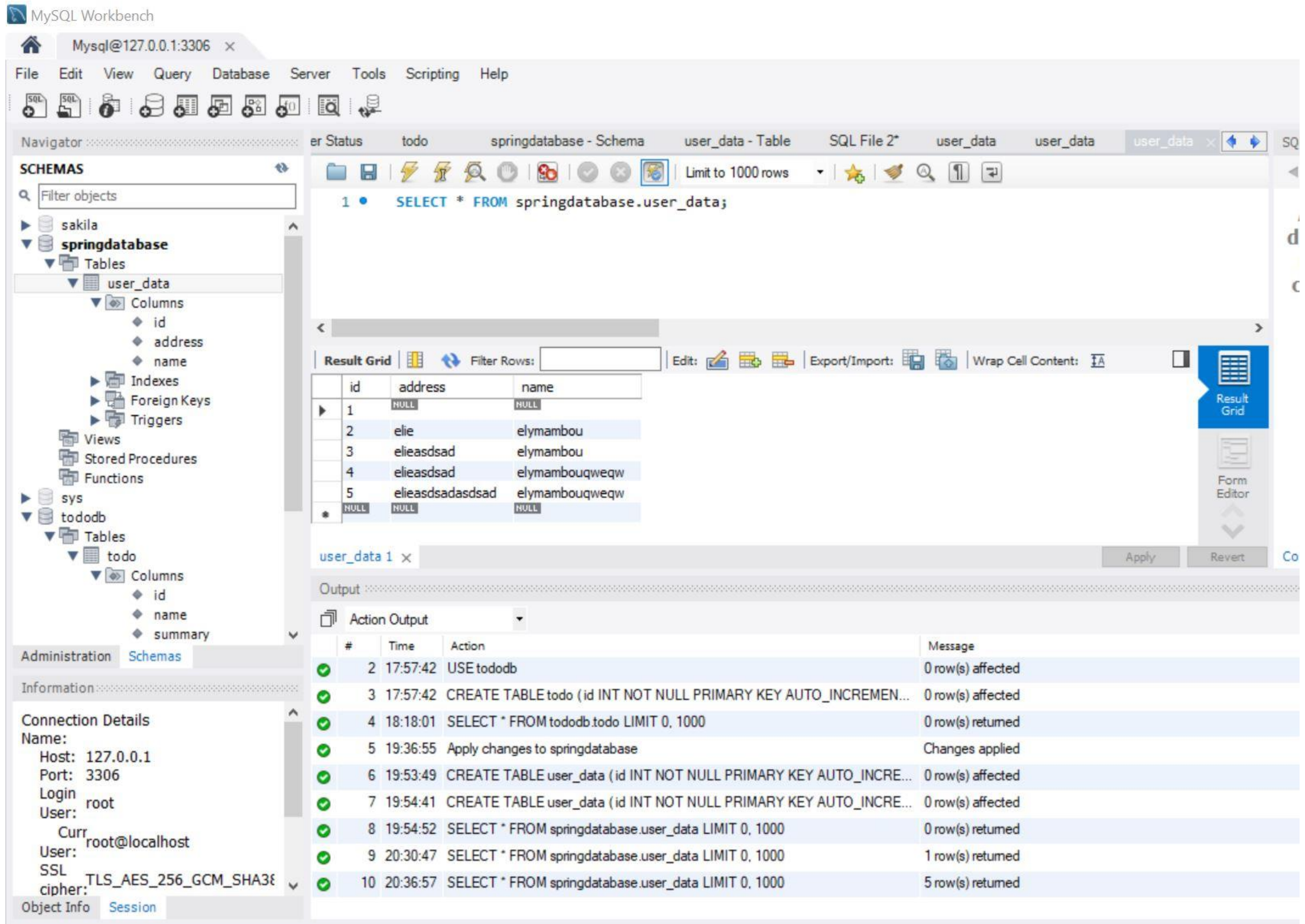
Elie Mambou, Ph.D.

(420-JD5-AB) Programming III

Summer 2022

# Install new things

- INSTALL MYSQL
  - [https://dev.mysql.com/downloads/installer/](https://dev.mysql.com/downloads/installer/)



```
SELECT * FROM springdatabase.user_data;
```

| id | address | name |
|---|---|---|
| 1 | NULL | NULL |
| 2 | elie | elymambou |
| 3 | elieasdsad | elymambou |
| 4 | elieasdsad | elymambouqweqw |
| 5 | elieasdsadasdsad | elymambouqweqw |
| NULL | NULL | NULL |

| # | Time | Action | Message |
|---|---|---|---|
| 2 | 17:57:42 | USE tododb | 0 row(s) affected |
| 3 | 17:57:42 | CREATE TABLE todo ( id INT NOT NULL PRIMARY KEY AUTO_INCREMEN... | 0 row(s) affected |
| 4 | 18:18:01 | SELECT * FROM tododb.todo LIMIT 0, 1000 | 0 row(s) returned |
| 5 | 19:36:55 | Apply changes to springdatabase | Changes applied |
| 6 | 19:53:49 | CREATE TABLE user_data ( id INT NOT NULL PRIMARY KEY AUTO_INCRE... | 0 row(s) affected |
| 7 | 19:54:41 | CREATE TABLE user_data ( id INT NOT NULL PRIMARY KEY AUTO_INCRE... | 0 row(s) affected |
| 8 | 19:54:52 | SELECT * FROM springdatabase.user_data LIMIT 0, 1000 | 0 row(s) returned |
| 9 | 20:30:47 | SELECT * FROM springdatabase.user_data LIMIT 0, 1000 | 1 row(s) returned |
| 10 | 20:36:57 | SELECT * FROM springdatabase.user_data LIMIT 0, 1000 | 5 row(s) returned |

Connection Details

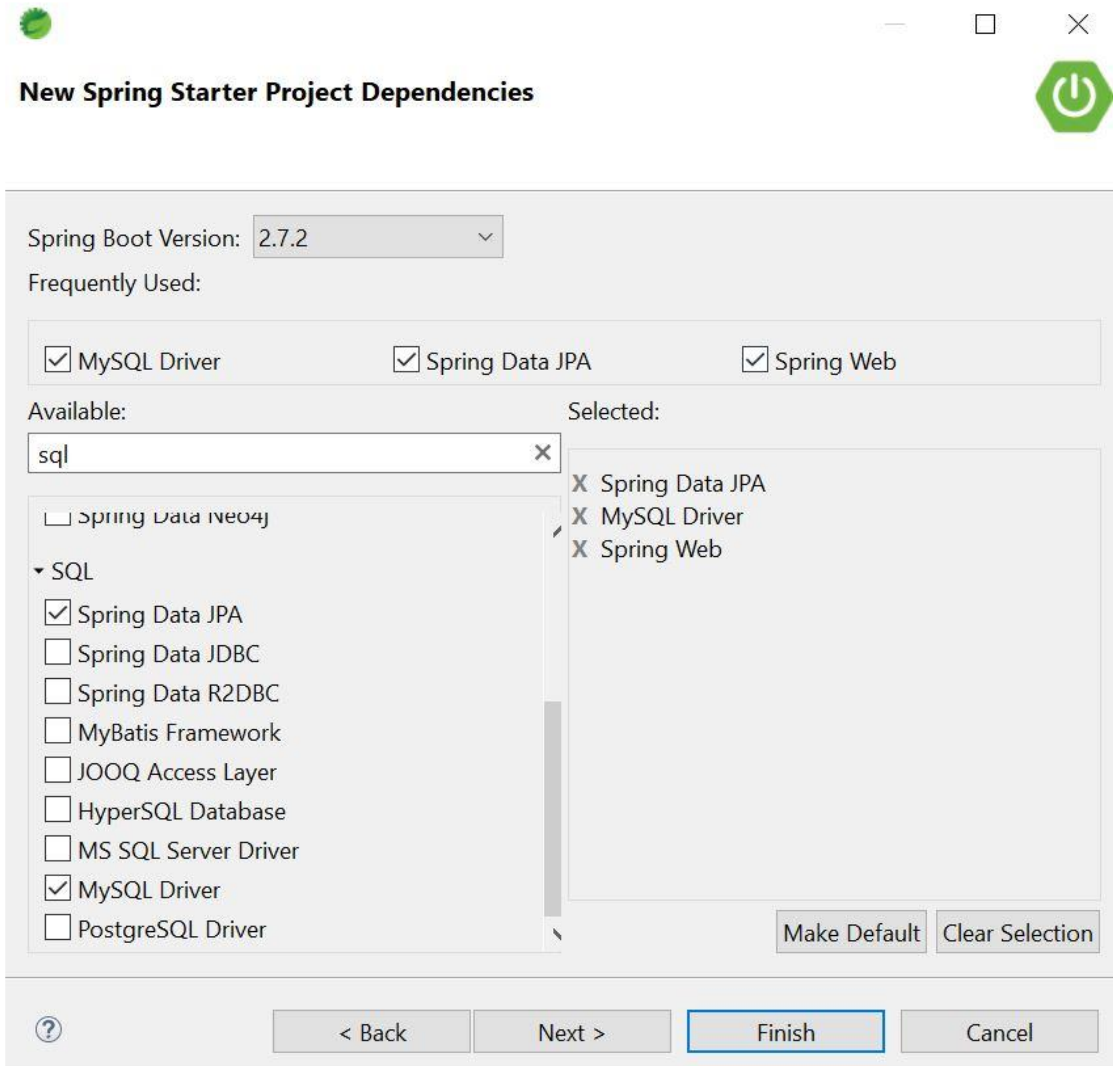
| | |
|---|---|
| Name: | |
| Host: | 127.0.0.1 |
| Port: | 3306 |
| Login User: | root |
| Curr User: | root@localhost |
| SSL cipher: | TLS_AES_256_GCM_SHA38 |

## Make a new spring boot project

JOHN ABBOTT CEGEP/COLLEGE

Name: ConnectDatabase

- Choose Java 11
- Maven Project
- Spring Web, Spring data JPA, and MySQL driver



New Spring Starter Project Dependencies

Spring Boot Version: 2.7.2

Frequently Used:

- [✓] MySQL Driver
- [✓] Spring Data JPA
- [✓] Spring Web

Available: sql

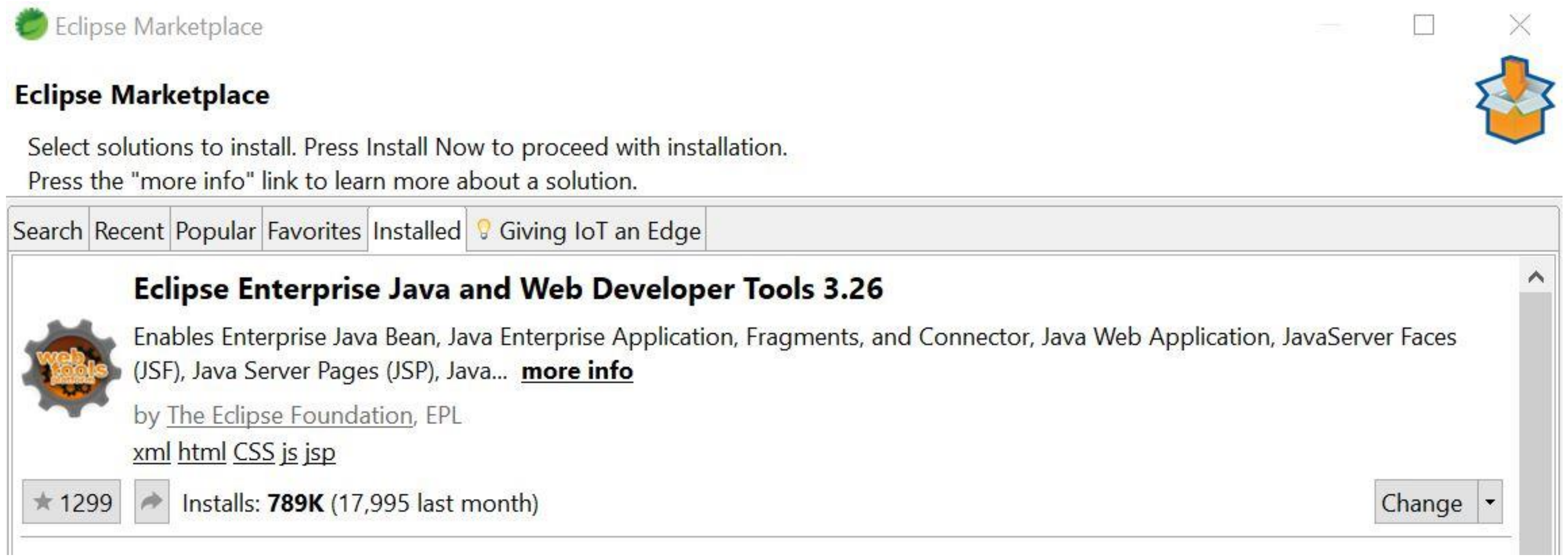
- [ ] Spring Data Neo4j
- SQL
  - [✓] Spring Data JPA
  - [ ] Spring Data JDBC
  - [ ] Spring Data R2DBC
  - [ ] MyBatis Framework
  - [ ] JOOQ Access Layer
  - [ ] HyperSQL Database
  - [ ] MS SQL Server Driver
  - [✓] MySQL Driver
  - [ ] PostgreSQL Driver

Selected:

- X Spring Data JPA
- X MySQL Driver
- X Spring Web

# Add package for file wizard

- Go to Help > Eclipse Marketplace
- Search and install Eclipse Enterprise Java and Web Developer Tools 3.26
- Reboot IDE



**Eclipse Marketplace**

Select solutions to install. Press Install Now to proceed with installation.
Press the "more info" link to learn more about a solution.

**Eclipse Enterprise Java and Web Developer Tools 3.26**

Enables Enterprise Java Bean, Java Enterprise Application, Fragments, and Connector, Java Web Application, JavaServer Faces (JSF), Java Server Pages (JSP), Java... **more info**

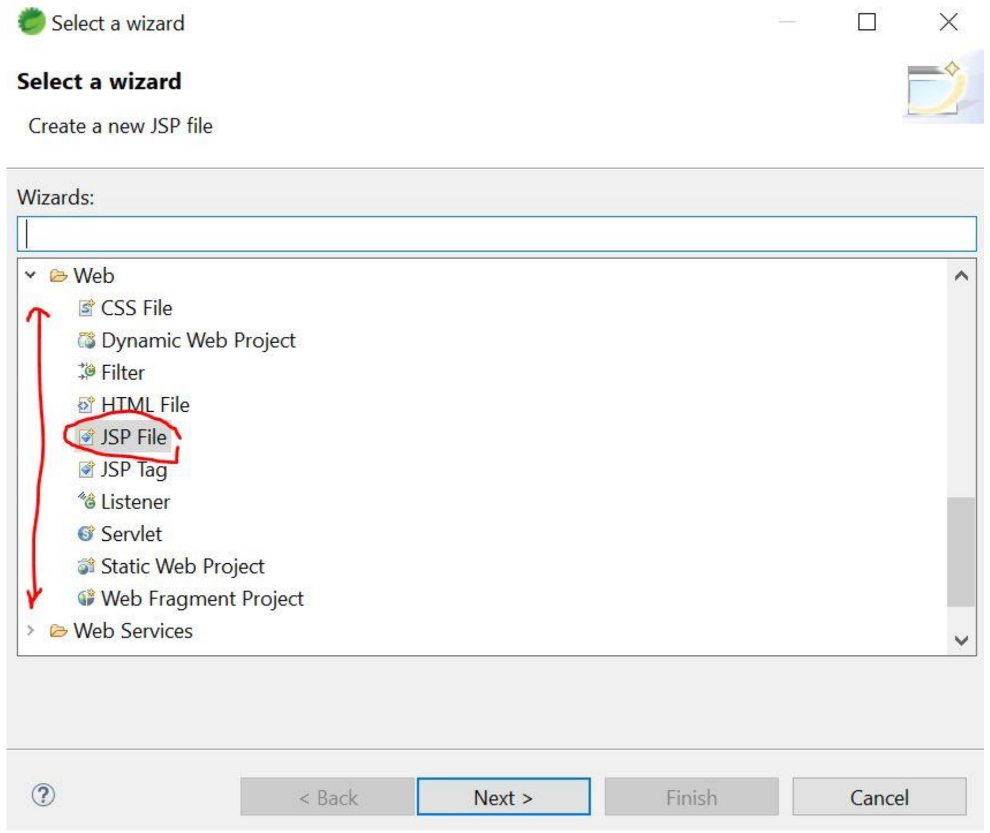
by The Eclipse Foundation, EPL

xml html CSS js jsp

Installs: **789K** (17,995 last month)

# Add index.jsp file

- Make a directory src/main/webapp/view/
- Right-click on view folder > new > other> Web> JSP File



```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="welcome" method="post" onsubmit="return validate()">
<label>Input name: </label>
<input type="text" name="name"><br>
<label>Input address: </label>
<input type="text" name="address"><br>
<input type="submit" value="Save">
</form>
</body>
</html>
```

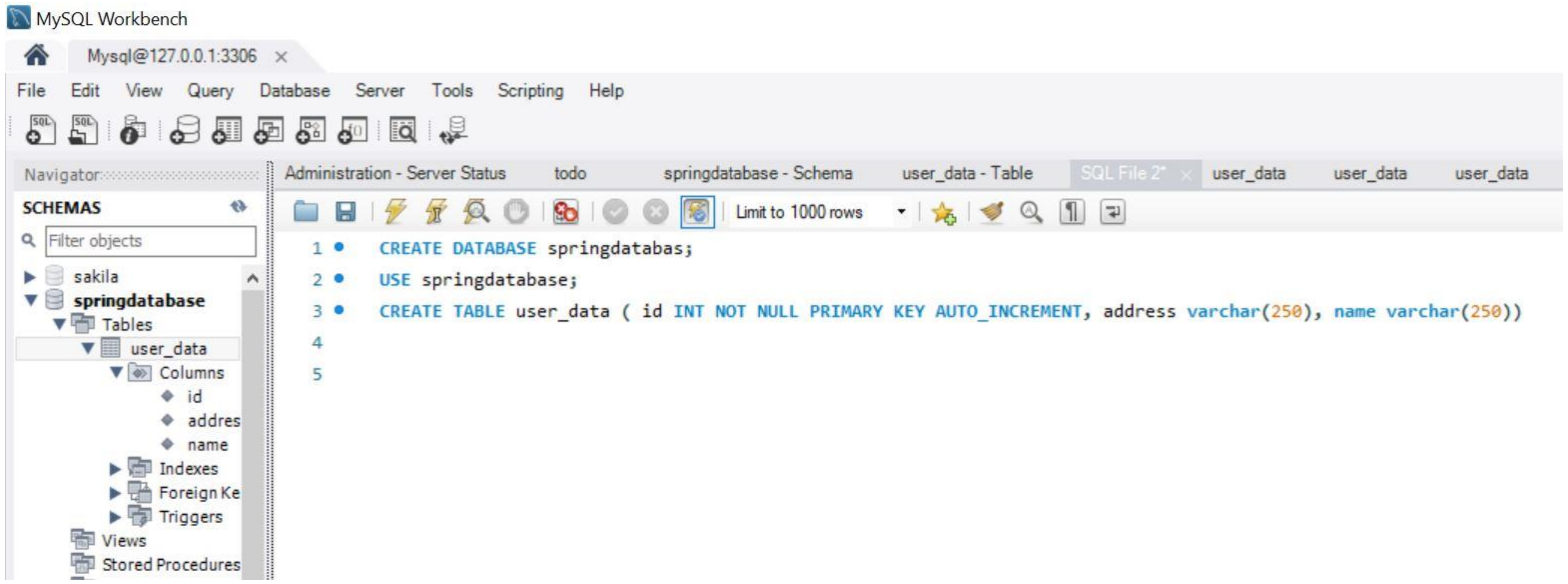
## Add new controller called DataController

- Under package com.data.controller

```
package com.data.controller;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;


@Controller
public class DataController {


@RequestMapping("/saveData")
public String indexPage() {

return "index";
}

   @PostMapping("/welcome")
   public String sayHello(@RequestParam("name") String name,
@RequestParam("address") String address, Model model) {
      model.addAttribute("name", name);
      model.addAttribute("address", address);
      return "welcome";
   }
}
```

# Add a new sql database

- Open MySQL workbench
    - Make a new sql and run



```sql
CREATE DATABASE springdatabas;
USE springdatabase;
CREATE TABLE user_data ( id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, address varchar(250), name varchar(250))
```

# Update application.properties and update pom.xml

- Under package src/main/resources

```
spring.mvc.view.prefix=/view/
spring.mvc.view.suffix=.jsp

spring.datasource.url=
jdbc:mysql://localhost:3306/springdatabase?autoR
econnect=true&useSSL=false
spring.datasource.username=root
spring.datasource.password=eliemambou

spring.jpa.hibernate.ddl-auto=update
server.port=8081
```

add this in pom.xml

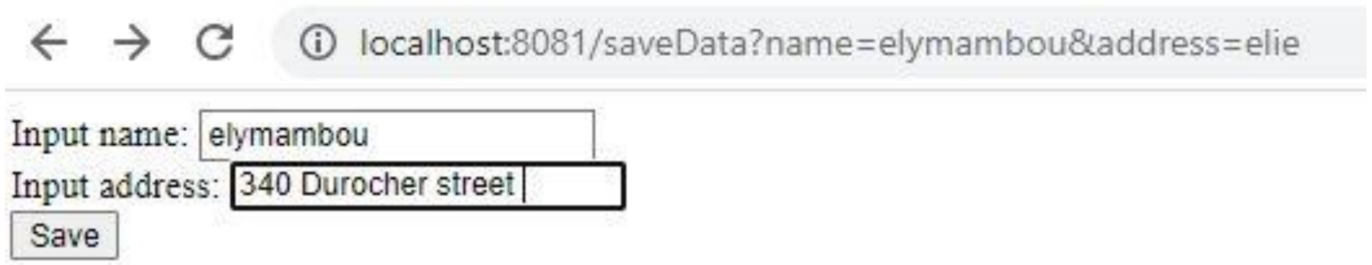
```
<!-- JSTL tag lib -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>jstl</artifactId>
    </dependency>

    <dependency>
      <groupId>taglibs</groupId>
      <artifactId>standard</artifactId>
      <version>1.1.2</version>
    </dependency>

    <!-- Tomcat for JSP rendering -->
    <dependency>
      <groupId>org.apache.tomcat.embed</groupId>
      <artifactId>tomcat-embed-jasper</artifactId>
      <scope>provided</scope>
    </dependency>
```

# Add welcome.jsp

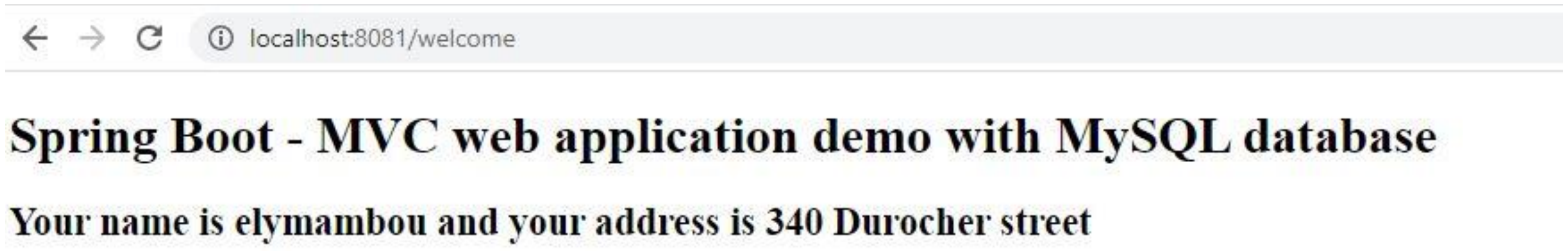
- Under src/main/webapp/view/

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>

  <h1>Spring Boot - MVC web application demo with MySQL database</h1>
  <hr>

  <h2>Your name is ${name} and your address is ${address} </h2>


</body>
</html>
```

## This is where we stopped last time!! Now let us add a model


localhost:8081/saveData?name=elymambou&address=elie

Input name: elymambou
Input address: 340 Durocher street
Save


localhost:8081/welcome

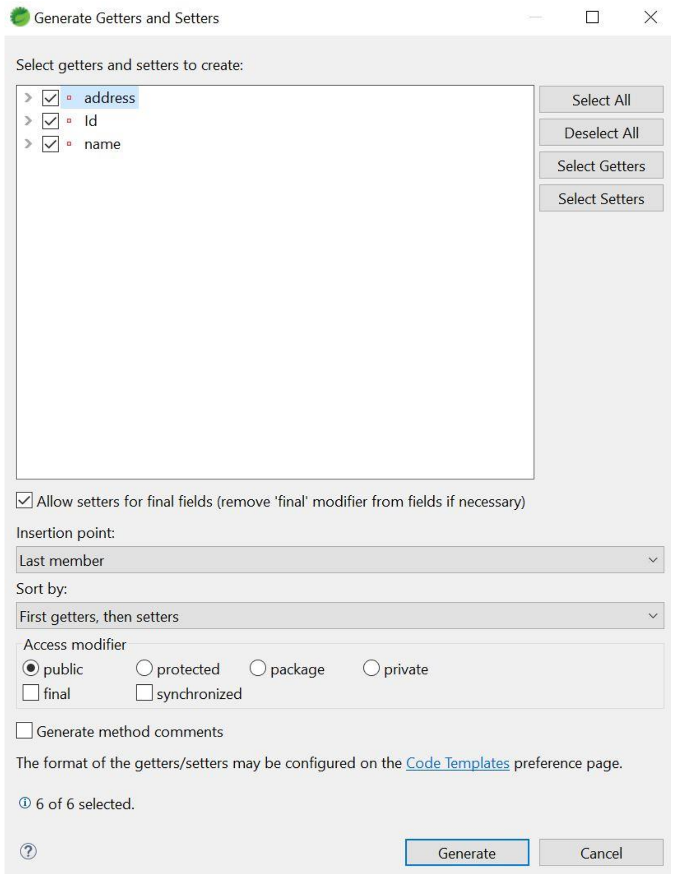
**Spring Boot - MVC web application demo with MySQL database**

**Your name is elymambou and your address is 340 Durocher street**

# Add a model under package com.data.model

- Under package src/main/java
- Make a class UserData
- Select all attributes then do Alt+Shift+s, then generate setters and getters
- Then add a constructor

```
package com.data.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;



@Entity
public class UserData {

@Id
@GeneratedValue(strategy= GenerationType.AUTO)
private int Id;
private String name;
private String address;



}
```



Generate Getters and Setters

Select getters and setters to create:

- ☑ address
- ☑ Id
- ☑ name

☑ Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: Last member

Sort by: First getters, then setters

Access modifier: ● public ○ protected ○ package ○ private
☐ final ☐ synchronized

☐ Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

6 of 6 selected.

```
package com.data.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;



@Entity
public class UserData {

@Id
@GeneratedValue(strategy= GenerationType.AUTO)
private int Id;
private String name;
private String address;
public int getId() {
return Id;
}
public String getName() {
return name;
}
public String getAddress() {
return address;
}
public void setId(int id) {
Id = id;
}
public void setName(String name) {
this.name = name;
}
public void setAddress(String address) {
this.address = address;
}

public UserData()
{
super();
this.name=name;
this.address=address;
} }
```

# Add a userDataRepo interface

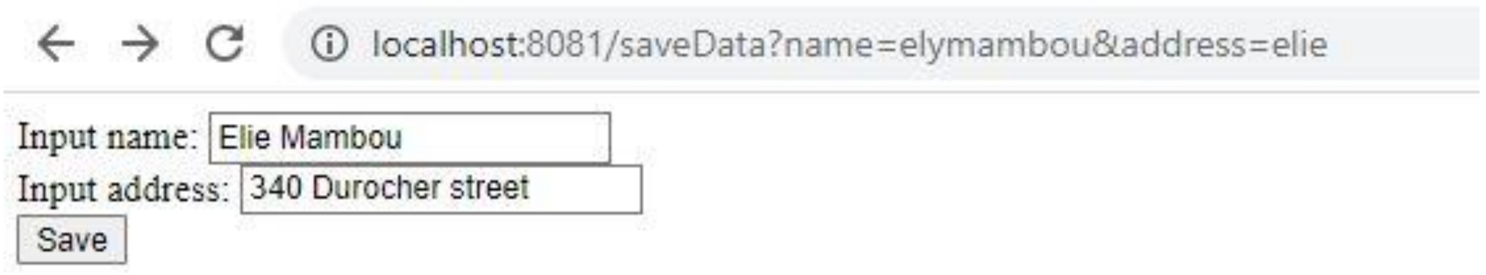
- Under package src/main/java/com.data.dao

```
package com.data.dao;

import org.springframework.data.jpa.repository.JpaRepository;

import com.data.model.UserData;

public interface UserDataRepo extends JpaRepository<UserData, Integer>{

}
```

## Update the controller (MVC magic!)

```
package com.data.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

import com.data.dao.UserDataRepo;
import com.data.model.UserData;

@Controller
public class DataController {

@Autowired
UserDataRepo repo;

@RequestMapping("/saveData")
public String indexPage() {

return "index";
}

   @RequestMapping("/welcome")
   @ResponseBody
   public String sayHello(UserData userData) {
      repo.save(userData);
      return "Welcome "+userData.getName()+" and you stay at "+userData.getAddress();
   }
}
```

# RUN the application on plain view @RequestMapping and @ResponseBody

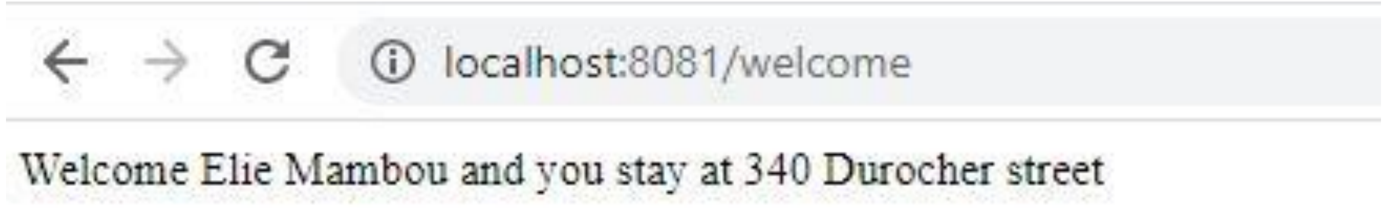


localhost:8081/saveData?name=elymambou&address=elie

Input name: Elie Mambou

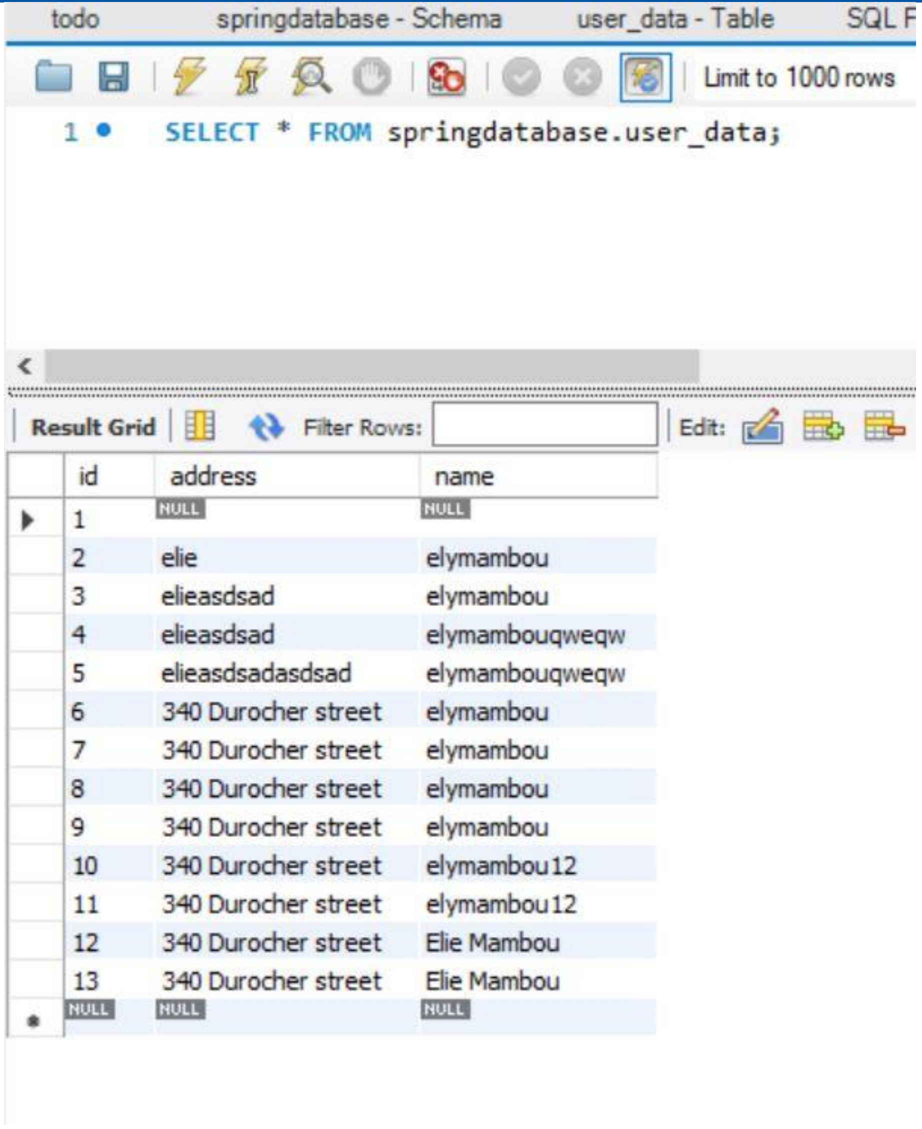
Input address: 340 Durocher street

Save



localhost:8081/welcome

Welcome Elie Mambou and you stay at 340 Durocher street



todo | springdatabase - Schema | user_data - Table | SQL F

Limit to 1000 rows

```
SELECT * FROM springdatabase.user_data;
```

Result Grid | Filter Rows: | Edit:

| id | address | name |
|---|---|---|
| 1 | NULL | NULL |
| 2 | elie | elymambou |
| 3 | elieasdsad | elymambou |
| 4 | elieasdsad | elymambouqweqw |
| 5 | elieasdsadasdsad | elymambouqweqw |
| 6 | 340 Durocher street | elymambou |
| 7 | 340 Durocher street | elymambou |
| 8 | 340 Durocher street | elymambou |
| 9 | 340 Durocher street | elymambou |
| 10 | 340 Durocher street | elymambou12 |
| 11 | 340 Durocher street | elymambou12 |
| 12 | 340 Durocher street | Elie Mambou |
| 13 | 340 Durocher street | Elie Mambou |
| NULL | NULL | NULL |

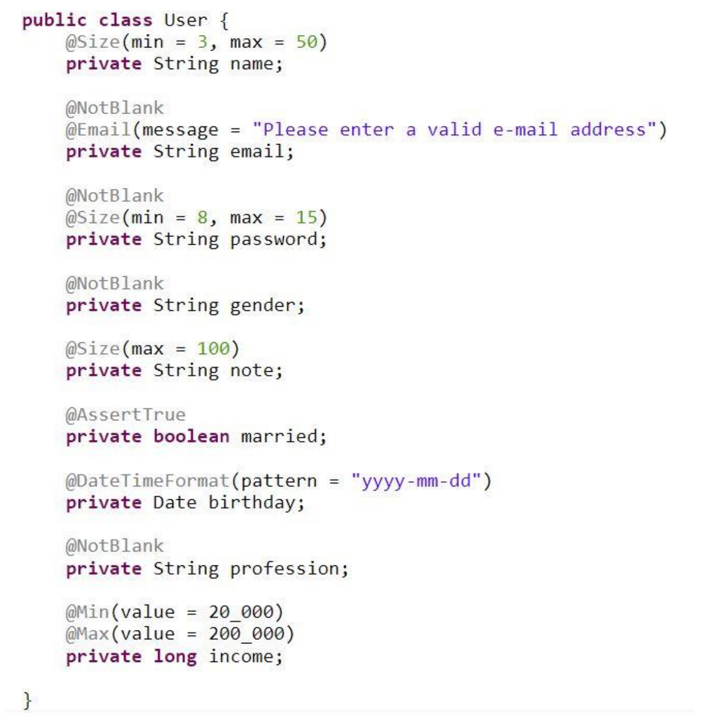
# Common Validation Annotations

JOHN ABBOTT CEGEP/COLLEGE

- **@NotEmpty:** to say that a list field must not empty.
- **@NotBlank:** to say that a string field must not be the empty string (i.e. it must have at least one character).
- **@Min and @Max:** to say that a numerical field is only valid when it's value is above or below a certain value.
- **@Pattern:** to say that a string field is only valid when it matches a certain regular expression.
- **@NotNull:** to say that a field must not be null.
- **@Email:** to say that a string field must be a valid email address.

https://www.codejava.net/frameworks/spring-boot/spring-boot-form-validation-tutorial

https://javaee.github.io/javaee-spec/javadocs/javax/validation/constraints/package-summary.html



```
public class User {
    @Size(min = 3, max = 50)
    private String name;

    @NotBlank
    @Email(message = "Please enter a valid e-mail address")
    private String email;

    @NotBlank
    @Size(min = 8, max = 15)
    private String password;

    @NotBlank
    private String gender;

    @Size(max = 100)
    private String note;

    @AssertTrue
    private boolean married;

    @DateTimeFormat(pattern = "yyyy-mm-dd")
    private Date birthday;

    @NotBlank
    private String profession;

    @Min(value = 20_000)
    @Max(value = 200_000)
    private long income;

}
```

# Q & A

JOHN ABBOTT
CEGEP/COLLEGE