**Hands On Lab 1**

# Spring Boot with H2 Database

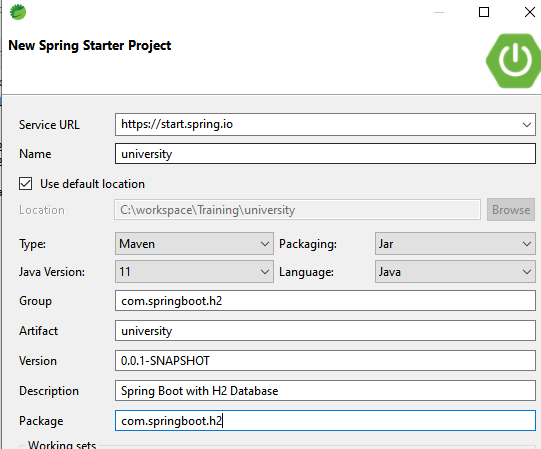
### **Introduction:**

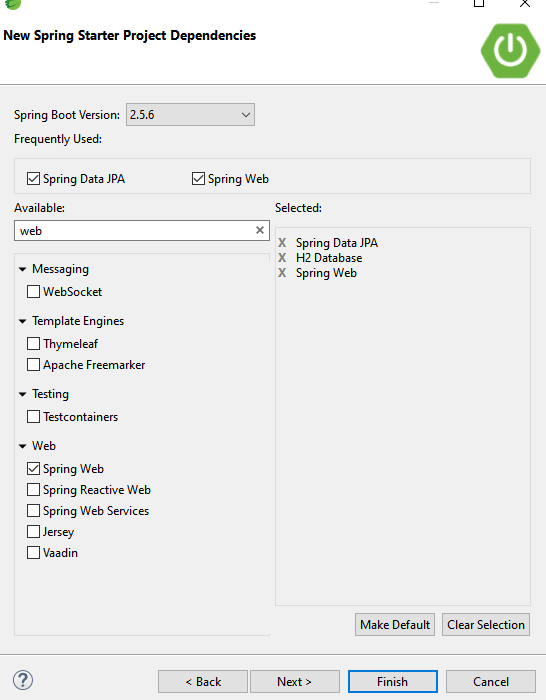
In this quick lab, we’ll bootstrap a simple Spring Boot application backed by an in-memory H2 database. We’ll use the Spring Data JPA to interact with our database.

### **Project Setup:**

Firstly, let’s generate our project template using [Spring Initializr:](https://start.spring.io/)

Alternatively, use the Spring tool Suite (STS) inbuilt tool to generate your project.





### **H2 Default Properties:**

Since we have added H2 database dependency, Spring Boot will auto-configure its related properties. The default configurations include:

spring.datasource.url=jdbc:h2:mem:testdb

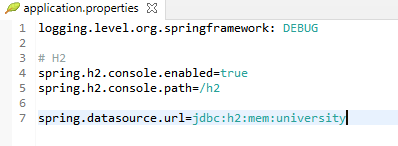
spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.h2.console.enabled=false

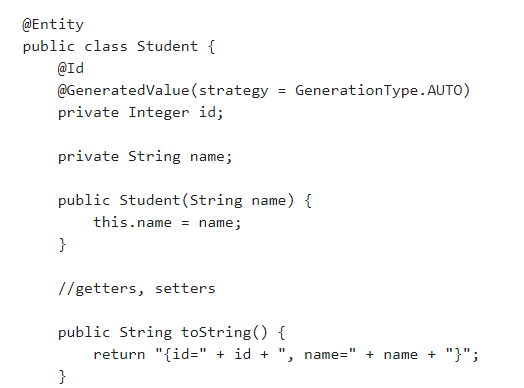
Let’s override a few of these properties by defining those in our application.properties file:



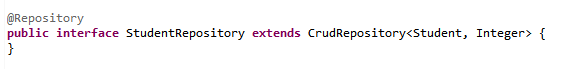
Here, our database name will be a university. We have also enabled the H2 Console and have set its context path.

### **Defining Entities:**

Now, we’ll define a Student entity:



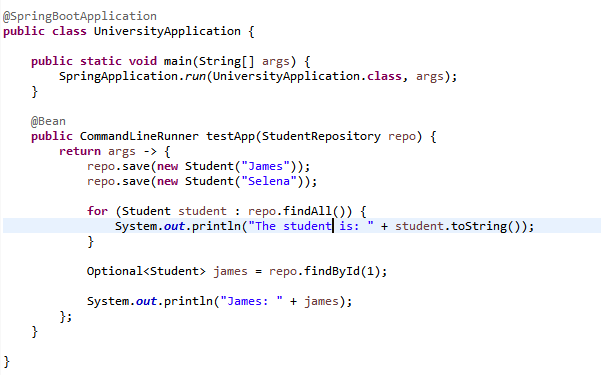
And its corresponding Spring Data JPA repository:



The Student entity will get mapped to the database table with exactly the same name. If we want, we can specify a different table name with @Table annotation.

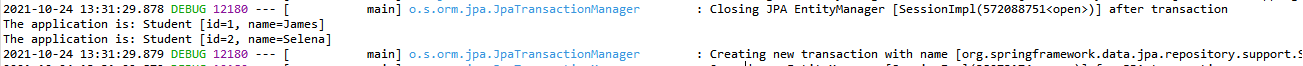
### **The Application Class:**

Finally, let’s implement our UniversityApplication class:



This class is the starting point for our Spring Boot application. Here, the **@SpringBootApplication**annotation is equivalent to having**@ComponentScan, @EnableAutoConfiguration,**and**@SpringConfiguration**together.****

We have also defined an instance of [CommandLineRunner](https://www.programmergirl.com/commandlinerunner-applicationrunner-spring-boot/). And so, when we run our application, our console logs will have:



Please note that in Spring Boot,****all entities should ideally be defined at either the same package level or at lower levels(in sub-packages) to that of the main application class****. If so, Spring Boot will auto-scan all those entities.

### **Accessing the H2 Console:**

We can also check the database entries on the H2 console.

To do that, we’ll open the URL: http://localhost:8080/h2 on any browser and use our database configurations to login. With it, we’ll be able to see all our created tables and entries easily on a UI Console dashboard.

### **Conclusion:**

In this lab, we bootstrapped a very simple Spring Boot application with a single entity. This application is integrated with the H2 database and uses Spring Data JPA.  
We can easily extend it to cater to a much wider application scope.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*