

# Lab JPA: Spring Data

## Objectives

In this lab, you will use Spring Data to implement a basic entity type.

## Instructions

### Step 1 Create the Spring Boot project

1. Go to <https://start.spring.io>.
2. We will be using the Spring data started and use an in-memory database called H2

Project	Language	Dependencies
<input checked="" type="radio"/> Maven Project	<input checked="" type="radio"/> Java	<b>H2 Database</b> <span>SQL</span> Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.
<input type="radio"/> Gradle Project	<input type="radio"/> Kotlin	<b>Spring Data JPA</b> <span>SQL</span> Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.
	<input type="radio"/> Groovy	
<b>Spring Boot</b>		
<input type="radio"/> 3.0.0 (SNAPSHOT)	<input type="radio"/> 3.0.0 (M1)	
<input type="radio"/> 2.7.0 (SNAPSHOT)	<input type="radio"/> 2.7.0 (M2)	
<input type="radio"/> 2.6.5 (SNAPSHOT)	<input checked="" type="radio"/> 2.6.4	
<input type="radio"/> 2.5.11 (SNAPSHOT)	<input type="radio"/> 2.5.10	
<b>Project Metadata</b>		
Group	com.example	
Artifact	data	
Name	data	
Description	Data Project for Spring Boot	
Package name	com.example.data	

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...

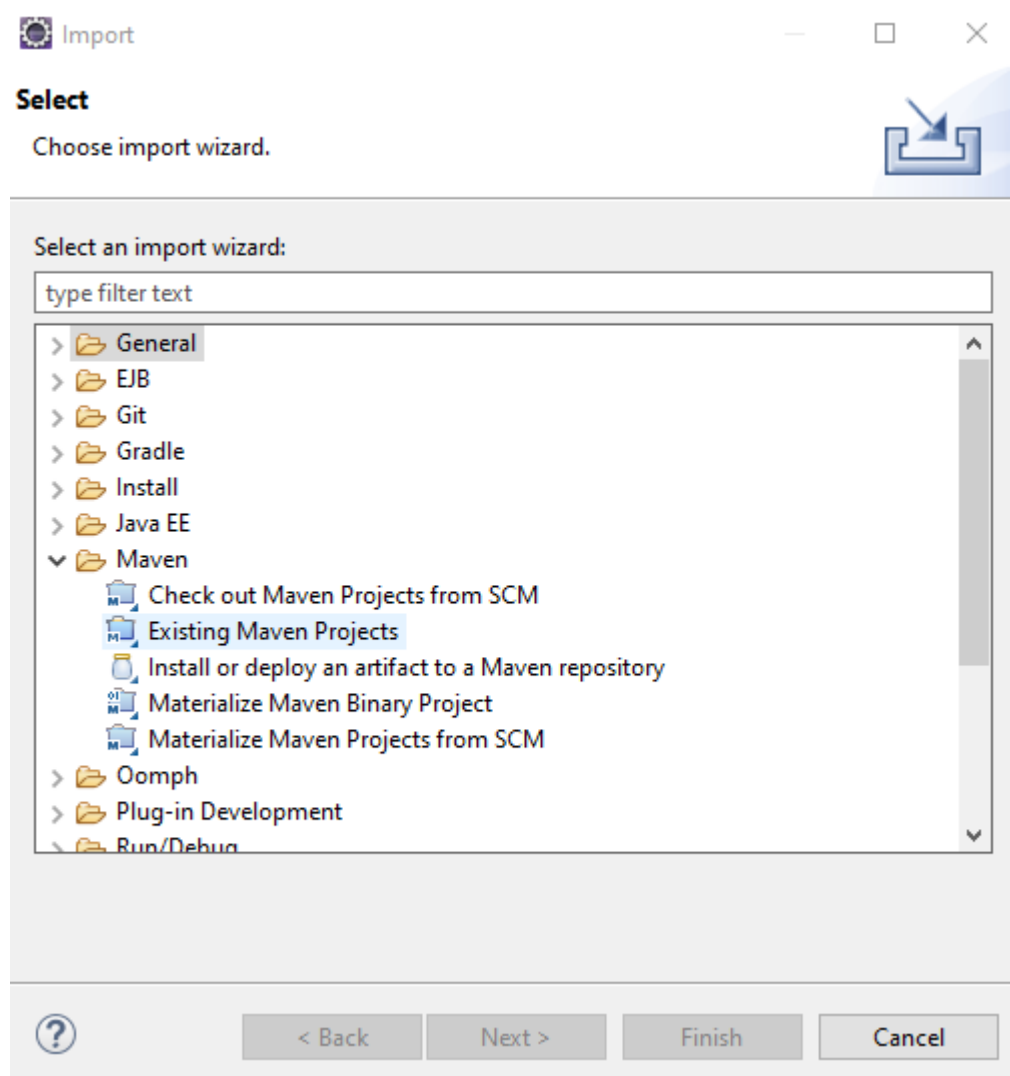
## **Step 2: Ensure Maven is installed**

1. Check to see Maven is installed by opening a command window and executing "mvn -version. You should see the following

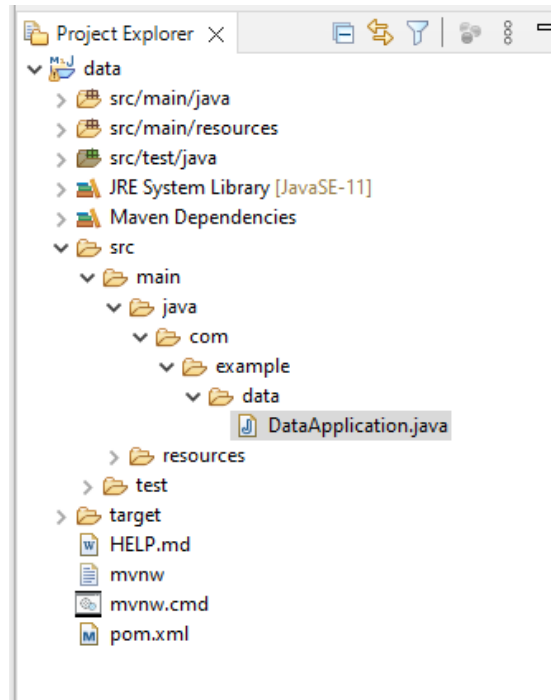
```
C:\Users\micro>mvn -version
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: C:\tools\apache-maven-3.8.4
Java version: 17.0.2, vendor: Oracle Corporation, runtime: C:\tools\java\jdk-17.0.2
Default locale: en_CA, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

## **Step 3: Create the project**

1. Create a new eclipse workspace
2. Use the file import option to import the unzipped director as a Maven project.



3. Once the project has been imported; Maven will build all the dependencies. This may take a while but eventually your project should look like this.



4. Run the DataApplication.java main() method as a Java Application to ensure that everything is working.

```
1 package com.example.data;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DataApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DataApplication.class, args);
10    }
11 }
12
13 }
```

terminated> DataApplication [Java Application] C:\tools\java\jdk-17.0.2\bin\javaw.exe (Feb. 27, 2022, 6:33:07 p.m. - 6:33:11 p.m.)

Spring Boot (v2.6.4)

```
2022-02-27 18:33:09.038 INFO 211124 --- [main] com.example.data.DataApplication :
2022-02-27 18:33:09.042 INFO 211124 --- [main] com.example.data.DataApplication :
2022-02-27 18:33:09.514 INFO 211124 --- [main] .s.d.r.c.RepositoryConfigurationDelegate :
2022-02-27 18:33:09.529 INFO 211124 --- [main] .s.d.r.c.RepositoryConfigurationDelegate :
2022-02-27 18:33:10.053 INFO 211124 --- [main] com.zaxxer.hikari.HikariDataSource :
2022-02-27 18:33:10.226 INFO 211124 --- [main] com.zaxxer.hikari.HikariDataSource :
2022-02-27 18:33:10.206 INFO 211124 --- [main] org.hibernate.jpa.internal.util.LogHelper :
2022-02-27 18:33:10.355 INFO 211124 --- [main] org.hibernate.Version :
2022-02-27 18:33:10.554 INFO 211124 --- [main] org.hibernate.annotations.common.Version :
2022-02-27 18:33:10.738 INFO 211124 --- [main] org.hibernate.dialect.Dialect :
2022-02-27 18:33:10.980 INFO 211124 --- [main] o.h.e.t.j.p.i.3taPlatformInitiator :
2022-02-27 18:33:10.990 INFO 211124 --- [main] j.LocalContainerEntityManagerFactoryBean :
2022-02-27 18:33:11.097 INFO 211124 --- [main] com.example.data.DataApplication :
2022-02-27 18:33:11.104 INFO 211124 --- [main] j.LocalContainerEntityManagerFactoryBean :
2022-02-27 18:33:11.104 INFO 211124 --- [main] .SchemaDropperImpl$DelayedDropActionImpl :
2022-02-27 18:33:11.107 INFO 211124 --- [main] com.zaxxer.hikari.HikariDataSource :
2022-02-27 18:33:11.110 INFO 211124 --- [main] com.zaxxer.hikari.HikariDataSource :
```

### Step 3: Create the Customer entity class and fields

1. Ensure that you are in the Java Perspective.
2. The code so far does nothing because all we have is a hibernate framework and no actual data
3. The entity to be persisted is a Customer object that has an ID which will serve as the primary key in the underlying database; and first and last name string fields.
4. Implement a Customer class as follows:

```
3 @Entity
4 public class Customer {
5
6     @Id
7     @GeneratedValue(strategy=GenerationType.AUTO)
8     private Long id;
9     private String firstName;
10    private String lastName;
11 }
12
```

5. The @Entity annotation indicates that Customer is a persistent entity
6. If the table that Customer is mapped to is not named Customer, there would be an additional @TableName annotation that would specify the table name
7. The @Id Annotation on the id field indicates it is the primary key and that it will be auto-generated.

#### Step 4: Create the constructors

1. JPA needs a constructor of no args which is protected.
2. The other constructor creates a Customer object populated with data
3. The toString() method give you a nice way to print out a customer

```
protected Customer() {}

public Customer(String firstName, String lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
}

@Override
public String toString() {
    return String.format(
        "Customer[id=%d, firstName='%s', lastName='%s']",
        id, firstName, lastName);
}
```

#### Sept 5: Create the getters

1. Finally, add the get methods for the data fields

```
public Long getId() {
    return id;
}

public String getFirstName() {
    return firstName;
}

public String getLastName() {
    return lastName;
}
```

### Step 6: Create a Repository Interface

1. Create a new Interface that extends the standard CRUD repository Interface
2. This new interface adds two new methods to the repository interface
3. The first looks up a Customer by id. It returns a single Customer object since the id is a primary key.
4. The second returns a list of all customers with the same last name.
5. The Spring JAP will then write the implementation of these methods using Hibernate when the application is run.

```
public interface CustomerRepository extends CrudRepository<Customer, Long> {  
  
    List<Customer> findByLastName(String lastName);  
    Customer findById(long id);  
}
```

### Step 7: Adding to the Runner class

1. The CommandLineRunner class takes a repository as an argument.
2. It returns a series of commands to be executed by the Spring Boot application.
3. The first set of methods create a few Customer objects and writes them into the H2 database

@Bean

```
public CommandLineRunner demo(CustomerRepository repository) {  
    return (args) → {  
        // save a few customers  
        repository.save(new Customer("Jack", "Bauer"));  
        repository.save(new Customer("Chloe", "O'Brian"));  
        repository.save(new Customer("Kim", "Bauer"));  
        repository.save(new Customer("David", "Palmer"));  
        repository.save(new Customer("Michelle", "Dessler"));  
    }  
}
```

4. The `findAll()` lists the contents of the Customer table. This is a method inherited from the CRUD repository interface.

```
// fetch all customers
System.out.println("Customers found with findAll():");
System.out.println("-----");
for (Customer customer : repository.findAll()) {
    System.out.println(customer.toString());
}
```

5. And the methods defined in the CustomerRepository are also usable.

```
// fetch an individual customer by ID
Customer customer = repository.findById(1L);
System.out.println("Customer found with findById(1L):");
System.out.println("-----");
System.out.println(customer.toString());
System.out.println("");

// fetch customers by last name
System.out.println("Customer found with findByLastName('Bauer'):");
System.out.println("-----");
repository.findByLastName("Bauer").forEach(bauer → {
    System.out.println(bauer.toString());
});
```

The output is shown on the next page.

```
2022-02-27 19:15:45.552 INFO 275116 --- [main] com.example.data.DataApplication
2022-02-27 19:15:45.554 INFO 275116 --- [main] com.example.data.DataApplication
2022-02-27 19:15:46.041 INFO 275116 --- [main] s.d.r.c.RepositoryConfigurationDelegate
2022-02-27 19:15:46.088 INFO 275116 --- [main] s.d.r.c.RepositoryConfigurationDelegate
2022-02-27 19:15:46.626 INFO 275116 --- [main] com.zaxxer.hikari.HikariDataSource
2022-02-27 19:15:46.801 INFO 275116 --- [main] com.zaxxer.hikari.HikariDataSource
2022-02-27 19:15:46.885 INFO 275116 --- [main] o.hibernate.jpa.internal.util.LogHelper
2022-02-27 19:15:46.979 INFO 275116 --- [main] org.hibernate.Version
2022-02-27 19:15:47.169 INFO 275116 --- [main] o.hibernate.annotations.common.Version
2022-02-27 19:15:47.332 INFO 275116 --- [main] org.hibernate.dialect.Dialect
2022-02-27 19:15:47.906 INFO 275116 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator
2022-02-27 19:15:47.915 INFO 275116 --- [main] j.LocalContainerEntityManagerFactoryBean
2022-02-27 19:15:48.386 INFO 275116 --- [main] com.example.data.DataApplication
Customers found with findAll():
-----
Customer[id=1, firstName='Jack', lastName='Bauer']
Customer[id=2, firstName='Chloe', lastName='O'Brian']
Customer[id=3, firstName='Kim', lastName='Bauer']
Customer[id=4, firstName='David', lastName='Palmer']
Customer[id=5, firstName='Michelle', lastName='Dessler']
Customer found with findById(1L):
-----
Customer[id=1, firstName='Jack', lastName='Bauer']

Customer found with findById(1L):
-----
Customer[id=1, firstName='Jack', lastName='Bauer']
Customer[id=3, firstName='Kim', lastName='Bauer']
```