

Professional Skills
Agile Fundamentals
Jira
Git
Databases Introduction
Java Beginner
Maven
Testing (Foundation)
Java Intermediate
HTML
CSS
Javascript
Spring Boot
<div><div></div>Introduction to Spring Boot</div>
<div><div></div>Multi-Tier Architecture</div>
<div><div></div>Beans</div>
<div><div></div>Bean Scopes</div>
<div><div></div>Bean Validation</div>
<div><div></div>Dependency Injection</div>
<div><div></div>Components</div>
<div><div></div>Configuration</div>
<div><div></div>Connecting to a Database</div>
<div><div></div>Entities</div>
<div><div></div>Postman</div>
<div><div></div>Controllers</div>
<div><div></div>Services</div>
<div><div></div>Repositories</div>
<div><div></div>Custom Queries</div>
<div><div></div>Data Transfer Objects</div>
<div><div></div>Lombok</div>
<div><div></div>Custom Exceptions</div>
<div><div></div>Swagger</div>
<div><div></div>Profiles</div>
<div><div></div>Pre-Populating Databases for Testing</div>
<div><div></div>Unit testing with Mockito</div>

# Entities

## Contents

- [Overview](#)
- [@Id](#)
- [@GeneratedValue](#)
  - [GenerationType](#)
- [@Column](#)
- [Tutorial](#)
- [Exercises](#)

## Overview

Entities are classes that represent tables in the database.  
Entity classes are annotated with `@Entity` and must have at least one field marked as an `@Id`, a *default* constructor *and* getters and setters.

## @Id

Denotes that the field represents a **Primary Key**.  
Can be applied to multiple columns in the case of a **composite Key**.

```
@Id
private Long id;
```

## @GeneratedValue

Sets a field to **Auto Increment**, can *only* be applied to a numeric field marked as an `@Id`.

```
@Id
@GeneratedValue
private Long id;
```

## GenerationType

There are a few strategies available for generating id's in Spring Boot but in order to set up an `AUTO_INCREMENT` field we will be using the `Identity` strategy.

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private long id;
```

## @Column

Fields in an entity are mapped to columns in that table by default (with variable names in **camelCase** being converted to **snake\_case** for the column names) but the addition of `@Column` allows for the *configuration* of those fields. In the following example the `username` field is set to be both **unique** and **not null** and the `password` field is set to be **not null** whereas the rest of the fields will default to being not-unique and nullable.

## Tutorial

► Example Entity  
The above example shows a possible way to model a user table as a Spring entity.

<div><div></div><div>Testing</div></div>
Selenium
Sonarqube
Advanced Testing (Theory)
Cucumber
MongoDB
Express
NodeJS
React
Express-Testing
Networking
Security
Cloud Fundamentals
AWS Foundations
AWS Intermediate
Linux
DevOps
Jenkins Introduction
Jenkins Pipeline
Markdown
IDE Cheatsheet

## Exercises

Create a new **Spring Starter Project** with this configuration:

New Spring Starter Project

Service URL

https://start.spring.io

Name

Account-Example

☒ Use default location

Location

C:\Users\jharry-work\Documents\sts-ws\Account-Example

Browse

Type:

Maven

Packaging:

Jar

Java Version:

8

Language:

Java

Group

com.qa

Artifact

AccountExample

Version

0.0.1-SNAPSHOT

Description

Example Spring application

Package

com.qa.demo

Working sets

☐ Add project to working sets

New...

Working sets:

Select...

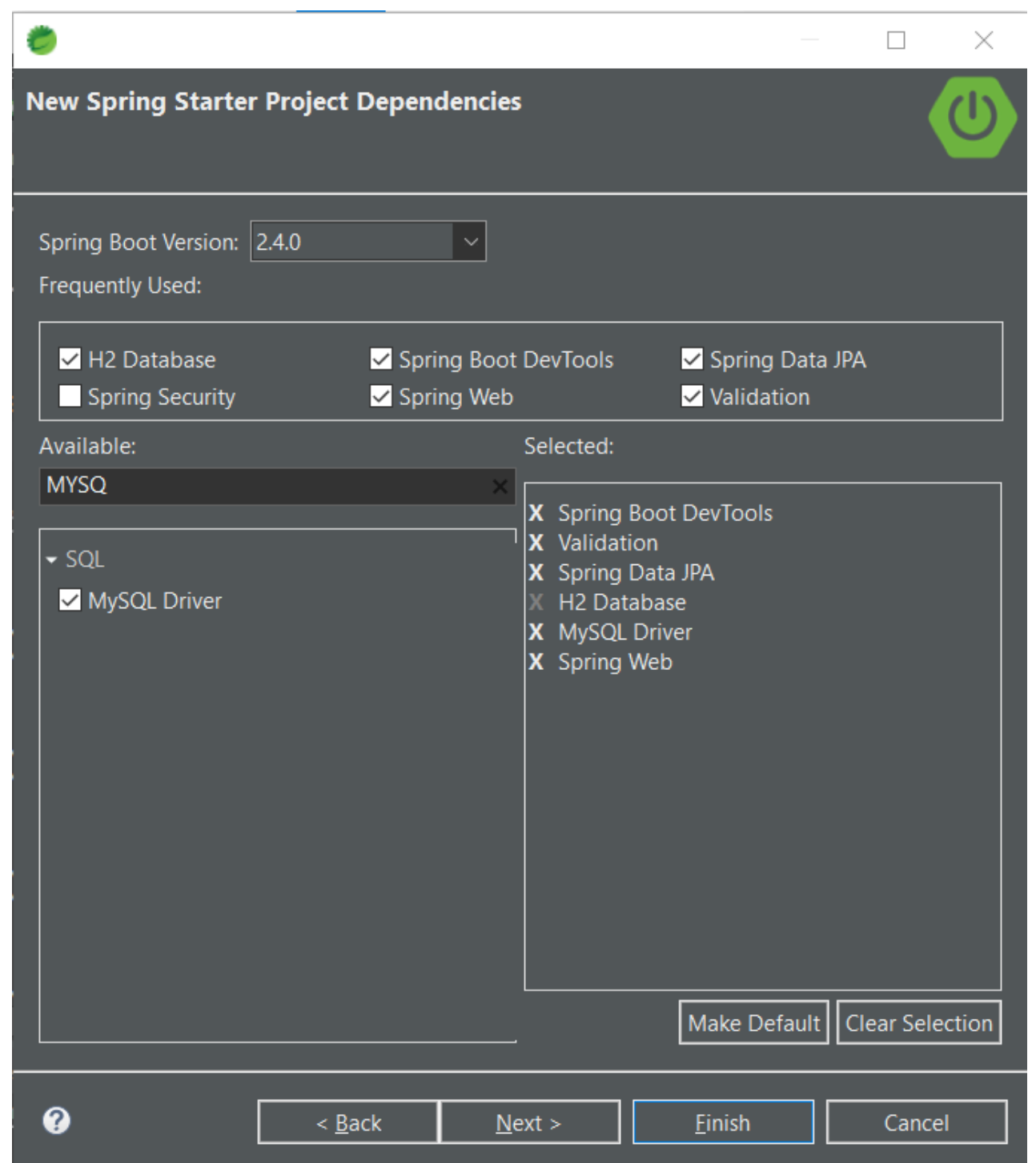
?

< Back

Next >

Finish

Cancel



Create a *valid* **Account** entity with:

1. A String **accountNumber** field that is both **unique** and **not null**.
2. A Long **id** field that is set to **AUTO\_INCREMENT**.
3. A String **name** field.

Make sure that your **Account** entity is in an appropriate package.