COURSEWARE

Connecting to a Database

Contents

Overview

As repositories are database independent, configuring the database connection is done externally in `application.properties`.

H2

H2 is the easiest database to get up and running - it is an in-memory database meant for testing and debugging.
In order to get an H2 up and running simply include the H2 dependency in the `pom.xml`:

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

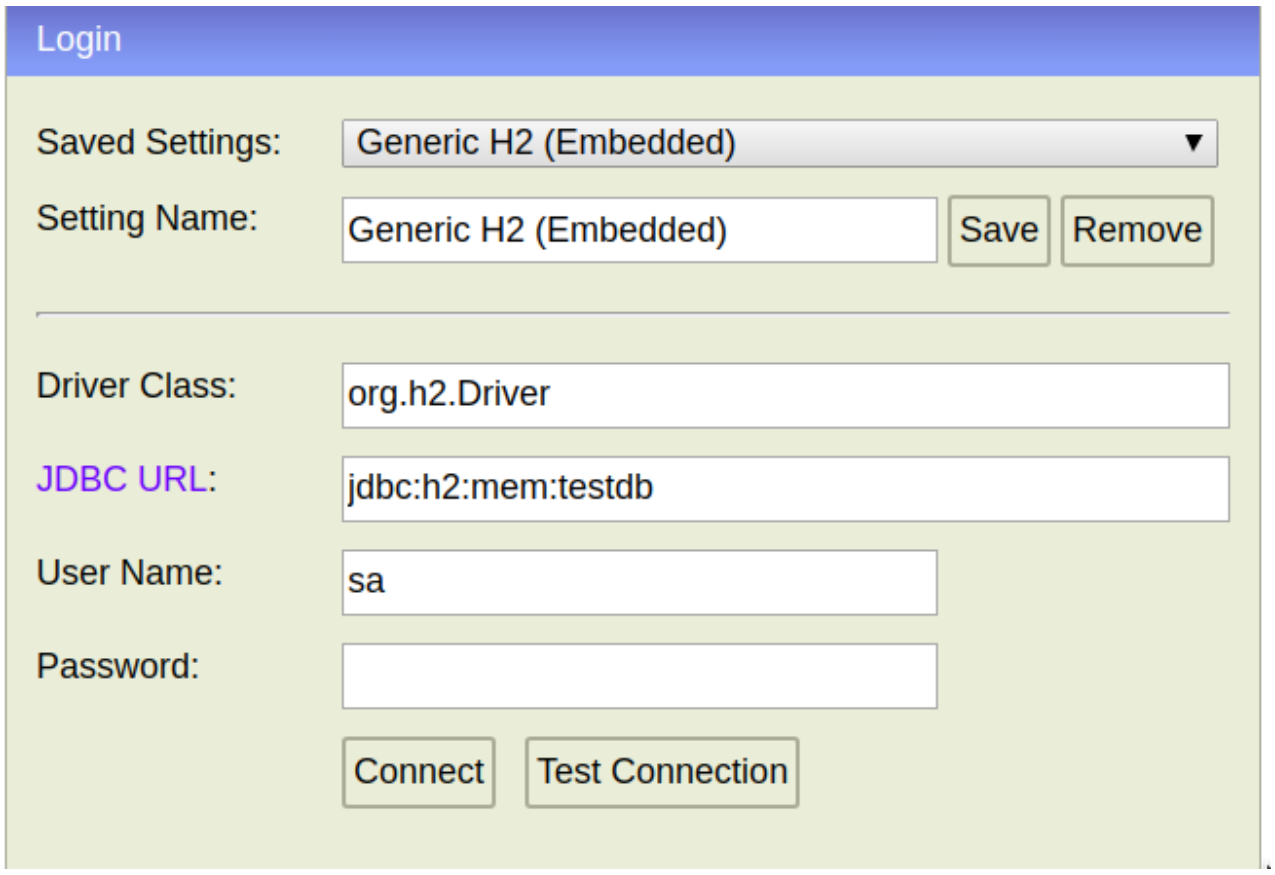
At runtime Spring will boot up an H2 instance and initialise the database from the entities in your application.

In order to view the database during runtime, the H2 console can be enabled and configured in `application.properties`.

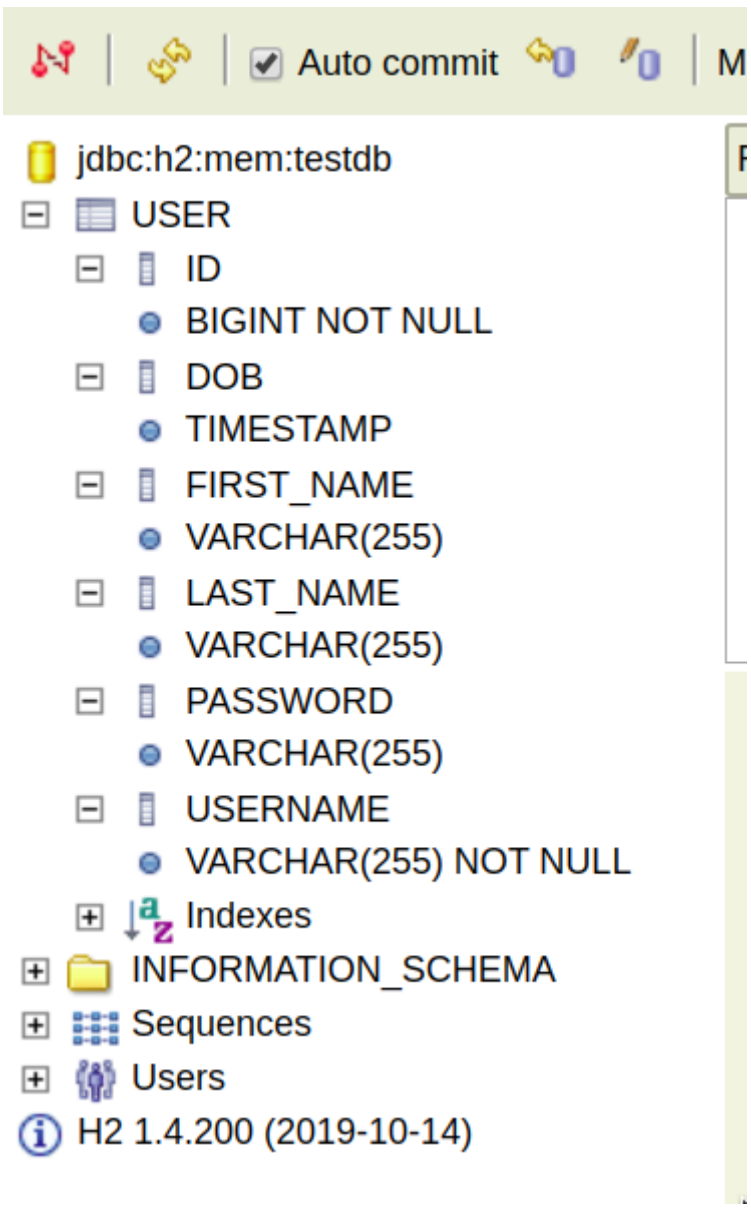
For instance, an embedded H2 called `testdb` would have the following `application.properties` file:

```
spring.h2.console.enabled=true
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=
spring.h2.console.path=/h2-console
```

Once enabled the console can be accessed from **/h2-console**, default login details can be seen below or set in **application.properties**.



Once logged into the H2 console the User table can be seen along with all of the generated columns.



MySQL

In order to connect to a MySQL database we need to add the correct dependency to the `pom.xml`:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

Then we need to set the JDBC URL and the user details for the database in `application.properties`.

```
spring.datasource.url=jdbc:mysql://localhost:3306/my_db
spring.datasource.username=root
spring.datasource.password=pass
```

DDL-auto

Spring Data has the power to automatically generate a database schema for you based off of the entities that are in your project.
This behaviour can be controlled using the `spring.jpa.hibernate.ddl-auto` property.

There are five valid values of this property:

none

The default for most databases, basically just disables this feature.

create

Tells Spring to generate the schema - this *will* override any existing schema in the database you connect to so any data that was there will be lost.

create-drop

As before except this time the generated schema will be destroyed when the Spring app stop - this is the default setting when connecting to an embedded database (such as an H2) as they will typically only persist for the duration of a single run of an application anyway.

update

Pretty much does what it says on the tin really; `update` will update the database schema to match your entities which can be really useful when you're more confident with Java than databases as it means you don't have to write any SQL.

validate

Very useful when dealing with persistent databases in a production environment; `validate` tells Spring to check your entities against the existing database schema

Tutorial

There is no tutorial for this module.

Exercises

Start up your `Account` project, setup a H2 database using **application.properties** and try and connect to the H2 console.