# Integrating an H2 Database in Your Spring Boot App

## H2

* H2 is one of the popular in memory databases. Spring Boot has very good integration for H2.
* *H2 is a relational database management system written in Java. It can be embedded in Java applications or run in the client-server mode.*
* H2 also provides a web console to maintain the database.
* An in-memory database is live only during the time of execution of the application. It is an efficient way to learn a framework.

## Spring Boot and H2

You need very little configuration to connect Spring Boot application with H2. In most situations, just adding the H2 runtime jar into dependencies should be sufficient.  
  
<dependency>  
<groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
</dependency>

## Spring Boot and H2 Magic

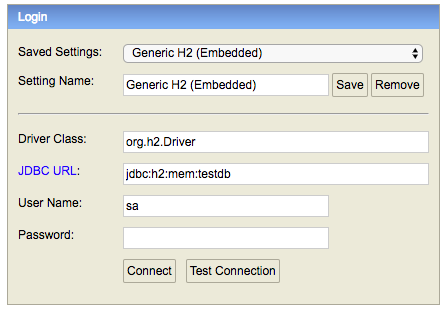
H2 provides a web interface called H2 Console to see the data. Let’s enable h2 console in the application.properties.

/src/main/resources/application.properties

*# Enabling H2 Console*

spring.h2.console.enabled=true

When you reload the application, you can launch up H2 Console at http://localhost:8080/h2-console.



*Tip - Make sure that you use jdbc:h2:mem:testdb as JDBC URL.*

### How did the Spring Boot Application connect to the database H2?

First thing you would need to understand is Spring Boot Auto Configuration.  
  
As far as H2 is concerned, as soon as Spring Boot sees H2 in the class path, it auto configures a data source similar to what you see below:

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

It knows that you are using an in memory database H2 and it uses the default url if you don’t provide one.

### *How did the Student table get created &* how does H2 and Spring Boot combination work?It’s because of Spring Boot Auto Configuration. If you are talking to an in memory db, by default, it looks at the entities and creates the database and the tables. However, if you connect to a mysql database, Spring Boot knows that its a permanent database. By default, it expects you to set up the database, set up the tables and it uses the connection that you established.

You can also populate some data into student table by adding a file called data.sql

/src/main/resources/data.sql

insert into student

values(10001,'Ranga', 'E1234567');

insert into student

values(10002,'Ravi', 'A1234568');

When you reload the application, you can launch up H2 Console to see that the student table is populated with the data from `data.sql’

* http://localhost:8080/h2-console.

How did all the magic happen? Let’s look at it question by question in the next section.

### Why is the data lost between restart?

H2 is an in memory database. Its not a persisted database.

### H2 Console is not Launched up?

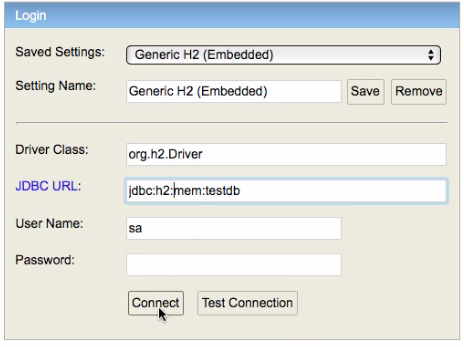
Try enabling it in the application.properties

spring.h2.console.enabled=true

### Error : Table is not created automatically in h2 embedded db or I’m unable to see the tables

Usually, the table’s are created but the url used in H2 GUI Console is wrong.

In the browser, change the database url to jdbc:h2:mem:testdb (Shown in the screen below).



You should be good to go!

### How did the insert query from data.sql run at application startup?

That’s part of the Spring Boot startup routine. Any queries in data.sql are run at application startup. You can read more here.

## Running H2 as a persisted database with Spring Boot

While we dont recommend this , it interesting to note that H2 has a persisted database mode

* With this configuration, the data is not lost even after spring boot restart and computer restart.

You would find H2 being very rarely used in this way. If you are really interested in a persistent database, we recommend exploring MySQL, Oracle or some other relational database.

application.properties

spring.datasource.name=yourdbname

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.initialize=false

spring.datasource.url=jdbc:h2:file:~/yourdbname;DB\_CLOSE\_ON\_EXIT=FALSE;IFEXISTS=TRUE;DB\_CLOSE\_DELAY=-1;

spring.jpa.hibernate.ddl-auto = update