**Spring boot H2 in memory database example**

By [Satish Varma](https://javabydeveloper.com/author/satishvarma/)

October 10, 2020

Spring boot H2 in-memory (or embedded) database example is a simple Spring boot application to demonstrate how to connect H2 database using Spring boot with JPA and test database results with a simple test case and from H2 console web browser.

Table of Content

 [[hide](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/)]

* [1. H2 In memory database](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/#0-1-h2-in-memory-database-)
* [2. Integrating Spring boot + H2 in memory database + JPA](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/#1-2-integrating-spring-boot-h2-in-memory-database-jpa-)
* [3. Conclusion](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/#2-3-conclusion)

**1.** [**H2 In memory database**](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/#h2-in-memory-database)

**What is H2 Databse?** : [**H2**](https://www.h2database.com/html/main.html) is a open-source relational database management system written in Java. It can be embedded in Java applications or run in client-server mode. It is one of the popular [In memory database](http://www.h2database.com/html/features.html#in_memory_databases). Spring Boot provides excellent integration support for H2.

**H2 Database Main Features :**

* Very fast and light weight database engine
* Open source
* Written in Java
* It is possible to create both **in-memory tables**, as well as **disk-based tables**.
* Supports standard SQL, JDBC API
* Embedded and Server mode, Clustering support
* Strong security features
* The PostgreSQL ODBC driver can be used
* Multi version concurrency
* Two full text search implementations are included, a native implementation and one using Lucene.

The complete list of features you can find at [H2 Features](https://www.h2database.com/html/features.html).

**2.** [**Integrating Spring boot + H2 in memory database + JPA**](https://javabydeveloper.com/spring-boot-h2-in-memory-database-example/#spring-boot-h2)

2.1. **Technologies used :**

1. H2 1.4.2
2. Spring Boot 2.2.2.RELEASE
3. Spring 5.2.2.RELEASE
4. Spring Data JPA 2.2.3.RELEASE
5. Junit 5.5.2
6. Maven 3
7. Java 8
8. Spring Tool Suite 3.9.8

2.2. **Project Structure :**

Following is the project structure I have created in this example.

2.3. **Maven pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.2.2.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>com.javabydeveloper</groupId>

<artifactId>Spring-Boot-jpa-h2-example</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Spring-Boot-jpa-h2-example</name>

<description>Spring boot2 in memory database example with jpa and H2</description>

<properties>

<!-- Dependency versions -->

<junit.jupiter.version>5.5.2</junit.jupiter.version>

<maven-surefire-plugin.version>2.22.2</maven-surefire-plugin.version>

<maven-jar-plugin.version>3.1.1</maven-jar-plugin.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

<!-- Maven plugin to use perticular java version to compile code -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

2.4. **User entity to save in H2 Database**

@Entity(name = "USER")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO, generator = "native")

@GenericGenerator(name = "native", strategy = "native")

@Column(name = "ID")

private Long id;

@Column(name = "USER\_NAME")

private String userName;

//@JsonIgnore

@Column(name = "PASSWORD")

private String password;

@Temporal(value = TemporalType.TIMESTAMP)

@Column(name = "CREATED\_TIME")

private Date creationTime;

@Temporal(value = TemporalType.TIMESTAMP)

@Column(name = "UPDATED\_TIME")

private Date updatedTime;

@Temporal(value = TemporalType.DATE)

@Column(name = "DOB")

private Date dateofBirth;

@Enumerated(value = EnumType.STRING)

@Column(name = "USER\_TYPE")

private UserType userType;

@Transient

private String dateOfBirthString;

// Setters and Getters

}

2.5. **UserRepository**

UserRepository.java to provide mechanism for storage, retrieval, search, update and delete operation to H2 in memory database on User entity.

@Repository

public interface UserRepository extends JpaRepository<User, Long>{

}

2.6. **H2 data source properties**

Following are the application properties in application.properties to create data source for H2. For in-memory embedded mode, *spring.datasource.url* need to update in spring boot properties file jdbc:h2:mem:<db\_name>. You can operate database in [other modes](http://www.h2database.com/html/features.html#database_url) also like server or mixed mode.

# H2 properties

spring.datasource.url=jdbc:h2:mem:jpa\_jbd

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.H2Dialect

spring.jpa.show-sql=true

2.7. **Enabling H2 console and web server port mapping**

Set following properties in application.properties to enable H2 console access for GUI access and update port to run application different port than 8080.

# webserver port mapping, default is 8080

server.port=8181

# custom root context, default is application name

server.servlet.context-path=/usermanager

server.error.whitelabel.enabled=false

# Enables H2 console

spring.h2.console.enabled=true

# custome H2 console url, Default is h2-console

spring.h2.console.path=/h2

2.8. **Configuration to Start Spring boot application**

Following code is to run Spring boot app. Spring calls @PostConstruct initDb() method after the initialization of bean properties.

@SpringBootApplication

public class MySpringBootApplication {

@Autowired

private UserRepository userRepository;

public static void main(String[] args) {

SpringApplication.run(MySpringBootApplication.class, args);

}

// spring calls after the initialization of bean properties

@PostConstruct

private void initDb() {

User user = new User();

user.setUserType(UserType.STUDENT);

user.setUserName("PeterM");

user.setPassword("ABC123abc\*");

user.setDateofBirth(new Date());

user.setCreationTime(new Date());

userRepository.save(user);

}

}

2.9. **Access H2 from console**

The [H2 Console](http://www.h2database.com/html/quickstart.html) lets you access a SQL database using a browser gui interface. It is a lightweight application used for examples only. It is not robust or scalable, is not supported, and should NOT be used in a production environment.

Start application using maven $ mvn spring-boot:run, so that application will get started under port 8181 on context root /usermanagement based on above configuration.

Np need to start H2 database separately, you can access H2 console using web browser http://127.0.0.1:8181/usermanager/h2 based on above configuration. You will see following window in browser. You can test and connect to H2 database.

Once you connect H2 database using console, you can see the tables created and apply query to see results.

2.10. **Spring Controller to Users results from browser**

@RestController

@RequestMapping("/api/users")

public class UserController {

@Autowired

private UserRepository userRepository;

@GetMapping

public List<User> getAllUsers() {

return userRepository.findAll();

}

@GetMapping("/{id}")

public ResponseEntity<User> getUserById(@PathVariable(value = "id") Long userId){

User user = userRepository.findById(userId)

.orElseThrow(() -> new NoSuchElementException("User not availbele for Id :"+userId));

return ResponseEntity.ok().body(user);

}

}

Hit the following url in browser http://127.0.0.1:8181/usermanager/api/users, you will see following results in browser.

2.11. **Test the configuration using Test Case**

Following is the simple [Junit 5](https://javabydeveloper.com/junit-5-tutorial/) tests case to save User entity in H2 database and test results.

@SpringBootTest

public class Spring\_boot\_H2\_test {

@Autowired

private UserRepository userRepository;

@Test

@DisplayName("Create User Test ")

void createUserTest() {

User created = userRepository.save(getUser());

assertTrue(created != null);

}

private User getUser() {

User user = new User();

user.setUserType(UserType.STUDENT);

user.setUserName("PeterM");

user.setPassword("ABC123abc\*");

user.setDateofBirth(new Date());

user.setCreationTime(new Date());

return user;

}

}

**3. Conclusion**

In this tutorial we went through H2 in memory database Spring Boot example. We have seen configurations to integrate Spring boot, H2 and JPA and how to access H2 console.

You can checkout source from our [github](https://github.com/javabydeveloper/spring-boot/tree/master/Spring-Boot-jpa-h2-example) repository.

**References**

1. [H2 database](https://www.h2database.com/html/main.html)
2. [Spring boot hello world](https://javabydeveloper.com/spring-boot-hello-world-example-rest/)
3. [Spring boot Junit 5 Test](https://javabydeveloper.com/spring-boot-junit-5-test-example/)
4. [H2(DBMS)](https://en.wikipedia.org/wiki/H2_(DBMS))
5. [JpaRepositiory](https://docs.spring.io/spring-data/jpa/docs/current/api/org/springframework/data/jpa/repository/JpaRepository.html)
6. [@Transient annotation in JPA and Hibernate to ignore fields](https://javabydeveloper.com/transient-annotation-non-persistent-field-using-jpa-hibernate/)
7. [JUnit 5 Tutorial](https://javabydeveloper.com/junit-5-tutorial/)