Spring Boot MySQL

# Docker spring boot MySQL example | Spring boot MySQL docker

<https://youtu.be/PAQvxqocb6A>

# Aplicación Spring Boot

El autor usa su propia aplicación. Yo voy a usar la que ya tengo hecha. Está en la carpeta crud. Es una aplicación de Spring Boot, con API REST. Usa una base de datos MySQL también propia de él, pero yo sigo con la que ya tenía hecha para mi aplicación.

## src\main\resources\application.properties

# Debe ser igual al artifactId del pom

spring.application.name=crud

server.port=8080

spring.datasource.driver-class-name =com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://${MYSQL\_HOST:localhost}:3306/anime

spring.datasource.username=cruduser

spring.datasource.password=crudPassword

# https://stackoverflow.com/questions/50322550/hibernate-dialect-for-mysql-8

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

#spring.jpa.show-sql: true

logging.level.org.hibernate.SQL=debug

# Arrancar WampServer

Necesitamos asegurarnos de que el puerto 3306 (el de MySQL) no está en uso. Una vez en WampServer, arrancar phpMyAdmin.

# \40-docker\anime.sql

En este archivo está el script SQL para crear la base de datos y el usuario. Copiar y pegar todo en el panel SQL de phpMyAdmin. Una vez ejecutado, tenemos la base de datos y el usuario.

# Ejecutar en la PC local desde el IDE con MySQL en WampServer

<https://youtu.be/PAQvxqocb6A?t=38>

Si uno tiene corriendo WampServer y ejecuta la aplicación desde el IDE, se ve que todo funciona correctamente. Hay que asegurarse de que el puerto 3306 no está siendo usado por un container, por ejemplo.

<https://youtu.be/PAQvxqocb6A?t=73>

En la URL

<http://localhost:8080/all>

se puede ver la lista de todos los animes cargados en la base de datos. Esto es lo que hemos hecho muchas veces, es conocido y no tiene nada de sorprendente.

# MySQL en un container

<https://youtu.be/PAQvxqocb6A?t=134>

Si no tengo la imagen de MySQL, la tengo que bajar de hub.docker.com:

docker pull mysql

Ahora que tengo la imagen en mi PC local, tengo que instanciar un container para MySQL. Asegurarse de que el puerto no está en uso, por ejemplo, hay que apagar WampServer:

docker run --name mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=my-secret-pw -d mysql

Ahora tengo MySQL server en un container, con las credenciales que elegí.

# MySQL Workbench 8.0 CE

Como apagué el Wamp, no puedo usar phpMyAdmin. Descargué MySQL Workbench Community Edition de:

<https://dev.mysql.com/downloads/workbench/>

Lo instalé, y ahora lo ejecuto. Me conecto usando las credenciales del MySQL que está corriendo en el container, y que está listening en el puerto 3306 de mi localhost, porque así lo configuré. Y como esto es lo que asume MySQL Workbench por defecto, no hay problemas.

En el file system del container, MySQL guarda las bases de datos en /var/lib/mysql

# \40-docker\anime.sql

En este archivo está el script SQL para crear la base de datos y el usuario. La base de datos que creamos antes con este mismo script está en el sistema de archivos de WampServer, en el lugar donde MySQL guarda las bases de datos. El MySQL que está ahora corriendo en el container está limpio, no tiene ninguna base de datos. Copiar y pegar todo en el panel Query 1 del Workbench. Una vez ejecutado, tenemos la base de datos y el usuario, todo en el container.

# Ejecutar en la PC local desde el IDE con MySQL en container

En la URL

<http://localhost:8080/all>

se puede ver la lista de todos los animes cargados en la base de datos. La aplicación está ahora conectada a la base de datos que está en el container de MySQL.

# Dockerizar la aplicación Spring Boot

Ya tenemos dockerizado el server de MySQL. Ahora tenemos que dockerizar la aplicación de Spring Boot.

# Dokerfile

Vamos a la carpeta de la aplicación, 40-docker\crud, creamos un archivo de texto y lo llamamos Dockerfile, sin extensión. En ese archivo pegamos lo siguiente:

# eclipse-temurin soporta versiones recientes del JDK.

# En muchos tutoriales usan openjdk, pero la documentación

# oficial dice que está deprecado.

FROM eclipse-temurin

ADD target/\*.jar app.jar

ENTRYPOINT ["java", "-jar", "app.jar"]

Para construir la imagen de la aplicación Spring Boot:

docker build -t crud .

Para instanciar un container a partir de esta imagen:

docker run -p 8080:8080 --name crud -e MYSQL\_HOST=mysql -e MYSQL\_USER=root -e MYSQL\_PASSWORD=my-secret-pw -e MYSQL\_PORT=3306 crud

docker run -p 8080:8080 --name crud -e MYSQL\_HOST=mysql -e MYSQL\_USER=cruduser -e MYSQL\_PASSWORD=crudPassword -e MYSQL\_PORT=3306 crud

Antes de ejecutar este comando, hay que asegurarse de que el puerto 8080 está libre. Entre otras cosas, si la aplicación está corriendo en el IDE, hay que apagarla. Si nos olvidamos, y nos da un error, el container de todos modos fue construido.

Puede dar el error de que la app fue compilada con una versión de Java posterior a la del container. Hay que recompilar la app.

# Docker compose

El formato del archivo docker-compose.yml está definido en:

<https://github.com/compose-spec/compose-spec/blob/master/spec.md#compose-file>

# Version top-level element

<https://github.com/compose-spec/compose-spec/blob/master/spec.md#version-top-level-element>

A pesar de que muchos tutoriales y ejemplos ponen este parámetro, está deprecado. Yo no lo voy a usar.

## docker-compose.yml

services:

  mysql:

    image: mysql

    environment:

      - MYSQL\_ROOT\_PASSWORD=mysql\_root\_password

      - MYSQL\_PASSWORD=crudPassword

      - MYSQL\_DATABASE=anime

    ports:

      - "3306:3306"

  app:

    image: crud

    environment:

      - SPRING\_DATASOURCE\_URL=jdbc:mysql://mysql:3306/anime

      - SPRING\_DATASOURCE\_USERNAME=cruduser

      - SPRING\_DATASOURCE\_PASSWORD=crudPassword

    ports:

      - "8080:8080"

    build:

      context: ./

      dockerfile: Dockerfile

    depends\_on:

      - mysql

Este archivo se ejecuta con docker-compose up.

Notar que se ocupa de buildear la imagen de la aplicación crud, si no existe.