VUE-FORUM-APP

Jaume López Mulet

INDEX

Disseny

En aquesta aplicació la part de frontend esta desenvolupada amb VUE, un framework de javascript basat en components. Damunt aquest frontent s'ha desenvolupat un backend en java EE emprant SPRING BOOT i HIBERNATE com eina per a gestionar la base de dades MYSQL.

A diferència de les pràctiques anteriors, en aquest cas, com que empram Spring Boot, el nostre servidor tomcat per a poder executar l'aplicació està EMBEDIT a dintre del JAR que es genera per a poder executar l'app.

El patró que hem escollit per a la nostra aplicació es el model MVC (model vista controlador) conjuntament amb el patró DAO per accedir a la base de dades amb una capa de servei.

Per a gestionar l'accés controlat a l'aplicació s'ha implementat l'accés mitjançant un JWT token, on es genera en el servidor i es comprova quan es realitzen les peticions per part del client.

Instruccions

Per descarregar el projecte i fer el deploy es pot fer un clone del repositori o descarregar-lo des de GITHUB des del seguent enllaç:

https://github.com/neo3kk/PracticaFinal

git@github.com:neo3kk/PracticaFinal.git

Una vegada descarregat el projecte, és necessari obrir-lo amb l'ide en el nostre cas InteliJ i comprovam que el nostre fitxer pom.xml està de la següent manera:

```
<?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>2.4.2</version>
     <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.rest</groupId>
  <artifactId>vue</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>vue</name>
  <description>Forum vue rest</description>
  <properties>
     <java.version>11</java.version>
     <maven.test.skip>true</maven.test.skip>
  </properties>
  <dependencies>
     <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
     </dependency>
     <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>runtime</scope>
     </dependency>
     <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
```

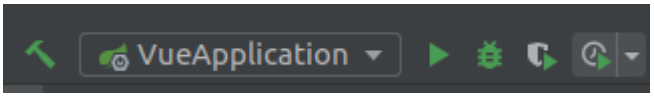
```
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.6</version>
        </dependency>
        <dependency>
            <groupId>com.auth0</groupId>
            <artifactId>java-jwt</artifactId>
            <version>3.12.0</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

Degut a que el projecte esta amb Spring Boot, la manera de inicialitzar l'aplicació es simplifica ja que el tomcat esta embedit a dintre i la propia aplicació es pot executar amb una clase main.



Abans de inicialitzar l'aplicació però, hem de comprovar el nostre fitxer application.properties:

```
server.port=5000

token.secret=sfgdsagsfdafsdfsfgdfdagsadf
token.expiration.time=1000000000

logging.level.org.springframework.jdbc.core=TRACE

spring.datasource.url = jdbc:mysql://localhost:3306/hibernate?allowPublicKeyRetrieval=true&useSSL=false&createDatabaseIfNotExist=true&serverTimezone=UTC
spring.datasource.username = root
spring.datasource.password = root
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.properties.show-sql = false
spring.jpa.properties.hibernate.format_sql = false
spring.jpa.hibernate.ddl-auto = update
```

Es important definir el nom de la base de dades així com la URL del servidor que volguem a la conexió, però en aquest cas, es crearà una nova base de dades si es que la que està definida no existeix, ja que s'executarà en un contenidor docker.
Una vegada fetes les comprovacions, es necessari iniciar els nostres contenidors docker per a poder executar l'aplicació en local….

DEVELOPMENT

Dins el directori **Dev,** executar l'script **startup.sh**

Aquest script executa:

- Contenidor node amb la comanda npm install i npm run serve per posar en funcionament el frontend al port 8080
- Contenidor mysql al port 3306
- Contenidor phpmyadmin al port 8090

Quan es té l'entorn aixecat, es pot executar el projecte a l'IDE i anar a l'adreça: http://localhost:8080

Una vegada finalitzada la sessió, executar l'escript stop.sh per a aturar i eliminar els contenidors.

Si volguesim generar un jar per a dur a producció, només hem de executar:

- maven package -P properties

Així ens aseguram que s'agafen les propietats del fitxer properties.

PRODUCTION

Per aixecar l'entorn de producció cal anar al directori Prod i executar la comanda

- docker-compose up -d

Ara tendrem els contenidors:

- Apache al port 8080 amb el dist del frontend
- mysql al port 3306
- phpmyadmin al port 8090
- openjdk amb el jar al port 5000

Accedirem al client amb la mateixa adreça:
http://localhost:8080

Aturarem l'entorn amb:

- docker-compose down