SPRING BOOT + SECURITY + BOOTSTRAP

SEGURIDAD CON SPRING BOOT + THYMELEAF Y MYSQL. CAPÍTULO 2. CREANDO EL PROYECTO

Bueno, pues vamos al lío. En la página de [Spring Initializr](https://start.spring.io/) creamos un proyecto llamado thymeleaf\_security (o el nombre que os plazca). El proyecto será tipo Maven, la versión de Spring Boot la 2.1.8. Las dependencias serán las siguientes:

* Web,
* JPA,
* MySQL Driver
* Spring Security
* Thymeleaf

Descargamos el proyecto pulsando el botón Generate the Project y lo guardamos. Se nos descargará un archivo tipo .zip. Lo descomprimimos y lo importamos IntelliJ.

**Añadir dependencias al pom.xml:**

Nos hacen falta un par de dependencias más (thymeleaf security y commons-lang3), así que las vamos añadir a mano en el pom.xml.  Nos deberá quedar así:

|  |  |
| --- | --- |
|  | <?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>2.1.8.RELEASE</version> |
|  | <relativePath/> <!-- lookup parent from repository --> |
|  | </parent> |
|  | <groupId>com.inezpre5</groupId> |
|  | <artifactId>thymeleaf\_security</artifactId> |
|  | <version>0.0.1-SNAPSHOT</version> |
|  | <name>thymeleaf\_security</name> |
|  | <description>Demo project for Spring Boot</description> |
|  |  |
|  | <properties> |
|  | <java.version>1.8</java.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-security</artifactId> |
|  | </dependency> |
|  | <dependency> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-starter-thymeleaf</artifactId> |
|  | </dependency> |
|  | <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>org.springframework.security</groupId> |
|  | <artifactId>spring-security-test</artifactId> |
|  | <scope>test</scope> |
|  | </dependency> |
|  | <!-- https://mvnrepository.com/artifact/org.thymeleaf.extras/thymeleaf-extras-springsecurity5 --> |
|  | <dependency> |
|  | <groupId>org.thymeleaf.extras</groupId> |
|  | <artifactId>thymeleaf-extras-springsecurity5</artifactId> |
|  | </dependency> |
|  | <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 --> |
|  | <dependency> |
|  | <groupId>org.apache.commons</groupId> |
|  | <artifactId>commons-lang3</artifactId> |
|  | <version>3.4</version> |
|  | </dependency> |
|  | </dependencies> |
|  |  |
|  | <build> |
|  | <plugins> |
|  | <plugin> |
|  | <groupId>org.springframework.boot</groupId> |
|  | <artifactId>spring-boot-maven-plugin</artifactId> |
|  | </plugin> |
|  | </plugins> |
|  | </build> |
|  |  |
|  | </project> |

[**view raw**](https://gist.github.com/cavanosa/bbe7412775eee4e914092d7788cd72d3/raw/0930506874e86ae9ecff1c006a3e1cc411171324/pom.xml)[**pom.xml**](https://gist.github.com/cavanosa/bbe7412775eee4e914092d7788cd72d3#file-pom-xml) hosted with  by [**GitHub**](https://github.com/)

**Crear la base de datos:**

En MySQL crearemos una base de datos a la que le pondremos el mismo nombre del proyecto (por supuesto que el nombre puede ser el que queráis, yo lo hago así para saber siempre qué base de datos se corresponde con qué proyecto). Podemos hacerlo tecleando el siguiente comando en la consola MySQL

*CREATE DATABASE thymeleaf\_security CHARACTER SET utf8 COLLATE utf8\_general\_ci;*

o bien con el editor visual de phpMyAdmin o con el de MySQL Workbench.

**Configuración de application.properties:**

Ahora toca conectar la base de datos con el proyecto por lo que tenemos que editar el archivo application.properties. Lo dejamos como se muestra a continuación:

|  |  |
| --- | --- |
|  | spring.datasource.url = jdbc:mysql://localhost:3306/thymeleaf\_security?useSSL=false&serverTimezone=UTC&useLegacyDateTimeCode=false |
|  |  |
|  | # nombre de usuario y contraseña |
|  | spring.datasource.username = username |
|  | spring.datasource.password = password |
|  |  |
|  | # mostrar sentencias SQL en la consola |
|  | spring.jpa.show-sql = true |
|  |  |
|  | # actualizar base de datos y crear entidades |
|  | spring.jpa.hibernate.ddl-auto = update |
|  |  |
|  | # hibernate genera SQL optimizado |
|  | spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect |
|  |  |
|  | spring.jmx.enabled= false |

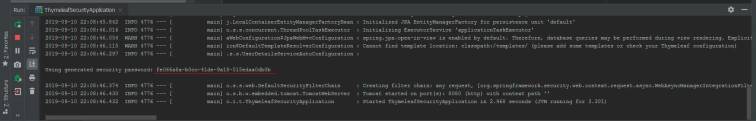
[**view raw**](https://gist.github.com/cavanosa/e0e95462e6e6d3bb7aa08697ef34315d/raw/caea518c8c8ca7ebb44160413cecee0f61a2d855/gistfile1.txt)[**gistfile1.txt**](https://gist.github.com/cavanosa/e0e95462e6e6d3bb7aa08697ef34315d#file-gistfile1-txt) hosted with  by [**GitHub**](https://github.com/)

Como siempre, el nombre de la base de datos tiene que estar bien puesto en la línea

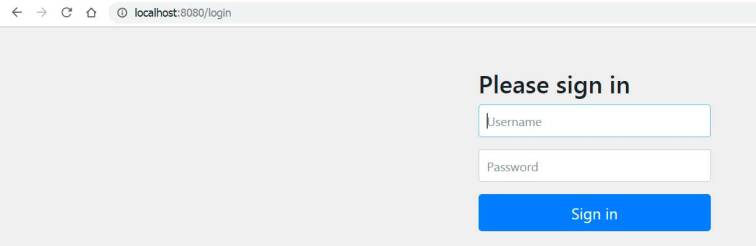
spring.datasource.url = jdbc:mysql://localhost:3306/login\_thymeleaf

Una vez que lo tenemos vamos a ejecutar el proyecto (sobre el ejecutable principal, en nuestro caso se llama ThymeleafSecurityApplication, botón derecho, run).

Al ejecutarse (y si todo ha ido bien, claro)  aparecerá una contraseña en la consola



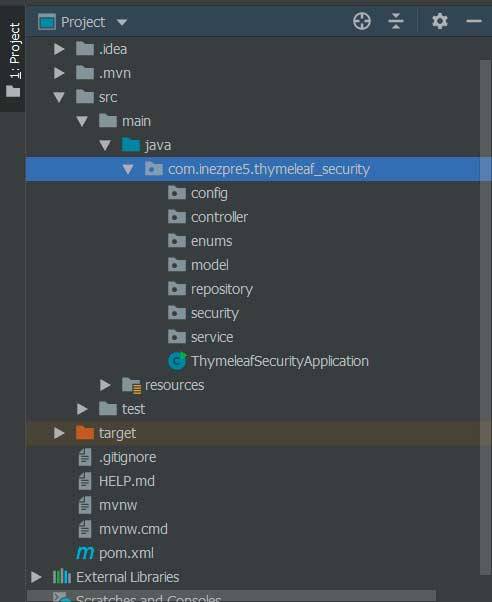
Si vamos a localhost:8080 nos aparece una ventana de login



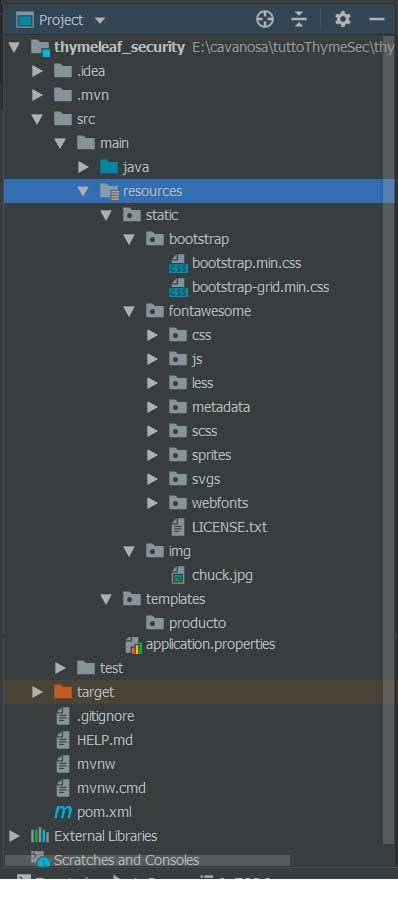
Como nombre de usuario ponemos user y la contraseña es la que aparece en consola. Le damos a Sign In y aparece la clásica página de Whitelabel Error 404 y bla, bla bla. Eso es por que aún no tenemos ninguna vista, pero la aplicación ya funciona pues no ha lanzado errores ni excepciones y nos ha mostrado la contraseña.

**Paquetes del proyecto:**

Ya para finalizar esta entrada vamos a crear los paquetes del proyecto. En java crearemos 7 (config, controller, enums, model, repository, security y service), os los muestro en la siguiente imagen:



Por su parte, el árbol de resources es el siguiente:



En *templates* hemos creado una carpeta nueva llamada producto, donde guardaremos las vistas de los productos (lista, detalle, crear, editar).

En *static* creamos 2 carpetas: img para guardar las imágenes, como su nombre indica, y otra llamada bootstrap.

Para descargar *Bootstrap* lo hacemos desde [este enlace](https://getbootstrap.com/docs/4.3/getting-started/download/), nos interesa el botón de los *css* y los *javascript* compilados. Se descargará un archivo tipo *zip*, lo descomprimimos y vemos que contiene dos carpetas: *css* y *js*. Pues bien, sólo nos interesan dos archivos dentro de la carpeta css: *bootstrap.min.css* y *bootsrap-grid.min.css*. Los copiamos y los guardamos en la carpeta *bootstrap* que acabamos de crear en el proyecto.

En cuanto a *Font Awesome* se descarga de [este enlace](https://fontawesome.com/download). Descomprimimos la carpeta y le cambiamos el nombre (mejor dicho, se lo acortamos) a *fontawesome* a secas. Guardamos esta carpeta en *static* y ya tenemos el árbol del proyecto listo.

Por supuesto, también puedes obtener *Bootstrap* y *Font Awesome* del repositorio del tutorial.

Y hasta aquí esta entrada, en el siguiente capítulo crearemos las entidades, los repositorios y los servicios.