Tabla de Contenidos

- Spring Boot y Spring Security (Autenticacion)
- URLs de Acceso y Error
- Spring Boot y Spring Security (Autorización)
- Spring MVC Login
- Otros artículos relacionados

Spring Boot y Spring Security son dos de las tecnologías que más usamos en nuestro día a día ya que todas las aplicaciones necesitan securizarse . Vamos a ver como podemos usar Spring Boot y Spring Security para generar una página de login básica que se autentifique. En primer lugar lo que tendremos que hacer es construirnos un proyecto con Spring Boot que contenga las dependencias necesarias a nivel de Starters.

```
<dependencies>
    <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>org.thymeleaf.extras</groupId>
      <artifactId>thymeleaf-extras-springsecurity5</artifactId>
    </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-boo
t-devtools -->
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

Spring Boot y Spring Security (Autenticacion)

Una vez tenemos configurados los starters es momento de configurar el fichero de SpringBoot que se encarga de gestionar la aplicación.

```
@SpringBootApplication
public class WebApplication extends WebSecurityConfigurerAdapter {

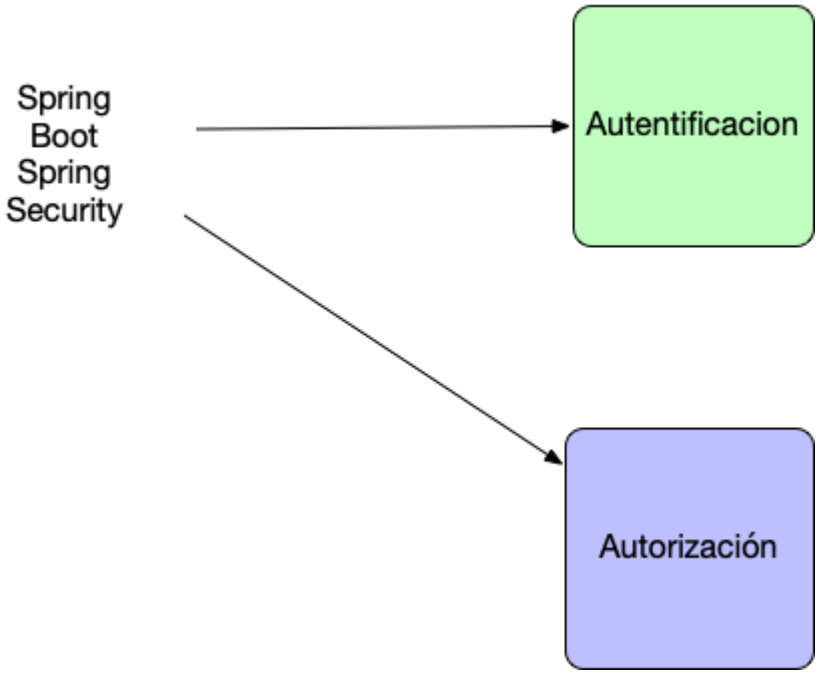
  public static void main(String[] args) {
    SpringApplication.run(WebApplication.class, args);
  }

  @Override
```

```
protected void configure(HttpSecurity http) throws Exception {

   http.authorizeRequests()
        .antMatchers("/login")
            .permitAll()
        .antMatchers("/**")
            .hasAnyRole("BASICO")
        .and()
            .formLogin()
            .loginPage("/login")
            .defaultSuccessUrl("/hola")
            .failureUrl("/login?error=true")
            .permitAll()
        .and()
            .logout()
            .logoutSuccessUrl("/login?logout=true")
            .invalidateHttpSession(true)
            .permitAll()
```

En este caso estamos configurando dos partes la autenticación y la autorización de forma paralela .



La autentificacion se gestiona a través del método configure . Este método se encarga de definir el tipo de autenticación y los roles que estan soportados para ella . En este caso estamos ante una autenticación que permite acceder a la url de login a todos los usuarios para que estos introduzcan sus credenciales y protege el resto de urls para que solo los usuarios que se encuentren en el rol básico puedan acceder.

```
http.authorizeRequests()
          .antMatchers("/login")
              .permitAll()
          .antMatchers("/**")
              .hasAnyRole("BASICO")
```

URLs de Acceso y Error

Una vez realizado este primer paso de definir las urls protegidas el siguiente paso que realiza el fichero de configuración es decidir que tipo de autenticación se realiza.

```
.and()
                .formLogin()
```

```
            .loginPage("/login")
            .defaultSuccessUrl("/hola")
            .failureUrl("/login?error=true")
            .permitAll()
        .and()
            .logout()
            .logoutSuccessUrl("/login?logout=true")
            .invalidateHttpSession(true)
            .permitAll()
            .and()
            .csrf()
            .disable();
```

En este caso se usa un método and() para añadir nuevas configuraciones en las cuales definimos el tipo de autenticación formlogin() (login con formulario) y las urls que tenemos asociadas a este casuística como son las url de error que nos devuelve al propio formulario como la url de existo que nos redirecciona a una página concreta /hola. Por ultimo desactivamos las restricciones de csrf() para esta página ya que es la de entrada.

Spring Boot y Spring Security (Autorización)

El siguiente paso es configurar que usuarios están autorizados para acceder a uno u otro contenido ya que dependiendo de tus permisos podrás acceder o no a la aplicación.

```
@Autowired
  public void configureGlobal(AuthenticationManagerBuilder auth)
throws Exception {
auth.inMemoryAuthentication().withUser("cecilio").password("{noop}micl
ave").roles("BASICO");
  }
```

Esa configuración también se realiza a nivel del fichero de configuración de Spring y nos

permite definir usuarios y roles . En este caso cecilio con clave miclave y rol básico

Spring MVC Login

Nos queda crear un Controlador que nos permita que al solicitar la url de /login usando una petición GET nos redirija a una pagina creada con Thymeleaf que solicite nombre y clave.

```
package com.arquitecturajava.we.b;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class LoginController {

  @RequestMapping("/login")
  public String login() {
    return "login";
  }
}
```

En este caso como vemos es muy sencillo es momento de mostrar el contenido de la plantilla con el usuario y la clave en un formulario personalizado.

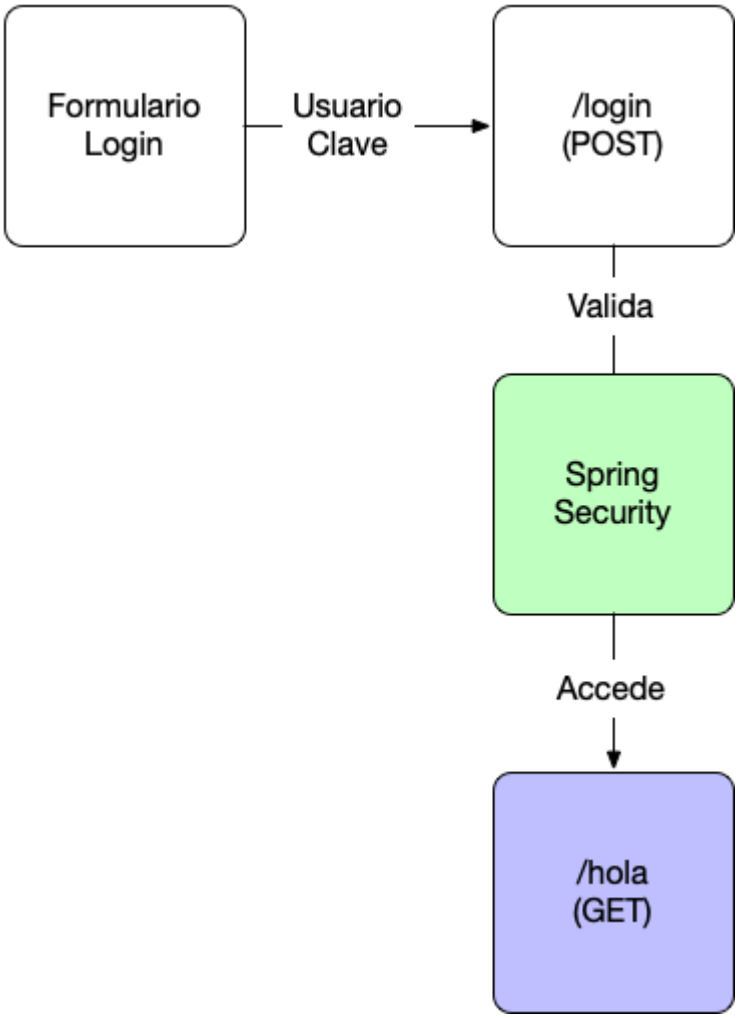
```
<html>
<head></head>
<body>
   <h1>Login</h1>
   <form name='f' action="login" method='POST'>
      <table>
         <tr>
            <td>Usuario:</td>
```

```
            <td><input type='text' name='username' value=''></td>
         </tr>
         <tr>
            <td>Clave:</td>
            <td><input type='password' name='password' /></td>
         </tr>
         <tr>
            <td><input name="submit" type="submit" value="submit"
/></td>
         </tr>
      </table>
  </form>
</body>
</html>
```

Si solicitamos con el navegador la url de /hola nos redireccionada el formulario de login que acabamos de definir:



Introducimos el usuario y la clave y accederemos al contenido de la aplicación .



Acabamos de configurar Spring Boot y Spring Security con un formulario personalizado.



Otros artículos relacionados

1. ¿Qué es Spring Boot?
2. Spring Boot JDBC y su configuración
3. Spring Boot Load Data y testing
4. Curso de Spring Boot Gratis en Español

5. Spring Security