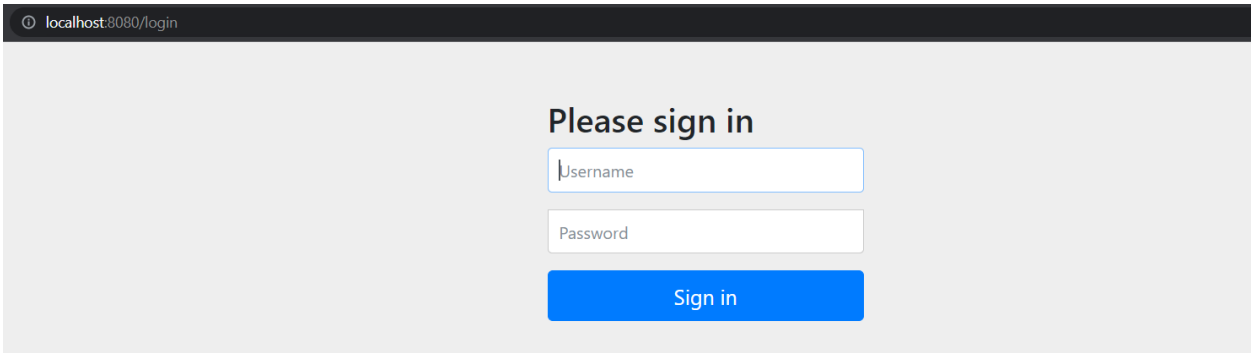
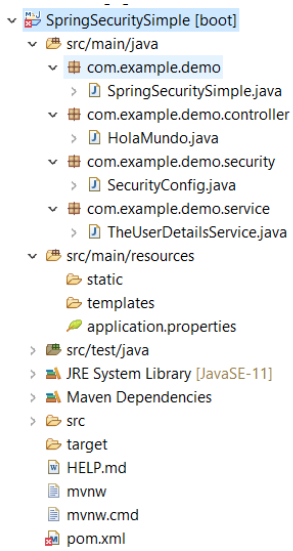
PAD

Spring Security - PADSA

Configurar la seguridad de nuestra API con Spring Security



Estuctura del projecto en Spring Boot Tool

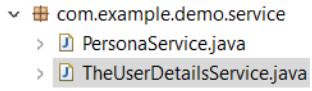


1. Se debe agregar la dependencia maven necesaria al pom.xml

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
    <version>2.6.3</version>
</dependency>
```

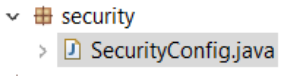
2. Para personalizar usuario y contraseña se procede a crear un nuevo servicio con la clase `TheUserDetailsService` la cual implementa la interface de Spring Security `UserDetailsService` en ella se sobre-escribe el método necesario `UserDetails` en el cual se retorna un nuevo usuario User de Spring Security en donde asignaremos los campos personalizados de usuario y contraseña.

   Importante agregar la notación `@Service` para poder inyectar el servicio.




```
package com.example.demo.service;

import java.util.ArrayList;

import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
public class TheUserDetailsService implements UserDetailsService  {

  @Override
  public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
    return new User("Edgar", "{noop}login", new ArrayList<>());
  }

}
```

3. Agregamos un nuevo paquete encargado de la gestión de la seguridad como `security` y dentro creamos la clase `SecurityConfig` que va a extender de la clase `WebSecurityConfigurerAdapter` que ya existe en Spring Security , además agregamos la notación @EnableWebSecurity para habilitarla.



4. Inyectamos nuestro servicio `TheUserDetailsService` con la notación @Autowired.
5. Para indicarle que los valores de usuario son los que definimos sobre-escribimos el método llamado `configure(AuthenticationManagerBuilder auth)` y le indicamos que la autenticación la manejaremos nosotros con el método `.userDetailsService()` le pasamos nuestras credenciales de usuario.

```
package security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
```

```
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import com.example.demo.service.TheUserDetailsService;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

  @Autowired
  private TheUserDetailsService theUserDetailsService;

  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    // TODO Auto-generated method stub
    auth.userDetailsService(theUserDetailsService);
  }
}
```

> El proyecto completo se encuentra en repositorio:

https://github.com/PADSA-github/Java-Basico/upload/main/SpringSecuritySimpl