SPRING SECURITY

SPRING SECURITY

- Spring Security es un potente y altamente personalizable marco de autenticación y control de acceso. Es el estándar de facto para asegurar las aplicaciones basadas en Spring.
- Spring Security es un marco que se enfoca en proporcionar tanto autenticación como autorización para aplicaciones Java. Al igual que todos los proyectos de Spring, el verdadero poder de Spring Security se encuentra en la facilidad con que se puede extender para cumplir con los requisitos personalizados.

DEMO : SPRING SECURITY - MARKET

DEPENDENCIAS EN EL ARCHIVO POM

- Agregar las siguientes dependencias en el archivo pom.xml

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

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

ENTIDADES USERS Y ROLE

- Crear las entidades Users y Role

```java
@Entity
@Table(name = "users")
public class Users implements Serializable {


    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(length = 30, unique = true)
    private String username;

    @Column(length = 60)
    private String password;

    private Boolean enabled;

    @OneToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL)
    @JoinColumn(name = "user_id")
    private List<Role> roles;
```

ENTIDADES USERS Y ROLE

- Crear las entidades Users y Role

```java
@Entity
@Table(name = "authorities", uniqueConstraints = { @UniqueConstraint(columnNames = { "user_id", "authority" }) })
public class Role implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String authority;
```

REPOSITORY USERREPOSITORY

- Crear interface UserRepository

```
@Repository
public interface UserRepository extends JpaRepository<Users, Long> {
    public Users findByUsername(String username);
}
```

SERVICE JPAUSERDETAILSSERVICE

- Crear clase JpaUserDetailsService

```java
@Service
public class JpaUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;


    @Transactional(readOnly = true)
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        Users user = userRepository.findByUsername(username);

        List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

        for (Role role : user.getRoles()) {
            authorities.add(new SimpleGrantedAuthority(role.getAuthority()));
        }

        return new User(user.getUsername(), user.getPassword(), user.getEnabled(), true, true, true, authorities);
    }

}
```

CLASE DE CONFIGURACION SPRINGSECURITYCONFIG

- Crear clase SpringSecurityConfig

```java
@EnableGlobalMethodSecurity(securedEnabled=true)
@EnableWebSecurity
@Configuration
public class SpringSecurityConfig  extends WebSecurityConfigurerAdapter{

    @Autowired
    private JpaUserDetailsService userDetailsService;

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;


    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests().antMatchers("/", "/css/**", "/js/**", "/img/**").permitAll()
        .anyRequest().authenticated()
        .and()
            .formLogin()
                .loginPage("/login")
            .permitAll()
        .and()
        .logout().permitAll()
        .and()
        .exceptionHandling().accessDeniedPage("/error");
    }


    @Autowired
    public void configurerGlobal(AuthenticationManagerBuilder build) throws Exception {
        build.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder);

    }
}
```

CONTROLADOR LOGINCONTROLLER

- Crear clase LoginController

```java
@Controller
@RequestMapping
public class LoginController {

    @GetMapping(value = { "/login", "/" })
    public String login(@RequestParam(value = "error", required = false) String error,
            @RequestParam(value = "logout", required = false) String logout, Model model, Principal principal,
            RedirectAttributes flash) {

        if (principal != null) {
            return "redirect:/dashboards/list";
        }

        if (error != null) {
            model.addAttribute("error",
                    "Error en el login: Nombre de usuario o contraseña incorrecta, por favor vuelva a intentarlo!");
        }

        if (logout != null) {
            model.addAttribute("success", "Ha cerrado sesión con éxito!");
        }

        return "login";
    }
}
```

GENERAR CLAVE DE USUARIO

- Agregar el siguiente código en la clase MarketApplication para generar clave encriptada.

```java
@SpringBootApplication
public class MarketApplication implements CommandLineRunner {

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    @Bean
    public BCryptPasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

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

    @Override
    public void run(String... args) throws Exception {

        String password = "12345";

        for (int i = 0; i < 2; i++) {
            String bcryptPassword = passwordEncoder.encode(password);
            System.out.println(bcryptPassword);
        }
    }

}
```

VISTAS

- Autorizar secciones de la vista según el rol

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"

      xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
```

```
<li class="nav-item dropdown" ><a sec:authorize="hasRole('ROLE_ADMIN')"
    class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
    role="button" data-toggle="dropdown" aria-haspopup="true"
    aria-expanded="false"> Mantenimiento </a>
    <div class="dropdown-menu" aria-labelledby="navbarDropdown">
        <a class="dropdown-item" href="#" th:href="@{/customers/list}">Clientes</a>
        <div class="dropdown-divider"></div>
        <a class="dropdown-item" href="#">Productos</a>


    </div>
</li>
```

CONTROLLER CUSTOMERCONTROLLER

- Autorizar métodos del controller

```
@Controller
@SessionAttributes("customer")
@RequestMapping("/customers/")
@Secured("ROLE_ADMIN")
public class CustomerController {
```

RUTA A VISTA ERROR

- Autorizar métodos del controller

```
package com.hampcode;

import org.springframework.web.servlet.config.annotation.ViewControllerRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

public class MvcConfig implements WebMvcConfigurer {

    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/error").setViewName("error");
    }

}
```