SPRING BOOT + SECURITY + BOOTSTRAP

# SEGURIDAD CON SPRING BOOT + THYMELEAF Y MYSQL. CAPÍTULO 4. CLASES DEL USUARIO

En el apartado anterior habíamos creado las clases del producto. En este crearemos las del usuario, sus roles y dos clases propias de Spring Security para la autenticación.

**Clases Usuario y Rol:**

Lo primero de todo será crear un enumerado llamado RolNombre (si has seguido los tutoriales de JWT te habrás dado cuenta de que las clases son las mismas).

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.enums; |
|  |  |
|  | public enum RolNombre { |
|  | ROLE\_ADMIN, |
|  | ROLE\_USER |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/6197119684b948fd201268017e70226d/raw/899ccaea787c0d6e6729101ed51a8ad01e8c865a/rolnombre.java)[**rolnombre.java**](https://gist.github.com/cavanosa/6197119684b948fd201268017e70226d#file-rolnombre-java) hosted with  by [**GitHub**](https://github.com/)

En el paquete model creamos las clases Usuario y Rol. Usuario es como la de los otros tutoriales pero para simplificar sólo he puesto los campos de nombre y contraseña (antes, además, teníamos el email y el nombre de usuario).

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.model; |
|  |  |
|  | import javax.persistence.\*; |
|  | import javax.validation.constraints.NotBlank; |
|  | import java.util.HashSet; |
|  | import java.util.Set; |
|  |  |
|  | @Entity |
|  | public class Usuario { |
|  | @Id |
|  | @GeneratedValue(strategy = GenerationType.IDENTITY) |
|  | private long id; |
|  | @NotBlank |
|  | @Column(unique = true) |
|  | private String nombre; |
|  | @NotBlank |
|  | private String password; |
|  | @ManyToMany(fetch = FetchType.EAGER) |
|  | @JoinTable(name = "usuario\_rol", joinColumns = @JoinColumn(name = "usuario\_id"), inverseJoinColumns = @JoinColumn(name = "rol\_id")) |
|  | private Set<Rol> roles = new HashSet<>(); |
|  |  |
|  | public Usuario() { |
|  | } |
|  |  |
|  | public Usuario(@NotBlank String nombre, @NotBlank String password) { |
|  | this.nombre = nombre; |
|  | this.password = password; |
|  | } |
|  |  |
|  | public long getId() { |
|  | return id; |
|  | } |
|  |  |
|  | public void setId(long id) { |
|  | this.id = id; |
|  | } |
|  |  |
|  | public String getNombre() { |
|  | return nombre; |
|  | } |
|  |  |
|  | public void setNombre(String nombre) { |
|  | this.nombre = nombre; |
|  | } |
|  |  |
|  | public String getPassword() { |
|  | return password; |
|  | } |
|  |  |
|  | public void setPassword(String password) { |
|  | this.password = password; |
|  | } |
|  |  |
|  | public Set<Rol> getRoles() { |
|  | return roles; |
|  | } |
|  |  |
|  | public void setRoles(Set<Rol> roles) { |
|  | this.roles = roles; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/8f04c49672284715727b813e013a2177/raw/10fc7835d937dd3fc1d57a919dd461d197ffc9ea/usuario.java)[**usuario.java**](https://gist.github.com/cavanosa/8f04c49672284715727b813e013a2177#file-usuario-java) hosted with  by [**GitHub**](https://github.com/)

El atributo roles lleva la anotación @ManyToMany, lo que siginifica que un usuario puede tener varios roles y un rol puede pertenecer a varios usuarios.  
Con @JoinTable le estamos diciendo que se cree una tabla llamada “usuario\_rol” con dos campos, “usuario\_id”, que apunta al id de un usuario y “rol\_id”, que apunta al id de un rol.

La clase Rol:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.model; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.enums.RolNombre; |
|  |  |
|  | import javax.persistence.\*; |
|  | import javax.validation.constraints.NotNull; |
|  |  |
|  | @Entity |
|  | public class Rol { |
|  | @Id |
|  | @GeneratedValue(strategy = GenerationType.IDENTITY) |
|  | private long id; |
|  | @Enumerated(EnumType.STRING) |
|  | @NotNull |
|  | @Column(unique = true) |
|  | private RolNombre rolNombre; |
|  |  |
|  | public Rol() { |
|  | } |
|  |  |
|  | public Rol(@NotNull RolNombre rolNombre) { |
|  | this.rolNombre = rolNombre; |
|  | } |
|  |  |
|  | public long getId() { |
|  | return id; |
|  | } |
|  |  |
|  | public void setId(long id) { |
|  | this.id = id; |
|  | } |
|  |  |
|  | public RolNombre getRolNombre() { |
|  | return rolNombre; |
|  | } |
|  |  |
|  | public void setRolNombre(RolNombre rolNombre) { |
|  | this.rolNombre = rolNombre; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/3e0f2d235d08501c376145666d650b86/raw/4e1e6c54403b0aac2d5778e836cfec50e406b285/rol.java)[**rol.java**](https://gist.github.com/cavanosa/3e0f2d235d08501c376145666d650b86#file-rol-java) hosted with  by [**GitHub**](https://github.com/)

**Repositorio y Servicio:**

La interfaz UsuarioRepository:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.repository; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.model.Usuario; |
|  | import org.springframework.data.jpa.repository.JpaRepository; |
|  | import org.springframework.stereotype.Repository; |
|  |  |
|  | import java.util.Optional; |
|  |  |
|  | @Repository |
|  | public interface UsuarioRepository extends JpaRepository<Usuario, Long> { |
|  | boolean existsByNombre(String nombre); |
|  | Optional<Usuario> findByNombre(String nombre); |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/9e0f49c52ef3df7d6520327ce8ccbb41/raw/628e60a7939dcf1cd5cd76d5271581188560a958/usuariorepository.java)[**usuariorepository.java**](https://gist.github.com/cavanosa/9e0f49c52ef3df7d6520327ce8ccbb41#file-usuariorepository-java) hosted with  by [**GitHub**](https://github.com/)

UsuarioService:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.service; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.model.Usuario; |
|  | import com.inezpre5.thymeleaf\_security.repository.UsuarioRepository; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.stereotype.Service; |
|  |  |
|  | import java.util.List; |
|  | import java.util.Optional; |
|  |  |
|  | @Service |
|  | public class UsuarioService { |
|  | @Autowired |
|  | UsuarioRepository usuarioRepository; |
|  |  |
|  | public List<Usuario> getList(){ |
|  | return usuarioRepository.findAll(); |
|  | } |
|  |  |
|  | public Optional<Usuario> getById(long id){ |
|  | return usuarioRepository.findById(id); |
|  | } |
|  |  |
|  | public Optional<Usuario> getByNombre(String nombre){ |
|  | return usuarioRepository.findByNombre(nombre); |
|  | } |
|  |  |
|  | public void save(Usuario usuario){ |
|  | usuarioRepository.save(usuario); |
|  | } |
|  |  |
|  | public boolean existsUsuarioId(long id){ |
|  | return usuarioRepository.existsById(id); |
|  | } |
|  |  |
|  | public boolean existsUsuarioNombre(String nombre){ |
|  | return usuarioRepository.existsByNombre(nombre); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/8fafee117b034f64c1787d16d6acac42/raw/40d959d592aaa182f8f45165429971b2bad5e693/usuarioservice.java)[**usuarioservice.java**](https://gist.github.com/cavanosa/8fafee117b034f64c1787d16d6acac42#file-usuarioservice-java) hosted with  by [**GitHub**](https://github.com/)

RolRepository:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.repository; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.enums.RolNombre; |
|  | import com.inezpre5.thymeleaf\_security.model.Rol; |
|  | import org.springframework.data.jpa.repository.JpaRepository; |
|  | import org.springframework.stereotype.Repository; |
|  |  |
|  | import java.util.Optional; |
|  |  |
|  | @Repository |
|  | public interface RolRepository extends JpaRepository<Rol, Long> { |
|  | boolean existsByRolNombre(RolNombre rolNombre); |
|  | Optional<Rol> findByRolNombre(RolNombre rolNombre); |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/104e23dc29f180a62ad65c9801a4c425/raw/a88717e4f65c045a074f5e617c3d8e381a749b11/rolrepository.java)[**rolrepository.java**](https://gist.github.com/cavanosa/104e23dc29f180a62ad65c9801a4c425#file-rolrepository-java) hosted with  by [**GitHub**](https://github.com/)

RolService:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.service; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.enums.RolNombre; |
|  | import com.inezpre5.thymeleaf\_security.model.Rol; |
|  | import com.inezpre5.thymeleaf\_security.repository.RolRepository; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.stereotype.Service; |
|  |  |
|  | import java.util.Optional; |
|  |  |
|  | @Service |
|  | public class RolService { |
|  |  |
|  | @Autowired |
|  | RolRepository rolRepository; |
|  |  |
|  | public Optional<Rol> getByRolNombre(RolNombre rolNombre){ |
|  | return rolRepository.findByRolNombre(rolNombre); |
|  | } |
|  |  |
|  | public boolean existsByRolNombre(RolNombre rolNombre){ |
|  | return rolRepository.existsByRolNombre(rolNombre); |
|  | } |
|  |  |
|  | public void save(Rol rol){ |
|  | rolRepository.save(rol); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/d8e3eea9a16377249322200606ae5103/raw/ec75cc6cae0474004b32031ac95db1772bba3faa/rolservice.java)[**rolservice.java**](https://gist.github.com/cavanosa/d8e3eea9a16377249322200606ae5103#file-rolservice-java) hosted with  by [**GitHub**](https://github.com/)

**UsuarioPrincipal y UsuarioDetailsServiceImpl:**

Estas dos clases, como dijimos, son propias de Spring Security (bueno, para ser exactos son nuestras, pero implementan interfaces de Spring Security). Dentro del paquete security creamos otro paquete llamado service en el que crearemos estas dos clases.

UsuarioPrincipal es como sigue:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.security.service; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.model.Usuario; |
|  | import org.springframework.security.core.GrantedAuthority; |
|  | import org.springframework.security.core.authority.SimpleGrantedAuthority; |
|  | import org.springframework.security.core.userdetails.UserDetails; |
|  |  |
|  | import java.util.Collection; |
|  | import java.util.List; |
|  | import java.util.stream.Collectors; |
|  |  |
|  | public class UsuarioPrincipal implements UserDetails { |
|  |  |
|  | private long id; |
|  | private String nombre; |
|  | private String password; |
|  | private Collection<? extends GrantedAuthority> authorities; |
|  |  |
|  | public UsuarioPrincipal(long id, String nombre, String password, Collection<? extends GrantedAuthority> authorities) { |
|  | this.id = id; |
|  | this.nombre = nombre; |
|  | this.password = password; |
|  | this.authorities = authorities; |
|  | } |
|  |  |
|  | public static UsuarioPrincipal build(Usuario usuario){ |
|  | List<GrantedAuthority> authorities= |
|  | usuario.getRoles().stream().map(rol -> new SimpleGrantedAuthority(rol.getRolNombre().name())).collect(Collectors.toList()); |
|  | return new UsuarioPrincipal(usuario.getId(), usuario.getNombre(), usuario.getPassword(), authorities); |
|  | } |
|  |  |
|  | public long getId() { |
|  | return id; |
|  | } |
|  |  |
|  | @Override |
|  | public Collection<? extends GrantedAuthority> getAuthorities() { |
|  | return authorities; |
|  | } |
|  |  |
|  | @Override |
|  | public String getPassword() { |
|  | return password; |
|  | } |
|  |  |
|  | @Override |
|  | public String getUsername() { |
|  | return nombre; |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isAccountNonExpired() { |
|  | return true; |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isAccountNonLocked() { |
|  | return true; |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isCredentialsNonExpired() { |
|  | return true; |
|  | } |
|  |  |
|  | @Override |
|  | public boolean isEnabled() { |
|  | return true; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/1b0ad1f5af5fbebd6f4e220983e1e675/raw/fc6de36b449460373ba6a2c612e2a2ac54f3ddb5/usuarioprincipal.java)[**usuarioprincipal.java**](https://gist.github.com/cavanosa/1b0ad1f5af5fbebd6f4e220983e1e675#file-usuarioprincipal-java) hosted with  by [**GitHub**](https://github.com/)

Esta clase guarda información sobre el usuario, que será utilizada para autenticarse.

La siguiente clase, UsuarioDetailsServiceImpl es la que obtendrá los datos del usuario de la base de datos:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.security.service; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.model.Usuario; |
|  | import com.inezpre5.thymeleaf\_security.service.UsuarioService; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.security.core.userdetails.UserDetails; |
|  | import org.springframework.security.core.userdetails.UserDetailsService; |
|  | import org.springframework.security.core.userdetails.UsernameNotFoundException; |
|  |  |
|  | @Service |
|  | public class UsuarioDetailsServiceImpl implements UserDetailsService { |
|  |  |
|  | @Autowired |
|  | UsuarioService usuarioService; |
|  |  |
|  | @Override |
|  | public UserDetails loadUserByUsername(String nombre) throws UsernameNotFoundException { |
|  | Usuario usuario = usuarioService.getByNombre(nombre).orElseThrow(() -> new UsernameNotFoundException(nombre)); |
|  | return UsuarioPrincipal.build(usuario); |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/6efe522588338e437610ef6cfdd1fe62/raw/e3c77a8b2322abdcfdd26844d43ecd44622a6101/usuariodetailsserviceimpl.java)[**usuariodetailsserviceimpl.java**](https://gist.github.com/cavanosa/6efe522588338e437610ef6cfdd1fe62#file-usuariodetailsserviceimpl-java) hosted with  by [**GitHub**](https://github.com/)

**Controlador principal (home):**

Nos queda por implementar el controlador de las vistas index, login y forbidden. Al index accederemos tanto a partir de la ruta raíz como a partir de la ruta index:

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.controller; |
|  |  |
|  | import org.springframework.stereotype.Controller; |
|  | import org.springframework.web.bind.annotation.GetMapping; |
|  | import org.springframework.web.bind.annotation.RequestMapping; |
|  |  |
|  | @Controller |
|  | @RequestMapping("/") |
|  | public class HomeController { |
|  |  |
|  | @GetMapping(value = {"", "index"}) |
|  | public String index(){ |
|  | return "index"; |
|  | } |
|  |  |
|  | @GetMapping("/login") |
|  | public String login(){ |
|  | return "login"; |
|  | } |
|  |  |
|  | @GetMapping("/forbidden") |
|  | public String forbidden(){ |
|  | return "forbidden"; |
|  | } |
|  |  |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/4f8a30fabbcba765ea963cc95cdc47d8/raw/b9a471fc3787ad14c807fd5521984d24d9af013c/homecontroller.java)[**homecontroller.java**](https://gist.github.com/cavanosa/4f8a30fabbcba765ea963cc95cdc47d8#file-homecontroller-java) hosted with  by [**GitHub**](https://github.com/)

**Controller:**

Para finalizar este post vamos con la clase UsuarioController, donde se crearán nuevos usuarios.

|  |  |
| --- | --- |
|  | package com.inezpre5.thymeleaf\_security.controller; |
|  |  |
|  | import com.inezpre5.thymeleaf\_security.enums.RolNombre; |
|  | import com.inezpre5.thymeleaf\_security.model.Rol; |
|  | import com.inezpre5.thymeleaf\_security.model.Usuario; |
|  | import com.inezpre5.thymeleaf\_security.service.RolService; |
|  | import com.inezpre5.thymeleaf\_security.service.UsuarioService; |
|  | import org.apache.commons.lang3.StringUtils; |
|  | import org.springframework.beans.factory.annotation.Autowired; |
|  | import org.springframework.security.crypto.password.PasswordEncoder; |
|  | import org.springframework.stereotype.Controller; |
|  | import org.springframework.web.bind.annotation.GetMapping; |
|  | import org.springframework.web.bind.annotation.PostMapping; |
|  | import org.springframework.web.bind.annotation.RequestMapping; |
|  | import org.springframework.web.servlet.ModelAndView; |
|  |  |
|  | import java.util.HashSet; |
|  | import java.util.Set; |
|  |  |
|  | @Controller |
|  | @RequestMapping("/usuario") |
|  | public class UsuarioController { |
|  |  |
|  | @Autowired |
|  | UsuarioService usuarioService; |
|  |  |
|  | @Autowired |
|  | RolService rolService; |
|  |  |
|  | @Autowired |
|  | PasswordEncoder passwordEncoder; |
|  |  |
|  | @GetMapping("/registro") |
|  | public String nuevo(){ |
|  | return "registro"; |
|  | } |
|  |  |
|  | @PostMapping("/registrar") |
|  | public ModelAndView registrar(String nombre, String password){ |
|  | ModelAndView mv = new ModelAndView(); |
|  | if(StringUtils.isBlank(nombre)){ |
|  | mv.addObject("error", "ese nombre no es válido"); |
|  | mv.setViewName("registro"); |
|  | return mv; |
|  | } |
|  | if(StringUtils.isBlank(password)){ |
|  | mv.addObject("error", "esa contraseña no es válida"); |
|  | mv.setViewName("registro"); |
|  | return mv; |
|  | } |
|  | if(usuarioService.existsUsuarioNombre(nombre)){ |
|  | mv.addObject("error", "ese nombre ya existe"); |
|  | mv.setViewName("registro"); |
|  | return mv; |
|  | } |
|  | String passwordEnc = passwordEncoder.encode(password); |
|  | Usuario usuario = new Usuario(nombre, passwordEnc); |
|  | Rol rol = rolService.getByRolNombre(RolNombre.ROLE\_USER).get(); |
|  | Set<Rol> roles = new HashSet<>(); |
|  | roles.add(rol); |
|  | usuario.setRoles(roles); |
|  | usuarioService.save(usuario); |
|  | mv.addObject("registroOK", " Cuenta creada, ingresa tus credenciales para iniciar sesión"); |
|  | mv.setViewName("login"); |
|  | return mv; |
|  | } |
|  | } |

[**view raw**](https://gist.github.com/cavanosa/5cbac7cbb210ddf9786856747151757e/raw/0bfa1215df895493f52921a7a3ef0a82944da51f/usuariocontroller.java)[**usuariocontroller.java**](https://gist.github.com/cavanosa/5cbac7cbb210ddf9786856747151757e#file-usuariocontroller-java) hosted with  by [**GitHub**](https://github.com/)

El primer método (nuevo()) devuelve la vista del formulario de registro. El dsiguiente método, registrar, es el que se encarga de crear y guardar al nuevo usuario. Comprueba como siempre que los campos sean correctos y, en caso afirmativo, se llamará al servicio para que lo guarde. La contraseña se cifra con el passwordEncoder. También se le añade un mensaje al ModelAndView para que la vista del login -que es la que va a devolver- informe al usuario de que la cuenta se ha creado con éxito. El rol va a ser de usuario pues el administrador lo crearemos nosotros más adelante.

Y aquí concluye el post, en el siguiente crearemos la clase que controla la seguridad de la aplicación.