Gmail **Exequiel Guzmán <guzman.exe@gmail.com>**

implementar 3 intentos

1 mensaje

Andrés Guzmán <aguzman@bolsadeideas.cl> 25 de junio de 2021, 18:32
Para: Exequiel Guzmán <guzman.exe@gmail.com>

Paso 1: Tener los atributos enabled e intentos en el entity y en la tabla de usuarios de la bbdd:

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

...

    private Boolean enabled;
    private Integer intentos;

        public Integer getIntentos() {
            return intentos;
        }

        public void setIntentos(Integer intentos) {
            this.intentos = intentos;
        }

        public Boolean getEnabled() {
            return enabled;
        }

        public void setEnabled(Boolean enabled) {
            this.enabled = enabled;
        }
```

==========
==========

Paso 2: Agregar el metodo findByUsername y save en el service JpaUserDetailsService:

```
@Service("jpaUserDetailsService")
public class JpaUserDetailsService implements UserDetailsService{

	@Autowired
	private IUsuarioDao usuarioDao;

	private Logger logger = LoggerFactory.getLogger(JpaUserDetailsService.class);

	@Override
	@Transactional(readOnly=true)
	public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

	Usuario usuario = usuarioDao.findByUsername(username);

	if(usuario == null) {
		logger.error("Error en el Login: no existe el usuario '" + username + "' en el sistema!");
		throw new UsernameNotFoundException("Username: " + username + " no existe en el sistema!");
	}

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

	for(Role role: usuario.getRoles()) {
		logger.info("Role: ".concat(role.getAuthority()));
		authorities.add(new SimpleGrantedAuthority(role.getAuthority()));
	}

	if(authorities.isEmpty()) {
		logger.error("Error en el Login: Usuario '" + username + "' no tiene roles asignados!");
		throw new UsernameNotFoundException("Error en el Login: usuario '" + username + "' no tiene roles
asignados!");
```

```
	}


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


		public Usuario findByUsername(String username) {
			return usuarioDao.findByUsername(username);
		}


		public Usuario save(Usuario usuario) {
			return usuarioDao.save(usuario);
		}


}
```

==========
==========

Paso 3: Crear la clase AuthenticationSuccessErrorHandler anotada con @Component y debe implementar la interface AuthenticationEventPublisher (dentro del package base del proyecto (com.bolsadeideas.springboot.app) ejemplo com.bolsadeideas.springboot.app.security.event):

```
package com.bolsadeideas.springboot.app.security.event;


import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.AuthenticationEventPublisher;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.stereotype.Component;
import com.bolsadeideas.springboot.app.models.service.JpaUserDetailsService;
import com.bolsadeideas.springboot.app.models.entity.Usuario;


@Component
```

```
public class AuthenticationSuccessErrorHandler implements AuthenticationEventPublisher {

	@Autowired
	private JpaUserDetailsService usuarioService;



	@Override
	public void publishAuthenticationSuccess(Authentication authentication) {

		Usuario usuario = usuarioService.findByUsername(authentication.getName());

		if(usuario.getIntentos() != null && usuario.getIntentos() > 0) {
			usuario.setIntentos(0);
			usuarioService.save(usuario);
		}
	}

	@Override
	public void publishAuthenticationFailure(AuthenticationException exception, Authentication authentication) {

		Usuario usuario = usuarioService.findByUsername(authentication.getName());

		if(usuario == null) {
	    throw new UsernameNotFoundException("Username: " + authentication.getName() + " no existe en el sistema!");
	}

		if (usuario.getIntentos() == null) {
			usuario.setIntentos(0);
		}

		usuario.setIntentos(usuario.getIntentos()+1);

		if(usuario.getIntentos() >= 3) {
```

```
                usuario.setEnabled(false);
            }

            usuarioService.save(usuario);

        }

}
```

==========
==========

Paso 4: En la clase SpringSecurityConfig inyectar el AuthenticationEventPublisher con @Autowired y registrarlo en el configurerGlobal

```
@EnableGlobalMethodSecurity(securedEnabled=true, prePostEnabled=true)
@Configuration
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter{

    @Autowired
    private AuthenticationEventPublisher eventPublisher;

...

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

...
```

---

implementar 3 intentos.txt
5K