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

Codigo completo: Z15-spring-boot-data-jpa-security-login-attempts

Andrés Guzmán <aguzman@bolsadeideas.cl> 28 de junio de 2021, 11:58
Para: Exequiel Guzmán <guzman.exe@gmail.com>

Paso 1 : mover el desbloqueado a la clase JpaUserDetailsService

....

```
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");

}


if(usuario.getLockoutEnd() != null && usuario.getEnabled() == false && usuario.getIntentos() >= AuthenticationSuccessErrorHandler.MAX_INTENTOS) {

Date fechaActual = new Date();

Date fechaBloqueo = usuario.getLockoutEnd();


if(fechaBloqueo.before(fechaActual)) {

usuario.setEnabled(true);

usuario.setIntentos(0);

usuario.setLockoutEnd(null);

usuarioDao.save(usuario);

logger.info("Usuario '" + username + "' ha sido habilitado nuevamente (desbloqueado) en el sistema");

}

}
```

...

Paso 2: y eliminarlo de AuthenticationSuccessErrorHandler, Cuidado con el **usuarioService.save(usuario);** debe ir al final del método y antes de lanzar la excepción,

quedando así:

```
@Component
public class AuthenticationSuccessErrorHandler implements AuthenticationEventPublisher {

	public final static Integer MAX_INTENTOS = 3;
	public final static Integer MINUTOS = 1;

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

	@Autowired
	private JpaUserDetailsService usuarioService;

	@Autowired
	private BCryptPasswordEncoder passwordEncoder;

	@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) {
			logger.error("Error en el Login: No existe el usuario '" + authentication.getName() + "' en
el sistema");
```

```
            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() == MAX_INTENTOS) {
                        //
                        Date fechaActual = new Date();
                        Calendar calendar = Calendar.getInstance();
                         calendar.setTime(fechaActual);
                        calendar.add(Calendar.MINUTE, MINUTOS);
                        Date fechaFutura = calendar.getTime();

                        usuario.setEnabled(false);
                        usuario.setLockoutEnd(fechaFutura);
                        usuarioService.save(usuario);

                        logger.error("Error en el Login: Usuario '" + authentication.getName() + "' ha sido bloqueado (deshabilitado) en el sistema");
            throw new DesbloqueadoException("Error en el Login: Usuario '" + authentication.getName() + "' ha sido bloqueado (deshabilitado) en el sistema.");
                    }

                    usuarioService.save(usuario);

            }

}
```

Paso 3: crear clase de exception llamada cl.excelsium.springboot.app.errors.DesbloqueadoException

```
package cl.excelsium.springboot.app.errors;
import org.springframework.security.core.AuthenticationException
public class DesbloqueadoException extends AuthenticationException{

	public DesbloqueadoException(String message) {
		super(message);
	}
}
```

Paso 4: crear o modificar la clase LoginFailedHandler anotada con @Component para el manejo de errores:

```
@Component
public class LoginFailedHandler extends SimpleUrlAuthenticationFailureHandler {

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

	@Autowired
	private IUsuarioDao usuarioDao;

	@Override
	public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,
			AuthenticationException exception) throws IOException, ServletException {

		SessionFlashMapManager flashMapManager = new SessionFlashMapManager();
		FlashMap flashMap = new FlashMap();

		if (exception instanceof DesbloqueadoException) {

			String username = request.getParameter("username");
			Usuario usuario = usuarioDao.findByUsername(username);
			String mensaje = "El Usuario: '" + username + "' se encuentra bloqueado hasta '"
					+ usuario.getLockoutEnd() + "'";
```

```
                    flashMap.put("warning", mensaje);
                    logger.error(mensaje);

                } else if(exception instanceof BadCredentialsException){
                    flashMap.put("error",  "Error en el login: Nombre de Usuario o Contraseña incorrecta, por favor vuelva a intentarlo!");
                }

                flashMapManager.saveOutputFlashMap(flashMap, request, response);
                super.onAuthenticationFailure(request, response, exception);

        }

}
```

Paso 5: registrar la clase LoginFailedHandler en SpringSecurityConfig:

```
        @Override
        protected void configure(HttpSecurity http) throws Exception {
                ...

                .and()
                        .formLogin()
                                .successHandler(successHandler)
                                .failureHandler(failedHandler)
                                .loginPage("/login")

                ...
        }
```