GUIDE II POUR LA RÉALISATION DU MINI PROJET "PFE" EN SPRING SECURITY

I. Utilisation de la méthode "*inMemoryAuthentication*"

1. Mise à jour du fichier pom.xml par les dépendences Spring-Security 4.0.1

```
<!-- Spring Security -->
<dependency>
	<groupId>org.springframework.security</groupId>
	<artifactId>spring-security-web</artifactId>
	<version>${spring.version}</version>
</dependency>
<dependency>
	<groupId>org.springframework.security</groupId>
	<artifactId>spring-security-config</artifactId>
	<version>${spring.version}</version>
</dependency>
```

2. Ajouter le filtre "DelegatingFilterProxy" dans le web.xml

```
<filter>
	<filter-name>springSecurityFilterChain</filter-name>
<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
	</filter>
<filter-mapping>
	<filter-name>springSecurityFilterChain</filter-name>
	<url-pattern>/*</url-pattern>
</filter-mapping>
```

3. **Créer une classe fille de "WebSecurityConfigurerAdapter" appeler la classe par exemple "SpringSecurityConfiguration" dans la couche "presentation"**
4. **Annoter la classe "SpringSecurityConfiguration" par les deux annotations suivantes: @Configuration et @EnableWebSecurity**

```
package presentation.controllers;

import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationMan
agerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
```

```
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;

@Configuration
@EnableWebSecurity
public class SpringSecurityConfiguration extends WebSecurityConfigurerAdapter{


}
```

5. Ajouter les deux méthodes suivantes à la classe SpringSecurityConfiguration

```
@Autowired
public void configureGlobalSecurity(AuthenticationManagerBuilder auth) throws Exception
{
auth.inMemoryAuthentication().withUser("user").password("abc123").roles("USER");
auth.inMemoryAuthentication().withUser("valid").password("root123").roles("VALID");
auth.inMemoryAuthentication().withUser("admin").password("root123").roles("ADMIN");
}

@Override
protected void configure(HttpSecurity http) throws Exception {
http.authorizeRequests().antMatchers("/", "/account/").permitAll().antMatchers("/crud")
.access("hasRole('USER')").antMatchers("/account/edit/**").access("hasRole('VALID')")
.antMatchers("/account/remove/**").access("hasRole('ADMIN')").and().formLogin().and()
.exceptionHandling().accessDeniedPage("/user/Access_Denied");
}
```

6. Créer “SpringSecurityController”

```
package presentation.controllers;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.logout.SecurityContextLogoutHandler;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/user")
public class SpringSecurityController {

        @RequestMapping(value = "/logout", method = RequestMethod.GET)
        public String logoutPage(HttpServletRequest request, HttpServletResponse response) {
                Authentication auth = SecurityContextHolder.getContext().getAuthentication();
```

```
                if (auth != null) {
                        new SecurityContextLogoutHandler().logout(request, response, auth);
                }
                return "redirect:/login";
        }
        @RequestMapping(value = "/Access_Denied", method = RequestMethod.GET)
        public String accessDeniedPage(ModelMap model) {
                model.addAttribute("user", getPrincipal());
                return "accessDenied";
        }

        private String getPrincipal() {
                String userName = null;
                Object principal =
SecurityContextHolder.getContext().getAuthentication().getPrincipal();
                if (principal instanceof UserDetails) {
                        userName = ((UserDetails) principal).getUsername();
                } else {
                        userName = principal.toString();
                }
                return userName;
        }
}
```

7. **Créer la page "WEB-INF/views/accessDenied.jsp"**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
   <title>AccessDenied page</title>
</head>
<body>
   Dear <strong>${user}</strong>, You are not authorized to access this page
   <a href="<c:url value="/user/logout" />">Logout</a>
</body>
</html>
```

II. Utilisation "*UserDetails*" Pour récupérer l'utilisateur et ses rôles à partir de la base.

Supposons que l'utilisateur peut avoir plusieurs rôles et qu'un rôle peut être affecté à plusieurs utilisateurs [Nous parlons d'une association de type MANY-TO-MANY entre la classe User et la classe Role]

8. **Créer la classe "Role" définie par l'id [Long, Primary Key] et name [String] avec les annotations de persistance nécessaires.**
9. **Créer l'attribut "roles" de type Set<Role> dans la classe User. Utiliser l'annotation @Many-to-Many**

10. Considerer username Primary Key dans la classe User. ci-après la classe User et la classe Role.

La Classe Role

```
package moldels;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
@Entity
public class Role {
	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private Long id;
	private String name;
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
}
```

La Classe User

```
package moldels;

import java.util.Set;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class User {
   @Id
   private String username;

   @ManyToMany
   private Set<Role> roles;
   private String password;
   public Set<Role> getRoles() {
		return roles;
	}
	public void setRoles(Set<Role> roles) {
```

```
		this.roles = roles;
	}
	public String getUsername() {
		return username;
	}
	public void setUsername(String username) {
		this.username = username;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	@Override
	public String toString() {
		return "User [username=" + username + ", roles=" + roles + ", password=" +
password + "]";
	}
 }
```

11. Créer les classes DAO pour l'entity User et créer la méthode getUser(String name)

L'interface IUserDao

```
package dao;

import moldels.User;
public interface IUserDao {
	public User getUser(String username);
}
```

L'implémentation "UserDaoImpl"

```
package dao;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

import org.springframework.stereotype.Repository;

import moldels.User;

@Repository
public class UserDaoImpl implements IUserDao {

	EntityManagerFactory emf = Persistence.createEntityManagerFactory("mysql_db");
	EntityManager em = emf.createEntityManager();

	@Override
	public User getUser(String username) {
		try {
			User u= (User) em.find(User.class, username);
			System.out.println("Call getUser from UserDaoImpl...."+u);
			return u;
```

```
                } catch (Exception ex) {
                        return null;// should have proper handling of Exception
                }
        }
}
```

12. Créer la classe Service "UserAuthServiceImpl". Cette classe doit implémenter l'interface Spring

org.springframework.security.core.userdetails.UserDetailsService

```
package service;

import java.util.HashSet;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import dao.IUserDao;
import moldels.Role;
import moldels.User;

@Service
public class UserAuthServiceImpl implements UserDetailsService {

        @Autowired
        private IUserDao userDao;

        @Override
        public UserDetails loadUserByUsername(String username)
                        throws UsernameNotFoundException {
                System.out.println("Call UserAuthService loadUserByUsername....." +
username);
                // load user
                final User user = userDao.getUser(username);

                if (user == null) {
                        throw new UsernameNotFoundException("Invalid User");
                }
                Set<GrantedAuthority> auths = new HashSet<GrantedAuthority>();
                for (Role r : user.getRoles())
                        auths.add(new SimpleGrantedAuthority("ROLE_" + r.getName()));

                org.springframework.security.core.userdetails.User ud;
ud= new org.springframework.security.core.userdetails.User(
                                        user.getUsername(), user.getPassword(), auths);
                return ud;
        }
}
```

13. Modifier les méthode de la classe SpringSecurityConfiguration en utilisant le service précédemment créé.

A modifier dans la classe "SpringSecurityConfiguration"

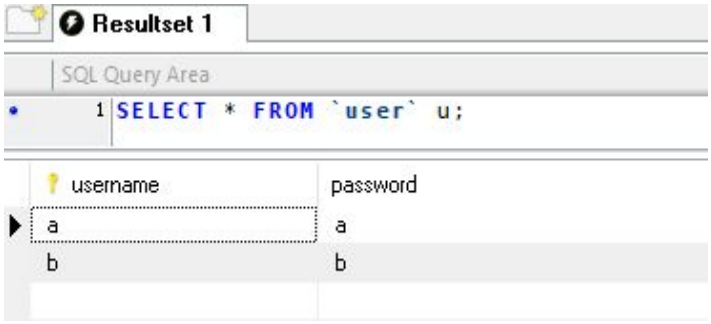
```
// Ajouter cet attribut….
@Autowired
private UserAuthServiceImpl userAuthService;

// Modifier cette méthode...
@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
auth.userDetailsService(userAuthService);
}
```

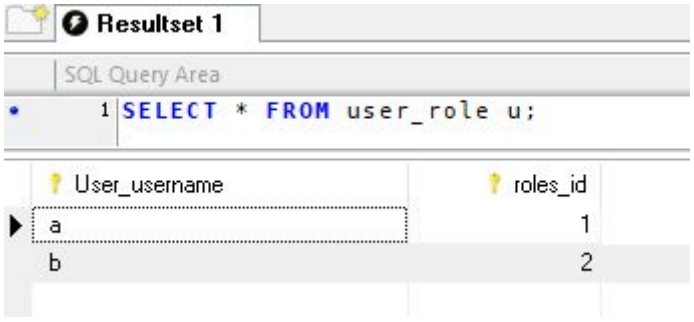
14. Pour tester, ajouter les données suivantes dans la base.

ROLE TABLE

USER TABLE

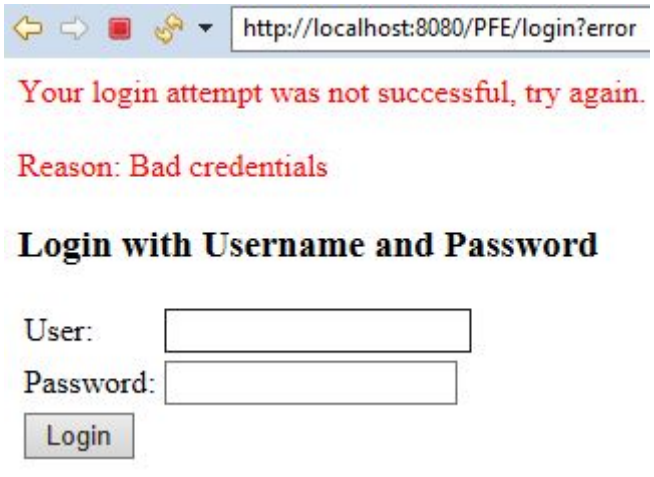


ROLES_USER TABLE



15. Déployer votre application sur Tomcat. Vérifier que le formulaire d'authentification est donné par Spring MVC via l'URL (/Project/login)

Accès à l'URL (/Project/login)

En cas d'erreur du login ou mot de passe

En cas d'accès à une action sans avoir les privilèges nécessaires.