Atelier 7 Spring Security : Les fondamentaux

Dans cet atelier on va ajouter la couche sécurité à notre application de gestion des produits en appliquant les règles de gestion suivantes :

- Toutes les opérations de l'application nécessitent une authentification,
- Les utilisateurs de l'application peuvent avoir les rôles suivants :
  - ***ADMIN*** : a le droit de faire toutes les opérations,
  - ***AGENT*** : a seulement le droit de consulter ou d'ajouter les produits, il ne peut ni les modifier ni les supprimer,
  - ***USER*** : a seulement le droit de consulter les produits.

Remarque :

Pour des raisons pédagogiques on s'intéresse ici seulement à l'entité Produit.

Objectifs :

1. Ajouter Spring Security au projet,
2. L'authentification basique,
3. L'authentification en utilisant des utilisateurs InMemory,
4. Sécuriser l'accès à l'application selon les rôles,
5. Ajouter un Password Encoder,
6. Contextualisation du menu selon l'utilisateur connecté.

Ajouter Spring Security au projet

1. Ajouter la dépendance Spring security au fichier pom.xml :

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

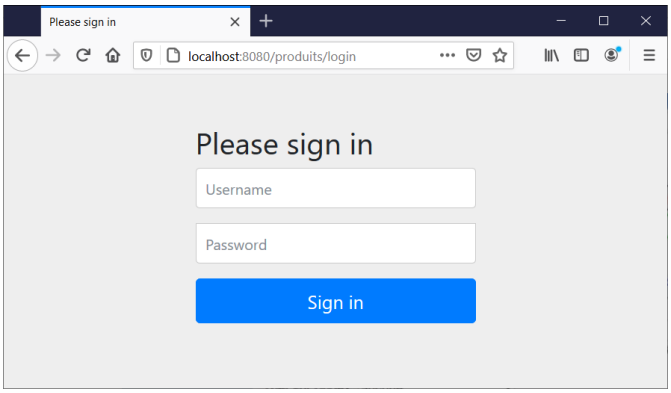
2. Créer dans le dossier templates le fichier index.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorator="template">

<link rel="stylesheet" type="text/css"
href="/webjars/bootstrap/4.3.1/css/bootstrap.min.css" />
<head>
<meta charset="utf-8">
<title>Gestion des Produits</title>
</head>
<body>
```

```
 <div layout:fragment="Mycontent">
</div>
</body>
</html>
```

3. Redémarrer l'application et tester : http://localhost:8080/produits/



Connectez-vous en tant que user, le mot de passe vous le trouvez dans la console au démarrage de l'application :

Using generated security password: ac3149b7-6e19-42a3-947e-b26f81d29ff3

NB : Le mot de passe change à chaque démarrage.

L'authentification basique

Ajouter des utilisateurs dans le fichier application.properties

4. Editer le fichier application.properties et ajouter les lignes suivantes :

```
spring.security.user.name=admin
spring.security.user.password=123
```

5. puis testez, de nouveau le lien : http://localhost:8080/produits/

L'authentification en utilisant des utilisateurs InMemory

6. Créer la classe SecurityConfig, placez la dans le package security :

```
package com.nadhem.produits.security;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
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.WebSecurityConfigurerAdapter;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("admin").password("{noop}123").roles("ADMIN");
auth.inMemoryAuthentication().withUser("nadhem").password("{noop}123").roles("AGENT","USER");
auth.inMemoryAuthentication().withUser("user1").password("{noop}123").roles("USER");
```

```
    }

  @Override
  protected void configure(HttpSecurity http) throws Exception {
    http.authorizeRequests().anyRequest().authenticated();
   http.formLogin();

  }
}
```

7. Commentez les lignes suivantes dans le fichier appilication.properties

```
#spring.security.user.name=admin
#spring.security.user.password=123
```

8. Avec un navigateur, authentifiez-vous, puis testez, de nouveau le lien : http://localhost:8080/produits/

Sécuriser l'accès à l'application selon les rôles

On se propose ici d'autoriser l'accès aux méthodes de la classe ProduitController selon le rôle de l'utilisateur authentifié. Et ce en appliquant les règles de gestion annoncées au début de l'atelier :

- Les utilisateurs de l'application peuvent avoir les rôles suivants :
  - ***ADMIN*** : a le droit de faire toutes les opérations,
  - ***AGENT*** : a seulement le droit de consulter ou d'ajouter les produits, il ne peut ni les modifier ni les supprimer,
  - ***USER*** : a seulement le droit de consulter les produits.

9. Modifier la méthode `configure(HttpSecurity http)` comme suit :

```
@Override
  protected void configure(HttpSecurity http) throws Exception {
   http.authorizeRequests().antMatchers("/showCreate").hasAnyRole("ADMIN","AGENT");
   http.authorizeRequests().antMatchers("/saveProduit").hasAnyRole("ADMIN","AGENT");
   http.authorizeRequests().antMatchers("/ListeProduits")
                           .hasAnyRole("ADMIN","AGENT","USER");

   http.authorizeRequests()
            .antMatchers("/supprimerProduit","/modifierProduit","/updateProduit")
            .hasAnyRole("ADMIN");

   http.authorizeRequests().anyRequest().authenticated();
   http.formLogin();
  }
```

Personnaliser la page Access Denied

10. Créer la classe contrôlleur *SecurityController* dans le package *security* :

```
package com.nadhem.produits.security;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class SecurityController {
    @GetMapping("/accessDenied")
    public String error()
    {
          return "accessDenied";
    }

}
```

11. Créer la page accessDenied.html :

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorator="template">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
 <div layout:fragment="Mycontent">
<div class="container">
 <h3 class="text-danger">Vous n'êtes pas autorisé</h3>
</div>
</div>
</body>
</html>
```

12. Ajouter la ligne suivante à la méthode `configure(HttpSecurity http)`

```
http.exceptionHandling().accessDeniedPage("/accessDenied");
```

13. Tester votre travail

Ajouter un Password Encoder :

14. Ajouter, à la classe SecurityConfig, la méthode passwordEncoder :

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

15. Modifier la méthode configure :

```
@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
           PasswordEncoder passwordEncoder = passwordEncoder ();
```

```
                auth.inMemoryAuthentication().withUser("admin")
                      .password(passwordEncoder.encode("123")).roles("ADMIN");
                auth.inMemoryAuthentication().withUser("nadhem")
                      .password(passwordEncoder.encode("123")).roles("AGENT","USER");
                auth.inMemoryAuthentication().withUser("user1")
                      .password(passwordEncoder.encode("123")).roles("USER");
    }
```

Contextualisation du menu selon l'utilisateur connecté

16. Ajouter la dépendance qui permet à Thymeleaf de communiquer avec Spring Security :

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

17. Ajouter le name space xml suivant au fichier template.html

```
xmlns:sec=https://www.thymeleaf.org/thymeleaf-extras-springsecurity5
```

18. Modifier le fichier template.html comme suit :

```
<ul class="navbar-nav ml-auto" >
     <li class="nav-item dropdown">
      <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-
toggle="dropdown">
      <span sec:authentication="name"></span>
      </a>

  <div class="dropdown-menu">
      <a sec:authorize="!isAuthenticated()" class="dropdown-item"
th:href="@{login}">login</a>
        <a sec:authorize="isAuthenticated()" class="dropdown-item"
th:href="@{logout}">logout</a>
          <a class="dropdown-item" href="#">Profile</a>
      </div>
    </li>
  </ul>
```

Cacher les boutons Supprimer et Editer pour les utilisateurs non autorisés

19. Modifier le fichier listeProduits.html comme suit :

```
<td sec:authorize="hasRole('ADMIN')" ><a class="btn btn-danger"
onclick="return confirm('Etes-vous sûr ?')"
th:href="@{supprimerProduit(id=${p.idProduit},
page=${currentPage},size=${size})}">Supprimer</a>
</td>

<td sec:authorize="hasRole('ADMIN')"><a class="btn btn-success"
th:href="@{modifierProduit(id=${p.idProduit})}">Editer</a></td>
```

20. Modifier également le fichier template.html pour cacher le menu « Ajouter » aux utilisateurs n'ayant pas le rôle ADMIN, comme suit :

```
<a sec:authorize="hasRole('ADMIN')" class="dropdown-item"
th:href="@{showCreate}">Ajouter</a>
```