Configuration Spring Security

Démonstration 1 du module 7

Les objectifs de cette démonstration sont

- La mise en place de la sécurité
- Configuration d'utilisateurs locaux
- Configuration de restriction sur les URLs

Contexte

- Imaginons une application qui permet de gérer des formateurs et des cours.
- La source de l'application Spring est fournie.
- Il faut interdire l'accès à certaines URLs selon les utilisateurs connectés.

Déroulement

Utiliser la démonstration fournie

- Intégrer là à votre IDE

Configuration

- Ajout des starters pour Spring Security dans build.gradle :

```
//Starter Spring Security
implementation 'org.springframework.boot:spring-boot-starter-security'

//Starter Spring Security Thymeleaf
implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity5'

//Starter Spring Security Test
testImplementation 'org.springframework.security:spring-security-test'
```

  - Faire un « Refresh Gradle Project »

- Créer la classe de configuration SecurityConfig :

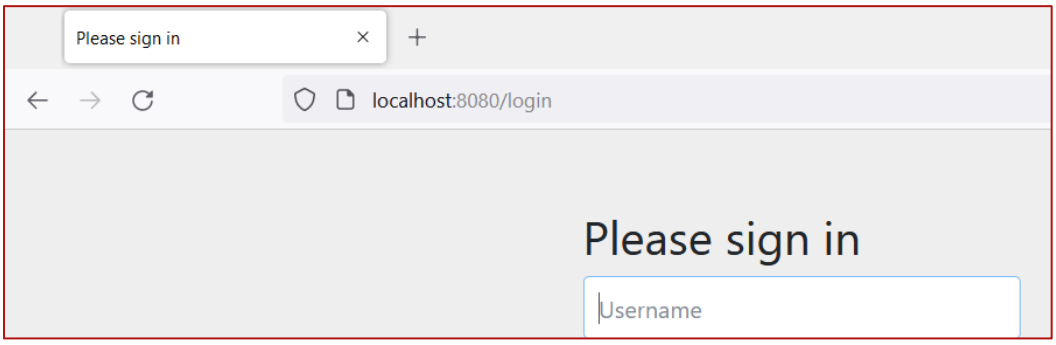
```
package fr.eni.demo.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
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;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.User.UserBuilder;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

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

		// add our users for in memory authentication
		UserBuilder users = User.builder();

		auth.inMemoryAuthentication().passwordEncoder(passwordEncoder())
		.withUser(users.username("abaille@campus-eni.fr").password(passwordEncoder().encode("Pa$$w0rd"))
					.roles("ADMIN", "TRAINER"))
		.withUser(users.username("sgobin@campus-eni.fr").password(passwordEncoder().encode("Pa$$w0rd"))
					.roles("TRAINER"))
		.withUser(users.username("jtrillard@campus-eni.fr").password(passwordEncoder().encode("Pa$$w0rd"))
					.roles("TRAINER"))
		.withUser(users.username("edoudard@campus-eni.fr").password(passwordEncoder().encode("Pa$$w0rd"))
					.roles("BUSINESS"));
	}

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

	@Override
	protected void configure(HttpSecurity httpSecurity) throws Exception {
		httpSecurity.authorizeRequests()
		.antMatchers("/trainers/detail*").hasAnyRole("ADMIN")
		.antMatchers("/trainers").hasAnyRole("TRAINER", "ADMIN")
		.antMatchers("/trainers/c*").hasAnyRole("ADMIN")
		.antMatchers("/resources/**").permitAll()
		.antMatchers("/**").permitAll()
		.antMatchers("/index").permitAll()
	.anyRequest()
	.authenticated()
	.and()
	.formLogin()
	.and()
	.logout()
	.logoutSuccessUrl("/");
	}
}
```

- Elle déclare les 4 utilisateurs qui peuvent être reconnus
- Elle leur donne des rôles (1 ou plusieurs)
- Elle permet de chiffrer le mot de passe
- Et elle renseigne les URL et les rôles associés

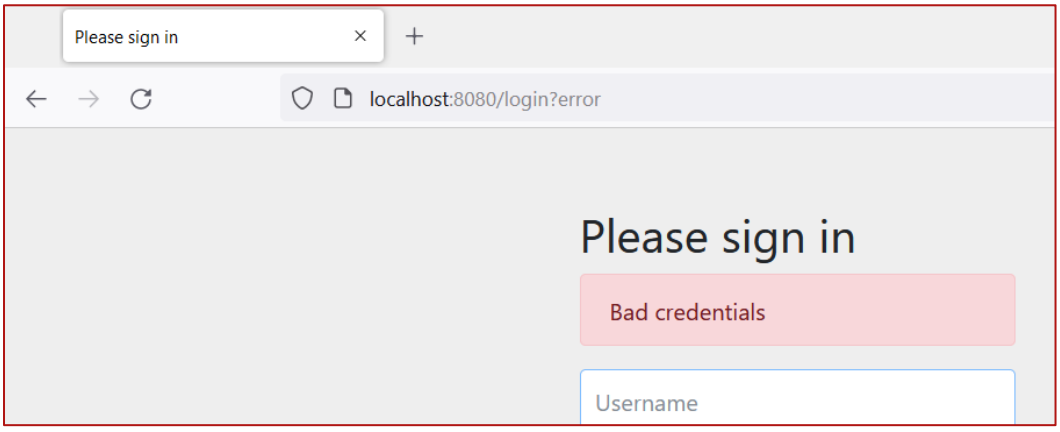
Exécution :

- Par défaut Spring fourni une page de connexion avec le module Spring Security.
  - Lancer l'application.
  - Les ressources à la racine du projet ou dans des URLs différentes de « /trainers » sont accessibles par tout le monde.
- La page index.html est accessible :

- Pour accéder à la liste des formateurs, il faut au moins être connecté et avoir un rôle (soit TRAINER soit ADMIN)
  - Spring Security nous redirige vers la page de connexion :

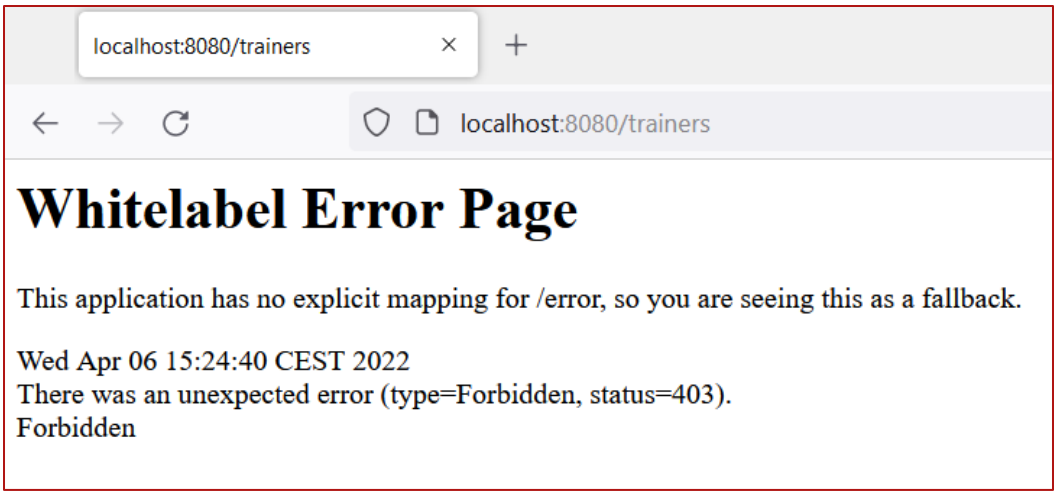


- Si nous essayons de nous connecté avec un compte inconnu :
  - Un message d'erreur s'affiche :

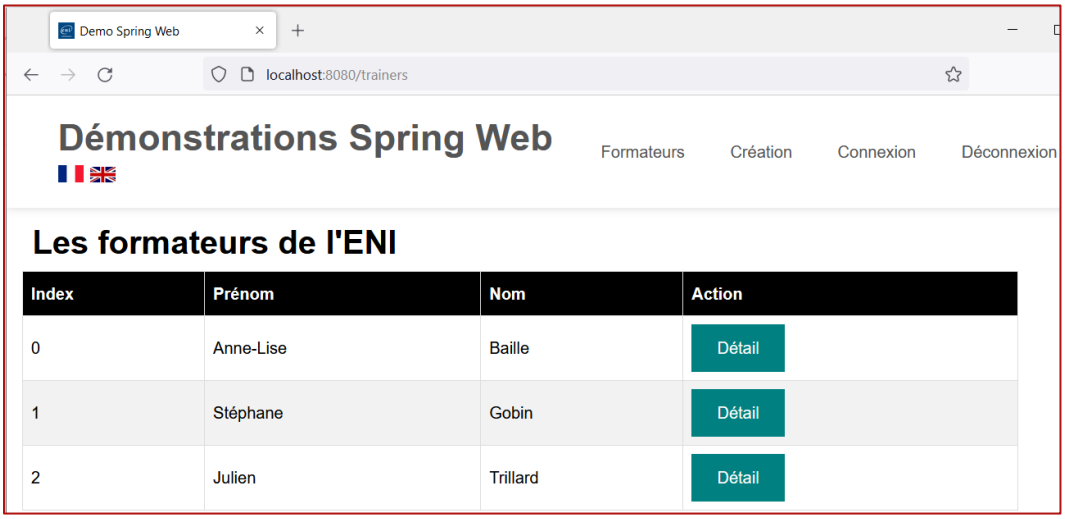


- Si nous nous connectons avec l'utilisateur : `edoudard@campus-eni.fr`

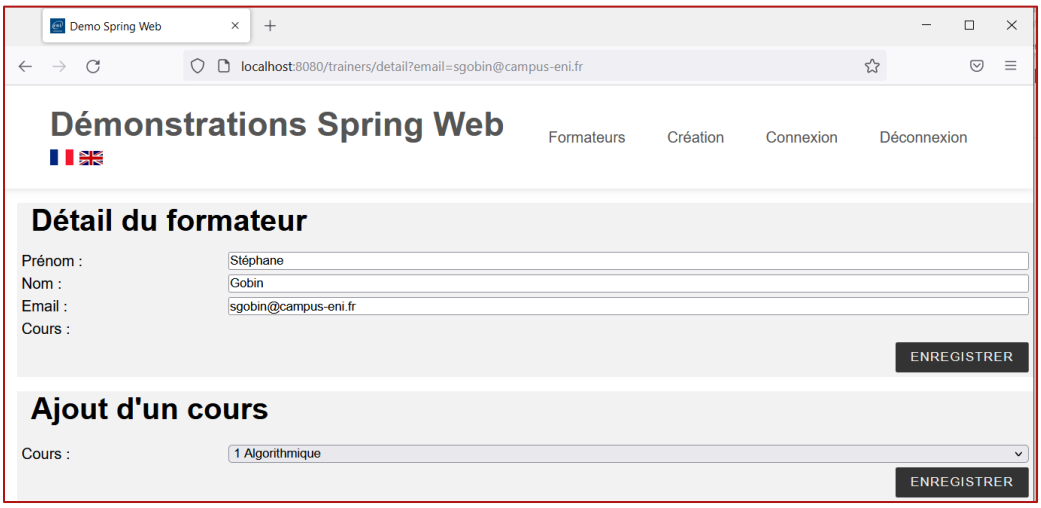
- Il nous informe que nous n'avons pas le droit d'accéder à cette ressource

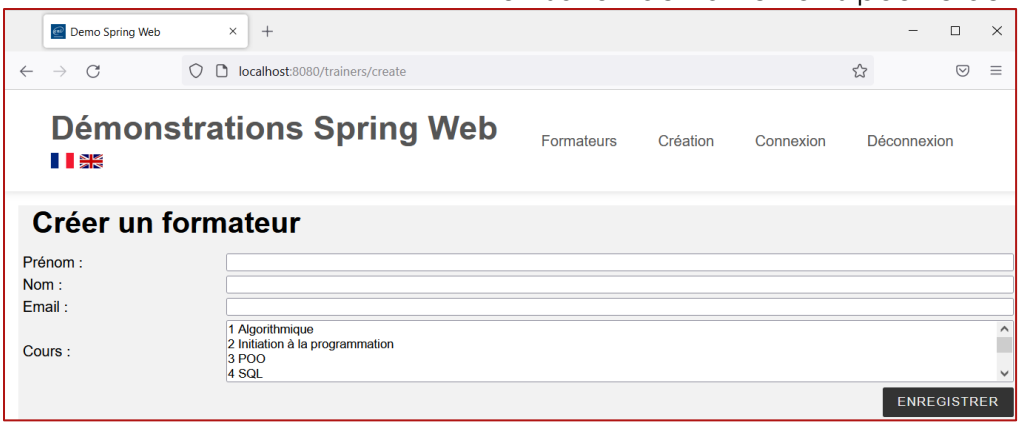


- Si nous nous connectons avec l'utilisateur : abaille@campus-eni.fr
  - Il a les rôles : TRAINER et ADMIN ; cela fonctionne



- Nous avons restreint l'affichage du détail d'un formateur et la création d'un formateur au rôle : ADMIN
  - Si nous utilisons l'utilisateur courant pour accéder à ces pages pas de soucis.





- Si vous tester avec les 3 autres, vous serez dans le cas où ils n'ont pas le droit.
    - Même en tapant à la main dans la barre de navigation l'URL.

Profil utilisateur

- Il est possible de se faire afficher les profils des utilisateurs
- Ajouter un contrôleur pour récupérer les profils connectés :

```
package fr.eni.demo.mmi.configuration;

import java.security.Principal;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/principal")
public class UserController {

    @GetMapping
    public Principal retrievePrincipal(Principal principal) {
        return principal;
    }
}
```

- L'exécution avec l'utilisateur abaille@campus-eni.fr donne :