eBook Gratuit

APPRENEZ spring-security

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#spring-security

Table des matières

À propos 1

Chapitre 1: Démarrer avec la sécurité des ressorts 2

Remarques 2

Versions 2

Examples 2

Installation ou configuration 2

Spring Securitiy pour protéger les points de terminaison de l'API REST 2

Spring-Security utilisant Spring-Boot et JDBC Authentication 4

Bonjour la sécurité du printemps 7

Application de sécurisation 7

Exécution d'une application Web sécurisée 9

Affichage du nom d'utilisateur 9

Déconnecter 10

Chapitre 2: Configuration de la sécurité du printemps 11

Examples 11

Configuration 11

Chapitre 3: Configuration de la sécurité Spring avec Java (non XML) 13

Introduction 13

Syntaxe 13

Examples 13

Sécurité de base avec annotation, source de données SQL 13

Crédits 14

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: spring-security

It is an unofficial and free spring-security ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official spring-security.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec la sécurité des ressorts

Remarques

Cette section fournit une vue d'ensemble de la sécurité des ressorts et des raisons pour lesquelles un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets dans la sécurité des ressorts, et établir un lien avec les sujets connexes. La documentation de Spring-Security étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
4.2.2	2017-03-02
3.2.10	2016-12-22
4.2.1	2016-12-21
4.1.4	2016-12-21
4.2.0	2016-11-10

Examples

Installation ou configuration

Instructions détaillées sur la configuration ou l'installation de la sécurité Spring.

Spring Securitiy pour protéger les points de terminaison de l'API REST

Ajoutez les entrées ci-dessous dans `pom.xml` .

```
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.1.0.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.1.0.RELEASE</version>
```

```
</dependency>
```

Important pour la version Spring supérieure à 3.1:

Erreur de création Bean pour `org.springframework.security.filterChains` vient quand vous utilisez la version Spring supérieure à 3,1 et n'ont pas ajouté les dépendances manuellement pour le `spring-aop` , `spring-jdbc` , `spring-tx` et `spring-expressions` dans votre `pom.xml` .

Ajouter les entrées ci-dessous dans le contexte Spring. Nous voulons protéger deux points de terminaison REST (helloworld & goodbye). Ajustez la version XSD selon la version Spring.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:security="http://www.springframework.org/schema/security"
   xmlns:context="http://www.springframework.org/schema/context"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
                       http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.1.xsd
                       http://www.springframework.org/schema/security
                       http://www.springframework.org/schema/security/spring-security-
3.1.xsd">

    <security:http auto-config='true' create-session="never">
        <security:intercept-url pattern="/helloworld/**" access="ROLE_USER" />
        <security:intercept-url pattern="/goodbye/**" access="ROLE_ADMIN" />
        <security:intercept-url pattern="/**" access="IS_AUTHENTICATED_ANONYMOUSLY" />
        <security:http-basic />
    </security:http>

    <security:authentication-manager>
        <security:authentication-provider>
            <security:user-service>
                <security:user name="username1" password="password1"
                    authorities="ROLE_USER" />
                <security:user name="username2" password="password2"
                    authorities="ROLE_ADMIN" />
            </security:user-service>
        </security:authentication-provider>
    </security:authentication-manager>
</beans>
```

Ajoutez les entrées ci-dessous dans `web.xml` .

```
<!-- Spring security-->
<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>

<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
```

```
</listener>

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:security-context.xml</param-value>
</context-param>
```

Spring-Security utilisant Spring-Boot et JDBC Authentication

Supposons que vous vouliez empêcher les utilisateurs non autorisés d'accéder à la page, vous devez alors leur interdire l'accès. Nous pouvons le faire en utilisant une sécurité de printemps qui fournit une authentification de base en sécurisant tous les points de terminaison HTTP. Pour cela, vous devez ajouter une dépendance à la sécurité des ressorts à votre projet. Dans maven, nous pouvons ajouter la dépendance en tant que:

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

Voici une configuration de sécurité qui garantit que seuls les utilisateurs authentifiés peuvent accéder.

```
@Configuration
@Order(SecurityProperties.ACCESS_OVERRIDE_ORDER)
public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired
DataSource datasource;

@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .authorizeRequests()
            .anyRequest()
            .fullyAuthenticated()
            .and()
        .formLogin()
            .loginPage("/login")
            .failureUrl("/login?error")
            .permitAll()
            .and()
        .logout()
            .logoutUrl("/logout")
            .logoutSuccessUrl("/login?logout")
            .permitAll()
            .and()
        .csrf();
}

@Override
protected void configure(AuthenticationManagerBuilder auth) throws Exception {
    auth.jdbcAuthentication().dataSource(datasource).passwordEncoder(passwordEncoder());
}

@Bean
```

```
public PasswordEncoder passwordEncoder() {
    PasswordEncoder encoder = new BCryptPasswordEncoder();
    return encoder;
}
}
```

Configuration	La description
`@Configuration`	Indique que la classe peut être utilisée par le conteneur Spring IoC comme source de définitions de bean.
`@Order (SecurityProperties.ACCESS_OVERRIDE_ORDER)`	Remplacez les règles d'accès sans modifier les autres fonctionnalités configurées automatiquement. Les valeurs inférieures ont une priorité plus élevée.
`WebSecurityConfigurerAdapter`	La classe `SecurityConfig` étend et remplace certaines de ses méthodes pour définir certaines caractéristiques de la configuration de sécurité.
`@Autowired` de `DataSource`	Fournit une fabrique pour les connexions à la source de données physique.
`configure(HttpSecurity)`	La méthode remplacée définit quels chemins d'URL doivent être sécurisés et lesquels ne doivent pas l'être.
`.authorizeRequests().anyRequest() .fullyAuthenticated()`	Indique que toute demande à notre application nécessite d'être authentifiée.
`.formLogin()`	Configure une connexion basée sur un formulaire
`.loginPage("/login").failureUrl("/login?error").permitAll()`	Spécifie l'emplacement de la page de connexion et tous les utilisateurs doivent être autorisés à accéder à la page.
`.logout().logoutUrl("/logout")` `.logoutSuccessUrl("/login?logout").permitAll()`	L'URL à rediriger après la

Configuration	La description
	déconnexion s'est produite. La valeur par défaut est / login? Logout.
`.csrf()`	Utilisé pour empêcher la falsification de requêtes inter-sites, la protection CSRF est activée (par défaut).
`configure(AuthenticationManagerBuilder){}`	Méthode remplacée pour définir la manière dont les utilisateurs sont authentifiés.
`.jdbcAuthentication().dataSource(datasource)`	Indique que nous utilisons l'authentification JDBC
`.passwordEncoder(passwordEncoder())`	Indique que nous utilisons un encodeur de mot de passe pour encoder nos mots de passe. (Un bean est créé pour renvoyer le choix du mot de passe Encoder, nous utilisons BCrypt dans ce cas)

Notez que nous n'avons configuré aucun nom de table à utiliser ou aucune requête, car la sécurité par défaut recherche par défaut les tables ci-dessous:

```
create table users (
  username varchar(50) not null primary key,
  password varchar(255) not null,
  enabled boolean not null) ;

create table authorities (
  username varchar(50) not null,
  authority varchar(50) not null,
  foreign key (username) references users (username),
  unique index authorities_idx_1 (username, authority));
```

Insérez les lignes suivantes dans les tableaux ci-dessus:

```
INSERT INTO authorities(username,authority)
VALUES ('user', 'ROLE_ADMIN');

INSERT INTO users(username,password,enabled)
VALUES('user', '$2a$10$JvqOtJaDys0yoXPX9w47YOqu9wZr/PkN1dJqjG9HHAzMyu9EV1R4m', '1');
```

Le **nom d'utilisateur** dans notre cas est `user` et le **mot de passe** est également crypté par l' `user` avec l'algorithme BCrypt

Enfin, configurez une source de données dans l'application.properties pour que le boot de printemps utilise:

```
spring.datasource.url = jdbc:mysql://localhost:3306/spring
spring.datasource.username = root
spring.datasource.password = Welcome123
```

Remarque: créez et configurez un contrôleur de connexion et associez-le au chemin d'accès `/login` et indiquez votre page de connexion à ce contrôleur.

Bonjour la sécurité du printemps

> **Remarque 1:** Vous devez avoir des connaissances préalables sur la ***page de servlet java (JSP)*** et ***Apache Maven*** avant de commencer cet exemple.

Démarrez le serveur Web (comme ***Apache tomcat***) avec un projet Web existant ou créez-en un.

Visitez le ***index.jsp*** .

N'importe qui peut accéder à cette page, c'est dangereux!

Application de sécurisation

1. Mettre à jour les dépendances Maven

Ajout de dépendances à votre fichier pom.xml

pom.xml

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-web</artifactId>
  <version>4.0.1.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>4.0.1.RELEASE</version>
</dependency>
```

> **Note 1:** Si vous n'utilisez pas "Spring" dans votre projet auparavant, il n'y a pas de dépendance à propos de "Spring-context". Cet exemple utilisera la configuration xml avec "spring-context". Alors ajoutez cette dépendance aussi.

```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>4.2.2.RELEASE</version>
</dependency>
```

Note 2: Si vous n'utilisez pas JSTL dans votre projet auparavant, cela ne dépend pas de cela. Cet exemple utilisera JSTL dans la page jsp. Alors ajoutez cette dépendance aussi.

```
<dependency>
  <groupId>org.glassfish.web</groupId>
  <artifactId>javax.servlet.jsp.jstl</artifactId>
  <version>1.2.1</version>
</dependency>
```

---

2. Créer un fichier de configuration de sécurité Spring

Rendez le dossier "spring" dans le dossier "WEB-INF" et créez le fichier security.xml. Copiez et collez à partir des codes suivants.

WEB-INF / spring / security.xml

```
<b:beans xmlns="http://www.springframework.org/schema/security"
         xmlns:b="http://www.springframework.org/schema/beans"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
                             http://www.springframework.org/schema/security
http://www.springframework.org/schema/security/spring-security.xsd">

  <http />

  <user-service>
    <user name="stackoverflow" password="pwd" authorities="ROLE_USER" />
  </user-service>

</b:beans>
```

---

3. Mettre à jour web.xml

Mettez à jour votre web.xml dans le dossier "WEB-INF"

WEB-INF / 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>
```

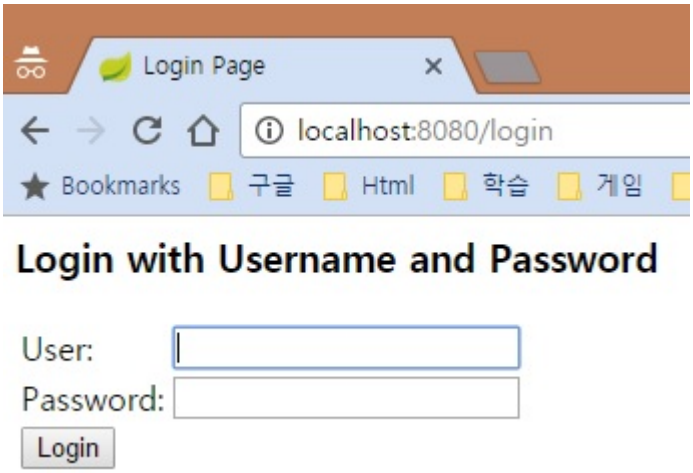
Remarque: Si vous n'utilisez pas "Spring" dans votre projet auparavant, il n'y a pas de configuration concernant le chargement des contextes Spring. Ajoutez donc ce paramètre et votre écouteur également.

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
    /WEB-INF/spring/*.xml
  </param-value>
</context-param>

<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
</listener>
```

Exécution d'une application Web sécurisée

Après avoir exécuté votre serveur Web et ***consulté index.jsp,*** vous verrez la page de connexion par défaut générée par la sécurité Spring. Parce que vous n'êtes pas authentifié.



Vous pouvez vous connecter

```
username : stackoverflow
password : pwd
```

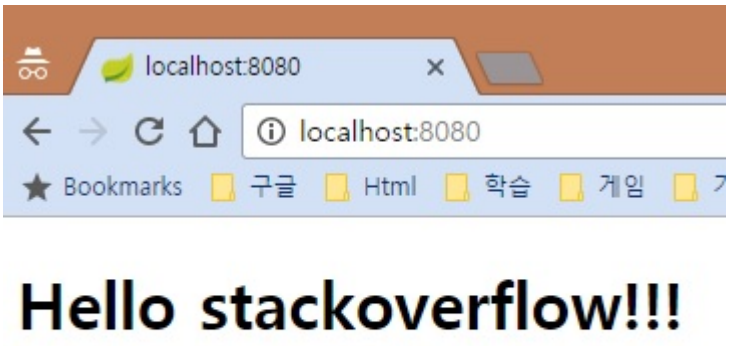
> **Remarque:** paramètre nom d'utilisateur et mot de passe sur ***WEB-INF / spring / security.xml***

Affichage du nom d'utilisateur

Ajouter le tag jstl après le "Hello", qui imprime le nom d'utilisateur

index.jsp

```
<h1>Hello <c:out value="${pageContext.request.remoteUser}" />!!</h1>
```



Déconnecter

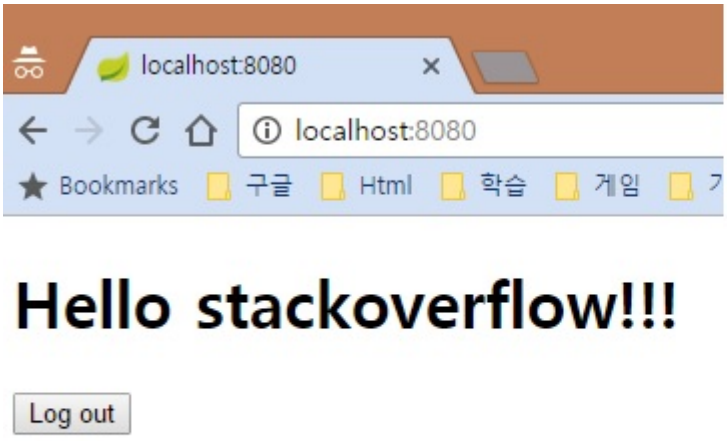
index.jsp

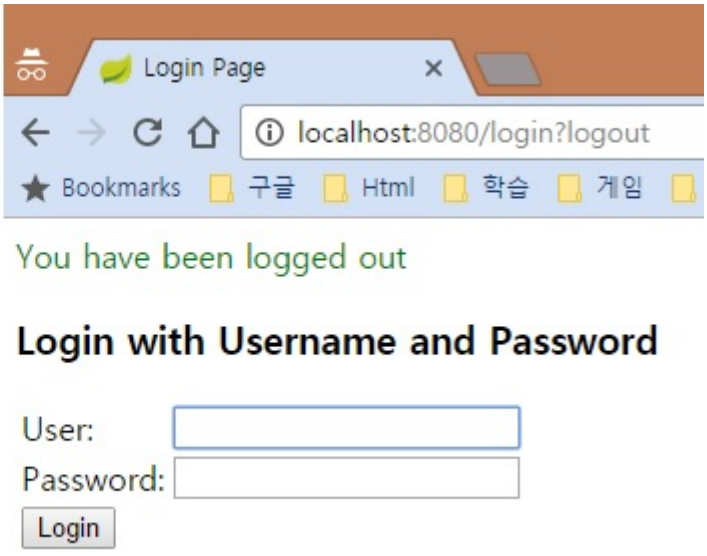
Ajout de la forme, des balises d'entrée après «Bonjour nom d'utilisateur», cette soumission a généré une ***déconnexion*** de l'URL ***/ déconnexion*** de la sécurité du printemps.

```
<h1>Hello <c:out value="${pageContext.request.remoteUser}" />!!</h1>

<form action="/logout" method="post">
  <input type="submit" value="Log out" />
  <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />
</form>
```

Lorsque vous vous déconnectez avec succès, vous voyez à nouveau la page de connexion générée automatiquement. À cause de vous ne sont pas authentifiés maintenant.





Lire Démarrer avec la sécurité des ressorts en ligne: https://riptutorial.com/fr/spring-security/topic/1434/demarrer-avec-la-securite-des-ressorts

Chapitre 2: Configuration de la sécurité du printemps

Examples

Configuration

Voici la configuration Java correspondante:

Ajoutez cette annotation à une classe `@Configuration` pour que la configuration de la sécurité Spring soit définie dans tout `WebSecurityConfigurer` ou plus probablement en étendant la classe de base `WebSecurityConfigurerAdapter` et en `WebSecurityConfigurerAdapter` les méthodes individuelles:

```
@Configuration
@EnableWebSecurity
@Profile("container")
public class XSecurityConfig extends WebSecurityConfigurerAdapter {
```

inMemoryAuthentication
Il définit un schéma d'authentification en mémoire avec un utilisateur ayant le nom d'utilisateur "utilisateur", le mot de passe "mot de passe" et le rôle "ROLE_USER".

```
  @Override
  protected void configure(AuthenticationManagerBuilder auth) throws Exception {
     auth
         .inMemoryAuthentication()
             .withUser("user")
             .password("password")
             .roles("ROLE_USER");
  }

  @Override
  public void configure(WebSecurity web) throws Exception {
     web
         .ignoring()
             .antMatchers("/scripts/**","/styles/**","/images/**","/error/**");
  }
```

HttpSecurity
Il permet de configurer la sécurité Web pour des requêtes HTTP spécifiques. Par défaut, il sera appliqué à toutes les requêtes, mais peut être restreint à l'aide de `requestMatcher(RequestMatcher)` ou d'autres méthodes similaires.

```
  @Override
  public void configure(HttpSecurity http) throws Exception {
     http
         .authorizeRequests()
             .antMatchers("/rest/**").authenticated()
             .antMatchers("/**").permitAll()
```

```
            .anyRequest().authenticated()
            .and()
        .formLogin()
            .successHandler(new AuthenticationSuccessHandler() {
                @Override
                public void onAuthenticationSuccess(
                    HttpServletRequest request,
                    HttpServletResponse response,
                    Authentication a) throws IOException, ServletException {
                        // To change body of generated methods,
                        response.setStatus(HttpServletResponse.SC_OK);
                }
            })
            .failureHandler(new AuthenticationFailureHandler() {
                @Override
                public void onAuthenticationFailure(
                    HttpServletRequest request,
                    HttpServletResponse response,
                    AuthenticationException ae) throws IOException, ServletException {
                        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
                }
            })
            .loginProcessingUrl("/access/login")
            .and()
        .logout()
            .logoutUrl("/access/logout")
            .logoutSuccessHandler(new LogoutSuccessHandler() {
                @Override
                public void onLogoutSuccess(
                    HttpServletRequest request,
                    HttpServletResponse response,
                    Authentication a) throws IOException, ServletException {
                        response.setStatus(HttpServletResponse.SC_NO_CONTENT);
                }
            })
            .invalidateHttpSession(true)
            .and()
        .exceptionHandling()
        .authenticationEntryPoint(new Http403ForbiddenEntryPoint())
            .and()
        .csrf() //Disabled CSRF protection
            .disable();
  }
}
```

Lire Configuration de la sécurité du printemps en ligne: https://riptutorial.com/fr/spring-security/topic/6600/configuration-de-la-securite-du-printemps

Chapitre 3: Configuration de la sécurité Spring avec Java (non XML)

Introduction

Configuration de base du ressort de base d'annotation, base de données d'annotation.

Syntaxe

1. configureGlobal () configure l'objet auth.
2. Les deux derniers SQL peuvent être facultatifs.
3. La méthode configure () indique à Spring mvc comment authentifier la requête
4. une URL que nous n'avons pas besoin d'authentifier
5. les autres redirigeront vers / login s'ils ne sont pas encore authentifiés.

Examples

Sécurité de base avec annotation, source de données SQL

```
@Configuration
public class AppSecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired
DataSource dataSource;

@Autowired
public void configureGlobal(AuthenticationManagerBuilder auth)
        throws Exception {
     auth.jdbcAuthentication().dataSource(dataSource)
    .passwordEncoder(new BCryptPasswordEncoder())
    .usersByUsernameQuery("select username,password, enabled from users where username=?")
    .authoritiesByUsernameQuery("select username, role from user_roles where username=?");

}
@Override
protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable();
    http.authorizeRequests().antMatchers(".resources/**", "/public/**")
            .permitAll().anyRequest().authenticated().and().formLogin()
            .loginPage("/login").permitAll().and().logout().permitAll();

    }

}
```

Lire Configuration de la sécurité Spring avec Java (non XML) en ligne: https://riptutorial.com/fr/spring-security/topic/8700/configuration-de-la-securite-spring-avec-java--non-xml-

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec la sécurité des ressorts	Alex78191, AMAN KUMAR, Community, dur, Gnanam, kartik, Panther, sayingu, Xtreme Biker
2	Configuration de la sécurité du printemps	dur, ojus kulkarni
3	Configuration de la sécurité Spring avec Java (non XML)	Maxi Wu