



EBook Gratuito

APPENDIMENTO spring-security

Free unaffiliated eBook created from
Stack Overflow contributors.

#spring-
security

Sommario

Di.....	1
Capitolo 1: Iniziare con la sicurezza di primavera.....	2
Osservazioni.....	2
Versioni.....	2
Examples.....	2
Installazione o configurazione.....	2
Spring Security per proteggere gli endpoint dell'API REST.....	2
Spring-Security utilizzando l'avvio a molla e l'autenticazione JDBC.....	4
Ciao Spring Security.....	7
Applicazione di protezione.....	7
Esecuzione di un'applicazione Web protetta.....	9
Visualizzazione del nome utente.....	9
Disconnettersi.....	10
Capitolo 2: Spring Security config con java (non XML).....	12
introduzione.....	12
Sintassi.....	12
Examples.....	12
Sicurezza primaverile di base con annotazione, origine dati SQL.....	12
Capitolo 3: Spring Security Configuration.....	13
Examples.....	13
Configurazione.....	13
Titoli di coda.....	15

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capitolo 1: Iniziare con la sicurezza di primavera

Osservazioni

Questa sezione fornisce una panoramica di ciò che è la sicurezza di primavera, e perché uno sviluppatore potrebbe voler usarlo.

Dovrebbe anche menzionare tutti i soggetti di grandi dimensioni entro la sicurezza di primavera e collegarsi agli argomenti correlati. Poiché la Documentazione per la sicurezza di primavera è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data di rilascio
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

Installazione o configurazione

Istruzioni dettagliate su come installare o installare la sicurezza della molla.

Spring Security per proteggere gli endpoint dell'API REST

Aggiungi sotto voci in `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>
```

Importante per la versione Spring maggiore di 3.1:

L'errore di creazione del bean per `org.springframework.security.filterChains` arriva quando si utilizza la versione Spring superiore a 3.1 e non si aggiungono manualmente le dipendenze per `spring-aop`, `spring-jdbc`, `spring-tx` e `spring-expressions` pom.xml nel proprio pom.xml .

Aggiungi sotto voci nel contesto di primavera. Vogliamo proteggere due endpoint REST (helloworld e addio). Regola la versione XSD in base alla versione 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>
```

Aggiungi sotto voci in `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 utilizzando l'avvio a molla e l'autenticazione JDBC

Supponiamo che tu voglia impedire agli utenti non autorizzati di accedere alla pagina, quindi devi metterli a barriera autorizzando l'accesso. Possiamo farlo utilizzando la sicurezza di primavera che fornisce l'autenticazione di base proteggendo tutti gli endpoint HTTP. Per questo è necessario aggiungere una dipendenza di sicurezza primaverile al progetto, in Maven possiamo aggiungere la dipendenza come:

```

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

```

Ecco una configurazione di sicurezza che garantisce l'accesso solo agli utenti autenticati.

```

@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;
}
}

```

Configurazione	Descrizione
@Configuration	Indica che la classe può essere utilizzata dal contenitore IoC di primavera come origine delle definizioni di bean.
@Order (SecurityProperties.ACCESS_OVERRIDE_ORDER)	Sovrascrivi le regole di accesso senza modificare altre funzionalità configurate automaticamente. I valori più bassi hanno una priorità più alta.
WebSecurityConfigurerAdapter	La classe <code>SecurityConfig</code> estende e sovrascrive un paio dei suoi metodi per impostare alcune specifiche della configurazione di sicurezza.
@Autowired di DataSource	Fornire factory per le connessioni all'origine dati fisica.
configure (HttpSecurity)	Il metodo sovrascritto definisce quali percorsi URL devono essere protetti e quali no.
.authorizeRequests () .anyRequest () .fullyAuthenticated ()	Indica a primavera che tutte le richieste per la nostra applicazione richiedono di essere autenticate.
.formLogin ()	Configura un accesso basato su modulo
.loginPage ("/login") .failureUrl ("/login?error") .permitAll ()	Specifica la posizione della pagina di accesso e a tutti gli utenti dovrebbe essere consentito di accedere alla pagina.

Configurazione	Descrizione
<pre>.logout().logoutUrl("/logout") .logoutSuccessUrl("/login?logout").permitAll()</pre>	Si è verificato l'URL da reindirizzare dopo il logout. L'impostazione predefinita è /login? Logout.
<pre>.csrf()</pre>	Utilizzato per prevenire la falsificazione di richieste intersito, la protezione CSRF è abilitata (impostazione predefinita).
<pre>configure(AuthenticationManagerBuilder){}</pre>	Metodo sottoposto a override per definire il modo in cui gli utenti sono autenticati.
<pre>.jdbcAuthentication().dataSource(datasource)</pre>	Indica che stiamo utilizzando l'autenticazione JDBC
<pre>.passwordEncoder(passwordEncoder())</pre>	Indica a primavera che stiamo usando un codificatore di password per codificare le nostre password. (Viene creato un bean per restituire la scelta della password Encoder, stiamo usando BCrypt in questo caso)

Si noti che non è stato configurato alcun nome di tabella da utilizzare o alcuna query, poiché la sicurezza di primavera per impostazione predefinita riguarda le seguenti tabelle:

```
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));
```

Inserisci le seguenti righe nelle tabelle precedenti:

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

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


Il **nome utente** nel nostro caso è `user` e la **password** è anche crittografata `user` con algoritmo BCrypt

Infine, configura un'origine dati in `application.properties` per l'avvio a molla da utilizzare:

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

Nota: creare e configurare un controller di accesso e associarlo al percorso `/login` e indirizzare la pagina di accesso a questo controller

Ciao Spring Security

Nota 1: prima di iniziare gli esempi, è necessario conoscere in anticipo la [pagina JSP servlet \(JSP\)](#) e [Apache Maven](#).

Avvia il web server (come [Apache tomcat](#)) con il progetto web esistente o creane uno.

Visita l' [indice.jsp](#).

Chiunque può accedere a quella pagina, è insicuro!

Applicazione di protezione

1. Aggiorna dipendenze Maven

Aggiunta di dipendenze al tuo file `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>
```

Nota 1: se non stai usando "Spring" nel tuo progetto in precedenza, non c'è alcuna dipendenza da "spring-context". Questo esempio userà la configurazione xml con "spring-context". Quindi aggiungi anche questa dipendenza.

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

```
<version>4.2.2.RELEASE</version>
</dependency>
```

Nota 2: se non si utilizza JSTL nel progetto in precedenza, non c'è alcuna dipendenza in merito. Questo esempio utilizzerà JSTL nella pagina jsp. Quindi aggiungi anche questa dipendenza.

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

2. Crea il file di configurazione di Spring Security

Crea il nome della cartella "molla" nella cartella "WEB-INF" e crea il file security.xml. Copia e incolla dai codici successivi.

WEB-INF / molla / 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. Aggiorna web.xml

Aggiorna il tuo web.xml all'interno della cartella "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>
```

Nota: se non si utilizza "Spring" nel progetto in precedenza, non è possibile caricare

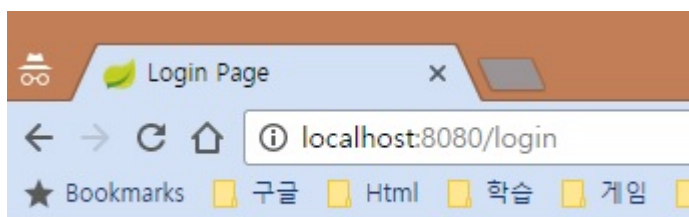
configurazioni relative ai contesti Spring. Quindi aggiungi anche questo parametro e ascoltatore.

```
<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>
```

Esecuzione di un'applicazione Web protetta

Dopo aver eseguito il server Web e visitato *index.jsp*, verrà visualizzata la pagina di accesso predefinita generata da Spring Security. Perché non sei autenticato.



Login with Username and Password

User:

Password:

Puoi accedere

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

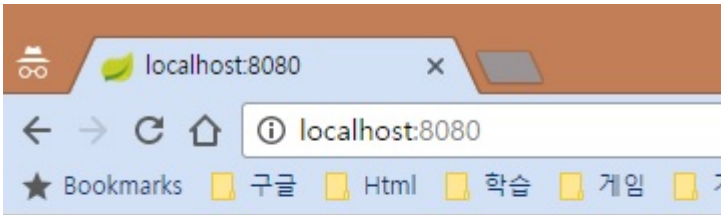
Nota: impostazione di nome utente e password su *WEB-INF / spring / security.xml*

Visualizzazione del nome utente

Aggiungendo il tag `jstl` dopo "Hello", che stampa il nome utente

index.jsp

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



Hello stackoverflow!!!

Disconnettersi

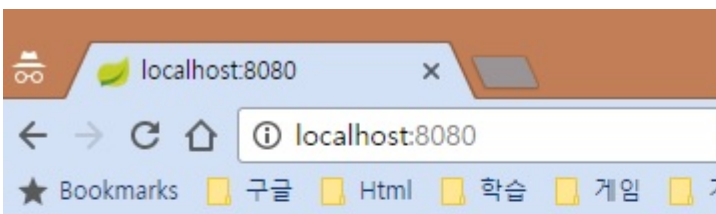
index.jsp

Aggiunta di un modulo, inserimento di tag dopo "Hello user name", che invia generato url / **logout** di **logout** dalla spring security.

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

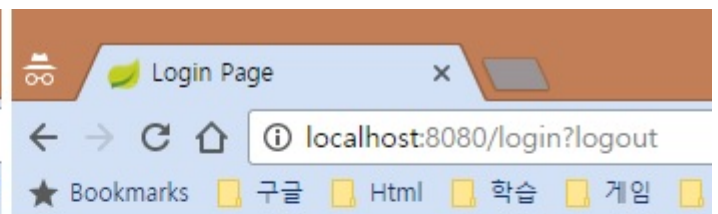
<form action="/logout" method="post">
  <input type="submit" value="Log out" />
  <input type="hidden" name="\${_csrf.parameterName}" value="\${_csrf.token}" />
</form>
```

Quando si esegue il logout, viene visualizzata nuovamente la pagina di accesso generata automaticamente. Perché non sei autenticato ora.



Hello stackoverflow!!!

Log out



You have been logged out

Login with Username and Password

User:

Password:

Login

Leggi Iniziare con la sicurezza di primavera online: <https://riptutorial.com/it/spring-security/topic/1434/iniziare-con-la-sicurezza-di-primavera>

Capitolo 2: Spring Security config con java (non XML)

introduzione

Database tipico supportato, impostazione della sicurezza della base di annotation base.

Sintassi

1. configureGlobal () configura l'oggetto auth.
2. Le due SQL successive potrebbero essere facoltative.
3. configure () metodo dice a molla mvc come autenticare la richiesta
4. alcuni URL non abbiamo bisogno di autenticarsi
5. altri reindirizzeranno a / login se non ancora autenticati.

Examples

Sicurezza primavera di base con annotazione, origine dati 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();
    }
}
```

Leggi Spring Security config con java (non XML) online: <https://riptutorial.com/it/spring-security/topic/8700/spring-security-config-con-java--non-xml->

Capitolo 3: Spring Security Configuration

Examples

Configurazione

Ecco la corrispondente configurazione Java:

Aggiungi questa annotazione a una classe `@Configuration` per fare in modo che la configurazione di Spring Security sia definita in qualsiasi `WebSecurityConfigurer` o più probabilmente estendendo la classe base `WebSecurityConfigurerAdapter` e sovrascrivendo i singoli metodi:

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

inMemoryAuthentication

Definisce uno schema di autenticazione in memoria con un utente che ha il nome utente "utente", la password "password" e il ruolo "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

Consente la configurazione della sicurezza basata sul Web per richieste HTTP specifiche. Per impostazione predefinita verrà applicato a tutte le richieste, ma può essere limitato utilizzando `requestMatcher(RequestMatcher)` o altri metodi simili.

```
@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();
    }
}

```

Leggi Spring Security Configuration online: <https://riptutorial.com/it/spring-security/topic/6600/spring-security-configuration>

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con la sicurezza di primavera	Alex78191 , AMAN KUMAR , Community , dur , Gnanam , kartik , Panther , sayingu , Xtreme Biker
2	Spring Security config con java (non XML)	Maxi Wu
3	Spring Security Configuration	dur , ojus kulkarni