Spring Security Example

Posted on August 16, 2011 , Last modified : April 2, 2014
By mkyong

In this tutorial, we will show you how to integrate Spring Security with a Spring MVC web application to secure a URL access. After implementing Spring Security, to access the content of an “admin” page, users need to key in the correct “username” and “password”.

Technologies used :

1. Spring 3.2.8.RELEASE
2. Spring Security 3.2.3.RELEASE
3. Eclipse 4.2
4. JDK 1.6
5. Maven 3

Note
Spring Security 3.0 requires Java 5.0 Runtime Environment or higher

1. Project Demo

2. Directory Structure

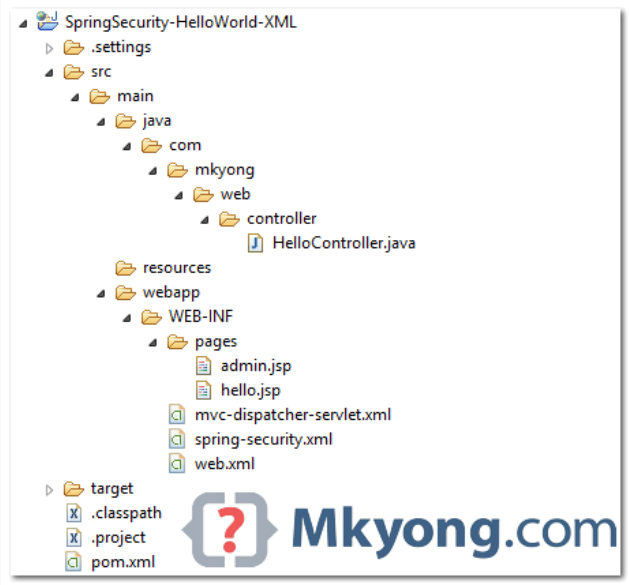
Review the final directory structure of this tutorial.

Recent Posts

Ant - How To Create A Jar File with external libraries
Ant - How to create a Java Project
Create a fat Jar file - Maven Assembly Plugin
Maven - Exclude log4j.properties in Jar file
Maven – Create a fat Jar file - One-JAR example

Popular Tutorials

Android Tutorial
JSF 2.0 Tutorial



3. Spring Security Dependencies

To use Spring security, you need `spring-security-web` and `spring-security-config`.

pom.xml

```
<properties>
 <jdk.version>1.6</jdk.version>
 <spring.version>3.2.8.RELEASE</spring.version>
 <spring.security.version>3.2.3.RELEASE</spring.security.version>
 <jstl.version>1.2</jstl.version>
</properties>

<dependencies>

 <!-- Spring dependencies -->
 <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>${spring.version}</version>
 </dependency>

 <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>${spring.version}</version>
 </dependency>

 <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>${spring.version}</version>
 </dependency>

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

 <dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-config</artifactId>
  <version>${spring.security.version}</version>
 </dependency>

 <!-- jstl for jsp page -->
 <dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>${jstl.version}</version>
 </dependency>

</dependencies>
```

4. Spring MVC Web Application

A simple controller :

1. If URL = `/welcome` or `/` , return hello page.
2. If URL = `/admin` , return admin page.

Later, we will show you how to use Spring Security to secure the “/admin” URL with a user login form.

HelloController.java

```
package com.mkyong.web.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class HelloController {

 @RequestMapping(value = { "/", "/welcome**" }, method = RequestMethod.GET)
 public ModelAndView welcomePage() {

  ModelAndView model = new ModelAndView();
  model.addObject("title", "Spring Security Hello World");
  model.addObject("message", "This is welcome page!");
  model.setViewName("hello");
  return model;

 }

 @RequestMapping(value = "/admin**", method = RequestMethod.GET)
 public ModelAndView adminPage() {

  ModelAndView model = new ModelAndView();
  model.addObject("title", "Spring Security Hello World");
  model.addObject("message", "This is protected page!");
  model.setViewName("admin");

  return model;

 }

}
```

Two JSP pages.

hello.jsp

```
<%@page session="false"%>
<html>
<body>
 <h1>Title : ${title}</h1>
 <h1>Message : ${message}</h1>
</body>
</html>
```

admin.jsp

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@page session="true"%>
<html>
<body>
 <h1>Title : ${title}</h1>
 <h1>Message : ${message}</h1>

 <c:if test="${pageContext.request.userPrincipal.name != null}">
    <h2>Welcome : ${pageContext.request.userPrincipal.name}
          | <a href="<c:url value="/j_spring_security_logout" />" > Logout</a></h2>
 </c:if>
</body>
</html>
```

mvc-dispatcher-servlet.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
 xmlns:context="http://www.springframework.org/schema/context"
 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-3.0.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.0.xsd">

 <context:component-scan base-package="com.mkyong.*" />

 <bean
   class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix">
  <value>/WEB-INF/pages/</value>
   </property>
   <property name="suffix">
  <value>.jsp</value>
   </property>
 </bean>

</beans>
```

5. Spring Security : User Authentication

Create a Spring Security XML file.

spring-security.xml

```
<beans:beans xmlns="http://www.springframework.org/schema/security"
 xmlns:beans="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-3.0.xsd
 http://www.springframework.org/schema/security
 http://www.springframework.org/schema/security/spring-security-3.2.xsd">

 <http auto-config="true">
  <intercept-url pattern="/admin**" access="ROLE_USER" />
 </http>

 <authentication-manager>
   <authentication-provider>
     <user-service>
  <user name="mkyong" password="123456" authorities="ROLE_USER" />
     </user-service>
   </authentication-provider>
 </authentication-manager>

</beans:beans>
```

It tells, only user “mkyong” is allowed to access the `/admin` URL.

6. Integrate Spring Security

To integrate Spring security with a Spring MVC web application, just declares `DelegatingFilterProxy` as a servlet filter to intercept any incoming request.

web.xml

```
<web-app id="WebApp_ID" version="2.4"
 xmlns="http://java.sun.com/xml/ns/j2ee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
 http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

 <display-name>Spring MVC Application</display-name>

 <!-- Spring MVC -->
 <servlet>
  <servlet-name>mvc-dispatcher</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
  <servlet-name>mvc-dispatcher</servlet-name>
  <url-pattern>/</url-pattern>
 </servlet-mapping>

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

        <!-- Loads Spring Security config file -->
 <context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
   /WEB-INF/spring-security.xml
  </param-value>
 </context-param>

 <!-- 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>

</web-app>
```

7. Demo

That's all, but wait… where's the login form? No worry, if you do not define any custom login form, Spring will create a simple login form automatically.

> **Custom Login Form**
> Read this "Spring Security form login example" to understand how to create a custom login form in Spring Security.

1. Welcome Page – `http://localhost:8080/spring-security-helloworld-xml/welcome`

2. Try to access `/admin` page, Spring Security will intercept the request and redirect to `/spring_security_login`, and a predefined login form is displayed.

3. If username and password is incorrect, error messages will be displayed, and Spring will redirect to this URL `/spring_security_login?login_error`.

4. If username and password are correct, Spring will redirect the request to the original requested URL and display the page.

Download Source Code

Download it – spring-security-helloworld-xml.zip (9 KB)

References

1. Spring Security Official Site
2. Spring 3 MVC hello world example
3. Spring Security form login example (authentication)

Tags : hello world | spring security

mkyong

Founder of Mkyong.com and HostingCompass.com, love Java and open source stuff. Follow him on Twitter, or befriend him on Facebook or Google Plus. If you like my tutorials, consider making a donation to this charity, thanks.

Related Posts

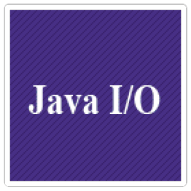
- Spring Security Hello World Annotation Example
- Log4j hello world example
- Spring Security + Hibernate Annotation Example
- Spring Security + Hibernate XML Example
- Spring Security : Encoded password does not look

Popular Posts

- Top 8 Java People You Should Know
- Top 20 Java Websites
- Top 5 Free Java eBooks
- Top 10 Java Regular Expression Examples
- Top 5 Open Source Q&A Systems

like BCrypt

You might also like following tutorials :

All Available Tutorials

Java Core Technologies :
Java I/O, Java RegEx, Java XML, Java JSON, JDBC, Java Misc
J2EE Frameworks :
Hibernate, JSF 2.0, Spring Core, Spring MVC, Spring Security, Spring MongoDB, Spring BatchApache Wicket, Struts 1.x, Struts 2.x
Web Service :
JAX-WS (SOAP), JAX-RS (REST)
Build Tools :
Maven, Archiva
Unit Test Frameworks :
jUnit, TestNG
Others :
Android, Google App Engine, jQuery, Java MongoDB, Quartz Scheduler, Log4j

Favorites Links

Android Getting Started
Google App Engine - Java
DZone - Fresh Links
Official Java EE 5 Tutorial
Official Java EE 6 Tutorial
Official Java EE 7 Tutorial
Spring 2.5.x documentation
Spring 3.2.x documentation
Spring Security 3.2.x documentation
Hibernate core 4.3 documentation
Java SE 6.0 API documentation
JSP home page
JSF home page
Eclipse IDE for Java developer
Struts 1.3.x documentation
Struts 2.3.x documentation
Maven home page
Maven central repository Search
Ant home page
JAX-WS Official Website
JAX-RS Official Website (Jersey)

Friends & Links

Java Code Geeks
PHP Tutorials
TenthOfMarch
Web Security Blog
Web Development
Cédric Beust (TestNG)

About Us

Mkyong.com is a weblog dedicated to Java/J2EE developers and Web Developers. We constantly publish useful tricks, tutorials on J2EE or web development.

All examples are simple, easy to read, and full source code available, and of course well tested in our development environment.

We're Social

1. Twitter - Follow Me
2. Facebook - Like Me
3. Google Plus - Add Me
4. RSS - Subscribe Me