[Spring Security](http://projects.spring.io/spring-security/" \t "_blank), is a flexible and powerful authentication and access control framework to secure Spring-based Java web application.

1. Spring Security Examples

* [Hello World XML Example](http://www.mkyong.com/spring-security/spring-security-hello-world-example/) Spring MVC + Spring Security XML-based project, using the default login form.
* [Hello World Annotation Example](http://www.mkyong.com/spring-security/spring-security-hello-world-annotation-example/)
* [Spring Security Custom Login Form XML Example](http://www.mkyong.com/spring-security/spring-security-form-login-example/)  
  Spring MVC + Spring Security XML-based project, custom login form, logout function, CSRF protection and in-memory authentication.
* [Spring Security Custom Login Form Annotation Example](http://www.mkyong.com/spring-security/spring-security-custom-login-form-annotation-example/)  
  Spring MVC + Spring Security annotations-based project, custom login form, logout function, CSRF protection and in-memory authentication.
* [Spring Security Form Login Using Database – XML and Annotation Example](http://www.mkyong.com/spring-security/spring-security-form-login-using-database/)  
  Database authentication, Spring Security, JSP taglibs, JDBC, customizes 403 access denied page and etc, both in XML and annotations.
* [Spring Security : Limit Login Attempts – XML and Annotation Example](http://www.mkyong.com/spring-security/spring-security-limit-login-attempts-example/)  
  Lock user accounts if a user tried 3 invalid login attempts.
* [Spring Security Remember Me Example](http://www.mkyong.com/spring-security/spring-security-remember-me-example/)  
  Remember me “Persistent Token Approach” example. Extra : user login from remember me cookie is unable to perform the update operation.
* [Spring Security password hashing example](http://www.mkyong.com/spring-security/spring-security-password-hashing-example/)  
  Password encoder with BCrypt algorithm.
* [Spring Security + Hibernate XML Example](http://www.mkyong.com/spring-security/spring-security-hibernate-xml-example/)  
  Using Hibernate to load users for database authentication.
* [Spring Security + Hibernate Annotation Example](http://www.mkyong.com/spring-security/spring-security-hibernate-annotation-example/)  
  Using Hibernate to load users for database authentication.

2. FAQs Some commonly asked questions in Spring Security.

* [Customize 403 access denied page](http://www.mkyong.com/spring-security/customize-http-403-access-denied-page-in-spring-security/)  
  Like topic, shows you how to customize a 403 access denied page, using the error-page attribute and custom handler.
* [Check if user is from remember me cookie](http://www.mkyong.com/spring-security/spring-security-check-if-user-is-from-remember-me-cookie/)  
  If authentication == RememberMeAuthenticationToken
* [Encoded password does not look like BCrypt](http://www.mkyong.com/spring-security/spring-security-encoded-password-does-not-look-like-bcrypt/)  
  Length of “password” is not enough to store the bcrypt hashing value.
* [How to get current login username in Spring Security](http://www.mkyong.com/spring-security/get-current-logged-in-username-in-spring-security/)  
  3 ways to get current login username in Spring Security.
* [ClassNotFoundException : org.springframework.security.web.savedrequest.DefaultSavedRequest](http://www.mkyong.com/spring-security/classnotfoundexception-defaultsavedrequest/)

Some outdated or obsoleted articles…may update in future.

* [Spring Security HTTP basic authentication example](http://www.mkyong.com/spring-security/spring-security-http-basic-authentication-example/)  
  The browser displays a login dialog for authentication.
* [Spring Security access control example](http://www.mkyong.com/spring-security/spring-security-access-control-example/)  
  Example to implement access control or authorization on web application.
* [Display custom error message in Spring Security](http://www.mkyong.com/spring-security/display-custom-error-message-in-spring-security/)  
  How to override default Spring Security error messages easily.
* [Spring Security logout example](http://www.mkyong.com/spring-security/spring-security-logout-example/)  
  Simple example to show you how to implement the logout function.

**hello world example**

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 :

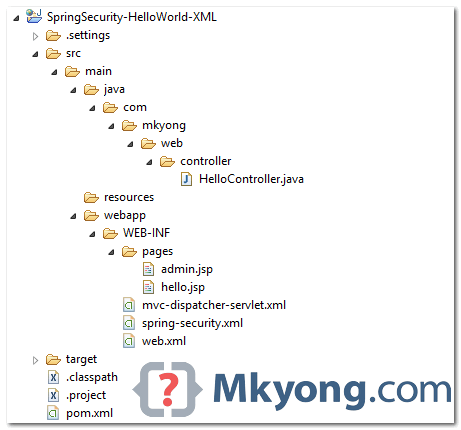
1. Spring 3.2.8.RELEASE ,Spring Security 3.2.3.RELEASE, Eclipse 4.2, JDK 1.6, Maven 3

**Note** Spring Security 3.0 requires Java 5.0 Runtime Environment or higher

Project Demo - https://youtu.be/hblHPyMuHJc

2. Directory Structure

Review the final directory structure of this 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 : If URL = /welcome or / , return hello page., 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>

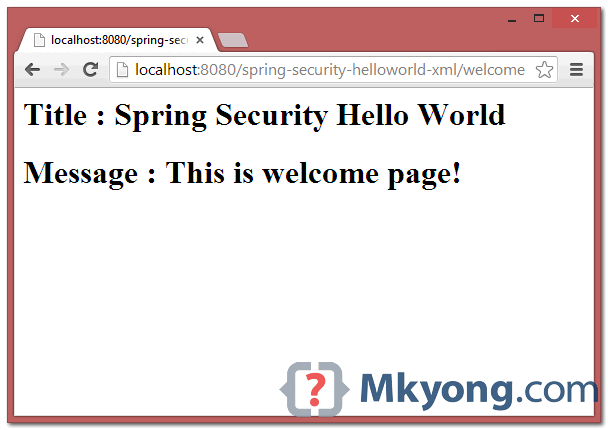
</web-app>

7. Demo

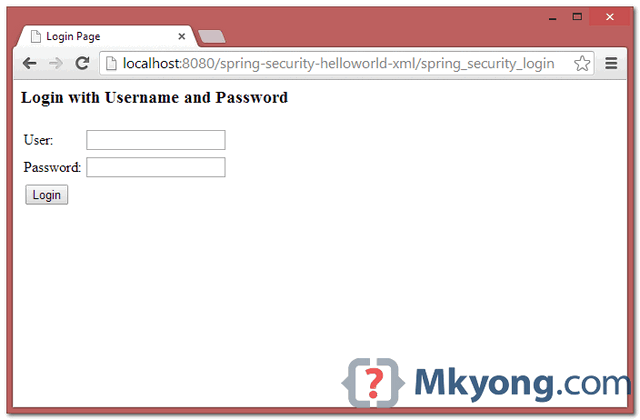
That’s all, but wait… where’s the login form? No worry, if you do not define any custom login form, Spring will create a simple login form automatically.

**Custom Login Form**  
Read this “[Spring Security form login example](http://www.mkyong.com/spring-security/spring-security-form-login-example/)” to understand how to create a custom login form in Spring Security.

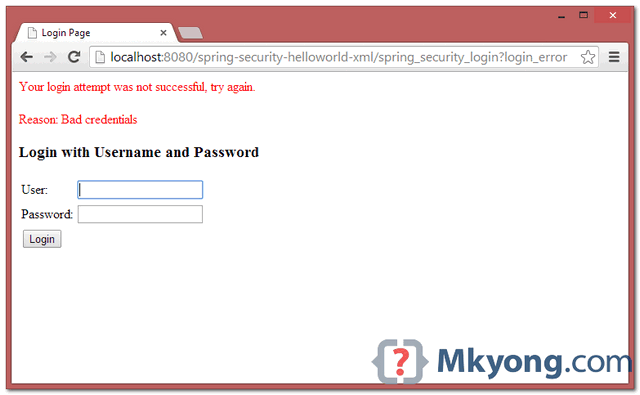
1. Welcome Page – http://localhost:8080/spring-security-helloworld-xml/welcome



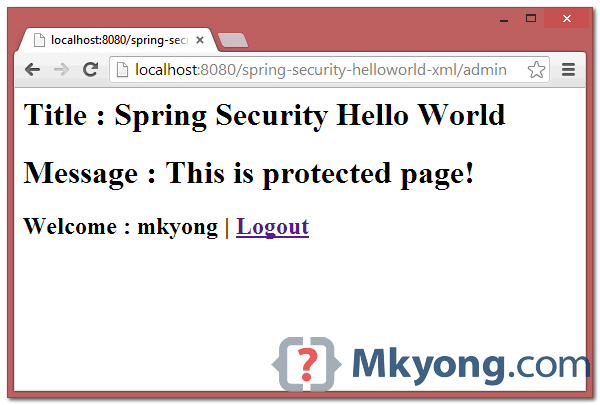
2. Try to access /admin page, Spring Security will intercept the request and redirect to /spring\_security\_login, and a predefined login form is displayed.



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](http://www.mkyong.com/wp-content/uploads/2011/08/spring-security-helloworld-xml.zip) (9 KB)

**Spring Security Hello World Annotation Example**

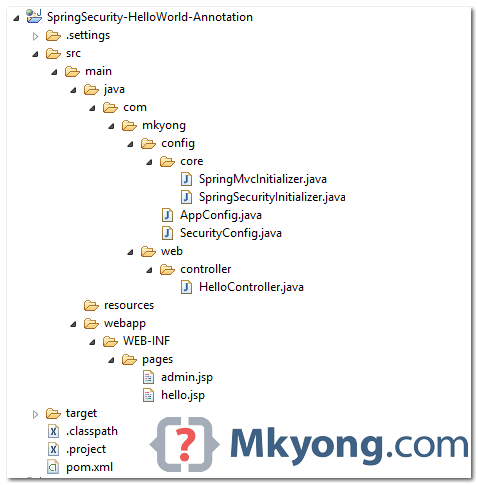
the previous XML-base Spring Security project into a pure Spring annotation project.

Technologies used : Spring 3.2.8.RELEASE, Spring Security 3.2.3.RELEASE,Maven 3,Tomcat 7 (Servlet 3.x)

1. This tutorial is using WebApplicationInitializer to load the Spring Context Loader automatically, which is supported in Servlet 3.x container only, for example, Tomcat 7 and Jetty 8.
2. Since we are using WebApplicationInitializer, the web.xml file is NOT required.
3. Spring Security annotations are supported in older Servlet 2.x container, for example, Tomcat 6. If you use the classic XML file to load the Spring context, this tutorial is still able to deploy on Servlet 2.x container, for example, Tomcat 6

1. Project Demo https://youtu.be/NKamWA6hDaU

2. Directory Structure Review the final directory structure of this tutorial.



3. 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 3 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. If URL = /admin , return admin page, If URL = /dba , return admin page. Later, we will secure the /admin and /dba URLs.

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 - Admin Page!");

model.setViewName("admin");

return model;

}

@RequestMapping(value = "/dba\*\*", method = RequestMethod.GET)

public ModelAndView dbaPage() {

ModelAndView model = new ModelAndView();

model.addObject("title", "Spring Security Hello World");

model.addObject("message", "This is protected page - Database Page!");

model.setViewName("admin");

return model;

}

}

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="/logout" />" > Logout</a></h2> </c:if>

</body>

</html>

5. Spring Security Configuration

*5.1* Create a Spring Security configuration file, and annotated with @EnableWebSecurity

SecurityConfig.java

package com.mkyong.config;

import org.springframework.beans.factory.annotation.Autowired;

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;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("mkyong").password("123456").roles("USER");

auth.inMemoryAuthentication().withUser("admin").password("123456").roles("ADMIN");

auth.inMemoryAuthentication().withUser("dba").password("123456").roles("DBA");

}

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests()

.antMatchers("/admin/\*\*").access("hasRole('ROLE\_ADMIN')")

.antMatchers("/dba/\*\*").access("hasRole('ROLE\_ADMIN') or hasRole('ROLE\_DBA')")

.and().formLogin();

}

}

The equivalent of the Spring Security xml file :

<http auto-config="true">

<intercept-url pattern="/admin\*\*" access="ROLE\_ADMIN" />

<intercept-url pattern="/dba\*\*" access="ROLE\_ADMIN,ROLE\_DBA" />

</http>

<authentication-manager>

<authentication-provider>

<user-service>

<user name="mkyong" password="123456" authorities="ROLE\_USER" />

<user name="admin" password="123456" authorities="ROLE\_ADMIN" />

<user name="dba" password="123456" authorities="ROLE\_DBA" />

</user-service>

</authentication-provider>

</authentication-manager>

*5.2* Create a class extends AbstractSecurityWebApplicationInitializer, it will load the  springSecurityFilterChainautomatically.

SpringSecurityInitializer.java

package com.mkyong.config.core;

import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;

public class SpringSecurityInitializer extends AbstractSecurityWebApplicationInitializer {

//do nothing

}

The equivalent of Spring Security in web.xml file :

<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>

6. Spring MVC Configuration

*6.1* A Config class, define the view’s technology and imports above SecurityConfig.java.

AppConfig.java

package com.mkyong.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.Import;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.view.InternalResourceViewResolver;

import org.springframework.web.servlet.view.JstlView;

@EnableWebMvc

@Configuration

@ComponentScan({ "com.mkyong.web.\*" })

@Import({ SecurityConfig.class })

public class AppConfig {

@Bean

public InternalResourceViewResolver viewResolver() {

InternalResourceViewResolver viewResolver

= new InternalResourceViewResolver();

viewResolver.setViewClass(JstlView.class);

viewResolver.setPrefix("/WEB-INF/pages/");

viewResolver.setSuffix(".jsp");

return viewResolver;

}

}

The equivalent of the Spring XML file :

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

<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>

*6.2* Create a Initializer class, to load everything. SpringMvcInitializer.java

package com.mkyong.config.core;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

import com.mkyong.config.AppConfig;

public class SpringMvcInitializer

extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override

protected Class<?>[] getRootConfigClasses() { return new Class[] { AppConfig.class }; }

@Override

protected Class<?>[] getServletConfigClasses() { return null; }

@Override

protected String[] getServletMappings() { return new String[] { "/" }; }

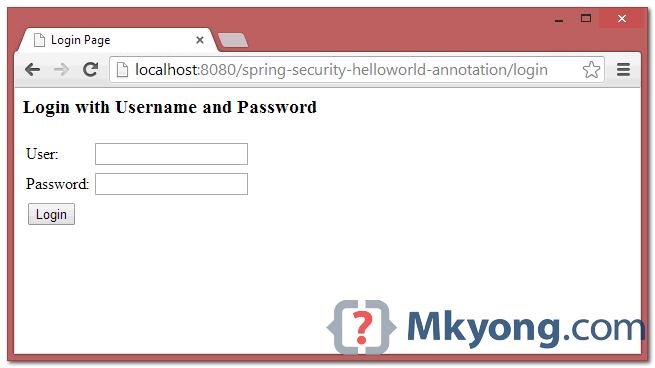
}

Done. **Note** In Servlet 3.x container environment + Spring container will detect and loads the Initializer classes automatically.

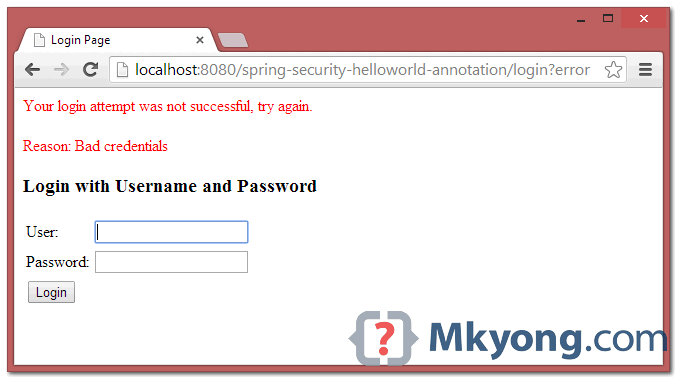
7. Demo 7.1. Welcome Page – http://localhost:8080/spring-security-helloworld-annotation/welcome



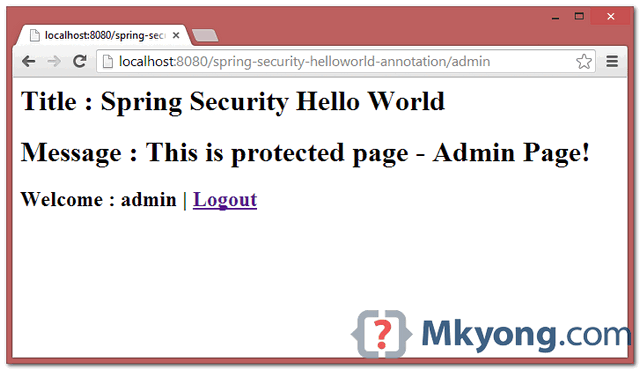
7.2 Try to access /admin page, Spring Security will intercept the request and redirect to /login, and a default login form is displayed.



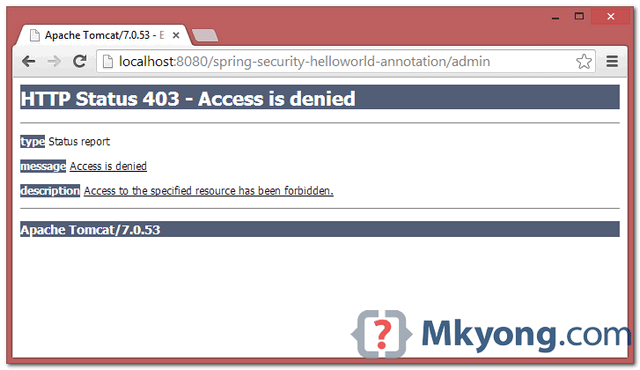
7.3. If username and password is incorrect, error messages will be displayed, and Spring will redirect to this URL /login?error.



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



7.5. For unauthorized user, Spring will display the 403 access denied page. For example, user “mkyong” or “dba” try to access the /admin URL.



Download Source Code

Download it – [spring-security-helloworld-annotation.zip](http://www.mkyong.com/wp-content/uploads/2014/04/spring-security-helloworld-annotation.zip) (12 KB)

References

1. [Spring Security](http://projects.spring.io/spring-security/)
2. [Spring Security Java Config Preview: Web Security](http://spring.io/blog/2013/07/03/spring-security-java-config-preview-web-security/)
3. [Hello Spring MVC Security Java Config](http://docs.spring.io/spring-security/site/docs/3.2.x/guides/hellomvc.html)
4. [Wikipedia : Java Servlet](http://en.wikipedia.org/wiki/Java_Servlet)
5. [Wikipedia : Apache Tomcat](http://en.wikipedia.org/wiki/Apache_Tomcat)
6. [Spring Security Hello World XML Example](http://www.mkyong.com/spring-security/spring-security-hello-world-example/)