Spring Security Custom Login Form Annotation Example

By mkyong (http://www.mkyong.com/author/mkyong/) | April 21, 2014 | Viewed : 123,383 times

In this tutorial, we will convert previous Spring Security custom login form (XML) (http://www.mkyong.com/spring-security/spring-security-form-login-example/) project to a pure annotation-based project.

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
6. Tomcat 7 (Servlet 3.x)

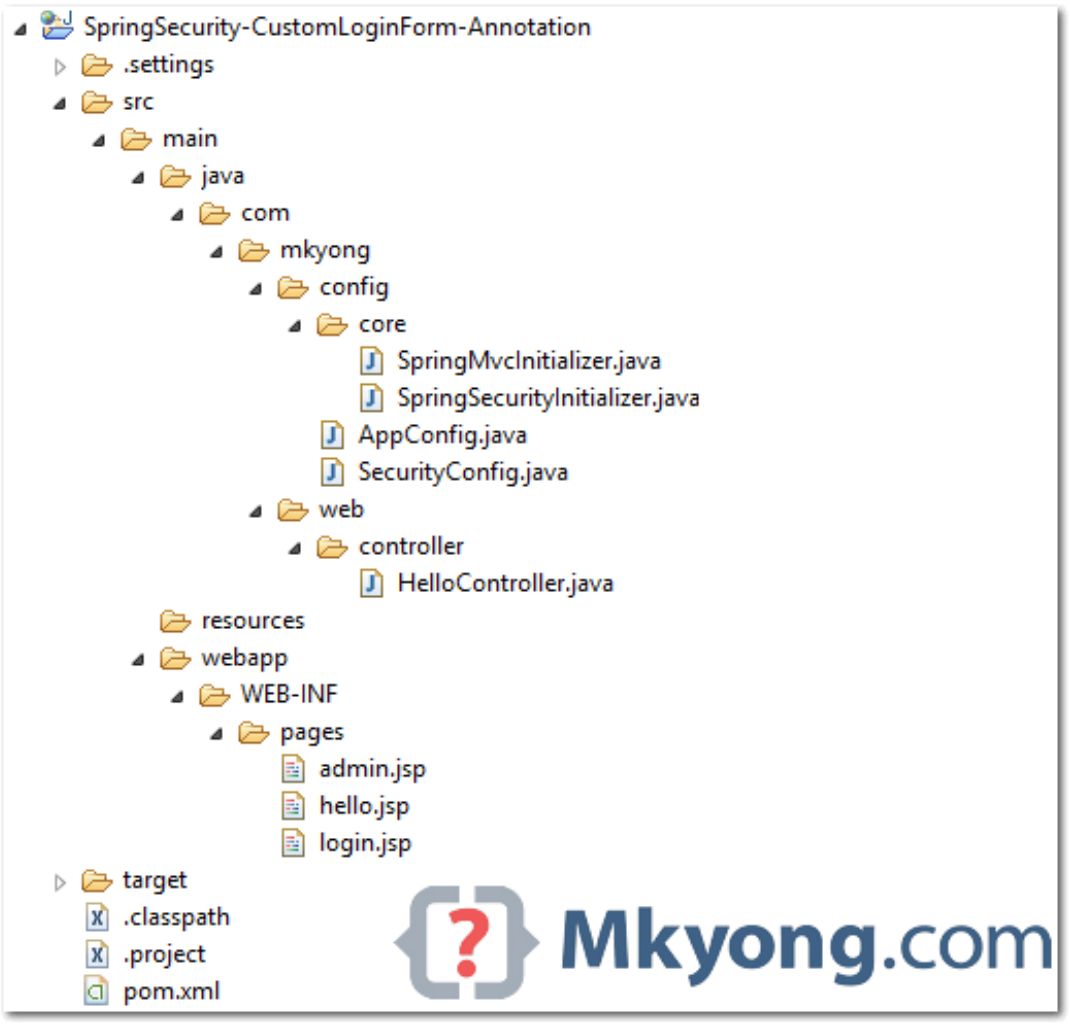
Note
In this example, last Spring Security hello world annotation example (http://www.mkyong.com/spring-security/spring-security-hello-world-annotation-example/) will be reused, enhance it to support a custom login form.

1. Project Demo

Spring Security Custom Login Form Annotation Example

2. Directory Structure

Review the final directory structure of this tutorial.



3. Spring Security Configuration

Spring Security configuration via annotations.

```
SecurityConfig.java
```

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.AuthenticationM
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.WebSecurityConfigurer

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.inMemoryAuthentication()
                    .withUser("mkyong").password("123456").roles("USER");
    }

    //.csrf() is optional, enabled by default, if using WebSecurityConfigurerAdapter construc
    @Override
    protected void configure(HttpSecurity http) throws Exception {

        http.authorizeRequests()
        .antMatchers("/admin/**").access("hasRole('ROLE_USER')")
        .and()
            .formLogin().loginPage("/login").failureUrl("/login?error")
            .usernameParameter("username").passwordParameter("password")
        .and()
            .logout().logoutSuccessUrl("/login?logout")
        .and()
            .csrf();
    }
}
```

The equivalent of the Spring Security XML file :

Markup

```
<http auto-config="true">
  <intercept-url pattern="/admin**" access="ROLE_USER" />
  <form-login
    login-page="/login"
        default-target-url="/welcome"
    authentication-failure-url="/login?error"
    username-parameter="username"
    password-parameter="password" />
  <logout logout-success-url="/login?logout" />
  <!-- enable csrf protection -->
  <csrf/>
</http>

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

4. Custom Login Form

4.1 Page to display the custom login form. If CSRF protection is enabled, remember to add `${_csrf.token}` in both login and logout form.

```
login.jsp
```

Markup

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<head>
<title>Login Page</title>
<style>
.error {
    padding: 15px;
    margin-bottom: 20px;
    border: 1px solid transparent;
    border-radius: 4px;
    color: #a94442;
    background-color: #f2dede;
    border-color: #ebccd1;
}

.msg {
    padding: 15px;
    margin-bottom: 20px;
    border: 1px solid transparent;
    border-radius: 4px;
    color: #31708f;
    background-color: #d9edf7;
    border-color: #bce8f1;
}

#login-box {
    width: 300px;
    padding: 20px;
    margin: 100px auto;
    background: #fff;
    -webkit-border-radius: 2px;
    -moz-border-radius: 2px;
    border: 1px solid #000;
}
</style>
</head>
<body onload='document.loginForm.username.focus();'>

    <h1>Spring Security Custom Login Form (Annotation)</h1>

    <div id="login-box">

        <h2>Login with Username and Password</h2>

        <c:if test="${not empty error}">
            <div class="error">${error}</div>
        </c:if>
        <c:if test="${not empty msg}">
            <div class="msg">${msg}</div>
        </c:if>

        <form name='loginForm'
```

```
            action="<c:url value='j_spring_security_check' />" method='POST'>

            <table>
            <tr>
                <td>User:</td>
                <td><input type='text' name='user' value=''></td>
            </tr>
            <tr>
                <td>Password:</td>
                <td><input type='password' name='pass' /></td>
            </tr>
            <tr>
                    <td colspan='2'>
                                <input name="submit" type="submit" value="submit" />
                                </td>
            </tr>
           </table>

           <input type="hidden"
                     name="${_csrf.parameterName}" value="${_csrf.token}" />
        </form>
    </div>

</body>
</html>
```

4.2 Page to display the welcome message, default page.

hello.jsp

Markup

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

4.3 This page is password protected, only authenticated user is allowed to access.

admin.jsp + logout

Markup

```
<%@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:url value="/j_spring_security_logout" var="logoutUrl" />

        <!-- csrt support -->
    <form action="${logoutUrl}" method="post" id="logoutForm">
        <input type="hidden"
            name="${_csrf.parameterName}"
            value="${_csrf.token}" />
    </form>

    <script>
        function formSubmit() {
            document.getElementById("logoutForm").submit();
        }
    </script>

    <c:if test="${pageContext.request.userPrincipal.name != null}">
        <h2>
            Welcome : ${pageContext.request.userPrincipal.name} | <a
                href="javascript:formSubmit()"> Logout</a>
        </h2>
    </c:if>

</body>
</html>
```

5. Spring MVC Controller

A simple controller.

HelloController.java

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.bind.annotation.RequestParam;
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 Custom Login Form");
        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 Custom Login Form");
        model.addObject("message", "This is protected page!");
        model.setViewName("admin");

        return model;

    }

    //Spring Security see this :
    @RequestMapping(value = "/login", method = RequestMethod.GET)
    public ModelAndView login(
        @RequestParam(value = "error", required = false) String error,
        @RequestParam(value = "logout", required = false) String logout) {

        ModelAndView model = new ModelAndView();
        if (error != null) {
            model.addObject("error", "Invalid username and password!");
        }

        if (logout != null) {
            model.addObject("msg", "You've been logged out successfully.");
        }
        model.setViewName("login");

        return model;
```

```
    }

}
```

6. Initializer Classes

Here are the Initializer classes to make this a pure annotation-based project.

6.1 Initializer class to enable the Spring Security configuration.

SpringSecurityInitializer.java

Java

```
package com.mkyong.config.core;

import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;
public class SpringSecurityInitializer extends AbstractSecurityWebApplicationInitializer {

}
```

6.2 Initializer class to enable the Spring MVC.

SpringMvcInitializer.java

Java

```
package com.mkyong.config.core;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletIniti
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[] { "/" };
    }

}
```

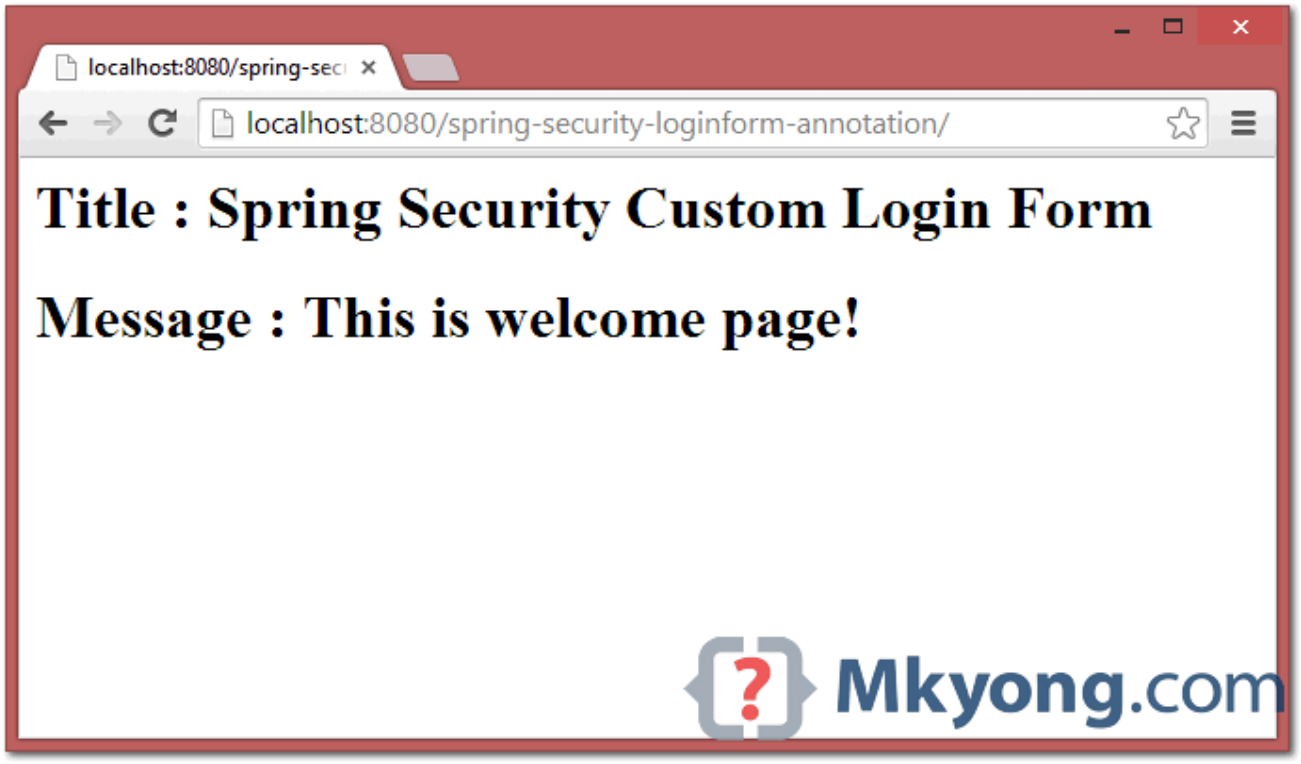
AppConfig.java

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;
    }

}
```

7. Demo

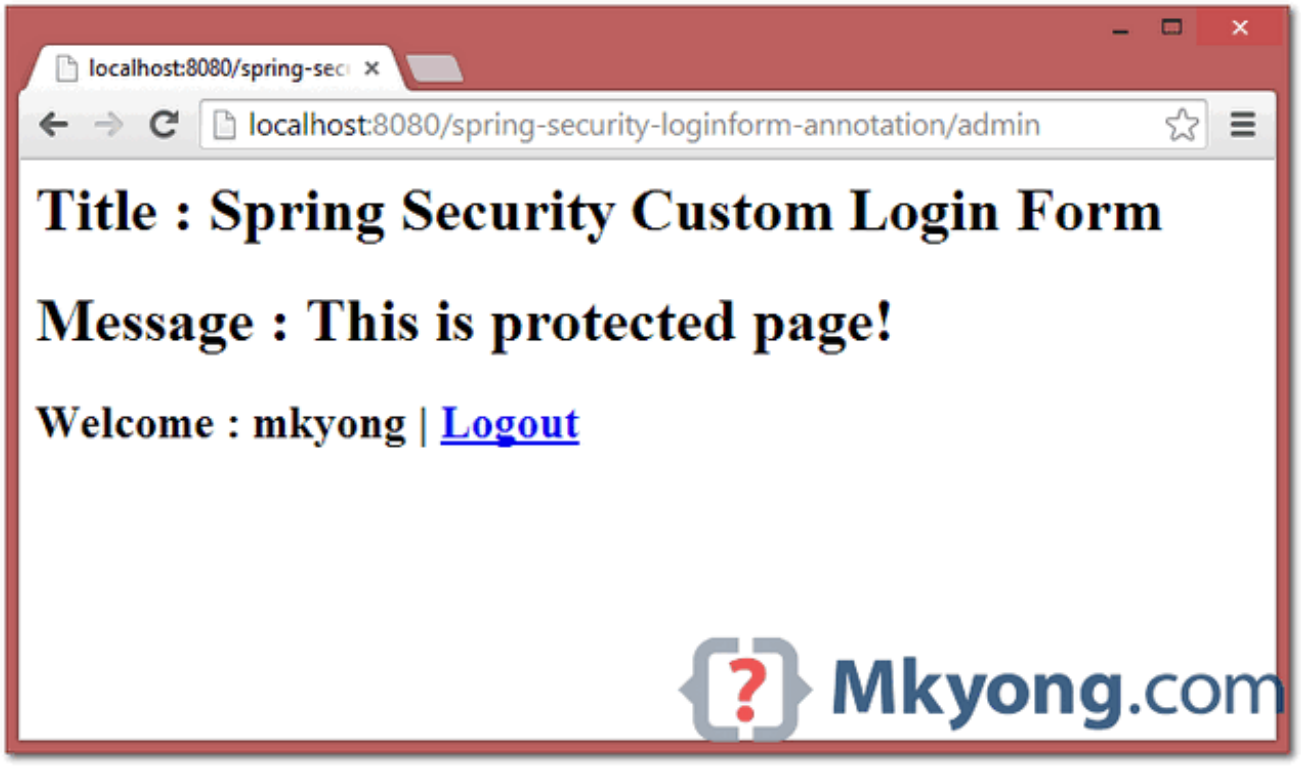
7.1. Welcome Page – *http://localhost:8080/spring-security-loginform-annotation/*



7.2 Try to access `/admin` page, your custom login form is displayed.

7.3. If username and password is incorrect, display `/login?error`.

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



7.5. Try to log out, it will redirect to `/login?logout` page.

Download Source Code

Download it – spring-security-custom-login-form-annotation.zip (http://www.mkyong.com/wp-content/uploads/2014/04/spring-security-custom-login-form-annotation.zip) (19 KB)

References

1. Spring Security Hello World Annotation Example (http://www.mkyong.com/spring-security/spring-security-hello-world-annotation-example/)
2. Creating a Custom Login Form (http://docs.spring.io/spring-security/site/docs/3.2.0.RELEASE/guides/form.html)
3. Spring Security 3.2.0.RC1 Highlights: CSRF Protection (http://spring.io/blog/2013/08/21/spring-security-3-2-0-rc1-highlights-csrf-protection/)
4. Wikipedia : Cross-site request forgery (http://en.wikipedia.org/wiki/Cross-site_request_forgery)

Tags : login form (http://www.mkyong.com/tag/login-form/) spring security (http://www.mkyong.com/tag/spring-security/)

Share this article on

Reader also read :




Spring Security + Hibernate Annotation Example (http://www.mkyong.com/spring-security/spring-security-hibernate-annotation-example/)

Spring Security + Hibernate XML Example (http://www.mkyong.com/spring-security/spring-security-hibernate-xml-example/)

Spring Security : Encoded password does not look like BCrypt (http://www.mkyong.com/spring-security/spring-security-encoded-password-does-not-look-like-bcrypt/)

Spring Security Remember Me Example (http://www.mkyong.com/spring-security/spring-security-remember-me-example/)

Spring Security : Check if user is from remember me cookie (http://www.mkyong.com/spring-security/spring-security-check-if-user-is-from-remember-me-cookie/)

About the Author

mkyong

Founder of Mkyong.com (http://mkyong.com) and HostingCompass.com (http://hostingcompass.com), love Java and open source stuff. Follow him on Twitter (https://twitter.com/mkyong), or befriend him on Facebook (http://www.facebook.com/java.tutorial) or Google Plus (https://plus.google.com/110948163568945735692?rel=author). If you like my tutorials, consider make a donation to these charities (http://www.mkyong.com/blog/donate-to-charity/).

Comments

Дмитрий Швейкус • 2 years ago

Hi, why you don't use the latest spring 4.

7 • Reply • Share ›

johaness vix ➚ Дмитрий Швейкус • 7 months ago

he doesn't know how to.

• Reply • Share ›

Premnath • a year ago

When I execute as it is, after entering username and password I am getting below error

HTTP ERROR 403

Problem accessing /login. Reason:

Expected CSRF token not found. Has your session expired?

4 • Reply • Share ›

Abul Fayes • a year ago

The username parameter is set as "username" and the password parameter is set as "password" in SecurityConfig.java but in the login.jsp the input fields uses "user" and "pass" for the name attributes so it didn't work for me without changing the input name attributes to match the parameter values. Otherwise a great tutorial thanks!

1 • Reply • Share ›

Guest • 2 years ago

This tutorial is not working :(

I got red mark in eclpse on.....

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

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

3.AbstractAnnotationConfigDispatcherServletInitializer

4.AbstractSecurityWebApplicationInitializer

The jars I'm using are:

javax.servlet-api-3.0.1.jar

jstl-1.2.jar

org.springframework.context-3.0.4.RELEASE.jar

spring-beans-3.0.4.RELEASE.jar

see more

1 • Reply • Share ›

Satish Pakalapati • a month ago

HI Mkyong,

can you please explain below question :

how to prevent multiple users login with same username and password in spring?

• Reply • Share ›

Satish Pakalapati • a month ago

Hi,
can you plz explain below question :

how to prevent multiple users login with same username and password in spring security.

• Reply • Share ›

Willians Martins • 2 months ago

Hello mkyong, I'm loving your posts, I copy this code tutorial and is running beautiful, but now, I want to use AuthenticationManagerBuilder getting the value of database, I get this code from another your tutorial, When the password is fixed(encrypt) on the database it is ok, but when I try to save in database one another user the password is write without encrypt(plaintext), can you understand me and help me?

• Reply • Share ›

yas • 6 months ago

This is for those of you who try to learn and make the best of it.

If you are using latest spring security which is "4.0.2.RELEASE"

and if you get blocked by http:xxxx/server_name/context_path/j_spring_security_check

add this to http in SecurityConfig "loginProcessingUrl("/j_spring_security_check")" thanks to Denis Ageev

and if you get blocked by http:xxxx/server_name/context_path/j_spring_security_logout

There are two ways possible:

1) .logoutRequestMatcher(new AntPathRequestMatcher("/j_spring_security_logout")) in SecurityConfig as the same way you did /j_spring_security_check

2) you put this <c:url value="/logout" var="logoutUrl"/>
insetad of <c:url value="/j_spring_security_check" var="logoutUrl"/> in admin.jsp

this is because of this reason
http://docs.spring.io/spring-s...

it says as follow

The URL that triggers log out to occur (default is /logout). If CSRF protection is enabled (default), then the request must also be a POST. For for information, please consult the JavaDoc.

• Reply • Share ›

johaness vix • 7 months ago

this is not very flexible if you need for example angular js for doing auth besides those html ugly spring tags.

• Reply • Share ›

Aspen Sugan • 9 months ago

Can anyone please reply me asap.

• Reply • Share ›

Jeff • a year ago

Hello Mr Mkyong

When I run the app on sever (Tomcat), in the browser I get this message "Ressource not available" , and the url is "http://localhost:8002/spring-security-loginform-annotation/"

What's the problem ?
Please help me. Thank you

• Reply • Share ›

PJ ➚ Jeff • 8 months ago

Check your main console (Tomcat) and check status of your application.

• Reply • Share ›

akshobhya kallapur • a year ago

Hi , thanks for this example .. but when I give correct username and password it is going to index page, May I know y?

• Reply • Share ›

Guest • a year ago

Thank you, good tutorial

• Reply • Share ›

Nacho • a year ago

Not working...

• Reply • Share ›

Juliuz • a year ago

Hello, the text for bad credentials and succesful logout isent in colour. Can you help?

• Reply • Share ›

Pawel • a year ago

wery good! It's really worked!

• Reply • Share ›

Priyanka Kulathilaka • 2 years ago

I got error and posted here it

[ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-

plugin:2.3.2:compile (default-compile) on project : Compilation failure: Compilation failure:
[ERROR] \..SecurityConfig.java:[12,7] error: cannot access Filter
[ERROR] \..SpringSecurityInitializer.java:[5,7] error: cannot access ServletException
[ERROR] -> [Help 1]

http://stackoverflow.com/quest...

• Reply • Share ›

Tushar Bhatt • 2 years ago

I am not able to deploy the application in Tomcat, on researching I found that there is an error in SecurityConfig.java file. Error is : "
Could not autowire. No beans of 'AuthenticationManagerBuilder' type found.
". Can somebody help me to resolve the issue.

• Reply • Share ›

Virmerson Santos • 2 years ago

To function correctly, and avoid 404 error I switched j_spring_security_logout by "/ logout" and also j_spring_security_check by "/ login" in login.jsp file and admin.jsp

• Reply • Share ›

Denis Ageev Virmerson Santos • a year ago

add this to http in SecurityConfig "loginProcessingUrl("/j_spring_security_check")"

• Reply • Share ›

RaunakShakya Denis Ageev • 9 months ago

thank you. your suggestion helped me solve a problem that had been cracking my brains for the past few days.

• Reply • Share ›

Philip Ardley • 2 years ago

The " spring-security-custom-login-f... (19 KB)" can't be imported to Eclipse ad run on Tomcat 7. These are the steps I took to make the imported (from file system) project work:

1) In the pom.xml change the javax.servlet.version from "3.1.0" to "3.0.1".
2) In the pom.xml change the jdk.version from "1.6" to "1.7".
3) Open the project properties, -> Java Build Path -> Libraries, remove all those links to the maven repository.
4) Right click your project, -> Configure -> Convert to maven.
5) Make sure your Tomcat library is in Java Build Path -> Libraries and also is the target runtime.

6) Run the project on the server like normal.

There are very often these types of problems with maven example projects from this website.

• Reply • Share ›

Guest • 2 years ago

The "last Spring Security hello world annotation example" can't be imported to Eclipse ad run on Tomcat 7. These are the steps I took to make the imported (from file system) project work:

1) In the pom.xml change the javax.servlet.version from "3.1.0" to "3.0.1".
2) In the pom.xml change the jdk.version from "1.6" to "1.7".
3) Open the project properties, -> Java Build Path -> Libraries, remove all those links to the maven repository.
4) Right click your project, -> Configure -> Convert to maven.
5) Make sure your Tomcat library is in Java Build Path -> Libraries and also is the target runtime.

6) Run the project on the server like normal.

There are very often these types of problems with maven example projects from this website.

• Reply • Share ›

Raghavulu Bussa • 2 years ago

Hi,

I'm getting

No mapping found for HTTP request with URI [/{app-context}/j_spring_security_check] in DispatcherServlet with name 'dispatcher' when using the java config while it worked with xml config

• Reply • Share ›

JukaBarros ➚ Raghavulu Bussa • 10 months ago

If you use Spring Security 3.2.x you should change "j_spring_security_check" to "login" and "j_spring_security_logout" to "logout"

And the parameter (username and password) in SecurityConfig.java must have the same name in input in login.jsp. Example:

SecurityConfig.java

usernameParameter("name").passwordParameter("pass")

login.jsp

```
<tr>
<td>User:</td>
<td><input type="text" name="name" value=""></td>
</tr>
<tr>
<td>Password:</td>
<td><input type="password" name="pass"/></td>
</tr>
```

• Reply • Share ›

RiccardoC ➚ Raghavulu Bussa • 10 months ago

as far as I know, this example won't work with Spring Security 3.2.x; they changed some urls.

j_spring_security_check is now /login (POST only, unless you configure it differently)

you have to provide your mapping for logout, this is is accomplished by something like (after usernameParameter("username").passwordParameter("password")):

.and().logout().logoutUrl("/logout");

• Reply • Share ›

Juan Manuel Flores ➚ Raghavulu Bussa • a year ago

Same error. Could you make it work?

• Reply • Share ›

Bono ➚ Raghavulu Bussa • 2 years ago

I've got the same error. Nice example but not working properly.

spring-context, spring-webmvc: 4.0.5.RELEASE
spring-security-core, spring-security-config, spring-security-web: 3.2.4.RELEASE

• Reply • Share ›

Favorites Links

Android Getting Started (http://developer.android.com/training/index.html)
Google App Engine – Java (https://cloud.google.com/appengine/docs/java/)

Spring 2.5.x Documentation (http://docs.spring.io/spring/docs/2.5.x/reference/index.html)
Spring 3.2.x Documentation (http://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/)
Spring 4.1.x Documentation (http://docs.spring.io/spring/docs/4.1.x/spring-framework-reference/html/)
Java EE 5 Tutorial (http://docs.oracle.com/javaee/5/tutorial/doc/docinfo.html)
Java EE 6 Tutorial (http://docs.oracle.com/javaee/6/tutorial/doc/docinfo.html)
Java EE 7 Tutorial (https://docs.oracle.com/javaee/7/tutorial/index.html)
Java 6 API (http://docs.oracle.com/javase/6/docs/api/overview-summary.html)
Java 7 API (http://docs.oracle.com/javase/7/docs/api/overview-summary.html)
Java 8 API (http://docs.oracle.com/javase/8/docs/api/overview-summary.html)
JSF Home Page (https://javaserverfaces.java.net/)
JSP Home Page (https://jsp.java.net/)
Maven Central Repository (http://search.maven.org/)
Hibernate ORM (http://hibernate.org/orm/)
JAX-WS Home Page (https://jax-ws.java.net/)
JAX-RS Home Page (Jersey) (https://jax-ws.java.net/)

Partners & Bookmarks

Java Code Geeks (http://www.javacodegeeks.com/)
TestNG Founder (http://beust.com/weblog/)
DZone (https://dzone.com)

About Mkyong.com

Mkyong.com is for Java and J2EE developers, all examples are simple and easy to understand, and well tested in my development environment.

Mkyong.com is created, written by, and maintained by Yong Mook Kim, aka Mkyong. It is built on WordPress (https://wordpress.org/), hosted by Liquid Web (http://mkyong.com/go/liquidweb/), and the caches are served by CloudFlare CDN.