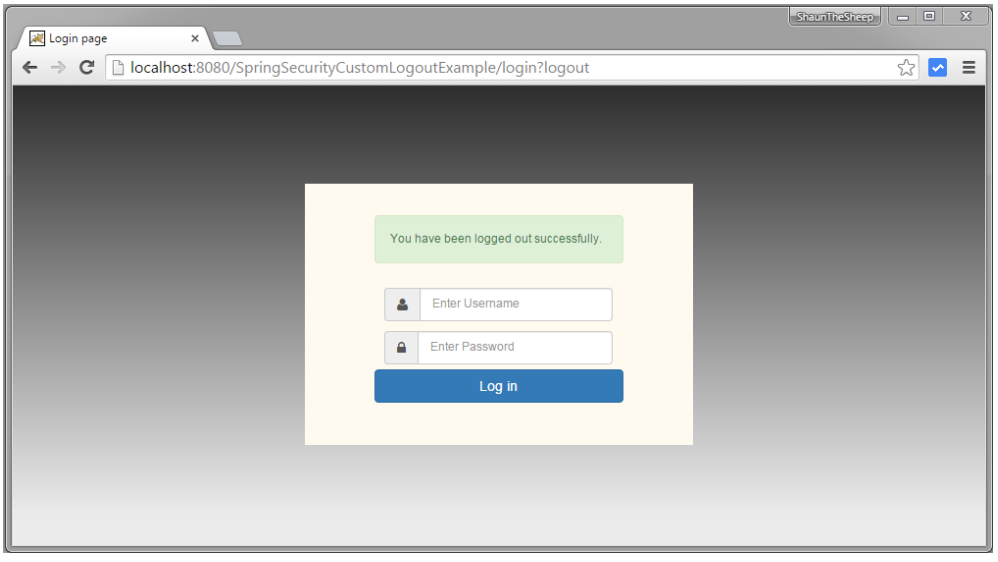
WebSystique
learn together

Spring Security 4 Logout Example

July 28, 2015 websystiqueadmin

This post shows you how to **programatically logout** a user in Spring Security. This works well with browser back button too.



Generally, In your views, you should be providing a simple **logout link** to logout a user, something like shown below:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO
    <title>Admin page</title>
</head>
<body>
```

Recent Posts

Spring 4 MVC+AngularJS CRUD Application using ngResource

Angularjs Server Communication using ngResource-CRUD Application

AngularJS Custom-Directives controllers, require option guide

AngularJS Custom-Directives transclude, ngTransclude guide

AngularJS Custom-Directives replace option guide

```
    Dear <strong>${user}</strong>, Welcome to Admin Page.
    <a href="<c:url value="/logout" />">Logout</a>
</body>
</html>
```

Nothing fancy about it. Now , We just need to map this **/logout** link in our controller. Create a new method like shown below:

```
@RequestMapping(value="/logout", method = RequestMethod.GET)
public String logoutPage (HttpServletRequest request, HttpServletRe
    Authentication auth = SecurityContextHolder.getContext().getAut
    if (auth != null){
        new SecurityContextLogoutHandler().logout(request, response
    }
    return "redirect:/login?logout";//You can redirect wherever you
}
```

Here firstly we identified if user was authenticated before using **SecurityContextHolder.getContext().getAuthentication()**. If he/she was, then we called **SecurityContextLogoutHandler().logout(request, response, auth)** to logout user properly.

This **logout** call performs following:

- Invalidates HTTP Session ,then unbinds any objects bound to it.
- Removes the Authentication from the SecurityContext to prevent issues with concurrent requests.
- Explicitly clears the context value from the current thread.

That's it. You don't need anything else anywhere in your application to handle logout. Notice that you don't even need to do anything special in your spring configuration(xml or annotation based), shown below just for information:

```
package com.websystique.springsecurity.configuration;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authenticatic
import org.springframework.security.config.annotation.web.builders.
import org.springframework.security.config.annotation.web.configura
import org.springframework.security.config.annotation.web.configura

@Configuration
@EnableWebSecurity
public class SecurityConfiguration extends WebSecurityConfigurerAda


    @Autowired
    public void configureGlobalSecurity(AuthenticationManagerBuilde
        auth.inMemoryAuthentication().withUser("bill").password("ab
        auth.inMemoryAuthentication().withUser("admin").password("r
```

```
        auth.inMemoryAuthentication().withUser("dba").password("roc
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {

      http.authorizeRequests()
        .antMatchers("/", "/home").permitAll()
        .antMatchers("/admin/**").access("hasRole('ADMIN')")
        .antMatchers("/db/**").access("hasRole('ADMIN') and hasRole
        .and().formLogin().loginPage("/login")
        .usernameParameter("ssoId").passwordParameter("password")
        .and().exceptionHandling().accessDeniedPage("/Access_Denied
    }
}
```

There is no special logout handling mentioned in above configuration.

Above security configuration in XML configuration format would be:

```
<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/security http://www.sprir

    <http auto-config="true" >
        <intercept-url pattern="/" access="hasRole('USER')" />
        <intercept-url pattern="/home" access="hasRole('USER')" />
        <intercept-url pattern="/admin**" access="hasRole('ADMIN')"
        <intercept-url pattern="/dba**" access="hasRole('ADMIN') ar
        <form-login  login-page="/login"
                     username-parameter="ssoId"
                     password-parameter="password"
                     authentication-failure-url="/Access_Denied" />
    </http>

    <authentication-manager >
        <authentication-provider>
            <user-service>
                <user name="bill"  password="abc123"  authorities="
                <user name="admin" password="root123" authorities="
                <user name="dba"   password="root123" authorities="
            </user-service>
        </authentication-provider>
    </authentication-manager>

</beans:beans>
```

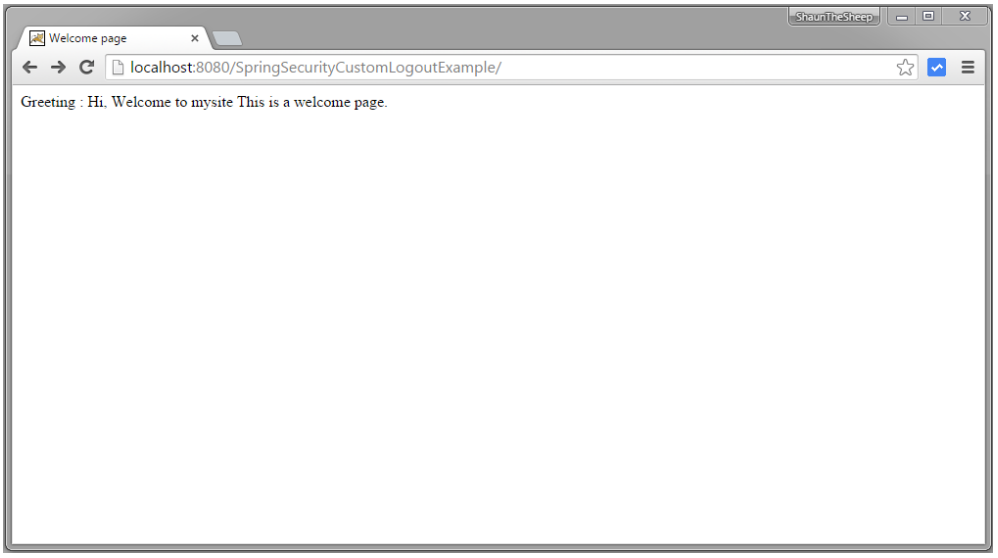
Rest of application code is same as mentioned in every post in this series. This Post (and actually every post in this series) shows this logout in action.

Deploy & Run

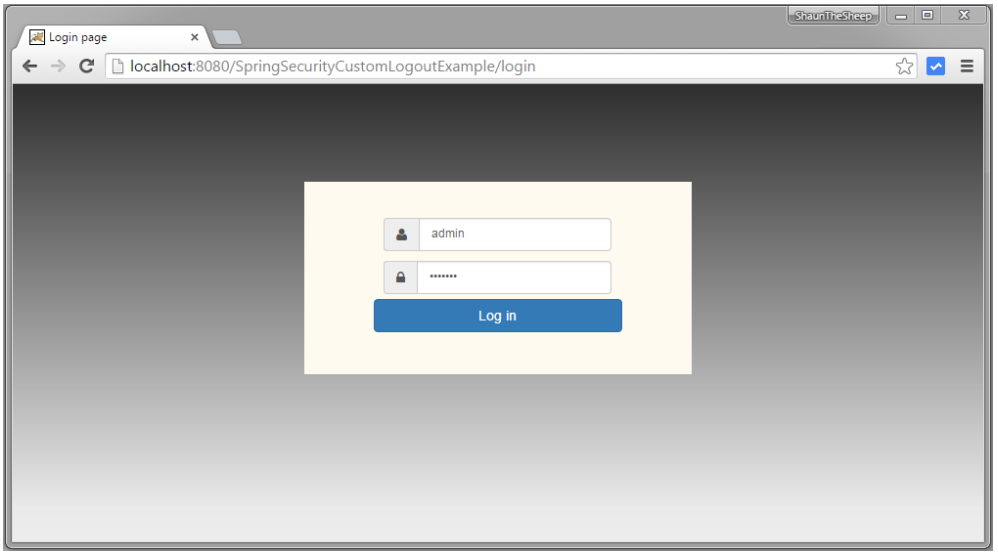
Download complete code of this project using download button shown at the bottom of this post. Build and deploy it on Servlet 3.0 container(Tomcat7/8).

Open your browser, go to

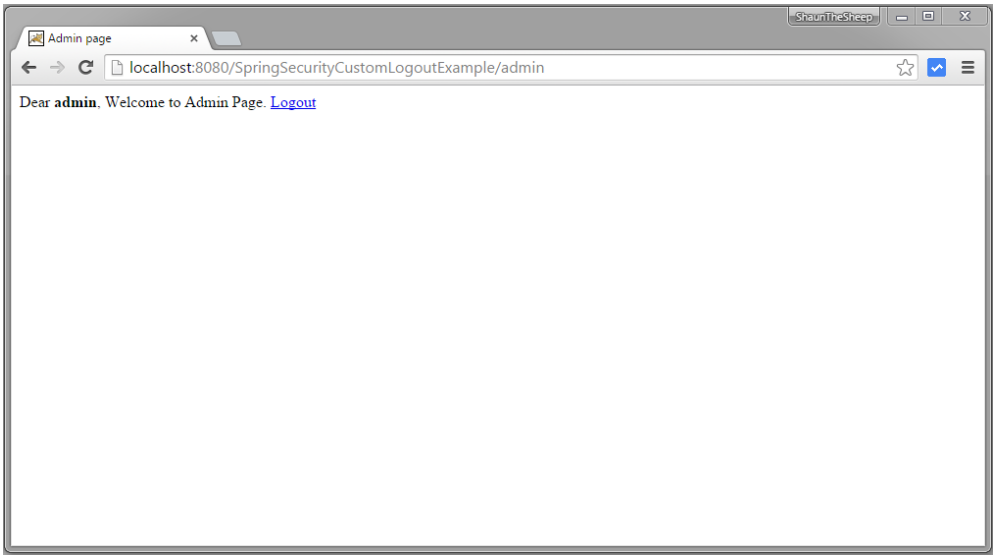
http://localhost:8080/SpringSecurityCustomLogoutExample/, home page will be shown



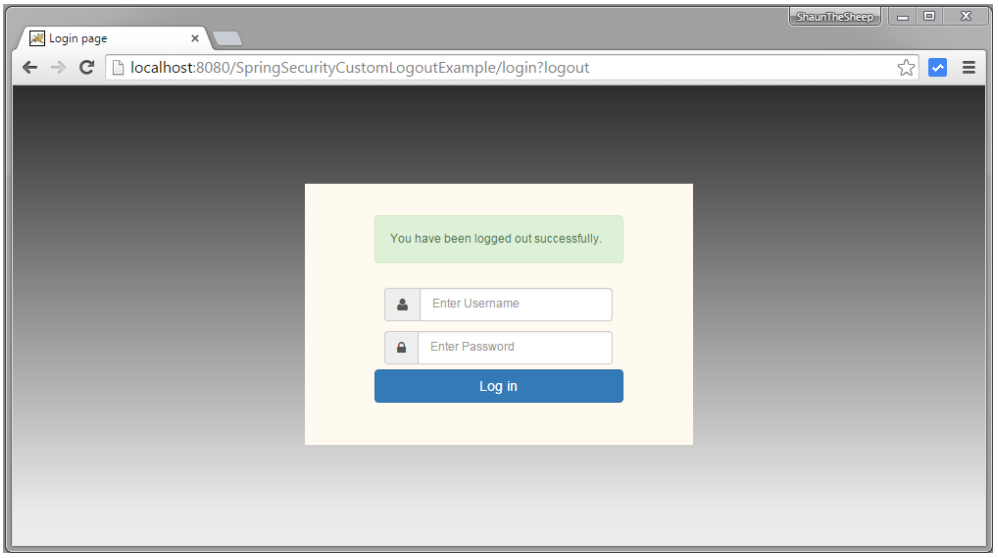
Now go to **http://localhost:8080/SpringSecurityCustomLogoutExample/admin**, you will be prompted for login



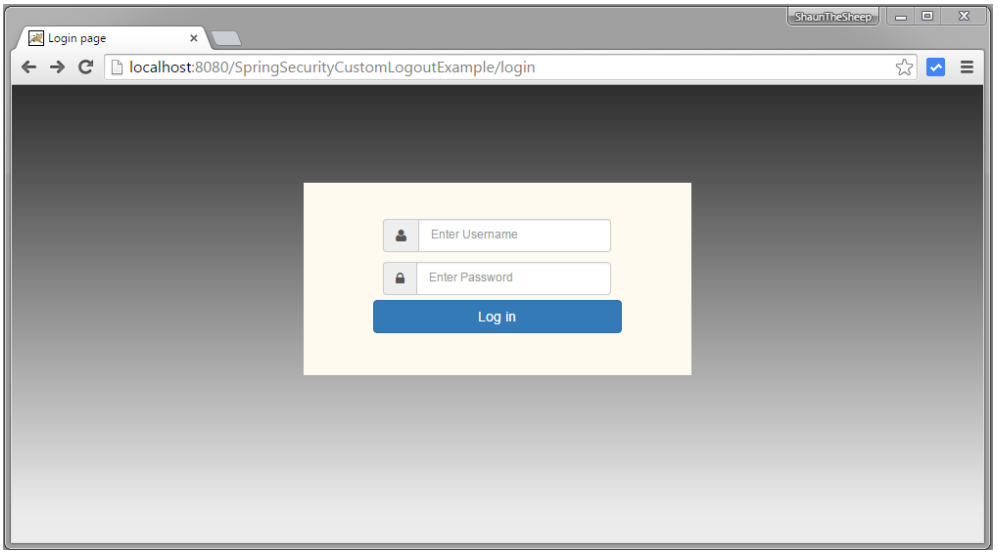
provide credentials, submit, you will see admin page.



click on logout, you will be taken to login page.



click on **browser back button**, you will remain at login screen.



That's it. Next post shows you how to show/hide parts of jsp/view based on logged-in user's roles, using Spring Security tags.

Download Source Code

Download Now!

References

- Spring Security 4 Project Page
- Spring Security 4 Reference Manual

websystiqueadmin

If you like tutorials on this site, why not take a step further and connect me on Facebook , Google Plus & Twitter as well? I would love to hear your thoughts on these articles, it will help me improve further our learning process.

If you appreciate the effort I have put in this learning site, help me improve the visibility of this site towards global audience by sharing and linking this site from within and beyond your network. You & your friends can always link my site from your site on www.websystique.com, and share the learning.

After all, we are here to learn together, aren't we?

Related Posts:

1. **Spring Security 4 Secure View Fragments using taglibs**
2. **Spring Security 4 Hibernate Role Based Login Example**
3. **Spring Security 4 Method security using @PreAuthorize,@PostAuthorize, @Secured, EL**
4. **Spring Security 4 Hello World Annotation+XML Example**

spring-security. permalink.

← Spring Security 4 Custom Login Form Annotation+XML Example

Spring Security 4 Secure View Fragments using taglibs →