# Spring Security 4 Custom Login Form Annotation+XML Example

We will demonstrate creating **custom login form** in Spring Security 4 and integrate it in Spring MVC web application.

In [Spring Security 4 Hello World Annotation+xml example](file:///D:\Q3_Spring_Security-Level2\ws\0001SpringSecurity4HelloWorldAnnotation), we have seen the default login form provided by Spring Security in case we don’t specify one. In this post, we will create our own Custom login form.

Basically, the idea is, in Security Configuration, attach a call to **loginPage(URL)** function with formLogin() like shown below

|  |
| --- |
| .and().formLogin().loginPage("/login") |

And then, Map this **‘/login’** URL in your Spring MVC Controller which will return the login view defined by you. Now, on login attempt, the specified login view will be displayed.Rest of the login functionality remains same.

Below provided is complete example for this scenario.

**Following technologies being used:**

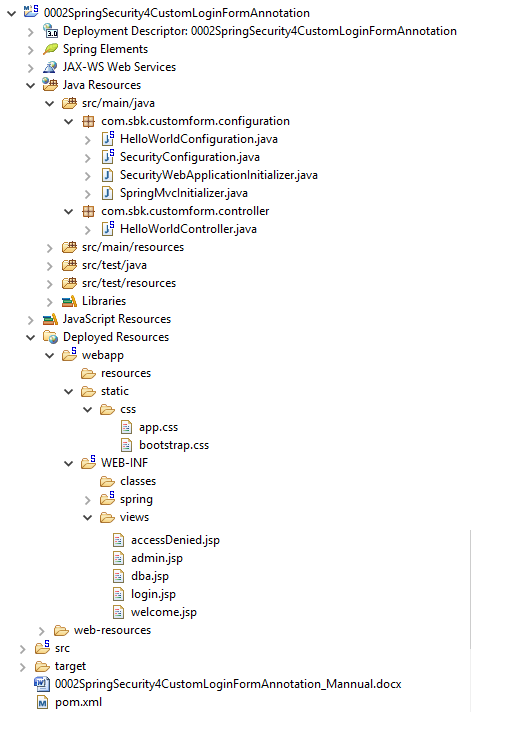
* Spring 4.1.6.RELEASE
* Spring Security 4.0.1.RELEASE
* Maven 3
* JDK 1.7
* Tomcat 7
* STS

Let’s begin.

1. **Create a Spring MVC project using STS**
2. **Delete the web.xml as we will be doing java Configuration**
3. **Delete the spring folder from WEB-INF**
4. **Update pom.xml**

* *First thing to notice here is the maven-war-plugin declaration. As we are using full annotation configuration,****we don’t even use web.xml****, so we will need to configure this plugin in order to avoid maven failure to build war package. We are using latest versions(at time of writing) of Spring and Spring Security.*
* *Along with that, we have also included JSP/Servlet/Jstl dependencies which we will be needing as we are going to use servlet api’s and jstl view in our code. In general, containers might already contains these libraries, so we can set the scope as ‘provided’ for them in pom.xml.*

1. **Following is the project structure**

****

1. **Let’s now add the content mentioned in above structure explaining each in detail**.
2. **Create *com.sbk.customform.configuratio.SecurityConfiguration*** ** class**

The first and foremost step to add spring security in our application is to create **Spring Security Java Configuration**. This configuration creates a Servlet Filter known as the springSecurityFilterChain which is responsible for all the security (protecting the application URLs, validating submitted username and passwords, redirecting to the log in form, etc) within our application.

This class is same as the one in [previous post](http://websystique.com/spring-security/spring-security-4-hello-world-annotation-xml-example/). Only differences are shown below

|  |
| --- |
| .and().formLogin().loginPage("/login")          .usernameParameter("ssoId").passwordParameter("password")          .and().csrf() |

This code creates a custom login page with **‘/login’** url, which will accept **ssoId** as username and **password** Http request parameters. We have also shown a call to csrf() which is optional as it is by default active in Spring Security 4. This call is, however, required if you want to disable [CSRF protection](http://docs.spring.io/spring-security/site/docs/3.2.x/reference/htmlsingle/#csrf) by using csrf().disable()although it is not a good idea to disable it.

**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/beans/spring-beans-4.1.xsd>  <http://www.springframework.org/schema/security> <http://www.springframework.org/schema/security/spring-security-4.0.xsd>">         <http auto-config="true" >          <intercept-url pattern="/" access="permitAll" />          <intercept-url pattern="/home" access="permitAll" />          <intercept-url pattern="/admin\*\*" access="hasRole('ADMIN')" />          <intercept-url pattern="/dba\*\*" access="hasRole('ADMIN') and hasRole('DBA')" />          <form-login  login-page="/login" username-parameter="ssoId" password-parameter="password" authentication-failure-url="/Access\_Denied" />          <csrf/>      </http>      <authentication-manager >          <authentication-provider>              <user-service>                  <user name="bill"  password="abc123"  authorities="ROLE\_USER" />                  <user name="admin" password="root123" authorities="ROLE\_ADMIN" />                  <user name="dba"   password="root123" authorities="ROLE\_ADMIN,ROLE\_DBA" />              </user-service>          </authentication-provider>      </authentication-manager>    </beans:beans> |

#### Register the springSecurityFilter with war com.sbk.customform.configuration.SecurityWebApplicationInitializer

Below specified initializer class registers the springSecurityFilter [created in previous step] with application war.



|  |
| --- |
| package com.websystique.springsecurity.configuration;  import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;  public class SecurityWebApplicationInitializer extends AbstractSecurityWebApplicationInitializer {    } |

This class is exactly same as in Previous demo[0001SpringSecurity4HelloWorldAnnotation](file:///D:\Q3_Spring_Security-Level2\ws\0001SpringSecurity4HelloWorldAnnotation)

**Above setup in XML configuration format would be(In 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> |

#### Add Controller com.sbk.customform.controller

#### 

In comparison to [previous demo](file:///D:\Q3_Spring_Security-Level2\ws\0001SpringSecurity4HelloWorldAnnotation\src\main\java\com\sbk\hellosecurity\controller\HelloWorldController.java) , only changes are new **loginPage** method to handle **‘/login’** requests and adapting**logout** to redirect to login page on logout, as shown below:

|  |
| --- |
| @RequestMapping(value = "/login", method = RequestMethod.GET)  public String loginPage() {      return "login";  }  @RequestMapping(value="/logout", method = RequestMethod.GET)  public String logoutPage (HttpServletRequest request, HttpServletResponse response) {      Authentication auth = SecurityContextHolder.getContext().getAuthentication();      if (auth != null){          new SecurityContextLogoutHandler().logout(request, response, auth);      }      return "redirect:/login?logout";  } |

#### Add SpringMVC Configuration Class

**com.sbk.customform.configuration**



Changes from [previous demo](file:///D:\Q3_Spring_Security-Level2\ws\0002SpringSecurity4CustomLoginFormAnnotation\src\main\java\com\sbk\customform\configuration\HelloWorldConfiguration.java) are extend from **WebMvcConfigurerAdapter** [just a convenience class] and implementing method **addResourceHandlers** which handles static resources(CSS/images/..) to be used in views.

#### Add Initializer class(This class is exactly same as in [previous post](file:///D:\Q3_Spring_Security-Level2\ws\0002SpringSecurity4CustomLoginFormAnnotation\src\main\java\com\sbk\customform\configuration\SpringMvcInitializer.java).)

**com.sbk.hellosecurity.configuration**



Notice that above initializer class extends AbstractAnnotationConfigDispatcherServletInitializerwhich is the base class for all WebApplicationInitializer implementations. Implementations of WebApplicationInitializer configures ServletContext programatically, for Servlet 3.0 environments. It means we won’t be using web.xml and we will deploy the app on Servlet 3.0 container.

#### Add Views

**

login.jsp  


login.jsp additionally contains CSS for login panel layout.

***Notice the CSRF (*Cross Site Request Forgery*)related line in above jsp:***

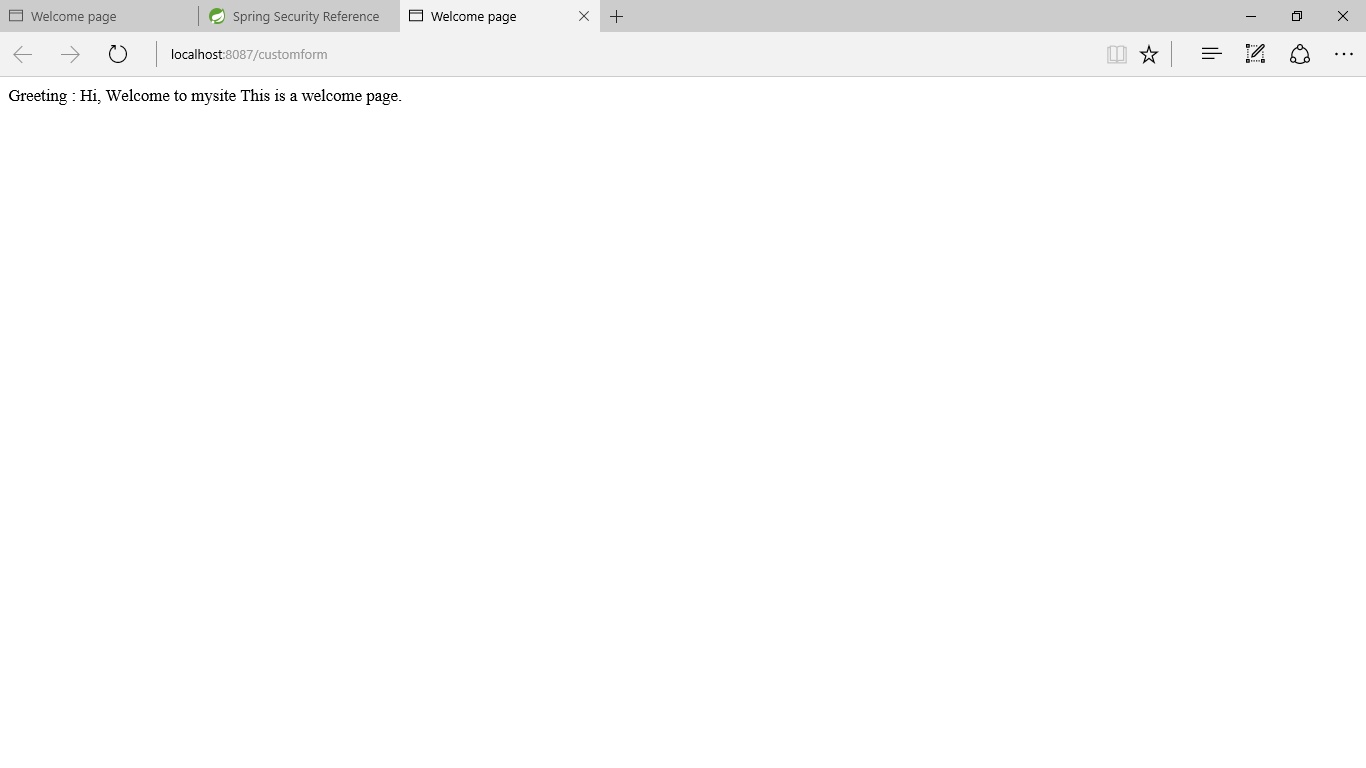
|  |
| --- |
| ***<input type="hidden" name="${\_csrf.parameterName}" value="${\_csrf.token}" /></strong>*** |

***This is required to protect against***[***CSRF attacks***](http://docs.spring.io/spring-security/site/docs/3.2.x/reference/htmlsingle/#csrf)***.***

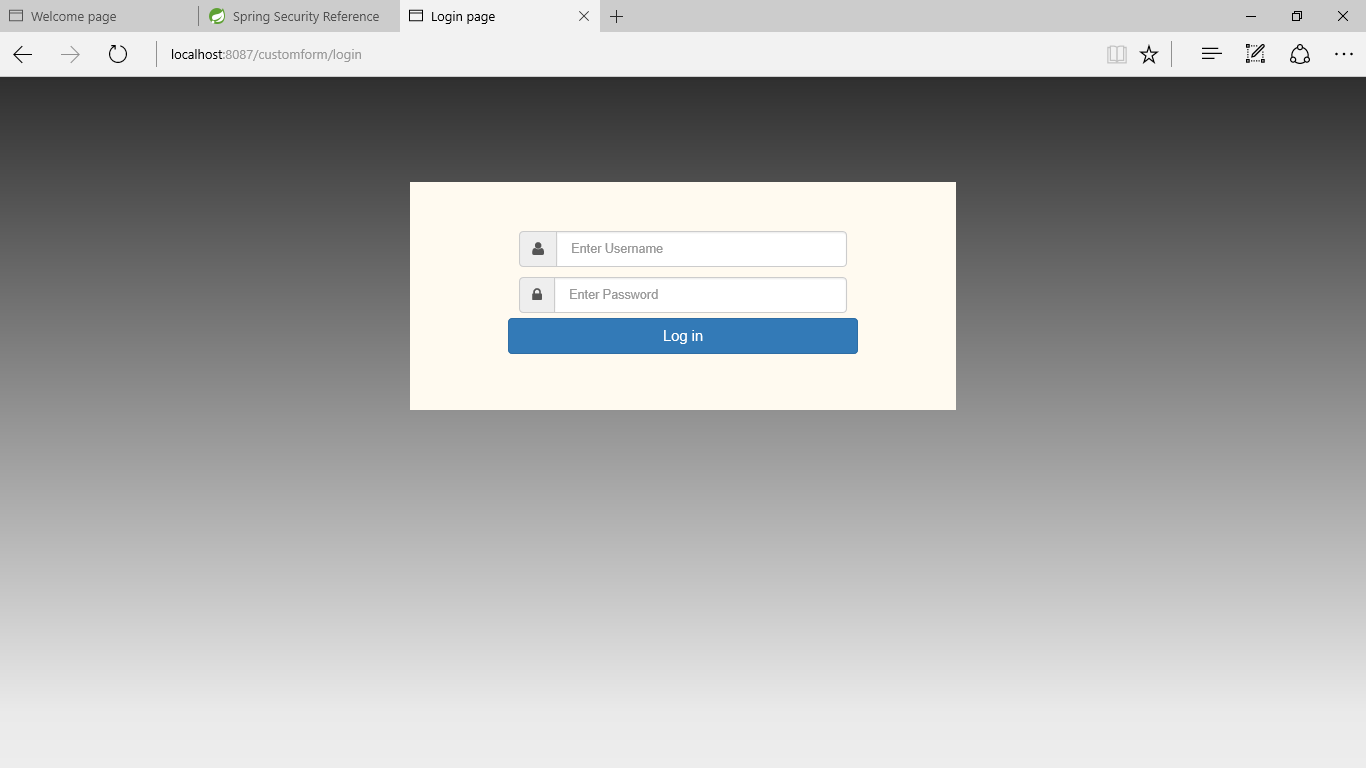
#### Build and Deploy the application

***NOTE : Before deploying, check the java build path(1.7), Check java compiler(1.7), java facets(jdk1.7,web module version 3.0), targeted runtime(tom cat7) and Add Maven dependency to build path***

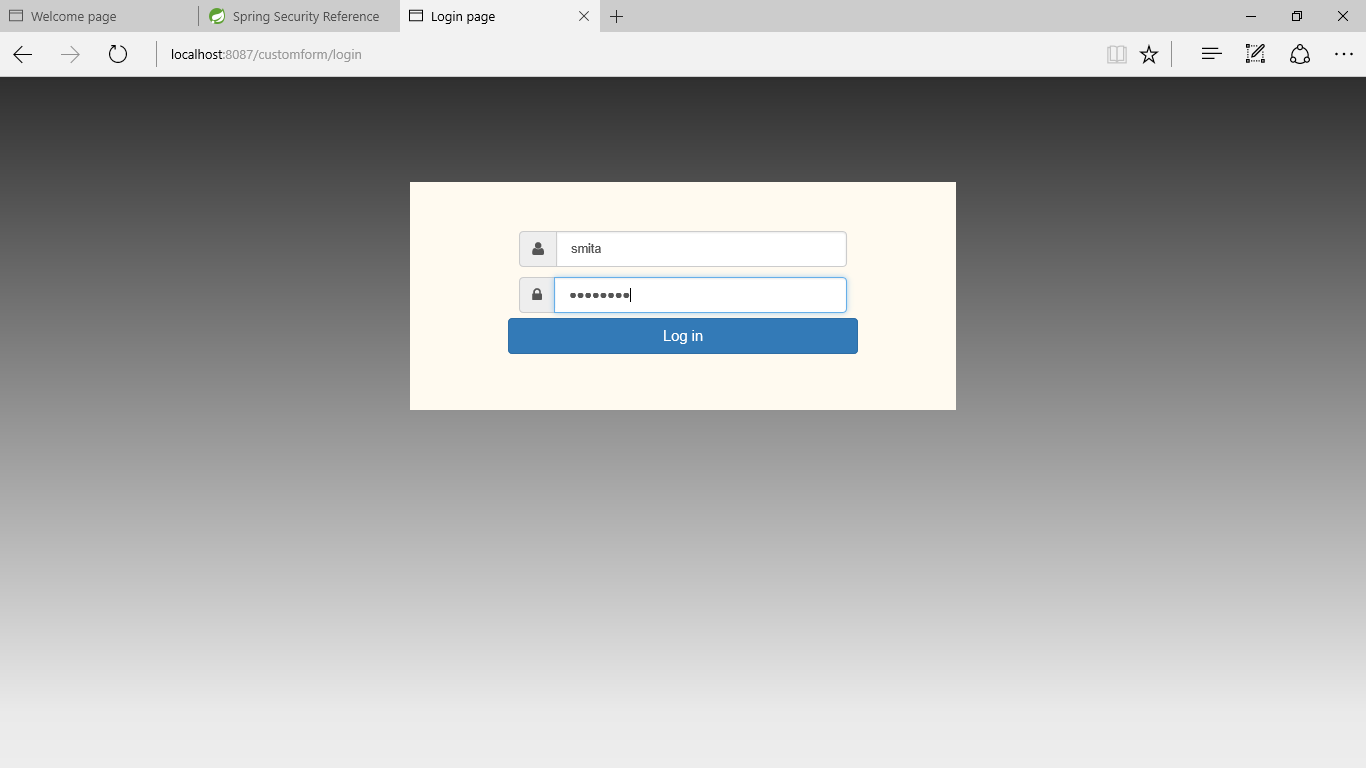
#### Run the application(http://localhost:8087/customform/)



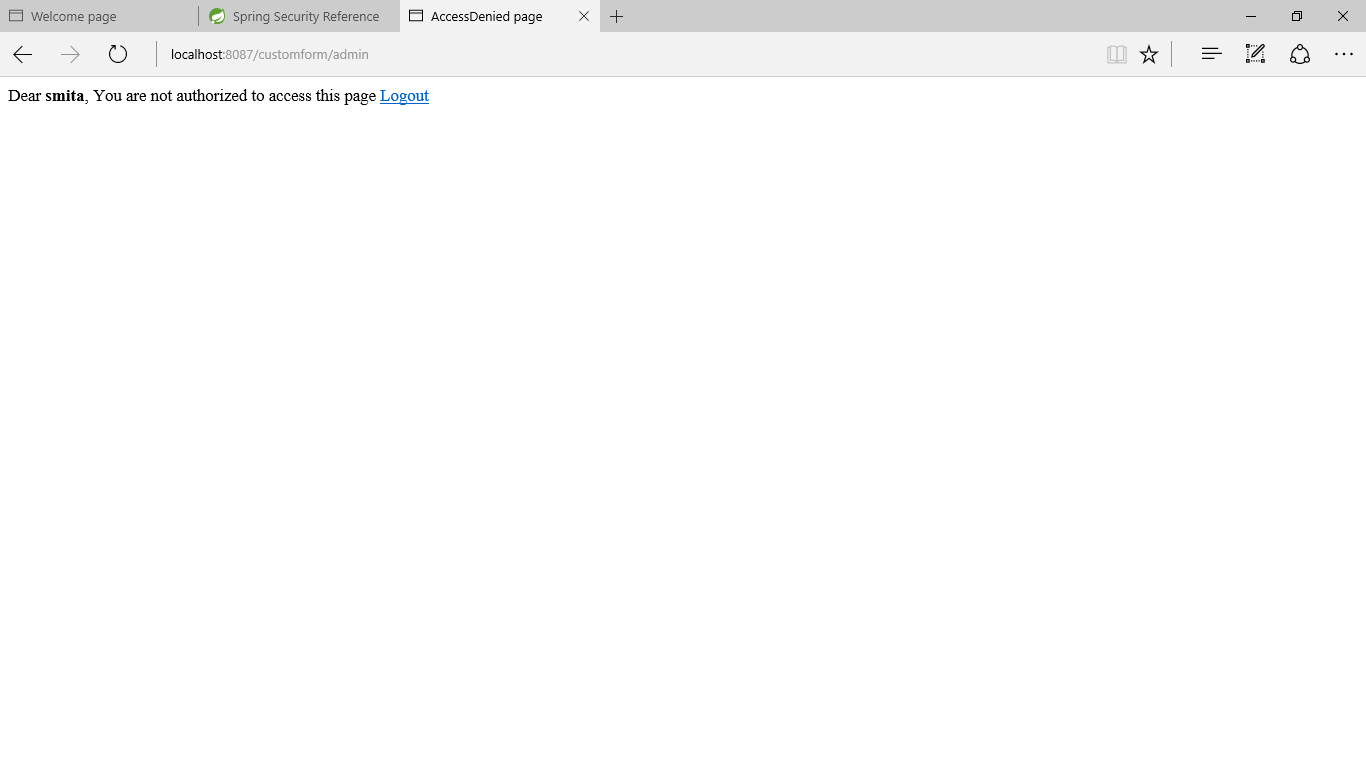
Now try to access admin page on http://localhost:8087/customform/admin, you will be prompted for login.

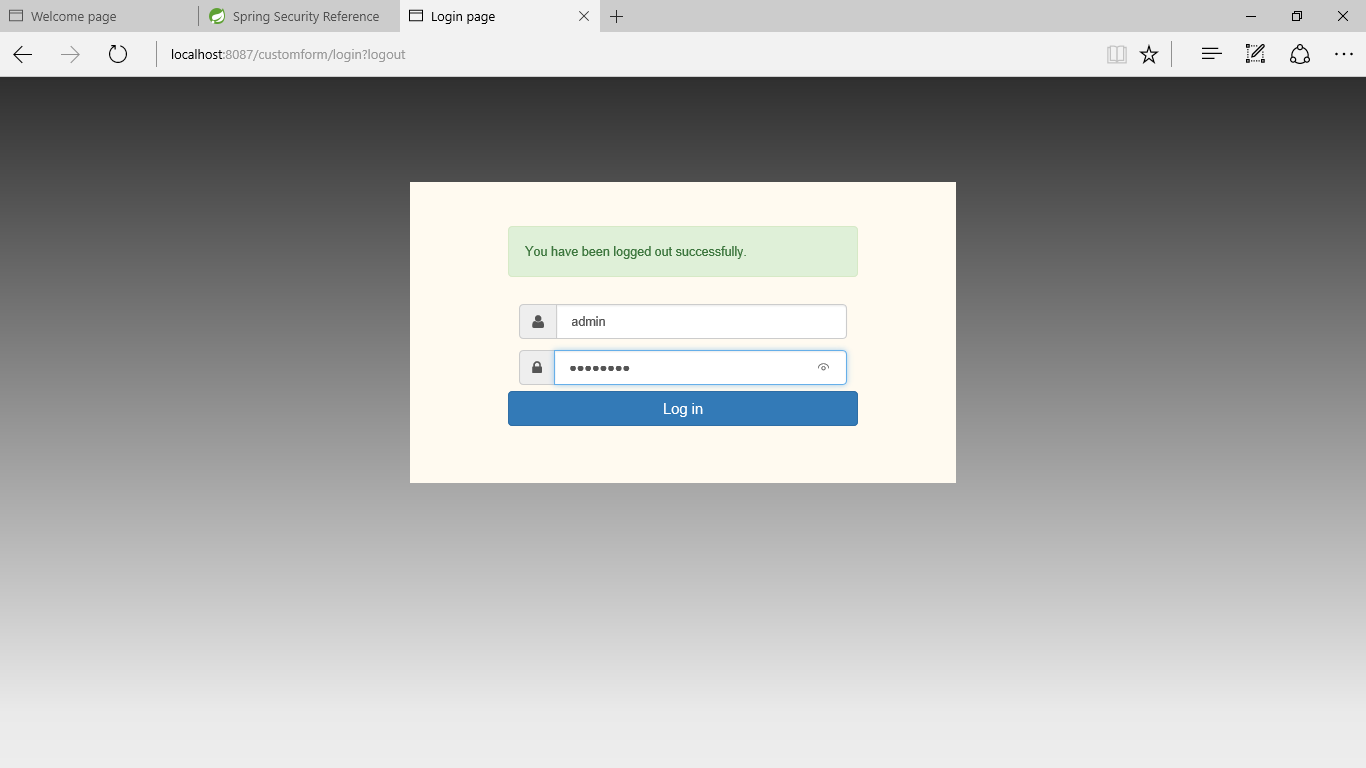


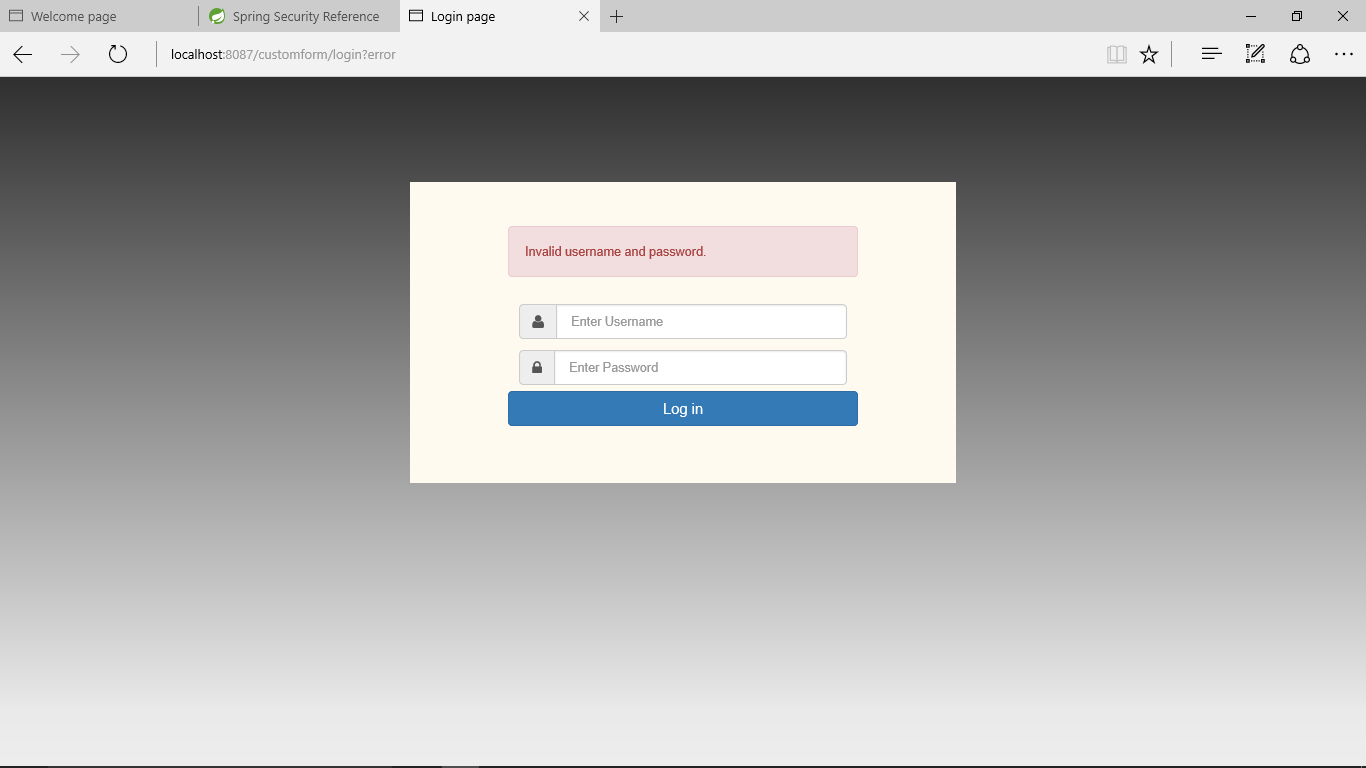
Provide credentials of a ‘USER’ role.*enter [smita,password]*



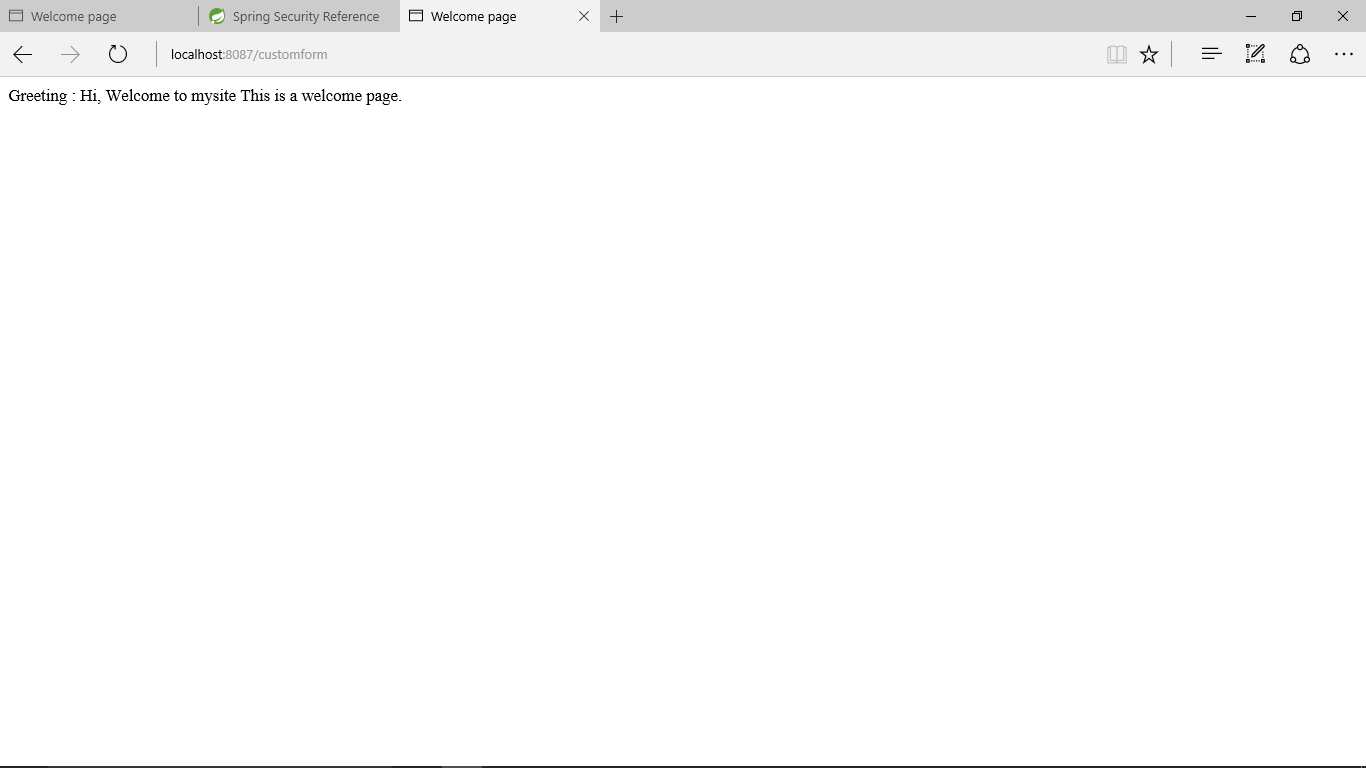
Submit, you will see AccessDenied Page



1. Logout. <http://localhost:8087/customform/login?logout> Login with ADMIN role credentials.[admin,wrongpassword]
2. 
3. Provide wrong password



Provide proper admin role credentials and login [admin,password]



Now try to access db page on **http://localhost:8087/customform/db**, you will get AccessDenied page.

