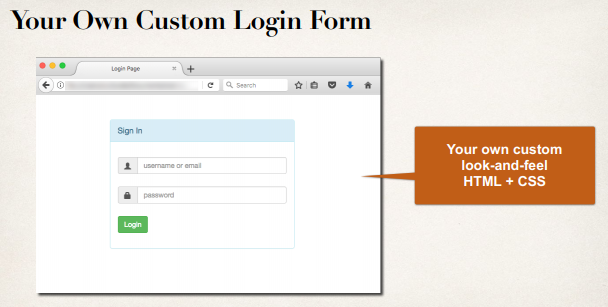
**48.3. Spring Security - Custom Login Form Overview**

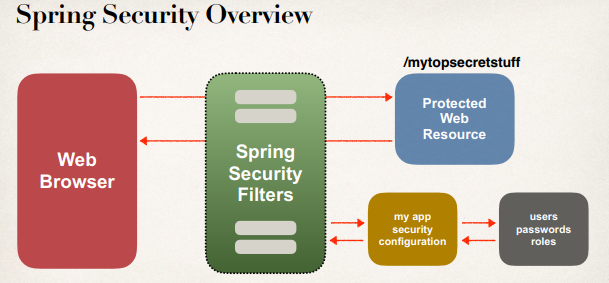
**Your Own Custom Login Form**:

Spring provide us default logging form but this is not ideal. Here we will create our own custom logging form. We can control the look and feel, use our own HTML, CSS, company logo and many more that we want.

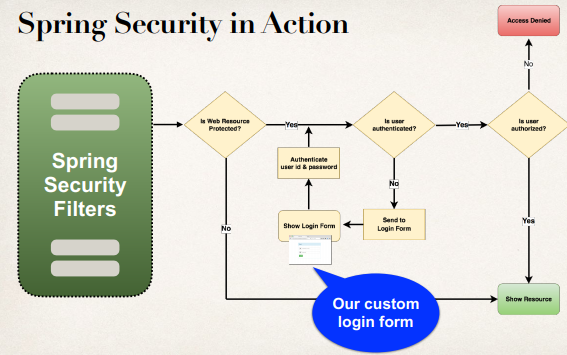


**Spring Security Overview**:

Here we have our web browser and we are trying to access some protected web resources. Here we have Spring security filters in place that will be actually kind of pre-process and post-process. They basically make use of their app configuration and read userid, password, and roles.



**Spring Security in Action**:



**Development Process (Step-by-Step)**:

1. Modify Spring Security Configuration to reference custom login form
2. Develop a Controller to show the custom login form
3. Create custom login form
   1. HTML (CSS optional)
   2. Spring MVC form tag **<form:form>**

**Step 1: Modify Spring Security Configuration**:

|  |  |
| --- | --- |
| Method | Description |
| configure(AuthenticationManagerBuilder | Configure users (in memory, database, ldap, etc) |
| configure(HttpSecurity) | Configure security of web paths in application, login, logout etc |

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.authorizeRequests()

.anyRequest().authenticated()

.and()

.formLogin()

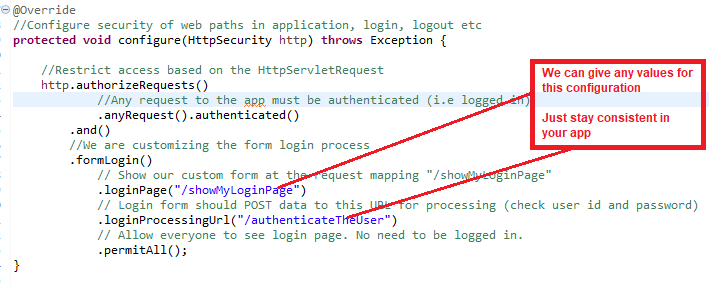
.loginPage("/showMyLoginPage")

.loginProcessingUrl("/authenticateTheUser")

.permitAll();

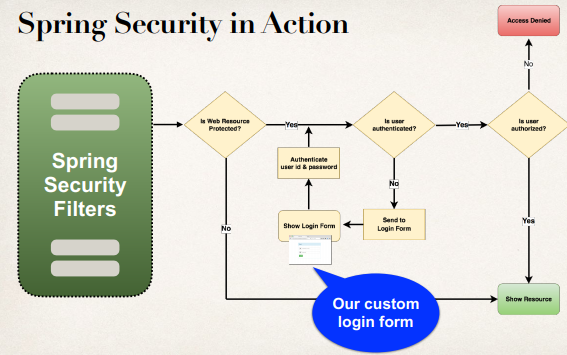
}

**Details about method**:



* We need to create a controller for the request mapping -> "/showMyLoginPage"
* No Controller Request Mapping for this. Spring Security provide this free -> "/authenticateTheUser"

**Configure Spring security to reference our custom login form**:



**Step 2: Develop a Controller to show the custom login form**:

@Controller

**public** **class** LoginController {

// map with 'loginPage("/showMyLoginPage")' in "DemoSecurityConfig.class"

@GetMapping("/showMyLoginPage")

**public** String showMyLoginPage() {

// "plain-login.jsp" page in "/WEB-INF/view/plain-login.jsp"

**return** "plain-login";

}

}

**Step-3: Create custom logon form**:

<%@ taglib prefix=*"form"* uri=*"http://www.springframework.org/tags/form"*%>

<%@ taglib prefix=*"c"* uri=*"http://java.sun.com/jsp/jstl/core"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Custom Login Page</title>

<style type=*"text/css"*>

*.failed*{

color: *red*;

}

</style>

</head>

<body>

<h3>My Custom Login Page</h3>

<form:form

action=*"*${pageContext.request.contextPath}*/authenticateTheUser"*

method=*"POST"*>

<!-- Check for login error -->

<c:if test=*"*${param.error != null }*"*>

<i class=*"failed"*>

Sorry!!! You entered invalid username/password

</i>

</c:if>

<p>

User Name: <input type=*"text"* name=*"username"* />

</p>

<p>

Password: <input type=*"password"* name=*"password"* />

</p>

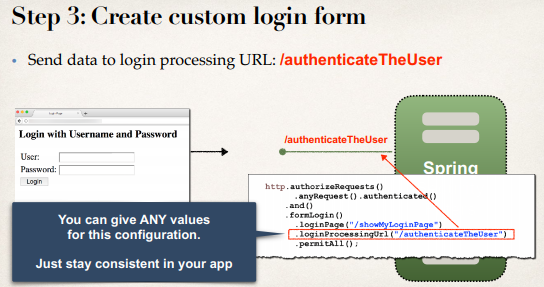
<input type=*"submit"* value=*"Login"* />

</form:form>

</body>

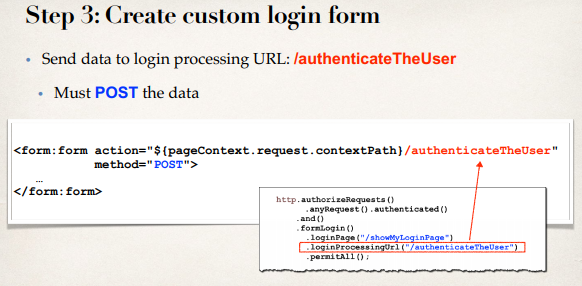
</html>

**Sending data to the logging process URL: */authenticateTheUser***:



* Send data to login processing URL: */authenticateTheUser*
* Login processing URL will be handled by Spring Security Filters
* We get it for free … no coding required
* Must ***POST*** the data.

**Send data to login process URL**:

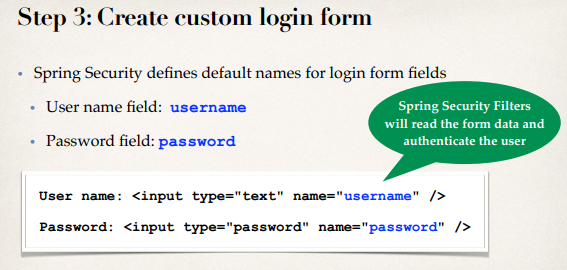


**Best Practice**:

* The best practice here, when we are creating the form, we have to make use of the Spring MVC form tag. <form:form>
* Provides automatic support for security defenses (more on this later)

**Spring Security and default form field**:

* Spring Security defines default names for login form fields
  + User name field: **username**
  + Password field: **password**



**Validate username and password by Spring Security in the Background**:



**Context Path and why we use it**:

* The "**Context Path**" is the root path of our web application.
  + **Properties -> Web Project Settings -> Context root**
  + <http://localhost:8080/spring-security-demo/showMyLoginPage> here context path is

" spring-security-demo "

* Allows us to dynamically reference context path of application
* Helps to keep links relative to application context path
* If you change context path of app, then links will still work
* Much better than hard-coding context path …

48.3. Spring Security - Custom Login Form Overview