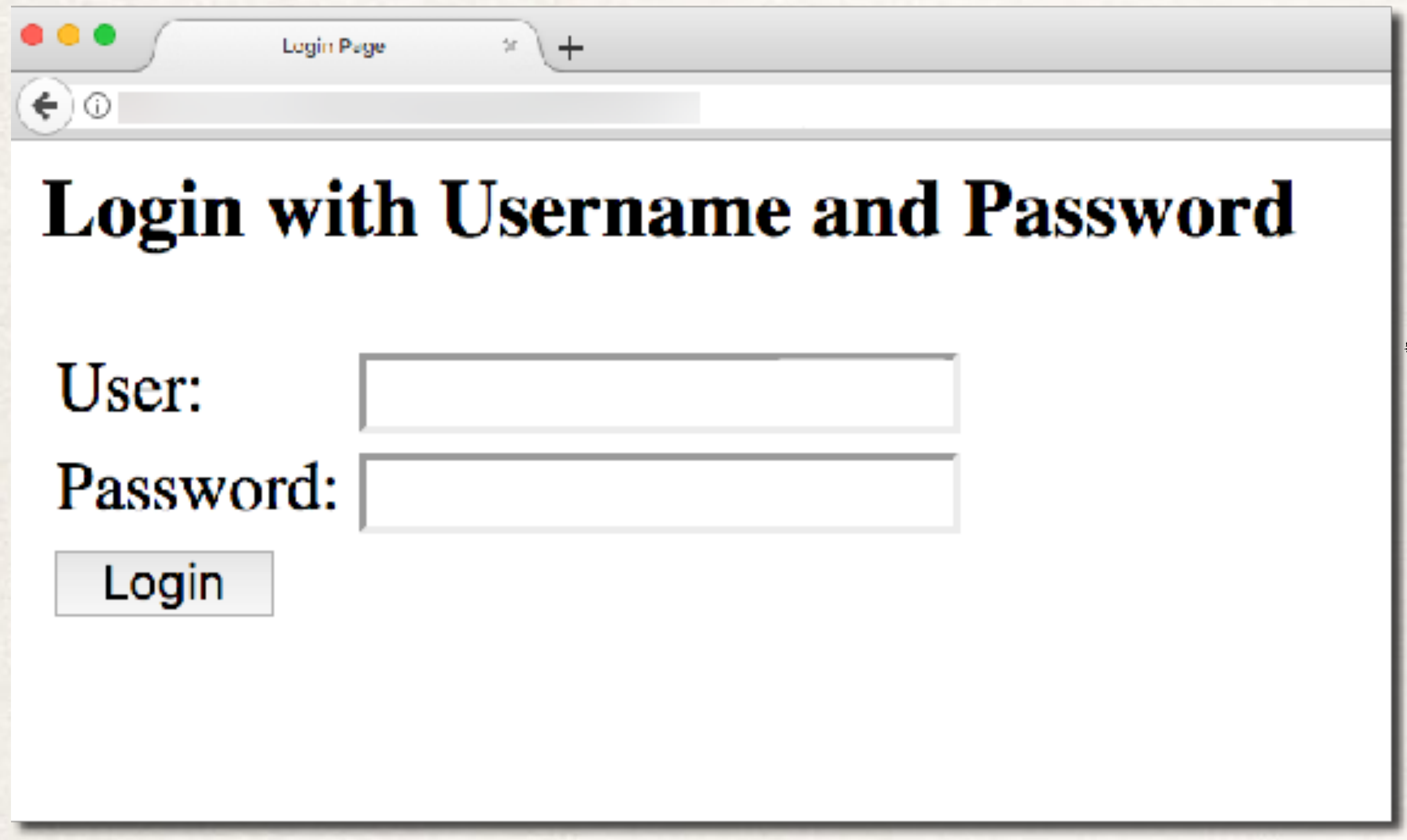
Step 3: Create custom login form

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

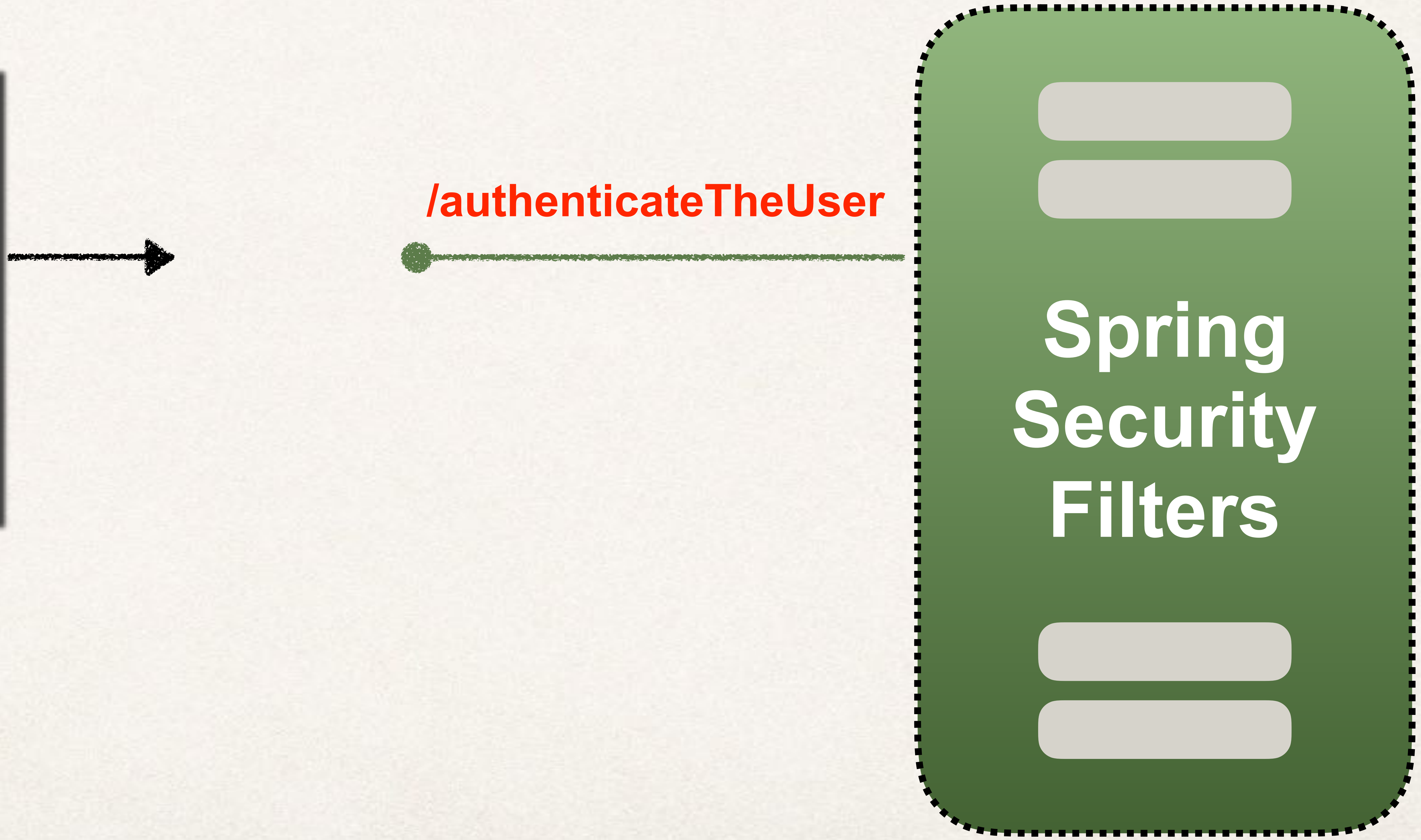


Login with Username and Password

User:

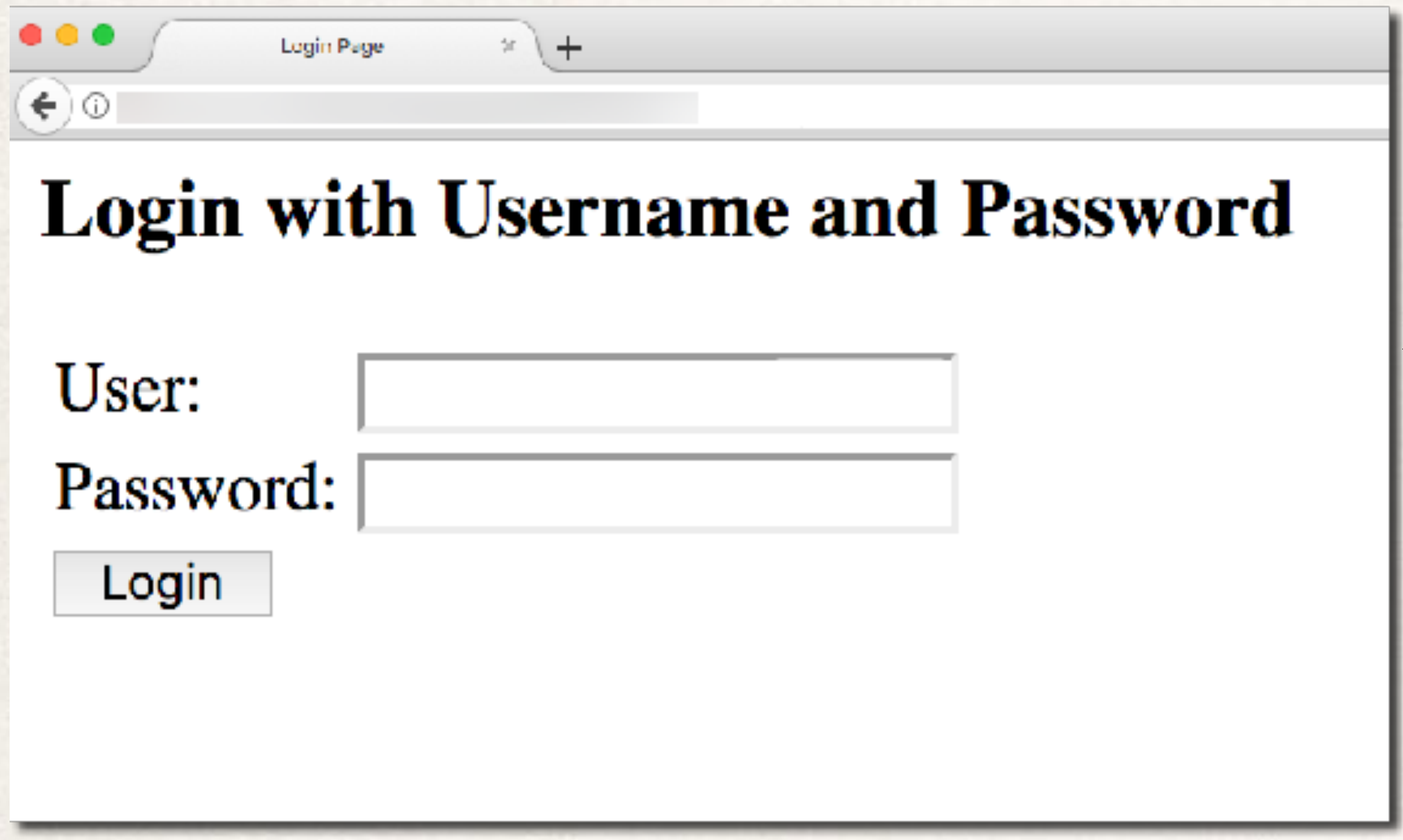
Password:

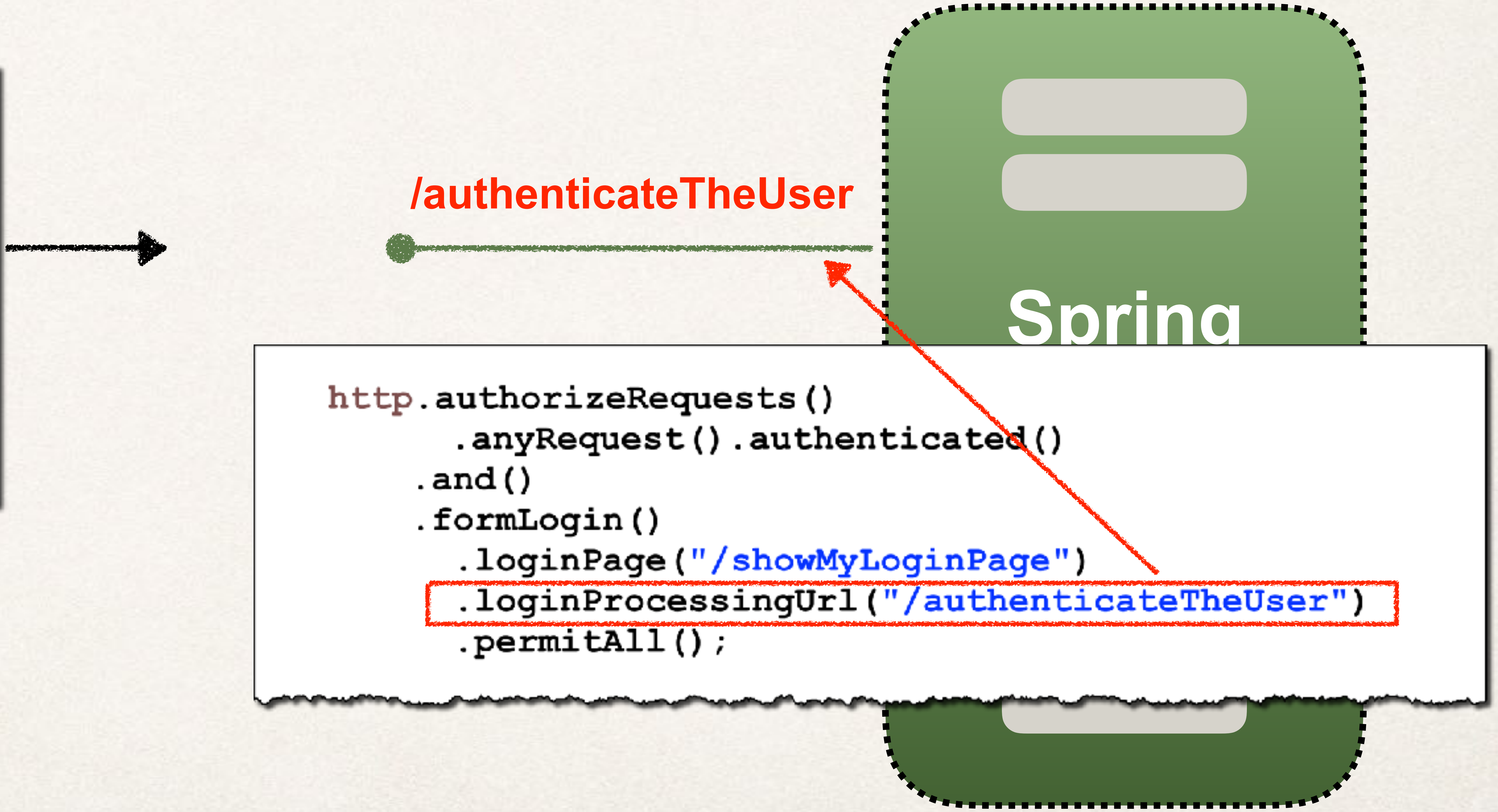
Login



Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

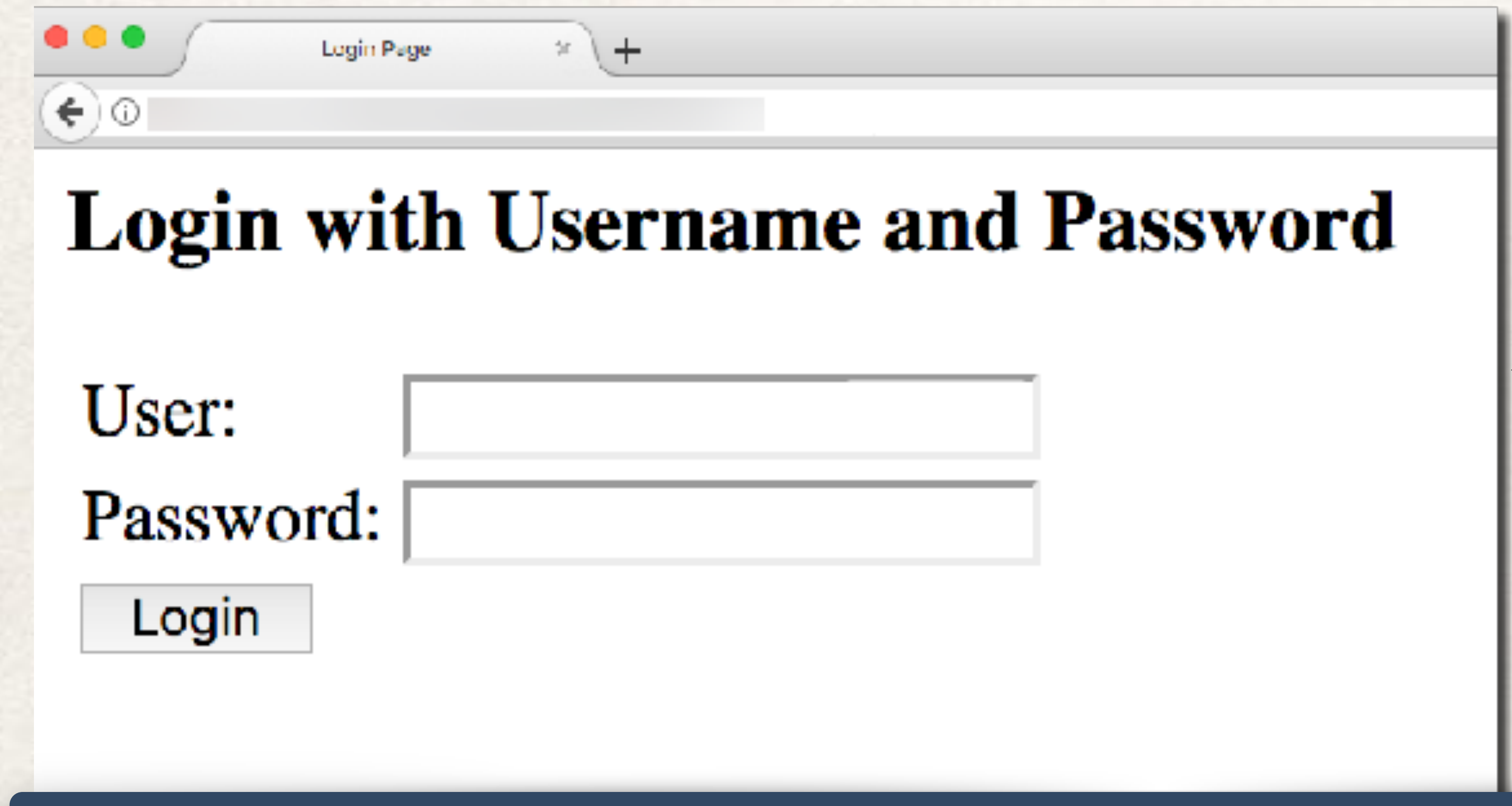




```
http.authorizeRequests()
        .anyRequest().authenticated()
    .and()
    .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();
```

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**



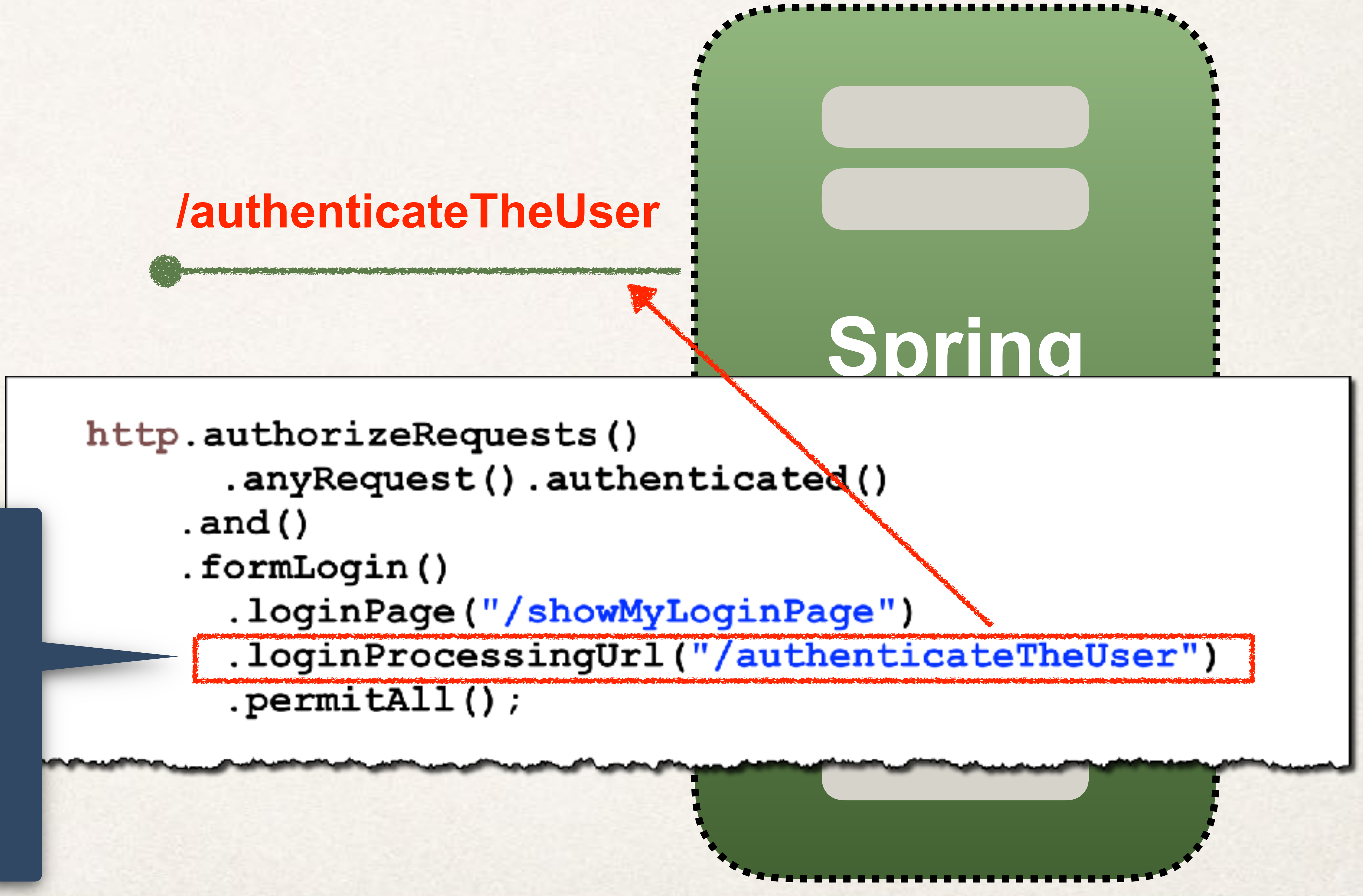
Login Page

Login with Username and Password

User:

Password:

Login



```
http.authorizeRequests()
        .anyRequest().authenticated()
    .and()
    .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();
```

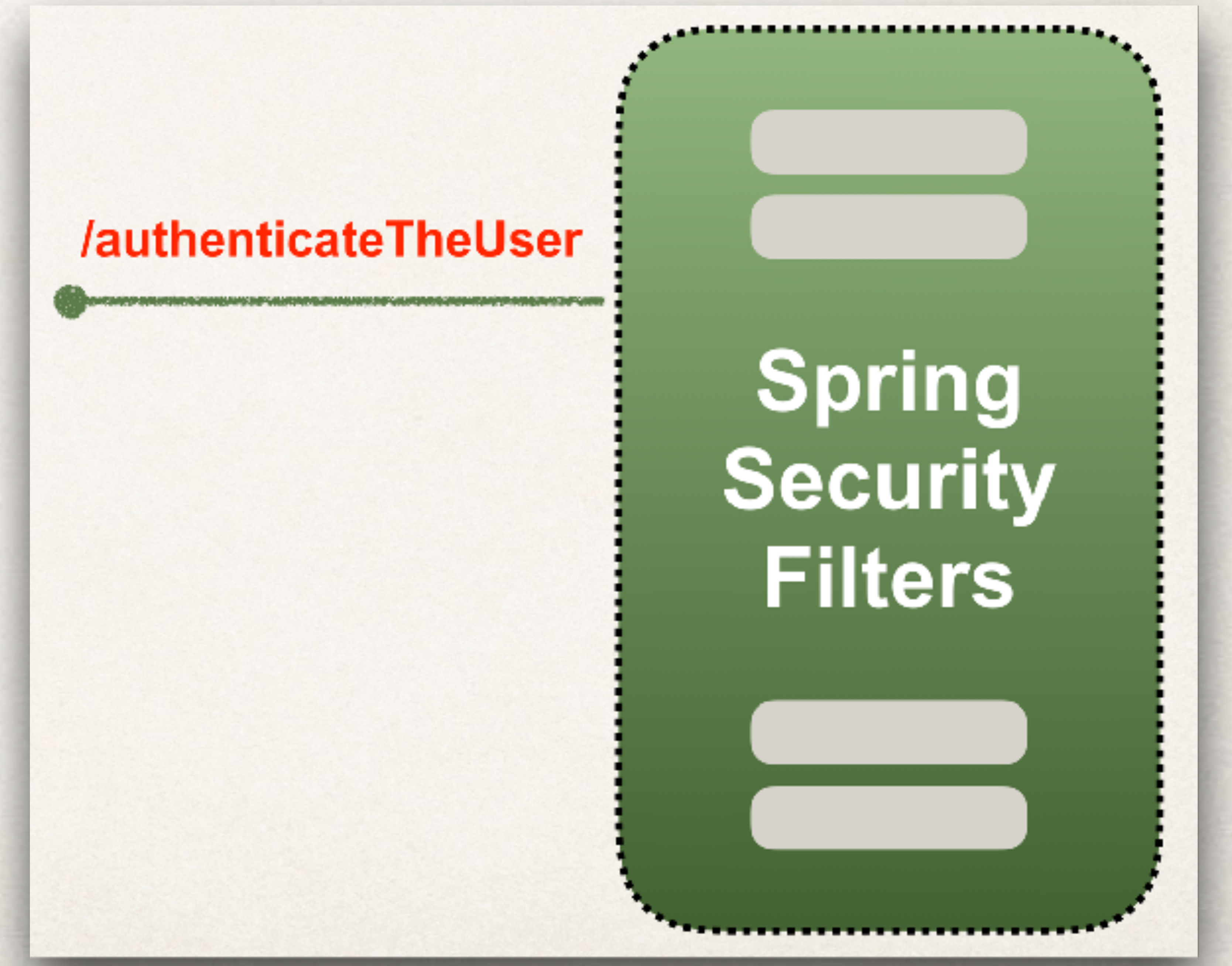
You can give ANY values for this configuration.

Just stay consistent in your app

Step 3: Create custom login form

Step 3: Create custom login form

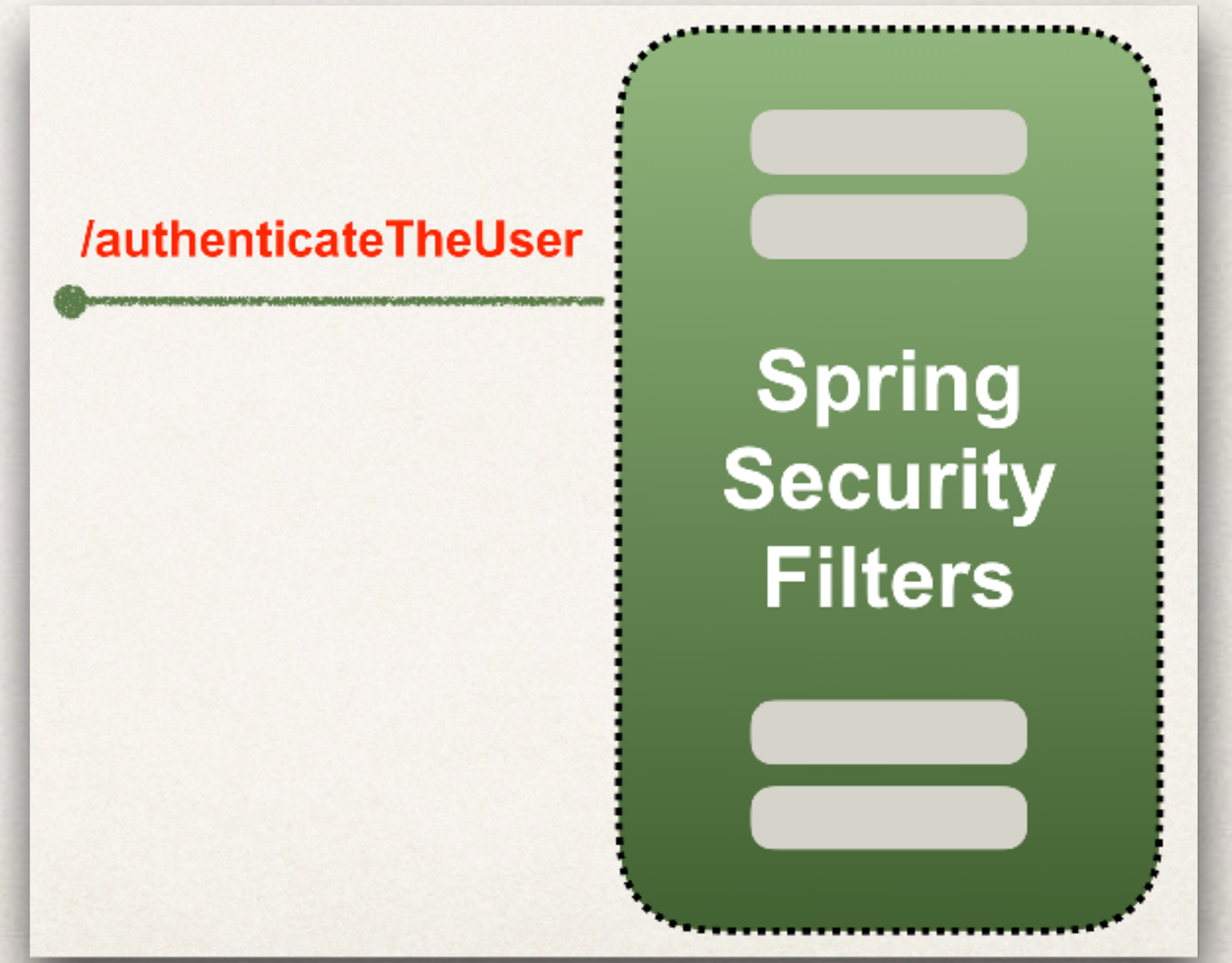
- Send data to login processing URL: **/authenticateTheUser**



Step 3: Create custom login form

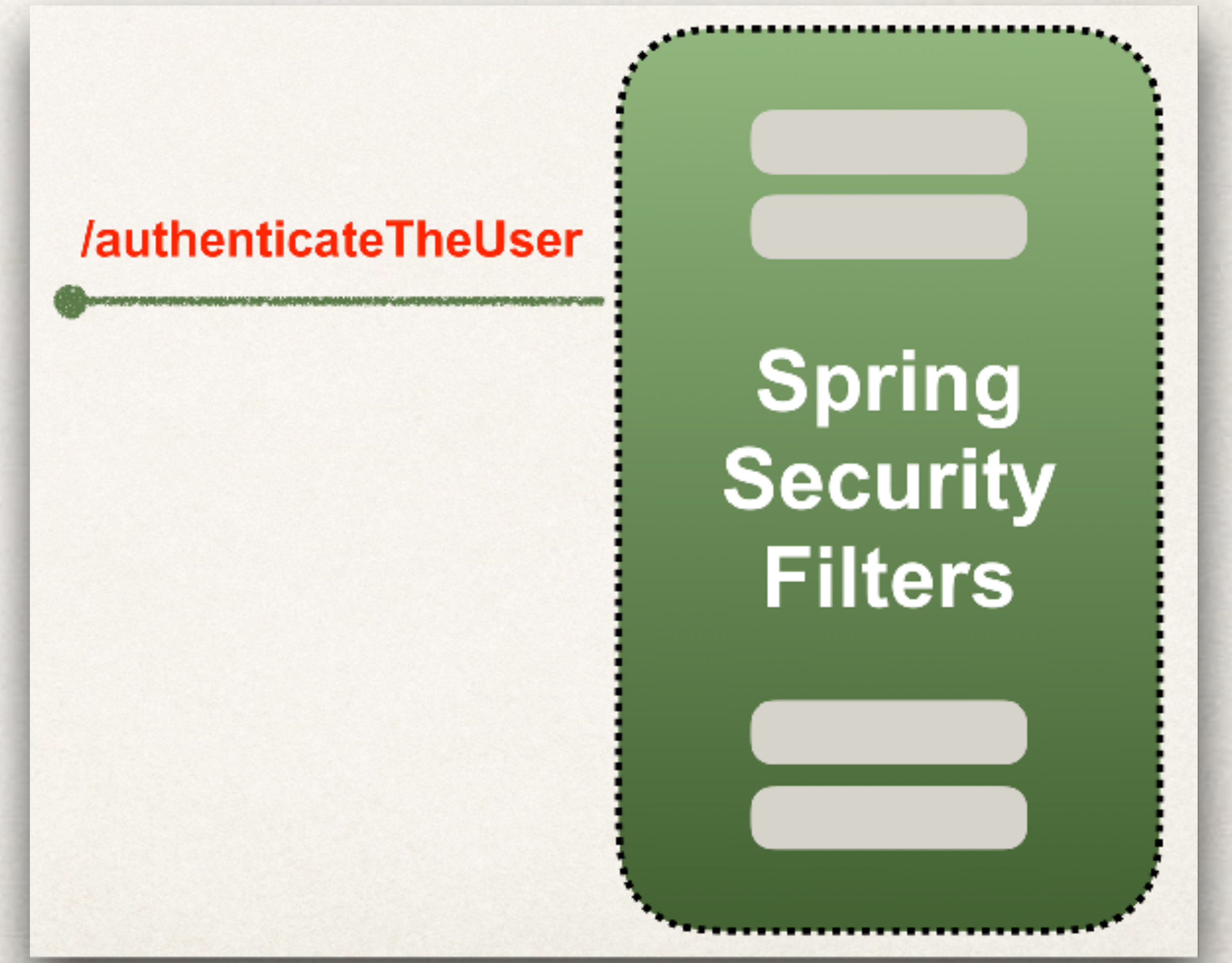
- Send data to login processing URL: **/authenticateTheUser**

- Login processing URL will be handled by Spring Security Filters



Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**
- Login processing URL will be handled by Spring Security Filters
- You get it for free … no coding required



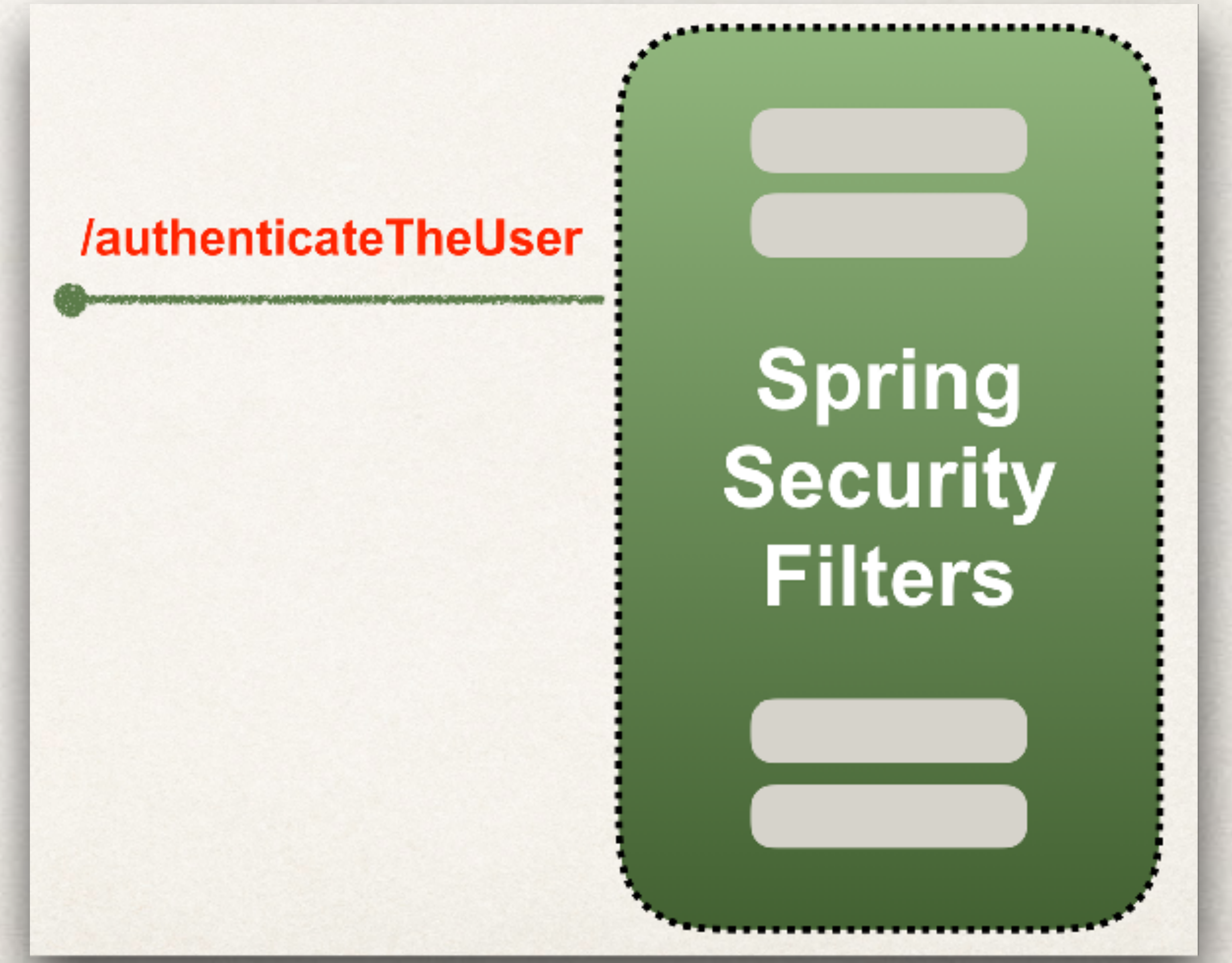
Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

- Login processing URL will be handled by Spring Security Filters

- You get it for free … no coding required

This is
Spring Security magic …
LOL



Step 3: Create custom login form

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
   …
</form:form>
```

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**
  - Must **POST** the data

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
    …
</form:form>
```

Step 3: Create custom login form

- Send data to login processing URL: **/authenticateTheUser**
  - Must **POST** the data

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
   …
</form:form>
```

```
http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();
```

Step 3: Create custom login form

Best Practice

Step 3: Create custom login form

Best Practice

- **Best practice** is to use the Spring MVC Form tag `<form:form>`

Step 3: Create custom login form

Best Practice

- **Best practice** is to use the Spring MVC Form tag `<form:form>`
  - Provides automatic support for security defenses (*more on this later)*

Step 3: Create custom login form

Best Practice

- **Best practice** is to use the Spring MVC Form tag `<form:form>`
  - Provides automatic support for security defenses (*more on this later)*

```
<form:form action="…" method="…" >
  …
</form:form>
```

Step 3: Create custom login form

Step 3: Create custom login form

- Spring Security defines default names for login form fields

Step 3: Create custom login form

- Spring Security defines default names for login form fields
  - User name field: `username`

Step 3: Create custom login form

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

Step 3: Create custom login form

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

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

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

Step 3: Create custom login form

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

Spring Security Filters will read the form data and authenticate the user

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

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

Step 3: Create custom login form

Pull It All Together

File: WEB-INF/view/plain-login.jsp

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

Step 3: Create custom login form

Pull It All Together

File: WEB-INF/view/plain-login.jsp

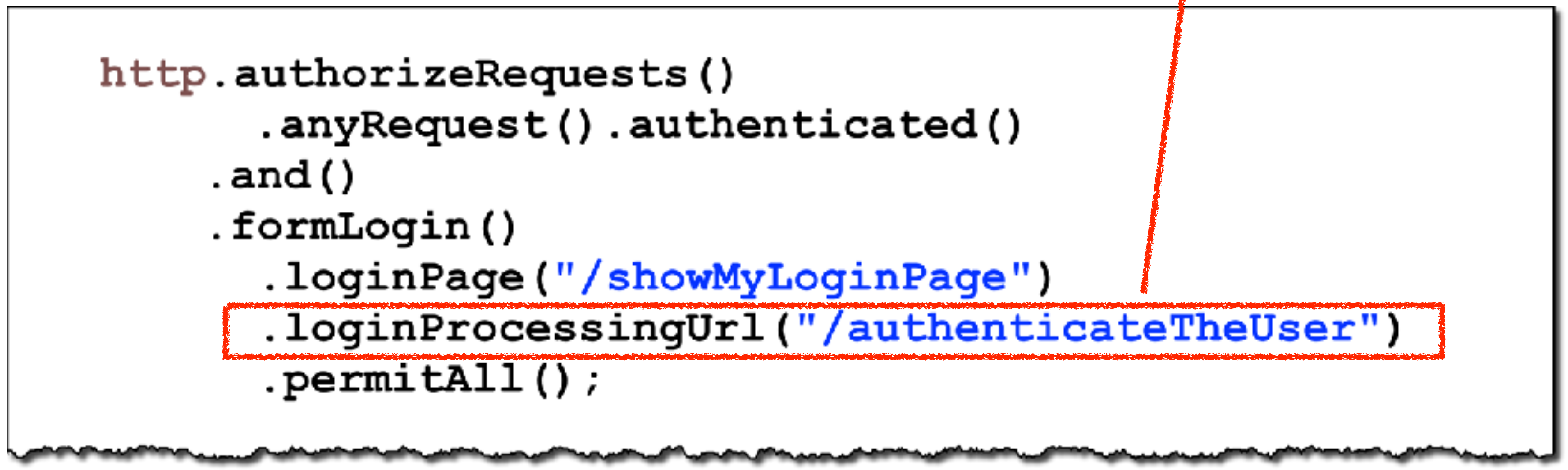
```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
…
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">









</form:form>
…
```

Step 3: Create custom login form

Pull It All Together

File: WEB-INF/view/plain-login.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
…
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
```



```
http.authorizeRequests()
        .anyRequest().authenticated()
      .and()
      .formLogin()
        .loginPage("/showMyLoginPage")
        .loginProcessingUrl("/authenticateTheUser")
        .permitAll();
```

```
</form:form>
…
```

Step 3: Create custom login form

Pull It All Together

File: WEB-INF/view/plain-login.jsp

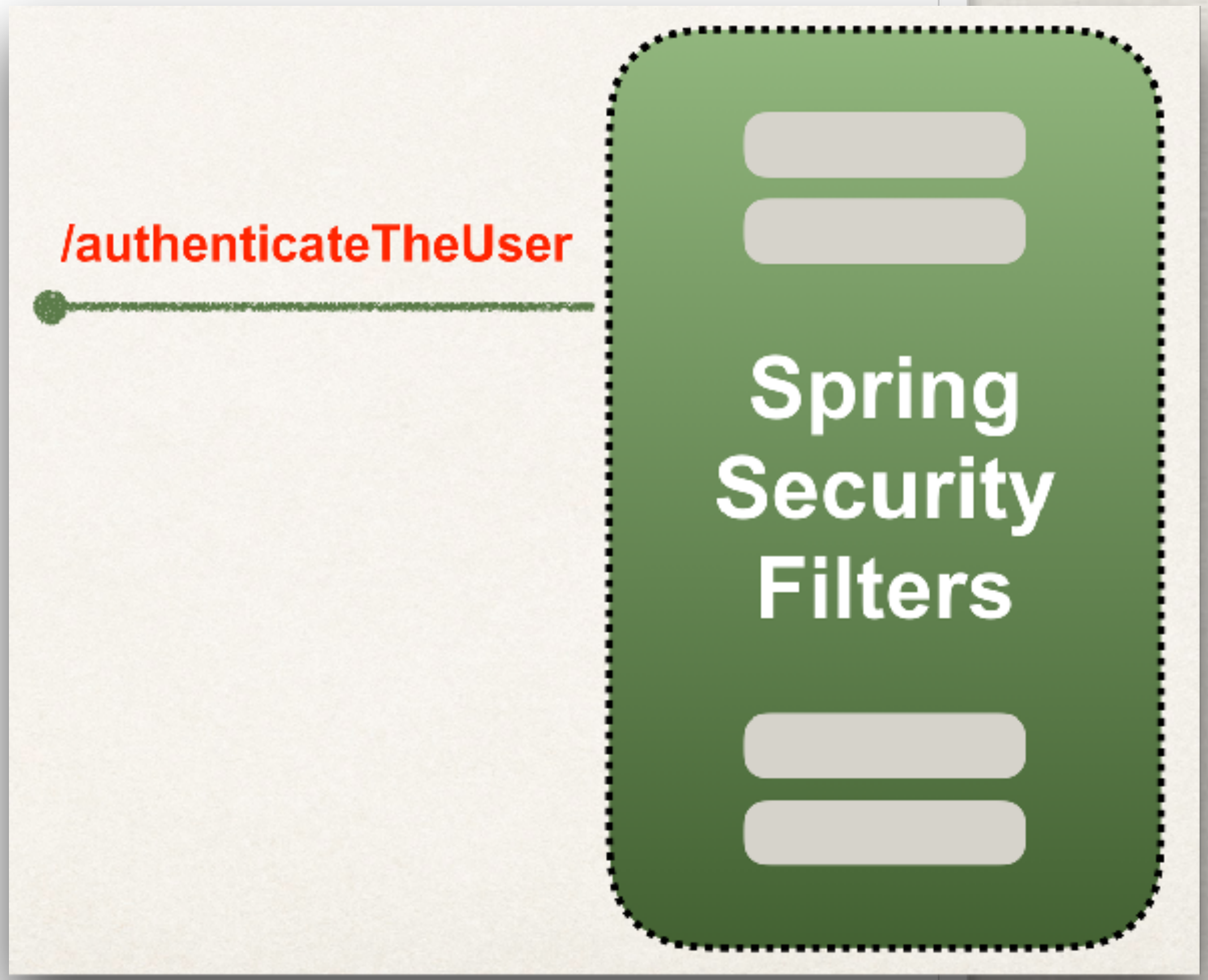
```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
…
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">

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

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

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

</form:form>
…
```

Step 3: Create custom login form

Pull It All Together

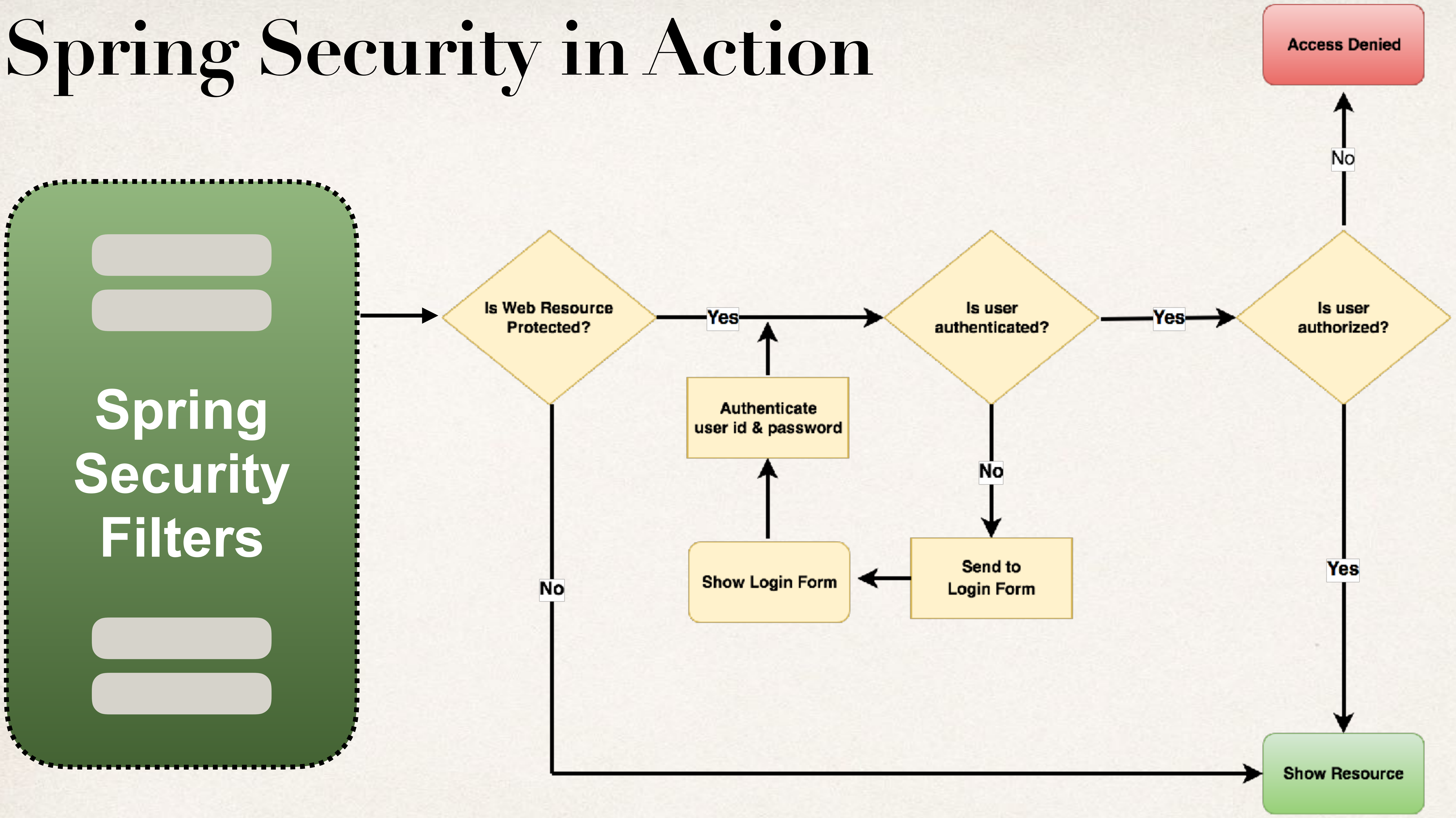
File: WEB-INF/view/plain-login.jsp

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
…
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">

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

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

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

</form:form>
…
```



Spring Security in Action

Spring Security in Action



More Info on Context Path

More Info on Context Path

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
    …
</form:form>
```

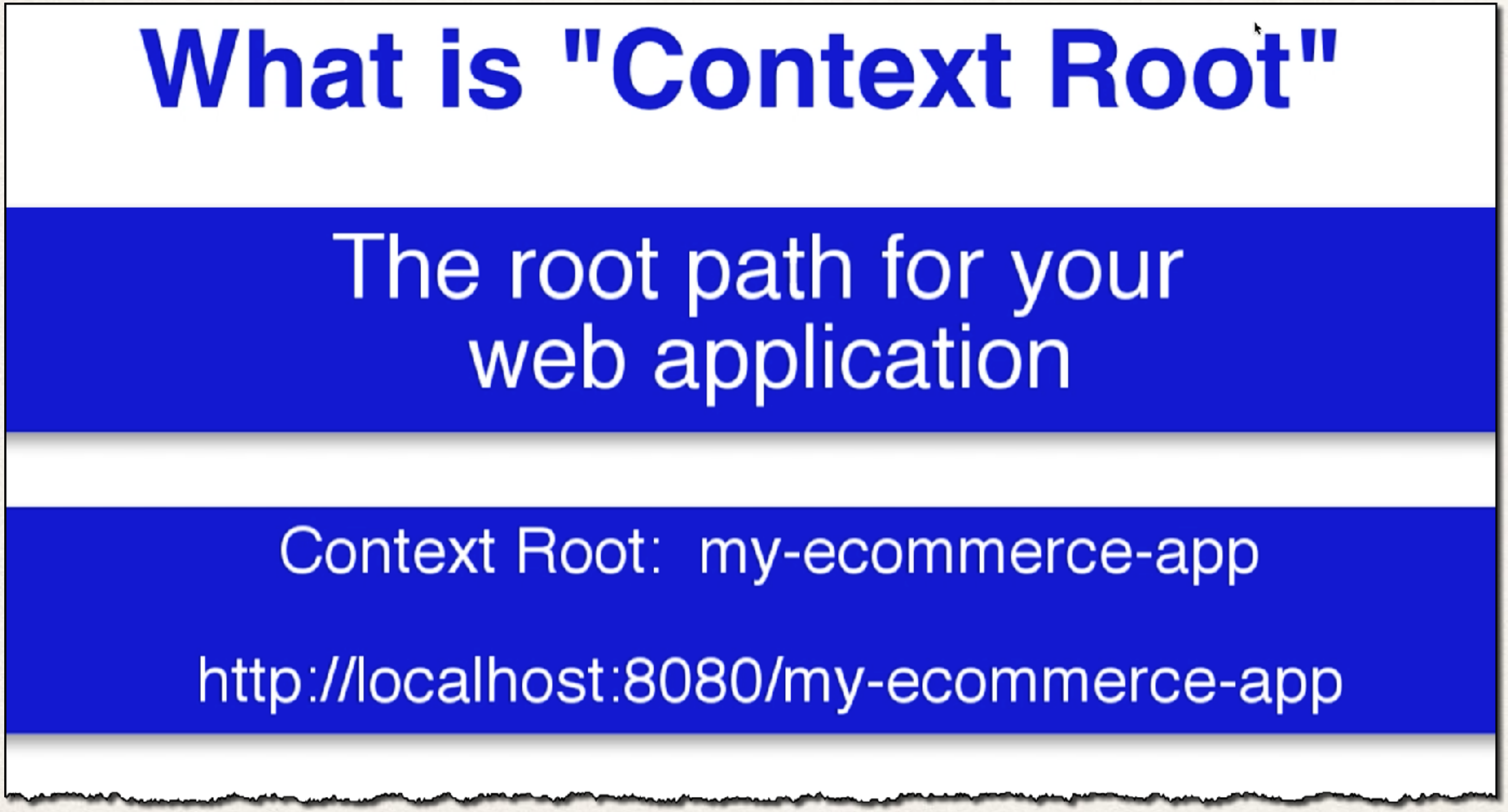
More Info on Context Path

What is this???

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
    …
</form:form>
```

More Info on Context Path

More Info on Context Path

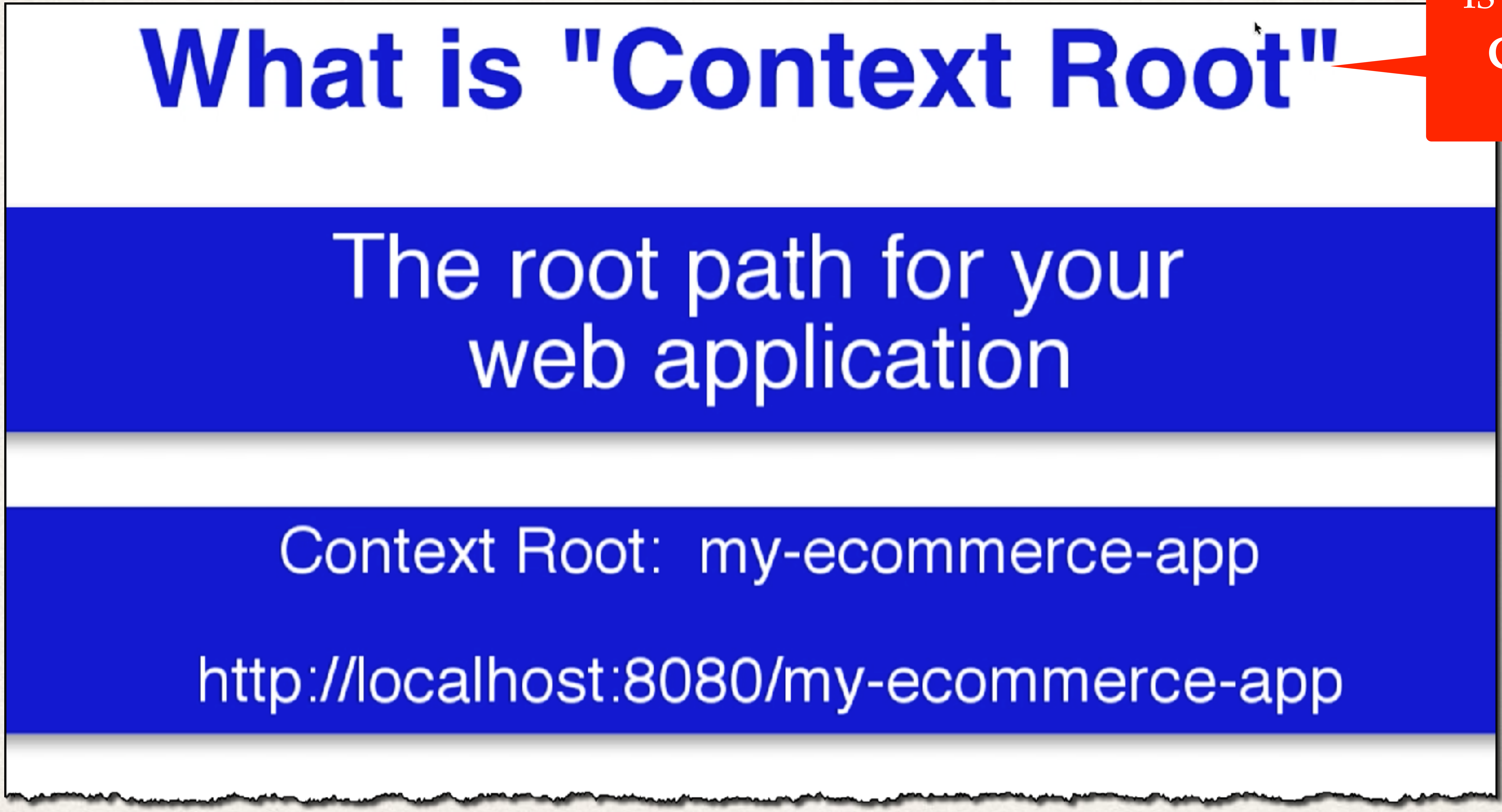

What is "Context Root"

The root path for your web application

Context Root: my-ecommerce-app

http://localhost:8080/my-ecommerce-app

More Info on Context Path


What is "Context Root"

The root path for your web application

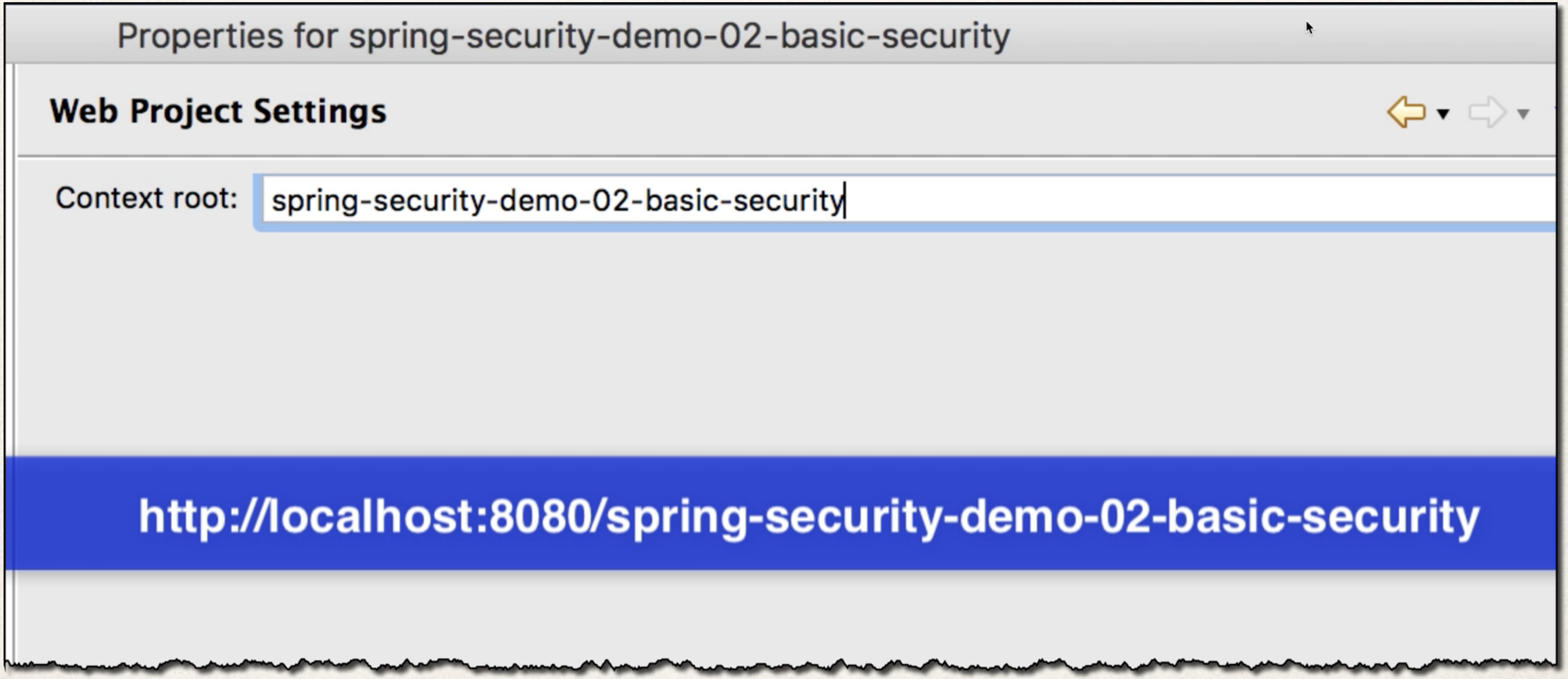
Context Root: my-ecommerce-app

http://localhost:8080/my-ecommerce-app

Context Path is same thing as **Context Root**

More Info on Context Path

More Info on Context Path



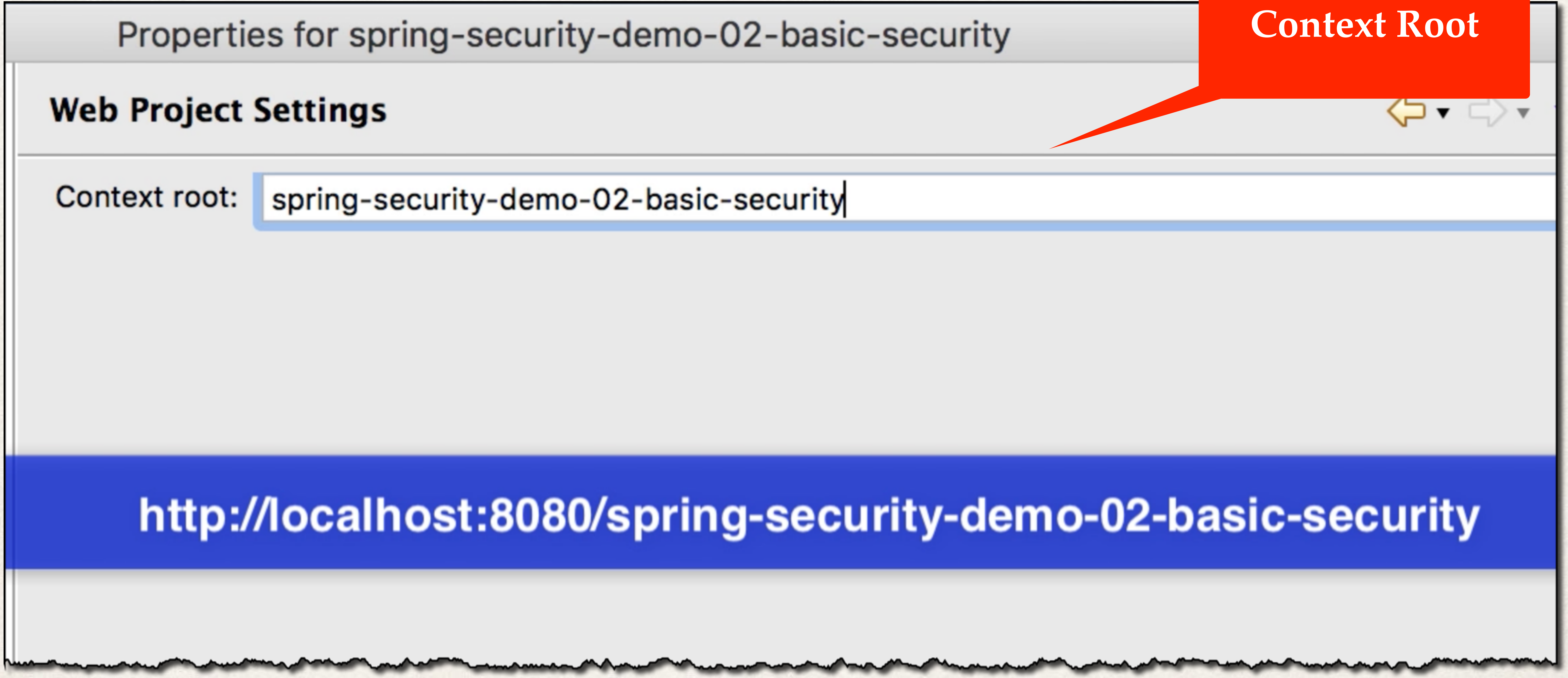
Properties for spring-security-demo-02-basic-security

Web Project Settings

Context root: spring-security-demo-02-basic-security

http://localhost:8080/spring-security-demo-02-basic-security

More Info on Context Path



Properties for spring-security-demo-02-basic-security

Web Project Settings

Context root: spring-security-demo-02-basic-security

http://localhost:8080/spring-security-demo-02-basic-security

Context Path is same thing as **Context Root**

More Info on Context Path

More Info on Context Path

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
    …
</form:form>
```

More Info on Context Path

Gives us access to context path dynamically

```
<form:form action="${pageContext.request.contextPath}/authenticateTheUser"
           method="POST">
    …
</form:form>
```

Why use Context Path?

Best Practice

Why use Context Path?

Best Practice

- Allows us to dynamically reference context path of application

Why use Context Path?

Best Practice

- Allows us to dynamically reference context path of application

- Helps to keep links relative to application context path

Why use Context Path?

Best Practice

- 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

Why use Context Path?

Best Practice

- 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 …