STRUTS FRAMEWORK COURSE

VALIDATIONS WITH STRUTS 2 FRAMEWORK

By the expert: Ubaldo Acosta

Global Mentoring

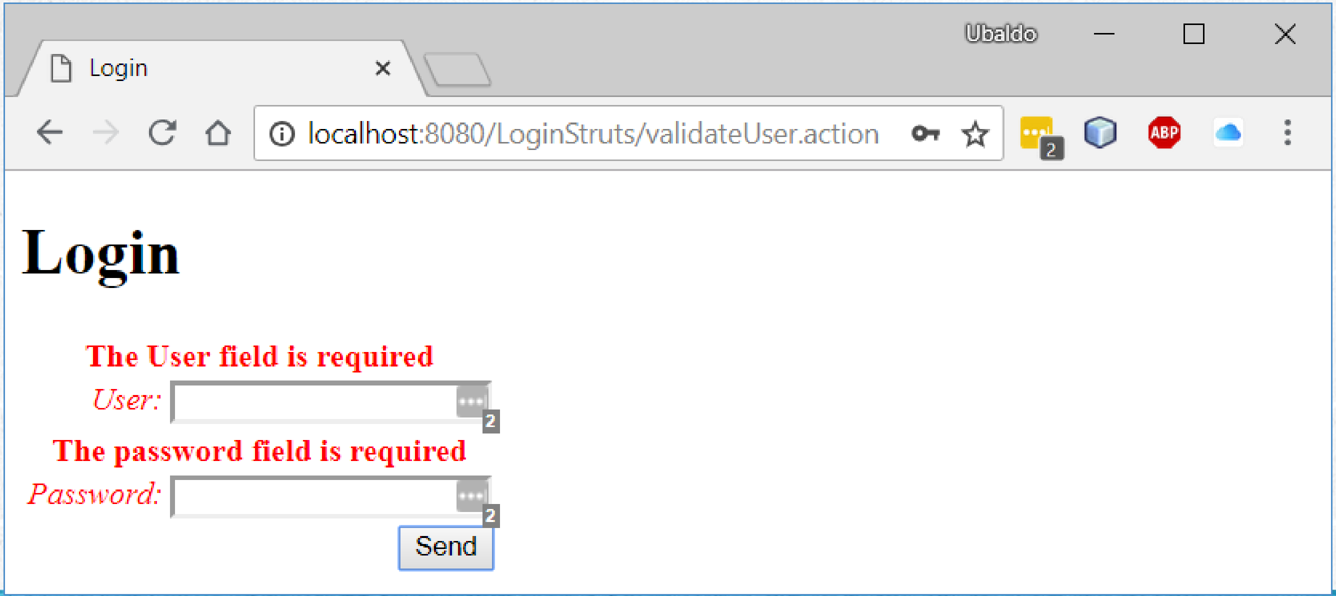
JAVA ONLINE COURSES UNIVERSITY

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx

EXERCISE OBJECTIVE

Create an application to put into practice the concept of Validations with Struts2. At the end we should observe the following:



Login

The User field is required

User:

The password field is required

Password:

Send

EXERCISE REQUIREMENT

We are going to start from the previous exercise called LoginStruts where the Annotations theme was applied.

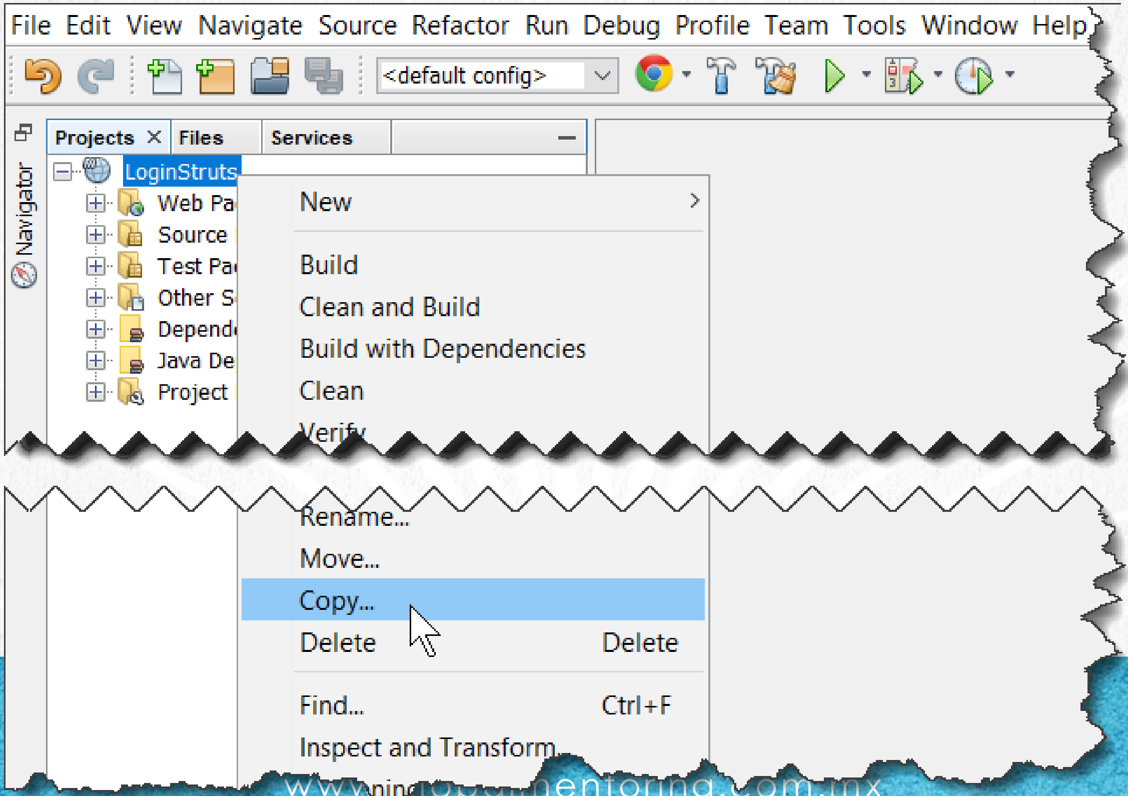
We are going to copy this project to work on this new lesson, in which we will add the concept of Validation and Error Handling in Struts2.

The login form must validate that both the user and password fields are not null, and also the password must contain at least 3 characters, not less.

And we will also add error handling. In case the value of the user field is different from "admin", we will return an error message and return to the login.jsp page, otherwise we will send a message that the user is correct and we will redirect to the page of welcome.jsp

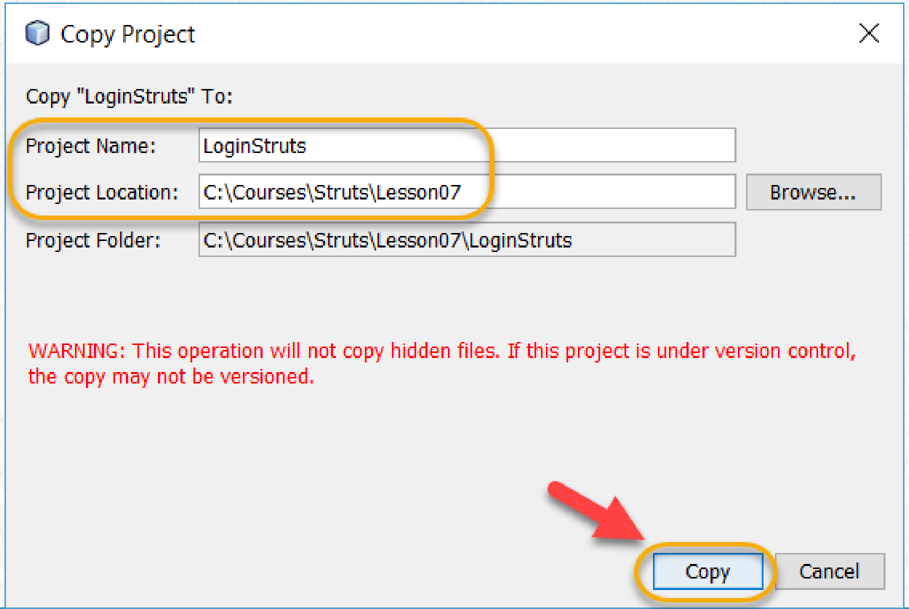
1. COPY THE PROJECT

- We created the new project using the copy function on the previous project:



1. COPY THE PROJECT

- We write the following values as shown and click on copy. This will not change the name of the Maven project, so we'll leave it:


Copy Project

Copy "LoginStruts" To:

Project Name: LoginStruts

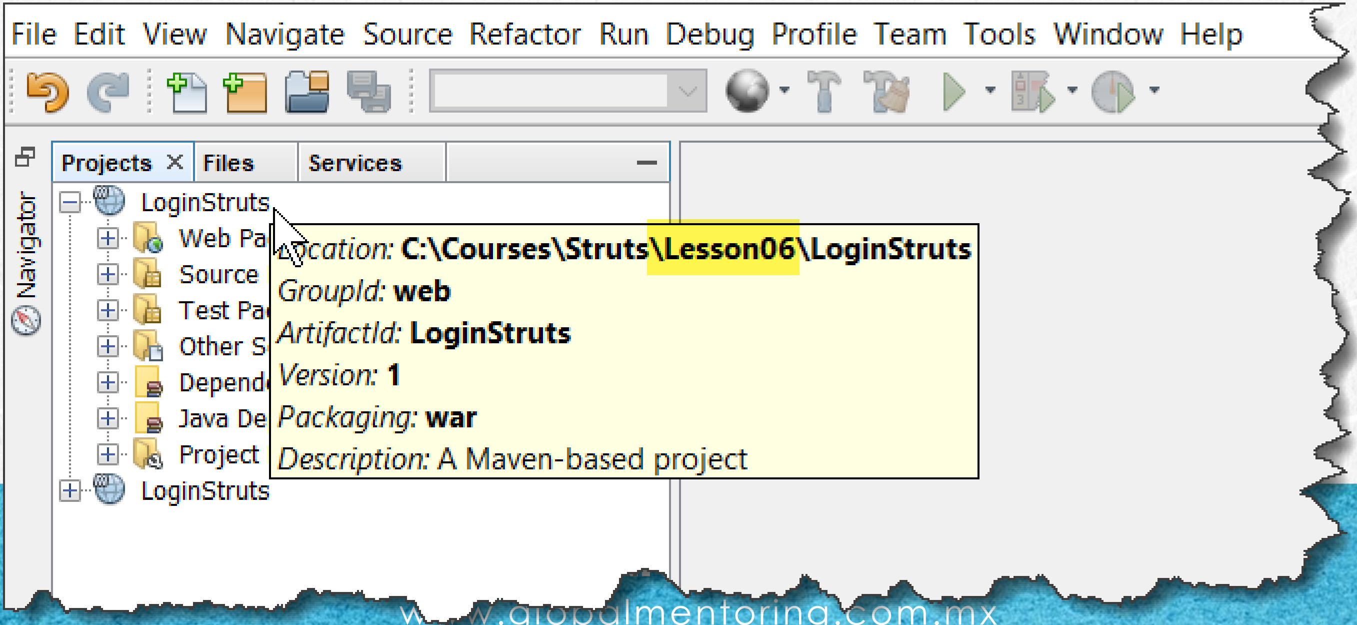
Project Location: C:\Courses\Struts\Lesson07

Project Folder: C:\Courses\Struts\Lesson07\LoginStruts

WARNING: This operation will not copy hidden files. If this project is under version control, the copy may not be versioned.

2. Close the project that we no longer use

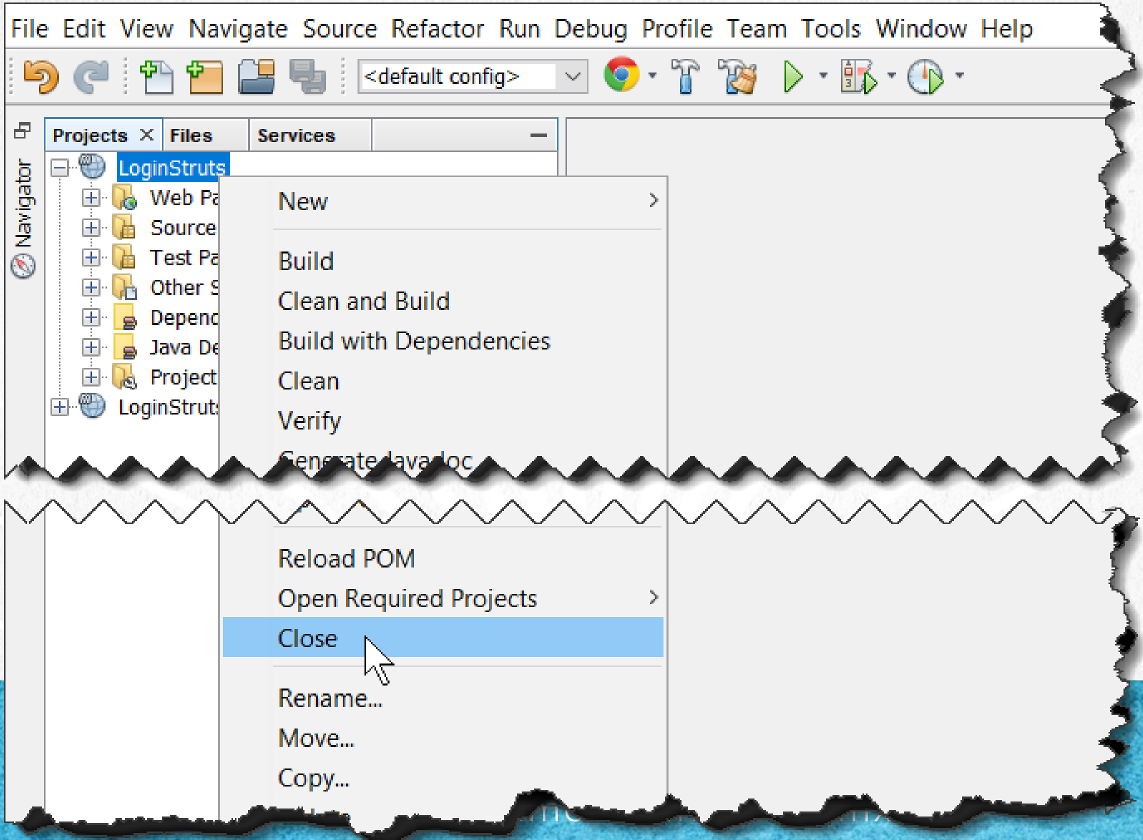
- We close the project that we no longer use, in this case to know which is the project that we must close (Lesson06), we can position ourselves on the name of the project and it will give us more information, including the route where the file is located. So we will know which project to close and which project to leave open:



Location: **C:\Courses\Struts\Lesson06\LoginStruts**
GroupId: **web**
ArtifactId: **LoginStruts**
Version: **1**
Packaging: **war**
Description: A Maven-based project

2. CLOSE THE PROJECT THAT WE NO LONGER USE

- We close the project that we no longer use.



3. MODIFY THE ACTION CLASS

We modify the ValidateUsuarioAction.java class so that we can add the concept of Validation and error handling in Struts 2.

To handle the concept of validations, what we are going to do is extend the ActionSupport class. As a next step we are going to overwrite the validate () method. And this is where we can take advantage of to make the necessary validations in the attributes of our class, or any other type of validation of our login form in this case.

- To add an error message associated with a field of the login form, we use the method:
`addFieldError(String fieldname, String errorMessage)`

- For handling errors we can use the methods we can use in case of a general error:
`addActionError(String generalErrorMessage)`

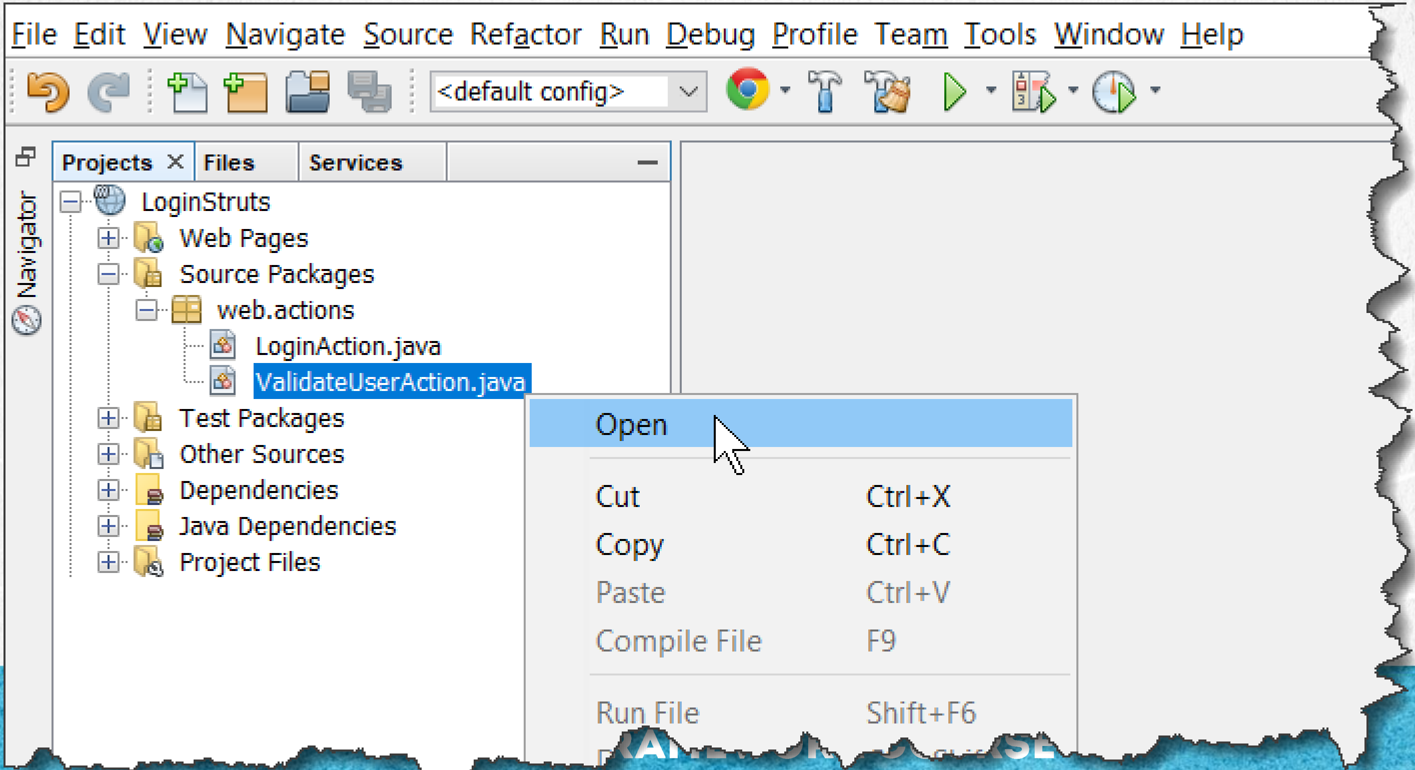
- For the handling of messages when it is NOT an error, but only an informative message we use the method:
`addActionMessage(String message)`

Finally, the @Result annotation with input value was modified to redirect directly to the login.jsp page and not to the action, otherwise the LoginAction.java class is reprocessed, and we do not want that , but once the validate and execute method of the class ValidarusuarioAction.java has been executed, we must go directly to login.jsp in case of error, otherwise we lose the messages that we add to the stack of Struts variables.

Let's see how our class is.

3. Modify the action class

- We open the ValidateUsuarioAction.java class to modify it:



3. MODIFY THE CODE

ValidateUserAction.java:

Click to download

```
package web.actions;

import static com.opensymphony.xwork2.Action.INPUT;
import static com.opensymphony.xwork2.Action.SUCCESS;
import com.opensymphony.xwork2.ActionSupport;
import org.apache.logging.log4j.*;
import org.apache.struts2.convention.annotation.*;

@Results({
    @Result(name = "success", location = "/WEB-INF/content/welcome.jsp"),
    @Result(name = "input", location = "/WEB-INF/content/login.jsp")
})
public class ValidateUserAction extends ActionSupport {

    private String user;
    private String password;
    Logger log = LogManager.getLogger(ValidateUserAction.class);
```

3. MODIFY THE CODE

ValidateUserAction.java:

Click to download

```
@Action("validateUser")
@Override
public String execute() {
    //If it is valid user we show the welcome.jsp page
    if (validUser()) {
        addActionMessage(getText("user.valid"));
        return SUCCESS;
    } else {
        //If it is user NOT valid, we return to the login
        return INPUT;
    }
}
```

3. MODIFY THE CODE

ValidateUserAction.java:

Click to download

```
@Override
public void validate() {
    if (this.user == null || "".equals(this.user.trim())) {
        addFieldError("user", getText("val.user"));
    } else if (!validUser()) {
        addActionError(getText("user.invalid"));
    }

    if (this.password == null || "".equals(this.password.trim())) {
        addFieldError("password", getText("val.password"));
    } else if (this.password.length() < 3) {
        addFieldError("password", getText("val.pass.min.length"));
    }
}
```

3. MODIFY THE CODE

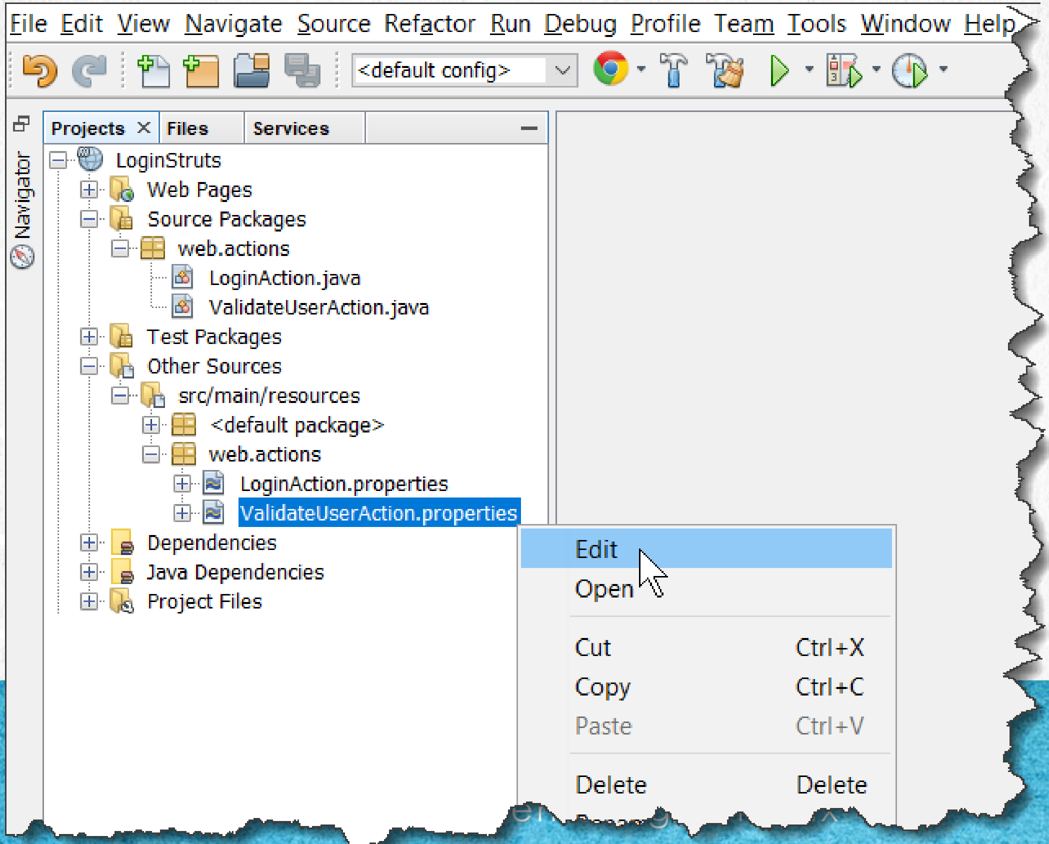
ValidateUserAction.java:

Click to download

```
    private boolean validUser() {
        //Value of valid user = "admin" in hard code
        return "admin".equals(this.user);
    }

    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

4. Modify the properties file

- We open the file ValidateUserAction.properties to modify it :



4. MODIFY THE CODE

ValidateUserAction.properties:

Click to download

```
#Values for the welcome page
welcome.title: Welcome
welcome.message: User Value:
welcome.return: Return

#Validations
user.valid: The user is valid
user.invalid: The user is invalid

#To reload the login page in case of error (result = input)
form.user: User
form.password: Password
form.button: Send
form.title: Login
val.user: The User field is required
val.password: The password field is required
val.pass.min.length: The password must contain at least 3 characters
```

5. MODIFY THE JSP

Now we modify the JSP of login.jsp to adapt it to the new changes in the validation topic and the messages that we must show.

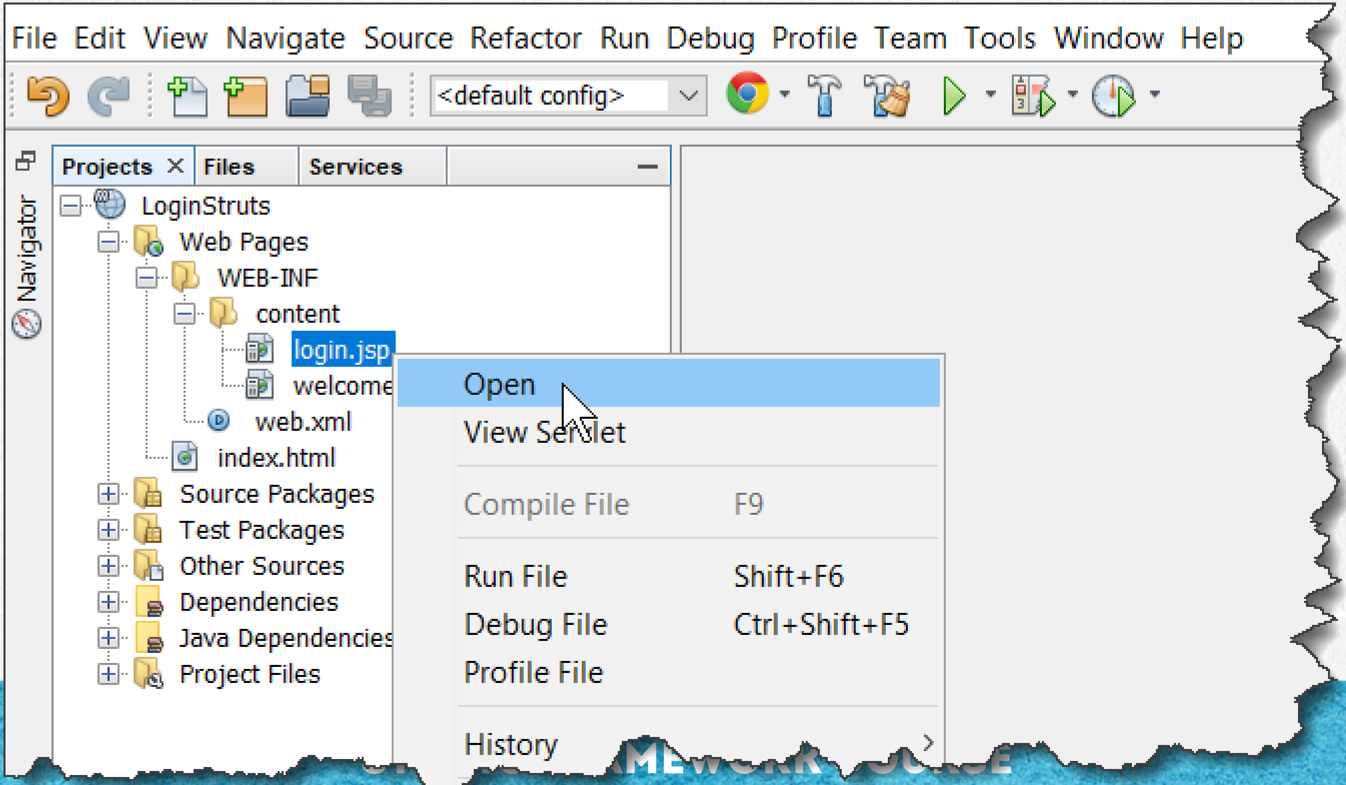
The changes are minimal, since most of the details are automatically handled by Struts, when associating from the moment an error is reported in the validate method of the ValidateUsuarioAction.java class, the name of the form field was indicated refers to that error in case it occurs.

However, more general errors, such as actionError, we must work with the tag:
<s:actionerror />

Finally, to add CSS styles automatically managed by Struts, we must add the tag <s: head /> before closing the tag </ head> of our HTML document.

5. MODIFY THE JSP

- We open the login.jsp file:



5. MODIFY THE CODE

login.jsp:

Click to download

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib  prefix="s" uri="/struts-tags" %>
<!DOCTYPE html>
<html>
    <head>
        <title><s:text name="form.title" /></title>
        <s:head />
    </head>
    <body>
        <h1><s:text name="form.title" /></h1>

        <s:actionerror/>

        <s:form action="validateUser">
            <s:textfield key="form.user" name="user" />
            <s:password key="form.password" name="password" />
            <s:submit key="form.button" name="submit" />
        </s:form>
    </body>
</html>
```

6. MODIFY THE JSP

Now we modify the JSP of welcome.jsp to adapt it to the new changes in the issue of validations and the messages that we must show.

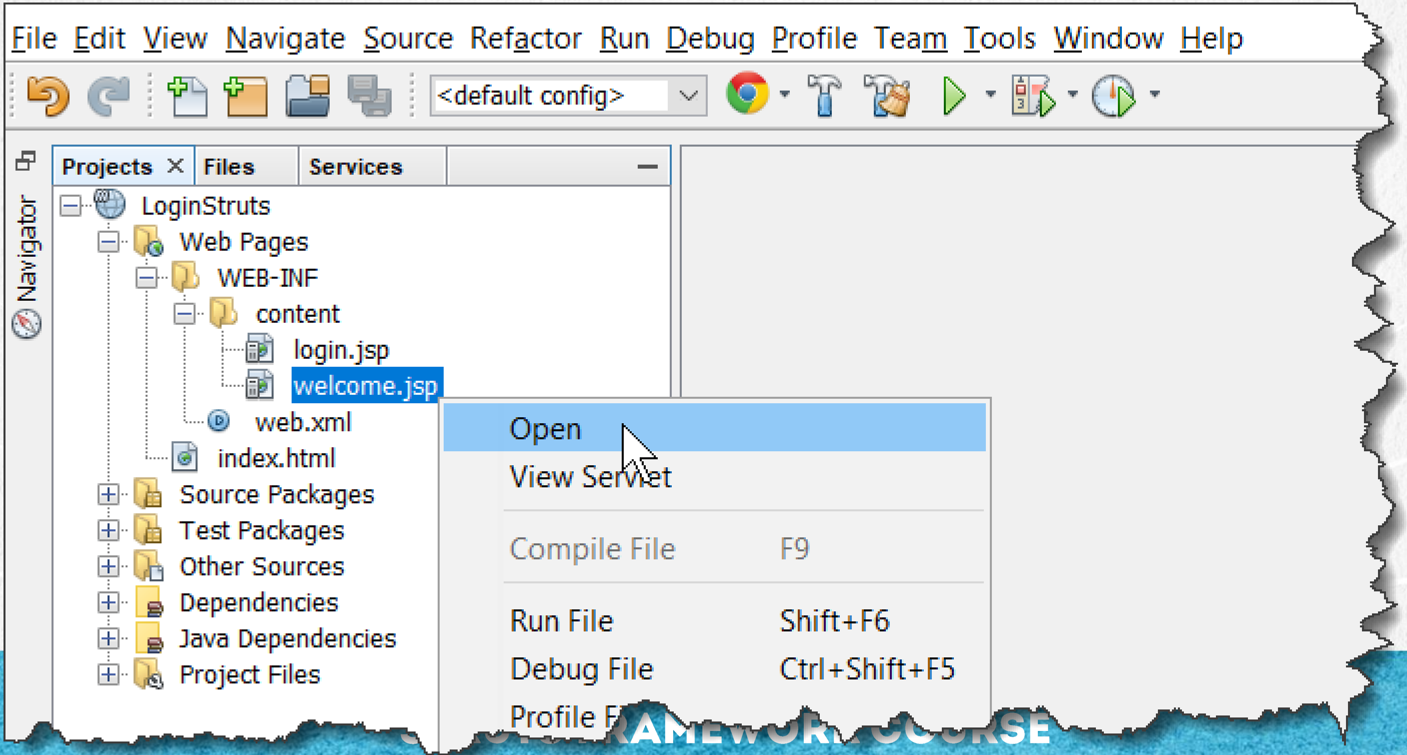
The changes are minimal. In this file what we are going to add is the CSS style so that Struts automatically adds the style to the messages that we add. For this we must add the tag <s:head /> before closing the tag </ head> of our HTML document.

And because this page is the one that will be used in case the flow is successful, then to display the success messages (actionMessage) sent from the execute method of the ValidateUserAction.java class when the provided user is correct, we must work with the tag: <s:actionmessage />

Let's see how our JSP is.

6. MODIFY THE JSP

- We open the file welcome.jsp:



6. MODIFY THE CODE

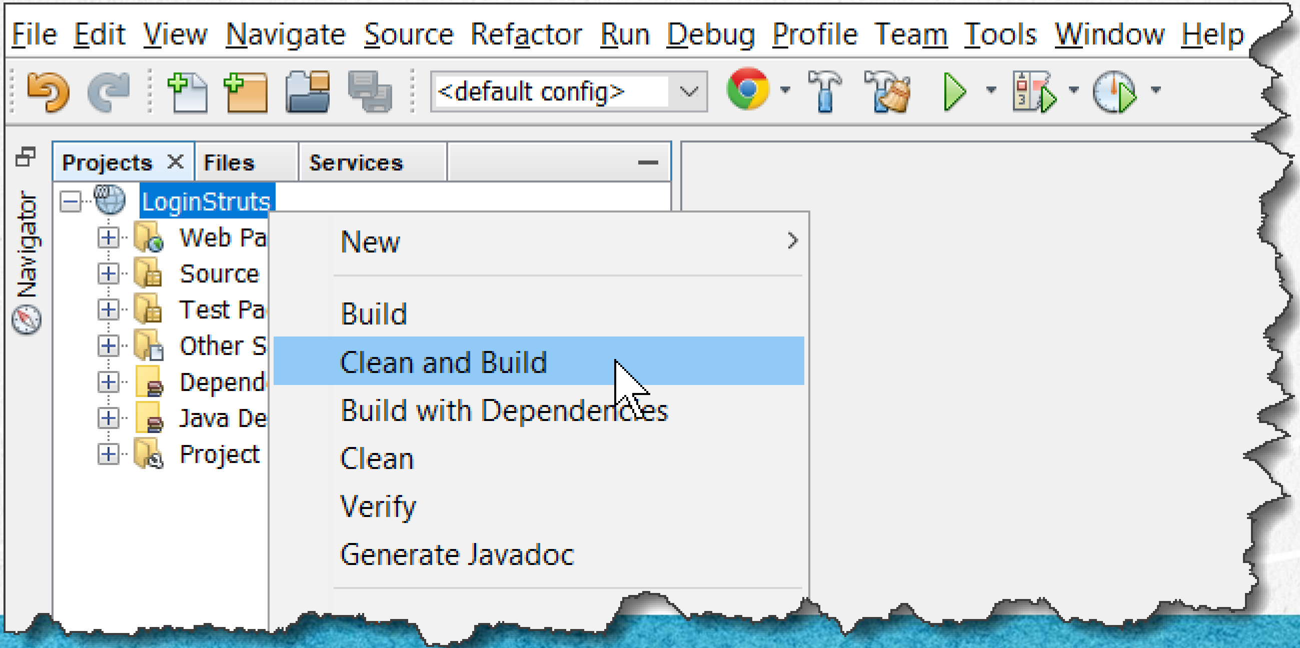
welcome.jsp:

Click to download

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags" %>
<html>
    <head>
        <title><s:text name="welcome.title" /></title>
        <s:head />
    </head>
    <body>
        <h1><s:text name="welcome.title" /></h1>

        <s:actionmessage/>

        <h2>
            <s:text name="welcome.message" /> <s:property value="user"/>
        </h2>
        <div>
            <a href="<s:url action="login"/>"><s:text name="welcome.return" /></a>
        </div>
    </body>
</html>
```

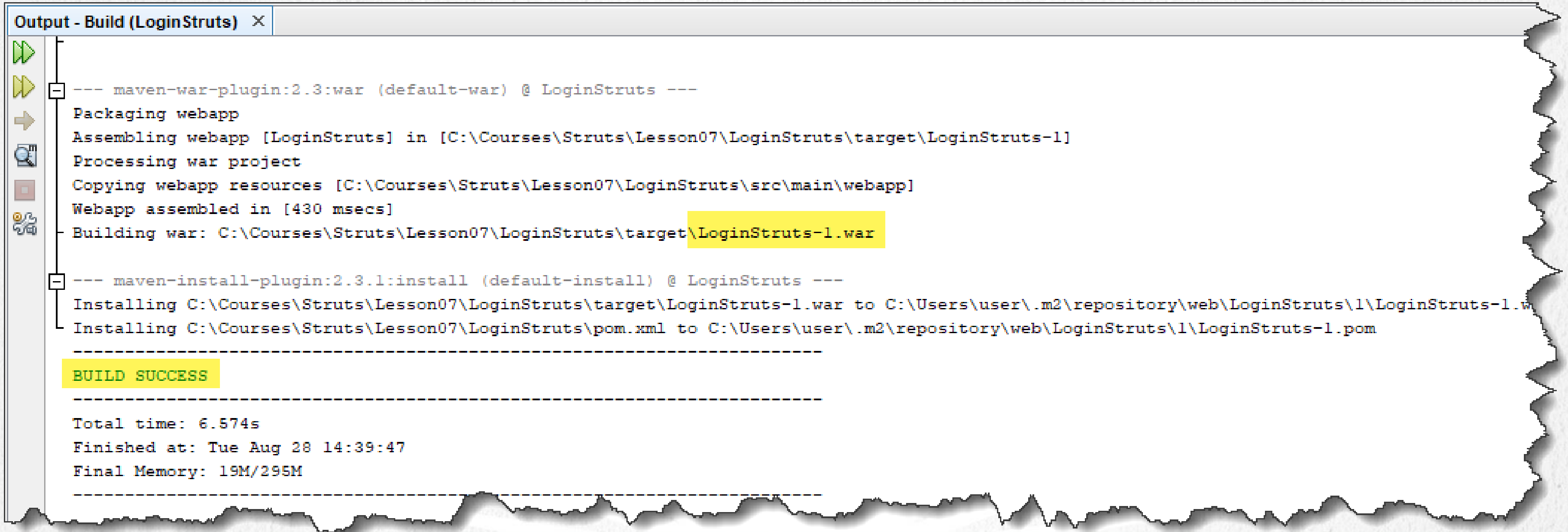
7. EXECUTE CLEAN & BUILD

- We execute the Clean & Build command as shown, to obtain the latest version of each file:



7. EXECUTE CLEAN & BUILD

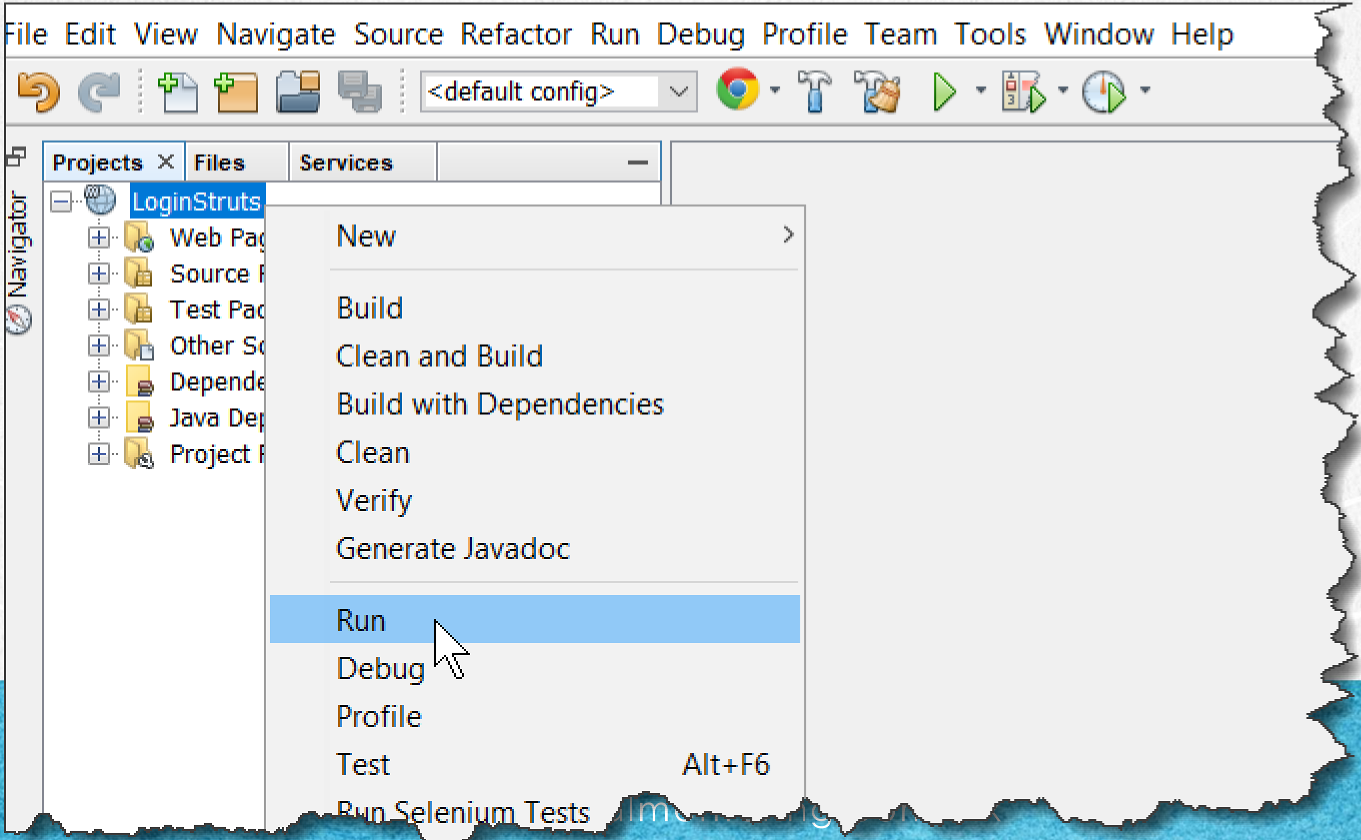
- We must observe a result similar to the following:



```
--- maven-war-plugin:2.3:war (default-war) @ LoginStruts ---
Packaging webapp
Assembling webapp [LoginStruts] in [C:\Courses\Struts\Lesson07\LoginStruts\target\LoginStruts-1]
Processing war project
Copying webapp resources [C:\Courses\Struts\Lesson07\LoginStruts\src\main\webapp]
Webapp assembled in [430 msecs]
Building war: C:\Courses\Struts\Lesson07\LoginStruts\target\LoginStruts-1.war

--- maven-install-plugin:2.3.1:install (default-install) @ LoginStruts ---
Installing C:\Courses\Struts\Lesson07\LoginStruts\target\LoginStruts-1.war to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.w
Installing C:\Courses\Struts\Lesson07\LoginStruts\pom.xml to C:\Users\user\.m2\repository\web\LoginStruts\1\LoginStruts-1.pom
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time: 6.574s
Finished at: Tue Aug 28 14:39:47
Final Memory: 19M/295M
------------------------------------------------------------------------
```

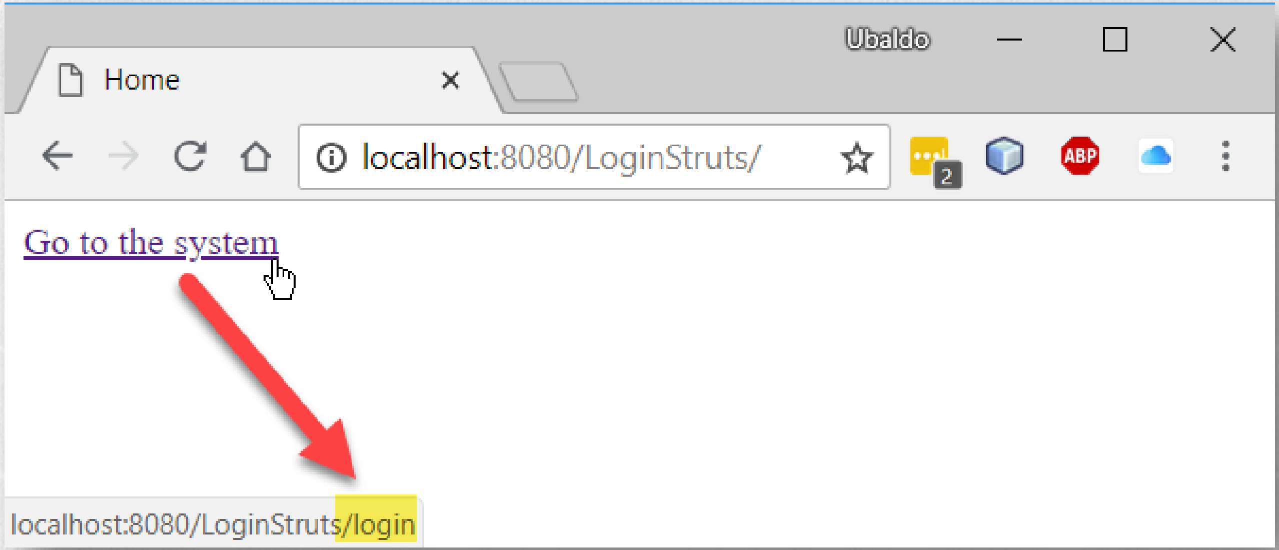
8. EXECUTE THE APPLICATION

- We execute the LoginStruts application as follows :



8. EXECUTE THE APPLICATION

- We run the application as follows:

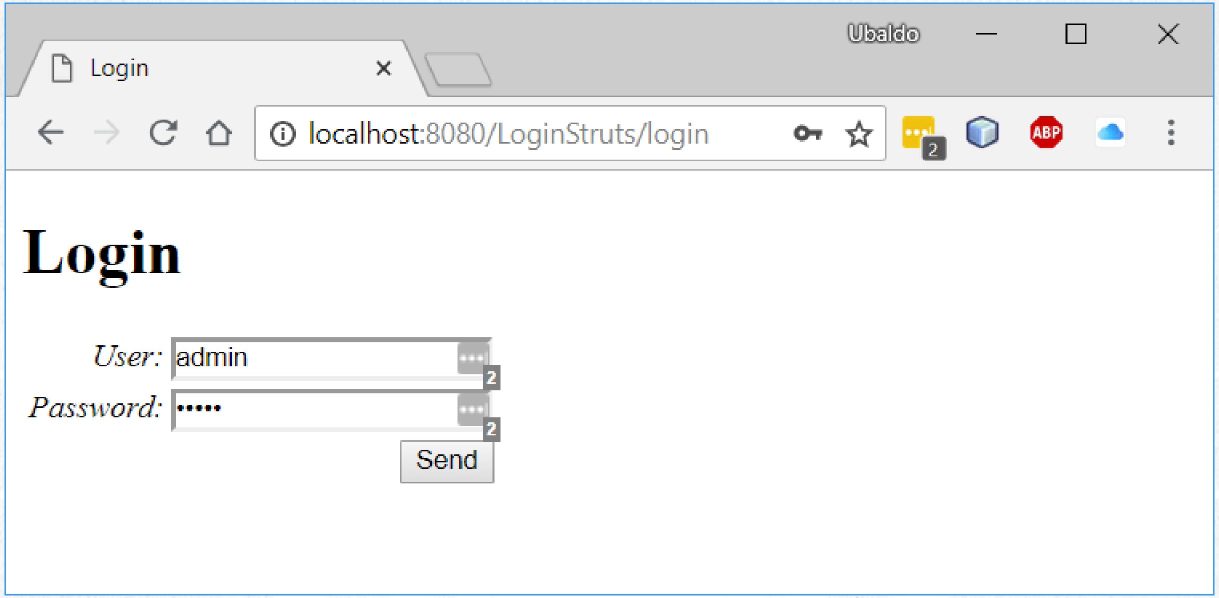


Go to the system

localhost:8080/LoginStruts/login

8. EXECUTE THE APPLICATION

- It shows us the login page with the new CSS styles of Struts 2 :



Login

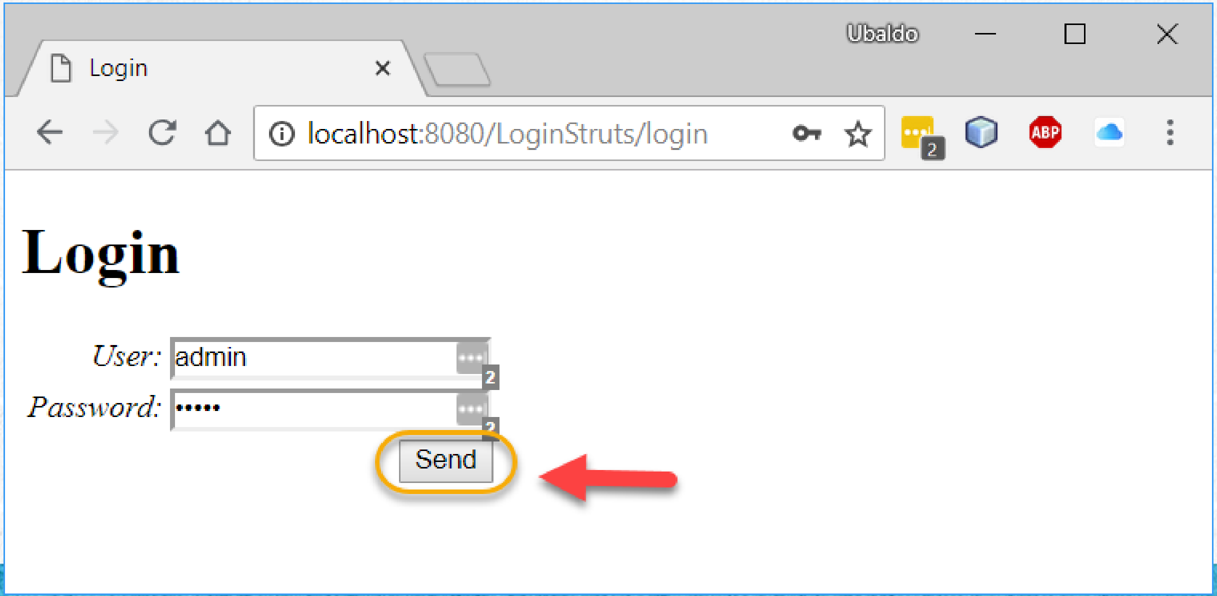
User: admin

Password: •••••

Send

8. EXECUTE THE APPLICATION

- We look at the form, and if we provide the value of "admin" in the user field it will direct us to the path of /validateUser and the result will be "success" showing us as a result the welcome.jsp view :



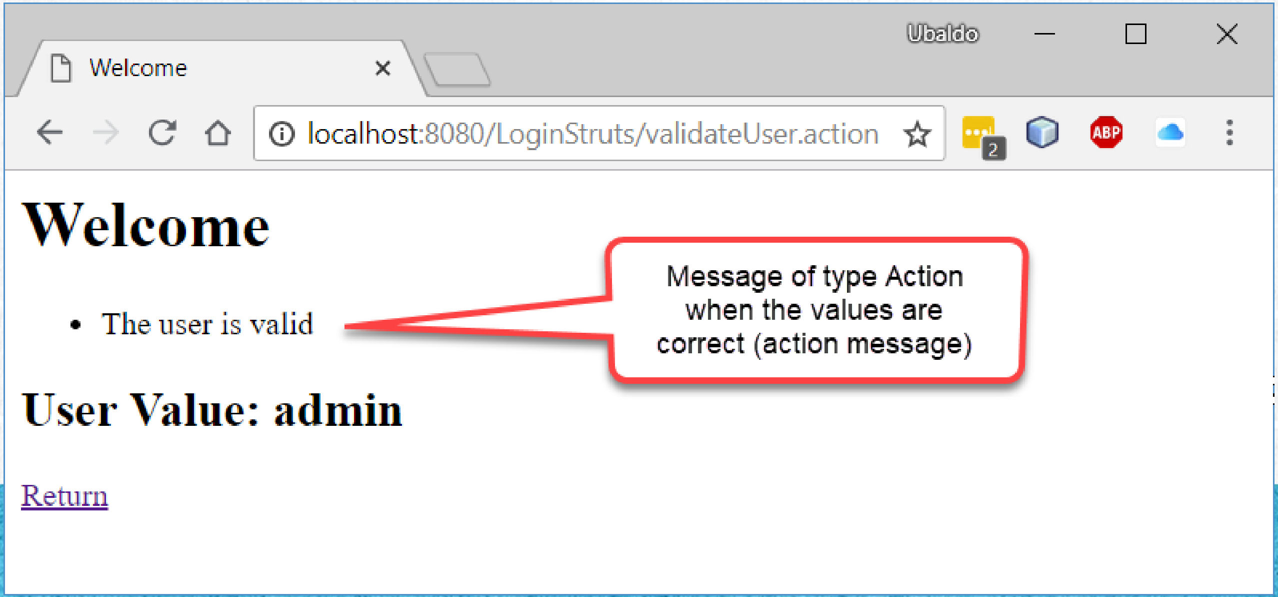
Login

User: admin

Password: •••••

8. EXECUTE THE APPLICATION

- We observe the value provided by the user, and we see that the action that was executed is the path of: /validateUser, showing us the view of welcome.jsp, and also the message of type action message indicating that the user is valid :



Welcome

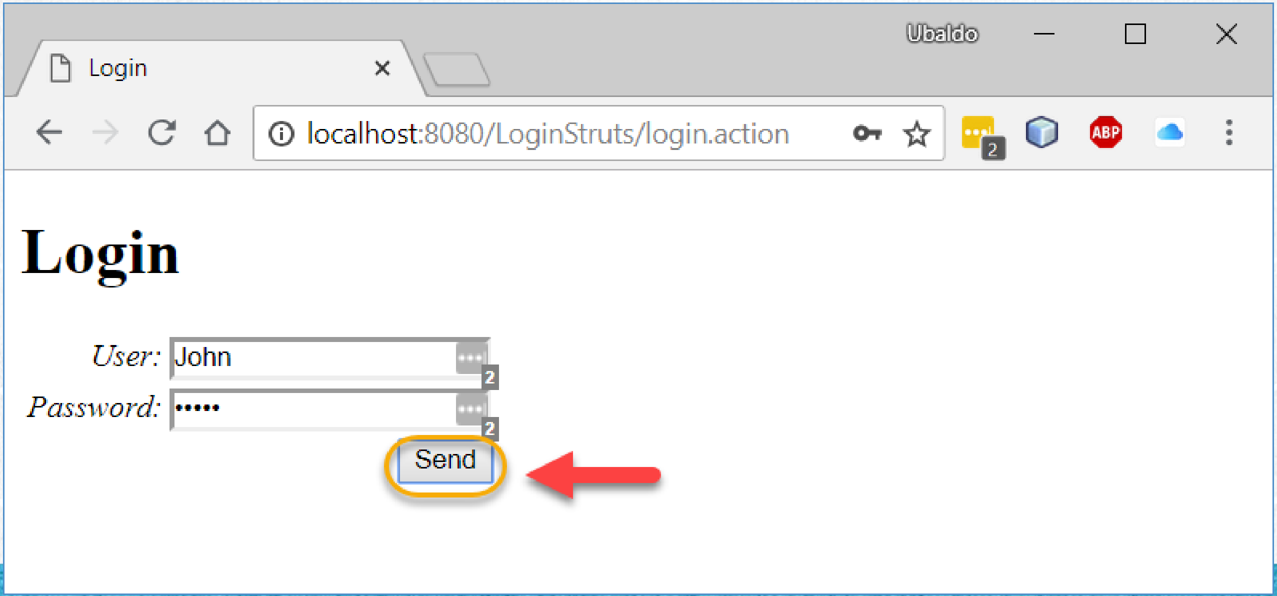
- The user is valid

Message of type Action when the values are correct (action message)

User Value: admin

8. EXECUTE THE APPLICATION

- We return to the login.jsp view and look at the form, and if we provide the value other than "admin" in the user field, it will direct us to the path of /validateUser and the result will be "input" showing us the result of login.jsp again. :



Login

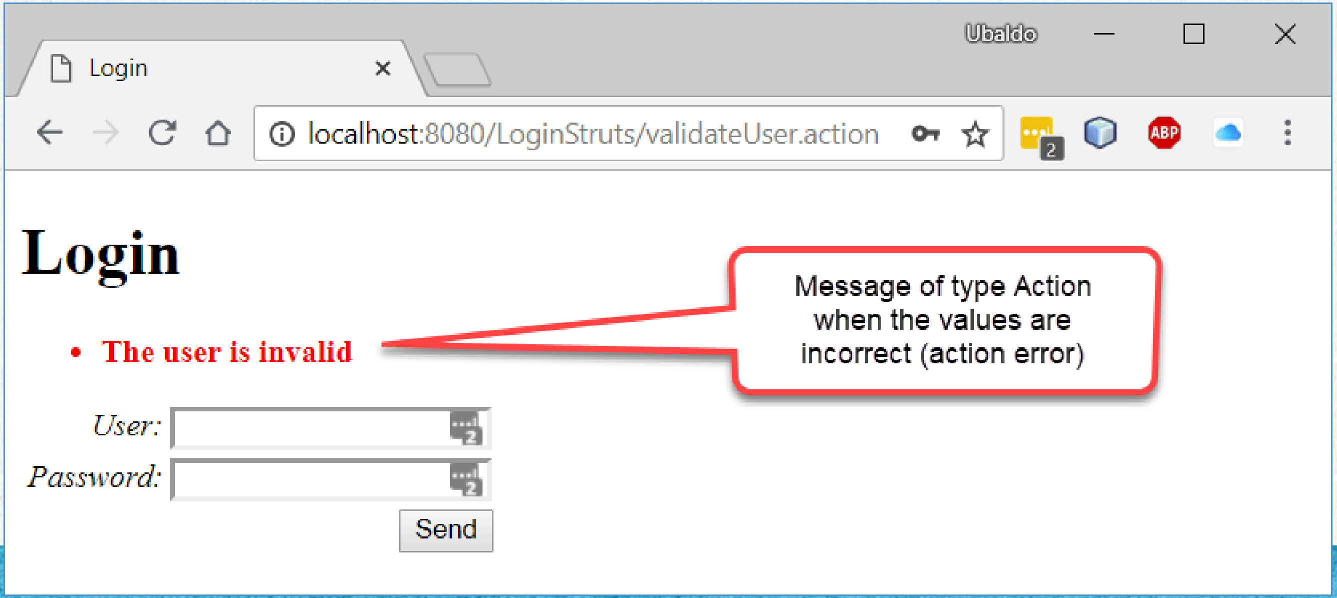
User: John

Password: •••••

Send

8. EXECUTE THE APPLICATION

- We see that the action that was executed is the path of: validateUser, and because the user value was not equal to "admin", then it returned as a result the "input" string, showing us again the login.jsp view. In addition, we add a message of type action error in the validate method of the class ValidateUserAction.java to indicate that the value of the user entered is wrong.



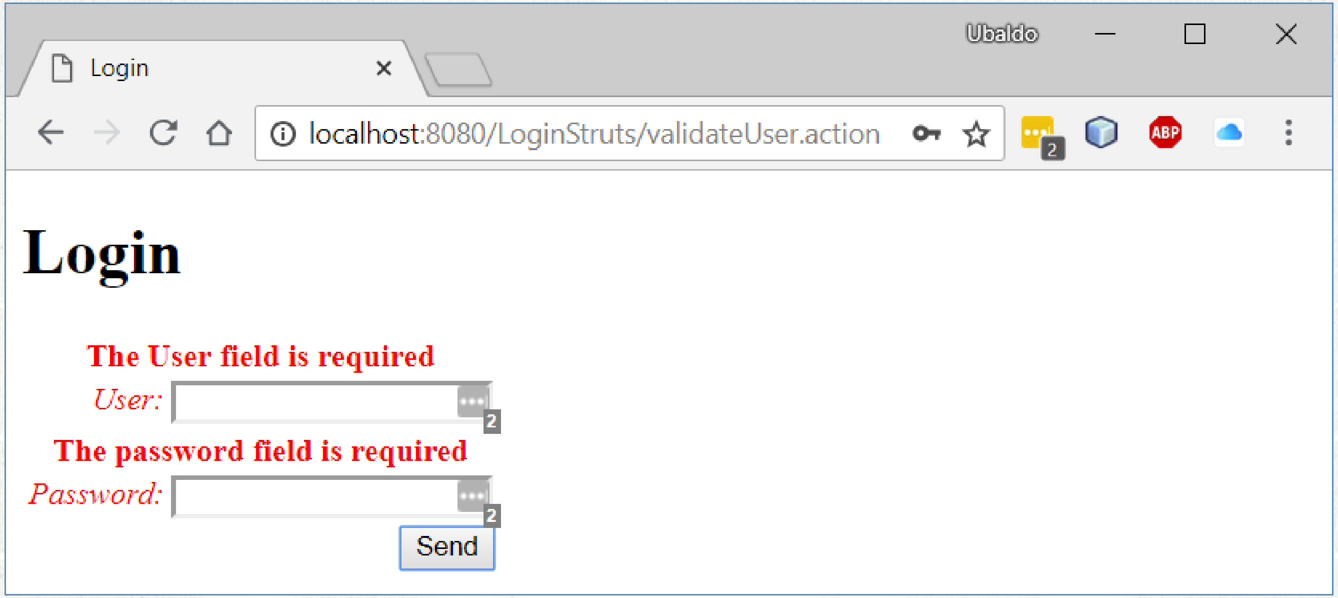
Login

- The user is invalid

Message of type Action when the values are incorrect (action error)

8. EXECUTE THE APPLICATION

- If we send empty values, we see that the validate method of the ValidateUsuarioAction.java class is executed and it returns the respective messages in each field that throws an error.



Login

The User field is required

User:

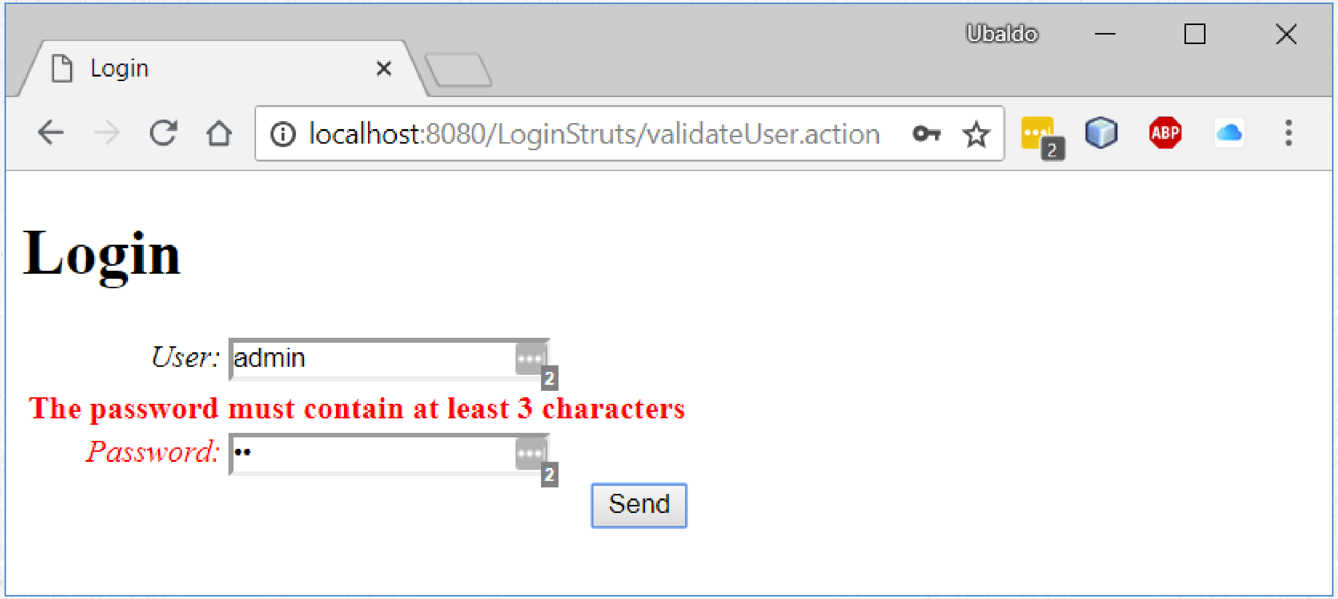
The password field is required

Password:

Send

8. EXECUTE THE APPLICATION

•If we send the correct user value, but the password value is less than 3 characters, we also receive an error, and it shows it on the login.jsp page :



Login

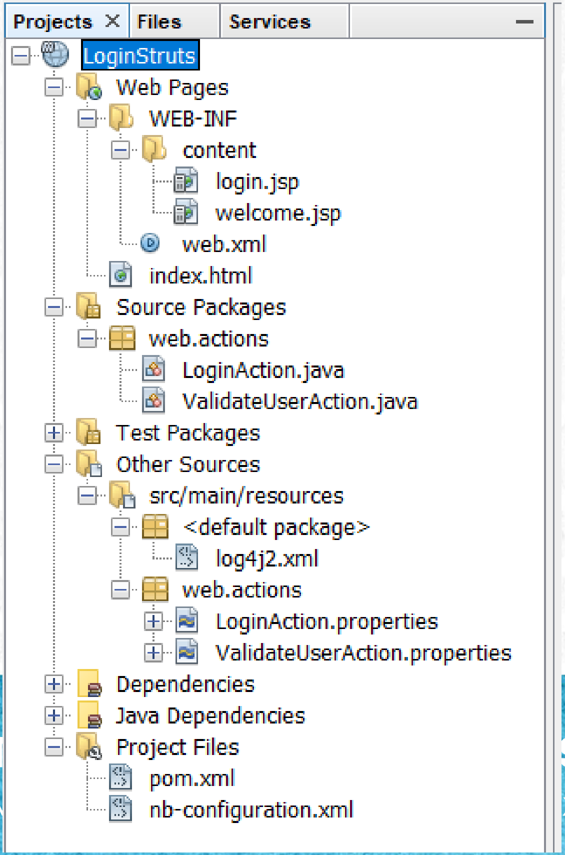
User: admin

The password must contain at least 3 characters

Password: ••

FINAL STRUCTURE OF THE PROJECT

At the end of the exercise the structure should be as follows.



```
LoginStruts
    Web Pages
        WEB-INF
            content
                login.jsp
                welcome.jsp
            web.xml
        index.html
    Source Packages
        web.actions
            LoginAction.java
            ValidateUserAction.java
    Test Packages
    Other Sources
        src/main/resources
            <default package>
                log4j2.xml
            web.actions
                LoginAction.properties
                ValidateUserAction.properties
    Dependencies
    Java Dependencies
    Project Files
        pom.xml
        nb-configuration.xml
```

FINAL RECOMMENDATIONS

If for some reason the exercise fails, several things can be done to correct it:

1. Stop the Glassfish server
2. Make a Clean & Build project to have the most recent version compiled
3. Restart the project (deploy the project to the server again)

If the above does not work, you can try loading the resolved project which is 100% functional and rule out configuration problems in your environment or any other code error.

EXERCISE CONCLUSION

With this exercise we put into practice the handling of Validations in Struts 2.

Although there are more ways to make validations in Struts 2, we already have the bases to start working with this powerful feature.

In addition we saw in a general way the way to send both error and success messages to the views, and in this way to notify not only the validations at the field level of the form, but also at a general level of the result of the error view (input) and welcome view (success).

ONLINE COURSE

STRUTS 2 FRAMEWORK

By: Eng. Ubaldo Acosta

STRUTS FRAMEWORK COURSE

www.globalmentoring.com.mx