STRUTS

STRUTS 2 FRAMEWORK
ADVANCED

Review

1. Send request — Browser → Web Server

2. Send request — Web Server → Filter Dispatcher

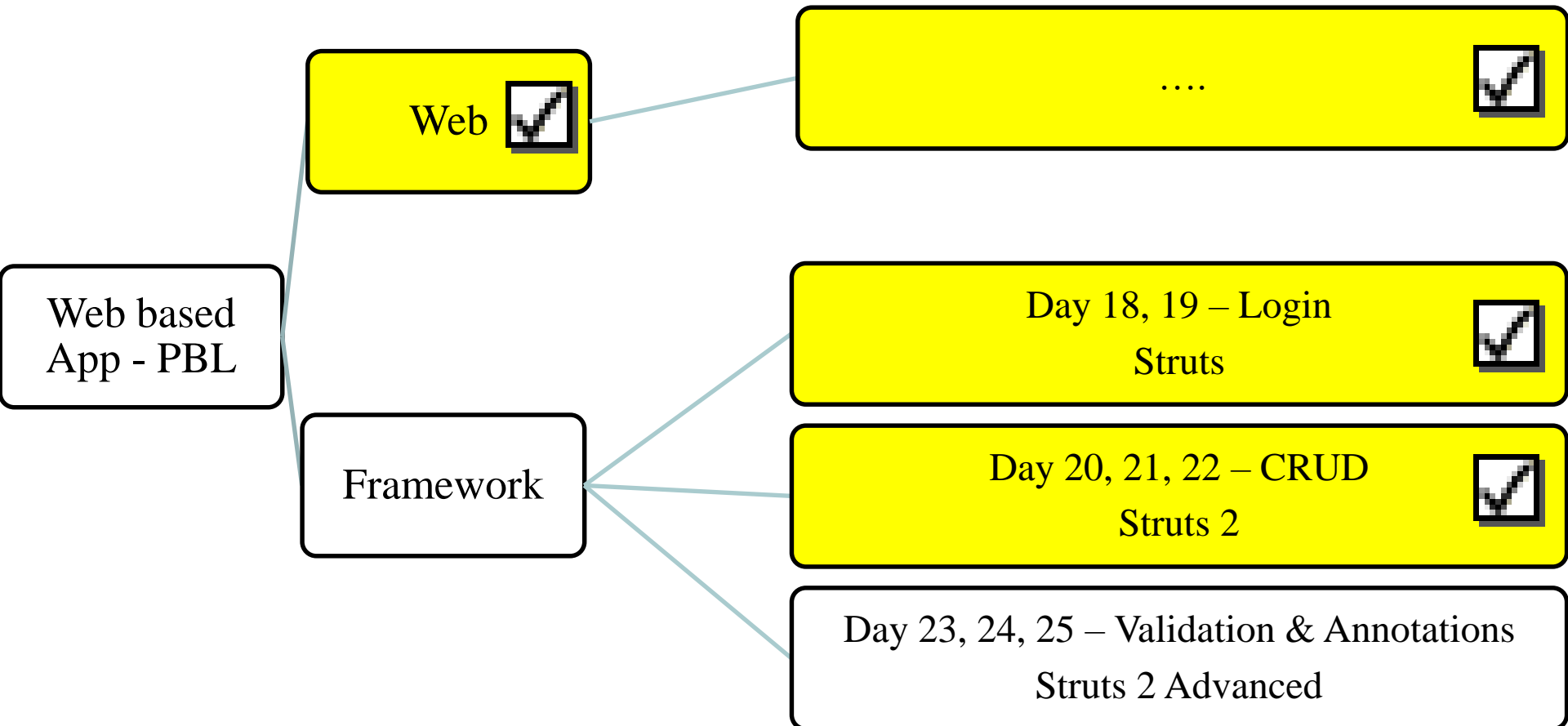**10. Send response**

6. forward

3. uses

5. return

4. read

7". Access

7'. Call

7. invoke

8. select

9. response

OGNL

Browser · Web Server · Filter Dispatcher

DB

DAO

Interceptor 1
Interceptor 2
…
Interceptor n

Action · Result

Configuration Manager

struts.xml

Action Context

Value Stack · *request* · *session* · *….*

OGNL

Client

Server

Review

Value Stack

Send Request

```
<s:form action="login">
  <s:textfield name="username" />
  <s:password name="password"/>
</s:form>
```

Reg Obj

OGNL

Login Action String username; Integer password;

Result

Send Response

```
result.html
Username: khanhkt
Password: 123456
```

```
result.jsp
Username: <s:property value="username"/>
Password: <s:property value="password"/>
```

OGNL

Objectives

- Validations
- Annotations

```
Web based          Web ☑ ───── ....  ☑
App - PBL
        │
        │              Day 18, 19 – Login    ☑
        │              Struts
     Framework ──────  Day 20, 21, 22 – CRUD  ☑
                       Struts 2
                       Day 23, 24, 25 – Validation & Annotations
                       Struts 2 Advanced
```

STRUTS 2
Expectation

Invalid username or password

Click here to try again!!!
Click here to sign up!!!

http://localhost:8084/Strut2Config/signUp

Create New Account

Username:
Password:
Confirm:
Full name:

Create New Account

Reset

http://localhost:8084/Strut2Config/createRecord.action;jsessionid=32BFC75038E92153980725EDC72011BB

STRUTS 2
Expectation

STRUTS 2

Expectation
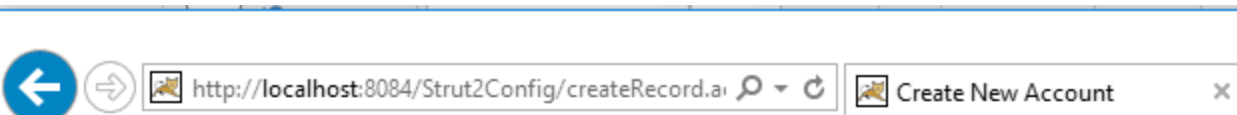
Create New Account

Username:	khanhkt
Password:	
Confirm:	

Lastname length is required 2 - 50 chars

Full name:	a

Create New Account

Reset

Login Page

Username	
Password	

Login   Reset

Create New Account

Username:	khanhkt
Password:	
Confirm:	
Full name:	Kieu Trong Khanh

Create New Account

Reset

khanhkt is existed!!!!

Validations
Overview

- User **input needs** to be **validated** so as to **ensure that correct data** has been entered as required by the application

- Main features of the Struts 2 framework is its **built-in validation support**
  - Helps to **reduce the code spec** and **makes** the tasks of validating the user input **easy and simple**

- Struts 2 framework **supports both server** and **client** side validation
  - Provides a wide range of validation rules and pre-defined validators which can be applied to form fields for validations
  - Developers can create custom validators based on the validation requirements

Validations
Validation Framework Architecture
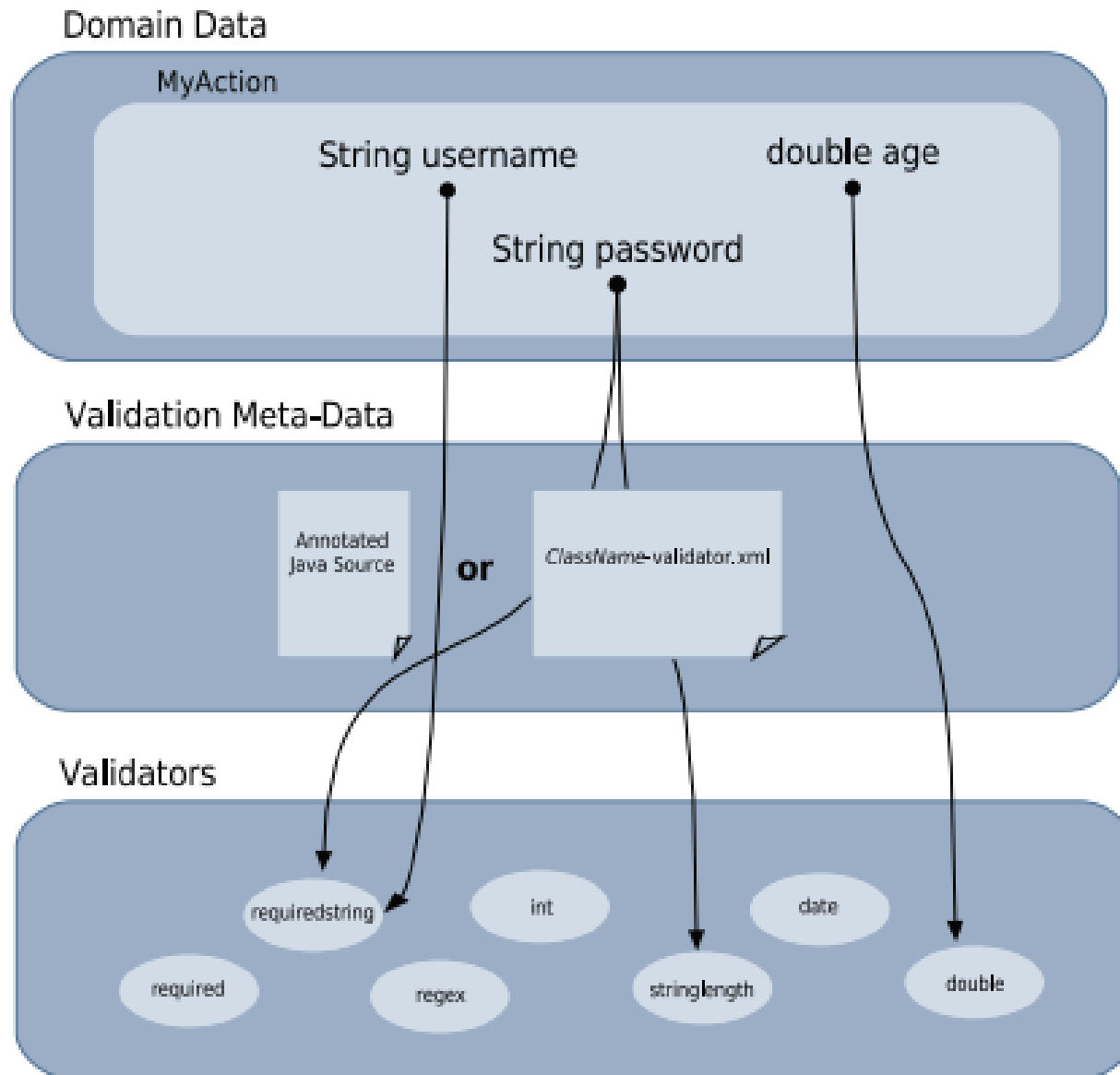
Fig 10.1, Strut2 in Action

Validations
Validation Framework Architecture

- Three layers
  - **Domain Data**: the **data entered** by the user using a browser **exists as properties** in an action class. Then these properties will store the data that the action will work with when it begins execution
  - **Validation Metadata**
    - This is the **middle layer component**
    - It will associate the **individual data properties with the validators** which are used for **verifying** the **correctness** of the values **during runtime**
    - There is no limit to the number of validators combining with each property
    - Mapping can be done **using Java annotations** or **XML files**
  - **Validators**
    - The **actual works of validating** the values present in the data properties is done
    - When the validation executes, each property is validated by the set of validators

Validations
Working of a Validator

- **Once** the user **submits** the **input form**, the **workflow Interceptor** is **executed**

- **First**, it **checks** whether **Validateable** interface is **implemented**

- If it is implemented, then the **validate**() **method** is **invoked** by the workflow Interceptor

- On **encountering an error**, the **validator creates** an **error message** and **adds** it to one of the methods of the ValidationAware interface

- When the **validate**() method is completed, the **workflow Interceptor** will **invoke** the **hasErrors**() method of ValidationAware interface
  - This method is invoked **to check** for the **existence** of **any error message**
  - If it is **existence**, the workflow Interceptor **stops further execution** of the action and **returns the input result**. This result takes the user back to the input form that was submitted

Validations
Scope

- **Two types of validators**
  - **Field** Validators
    - **A field validator** is **associated** with a **form field** and works by **verifying a value before** the value is **assigned** to an **action property**
  - **Plain** validators (non-field validators)
    - Is **not associated** with a field and is used to **test if a certain condition has been met.**

Validations
Field Validators

- **Syntax**

```
<validators>
  <field name="fieldName">
   <field-validator type="dataType">
    [<param name="par">value</param>]

    …
    <message [key="keyName"]></message>
   </field-validator>
  </field>

  ...
</validators>
```

- **Validators: declaring** of the individual validators is applied
- **name**: **specifies** the **form field** to be validated
- **field-validator: declaring** validator
- **type**: **refers** to the **name** of the **registered validator**.
- **message**: **contains** the **error message**
- **param**: declare **parameters supporting** to type of validator

Validations
Built-in Validators

Name	Parameters	Description
required		Checks that the value is not null
requiredstring	trim	Checks that the value is not null and is not empty. The default value for the attribute trim is true indicating that it trims white space
stringlength	trim, minLength, maxLength	Checks that the length of the string is within the parameters specified by minlength and maxlength. If length is not specified then checks are not made. In minlength is not specified then an empty string would pass validation
int	min, max	Checks that the integer value is within the value specified by min and max
double	minInclusive, maxInclusive, minExclusive, maxExclusive	Checks that the double value is within the inclusive or exclusive specified parameters
url		Checks the URL format

Validations
Built-in Validators

Name	Parameters	Description
date	min, max	Checks that the date value is within the specified date as specified by min and max attribute. Format of the date value is MM/DD/YYYY
email		Checks the format of the email address
url		Checks the URL format
fieldexpression	expression	Evaluates the OGNL expression against the current ValueStack and returns true or false depending on the result of evaluation
regex	expression, caseSensitive, trim	Checks that the string confirms to the regular expression
expression	expression	Used at action level and is same as fieldexpression
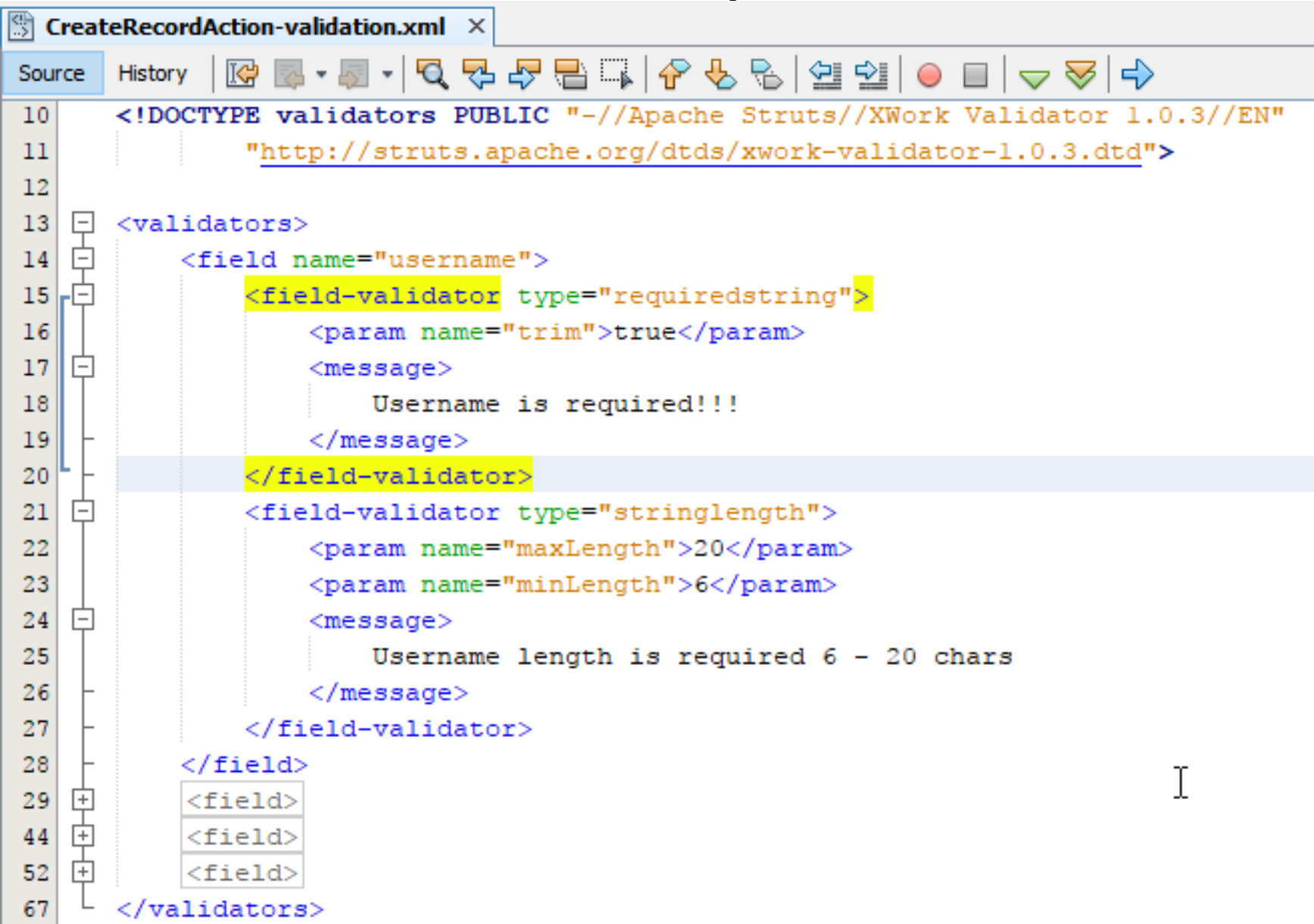
Validation
Validator Implementation

- **Step 1**: **Create all view** with JSP that can or cannot contain the htlm form
  - The **<s:head />** tag should be **placed** in the **head section** of the HTML page in the page applying the validation.
  - The s:head tag **automatically generates** links to the **css** and **JavaScript** libraries that are necessary to render the form elements
- **Step 2**: Create the **Action class**
  - **Extends** the **ActionSupport class**
- **Step 3**: **Create** the **validation metadata** file formed as **ActionClassName-validation.xml**
- **Step 4**: **Mapping action** (struts.xml)
  - The result with input value must be declared to present the input form in path that when errors occurring, the page is thrown backward to input form

Validation
Validator Implementation

Source | History

```xml
10  <!DOCTYPE validators PUBLIC "-//Apache Struts//XWork Validator 1.0.3//EN"
11          "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">
12
13  <validators>
14      <field name="username">
15          <field-validator type="requiredstring">
16              <param name="trim">true</param>
17              <message>
18                  Username is required!!!
19              </message>
20          </field-validator>
21          <field-validator type="stringlength">
22              <param name="maxLength">20</param>
23              <param name="minLength">6</param>
24              <message>
25                  Username length is required 6 - 20 chars
26              </message>
27          </field-validator>
28      </field>
29      <field>
44      <field>
52      <field>
67  </validators>
```

Exception Handling
Handle per Action

- **To handle exception** that throws from execute methods in Action class on the view, **the exception-mapping node** within the action node **is recommended**
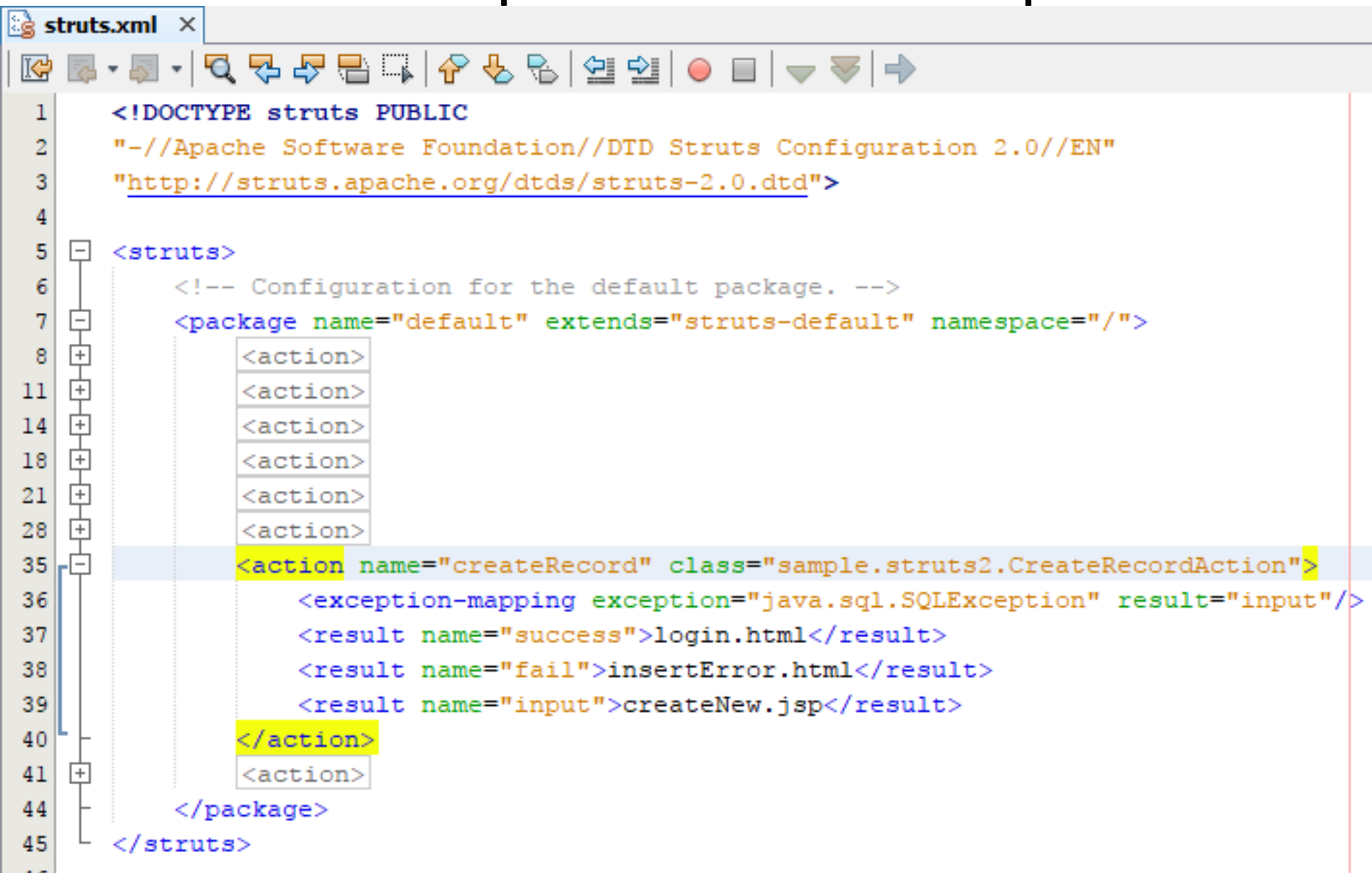
  Syntax

  <exception-mapping exception="exception class" result="name" />

  - If **execute() method throws** an uncaught exception of type **exception_class**, the Struts 2 framework should **return a name result**

  - **The syntax should be put before the result tag**

- In the **view**, the **exception** object is **available** in object Stack **for accessing** using Struts tag library

Exception Handling
Handle per Action – Example

```
struts.xml  ×

1   <!DOCTYPE struts PUBLIC
2   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
3   "http://struts.apache.org/dtds/struts-2.0.dtd">
4
5   <struts>
6       <!-- Configuration for the default package. -->
7       <package name="default" extends="struts-default" namespace="/">
8           <action>
11          <action>
14          <action>
18          <action>
21          <action>
28          <action>
35          <action name="createRecord" class="sample.struts2.CreateRecordAction">
36              <exception-mapping exception="java.sql.SQLException" result="input"/>
37              <result name="success">login.html</result>
38              <result name="fail">insertError.html</result>
39              <result name="input">createNew.jsp</result>
40          </action>
41          <action>
44      </package>
45   </struts>
```

Annotations
Overview

- struts.xml is **eliminated**
- The **web.xml configuration** file **must be modified**
  - The packages that have actions **must be defined** in the **<init-param>** element of the filter configuration
    - **<param-name> and the <param-value>** elements have been used to **specify the location of the packages containing** the **action classes** using annotations
  - **Each of** these **packages** and **sub** packages will be **checked** for classes that **implement Action or ActionSupport** class or **whose name ends** in **Action**
  - **Syntax**

    <filter>
      <filter-name>struts2</filter-name>
      <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
      <init-param>
       <param-name>paramName</param-name>
       <param-value>package1[,package2..n]</param-value>
      </init-param>
    </filter>

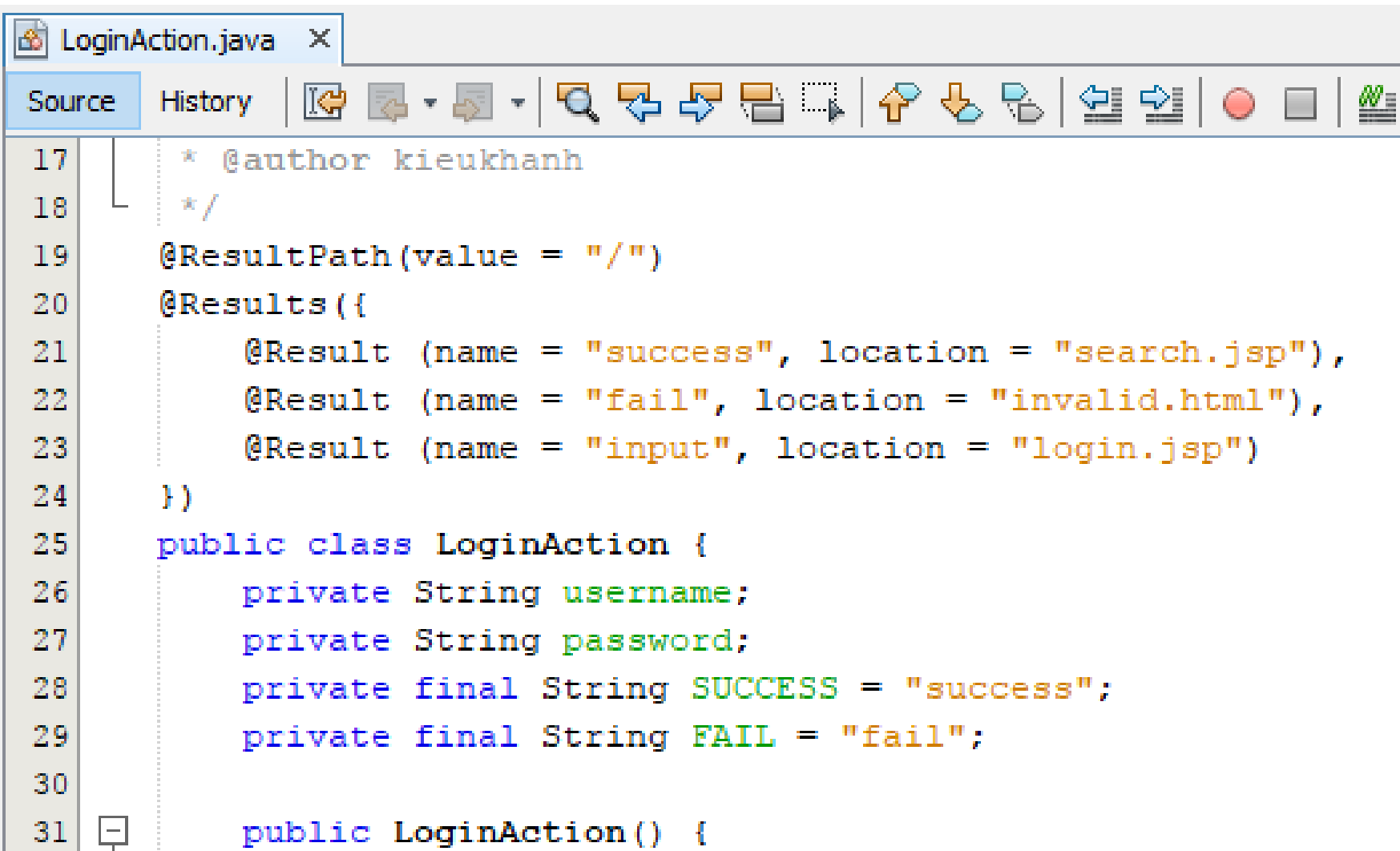# Annotations
Example

Annotations
Action

- **@Namespace**
  - Helps to **set the path** of the **action URL**
  - **Placed before** the **action class** and it applies to all actions defined in the class
- **@Result and @Results**
  - Used for **configuring** a **result** for the return value
  - Can be declared at global (class level) or local level (method)
  - **Accepts four parameters** for configuration:
    - **name**: the **string value returned** from the execute methods that processes the request
    - **location**: a **value** that the **result path** uses in execute methods
    - **type**: the class of the **result type**
    - **parameters**: an array of **string parameters** *as 'name', 'value'[, 'name', 'value', …]*
  - **@Results** is used for configuring **multiple results that** contains @Result
- **@ResultPath**
  - Is used to **change default location** of results
- **@Action**
  - Change the URL that they are mapped to using the Action annotation
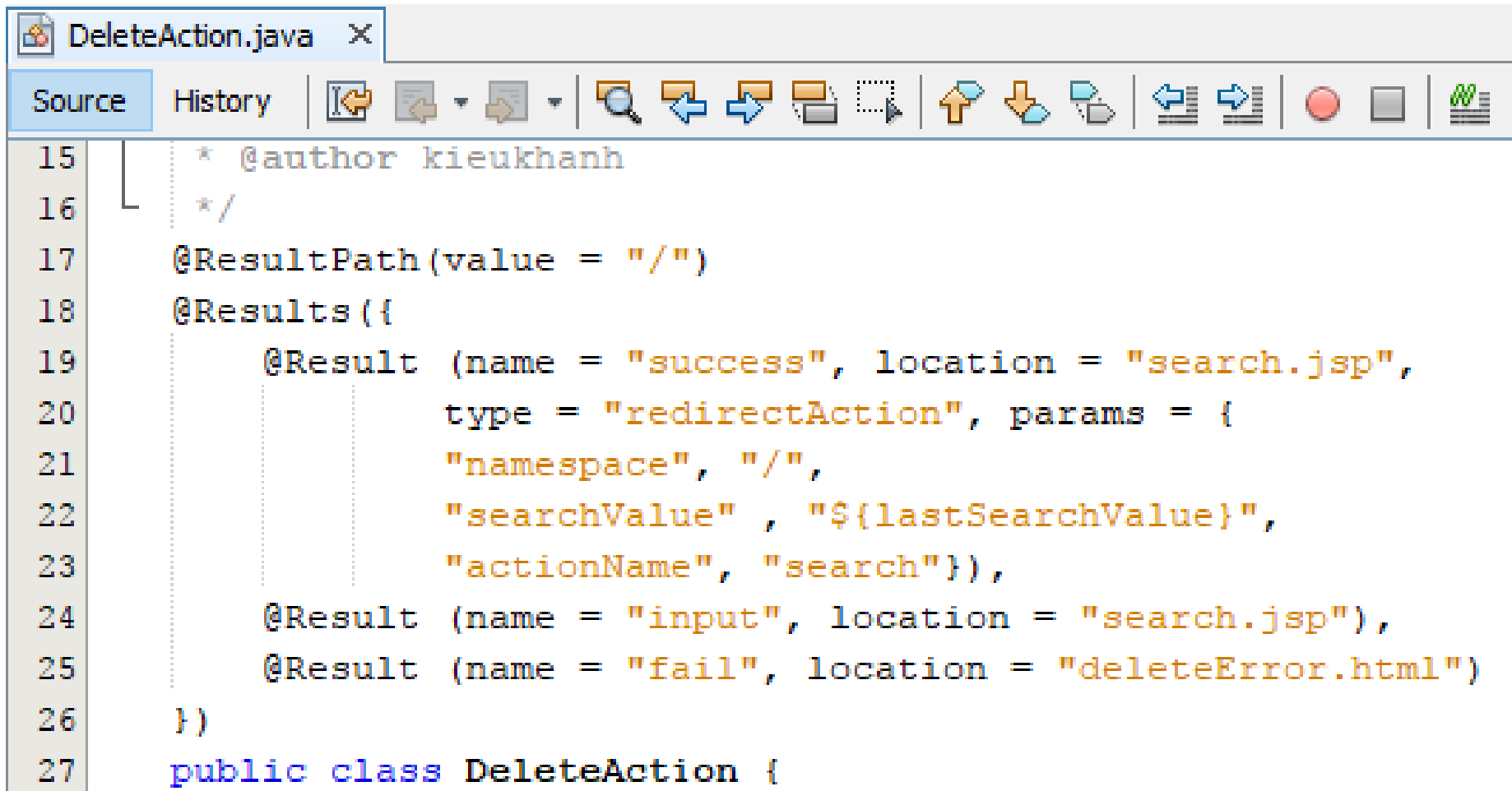 - Must be **defined** on **action methods or put into @Actions**

Annotations
Action – Example

Source | History

```java
17      * @author kieukhanh
18      */
19     @ResultPath(value = "/")
20     @Results({
21         @Result (name = "success", location = "search.jsp"),
22         @Result (name = "fail", location = "invalid.html"),
23         @Result (name = "input", location = "login.jsp")
24     })
25     public class LoginAction {
26         private String username;
27         private String password;
28         private final String SUCCESS = "success";
29         private final String FAIL = "fail";
30
31         public LoginAction() {
```

Annotations
Action – Example


```java
     * @author kieukhanh
     */
    @ResultPath(value = "/")
    @Results({
        @Result (name = "success", location = "search.jsp",
                type = "redirectAction", params = {
                "namespace", "/",
                "searchValue" , "${lastSearchValue}",
                "actionName", "search"}),
        @Result (name = "input", location = "search.jsp"),
        @Result (name = "fail", location = "deleteError.html")
    })
    public class DeleteAction {
```

Annotations
Custom Validators

- Used for configuring custom validators and to group validators for a property or a class

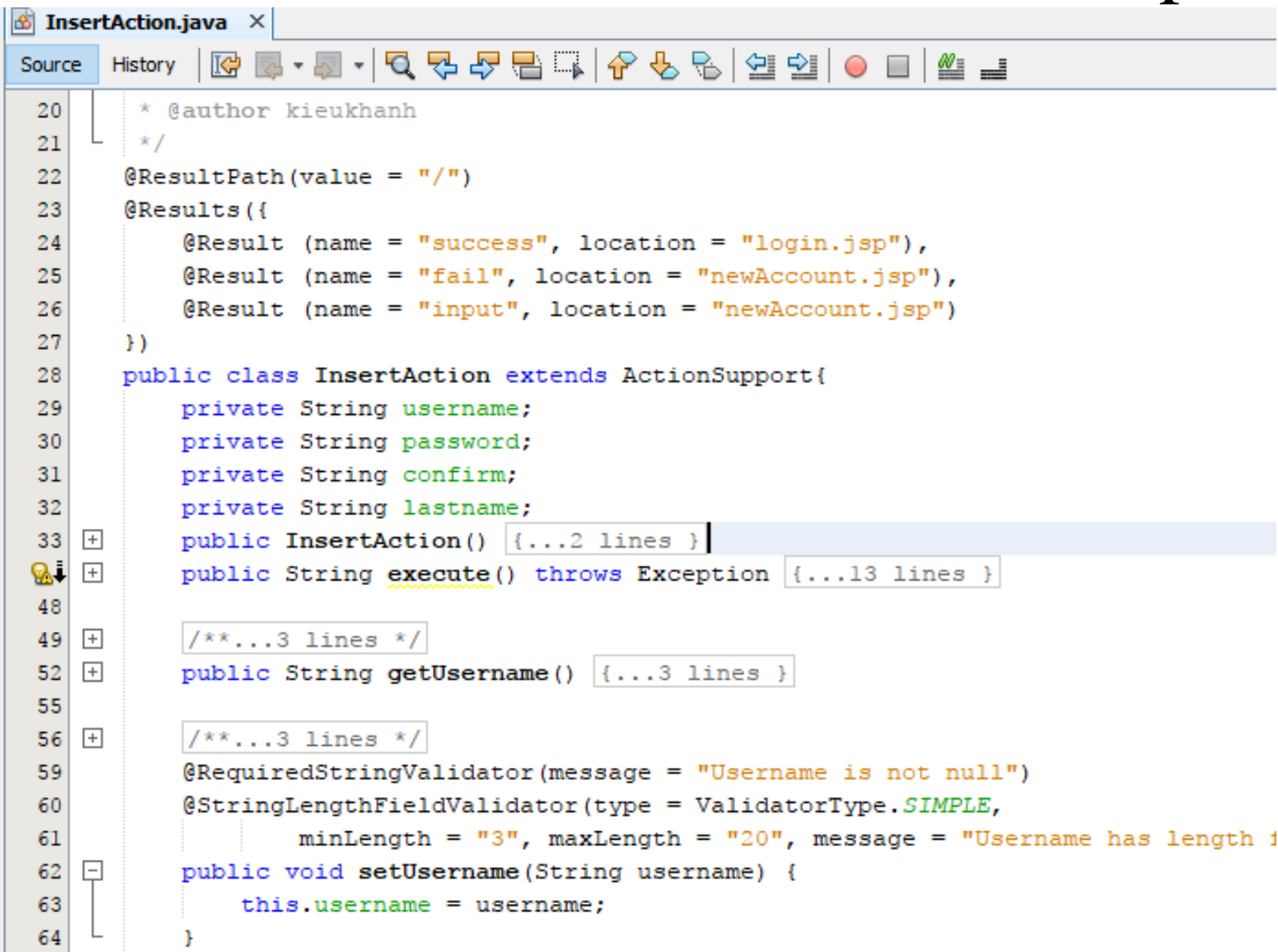| Annotation Name	XML	Description
RequiredFieldValidator	required	Ensures that the property is not null
RequiredStringValidator	requiredstring	Ensures that the String property is not null or empty
StringLengthFieldValidator	stringlength	Checks that the String length is within the specific range
IntRangeFieldValidator	int	Checks that the integer property is within the specific range
DateRangeFieldValidator	date	Checks that the date property is within the specific range
ExpressionVaidator	expression	Evaluates the OGNL expression using the ValueStack
FieldExpressionValidator	fieldexpression	Validates a field using the OGNL expression

Annotations
Custom Validators

Annotation Name	XML	Description
EmailValidator	email	Ensures that the property is a valid email address
UrlValidator	url	Ensures that the property is a valid URL
RegexFieldValidator	regex	Checks whether the property matches the regular expression
Validation		Signifies that the class using annotation based validation can be used on interfaces or classes
Validations		Groups multiple validations for a property or a class

Annotations
Custom Validators – Example

Source | History

```
20        * @author kieukhanh
21        */
22      @ResultPath(value = "/")
23      @Results({
24          @Result (name = "success", location = "login.jsp"),
25          @Result (name = "fail", location = "newAccount.jsp"),
26          @Result (name = "input", location = "newAccount.jsp")
27      })
28      public class InsertAction extends ActionSupport{
29          private String username;
30          private String password;
31          private String confirm;
32          private String lastname;
33          public InsertAction() {...2 lines }
         public String execute() throws Exception {...13 lines }
48
49          /**...3 lines */
52          public String getUsername() {...3 lines }
55
56          /**...3 lines */
59          @RequiredStringValidator(message = "Username is not null")
60          @StringLengthFieldValidator(type = ValidatorType.SIMPLE,
61                  minLength = "3", maxLength = "20", message = "Username has length 1
62          public void setUsername(String username) {
63              this.username = username;
64          }
```
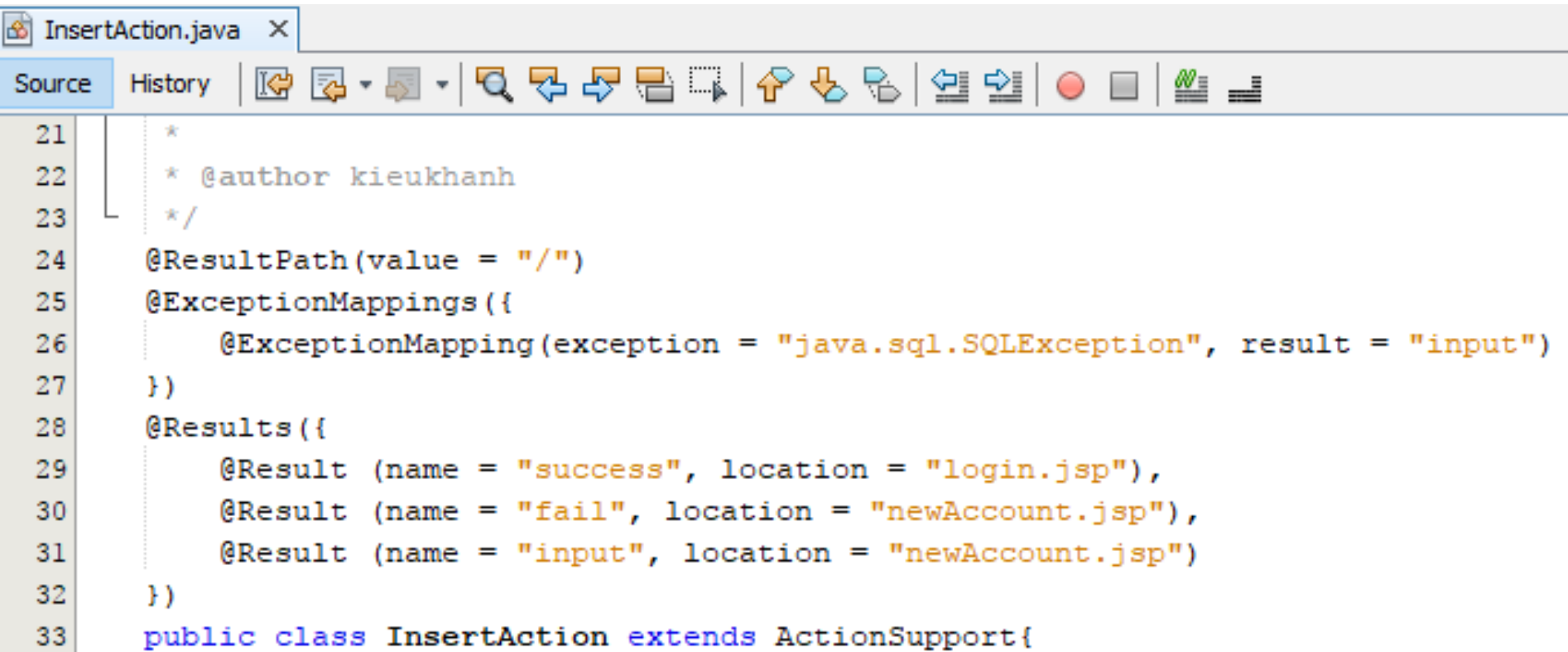
Annotations
Handle Exception per Action

- **@ExceptionMapping**
  - Define **exception mappings** to actions
  - Can be **applied** to the **class level** (all action) that contains in **@ExceptionMappings**
  - Or, the **action level** (method) that contains in **Action** with exceptionMappings **attribute**
  - Accepts **03 parameters**
    - **exception**: define exception class that is thrown
    - **result**: define returned result
    - **params**: all strings parameter is passed

Annotations
Handle Exception per Action – Example

Source   History

```
21        *
22        * @author kieukhanh
23        */
24       @ResultPath(value = "/")
25       @ExceptionMappings({
26           @ExceptionMapping(exception = "java.sql.SQLException", result = "input")
27       })
28       @Results({
29           @Result (name = "success", location = "login.jsp"),
30           @Result (name = "fail", location = "newAccount.jsp"),
31           @Result (name = "input", location = "newAccount.jsp")
32       })
33       public class InsertAction extends ActionSupport{
```

Annotations
Create the Struts 2 projects

- Steps for creating the web application using Struts 2
    - **Step 1**: Create Web application projects with
        - Choosing Web Servers and JavaEE 5
        - Choosing the Struts 2 Frameworks support
    - **Step 2**: Create all views of application that provide GUI
        - Create the form with action as **actionName** that implements the action class in next steps
    - **Step 3**: Create all action class
        - With class form **ActionName**Action (the **first** character must be **capitalized**)
        - Define input properties and accessor methods
        - Implement the execute() method to perform the action
    - **Step 4**: Declare the class with annotation to get result/ results or validation (if it is necessary) without configuring the struts.xml
    - **Step 5**: Build, Deploy and Test application

Summary

Domain Data

MyAction

String username   double age

String password

Validation Meta-Data

Annotated Java Source   **or**   *ClassName*-validator.xml

Validators

requiredstring   int   date

required   regex   stringlength   double
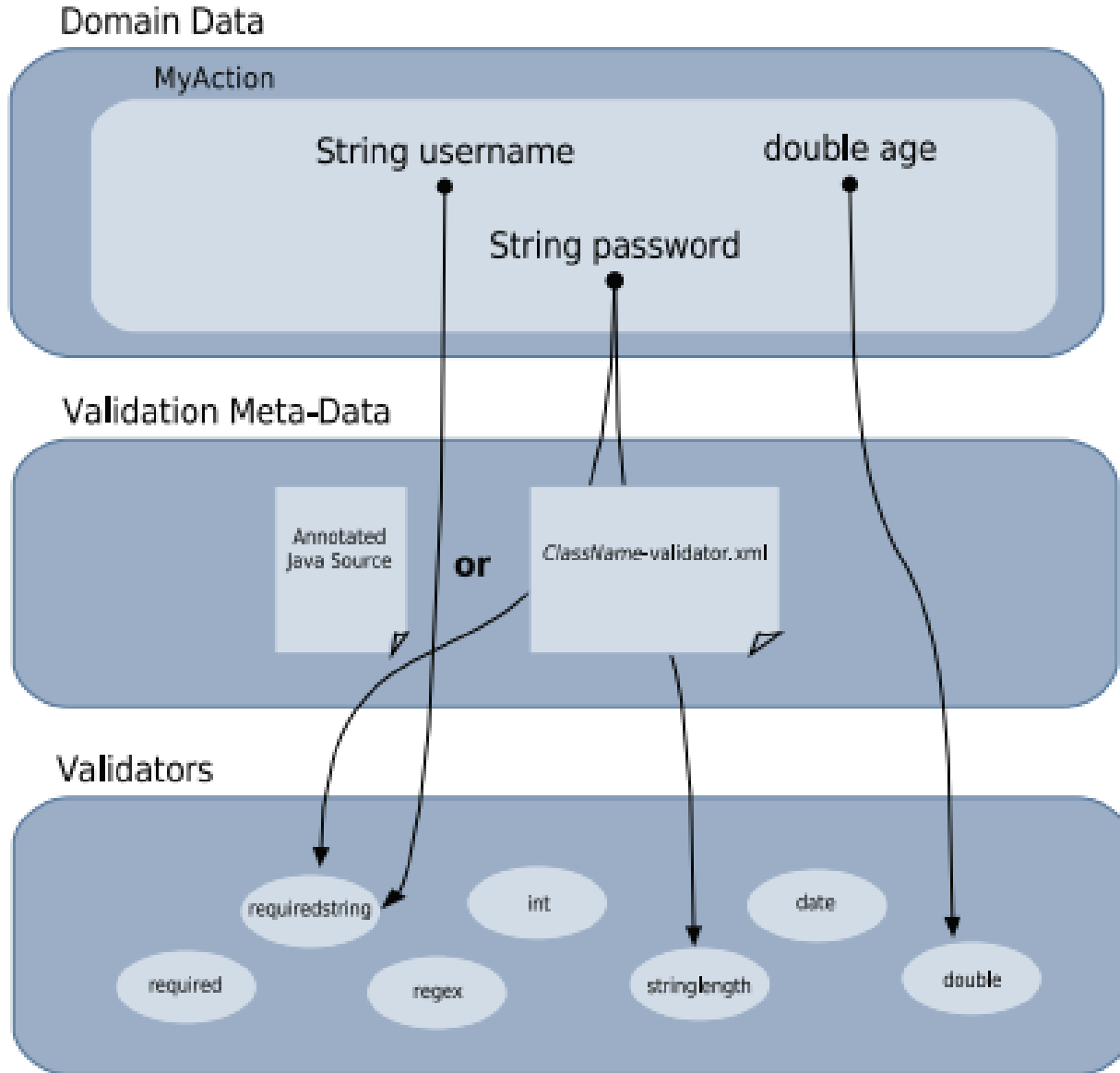
How to use Annotations in Struts 2?

Q&A

Summary

Web based App - PBL

Web ✓

.... ✓

Framework ✓

Day 18, 19 – Login
Struts ✓

Day 20, 21, 22 – CRUD
Struts 2 ✓

Day 23, 24, 25 – Validation & Annotations
Struts 2 Advanced ✓
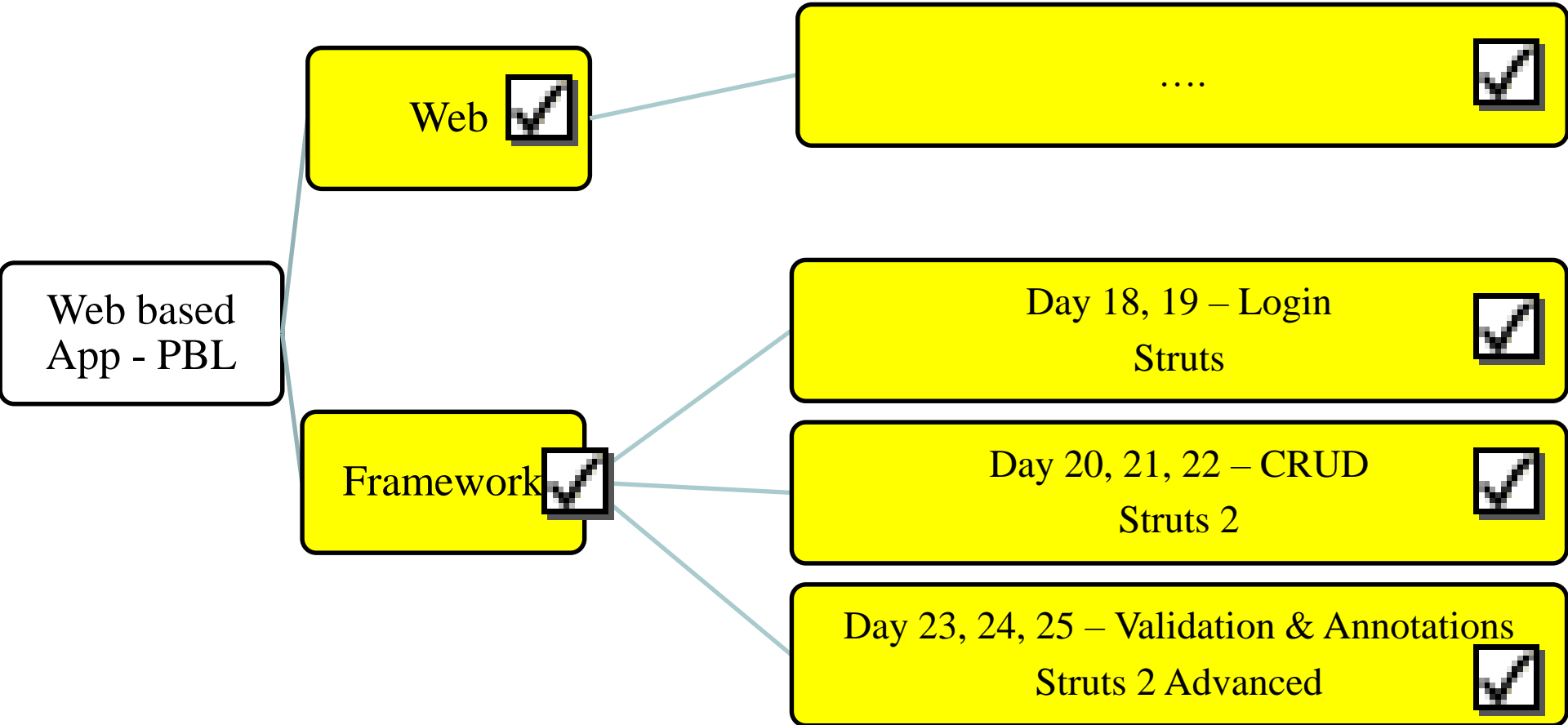
Next Days

- Practice with exercises
- Practical Test
- Presentations

Good luck to you!!!

EXAMPLE – CREATE VALIDATION FILE

- Create the file with ActionClass-validation.xml
  - Create the RegisterActon-validation.xml
- Mapping the visual to support programming the validation file
  - Required: the file dtd with name xwork-validation-1.0.2.dtd
  - In Netbeans, choose tools, choose DTDs and XML Schemas
  - Choose User Catalog, Click Add Local DTD or Schema button
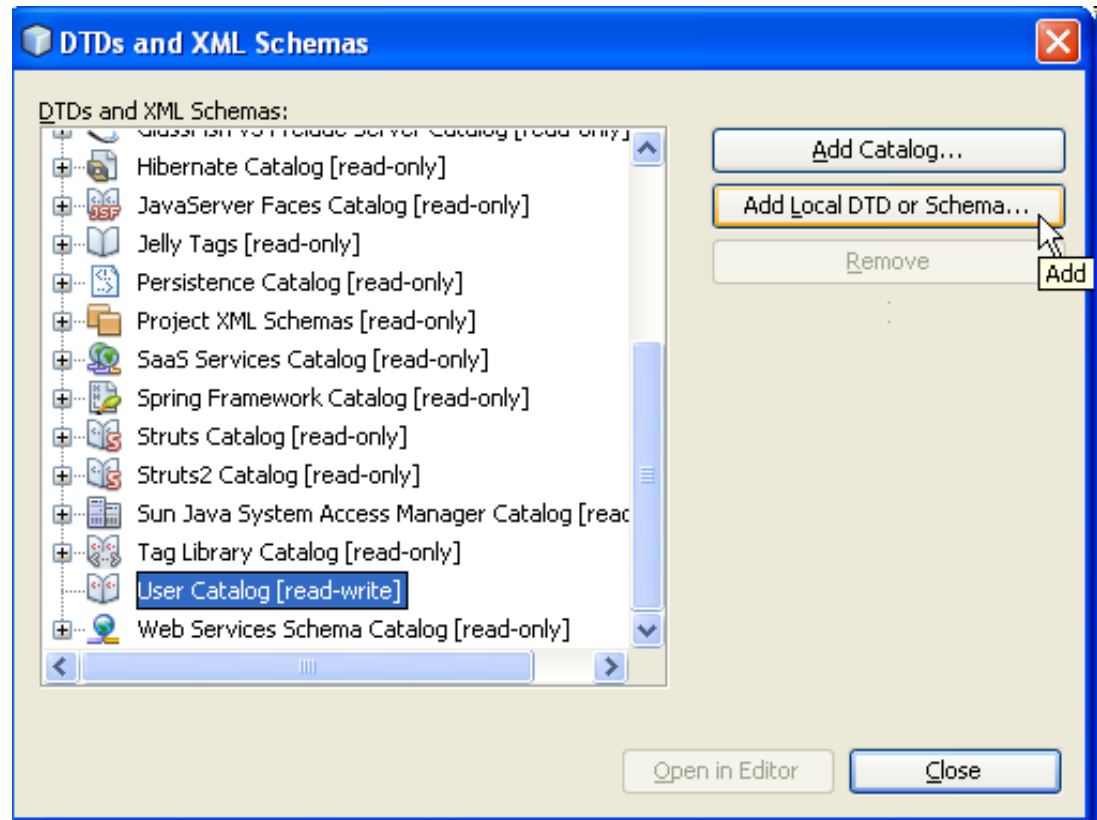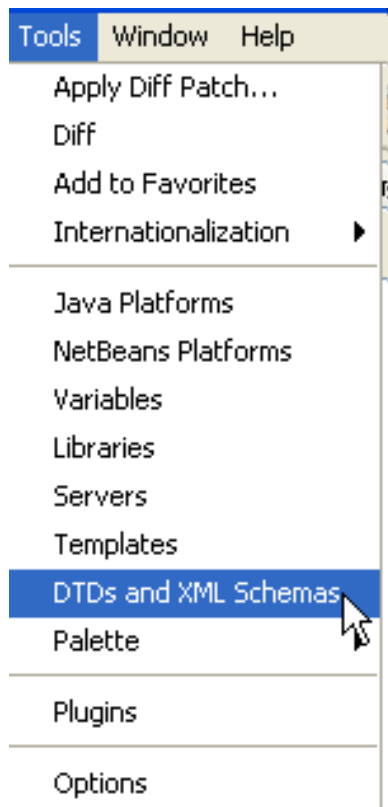
Mapping DTD to Netbeans to support visual code XML for configuring validation file

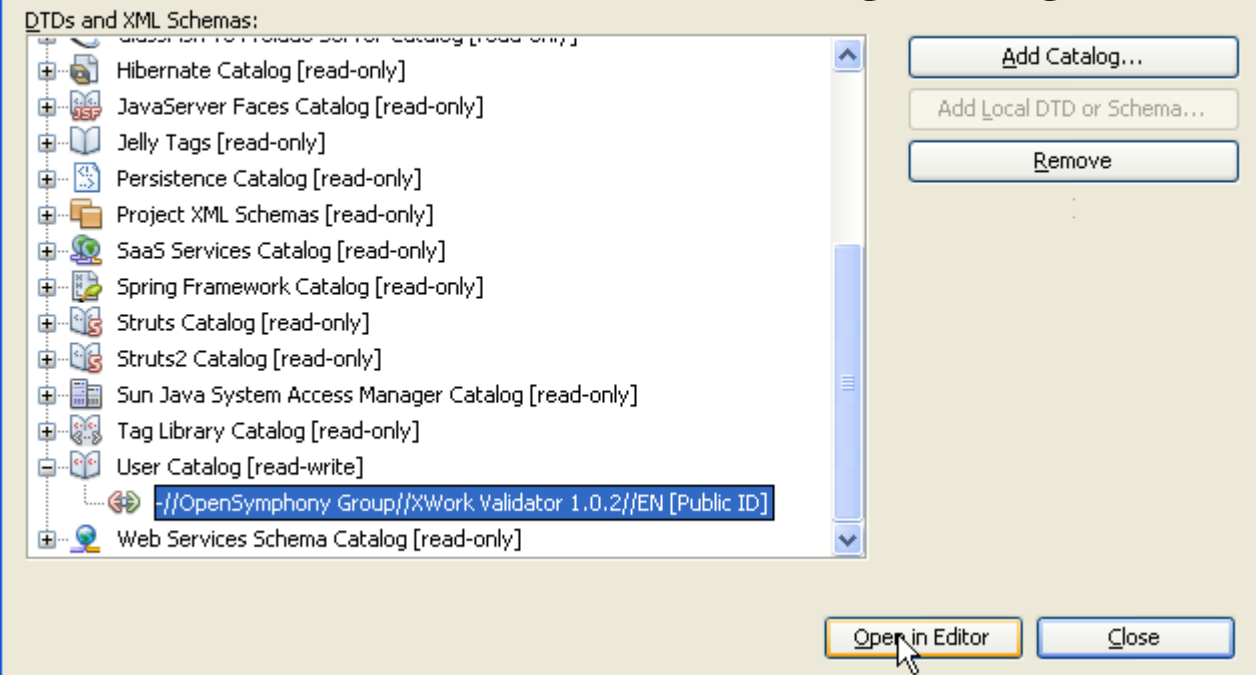- Type or copy Public ID form the file xwork-validation-1.0.3.dtd

Mapping DTD to Netbeans to support visual code XML for configuring validation file
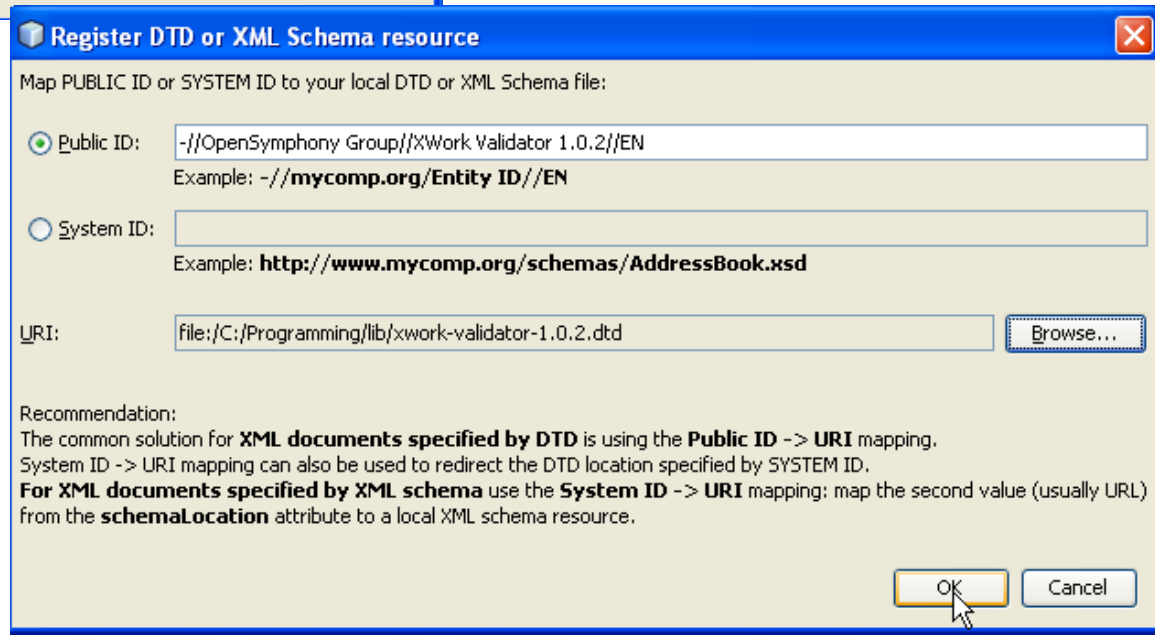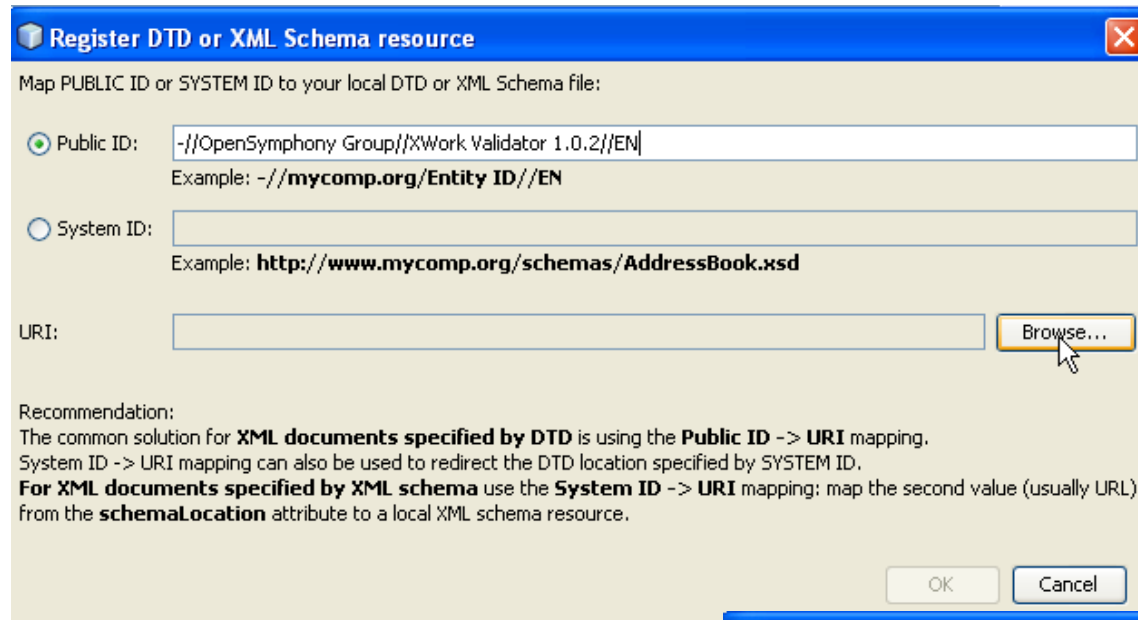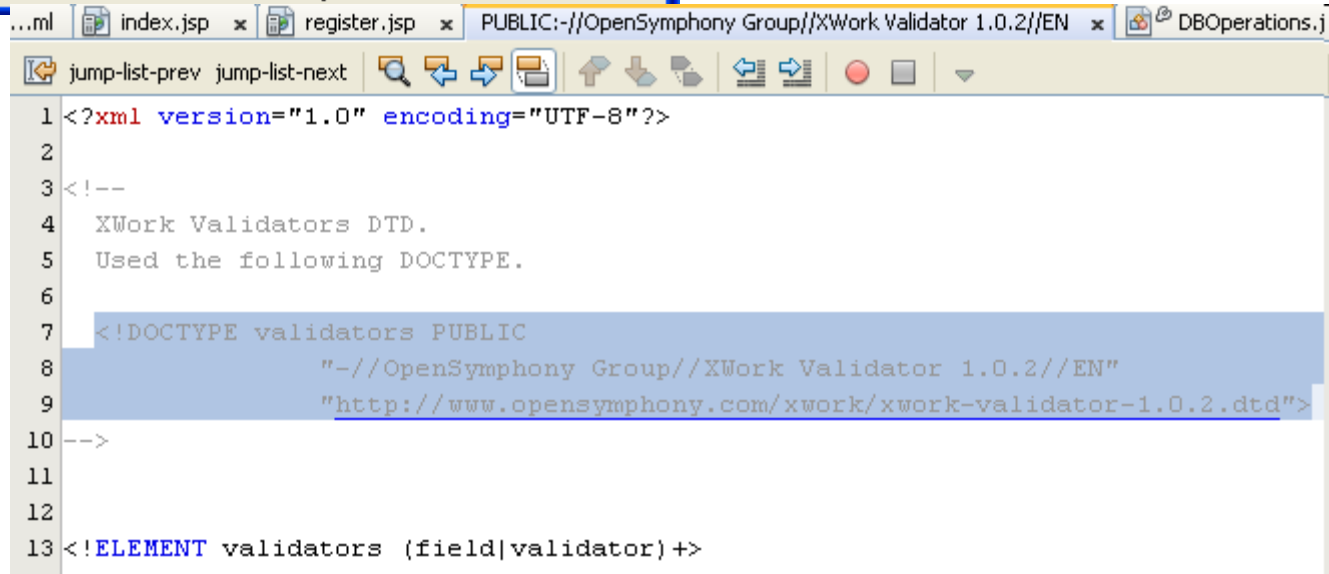

- Click Browse… button from URI to choose the dtd file from hdd to add the Netbeans
- Click OK

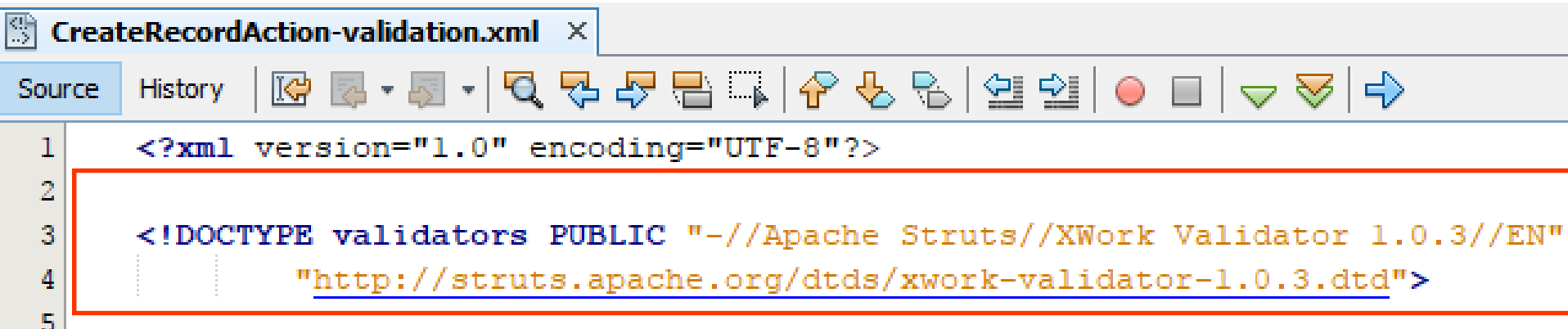# Mapping DTD to Netbeans to support visual code XML for configuring validation file

- Choose added file, then click Open in Editor to open file in Netbeans.

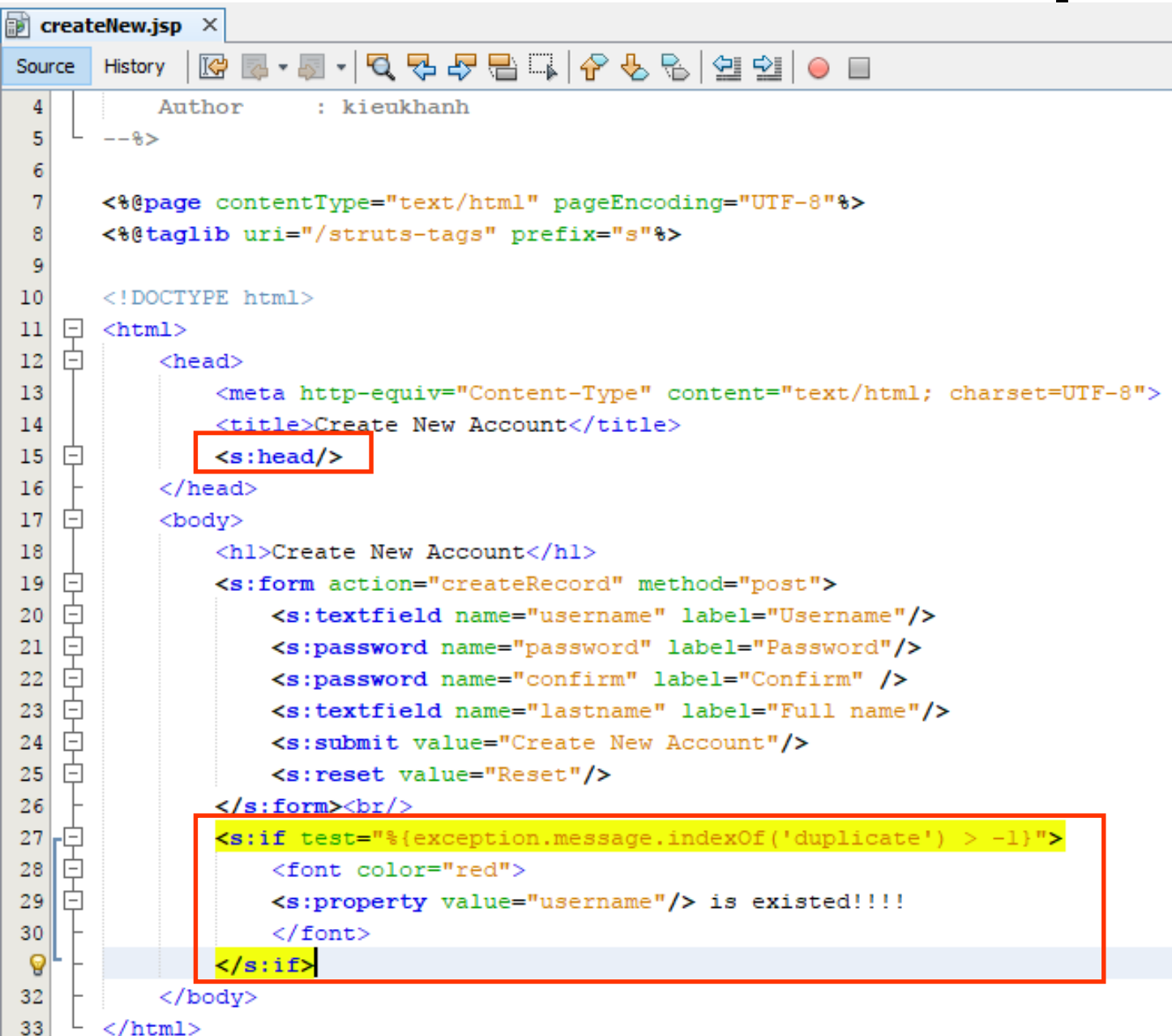- Then, copy all DOCTYPE to pass the validation xml file

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <!--
4     XWork Validators DTD.
5     Used the following DOCTYPE.
6
7     <!DOCTYPE validators PUBLIC
8             "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
9             "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
10  -->
11
12
13  <!ELEMENT validators (field|validator)+>
```

Mapping DTD to Netbeans to support visual code XML for configuring validation file

```
CreateRecordAction-validation.xml  ×

Source   History

1    <?xml version="1.0" encoding="UTF-8"?>
2
3    <!DOCTYPE validators PUBLIC "-//Apache Struts//XWork Validator 1.0.3//EN"
4            "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">
5
```
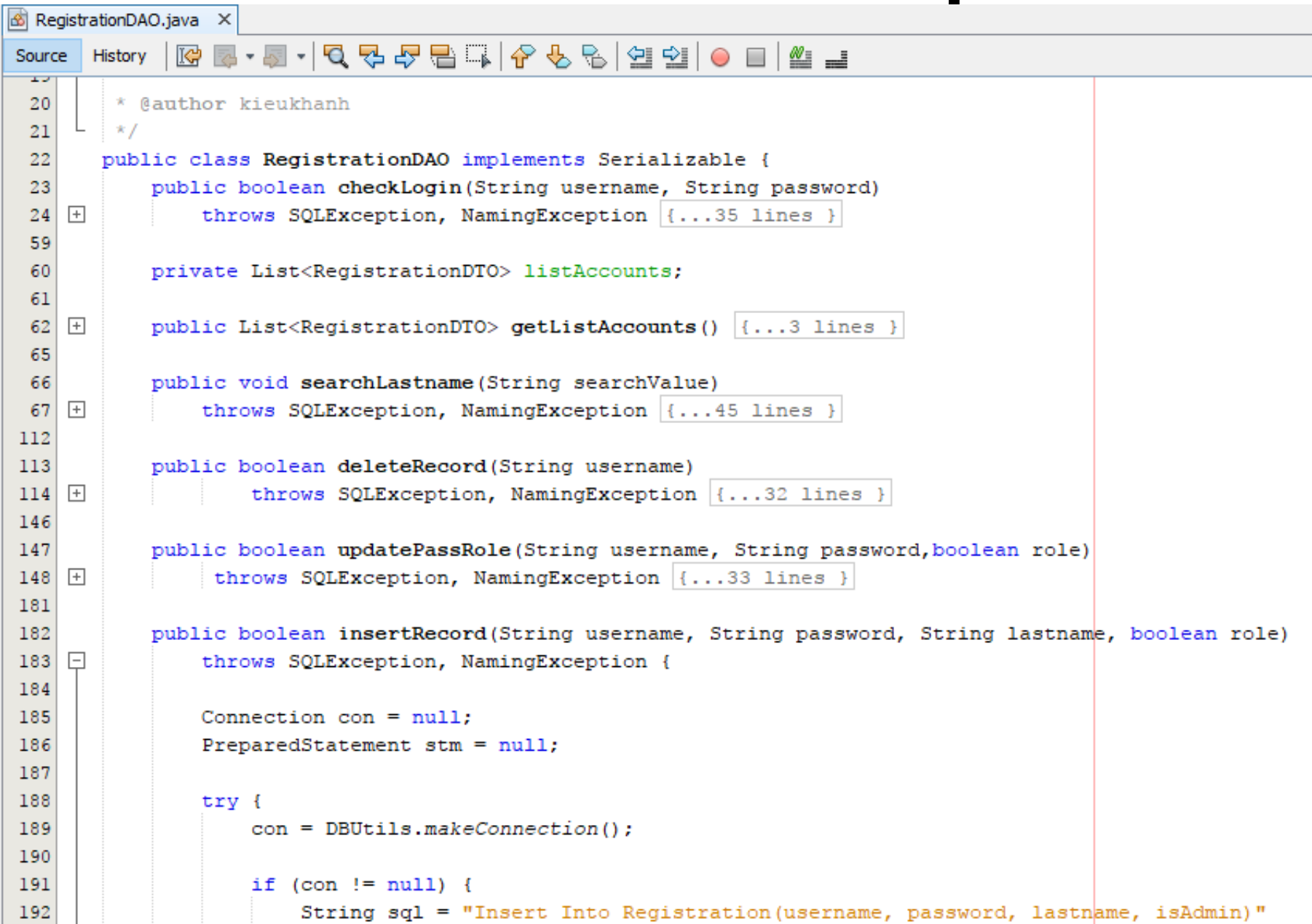
Validations Example

```jsp
     Author     : kieukhanh
   --%>

   <%@page contentType="text/html" pageEncoding="UTF-8"%>
   <%@taglib uri="/struts-tags" prefix="s"%>

   <!DOCTYPE html>
   <html>
       <head>
           <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
           <title>Create New Account</title>
           <s:head/>
       </head>
       <body>
           <h1>Create New Account</h1>
           <s:form action="createRecord" method="post">
               <s:textfield name="username" label="Username"/>
               <s:password name="password" label="Password"/>
               <s:password name="confirm" label="Confirm" />
               <s:textfield name="lastname" label="Full name"/>
               <s:submit value="Create New Account"/>
               <s:reset value="Reset"/>
           </s:form><br/>
           <s:if test="%{exception.message.indexOf('duplicate') > -1}">
               <font color="red">
               <s:property value="username"/> is existed!!!!
               </font>
           </s:if>
       </body>
   </html>
```

Validations Example


```java
    * @author kieukhanh
    */
   public class RegistrationDAO implements Serializable {
       public boolean checkLogin(String username, String password)
           throws SQLException, NamingException {...35 lines }

       private List<RegistrationDTO> listAccounts;

       public List<RegistrationDTO> getListAccounts() {...3 lines }

       public void searchLastname(String searchValue)
           throws SQLException, NamingException {...45 lines }

       public boolean deleteRecord(String username)
               throws SQLException, NamingException {...32 lines }

       public boolean updatePassRole(String username, String password,boolean role)
            throws SQLException, NamingException {...33 lines }

       public boolean insertRecord(String username, String password, String lastname, boolean role)
           throws SQLException, NamingException {

           Connection con = null;
           PreparedStatement stm = null;

           try {
               con = DBUtils.makeConnection();

               if (con != null) {
                   String sql = "Insert Into Registration(username, password, lastname, isAdmin)"
```

Validations Example

```java
192          String sql = "Insert Into Registration(username, password, lastname, isAdmin)"
193                  + " Values(?, ?, ?, ?)";
194
195          stm = con.prepareStatement(sql);
196          stm.setString(1, username);
197          stm.setString(2, password);
198          stm.setString(3, lastname);
199          stm.setBoolean(4, role);
200
201          int row = stm.executeUpdate();
202
203          if (row > 0) {
204              return true;
205          }
206      }
207  } finally {
208      if (stm != null) {
209          stm.close();
210      }
211      if (con != null) {
212          con.close();
213      }
214  }
215
216  return false;
217  }
218
219  }
220
```
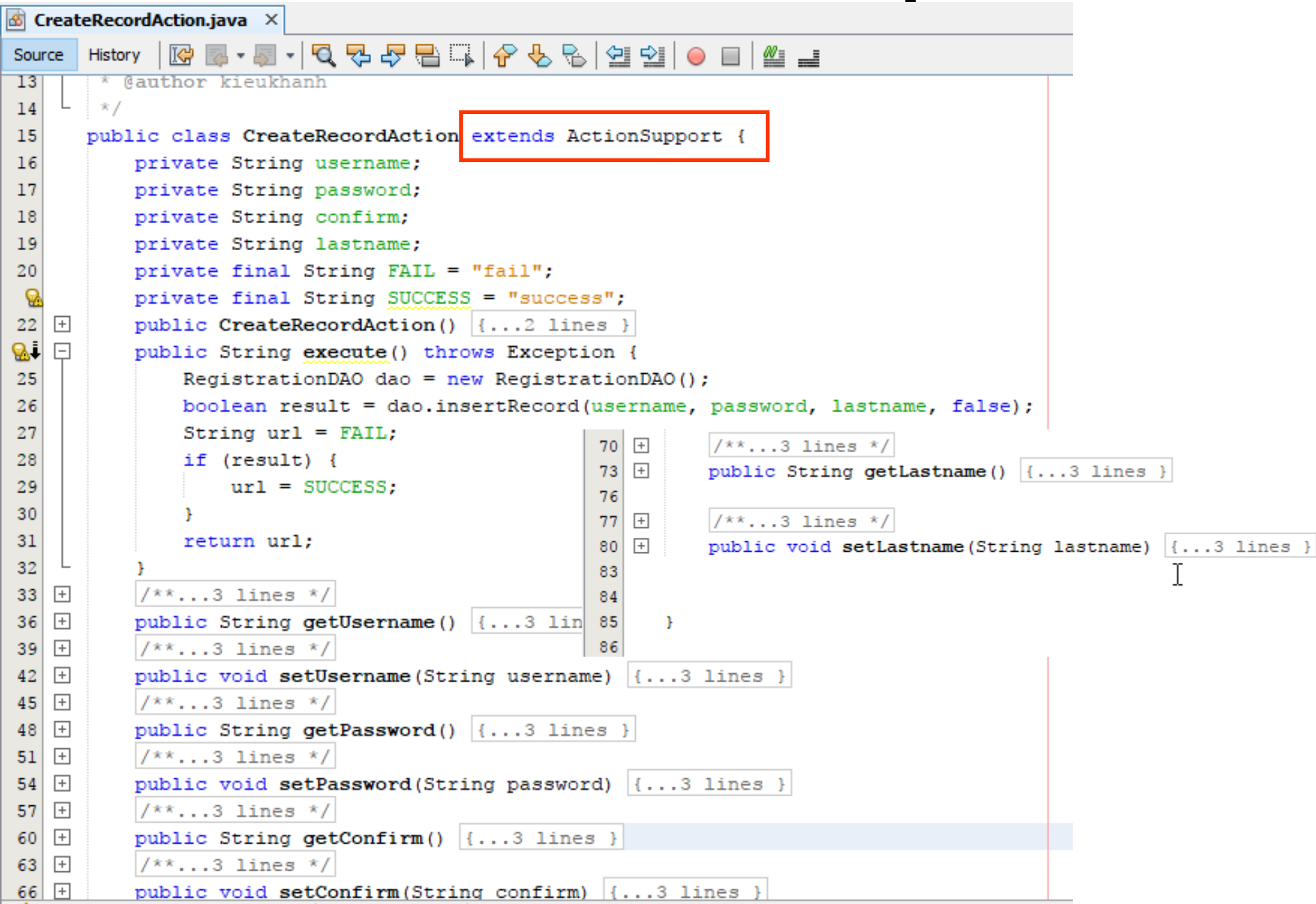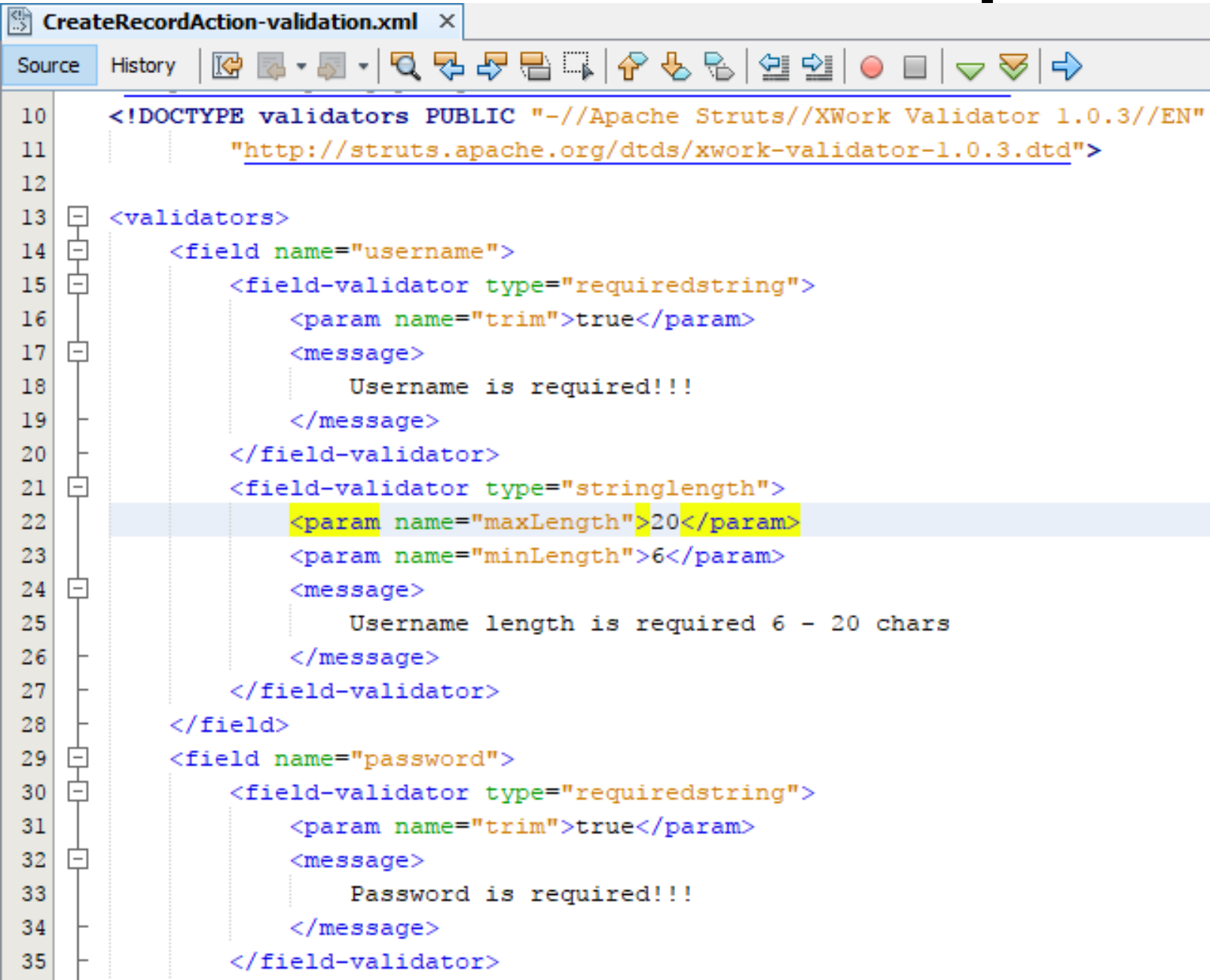
Validations Example

```java
 * @author kieukhanh
 */
public class CreateRecordAction extends ActionSupport {
    private String username;
    private String password;
    private String confirm;
    private String lastname;
    private final String FAIL = "fail";
    private final String SUCCESS = "success";
    public CreateRecordAction() {...2 lines }
    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();
        boolean result = dao.insertRecord(username, password, lastname, false);
        String url = FAIL;
        if (result) {
            url = SUCCESS;
        }
        return url;
    }
    /**...3 lines */
    public String getUsername() {...3 lin
    /**...3 lines */
    public void setUsername(String username) {...3 lines }
    /**...3 lines */
    public String getPassword() {...3 lines }
    /**...3 lines */
    public void setPassword(String password) {...3 lines }
    /**...3 lines */
    public String getConfirm() {...3 lines }
    /**...3 lines */
    public void setConfirm(String confirm) {...3 lines }
```

```java
    /**...3 lines */
    public String getLastname() {...3 lines }

    /**...3 lines */
    public void setLastname(String lastname) {...3 lines }

    }
```

Validations Example

Source   History

```xml
10  <!DOCTYPE validators PUBLIC "-//Apache Struts//XWork Validator 1.0.3//EN"
11          "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">
12
13  <validators>
14      <field name="username">
15          <field-validator type="requiredstring">
16              <param name="trim">true</param>
17              <message>
18                  Username is required!!!
19              </message>
20          </field-validator>
21          <field-validator type="stringlength">
22              <param name="maxLength">20</param>
23              <param name="minLength">6</param>
24              <message>
25                  Username length is required 6 - 20 chars
26              </message>
27          </field-validator>
28      </field>
29      <field name="password">
30          <field-validator type="requiredstring">
31              <param name="trim">true</param>
32              <message>
33                  Password is required!!!
34              </message>
35          </field-validator>
```

Validations Example

```xml
36        <field-validator type="stringlength">
37            <param name="maxLength">30</param>
38            <param name="minLength">6</param>
39            <message>
40                Password length is required 6 - 30 chars
41            </message>
42        </field-validator>
43    </field>
44    <field name="confirm">
45        <field-validator type="fieldexpression">
46            <param name="expression">confirm==password</param>
47            <message>
48                Confirm must match password!!!
49            </message>
50        </field-validator>
51    </field>
52    <field name="lastname">
53        <field-validator type="requiredstring">
54            <param name="trim">true</param>
55            <message>
56                Lastname is required!!!
57            </message>
58        </field-validator>
59        <field-validator type="stringlength">
60            <param name="maxLength">50</param>
61            <param name="minLength">2</param>
62            <message>
63                Lastname length is required 2 - 50 chars
64            </message>
65        </field-validator>
66    </field>
67 </validators>
```

Validations Example

```
1   <!DOCTYPE struts PUBLIC
2   "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
3   "http://struts.apache.org/dtds/struts-2.0.dtd">
4
5   <struts>
6       <!-- Configuration for the default package. -->
7       <package name="default" extends="struts-default" namespace="/">
8           <action>
11          <action name="try">
12              <result>login.html</result>
13          </action>
14          <action>
18          <action>
21          <action>
28          <action>
35          <action name="createRecord" class="sample.struts2.CreateRecordAction">
36              <exception-mapping exception="java.sql.SQLException" result="input"/>
37              <result name="success">login.html</result>
38              <result name="fail">insertError.html</result>
39              <result name="input">createNew.jsp</result>
40          </action>
41          <action name="signUp">
42              <result>createNew.jsp</result>
43          </action>
44      </package>
45  </struts>
```

Validations Example


```html
<!DOCTYPE html>
...5 lines
<html>
    <head>
        <title>Errors</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <h1>
            <font color="red">
                Errors occur in progress!!!!
            </font>
        </h1>
    </body>
</html>
```
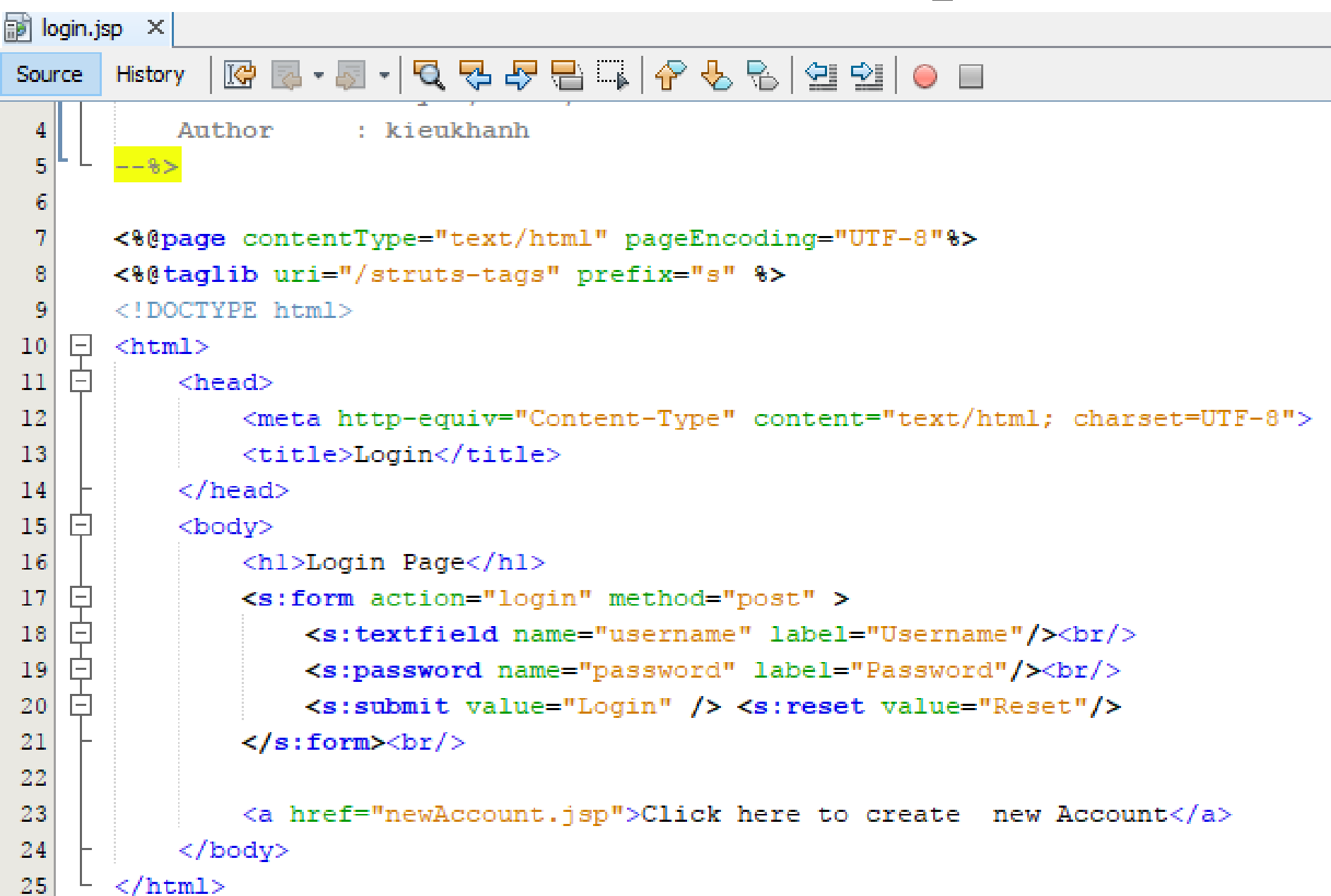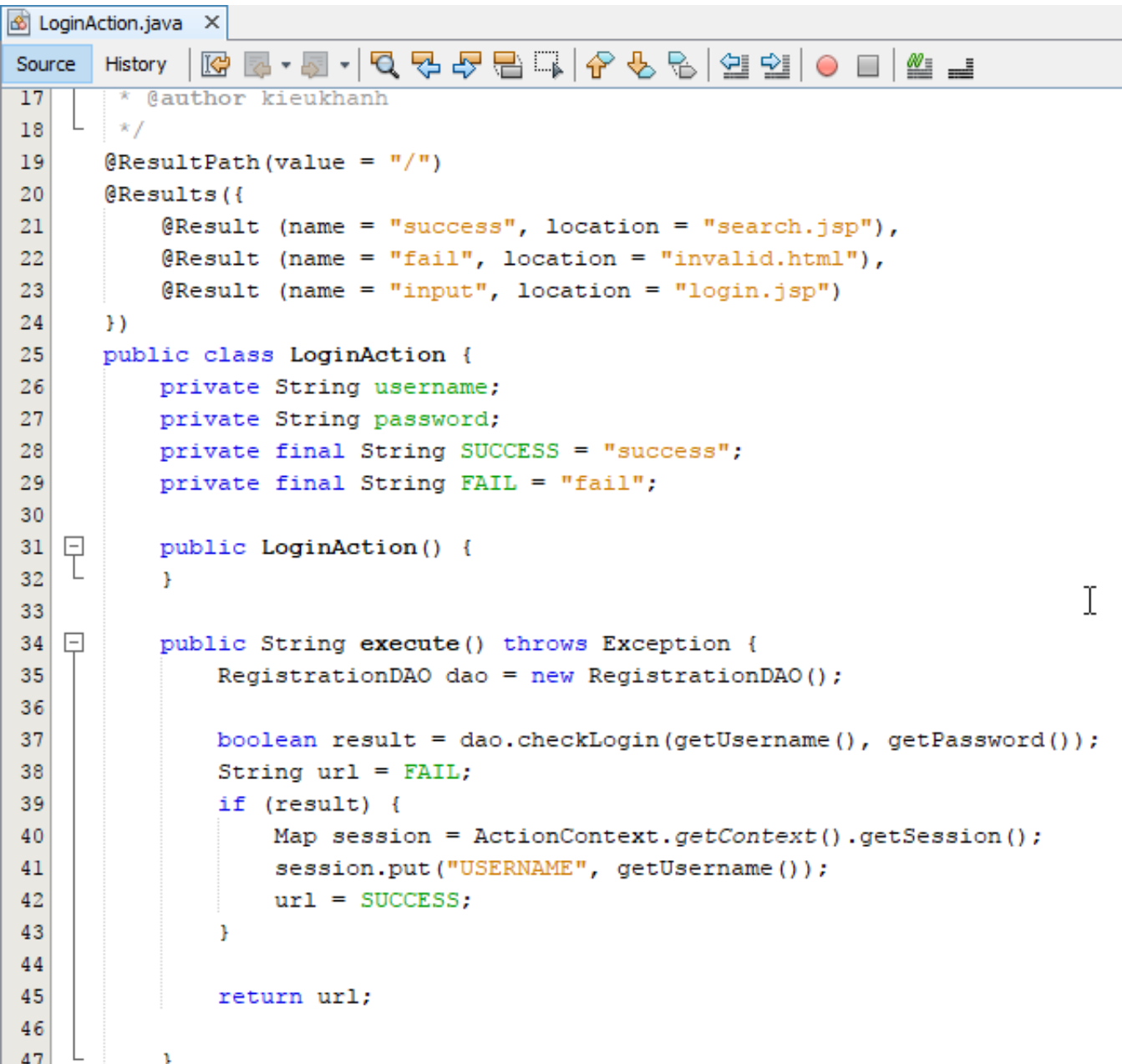
Source Packages
  <default package>
    struts.xml
  sample.registration
    RegistrationDAO.java
    RegistrationDTO.java
  sample.struts2
    CreateRecordAction-validation.xml
    CreateRecordAction.java

Annotations Example

```jsp
 4        Author      : kieukhanh
 5      --%>
 6
 7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
 8      <%@taglib uri="/struts-tags" prefix="s" %>
 9      <!DOCTYPE html>
10      <html>
11          <head>
12              <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13              <title>Login</title>
14          </head>
15          <body>
16              <h1>Login Page</h1>
17              <s:form action="login" method="post" >
18                  <s:textfield name="username" label="Username"/><br/>
19                  <s:password name="password" label="Password"/><br/>
20                  <s:submit value="Login" /> <s:reset value="Reset"/>
21              </s:form><br/>
22
23              <a href="newAccount.jsp">Click here to create  new Account</a>
24          </body>
25      </html>
```

Annotations

```java
   17      * @author kieukhanh
   18      */
   19     @ResultPath(value = "/")
   20     @Results({
   21         @Result (name = "success", location = "search.jsp"),
   22         @Result (name = "fail", location = "invalid.html"),
   23         @Result (name = "input", location = "login.jsp")
   24     })
   25     public class LoginAction {
   26         private String username;
   27         private String password;
   28         private final String SUCCESS = "success";
   29         private final String FAIL = "fail";
   30
   31         public LoginAction() {
   32         }
   33
   34         public String execute() throws Exception {
   35             RegistrationDAO dao = new RegistrationDAO();
   36
   37             boolean result = dao.checkLogin(getUsername(), getPassword());
   38             String url = FAIL;
   39             if (result) {
   40                 Map session = ActionContext.getContext().getSession();
   41                 session.put("USERNAME", getUsername());
   42                 url = SUCCESS;
   43             }
   44
   45             return url;
   46
   47         }
```

Annotations

```
invalid.html  ×
Source  History  [toolbar icons]

 1    <!DOCTYPE html>
 2    <!--
 3    To change this license header, choose License Headers in Project Properti
 4    To change this template file, choose Tools | Templates
 5    and open the template in the editor.
 6    -->
 7    <html>
 8        <head ...5 lines />
13        <body>
14            <h1>
 ♀               <font color="red">
16                   Invalid username or password!!!
17               </font>
18           </h1>
19
20           <a href="newAccount.jsp">Click here to create  new Account</a>
21       </body>
22   </html>
```

```
search.jsp  ×
Source  History  [toolbar icons]

 4        Author       : kieukhanh
 5        --%>
 6
 7    <%@page contentType="text/html" pageEncoding="UTF-8"%>
 8    <%@taglib uri="/struts-tags" prefix="s"%>
 9    <!DOCTYPE html>
10    <html>
11        <head>
12            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13            <title>Search</title>
14        </head>
15        <body>
16            <font color ="red">
17            Welcome, <s:property value="%{#session.USERNAME}"/>
18            </font>
19            <h1>Search Page</h1>
20            <s:form action="search">
21                <s:textfield name="searchValue" label="Search Value"/><br/>
22                <s:submit value="Search"/>
23            </s:form>
24            <br/>
```
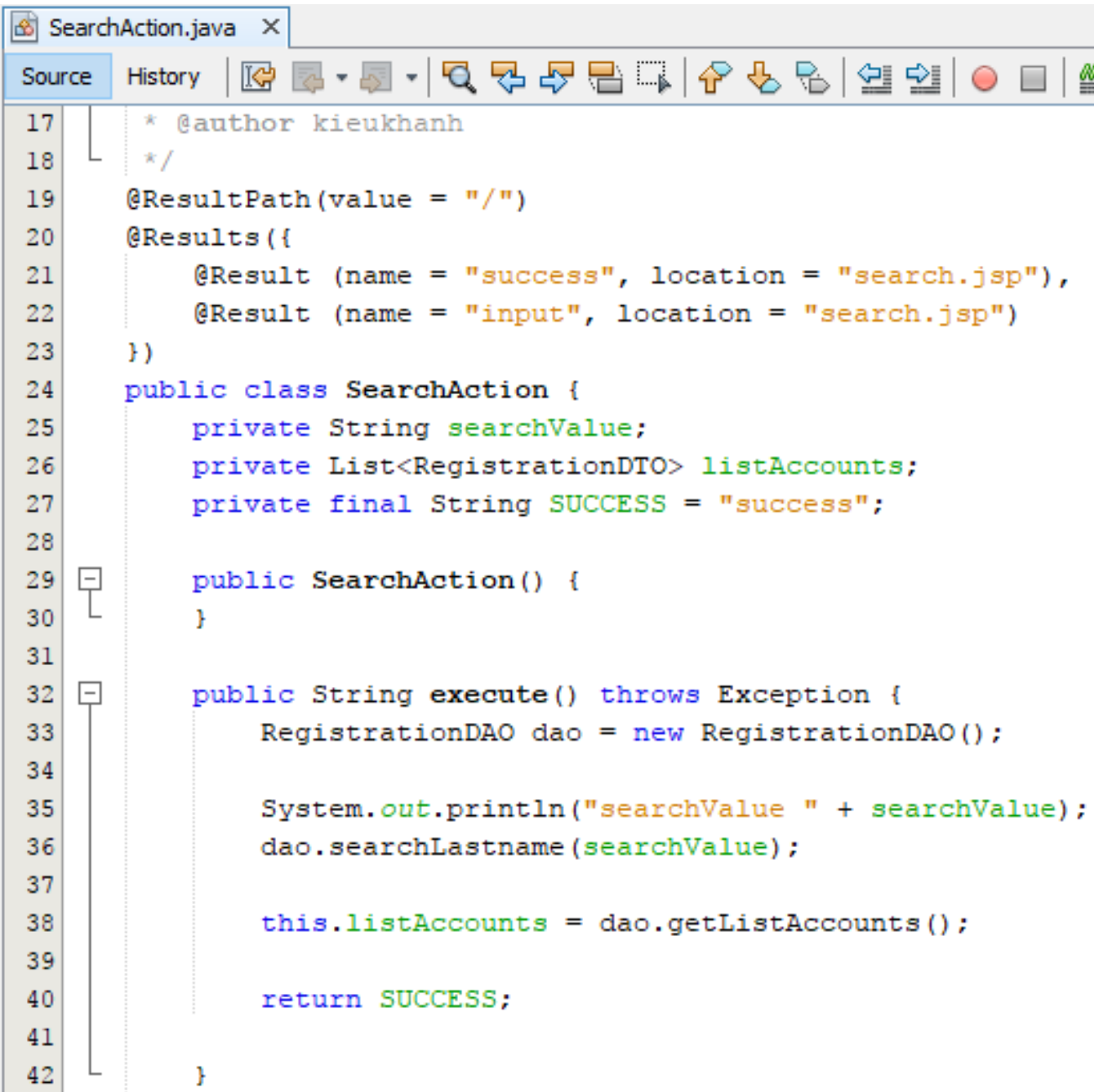
Annotations


```
web.xml  ✕

Source    General    Servlets    Filters    Pages    References    Security    History

 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://w
 3      <filter>
 4          <filter-name>struts2</filter-name>
 5          <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
 6          <init-param>
 7              <param-name>actionPackages</param-name>
 8              <param-value>sample.struts</param-value>
 9          </init-param>
10      </filter>
11      <filter-mapping>
12          <filter-name>struts2</filter-name>
13          <url-pattern>/*</url-pattern>
14      </filter-mapping>
15      <session-config>
20      <welcome-file-list>
23      <resource-ref>
29  </web-app>
```
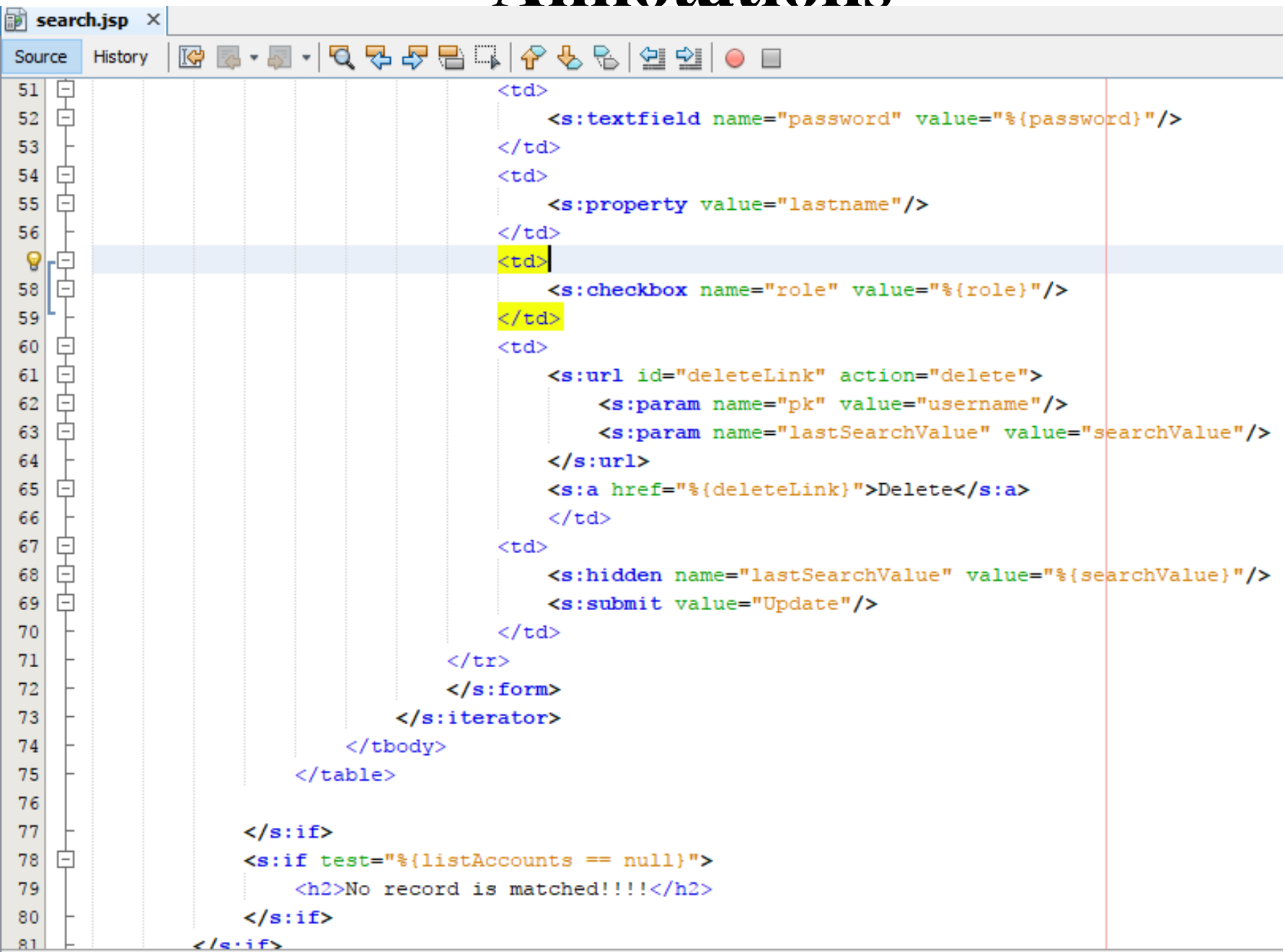
Annotations

```
17      * @author kieukhanh
18      */
19     @ResultPath(value = "/")
20     @Results({
21         @Result (name = "success", location = "search.jsp"),
22         @Result (name = "input", location = "search.jsp")
23     })
24     public class SearchAction {
25         private String searchValue;
26         private List<RegistrationDTO> listAccounts;
27         private final String SUCCESS = "success";
28
29         public SearchAction() {
30         }
31
32         public String execute() throws Exception {
33             RegistrationDAO dao = new RegistrationDAO();
34
35             System.out.println("searchValue " + searchValue);
36             dao.searchLastname(searchValue);
37
38             this.listAccounts = dao.getListAccounts();
39
40             return SUCCESS;
41
42         }
```

Annotations

```
5     --%>
```

```
23          </s:form>
24          <br/>
25          <s:if test="%{searchValue != '' and searchValue != null}">
26              <s:if test="%{listAccounts != null}">
27                  Result of Search <br/>
28                  <table border="1">
29                      <thead>
30                          <tr>
31                              <th>No.</th>
32                              <th>Username</th>
33                              <th>Password</th>
34                              <th>Last name</th>
35                              <th>Role</th>
36                              <th>Delete</th>
37                              <th>Update</th>
38                          </tr>
39                      </thead>
40                      <tbody>
41                          <s:iterator value="listAccounts" status="dto">
42                              <s:form action="update" theme="simple">
43                                  <tr>
44                                      <td>
45                                          <s:property value="%{#dto.count}"/>
46                                          .</td>
47                                      <td>
48                                          <s:property value="username"/>
49                                          <s:hidden name="username" value="%{username}"/>
50                                      </td>
```
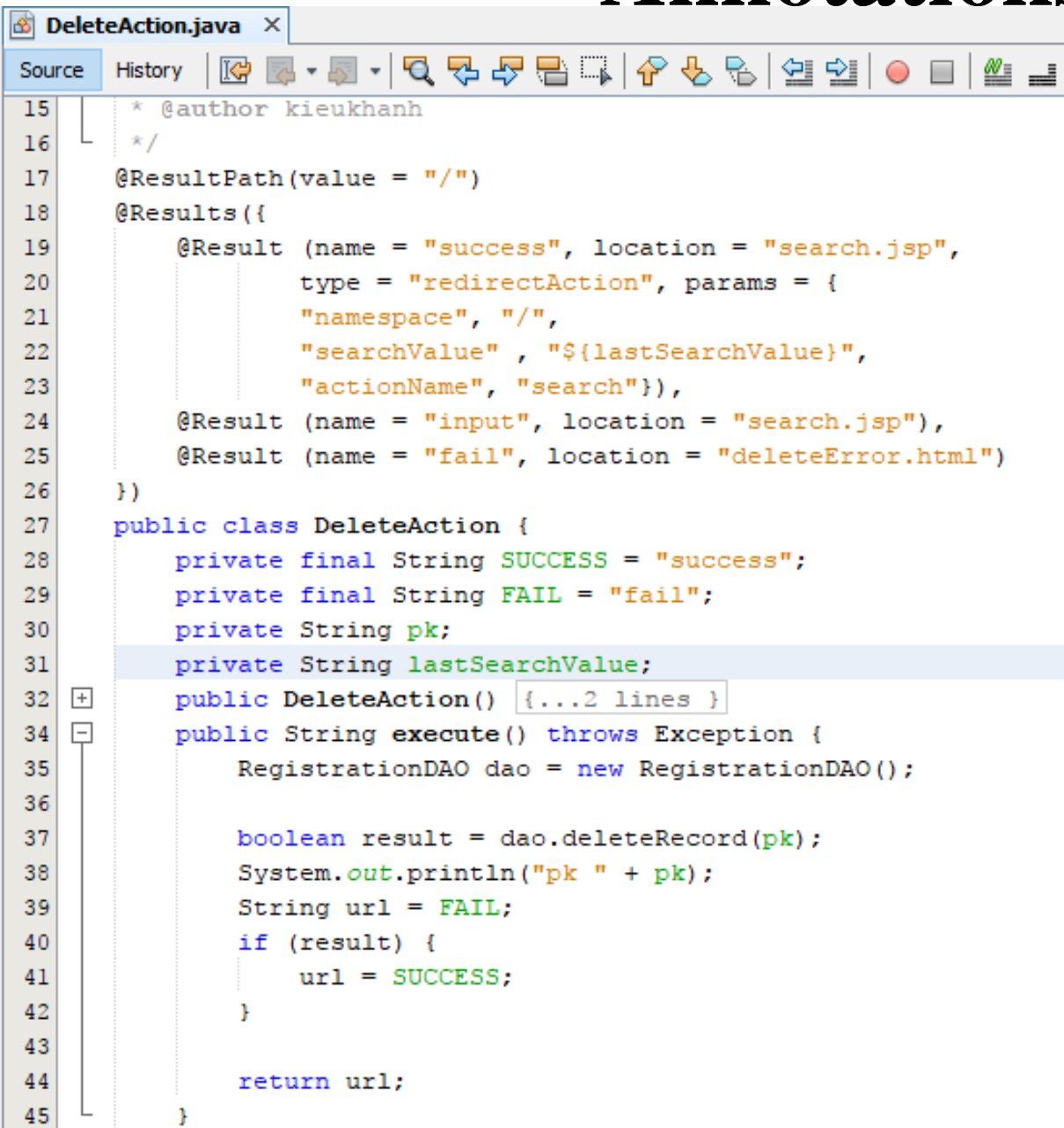
Annotations

Source | History

```
51        <td>
52            <s:textfield name="password" value="%{password}"/>
53        </td>
54        <td>
55            <s:property value="lastname"/>
56        </td>
57        <td>
58            <s:checkbox name="role" value="%{role}"/>
59        </td>
60        <td>
61            <s:url id="deleteLink" action="delete">
62                <s:param name="pk" value="username"/>
63                <s:param name="lastSearchValue" value="searchValue"/>
64            </s:url>
65            <s:a href="%{deleteLink}">Delete</s:a>
66            </td>
67        <td>
68            <s:hidden name="lastSearchValue" value="%{searchValue}"/>
69            <s:submit value="Update"/>
70        </td>
71                    </tr>
72                </s:form>
73            </s:iterator>
74        </tbody>
75    </table>
76
77    </s:if>
78    <s:if test="%{listAccounts == null}">
79        <h2>No record is matched!!!!</h2>
80    </s:if>
81    </s:if>
```
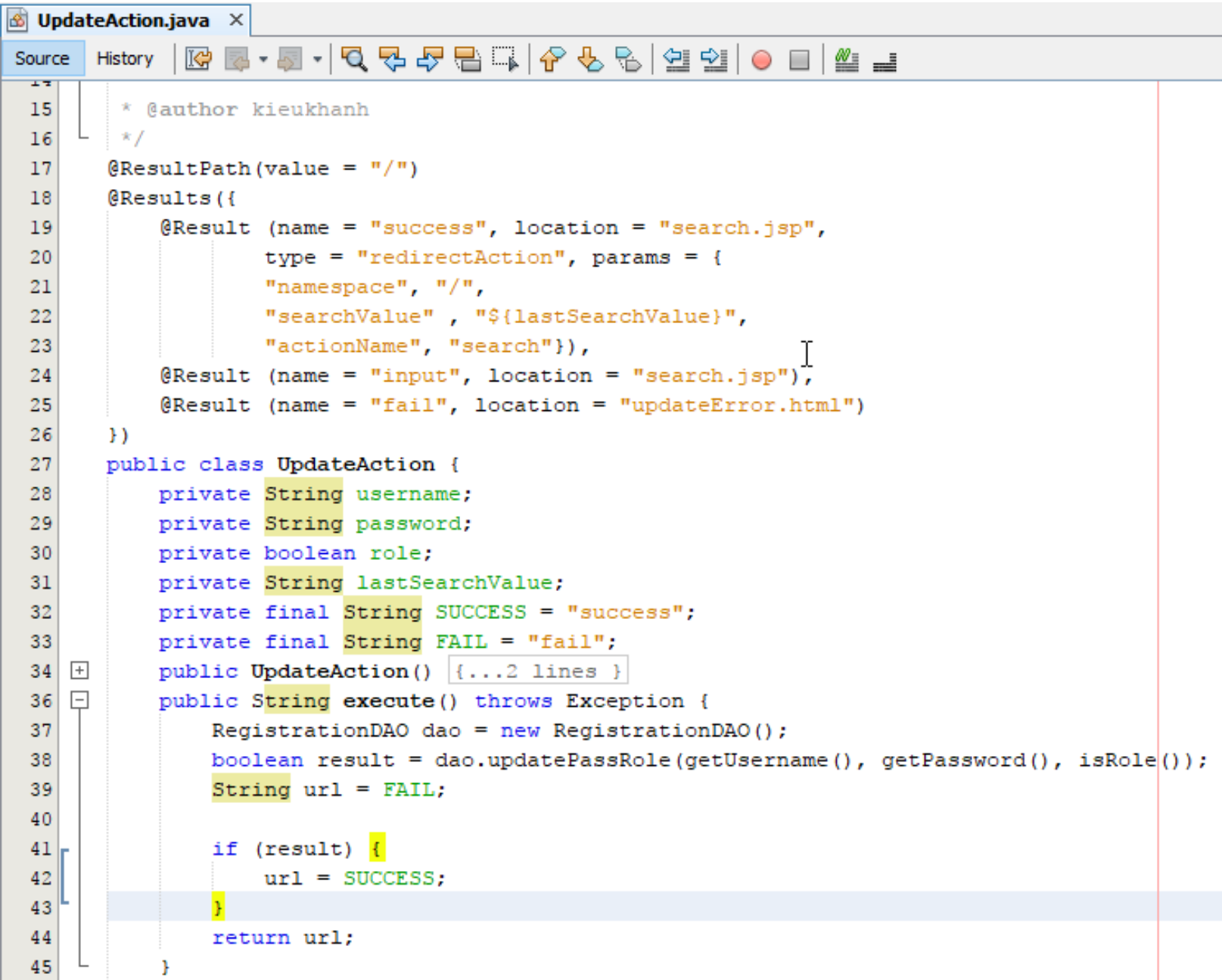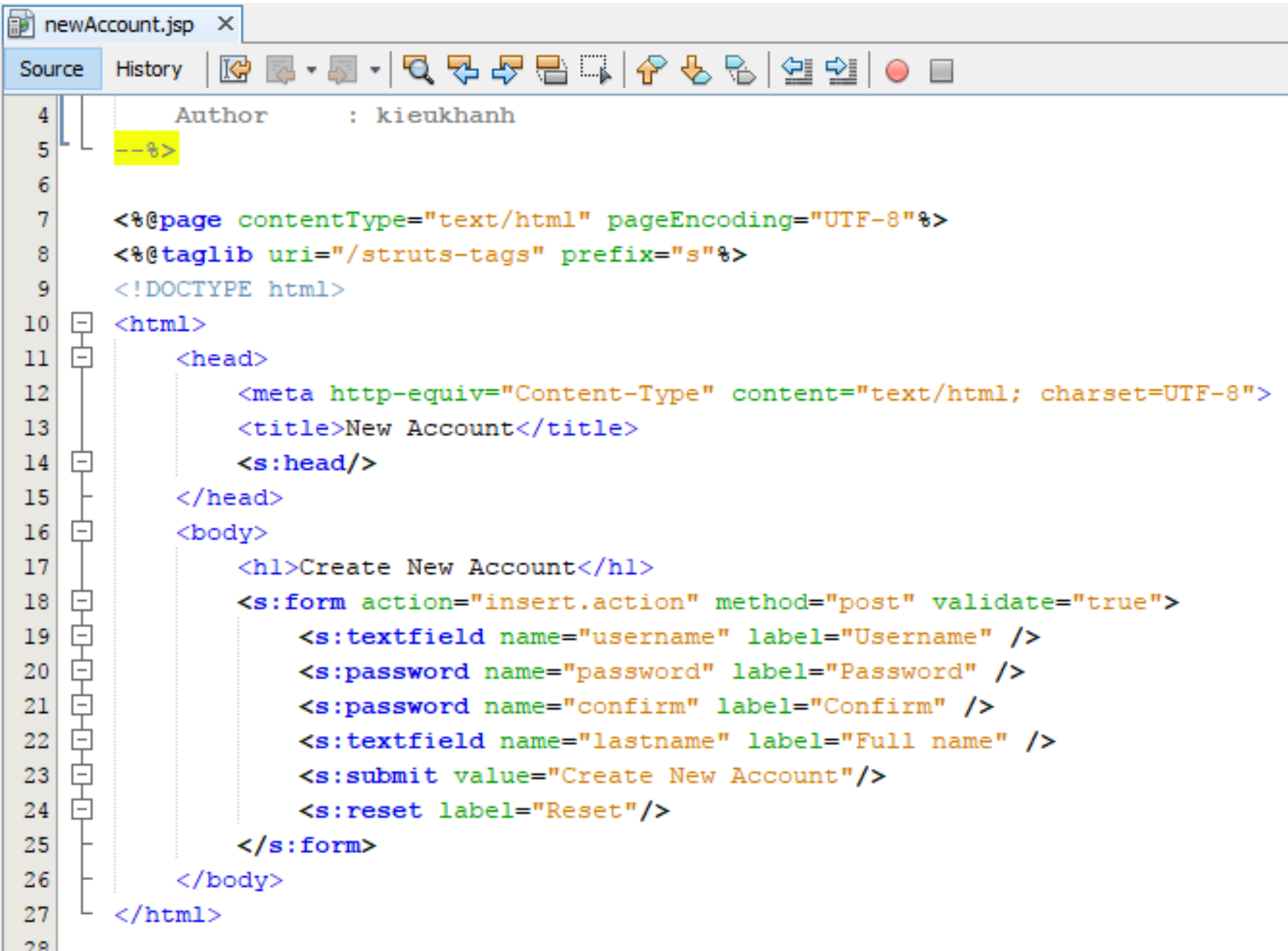
Annotations

```
DeleteAction.java ×

Source  History

15      * @author kieukhanh
16      */
17     @ResultPath(value = "/")
18     @Results({
19         @Result (name = "success", location = "search.jsp",
20                 type = "redirectAction", params = {
21                 "namespace", "/",
22                 "searchValue" , "${lastSearchValue}",
23                 "actionName", "search"}),
24         @Result (name = "input", location = "search.jsp"),
25         @Result (name = "fail", location = "deleteError.html")
26     })
27     public class DeleteAction {
28         private final String SUCCESS = "success";
29         private final String FAIL = "fail";
30         private String pk;
31         private String lastSearchValue;
32         public DeleteAction() {...2 lines }
34         public String execute() throws Exception {
35             RegistrationDAO dao = new RegistrationDAO();
36
37             boolean result = dao.deleteRecord(pk);
38             System.out.println("pk " + pk);
39             String url = FAIL;
40             if (result) {
41                 url = SUCCESS;
42             }
43
44             return url;
45         }
```

Annotations

```java
 *  @author kieukhanh
 */
@ResultPath(value = "/")
@Results({
    @Result (name = "success", location = "search.jsp",
             type = "redirectAction", params = {
             "namespace", "/",
             "searchValue" , "${lastSearchValue}",
             "actionName", "search"}),
    @Result (name = "input", location = "search.jsp"),
    @Result (name = "fail", location = "updateError.html")
})
public class UpdateAction {
    private String username;
    private String password;
    private boolean role;
    private String lastSearchValue;
    private final String SUCCESS = "success";
    private final String FAIL = "fail";
    public UpdateAction() {...2 lines }
    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();
        boolean result = dao.updatePassRole(getUsername(), getPassword(), isRole());
        String url = FAIL;

        if (result) {
            url = SUCCESS;
        }
        return url;
    }
```
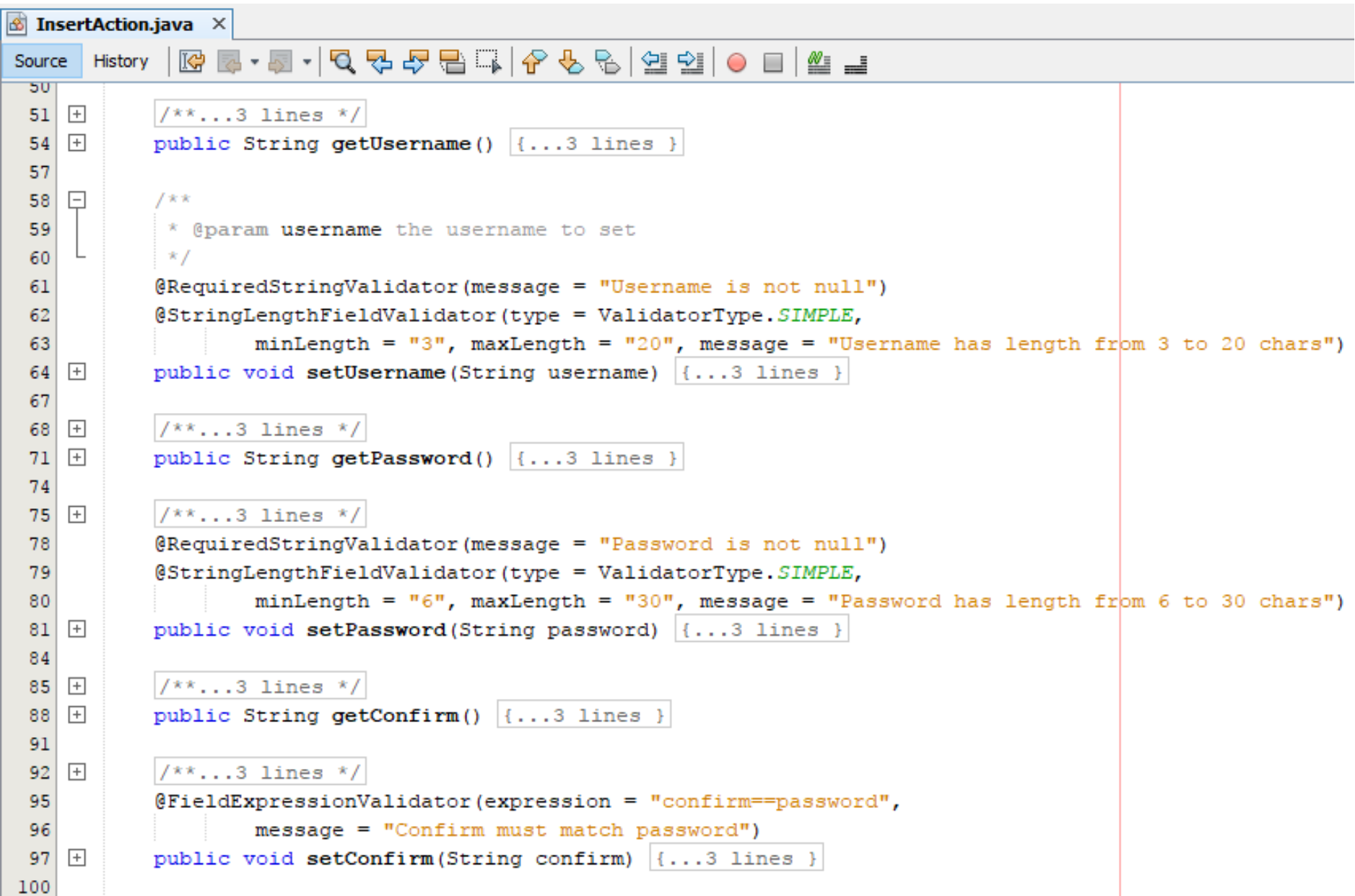
Annotations

```
     newAccount.jsp   ×

  Source    History    [toolbar icons]

 4 |      Author        : kieukhanh
 5 |      --%>
 6 |
 7 |    <%@page contentType="text/html" pageEncoding="UTF-8"%>
 8 |    <%@taglib uri="/struts-tags" prefix="s"%>
 9 |    <!DOCTYPE html>
10 |    <html>
11 |        <head>
12 |            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 |            <title>New Account</title>
14 |            <s:head/>
15 |        </head>
16 |        <body>
17 |            <h1>Create New Account</h1>
18 |            <s:form action="insert.action" method="post" validate="true">
19 |                <s:textfield name="username" label="Username" />
20 |                <s:password name="password" label="Password" />
21 |                <s:password name="confirm" label="Confirm" />
22 |                <s:textfield name="lastname" label="Full name" />
23 |                <s:submit value="Create New Account"/>
24 |                <s:reset label="Reset"/>
25 |            </s:form>
26 |        </body>
27 |    </html>
28
```

Annotations

```java
 *
 * @author kieukhanh
 */
@ResultPath(value = "/")
@Results({
    @Result (name = "success", location = "login.jsp"),
    @Result (name = "fail", location = "newAccount.jsp"),
    @Result (name = "input", location = "newAccount.jsp")
})
public class InsertAction extends ActionSupport{
    private String username;
    private String password;
    private String confirm;
    private String lastname;

    public InsertAction() {...2 lines }

    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();

        boolean result = dao.insertRecord(getUsername(), getPassword(), getLastname(), false);

        String url = "fail";

        if (result) {
            url = "success";
        }

        return url;
    }
```

Annotations

```java
 50
 51 ⊞     /**...3 lines */
 54 ⊞     public String getUsername() {...3 lines }
 57
 58 ⊟     /**
 59      * @param username the username to set
 60      */
 61     @RequiredStringValidator(message = "Username is not null")
 62     @StringLengthFieldValidator(type = ValidatorType.SIMPLE,
 63             minLength = "3", maxLength = "20", message = "Username has length from 3 to 20 chars")
 64 ⊞     public void setUsername(String username) {...3 lines }
 67
 68 ⊞     /**...3 lines */
 71 ⊞     public String getPassword() {...3 lines }
 74
 75 ⊞     /**...3 lines */
 78     @RequiredStringValidator(message = "Password is not null")
 79     @StringLengthFieldValidator(type = ValidatorType.SIMPLE,
 80             minLength = "6", maxLength = "30", message = "Password has length from 6 to 30 chars")
 81 ⊞     public void setPassword(String password) {...3 lines }
 84
 85 ⊞     /**...3 lines */
 88 ⊞     public String getConfirm() {...3 lines }
 91
 92 ⊞     /**...3 lines */
 95     @FieldExpressionValidator(expression = "confirm==password",
 96             message = "Confirm must match password")
 97 ⊞     public void setConfirm(String confirm) {...3 lines }
100
```
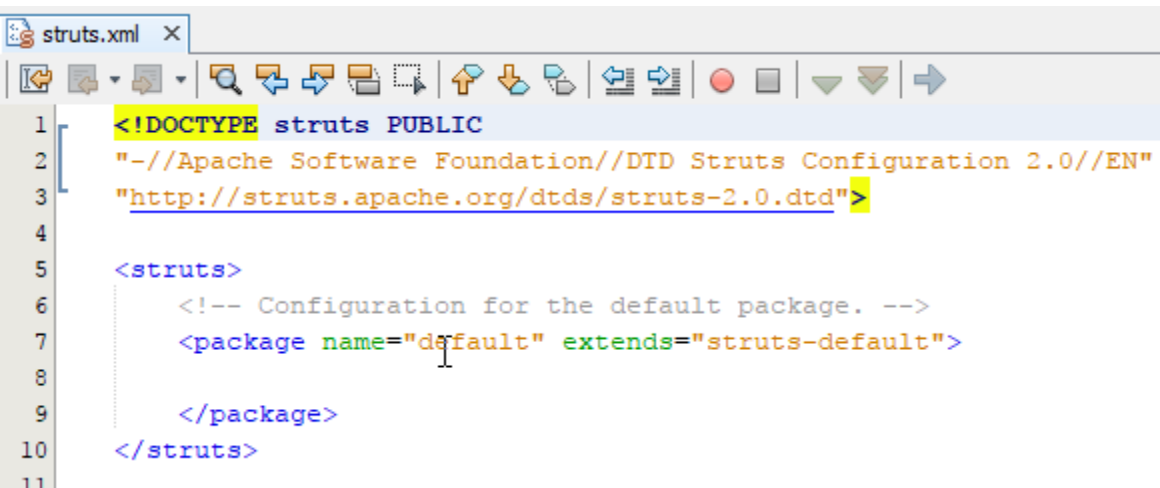
Annotations

Source | History

```
85    /**...3 lines */
88    public String getConfirm()  {...3 lines }
91

92    /**...3 lines */
95    @FieldExpressionValidator(expression = "confirm==password",
96              message = "Confirm must match password")
97    public void setConfirm(String confirm)  {...3 lines }
100

101   /**...3 lines */
104   public String getLastname()  {...3 lines }
107

108   /**...3 lines */
111   @RequiredStringValidator(message = "Full name is not null")
112   @StringLengthFieldValidator(type = ValidatorType.SIMPLE,
113           minLength = "2", maxLength = "50", message = "Full name has length from 6 to 30 chars")
114   public void setLastname(String lastname)  {...3 lines }
117

118

119   }
120
```

Annotations