Struts

Struts 2 Framework
Basic

#Struts #Struts2 #FilterDispatcher

#OGNL #valueStack

Review

struts-config.xml

Browser

Html/jsp page

ActionServlet

Controler

Action

1. Request

2. match action description

5. invoke Action (dispatches)

6. match description

10. Response

Controller

3. Create FormBean with parameters

8. forward

7". Call without ActionFormBean

9. Create View to display info

Html/jsp pages

7. Call

View

ActionFormBean

4. Map

Validation (if any)

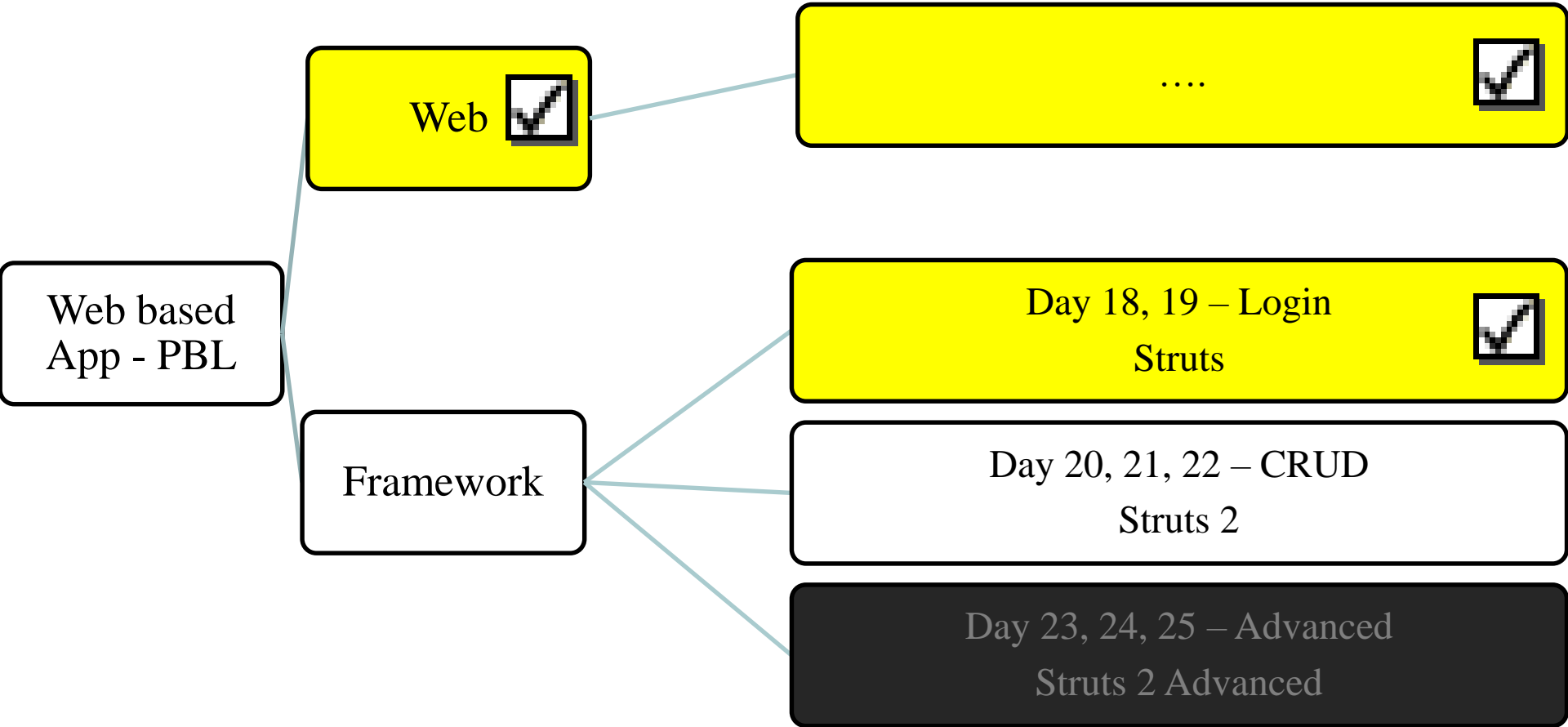ActionForm

Model

7"'. Access

DB

7'. Call

DAO

Objectives

- **How to build the web application using Struts 2 Framework?**
  - Struts 2 (**Architecture & Mechanism**)
  - Struts 2 **Components** (Filter Dispatcher, Action, Struts Configuration File, Result & Result Type)
  - **How to access** action's **properties** and **scope variable**? (Using Value Stack, Action Context, OGNL)
  - Struts 2 **Tag**

Objectives

STRUTS 2
Requirement

- Building the web application with CRUD functional combining with DB using Struts 2 framework

STRUTS 2
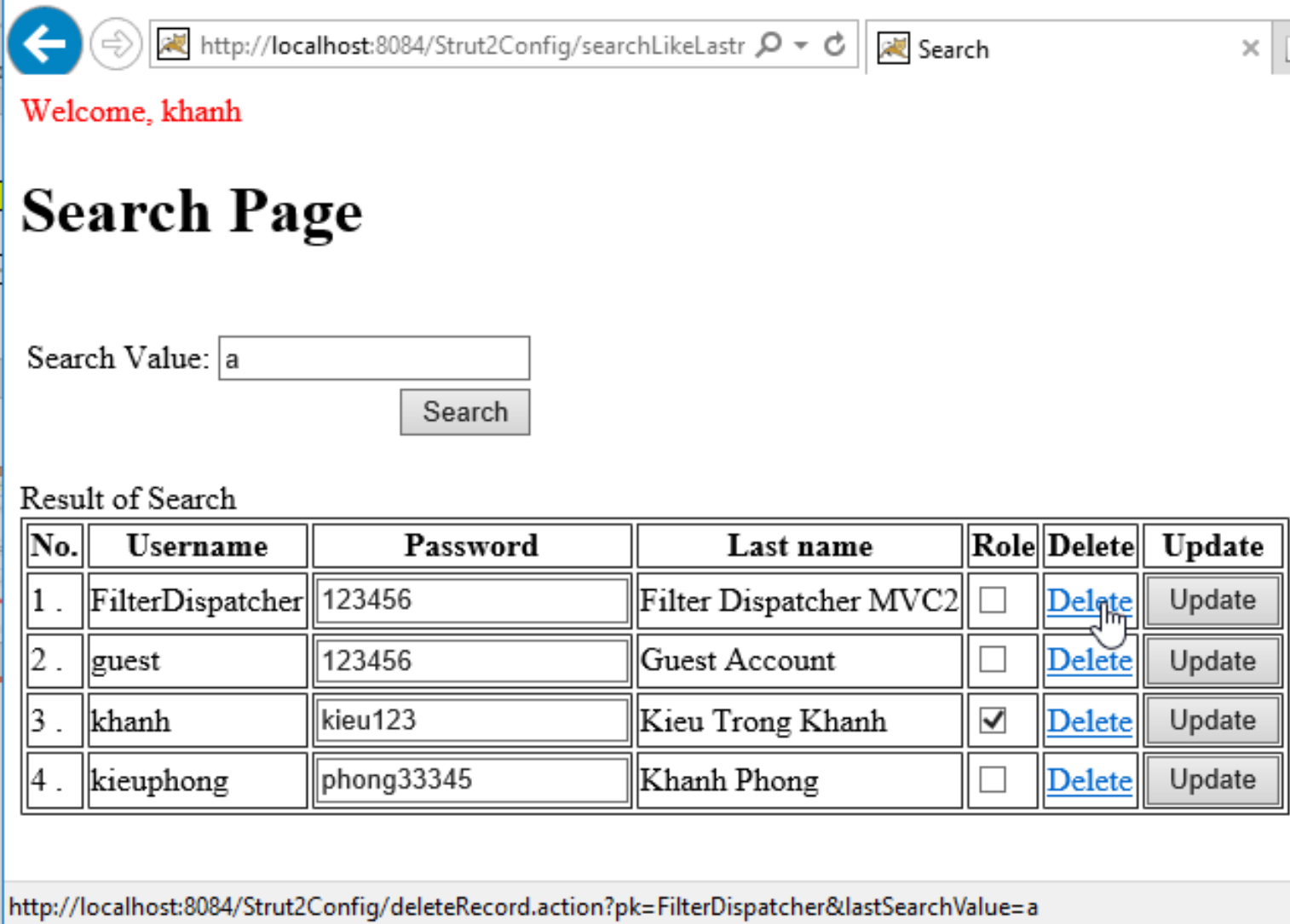## Expectation

Welcome, khanh

Search Page

Search Value: a

Search

Result of Search

No.	Username	Password	Last name	Role	Delete	Update
1 .	FilterDispatcher	123456	Filter Dispatcher MVC2	☐	Delete	Update
2 .	guest	123456	Guest Account	☐	Delete	Update
3 .	khanh	kieu123	Kieu Trong Khanh	☑	Delete	Update
4 .	kieuphong	phong33345	Khanh Phong	☐	Delete	Update

STRUTS 2
Expectation

Welcome, khanh

Search Page

Search Value: [a]

[Search]

Result of Search

No.	Username	Password	Last name	Role	Delete	Update
1 .	FilterDispatcher	123456	Filter Dispatcher MVC2	☐	Delete	Update
2 .	guest	123456	Guest Account	☐	Delete	Update
3 .	khanh	kieu123	Kieu Trong Khanh	☑	Delete	Update
4 .	kieuphong	phong33345	Khanh Phong	☐	Delete	Update

http://localhost:8084/Strut2Config/updateRecord.action

STRUTS 2
Expectation

Projects

- Strut2Config
  - Web Pages
    - META-INF
      - context.xml
    - WEB-INF
      - web.xml
    - deleteError.html
    - index.jsp
    - invalid.html
    - login.html
    - search.jsp
    - updateError.html
  - Source Packages
    - <default package>
      - struts.xml
    - sample.registration
      - RegistrationDAO.java
      - RegistrationDTO.java
    - sample.struts2
      - DeleteRecordAction.java
      - LoginAction.java
      - SearchLikeLastnameAction.java
      - UpdateRecordAction.java
    - sample.utils
      - DBUtils.java
  - Test Packages
  - Libraries
  - Test Libraries
  - Configuration Files

- Libraries
  - Struts2 Core 2.3.15 - xwork-core-2.3.15.3.jar
  - Struts2 Core 2.3.15 - struts2-core-2.3.15.3.jar
  - Struts2 Core 2.3.15 - struts2-convention-plugin-2.3.15.3.jar
  - Struts2 Core 2.3.15 - ognl-3.0.6.jar
  - Struts2 Core 2.3.15 - javassist-3.11.0.GA.jar
  - Struts2 Core 2.3.15 - freemarker-2.3.19.jar
  - Struts2 Core 2.3.15 - commons-lang3-3.1.jar
  - Struts2 Core 2.3.15 - commons-io-2.0.1.jar
  - Struts2 Core 2.3.15 - commons-fileupload-1.3.jar
  - Struts2 Core 2.3.15 - asm-tree-3.3.jar
  - Struts2 Core 2.3.15 - asm-commons-3.3.jar
  - Struts2 Core 2.3.15 - asm-3.3.jar
  - JDK 1.8 (Default)
  - Apache Tomcat 8.0.27.0
- Test Libraries

STRUTS 2
Overview

- Achieved **easy development** of Web application by **reducing XML configurations**, or **using Annotations**

- Is a **MVC2** Design Pattern **using Filter as Controller**

- **Features**
  - **JavaBeans** usage **instead of Action form**
  - Annotation and XML configuration **options**
  - **POJO based actions** for easy testing
  - Object Graphic Notation Language (**OGNL**) expression language usage **in integration**
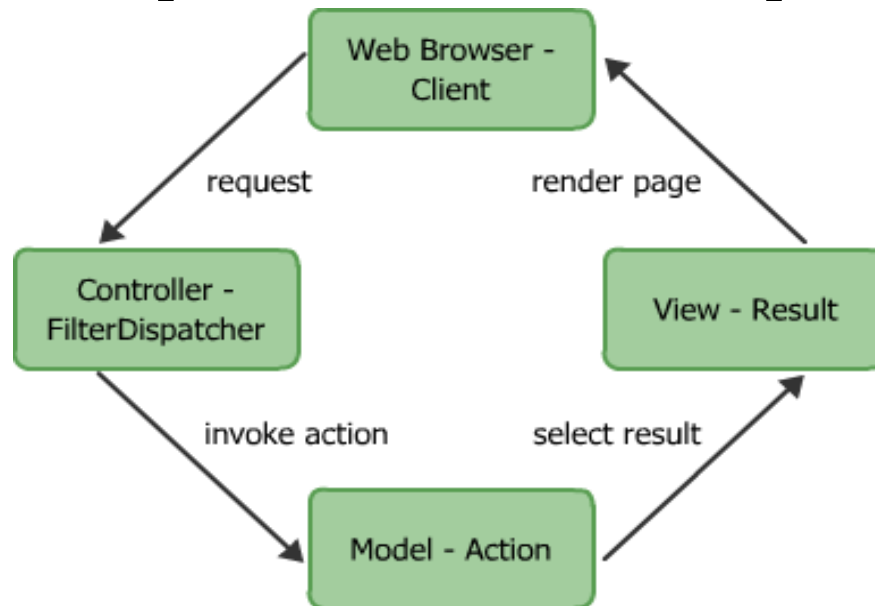 - Usage of **plug-ins to extend** and **modify**

STRUTS2
Comparing with STRUTS 1

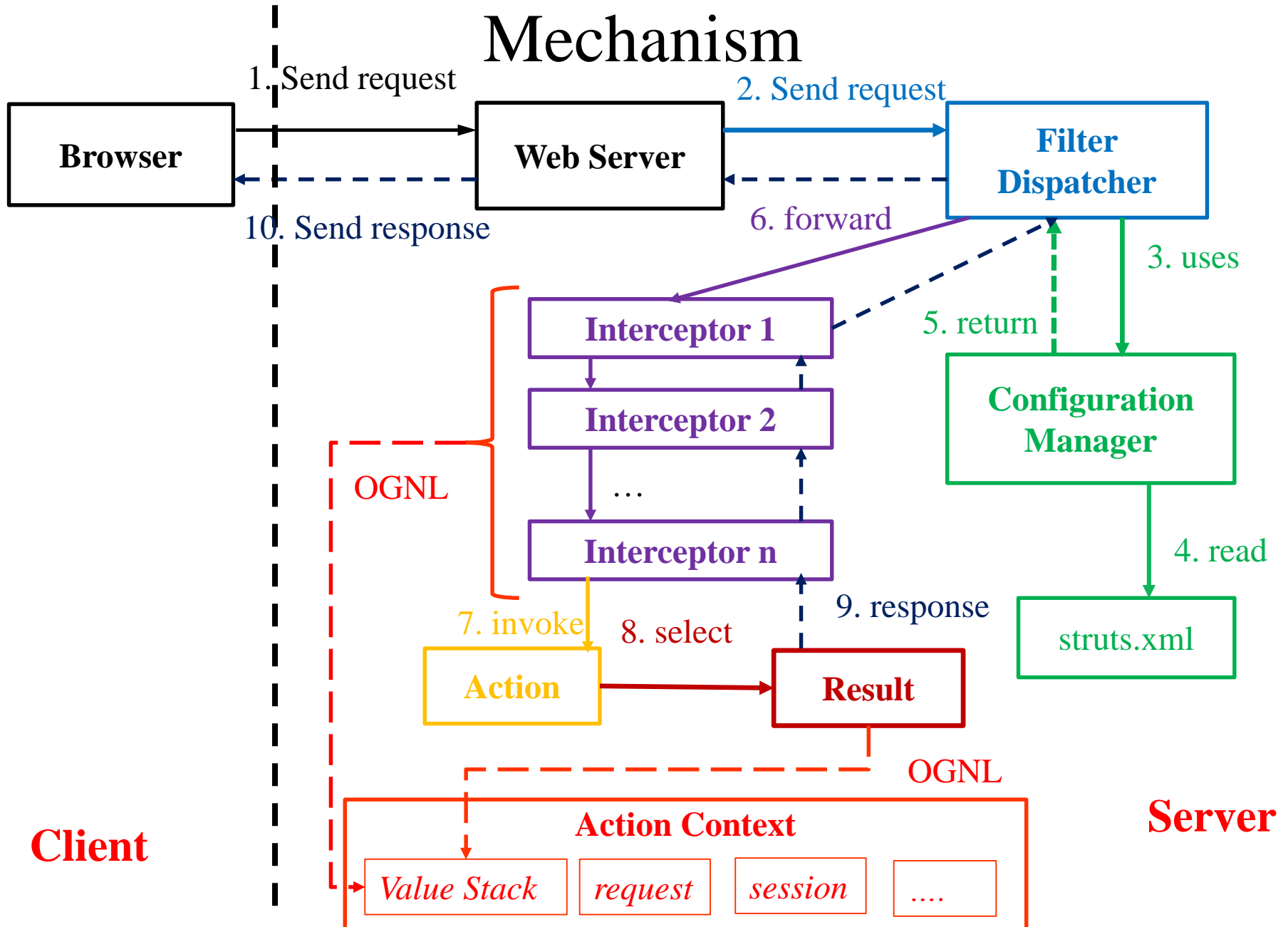| Struts 1	Struts 2
Uses a **Servlet Controller** (**ActionServlet** class)	Uses a **Filter** to act as a **Controller**
Uses an **HTML form mapped** to an **ActionForm object** to capture inputs	**Action properties** are used for capturing input
Implements the **Action interface**	**Action classes do not require** to **implement** the **Action interface** (**POJO**)
Only one instance of the **user-defined Action class** is created for **handling** all **incoming requests** and so it is required to be thread-safe	**Creates instance** of **user-defined Action class for each** incoming **request**
Provides multiple tag libraries such as HTML, Bean, and Logic	**Provide** a **single** tag library
Uses JSTL as an expression language	Uses **OGNL** as an **expression language** which is powerful
Uses JSP mechanism of **binding object** into the page contexts for access	Uses **ValueStack for** the **taglibs** to **access values**
Has a **separate lifecycle** for **each of the module**. All actions share same lifecycle in the module	**Create separate lifecycles** for each action **using the Interceptor Stack**

STRUTS 2
Architecture

- Struts 2 follow the MVC design pattern using Filter as Controller

- Struts 2 is based on a **pull-MVC** framework
  - The data that is to be displayed is pulled from action
  - The action class is a **POJO** class that acts as a model
  - Providing the powerful APIs for configuring the Validators/Interceptors that reduces the processing on the action class

```
        ┌──────────────────┐
        │  Web Browser -   │
        │     Client       │
        └──────────────────┘
       ↙ request    render page ↖

┌──────────────────┐        ┌──────────────────┐
│  Controller -    │        │   View - Result  │
│ FilterDispatcher │        │                  │
└──────────────────┘        └──────────────────┘
       ↘ invoke action    select result ↗
        ┌──────────────────┐
        │  Model - Action  │
        └──────────────────┘
```

STRUTS 2
Mechanism

Browser — 1. Send request → **Web Server** — 2. Send request → **Filter Dispatcher**

10. Send response

6. forward

3. uses

5. return

Interceptor 1

Interceptor 2

…

Interceptor n

OGNL

Configuration Manager

4. read

struts.xml

7. invoke

8. select

9. response

Action → **Result**

OGNL

Action Context

Value Stack | *request* | *session* | *....*
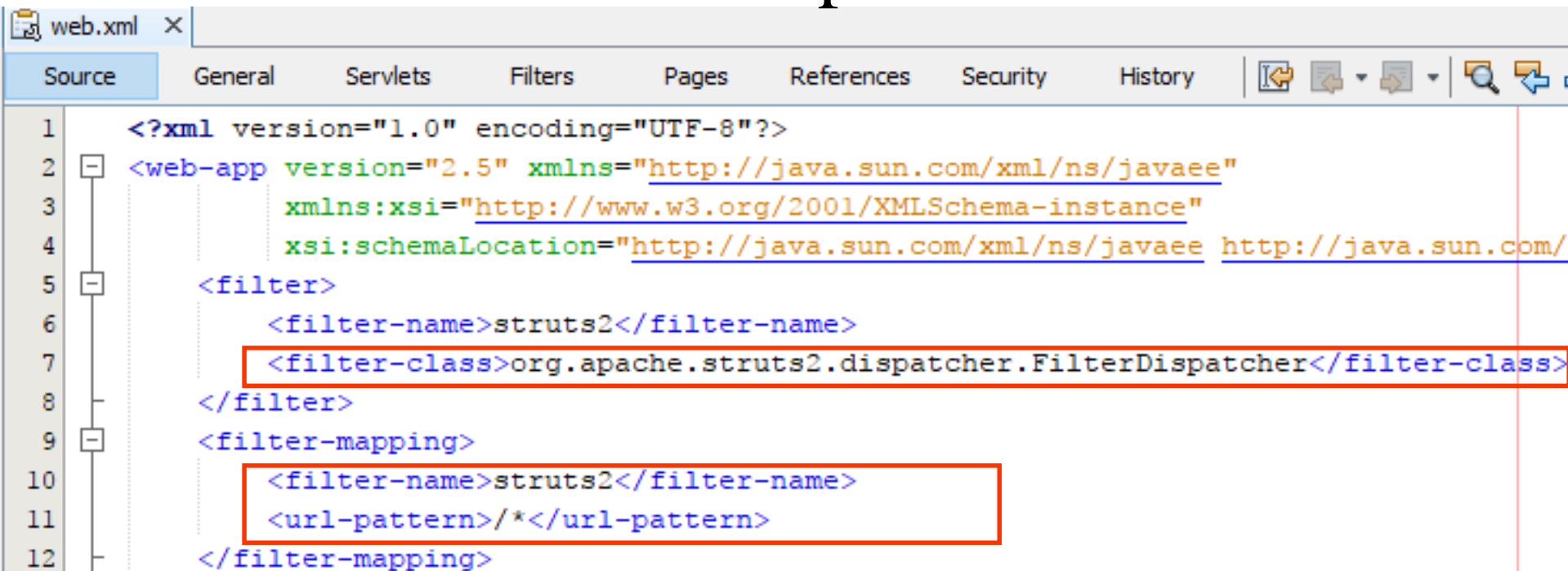
Client

Server

STRUT2
Filter Dispatcher

- **Instantiation** of the **Action** class

  – **Execution** of the **method specified** in the configuration file

  – **Reading** of the **control string**

- **While handling** an action, the Struts 2 **filter performs** the following steps

  – **Looks** for the **execute() method** in the user-defined action class, **if** the **configuration file does not contain** any **other method declaration**

  – **Generates** the **default view**, **when** the **execute()** method **returns any other String constants** besides success

STRUT2
Filter Dispatcher

```xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/
5     <filter>
6         <filter-name>struts2</filter-name>
7         <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
8     </filter>
9     <filter-mapping>
10        <filter-name>struts2</filter-name>
11        <url-pattern>/*</url-pattern>
12    </filter-mapping>
```

STRUT2
Actions

- Are at the **core** of the framework
- **Encapsulate** the unit of work (operation of application)
  - Each URL is mapped to a specific action **containing the business logic**
  - **Encapsulates** the logic to be **executed** for a given **request based** on the **input parameters**
  - All business code should be processed in **execute() method**
- **Provide locus** for data transfer
  - Struts **automatically maps** form's **parameters to JavaBeans'** properties in action and **expose** the **data to the result**
- **Return control string** for result routing
  - **Determines the result** that will be returned in the response
- Struts 2 defines 2 helper interface
  - **Action** and **ActionSupport** class

STRUT2
Action class

- Acts as a **Model** as per the MVC Design pattern
- Is an ordinary Java class
  - That follows **same rules as JavaBeans**
  - Must have **at least one method** that will be invoked when the action is called
  - An action class may be **associated with multiple actions**. In this case, the action class may **provide a different method** for each action
  - Consists of a method named **execute**()
    - **Performs** the **request processing**
    - **Provides encapsulation** of business logic
    - Does **not accept any argument**
    - **Returns** a **String or Result** object
- Some resource as ServletContext, HttpSession, HttpServletRequest, and HttpServletResponse objects can be accessed either through the ServletActionContext object or by implementing Aware interfaces
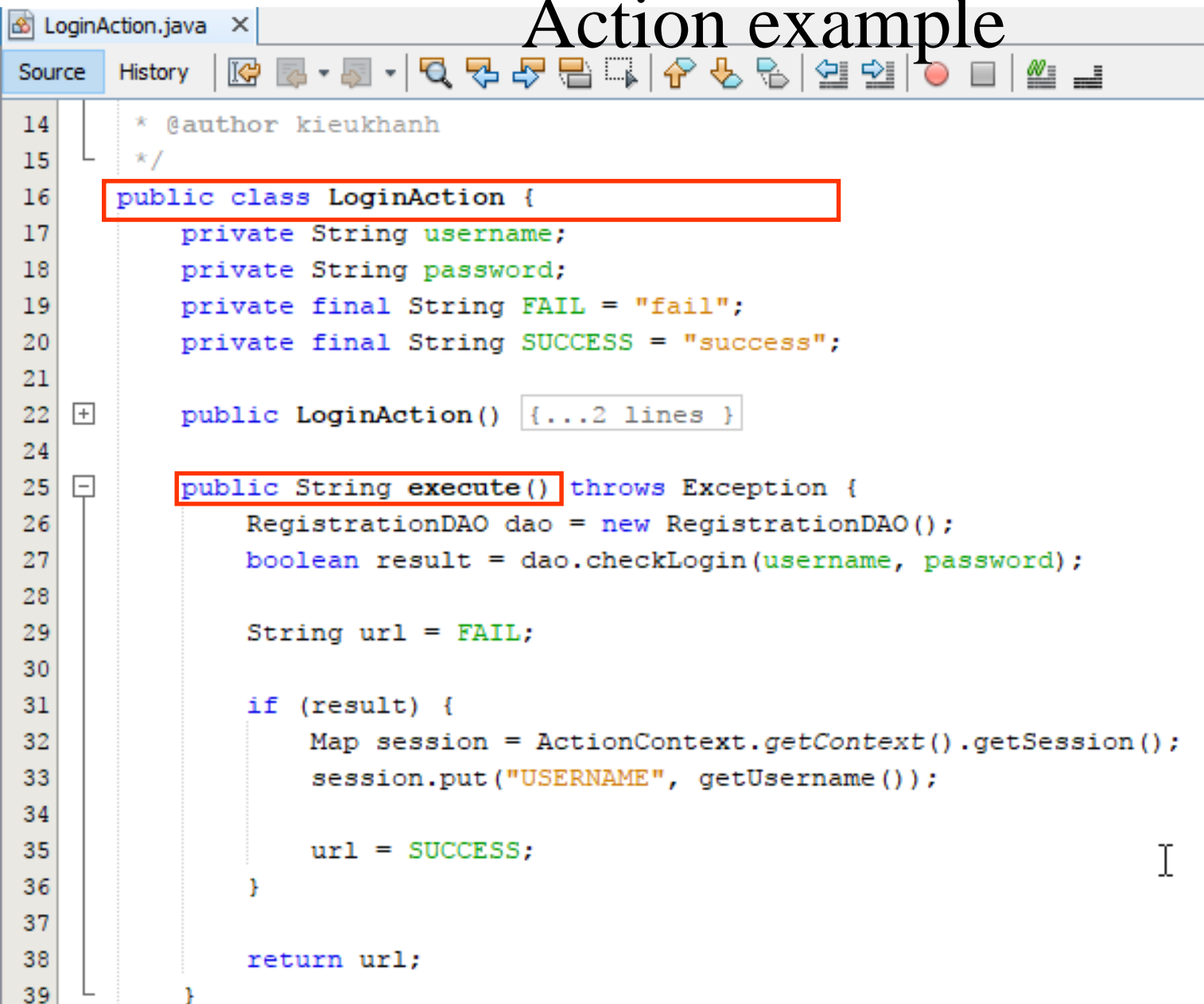
STRUTS 2
Static field in Action

- **SUCCESS**
  - Action **execution** was **successful** and the result view should be shown
  - public static final String SUCCESS = "success";
- **NONE**
  - Action **execution** was **successful but no result view** should be shown
  - public static final String NONE = "none";
- **ERROR**
  - Action execution **failed and an error view** should be sent
  - public static final String ERROR = "error";
- **INPUT**
  - Input validation **failed and the form** that had been **used to take user input should be shown again.**
  - public static final String INPUT = "input";
- **LOGIN**
  - Action could **not execute because the user was not logged in** and the login view should be shown
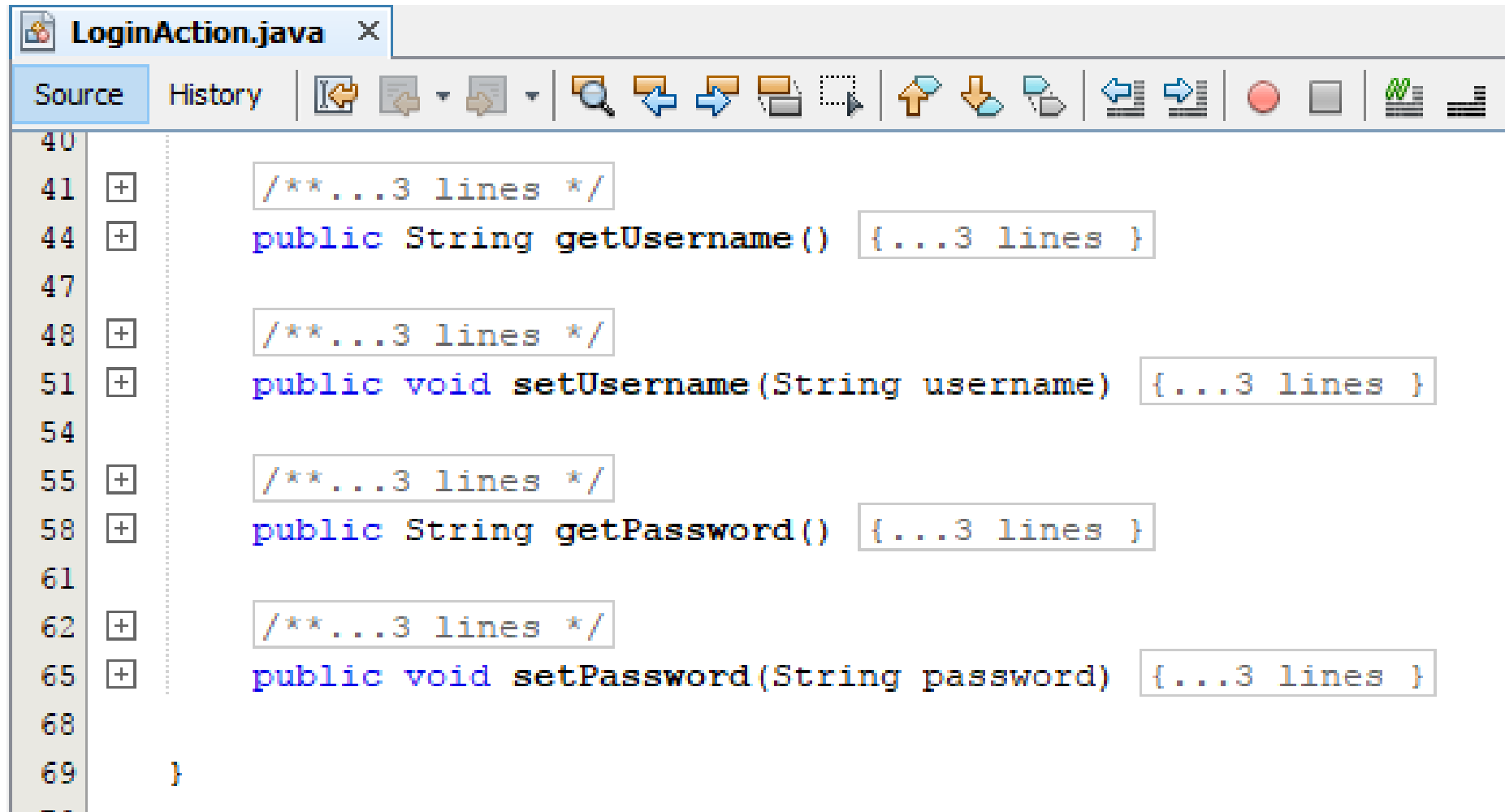 - public static final String LOGIN = "login";

STRUTS 2
Action example

```java
 * @author kieukhanh
 */
public class LoginAction {
    private String username;
    private String password;
    private final String FAIL = "fail";
    private final String SUCCESS = "success";

    public LoginAction() {...2 lines }

    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();
        boolean result = dao.checkLogin(username, password);

        String url = FAIL;

        if (result) {
            Map session = ActionContext.getContext().getSession();
            session.put("USERNAME", getUsername());

            url = SUCCESS;
        }

        return url;
    }
}
```

STRUTS 2
Action example

Source History

```java
40
41  /**...3 lines */
44  public String getUsername() {...3 lines }
47
48  /**...3 lines */
51  public void setUsername(String username) {...3 lines }
54
55  /**...3 lines */
58  public String getPassword() {...3 lines }
61
62  /**...3 lines */
65  public void setPassword(String password) {...3 lines }
68
69  }
```

STRUT2
Action Support class

- A convenience class that **provides default implementations** of the **Action** interface and several other **useful interfaces**, **giving** such things as **data validation** and **localization of error messages**.
- Is **default action class**
  - Struts will create an instance of this class if an **action declaration does not specify an action class**
- Has **already implemented execute() method** that simply **returns** Action.**SUCCESS**
- It implements the **Validateable and ValidationAware** interfaces that are used to **support validation in application**
  - The validate() method in the Validateable interface contains the validation code
  - The ValidationAware interfaces contains methods for storing the error messages generated when validation of a property fails

STRUT2
Action Support class – Example

CreateRecordAction.java

```
     * @author kieukhanh
     */
15   public class CreateRecordAction extends ActionSupport {
16       private String username;
17       private String password;
18       private String confirm;
19       private String lastname;
20       private final String FAIL = "fail";
         private final String SUCCESS = "success";
22
23       public CreateRecordAction() {...2 lines }
         public String execute() throws Exception {...11 lines }
36       /**...3 lines */
39       public String getUsername() {...3 lines }
42       /**...3 lines */
45       public void setUsername(String username) {...3 lines }
48       /**...3 lines */
51       public String getPassword() {...3 lines }
54       /**...3 lines */
57       public void setPassword(String password) {...3 lines }
60       /**...3 lines */
63       public String getConfirm() {...3 lines }
66       /**...3 lines */
69       public void setConfirm(String confirm) {...3 lines }
72       /**...3 lines */
75       public String getLastname() {...3 lines }
78       /**...3 lines */
81       public void setLastname(String lastname) {...3 lines }
84   }
```

STRUT2
Struts Configuration File

- **Defines all aspects** in web application, **including** the **actions**, the **interceptors** that need to be called for each action, and the **possible results** for each action
- **struts.xml file**
  - Defines all the actions

STRUT2

struts.xml – package element

- **Actions** are **grouped** into **packages**
- A typical **struts**.xml file can **have one or many packages**
- **Syntax**

<package **name="name"** [namespace="/|uri"] extends="struts-default">

 <action name="..."/>

 <action name="..."/>

</package>

 – The **namespace** attribute is **optional** and if it is **not present**, the default **value "/" is assumed**

 – A **package element** almost always **extends** the **struts-default package**

STRUT2

struts.xml – action element

- Is **nested** within a package element and represents an action
- Syntax

  <action   name="action_Name"   [class="package.className"] [method="name"]>
       <result ….>…</result>

       …
  </action>

  - An **action** that **does not specify an action class** will be given an instance of the default action class
  - The **method** in the action class will be **executed when** the **action** is **invoked**
  - If the **class** attribute is **present** but the **method** attribute **is not**, execute is **assumed for the method name**

STRUT2

struts.xml – result element

- Is a **sub element** of action and tells Struts **where** application want the **action to be forwarded to**
- **Corresponds** to the **return value** of an action method
- Syntax

  <result   **name="success|return_value"**   type="**dispatcher**|type">

  /pageview

  [<param name="name">value</param>…]

  </result>

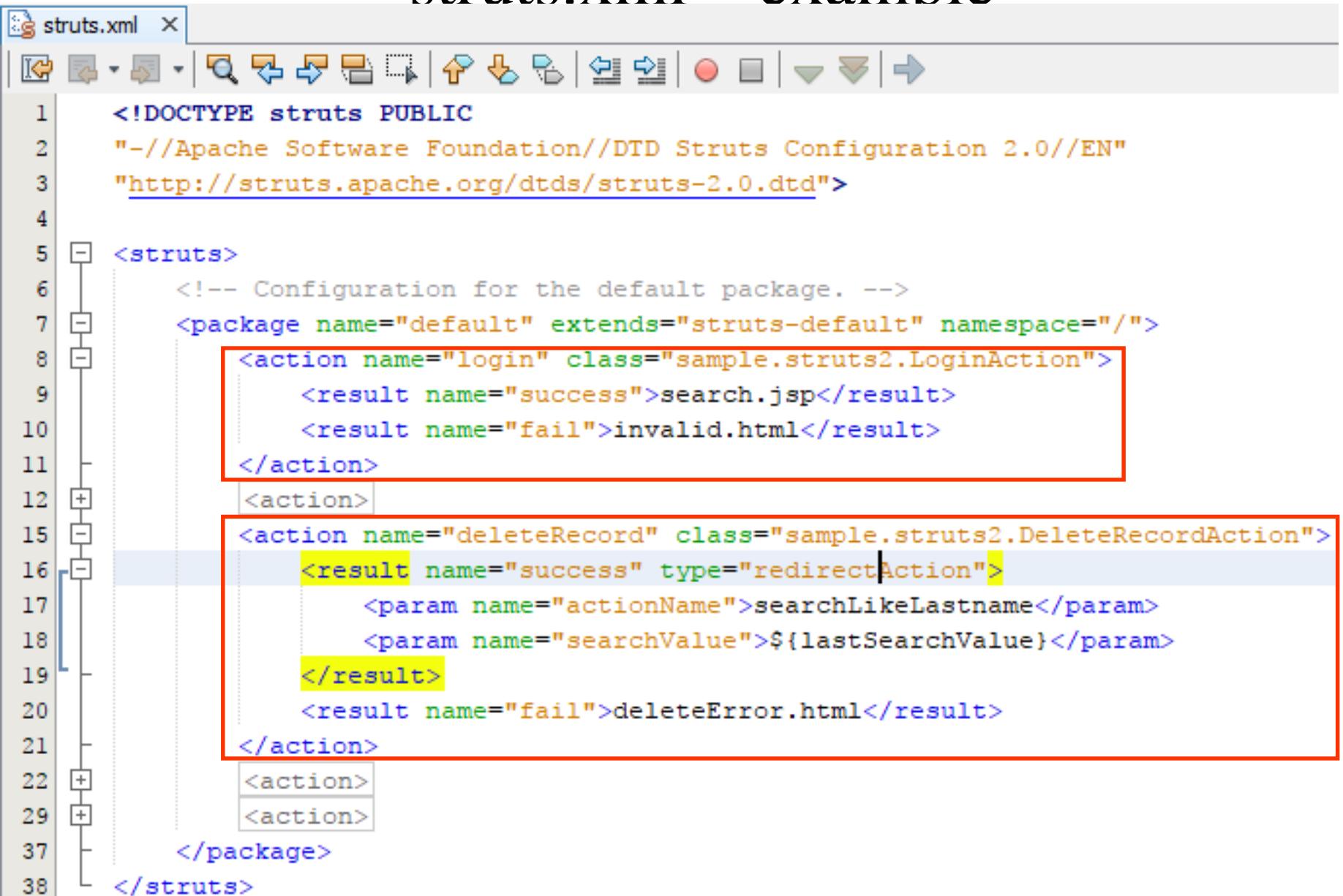 – The **type** attribute of a result element **specifies** the **result type**

STRUT2
struts.xml – param element

- Can be **nested** within **another element** such as action, result-type, … to **pass** a **value** to the **enclosing object**
- **Used** within an **action element**, param can be **used to set an action property**
- Syntax

  <param **name="property"**>value</param>

STRUT2
struts.xml – example

```xml
<!DOCTYPE struts PUBLIC
"-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
"http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
    <!-- Configuration for the default package. -->
    <package name="default" extends="struts-default" namespace="/">
        <action name="login" class="sample.struts2.LoginAction">
            <result name="success">search.jsp</result>
            <result name="fail">invalid.html</result>
        </action>
        <action>
        <action name="deleteRecord" class="sample.struts2.DeleteRecordAction">
            <result name="success" type="redirectAction">
                <param name="actionName">searchLikeLastname</param>
                <param name="searchValue">${lastSearchValue}</param>
            </result>
            <result name="fail">deleteError.html</result>
        </action>
        <action>
        <action>
    </package>
</struts>
```

STRUTS 2
Results

- Are sent in **two parts**
  - **Result**
    - **Defines** the **next** action after the processing of the Action is complete
    - **Result** element in struts.xml allows to select the **String result values**
  - **Result type**
    - **Specifies** the **implementation details** for the **type** of **information** that is **returned to the user**
    - Are **preconfigured** or provided as a **plugin**
    - The **default value is a dispatcher** which uses JSP to render the response to the users

STRUTS 2
Result Type

Method	Description
dispatcher	Forwards to a JSP (default type)
chain	Chains actions with each other
httpheader	Send HTTP headers back to the browser
redirect	Redirects the user to an another URL
redirectAction	Redirects the user to an another action
stream	Streams raw data to the browser Is useful for downloadable content and images
xslt	Renders XML to the browser, which is transformed using XSLT
plaintext	Returns the content as plain text

STRUTS 2
Result & Result Type – Example

```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 name="login" class="sample.struts2.LoginAction">
9                <result name="success">search.jsp</result>
10               <result name="fail">invalid.html</result>
11           </action>
12           <action name="searchLikeLastname" class="sample.struts2.SearchLikeLastnameAction">
13               <result name="success">search.jsp</result>
14           </action>
15           <action name="deleteRecord" class="sample.struts2.DeleteRecordAction">
16               <result name="success" type="redirectAction">
17                   <param name="actionName">searchLikeLastname</param>
18                   <param name="searchValue">${lastSearchValue}</param>
19               </result>
20               <result name="fail">deleteError.html</result>
21           </action>
22           <action>
29           <action name="createRecord" class="sample.struts2.CreateRecordAction">
30               <result name="success">login.html</result>
31               <result name="fail">insertError.html</result>
32               <result name="input">createNew.jsp</result>
33           </action>
34       </package>
35   </struts>
```

STRUTS 2
Value Stack

- Is a **stack of objects** with properties **that all the properties** of these objects appear **as properties of theValueStack** itself
  - If two properties have the **same name**, the value of the **highest object** is returned
- Is a **storage area** that holds the **application data associated** with **request processing**
  - So all the **form properties** will be **stored** on the ValueStack.
- Is **accessed during processing** and by the view to display the action and other information
- Struts **pushes** the **action and related objects to** the **Object Stack** and **pushes various maps to the Context Map**
- Is often used to refer to the Object Stack in the Value Stack
- **OGNL expression** with **a #** is used to **access Context Map**
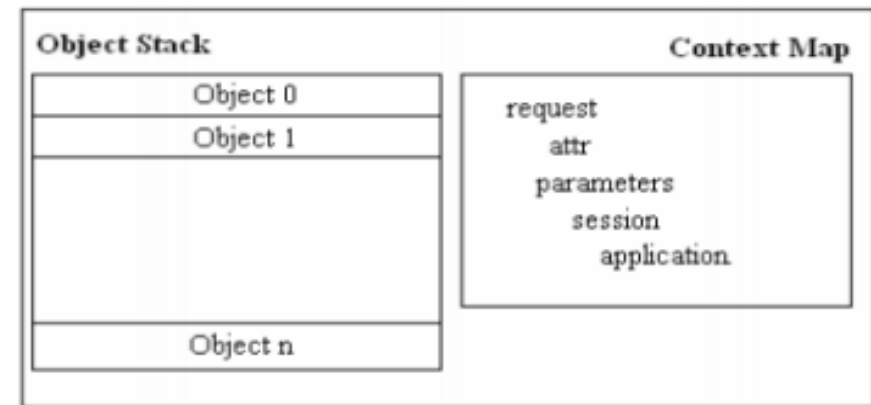- OGNL expression **without a #** is used to **access Object Stack**

Fig4.1, Strut2 Design & Programming: A Tutorial

STRUTS 2
Value Stack Maps

Map Name	Description
parameters	Contains the request parameters for the current request **Notes:** - A request parameter always returns an array of Strings, not a String. - To access the number of request parameters, this example #parameters.count[0] is used
request	Contains all the request attributes for the current request
session	Contains the session attributes for the current user
application	Contains the ServletContext attributes for the current application
attr	Searches for attributes in this order: request, session, and application

STRUTS2
Action Context

- Is a **global storage area** that **holds all** the **data** associated with the processing of a request
  - The **action resides** on the **ValueStack** which is a **part of the ActionContext**
  - When a **request comes**, the **params** interceptor helps in **moving the request data to the ValueStack**

- **HttpServletRequest, Attributes in Session** can be obtained **by asking the ActionContext**
  - HttpServletRequest request = ServletActionContext.getRequest();
  - Map attributes = ActionContext.getContext().getSession();

STRUTS 2
OGNL

- Stands for the **Object Graph Navigation Language**

- Consists of two things: an **expression** language and **type converters**

- **Allows** the **developer to refer** and **manipulate** the data present on the ValueStack

  - **Helps** in data transfer and type conversion (**TypeConverter**) **between data forms and action class** (**form to model object**)

  - Is **an expression and binding language** used for getting/setting the properties of the Java objects

  - **Acts** as a **glue between** the **frameworks** string based **input and output**, and the Java based **internal processing**

- Navigates object graphs **using dot notation** and **evaluate expression**

STRUTS 2
OGNL

1. Send Request

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

Reg Obj

2. Push into

OGNL

Value Stack

Login Action
String username;
Integer password;

3. Access

4. Return Result

5. Send Response

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

```
result.jsp
Username: <s:property value="username"/>
Password: <s:property value="password"/>
```
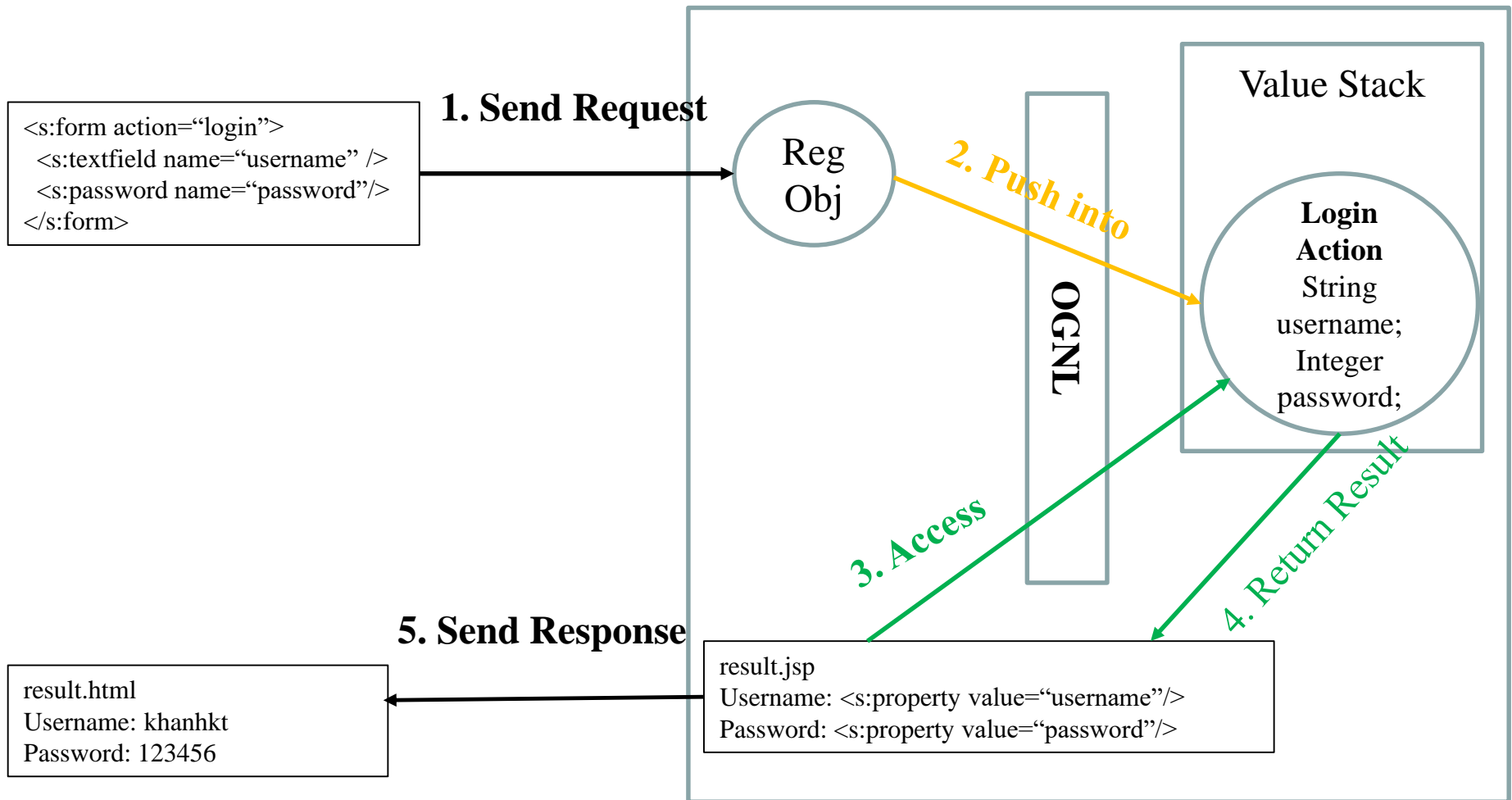
STRUTS 2
OGNL

- The **Action instance** is **always pushed** onto the value stack.
  - **Action properties** can be **referenced without the # marker**.
    - Ex: <s:property value="postalCode"/>
- To **access** the property of an object in the **Object Stack**, **use object.propertyName**
- To **access** the property of an object in the **Context Map**, use **#object.propertyName**
- The **expression** escape sequence is **"%{ ... }"**
  - Any text embedded in the escape sequence is evaluated as an expression.
  - **Refer a value** on the ActionContext
- There are times when OGNL and the Struts custom tags **are not the best choice (ex: configure in struts.xml file)**
  - **JSP EL is a candidate**
  - **Ex**: ${username} replaces <s:property name="username"/>

STRUTS 2
UI Tags

- Are **simple** and **easy to use**.
- Will **automatically generate** them for developer based on the theme him/her select.
  - The developer need not write any HTML code
  - By default the XHTML theme is used.
    - The XHTML theme uses tables to position the form elements
- The **textfield tag** is used to create a textfield or input text
  - **Syntax**: <s:textfield name="name" label="label" [value="val"] />
  - The name attribute is the one which links the property on the ValueStack.
  - The value attribute populates some default value for a specific field
  - The value set in the label attribute will be used to render the label for that particular field while generating the HTML code
- The **password tag** is used to create a password field
  - **Syntax:** <s:password name="name" label="label" />
- The **hidden tag** renders an HTML input element of type hidden, populated by the specified property from the ValueStack.
  - **Syntax:** <s:hidden name="name" value="label" />

- The **radio tag creates** radio buttons
  - **Syntax:** <s:radio name="name" label="label" list="value" />
  - The list attribute of the radio tag is used to specify the option values. The value of the list attribute can be a Collection, Map, Array or Iterator, or constant {'val1', 'val2', ..}

- The **select tag** is drop down control
  - **Syntax**: <s:select name="name" list="values" listKey="refName" listValue="showName" headerKey="0" headerValue="header" label="label" />
  - The list attribute of the radio tag is used to specify the option values. The value of the list attribute can be a Collection, Map, Array or Iterator, or constant {'val1', 'val2', ..}
  - The listKey attribute displays value in the frontend
  - The listValue attribute displays value and store it in the backend
  - The first value can be specified using the headerValue attribute and the corresponding key value is specified using the headerKey attribute

- The **textarea tag** is used to create a text area
  - **Syntax:** <s:textarea name="name" label="label" />

STRUT2 – UI Tags

- The **checkboxlist tag i**s similar to that of the select tag, the only difference is that it displays boxes for each option instead of a dropdown.
  - It returns an array of String values
  - **Syntax:** <s:checkboxlist list="values" name="name" label="label" />
- The checkbox tag returns a boolean value.
  - If the checkbox is checked then true is returned else false is returned
  - **Syntax**: <s:checkbox name="name" label="label" />
- The submit tag is used to create the Submit button
  - **Syntax:** <s:submit name="name" value="label" />
- The reset tag is used to create the Reset button
  - **Syntax:** <s:reset value="label" />
- The form tag Renders HTML an input form
  - **Syntax:** <s:form action="action" method="httpMethod" />
  - **Mechanism**
    - If the action attribute is not specified, then the current request will be used to determine the posting url
    - If the action is given, Struts will try to obtain an ActionConfig. This will be successfull if the action attribute is a valid action alias defined struts.xml.
    - If the action is given and is not an action alias defined in struts.xml, Struts will used the action attribute as if it is the posting url, separting the namespace from it and using UrlHelper to generate the final url

STRUT2
Data Tags

- Are access data, assign value, print or output value, create link, ...
- The property tag
  - Is used to retrieve the data from the ValueStack or some other object in the ActionContext like application or session
  - Is used to get the property value from a class, which will default to the current Action class (top of the stack) property if none is specified
  - Is used to get the property value from a bean class
  - **Syntax**: <s:property value="propertyName| expressionLanguage " />
    - In default, the tag access the current Action class.
    - If propertyName is object/bean, the dot (.) operator with one more level is used to access the property value
- The **set tag**
  - Is used to assign a property value to another name
  - Helps in accessing the property in a faster and easier way
    - Do not need to go one level deeper and fetch it when the property tag is used, instead the set tag can assign the value to another property in the ActionContext and access it directly
  - Assign a value to a variable in a specified scope (application, session, request, page, or action), the action is the default scope
  - **Syntax:** <s:set name="propername" value="value" [scope=""] />

STRUT2
Data Tags

- The **url tag**
  - Is used to create an URL and output it as a text format.
  - Is never work by itself, but it can provides URL to other tags like <s:a> to create a hyperlink or <img> to render an image
  - **Syntax**:
    - Create in image <img src="**<s:url value="/path"/>**" />
    - Create link **<a href="<s:url value="url" />">value**
    - Create an Action URL with one or more parameters

    <s:url action="actionName" >
       <s:param name="name">value</s:param>

       …
    </s:url>

    - Create an Action URL with one or more parameter, and combine with <s:a> tag via <s:property>.

    <s:url action="actionName" var="varName" >
       <s:param name="name">value</s:param>
    </s:url>
    <a href="<s:property value="#varName"| "%{varName}" />" >value

STRUT2
Data Tags

- ## The **a tag**
  - Is used to render a HTML "<a>" tag.

  - The best practice is always use "<s:url>" tag to create the URL and embed it into the "a" tag

  - **Syntax**:

    <s:url action="url" id="varName" />
    <s:a href="%{varName}">value</s:a>

STRUTS 2
Control Tags

- Are used to perform basic condition checking
  - The 'If' tag could be used by itself

<s:if test="expressionCondion">
    statement or text
</s:if>

  - The elseif tag can use it if there are multiple conditions to check

<s:if test=" expressionCondion ">
    statement or text
</s:if>
<s:elseif test=" expressionCondion}">
    statement or text
</s:elseif>

  - The else tag

<s:if test=" expressionCondion ">
    statement or text
</s:if>
<s:elseif test=" expressionCondion}">
    statement or text
</s:elseif>
<s:else>

    statement or text

</s:else>

STRUTS 2 – Control Tags – Iteration Tag

- Is used to loop over a collection of objects.
- Can iterate over any Collection, Map, Enumeration, Iterator, or array

  <s:iterator value="propertyName|expression|values" [status="statusName"]>

  statement or text

  </s:iterator>

  – The value property references the object that can be iterated
  – The status attribute create an instance of the IteratorStatus object
    - The newly created instance will be placed in the ActionContext which can be refered using the the following OGNL expression #statusName
  – The table show the different **properties** of the **IteratorStatus object.**

Name	Return Type	Description
index	int	zero-based index value.
count	int	index + 1
first	boolean	returns true if it is the first element
last	boolean	returns true if it is the last element
even	boolean	returns true if the count is an even number.
odd	boolean	returns true if the count is an odd number.
modulus	int	takes an int value and returns the modulus of count.

STRUTS 2
Create the Struts 2 projects in NetBeans

- 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
  - **Step 3**: **Create all action** class
    - Define input properties and accessor methods
    - Implement the execute() method to perform the action
  - **Step 4**: **Configure** the **struts.xml or set the annotation** for establishing relationship between the Views and Controllers
  - **Step 5**: Build, Deploy and Test application

Summary

Browser — 1. Send request → **Web Server** — 2. Send request → **Filter Dispatcher**

10. Send response

6. forward

3. uses

5. return

Configuration Manager

Interceptor 1

Interceptor 2

…

Interceptor n

OGNL

4. read

struts.xml

7". Access

7'. Call

7. invoke

8. select

9. response

DB

DAO

Action

Result

OGNL

Action Context

Value Stack *request* *session* ....

Client

Server

Q&A

Summary

Send Request

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

Value Stack

Reg Obj

OGNL

Login Action
String username;
Integer password;

Result

Send Response

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

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

OGNL - Q&A

Next Lecture

- How to build web application using Struts 2 Framework Combining Validations?

- How to build web application using Struts 2 Framework with Annotations?

Next Lecture

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
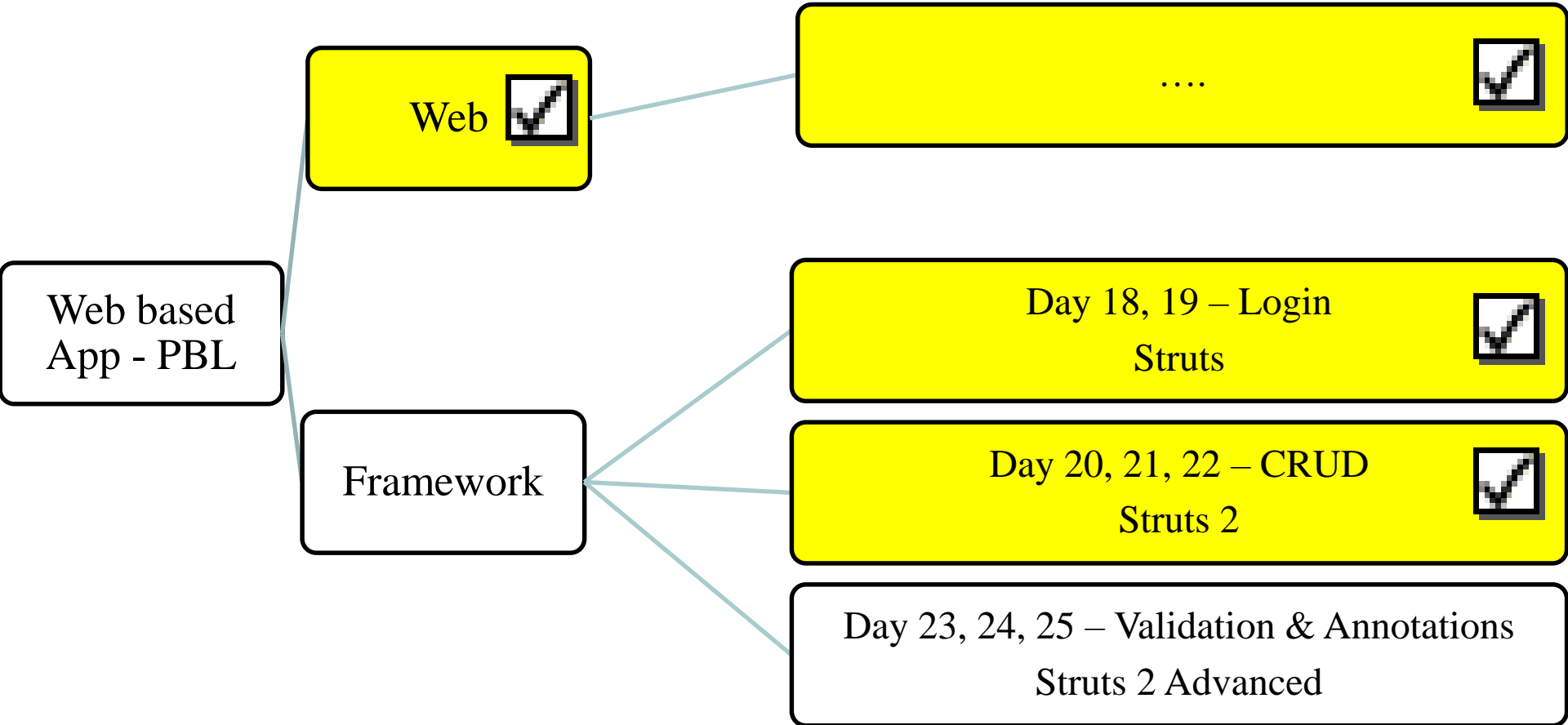
APPENDIX
Struts 2 Architecture

- The general mechanism bases on MVC pattern
  - The client – browser sends request to the server/ container
  - The container receives the request, then sends to the controller
  - The controller checks the mapping (basing on XML or annotation configuring) to determine the correct action
  - The controller invokes the action and hands over control of the request processing to the action
  - The action completes the request processing it forwards the result to the view
    - The action has to choose which result should be displayed such as success result or error result

APPENDIX
Struts 2 Architecture

- Controller – FilterDispatcher
  - Is a servlet filter
  - Checks each incoming requests and determines the Struts 2 action to handle the request
  - Is specified either through XML based configuration files or annotations
- Model – Action
  - Is a business object and has no user interface
    - Is the core of an application as it contains both the data model and the business logic
    - Is defined as the internal state of an application
  - Is implemented by the action component
  - Serves two roles:
    - encapsulates the business logic
    - focus of data transfer
  - Controller invokes the action and hands over control of the request processing to the action
- View – Result
  - Returns the result page to the Web browser
  - Translates the state of the application to a visual presentation with which the user can interact
  - Is the outcome of the action processing

APPENDIX
Configure Struts 2 plugin in NetBeans

- Install plugin steps
  - Download strut2 support netbeans from netbeans plugin website www.netbeans.org (struts2-suite-1.3.5-nb74.zip)
  - Unzip the zip file that can get 3 following files
    - org-netbeans-modules-web-frameworks-struts2lib-v2_3_15.nbm
    - org-netbeans-modules-web-frameworks-struts2.nbm
    - org-netbeans-modules-framework-xwork.nbm
  - Install plugin in netbeans
    - Tools\ Plugins\ Downloaded\ Click Add Plugins …
    - Browser to above 3 files, Click Install
  - Restart or Start again Netbeans

APPENDIX
Configure Struts 2 plugin in NetBeans

APPENDIX
Configure Struts 2 plugin in NetBeans

- Checking on Netbeans IDE
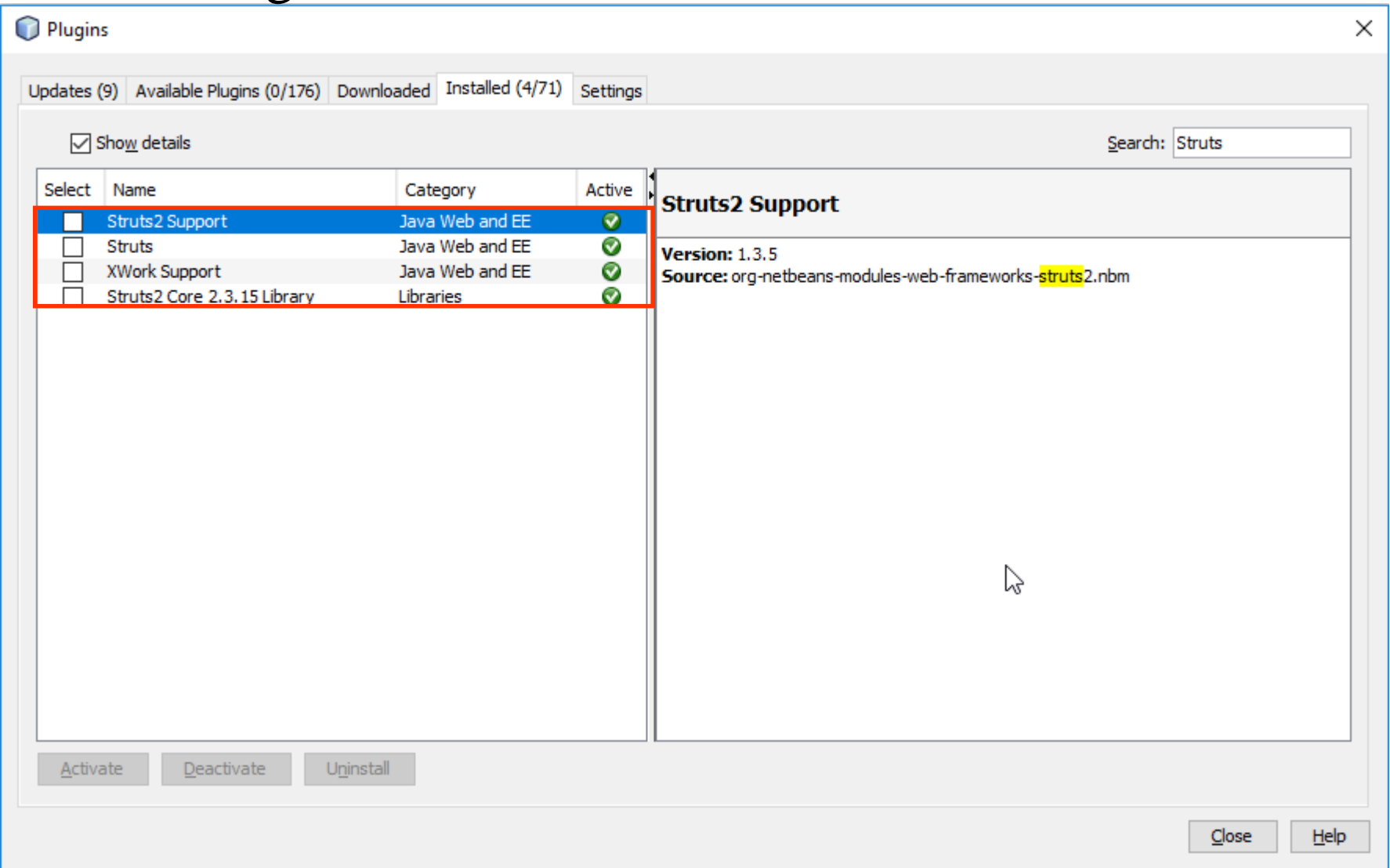
APPENDIX
Configure Struts 2 plugin in NetBeans
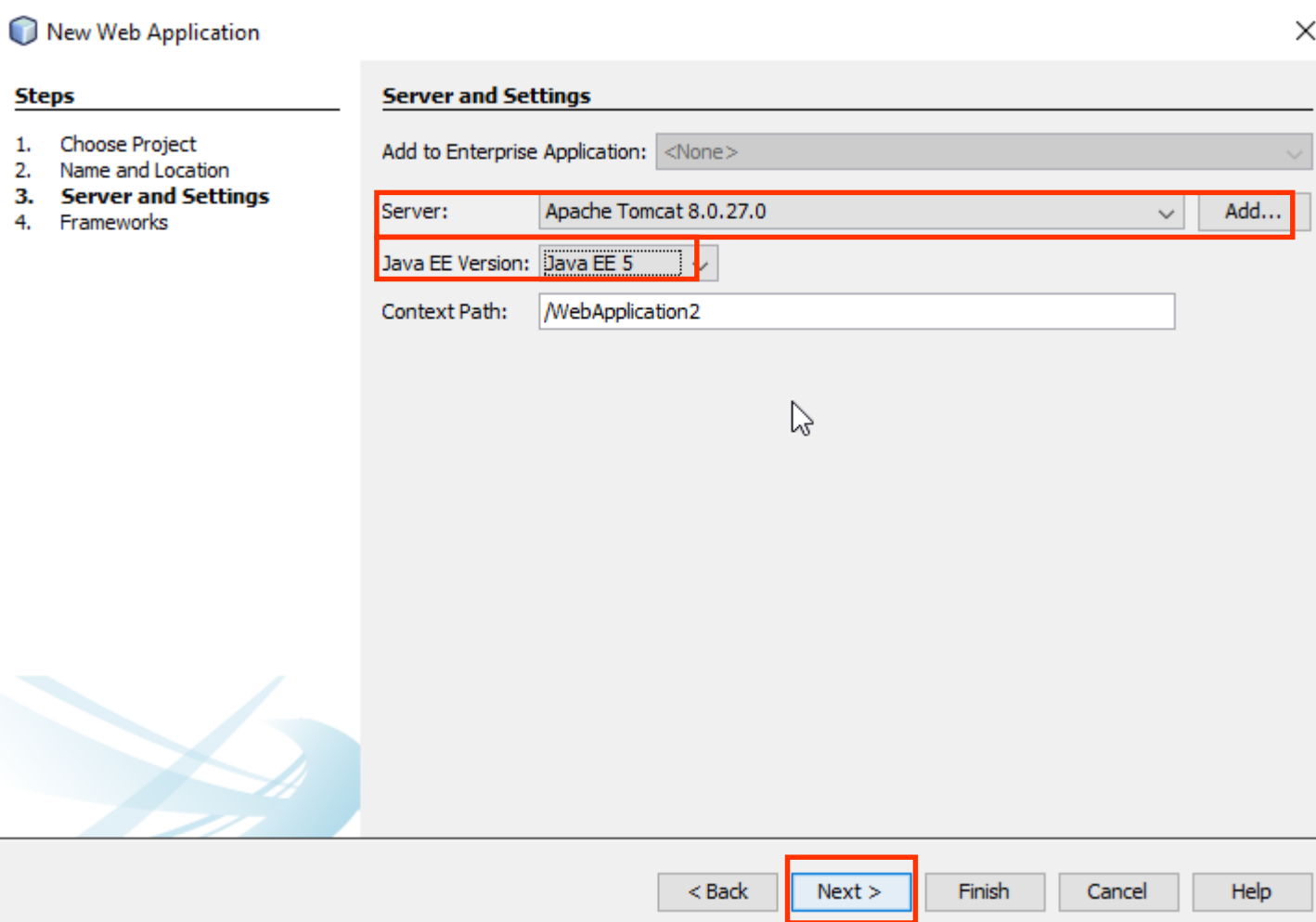
- Checking on Netbeans IDE
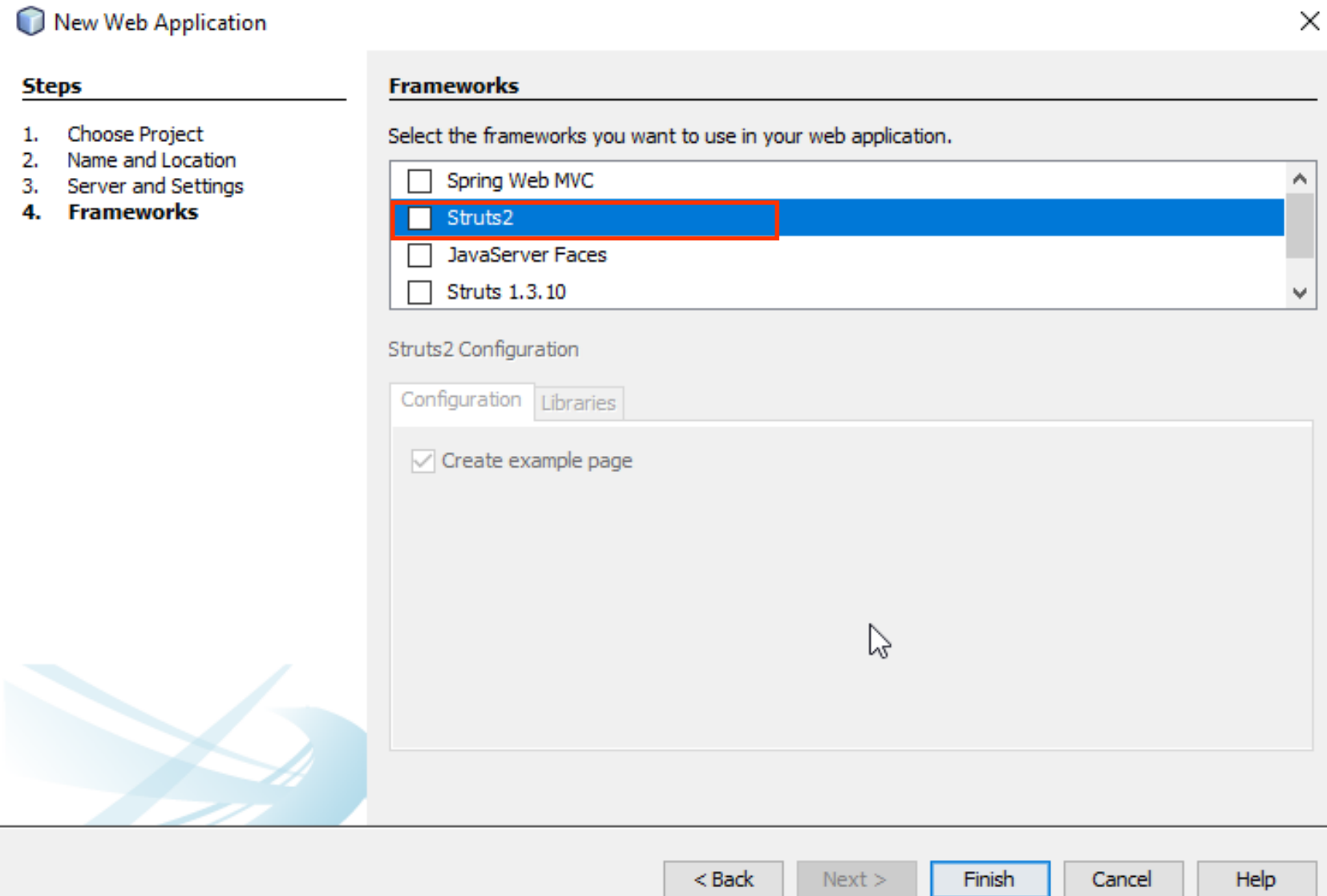
Create Web application projects

- Creating Web application projects
- In Server and Settings
  - Choose Server: Apache Tomcat
  - JavaEE version: JavaEE 5
  - Click Next

Create Web application projects

- In Choosing Frameworks
  - Choose Struts2
  - In Struts2 Configuration, choose tab Configuration
    - Uncheck checkbox "Create example pages"
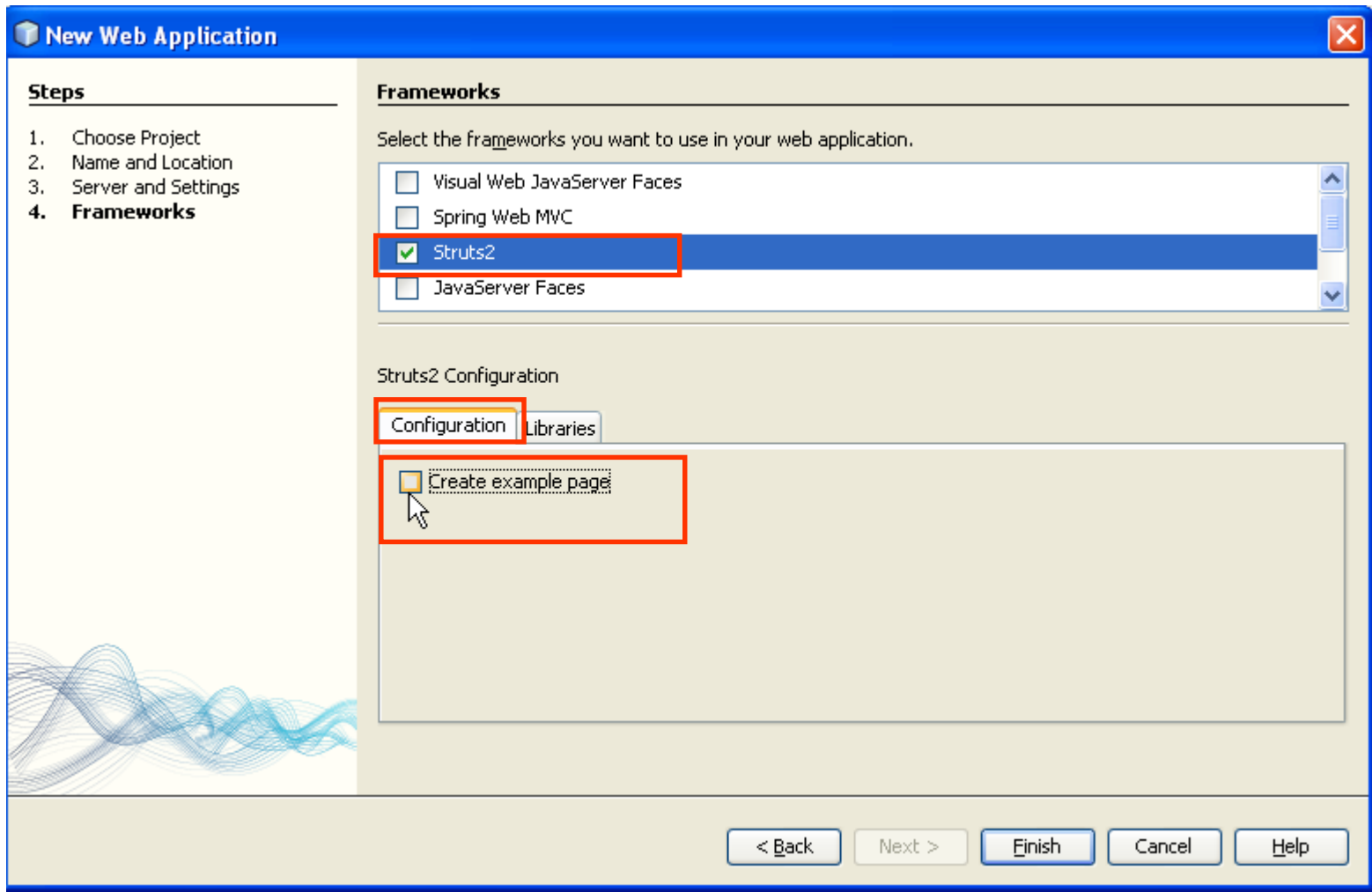
Create Web application projects

- In Choosing Frameworks (cont)
  - Choose tab Libraries
    - Choose Register Libaries
    - Choose Struts2 Core 2.3.15
    - Click Finish
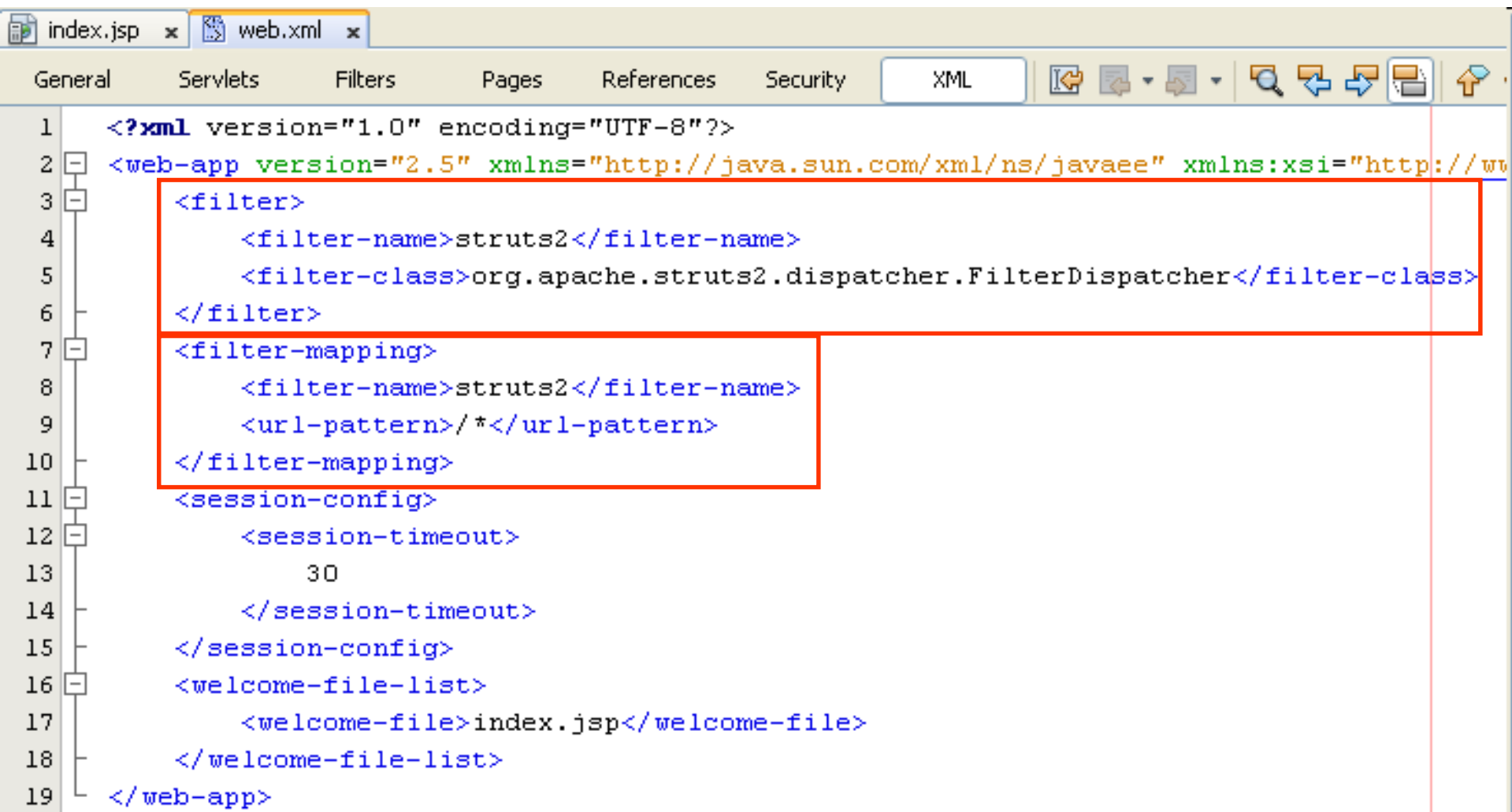
Create Web application projects

Create Web application projects


```xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www
3      <filter>
4          <filter-name>struts2</filter-name>
5          <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
6      </filter>
7      <filter-mapping>
8          <filter-name>struts2</filter-name>
9          <url-pattern>/*</url-pattern>
10     </filter-mapping>
11     <session-config>
12         <session-timeout>
13             30
14         </session-timeout>
15     </session-config>
16     <welcome-file-list>
17         <welcome-file>index.jsp</welcome-file>
18     </welcome-file-list>
19 </web-app>
```

EXAMPLE

- Create the web application using Struts 2 that have all function as
  - Authentication
  - Search last name
  - Delete, Update data row on data grid
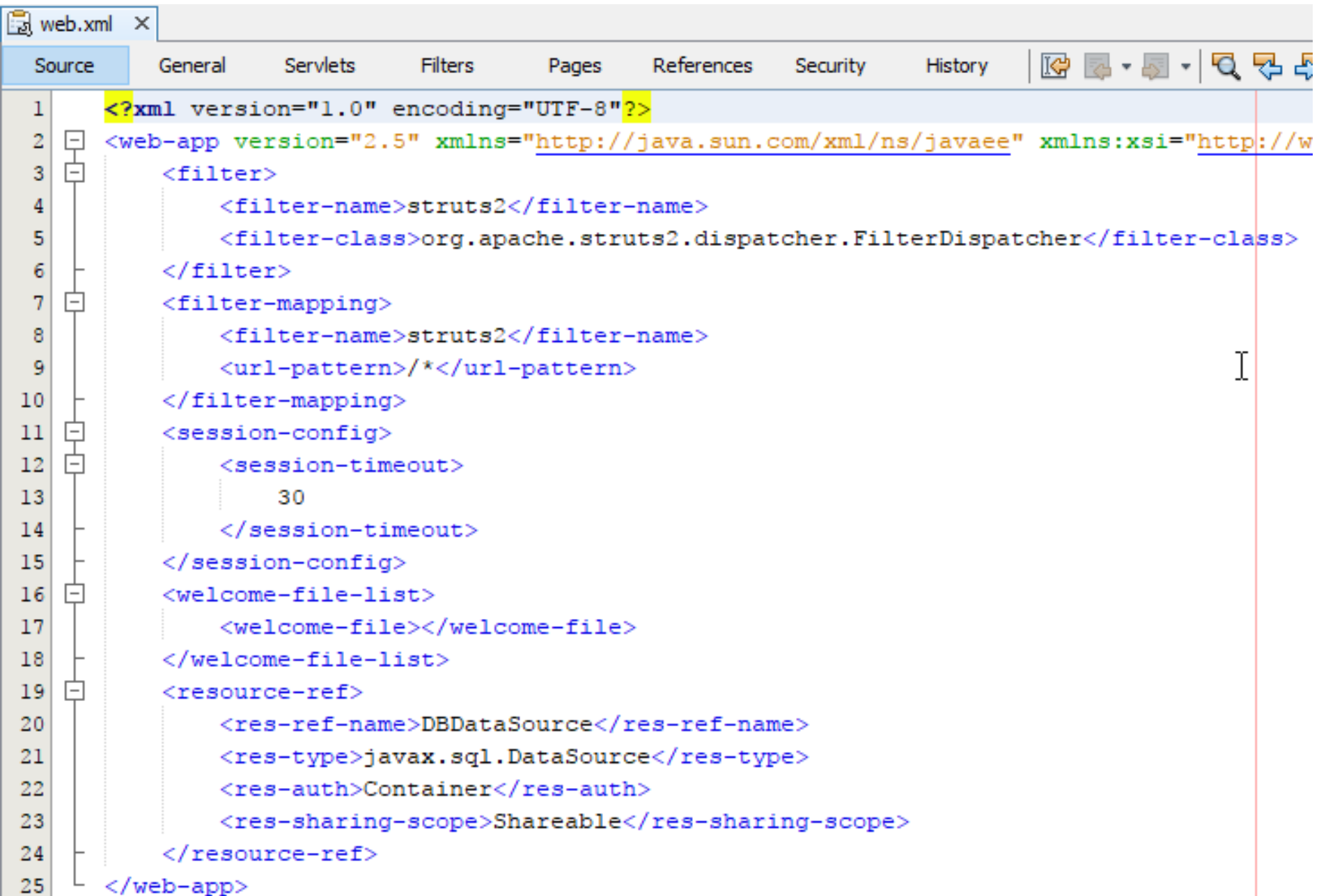
EXAMPLE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/Strut2Config">
    <Resource name="DBDataSource" type="javax.sql.DataSource" auth="Container"
        driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
        url="jdbc:sqlserver://localhost:1433;databaseName=Sinhvien2K14;instanceName=SQL2014"
        username="sa" password="trongkhanh"/>
</Context>
```
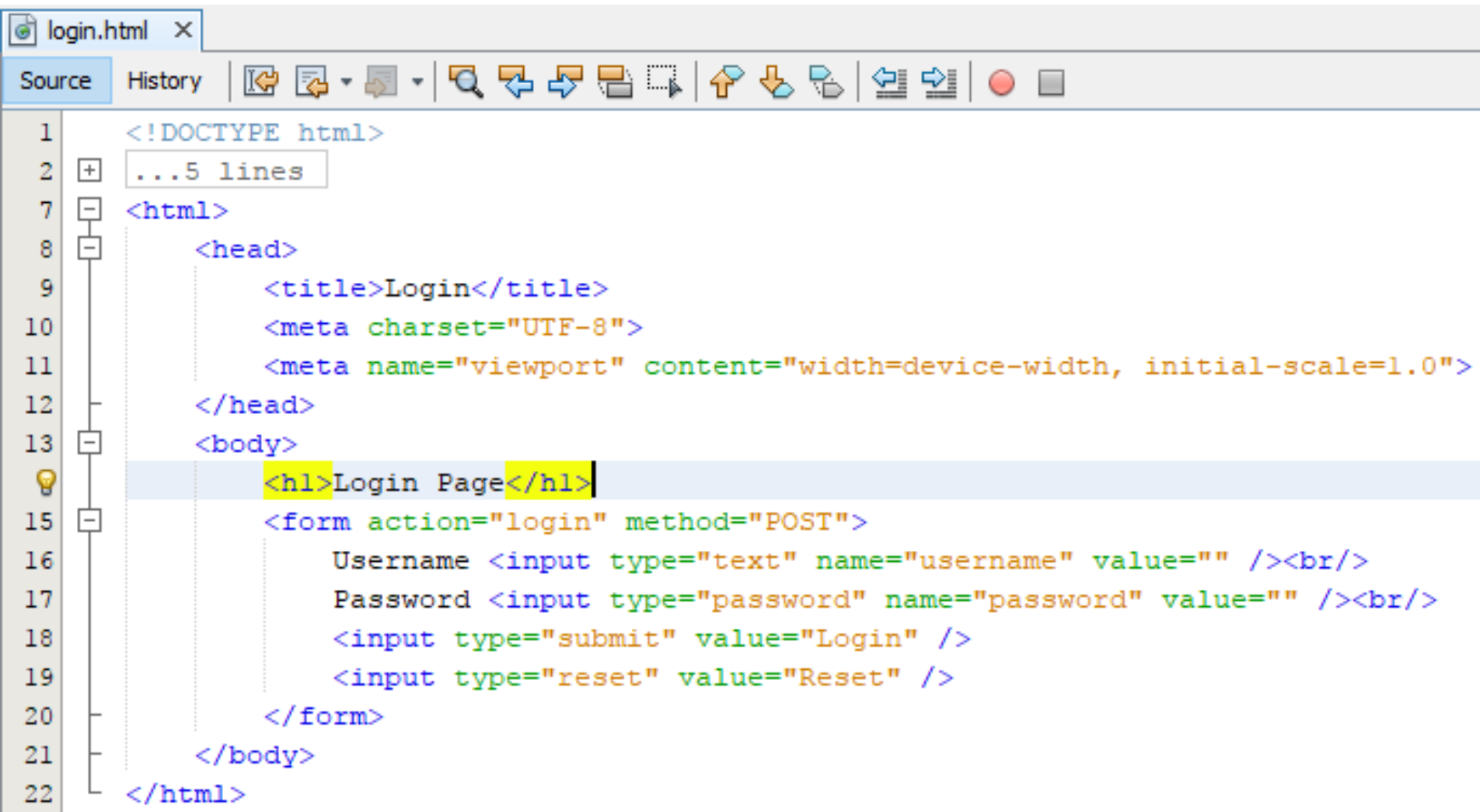
EXAMPLE

Source	General	Servlets	Filters	Pages	References	Security	History

```xml
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <web-app version="2.5" 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      </filter>
 7      <filter-mapping>
 8          <filter-name>struts2</filter-name>
 9          <url-pattern>/*</url-pattern>
10      </filter-mapping>
11      <session-config>
12          <session-timeout>
13              30
14          </session-timeout>
15      </session-config>
16      <welcome-file-list>
17          <welcome-file></welcome-file>
18      </welcome-file-list>
19      <resource-ref>
20          <res-ref-name>DBDataSource</res-ref-name>
21          <res-type>javax.sql.DataSource</res-type>
22          <res-auth>Container</res-auth>
23          <res-sharing-scope>Shareable</res-sharing-scope>
24      </resource-ref>
25  </web-app>
```
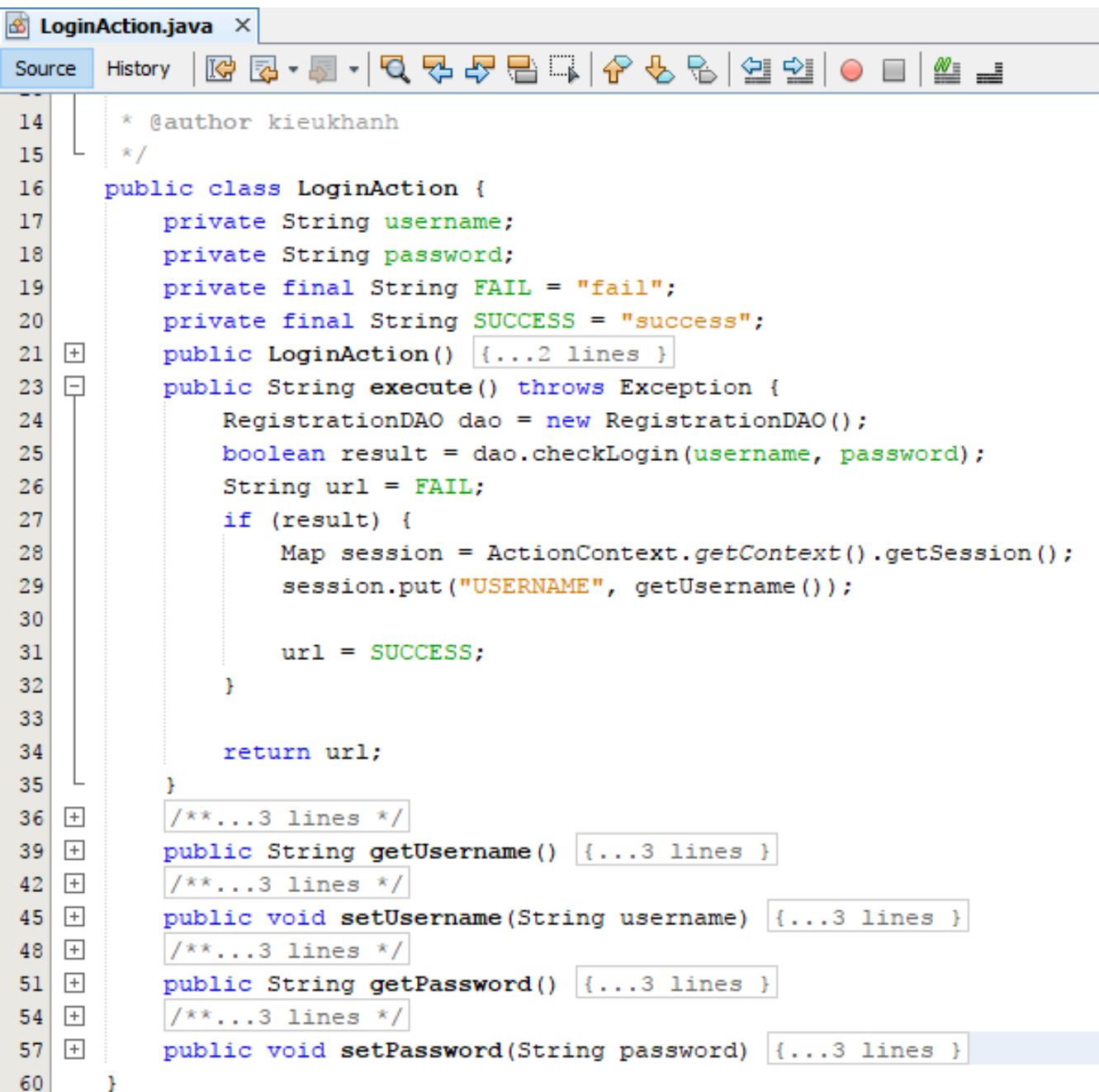
EXAMPLE

```
login.html  ×

Source   History

 1    <!DOCTYPE html>
 2  +  ...5 lines
 7  -  <html>
 8  -      <head>
 9            <title>Login</title>
10            <meta charset="UTF-8">
11            <meta name="viewport" content="width=device-width, initial-scale=1.0">
12        </head>
13  -     <body>
          <h1>Login Page</h1>
15  -         <form action="login" method="POST">
16                Username <input type="text" name="username" value="" /><br/>
17                Password <input type="password" name="password" value="" /><br/>
18                <input type="submit" value="Login" />
19                <input type="reset" value="Reset" />
20            </form>
21        </body>
22    </html>
```

EXAMPLE

```java
 * @author kieukhanh
 */
public class LoginAction {
    private String username;
    private String password;
    private final String FAIL = "fail";
    private final String SUCCESS = "success";
    public LoginAction() {...2 lines }
    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();
        boolean result = dao.checkLogin(username, password);
        String url = FAIL;
        if (result) {
            Map session = ActionContext.getContext().getSession();
            session.put("USERNAME", getUsername());

            url = SUCCESS;
        }

        return url;
    }
    /**...3 lines */
    public String getUsername() {...3 lines }
    /**...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 }
}
```

EXAMPLE

```java
/**
 *
 * @author kieukhanh
 */
public class DBUtils implements Serializable {
    public static Connection makeConnection()
        throws SQLException, NamingException {

        Context context = new InitialContext();
        Context tomcatCtx = (Context)context.lookup("java:comp/env");

        DataSource ds = (DataSource) tomcatCtx.lookup("DBDataSource");

        Connection con = ds.getConnection();

        return con;
    }

}
```

DBUtils.java

RegistrationDAO.java

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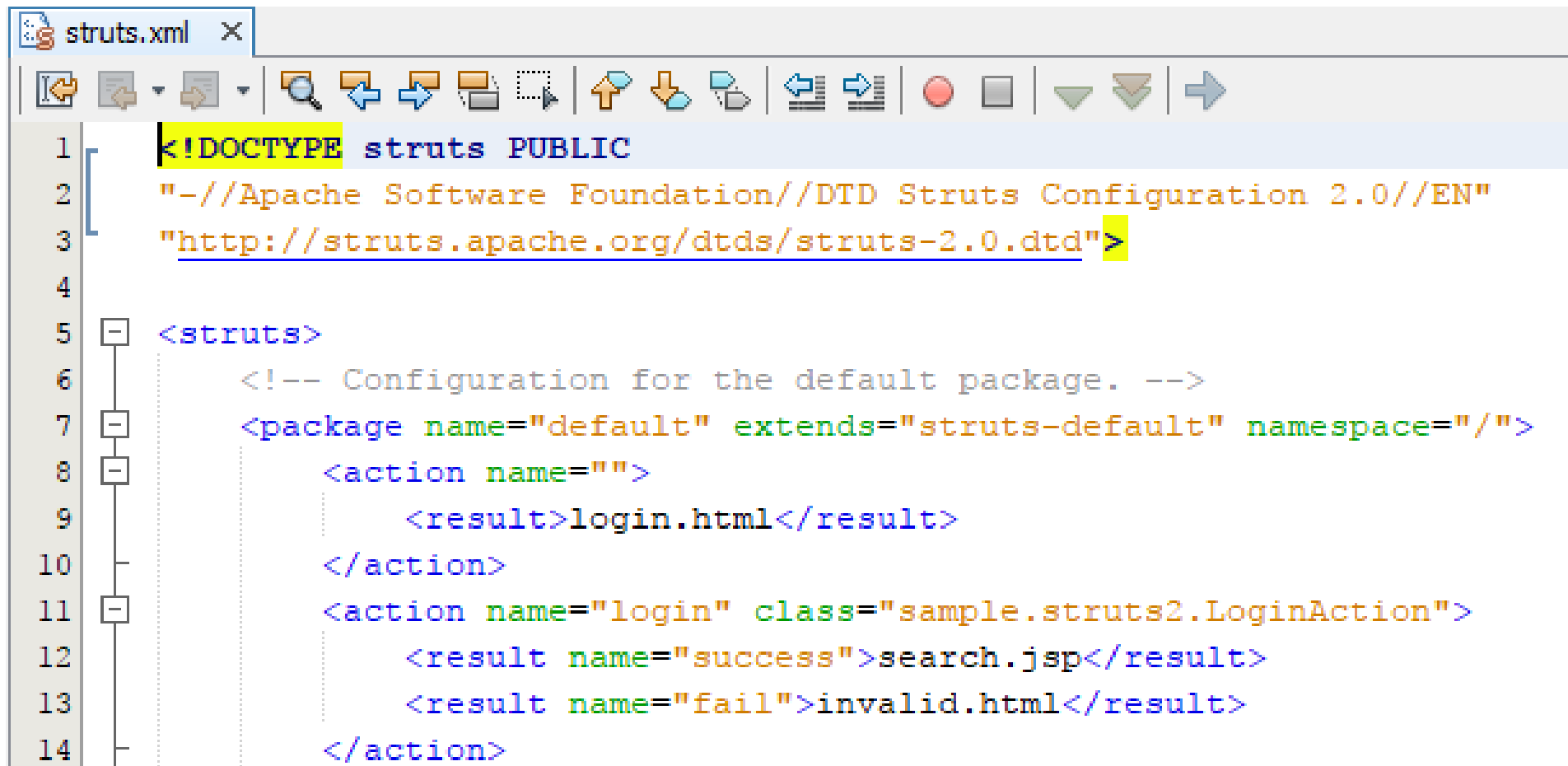
EXAMPLE

```html
1   <!DOCTYPE html>
2   ...5 lines
7   <html>
8       <head>
9           <title>Invalid</title>
10          <meta charset="UTF-8">
11          <meta name="viewport" content="width=device-width, initial-scale=1.0">
12      </head>
13      <body>
14          <h1>
                <font color="red">
16                  Invalid username or password
17              </font>
18          </h1>
19
20          <a href="">Click here to try again!!!</a><br/>
```
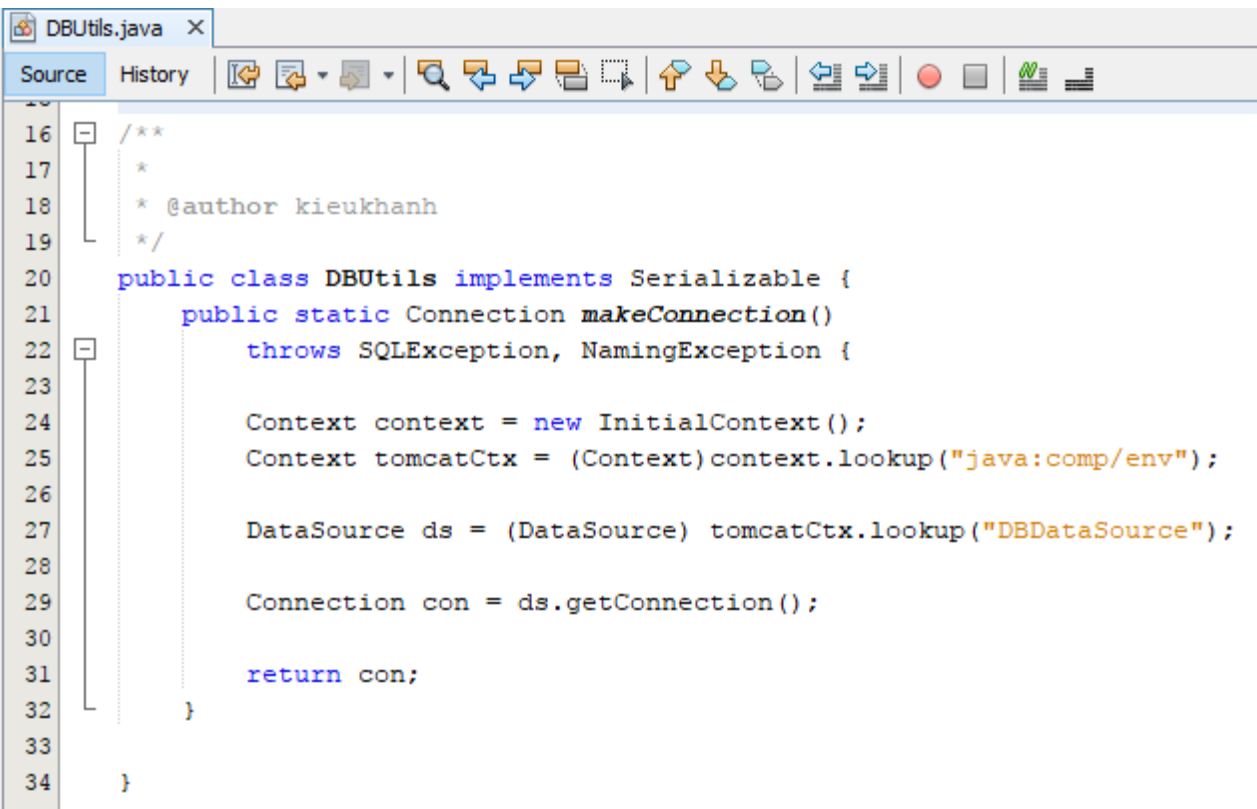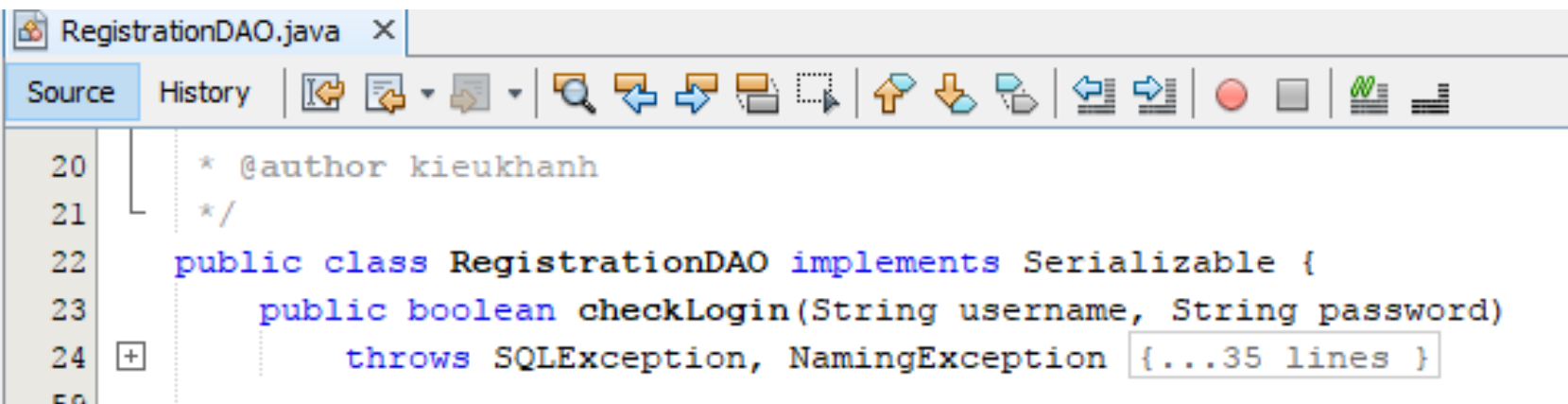
```jsp
4          Author      : kieukhanh
5   --%>
6
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <%@taglib uri="/struts-tags" prefix="s"%>
9   <!DOCTYPE html>
10  <html>
11      <head>
12          <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13          <title>Search</title>
14      </head>
15      <body>
16          <font color="red">
17          Welcome, <s:property value="%{#session.USERNAME}"/>
18          </font>
19          <h1>Search Page</h1>
```

EXAMPLE
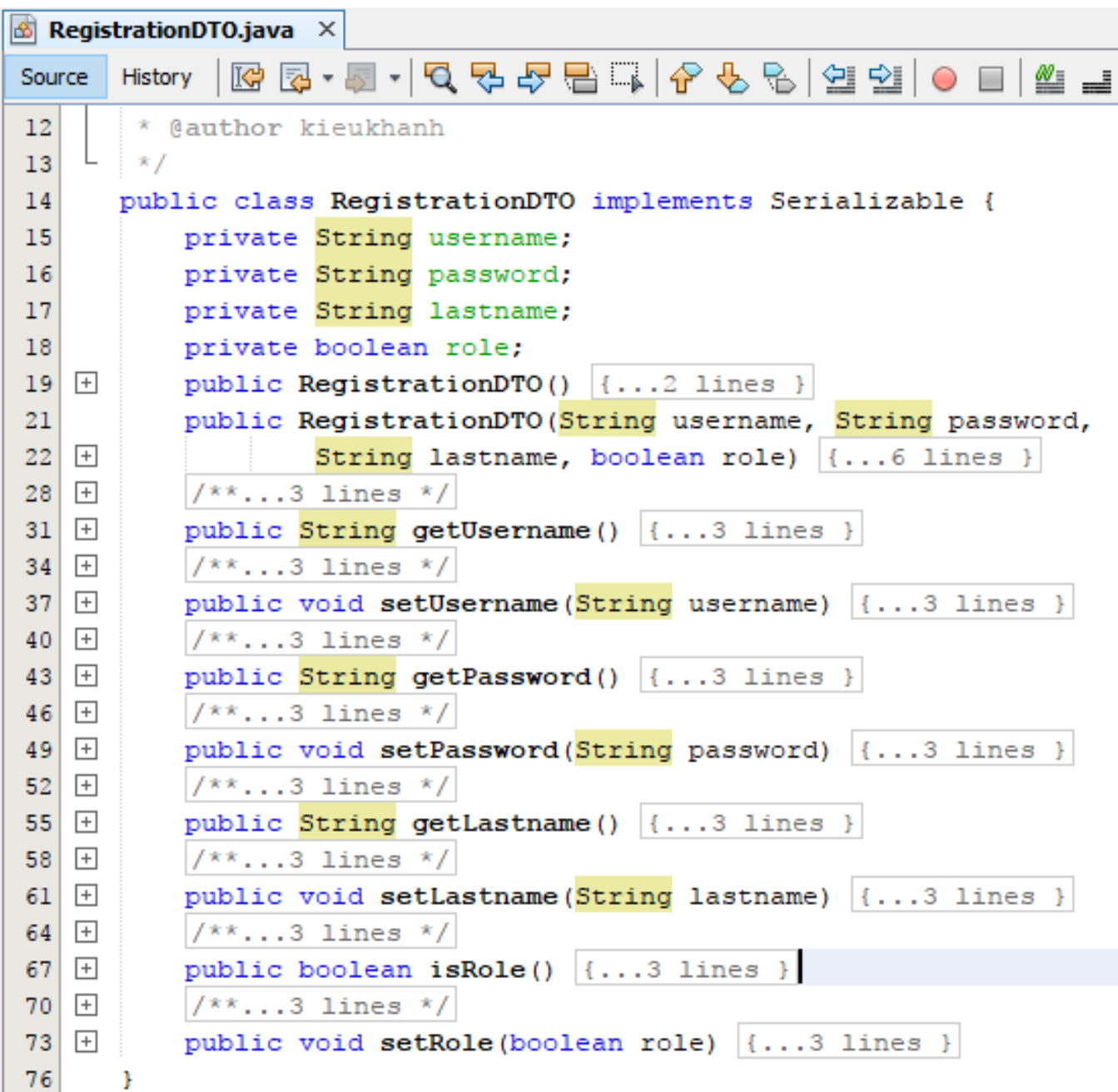
```
struts.xml  ×

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 name="">
9                <result>login.html</result>
10           </action>
11           <action name="login" class="sample.struts2.LoginAction">
12               <result name="success">search.jsp</result>
13               <result name="fail">invalid.html</result>
14           </action>
```

EXAMPLE
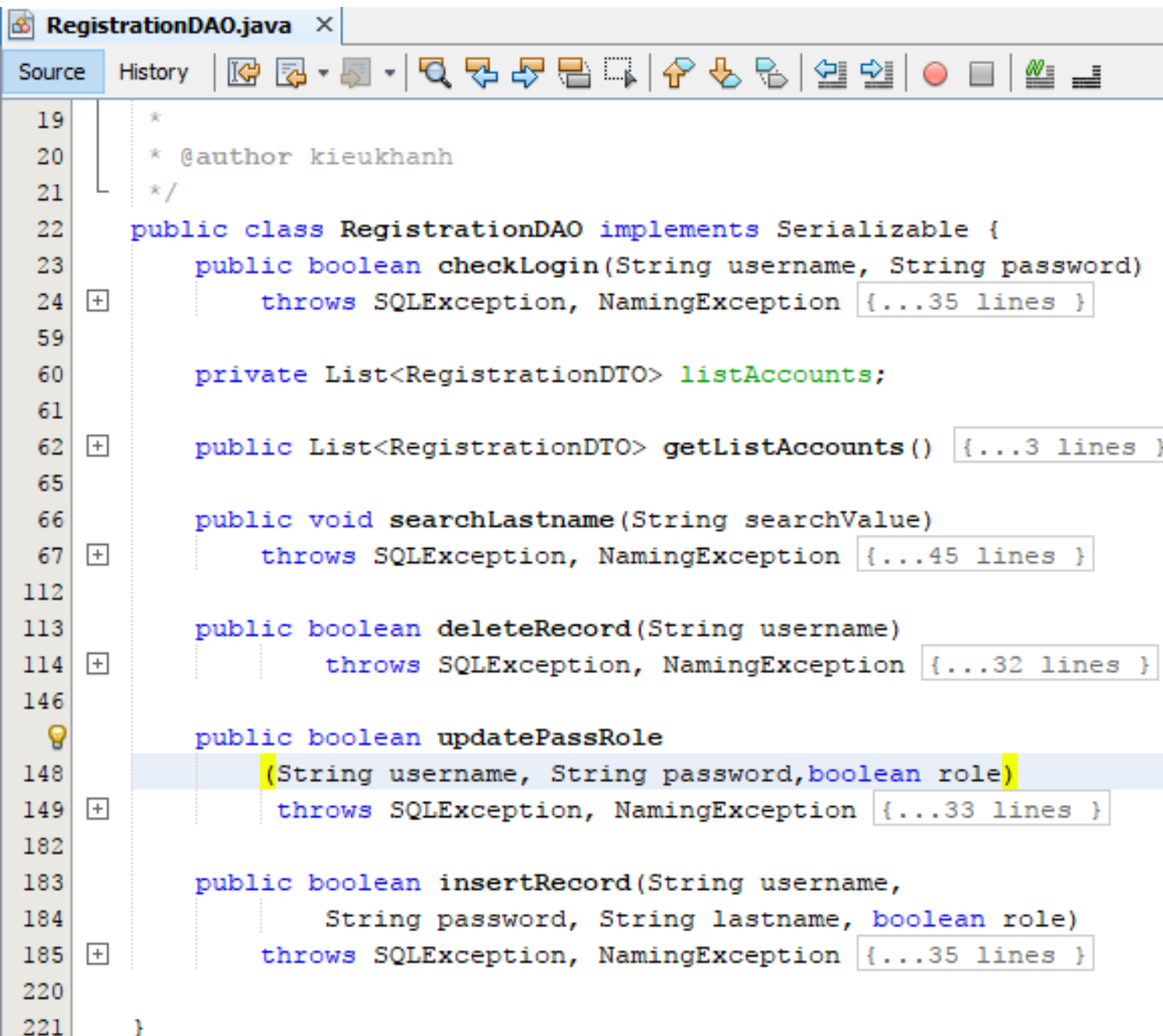
Source | History

```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>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="searchLikeLastname">
21               <s:textfield name="searchValue" label="Search Value"/><br/>
22               <s:submit value="Search"/>
23           </s:form><br/>
```
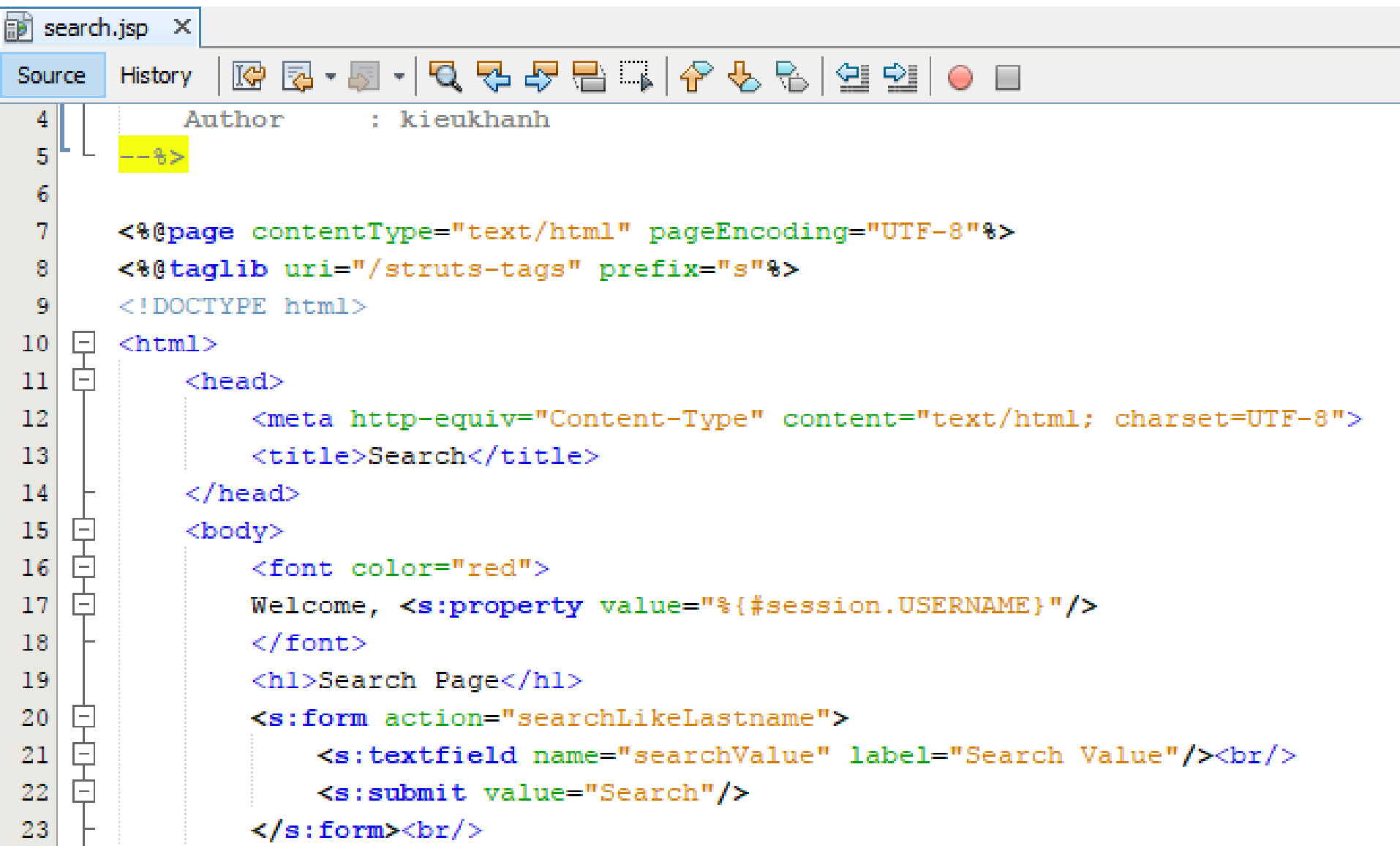
EXAMPLE

Source History
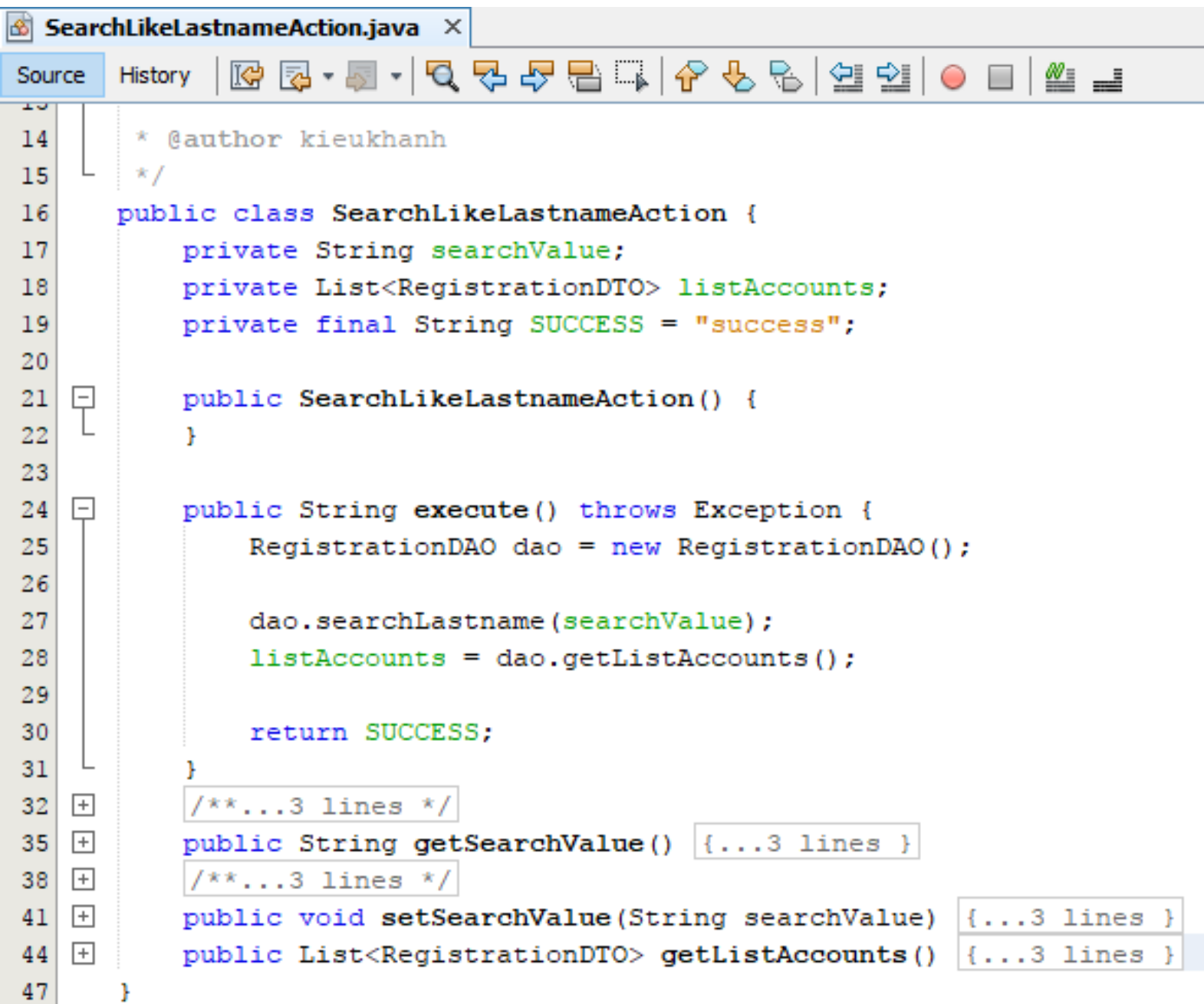
```java
12      * @author kieukhanh
13      */
14     public class RegistrationDTO implements Serializable {
15         private String username;
16         private String password;
17         private String lastname;
18         private boolean role;
19         public RegistrationDTO() {...2 lines }
21         public RegistrationDTO(String username, String password,
22                 String lastname, boolean role) {...6 lines }
28     /**...3 lines */
31     public String getUsername() {...3 lines }
34     /**...3 lines */
37     public void setUsername(String username) {...3 lines }
40     /**...3 lines */
43     public String getPassword() {...3 lines }
46     /**...3 lines */
49     public void setPassword(String password) {...3 lines }
52     /**...3 lines */
55     public String getLastname() {...3 lines }
58     /**...3 lines */
61     public void setLastname(String lastname) {...3 lines }
64     /**...3 lines */
67     public boolean isRole() {...3 lines }
70     /**...3 lines */
73     public void setRole(boolean role) {...3 lines }
76     }
```

EXAMPLE

Source | History

```java
 19       *
 20       * @author kieukhanh
 21       */
 22      public class RegistrationDAO implements Serializable {
 23          public boolean checkLogin(String username, String password)
 24              throws SQLException, NamingException {...35 lines }
 59
 60          private List<RegistrationDTO> listAccounts;
 61
 62          public List<RegistrationDTO> getListAccounts() {...3 lines }
 65
 66          public void searchLastname(String searchValue)
 67              throws SQLException, NamingException {...45 lines }
112
113          public boolean deleteRecord(String username)
114              throws SQLException, NamingException {...32 lines }
146
              public boolean updatePassRole
148              (String username, String password,boolean role)
149              throws SQLException, NamingException {...33 lines }
182
183          public boolean insertRecord(String username,
184              String password, String lastname, boolean role)
185              throws SQLException, NamingException {...35 lines }
220
221      }
```

EXAMPLE

Source   History

```java
13
14        * @author kieukhanh
15        */
16       public class SearchLikeLastnameAction {
17           private String searchValue;
18           private List<RegistrationDTO> listAccounts;
19           private final String SUCCESS = "success";
20
21           public SearchLikeLastnameAction() {
22           }
23
24           public String execute() throws Exception {
25               RegistrationDAO dao = new RegistrationDAO();
26
27               dao.searchLastname(searchValue);
28               listAccounts = dao.getListAccounts();
29
30               return SUCCESS;
31           }
32           /**...3 lines */
35           public String getSearchValue() {...3 lines }
38           /**...3 lines */
41           public void setSearchValue(String searchValue) {...3 lines }
44           public List<RegistrationDTO> getListAccounts() {...3 lines }
47       }
```

EXAMPLE

```jsp
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="updateRecord" 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>
51                                  <td>
52                                      <%--<s:property value="password"/>--%>
53                                      <s:textfield name="password" value="%{password}"/>
54                                  </td>
```

EXAMPLE

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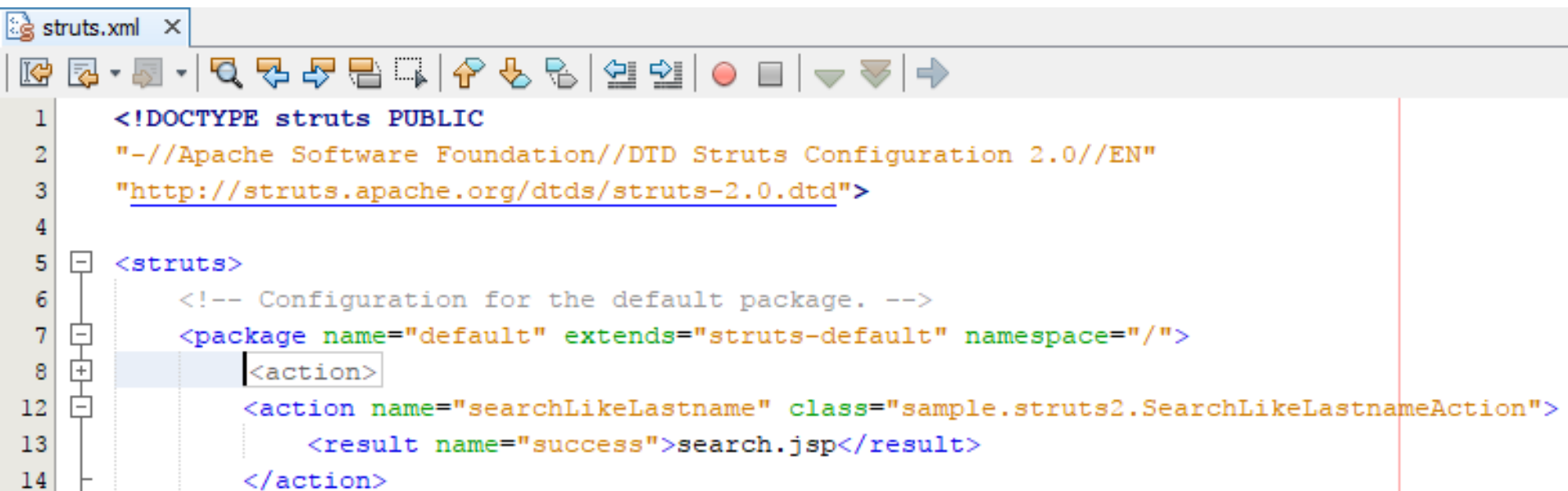
EXAMPLE

```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>
12          <action name="searchLikeLastname" class="sample.struts2.SearchLikeLastnameAction">
13              <result name="success">search.jsp</result>
14          </action>
```
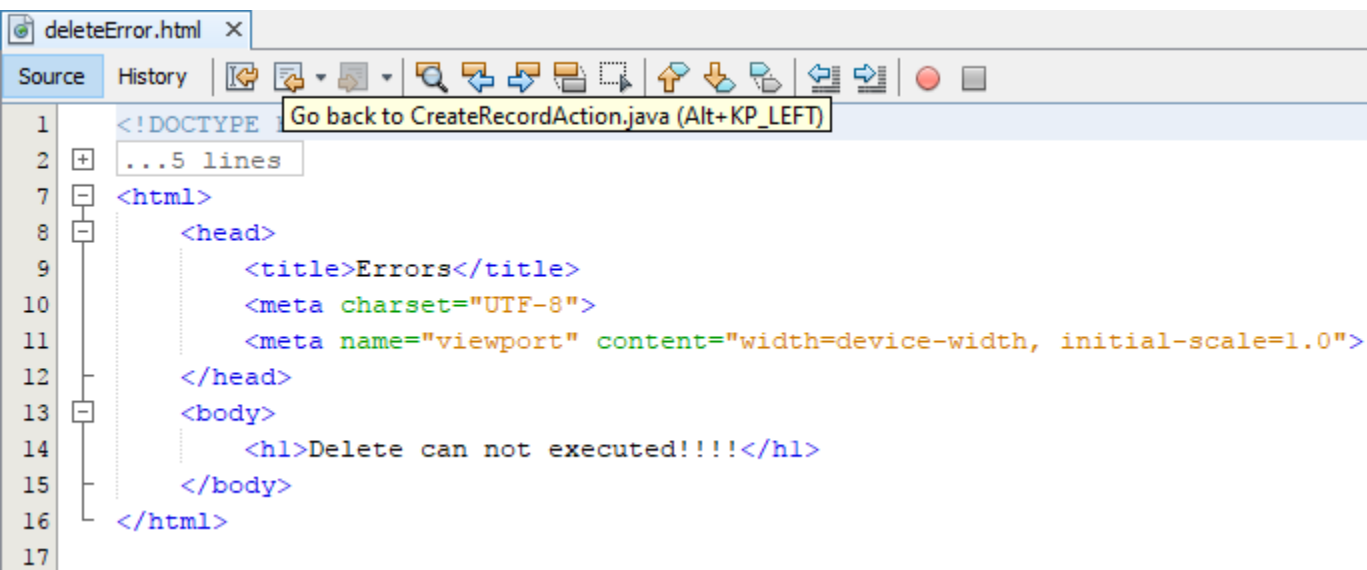
Go back to CreateRecordAction.java (Alt+KP_LEFT)

```html
1   <!DOCTYPE
2   ...5 lines
7   <html>
8       <head>
9           <title>Errors</title>
10          <meta charset="UTF-8">
11          <meta name="viewport" content="width=device-width, initial-scale=1.0">
12      </head>
13      <body>
14          <h1>Delete can not executed!!!!</h1>
15      </body>
16  </html>
17
```

EXAMPLE
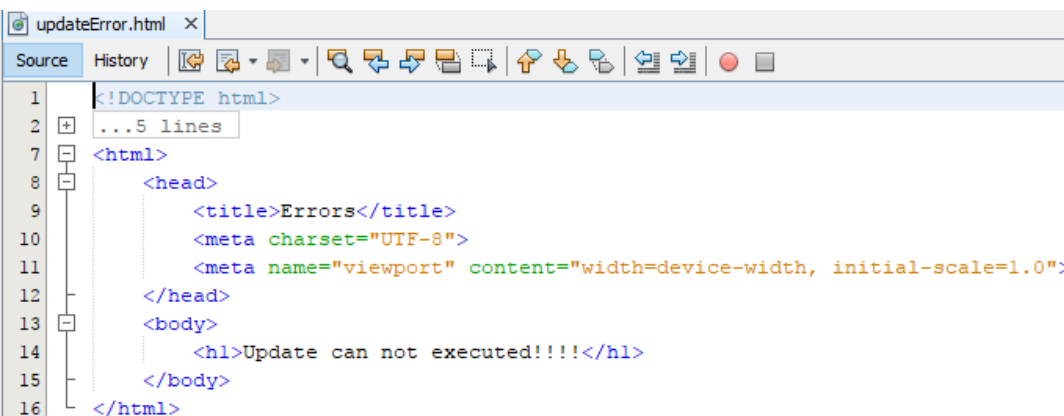

```java
 * @author kieukhanh
 */
public class DeleteRecordAction {
    private String pk;
    private String lastSearchValue;
    private final String FAIL = "fail";
    private final String SUCCESS = "success";
    public DeleteRecordAction() {...2 lines }
    public String execute() throws Exception {
        RegistrationDAO dao = new RegistrationDAO();

        boolean result = dao.deleteRecord(pk);
        String url = FAIL;

        if (result) {
            url = SUCCESS;
        }

        return url;
    }
    /**...3 lines */
    public String getPk() {...3 lines }
    /**...3 lines */
    public void setPk(String pk) {...3 lines }
    /**...3 lines */
    public String getLastSearchValue() {...3 lines }
    /**...3 lines */
    public void setLastSearchValue(String lastSearchValue) {...3 lines }
}
```

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>
12            <action>
15            <action name="deleteRecord" class="sample.struts2.DeleteRecordAction">
16                <result name="success" type="redirectAction">
17                    <param name="actionName">searchLikeLastname</param>
18                    <param name="searchValue">${lastSearchValue}</param>
19                </result>
20                <result name="fail">deleteError.html</result>
21            </action>
```
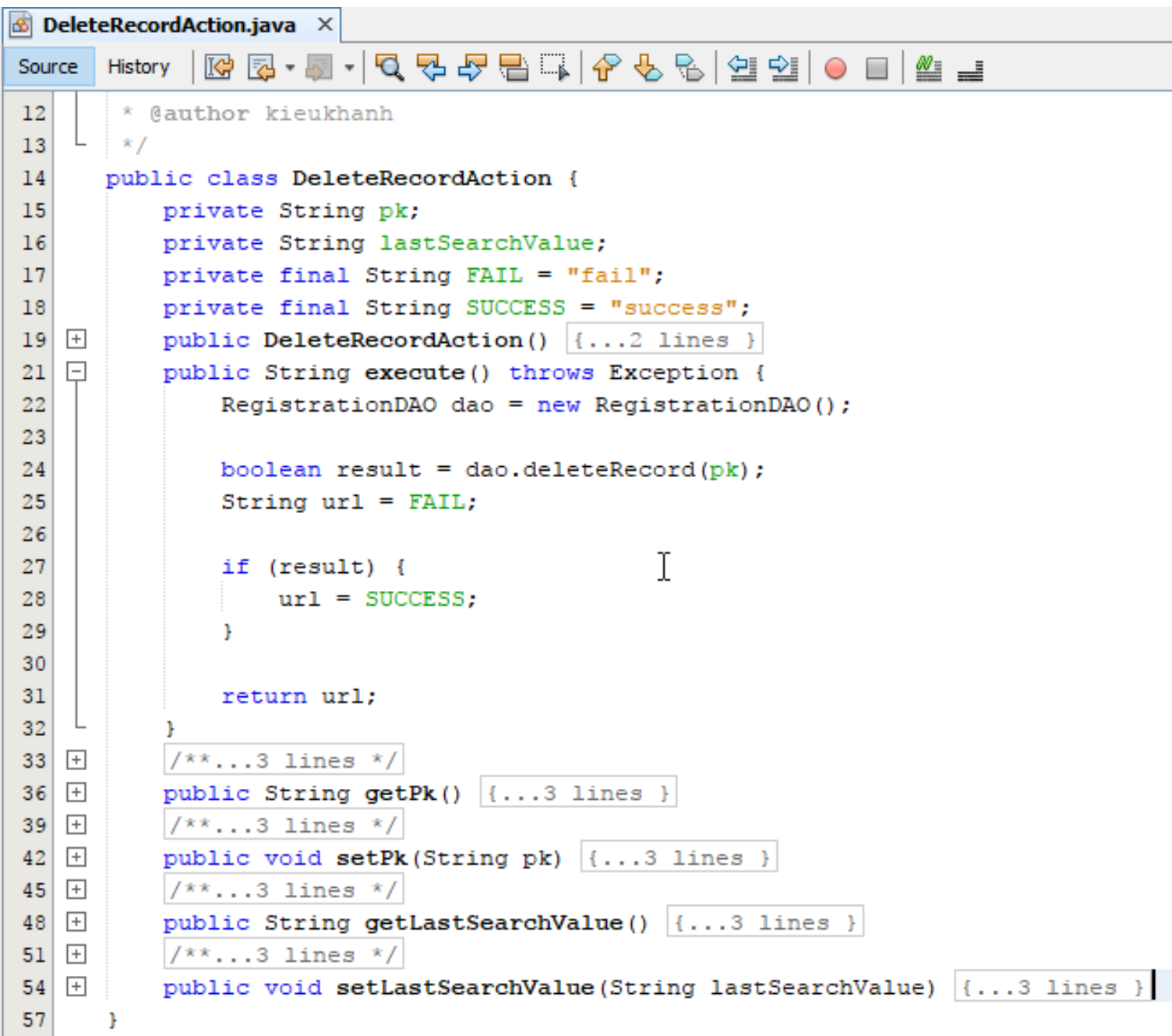
```
updateError.html  ×
Source  History

 1   <!DOCTYPE html>
 2   ...5 lines
 7   <html>
 8       <head>
 9           <title>Errors</title>
10           <meta charset="UTF-8">
11           <meta name="viewport" content="width=device-width, initial-scale=1.0">
12       </head>
13       <body>
14           <h1>Update can not executed!!!!</h1>
15       </body>
16   </html>
```
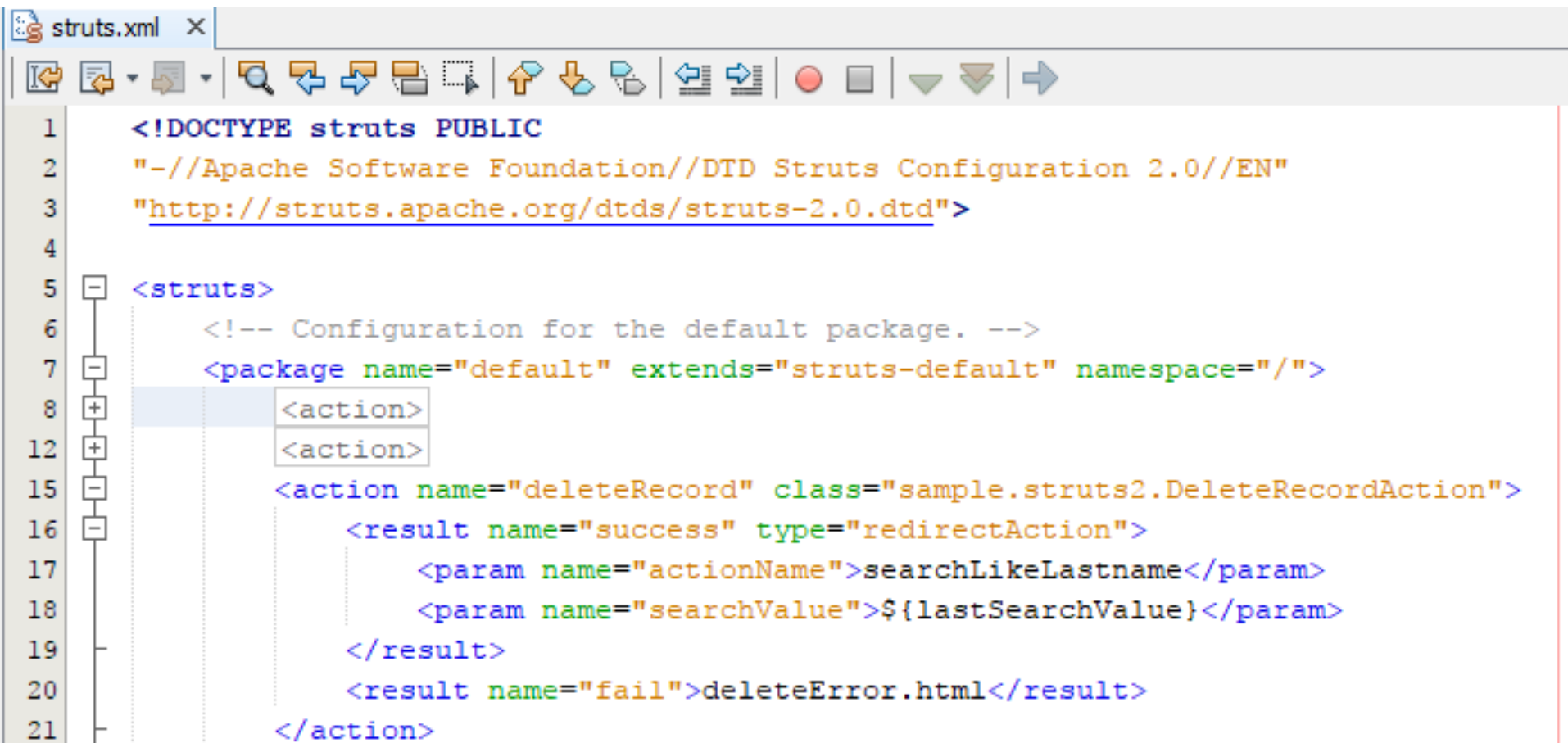
EXAMPLE

Source | History

```java
12       * @author kieukhanh
13       */
14      public class UpdateRecordAction {
15          private String username;
16          private String password;
17          private boolean role;
18          private String lastSearchValue;
19          private final String FAIL = "fail";
20          private final String SUCCESS = "success";
21          public UpdateRecordAction() {...2 lines }
23          public String execute() throws Exception {
24              RegistrationDAO dao = new RegistrationDAO();
25              boolean result = dao.updatePassRole(username, password, role);
26              String url = FAIL;
27              if (result) {
28                  url = SUCCESS;
29              }
30              return url;
31          }
32          /**...3 lines */
35          public String getUsername() {...3 lines }
38          /**...3 lines */
41          public void setUsername(String username) {...3 lines }
44          /**...3 lines */
47          public String getPassword() {...3 lines }
50          /**...3 lines */
53          public void setPassword(String password) {...3 lines }
56          /**...3 lines */
59          public boolean isRole() {...3 lines }
62          /**...3 lines */
65          public void setRole(boolean role) {...3 lines }
```

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>
12         <action>
15         <action>
22         <action name="updateRecord" class="sample.struts2.UpdateRecordAction">
23             <result name="success" type="redirectAction">
24                 <param name="actionName">searchLikeLastname</param>
25                 <param name="searchValue">${lastSearchValue}</param>
26             </result>
27             <result name="fail">updateError.html</result>
28         </action>
```
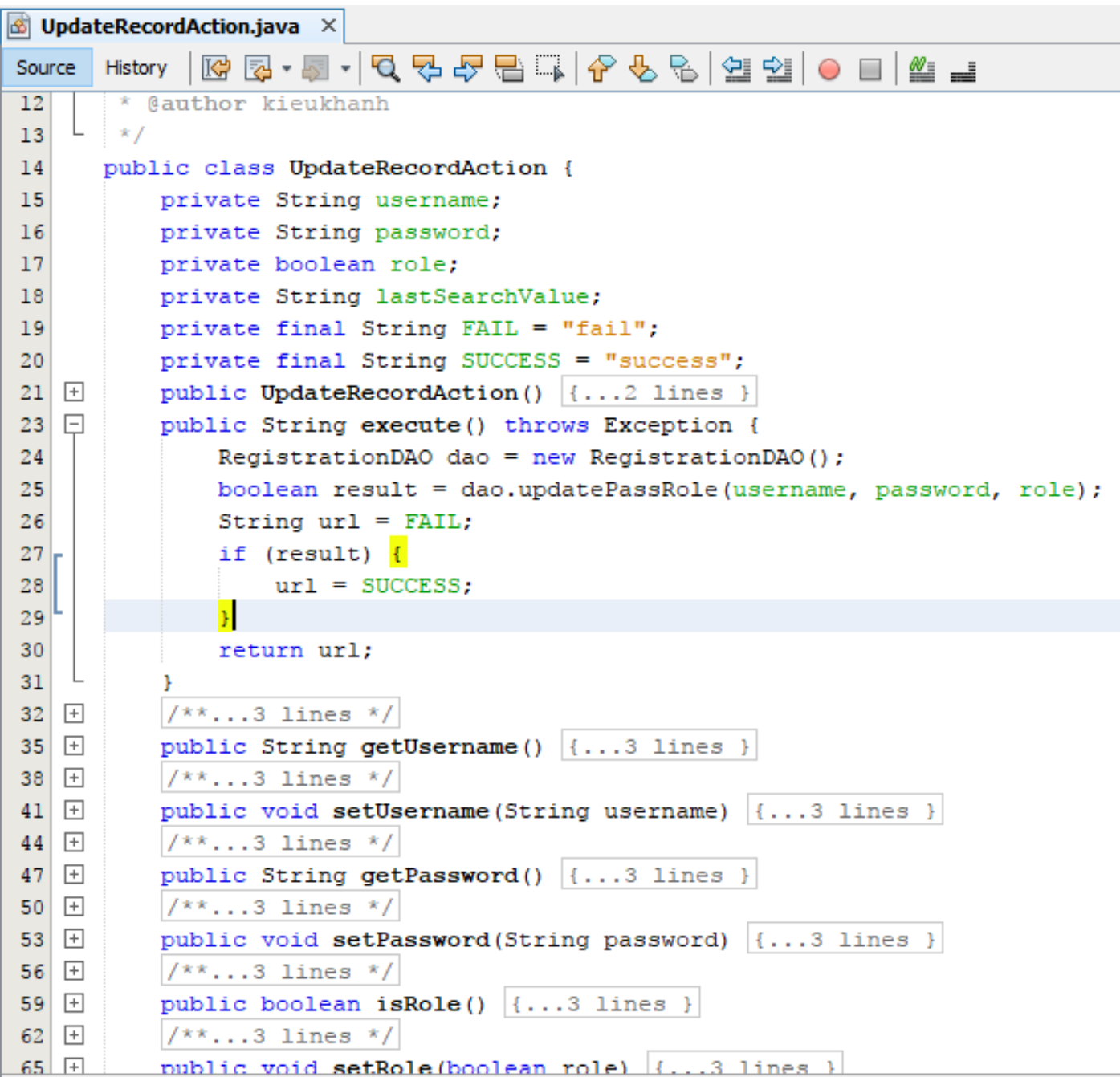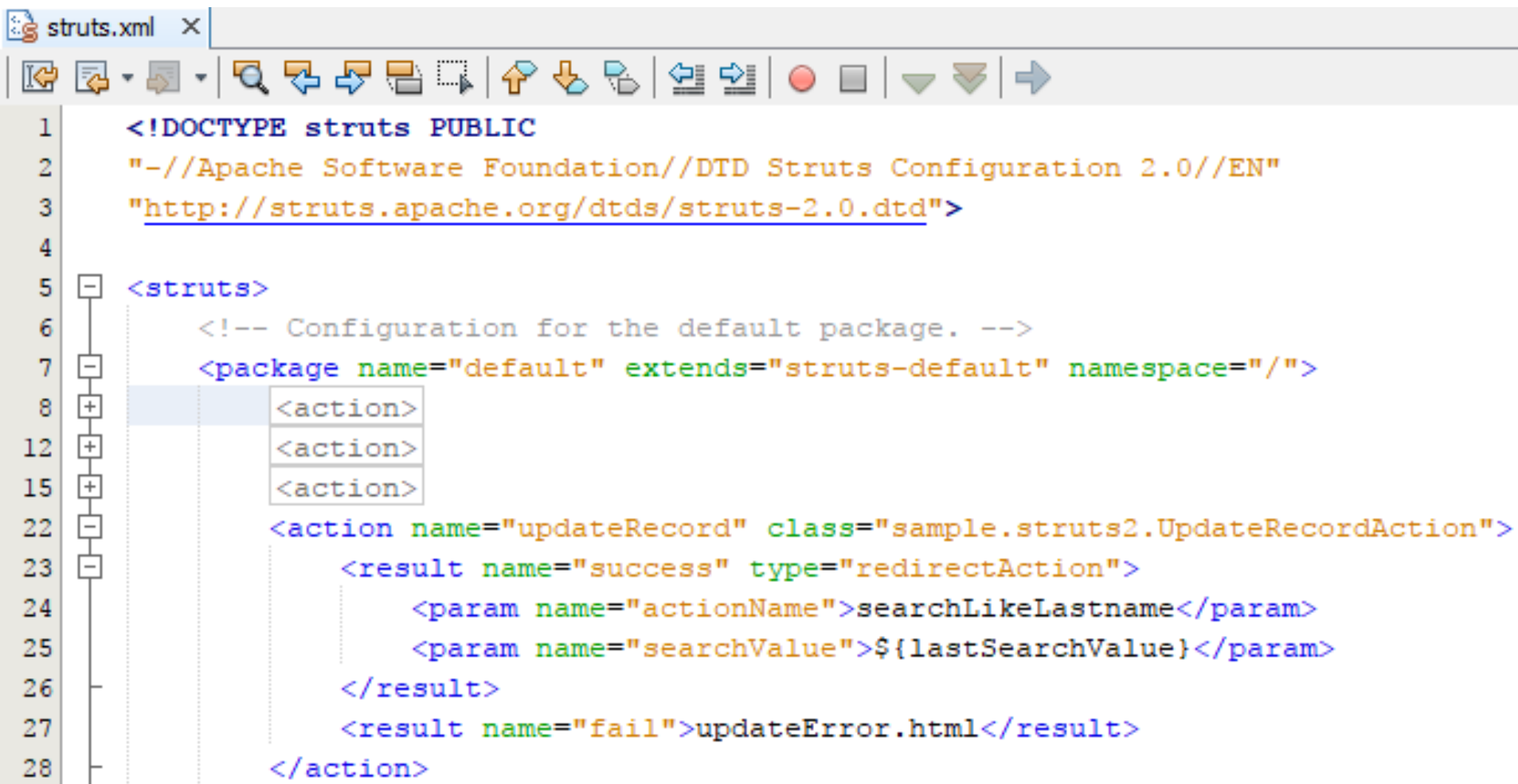
EXAMPLE

```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>
12          <action>
15          <action>
22          <action name="updateRecord" class="sample.struts2.UpdateRecordAction">
23              <result name="success" type="redirectAction">
24                  <param name="actionName">searchLikeLastname</param>
25                  <param name="searchValue">${lastSearchValue}</param>
26              </result>
27              <result name="fail">updateError.html</result>
28          </action>
```
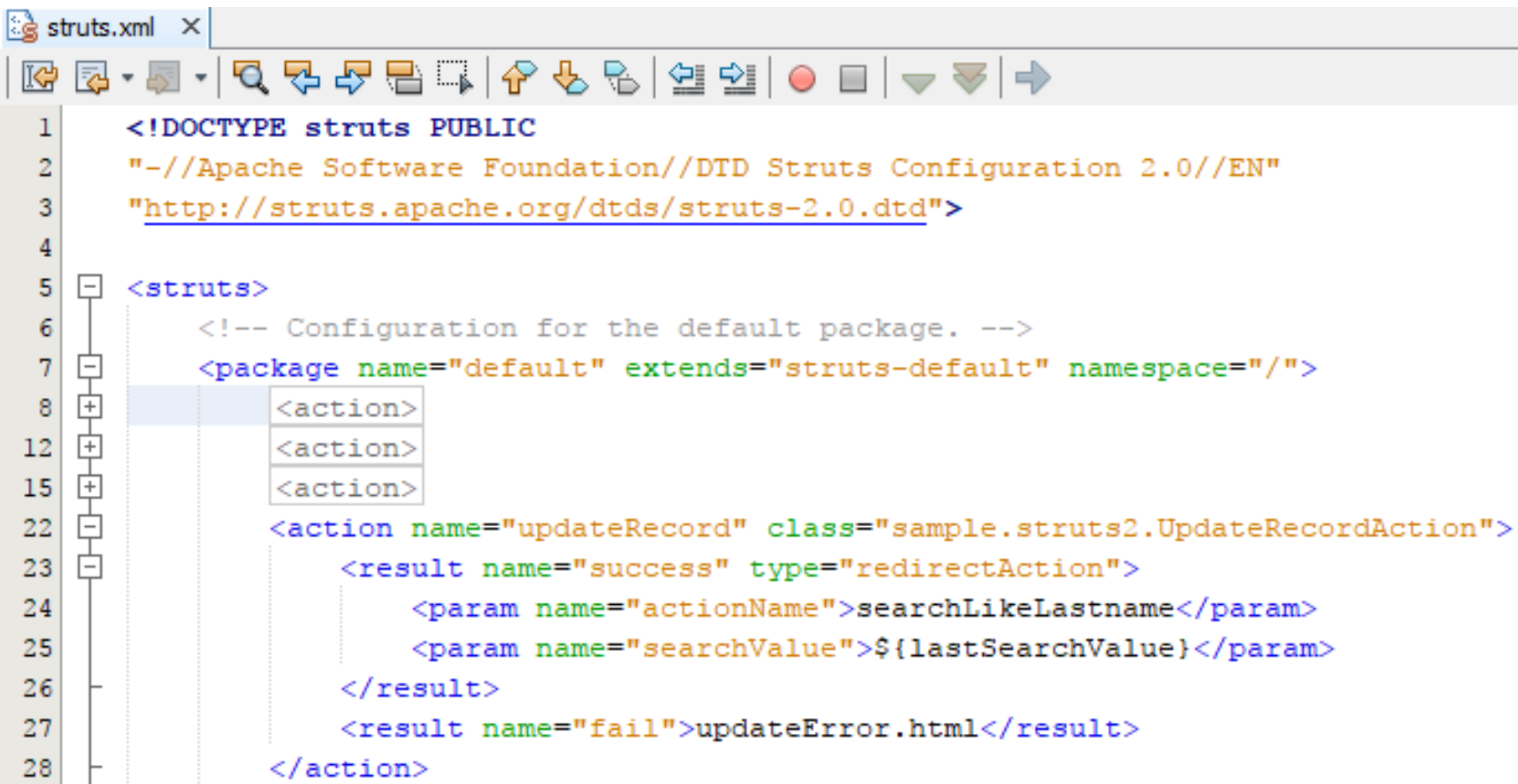
EXAMPLE

Projects

- Strut2Config ⚠
  - Web Pages
    - META-INF
      - context.xml
    - WEB-INF
      - web.xml
    - deleteError.html
    - index.jsp
    - invalid.html
    - login.html
    - search.jsp
    - updateError.html
  - Source Packages
    - <default package>
      - struts.xml
    - sample.registration
      - RegistrationDAO.java
      - RegistrationDTO.java
    - sample.struts2
      - DeleteRecordAction.java
      - LoginAction.java
      - SearchLikeLastnameAction.java
      - UpdateRecordAction.java
    - sample.utils
      - DBUtils.java
  - Test Packages
  - Libraries
  - Test Libraries
  - Configuration Files