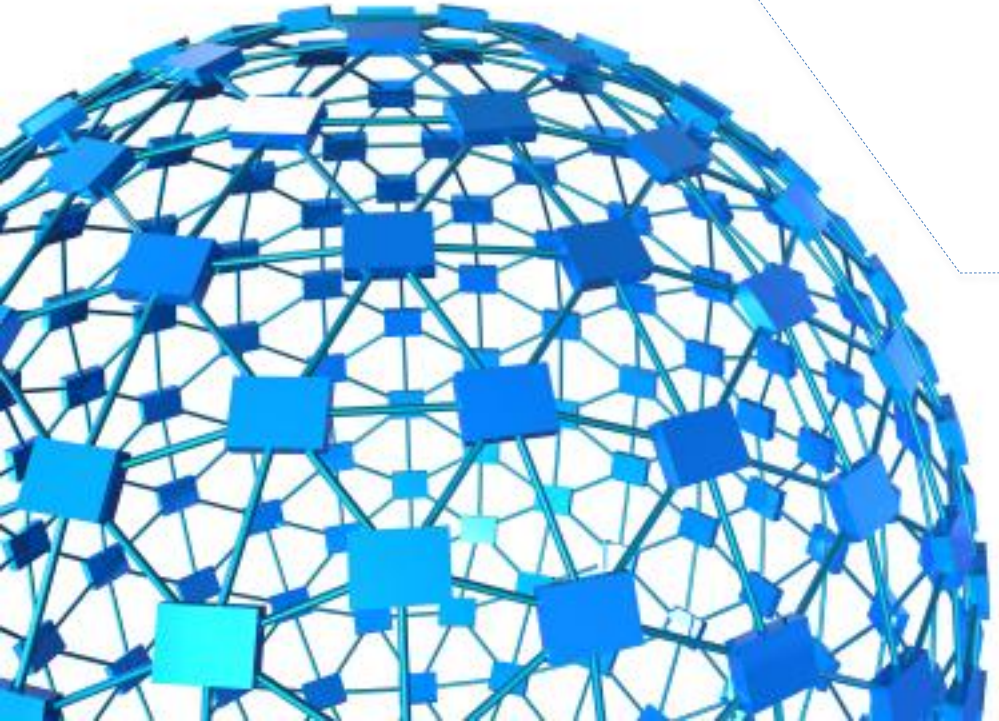
Write Once, Run Anywhere

SERVLET/JSP



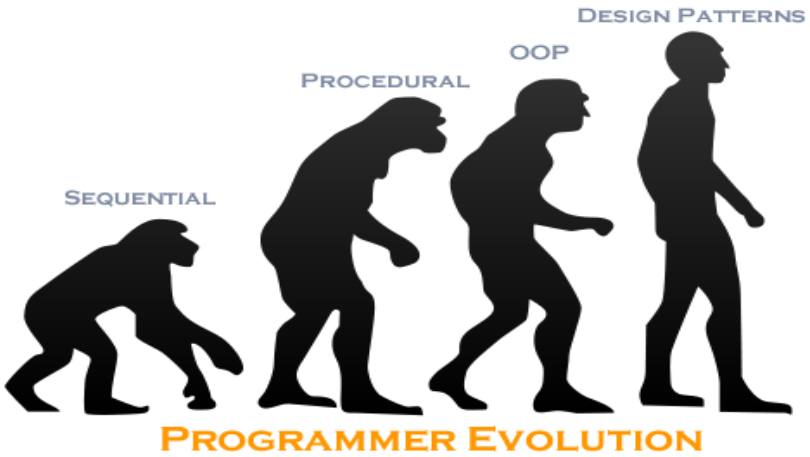
STRUTS 2X

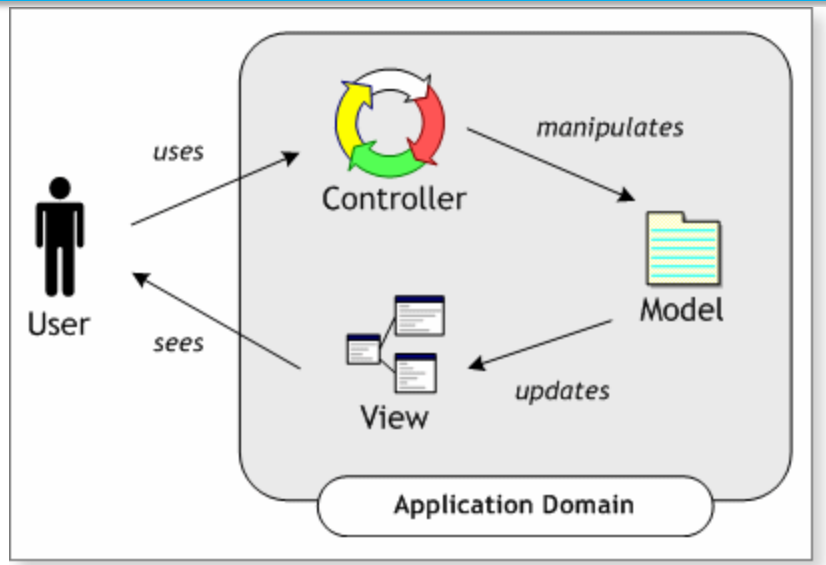
ThS. Nguyễn Nghiệm
0913.745.789
nnghiem@yahoo.com

Nội dung trình bày

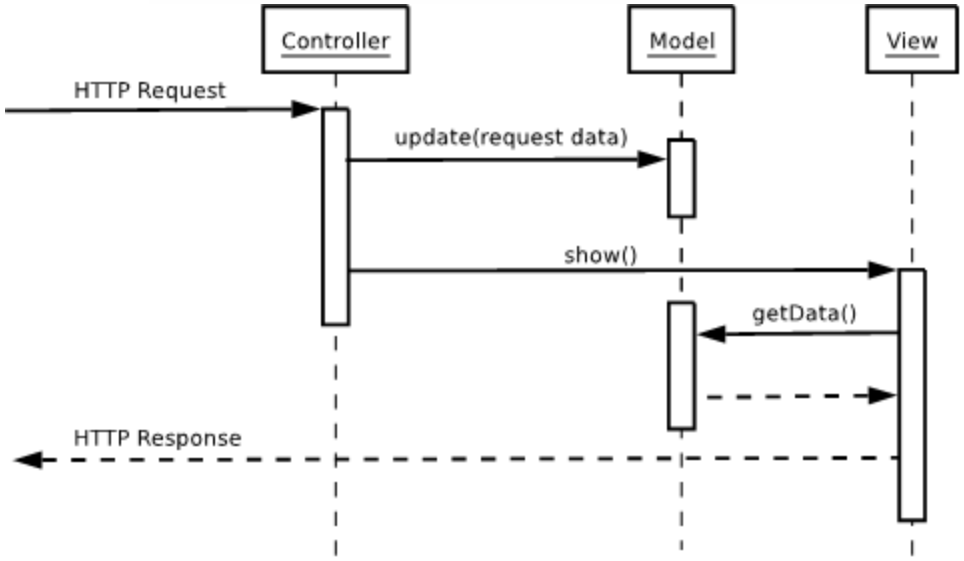
- ☆Mô hình lập trình MVC
- ☆Kiến trúc tổ chức của Struts2
- ☆Hello World
- ☆ Actions
- ☆Cấu hình struts.xml
- ☆ModelDriven
- ☆Struts2 Annotation
- ☆Thẻ struts2
- ☆ Kiểm lỗi
- ☆ Tiles framework
- ☆ Quốc tế hóa website
- ☆Chia sẽ dữ liệu

Kiến trúc mô hình MVC



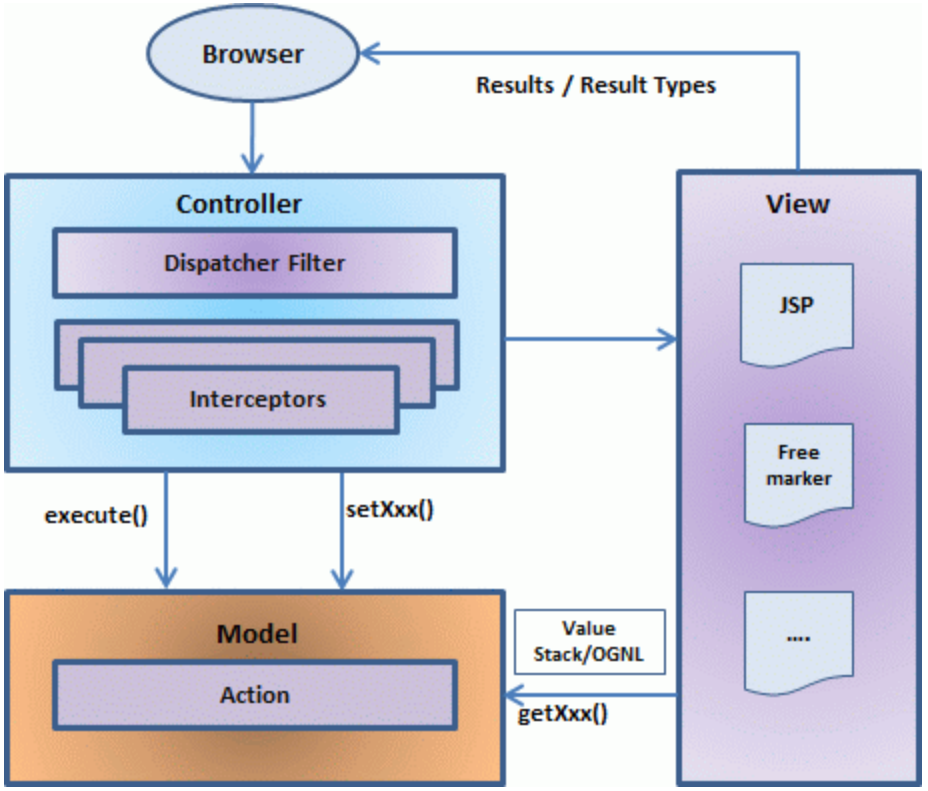


- MVC Pattern
  - Model
  - Controller
  - View
- MVC Frameworks
  - Java: Struts, JSF, Spring…
  - PHP: Zend
  - MS.NET: MVC

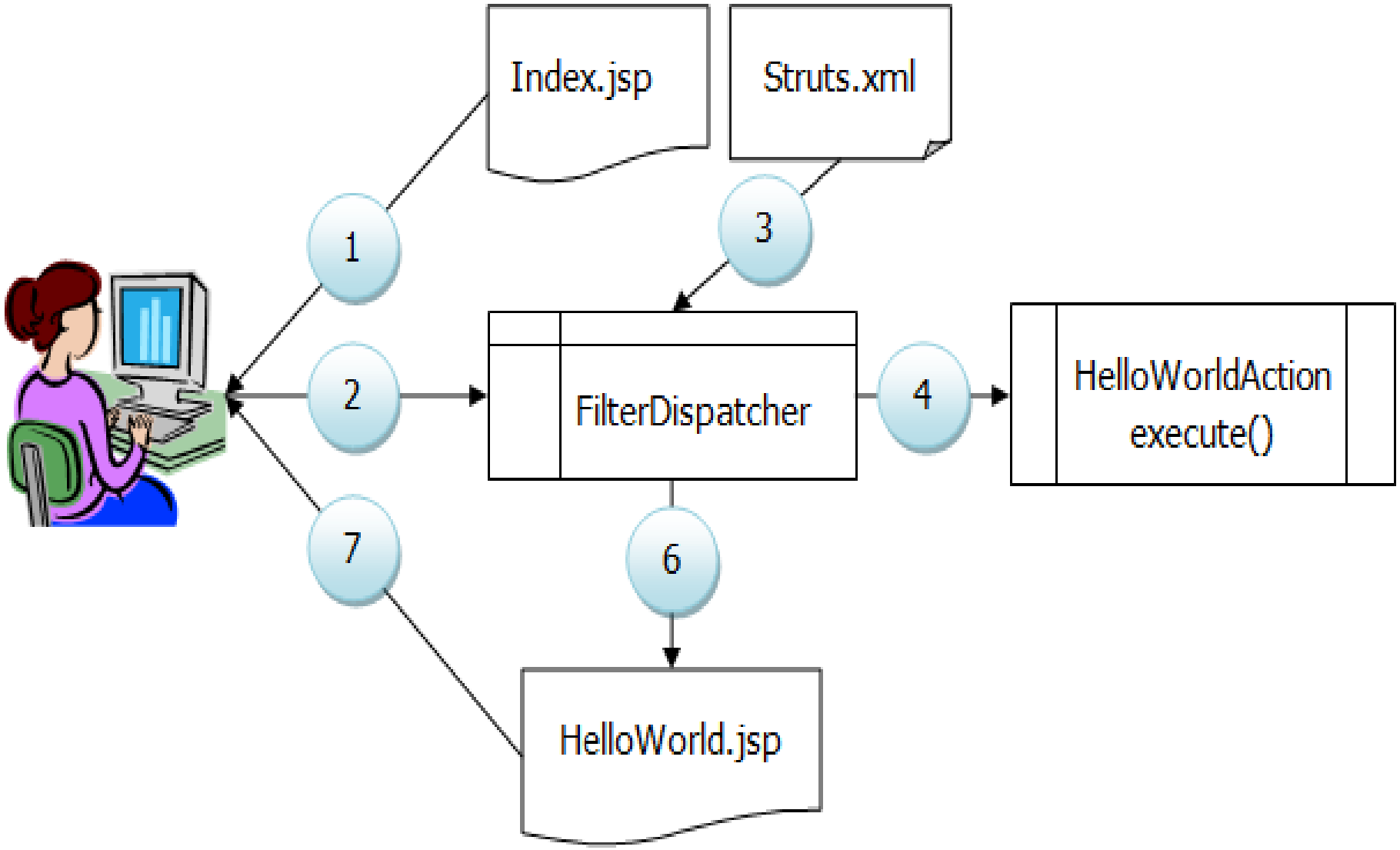


Kiến trúc tổ chức Struts 2

- Struts là công nghệ Java (WORAW)
- Struts2 là MVC Framework
  - Model
    - **Action**
  - Controller
    - Dispatcher Filter (bộ lọc)
    - **Interceptors** (lá chắn)
  - View
    - **JSPs**



HelloWorld

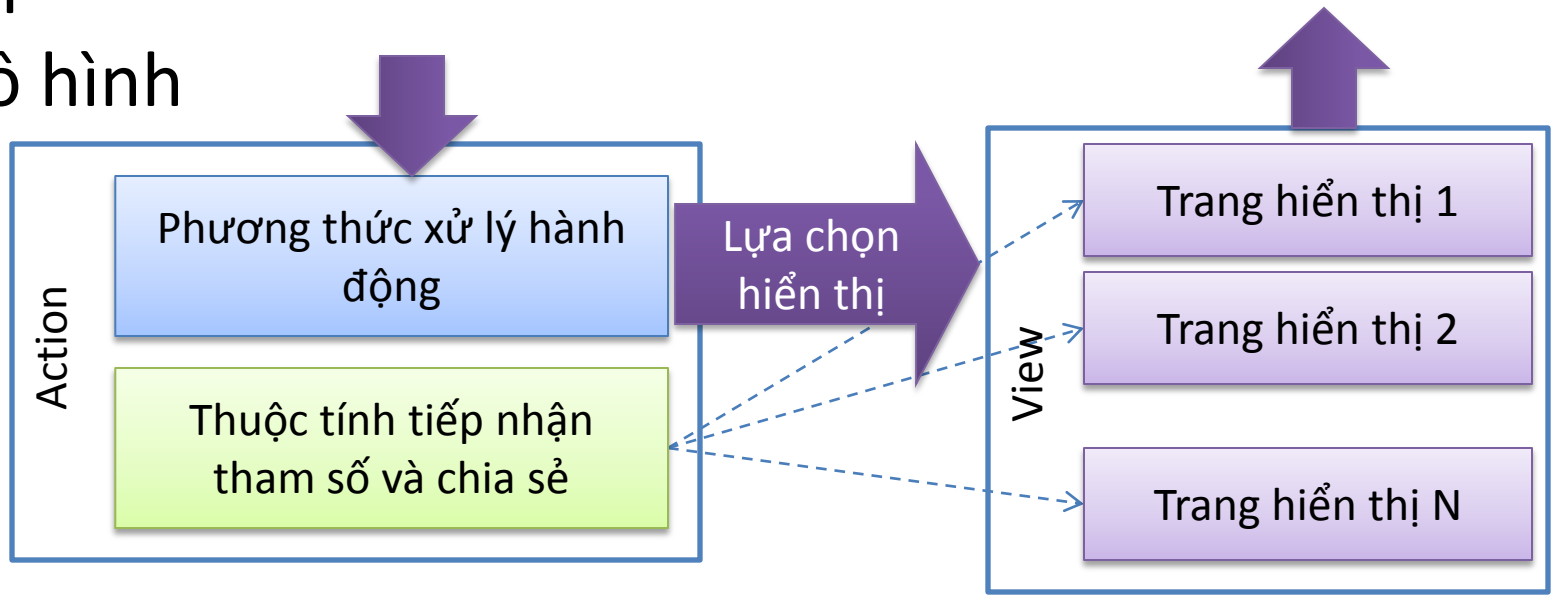

Index.jsp
Struts.xml
1
3
2
FilterDispatcher
4
HelloWorldAction
execute()
7
6
HelloWorld.jsp

Các bước thực hiện

- Bước 1: Tạo Dynamic Web Project
  - Bổ sung thư viện
- Bước 2: Tạo lớp Action (HelloWorldAction.java)
- Bước 3: Tạo View (HelloWorld.jsp)
- Bước 4: Tạo form nhập (index.jsp)
- Bước 5: Cấu hình
  - Tập tin cấu hình struts.xml
  - Tập tin cấu hình web.xml
- Bước 6: Chạy thử trong eclipse

Lớp action

- Là thành phần quan trọng nhất của Struts2
  - Tiếp nhận dữ liệu từ người dùng
  - Xử lý các hành động tương tác của người dùng
- Một Action phải có tối thiểu một phương thức xử lý hành động người dùng và (có hoặc không) các thuộc tính (get/set) để tiếp nhận tham số cùng tên
- Mô hình



Lớp action

```
package nnghiem.struts;

import com.opensymphony.xwork2.ActionSupport;

public class <tên lớp hành động> extends ActionSupport{

	<định nghĩa các phương thức hành động và danh sách các thuộc tính>
}
```

```
public String <tên phương thức action>() throws Exception{
	...thực hiện công việc...
	return <giá trị kết quả>;
}
```

```
int age;
public int getAge(){
	return age;
}
public void setAge(int age){
	this.age = age;
}
```

Ví dụ lớp action

```
public class MemberAction extends ActionSupport{

	public String input() throws Exception {
		return "input";
	}

	public String signIn() throws Exception {
		if(userId.equals("user") && password.equals("123") ){
			return "success";
		}
		return "error";
	}

	String userId, password;

	public String getUserId() {
		return userId;
	}
	public void setUserId(String userId) {
		this.userId = userId;
	}

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

Action đơn giản

```
package nnghiem.struts.acti
import com.opensymphony.xw

public class WelcomeAction ex      ActionSupport{

    public String chao() throws Exception {
        // mã xử lý hành động
        return "hienthi";
    }
    // thuộc tính tiếp nhận tham số và chia sẻ với View
    private String ten;
    public String getTen() {return ten;}
    public void setTen(String ten) {this.ten = ten;}
}
```

Phương thức xử lý hành động do người dùng tương tác (nút, liên kết)

Lựa chọn View để hiển thị

Theo qui ước của javabean (set/get)

Cấu hình struts.xml

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

<struts>
	<package name="default" extends="default">

	<action name="viewLogin" method="input" class="nnghiem.struts.MemberAction">
		<result name="input">/LoginInput.jsp</result>
	</action>

	<action name="login" method="signIn" class="nnghiem.struts.MemberAction">
		<result name="error">/LoginError.jsp</result>
		<result name="success">/MyAccount.jsp</result>
	</action>

	</package>

</struts>
```

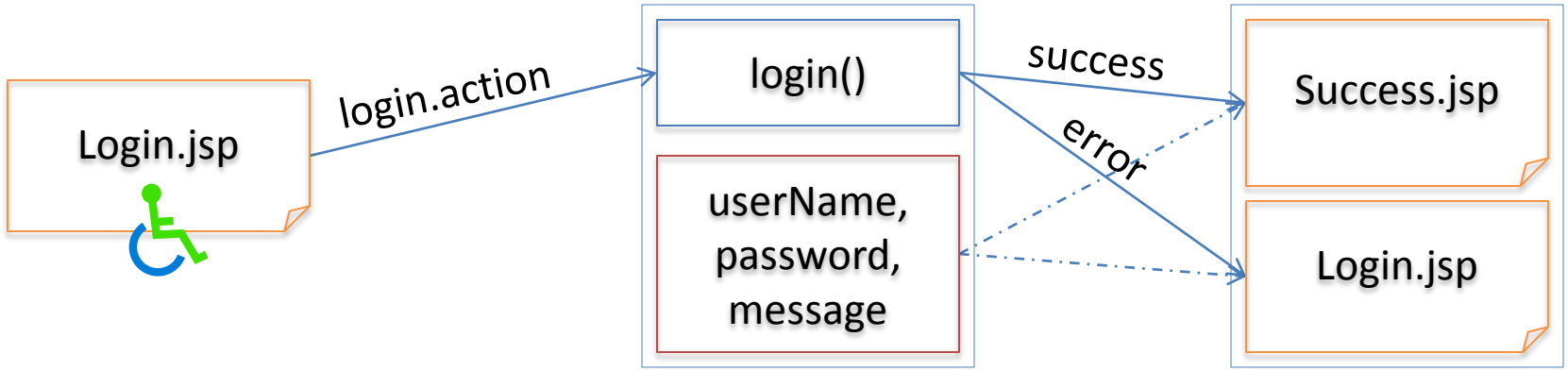
Ánh xạ action động

```
<struts>
   <package name="default" extends="struts-default">

      <action name="*Member" method="{1}"
                class="nnghiem.struts.MemberAction">
            <result name="input">/Input.jsp</result>
            <result name="error">/Error.jsp</result>
            <result name="success">/Success.jsp</result>
            <result name="login">/Login.jsp</result>
      </action>

   </package>
</struts>
```

☆***Tên phương thức action {1}*** tương ứng với *

☆Nếu * rỗng thì tên phương thức action là ***execute()***

Đề mô login



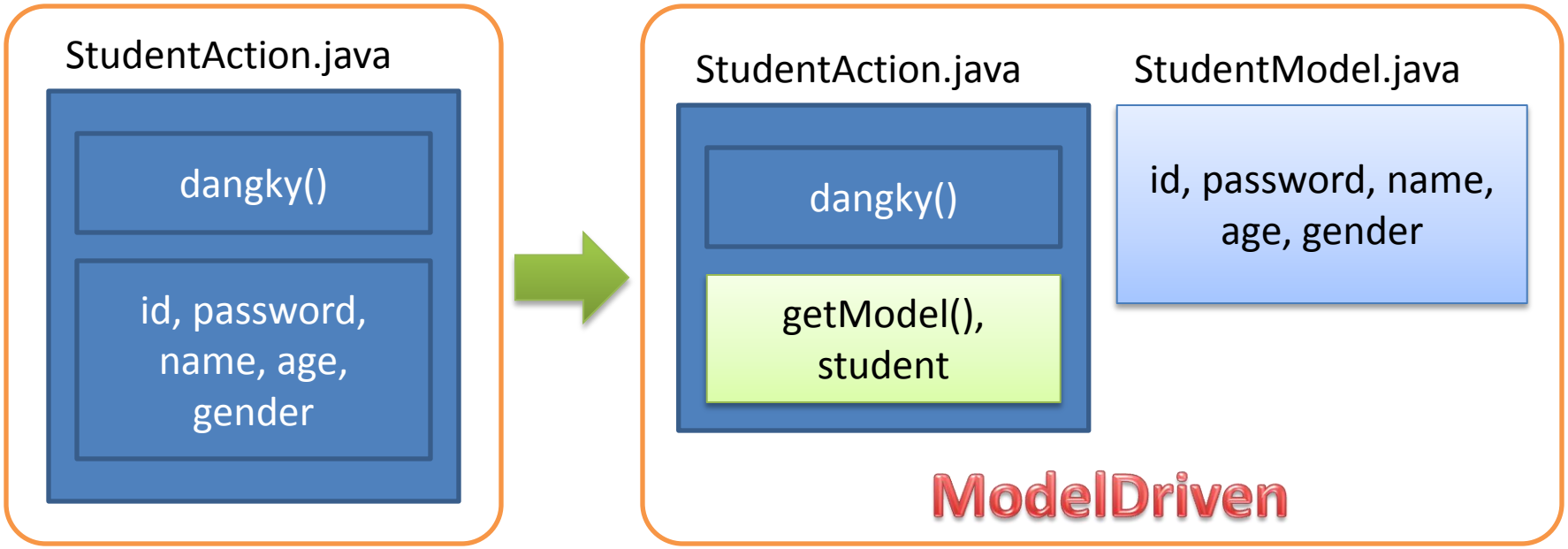
- **Login.jsp**: chứa form đăng nhập ->login.action
- **LoginAction.java**
  - Phương thức hành động **login**()
  - Thuộc tính **userName** và **password** để nhận tham số từ form; message chứa thông báo lỗi hiển thị trên trang Login.jsp khi đăng nhập sai.
- **Success.jsp**: hiển thị userName và password đã đăng nhập
- **Struts.xml**
  - Ánh xạ hành động (login.action-> LoginAction.login())

ModelDriven

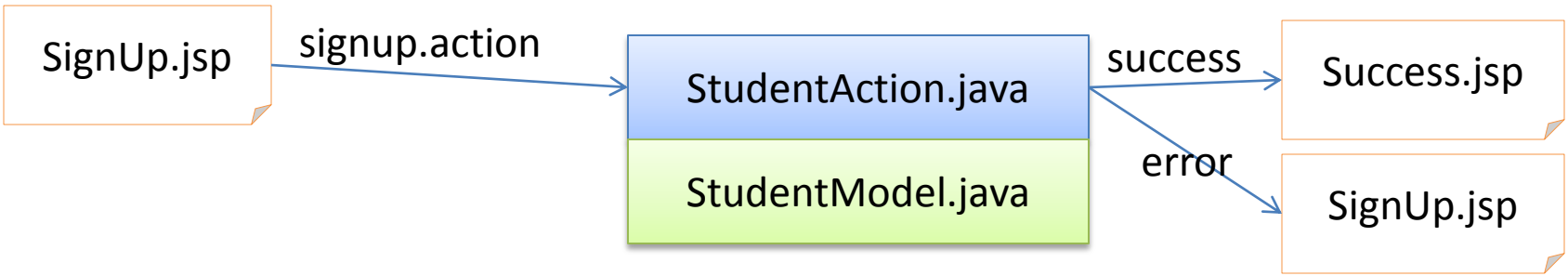
- Kỹ thuật tách mô hình dữ liệu ra khỏi Action
  - Rõ ràng, đơn giản và dễ quản lý
  - Tái sử dụng
- Thực hiện
  - Định nghĩa lớp Model chứa các thuộc tính cần tách rời.
  - Lớp Achtion phải implements interface DrivenModel<Model> và viết mã cho phương thức getModel() theo qui định của interface.
  - Định nghĩa thuộc tính model để dẫn xuất dữ liệu thuộc tính vào model.

```
public class MyAction
                    implements DrivenModel<Model> {
        public Model getModel(){...}
}
```

Đề mô ModelDriven



- Định nghĩa lớp **StudentModel** chứa các thuộc tính **id, password, name, age, gender**
- Viết mã cho **getModel()** theo qui định của interface **DrivenModel<StudentModel>**
- Định nghĩa thuộc tính student để có thể đọc ghi Model



Annotation

- Thay cho struts.xml
- Các annotation
  - @Results(result=Result[])
  - @Result(name, location, type, param)
  - @Action(value, results)

Annotation

```
@Results(value={
        @Result(name="input", location="/register.jsp"),
        @Result(name="error", location="/error.jsp", type="redirect"),
        @Result(name="login", location="/login.jsp")
})
public class MemberAction extends ActionSupport {
        /*
         * Đăng nhập
         */
        @Action(value="signIn",
                results={
                        @Result(name="account", location="/account.jsp")
                }
        )
        public String signIn() throws Exception {
                return "account";
        }
        /*
         * Đăng ký
         */
        @Action(value="signUp")
        public String signUp() throws Exception {
                return "login";
        }
```

Thư viện thẻ

- ☆Thẻ điều khiển
- ☆Thể giao diện
- ☆Thẻ dữ liệu

Thẻ điều khiển - <s:if>

```
<s:if test="điều kiện 1">
    Giao diện 1
</s:if>
<s:elseif test="điều khiện 2">
    Giao diện 2
</s:elseif>
<s:elseif test="điều khiện 3">
    Giao diện 3
</s:elseif>
<s:else>
    Giao diện N + 1
</s:else>
```

```
<!--&lt; là dấu nhỏ hơn, &gt; là dấu lớn hơn-->
<s:if test="#session.user != null">
    <li><a href="editProfile">Tài khoản</a></li>
    <li><a href="changePassword">Đổi mật khẩu</a></li>
</s:if>
<s:else>
    <li><a href="login">Đăng nhập</a></li>
    <li><a href="forgotPassword">Đổi mật khẩu</a></li>
    <li><a href="register">Đăng ký thành viên</a></li>
</s:else>
```

Thẻ điều khiển <s:iterate>

☆Tương tự <c:forEach> của JSTL

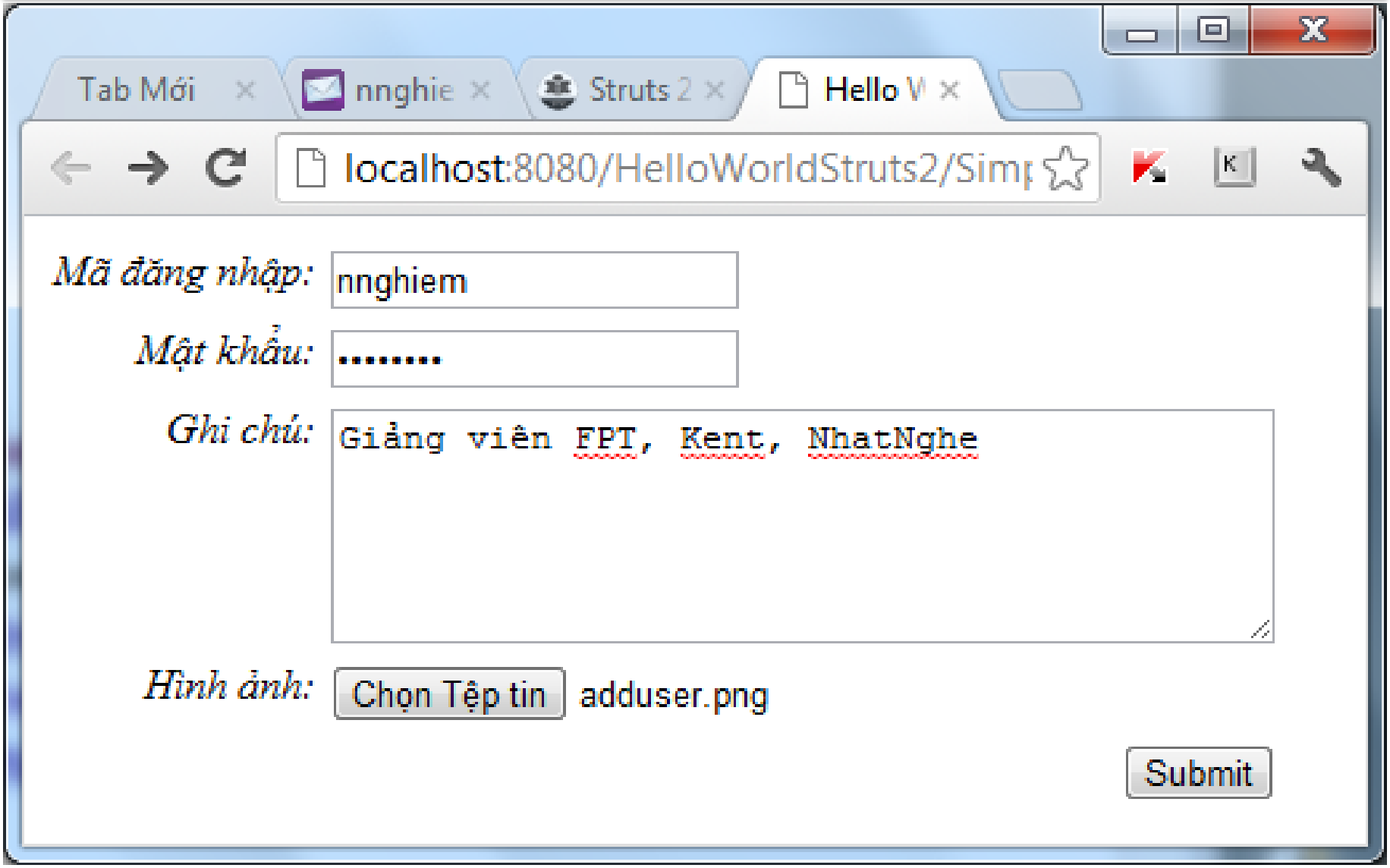
```
<s:iterate value="users">
<ul>
    <li>User Id: ${userId}</li>
    <li>Password: ${password}</li>
</ul>
</s:else>
```

Thẻ form đơn giản

```
<%@ page contentType="text/html; charset=utf8" pageEncoding="utf8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
	<s:head/>
	<title>Hello World</title>
</head>
<body>
<s:form action="hello" method="post" enctype="multipart/form-data">
	<s:textfield name="from" label="Mã đăng nhập"/>
	<s:password name="password" label="Mật khẩu"/>
	<s:textarea name="notes" label="Ghi chú"/>
	<s:file name="attachment" label="Hình ảnh"/>
	<s:hidden name="country" value="VN"/>
	<s:submit value="Submit" />
</s:form>
</body>

</html>
```

- ☆ Sử dụng thẻ struts để kết hợp với model tự động
  - Dữ liệu các field được chuyển thẳng vào model
  - Dữ liệu trong model kết nối tự động lên các field

Kết quả


Tab Mới
nnghie
Struts 2
Hello W
localhost:8080/HelloWorldStruts2/Simp
Mã đăng nhập:
nnghiem
Mật khẩu:
Ghi chú:
Giảng viên FPT, Kent, NhatNghe
Hình ảnh:
Chọn Tệp tin
adduser.png
Submit

Mã HTML

Thẻ <s:form>

```
<s:form action="hello" method="post" enctype="multipart/form-data">
...
</s:form>
```

Sẽ sinh mã tương ứng

```
<form id="hello" name="hello" action="/HelloWorldStruts2/hello.action" method="post"
enctype="multipart/form-data">
<table class="wwFormTable">
...
</table>
</form>
```

Thẻ <s:textfield>

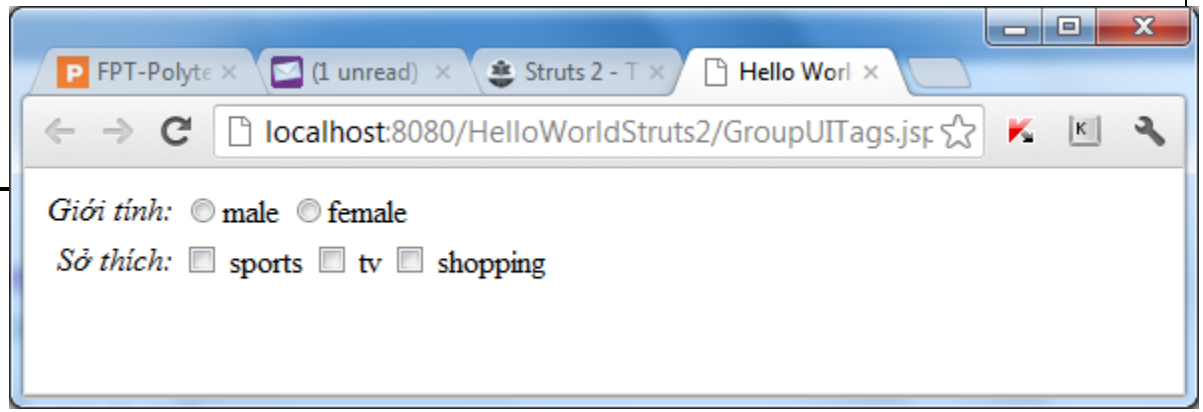
```
<s:textfield name="from" label="Mã đăng nhập"/>
```

Sẽ sinh mã tương ứng là

```
<tr>
    <td class="tdLabel">
      <label for="hello_from" class="label">Mã đăng nhập:</label>
    </td>
    <td>
      <input type="text" name="from" value="" id="hello_from"/>
    </td>
</tr>
```

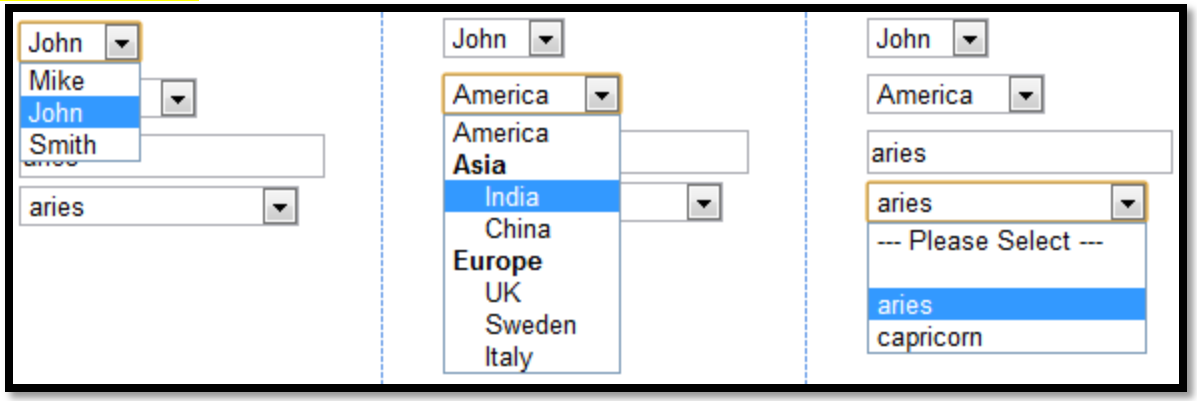
Thẻ nhóm

```
<%@ page contentType="text/html; charset=utf8" pageEncoding="utf8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
	<title>Hello World</title>
	<s:head />
</head>
<body>
<s:form action="hello.action">
	<s:radio name="gender"  label="Giới tính"
		list="#{'1':'male','0':'female'}" />

	<s:checkboxlist name="hobbies" label="Sở thích"
		list="{'sports','tv','shopping'}" />
</s:form>
</body>

</html>
```



Thẻ form - select

```
<s:select name="username" label="Username"
   list="{'Mike','John','Smith'}" />

<s:select label="Company Office" name="mySelection"
   value="{'America'}"
   list="#{'America':'America'}">
      <s:optgroup label="Asia"
         list="#{'India':'India','China':'China'}" />
      <s:optgroup label="Europe"
         list="#{'UK':'UK','Sweden':'Sweden','Italy':'Italy'}" />
</s:select>

<s:combobox label="My Sign" name="mySign"
   list="#{'aries':'aries','capricorn':'capricorn'}"
   headerKey="-1"
   headerValue="--- Please Select ---"
   emptyOption="true"
   value="capricorn" />
```



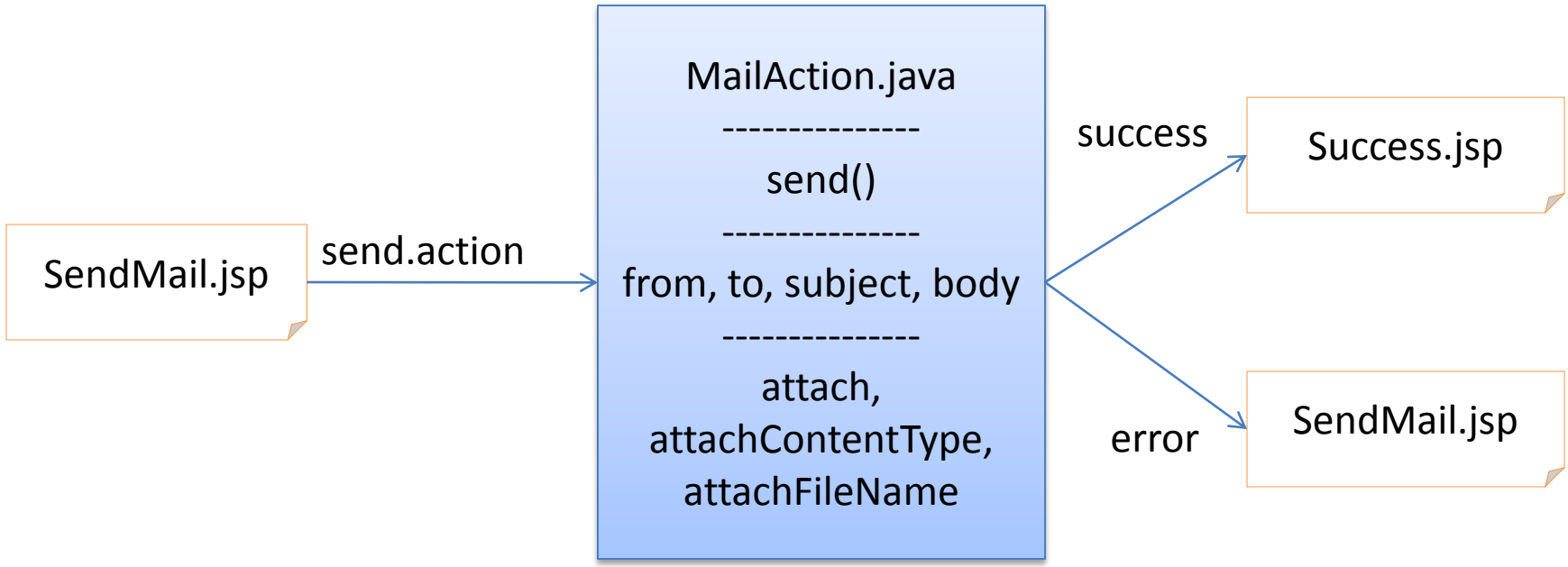
Upload File

- Sự cần thiết của upload trong ứng dụng web
  - Tải tài liệu, hồ sơ lên server
  - Đính kèm theo mail
- Kỹ thuật upload trong Struts2
  - Quản lý file vừa upload (lưu trữ, tên, loại)
    - ✓Định nghĩa các thuộc tính trong Action
      - xyz: quản lý file
      - xyzContentType: quản lý kiểu file
      - xyzFileName: quản lý tên fiel
  - Hạn chế loại và kích thước
    - ✓Khai báo interceptors trong struts.xml

Ví dụ upload

- <s:form **enctype="multipart/form-data"**>
  - **<s:file name="photo"/>**
- Lớp Action
  - **photo, photoContentType, photoFileName**
- Struts.xml
  - <interceptor-ref name="fileUpload">
  - <param name="maximumSize">**10240**</param>
  - <param name="allowedTypes">
    image/jpeg,image/gif,image/png</param>
  - </interceptor-ref>
  - <interceptor-ref name="params"/>

Đề mô gửi email có đính kèm file



- ☆ Chú ý cấu hình Struts.xml chỉ hạn chế kích thước
  - <interceptor-ref name="fileUpload">
  - <param name="maximumSize">**10240**</param>
  - </interceptor-ref>
  - <interceptor-ref name="params"/>

Thẻ dữ liệu

- Truy xuất thuộc tính trên ValueStack

```
<s:property name="myBeanProperty" />
```

- Truy xuất tài nguyên

```
<s:text name="main.title" />
```

- Nhúng 1 action

```
<s:action name="myAction" executeResult="true">
```

- Bao hàm 1 jsp

```
<s:include value="myJsp.jsp">
  <s:param name="param1" value="value2" />
  <s:param name="param2" value="value2" />
</s:include>
```

Validation

- Kiểm soát để dữ liệu nhận được từ người dùng luôn hợp lệ luôn đóng vai trò quan trọng.
- Ví dụ:
  - Không để trống, độ dài tối thiểu
  - Dạng số, ngày, email, creadicard, điện thoại, số xe máy
  - Phạm vi từ nhỏ nhất, lớn nhất
- Trong Struts2 có thể dùng 3 phương pháp khác nhau
  - Mã Java **bằng tay**
  - Khai báo bằng **XML**
  - Sử dụng **Annotation**

Validation bằng tay

- Viết đè phương thức validate() của ActionSupport
- Ví dụ sau là mã kiểm lỗi cho field userName và password.

```
@Override
public void validate() {
    if(userName == null || userName.length() == 0){
        addFieldError("userName", getText("userName.required"));
    }

    if(password == null || password.length() == 0){
        addFieldError("password", getText("password.required"));
    }
    else if(password.length() < 6){
        addFieldError("password", getText("password.minlength"));
    }
}
```

Validation bằng XML

```
<!DOCTYPE validators
    PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
    "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
    <field name="userName">
        <field-validator type="requiredstring">
            <message>User Name is required.</message>
        </field-validator>
    </field>
    <field name="password">
        <field-validator type="requiredstring">
            <message key="password.required"/>
        </field-validator>
        <field-validator type="stringlength">
             <param name="minLength">6</param>
             <param name="trim">true</param>
            <message>It nhat ${minLength} ky tu</message>
        </field-validator>
    </field>
</validators>
```

LoginAction-validation.xml

Phải được đặt cùng chỗ với LoginAction.java

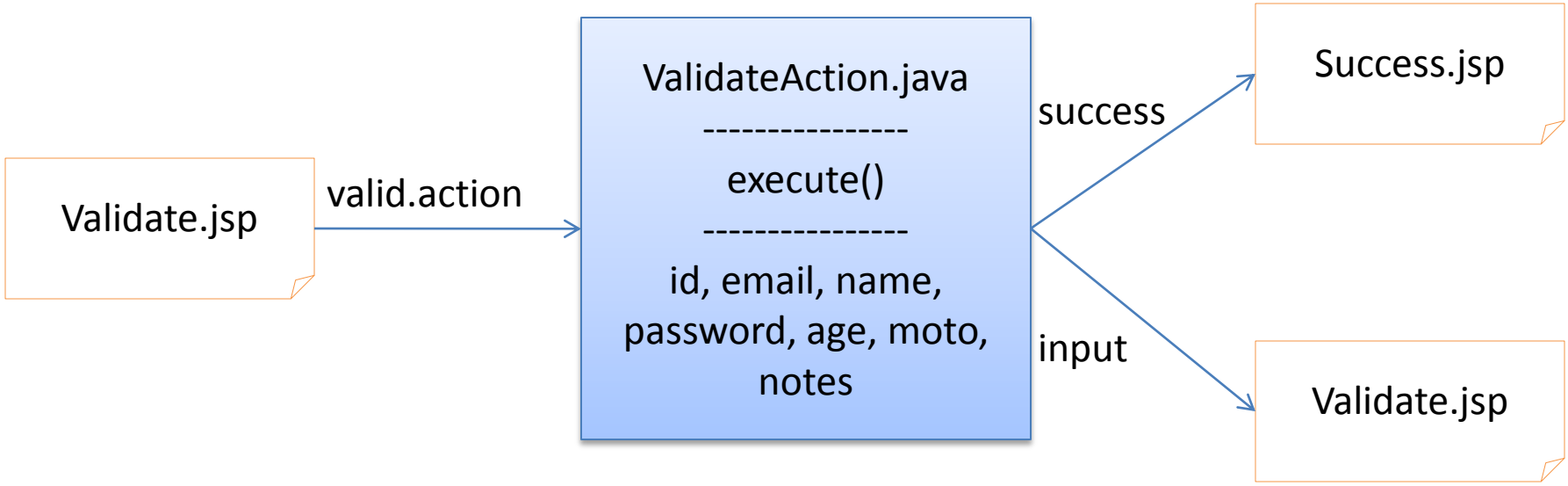
Validation bằng Annotation

- Đính kèm các Annotation phù hợp đối với các thuộc tính muốn kiểm soát.

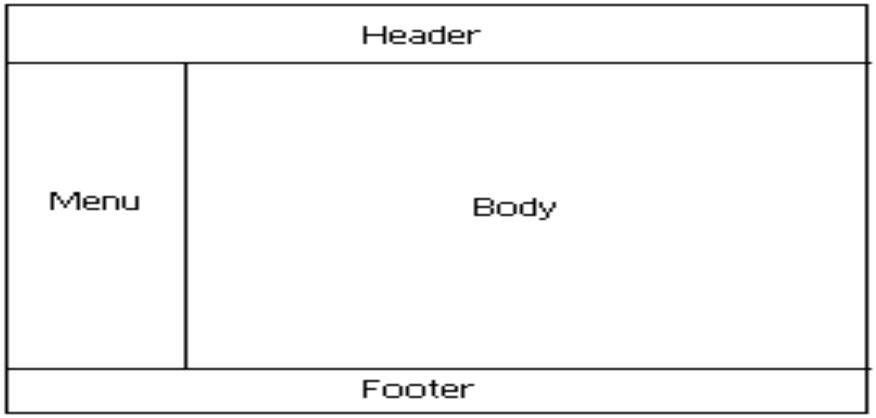
```
@RequiredStringValidator(message="userName.required")
public String getUserName() {
    return userName;
}
public void setUserName(String userName) {
    this.userName = userName;
}

@RequiredStringValidator(key="password.required")
@StringLengthFieldValidator(minLength="6",
    message="Ít nhất phải ${minLength} ký tự !")
public String getPassword() {
    return password;
}
public void setPassword(String password) {
    this.password = password;
}
```

Đề mô kiểm soát thông tin đăng ký

- ☆Không để trống id, name, email
- ☆Password ít nhất 6 ký tự, notes tối đa 255 ký tự
- ☆Email phải đúng dạng
- ☆Age phải từ 18 đến 55
- ☆Moto phải đúng dạng số Saigon



Tiles



Trong trang khung mẫu: **Header**, **Menu**, **Footer** và **Body** là các vùng trống cần được bổ sung ở các trang áp dụng mẫu này

- Khung mẫu áp dụng cho các trang trong website.
- Thống nhất tổ chức theo văn hóa doanh nghiệp
- Thực hiện:
  - Tạo trang mẫu và đánh dấu vùng trống bằng <tiles:insertAttribute>
  - Tạo trang web thành viên và lắp nội dung cho các vùng trống bằng <titles:putAttribute>

Cấu trúc trang khung mẫu

```
<%@ taglib prefix="tiles"
    uri="http://tiles.apache.org/tags-tiles" %>
...
<tiles:insertAttribute name="Header" />
...
<tiles:insertAttribute name="Menu" />
...
<tiles:insertAttribute name="Body" />
...
<tiles:insertAttribute name="Footer" />
....
```

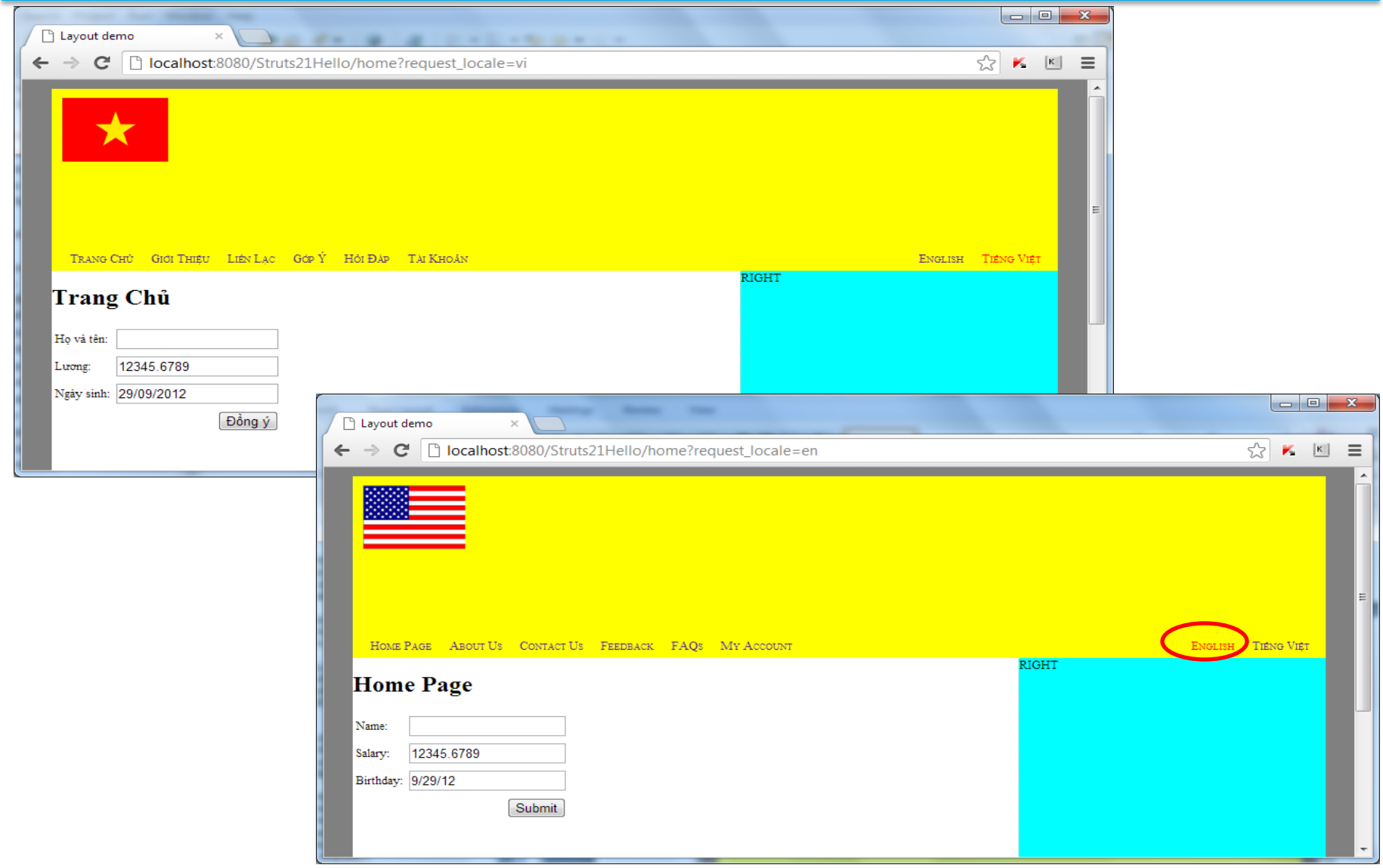
☆Header, Menu, Body và Footer là các vùng cần chèn nội dung để tao trang thành viên

Cấu trúc trang áp dụng khung mẫu

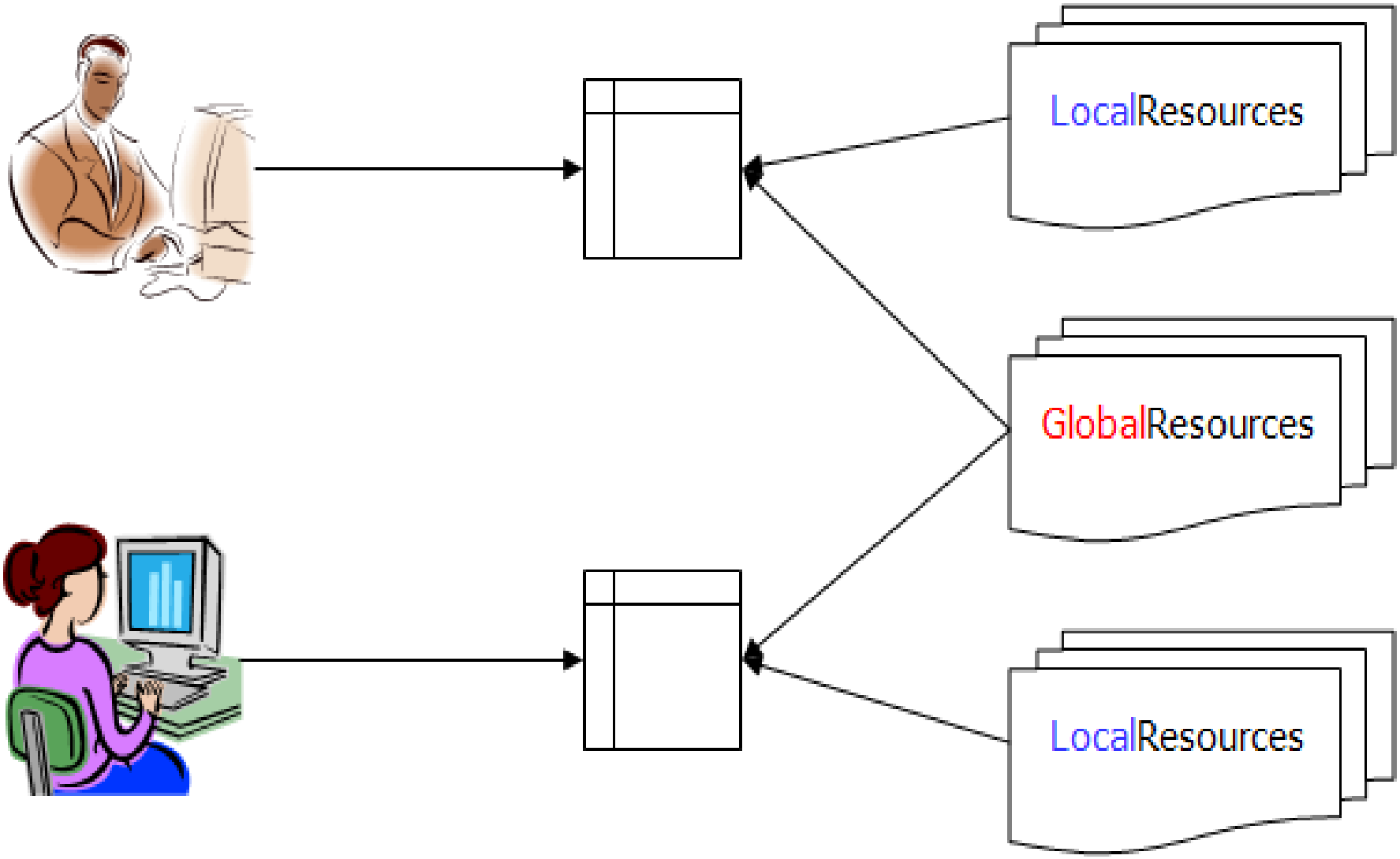
```
<%@ taglib prefix="tiles"
    uri="http://tiles.apache.org/tags-tiles" %>

<tiles:insertTemplate template="MyLayout.jsp">
    <tiles:putAttribute name="Header" value="/Header.jsp"/>
    <tiles:putAttribute name="Menu" value="/Menu.jsp"/>
    <tiles:putAttribute name="body">
        <h1>NỘI DUNG PHẦN BODY</h1>
    </tiles:putAttribute>
    <tiles:putAttribute name="Footer" value="/Footer.jsp"/>
</tiles:insertTemplate>
```

☆Các phần còn thiếu có thể chèn nội dụng của một trang (như Header, Footer và Menu) hoặc nội dung trực tiếp (như “Body”)

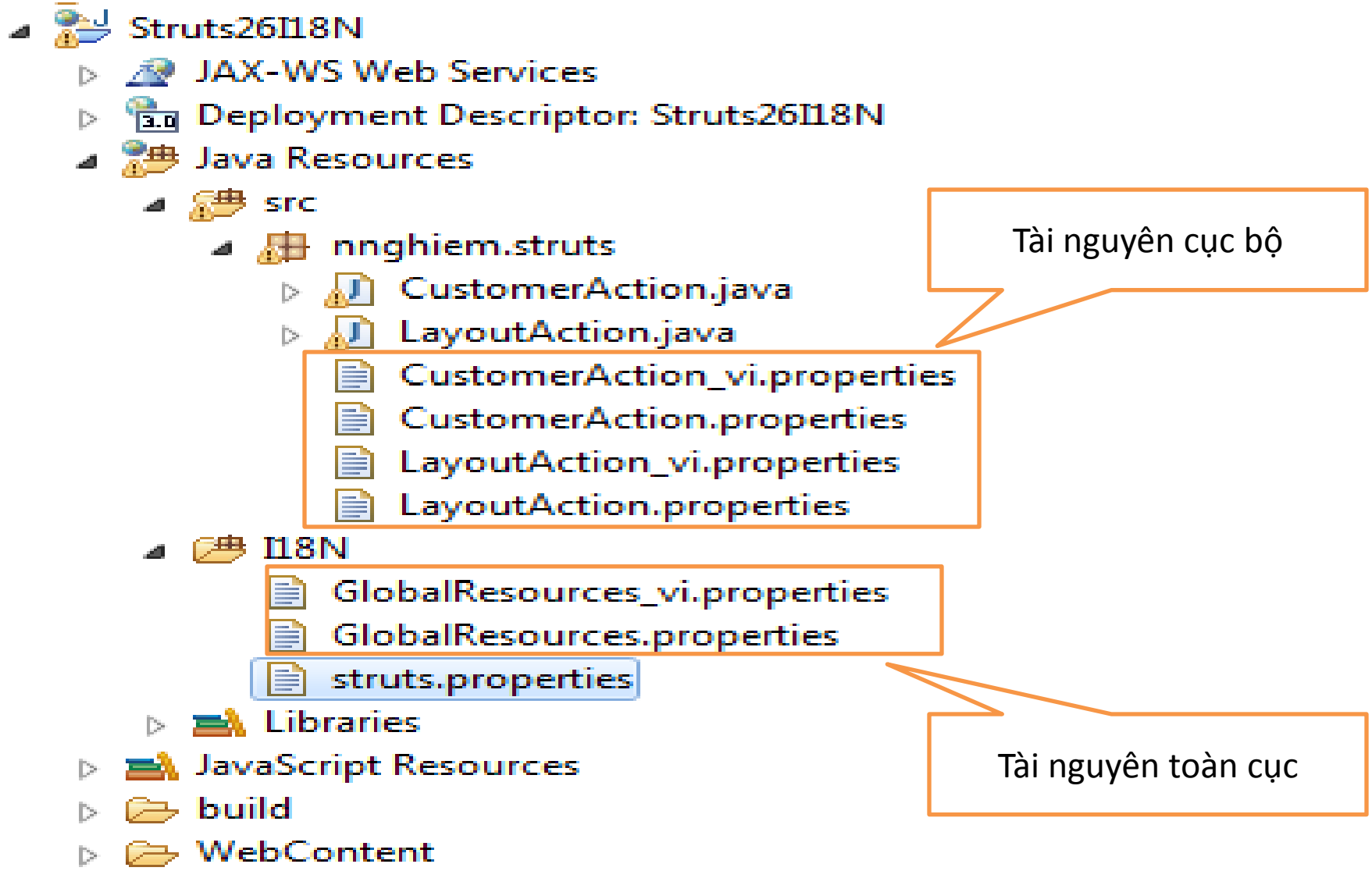
Tổ chức website đa ngôn ngữ



Tổ chức website



Tập tin tài nguyên



Lựa chọn ngôn ngữ và truy xuất tài nguyên

☆Chọn ngôn ngữ:

- ✍Chỉ cần truyền tham số request_locale=<mã ngôn ngữ> cho bất kỳ action nào

```
<a href="?request_locale=en">English</a>
<a href="?request_locale=vi">Tiếng Việt</a>
```

☆Sử dụng các thẻ

- ✍<s:text> và <s:các trường trên form>

```
<h1><s:text name="layout.home"/></h1>

<s:form action="locale">
   <s:textfield name="name" key="home.name" size="20" />
   <s:textfield name="salary" key="home.salary" size="20" />
   <s:textfield name="birthday" key="home.birthday" size="20" />
   <s:submit name="submit" key="home.submit" />
</s:form>
```

Chia sẽ dữ liệu

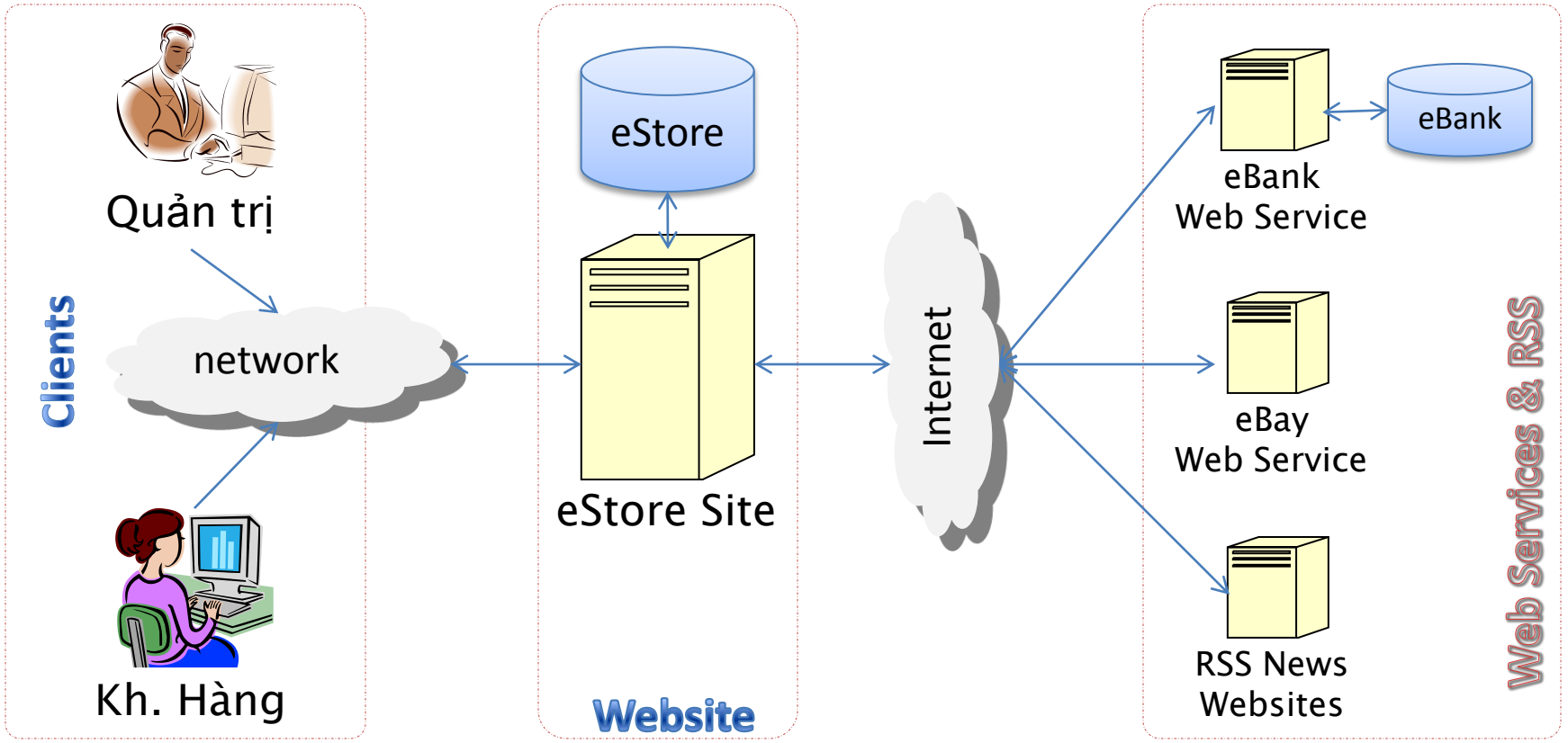
- Sử dụng ServletActionContext

```
HttpServletRequest req = ServletActionContext.getRequest();
HttpServletResponse resp = ServletActionContext.getResponse();
HttpSession session = req.getSession();
ServletContext application = ServletActionContext.getServletContext();
```

- Implements các interface

```
@Result(name="success", location="/Success.jsp")
public class MyAction extends ActionSupport
        implements ServletRequestAware, ServletResponseAware,
                   SessionAware, ServletContextAware{
    @Override
    public void setServletContext(ServletContext servletContext) {}
    @Override
    public void setSession(Map<String, Object> session) {}
    @Override
    public void setServletResponse(HttpServletResponse response) {}
    @Override
    public void setServletRequest(HttpServletRequest request) {}
}
```

Project (eStore)



☆ Thư viện lưu trữ kỹ thuật lập trình thường gặp để tra cứu trong quá trình học tập và công tác sau này.

Chức năng cơ bản của eStore

- Các trang thành viên
  - Đăng ký, đăng nhập, quên mật khẩu, đổi mặt khẩu, sửa đổi hồ sơ, quản lý hàng hóa yêu thích, quản lý hàng hóa đã gửi cho bạn.

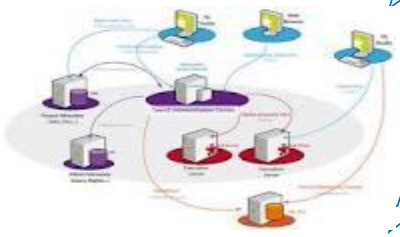
- Các trang hàng hóa
  - Hàng hóa, Loại hàng, nhà cung cấp, tìm kiếm, xem chi tiết.
  - Thống kê hàng hóa bán chạy, được yêu thích nhất, mới nhất, được xem nhiều nhất
  - Tích hợp với eBay: tìm và hiển thị hàng hóa eBay theo từ khóa

- Các trang bán hàng
  - Quản lý giỏ hàng, thao tác chọn hàng, cập nhật thông tin giỏ hàng, đặt hàng, thanh toán trực tuyến, quản lý đơn hàng.

Chức năng cơ bản của eStore (tt)

- Các trang tiếp thị
  - Send to friend, Mark as favorite, Newsletter, Google map, SEO, giảm giá, các mặt hàng liên quan, hỗ trợ trực tuyến.

- Các trang tin tức
  - Liệt kê, phân loại, tìm kiếm tin tức, quản lý mối quan hệ tin tức.



- Tích hợp thanh toán
  - Xây dựng ngân hàng ảo, tích hợp với website để thực hiện thanh toán khi đặt hàng

- Quản trị website
  - Cấu hình trang chủ, quản lý CSDL, phân quyền, báo cáo thống kê, xử lý phản hồi, quan hệ khách hàng, gửi quảng cáo.