Bài 10
JSP & JSTL

Module: JWBD

Kiểm tra bài trước

Hỏi và trao đổi về các khó khăn gặp phải trong bài "Tổng quan ứng dung Web"

Tóm tắt lại các phần đã học từ bài "Câu lệnh truy vấn SQL"

Mục tiêu

- Sử dụng được JSP
- Sử dụng được form trong ứng dụng web dựa trên Servlet & JSP
- Sử dụng được các thẻ jstl thông dụng

Java Server Page

Tìm hiểu về JSP

Luồng hoạt động của một ứng dụng web sử dụng JSP

Vòng đời JSP

JSP là gì?

- JSP (JavaServer Pages) là một ngôn ngữ kịch bản phía server, cho phép người dùng tạo ra các trang web động.
- JSP được phát hành vào năm 1999 bởi Sun Microsystems, được chạy trên nền JDK 1.3 trở về sau, là một công nghệ không thể thiếu của Java EE. Phiên bản mới nhất của JSP là 2.2.
- JSP là một tài liệu text có thể trả về cả static content (như HTML, XML, text) và dynamic content (như mã nguồn java, các thuộc tính của các lớp Java Bean, các custom tag) cho trình duyệt. Static content và dynamic content có thể đan xen lẫn nhau.
- Các file JSP được lưu với phần mở rộng là .jsp.
- Không giống như Servlet, các file JSP sẽ tự động biên dịch và triển khai khi có bất kỳ thay đổi nào.

Ví dụ file index.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <h1><% out.print("Hello World"); %></h1>
    </body>
</html>
```

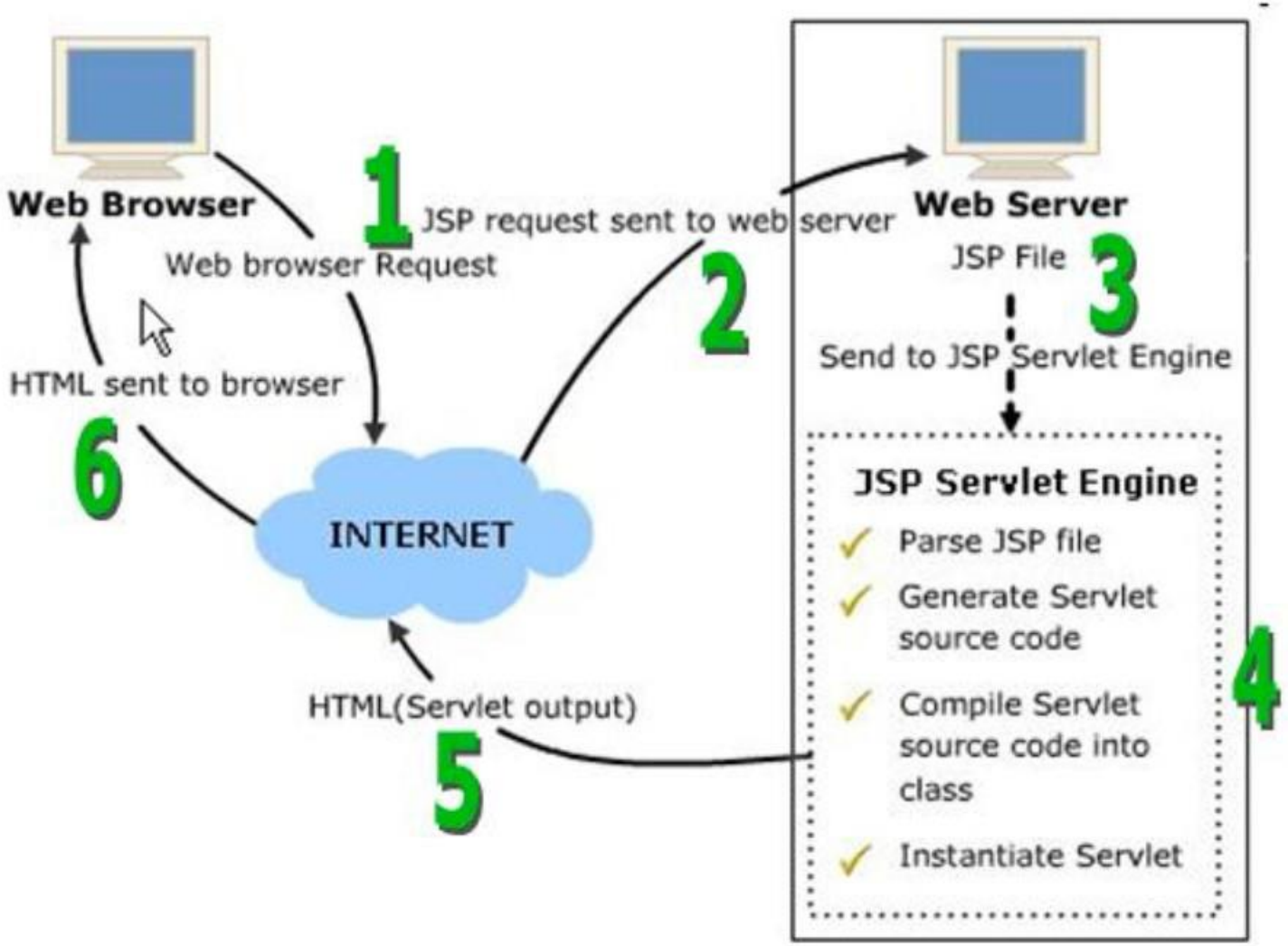
So sánh JSP và Servlet

- Về bản chất, JSP cũng chính là Servlet. Vì trong quá trình biên dịch, JSP sẽ được chuyển thành Servlet rồi Servlet này sẽ được biên dịch. Servlet của trang JSP có vòng đời giống như các Servlet thông thường.
- **Giống nhau**
  - Đều nằm ở phía server.
  - Đều xử lý dữ liệu "động".
  - Đều chạy với web container.
  - Đều là những công nghệ quan trọng của Java EE.

So sánh JSP và Servlet

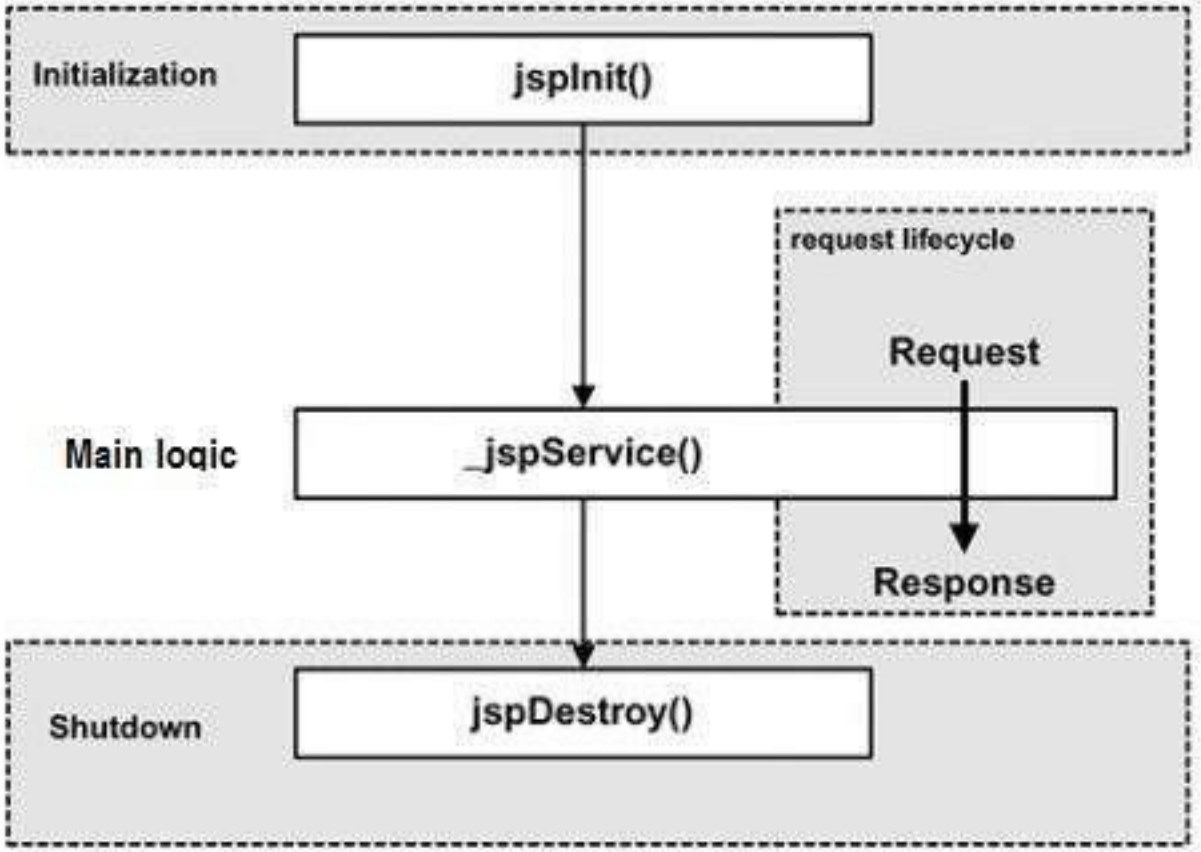
- **Khác nhau**
  - Servlet rất mạnh về xử lý và điều phối, nhưng lại rất yếu về tạo giao diện và bảo trì web.
  - JSP mạnh về xử lý hiển thị nhưng lại yếu về xử lý nghiệp vụ và điều phối.
  - Trong thực tế, chúng ta kết hợp sức mạnh của Servlet và JSP vào mô hình MVC thì: Servlet đóng vai trò Controller, thì JSP đóng vai trò View.
  - Ở Servlet, mã HTML nằm trong mã Java. Còn ở JSP, mã Java nằm trong mã HTML.

Cách xử lý trang JSP



Vòng đời của một trang JSP

- Vòng đời của một JSP được tính từ khi JSP đó được tạo ra cho đến khi bị hủy bỏ.
- Các giai đoạn trong vòng đời trang JSP
  - Biên dịch
  - Khởi tạo
  - Thực thi
  - Huỷ



1. Biên dịch

- Khi trình duyệt yêu cầu 1 trang JSP, JSP engine đầu tiên sẽ kiểm tra xem có cần biên dịch trang đó không. Nếu trang JSP chưa bao giờ được biên dịch hoặc đã được chỉnh sửa kể từ lần biên dịch cuối cùng thì JSP engine sẽ biên dịch trang JSP.
- Quá trình biên dịch gồm 3 bước:
  - Phân tích trang JSP
  - Chuyển trang JSP sang Servlet
  - Biên dịch Servlet

2. Khởi tạo

- Khi JSP container nạp nạp trang JSP, nó sẽ gọi phương thức jspInit() trước khi trả lời các request khác.
- Nếu bạn cần thực hiện sự khởi tạo JSP riêng, ghi đề phương thức jspInit():

```
public void jspInit(){
  // Initialization code...
}
```

- Việc khởi tạo được thực hiện chỉ một lần và với phương thức int của Servlet
- Trong jspInit(), chúng ta có thể khởi tạo kết nối tới CSDL, mở file,…

3. Thực thi

- Khi trình duyệt yêu cầu một trang JSP, trang này đã được nạp và khởi tạo, thì JSP engine sẽ gọi phương thức _jspService().
- Phương thức _jspService() nhận một HttpServletRequest và một HttpServletResponse như là các tham số của nó.
- Phương thức _jspService() của JSP được triệu hồi một lần cho mỗi yêu cầu và nó chịu trách nhiệm tạo Response cho Request đó.

```
void _jspService(HttpServletRequest request,
                 HttpServletResponse response)
{
   // Service handling code...
}
```

- Phương thức này cũng chịu trách nhiệm tạo các phản hồi tới tất cả phương thức của HTTP, ví dụ: GET, POST, DELETE, …

4. Huỷ

- Giai đoạn hủy một JSP trong vòng đời JSP biểu thị khi nào thì một JSP bị gỡ bỏ khỏi một container.
- Ghi đè phương thức jspDestroy khi bạn cần thực hiện bất kỳ quá trình hủy nào, ví dụ như giải phóng kết nối với Database, hoặc đóng các file.

```
public void jspDestroy()
{
   // Your cleanup code goes here.
}
```

Ví dụ

index.jsp

```jsp
<%-- Created by IntelliJ IDEA. --%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%!
    int counter;
    public void jspInit() {
        counter = 0;
        System.out.println("The lifecycle jsp has been initialized");
    }
%>

<html>
<head>
    <title>JSP Life Cycle Example</title>
</head>
<body>
<%
    System.out.println("The lifecycle jsp has received a request");
    counter++;
%>
<h2>JSP Life cycle : Request counter</h2>
<p>This page has been called <%=counter %> times </p>
</body>
</html>
<%!
    public void jspDestroy() {
        System.out.println("The lifecycle jsp is being destroyed");
    }
%>
```

Cú pháp cơ bản trong JSP

- Để chèn các đoạn mã nguồn java vào bên trong trang JSP sử dụng 3 dạng cơ bản sau:
  - Expressions: <%=Expressions%>
  - Scriptlets: <%Code%>
  - Declarations: <%!Declarations%>

Các đối tượng ẩn

- Đối tượng ẩn (Implicit Object) là các đối tượng Java mà JSP Container cung cấp cho các nhà phát triển trong mỗi trang.
- Các đối tượng này được gọi trực tiếp mà không cần khai báo. Còn được gọi là các biến được định nghĩa trước.

Các đối tượng ẩn

Đối tượng	Mô tả
request	Đây là đối tượng HttpServletRequest được liên kết với request
response	Đây là đối tượng HttpServletResponse được liên kết với response tới client
out	Đây là đối tượng PrintWriter được sử dụng để gửi dữ liệu tới client
session	Đây là đối tượng HttpSession được liên kết với request
application	Đây là đối tượng ServletContext được liên kết với application context
config	Đây là đối tượng **ServletConfig** được liên kết với page.
pageContext	Điều này đóng gói việc sử dụng các tính năng cụ thể của server như **JspWriters** với hiệu suất cao hơn.
page	Đơn giản là một từ đồng nghĩa với **this**, được sử dụng để gọi các phương thức được định nghĩa bởi lớp servlet được biên dịch.
exception	Đối tượng **Exception** cho phép các dữ liệu ngoại lệ được truy cập bằng JSP được chỉ định.

JSTL

- JSTL là bộ thư viện thẻ chuẩn được bổ sung với mục dicdhs tối ưu lập trình **trong** JSP
- Các thư viện cần thiết cho JSTL gồm
- Jstl-api.jar
- Jstl-impl.jar
- Trong JSTL có rất nhiều bộ thẻ để xử lý các vấn đề khác nhau
  - Core: Chứa các thẻ điều khiển cơ bản
  - Format: Chứa các thẻ định dạng và đa ngôn ngữ
  - XML: Chứa các thẻ xử lý tài liệu xml
  - SQL: Chứa các thẻ làm việc với CSDL
  - Function: Chứa các thẻ cung cấp các hàm hỗ trợ cho Expression Language

Trong phạm vi bài học này, chúng tả sẽ sử dụng bộ thẻ cơ bản

Các thẻ cơ bản trong jslt

- Các thẻ cơ bản (Core Tags)
  - Chứa các lệnh điều khiển như như if, choose when otherwise,vòng lặp

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

Các thẻ JSTL cơ bản

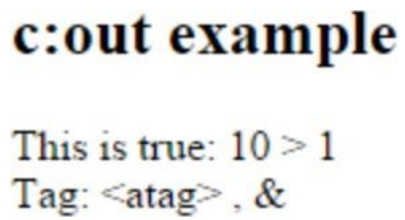
Thẻ	Mô tả
Thẻ <c:out> trong JSTL	Giống <%= ... >, nhưng cho các Expression
Thẻ <c:set> trong JSTL	Thiết lập kết quả của một ước lượng Expression trong một 'scope'
Thẻ <c:remove> trong JSTL	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
Thẻ <c:catch> trong JSTL	Bắt bất kỳ Throwable mà xuất hiện trong thân của nó và trưng bày nó một cách tùy ý
Thẻ <c:if> trong JSTL	Thẻ điều kiện đơn giản, mà ước lượng phần thân của nó nếu điều kiện đã cho là true
Thẻ <c:choose> trong JSTL	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise>
Thẻ <c:when> trong JSTL	Thẻ phụ của <choose> mà include phần thân của nó nếu điều kiện được ước lượng là true

Các thẻ JSTL cơ bản

Thẻ	Mô tả
Thẻ <c:otherwise >trong JSTL	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện trước được ước lượng là 'false'
Thẻ <c:import> trong JSTL	Thu nhận một URL tuyệt đối hoặc quan hệ và trưng bày nội dung của nó tới hoặc trang đó, một String trong 'var', hoặc một Reader trong 'varReader'
Thẻ <c:forEach >trong JSTL	Thẻ lặp cơbản, chấp nhận nhiều kiểu tập hợp khác nhau và hỗ trợ subsetting(chia tập con) và tính năng khác
Thẻ <c:forTokens> trong JSTL	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
Thẻ <c:param> trong JSTL	Thêm một parameter tới một URL của thẻ đang chứa 'import'
Thẻ <c:redirect >trong JSTL	Redirect tới một URL mới
Thẻ <c:url> trong JSTL	Tạo một URL với các tham số truy vấn tùy ý

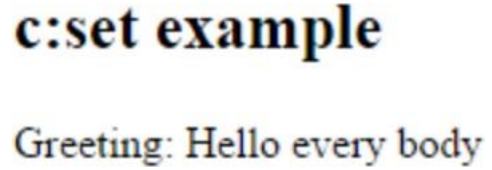
Ví dụ sử dụng c:out

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>c:out example</title>
</head>
<body>
<h2>c:out example</h2>
<c:out value="${'This is true: 10 > 1 '}" />
<br/>
Tag: <c:out value="${'<atag> , &'}"/>
</body>
</html>
```



Ví dụ sử dụng c:set

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>c:set example</title>
</head>
<body>
<h2>c:set example</h2>
<c:set scope="request" var="greeting" value="Hello every  body" />
Greeting: <c:out value="${greeting}"/>
</body>
</html>
```



Ví dụ sử dụng c:if

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<html>
<head>
  <title><c:if> Tag Example</title>
</head>
<body>
<c:set var = "salary" scope = "session" value = "${2000*2}"/>
<c:if test = "${salary > 2000}">
<p>My salary is: <c:out value = "${salary}"/><p>
  </c:if>
</body>
</html>
```

```
My salary is: 4000
```

Ví dụ sử dụng c:choose - c:when - c:otherwise

```
<%@ taglib uri = "http://java.sun.com/jsp/jstl/core" prefix = "c" %>
<html>
<head>
<title><c:choose> Tag Example</title>
 </head>
 <body>
 <c:set var = "salary" scope = "session" value = "${2000*2}"/>
 <p>Your salary is : <c:out value = "${salary}"/></p>
 <c:choose>
  <c:when test = "${salary <= 0}">  Salary is very low to survive.
  </c:when>
  <c:when test = "${salary > 1000}">  Salary is very good.
  </c:when>
  <c:otherwise>
   No comment sir...
  </c:otherwise>
 </c:choose>
 </body>
 </html>
```

```
Your salary is : 4000
Salary is very good.
```

c:forEach

```
<%@ taglib uri =  "http://java.sun.com/jsp/jstl/core" prefix =
"c" %>
<html>
<head>
<title><c:forEach> Tag Example</title>
 </head>
 <body>
 <c:forEach var = "i" begin = "1" end = "5">  Item
  <c:out value = "${i}"/><p>
 </c:forEach>
 </body>
 </html>
```

```
Item 1
Item 2
Item 3
Item 4
Item 5
```

Tóm tắt bài học

- JSP (JavaServer Pages) là một ngôn ngữ kịch bản phía server, cho phép người dùng tạo ra các trang web động.
- Toàn bộ trang JSP được thông dịch sang Servlet (một lần) và Servlet được thực thi khi yêu cầu của client gửi đến.

Hướng dẫn

Hướng dẫn làm bài thực hành và bài tập

Chuẩn bị bài tiếp theo: MVC