



ĐẠI HỌC BÁCH KHOA HÀ NỘI



Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

LẬP TRÌNH WEB HƯỚNG JAVA

Bài 09: Cơ bản về JSP

Giảng viên: ThS. Trịnh Tuấn Đạt
Bộ môn CNPM
Email: trinhthuandat.bk@gmail.com / dattt@soict.hut.edu.vn

Nội dung

- 1. JSP trong kiến trúc J2EE
- 2. Giới thiệu về JSP
- 3. Vòng đời của trang JSP
- 4. Các bước phát triển ứng dụng Web với JSP
- 5. Kỹ thuật sinh nội dung động với JSP
- 6. Gọi mã nguồn Java sử dụng **JSP scripting elements**
- 7. JavaBeans
- 8. Xử lý lỗi (Error handling)
- 9. Ví dụ: Date Website

DatTT-DSE-SOICT-HUST 2


1. JSP trong kiến trúc J2EE

DatTT-DSE-SOICT-HUST 3

JSP & Servlet trong kiến trúc J2EE 1.2

Một công nghệ Web mở rộng, kết hợp với các đối tượng Java, trả về nội dung động cho client dưới dạng HTML hoặc XML. Client thường là Web Browser

Java Servlet: 1 chương trình Java, mở rộng chức năng 1 web server, sinh nội dung động và tương tác với web clients sử dụng mô hình request-response



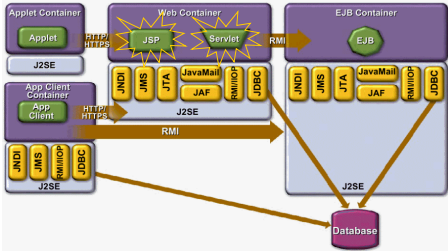
DatTT-DSE-SOICT-HUST 4

Thế nào là Static & Dynamic Contents?

- **Static contents**
 - Điển hình là các trang HTML tĩnh
 - Hiển thị như nhau cho tất cả mọi người
- **Dynamic contents**
 - Nội dung được sinh tự động theo 1 số conditions
 - Các Conditions có thể là
 - Tài khoản người dùng
 - Thời gian
 - Giá trị User nhập vào trên forms hoặc qua lựa chọn

DatTT-DSE-SOICT-HUST 5

JSP & Servlet - Web Components



DatTT-DSE-SOICT-HUST 6

2. Giới thiệu về JSP

DatTT-DSE-SOICT-HUST

7

Trang JSP là gì?

- Là 1 tài liệu text có thể trả về cả static và dynamic content cho trình duyệt
- Static content và dynamic content có thể được ghép lẫn với nhau
- Static content
 - HTML, XML, Text
- Dynamic content
 - Mã Java
 - Các thuộc tính hiển thị của JavaBeans
 - Các thẻ Custom tags

DatTT-DSE-SOICT-HUST

8

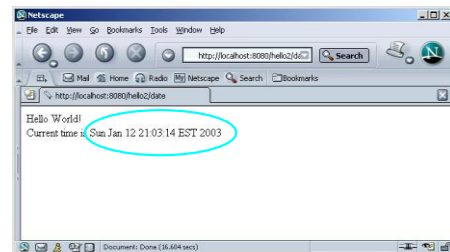
Ví dụ trang JSP (Blue: static, Red: Dynamic contents)

```
<html>
<body>
  Hello World!
<br>
  Current time is <%= new java.util.Date() %>
</body>
</html>
```

DatTT-DSE-SOICT-HUST

9

Kết quả hiển thị



DatTT-DSE-SOICT-HUST

10

2.1. JSP và Servlet

DatTT-DSE-SOICT-HUST

11

Servlets

- HTML code in Java
- Not easy to author

JSP

- Java-like code in HTML
- Very easy to author
- Code is compiled into a servlet

DatTT-DSE-SOICT-HUST

12

Ưu điểm của JSP

- Tách biệt nội dung & cách trình bày
- Đơn giản hóa việc phát triển ứng dụng Web với JSP, JavaBeans và custom tags
- Hỗ trợ tái sử dụng phần mềm qua các components (JavaBeans, Custom tags)
- Tự động triển khai
 - Tự biên dịch lại các trạng JSP khi có thay đổi
- Độc lập platform
- Dễ dàng hơn cho người thiết kế (không cần hiểu rõ Java)

DatTT-DSE-SOICT-HUST

13

Ưu điểm của JSP so với Servlet?

- Servlets:
 - Sử dụng lệnh `println()` để sinh các trang HTML
 - → Hạn chế trong bảo trì các trang HTML
 - Khi thay đổi, phải biên dịch lại, (đóng gói lại), deploy lại
- JSP:
 - Khắc phục 2 hạn chế trên

DatTT-DSE-SOICT-HUST

14

Nên dùng JSP thay cho Servlet hay ngược lại?

- Cần khai thác đồng thời 2 công nghệ
 - Sức mạnh của Servlet là "**controlling and dispatching**"
 - Sức mạnh của JSP là "**displaying**"
- Trong thực tế, cả servlet và JSP được sử dụng trong mẫu thiết kế MVC (Model-View-Controller)
 - Servlet xử lý phần Controller
 - JSP xử lý phần View

DatTT-DSE-SOICT-HUST

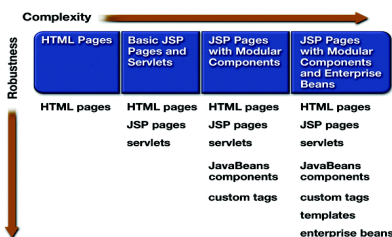
15

2.2. Kiến trúc JSP

DatTT-DSE-SOICT-HUST

16

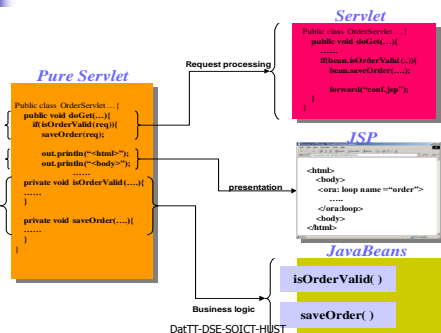
Kiến trúc/Thiết kế ứng dụng Web



DatTT-DSE-SOICT-HUST

17

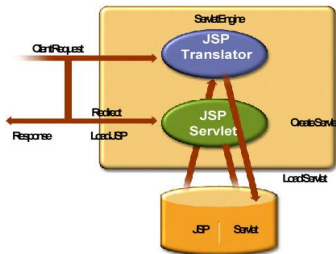
Tách biệt xử lý Request với hiển thị (Presentation)



DatTT-DSE-SOICT-HUST

18

Kiến trúc JSP



DatTT-DSE-SOICT-HUST

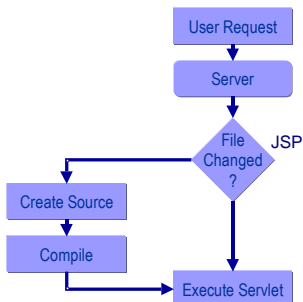
19

3. Vòng đời của 1 trang JSP

DatTT-DSE-SOICT-HUST

20

JSP làm việc như thế nào?



DatTT-DSE-SOICT-HUST

21

Các giai đoạn trong vòng đời trang JSP

- Translation
- Compile
- Execution

DatTT-DSE-SOICT-HUST

22

Giai đoạn Translation/Compilation

- Các file JSP được dịch thành mã Servlet. Sau đó mã này mới được biên dịch tiếp
- Thực hiện tự động nhờ container, ở lần đầu tiên trang JSP được truy cập (hoặc khi chỉnh sửa)
- Với trang JSP tên là "pageName", mã dịch sẽ nằm ở
 - <AppServer_HOME>/work/Standard Engine/localhost/context_root/pageName\$.jsp.java
- Ví dụ:
 - <AppServer_HOME>/work/Standard Engine/localhost/date/index\$.jsp.java

DatTT-DSE-SOICT-HUST

23

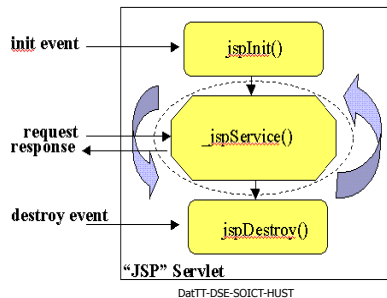
Giai đoạn Translation/Compilation

- Dữ liệu tính được chuyển thành mã Java, tác động tới output stream trả dữ liệu về cho client
- Các phần tử JSP được xử lý khác nhau:
 - Các chỉ dẫn (**Directives**) được dùng để điều khiển Web container biên dịch và thực thi trang JSP
 - Phần tử **Scripting** được thêm vào lớp servlet tương ứng của trang JSP
 - Phần tử dạng <jsp:xxx .../> được chuyển thành lời gọi phương thức tới JavaBeans components

DatTT-DSE-SOICT-HUST

24

Các phương thức trong giai đoạn thực thi



DatTT-DSE-SOICT-HUST

25

Khởi tạo trang JSP

- Có thể khai báo phương thức khởi tạo thực hiện nhiệm vụ
 - Đọc tham số cấu hình
 - Khởi tạo tài nguyên
 - Thực hiện bất kỳ công việc khởi tạo nào khác bằng việc override phương thức `jspInit()` của giao diện `JspPage`

DatTT-DSE-SOICT-HUST

26

Kết thúc trang JSP

- Khai báo phương thức thực hiện nhiệm vụ
 - Đọc tham số cấu hình
 - Giải phóng tài nguyên
 - Thực hiện bất kỳ công việc dọn dẹp nào bằng cách override phương thức `jspDestroy()` của giao diện `JspPage`

DatTT-DSE-SOICT-HUST

27

Ví dụ: initdestroy.jsp

```

<%@ page import="database.*" %>
<%@ page errorPage="errorpage.jsp" %>
<!-- Declare initialization and finalization methods using JSP declaration --%>
<%!

private BookDBAO bookDBAO;
public void jspInit() {

    // retrieve database access object, which was set once per web application
    bookDBAO =
        (BookDBAO)getContext().getAttribute("bookDB");
    if (bookDBAO == null)
        System.out.println("Couldn't get database.");
}

public void jspDestroy() {
    bookDBAO = null;
}

%>
  
```

DatTT-DSE-SOICT-HUST

28

4. Các bước phát triển ứng dụng Web với JSP

(Xem lại bài trước)

DatTT-DSE-SOICT-HUST

29

3. Các bước phát triển và triển khai ứng dụng Web

- Viết code (và biên dịch) cho các Web component (Servlet or JSP), các **helper classes** sử dụng trong web component
- Tạo các tài nguyên tĩnh (Images, các trang HTML)
- Viết file deployment descriptor (web.xml)
- Build ứng dụng Web (Tạo file *.war hoặc thư mục dạng chứa đóng gói nhưng triển khai được)
- Triển khai ứng dụng Web trên 1 Web container
 - Web clients có thể truy cập ứng dụng qua URL

DatTT-DSE-SOICT-HUST

30

So sánh Hello1 Servlet & Hello2 JSP

DatTT-DSE-SOICT-HUST

31

GreetingServlet.java (1)

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

/**
 * This is a simple example of an HTTP Servlet. It responds to the GET
 * method of the HTTP protocol.
 */
public class GreetingServlet extends HttpServlet {

    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException

    {

        response.setContentType("text/html");
        response.setBufferSize(8192);
        PrintWriter out = response.getWriter();

        // then write the data of the response
        out.println("<html>" +
                   "<head><title>Hello</title></head>");
    }
}
```

DatTT-DSE-SOICT-HUST

32

GreetingServlet.java (2)

```
// then write the data of the response
out.println("<body bgcolor=\"<#ffffff>\">" +
"<img src=\"duke.waving.gif\">" +
"<h2>Hello, my name is Duke. What's yours?</h2>" +
"<form method=\"get\">" +
"<input type=\"text\" name=\"username\" size=\"25\">" +
"<p></p>" +
"<input type=\"submit\" value=\"Submit\">" +
"<input type=\"reset\" value=\"Reset\">" +
"</form>");

String username = request.getParameter("username");

// dispatch to another web resource
if ( username != null && username.length() > 0 ) {
    RequestDispatcher dispatcher =
        getServletContext().getRequestDispatcher("/response");

    if (dispatcher != null)
        dispatcher.include(request, response);
}

out.println("</body></html>");
out.close();
}
```

DatTT-DSE-SOICT-HUST

33

GreetingServlet.java (3)

```
public String getServletInfo() {
    return "The Hello servlet says hello.";
}
}
```

DatTT-DSE-SOICT-HUST

34

greeting.jsp

```
<html>
<head><title>Hello</title></head>
<body bgcolor="white">

<h2>My name is Duke. What is yours?</h2>

<form method="get">
<input type="text" name="username" size="25">
<p></p>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>

<%
String username = request.getParameter("username");
if ( username != null && username.length() > 0 ) {
%>
<%include file="response.jsp" %>
}
%>
</body>
</html>
```

DatTT-DSE-SOICT-HUST

35

ResponseServlet.java

```
import java.io.*;
import java.util.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;

// This is a simple example of an HTTP Servlet. It responds to the GET
// method of the HTTP protocol.
public class ResponseServlet extends HttpServlet {

    public void doGet (HttpServletRequest request,
                      HttpServletResponse response)
        throws ServletException, IOException

    {
        PrintWriter out = response.getWriter();

        // then write the data of the response
        String username = request.getParameter("username");
        if ( username != null && username.length() > 0 )
            out.println("<h2>Hello, " + username + "!</h2>");
    }

    public String getServletInfo() {
        return "The Response servlet says hello.";
    }
}
```

DatTT-DSE-SOICT-HUST

36

response.jsp

```
<h2><font color="black">Hello, <%=username%>!</font></h2>
```

DatTT-DSE-SOICT-HUST

37

JSP "là" Servlet!

DatTT-DSE-SOICT-HUST

38

JSP là "Servlet"

- Các trang JSP được dịch thành servlet
 - Tomcat biên dịch greeting.jsp thành greeting\$jsp.java
- Scriptlet (Java code) trong trang JSP sẽ được chèn vào trong phương thức `jspService()` của servlet tương ứng
- Các đối tượng Servlet có thể được truy cập từ trang JSP, mã nguồn phát triển JavaBeans, hoặc custom tag.

DatTT-DSE-SOICT-HUST

39

greeting\$jsp.java (1)

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import org.apache.jasper.runtime.*;

public class greeting$jsp extends HttpJspBase {

    static {
    }

    public greeting$jsp( ) {
    }

    private static boolean _jspx_inited = false;

    public final void _jspx_init()
        throws org.apache.jasper.runtime.JspException {
    }
}
```

DatTT-DSE-SOICT-HUST

40

greeting\$jsp.java (2)

```
public void _jspService(HttpServletRequest request,
    HttpServletResponse response)
    throws java.io.IOException, ServletException
{
    JspFactory _jspxFactory = null;
    PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    String _value = null;
```

DatTT-DSE-SOICT-HUST

41

greeting\$jsp.java (3)

```
try {
    if (_jspx_inited == false) {
        synchronized (this) {
            if (_jspx_inited == false) {
                _jspx_init();
                _jspx_inited = true;
            }
        }
    }
    _jspxFactory = JspFactory.getDefaultFactory();
    response.setContentType("text/html; charset=ISO-8859-1");
    pageContext = _jspxFactory.getPageContext(this, request,
        response, "", true, 8192, true);

    application = pageContext.getServletContext();
    config = pageContext.getServletConfig();
    session = pageContext.getSession();
    out = pageContext.getOut();
}
```

DatTT-DSE-SOICT-HUST

42

(b) Gọi mã Java gián tiếp

- Phát triển các lớp tiện ích độc lập
- Chèn vào trang JSP mã nguồn Java gọi đến các lớp tiện ích
- Tốt hơn cách 1: phân tách tốt hơn phần sinh nội dung với phần presentation
- Tái sử dụng, bảo trì tốt hơn so với cách 1
- Tuy nhiên, sự tách biệt giữa contents và presentation vẫn chưa rõ ràng

DatTT-DSE-SOICT-HUST

49

(c) Sử dụng JavaBeans

- Phát triển các lớp tiện ích dưới dạng JavaBeans
- Khai thác được các hỗ trợ của JSP cho JavaBeans
- Dễ sử dụng hơn với người phát triển Web
- Dễ tái sử dụng, bảo trì hơn 2 cách đầu

DatTT-DSE-SOICT-HUST

50

(d) Phát triển và sử dụng các Custom Tags

- Phát triển các components gọi là các custom tags
 - Custom tags được thiết kế đặc biệt cho JSP
- Mạnh mẽ hơn JavaBeans component
 - (JavaBeans: chỉ có phương thức getter & setter)
- Tái sử dụng, bảo trì tốt hơn, ổn định hơn
- Tuy nhiên, phát triển custom tags khó hơn tạo các JavaBeans

DatTT-DSE-SOICT-HUST

51

(e) Sử dụng 3rd-party custom tags hoặc JSTL

- Có nhiều custom tags mã nguồn mở/thương mại
 - Ví dụ: Apache Struts
- JSTL (JSP Standard Tag Library) chuẩn hóa tập các custom tags tối thiểu cho Java EE
 - Với vai trò developer hoặc deployer, cần đảm bảo có đủ các custom tags này (Từ phiên bản J2EE 1.3)

DatTT-DSE-SOICT-HUST

52

(f) Thiết kế/Sử dụng mẫu MVC

- Tuân theo mẫu thiết kế MVC
 - Model: sử dụng các công nghệ sẵn có khác (JDBC, hibernate, ...)
 - View: sử dụng JSP
 - Controller: sử dụng Servlet
- Tự thiết kế/tạo mẫu kiến trúc MVC:
 - Khó khăn, không nên làm (Nên sử dụng sẵn)

DatTT-DSE-SOICT-HUST

53

(g) Sử dụng các MVC Model2 Frameworks đã kiểm chứng

- Có rất nhiều lựa chọn
 - Apache Struts
 - JavaServer Faces (JSF)
 - Các framework khác: Echo, Tapestry, WebWorks, Wicket

DatTT-DSE-SOICT-HUST

54

6. Gọi mã nguồn Java sử dụng JSP scripting elements

DatTT-DSE-SOICT-HUST

55

JSP Scripting Elements

- Cho phép chèn mã nguồn Java vào servlet được sinh tương ứng cho trang JSP
- Nên **GIẢM THIỂU** sử dụng JSP scripting elements trong trang JSP nếu có thể
- Có 3 dạng:
 - Expressions: `<%= Expressions %>`
 - Scriptlets: `<% Code %>`
 - Declarations: `<%! Declarations %>`

DatTT-DSE-SOICT-HUST

56

Expressions

- Trong giai đoạn thực thi trang JSP:
 - Expression được tính giá trị và chuyển thành một String
 - String sau đó được chèn trực tiếp vào output stream của Servlet tương ứng
 - Kết quả tương đương với lệnh `out.println(expression)`
 - Có thể sử dụng các biến định nghĩa trước đó (implicit objects) trong 1 expression
- Cú pháp
 - `<%= Expression %>` hoặc
 - `<jsp:expression>Expression</jsp:expression>`
 - **KHÔNG** được dùng dấu ; trong các expressions

DatTT-DSE-SOICT-HUST

57

Ví dụ: Expressions

- Hiện thị thời gian hiện tại sử dụng lớp Date
 - Current time: `<%= new java.util.Date() %>`
- Hiện thị 1 số ngẫu nhiên sử dụng lớp Math
 - Random number: `<%= Math.random() %>`
- Sử dụng các **implicit objects**
 - Hostname: `<%= request.getRemoteHost() %>`
 - Parameter: `<%= request.getParameter("yourParameter") %>`
 - Server: `<%= application.getServerInfo() %>`
 - Session ID: `<%= session.getId() %>`

DatTT-DSE-SOICT-HUST

58

Scriptlets

- Sử dụng để chèn mã Java bất kỳ vào phương thức `jspService()` của Servlet tương ứng
- Có thể làm được các việc mà expressions không thể
 - Thiết lập **response headers** và **status codes**
 - Ghi log cho server
 - Cập nhật CSDL
 - Thực thi mã nguồn có điều khiển lặp, rẽ nhánh
- Có thể sử dụng các biến đã định nghĩa (implicit objects)
- Cú pháp:
 - `<% Java code %>` hoặc
 - `<jsp:scriptlet> Java code</jsp:scriptlet>`

DatTT-DSE-SOICT-HUST

59

Ví dụ: Scriptlets

- Hiện thị **query string** (của URL yêu cầu)


```
<%
String queryData = request.getQueryString();
out.println("Attached GET data: " +
    queryData);
%>
```
- Thiết lập **response type**

```
<% response.setContentType("text/plain"); %>
```

DatTT-DSE-SOICT-HUST

60

Ví dụ: Scriptlet với điều khiển lặp

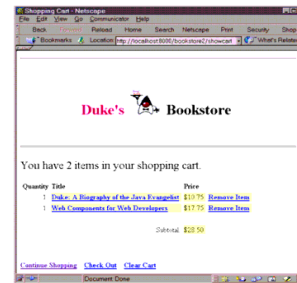
```
<%
Iterator i = cart.getItems().iterator();
while (i.hasNext()) {
    ShoppingCartItem item =
        (ShoppingCartItem)i.next();
    BookDetails bd = (BookDetails)item.getItem();
}%>

<tr>
<td align="right" bgcolor="#ffffff">
<%=item.getQuantity()%>
</td>
<td bgcolor="#ffffaa">
<strong><a href="
<%=request.getContextPath()%>/bookdetails?bookId=
<%=bd.getBookId()%>"><%=bd.getTitle()%></a></strong>
</td>
</tr>
...
// End of while
}%>
```

DatTT-DSE-SOICT-HUST

61

Ví dụ: Scriptlet với điều khiển lặp (2)



DatTT-DSE-SOICT-HUST

62

Ví dụ: biên dịch trang JSP

- Giả sử ta có đoạn mã JSP như sau:

```
<H2> sangHTML </H2>
<%= sangExpression() %>
<% sangScriptletCode(); %>
```

DatTT-DSE-SOICT-HUST

63

Ví dụ: biên dịch trang JSP

```
public void _jspService(HttpServletRequest request,
    HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    HttpSession session = request.getSession(true);
    JSPWriter out = response.getWriter();

    // Static HTML fragment is sent to output stream in "as is" form
    out.println("<H2>sangHTML</H2>");

    // Expression is converted into String and then sent to output
    out.println(sangExpression());

    // Scriptlet is inserted as Java code within _jspService()
    sangScriptletCode();
    ...
}
```

DatTT-DSE-SOICT-HUST

64

Declarations

- Được sử dụng để định nghĩa các biến hoặc phương thức (sẽ được chèn vào trong lớp Servlet tương ứng)
 - Nằm ngoài phương thức `_jspService()`
 - Declarations không được truy cập các Implicit objects (VD: đối tượng `HttpSession`)
- Thường được sử dụng với Expressions hoặc Scriptlets
- Để thực hiện thao tác khởi tạo và dọn dẹp trang in JSP pages, sử dụng declarations để override phương thức `jspInit()` và `jspDestroy()`
- Cú pháp:
 - `<%! Mã nguồn khai báo biến hoặc phương thức %>`
 - `<jsp:declaration>` Mã nguồn khai báo biến hoặc phương thức `</jsp:declaration>`

DatTT-DSE-SOICT-HUST

65

Ví dụ (1)

```
<H1>Some heading</H1>
<%!
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }
}%>
<%= randomHeading() %>
```

DatTT-DSE-SOICT-HUST

66

Ví dụ (2): Servlet tương ứng

```
public class xxxx implements HttpJSPPage {
    private String randomHeading() {
        return("<H2>" + Math.random() + "</H2>");
    }

    public void _jspService(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        HttpSession session = request.getSession(true);
        JSPWriter out = response.getWriter();
        out.println("<H1>Some heading</H1>");
        out.println(randomHeading());
        ...
    }
    ...
}
```

DatTT-DSE-SOICT-HUST

67

Ví dụ: Declaration

```
<%!
private BookDBAO bookDBAO;

public void jspInit() {
    ...
}
public void jspDestroy() {
    ...
}
%>
```

DatTT-DSE-SOICT-HUST

68

Tại sao nên sử dụng cú pháp XML?

- Hỗ trợ từ JSP 1.2
- Ví dụ
 - <jsp:expression>Expression</jsp:expression>
 - <jsp:scriptlet> Java code</jsp:scriptlet>
 - <jsp:declaration> declaration code </jsp:declaration>
- Lợi ích
 - XML validation (qua XML schema)
 - Có sẵn nhiều công cụ XML
 - editor
 - transformer
 - Java APIs

DatTT-DSE-SOICT-HUST

69

Including và Forwarding tới tài nguyên Web khác

DatTT-DSE-SOICT-HUST

70

Including các Contents trong 1 trang JSP

- Có 2 kỹ thuật để đính kèm (including) một tài nguyên Web (Web resource) trong 1 trang JSP
 - Sử dụng **include directive**
 - Sử dụng phần tử **jsp:include**

DatTT-DSE-SOICT-HUST

71

Include Directive

- Được xử lý khi trang JSP được **dịch thành Servlet**
- Hoạt động của directive là chèn text chứa trong file khác (hoặc nội dung tĩnh hoặc 1 trang JSP khác) vào trong trang JSP đang xét
- Thường được dùng để đính kèm các thông tin banner, copyright, hoặc bất kỳ nội dung nào để tái sử dụng cho các trang khác
- Cú pháp
 - <%@ include file="filename" %>
 - <%@ include file="banner.jsp" %>

DatTT-DSE-SOICT-HUST

72

Phần tử jsp:include

- Được xử lý **khi thực thi** 1 trang JSP
- Cho phép chèn tài nguyên tĩnh/động vào 1 file JSP
 - static: Nội dung của resource được chèn vào file JSP đang xét
 - dynamic: Một yêu cầu được gửi tới resource cần được đính kèm, trang cần đính kèm (included page) sẽ được thực thi, và kết quả sẽ được đính vào response rồi trả về cho trang JSP ban đầu
- Cú pháp
 - `<jsp:include page="includedPage" />`
 - `<jsp:include page="date.jsp"/>`

DatTT-DSE-SOICT-HUST

73

Sử dụng kỹ thuật nào?

- Sử dụng include directive nếu file ít khi thay đổi
 - Nhanh hơn jsp:include
- Sử dụng jsp:include cho nội dung thay đổi thường xuyên
- Sử dụng jsp:include khi chưa quyết định được sẽ đính kèm trang nào cho đến khi main page được request

DatTT-DSE-SOICT-HUST

74

Forward tới Web component khác

- Cùng cơ chế như trong Servlet
- Cú pháp


```
<jsp:forward page="/main.jsp" />
```
- Đối tượng request ban đầu sẽ được chuyển cho trang đích. Có thể đính thêm các tham số mới


```
<jsp:forward page="..." >
  <jsp:param name="param1" value="value1"/>
</jsp:forward>
```

DatTT-DSE-SOICT-HUST

75

Directives

DatTT-DSE-SOICT-HUST

76

Directives

- Directives là các thông điệp (messages) chuyển đến JSP container hướng dẫn cách biên dịch trang JSP
- Không sinh ra output
- Cú pháp
 - `<%@ directive {attr=value}* %>`

DatTT-DSE-SOICT-HUST

77

3 loại Directives

- page:** Các thuộc tính của trang JSP
 - `<%@ page import="java.util.*" %>`
- include:** Được sử dụng để đính kèm text/mã nguồn khi dịch trang JSP thành Servlet
 - `<%@ include file="header.html" %>`
- Taglib:** Chỉ ra 1 thư viện thẻ mà JSP container cần phải **interpret**
 - `<%@ taglib uri="mytags" prefix="codecamp" %>`

DatTT-DSE-SOICT-HUST

78

Page Directives

- Đưa thêm các thông tin vào Servlet biên dịch từ trang JSP tương ứng
- Cho phép điều khiển
 - Import các class nào
 - `<%@ page import="java.util.*" %>`
 - Loại MIME sinh ra là gì
 - `<%@ page contentType="MIME-Type" %>`
 - Xử lý đa luồng như thế nào
 - `<%@ page isThreadSafe="true" %>` `<%!--Default --%>`
 - `<%@ page isThreadSafe="false" %>`
 - Xử lý các lỗi không mong đợi như thế nào
 - `<%@ page errorPage="errorpage.jsp" %>`

DatTT-DSE-SOICT-HUST

79

Implicit Objects

- Một trang JSP có thể truy cập tới các implicit objects
- Được tạo bởi container
- Tương ứng với các lớp định nghĩa trong Servlet (đã học ở bài trước)

DatTT-DSE-SOICT-HUST

80

Implicit Objects

- request (HttpServletRequest)
- response (HttpServletResponse)
- session (HttpSession)
- application (ServletContext)
- out (of type JspWriter)
- config (ServletConfig)
- pageContext

DatTT-DSE-SOICT-HUST

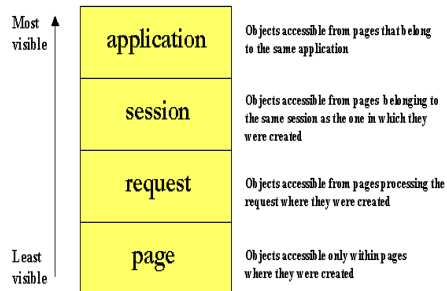
81

Scope Objects

DatTT-DSE-SOICT-HUST

82

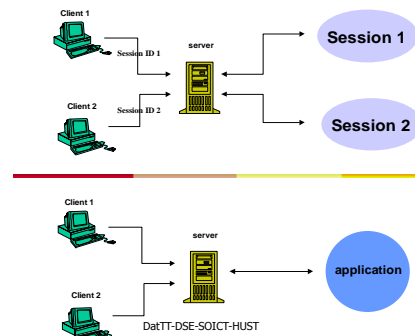
Các loại Scope



DatTT-DSE-SOICT-HUST

83

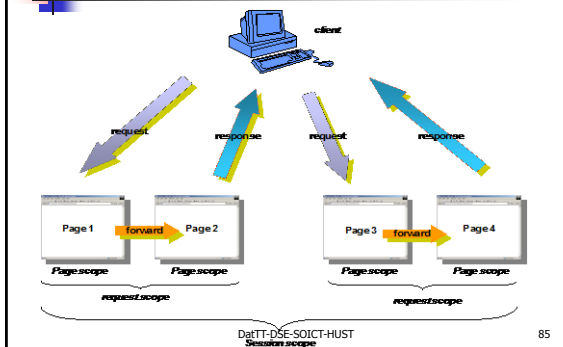
Session và Application Scope



DatTT-DSE-SOICT-HUST

84

Session, Request, Page Scope



DatTT-DSE-SOICT-HUST

85

7. JavaBeans Components

DatTT-DSE-SOICT-HUST

86

JavaBeans là gì?

- Các lớp Java classes có thể được tái sử dụng trong các ứng dụng
- Bất kỳ Java class nào tuân theo các quy định thiết kế đều thành một JavaBeans
 - Quy định về các thuộc tính của lớp
 - Quy định về phương thức **public** get, set của các thuộc tính
- Trong 1 trang JSP, có thể khởi tạo các Beans và truy cập/thiết lập (get/set) các thuộc tính của nó
- JavaBeans có thể chứa các xử lý nghiệp vụ/truy cập database (business logic/data base access logic)

DatTT-DSE-SOICT-HUST

87

Quy ước thiết kế JavaBeans

- JavaBeans có các thuộc tính (properties)
- Một thuộc tính có thể là
 - Read/write, read-only, hoặc write-only
 - Kiểu đơn hoặc kiểu mảng
- Các thuộc tính được truy cập/thiết lập qua phương thức getXxx và setXxx
 - PropertyClass getProperty() { ... }
 - PropertyClass setProperty() { ... }
- JavaBeans phải có constructor mặc định

DatTT-DSE-SOICT-HUST

88

Ví dụ: JavaBeans

```

public class Currency {
    private Locale locale;
    private double amount;
    public Currency() {
        locale = null;
        amount = 0.0;
    }
    public void setLocale(Locale l) {
        locale = l;
    }
    public void setAmount(double a) {
        amount = a;
    }
    public String getFormat() {
        NumberFormat nf =
        NumberFormat.getCurrencyInstance(locale);
        return nf.format(amount);
    }
}

```

DatTT-DSE-SOICT-HUST

89

Tại sao sử dụng JavaBeans trong các trang JSP?

- Một trang JSP có thể tạo và sử dụng bất kỳ đối tượng Java nào trong 1 **declaration** hoặc **scriptlet** như sau:

```

<%
    ShoppingCart cart =
    (ShoppingCart)session.getAttribute("cart");
    // If the user has no cart, create a new one
    if (cart == null) {
        cart = new ShoppingCart();
        session.setAttribute("cart", cart);
    }
%>

```

DatTT-DSE-SOICT-HUST

90

Tại sao sử dụng JavaBeans trong các trang JSP?

- Các trang JSP pages sử dụng phần tử JSP để tạo và truy nhập đối tượng JavaBean


```
<jsp:useBean id="cart"
class="cart.ShoppingCart" scope="session"/>
```
- Tác dụng:
 - Tạo 1 thể hiện của lớp "ShoppingCart" nếu chưa tồn tại, lưu trữ làm 1 thuộc tính của đối tượng session
 - Bean được tạo được truy cập trên toàn session với id là "cart"

DatTT-DSE-SOICT-HUST

91

So sánh 2 đoạn code

```
<%
ShoppingCart cart = (ShoppingCart)session.getAttribute("cart");
// If the user has no cart object as an attribute in Session scope
// object, then create a new one. Otherwise, use the existing
// instance.
if (cart == null) {
    cart = new ShoppingCart();
    session.setAttribute("cart", cart);
}
%>
```

versus

```
<jsp:useBean id="cart" class="cart.ShoppingCart"
scope="session"/>
```

DatTT-DSE-SOICT-HUST

92

Tại sao sử dụng JavaBeans trong các trang JSP?

- Người thiết kế trang Web không cần học Java
- Tách biệt rõ ràng giữa **content** và **presentation**
- Tái sử dụng mã nguồn hiệu quả
- Chia sẻ đối tượng dễ dàng (cơ chế **built-in**)
- Thuận tiện khi đọc tham số vào (request parameters) và chuyển thành các thuộc tính của 1 đối tượng (ví dụ ở phần sau)

DatTT-DSE-SOICT-HUST

93

Tạo một JavaBeans

- Trang JSP khai báo sử dụng bean có tầm vực nào đó với phần tử jsp:useBean

```
<jsp:useBean id="beanName"
class="fully_qualified_classname"
scope="scope"/>
```

- Hoặc

```
<jsp:useBean id="beanName"
class="fully_qualified_classname"
scope="scope">
    <jsp:setProperty .../>
</jsp:useBean>
```

DatTT-DSE-SOICT-HUST

94

Thiết lập thuộc tính cho JavaBeans

- 2 cách thiết lập thuộc tính cho bean
- Qua scriptlet
 - `<% beanName.setPropName(value); %>`
- Qua JSP:setProperty
 - `<jsp:setProperty name="beanName" property="propName" value="string constant"/>`
 - "beanName" phải trùng với id chỉ ra trong phần tử useBean
 - Phải có phương thức setPropName trong Bean

DatTT-DSE-SOICT-HUST

95

Thiết lập thuộc tính cho JavaBeans

- Cú pháp jsp:setProperty phụ thuộc vào đầu vào của thuộc tính
 - `<jsp:setProperty name="beanName" property="propName" value="string constant"/>`
 - `<jsp:setProperty name="beanName" property="propName" param="paramName"/>`
 - `<jsp:setProperty name="beanName" property="propName"/>`
 - `<jsp:setProperty name="beanName" property="*" />`
 - `<jsp:setProperty name="beanName" property="propName" value="<%= expression %>"/>`

DatTT-DSE-SOICT-HUST

96

Ví dụ: jsp:setProperty với đầu vào là Request parameter

```
<jsp:setProperty name="bookDB" property="bookId"/>
```

tương đương với

```
<%
//Get the identifier of the book to display
String bookId = request.getParameter("bookId");
bookDB.setBookId(bookId);
...
%>
```

DatTT-DSE-SOICT-HUST

97

Ví dụ: jsp:setProperty với đầu vào là 1 Expression

```
<jsp:useBean id="currency" class="util.Currency"
scope="session">
  <jsp:setProperty name="currency" property="locale"
value="<%= request.getLocale() %>"/>
</jsp:useBean>

<jsp:setProperty name="currency" property="amount"
value="<%= cart.getTotal() %>"/>
```

DatTT-DSE-SOICT-HUST

98

Truy cập các thuộc tính của JavaBeans

- 2 cách truy cập 1 thuộc tính của bean:
 - CONVERT** giá trị thuộc tính thành String và chèn giá trị vào đối tượng "out" (implicit object)
 - Lấy giá trị của thuộc tính mà **KHÔNG** cần chuyển thành String và chèn vào đối tượng "out"

DatTT-DSE-SOICT-HUST

99

Truy cập thuộc tính của JavaBeans và convert thành String rồi chèn vào out

- 2 cách
 - Qua scriptlet
 - <%= beanName.getPropName() %>
 - Qua JSP:setProperty
 - <jsp:setProperty name="beanName" property="propName"/>
- Yêu cầu
 - "beanName" phải trùng với thuộc tính id trong phần tử useBean lúc khai báo
 - Phải có phương thức "getPropName()" trong bean

DatTT-DSE-SOICT-HUST

100

Truy cập thuộc tính của JavaBeans mà không convert thành String

- Phải sử dụng **scriptlet**
- Cú pháp:

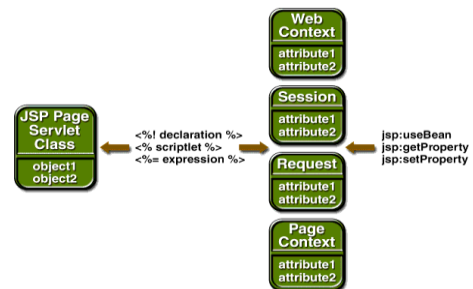

```
<% Object o = beanName.getPropName(); %>
```
- Ví dụ:


```
<%
// Print a summary of the shopping cart
int num = cart.getNumberOfItems();
if (num > 0) {
%>
```

DatTT-DSE-SOICT-HUST

101

Truy cập đối tượng trong trang JSP



DatTT-DSE-SOICT-HUST

102

8. Xử lý lỗi

DatTT-DSE-SOICT-HUST

103

Tạo trang thông báo lỗi

- Xác định ngoại lệ được tung ra
- Với mỗi trang JSP, thêm directive sau
 - `<%@ page errorPage="errorpage.jsp" %>`
- Viết 1 trang thông báo lỗi (error page), nên có directive sau
 - `<%@ page isErrorPage="true" %>`
- Trong **error page**, sử dụng đối tượng exception để hiển thị thông tin
 - `<%= exception.toString() %>`

DatTT-DSE-SOICT-HUST

104

Ví dụ: initdestroy.jsp

```

<%@ page import="database.*" %>
<%@ page errorPage="errorpage.jsp" %>
<%!

private BookDBAO bookDBAO;
public void jspInit() {

    // retrieve database access object, which was set once per web application
    bookDBAO =
        (BookDBAO)getContext().getAttribute("bookDB");
    if (bookDBAO == null)
        System.out.println("Couldn't get database.");
}

public void jspDestroy() {
    bookDBAO = null;
}
%>

```

DatTT-DSE-SOICT-HUST

105

Example: errorpage.jsp

```

<%@ page isErrorPage="true" %>
<%@ page import="java.util.*" %>
<%

ResourceBundle messages =
    (ResourceBundle)session.getAttribute("messages");
if (messages == null) {
    Locale locale=null;
    String language = request.getParameter("language");

    if (language != null) {
        if (language.equals("English")) {
            locale=new Locale("en", "");
        } else {
            locale=new Locale("es", "");
        }
    } else
        locale=new Locale("en", "");

    messages = ResourceBundle.getBundle("BookStoreMessages", locale);
    session.setAttribute("messages", messages);
}
%> ...

```

DatTT-DSE-SOICT-HUST

106

Example: errorpage.jsp

```

... (continued)
<html>
<head>
<title><%=messages.getString("ServerError")%></title>
</head>
<body bgcolor="white">
<h3>
<%=messages.getString("ServerError")%>
</h3>
<p>
<%= exception.getMessage() %>
</body>
</html>

```

DatTT-DSE-SOICT-HUST

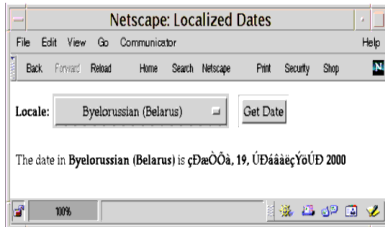
107

9. Ví dụ: Date website

DatTT-DSE-SOICT-HUST

108

9. Ví dụ: Date website -output



DatTT-DSE-SOICT-HUST

109

9. Ví dụ: Date website

```
<%@ page import="java.util.*" %>
<%@ page import="MyLocales" %>
<%@ page contentType="text/html; charset=ISO-8859-5" %>
<html>
<head><title>Localized Dates</title></head>
<body bgcolor="white">
<jsp:useBean id="locales" scope="application" class="MyLocales"/>
<form name="localeForm" action="index.jsp" method="post">
<b>Locale:</b>
```

DatTT-DSE-SOICT-HUST

110

9. Ví dụ: Date website

```
<select name="locale">
<%
Iterator i = locales.getLocaleNames().iterator();
String selectedLocale = request.getParameter("locale");
while (i.hasNext()) {
String locale = (String)i.next();
if (selectedLocale != null && selectedLocale.equals(locale) ) { %>
<option selected><%=locale%></option>
<%} else { %>
<option><%=locale%></option>
<%}
}%>
</select>
<input type="submit" name="Submit" value="Get Date">
</form>
```

DatTT-DSE-SOICT-HUST

111

9. Ví dụ: Date website

```
<p>
<jsp:include page="date.jsp" flush="true" />
</body>
</html>
```

DatTT-DSE-SOICT-HUST

112