Bài 3: Cơ bản JSP

Nôi dung bài học

- JSP là gì?
- - Vòng đời của một JSP
- - Mối quan hệ giữa JSP và Servlet
- - Kỹ thuật sinh nội dung động với JSP
- - Gọi mã nguồn Java sử dụng JSP scripting elements
- - Xử lý lỗi

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

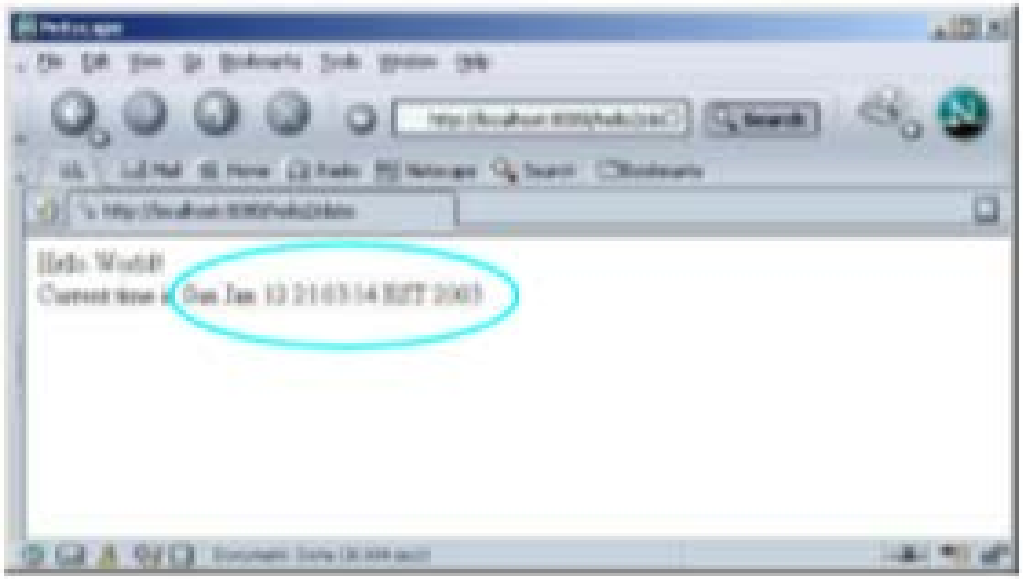
Giá trị User nhập vào trên forms hoặc qua lựa chọn

Trang JSP là gì?

- Thiết kế các trang web sử dụng HTML chuẩn
- Vị trí nào cần tạo ra nội dung động chỉ cần chèn các thẻ Java vào bên trong HTML.
- Toàn bộ trang JSP được thông dịch sang Servlet (một lần) và Servlet được thực thi khi yêu cầu của client gửi đến

Ví dụ

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



JSP và Servlet

- Servlet

Thuận lợi	Bất lợi
-Đọc dữ liệu từ Form -Đọc các HTTP Request Header -Gán HTTP Status Code và Response Header -Sử dụng Cookie và Session -Chia sẽ dữ liệu giữa các Servlet -Xử lý cơ sở dữ liệu, ...	-Sử dụng câu lệnh println để phát sinh HTML - Khi thay đổi, phải biên dich lại, (đóng gói lại), deploy lại

→ Servlet rất mạnh về xử lý và điều phối, nhưng Servlet lại rất yếu về tạo giao diện và bảo trì web

JSP và Servlet

JSP

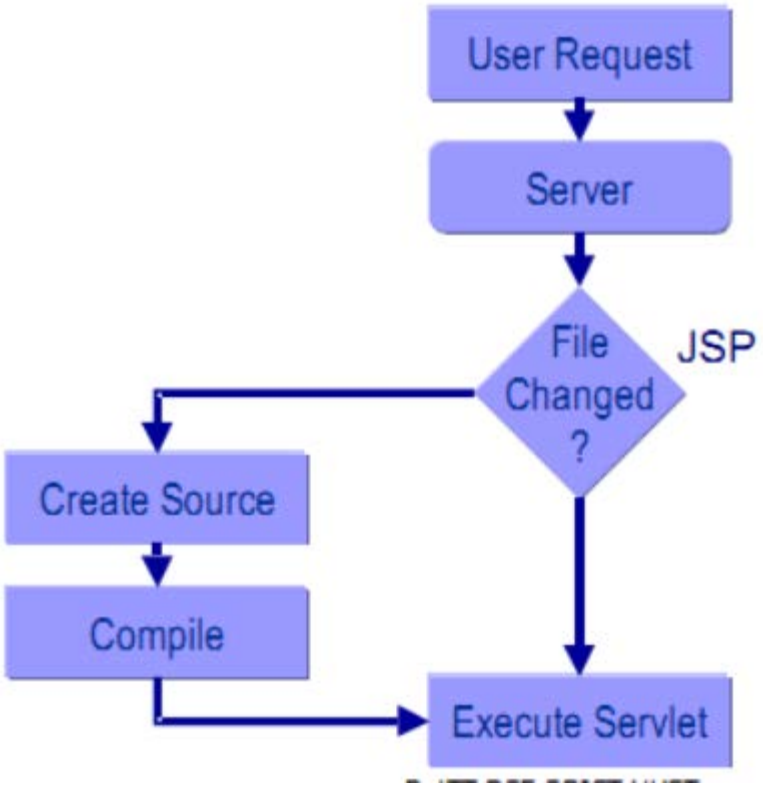
- Đơn giản hóa việc phát triển ứng dụng Web với JSP, JavaBeans và custom tags
- Hỗ trợ tái sử dụng phần mềm qua các components (JavaBeans, Custom tags)
- Tự động triển khai
- Tự biên dịch lại các trạng JSP khi có thay đổi
- Độc lập playform

→JSP mạnh về xử lý hiển thị nhưng lại yếu về xử lý nghiệp vụ và điều phối

JSP và Servlet

- Trong thực tế, chúng ta kết hợp sức mạnh của Servlet và JSP vào mô hình MVC (Model-View-Controller)
  - Các Servlet đóng vai trò làm Controller
  - Các trang JSP đóng vai trò làm View
  - Model: sử dụng các công nghệ sẵn có khác (JDBC, hibernate, ...)

Vòng đời của một trang JSP



Vòng đời của một trang JSP

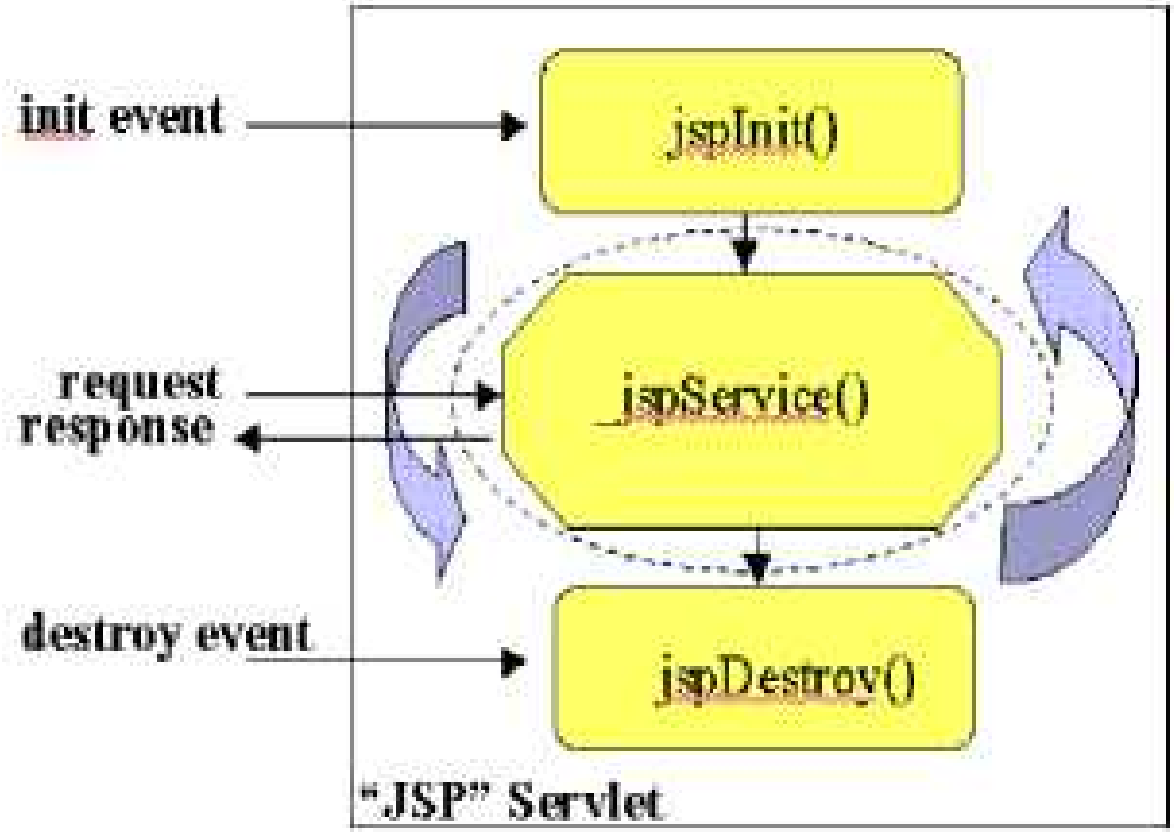
- Các giai đoạn trong vòng đời trang JSP
  - Translation
  - Compile
  - Execution

Vòng đời của một trang JSP

- 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$jsp.java
  - Ví dụ:
  - <AppServer_HOME>/work/Standard Engine/localhost/date/mdex$jsp.java
  - 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

Vòng đời của một trang JSP

- Các phương thức trong giai đoạn thực thi



Vòng đời của một trang JSP

- 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
-
- 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

Các bước phát triển ứng dụng Web với JSP

- 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 úng dụng Web (Tạo file *.war hoặc thư mục dạng chưa 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

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 jspServiceQ của servlet tương ứng
- Các đối tượng Servlet có thể được truy cập từ trang JSP, mã nguồn phát triển JavaBeans, hoặc custom tag.

Kỹ thuật sinh nội dung động trong JSP

- Có thể áp dụng các kỹ thuật khác nhau, tùy các yếu tố sau
  - Kích thước, độ phức tạp của project
  - Yêu cầu về tái sử dụng code, bảo trì, ...
- Có đầy đủ các kỹ thuật từ đơn giản tới phức tạp

Kỹ thuật sinh nội dung động trong JSP

- Gọi mã Java trực tiếp trong JSP
- Gọi mã Java gián tiếp trong JSP
- Sử dụng JavaBeans
- Tự phát triển và sử dụng các custom tags
- Sử dụng 3rd-party custom tags hoặc JSTL (JSP Standard Tag Library)
- Sử dụng mẫu thiết kế MVC
- Sử dụng Model2 frameworks đã được kiểm chứng

Gọi mã nguồn Java sử dụng JSP scripting elements

- Cho phép chèn các đoạn mã nguồn java vào bên trong trang JSP
- Khi trang JSP thông dịch các đoạn mã nguồn này sẽ được chèn vào bên trong phương thức _jspService của Servlet.
- Có 3 dạng:
  - Expressions: <%= Expressions %>
  - Scriptlets: <% Code %>
  - Declarations: <%! Declarations %>

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Expression
- Định dạng:
  - JSP : <%= Java Expression %>
- Lưu ý:- Không được phép sử dụng dấu ; trong các Expression

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Expression

Kết quả

- Expression sau tính toán ra kết quả sẽ được chuyển thành một String
- String được chèn trực tiếp vào bên trong Ouput Stream của Servlet.
- Kết quả tương tự như:
  - **out.println(Expression);**
- Trong Expression có thể sử dụng các biến:
  - Các biến được định nghĩa tường minh
  - Các đối tượng được tạo sẵn ngầm định

Gọi mã nguồn Java sử dụng JSP scripting elements

- Sử dụng JSP Expression


```
7   <%@page contentType="text/html" pageEncoding="UTF-8"%>
8   <!DOCTYPE html>
9   <html>
10      <head>
11          <meta http-equiv="Content-Type" content="text/html;
12                charset=UTF-8">
13          <title>JSP Page</title>
14      </head>
15      <body>
16          <h1>Ngày hôm nay:  <%=new java.util.Date()%> </h1>
17      </body>
18  </html>
```

Gọi mã nguồn Java sử dụng JSP scripting elements

- Sử dụng JSP Expression

```
33  response.setContentType("text/html;charset=UTF-8");
34  PrintWriter out = response.getWriter();
35  out.println("<!DOCTYPE html>");
36  out.println("<html>");
37  out.println("    <head>");
38  out.println("        <meta http-equiv='Content-Type' content='text/html;");
39  out.println("              charset=UTF-8'>");
40  out.println("        <title>JSP Page</title>");
41  out.println("    </head>");
42  out.println("    <body>");
43  out.println("        <h1>Ngày hôm nay:  "+ new java.util.Date() +" </h1>");
44  out.println("    </body>");
45  out.println("</html>");
```

Gọi mã nguồn Java sử dụng JSP scripting elements

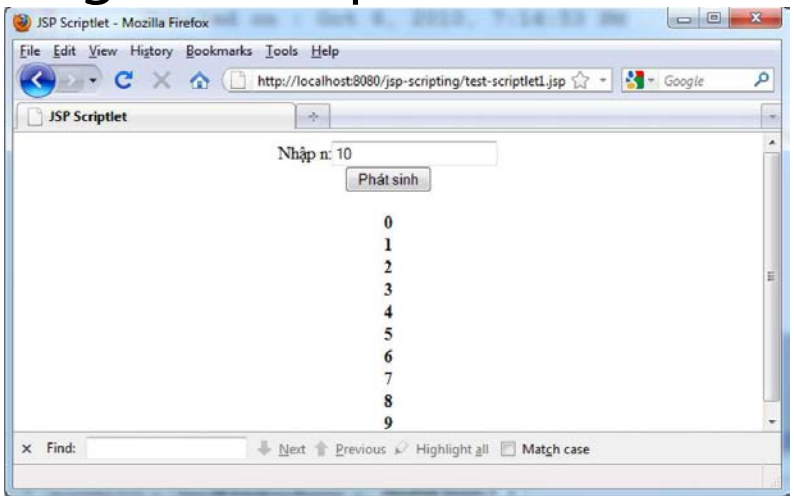
- JSP Scriptlet
- Định dạng:
  - JSP : <% Java Code %>

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Scriptlet
- Kết quả:
  - Sau khi trang JSP được thông dịch sang Servlet, mã nguồn java trong scriptlet được chèn tương ứng vào bên trong phương thức _jspServiceO
- Trong Scriptlet có thể sử dụng các biến:
  - Các biến được định nghĩa tường minh
  - Các đối tượng được tạo sẵn ngầm định
- Trong scriptlet được phép khai báo biến, sử dụng các câu lệnh điều kiện, vòng lặp, gọi phương thức,...

Gọi mã nguồn Java sử dụng JSP scripting elements

- Sử dụng JSP Scriptlet




```
1   <body>
2      <center>
3         <form name="frm" method="get">
4            Nhập n:<input type="text" name="soLuong"/><br/>
5            <input type="submit" name="bt" value="Phát sinh"/>
6         </form>
7         <%
8               String s = request.getParameter("soLuong");
9               if (s != null) {
10                 int n = Integer.parseInt(s);
11                 for (int i = 0; i < n; i++) {
12                    out.println("<b>" + i + "</b>");
13                    out.println("<br/>");
14                 }
15              }
16        %>
17     </center>
18  </body>
```

Gọi mã nguồn Java sử dụng JSP scripting elements

- Sử dụng JSP Scriptlet

```
33  response.setContentType("text/html;charset=UTF-8");
34  PrintWriter out = response.getWriter();
35  out.println("<body>");
36  out.println("   <center>");
37  out.println("      <form name='frm' method='get'>");
38  out.println("         Nhập n:<input type='text' name='soLuong'/><br/>");
39  out.println("         <input type='submit' name='bt' value='Phát sinh'/>");
40  out.println("      </form>");
41  String s = request.getParameter("soLuong");
42  if (s != null) {
43      int n = Integer.parseInt(s);
44      for (int i = 0; i < n; i++) {
45          out.println("<b>" + i + "</b>");
46          out.println("<br/>");
47      }
48  }
49  out.println("</center>");
50  out.println("</body>");
```

Gọi mã nguồn Java sử dụng JSP scripting elements

- Sử dụng Scriptlet + Expression + HTML

```
33  response.setContentType("text/html;charset=UTF-8");
34  PrintWriter out = response.getWriter();
35  out.println("<body>");
36  out.println("        <form name='frm' method='get'>");
37  out.println("            Nhập n:<input type='text' name='soLuong'/><br/>");
38  out.println("            <input type='submit' name='bt' value='Phát sinh'/>");
39  out.println("        </form>");
40  String s = request.getParameter("soLuong");
41  if (s != null) {
42      int n = Integer.parseInt(s);
43      out.println("        <table border='1'><tr><th>Chương <%=n%></th></tr>");
44      for (int i = 1; i <= 10; i++) {
45          out.println("                <tr><td><b>");
46          out.println(n + " * " + i + " = " + i * n);
47          out.println("                </b></td></tr>");
48      }
49      out.println("        </table>");
50  }
51  out.println("    </body>");
```

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Declaration
- Định dạng:
  - JSP : <%! Khai báo các thuộc tính
  - Định nghĩa các phương thức %>

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Declaration
- Sau khi trang JSP được thông dịch thành Servlet thì các khai báo thuộc tính và định nghĩa phương thức được chèn vào bên trong Servlet.
- JSP Declaration được sử dụng với Scriptlet và Expression

Gọi mã nguồn Java sử dụng JSP scripting elements

- JSP Declaration được sử dụng với Scriptlet và Expression
- JSP Declaration cho phép -Định nghĩa phương thức mới
- Cài đặt lại các phương thức jspInit(), jspDestroy
- Không được phép cài đặt lại phương thức _jspService()
- JSP Declaration không được phép sử dụng các đối tượng được định nghĩa ngầm định

Gọi mã nguồn Java sử dụng JSP scripting elements

- Khai báo JSP Declaration

```
1  <%@page contentType="text/html" pageEncoding="UTF-8"%>
2  <%@page import="dao.*, pojo.*"%>
3  <%!       private ArrayList<DanhMuc> ds;
4            // Cài đặt lại phương thức jspInit
5            public void jspInit() {
6               this.ds = DanhMucDAO.layDanhSachDanhMuc();
7            }
8            //Cài đặt lại phương thức jspDestroy
9            public void jspDestroy() {
10              this.ds = null;
11           }
12           //Định nghĩa phương thức mới
13           public void xxx() {
14              . . .
15           }
16 %>
17 <html><head><title>DanhSachSach</title></head>
18    <body> ... </body>
19 </html>
```

XIN CẢM ƠN!