



LẬP TRÌNH JAVA 4

BÀI 3: JSP, JSP LIFE CYCLE, SCRIPTING ELEMENTS, COMMENTS

PHẦN 1

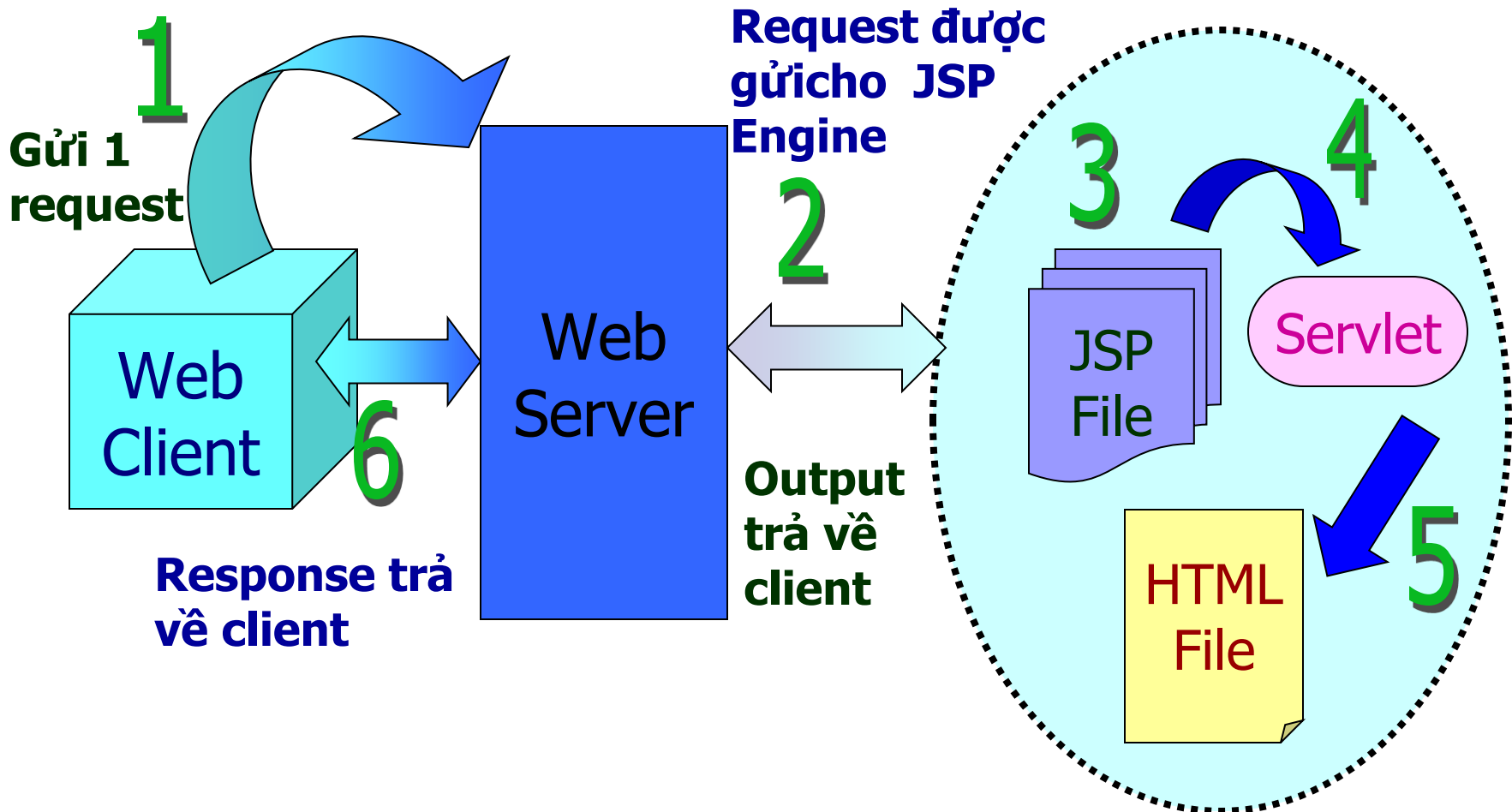
🎯 Kết thúc bài học này bạn có khả năng

- ❖ 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
- ❖ Comments - Chú thích



- ☐ Một file JSP có dạng tương tự như một file HTML nhưng có thêm các thành phần sẽ được xử lý bởi JSP engine.
- ☐ Thiết kế các trang web jsp 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
- ☐ JSP là công nghệ nằm ở tầng VIEW.
- ☐ Chuyển trang Web từ .htm sang .jsp trước hết chỉ đơn giản là ta đổi phần mở rộng.
- ☐ Là một phương cách tạo trang web động, xử lý server side, dựa trên công nghệ java

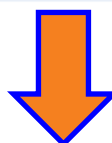
CÁC BƯỚC XỬ LÝ REQUEST



simpleDate.jsp

Server Page Template

```
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>
8 <!DOCTYPE html>
9 <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>JSP Page</title>
13 </head>
14 <body COLOR=#ffffff>
15     The time on the server is
16     <%= new java.util.Date() %>
17 </body>
</html>
```



Resulting HTML

```
4 <!DOCTYPE html>
5 <html>
6 <head>
7     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8     <title>JSP Page</title>
9 </head>
10 <body COLOR=#ffffff>
11     The time on the server is
12     Fri Jun 02 23:04:39 ICT 2017
13 </body>
14 </html>
15
```



DEMO

Chạy và giải thích



☐ Servlet

Thuận lợi

- Đọc dữ liệu từ Form
- Đọc các HTTP Request Header
- Gán HTTP Status Code và Response Header -Sử dụng Cookie và Session
- Chia sẻ dữ liệu giữa các Servlet
- Xử lý cơ sở dữ liệu, ...

Bất lợi

- Sử dụng câu lệnh println để phát sinh HTML
- Khi thay đổi, phải biên dịch lại, (đóng gói lại), deploy lại

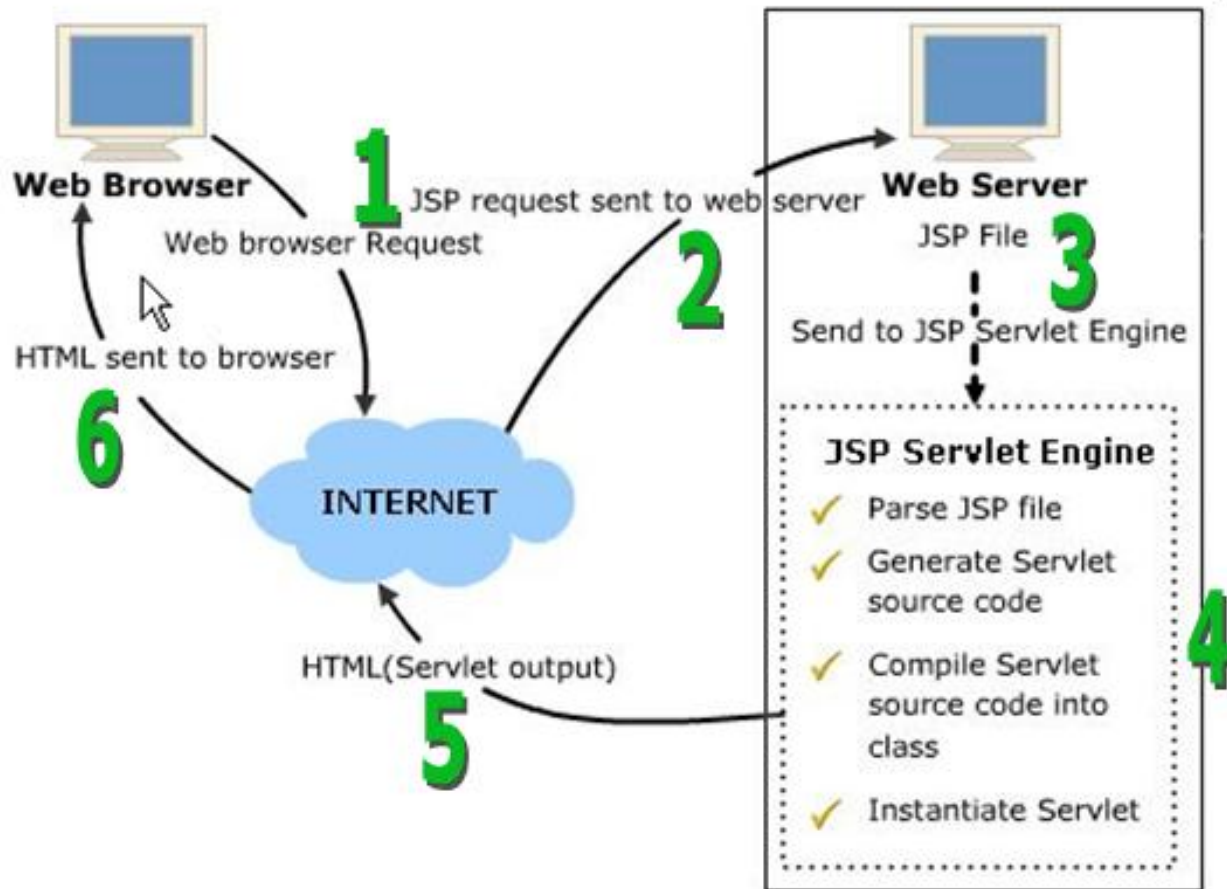
→ Servlet rất mạnh về xử lý và điều phối, nhưng Servlet lại rất yếu về tạo giao diện và bảo trì web

□ JSP

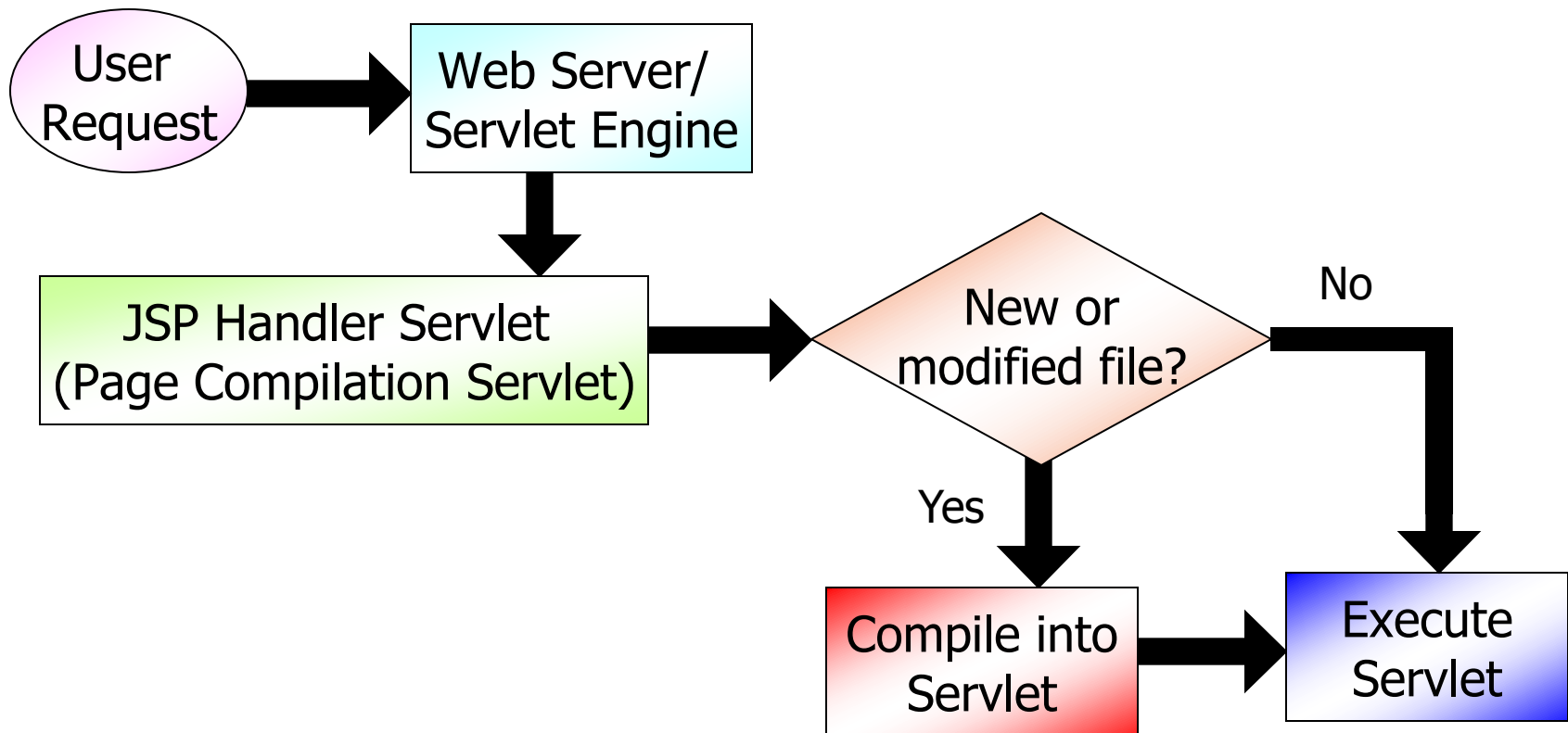
- ❖ Đơn giản hóa việc phát triển ứng dụng Web với JSP, JavaBeans và custom tags
 - ❖ Hỗ trợ tái sử dụng phần mềm qua các components (JavaBeans, Custom tags)
 - ❖ Tự động triển khai
 - ❖ Tự biên dịch lại các trạng JSP khi có thay đổi
 - ❖ Độc lập platform
- JSP mạnh về xử lý hiển thị nhưng lại yếu về xử lý nghiệp vụ và điều phối

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

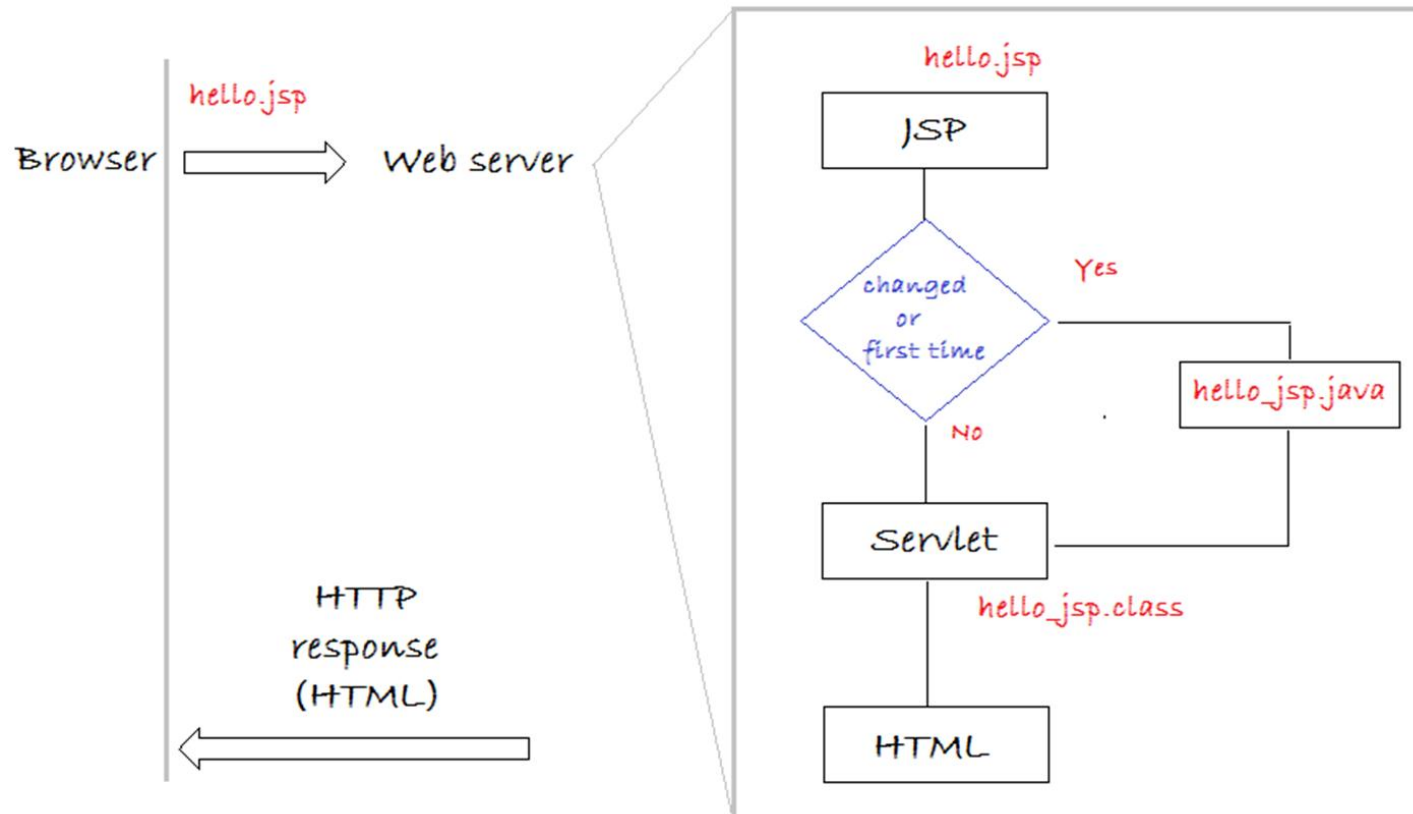
VÒNG ĐỜI CỦA MỘT TRANG JSP



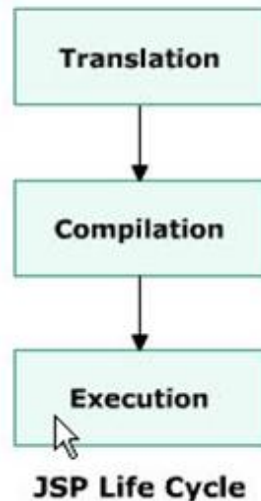
❑ Cách xử lý trang JSP



❑ Cách xử lý trang JSP



- ❑ Vòng đời của JSP có thể được định nghĩa như là toàn bộ tiến trình từ khi tạo ra đến khi hủy nó, tương tự như vòng đời của một Servlet, nhưng thêm một bước để biên dịch một JSP thành Servlet.
- ❑ Các giai đoạn trong vòng đời trang JSP
 - ❖ Translation
 - ❖ Compile
 - ❖ Execution

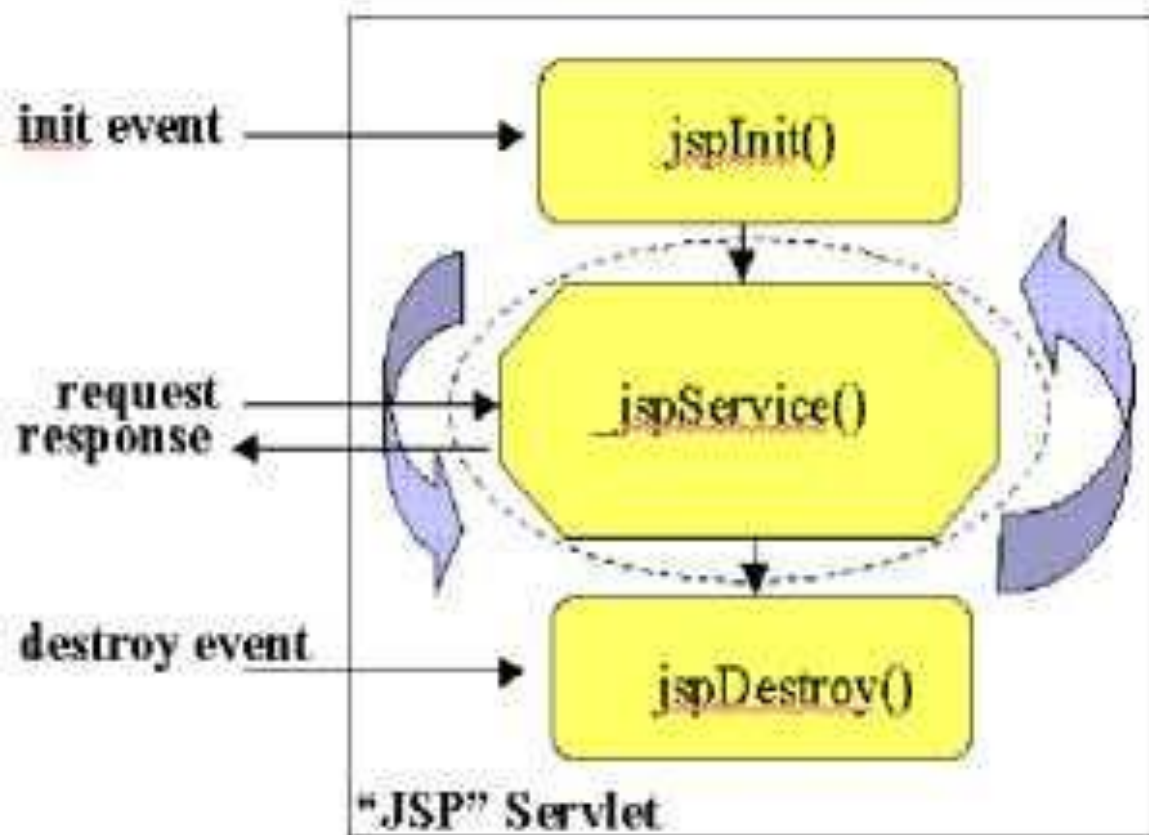


❑ Giai đoạn Translation/Compilation

- ❖ 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)
- ❖ 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



❑ Khởi tạo trang JSP

- ❖ Khi container tải một JSP, nó gọi phương thức `jspInit()` trước khi phục vụ bất kỳ yêu cầu nào. Nếu bạn cần thực hiện sự khởi tạo JSP riêng, ghi đè phương thức `jspInit()`:

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

❑ Thực thi JSP

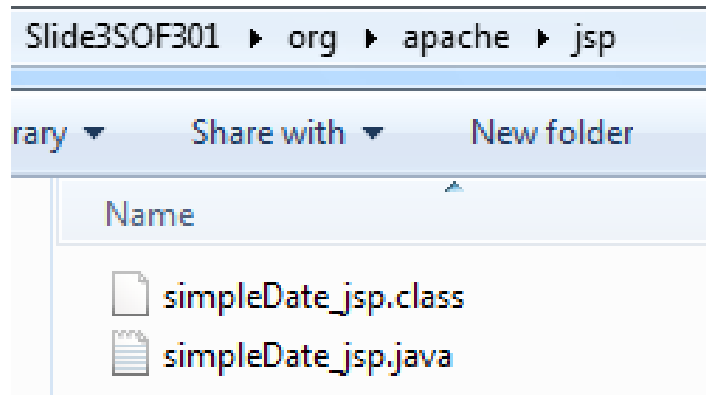
- ❖ Khi một trình duyệt yêu cầu một JSP và trang đã được tải và được khởi tạo, thì JSP engine triệu hồi phương thức **`_jspService()`** trong JSP đó.
- ❖ Phương thức `_jspService()` nhận một **`HttpServletRequest`** và một **`HttpServletResponse`** như là các tham số của nó.
- ❖ Phương thức `_jspService()` 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...  
}
```


❑ Hủy JSP

- ❖ Phương thức **jspDestroy()** trong JSP là phương thức hủy tương đương với trong Servlet. 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, giải phóng tài nguyên...

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



```
public void _jspService(final
javax.servlet.http.HttpServletRequest request, final
javax.servlet.http.HttpServletResponse response)
    throws java.io.IOException,
javax.servlet.ServletException {...}
```

```
out.write("\n");
out.write("\n");
out.write("\n");
out.write("<!DOCTYPE html>\n");
out.write("<html>\n");
out.write("    <head>\n");
out.write("        <meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">\n");
out.write("        <title>JSP Page</title>\n");
out.write("    </head>\n");
out.write("    <body COLOR=#ffffff>\n");
out.write("        The time on the server is\n");
out.write("    ");
out.print( new java.util.Date() );
out.write("\n");
out.write("    </body>\n");
out.write("</html>\n");
```



DEMO

Chạy và giải thích



- ❑ 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 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

- ☐ 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.

- ☐ 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



LẬP TRÌNH JAVA 4

BÀI 3: JSP, JSP LIFE CYCLE, SCRIPTING ELEMENTS, COMMENTS

PHẦN 2

- ❑ Cho phép chèn các đoạn mã nguồn java vào bên trong trang JSP
- ❑ Có 3 dạng:
 - ❖ Expressions: `<%= Expressions %>`
 - ❖ Scriptlets: `<% Code %>`
 - ❖ Declarations: `<%! Declarations %>`

❑ JSP Expression

❖ Cú pháp:

➤ JSP : `<%= Java Expression %>`

- ❖ Lưu ý: Không được phép sử dụng dấu ; trong các Expression
- ❖ Expression sau tính toán ra kết quả sẽ được chuyển thành một String
- ❖ String được chèn trực tiếp vào bên trong Output 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

❑ Ví dụ: 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>
```

□ Ví dụ: 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>");
```

❑ JSP Scriptlet

❖ Cú pháp:

➤ JSP : `<% Java Code; %>`

❖ Sau khi trang JSP được thông dịch sang Servlet, mã nguồn java trong scriptlet được chèn tương ứng vào bên trong phương thức `_jspService()`

❖ Trong Scriptlet có thể sử dụng các biến:

➤ Các biến được định nghĩa tường minh

➤ Các đối tượng được tạo sẵn ngầm định

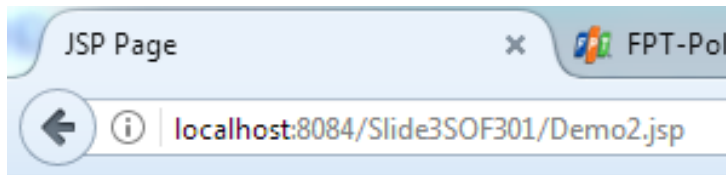
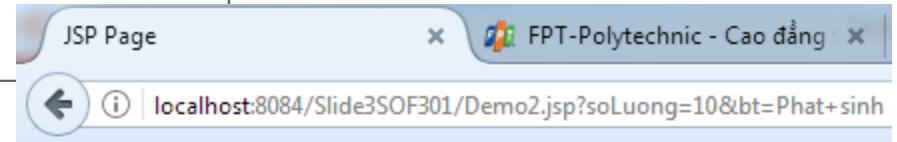
❖ Trong scriptlet được phép khai báo biến, sử dụng các câu lệnh điều kiện, vòng lặp, gọi phương thức,...

❑ Ví dụ: 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>

```



Nhap n:

Nhap n:

0
1
2
3
4
5
6
7
8
9

❑ Ví dụ: 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>");
```

❑ JSP Declaration


❖ Cú pháp:

➤ JSP : `<%! Code java; %>`

- ❖ 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
- ❖ 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

❑ Ví dụ: JSP Declaration

<%! String x; %> or
<%! String x = new String("hello"); %>



```
public class BasicDeclaration_jsp extends HttpJspBase
{
    String x;
    int y;
    String z = new String( "hello" );

    private static java.util.Vector _jspx_includes;

    public java.util.List getIncludes()
    {
        return _jspx_includes;
    }
}
```


❑ Ví dụ: JSP Declaration

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

❑ Ví dụ: JSP Declaration

EXAMPLE (cont)

```
<%!
    int num;
    public void jspInit() {
        System.out.println ("jspInit");
        num=10;
    }

    public void jspDestroy() {
        System.out.println ("jspDestroy");
        num=0;
    }

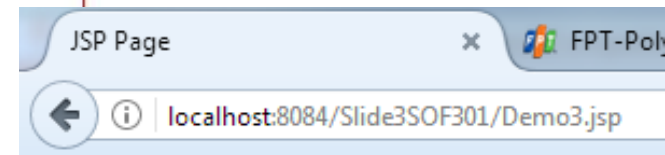
    public int add(int n) {
        System.out.println ("in Add");
        num+=n;
        return num;
    }
%>
```

jspInit

```
jspInit
in Add
jspDestroy
jspInit
in Add
in Add
```

Declarations

jspDestroy



```
Init Num: <%= num %><br/>
<% out.println("Result is: " + add(5)); %>
```

Use init and destroy

Init Num: 10
Result is: 15



DEMO

Chạy và giải thích



❑ Trong trang JSP, có thể sử dụng 2 loại cú pháp cho việc chú thích:

<code><%-- chú thích của JSP --%></code>	Nội dung chú thích được bỏ qua trong quá trình dịch JSP → Servlet. Nếu trong chú thích có các chỉ thị, câu lệnh JSP thì chúng cũng bị bỏ qua.
<code><!-- Chú thích HTML --></code>	Nội dung này vẫn được đưa vào response HTML. Nếu trong nội dung chú thích cày có các chỉ thị JSP thì các chỉ thị này vẫn được thực thi bình thường.

- ❖ 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
- ❖ Comments - Chú thích





Cảm ơn