VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

Chapter 7

JSP

Lecturer: MSc. Kiet Van Nguyen
Faculty of Information Science and Engineering
University of Information Technology, VNU-HCM

Giới Thiệu Về JSP

- Sự kết hợp các thành phần của các đối tượng HTML, XML, Servlet (mở rộng từ Servlet) và Java Code để tạo ra trang Web động
- Được xử lý trên server (server side)
- Được thực thi trên các web (application) server như **Tomcat**, JBoss, Sun, JOnAS, ...
- Sử dụng ngôn ngữ Java
- Ngôn ngữ lập trình có phân biệt chữ hoa – thường (case sensitive) và hỗ trợ các thành phần trong xử lý lập trình (JSP tag, scripting …)
- **Tách biệt giữa giao diện và phần xử lý**

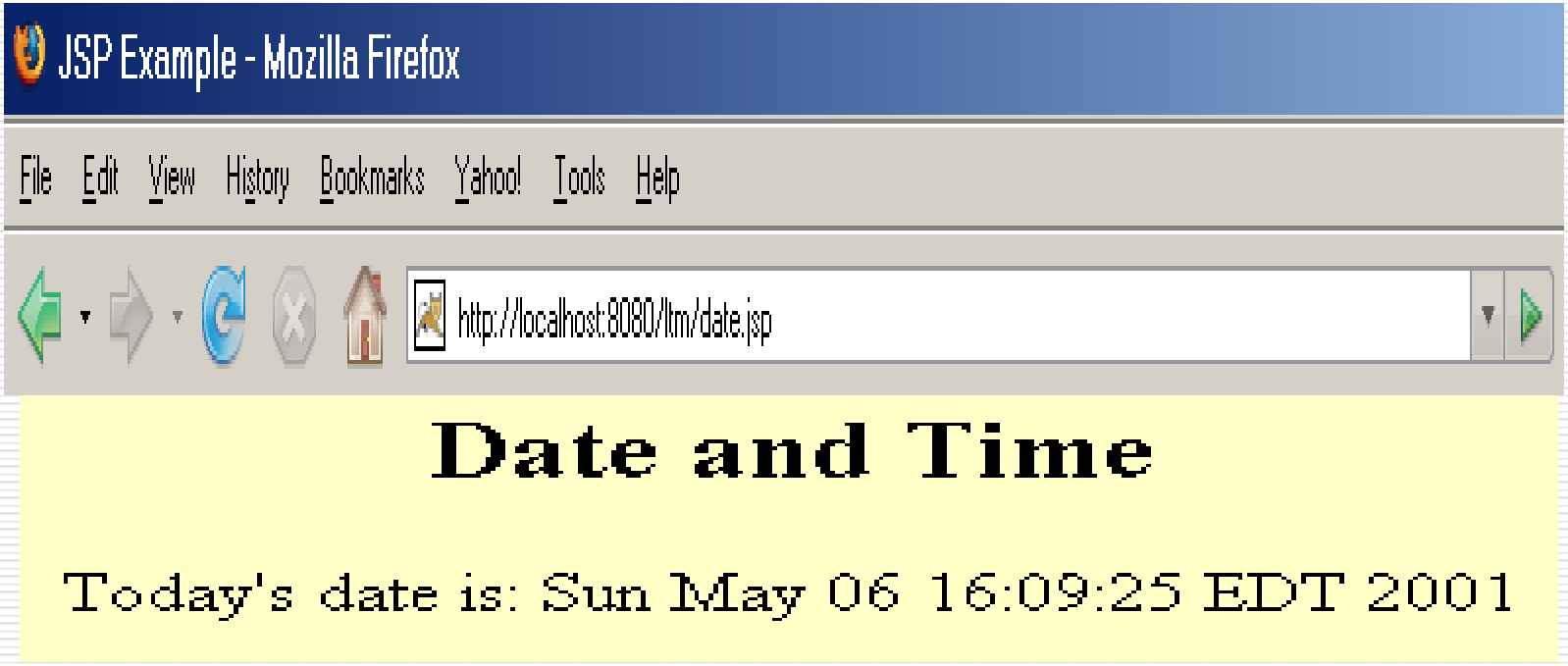
Trang JSP Mẫu

- Trang JSP đơn giản hiện thị ngày tháng hiện hành

```
1.  <HTML>
2.  <HEAD> <TITLE>JSP Example</TITLE> </HEAD>
3.  <BODY BGCOLOR="ffffcc">
4.  <CENTER>
5.  <H2>Date and Time</H2>
6.  <%
7.  java.util.Date today = new java.util.Date();
8.  out.println("Today's date is: "+today);
9.  %>
10. </CENTER>
11. </BODY>
12. </HTML>
```

- Server xử lý thành phần của JSP để chuyển đổi thành dữ liệu tĩnh trên trang HTML để trình bày trên Web browser

Trang Kết Quả



Ưu Điểm Của JSP

- Thiết kế giao diện Web dễ dàng hơn.
- Có thể viết ở một nơi và chạy ở bất cứ nơi nào.
- Cho phép tạo những trang web động.

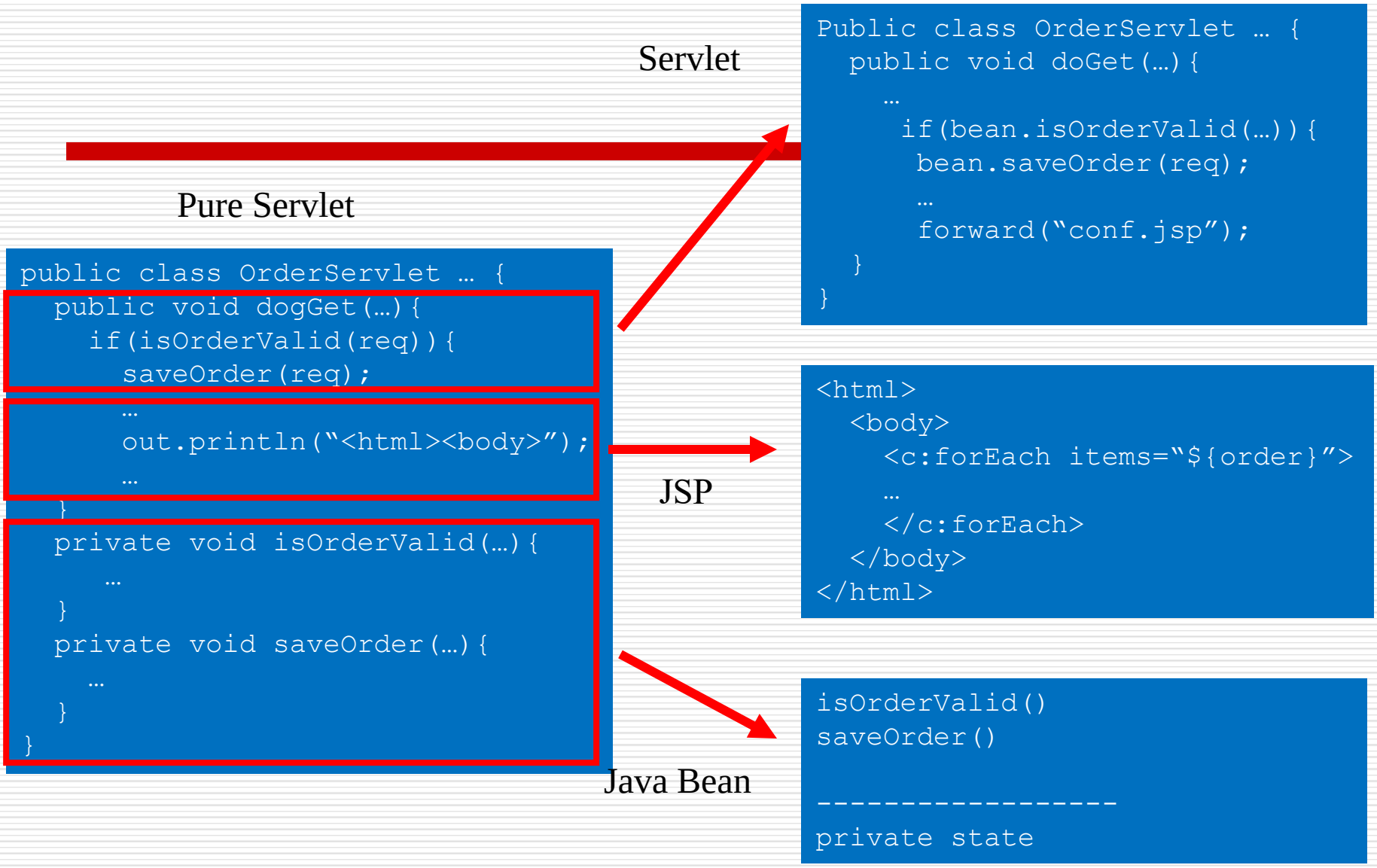
JSP vs Servlet

JSP

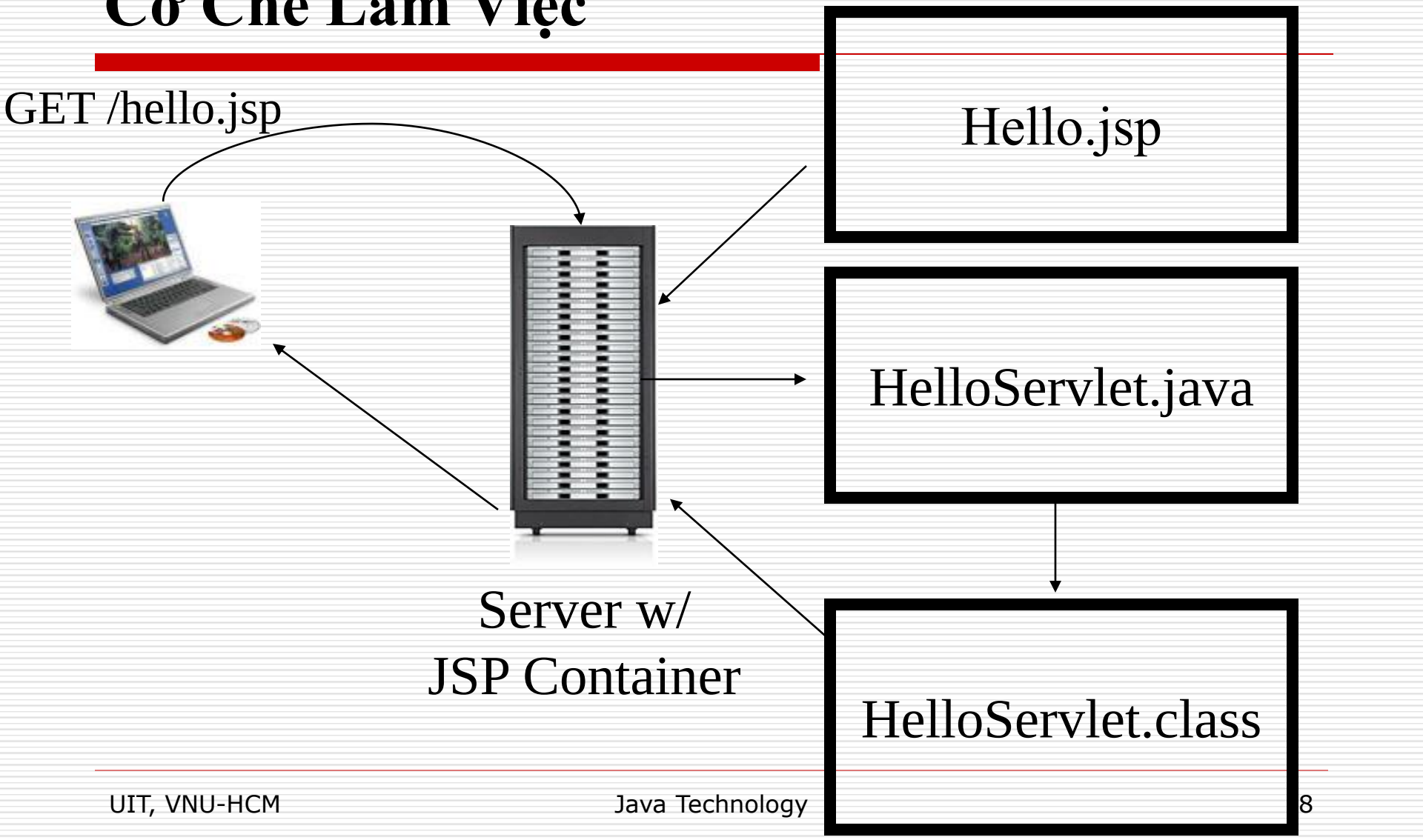
- **Mã ngôn ngữ Java nhúng trong trang HTML**
- Rất dễ soạn thảo
- Mã được biên dịch thành servlet
- Rất dễ triển khai

Servlet

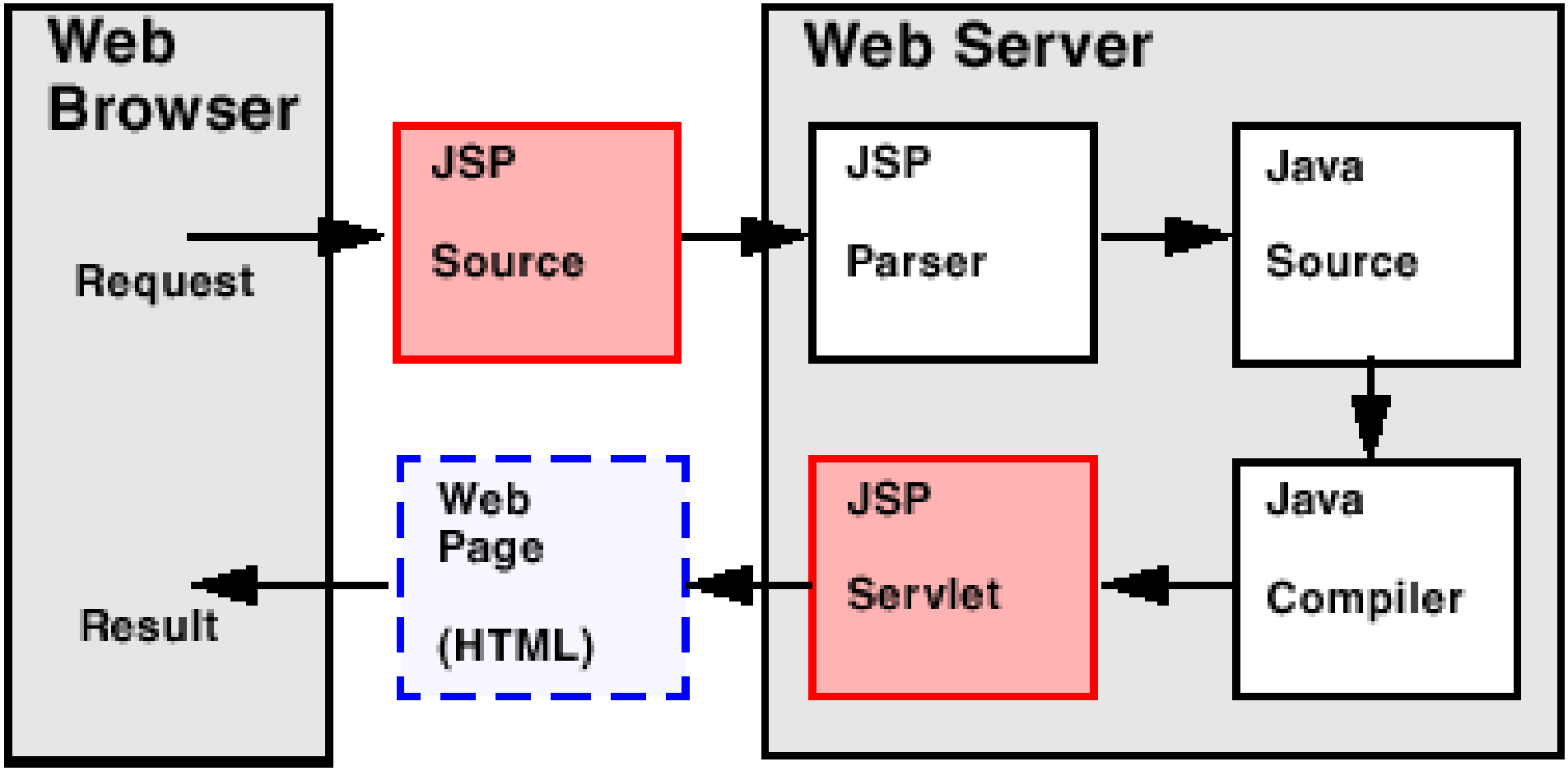
- **Mã HTML nhúng trong mã Java**
- Khó phát sinh nội dung trang web
- Đóng gói và triển khai phức tạp


Servlet
Pure Servlet
JSP
Java Bean
public class OrderServlet … {
public void dogGet(…){
if(isOrderValid(req)){
saveOrder(req);
…
out.println("<html><body>");
…
}
private void isOrderValid(…){
…
}
private void saveOrder(…){
…
}
}
Public class OrderServlet … {
public void doGet(…){
…
if(bean.isOrderValid(…)){
bean.saveOrder(req);
…
forward("conf.jsp");
}
}
<html>
<body>
<c:forEach items="${order}">
…
</c:forEach>
</body>
</html>
isOrderValid()
saveOrder()
----
private state

Cơ Chế Làm Việc



Thi Hành JSP Lần Đầu Tiên



Cấu trúc trang JSP

- Một trang JSP là 1 file *.jsp , ví dụ: “hello.jsp”
- Viết một trang JSP bằng bất kỳ trình soạn thảo nào kể cả Notepad, rồi lưu lại với tên *.jsp.
- JSP thường dùng các đối tượng Java chính quy để xử lý các giải thuật phức tạp. Các đối tượng java này sẽ được đóng gói dạng JavaBean.
- Một file JSP gồm 2 thành phần:
  - Template Text: mã HTML
  - JSP element: các thành phần của JSP

JSP File

```
<%@ page language="java" contentType="text/html" %>
```
JSP element

```
<html>
 <body bgcolor="white">
```
template text

```
 <jsp:useBean
  id="userInfo"
  class="com.ora.jsp.beans.userinfo.UserInfoBean">
  <jsp:setProperty name="userInfo" property="*"/>
 </jsp:useBean>
```
JSP element

```
  The following information was saved:
  <ul>
   <li>User Name:
```
template text

```
    <jsp:getProperty name="userInfo"
     property="userName"/>
```
JSP element

```
   <li>Email Address:
```
template text

```
    <jsp:getProperty name="userInfo"
    property="emailAddr"/>
```
JSP element

```
  </ul>
 </body>
</html>
```
template text

Thành Phần Của JSP

- **Scripting:** những đoạn mã Java mà sẽ được tạo ra trong servlet tương ứng.
- **Directive:** những câu lệnh dùng để kiểm soát cấu trúc của servlet.
- **Action:** các câu lệnh dùng để khai báo sử dụng các thành phần đã có hoặc dùng để kiểm soát hoạt động của JSP engine.

Các Giai Đoạn Trong Chu Kỳ Hoạt Động

- Translation phase
- Compilation phase
- Execution phase

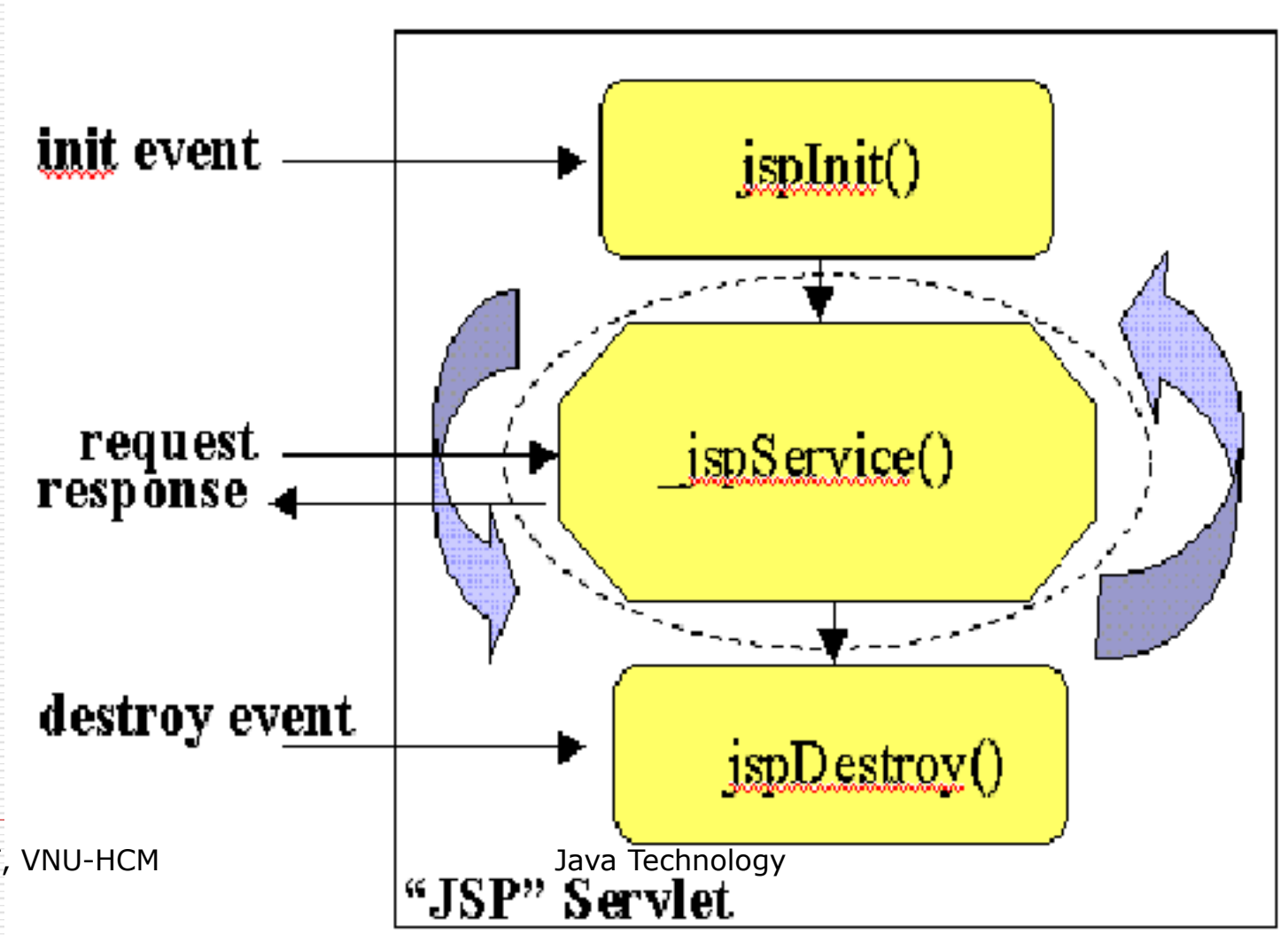
Translation/Compilation Phase

- JSP file được chuyển thành mã servlet và sau đó biên dịch mã servlet.
- Được thực hiện tự động bởi Web Container.
- Được thực hiện ở lần đầu tiên trang JSP được truy cập

Translation/Compilation Phase

- Template text được chuyển thành mã để có thể gửi vào luồng
- JSP elements:
  - **Scripting**: được chèn vào lớp servlet tương ứng của trang JSP
  - **Directive**: được dùng để chỉ thị web container cách thức dịch và thi hành trang JSP
  - **Action**: được chuyển thành các lời gọi hàm đến các thành phần JavaBean
  - Comment: các ghi chú của JSP

Execution Phase



Các Bước Xây Dựng ỨD Web với JSP

1. Viết và biên dịch các thành phần trang web (servlet, JSP) và các lớp helper
2. Tạo các tài nguyên tĩnh (trang HTML, ảnh)
3. Tạo file deploy descriptor (web.xml)
4. Đóng gói ứng dụng web (*.war)
5. Deploy ứng dụng web vào web container

1. Viết và biên dịch thành phần web

- Tạo cấu trúc cây thư mục cho ứng dụng web
- Viết mã servlet/JSP cùng các lớp helper
- Tạo file build.xml (nếu sử dụng ant)
  - Root directory
    - Build.xml (optional)
    - Context.xml (optional – application configuration)
    - Src: chứa mã Java của servlet và JavaBean
    - Web:
      - JSP pages, HTML pages, images
      - WEB-INF

2. Tạo các tài nguyên tĩnh

- Trang HTML
  - Custom page
  - Login page
  - Error page
- Tập tin ảnh được dùng trong các trang HTML hoặc JSP

3. Tạo web.xml

- Tập tin deploy descriptor chứa các thông tin hướng dẫn web container trong quá trình thi hành ứng dụng web
  - URN client dùng để truy cập vào thành phần của ứng dụng web
- Mọi trang web đều phải có web.xml

Các Đối Tượng Sẵn Có Trong JSP

- request: javax.servlet.HttpServletRequest
  - Đối tượng chứa thông tin yêu cầu của client
- response: javax.servlet.HttpServletResponse
  - Đối tượng response chứa thông tin phản hồi của server
- session: javax.servlet.http.HttpSession
  - Đối tượng session chứa thông tin về phiên làm việc của client
- application: javax.servlet.ServletContext
  - Chứa các thông tin chung của web application (chia sẻ bởi các servlet)
- out: javax.servlet.jsp.JspWriter
  - Luồng ghi ký tự của trang JSP
- pageContext : javax.servlet.jsp.PageContext
  - Đối tượng dùng để truy cập vào các thuộc tính của một trang
- page : java.lang.Object
  - Tương tực tham chiếu this

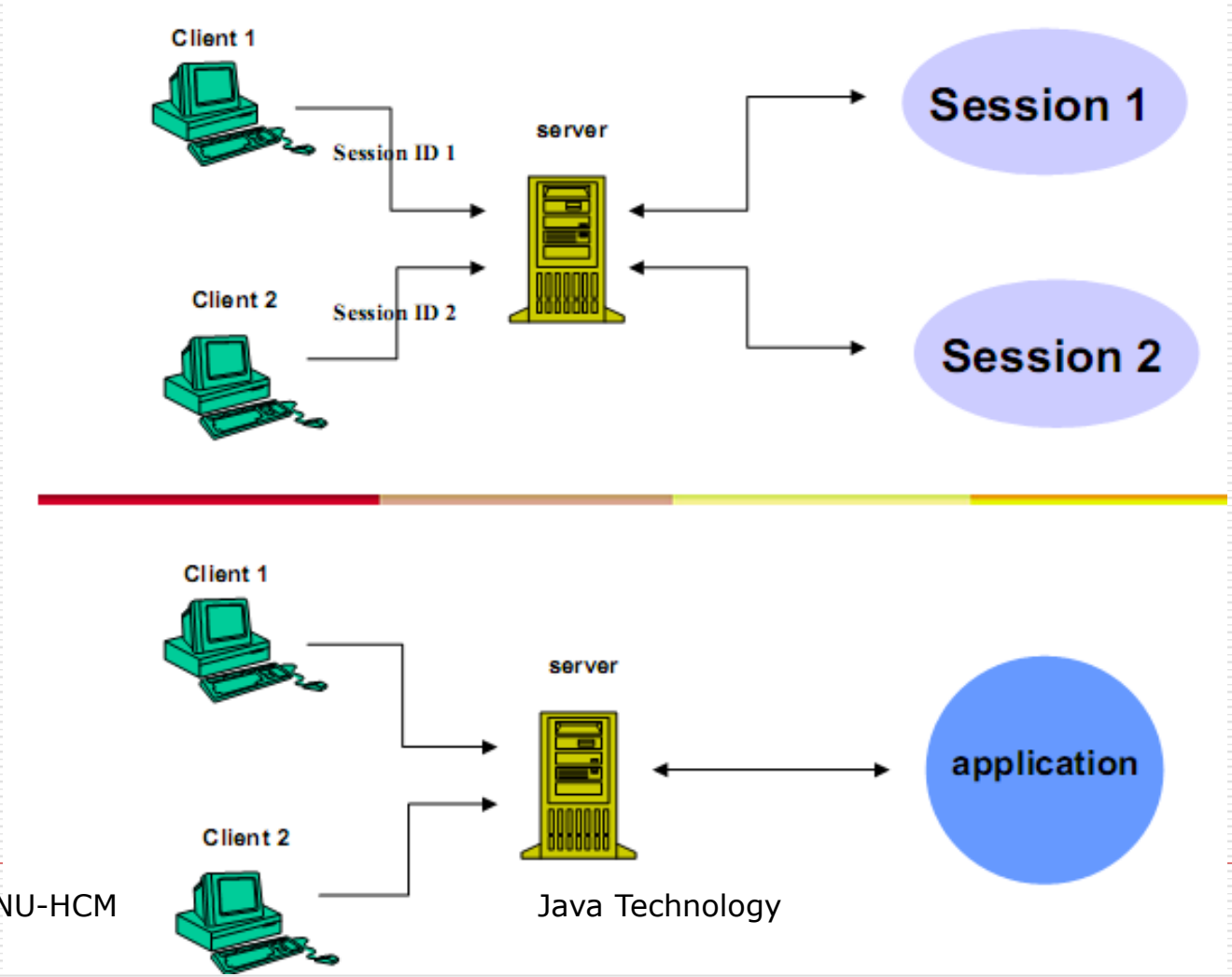
Phạm Vi Của Đối Tượng

Rộng nhất

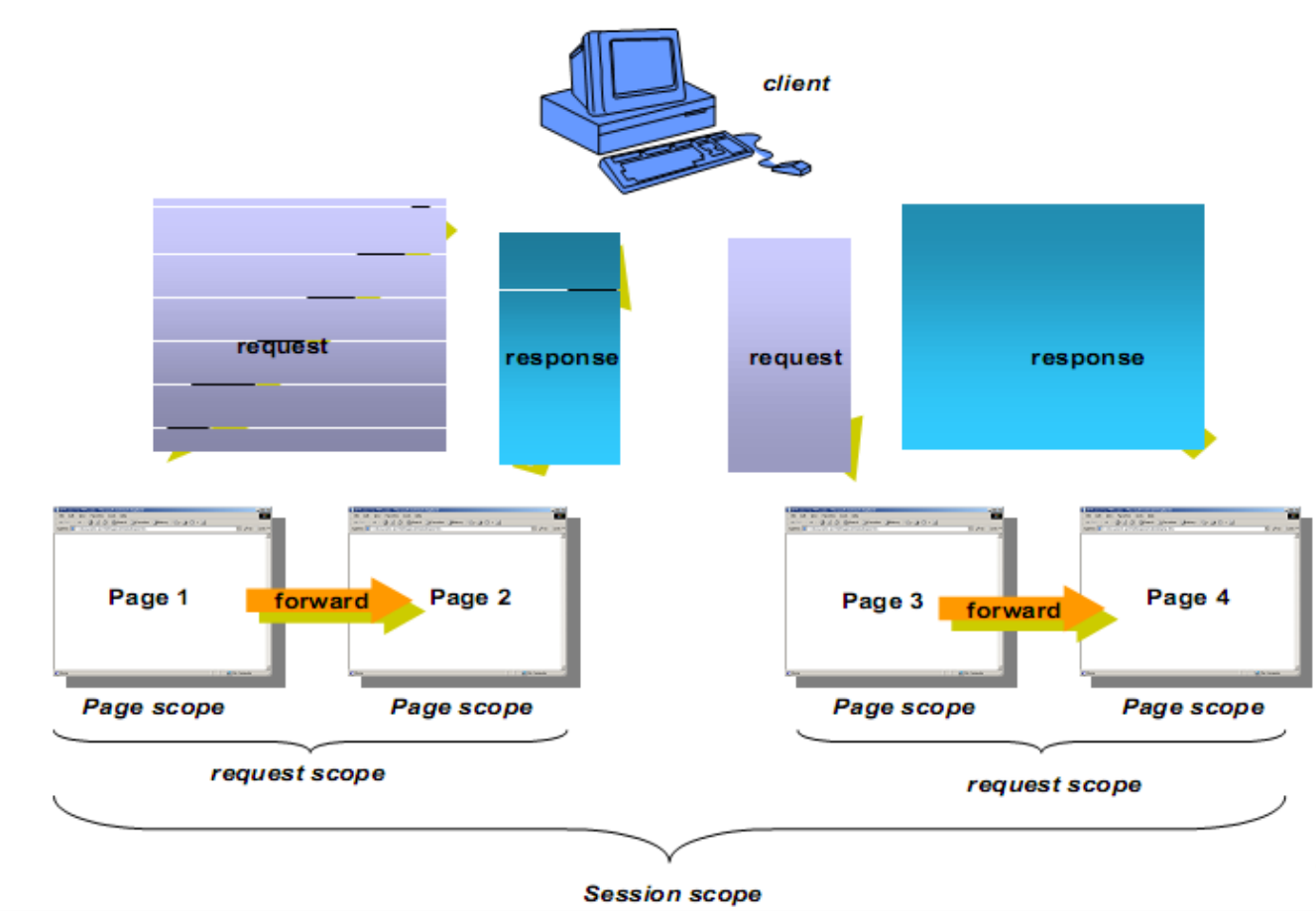
Hẹp nhất

application	**Đối tượng có thể được truy cập từ các trang trong ứng dụng**
session	Đối tượng có thể được truy cập từ các trang trong cùng 1 phiên làm việc
request	Đối tượng có thể được truy cập từ các trang đang xử lý yêu cầu
page	Đối tượng chỉ được truy cập trong nội bộ trang

Phạm Vi Session/Application



Phạm Vi Session/Request/Page



JSP SCRIPTING ELEMENT

JSP Scripting Element

- Cho phép chèn mã Java vào servlet được phát sinh từ trang JSP
- Có các dạng sau:
  - Comment: tạo các chú thích
  - Expressions: <%= expressions %>
  - Scriptlets: <% java code %>
  - Declarations: <%! Declarations %>

JSP Comment

- Có 2 dạng:
  - Output comment: lời chú thích được hiển thị trên browser

    <!-- comments ... -->
  - Secret comment: không hiển thị lên browser

    <%-- comment text --%>

Expression

- Trong quá trình thi hành:
  - Biểu thức được ước lượng giá trị và chuyển thành chuỗi.
  - Chuỗi kết quả được chèn trực tiếp vào luồng in kết quả của servlet.
  - Kết quả tương tự sử dụng out.println(expression) trong servlet
  - Có thể sử dụng các biến định nghĩa sẵn
- Cú pháp
  - **<%= expression %>** hoặc
  - **<jsp:expression>expression</jsp:expression>**

Ví Dụ Expression

- Hiển thị thời điểm hiện tại sử dụng lớp Date
  - Current time: <%= new java.uitl.Date()%>
- Hiển thị một con số ngẫu nhiên
  - Random number: <%= Math.random()%>
- Sử dụng các đối tượng có sẵn
  - Your host name: <%=request.getRemoteHost()%>
  - Your parameter:

  <%= request.getParameter("parameter")%>

Expression.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
    Transitional//EN">
<HTML>
<HEAD>
<TITLE>JSP Expressions</TITLE>
</HEAD>
<BODY>
<H2>JSP Expressions</H2>
<UL>
  <LI>Current time: <%= new java.util.Date() %>
  <LI>Your hostname: <%= request.getRemoteHost() %>
  <LI>Your session ID: <%= session.getId() %>
</BODY>
</HTML>
```

Declarations

- Cú pháp:

  <%! Declaration statement **%>**
- Declaration dùng để tạo ra các biến và đối tượng
- Các biến và đối tượng này được dùng trong các expression và scriptlet trong cùng một trang jsp.
- Declaration còn dùng để khai báo các hàm thành viên

Declaration

- Các khai báo này trở thành thành viên dữ liệu hoặc phương thức thành viên trong servlet được tạo ra từ trang jsp sau giai đoạn translation (nằm ngoài phương thức _jspService()).
- Ví dụ:

  <%! int i; **%>**

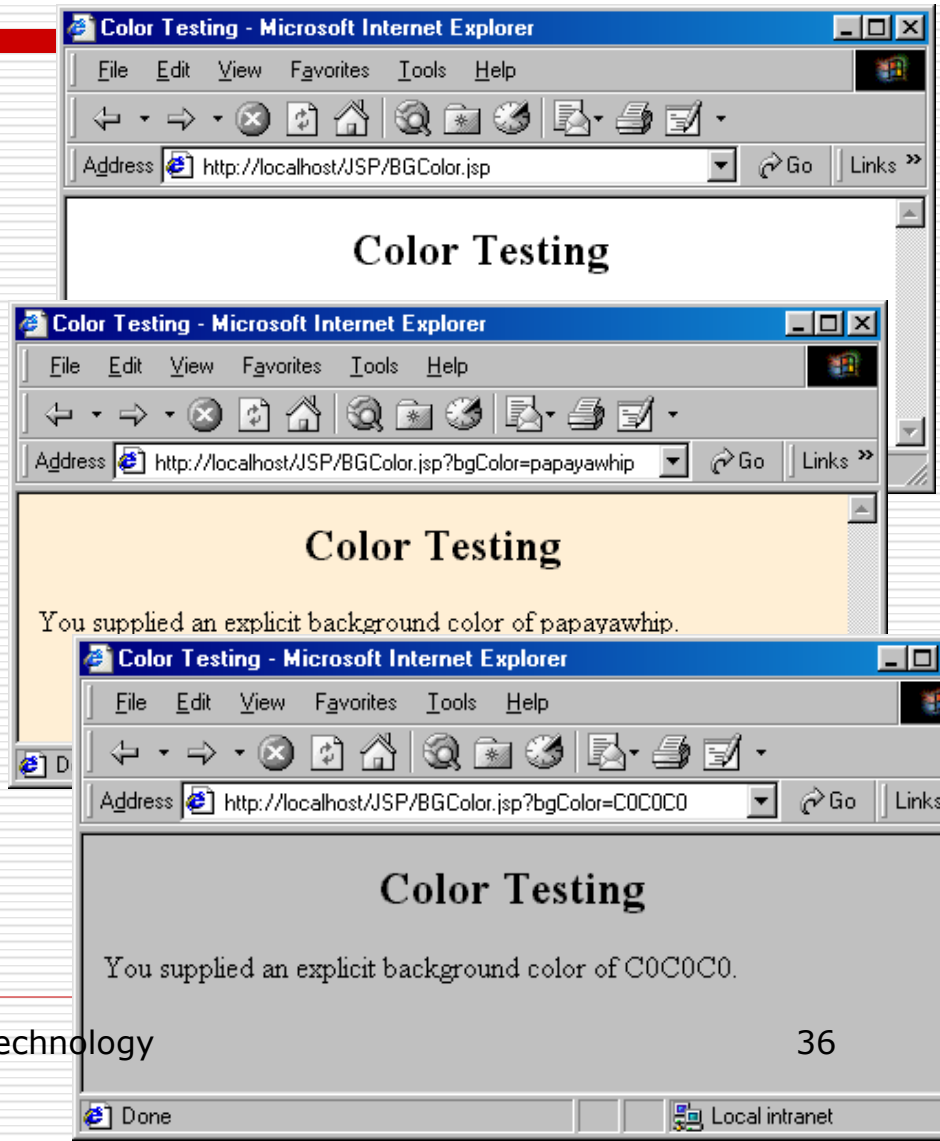
- Có thể dùng các **access modifiers: *public***, ***protected*** and ***private***
- **<%!** private int salary=5000; **%>**

Scriptlet

- Cú pháp: <% java code %>
- Scriptlet cho phép chèn mã java vào phương thức _jspService() trong servlet được phát sinh sau giai đoạn translation.
- Mã java trong scriptlet được thi hành khi có yêu cầu (request) gửi lên server.
- Biến được khai báo bên trong scriptlet được xem là biến local.

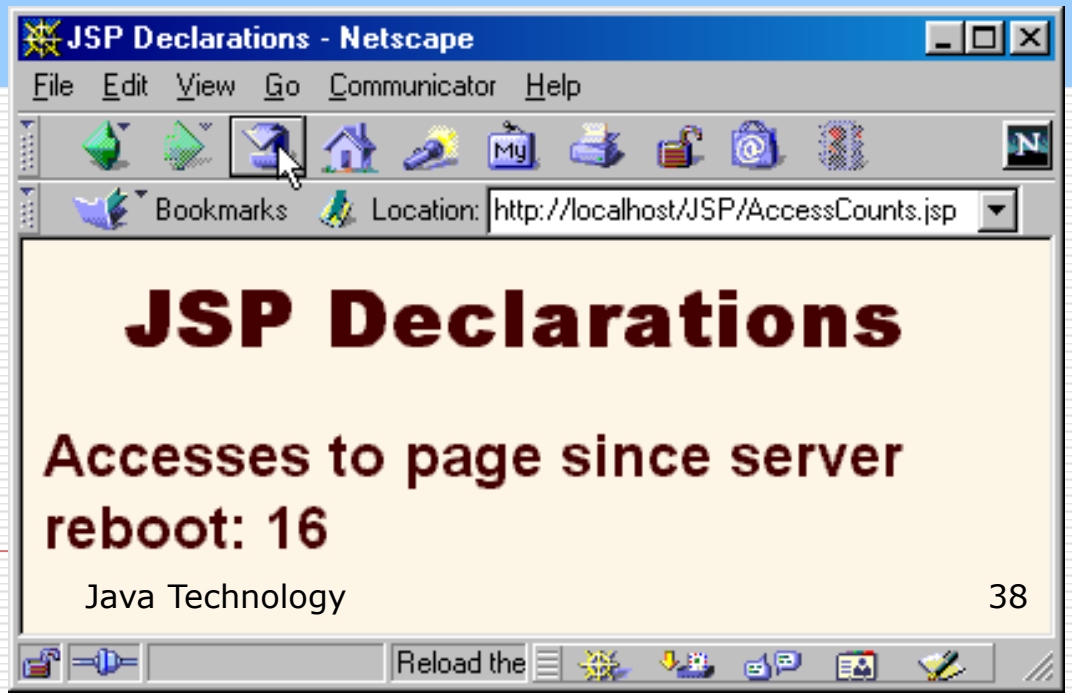
Example Using JSP Scriptlets

```
<HTML>
<HEAD>
  <TITLE>Color Testing</TITLE>
</HEAD>
<%
String bgColor =
    request.getParameter("bgColor");
boolean hasExplicitColor;
if (bgColor != null) {
  hasExplicitColor = true;
} else {
  hasExplicitColor = false;
  bgColor = "WHITE";
}
%>
<BODY BGCOLOR="<%= bgColor %>">
```



Example Using JSP Declarations

```
...
<body>
<h1>JSP Declarations</h1>
<%! private int accessCount = 0; %>
<h2>Accesses to page since server reboot:
<%= ++accessCount %></h2>
</body></html>
```



Q&A

Cảm ơn!