Hackademics Hanoi
Hack your passion, Code your life!

CHƯƠNG 13
JAVASERVER PAGES

Nội dung

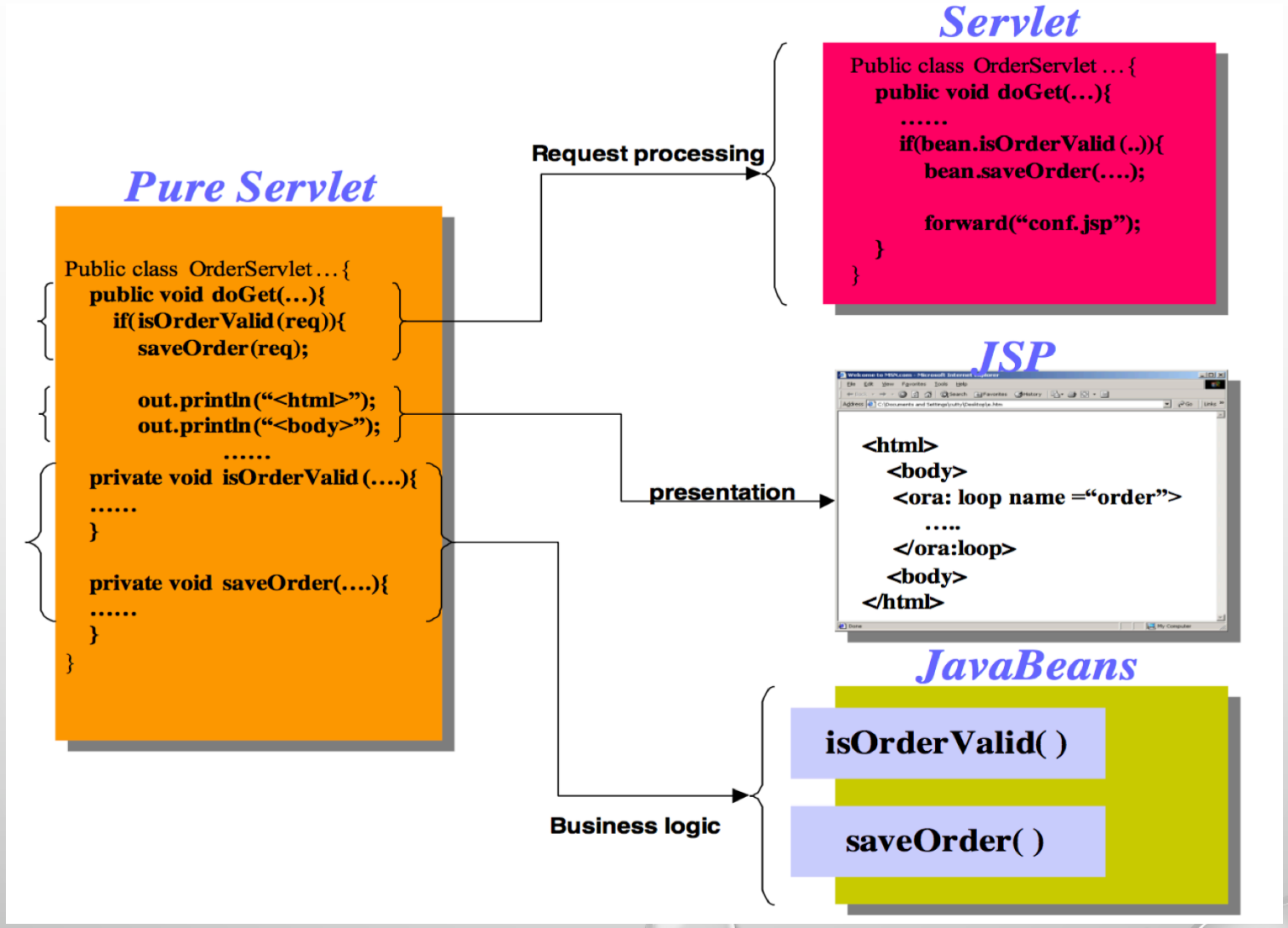
1. Giới thiệu về JSP
2. JSP và Servlet
3. Vòng đời của JSP
4. JSP Tag
5. Biến ẩn trong trang JSP
6. Session
7. Cookie

1. Giới thiệu về JSP

Giới thiệu về Servlet

- 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

Kiến trúc jsp



ví dụ

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

Blue: static,
Red: Dynamic contents

2. So sánh JSP và Servlet

Servlet và JSP

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 sinh các trang HTML
  - Khó cho lập trình giao diện: khó được giao diện đẹp, khó maintain, bảo trì
  - Khi thay đổi, phải biên dịch lại, (đóng gói lại), deploy lại

Servlet và JSP

- Servlet rất mạnh về xử lý và điều phối, nhưng Servlet lại rất yếu về tạo giao diện

JSP:

- 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

→ Dễ dàng hơn rất nhiều cho lập trình viên

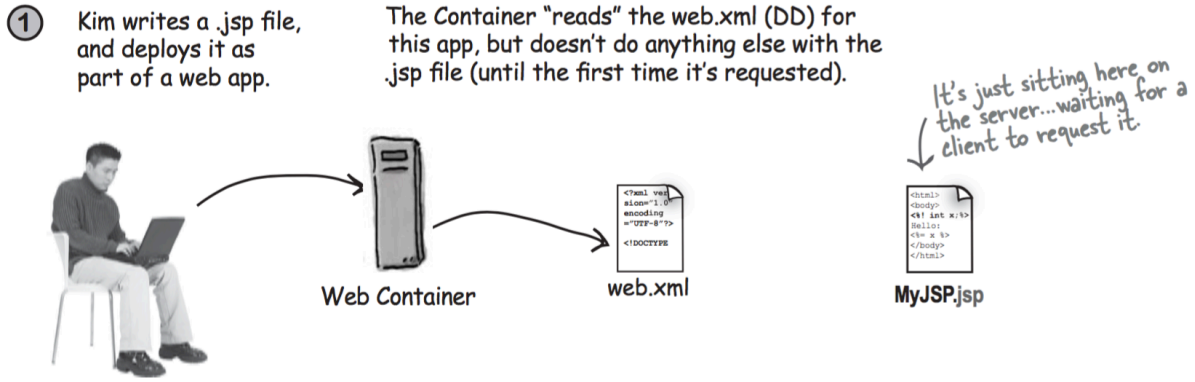
- Thuận lợi
  - Tiện khi tạo ra trang web HTML
- Có nhiều công cụ hỗ trợ thiết kế HTML
- Phân cách thiết kế web và xử lý mã nguồn java
  - Đội thiết kế HTML chuyên giao diện hơn lập trình viên java
- Thuận lợi hơn Servlet
  - Thuận tiện trong việc tạo ra trang web HTML
  - Sử dụng các công cụ thiết kế như DreamWeaver
  - Phân cách xử lý và giao diện
- JSP ra đời để thay thế Servlet?

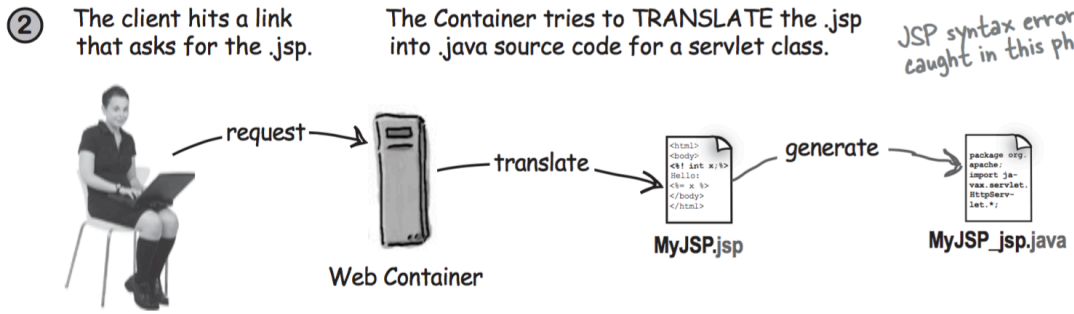
NÊN DÙNG JSP HAY SERVLET?

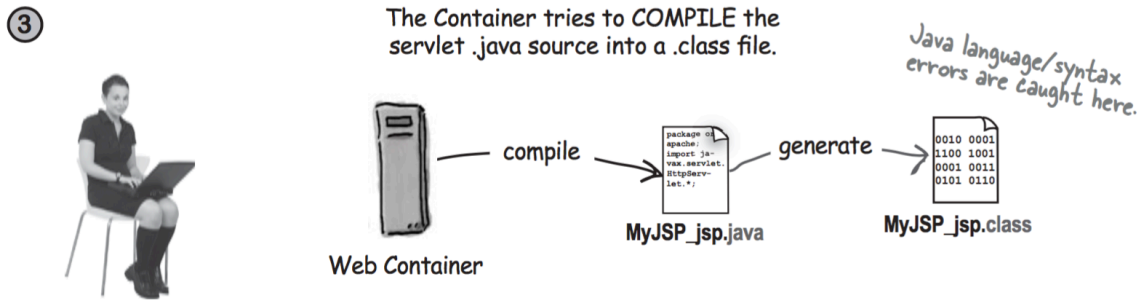
- Servlet mạnh về xử lý nghiệp vụ và điều phối nhưng lại rất yếu về hiển thị
- SP mạnh về xử lý hiển thị nhưng lại yếu về xử lý nghiệp vụ và điều phối
- Cần khai thác đồng thời 2 điểm mạnh công nghệ
- Trong thực tế, cả servlet và JSP được sử dụng trong mẫu thiết kế MVC (Model-View- Controller)
  - Servlet xử lý phần Controller
  - JSP xử lý phần View

2. Vòng đời của JSP

Vòng đời của JSP

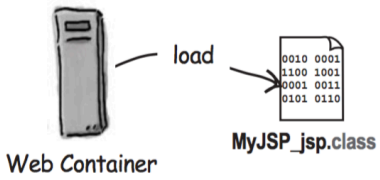






④

The Container LOADS the newly-generated servlet class.

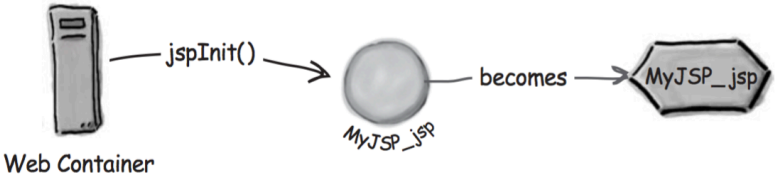


---

⑤

The Container instantiates the servlet and causes the servlet's jspInit() method to run.

The object is now a full-fledged servlet, ready to accept client requests.



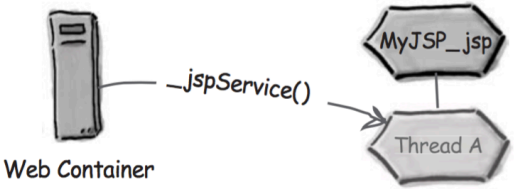
---

⑥

The Container creates a new thread to handle this client's request, and the servlet's _jspService() method runs.

Everything that happens after this is just plain old servlet request-handling.

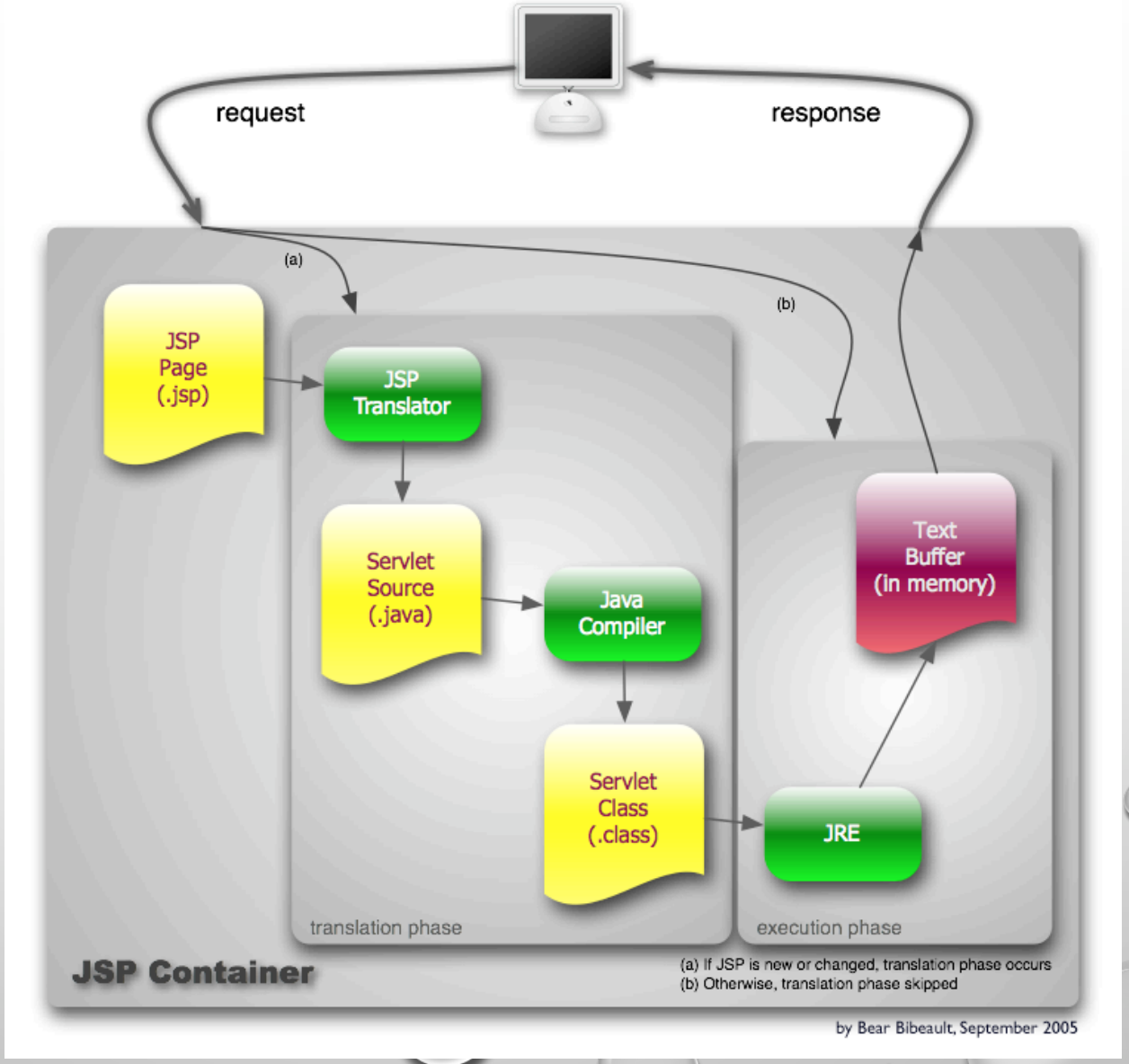
Eventually the servlet sends a response back to the client (or forwards the request to another web app component).



Các phương thức trong vòng đời

Các giai đoạn:

- Translation
- Compile
- Execution



3. JSP Tag

- Deriective: <%@ directive attribute="value" %>
- Declarations: <%! int i = 0; %>
- Scriptlet: <% code %>
- Expression: <%= expression %>
- Action: <jsp:action_name attribute="value" />
- Comment: <%-- This is JSP comment --%>

DERIECTIVE

- Cung cấp thông tin tổng quát về các trang JSP cho Jsp engine

Có ba loại:

- <%@ page ... %> Định nghĩa một thuộc tính page-dependent (phụ thuộc trang), như ngôn ngữ scripting, trang lỗi và các yêu cầu bộ đệm
- <%@ include ... %> Include một file trong suốt giai đoạn phiên dịch
- <%@ taglib ... %> Khai báo một thư viện thẻ, chứa các action tùy biến, được sử dụng trong trang đó

DECLARATIONS

- Khai báo biến và các phương thức có thể được sử dụng trong các trang JSP
- Cấu trúc:
  - <%! // code %>
  - Ví dụ:
    - <%! int i = 0; %>
    - <%! int a, b, c; %>
    - <%! Circle a = new Circle(2.0); %>

SCRIPTLET

- Là đoạn mã Java được nhúng vào trong các trang JSP
- Được thực thi mỗi khi trang web được truy cập
- Cú pháp: <% //code %>
- Ví dụ:

```
<%
out.println("Your IP address is " + request.getRemoteAddr());
%>
```

EXPRESSION

- Một phần tử Expression trong JSP chứa một biểu thức ngôn ngữ Scripting mà được tính toán, được biến đổi thành một String, và được chèn tại nơi Expression đó xuất hiện trong JSP file.
- Bởi vì giá trị của một Expression được biến đổi thành một String, bạn có thể sử dụng một Expression bên trong một dòng text, có hoặc không nó được tag với HTML, trong một JSP file.
- Cú pháp: <%= %>

ACTION

- jsp:include
- jsp:useBean
- jsp:setProperty
- jsp:getProperty
- jsp:forward
- jsp:plugin
- jsp:element
- jsp:attribute
- jsp:body
- jsp:text

COMMENT

- <%-- comment --%> Một JSP comment. Được bỏ qua bởi JSP engine
- <!-- comment --> Một HTML comment. Được bỏ qua bởi trình duyệt
- <\% Biểu diễn một static <% literal.
- %\> Biểu diễn một static %> literal.
- \' Một trích dẫn đơn trong một thuộc tính mà sử dụng các trích dẫn đơn
- \" Một trích dẫn kép trong một thuộc tính mà sử dụng các trích dẫn kép

5. Biến ẩn

Biến ẩn

- **Đối tượng**
- request: Đây là đối tượng **HttpServletRequest** mà liên kết với Request
- response: Đây là đối tượng **HttpServletResponse** mà liên kết với Response tới Client
- out: Đây là đối tượng **PrintWriter** được sử dụng để gửi output tới Client
- session: Đây là đối tượng **HttpSession** mà liên kết với Request
- application: Đây là đối tượng **ServletContext** mà liên kết với application context

Biến ẩn

- config: Đây là đối tượng **ServletConfig** mà liên kết với page
- pageContext: Sự gói gọn này sử dụng các đặc trưng Server-Spcific giống như hiệu năng cao **JspWriters**.
- page: Được sử dụng để gọi các phương thức được định nghĩa bởi lớp Servlet đã được phiên dịch
- Exception: Đối tượng **Exception** cho phép dữ liệu exception để được truy cập bởi JSP đã chỉ rõ

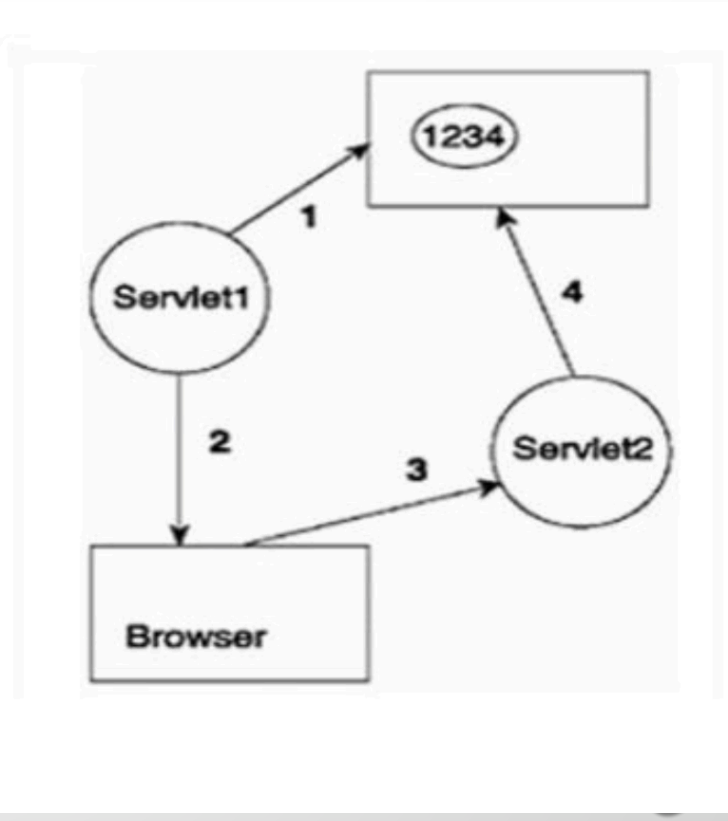
6. SESSION

Trạng thái

- Thông tin trong quá trình tương tác giữa client – server
- Giao thức http không có trạng thái: không biết được các yêu cầu gửi cùng một client hay không
- Tuy nhiên, một số trường hợp ứng dụng web cần lưu trữ trạng thái của client
- Ví dụ:
  - trạng thái người dùng
  - Shopping cart

SESSION

- Session giúp server lưu trữ trạng thái giao tiếp của client
- Session bắt đầu khi client gửi yêu cầu đầu tiên đến web application và kết thúc khi client đóng kết nối hoặc quá thời gian tương tác(time out)
- Cơ chế hoạt động:
  - Server tạo ra session ID cho lần yêu cầu đầu tiên và gửi cho client
  - Client gửi yêu cầu lại sẽ chứa session ID và server dựa trên session ID để xác định trạng thái giao dịch của client đó



Giao diện httpsession

- package javax.servlet.http
- Hiện thực bởi các servlet container và cung cấp cách theo dõi trạng thái giao dịch của client
- servlet container tạo ra đối tượng session khi bắt đầu phiên giao dịch

Sử dụng đối tượng HttpSession

1. Khởi tạo session đối với người dùng.

•Ta khởi tạo ra một đối tượng của HttpSession.

2. Lưu trữ và lấy dữ liệu từ đối tượng HttpSession.

•Sau khi người dùng tiến hành đăng nhập xong, ta sẽ gán giá trị cho session đã khởi tạo dữ liệu của người dùng (thường là username, password..v.v).

- + Để gán dữ liệu cho session ta dùng phương thức: putValue(),
- + Để lấy dữ liệu từ session ta dùng phương thức getValue();

•3. Vô hiệu hoá Session

•Sau khi người dùng đã đăng xuất hoặc tắt trình duyệt ..v.v ta sẽ vô hiệu hoá session bằng cách gọi ra phương thức: invalidate().

phương thức của HttpSession hay dùng

- getCreationTime() : Lấy ra thời gian khởi tạo.
- getID() : Lấy ra id của session.
- getLastAccessedTime(): lấy ra thời gian lần truy cập cuối cùng.
- getValueName() : lấy ra tên của đối tượng.
- getValue() : lấy ra dữ liệu được lưu trong session.
- setAttribute : Gán dữ liệu cho session.// Thực ra thì trước đây người ta dùng phương thức putValue() nhưng do nó quá cũ và không đáp ứng được nhu cầu mới cho nên đã bị thay thế/
- invalidate() : Vô hiệu hoá session.
- removeValue() : Xoá dữ liệu đã gán vào cho session.

7. Cookie

COOKIE

- Cookie là các text file được lưu giữ trên máy tính Client và chúng được giữ cho mục đích theo dõi các thông tin đa dạng. Rõ ràng một điều là, Java Servlet hỗ trợ các HTTP cookie.
- Có 3 bước liên quan trong việc nhận diện việc phản hồi người dùng:
  - Server script gửi một tập hợp các Cookie tới trình duyệt. Ví dụ: name, age, hoặc số chứng minh thư, …
  - Trình duyệt lưu giữ thông tin này trên thiết bị nội bộ để sử dụng trong thời gian tới.
  - Trong lần tới, trình duyệt gửi bất kỳ yêu cầu nào tới Web server, thì nó gửi những thông tin Cookie này tới Server và Server này sử dụng thông tin đó để nhận diện người dùng.

Thiết lập Cookie với Servlet

3 bước sau để thiết lập Cookie với Servlet:

- **Tạo một đối tượng Cookie**
  - Cookie cookie = new Cookie("key","value");
- **Thiết lập tuổi tối đa**
  - cookie.setMaxAge(60*60*24);
- **Gửi Cookie vào trong các trường HTTP Response Header:**
  - response.addCookie(cookie);

Đọc xóa Cookie

- Đọc Cookie với Servlet:

```
Cookie[] cookies = null;
// Get an array of Cookies associated with this domain
cookies = request.getCookies();
```

- Xóa Cookie với Servlet: Nếu bạn muốn xóa một Cookie, đơn giản bạn chỉ cần theo 3 bước sau:
  - Đọc một cookie đang tồn tại và lưu nó trong đối tượng Cookie.
  - Thiết lập tuổi của cookie về 0 bởi sử dụng phương thức **setMaxAge()** để xóa một cookie đang tồn tại.
  - Thêm cookie này trở lại bên trong trường Response header.

Thực hành

HẾT CHƯƠNG 13