CHƯƠNG VIII
LẬP TRÌNH WEB VỚI JAVA SERVLET - JSP

Chương 8: Lập trình web với java

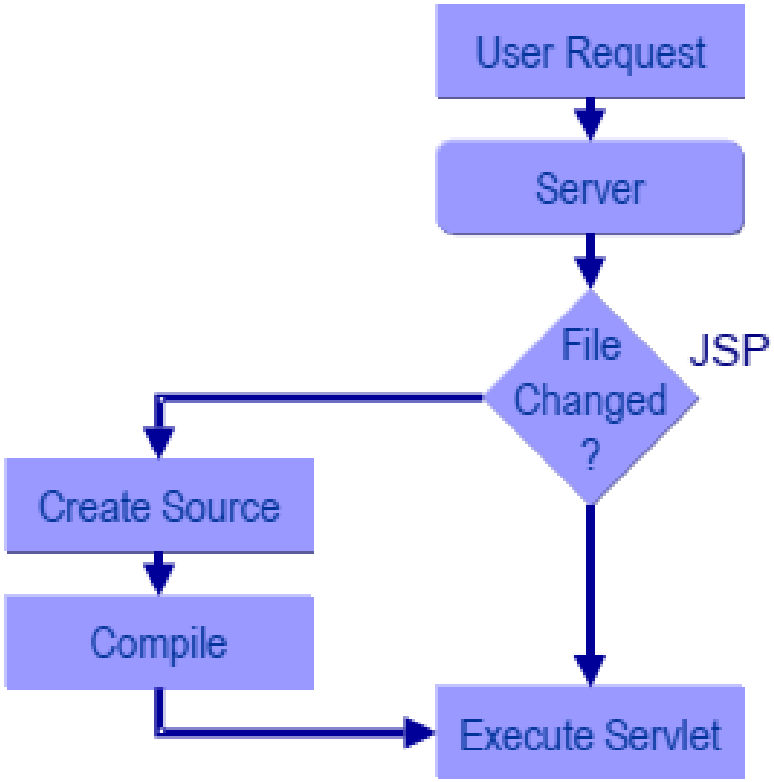
1. Tổng quan nền tảng Java EE 7
2. Web server - Servlet
3. Ngôn ngữ JSP

Ngôn ngữ JSP

- Trang JSP
  - 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

Ngôn ngữ JSP

- Vòng đời của 1 trang JSP



Ngôn ngữ JSP

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

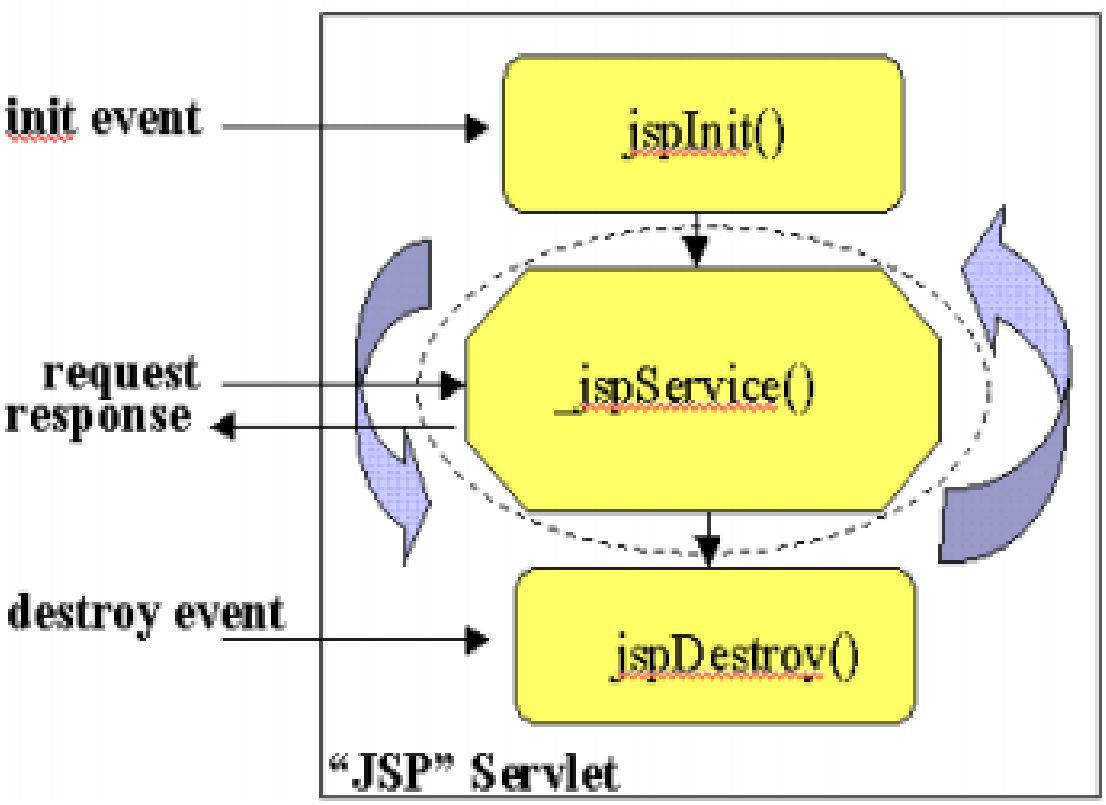
Giai đoạn Translation/Compilation

- 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)
- 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

Giai đoạn Translation/Compilation(tt)

- 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 phơng thức tới JavaBeans components

Các phương thức trong giai đoạn Execution



Các phương thức trong giai đoạn Execution

Khởi tạo trang

- 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`

Các phương thức trong giai đoạn Execution

Kết thúc trang

- 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 phương thức trong giai đoạn Execution

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 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

Gọi mã nguồn Java sử dụng JSP scripting elements

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 %>

Gọi mã nguồn Java sử dụng JSP scripting elements

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

Gọi mã nguồn Java sử dụng JSP scripting elements

Expressions, ví dụ:

- Hiển thị thời gian hiện tại sử dụng lớp Date
  - Current time: <%= new java.util.Date() %>
- Hiển 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() %>

Gọi mã nguồn Java sử dụng JSP scripting elements

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>

Gọi mã nguồn Java sử dụng JSP scripting elements

Scriptlets, ví dụ:

- Hiển 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"); %>
```

Gọi mã nguồn Java sử dụng JSP scripting elements

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 _jspSevice()
  - Declerations 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>

Including và Forwarding tới tài nguyên Web khác

- 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

Including và Forwarding tới tài nguyên Web khác

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" %>

Including và Forwarding tới tài nguyên Web khác

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>
```

Including và Forwarding tới tài nguyên Web khác

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"/>

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}* %>

Directives

Có 3 loại

- 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" %>

Directives

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" %>

Implicit Object

- 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)

Directives

Implicit Objects

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

JavaBeans

- 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)

Qui 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

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>
```

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

Thiết lập thuộc tính cho JavaBeans(tt)

- 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 %>"/>

Truy cập các thuộc tính

- 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"

JavaBeans

Truy cập các thuộc tính, convert –> string -> chèn vào out

- 2 cách
  - Qua scriptlet
    - <%= beanName.getPropName() %>
  - Qua JSP:setProperty
    - <jsp:getProperty 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

Truy cập các thuộc tính không convert

- 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) {
%>
```