Hackademics Hanoi
Hack your passion, Code your life!

CHƯƠNG 14 – JSP ADVANCED

Nội dung

1. Khái niệm về Custom Tag
2. Sử dụng lại các thành phần
3. JavaBeans
4. Expression Language (EL)
5. JSP Standard Tag Library (JSTL) Servlet Context

1. CUSTOM TAG

Hackademics Hanoi
Hack your passion, Code your life!

Giới thiệu

- Thẻ JSP có phần hạn chế và không cung cấp hỗ trợ cho các trình bày logic cần thiết cho định dạng dữ liệu động
- Công nghệ JSP cung cấp một tính năng cho phép nhà phát triển tạo ra các thẻ mới và định nghĩa hành vi của chúng
- Các thẻ người dùng định nghĩa được gọi là Custom Tags

Custom Tag

- Custom tags có thể làm gì
  - Được tùy biến thông qua các thuộc tính truyền từ trang JSP gọi chúng
  - Có thể truyền lại tham số cho trang đang gọi
  - Truy cập được tất cả các đối tượng có trong trang JSP
  - Được lồng vào nhau, giao tiếp thông qua các biến cục bộ
- Một số Custum Tag:
  - Thiết lập / truy cập các Implicit object
  - Xử lý forms
  - Truy cập database
  - Điều khiển (rẽ nhánh, vòng lặp...)
  - ...

Tại sao cần Custom Tag

- Tách biệt phần presentation với phần business logic (và các chức năng khác)
  - Thiết kế trang: thực hiện các presontation
  - Business logic developers: tạo các custom tags
- Đóng gói phần bussiness logic
  - Tái sử dụng và bảo trì dễ dàng
- Đơn giản cho người thiết kế trang
  - Không cần biết Java, sử dụng cú pháp đơn giản

Các thành phần

- Lớp xử lý Tag (Tag handler class)
  - Định nghĩa các hành động với tag
- Tag library descriptor (TLD)
  - Ánh xạc các phần tử XML tới lớp xử lý tag
- JSP file
  - Sử dụng các tags

- Tag handler
  - Là một đối tượng được quản lý bởi Container
  - Tag hadnler đánh giá hành động tùy chỉnh trong khi thực hiện một trang JSP
  - Là một lib Java thực hiện giao diện Tag, IterationTag hoặc BodyTag
  - Trong gói javax.servlet.jsp.tagext

- Tag library
  - Một tập của các hành động gói gọn một số chức năng được sử dụng bên trong một trang JSP
  - Chúng ta sẽ thiết kế và phát triển một tập hợp các thẻ làm việc cùng nhau và giúp giải quyết các yêu cầu thường gặp
  - Một bộ các thẻ tùy chỉnh như vậy được gọi là mọt thư viện thẻ

- Tag library description
  - Khi chúng ta sử dụng các thẻ tùy chỉnh trong một trang JSP, JSP engine cần phải biết các lớp xử lý cho các thẻ này
  - Tong đó thẻ thư viện chúng nằm ở đâu và làm thế nào chúng được sử dụng
  - Siêu dữ liệu này được lưu trữ trong một tập tin gọi là thư viện thẻ mô tả (TLD)

Cài đặt – sử dụng

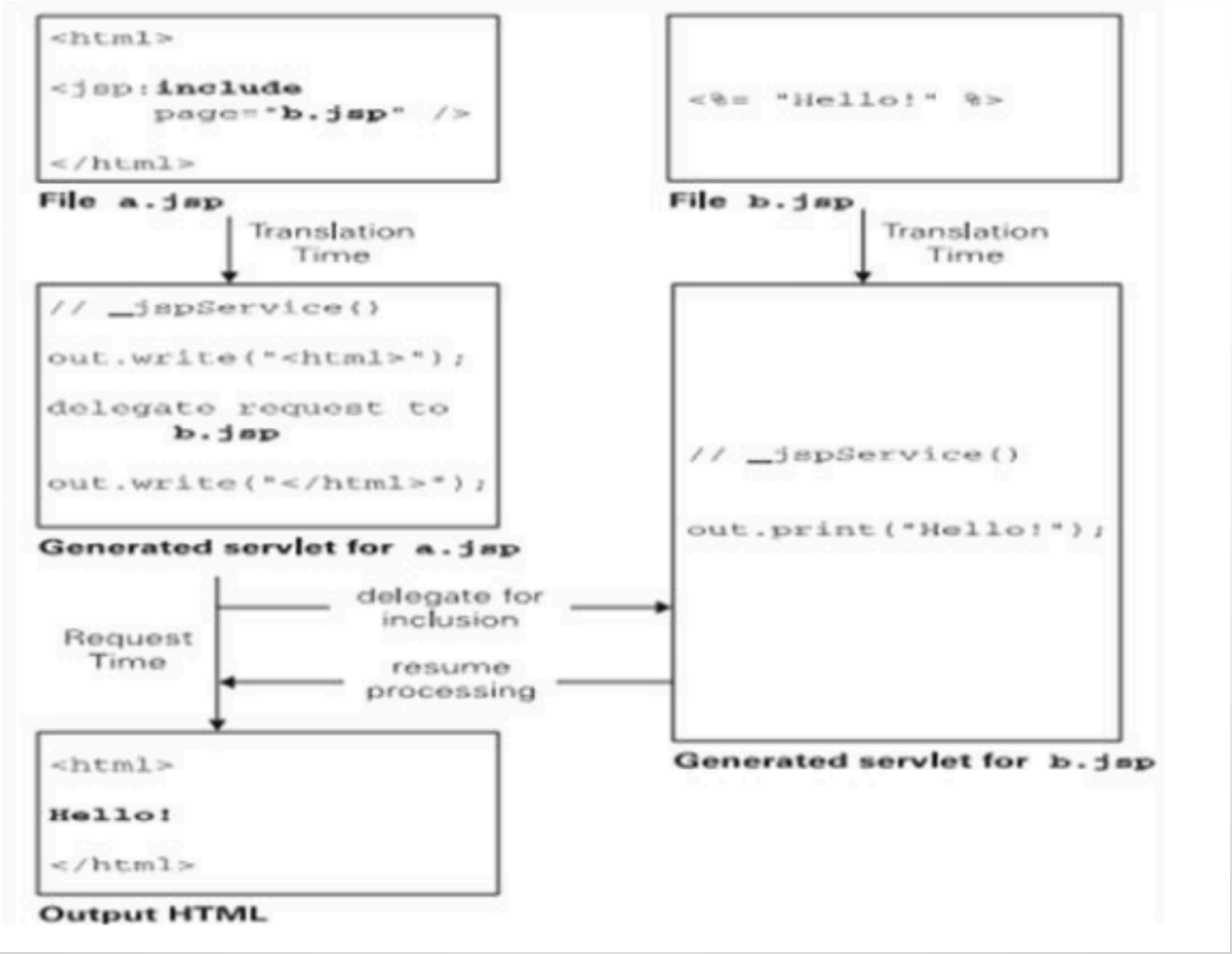
- Cài đặt các Custom Tag
  - Viết các lớp xử lý tag
  - Viết các file Tag Libraries Descriptor (TLD)
  - Xây dựng lớp Tag Handler
- Cấu hình TLD: 1 trong 2 cách sau
  - Khai báo trực tiếp thông qua thành phần **directive taglig** của trang JSP: **<%@ taglib** uri="**/WEB-INF/tlds/boDuaTag.tld**" prefix="**bd**" %>
  - Hoặc khai báo trong **web.xml**
- Sử dụng các Custum Tag
  - Viết các trang JSP sử dụng Tag

```
<tag>
  <name>custom</name>
  <tag-class>customTag.customTagHandler</tag-class>
  <body-content>empty</body-content>
  <attribute>
    <name>ten</name>
  </attribute>
  <attribute>
    <name>kieuNgay</name>
  </attribute>
</tag>
```

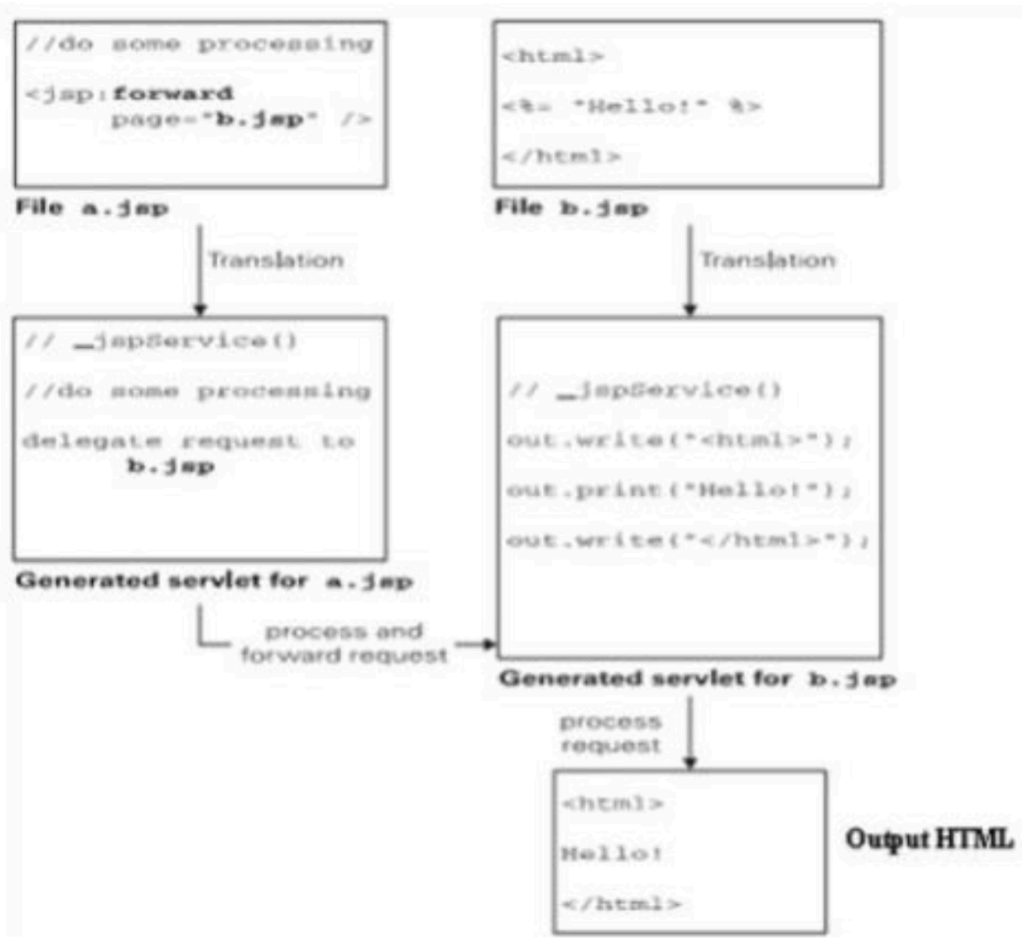
2. Sử dụng lại các thành phần web

JSP bao gộp trang JSP khác

- Đôi khi chèn thêm các nội dung tĩnh hoặc động giống nhau trong web resource: thêm copyright information, footer, banner, widget,...
- Static resource:
  - Dùng JSP Directive
  - Thuộc tính file tham chiếu đến file có thể là text, html, jsp, xml,..
  - Đường dẫn là đường dẫn tương đối
- Dynamic inclusion:
  - Dùng JSP action: <jsp:include page="relativeURL">
  - Đường dẫn tương đối có thể là biểu thức: <jsp:include page="<%= pageURL %>">
  - Có thể truyền đối số cho dynamic include


<html>
<jsp:include
page="b.jsp" />
</html>
File a.jsp
Translation Time
// _jspService()
out.write("<html>");
delegate request to
b.jsp
out.write("</html>");
Generated servlet for a.jsp
<%= "Hello!" %>
File b.jsp
Translation Time
// _jspService()
out.print("Hello!");
Generated servlet for b.jsp
Request Time
delegate for inclusion
resume processing
<html>
Hello!
</html>
Output HTML

JSP Foward


//do some processing
<jsp:forward
page="b.jsp" />
File a.jsp
<html>
<%= "Hello!" %>
</html>
File b.jsp
Translation
Translation
// _jspService()
//do some processing
delegate request to
b.jsp
Generated servlet for a.jsp
// _jspService()
out.write("<html>");
out.print("Hello!");
out.write("</html>");
process and
forward request
Generated servlet for b.jsp
process
request
<html>
Hello!
</html>
Output HTML

FORWARD

Làm thế nào để Forwarding tới một web resource khác?

- Lấy đối tượng RequestDispatcher từ đối tượng HttpServletRequest

```
RequestDispatcher dispatcher =
request.getRequestDispatcher("/template.jsp");
```

- Nếu cần giữ lại URL gốc để xử lý thêm, có thể lưu lại thành thuộc tính request

- Sau đó, gọi phương thức forward() của đối tượng RequestDispatcher với tham số là đối tượng request và response

```
dispatcher.forward(request, response);
```

CÁC THẺ CHUẨN TĂC

- <jsp:include>
  - Include output của 1 file JSP khác
- <jsp:forward>
  - Forwards request đến trang HTML/JSP hoặc servlet khác
- <jsp:plugin>
  - Downloads Java plugin về trình duyệt để thực thi applet hoặc bean
- <jsp:useBean>
  - Khởi tạo một đối tượng JavaBean
- <jsp:getProperty>: Truy cập các thuộc tính bean
- <jsp:setProperty>: Thiết lập các thuộc tính bean

3. JavaBeans

Hackademics Hanoi
Hack your passion, Code your life!

JAVABEANS

- Mô hinh các thành phần phần mềm được thiết kế sao cho dễ tái sử dụng và thay thế nhau
- Các thành phần tái sử dụng được viết trong java và đóng gói các hành vi, trạng thái
- JavaBeans định nghĩa ra những giao diện và những lớp trong package java.bean để tạo ra bean và điều khiển các thuộc tính của nó.

Cấu trúc của javabeans

- Là lớp java phải có hàm khởi tạo không có đối số
- Tất cả các thuộc tính phải được đặc tả private
- Phương thức getter để truy cập giá trị thuộc tính nào đó trong bean
- Phương thức setter để thiết lập giá trị cho một thuộc tính nào đó trong bean

Ví dụ

```java
public class Person {
    private String username;
    private String password;
    private int age;
    private String hometown;

    public Person(String username, String password, int age, String hometown) {
        this.username = username;
        this.password = password;
        this.age = age;
        this.hometown = hometown;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getHometown() {
        return hometown;
    }

    public void setHometown(String hometown) {
        this.hometown = hometown;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

}
```

4. Expression Language (EL)

Hackademics Hanoi
Hack your passion, Code your life!

EXPRESSION LANGUAGE

- EL cung cấp cấu trúc cho phép truy xuất dữ liệu của JavaBean, maps, array và danh sách được lưu trữ trong các thuộc tính của ứng dụng web
- JSP EL cho phép bạn tạo ra các biểu thức cả số học và logic.
- Biểu thức JSP EL có thể sử dụng số nguyên, số thực, chuỗi, boolean và giá trị null

Ưu điểm EL

- Ngắn gọn và súc tích hơn JSP chuẩn
- Cho phép truy cập các thuộc tính lồng cấp
- Cho phép truy cập collections như maps, arrays, list
- Xử lý giá trị null tốt hơn JSP chuẩn
- Cung cấp nhiều chức năng

Nhược điểm

- Không tạo ra JavaBean nếu chưa tồn tại
- Không cung cấp cách để thiết lập giá trị cho thuộc tính

CÚ pháp

- Định dạng

  ${ Expression }

- Expression có thể là:
  - Literals
  - Operators
  - Variables (đối tượng tham chiếu)
  - Gọi hàm thông qua các thuộc tính

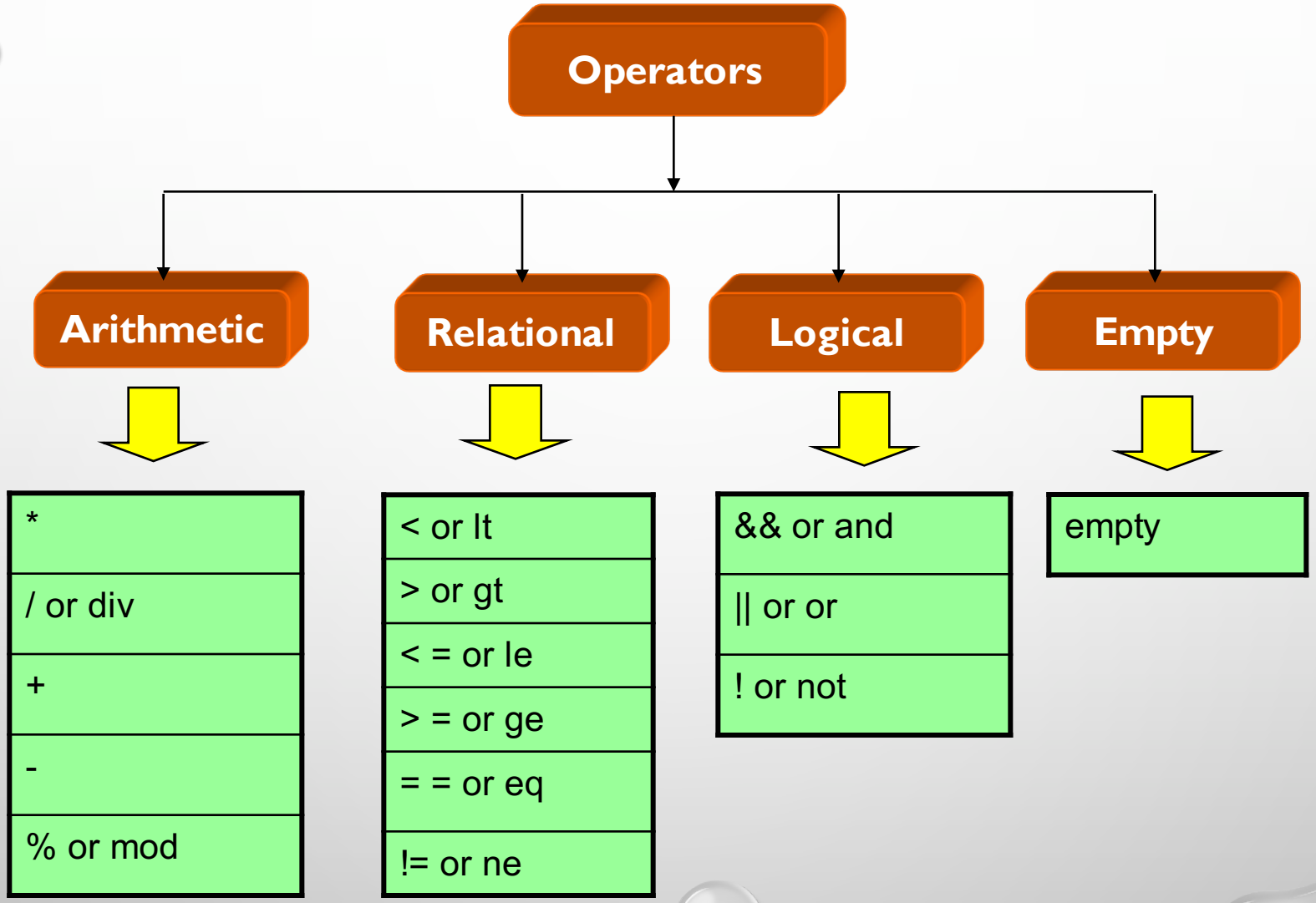
EL Literals

Literals	Literal Values
Boolean	true or false
Integer	Similar to Java e.g. 243, -9642
Floating Point	Similar to Java e.g. 54.67, 1.83
String	Any string delimited by single or double quote e.g. “hello” , ‘hello’
Null	null

${ true }

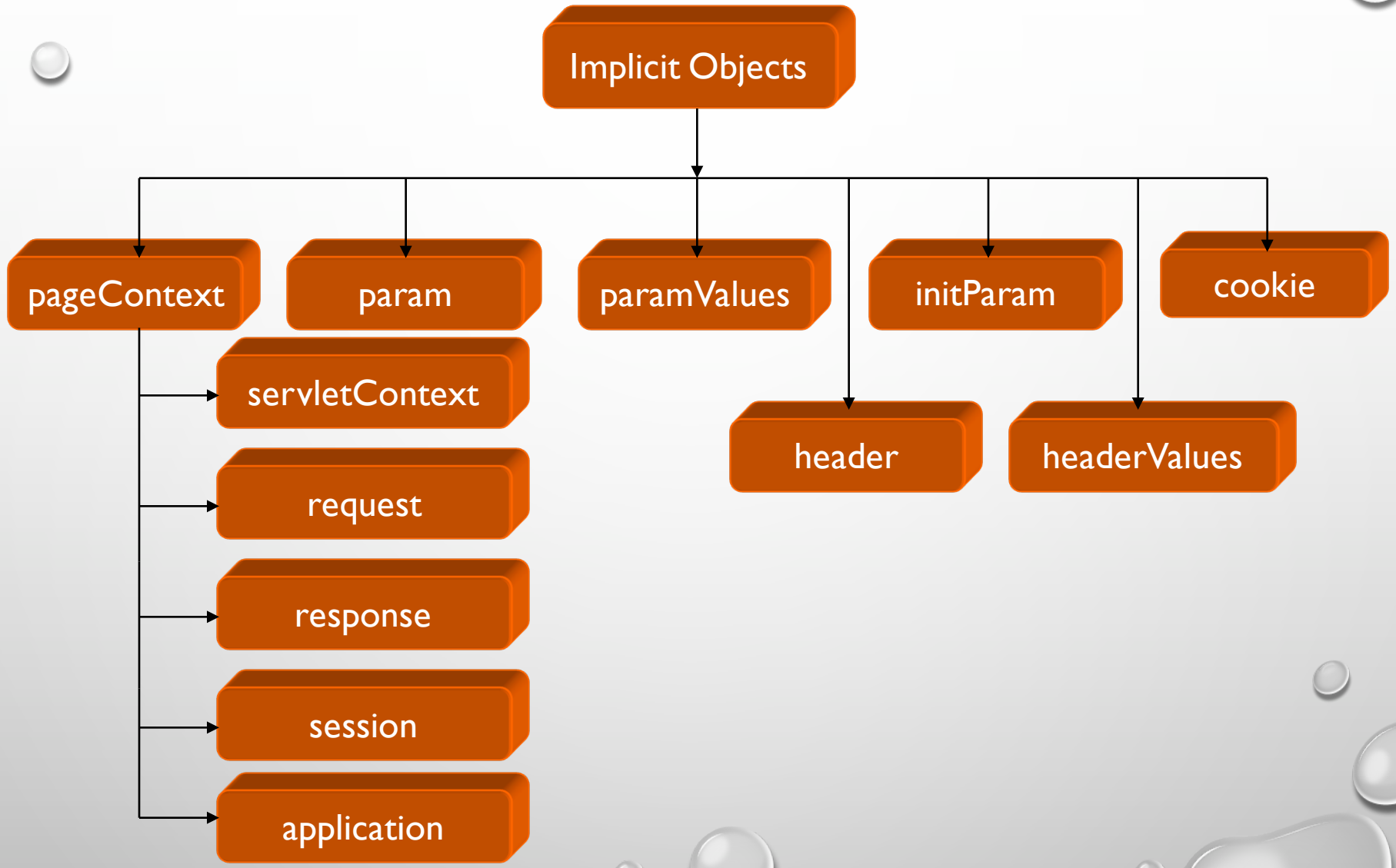
${ 10*10 }

EL OPERATOR



Hackademics Hanoi
Hack your passion, Code your life!

EL IDENTIFIERS



EL IDENTIFIERS

Category	Implicit Object	Description
JSP	pageContext	used to access JSP implicit objects
Scopes	pageScope	A *Map* associating names & values of page scoped attributes
	requestScope	A *Map* associating names & values of request scoped attributes
	sessionScope	A *Map* associating names & values of session scoped attributes
	applicationScope	A *Map* associating names & values of page scoped attributes

EL IDENTIFIERS

Category	Operator	Description
Request Parameters	Param	Maps a request parameter name to a single String parameter value
	paramValues	Maps a request parameter name to an array of values
Request Headers	header	Maps a request header name to a single header value
	headerValues	Maps a request header name to an array of values
Cookies	cookie	A *Map* storing the cookies accompanying the request by name
Initialization parameters	initParam	A *Map* storing the context initialization parameters of the web application by name

EL IDENTIFIERS

- ${ pageContext.response }
  - Trả về đối tượng response của JSP
- ${ param.name }
  - Tương ứng với request.getParameter(“name”)
- ${ cookie.name.value}
  - Tương đương:

```
if(cookie.getName()).equals(“name”){
        String val = cookie.getValue();
}
```

TOán tử “.”

- Được dùng để truy cập các thuộc tính của đối tượng
- Example:
  - ${ person.name}
  - ${ person.city.name}

TOán tử [ ]

- Với array và collection list
  - Dùng để truy cập các phần tử
  - ${ arrayList[n] } : truy cập phần tử thứ n
- Collection Map:
  - ${ Map[“key”] }

5. JSP Standard Tag Library (JSTL)

Hackademics Hanoi
Hack your passion, Code your life!

JSTL

- JavaServer Pages Standard Tag Library (JSTL) là một tập hợp các thẻ JSP hữu ích, mà gói các tính năng lõi phổ biến tới các ứng dụng JSP.
- JSTL hỗ trợ tới các tác vụ phổ biến và có tính cấu trúc, ví dụ như các tính lặp và điều kiện, các thẻ để thao tác tài liệu XML, các thẻ đa ngôn ngữ, và các thẻ SQL.

  Nó cũng cung cấp Framework để tích hợp các **Custom Tags** với các thẻ JSTL.

JSTL

- Các thẻ JSTL có thể được phân loại, theo tính năng của nó, thành các nhóm thư viện thẻ JSTL sau, mà có thể được sử dụng khi tạo một JSP page:
  - **Core Tags:** Nhóm thẻ cơ bản
  - **Formatting tags:** Nhóm thẻ định dạng
  - **SQL tags:** Nhóm thẻ SQL
  - **XML tags:** Nhóm thẻ XML
  - **JSTL Functions:** Nhóm hàm JSTL

NHÓM CORE TAGS

- <%@ taglib prefix="c" uri="http://java.sun.com/jsp/html/core" %>

Thẻ	Miêu tả
Thẻ <c:out > trong JSTL	Giống <%= ... >, nhưng cho các Expression
Thẻ <c:set > trong JSTL	Thiết lập kết quả của một ước lượng Expression trong một 'scope'
Thẻ <c:remove > trong JSTL	Gỡ bỏ một biến mục tiêu (từ một biến scope cụ thể, nếu đã xác định)
Thẻ <c:catch> trong JSTL	Bắt bất kỳ Throwable mà xuất hiện trong thân của nó và trưng bày nó một cách tùy ý
Thẻ <c:if> trong JSTL	Thẻ điều kiện đơn giản, mà ước lượng phần thân của nó nếu điều kiện đã cho là true
Thẻ <c:choose> trong JSTL	Thẻ điều kiện đơn giản mà thiết lập một context cho các hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise>
Thẻ <c:when> trong JSTL	Thẻ phụ của <choose> mà include phần thân của nó nếu điều kiện được ước lượng là true
Thẻ <c:otherwise > trong JSTL	Thẻ phụ của <choose> mà theo sau thẻ <when> và chỉ chạy nếu tất cả điều kiện trước được ước lượng là 'false'
Thẻ <c:import> trong JSTL	Thu nhận một URL tuyệt đối hoặc quan hệ và trưng bày nội dung của nó tới hoặc trang đó, một String trong 'var', hoặc một Reader trong 'varReader'
Thẻ <c:forEach > trong JSTL	Thẻ lặp cơ bản, chấp nhận nhiều kiểu tập hợp khác nhau và hỗ trợ subsetting (chia tập con) và tính năng khác
Thẻ <c:forTokens> trong JSTL	Lặp qua các token, được phân biệt bởi các dấu phân tách (delimiter) đã cung cấp
Thẻ <c:param> trong JSTL	Thêm một parameter tới một URL của thẻ đang chứa 'import'
Thẻ <c:redirect > trong JSTL	Redirect tới một URL mới
Thẻ <c:url> trong JSTL	Tạo một URL với các tham số truy vấn tùy ý

NHÓM FORMATTING TAGS

- <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/html/fmt" %>

Thẻ	Miêu tả
Thẻ <fmt:formatNumber> trong JSTL	Trả lại giá trị số với định dạng cụ thể
Thẻ <fmt:parseNumber> trong JSTL	Parse biểu diễn chuỗi của một số, tiền tệ, phần trăm
Thẻ <fmt:formatDate> trong JSTL	Định dạng một date/time bởi sử dụng Style và Pattern đã cho
Thẻ <fmt:parseDate> trong JSTL	Parse biểu diễn chuỗi của một date/time
Thẻ <fmt:bundle> trong JSTL	Tải một Resource Bundle để được sử dụng bởi phần thân thẻ
Thẻ <fmt:setLocale> trong JSTL	Lưu giữ Locale đã cho trong biến cấu hình locale
Thẻ <fmt:setBundle> trong JSTL	Tải một Resource Bundle và lưu giữ nó trong biến scope đã đặt tên hoặc biến cấu hình bundle
Thẻ <fmt:timeZone> trong JSTL	Xác định timezone cho bất kỳ định dạng time nào hoặc parse các action được lặp trong phần thân của nó
Thẻ <fmt:setTimeZone> trong JSTL	Lưu giữ timezone đã cung cấp biến cấu hình time zone đó
Thẻ <fmt:message> trong JSTL	Hiển thị một thông báo đa ngôn ngữ
Thẻ <fmt:requestEncoding> trong JSTL	Thiết lập mã hóa ký tự cho request

NHÓM SQL TAGS

- <%@ taglib prefix="sql" uri="http://java.sun.com/jsp/html/sql" %>
- Nhóm SQL Tags trong JSTL cung cấp các thẻ để tương tác với các Relational Database (RDBMSs), ví dụ như Oracle, MySQL, hoặc Microsoft SQL Server.

Thẻ	Miêu tả
Thẻ <sql:setDataSource> trong JSTL	Tạo một DataSource đơn giản chỉ thích hợp cho prototype
Thẻ <sql:query> trong JSTL	Thực thi SQL query được định nghĩa trong phần thân của nó hoặc thông qua thuộc tính sql
Thẻ <sql:update> trong JSTL	Thực thi SQL update được định nghĩa trong phần thân của nó hoặc thông quan thuộc tính sql
Thẻ <sql:param> trong JSTL	Thiết lập một parameter trong một lệnh SQL tới giá trị đã xác định
Thẻ <sql:dateParam> trong JSTL	Thiết lập một parameter trong một lệnh SQL tới giá trị java.util.Date đã xác định
Thẻ <sql:transaction > trong JSTL	Cung cấp các phần tử database action được lặp với một Connection đã chia sẻ, thiết lập để thực thi tất cả các lệnh

NHÓM XML TAGS

- Nhóm XML Tags trong JSTL cung cấp một cách để tạo và thao tác các tài liệu XML.

<%@ taglib prefix="x" uri="http://java.sun.com/jsp/html/xml" %>

Thẻ	Miêu tả
Thẻ <x:out> trong JSTL	Giống <%= ... >, nhưng dành cho các XPath Expression
Thẻ <x:parse> trong JSTL	Sử dụng để *parse* XML data được xác định hoặc thông qua một thuộc tính hoặc trong phần thân thẻ
Thẻ <x:set > trong JSTL	Thiết lập một biến tới giá trị của một XPath expression
Thẻ <x:if > trong JSTL	Ước lượng XPath Expression và nếu nó là true, thì nó xử lý phần thân thẻ. Nếu điều kiện là false, phần thân bị bỏ qua
Thẻ <x:forEach> trong JSTL	Lặp qua các node trong một tài liệu XML
Thẻ <x:choose> trong JSTL	Thẻ điều kiện đơn giản mà thiết lập một context cho hoạt động điều kiện loại trừ, được đánh dấu bởi <when> và <otherwise> trong JSTL
Thẻ <x:when > trong JSTL	Thẻ phụ của <choose> mà bao phần thân thẻ nếu Expression được ước lượng là 'true'
Thẻ <x:otherwise > trong JSTL	Thẻ phụ của <choose> mà theo sau các thẻ <when> và chỉ chạy khi nếu tất cả điều kiện trước là 'false'
Thẻ <x:transform > trong JSTL	Áp dụng một phép biến đổi XSL trên một tài liệu XML
Thẻ <x:param > trong JSTL	Sử dụng cùng với thẻ transform để thiết lập một parameter trong XSLT stylesheet

NHÓM JSTL FUNCTIONS

<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/html/functions" %>

Hàm	Miêu tả
Hàm fn:contains() trong JSTL	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho
Hàm fn:containsIgnoreCase() trong JSTL	Kiểm tra nếu một chuỗi input chứa chuỗi phụ đã cho trong trường hợp không phân biệt kiểu chữ
Hàm fn:endsWith() trong JSTL	Kiểm tra nếu một chuỗi input kết thúc với suffix đã cho
Hàm fn:escapeXml() trong JSTL	Các ký tự thoát mà có thể được phiên dịch như XML markup
Hàm fn:indexOf() trong JSTL	Trả về index bên trong một chuỗi về sự xuất hiện đầu tiên của chuỗi phụ
Hàm fn:join() trong JSTL	Kết hợp tất cả phần tử trong một mảng thành một chuỗi
Hàm fn:length() trong JSTL	Trả về số item trong một tập hợp, hoặc số ký tự trong một chuỗi
Hàm fn:replace() trong JSTL	Trả về một chuỗi là kết quả của việc thay thế một chuỗi input với một chuỗi đã cho
Hàm fn:split() trong JSTL	Chia một chuỗi thành một mảng các chuỗi phụ
Hàm fn:startsWith() trong JSTL	Kiểm tra nếu một chuỗi input bắt đầu với prefix đã cho
Hàm fn:substring() trong JSTL	Trả về một tập con của một chuỗi
Hàm fn:substringAfter() trong JSTL	Trả về một tập con của một chuỗi ở sau một chuỗi phụ đã cho
Hàm fn:substringBefore() trong JSTL	Trả về một tập con của một chuỗi ở trước một chuỗi phụ đã cho
Hàm fn:toLowerCase() trong JSTL	Biến đổi tất cả ký tự của một chuỗi thành chữ thường
Hàm fn:toUpperCase() trong JSTL	Biến đổi tất cả ký tự của một chuỗi thành chữ hoa

Hackademics Hanoi
Hack your passion, Code your life!

HẾT CHƯƠNG 14