




Bộ môn Công nghệ Phần mềm
Viện CNTT & TT
Trường Đại học Bách Khoa Hà Nội

LẬP TRÌNH WEB HƯỚNG JAVA

Bài 12: JSP 1.2. custom tags

Giảng viên: ThS. Trịnh Tuấn Đạt
Bộ môn CNPM
Email: trinhthuandat.bk@gmail.com/dattt@soict.hut.edu.vn

Nội dung

- 1. Giới thiệu
 - 1.1. Quá trình phát triển thiết kế của ứng dụng Web
 - 1.2. Custom tags là gì?
 - 1.3. Tại sao cần Custom tags?
 - 1.4. Các thành phần trong kiến trúc custom tag
 - 1.5. Cách thức xây dựng, cấu hình và triển khai thư viện thẻ

DatTT-DSE-SOICT-HUST 2

Nội dung (2)

- 2. Các ví dụ sử dụng custom tags (tăng dần về độ phức tạp)
 - 2.1. Định nghĩa tag cơ bản (không có thuộc tính, không có phần body)
 - 2.2. Tạo các thuộc tính cho thẻ
 - 2.3. Sử dụng phần thân thẻ (tag body)
 - 2.4. Tùy chọn sử dụng phần thân thẻ
 - 2.5. Xử lý phần thân thẻ
 - 2.6. Sử dụng hoặc xử lý phần thân thẻ nhiều lần
 - 2.7. Sử dụng thẻ lồng

DatTT-DSE-SOICT-HUST 3

Nội dung (3)

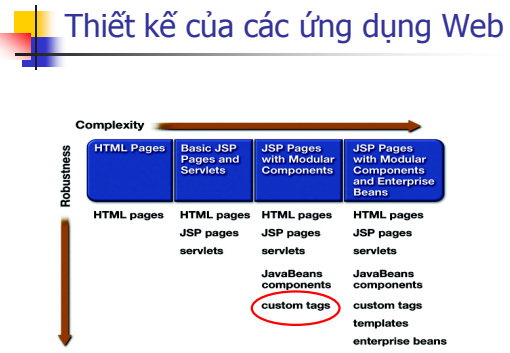
- 3. Kiến thức bổ sung
 - 3.1. Khai báo thư viện thẻ sử dụng taglib directive
 - 3.2. Tổ chức thư viện thẻ cho ứng dụng Web
 - 3.3. Hoạt động của custom tags?

DatTT-DSE-SOICT-HUST 4

1.1. Quá trình phát triển thiết kế của ứng dụng Web

DatTT-DSE-SOICT-HUST 5

Thiết kế của các ứng dụng Web

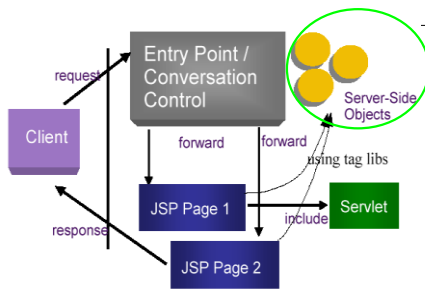


The diagram illustrates the progression of web application design. It features a 2D coordinate system with 'Complexity' on the horizontal axis (increasing to the right) and 'Robustness' on the vertical axis (increasing downwards). Four stages are shown as blue boxes:

- HTML Pages:** Simple HTML pages.
- Basic JSP Pages and Servlets:** HTML pages, JSP pages, and servlets.
- JSP Pages with Modular Components:** HTML pages, JSP pages, servlets, and JavaBeans components. The 'custom tags' sub-category is circled in red.
- JSP Pages with Modular Components and Enterprise Beans:** HTML pages, JSP pages, servlets, JavaBeans components, custom tags, templates, and enterprise beans.

DatTT-DSE-SOICT-HUST 6

Vị trí của custom tags?



DatTT-DSE-SOICT-HUST

7

Các thẻ chuẩn tắc

- `<jsp:useBean>`
 - Khởi tạo 1 đối tượng JavaBean
- `<jsp:getProperty>`
 - Truy cập các thuộc tính bean
- `<jsp:setProperty>`
 - Thiết lập các thuộc tính bean

DatTT-DSE-SOICT-HUST

8

Các thẻ chuẩn tắc

- `<jsp:forward>`
 - Forwards request đến trang HTML/JSP hoặc servlet khác
- `<jsp:include>`
 - Includes output của 1 file JSP khác
- `<jsp:plugin>`
 - Downloads Java plugin về trình duyệt để thực thi applet hoặc bean

DatTT-DSE-SOICT-HUST

9

1.2. Custom tags là gì?



DatTT-DSE-SOICT-HUST

10

Custom tags là gì?

- Là các phần tử JSP do User tự định nghĩa (ngược với các thẻ chuẩn tắc: **standard tags**)
- Đóng gói các tác vụ phải thực hiện nhiều lần
- Lấy từ thư viện thẻ (**tag library**) tự định nghĩa

DatTT-DSE-SOICT-HUST

11

Custom Tags có thể

- Được tùy biến thông qua các thuộc tính truyền từ trang JSP gọi chúng
- Có thể truyền lại tham số cho trang 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

DatTT-DSE-SOICT-HUST

12

Ví dụ các Custom Tag

- Thiết lập/truy cập các **Implicit objects**
- Xử lý forms
- Truy cập database
- Điều khiển (rẽ nhánh, vòng lặp)
- ...

DatTT-DSE-SOICT-HUST

13

Các thư viện thẻ (Custom Tag Library) đã có sẵn:

- Java Standard Tag Library (JSTL)
 - Tags for setting/getting attributes, iteration, etc
 - Tags truy cập database
 - Tags for internationalized formatting
 - Tags for XML
- Jakarta-Taglibs

DatTT-DSE-SOICT-HUST

14

1.3. Tại sao cần custom tags?

Tại sao cần custom tags?

- Tách biệt phần presentation với phần business logic (và các chức năng khác)
 - Thiết kế trang: thực hiện phần presentation
 - Business logic developers: tạo các custom tags
- Đóng gói phần business 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

DatTT-DSE-SOICT-HUST

15

DatTT-DSE-SOICT-HUST

16

Custom Tags vs. JavaBeans

- Ưu điểm
 - Custom tags có thể xử lý các nội dung trang JSP, trong khi beans không làm được
 - Các thao tác phức tạp sẽ chuyển thành dạng đơn giản hơn, nếu dùng custom tags thay cho beans
- Nhược điểm
 - Mặc dù đơn giản khi sử dụng, nhưng tạo Custom phức tạp hơn tạo các beans

DatTT-DSE-SOICT-HUST source: more Servlets and JSP[2] 17

1.4. Các thành phần trong kiến trúc custom tag (Trong JSP 1.2)

DatTT-DSE-SOICT-HUST

18

3 thành phần trong kiến trúc custom tag

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

DatTT-DSE-SOICT-HUST

19

Các bước cài đặt, sử dụng & triển khai các custom tags

- Cài đặt các custom tags
 - Viết các lớp xử lý tag
 - Viết các file **Tag library descriptor** (TLD) file
 - Đóng gói các lớp xử lý tag, các file TLD thành thư viện thẻ - tag library (hoặc ở dạng đóng gói hoặc không)
- Sử dụng các custom tags
 - Viết các trang JSP sử dụng tags
- Triển khai các custom tags như một phần của ứng dụng Web
 - Cấu hình và triển khai các tag lib cùng với các trang JSP

DatTT-DSE-SOICT-HUST

20

Lớp xử lý thẻ (Tag Handler)

- Thực thi giao diện
 - `javax.servlet.jsp.tagext.Tag` hoặc
 - `javax.servlet.jsp.tagext.Bodytag`
- Thường kế thừa lớp tiện ích:
 - `javax.servlet.jsp.tagext.TagSupport` hoặc
 - `javax.servlet.jsp.tagext.BodyTagSupport`
- Nằm cùng thư mục với các lớp Servlet
 - `/WEB-INF/classes/` <package-directory-structure>

DatTT-DSE-SOICT-HUST

21

Ví dụ: Lớp xử lý thẻ ExampleTag.java

```
package moreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

/** Very simple JSP tag that just inserts a string
 * ("Custom tag example...") into the output.
 * The actual name of the tag is not defined here;
 * that is given by the Tag Library Descriptor (TLD)
 * file that is referenced by the taglib directive
 * in the JSP file.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * © 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

22

Ví dụ: Lớp xử lý thẻ ExampleTag.java (2)

```
public class ExampleTag extends TagSupport {
    public int doStartTag() {
        try {
            JspWriter out = pageContext.getOut();
            out.print("Custom tag example " +
                "(moreservlets.tags.ExampleTag)");
        } catch (IOException ioe) {
            System.out.println("Error in ExampleTag: " + ioe);
        }
        return(SKIP_BODY);
    }
}
```

DatTT-DSE-SOICT-HUST

23

Tag Library Descriptor (TLD)

- Là file XML đặc tả
 - Tên thẻ: tag name
 - Nội dung: bodycontent
 - Các thuộc tính: attributes
 - Lớp xử lý thẻ tương ứng
- Container biết được 1 tag sẽ có lớp xử lý tương ứng là gì dựa trên file XML này
- Thường nằm ở trong thư mục WEB-INF
- Custom location can specified in JSP file
 - Qua thuộc tính uri hoặc **taglib directive**

DatTT-DSE-SOICT-HUST

24

Ví dụ: msajsp-tags.tld

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc./DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/tdts/web-jsp/taglibrary_1_1.dtd">

<!-- a tag library descriptor -->

<taglib>
  <tliversion>1.0</tliversion>
  <jspversion>1.1</jspversion>
  <shortname>msajsp-tags</shortname>
  <info>
    A tag library from More Servlets and JavaServer Pages,
    http://www.moreservlets.com/.
  </info>
```

DatTT-DSE-SOICT-HUST

25

Ví dụ: msajsp-tags.tld (2)

```
<tag>
  <name>example</name>
  <tagclass>moreservlets.tags.ExampleTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>Simplest example: inserts one line of output</info>
</tag>

<tag>
  <name>simplePrime</name>
  <tagclass>moreservlets.tags.SimplePrimeTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>Outputs a random 50-digit prime.</info>
</tag>

...
</taglib>
```

DatTT-DSE-SOICT-HUST

26

Thư viện thẻ là gì?

- Là tập các thẻ cùng chung mục đích
 - Một hoặc nhiều thẻ có thể được đóng gói thành thư viện thẻ
- Thường được đóng gói thành file Jar, chứa:
 - Một file tag library descriptor (TLD)
 - Ví dụ: META-INF/taglib.tld
 - Các file *.class – các lớp xử lý thẻ
 - Các resource đi kèm nếu cần

DatTT-DSE-SOICT-HUST

27

Các trang JSP

- Khai báo thư viện thẻ qua **taglib directive**
- Sử dụng thẻ theo đúng cú pháp quy định

DatTT-DSE-SOICT-HUST

28

Khai báo một thư viện thẻ

- Phải có **taglib directive** trước khi sử dụng các thẻ
- Cú pháp
 - <%@ taglib prefix="myprefix" uri="myuri" %>
 - prefix: tên của thư viện thẻ (tùy chọn tên)
 - uri: xác định duy nhất **tag library descriptor** (TLD), trực tiếp hoặc gián tiếp

DatTT-DSE-SOICT-HUST

29

Cú pháp Custom Tag trong trang JSP

```
<prefix:tag attr1="value" ... attrN="value" />
■ Hoặc
<prefix:tag attr1="value" ... attrN="value" >
  body
</prefix:tag>
```

- **prefix**: tên của thư viện thẻ
- **tag**: định danh của thẻ (trong thư viện thẻ tương ứng)
- **attr1 ... attrN**: các thuộc tính của thẻ

DatTT-DSE-SOICT-HUST

30

Ví dụ: SimpleExample.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
```

```
<!--
```

Illustration of very simple JSP custom tag.

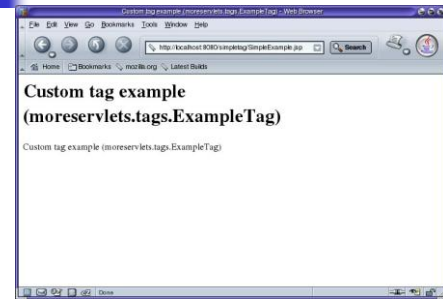
Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
<http://www.moreservlets.com/>.
(C) 2002 Marty Hall; may be freely used or adapted.

```
-->
<HTML>
<HEAD>
<%@ taglib uri="/WEB-INF/msajsp-taglib.tld" prefix="msajsp" %>
<TITLE><msajsp:example /></TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1><msajsp:example /></H1>
<msajsp:example />
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

31

Ví dụ: truy cập trang SimpleExample.jsp



DatTT-DSE-SOICT-HUST

32

1.5. Cách thức xây dựng, cấu hình và triển khai thư viện thẻ

DatTT-DSE-SOICT-HUST

33

Cấu trúc thư mục mã nguồn

- Nên sử dụng cấu trúc mã nguồn (như trong thư mục web/iterator của J2EE 1.4 Tutorial)
 - <j2ee14tutorial-install>/j2eetutorial14/examples/web
 - iterator
 - build.xml (ant build file)
 - src (Thư mục chứa các file mã nguồn java, với cấu trúc package phù hợp)
 - web (dir)
 - WEB-INF (dir)
 - *.tld
 - tags (thư mục file thẻ JSP 2.0)
 - *.jsp

DatTT-DSE-SOICT-HUST

34

Cấu trúc thư mục Build

- Cấu trúc thư mục build nên được tổ chức như sau: (như trong thư mục web/iterator của J2EE 1.4 Tutorial)
 - <j2eetutorial14-install>/j2eetutorial14/examples/web
 - iterator
 - build (unpacked *.war file)
 - dist (packed *.war file)

DatTT-DSE-SOICT-HUST

35

2. Các ví dụ sử dụng Custom tags (trong JSP 1.2)

DatTT-DSE-SOICT-HUST

36

2. Các ví dụ sử dụng Custom tags

- 2.1. Định nghĩa tag cơ bản (không có thuộc tính, không có phần body)
- 2.2. Tạo các thuộc tính cho thẻ
- 2.3. Sử dụng phần thân thẻ (tag body)
- 2.4. Tùy chọn sử dụng phần thân thẻ
- 2.5. Xử lý phần thân thẻ
- 2.6. Sử dụng hoặc xử lý phần thân thẻ nhiều lần
- 2.7. Sử dụng thẻ lồng

DatTT-DSE-SOICT-HUST

37

2.1. Định nghĩa một thẻ đơn giản



DatTT-DSE-SOICT-HUST

38

2.1. Định nghĩa một thẻ đơn giản

- Một thẻ không có thuộc tính, không có body
- Kế thừa lớp TagSupport
- Chỉ cần override phương thức doStartTag()
 - doStartTag(): sẽ được gọi trong thời điểm đang request khi gặp thẻ mở
 - doStartTag(): trả về SKIP_BODY vì thẻ không có body
 - Chỉ dẫn container bỏ qua phần nội dung body giữa thẻ mở và thẻ đóng

DatTT-DSE-SOICT-HUST

39

2.2. Tạo các thuộc tính cho thẻ



DatTT-DSE-SOICT-HUST

40

Tại sao cần tạo các thuộc tính cho thẻ?

- Cung cấp cách thức truyền các cặp giá trị attribute/value từ trang JSP đến bộ xử lý thẻ

DatTT-DSE-SOICT-HUST

41

Lớp xử lý thẻ

- Sử dụng thuộc tính X trong trang JSP sẽ dẫn đến lời gọi phương thức setX() trong lớp xử lý thẻ
 - Lớp xử lý thẻ phải cài đặt phương thức setX()

```
public void setX(String value1) {
    doSomethingWith(value1);
}
```

DatTT-DSE-SOICT-HUST

42

Ví dụ: Lớp xử lý thẻ SimplePrimeTag.java

```
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import java.math.*;
import javax.servlet.*;

/** Generates a prime of approximately 50 digits.
 * (50 is actually the length of the random number
 * generated -- the first prime above that number will
 * be returned.)
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * &copy; 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

43

Ví dụ: Lớp xử lý thẻ SimplePrimeTag.java (2)

```
public class SimplePrimeTag extends TagSupport {
    protected int len = 50;

    public int doStartTag() {
        try {
            JspWriter out = pageContext.getOut();
            BigInteger prime = Primes.nextPrime(Primes.random(len));
            out.print(prime);
        } catch (IOException ioe) {
            System.out.println("Error generating prime: " + ioe);
        }
        return(SKIP_BODY);
    }
}
```

DatTT-DSE-SOICT-HUST

44

Ví dụ: Lớp xử lý thẻ PrimeTag.java

```
package moreservlets.tags;

/** Generates an N-digit random prime (default N = 50).
 * Extends SimplePrimeTag, adding a length attribute
 * to set the size of the prime. The doStartTag
 * method of the parent class uses the len field
 * to determine the length of the prime.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * &copy; 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

45

Ví dụ: Lớp xử lý thẻ PrimeTag.java (2)

```
public class PrimeTag extends SimplePrimeTag {
    public void setLength(String length) {
        try {
            len = Integer.parseInt(length);
        } catch (NumberFormatException nfe) {
            len = 50;
        }
    }

    /** Servers are permitted to reuse tag instances
     * once a request is finished. So, this resets
     * the len field.
     */
    public void release() {
        len = 50;
    }
}
```

DatTT-DSE-SOICT-HUST

46

TLD

- Các thuộc tính phải được khai báo trong phần tử con **attribute** của phần tử **tag**
- Phần tử con **attribute** lại có 5 phần tử con trong nó:
 - name (bắt buộc)
 - required (bắt buộc)
 - rtexprvalue (tùy chọn)
 - type (tùy chọn)
 - example (tùy chọn)

DatTT-DSE-SOICT-HUST

47

Phần tử con rtexprvalue và type

- rtexprvalue (tùy chọn)
 - true nếu giá trị một thuộc tính có thể là <%= expression %>
 - false nếu giá trị một thuộc tính là 1 string fix sẵn (mặc định)
- type (tùy chọn)
 - Chỉ định class mà giá trị của rtexprvalue được ép kiểu về
 - Chỉ hợp lệ khi rtexprvalue được thiết lập true

DatTT-DSE-SOICT-HUST

48

Ví dụ: TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">

<!-- a tag library descriptor -->

<taglib>
  <tliversion>1.0</tliversion>
  <jspversion>1.1</jspversion>
  <shortname>msajsp-tags</shortname>
  <info>
    A tag library from More Servlets and JavaServer Pages,
    http://www.moreservlets.com/.
  </info>
```

DatTT-DSE-SOICT-HUST

49

Ví dụ: TLD (2)

```
...
<tag>
  <name>prime</name>
  <tagclass>moreservlets.tags.PrimeTag</tagclass>
  <bodycontent>empty</bodycontent>
  <info>Outputs a random N-digit prime.</info>
  <attribute>
    <name>length</name>
    <required>false</required>
  </attribute>
</tag>

...
</taglib>
```

DatTT-DSE-SOICT-HUST

50

Ví dụ: trang JSP (PrimeExample.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!-- Illustration of PrimeTag tag.

Taken from More Servlets and JavaServer Pages from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/(C) 2002 Marty Hall; may be freely used or adapted.

-->
<HTML>
<HEAD>
<TITLE>Some N-Digit Primes</TITLE>
<LINK REL=STYLESHEET
  HREF="JSP-Styles.css"
  TYPE="text/css">
</HEAD>
<BODY>
<H1>Some N-Digit Primes</H1>
<%@ taglib uri="/WEB-INF/msajsp-taglib.tld" prefix="msajsp" %>
<UL>
  <LI>20-digit: <msajsp:prime length="20" />
  <LI>40-digit: <msajsp:prime length="40" />
  <LI>60-digit: <msajsp:prime length="60" />
  <LI>Default (50-digit): <msajsp:prime />
</UL>
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

51

Ví dụ: truy cập trang PrimeExample.jsp



DatTT-DSE-SOICT-HUST

52

2.3. Sử dụng phần thân thẻ (include tag body)

Sử dụng phần thân thẻ

- <prefix:tagname>body</prefix:tagname>
- Body content: chứa đoạn mã JSP giữa thẻ mở và thẻ đóng
 - Được dịch khi trang JSP tương ứng dịch sang mã Servlet
- Trong phần này, chỉ include phần body content, không cần chỉnh sửa hay xử lý gì.

DatTT-DSE-SOICT-HUST

53

DatTT-DSE-SOICT-HUST

54

Lớp xử lý thẻ

- Phương thức `doStartTag()` cần trả về `EVAL_BODY_INCLUDE`
- `doEndTag()` được gọi sau khi đọc xong phần `body`
 - Trả về `EVAL_PAGE` để xử lý tiếp
 - Trả về `SKIP_PAGE` để bỏ qua xử lý phần còn lại của trang

DatTT-DSE-SOICT-HUST

55

Ví dụ: Lớp xử lý thẻ `HeadingTag.java`

```
package moreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

/** Generates an HTML heading with the specified background
 * color, foreground color, alignment, font, and font size.
 * You can also turn on a border around it, which normally
 * just barely encloses the heading, but which can also
 * stretch wider. All attributes except the background
 * color are optional.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * &copy; 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

56

Ví dụ: Lớp xử lý thẻ `HeadingTag.java` (2)

```
public class HeadingTag extends TagSupport {
    private String bgColor; // The one required attribute
    private String color = null;
    private String align="CENTER";
    private String fontSize="36";
    private String fontList="Arial, Helvetica, sans-serif";
    private String border="0";
    private String width=null;

    public void setBgColor(String bgColor) {
        this.bgColor = bgColor;
    }

    public void setColor(String color) {
        this.color = color;
    }
}
```

...

DatTT-DSE-SOICT-HUST

57

Ví dụ: Lớp xử lý thẻ `HeadingTag.java` (3)

```
public int doStartTag() {
    try {
        JspWriter out = pageContext.getOut();
        out.print("<TABLE BORDER=" + border +
            " BGCOLOR=\"" + bgColor + "\"" + " ALIGN=\"" + align + "\"");
        if (width != null) {
            out.print(" WIDTH=\"" + width + "\"");
        }
        out.print("><TR><TH>");
        out.print("<SPAN STYLE=\"" +
            "font-size: " + fontSize + "px; " + "font-family: " + fontList + ";");
        if (color != null) {
            out.println("color: " + color + ";");
        }
        out.print("\>"); // End of <SPAN ...>
    } catch (IOException ioe) {
        System.out.println("Error in HeadingTag: " + ioe);
    }
    return(EVAL_BODY_INCLUDE); // Include tag body
}
```

DatTT-DSE-SOICT-HUST

58

Ví dụ: Lớp xử lý thẻ `HeadingTag.java` (4)

```
public int doEndTag() {
    try {
        JspWriter out = pageContext.getOut();
        out.print("</SPAN></TABLE>");
    } catch (IOException ioe) {
        System.out.println("Error in HeadingTag: " + ioe);
    }
    return(EVAL_PAGE); // Continue with rest of JSP page
}
```

DatTT-DSE-SOICT-HUST

59

TLD

- Phần tử `bodycontent` cần chứa giá trị là `JSP`
 - `<bodycontent>JSP</bodycontent>`

DatTT-DSE-SOICT-HUST

60

Ví dụ: TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ...>

<taglib>
  ...
  <tag>
    <name>heading</name>
    <tagclass>moreservlets.tags.HeadingTag</tagclass>
    <bodycontent>JSP</bodycontent>
    <info>Outputs a 1-cell table used as a heading.</info>
    <attribute>
      <name>bgColor</name>
      <required>true</required> <!-- bgColor is required -->
    </attribute>
    <attribute>
      <name>color</name>
      <required>false</required>
    </attribute>
  </tag>
</taglib>
```

DatTT-DSE-SOICT-HUST

61

Ví dụ: trang JSP (HeadingExample.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Illustration of HeadingTag tag.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
<TITLE>Some Tag-Generated Headings</TITLE>
</HEAD>
<BODY>
  <%@ taglib uri="/WEB-INF/msajsp-taglib.tld" prefix="msajsp" %>
  <msajsp:heading bgColor="#C0C0C0">
    Default Heading
  </msajsp:heading>
  <msajsp:heading bgColor="BLACK" color="WHITE">
    White on Black Heading
  </msajsp:heading>
```

DatTT-DSE-SOICT-HUST

62

Ví dụ: trang JSP (HeadingExample.jsp) (2)

```
<P>
<msajsp:heading bgColor="#EF8429" fontSize="60" border="5">
  Large Bordered Heading
</msajsp:heading>
<P>
<msajsp:heading bgColor="CYAN" width="100%">
  Heading with Full-Width Background
</msajsp:heading>
<P>
<msajsp:heading bgColor="CYAN" fontSize="60"
  fontList="Brush Script MT, Times, serif">
  Heading with Non-Standard Font
</msajsp:heading>
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

63

Truy cập trang HeadingExample.jsp



DatTT-DSE-SOICT-HUST

64

2.4. Tùy chọn sử dụng phần thân thẻ

Tùy chọn sử dụng phần thân thẻ

- Việc sử dụng phần thân thẻ được quyết định tại thời điểm request
- Trong ví dụ này, vẫn không chỉnh sửa body content

DatTT-DSE-SOICT-HUST

65

DatTT-DSE-SOICT-HUST

66

Lớp xử lý thẻ

- Phương thức `doStartTag()` trả về `EVAL_BODY_INCLUDE` hoặc `SKIP_BODY`
 - Phụ thuộc vào giá trị của 1 biểu thức nào đó, hoặc tùy theo xử lý nghiệp vụ
- Gọi phương thức `getRequest()` từ thuộc tính `pageContext` của lớp `TagSupport`
 - Ép kiểu trả về của phương thức `getRequest()` thành `HttpServletRequest` (phương thức `getRequest()` trả về `ServletRequest`)

DatTT-DSE-SOICT-HUST

67

Ví dụ: Lớp xử lý thẻ DebugTag.java

```
package moreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;

/** A tag that includes the body content only if
 * the "debug" request parameter is set.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * &copy; 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

68

Ví dụ: Lớp xử lý thẻ DebugTag.java (2)

```
public class DebugTag extends TagSupport {
    public int doStartTag() {
        ServletRequest request = pageContext.getRequest();
        String debugFlag = request.getParameter("debug");
        if ((debugFlag != null) &&
            (!debugFlag.equalsIgnoreCase("false"))) {
            return(EVAL_BODY_INCLUDE);
        } else {
            return(SKIP_BODY);
        }
    }
}
```

DatTT-DSE-SOICT-HUST

69

Ví dụ: TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ...>

<taglib>
...
<tag>
  <name>debug</name>
  <tagclass>moreservlets.tags.DebugTag</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>Includes body only if debug param is set.</info>
</tag>

...
```

DatTT-DSE-SOICT-HUST

70

Ví dụ: Trang JSP (DebugExample.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Illustration of DebugTag tag.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/.
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
<TITLE>Using the Debug Tag</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Using the Debug Tag</H1>
<%@ taglib uri="/WEB-INF/msajsp-taglib.tld" prefix="msajsp" %>
Top of regular page. Blah, blah, blah. Yadda, yadda, yadda.
<P>
```

DatTT-DSE-SOICT-HUST

71

Ví dụ: Trang JSP (DebugExample.jsp) (2)

```
<msajsp:debug>
<B>Debug:</B>
<UL>
<LI>Current time: <%= new java.util.Date() %>
<LI>Requesting hostname: <%= request.getRemoteHost() %>
<LI>Session ID: <%= session.getId() %>
</UL>
</msajsp:debug>
<P>
Bottom of regular page. Blah, blah, blah. Yadda, yadda, yadda.
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

72

DebugTag trong trường hợp không có tham số "debug"



73

DebugTag trong trường hợp có tham số "debug=true"



74

2.5. Xử lý phần thân thẻ

DatTT-DSE-SOICT-HUST

75

Lớp xử lý thẻ

- Lớp xử lý thẻ cần kế thừa lớp **BodyTagSupport**
- Lớp **BodyTagSupport** có 2 phương thức
 - **doAfterBody()**: ghi đè phương thức này để xử lý phần thân thẻ
 - Trả về **SKIP_BODY** nếu không cần xử lý thêm phần body nữa
 - Trả về **EVAL_BODY_TAG** (JSP 1.1) hoặc **EVAL_BODY_AGAIN** (JSP 1.2) nếu nội dung phần body cần được xử lý lại thêm nữa
 - **getBodyContent()**: trả về đối tượng **BodyContent**

DatTT-DSE-SOICT-HUST

76

Lớp BodyContent

- Đóng gói các thao tác đọc phần body
- Là lớp con của **JspWriter**
- Lớp **BodyContent** có 3 phương thức:
 - **getEnclosingWriter()**: trả về **JspWriter**
 - **getReader()**: Trả về một **Reader** có thể đọc phần thân thẻ (tag body)
 - **getString()**: trả về 1 **String** chứa toàn bộ tag body

DatTT-DSE-SOICT-HUST

77

Ví dụ: Lớp xử lý thẻ FilterTag.java

```
package moreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import moreservlets.*;

/** A tag that replaces <, >, " and & with their HTML
 * character entities (&lt;, &gt;, &quot;, and &amp;).
 * After filtering, arbitrary strings can be placed
 * in either the page body or in HTML attributes.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * © 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

78

Ví dụ: Lớp xử lý thẻ FilterTag.java (2)

```
public class FilterTag extends BodyTagSupport {
    public int doAfterBody() {
        BodyContent body = getBodyContent();
        String filteredBody =
            ServletUtilities.filter(body.getString());
        try {
            JspWriter out = body.getEnclosingWriter();
            out.print(filteredBody);
        } catch (IOException ioe) {
            System.out.println("Error in FilterTag: " + ioe);
        }
        // SKIP_BODY means we're done. If we wanted to evaluate
        // and handle the body again, we'd return EVAL_BODY_TAG
        // (JSP 1.1/1.2) or EVAL_BODY_AGAIN (JSP 1.2 only)
        return(SKIP_BODY);
    }
}
```

DatTT-DSE-SOICT-HUST

79

Ví dụ: TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ...>

<taglib>
...
<tag>
  <name>filter</name>
  <tagclass>moreservlets.tags.FilterTag</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>Replaces HTML-specific characters in body.</info>
</tag>

...
```

DatTT-DSE-SOICT-HUST

80

Ví dụ: Trang JSP (FilterExample.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Illustration of FilterTag tag.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/,
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
<TITLE>HTML Logical Character Styles</TITLE>
<LINK REL="STYLESHEET"
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>HTML Logical Character Styles</H1>
Physical character styles (B, I, etc.) are rendered consistently
in different browsers. Logical character styles, however,
may be rendered differently by different browsers.
Here's how your browser
(<%= request.getHeader("User-Agent") %>)
renders the HTML 4.0 logical character styles
<P>
```

DatTT-DSE-SOICT-HUST

81

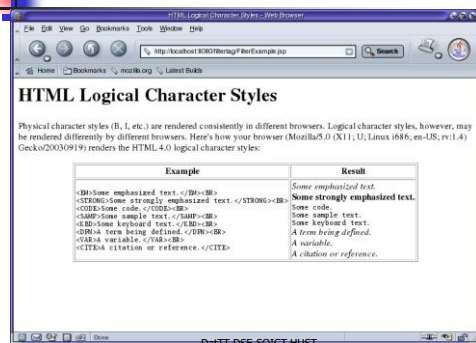
Ví dụ: Trang JSP (FilterExample.jsp) (2)

```
<%@ taglib uri="/WEB-INF/masajsp-taglib.tld" prefix="msajsp" %>
<TABLE BORDER=1 ALIGN="CENTER">
<TR CLASS="COLORED"><TH>Example<TH>Result
<TR>
<TD><PRE><msajsp:filter>
<EM>Some emphasized text.</EM><BR>
<STRONG>Some strongly emphasized text.</STRONG><BR>
<CODE>Some code.</CODE><BR>
<SAMP>Some sample text.</SAMP><BR>
<KBD>Some keyboard text.</KBD><BR>
<DFN>A term being defined.</DFN><BR>
<VAR>A variable.</VAR><BR>
<CITE>A citation or reference.</CITE>
</msajsp:filter></PRE>
<TD>
<EM>Some emphasized text.</EM><BR>
<STRONG>Some strongly emphasized text.</STRONG><BR>
<CODE>Some code.</CODE><BR>
<SAMP>Some sample text.</SAMP><BR>
<KBD>Some keyboard text.</KBD><BR>
<DFN>A term being defined.</DFN><BR>
<VAR>A variable.</VAR><BR>
<CITE>A citation or reference.</CITE>
</TABLE>
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

82

Truy cập trang FilterExample.jsp



DatTT-DSE-SOICT-HUST

83

2.6. Sử dụng hoặc xử lý phần thân thẻ nhiều lần

DatTT-DSE-SOICT-HUST

84

Ví dụ: lớp xử lý thẻ RepeatTag.java

- Lớp xử lý thẻ cần kế thừa lớp `BodyTagSupport`
- `doAfterBody()` trả về `EVAL_BODY_TAG` (`EVAL_BODY_AGAIN` trong JSP 1.2)

DatTT-DSE-SOICT-HUST

85

Ví dụ: lớp xử lý thẻ RepeatTag.java

```
package moreservlets.tags;

import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;

/** A tag that repeats the body the specified
 * number of times.
 * <P>
 * Taken from More Servlets and JavaServer Pages
 * from Prentice Hall and Sun Microsystems Press,
 * http://www.moreservlets.com/.
 * &copy; 2002 Marty Hall; may be freely used or adapted.
 */
```

DatTT-DSE-SOICT-HUST

86

Ví dụ: lớp xử lý thẻ RepeatTag.java (2)

```
public class RepeatTag extends BodyTagSupport {
    private int reps;

    public void setReps(String repeats) {
        try {
            reps = Integer.parseInt(repeats);
        } catch (NumberFormatException nfe) {
            reps = 1;
        }
    }
}
```

DatTT-DSE-SOICT-HUST

87

Ví dụ: lớp xử lý thẻ RepeatTag.java (3)

```
public int doAfterBody() {
    if (reps-- >= 1) {
        BodyContent body = getBodyContent();
        try {
            JspWriter out = body.getEnclosingWriter();
            out.println(body.getString());
            body.clearBody(); // Clear for next evaluation
        } catch (IOException ioe) {
            System.out.println("Error in RepeatTag: " + ioe);
        }
        // Replace EVAL_BODY_TAG with EVAL_BODY_AGAIN in JSP 1.2.
        return(EVAL_BODY_TAG);
    } else {
        return(SKIP_BODY);
    }
}
```

DatTT-DSE-SOICT-HUST

88

Ví dụ: TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE ...>

<taglib>
...
<tag>
  <name>repeat</name>
  <tagclass>moreservlets.tags.RepeatTag</tagclass>
  <bodycontent>JSP</bodycontent>
  <info>Repeats body the specified number of times.</info>
  <attribute>
    <name>reps</name>
    <required>true</required>
    <!-- rtexprvalue indicates whether attribute
         can be a JSP expression. -->
    <rtexprvalue>true</rtexprvalue>
  </attribute>
</tag>
...
```

DatTT-DSE-SOICT-HUST

89

Ví dụ: trang JSP (RepeatExample.jsp)

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<!--
Illustration of RepeatTag tag.

Taken from More Servlets and JavaServer Pages
from Prentice Hall and Sun Microsystems Press,
http://www.moreservlets.com/,
(C) 2002 Marty Hall; may be freely used or adapted.
-->
<HTML>
<HEAD>
<TITLE>Some 40-Digit Primes</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>Some 40-Digit Primes</H1>
Each entry in the following list is the first prime number
higher than a randomly selected 40-digit number.
```

DatTT-DSE-SOICT-HUST

90

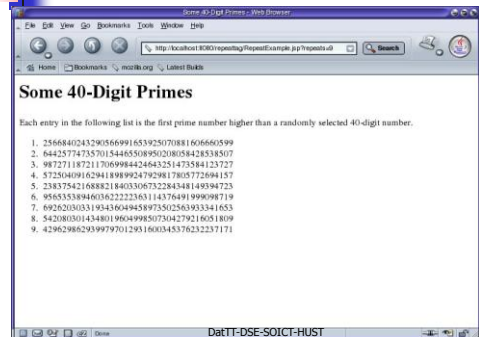
Ví dụ: trang JSP (RepeatExample.jsp) (2)

```
<%@ taglib uri="/WEB-INF/msajsp-taglib.tld" prefix="msajsp" %>
<OL>
<!-- Repeats N times. A null reps value means repeat once. -->
<msajsp:repeat reps='<%= request.getParameter("repeats") %>'>
<LI><msajsp:prime length="40" />
</msajsp:repeat>
</OL>
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

91

Truy cập trang RepeatExample.jsp



DatTT-DSE-SOICT-HUST

92

2.7. Tạo các thẻ lồng

DatTT-DSE-SOICT-HUST

93

Tạo các thẻ lồng

Bài toán:

```
<csajsp:if>
  <csajsp:condition><%= someExpression %> </csajsp:condition>
  <csajsp:then>JSP to include if condition is
  true</csajsp:then>
  <csajsp:else>JSP to include if condition is
  false</csajsp:else>
</csajsp:if>
```

Trong thẻ if có 3 thẻ con

- Thẻ kiểm tra điều kiện
- Thẻ chứa nội dung được thực hiện nếu điều kiện đúng
- Thẻ chứa nội dung được thực hiện nếu điều kiện sai

DatTT-DSE-SOICT-HUST

94

Lớp xử lý thẻ

- Có thể kế thừa TagSupport hoặc BodyTagSupport, tùy thuộc chỉ sử dụng hay phải xử lý phần body
- Sử dụng phương thức tiện ích: findAncestorWithClass:
 - Tham số thứ nhất: this
 - Tham số thứ hai: đối tượng Class của lớp thẻ bao bên ngoài nó

DatTT-DSE-SOICT-HUST

95

Ví dụ: Lớp xử lý thẻ IfTag.java

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;
/** A tag that acts like an if/then/else.
 */
public class IfTag extends TagSupport {
  private boolean condition;
  private boolean hasCondition = false;
  public void setCondition(boolean condition) {
    this.condition = condition;
    hasCondition = true;
  }
  public boolean getCondition() {
    return(condition);
  }
}
```

DatTT-DSE-SOICT-HUST

96

Ví dụ: Lớp xử lý thẻ IfTag.java (2)

```
public void setHasCondition(boolean flag) {
    this.hasCondition = flag;
}
/** Has the condition field been explicitly set? */
public boolean hasCondition() {
    return(hasCondition);
}
public int doStartTag() {
    return(EVAL_BODY_INCLUDE);
}
}
```

DatTT-DSE-SOICT-HUST

97

Ví dụ: Lớp xử lý thẻ IfConditionTag.java

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;
/** The condition part of an if tag.
 */
public class IfConditionTag extends BodyTagSupport {
    public int doStartTag() throws JspTagException {
        IfTag parent =
            (IfTag)findAncestorWithClass(this, IfTag.class);
        if (parent == null) {
            throw new JspTagException("condition not inside if");
        }
        return(EVAL_BODY_TAG);
    }
}
```

DatTT-DSE-SOICT-HUST

98

Ví dụ: Lớp xử lý thẻ IfConditionTag.java (2)

```
public int doAfterBody() {
    IfTag parent =
        (IfTag)findAncestorWithClass(this, IfTag.class);
    String bodyString = getBodyContent().getString();
    if (bodyString.trim().equals("true")) {
        parent.setCondition(true);
    } else {
        parent.setCondition(false);
    }
    return(SKIP_BODY);
}
}
```

DatTT-DSE-SOICT-HUST

99

Ví dụ: Lớp xử lý thẻ IfThenTag.java

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;
/** The then part of an if tag.
 */
public class IfThenTag extends BodyTagSupport {
    public int doStartTag() throws JspTagException {
        IfTag parent =
            (IfTag)findAncestorWithClass(this, IfTag.class);
        if (parent == null) {
            throw new JspTagException("then not inside if");
        } else if (!parent.hasCondition()) {
            String warning =
                "condition tag must come before then tag";
            throw new JspTagException(warning);
        }
        return(EVAL_BODY_TAG);
    }
}
```

DatTT-DSE-SOICT-HUST

100

Ví dụ: Lớp xử lý thẻ IfThenTag.java (2)

```
public int doAfterBody() {
    IfTag parent =
        (IfTag)findAncestorWithClass(this, IfTag.class);
    if (parent.getCondition()) {
        try {
            BodyContent body = getBodyContent();
            JspWriter out = body.getEnclosingWriter();
            out.print(body.getString());
        } catch (IOException ioe) {
            System.out.println("Error in IfThenTag: " + ioe);
        }
    }
    return(SKIP_BODY);
}
}
```

DatTT-DSE-SOICT-HUST

101

Ví dụ: Lớp xử lý thẻ IfElseTag.java

```
package coreservlets.tags;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;
import java.io.*;
import javax.servlet.*;
/** The else part of an if tag.
 */
public class IfElseTag extends BodyTagSupport {
    public int doStartTag() throws JspTagException {
        IfTag parent =
            (IfTag)findAncestorWithClass(this, IfTag.class);
        if (parent == null) {
            throw new JspTagException("else not inside if");
        } else if (!parent.hasCondition()) {
            String warning =
                "condition tag must come before else tag";
            throw new JspTagException(warning);
        }
        return(EVAL_BODY_TAG);
    }
}
```

DatTT-DSE-SOICT-HUST

102

Ví dụ: Lớp xử lý thẻ IfElseTag.java (2)

```
public int doAfterBody() {
    IfTag parent =
        (IfTag)findAncestorWithClass(this, IfTag.class);
    if (!parent.getCondition()) {
        try {
            BodyContent body = getBodyContent();
            JspWriter out = body.getEnclosingWriter();
            out.print(body.getString());
        } catch (IOException ioe) {
            System.out.println("Error in IfElseTag: " + ioe);
        }
    }
    return (SKIP_BODY);
}
```

DatTT-DSE-SOICT-HUST

103

Ví dụ: TLD (1)

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library
1.1/EN"
"http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<!-- a tag library descriptor -->
<taglib>
  <!-- after this the default space is
        "http://java.sun.com/j2ee/dtds/jsptaglibrary_1_2.dtd"
  -->
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>csajsp</shortname>
  <urn></urn>
  <info>
    A tag library from Core Servlets and JavaServer Pages,
    http://www.coreservlets.com/.
  </info>
  <!-- Other tags defined in IfElseTag.java -->
```

104

Ví dụ: TLD (2)

```
<tag>
  <name>if</name>
  <tagclass>coreservlets.tags.IfTag</tagclass>
  <info>if/condition/then/else tag.</info>
  <bodycontent>JSP</bodycontent>
</tag>

<tag>
  <name>condition</name>
  <tagclass>coreservlets.tags.IfConditionTag
  </tagclass>
  <info>condition part of if/condition/then/else tag.
  </info>
  <bodycontent>JSP</bodycontent>
</tag>
```

DatTT-DSE-SOICT-HUST

105

Ví dụ: TLD (3)

```
<tag>
  <name>then</name>
  <tagclass>coreservlets.tags.IfThenTag</tagclass>
  <info>then part of if/condition/then/else tag.
  </info>
  <bodycontent>JSP</bodycontent>
</tag>

<tag>
  <name>else</name>
  <tagclass>coreservlets.tags.IfElseTag</tagclass>
  <info>else part of if/condition/then/else tag.
  </info>
  <bodycontent>JSP</bodycontent>
</tag>
</taglib>
```

DatTT-DSE-SOICT-HUST

106

Ví dụ: Trang JSP IfExample.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<HEAD>
<TITLE>If Tag Example</TITLE>
<LINK REL=STYLESHEET
      HREF="JSP-Styles.css"
      TYPE="text/css">
</HEAD>
<BODY>
<H1>If Tag Example</H1>
<%@ taglib uri="csajsp-taglib.tld" prefix="csajsp" %>
<csajsp:if>
  <csajsp:condition>true</csajsp:condition>
  <csajsp:then>Condition was true</csajsp:then>
  <csajsp:else>Condition was false</csajsp:else>
</csajsp:if>
```

DatTT-DSE-SOICT-HUST

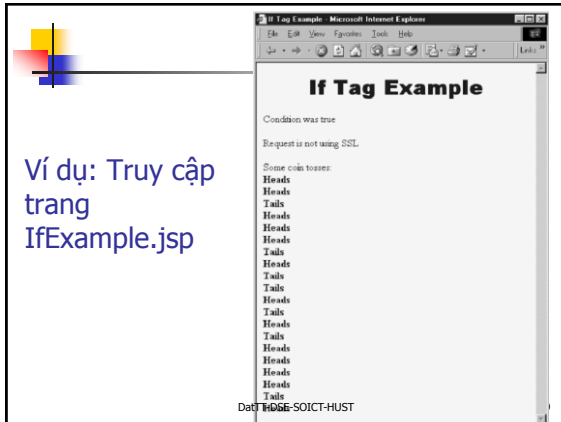
107

Ví dụ: Trang JSP IfExample.jsp

```
<P>
<csajsp:if>
  <csajsp:condition><%= request.isSecure()
  %></csajsp:condition>
  <csajsp:then>Request is using SSL (https)</csajsp:then>
  <csajsp:else>Request is not using SSL</csajsp:else>
</csajsp:if>
<P>
Some coin tosses:<BR>
<csajsp:repeat reps="20">
  <csajsp:if>
    <csajsp:condition>
      <%= Math.random() > 0.5 %>
    </csajsp:condition>
    <csajsp:then><B>Heads</B><BR></csajsp:then>
    <csajsp:else><B>Tails</B><BR></csajsp:else>
  </csajsp:if>
</csajsp:repeat>
</BODY>
</HTML>
```

DatTT-DSE-SOICT-HUST

108



3.1. Khai báo thư viện thẻ sử dụng taglib directive

DatTT-DSE-SOICT-HUST

110

Khai báo thư viện thẻ: thuộc tính uri

- Cách 1: Tham chiếu trực tiếp đến file TLD


```
<%@ taglib prefix="tlt" uri="/WEB-INF/iterator.tld"%>
```
 - Cách 2: Tham chiếu gián tiếp đến file TLD sử dụng định danh logic


```
<%@ taglib prefix="tlt" uri="/tlt"%>
```

 - Trong web.xml, cần có Mapping giữa định danh logic với đường dẫn của file TLD


```
<jsp-config>
  <taglib>
    <taglib-uri>/tlt</taglib-uri>
    <taglib-location>/WEB-INF/iterator.tld</taglib-location>
  </taglib>
</jsp-config>
```
- 11

DatTT-DSE-SOICT-HUST

111

Khai báo thư viện thẻ: thuộc tính uri

- Cách 3: URI tuyệt đối- ví dụ các thư viện thẻ JSTL
- ```
<%@ taglib prefix="core"
uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="xml"
uri="http://java.sun.com/jsp/jstl/xml"%>
<%@ taglib prefix="fmt"
uri="http://java.sun.com/jsp/jstl/fmt"%>
<%@ taglib prefix="sql"
uri="http://java.sun.com/jsp/jstl/sql"%>
<%@ taglib prefix="fn"
uri="http://java.sun.com/jsp/jstl/functions"%>
```

DatTT-DSE-SOICT-HUST

112

### 3.2. Tổ chức thư viện thẻ cho ứng dụng Web

DatTT-DSE-SOICT-HUST

113

## Tổ chức viên thẻ cho từng Web app

- **Dạng unpacked:**
  - Các lớp xử lý thẻ được đóng gói trong `/WEB-INF/classes/` directory
  - Các file `*.tld` nằm trong thư mục `/WEB-INF/` directory
- **Dạng packaged (\*.jar file)**
  - Các file `*.jar` nằm trong thư mục `/WEB-INF/lib/` directory

DatTT-DSE-SOICT-HUST

114

### 3.3. Hoạt động của custom tags?

DatTT-DSE-SOICT-HUST

115

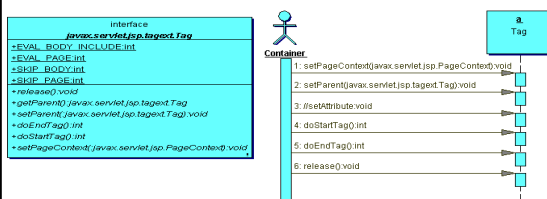
### Quá trình hoạt động?

- Bộ dịch JSP (thành servlet) xác định vị trí file TLD qua thuộc tính uri hoặc qua **taglib directive**
  - File TLD mô tả mối quan hệ giữa lớp xử lý thẻ với thẻ tương ứng
- Sử dụng 1 thẻ trong trang JSP sẽ sinh ra mã servlet tương ứng
  - Mã servlet đó chứa phần code xử lý thẻ

DatTT-DSE-SOICT-HUST

116

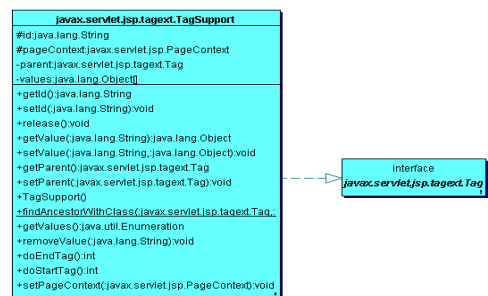
### Chuỗi các lời gọi từ Container



DatTT-DSE-SOICT-HUST

117

### Lớp TagSupport thực thi giao diện Tag



DatTT-DSE-SOICT-HUST

118

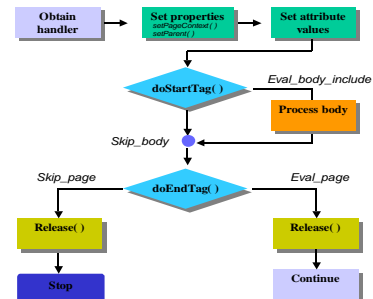
### doStartTag() và doEndTag()

- doStartTag()**
  - Xử lý thẻ mở
  - Trả về EVAL\_BODY\_INCLUDE để xử lý phần body
  - Trả về SKIP\_BODY để bỏ qua phần body
- doEndTag()**
  - Hoàn thiện việc xử lý thẻ
  - Trả về EVAL\_PAGE để xử lý tiếp phần còn lại của trang
  - Trả về SKIP\_PAGE để kết thúc xử lý phần còn lại của trang
- Cả 2 phương thức có thể trả về JspException

DatTT-DSE-SOICT-HUST

119

### Chuỗi các lời gọi từ Container



DatTT-DSE-SOICT-HUST

120

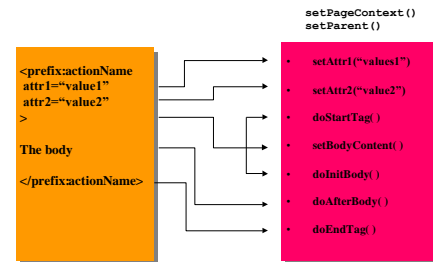
## Xử lý thẻ như thế nào?

- Công nghệ JSP sẽ tạo (hoặc sử dụng lại) một instance của lớp Tag ứng với phần tử thẻ trong trang JSP
  - Lớp được chỉ định trong file TLD
- Container gọi phương thức `setPageContext()` và `setParent()`
  - `parent` (có thể là null)
  - `PageContext` cung cấp truy cập tới: request, response, out, session, page và các thuộc tính (attributes)
- Container gọi phương thức `setX()` nếu thẻ sử dụng thuộc tính X này
- Container gọi phương thức `doStartTag()` và `doEndTag()`
- Container gọi phương thức `release()` để trả tự do cho instance của lớp Tag đang dùng (để sử dụng lại về sau)

DatTT-DSE-SOICT-HUST

121

## Xử lý thẻ như thế nào?



DatTT-DSE-SOICT-HUST

122

## Response Output của trang JSP

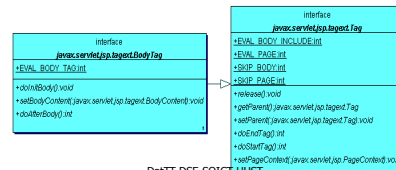
- Output của trang được viết vào đối tượng `JSPWriter`
  - Mặc định đã được xử lý buffer
- Nội dung từ `JSPWriter` sau đó sẽ được chuyển hết vào output stream của `ServletResponse`.
- Tags output sẽ được lồng vào trong `JSPWriter` streams.

DatTT-DSE-SOICT-HUST

123

## BodyTag, BodyTagSupport

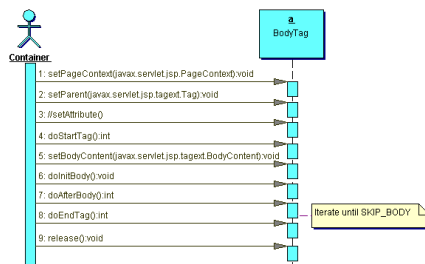
- Để xử lý phần body nhiều lần, sử dụng lớp `BodyTagSupport` thay cho lớp `BodyTag`
- Ví dụ:
  - Các item trong đơn đặt hàng, kết quả tìm kiếm, ...



DatTT-DSE-SOICT-HUST

124

## BodyTag hoạt động như thế nào?



DatTT-DSE-SOICT-HUST

125

## BodyContent

```

javax.servlet.jsp.tagext.BodyContent
-enclosingWriter: javax.servlet.jsp.JspWriter
+BodyContent(javax.servlet.jsp.JspWriter)
+getString(): java.lang.String
+getReader(): java.io.Reader
+writeOut(java.io.Writer): void
+getEnclosingWriter(): javax.servlet.jsp.JspWriter
+clearBody(): void

```

DatTT-DSE-SOICT-HUST

126

## Xử lý BodyContent (1)

Công nghệ JSP tạo stream vào (BodyContent) để chứa phần text của body

- BodyContent được truyền cho BodyTag qua phương thức setBodyContent()
- Gọi phương thức doStartBody()
- Gọi phương thức doInitBody()
- Phần text của body sẽ được đọc vào BodyContent

DatTT-DSE-SOICT-HUST

127

## Xử lý BodyContent (2)

- Tiếp đến, gọi phương thức doAfterBody():
  - Trong đó, phải xử lý phần nội dung của stream vào (chính là BodyContent), và ghi dữ liệu vào stream ra (chính là JSPWriter)
  - Có thể lại đọc lại phần body, bằng cách trả về EVAL\_BODY\_TAG
- Cuối cùng, gọi phương thức doEndTag().

DatTT-DSE-SOICT-HUST

128

## Ý nghĩa các phương thức trong giao diện BodyTag

- doStartTag():
  - Tương tự như trong giao diện Tag, ngoại trừ:
    - Trả về EVAL\_BODY\_TAG để xử lý phần body
- doBodyInit():
  - Chuẩn bị xử lý phần body
- doAfterBody() :
  - Xử lý phần body chứa trong BodyContent
  - Có thể trả về EVAL\_BODY\_TAG để xử lý lại
- DoEndTag() :
  - Như trong giao diện Tag

DatTT-DSE-SOICT-HUST

129