Bài 6
Các Thẻ Tùy Biến

MỤC TIÊU

- Giải thích các thẻ tùy biến trong các trang JSP
- Mô tả kỹ thuật chung được sử dụng trong các thẻ tùy biến
- Giải thích cơ chế làm việc của các thư viện thẻ tùy biến
- Giải thích các kiểu khác nhau của các thẻ tùy biến
- Giải thích cách tạo các thẻ tùy biến

Giới thiệu các thẻ tùy biến

- Thẻ tùy biến cho phép nhúng các mã Java vào trong các trang jsp
- Sử dụng thẻ tùy biến liên quan đến 3 bước:

Tạo một mô tả thư viện thẻ (TLD)

- File TLD chứa thông tin cho mỗi thẻ có trong thư viện thẻ.
- Nó là một tài liệu XML.
- Được sử dụng để xác thực các thẻ.

Tạo một lớp điều khiển thẻ

- Điều khiển thẻ là một lớp Java mà định nghĩa một mô tả thẻ trong file TLD.

Sử dụng các thẻ tùy biến trong các trang jsp

- Trang JSP sẽ sử dụng các thẻ đã được định nghĩa trong một thư viện thẻ bằng cách sử dụng chỉ thị xử lý taglib trong trang trước khi bất kỳ thẻ nào được sử dụng.

File mô tả thư viện thẻ (1/3)

- Có phần đuôi mở rộng .tld, chứa định nghĩa của các thẻ tùy biến
- Được đưa vào trang jsp với chỉ thị xử lý <%@tablib>

```
<%@taglib prefix="u" uri="/WEB-INF/tlds/sampleLib.tld" %>
```

  - uri: Đường dẫn xác định duy nhất file TLD của thẻ
  - prefix: Tiền tố để phân biệt các thẻ khác nhau trong một trang jsp

- Vị trí file mô tả thư viện thẻ
  - Thường đặt dưới thư mục: **`../WEB-INF/tld/sampleLib.tld`**

File mô tả thư viện thẻ (2/3)

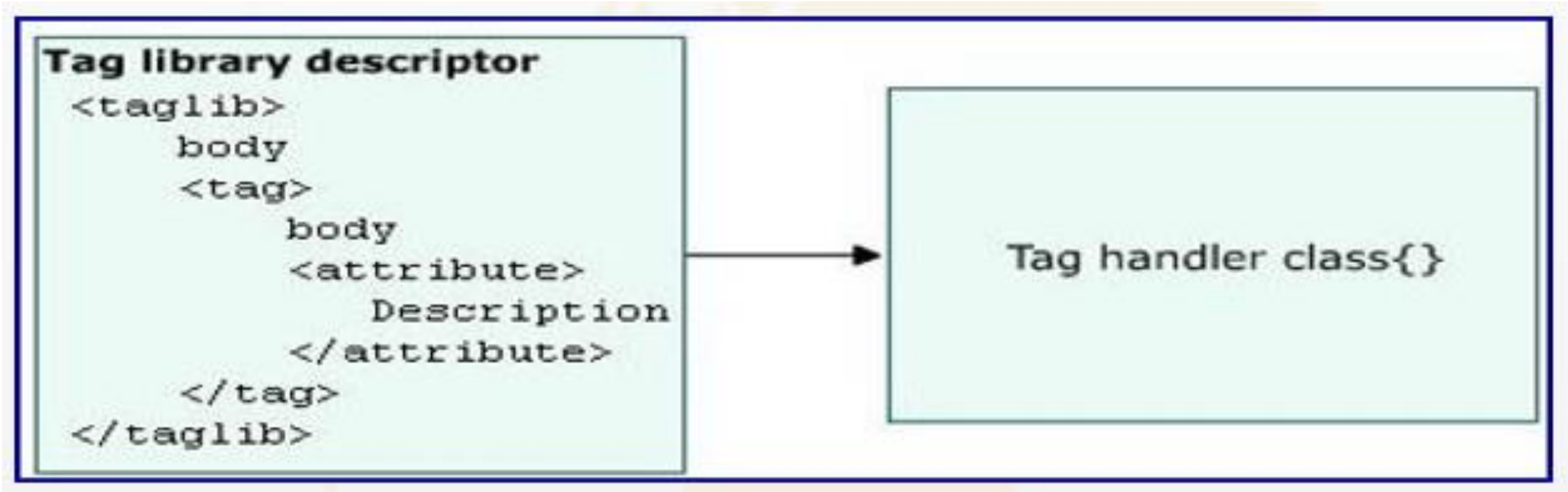
```
<taglib version="2.0" xmlns="http://java.sun.com/xml/ j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema- instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee/web-
jsptaglibrary_2_0.xsd">
<!-- The tag library and the tag is described inside this
tag-->
<taglib>
<!-- The version of the tag library -->
    <tlib-version>2.0</tlib-version>
<!-- The JSP specification version for the tag library --
>
    <jsp-version>2.0</jsp-version>
<!-- This is the name assigned to the tag library -->
    <short-name>tags</short-name>
<!-- This specifies the path of the TLD file-->
    <uri>tag lib version id</uri>
```

File mô tả thư viện thẻ (3/2)

```
<!-- Provides a short description of the tag library-->
<description>The tag library </description>
<!-- Provides a detailed description of the tag-->
<tag>
    <!-- This is the name of the tag used in the JSP page-->
    <name>name</name>
    <!-- This is the name of the tag handler class for the  concerned tag-->
    <tag-class>tags.NameTag</tag-class>
    <!-- This specifies the type of body content. It can be empty,JSP or tag-dependent -->
    <body-content>empty</body-content>
    <!-- This describes the attribute passed with the tag-->
    <attribute>
       <!-- This specifies the name of the attribute passed with the tag- ->
       <name>name</name>
    </attribute>
</tag>
</taglib>
```

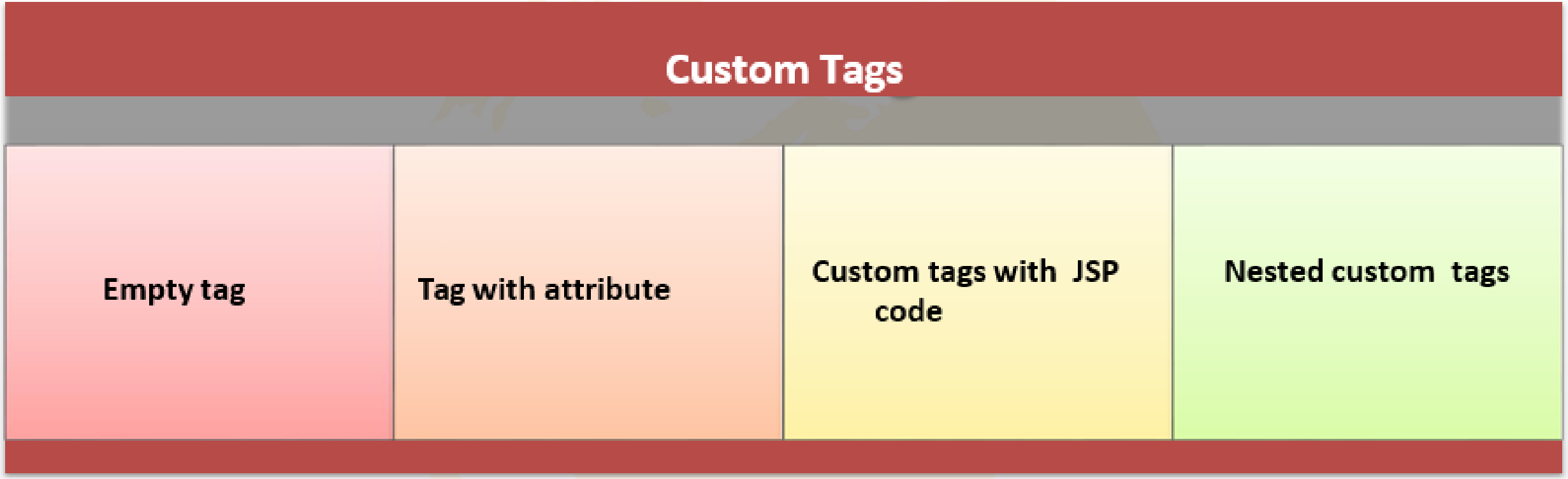
Lớp điều khiển thẻ

- Là một file class Java đơn giản chứa mã điều khiển cho chức năng của thẻ
- Có 2 kiểu là classic và simple



Các loại thẻ tùy biến

- Có 4 loại thẻ tùy biến sau:



Thẻ với các thuộc tính

- Còn gọi là các thẻ được tham số hóa
- Các thuộc tính được truyền tới thẻ giống như các đối số được truyền tới một phương thức

```
<html>
<body>
      <%@ taglib prefix="tagPrefix" uri="sampleLib.tld"%>
      <h1>
            <tagPrefix:TagsWithAttributesuserName="userName1"/>
      </h1>
</body>
</html>
```

Thẻ với mã JSP

- Có thể chứa mã JSP như là nội dung bên trong chúng
- Mã JSP được tính toán trước khi thẻ được thực thi

```
<html><body>
   <%@ taglib prefix="tagPrefix" uri="sampleLib.tld" %>
    <tagPrefix:if condition="true">
          UserName is: <%= request.getParameter("userName") %>
   </tagPrefix:if>
</body></html>
```

Thẻ tùy biến lồng

- Đó là các thẻ được khai báo bên trong một thẻ khác
- Một thẻ có thể chứa thẻ khác, còn gọi là thẻ cha. Các thẻ bên trong gọi là thẻ con của nó.

```
<html><body>
<%@ taglib prefix="tagPrefix" uri="sampleLib.tld" %>
<tagPrefix:switch conditionValue='<%=request. getParameter("userName")%>'>
    <tagPrefix:case caseValue=" userName1"> First User</tagPrefix:case>
    <tagPrefix:case caseValue=" userName2" >Second User</tagPrefix:case>
</tagPrefix:switch>
</body></html>
```

Định nghĩa thẻ trong file .tld

```
<tag>
<name>MyTag</name>
<tag-class>pkg.MyTag</tag-class>
<body-content>JSP</body-content>

<attribute>
    <name>attr1</name>
    <required>true</required>
</attribute>

<attribute>
    <name>attr2</name>
    <required>false</required>
    <rtexprvalue>true</rtexprvalue>
</attribute>

</tag>
```

- body-content có thể là các giá trị: empty, JSP, hoặc tabdependent
- rtexprvalue: Giá trị của thuộc tính có được tính toán động hay không

Các giao diện và lớp định nghĩa các thẻ tùy biến

- Các giao diện:
  - Tag: Kết nối giữa một tag handler và servlet của một trang jsp
  - IterationTag: Sử dụng bởi các tag handler mà yêu cầu thực thi tag, kế thừa giao diện Tag
  - BodyTag: Kế thừa giao diện IterationTag và thêm vào 2 phương thức để xử lý phần thân của thẻ, doInitBody() và setBodyContent()
- Các lớp:
  - TagSupport: Thực thi giao diện Tag, IterationTag, chứa các phương thức doStartTag(), doEndTag() và doAfterBody().
  - BodyTagSupport: Thực thi giao diện BodyTag và kế thừa lớp TagSupport

Các phương thức của giao diện Tag

```
//The evaluation of start tag starts
public int doStartTag() throws JspException {
try {
        pageContext.getOut().print("Creating Custom Tag by  Implementing Tag Interface");
} catch (IOException ex) {
    ex.printStackTrace();
}
        // The SKIP BODY is returned
            return SKIP_BODY;
}
//The evaluation of end tag starts
public int doEndTag() throws JspException {
        // the SKIP PAGE is returned
            return SKIP_PAGE;
    }
//All the resources held by the tag handler is released
public void release() {
    }

//The current value of the pageContext is set
public void setPageContext(PageContext pc) {
    this.pageContext = pc;
  }
  }
```

Các phương thức của giao diện IterationTag

- Giao diện này có phương thức `doAfterBody()`.
- `doAfterBody()`:
  - Cho phép tính toán lại điều kiện trong phần thân của thẻ.
  - Trả về hằng số `SKIP_BODY` hoặc `EVAL_BODY_AGAIN`, nếu phương thức `doStartTag()` trả về `EVAL_BODY_INCDLUDE`, khi đó phương thức này trả về `EVAL_BODY_AGAIN` và phương thức `doStartTag()` được tính toán lại một lần nữa
  - Khi phương thức `doAfterBody()` trả về `SKIP_PAGE`, khi đó phương thức `doEndTag()` được gọi.

Các phương thức của giao diện BodyTag

- setBodyContent():
  - Thiết lập đối tượng bodyContent cho lớp điều khiển thẻ
  - Đóng gói nội dung thân của thẻ tùy biến để xử lý
  - Được tự động gọi bởi trang JSP trước phương thức doInitBody()

```
public void setBodyContent(BodyContent b){ }
```

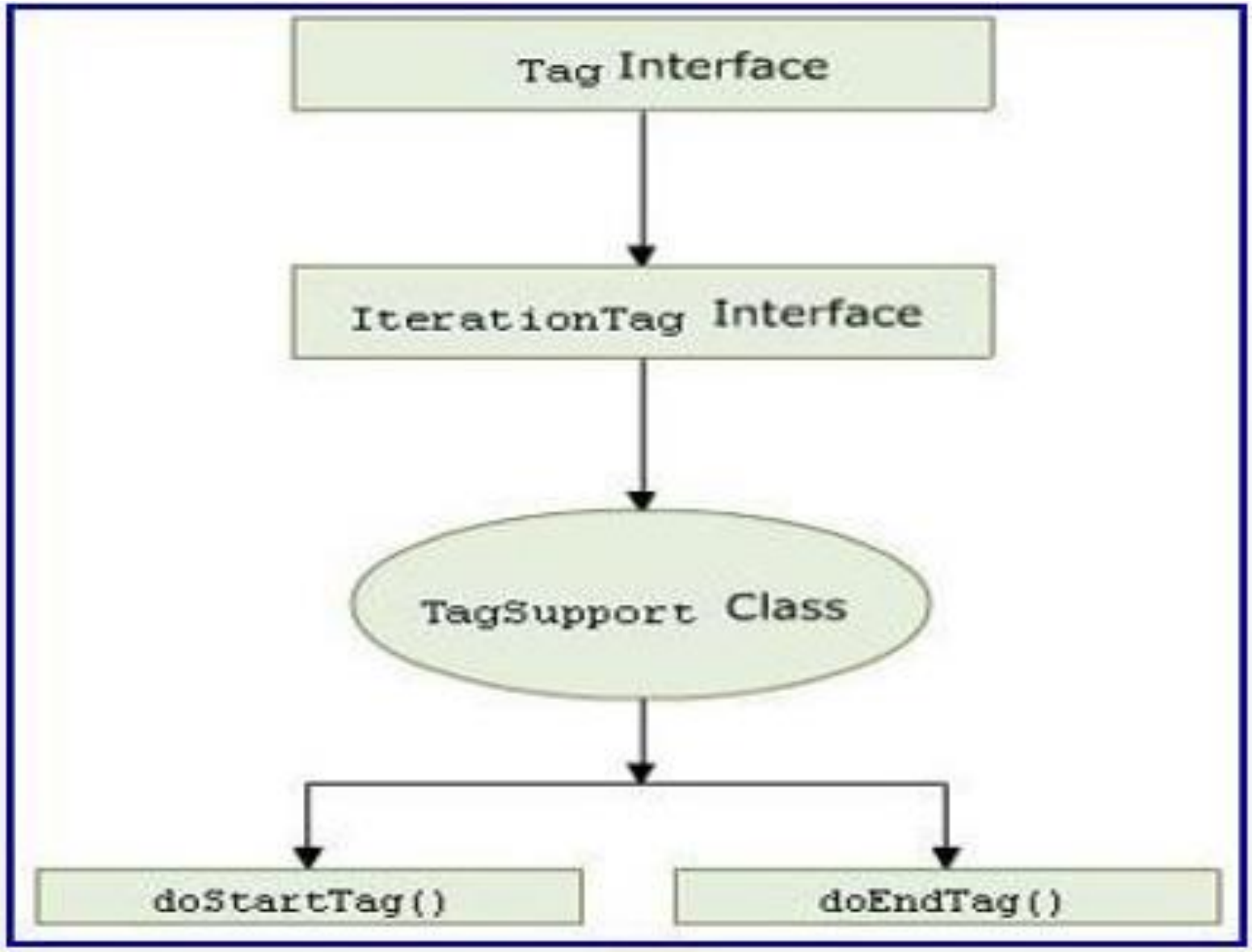
- doInitBody():
  - Được gọi ngay sau phương thức setBodyContent() đã trả về đối tượng bodyContent và trước khi tính toán phần thân lần đầu tiên

```
public int doAfterBody()throws JspException
```

Lớp TagSupport

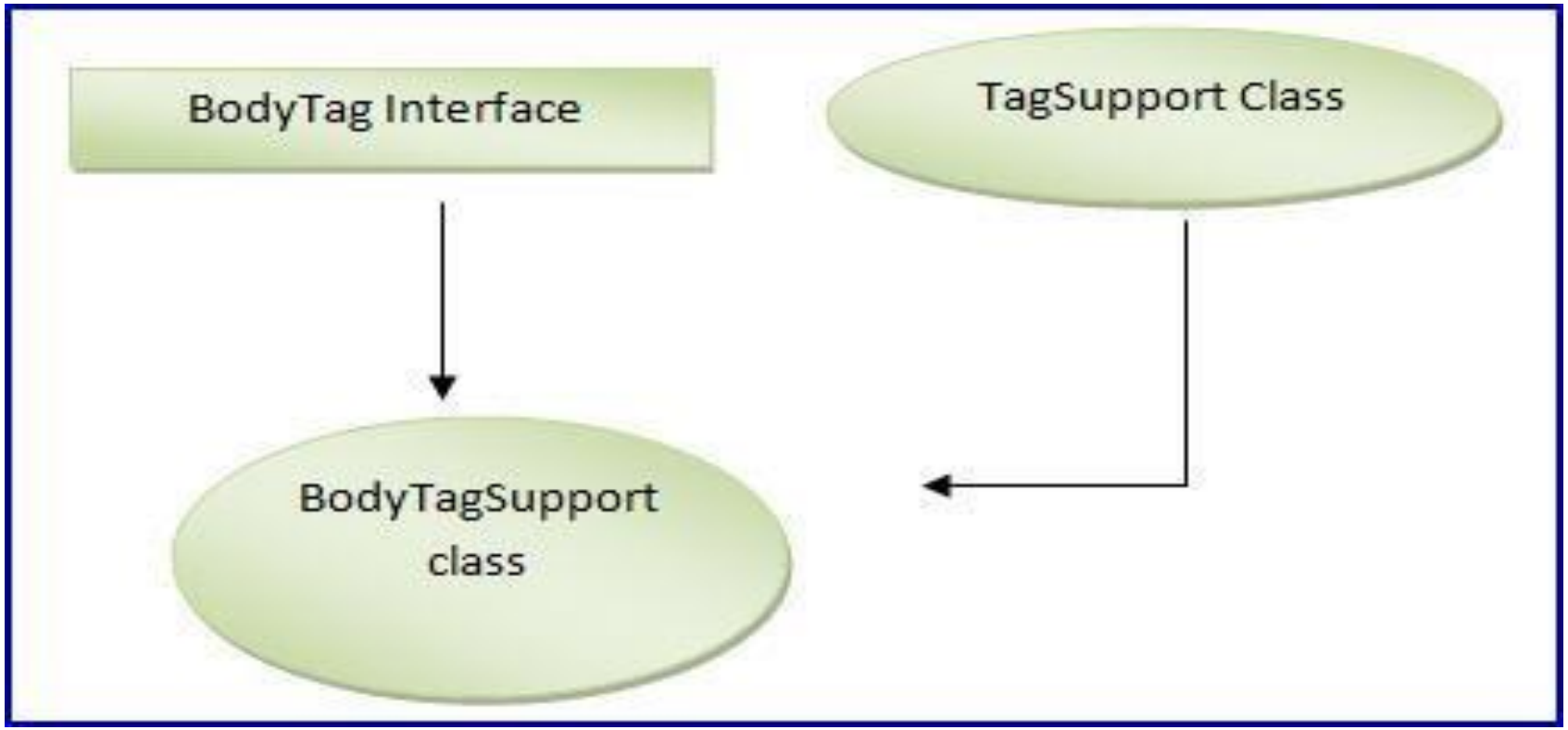
- Thực thi giao diện IterationTag
- Cung cấp thực thi mặc định cho các phương thức của giao diện Tag và IterationTag
- Có các phương thức:
  - doStartTag()
  - doEndTag()

Kiến trúc lớp TagSupport



Lớp BodyTagSupport

- Thực thi giao diện BodyTag và kế thừa lớp TagSupport
- Tất cả các phương thức của lớp TagSupport và giao diện BodyTag được thực thi mặc định trong lớp này



HỎI ĐÁP

TRẢI NGHIỆM THỰC HÀNH

TRÂN TRỌNG CẢM ƠN!

238 Hoàng Quốc Việt, Bắc Từ Liêm, Hà Nội

0968.27.6996

tuyensinh@bachkhoa-aptech.edu.vn

www.bachkhoa-aptech.edu.vn