



LẬP TRÌNH JAVA 5

BÀI 4: EL & JSTL

- ◎ **Nắm vững kỹ thuật lập trình giao diện trong JSP**
 - ◎ Expression Language (EL)
 - ◎ Java Standard Tag Library (JSTL)



- ❑ EL là sự rút ngắn tuyệt vời trong việc viết mã làm việc với các attribute đặt trong các scope (page, request, session và application)
- ❑ EL được giới thiệu trong phiên bản JSP 2.0
- ❑ Trong phần này chúng ta nghiên cứu sử dụng EL để truy xuất
 - ❖ Attribute trong các scope
 - ❖ Thuộc tính của bean
 - ❖ Phần tử trong Collection
 - ❖ Phần tử trong Map
 - ❖ Tham số, cookie và header

$\{biểu\}$ thức

□ Cú pháp:

$\${<\text{biểu thức}>}$

- ❖ **$<\text{biểu thức}>$** là một biểu thức cho một giá trị được kết xuất tại vị trí đặt biểu thức EL.
- ❖ Trong biểu thức này có thể có thể chứa **attribute**, **parameter**, **cookie** hay **header**

□ Ví dụ:

- ❖ **$\${\text{salary} * 2}$** : nhân đôi giá trị của attribute salary và kết xuất giá trị của biểu thức
- ❖ **$\${\text{sessionScope['salary']}}$** : kết xuất giá trị của attribute là salary đặt trong session
- ❖ **$\${\text{param.salary}}$** : kết xuất giá trị của tham số salary

Controller

```
@RequestMapping("/el/demo1")
public String sayHello(ModelMap model, HttpSession session){
    session.setAttribute("name", "Tèo");
    model.addAttribute("salary", 2000)
}
```

JSP

```
<li>name: ${name}</li>
<li>salary: ${salary}</li>
<li>requestScope.name: ${requestScope.name}</li>
<li>requestScope.salary: ${requestScope.salary}</li>
<li>sessionScope.name: ${sessionScope.name}</li>
<li>sessionScope.salary: ${sessionScope.salary}</li>
```

- name: Tèo
- salary: 2000
- requestScope.name:
- requestScope.salary: 2000
- sessionScope.name: Tèo
- sessionScope.salary:

Why?

❑ Như đã biết trong JSP có 4 scope chia sẻ dữ liệu

- ❖ Page: pageScope
- ❖ Request: requestScope
- ❖ Session: sessionScope
- ❖ Application: applicationScope

❑ Scope API gồm

- ❖ `setAttribute(name, value)`
- ❖ `getAttribute(name)`
- ❖ `removeAttribute(name)`
- ❖ `getAttributeNames()`

Các phương thức này vẫn hữu dụng với viết mã Java. Tuy nhiên trong JSP, theo phong cách mới thì lập trình viên sử dụng EL và JSTL.

- ❑ Các biểu thức EL sau đây sẽ kết xuất attribute x đặt trong scope cụ thể
 - ❖ `${pageScope['x']}` hoặc `${pageScope.x}`
 - ❖ `${requestScope['x']}` hoặc `${requestScope.x}`
 - ❖ `${sessionScope['x']}` hoặc `${sessionScope.x}`
 - ❖ `${applicationScope['x']}` hoặc `${applicationScope.x}`
- ❑ Biểu thức EL sau đây sẽ truy tìm và kết xuất giá trị của attribute message trong tất cả Scope `${message}`
 - ❖ Trình tự tìm kiếm attribute message là
`pageScope->requestScope->sessionScope->applicationScope`
 - ❖ Nếu tìm thấy thì dừng lại, ngược lại cho giá trị rỗng

- ❑ Nếu attribute là một bean thì EL cho phép truy xuất các thuộc tính một cách đơn giản
- ❑ Lớp JavaBean là lớp
 - ❖ Phải khai báo là public
 - ❖ Có Constructor mặc định không tham số
 - ❖ Đọc/ghi dữ liệu thông qua phương thức getter/setter
- ❑ Cú pháp truy xuất thuộc tính bean:
 - ❖ **`${bean.property}`**
Kết xuất giá trị của thuộc tính property của attribute bean. Có nghĩa là kết xuất kết quả của phương thức bean.getProperty()
- ❑ Ví dụ:
 - ❖ `${student.mark}` ~ xuất student.getMark()

Controller

```
@RequestMapping("/el/demo2")  
public String demo2(ModelMap model) {  
    Student student = new Student("Phương", 10.0, "APP");  
    model.addAttribute("student", student);  
    return "el/demo2";  
}
```

JSP

```
<li>name: ${student.name}</li>  
<li>mark: ${student.mark}</li>
```

- name: Phương
- mark: 10.0

- ❑ Nếu attribute là mảng hoặc tập hợp thì EL cho phép sử dụng chỉ số để truy xuất các phần tử.

Controller

```
@RequestMapping("/el/demo3")  
public String demo3(ModelMap model) {  
    List<String> list = new ArrayList<>();  
    list.add("Phương");  
    list.add("Cường");  
    model.addAttribute("items", list);  
    return "el/demo3";  
}
```

JSP

```
<li>${items[0]}</li>  
<li>${items[1]}</li>
```

- Phương
- Cường

- ❑ Nếu attribute là Map thì EL cho phép sử dụng key để truy xuất các phần tử.

Servlet hoặc Controller

```
@RequestMapping("demo3")
public String demo3(ModelMap model) {
    Map<String, Object> map = new HashMap<>();
    map.put("name", "Phương");
    map.put("mark", 9.5);
    model.addAttribute("student", map);
    return "el/demo3";
}
```

Chú ý: Có 2 cách truy xuất

1. Map[key]
2. Map.key

JSP

`${student['name']}`
`${student.mark}`

- name: Phương
- mark: 9.5



DEMO



Giải thích el/demo1-4.htm

- ❑ Với EL, bạn có thể truy xuất tham số và cookie trong JSP một cách đơn giản
- ❑ Truy xuất tham số
 - ❖ `${param[<tên tham số>]}`
 - ❖ Hoặc `${param.<tên tham số>}`
- ❑ Truy xuất cookie
 - ❖ `${cookie[<tên cookie>].value}`
 - ❖ Hoặc `${cookie.<tên cookie>.value}`
- ❑ Ví dụ
 - ❖ `${param.salary}`
 - ❖ `<input value="${cookie.userid.value}">`



DEMO

Giải thích el/login.htm





LẬP TRÌNH JAVA 5

PHẦN 2

- ❑ JSTL là bộ thư viện thẻ chuẩn được bổ sung với mục đích tối ưu lập trình giao diện trong JSP
- ❑ Các thư viện cần thiết cho JSTL gồm
 - ❖ jstl-api.jar
 - ❖ jstl-impl.jar
- ❑ Trong JSTL có rất nhiều bộ thẻ để xử lý các vấn đề khác nhau
 - ❖ Core: chứa các thẻ lệnh điều khiển cơ bản
 - ❖ Format: chứa các thẻ định dạng và đa ngôn ngữ
 - ❖ Xml: chứa các thẻ xử lý xml
 - ❖ Sql: chứa các thẻ làm việc với CSDL
 - ❖ Function: cung cấp các hàm hỗ trợ cho EL

❑ Trong phạm vi môn học này, các bạn sẽ sử dụng các bộ thẻ sau

❖ Thư viện cơ bản (core)

<%@ taglib uri="http://java.sun.com/jstl/**core_rt**" prefix="c" %>

❖ Thư viện định dạng (format)

<%@ taglib uri="http://java.sun.com/jstl/**fmt_rt**" prefix="fmt" %>

❖ Thư viện hàm (function)

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

❑ Core chứa các thẻ thay thế các lệnh cơ bản trong Java để phù hợp với lập trình giao diện theo cú pháp thẻ.

❖ `<c:if>`

➤ Tương tự lệnh if trong java

❖ `<c:choose>`

➤ Tương tự if...else if...else trong java

❖ `<c:forEach>`

➤ Tương tự for-each trong java

❖ `<c:set>`

➤ Tương tự: `<scope>.setAttribute()` trong java

❖ `<c:remove>`

➤ Tương tự `<scope>.removeAttribute()` trong java

❑ Cú pháp

- ❖ <c:if test="{<điều kiện>}">Nội dung</c:if>
- ❖ Nội dung sẽ được kết xuất nếu điều kiện có giá trị là true

❑ Ví dụ

```
<ul>  
  <li>Họ và tên: ${student.name}</li>  
  <li>Điểm TB: ${student.mark}</li>  
  <li>Chuyên ngành: ${student.major}</li>  
  <c:if test="${student.mark > 9}">  
    <li><strong>Danh hiệu ong vàng</strong></li>  
  </c:if>  
</ul>
```

- Họ và tên: Phương
- Điểm TB: 10.0
- Chuyên ngành: APP
- **Danh hiệu ong vàng**

□ Cú pháp

<c:choose>

<c:when test="\$ {<ĐK 1>}">Nội dung 1</c:when>

<c:when test="\$ {<ĐK 2>}">Nội dung 2</c:when>

...

<c:otherwise> Nội dung N+1</c:otherwise>

</c:choose>

□ Diễn giải

- ❖ Xét các điều kiện từ trên xuống, nếu điều kiện thứ i đúng thì kết xuất Nội dung i. Nếu không có điều kiện nào thỏa mãn thì kết xuất nội dung thứ N+1 (<c:otherwise>).

```
<ul>
  <li>Họ và tên: ${student.name}</li>
  <li>Điểm TB: ${student.mark}</li>
  <li>Chuyên ngành: ${student.major}</li>
  <li>Xếp loại:
    <c:choose>
      <c:when test="${student.mark < 5}">Yếu/kém</c:when>
      <c:when test="${student.mark > 9}">Giỏi</c:when>
      <c:when test="${student.mark > 6.5}">Khá</c:when>
      <c:otherwise>Trung bình</c:otherwise>
    </c:choose>
  </li>
</ul>
```

- Họ và tên: Cường
- Điểm TB: 8.5
- Chuyên ngành: BIZ
- Xếp loại: Khá

□ Cú pháp

<c:forEach

var="biến chứa phần tử hiện tại"

items="tập hợp các phần tử"

begin="vị trí của phần tử bắt đầu mặc định là 0"

end="vị trí của phần tử cuối cùng mặc định là vị trí phần tử cuối cùng"

varStatus="biến trạng thái" >

Nội dung

</c:forEach>

□ Diễn giải

- ❖ Duyệt các phần tử từ vị trí **begin** đến **end** trong tập hợp **items**. **Var** sẽ nắm phần tử hiện tại còn **varStatus** sẽ nắm thông tin trạng thái của phần tử hiện tại.

```
<c:forEach begin="1" end="6" varStatus="status">  
    <h${status.index}>Hello World</h${status.index}>  
</c:forEach>
```

❑ Vòng lặp trên sẽ kết xuất

<h1>Hello World</h1>

<h2>Hello World</h2>

<h3>Hello World</h3>

<h4>Hello World</h4>

<h5>Hello World</h5>

<h6>Hello World</h6>

```
<c:forEach var="p" items="{products}" begin="10" end="25" varStatus="status">
    <li>Vị trí: ${status.index} </li>
    <li>Tên hàng hóa: ${p.name} </li>
</c:forEach>
```

❑ Ví dụ trên sẽ kết xuất thông tin của các phần tử từ 10 đến 25 trong tập hợp products. Mỗi phần tử sẽ xuất vị trí và tên sản phẩm:

- ❖ Vị trí: 10
- ❖ Tên hàng hóa: Abc
- ❖ Vị trí: 11
- ❖ Tên hàng hóa: Xyz
- ❖ ...

VÍ DỤ 3 - <C:FOREACH>

```
<table border="1" style="width:100%">
<tr>
  <th>Họ và tên</th>
  <th>Điểm</th>
  <th>Chuyên ngành</th>
  <th>Xếp loại</th>
</tr>
<c:forEach var="student" items="{students}">
  <tr>
    <td>${student.name}</td>
    <td>${student.mark}</td>
    <td>${student.major}</td>
    <td>
      <c:choose>
        <c:when test="${student.mark < 5}">Yếu/kém</c:when>
        <c:when test="${student.mark > 9}">Giỏi</c:when>
        <c:when test="${student.mark > 6.5}">Khá</c:when>
        <c:otherwise>Trung bình</c:otherwise>
      </c:choose>
    </td>
  </tr>
</c:forEach>
</table>
```

Họ và tên	Điểm	Chuyên ngành	Xếp loại
Phương	10.0	BIZ	Giỏi
Cường	8.5	APP	Khá
Hạnh	3.5	MOB	Yếu/kém
Hương	5.5	MUL	Trung bình

- ❑ <c:set> được sử dụng để tạo một attribute tương tự <scope>.setAttribute("name", "value") trong java
- ❑ Cú pháp
 - ❖ <c:set var="name" value="value" scope="session"/>
 - ❖ <c:set var="name" scope="session">value</c:set>
- ❑ <c:remove> được sử dụng để xóa một attribute tương tự <scope>.removeAttribute("name") trong java
- ❑ Cú pháp
 - ❖ <c:remove var="name" scope="session"/>



DEMO

Giải thích

+ [jstl/core-if.htm](#)

+ [jstl/core-choose.htm](#)

+ [jstl/core-foreach.htm](#)



□ Định dạng số

❖ `<fmt:formatNumber value="" type="">`

- Value: số cần định dạng
- Type: kiểu định dạng

❖ Ví dụ

`<fmt:formatNumber value="1000000" type="currency" />`

`<fmt:formatNumber value="0.51" type="percent" />`

□ Định dạng thời gian

❖ `<fmt:formatDate value="" pattern="">`

- Value: thời gian cần định dạng
- Pattern: mẫu định dạng thời gian

❖ Ví dụ

`<fmt:formatDate value="${date}" pattern="dd-MM-yyyy" />`

Bean Class

```
public class Product {
    private String name;
    private Date date;
    private Double price;
    private Double discount;

    public Product() {
        this.date = new Date();
    }

    public Product(String name, Double price, Double discount) {
        this.name = name;
        this.price = price;
        this.discount = discount;
        this.date = new Date();
    }
    getter/setter
}
```

```
@RequestMapping("format")
public String format(ModelMap model) {
    Product product = new Product("iPhone 9", 2579.5, 0.05);
    model.addAttribute("product", product);
    return "jstl/format";
}
```

Controller Class

```

<ul>
  <li>Tên sản phẩm: ${product.name}</li>
  <li>Đơn giá:
    <fmt:formatNumber value="${product.price}" type="currency"/>
  </li>
  <li>Giảm giá:
    <fmt:formatNumber value="${product.discount}" type="percent"/>
  </li>
  <li>Ngày nhập:
    <fmt:formatDate value="${product.date}" pattern="dd-MM-yyyy"/>
  </li>
</ul>

```

- Tên sản phẩm: iPhone 9
- Đơn giá: \$2,579.50
- Giảm giá: 5%
- Ngày nhập: 18-11-2016



DEMO

Giải thích
+ [jstl/format.htm](#)



❑ JSTL cung cấp tập các hàm hỗ trợ xử lý cho biểu thức EL. Các hàm này tập trung xử lý chuỗi và tập hợp.

❑ Ví dụ 1:

`${fn:substring(0, 100, description)}...`

❖ Ví dụ này chỉ hiển thị 100 ký tự đầu tiên của attribute description

❑ Ví dụ 2:

`<c:if test="${fn:startsWith("Nguyễn ", fullname)}">`

Người này họ Nguyễn

`</c:if>`

❖ Ví dụ này kiểm tra attribute fullname có phải bắt đầu bởi chữ "Nguyễn " hay không

Tên hàm	Đối số	Trả về	Mô tả mục đích
fn:contains	String, String	boolean	Chuỗi (1) có chứa chuỗi (2) hay không
fn:containsIgnoreCase	String, String	boolean	Chuỗi (1) có chứa chuỗi (2) hay không (không phân biệt hoa thường)
fn:endsWith	String, String	boolean	Chuỗi (1) có kết thúc bởi (2) hay không
fn:escapeXML	String	String	Mã hóa thành thực thể các ký tự phạm cú pháp XML
fn:indexOf	String, String	int	Tìm vị trí xuất hiện đầu tiên của chuỗi (2) trong chuỗi (1)
fn:join	String[], String	String	Gia nhập các phần tử trong mảng (1) thành chuỗi sử dụng chuỗi(2) như là chuỗi phân cách.
fn:length	Map; array; Collection; Iterator; Enumeration; or String	int	Tìm độ dài của chuỗi hay số lượng các phần tử trong tập hợp.

Tên hàm	Đối số	Trả về	Mô tả mục đích
fn:replace	String, String, String	String	Thay thế chuỗi (1) bởi chuỗi (3) trong chuỗi (1)
fn:split	String, String	String[]	Tách chuỗi (1) thành mảng sử dụng chuỗi (2) như chuỗi phân cách
fn:startsWith	String, String	boolean	Chuỗi đối số thứ nhất có bắt đầu bởi chuỗi đối số thứ hai hay không
fn:substring	String, int, int	String	Lấy chuỗi trong chuỗi (1) tính từ vị trí (1) cho đến vị trí (3)
fn:substringAfter	String, String	String	Lấy chuỗi con trong chuỗi (1) đứng sau chuỗi (2)
fn:substringBefore	String, String	String	Lấy chuỗi con trong chuỗi (1) đứng trước chuỗi (2)
fn:toLowerCase	String	String	Đổi chuỗi sang chữ thường
fn:toUpperCase	String	String	Đổi chuỗi sang chữ HOA
fn:trim	String	String	Cắt bỏ khoảng trắng 2 đầu chuỗi

☑ EL

- ☑ Truy xuất attribute trong các scope
- ☑ Truy xuất thuộc tính bean
- ☑ Truy xuất phần tử mảng và tập hợp
- ☑ Truy xuất phần tử của map
- ☑ Truy xuất tham số, cookie

☑ JSTL

- ☑ Core
- ☑ Format
- ☑ Function





Cảm ơn