JSP 内置对象作用域

4个

page、request、session、application

setAttribute、getAttribute

page 作用域：对应的内置对象是 pageContext。

request 作用域：对应的内置对象是 request。

session 作用域：对应的内置对象是 session。

application 作用域：对应的内置对象是 application。

page < request < session < application

page 只在当前页面有效。

request 在一次请求内有效。

session 在一次会话内有效。

application 对应整个 WEB 应用的。

- 网站访问量统计

```
<%
  Integer count = (Integer) application.getAttribute("count");
  if(count == null){
    count = 1;
    application.setAttribute("count",count);
  }else{
    count++;
    application.setAttribute("count",count);
  }
%>
您是当前的第<%=count%>位访客
```

EL 表达式

Expression Language 表达式语言，替代 JSP 页面中数据访问时的复杂编码，可以非常便捷地取出域对象（pageContext、request、session、application）中保存的数据，前提是一定要先 setAttribute，EL 就相当于在简化 getAttribute

${变量名} 变量名就是 setAttribute 对应的 key 值。

1、EL 对于 4 种域对象的默认查找顺序：

pageContext -》request-〉session-》application

按照上述的顺序进行查找，找到立即返回，在 application 中也无法找到，则返回 null

2、指定作用域进行查找

pageContext：${pageScope.name}

request：${requestScope.name}

session：${sessionScope.name}

application：${applicationScope.name}

数据级联：

```
<%
//        pageContext.setAttribute("name","page");
//        request.setAttribute("name","request");
//        session.setAttribute("name","session");
//        application.setAttribute("name","application");
        User user = new User(1,"张三",86.5,new Address(1,"小寨"));
        System.out.println(user.toString());
        pageContext.setAttribute("user",user);
    %>
    <table>
        <tr>
            <th>编号</th>
            <th>姓名</th>
            <th>成绩</th>
            <th>地址</th>
        </tr>
        <tr>
            <td>${user.id}</td>
            <td>${user.name}</td>
            <td>${user.score}</td>
            <td>${user.address}</td>
        </tr>
    </table>
```

${user["id"]}

EL 执行表达式

```
${num1&&num2}
&& || ! < > <= <= ==

&& and
|| or
! not
== eq
!= ne
< lt
> gt
<= le
>= ge
empty 变量为 null，长度为0的String，size为0的集合
```

JSTL

JSP Standard Tag Library JSP 标准标签库，JSP 为开发者提供的一系列的标签，使用这些标签可以完成一些逻辑处理，比如循环遍历集合，让代码更加简洁，不再出现 JSP 脚本穿插的情况。

实际开发中 EL 和 JSTL 结合起来使用，JSTL 侧重于逻辑处理，EL 负责展示数据。

JSTL 的使用

1、需要导入 jar 包（两个 jstl.jar standard.jar）存放的位置 web/WEB-INF

2、在 JSP 页面开始的地方导入 JSTL 标签库

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

3、在需要的地方使用

```
<c:forEach items="${list}" var="user">
  <tr>
    <td>${user.id}</td>
    <td>${user.name}</td>
    <td>${user.score}</td>
    <td>${user.address.value}</td>
  </tr>
</c:forEach>
```

JSTL 优点：

1、提供了统一的标签

2、可以用于编写各种动态功能

核心标签库常用标签：

- set、out、remove、catch

set：向域对象中添加数据

```
<%
  requset.setAttribute(key,value)
%>

<c:set var="name" value="tom" scope="request"></c:set>
${requestScope.name}

<%
User user = new User(1,"张三",66.6,new Address(1,"科技路"));
request.setAttribute("user",user);
%>
${user.name}
<hr/>
<c:set target="${user}" property="name" value="李四"></c:set>
${user.name}
```

out：输出域对象中的数据

```
<c:set var="name" value="tom"></c:set>
<c:out value="${name}" default="未定义"></c:out>
```

remove：删除域对象中的数据

```
<c:remove var="name" scope="page"></c:remove>
<c:out value="${name}" default="未定义"></c:out>
```

catch：捕获异常

```
<c:catch var="error">
  <%
  int a = 10/0;
  %>
</c:catch>
${error}
```

- 条件标签：if choose

```
<c:set var="num1" value="1"></c:set>
<c:set var="num2" value="2"></c:set>
<c:if test="${num1>num2}">ok</c:if>
<c:if test="${num1<num2}">fail</c:if>
<hr/>
<c:choose>
  <c:when test="${num1>num2}">ok</c:when>
  <c:otherwise>fail</c:otherwise>
</c:choose>
```

- 迭代标签：forEach

```
<c:forEach items="${list}" var="str" begin="2" end="3" step="2"
varStatus="sta">
  ${sta.count}、${str}<br/>
</c:forEach>
```

格式化标签库常用的标签：

```
<%
request.setAttribute("date",new Date());
%>
<fmt:formatDate value="${date}" pattern="yyyy-MM-dd HH:mm:ss">
</fmt:formatDate><br/>
<fmt:formatNumber value="32145.23434" maxIntegerDigits="2"
maxFractionDigits="3"></fmt:formatNumber>
```

函数标签库常用的标签：

```
<%
request.setAttribute("info","Java,C");
%>
${fn:contains(info,"Python")}<br/>
${fn:startsWith(info, "Java")}<br/>
${fn:endsWith(info, "C")}<br/>
${fn:indexOf(info, "va")}<br/>
${fn:replace(info, "C","Python")}<br/>
${fn:substring(info, 2, 3)}<br/>
${fn:split(info, ",")[0]}-${fn:split(info, ",")[1]}
```