EL与JSTL

一、EL技术(主要是取值，不能进行逻辑判断)

1.1 EL表达式概述

EL(Express Lanuage)表达式可以嵌入在jsp页面内部，减少jsp脚本的编写，EL出现的目的是要替代jsp页面中脚本的编写。

1.2 EL表达式从域中取数据

```
<!-- 模拟域中的数据 -->
<% 将会在service方法中出现下列代码，局部的
    pageContext.setAttribute("company", "传智播客");  页面域存数据

    //存储字符串
    request.setAttribute("company", "黑马程序员");  request域存数据

    //存储一个对象
    User user = new User();
    user.setId(1);
    user.setName("zhangsan");
    user.setPassword("123");
    session.setAttribute("user", user);  session域存数据

    //存储一个集合
    List<User> list = new ArrayList<User>();
    User user1 = new User();
    user1.setId(2);
    user1.setName("lisi");
    user1.setPassword("123");
    list.add(user1);
    User user2 = new User();
    user2.setId(3);
    user2.setName("wangwu");
    user2.setPassword("123");
    list.add(user2);
    application.setAttribute("list", list);  servletContext 应用域(域)
%>
```

这些都是在jsp页面的java代码

普通java脚本从域中取值:

```
<!-- 脚本法是取出域中的值 -->
<%=request.getAttribute("company") %>  需要用该域的getAttribute方法按照键找出值，取出的是一个对象，如若需要可以强制转换
<%
    User sessionUser = (User)session.getAttribute("user");
    out.write(sessionUser.getName());
%>
```

使用EL表达式:${域名.键名}

```
EL获得pageContext域中的值：${pageScope.key};
EL获得request域中的值：${requestScope.key};
EL获得session域中的值：${sessionScope.key};
EL获得application域中的值：${applicationScope.key};
```

```
<!-- 使用EL表达式获得域中的值 -->
${requestScope.company }
${sessionScope.user.name }
${applicationScope.list[1].name}
```

1.3 使用EL表达式实现全域查找

EL从四个域中获得某个值${key};---依次从pageContext域，request域，session域，application域中获取属性，在某个域中获取后将不再向后寻找。

```
<!-- 使用el表达式 全域查找 -->
${company }
${user.name }
${list[1].name}
```

1.5 EL内置对象

再次提醒：**getParameter是得到请求体中的参数，getAttribute是得到域中的数据**

EL表达式在获取Map的值或Bean的属性值是，可以使用“点”的方法，也可以使用“下标”的方法。`${initParam.a}`与`${initParam['a']}`，它们是完成的东西相同的。但是，如果Map的键或Bean的属性名中包含下划线或横岗时，那么就必须使用“下标”方法，例如：`${initParam['a_a']}`

1.6 EL执行运算表达式

EL语言的运算符

运算符类型	运算符	说明	范例	结果
算术运算符	+	加	${16+5}	21
	-	减	${16-5}	11
	*	乘	${16*5}	80
	/或div	除	${16/5}	3.2
	%或mod	模（求余）	${16%5}	1
关系运算符	==或eq	等于	${16==5}	false
	!=或ne	不等于	${16!=5}	true
	<或lt	小于	${16<5}	false
	>或gt	大于	${16>5}	true
	<=或le	小于等于	${16<=5}	false
	>=或ge	大于等于	${16>=5}	true
逻辑运算符	&&或and	逻辑与	${16>5&&16<18}	true
	||或or	逻辑或	${16>5||16<18}	true
	!或not	逻辑非	${!(16>5)}	false
empty运算符	empty	检查是否为空值	${empty var}	如果变量var为null，就返回true
条件运算符	a?b:c	条件运算符	${16>5?16:5}	16

二、JSTL技术

2.1 JSTL概述

JSTL（JSP Standard Tag Library），JSP标准标签库，可以嵌入在jsp页面中使用标签的形式完成业务逻辑等功能。jstl出现的目的同el一样也是要代替jsp页面中的脚本代码。JSTL标准标签库有5个子库，但随着发展，目前常使用的是他的核心库。

标签库	标签库的 URI	前缀
Core	http://java.sun.com/jsp/jstl/core	c
I18N	http://java.sun.com/jsp/jstl/fmt	fmt
SQL	http://java.sun.com/jsp/jstl/sql	sql
XML	http://java.sun.com/jsp/jstl/xml	x
Functions	http://java.sun.com/jsp/jstl/functions	fn

2.2 JSTL下载与导入

JSTL下载:

从Apache的网站下载JSTL的JAR包。进入下载链接网址下载JSTL的安装包。jakarta-taglibs-standard-1.1.2.zip，然后将下载好的JSTL安装包进行解压，此时，在lib目录下可以看到两个JAR文件，分别为jstl.jar和standard.jar。其中，jstl.jar文件包含JSTL规范中定义的接口和相关类，standard.jar文件包含用于实现JSTL的.class文件以及JSTL中5个标签库描述符文件（TLD）

JSTL导入:

一般导入核心库，最常用，别的基本不用了

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

2.3 JSTL核心库的常用标签

2.3.1 if标签

```
<%
    request.setAttribute("count", 10);

%>
<!-- jstl标签经常会和el配合使用 -->
<!-- test代表的返回boolean的表达式 -->
<c:if test="${count==9 }">用EL表达式取值，会查找四大域的范围
    xxxx
</c:if>
```

核心标签库：<c:标签名>逻辑代码</c:标签名>

if标签有三大属性：

1)test:用于设置逻辑表达式

2)var:用于指定逻辑表达式中变量的名字

3)scope属性用于指定var变量的作用范围，默认值为page。

注意：没有else标签

2.3.2 forEach标签

```
<!-- forEach模拟
    for(int i=0;i<=5;i++){
        syso(i)
    }
-->
<c:forEach begin="0" end="5" var="i">
    ${i }<br/>
</c:forEach>
```

begin属性：begin属性用于指定从集合中第几个元素开始进行迭代，begin的索引值从0开始，如果

没有指定items属性，就从begin指定的值开始迭代，直到迭代结束为止
end属性：类推
var属性：用于指将当前迭代到的元素保存到page域中的名称;
items属性：表示一个集合或者数组
step属性：顾名思义

1.取List

```
//模拟List<String> strList
List<String> strList = new ArrayList<String>();
strList.add("itcast");
strList.add("itheima");
strList.add("boxuegu");
strList.add("shandingyu");
request.setAttribute("strList", strList);
```

```
<h1>取出strList的数据</h1>
<c:forEach items="${strList }" var="str">
    ${str }<br/>
</c:forEach>
```

2.取自定义对象的List集合

```
//遍历List<User>的值
List<User> userList = new ArrayList<User>();
User user1 = new User();
user1.setId(2);
user1.setName("lisi");
user1.setPassword("123");
userList.add(user1);
User user2 = new User();
user2.setId(3);
user2.setName("wangwu");
user2.setPassword("123");
userList.add(user2);
application.setAttribute("userList", userList);
```

```
<h1>取出userList的数据</h1>
<c:forEach items="${userList}" var="user">
    user的name: ${user.name }------user的password: ${user.password }<br/>
</c:forEach>
```

3.取Map

```java
//遍历Map<String,String>的值
Map<String,String> strMap = new HashMap<String,String>();
strMap.put("name", "lucy");
strMap.put("age", "18");
strMap.put("addr", "西三旗");
strMap.put("email", "licy@itcast.cn");
session.setAttribute("strMap", strMap);

//遍历Map<String,User>的值
Map<String,User> userMap = new HashMap<String,User>();
userMap.put("user1", user1);
userMap.put("user2", user2);
request.setAttribute("userMap", userMap);
```

```jsp
<h1>取出strMap的数据</h1>
<c:forEach items="${strMap }" var="entry">
    ${entry.key }====${entry.value }<br/>
</c:forEach>

<h1>取出userMap的数据</h1>
<c:forEach items="${userMap }" var="entry">
    ${entry.key }:${entry.value.name }--${entry.value.password }<br/>
</c:forEach>
```