EL表达式

- 什么是EL表达式?

  Expression Language（表达式语言）

- EL表达式能干什么?

  替代JSP页面中数据访问时的复杂编码，JSP页面中数据访问是指在JSP文件中取出域对象中的数据

  域对象（pageContext，request，session，application），getAttribute(name)取出数据。

- EL表达式的特点

  使用简单，EL只能在JSP中使用，JSP提供的一个组件，可以用更加简洁的方法来完成数据的访问，EL也属于中间层的组件，最终还是会转为底层的数据访问方式。

- 如何使用EL表达式?

  语法：${EL表达式}

  通过变量名取值（取出域对象中保存的数据，使用EL表达式的前提是首先在域对象中存入数据）

  变量名指存入域对象时的key值。

通过变量名.属性名取出对象中的属性值，如${user.id}

${user.id}并不是直接去访问对象的id属性，因为属性都是私有的，在外部不能直接访问。

${user["id"]}

```
${user.id}<br/>
${user.name}<br/>
${user.age}
<hr/>
${user["id"]}<br/>
${user["name"]}<br/>
${user["age"]}
```

EL表达式中的user.id实际上是在调用对象的getId()方法，

id--->Id--->getId()

EL表达式是从域对象中取出数据，域对象共有4种，如果4个域对象中都存有key值相同的数据时，EL表达式如何来取值?

EL表达式按照如下顺序进行取值：

pageContext>request>session>application

EL表达式会首先在pageContext中查找，若存在直接返回，若不存在来到request中继续查找，若存在直接返回，若不存在来到session中继续查找，若存在直接返回，若不存在来到application中继续查找，若存在直接返回，若不存在返回空。

以上就是EL表达式默认的查找方式，同时我们也可以强制指定EL表达式的查找方式，当4个域对象中同时存在key值相同的数据时，可以强制让EL表达式只从某个域对象中查找，而不遵循默认的查找顺序。

${指定作用域.变量名}

指定page作用域：${pageScope.name}

指定request作用域：${requestScope.name}

指定session作用域：${sessionScope.name}

指定application作用域：${applicationScope.name}

```
<%
        String pageMessage = "This is page";
        String requestMessage = "This is request";
        String sessionMessage = "This is session";
        String applicationMessage = "This is application";
        pageContext.setAttribute("message",pageMessage);
        request.setAttribute("message",requestMessage);
        session.setAttribute("message",sessionMessage);
        application.setAttribute("message",applicationMessage);
%>

${applicationScope.message}
```

EL表达式取出域对象中的集合数据

- List

```
${requestScope.list[0].id}
```

- Map

```
${requestScope.map.user.name}
${requestScope.map["user1"].name}
```

- EL表达式进行逻辑运算

  把运算符和表达式写到同一个${}中，不能分开写，如

  ```
  ${requestScope.flag1} || ${requestScope.flag}
  ```

  应该这样写

```
${requestScope.flag1 || requestScope.flag}
```

```
与运算              ${表达式1 && 表达式2} / ${表达式1 and 表达式2}
或运算              ${表达式1 || 表达式2} / ${表达式1 or 表达式2}
非运算              ${!表达式} / ${not 表达式}
判断是否相等        ${表达式1 == 表达式2} / ${表达式1 eq 表达式2}
判断不相等          ${表达式1 != 表达式2} / ${表达式1 ne 表达式2}
小于                ${表达式1 < 表达式2} / ${表达式1 lt 表达式2}
大于                ${表达式1 > 表达式2} / ${表达式1 gt 表达式2}
小于等于            ${表达式1 <= 表达式2} / ${表达式1 le 表达式2}
大于等于            ${表达式1 >= 表达式2} / ${表达式1 ge 表达式2}
```