request ---> HttpServletRequest

session ---> HttpSession

application ---> ServletContext

EL表达式就是用来替换从域对象中取值的复杂编码。

${EL表达式}

${变量名} 变量名是将数据保存到域对象中时的key值

${变量名.属性名} 底层是在调用属性对应的getter方法

${变量名["属性名"]}

${变量名[0].属性名}

${变量名.key.属性名}

${变量名[“key”].属性名}

EL表达式默认的取值顺序，按照pageContext---》request---〉session---》application进行取值。

指定EL表达式从某个具体的域对象中查找

${pageScope.变量名}： 从pageContext中查找

${requestScope.变量名}：从request中查找

${sessionScope.变量名}：从session中查找

${applicationScope.变量名}：从application中查找

EL表达式进行简单的逻辑运算

${表达式1 逻辑运算符 表达式2}

EL表达式进行非空验证，${empty 变量名} 返回值为true/false，分别表示为空和不为空

变量如果为null，认为是空

集合size为0，认为是空

长度为0的字符串，认为是空

```
${not empty requestScope.str}
```

request的getAttribute(String name) 和 getParameter(String name)的区别：

服务端资源(Servlet,JSP)之间跳转，数据用getAttribute()获取。

客户端和服务端进行交互时，请求中的参数用getParameter()获取。

使用getAttribute()之前，必须有setAttribute()。

使用getParameter()可以直接取。

EL隐式对象

- 作用域访问对象（pageScope，requestScope，sessionScope，applicationScope）
- 参数访问对象（从HTTP请求中获取参数）${param.name}

```
${param.name}
${paramValues.name}
```

- 获取pageContext对象，获取JSP对应的Servlet的信息和JSP内置对象

```
${pageContext.servletConfig}<br/>
${pageContext.page}<br/>
${pageContext.servletContext}<br/>
${pageContext.servletConfig.servletName}<br/>
${pageContext.servletContext.getRealPath("")}<br/>
${pageContext.session}<br/>
${pageContext.request}<br/>
${pageContext.response}<b/>
```

JSTL

JSTL(JSP Standard Tag Library) JSP标准标签库

为什么要使用JSTL?

使用EL可以简化JSP页面访问数据的编码，但是功能不够完善，如果要进行遍历集合操作，EL表达式无法完成，就需要借助JSTL来完成集合的遍历，实际开发中，EL表达式和JSTL结合起来使用，JSTL负责处理逻辑，EL负责展示。

JSTL的优点：

- 提供了一组标准标签。
- 可用于编写各种动态功能。

如何使用

1.导入jar包

2.在JSP页面导入标签库

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

3.在JSP页面中使用JSTL

JSTL常用标签库：

核心标签库：

通用标签：set out remove catch

set：向域对象中添加数据

```
<c:set var="key" value="value" />
```

默认是将数据保存到page作用域中，pageContext中，相当于pageContext.setAttribute("key",value)。

如果需要指定作用域，通过scope属性来完成。

scope = "page" 默认，表示将数据保存到page作用域，pageContext对象中。

scope = "reqeust"， 表示将数据保存到request作用域，request对象中。

scope = "session"，表示将数据保存到session作用域，session对象中。

scope = "application"，表示将数据保存到application作用域，application对象中。

设置对象的属性值

```
<c:set target="${user}" property="name" value="张三" ></c:set>
```

out:输出域对象中的数据，于EL表达式功能类似

```
<c:out value="${mess}" default="未定义"></c:out>
```

remove:删除域对象中的数据

```
<c:remove var="mess"></c:remove>
```

catch：捕获异常

```
<c:catch var="error">
    <%
        int[] array = {1,2,3};
        int num = array[3];
    %>
</c:catch>
${error}
```

条件标签：if choose when otherwise

```
<c:set var="num1" value="1" scope="request"></c:set>
<c:set var="num2" value="2" scope="request"></c:set>
<c:if test="${num1 > num2}">${num1}</c:if>
<c:if test="${num1 < num2}">${num2}</c:if>
```

```
<c:choose>
    <c:when test="${num1 > num2}">${num1}</c:when>
    <c:when test="${num1 < num2}">${num2}</c:when>
</c:choose>
类似于Java中switch-case的写法/多个if条件
```

```
<c:choose>
    <c:when test="${num1 > num2}">${num1}</c:when>
    <c:otherwise>${num2}</c:otherwise>
</c:choose>
类似于Java中的if-else
```

迭代标签：forEach

```
<table border="1">
        <tr>
            <th>用户编号</th>
            <th>用户姓名</th>
        </tr>
        <c:forEach items="${list}" var="user">
            <tr>
                <td>${user.id}</td>
                <td>${user.name}</td>
            </tr>
        </c:forEach>
    </table>
```