javaweb 学习总结(二十九)——EL 表达式

一、EL 表达式简介

EL 全名为 Expression Language。EL 主要作用：

1、获取数据

EL 表达式主要用于替换 JSP 页面中的脚本表达式，以从各种类型的 web 域 中检索 java 对象、获取数据。(某个 web 域 中的对象，访问 javabean 的属性、访问 list 集合、访问 map 集合、访问数组)

2、执行运算

利用 EL 表达式可以在 JSP 页面中执行一些基本的关系运算、逻辑运算和算术运算，以在 JSP 页面中完成一些简单的逻辑运算。${user==null}

3、获取 web 开发常用对象

EL 表达式定义了一些隐式对象，利用这些隐式对象，web 开发人员可以很轻松获得对 web 常用对象的引用，从而获得这些对象中的数据。

4、调用 Java 方法

EL 表达式允许用户开发自定义 EL 函数，以在 JSP 页面中通过 EL 表达式调用 Java 类的方法。

1.1、获取数据

使用 EL 表达式获取数据语法："**${标识符}**"

EL 表达式语句在执行时，会调用 pageContext.findAttribute 方法，用标识符为关键字，分别从 page、request、session、application 四个域中查找相应的对象，找到则返回相应对象，找不到则返回""（注意，不是 null，而是空字符串）。

EL 表达式可以很轻松获取 JavaBean 的属性，或获取数组、Collection、Map 类型集合的数据

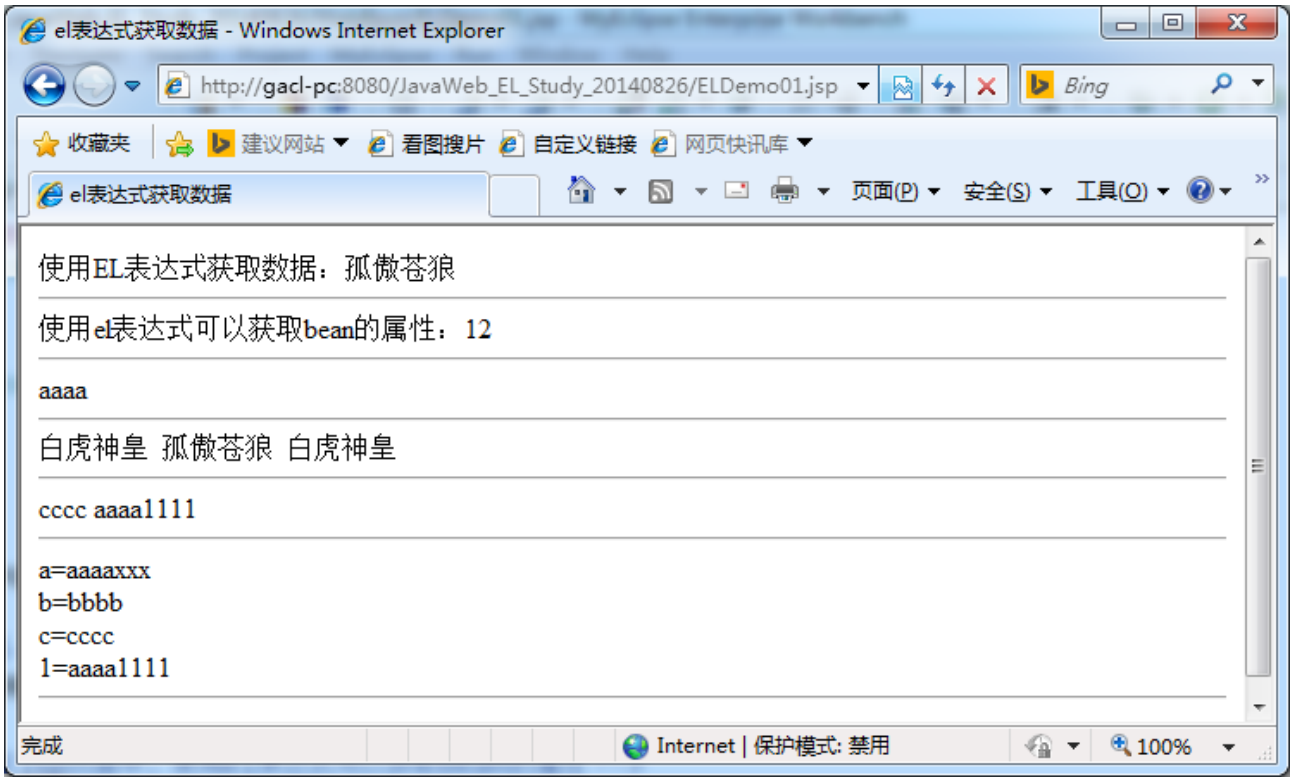
el 表达式获取数据范例：

```
1  <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2  <%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
3  <%@page import="me.gacl.domain.Person"%>
4  <%@page import="me.gacl.domain.Address"%>
5  <!DOCTYPE HTML>
6  <html>
7    <head>
8      <title>el 表达式获取数据</title>
9    </head>
10
11   <body>
12     <%
13         request.setAttribute("name","孤傲苍狼");
14     %>
15         <%--${name}等同于 pageContext.findAttribute("name") --%>
16         使用 EL 表达式获取数据：${name}
```

```
17      <hr>
18      <!-- 在 jsp 页面中，使用 el 表达式可以获取 bean 的属性 -->
19      <%
20          Person p = new Person();
21          p.setAge(12);
22          request.setAttribute("person",p);
23      %>
24          使用 el 表达式可以获取 bean 的属性：${person.age}
25      <hr>
26      <!-- 在 jsp 页面中，使用 el 表达式可以获取 bean 中的。。。。。。。。。的属性
--->
27      <%
28          Person person = new Person();
29          Address address = new Address();
30          person.setAddress(address);
31
32          request.setAttribute("person",person);
33      %>
34         ${person.address.name}
35       <hr>
36      <!-- 在 jsp 页面中，使用 el 表达式获取 list 集合中指定位置的数据 -->
37      <%
38          Person p1 = new Person();
39          p1.setName("孤傲苍狼");
40
41          Person p2 = new Person();
42          p2.setName("白虎神皇");
43
44          List<Person> list = new ArrayList<Person>();
45          list.add(p1);
46          list.add(p2);
47
48          request.setAttribute("list",list);
49      %>
50
51      <!-- 取 list 指定位置的数据 -->
52      ${list[1].name}
53
54      <!-- 迭代 List 集合 -->
55      <c:forEach var="person" items="${list}">
56          ${person.name}
57      </c:forEach>
58      <hr>
59      <!-- 在 jsp 页面中，使用 el 表达式获取 map 集合的数据 -->
60      <%
```

```
61          Map<String,String> map = new LinkedHashMap<String,String>();
62          map.put("a","aaaaxxx");
63          map.put("b","bbbb");
64          map.put("c","cccc");
65          map.put("1","aaaa1111");
66          request.setAttribute("map",map);
67      %>
68
69      <!-- 根据关键字取 map 集合的数据 -->
70        ${map.c}
71        ${map["1"]}
72        <hr>
73        <!-- 迭代 Map 集合 -->
74        <c:forEach var="me" items="${map}">
75            ${me.key}=${me.value}<br/>
76        </c:forEach>
77      <hr>
78    </body>
79 </html>
```

运行效果如下：



1.2、执行运算

语法：${运算表达式}，EL 表达式支持如下运算符：

1、关系运算符

关系运算符	说　　明	范　　例	结　果
= = 或 eq	等于	${ 5 = = 5 } 或 ${ 5 eq 5 }	true
!= 或 ne	不等于	${ 5 != 5 } 或 ${ 5 ne 5 }	false
< 或 lt	小于	${ 3 < 5 }或 ${ 3 lt 5 }	true
> 或 gt	大于	${ 3 > 5 }或 ${ 3 gt 5 }	false
<= 或 le	小于等于	${ 3 <= 5 }或 ${ 3 le 5 }	true
>= 或 ge	大于等于	${ 3 >= 5 }或 ${ 3 ge 5 }	false

2、逻辑运算符：

逻辑运算符	说　明	范　　例	结　果
&& 或 and	交集	${ A && B } 或 ${ A and B }	true / false
|| 或 or	并集	${ A || B } 或 ${ A or B }	true / false
! 或 not	非	${ !A } 或 ${ not A }	true / false

3、empty 运算符：检查对象是否为 null(空)

4、二元表达式：${user!=null?user.name :""}

5、[ ] 和 . 号运算符

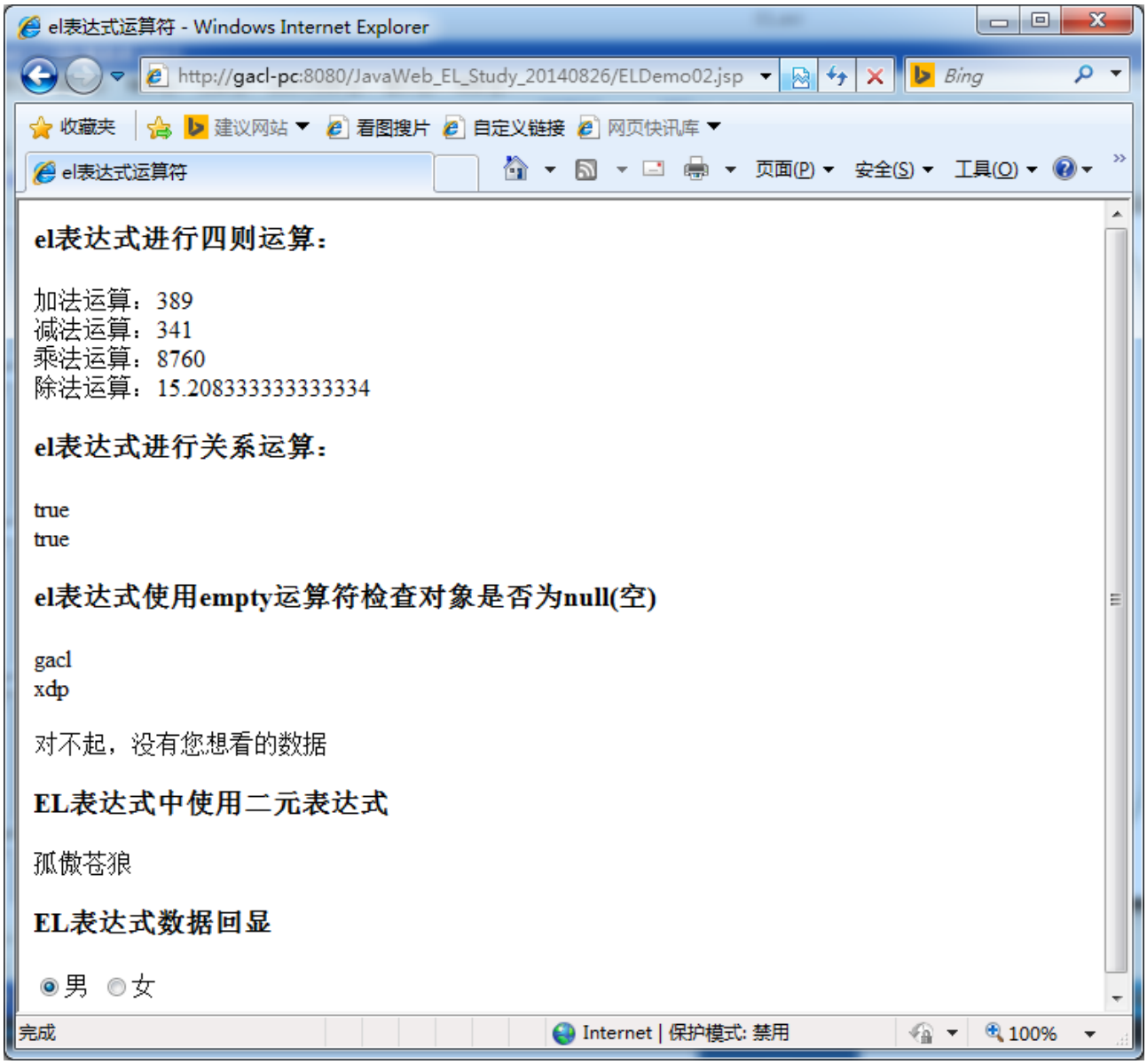
使用 EL 表达式执行运算范例：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@page import="me.gacl.domain.User"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el 表达式运算符</title>
  </head>

  <body>
  <h3>el 表达式进行四则运算：</h3>
        加法运算：${365+24}<br/>
        减法运算：${365-24}<br/>
        乘法运算：${365*24}<br/>
        除法运算：${365/24}<br/>

    <h3>el 表达式进行关系运算：</h3>
```

```
18      <%--${user == null}和 ${user eq null}两种写法等价--%>
19          ${user == null}<br/>
20          ${user eq null}<br/>
21
22          <h3>el 表达式使用 empty 运算符检查对象是否为 null(空)</h3>
23      <%
24
25          List<String> list = new ArrayList<String>();
26          list.add("gacl");
27          list.add("xdp");
28          request.setAttribute("list",list);
29      %>
30      <%--使用 empty 运算符检查对象是否为 null(空) --%>
31      <c:if test="${!empty(list)}">
32          <c:forEach var="str" items="${list}">
33              ${str}<br/>
34          </c:forEach>
35      </c:if>
36      <br/>
37      <%
38          List<String> emptyList = null;
39      %>
40      <%--使用 empty 运算符检查对象是否为 null(空) --%>
41      <c:if test="${empty(emptyList)}">
42          对不起，没有您想看的数据
43      </c:if>
44
45      <br/>
46
47      <h3>EL 表达式中使用二元表达式</h3>
48      <%
49          session.setAttribute("user",new User("孤傲苍狼"));
50      %>
51      ${user==null? "对不起，您没有登陆 " : user.username}
52
53      <br/>
54
55      <h3>EL 表达式数据回显</h3>
56      <%
57          User user = new User();
58          user.setGender("male");
59          //数据回显
60          request.setAttribute("user",user);
61      %>
```

```
62          <input type="radio" name="gender" value="male"
${user.gender=='male'?'checked':''}>男
63          <input type="radio" name="gender" value="female"
${user.gender=='female'?'checked':''}>女
64      <br/>65    </body>
66 </html>
```

运行结果如下：



1.3、获得 web 开发常用对象

EL 表达式语言中定义了 11 个隐含对象，使用这些隐含对象可以很方便地获取 web 开发中的一些常见对象，并读取这些对象的数据。

语法：**${隐式对象名称}**：获得对象的引用

序号	隐含对象名称	描　　述
1	pageContext	对应于 JSP 页面中的 pageContext 对象（注意：取的是 pageContext 对象
2	pageScope	代表 page 域中用于保存属性的 Map 对象
3	requestScope	代表 request 域中用于保存属性的 Map 对象
4	sessionScope	代表 session 域中用于保存属性的 Map 对象
5	applicationScope	代表 application 域中用于保存属性的 Map 对象
6	param	表示一个保存了所有请求参数的 Map 对象
7	paramValues	表示一个保存了所有请求参数的 Map 对象，它对于某个请求参数，返回的
8	header	表示一个保存了所有 http 请求头字段的 Map 对象，注意：如果头里面有 header[“Accept-Encoding”]
9	headerValues	表示一个保存了所有 http 请求头字段的 Map 对象，它对于某个请求参数，有“-” ，例 Accept-Encoding，则要 headerValues[“Accept-Encoding
10	cookie	表示一个保存了所有 cookie 的 Map 对象
11	initParam	表示一个保存了所有 web 应用初始化参数的 map 对象

测试 EL 表达式中的 11 个隐式对象：

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<!DOCTYPE HTML>
<html>
  <head>
    <title>el 隐式对象</title>
  </head>

  <body>
   <br/>----------------1、pageContext 对象：获取 JSP 页面中的 pageContext 对象-
----------------------<br/>
        ${pageContext}
    <br/>----------------2、pageScope 对象：从 page 域(pageScope)中查找数据----
--------------------<br/>
    <%
        pageContext.setAttribute("name","孤傲苍狼");  //map
    %>
        ${pageScope.name}
```

```
16          <br/>---------------3、requestScope 对象：从 request 域(requestScope)中
获取数据----------------------<br/>
17          <%
18              request.setAttribute("name","白虎神皇");  //map
19          %>
20              ${requestScope.name}
21          <br/>---------------4、sessionScope 对象：从 session 域(sessionScope)中
获取数据----------------------<br/>
22          <%
23              session.setAttribute("user","xdp");  //map
24          %>
25              ${sessionScope.user}
26          <br/>---------------5、applicationScope 对象：从 application 域
(applicationScope)中获取数据----------------------<br/>
27          <%
28              application.setAttribute("user","gacl");  //map
29          %>
30              ${applicationScope.user}
31          <br/>--------------6、param 对象：获得用于保存请求参数 map，并从 map 中
获取数据----------------------<br/>
32          <!--
http://localhost:8080/JavaWeb_EL_Study_20140826/ELDemo03.jsp?name=aaa  -->
33              ${param.name}
34          <!-- 此表达式会经常用在数据回显上 -->
35          <form
action="${pageContext.request.contextPath}/servlet/RegisterServlet"
method="post">
36              <input type="text" name="username" value="${param.username}">
37              <input type="submit" value="注册">
38          </form>
39          <br/>--------------7、paramValues 对象：paramValues 获得请求参数
//map{"",String[]}----------------------<br/>
40          <!--
http://localhost:8080/JavaWeb_EL_Study_20140826/ELDemo03.jsp?like=aaa&like=bbb -
->
41              ${paramValues.like[0]}
42              ${paramValues.like[1]}
43          <br/>--------------8、header 对象：header 获得请求头----------------
-----<br/>
44              ${header.Accept}<br/>
45              <%--${header.Accept-Encoding} 这样写会报错
46                  测试 headerValues 时，如果头里面有“-” ，例 Accept-Encoding，
则要 headerValues[“Accept-Encoding”]
47              --%>
48              ${header["Accept-Encoding"]}
```

```
49        <br/>--------------9、headerValues 对象：headerValues 获得请求头的值----------------------<br/>
50            <%--headerValues 表示一个保存了所有 http 请求头字段的 Map 对象，它对于某个请求参数，返回的是一个 string[]数组
51            例如：headerValues.Accept 返回的是一个 string[]数组 ，headerValues.Accept[0]取出数组中的第一个值
52            --%>
53            ${headerValues.Accept[0]}<br/>
54            <%--${headerValues.Accept-Encoding} 这样写会报错
55                测试 headerValues 时，如果头里面有“-” ，例 Accept-Encoding，则要 headerValues[“Accept-Encoding”]
56                headerValues["Accept-Encoding"]返回的是一个 string[]数组，headerValues["Accept-Encoding"][0]取出数组中的第一个值
57            --%>
58            ${headerValues["Accept-Encoding"][0]}
59        <br/>--------------10、cookie 对象：cookie 对象获取客户机提交的 cookie----------------------<br/>
60        <!-- 从 cookie 隐式对象中根据名称获取到的是 cookie 对象,要想获取值，还需要.value -->
61            ${cookie.JSESSIONID.value}  //保存所有 cookie 的 map
62        <br/>--------------11、initParam 对象：initParam 对象获取在 web.xml 文件中配置的初始化参数-------------------------<br/>
63        <%--
64         <!-- web.xml 文件中配置初始化参数 -->
65       <context-param>
66           <param-name>xxx</param-name>
67           <param-value>yyyy</param-value>
68       </context-param>
69       <context-param>
70           <param-name>root</param-name>
71           <param-value>/JavaWeb_EL_Study_20140826</param-value>
72       </context-param>
73         --%>
74        <%--获取 servletContext 中用于保存初始化参数的 map --%>
75            ${initParam.xxx}<br/>
76            ${initParam.root}
77   </body>
78 </html>
```

RegisterServlet 的代码如下：

```
1 package me.gacl.web.controller;
2
3 import java.io.IOException;
```

```
 4 import javax.servlet.ServletException;
 5 import javax.servlet.http.HttpServlet;
 6 import javax.servlet.http.HttpServletRequest;
 7 import javax.servlet.http.HttpServletResponse;
 8
 9 public class RegisterServlet extends HttpServlet {
10     /*
11      * 处理用户注册的方法
12      */
13     public void doGet(HttpServletRequest request, HttpServletResponse
response)
14             throws ServletException, IOException {
15         //1、接收参数
16         String userName = request.getParameter("username");
17         /**
18          * 2、直接跳转回/ELDemo03.jsp 页面，没有使用
request.setAttribute("userName", userName)将 userName 存储到 request 对象中
19          * 但是在 ELDemo03.jsp 页面中可以使用${param.username}获取到 request 对
象中的 username 参数的值
20          */
21         request.getRequestDispatcher("/ELDemo03.jsp").forward(request,
response);
22     }
23
24     public void doPost(HttpServletRequest request, HttpServletResponse
response)
25             throws ServletException, IOException {
26         doGet(request, response);
27     }
28 }
```

测试结果如下：

注意：

测试 header 和 headerValues 时，如果头里面有"-"，例 Accept-Encoding，则要 header["Accept-Encoding"]、headerValues["Accept-Encoding"]

测试 cookie 时，例${cookie.key}取的是 cookie 对象，如访问 cookie 的名称和值，须 ${cookie.key.name}或${cookie.key.value}

1.4、使用 EL 调用 Java 方法

EL 表达式语法允许开发人员开发自定义函数，以调用 Java 类的方法。语法：${prefix：method(params)}

在 EL 表达式中调用的只能是 Java 类的静态方法，这个 Java 类的静态方法需要在 TLD 文件中描述，才可以被 EL 表达式调用。

EL 自定义函数用于扩展 EL 表达式的功能，可以让 EL 表达式完成普通 Java 程序代码所能完成的功能。

1.5、EL Function 开发步骤

一般来说， EL 自定义函数开发与应用包括以下三个步骤：

1、编写一个 Java 类的静态方法

2、编写标签库描述符（tld）文件，在 tld 文件中描述自定义函数。

3、在 JSP 页面中导入和使用自定义函数

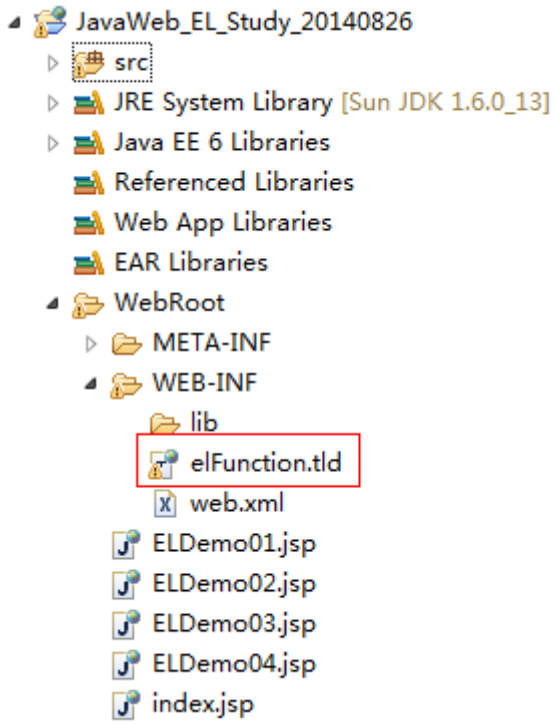
示例：开发对 html 标签进行转义的 el function

1、编写 html 转义处理工具类，工具类中添加对 html 标签进行转义的静态处理方法，如下：

```
package me.gacl.util;

/**
* @ClassName: HtmlFilter
* @Description: html 转义处理工具类
* @author: 孤傲苍狼
* @date: 2014-8-27 上午 12:09:15
*
*/
public class HtmlFilter {

    /**
    * @Method: filter
    * @Description: 静态方法，html 标签转义处理
    * @Anthor:孤傲苍狼
    *
    * @param message 要转义的内容
    * @return 转义后的内容
    */
    public static String filter(String message) {

        if (message == null)
            return (null);

        char content[] = new char[message.length()];
        message.getChars(0, message.length(), content, 0);
        StringBuffer result = new StringBuffer(content.length + 50);
        for (int i = 0; i < content.length; i++) {
            switch (content[i]) {
            case '<':
                result.append("&lt;");
                break;
            case '>':
                result.append("&gt;");
                break;
            case '&':
                result.append("&amp;");
                break;
            case '"':
                result.append("&quot;");
                break;
            default:
```

```
43                  result.append(content[i]);
44              }
45          }
46          return (result.toString());
47      }
48  }
```

2、在 WEB-INF 目录下编写标签库描述符（tld）文件，在 tld 文件中描述自定义函数



elFunction.tld 的代码如下：

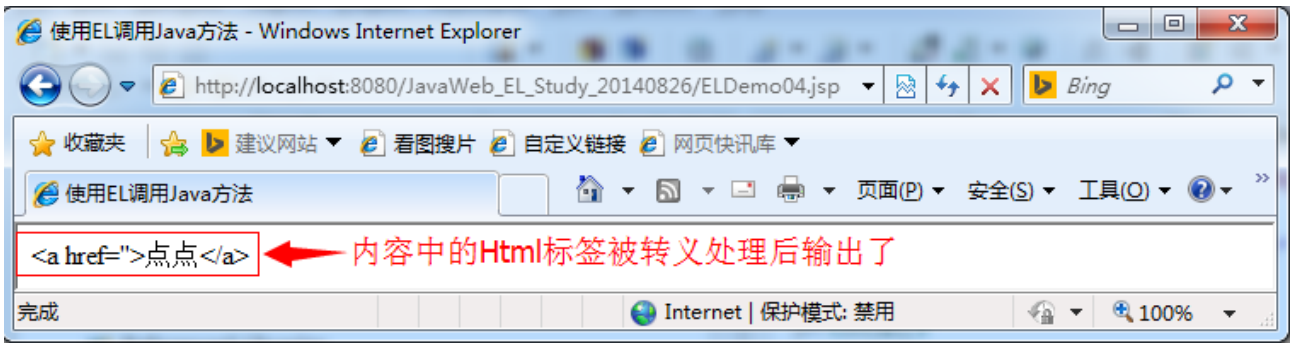
```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd">
4   <tlib-version>1.0</tlib-version>
5   <short-name>EL Function</short-name>
6   <!--
7       自定义 EL 函数库的引用 URI，
8       在 JSP 页面中可以这样引用：<%@taglib uri="/ELFunction" prefix="fn" %>
9   -->
10  <uri>/ELFunction</uri>
11
12  <!--<function>元素用于描述一个 EL 自定义函数 -->
13   <function>
14         <description>html 标签转义处理方法</description>
```

```
15          <!--<name>子元素用于指定 EL 自定义函数的名称-->
16          <name>filter</name>
17          <!--<function-class>子元素用于指定完整的 Java 类名-->
18          <function-class>me.gacl.util.HtmlFilter</function-class>
19          <!--<function-signature>子元素用于指定 Java 类中的静态方法的签名，
20              方法签名必须指明方法的返回值类型及各个参数的类型，各个参数之间用
逗号分隔。-->
21          <function-signature>java.lang.String
filter(java.lang.String)</function-signature>
22      </function>
23
24 </taglib>
```

3、在 JSP 页面中导入和使用自定义函数

```
1 <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2 <%--引入 EL 自定义函数库 --%>
3 <%@taglib uri="/ELFunction" prefix="fn" %>
4 <!DOCTYPE HTML>
5 <html>
6   <head>
7     <title>使用 EL 调用 Java 方法</title>
8   </head>
9
10  <body>
11      <%--使用 EL 调用 filter 方法--%>
12      ${fn:filter("<a href=''>点点</a>")}
13  </body>
14 </html>
```

运行结果如下：



1.6、开发 EL Function 注意事项

编写完标签库描述文件后，需要将它放置到<web 应用>\WEB-INF 目录中或 WEB-INF 目录下的除了 classes 和 lib 目录之外的任意子目录中。

TLD 文件中的<uri> 元素用指定该 TLD 文件的 URI，在 JSP 文件中需要通过这个 URI 来引入该标签库描述文件。

<function>元素用于描述一个 EL 自定义函数，其中：

<name>子元素用于指定 EL 自定义函数的名称。

<function-class>子元素用于指定完整的 Java 类名，

<function-signature>子元素用于指定 Java 类中的静态方法的签名，方法签名必须指明方法的返回值类型及各个参数的类型，各个参数之间用逗号分隔。

1.7、EL 注意事项

EL 表达式是 JSP 2.0 规范中的一门技术 。因此，若想正确解析 EL 表达式，需使用支持 Servlet2.4/JSP2.0 技术的 WEB 服务器。

注意：有些 Tomcat 服务器如不能使用 EL 表达式

（1）升级成 tomcat6

（2）在 JSP 中加入<%@ page isELIgnored="false" %>

1.8、EL 表达式保留关键字

And	eq	gt	true
Or	ne	le	false
No	lt	ge	null
instanceof	empty	div	mod