EL 表达式详解

一、EL 表达式介绍

- Expression Language 表达式语言
- 是一种在 JSP 页面获取数据的简单方式(只能获取数据，不能设置数据)
- 在 JSP2.0 开始引入概念

语法格式

在 JSP 页面的任何静态部分均可通过：${expression}来获取到指定表达式的值

二、EL 获取数据(从四大域中获取属性)

- EL 只能从四大域中获取属性

1、如果没有使用 EL 的内置对象，则查找数据顺序是依次按照由小到大范围从四大域中查找指定名称的属性值

```
- pageContext<request<session<application

       <%@ page language="java" contentType="text/html; charset=UTF-8"
           pageEncoding="UTF-8"%>
       <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
       <html>
       <head>
       <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
       <title>Insert title here</title>
```

```
</head>
<body>
    <%
        pageContext.setAttribute("name", "linjie");
        request.setAttribute("name", "lucy");
        session.setAttribute("name", "king");
        application.setAttribute("name", "bilibili");
    %>
    name=${name }
</body>
</html>
```

```
17
18
19
20
```

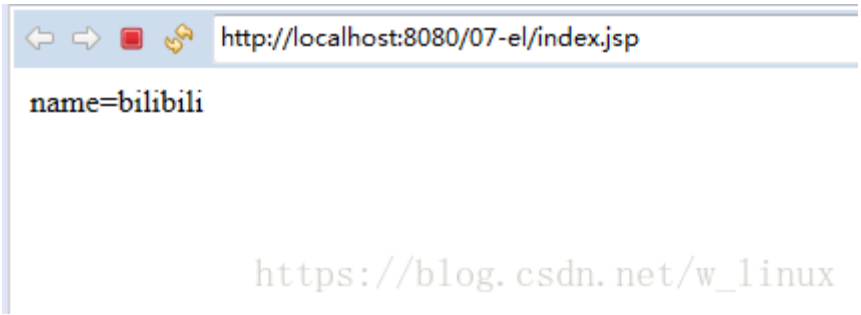
可以看出没有使用 EL 内置对象时查找顺序是由小到大，所以最先显示的是 pageContext 域中的

2、使用 EL 内置对象，从指定域中获取数据，提高了查找效率

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        pageContext.setAttribute("name", "linjie");
        request.setAttribute("name", "lucy");
        session.setAttribute("name", "king");
        application.setAttribute("name", "bilibili");
    %>
    name=${applicationScope.name }
</body>
</html>
```

```
1
```

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```



可以看出，使用 applicationScope 即可指定 application 域中的 name 输出，当然其他域也是类似，下文会说这四大域属性相关的内置对象

三、EL 中的内置对象

EL 有 11 个内置对象，这里主要讲域属性相关的 4 个和其他 4 个

EL 的 11 个内置对象，除了 pageContext 以外，其他 10 个内置对象的类型都是 java.util.Map 类型

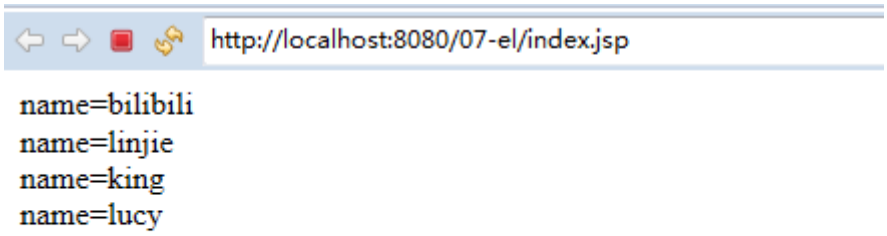
1、域属性相关(4 个)

1. pageScope：从 page 范围域属性空间中查找指定的 key
2. requestScope：从 request 范围域属性空间中查找指定的 key
3. sessionScope：从 session 范围域属性空间中查找指定的 key
4. applicationScope：从 application 范围域属性空间中查找指定的 key

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        pageContext.setAttribute("name", "linjie");
        request.setAttribute("name", "lucy");
        session.setAttribute("name", "king");
        application.setAttribute("name", "bilibili");
    %>

    name=${applicationScope.name }<br>
```

```
22.    name=${pageScope.name }<br>
23.    name=${sessionScope.name }<br>
24.    name=${requestScope.name }<br>
25.</body>
26.</html>
```

```
21
22
```



2、其他重要内置对象(4 个)

1、pageContext

该 pageContext 与 JSP 内置对象 pageContext 是同一个对象。通过该对象，可以获取到 request、response、session、servletContext、servletConfig 等对象**注意：**这些对象在 EL 里不是内置对象，这些对象只能通过 pageContext 获取

- 在 EL 中直接${pageContext.request}即可获取 request 对象，其底层调用的是 pageContext.getRequest()方法。同理，也可以通过类似方法获取其他对象

重点：其中最常用的：${pageContext.request.contextPath },代表 web 应用下的根，可以看出下面 action 中的路径可读性更强了

Regster.java

```
package linjie.com;

import java.io.IOException;
```

```
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Regster extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
        doGet(request, response);
    }

}
```

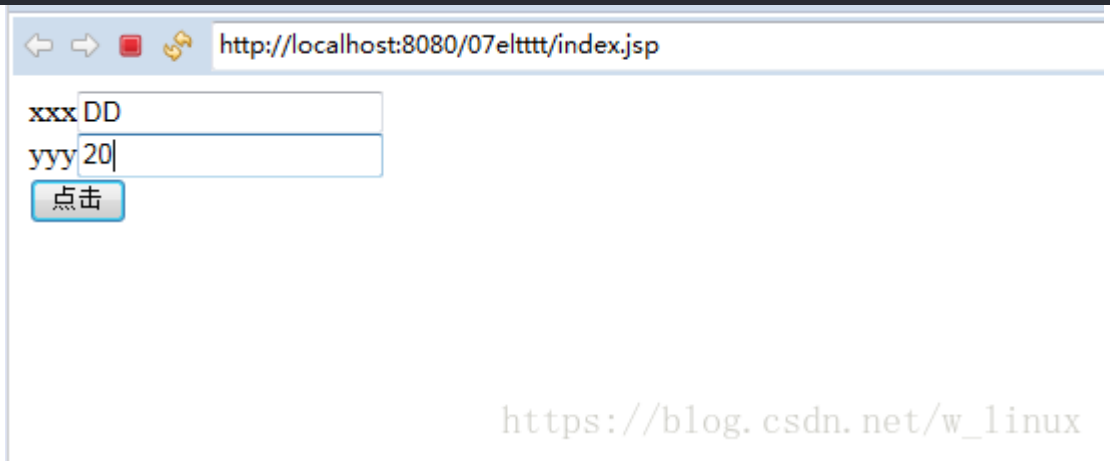
```
14
15
16
17
18
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%-- ${pageContext.request.contextPath }代表web应用的根 --%>
    <form action="${pageContext.request.contextPath }/regster" method="POST">
        xxx<input type="text" name="name"/><br>
        yyy<input type="text" name="age"/><br>
        <input type="submit" value="点击">
    </form>
</body>
</html>
```

```
1
2
3
4
5
6
7
```

```
8
9
10
11
12
13
14
15
16
17
```



2、param（获取请求中的指定参数）

- 其底层实际调用 request.getParameter()

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%-- ${pageContext.request.contextPath }代表web应用的根 --%>
    <form action="${pageContext.request.contextPath }/show.jsp" method="POST">
        xxx<input type="text" name="name"/><br>
        yyy<input type="text" name="age"/><br>
        <input type="submit" value="点击">
    </form>
</body>
</html>
```

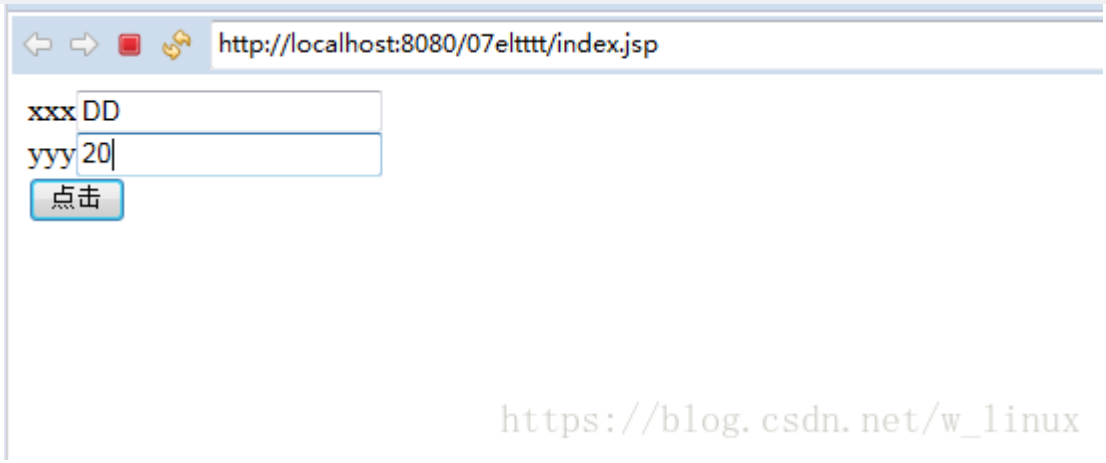
```
17
```

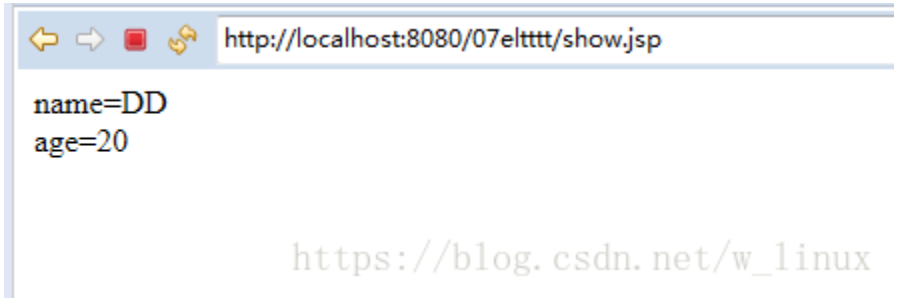
show.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    name=${param.name }<br>
    age=${param.age }<br>
</body>
</html>
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
```

客户浏览器访问结果





3、paramValues

获取请求中的指定参数的所以值，其底层实际调用 request.getParameterValues()

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<%-- ${pageContext.request.contextPath }代表 web 应用的根 --%>
    <form action="${pageContext.request.contextPath }/show.jsp" method="POST">
```

```
        xxx<input type="text" name="name"/><br>
        yyy<input type="text" name="age"/><br>

        爱好:
        <input type="checkbox" name="hobby" value="sleep">睡觉
        <input type="checkbox" name="hobby" value="play">玩
        <input type="checkbox" name="hobby" value="eat">吃
        <input type="submit" value="点击">
    </form>
</body>
</html>
```

```
17
18
19
20
21
22
```

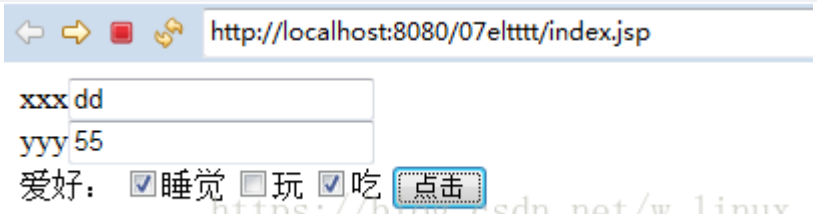
show.jsp

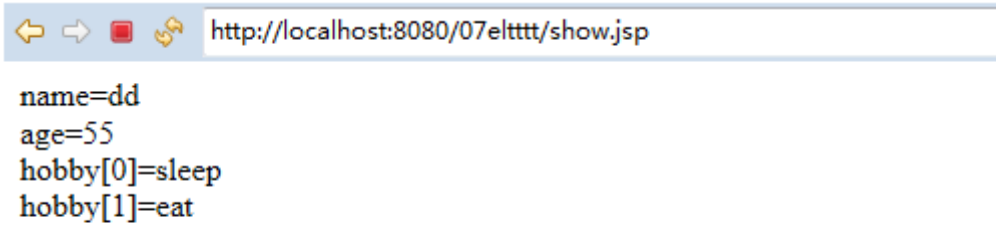
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    name=${param.name }<br>
    age=${param.age }<br>


    hobby[0]=${paramValues.hobby[0] }<br>
    hobby[1]=${paramValues.hobby[1] }<br>
</body>
</html>
```

```
1
2
3
4
5
6
```

```
7
8
9
10
11
12
13
14
15
16
17
```

客户浏览器显示结果





4、initParam

获取初始化参数，其底层调用的是 ServletContext.getInitParameter()

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>07eltttt</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

<!--初始化参数 -->
  <context-param>
    <param-name>name</param-name>
    <param-value>林杰</param-value>
  </context-param>


  <servlet>
    <display-name>Regster</display-name>
    <servlet-name>Regster</servlet-name>
    <servlet-class>linjie.com.Regster</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Regster</servlet-name>
    <url-pattern>/regster</url-pattern>
  </servlet-mapping>
</web-app>
```

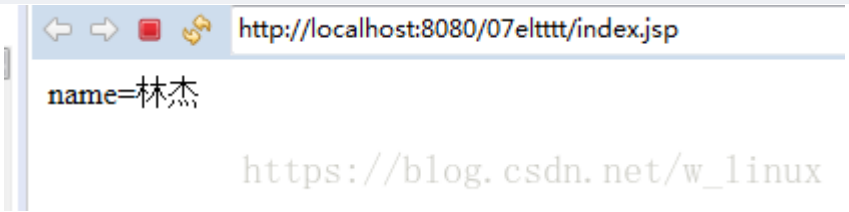
```
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    name=${initParam.name }
</body>
</html>
```

客户浏览器显示结果



四、EL 访问 Bean 的属性

1、什么是 java Bean

JavaBean 是公共 Java 类，但是为了编辑工具识别

需要满足至少三个条件

- 有一个 public 默认构造器（例如无参构造器）
- 属性使用 public 的 get，set 方法访问，也就是说设置成 private 同时 get，set 方法与属性名的大小也需要对应。例如属性 name，get 方法就要写成，public String getName(){},N 大写。
- 需要序列化。这个是框架，工具跨平台反映状态必须的

2、EL 访问 Bean 属性

EL 可以通过${key.属性}的方式获取到指定值，其底层实际调用的是该对象的相应属性的 get 方法

Demo.java

```
package linjie.com;
/*
 *Bean
 */
public class Demo {
    private String name;
    private int age;
    public Demo(String name,int age){
        this.name=name;
        this.age=age;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    @Override
    public String toString() {
        return super.toString();
    }

}
```

```
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    import="linjie.com.Demo"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        Demo test=new Demo("linjie",12);
        request.setAttribute("ELttt", test);
```

```
    %>
    name=${requestScope.ELttt.name }<br>
    age=${requestScope.ELttt.age }<br>

    <!-- 若访问为 null 的对象的属性，EL 是不会抛出空指针异常的，只是不显示而已 -->
    names=${requestScope.ELtttxx.name }<br>

</body>
</html>
```

```
18
19
20
21
22
```

客户浏览器显示结果

五、EL 访问数组中的数据

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        String[] names={"xlj","lucy","king"};
        pageContext.setAttribute("names", names);
    %>
    name[1]=${names[1] }<br>

    <!-- 若访问的数组元素下标超出了数组下标上限,EL 不会抛出越界异常，只是不显示 -->
    names[5]=${names[5] }<br>
```

```
</body>
</html>
```

下面是访问类的数组

Stu.java

```
package linjie.com;
/*
 *Bean
 */
public class Stu {
    private String sname;
    private String address;
    public Stu() {
        super();
    }

    public Stu(String sname, String address) {
        super();
        this.sname = sname;
        this.address = address;
    }

    public String getSname() {
        return sname;
    }

    public void setSname(String sname) {
        this.sname = sname;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return super.toString();
    }


}
```

```
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    import="linjie.com.*"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Insert title here</title>
</head>
<body>

    <%
        Stu[] stus=new Stu[3];
        stus[0]=new Stu("xlj","A");
        stus[1]=new Stu("lucy","B");
        stus[2]=new Stu("kingA","C");
        pageContext.setAttribute("stus",stus);
    %>
    stus[1].Sname=${stus[1].sname }
</body>
</html>
```

```
16
17
18
19
20
21
```

客户浏览器显示结果

六、EL 获取 list 中数据

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        List<String> names=new ArrayList<String>();
```

```
        names.add("xlj");
        names.add("lucy");
        pageContext.setAttribute("names", names);
    %>

    <!-- 因为 List 底层是数组，所以可以这样写 -->
    names[1]=${names[1] }<br>
</body>
</html>
```

```
18
19
20
21
```

客户浏览器显示结果

注意：

EL 可以通过索引访问 List，但无法访问 Set。因为 Set 中没有索引概念

七、EL 访问 Map

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        Map<String,Object> map=new HashMap<String,Object>();
        map.put("age", 20);
        map.put("name", "xlj");
```

```
        pageContext.setAttribute("map", map);
    %>
    name=${map.name }<br>
    age=${map.age }<br>
</body>
</html>
```

客户浏览器显示结果

八、EL 中的运算符(empty)

1、先说说几个常用运算符

- 算术运算符：+、-、*、/、%(不支持++、--)
- 关系运算符：==、!=、>、>=、<、<=
- 逻辑运算符：!、&&、||、not、and、or
- 条件运算符：?:
- 取值运算符：[]、点号

2、empty 运算符

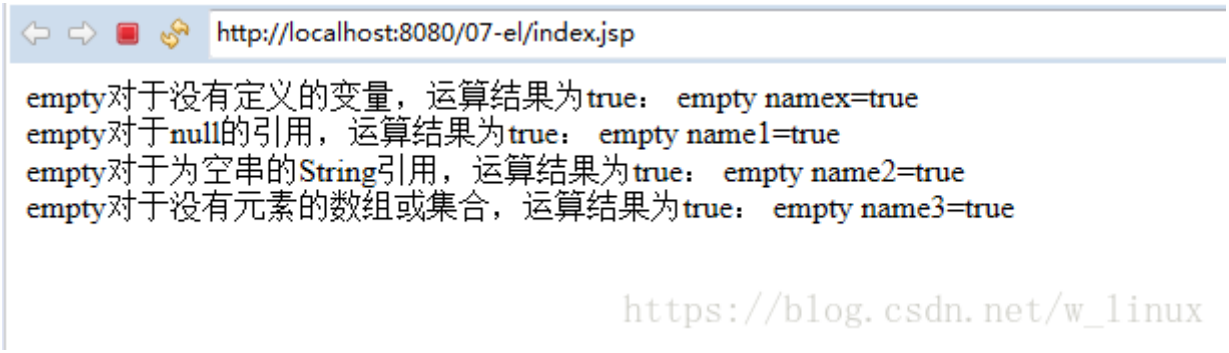
用法为${empty 变量}，结果为布尔值

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <%
        String name1=null;
        String name2="";
        List<String> name3=new ArrayList<String>();

        pageContext.setAttribute("name1", name1);
        pageContext.setAttribute("name2", name2);
        pageContext.setAttribute("name3", name3);
    %>
    empty 对于没有定义的变量，运算结果为 true：
    empty namex=${empty namex }<br>

    empty 对于 null 的引用，运算结果为 true：
    empty name1=${empty name1 }<br>

    empty 对于为空串的 String 引用，运算结果为 true：
    empty name2=${empty name2 }<br>

    empty 对于没有元素的数组或集合，运算结果为 true：
    empty name3=${empty name3 }<br>
</body>
</html>
```

```
30
31
32
```

客户浏览器显示结果



九、自定义 EL 函数

因为 EL 本身不具有处理字符串能力，所以可以自定义 EL 函数

- 定义函数(新建 MyEL.java 类)

- 注册：先找到 jsp2-example-taglib.tld，将头部以及注册函数复制到自己创建的.tld 文件中(.tld 放在 WEB-INF 下)

- 在 index.jsp 中使用，使用时需要<%@ taglib uri=" http://tomcat.apache.org/jsp2-example-taglib" prefix=" MyEL" %>

1、定义函数 MyEL.java

```
package linjie.com;



//自定义函数
//该类及其函数,需要在扩展名为.tld 的 xml 文件中注册
//tld: tag library definition（标签库定义）
//xml 文件是需要约束的,即需要配置文件头部。这个头部约束可以从一下文件中进行复制
```

```
//在 Tomcat 安装目录下：webapps\examples\WEB-INF\jsp2
//文件为：jsp2-example-taglib.tld

//这个.tld 的 xml 文件，需要定义在当前 web 项目的 WEB-INF 目录下，在此目录下创建以.tld 结尾的
xml 文件
//将 jsp2-example-taglib.tld 中头部复制到创建的 xml 文件中

//再将函数注册，还是在 jsp2-example-taglib.tld 底部中复制
public class MyEL {
    private static MyEL instance;
    public static MyEL getInstance() {
        if(instance==null)
        {
            instance=new MyEL();
        }
        return instance;
    }

    //字符串小写变大写
    public static String LowerToUpper(String str) {
        return str.toUpperCase();
    }
}
```

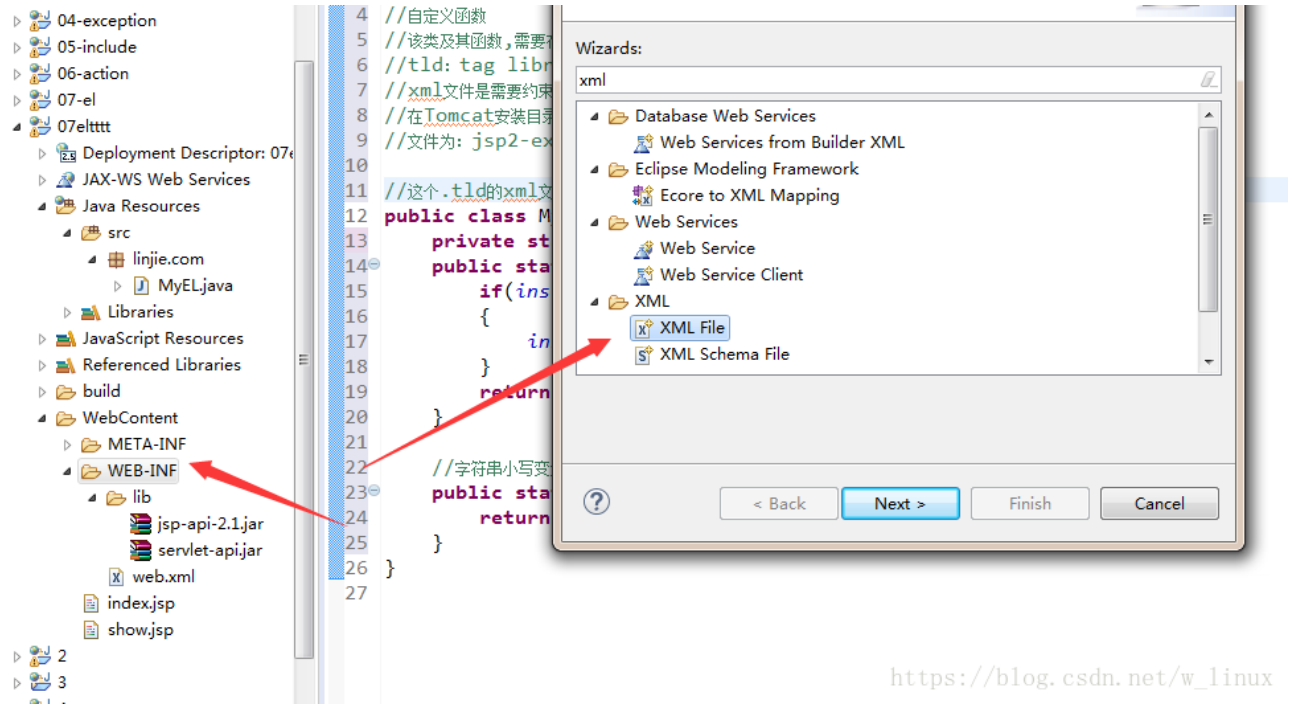
```
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

2、将 jsp2-example-taglib.tld 中头部部分以及底部的注册函数部分复制到自己创建的 tld(在 WEB-INF 下)文件中

```
 limitations under the License.
-->

<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
    version="2.0">
    <description>A tag library exercising SimpleTag handlers.</description>
    <tlib-version>1.0</tlib-version>
    <short-name>SimpleTagLibrary</short-name>
    <uri>http://tomcat.apache.org/jsp2-example-taglib</uri>
```



```
    <function>
        <description>Converts the string to all caps</description>
        <name>caps</name>
        <function-class>jsp2.examples.el.Functions</function-class>
        <function-signature>java.lang.String caps( java.lang.String )</function-signature>
    </function>
</taglib>
```


```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xs
    version="2.0">

    <!-- 定义标签库信息 -->
    <description>A tag library exercising SimpleTag handlers.</description>
    <tlib-version>1.0</tlib-version>
    <short-name>MyEL</short-name><!-- 标签库名称，一般定义成和文件名一样 -->
    <uri>http://tomcat.apache.org/jsp2-example-taglib</uri>

   <!--  注册函数 -->
      <function>
        <name>MyLowerToUpper</name>
        <function-class>linjie.com.MyEL</function-class><!-- 方法得类 -->
        <function-signature>java.lang.String LowerToUpper( java.lang.String )</function-signature><!-- 方法签名：需
    </function>
</taglib>
```

MyEL.tld

```
<?xml version="1.0" encoding="UTF-8"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
    version="2.0">

    <!-- 定义标签库信息 -->
    <description>A tag library exercising SimpleTag handlers.</description>
    <tlib-version>1.0</tlib-version>
    <short-name>MyEL</short-name><!-- 标签库名称，一般定义成和文件名一样 -->
    <uri>http://tomcat.apache.org/jsp2-example-taglib</uri>

   <!--  注册函数 -->
     <function>
        <name>MyLowerToUpper</name>
        <function-class>linjie.com.MyEL</function-class><!-- 方法得类 -->
        <function-signature>java.lang.String LowerToUpper(
java.lang.String )</function-signature><!-- 方法签名 ：需要返回值以及方法名、参数-->
    </function>
</taglib>
```

```
11
12
13
14
15
16
17
18
19
```

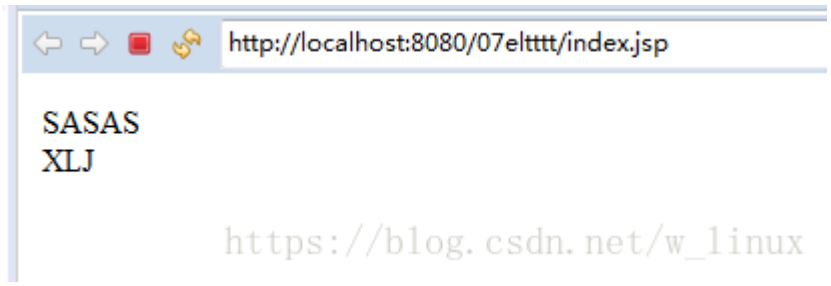
3、在 index.jsp 中使用，使用时需要<%@ taglib uri=" http://tomcat.apache.org/jsp2-example-taglib" prefix=" MyEL" %>

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://tomcat.apache.org/jsp2-example-taglib" prefix="MyEL" %><!--
tld 中的 uri 和 short-name -->
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
    <!-- 这个方法名是在 tld 注册时的 name -->
    ${MyEL:MyLowerToUpper("sasas") }<br>


    <!-- EL 函数只能处理四大域中的属性值及 String 常量 -->
    <%
```

```
        String name="xlj";
        pageContext.setAttribute("name", name);
    %>
    ${MyEL:MyLowerToUpper(name) }<br>

</body>
</html>
```

```
19
20
21
22
23
```

客户浏览器显示结果



十、EL 总结

- EL 表达式不能出现在 Java 代码块、表达式块等 JSP 动态代码部分
- EL 只能从四大域属性空间中获取数据(pageContext、request、session、application)
- EL 不会抛出空指针异常，只会不显示
- EL 不会抛出数组越界异常，只会不显示
- EL 不具有对字符串进行处理的能力(可以使用 JSTL 的 EL 或者自定义 EL 函数)