**表达式语言（EL）**

# 授课内容

授课科目：JavaEE

授课内容：表达式语言（EL）

授课类型：讲授+实践

授课时间：3+1学时

主讲教师：武永亮

# 教学目标要求

## 思想目标：

## 能力目标：

1. 培养学生通过EL表达式在页面中显示Java数据的能力；
2. 培养学生了解EL表达式的应用场景

## 知识目标：

1. 表达式语言简介；
2. 表达式语言运算符；
3. 表达式语言函数。

# 课件分析

概 述：首先介绍表达式语言的发展历程，说明在不同的作用域中都可通过表达式语言进行数据展示，然后通过实例对比使用JSP代码展示数据和通过EL表达式展示数据的差异，说明所有通过JSP代码进行的数据展示方式都可以通过EL表达式进行展示。接着详细介绍EL表达式的不同展示方式、EL的隐含对象，同时对比JSP的9个隐含对象、然后通过EL表达式获取变量数据。EL表达式不但能获取显示基本的变量数据，还能够通过表达式中的运算符进行基本的逻辑运算，而且还能通过定义tld文件自定义EL函数进行使用。

教学重点：EL表达式的基本用法。

教学难点：通过定义tld文件实现EL函数。

# 教学方法

问题教学法、案例法等。

# 教学过程

## 课前补充

（10分钟）

以提问的方式要求学生回顾JSP的基本用法:

1.如何通过JSP进行Java变量的声明及显示？

答：JSP中声明变量的语法：<%! 声明部分 %>

变量的输出表达式：<%=表达式%>

2.JSP页面中的内置对象及对象作用域有哪些？

答：JSP中的内置对象：

request对象 客户端请求，此请求会包含来自GET/POST请求的参数通过它才能了解到客户的需求，然后做出响应。

response对象 响应客户请求的有关信息。

session对象 它指的是客户端与服务器的一次会话，从客户端连到服务器的一个WebApplication开始，直到客户端与服务器断开连接为止。

out对象 它是JspWriter类的实例,是向客户端输出内容常用的对象。

page对象 它是指向当前JSP页面本身，有点像类中的this指针，它是Java.lang.Object类的实例。

application对象 它实现了用户间数据的共享，可存放全局变量。它开始于服务器的启动，直到服务器的关闭。

exception对象 它是一个例外对象，当一个页面在运行过程中发生了例外，就产生这个对象。

pageContext对象 它提供了对JSP页面内所有的对象及名字空间的访问。

config对象 它是在一个Servlet初始化时，JSP引擎向它传递信息用的。

JSP中的作用域：

page范围(pageContext)：当前页中有效，跳转即无效。

request范围：在同一个请求范围有效，客户端跳转无效。

session范围：跳转有效，新开浏览器无效。

application范围：所有用户有效，重启服务器无效。

## 上节回顾

（10分钟）

上节课的内容讲到JSP页面可以通过组件复用来实现模块可重用，其中复用组件最主要的方法有静态包含、动态包含。静态包含在JSP源文件转成对应的Servlet源文件时将当前JSP文件和其包含的所有其他文件合并成一个Servlet文件，使用的语法是<%@include file=“地址”%>。动态包含指的是客户端请求发送时，可以将请求转发给当前JSP文件包含的其他对象。主要实现形式是使用JSP的include动作指令和forward动作指令。include动作指令的作用是把当前页面的客户端请求转给另一个对象去处理，执行完毕后回到当前对象继续执行。forward动作指令是把当前页面的客户端请求完全交给另一个对象去处理，结果也不需要让当前页面知道。通过这两种方式可以引入多种页面对象相互协作完成当前页面的请求。

## 作业点评

（0分钟）

## 导入新课

（5分钟）

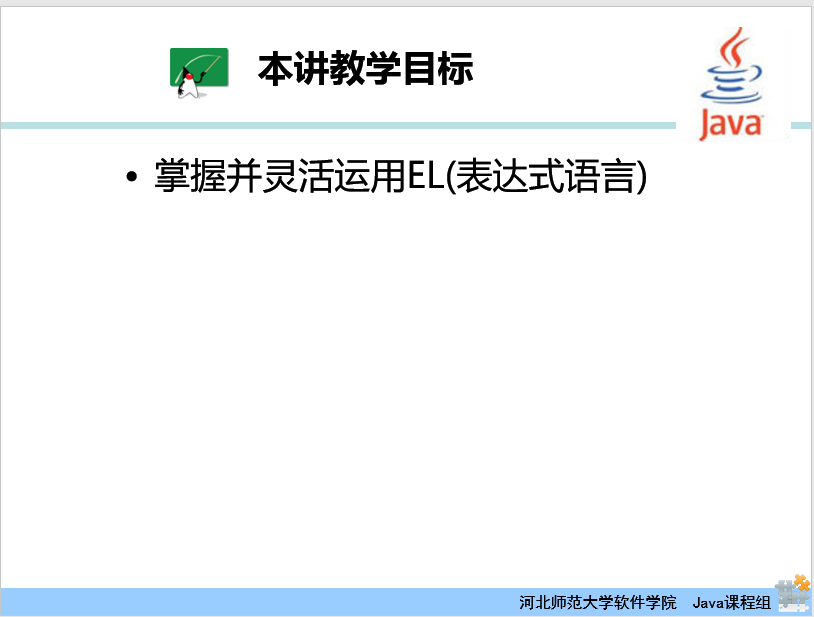
Web组件复用主要内容如下：

1.对于一个页面的响应结果可能是有多个对象共同作用来完成的

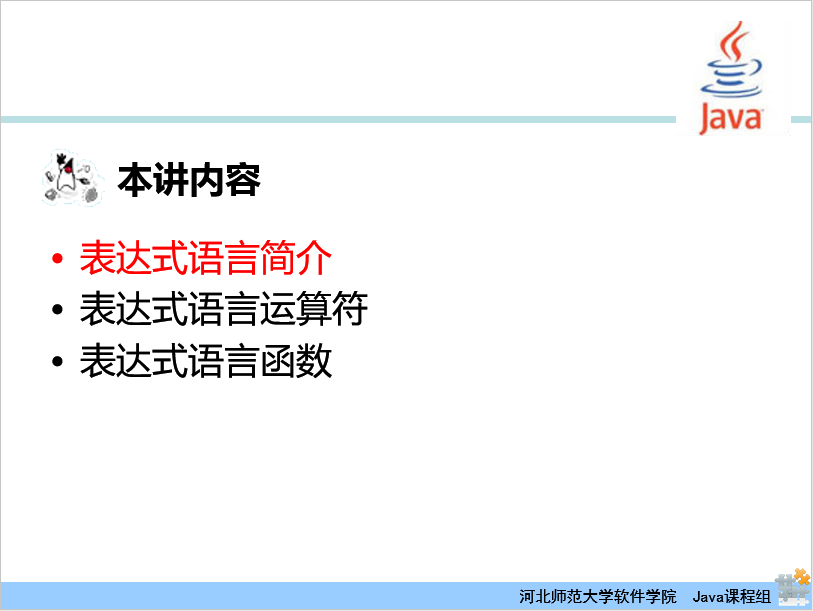
2.通过JSP可进行对象获取，对象逻辑操作。但是在JSP页面中，这些操作相对比较复杂，在一个JSP页面中既有HTML代码又有Java代码，不利于代码分层，JSP主要侧重页面数据展示，通过EL可以更加方便的实现展示效果。本节来讲解EL表达式的相关知识。

## 讲授新课

（120分钟）

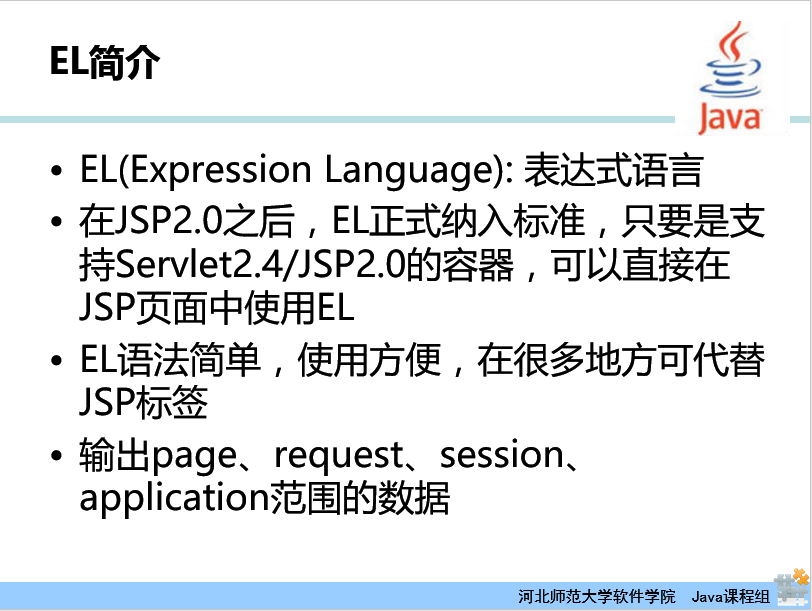


本章的教学目标：要求学生掌握表达式语言的用法，能够熟练应用EL表达式进行数据展示。把握EL表达式的应用场景，熟练掌握EL表达式的逻辑运算。



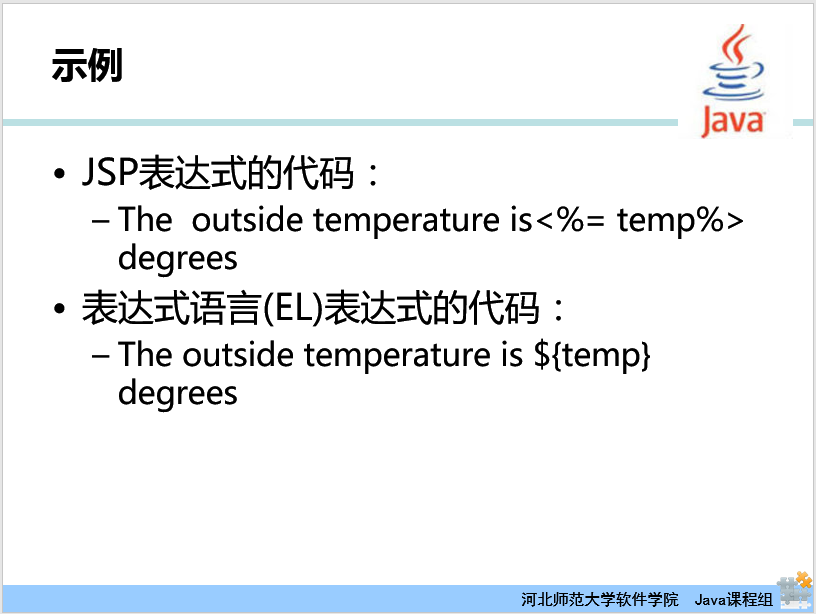
本节内容分为三个部分，第一部分介绍表达式语言的发展历程，第二部分介绍表达式语言的运算符及用法，第三部分简单介绍表达式语言的函数使用。

首先介绍第一部分内容—表达式语言简介。



EL是Expression Language的缩写，意思就是表达式语言，顾名思义就是一种基于表达式的语言，它是从JSP2.0之后被正式纳入标准的，这就意味着只要支持JSP2.0的容器，都支持表达式语言。现阶段常见的JSP容器都支持EL表达式。EL表达式的作用是通过简单的语法，在JSP中显示数据，它作用就是替代JSP脚本元素在JSP中进行数据显示。 EL表达式中显示的数据和JSP显示的数据作用域相同，都包括page作用域、request作用域、session作用域、application作用域。

接下来通过一个示例展示JSP如何显示Java中的变量数据，并且对比通过EL显示变量数据。



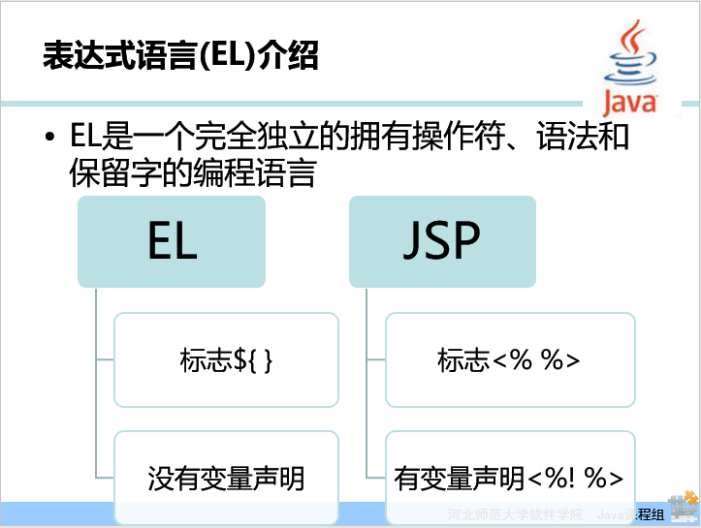
使用JSP表达式进行Java变量的显示，代码如下：

|  |
| --- |
| The outside temperature is <%= temp%> degrees |

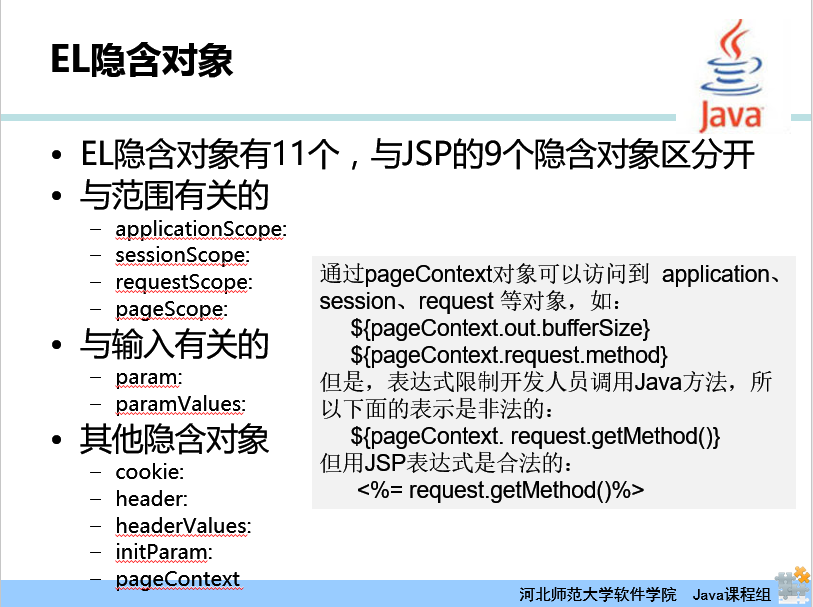
在这段JSP代码中通过<%= temp%> JSP标签显示了temp变量的内容。当temp变量值为30时，上述代码在页面显示The outside temperature is 30 degrees。这种方式可以显示temp变量的值，但是在JSP页面中插入了JSP标签，代码看起来比较混乱。通过EL表达式的方式显示temp变量，代码如下：

|  |
| --- |
| The outside temperature is ${temp} degrees |

这就是通过EL表达式来显示temp变量的源码。页面展示效果同上例效果，但是这样方式在页面中看不到JSP标签，使代码简单很多。其中通过${temp}来获取temp变量的值。这种方法就是EL表达式获取变量的方法。接下来详细介绍一下EL表达式的具体语法。



EL是一种表达式语言，它拥有完全独立的操作符、语法和关键字。在JSP中常用的标签是<% %>，其中包含Java的语法，对于变量声明使用<%! %>标签。EL表达式的用法是使用${ }显示变量的值，无法对变量进行声明操作。



JSP标签：

1.request对象 客户端请求，此请求会包含来自GET/POST请求的参数通过它，解到客户的需求，然后做出响应。

2.response对象 响应客户请求的有关信息。

3.session对象 它指的是客户端与服务器的一次会话，从客户端连到服务器的一个WebApplication开始，直到客户端与服务器断开连接为止。

4.out对象 它是JspWriter类的实例,是向客户端输出内容常用的对象。

5.page对象 它是指向当前JSP页面本身，有点象类中的this指针，它是 Java.lang.Object类的实例。

6.application对象 它实现了用户间数据的共享，可存放全局变量。它开始于服务器的启动，直到服务器的关闭。

7.exception对象 它是一个异常对象，当一个页面在运行过程中发生了例外，就产生这个对象。

8.pageContext对象 它提供了对JSP页面内所有的对象及名字空间的访问。

9.config对象 它是在一个Servlet初始化时，JSP引擎向它传递信息用的。

对比JSP的9个隐含对象，EL具有11隐含对象。这里的EL的隐含对象和JSP的隐含对象是不一样的，JSP的隐含对象有9个。EL的隐含对象主要分为与范围相关的隐含对象、与输入输出有关的隐含对象、其他隐含对象。与范围相关的隐含对象有：

例如，已有代码：

|  |
| --- |
| <%  pageContext.setAttribute("userName ","pageName");  request.setAttribute("userName", "xiaoming");  session.setAttribute("userName", "xiaoli");  application.setAttribute("userName", "xiaowang");  %> |

applicationScope:代表整个应用程序的作用域范围，例如：${ applicationScope.userName}代表在application作用域内查找userName变量。

sessionScope:代表在session的作用域范围，例如：${ sessionScope.userName}代表在session作用域内查找userName变量。

requestScope:代表在本次请求的作用域范围，例如：${requestScope.userName}代表在request作用域内查找userName变量。

pageScope:代表在当前页面的作用范围，例如：${ pageScope.userName}代表在page作用域内查找userName变量。

与输入相关的隐含对象有：

|  |
| --- |
| 假设请求地址是：<http://127.0.0.1:8080/demo/param.jsp?name>=zhangsan&age=10&age=20  <!--获取name-->  ${param.name}  <!--获取age数组-->  ${paramValues.age[0] }<!--得到10-->  ${paramValues.age[1] }<!--得到20--> |

param:表示一个保存了所有请求参数的Map对象。

paramValues:表示一个保存了所有请求参数的Map对象，对于某个请求参数，返回的是string[]。

对于其他隐含对象有：

header：表示一个保存了所有http请求头字段的Map对象。

headerValues: 表示一个保存了所有http请求头字段的Map对象，返回string[]数组。注意：如果头里面有“-” ，例Accept-Encoding，则要headerValues[“Accept-Encoding”]。

cookie:表示一个保存了所有cookie的Map对象。

initParam:表示一个保存了所有Web应用初始化参数的map对象。

pageContext:对应于JSP页面中的pageContext对象（注意：取的是pageContext对象。）

EL程序可以通过pageContext对象访问到JSP的隐含对象application、session、request等对象，可以说pageContext对象提供了一种对应EL和Java对象的方法，如：

${pageContext.out.bufferSize} 通过这种方式访问JSP的隐含对象out

${pageContext.request.method} 通过这种方式访问JSP的隐含对象request

但是要注意EL限制开发人员去调用Java中的方法，所以下面的表示是非法的：

${pageContext. request.getMethod()} 不可以在${ } 中调用getMethod方法。

但用在JSP表达式中是合法的：

<%= request.getMethod()%>



这里列出了我刚才给大家介绍过的11中EL隐含对象所代表的含义如下：

pageContext:对应于JSP页面中的pageContext对象

applicationScope:代表整个应用程序的作用域范围

sessionScope:代表在session的作用域范围

requestScope:代表在本次请求的作用域范围

pageScope:代表在当前页面的作用范围

Param:表示一个保存了所有请求参数的Map对象

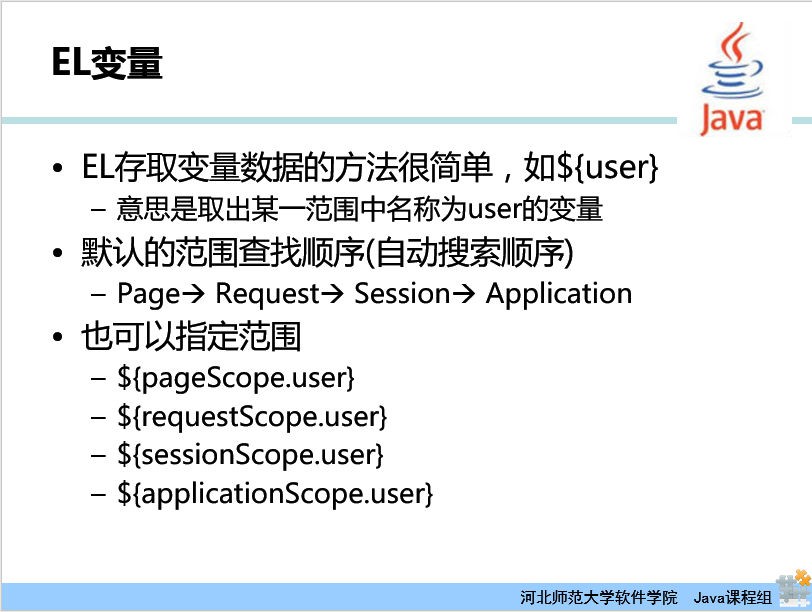
paramValues:表示一个保存了所有请求参数的Map对象，用数组方式存储

Header：表示一个保存了所有http请求头字段的Map对象

headerValues: 表示一个保存了所有http请求头字段的Map对象，用数组方式存储

cookie:表示一个保存了所有cookie的Map对象

initParam:表示一个保存了所有web应用初始化参数的map对象



通过EL可以展示变量，EL存取变量的语法非常简单，例如：使用${user}可以取出user变量，当然如果直接获取user变量，user的作用域是按照从小到大开始搜索的，默认查找顺序是从page作用域到request作用域，再到session作用域，最后是application作用域。当然确定user变量的作用域，也可以直接指定从某个范围取出user变量。例如：

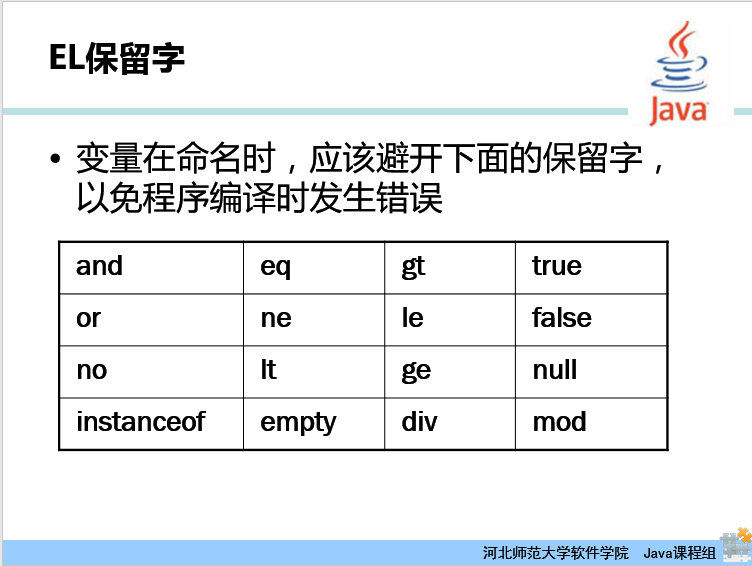
${pageScope.user}：取出page作用域的user变量

${requestScope.user}：取出request作用域的user变量

${sessionScope.user}：取出session作用域的user变量

${applicationScope.user}：取出application作用域的user变量

讲完这部分，我相信大家都可以通过EL来获取Java中的变量了。



需要注意一下，在变量命名时，应避开EL保留字（关键字），否则会出现编译错误。

and ：&&运算

eq ：==运算

gt ：>运算

true：真值

false：假值

or ： ||运算

ne ：!= 运算

ln ： <=运算

ge ： >=运算

no：不存在

lt ：<

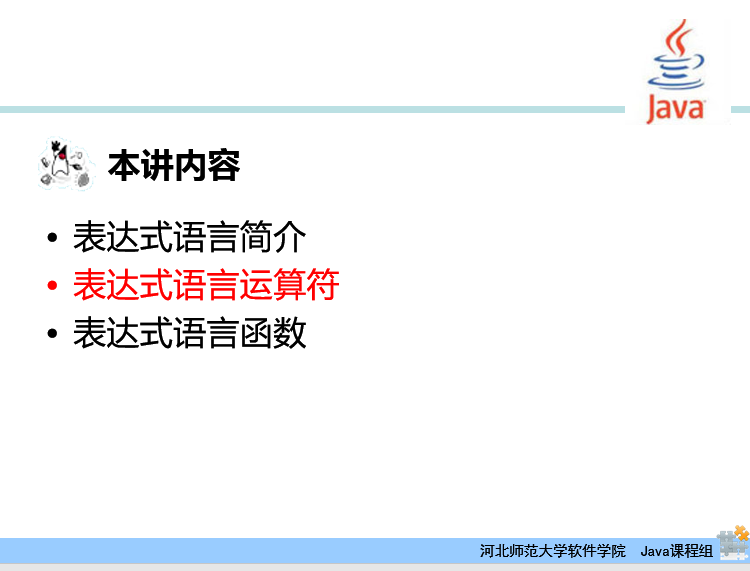
null：空值

instanceof ：判断对象

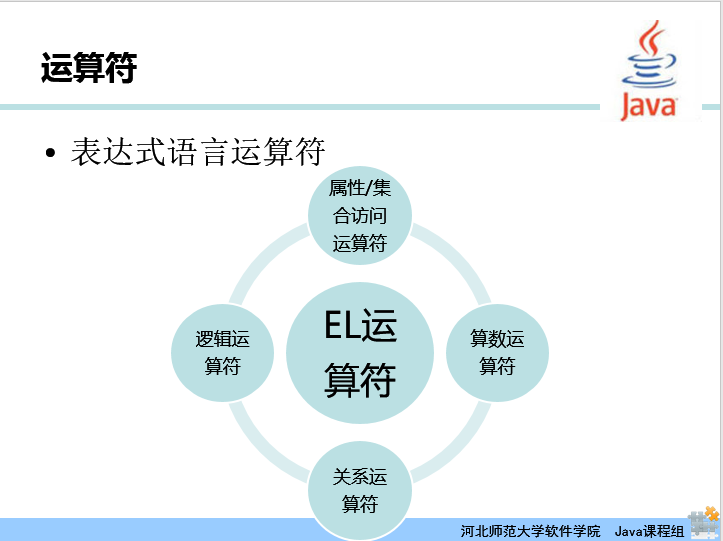
empty：判断是否为空

div ： /运算

mod ：%运算



第一部分主要介绍如何通过EL获取Java变量的值及注意情况。EL不单可以只获取变量值，还能对获取的变量值进行简单的运算，那就是通过表达式运算符来实现。下面我们开始第二部分内容的讲解。



对于EL的运算符主要分为四大类：

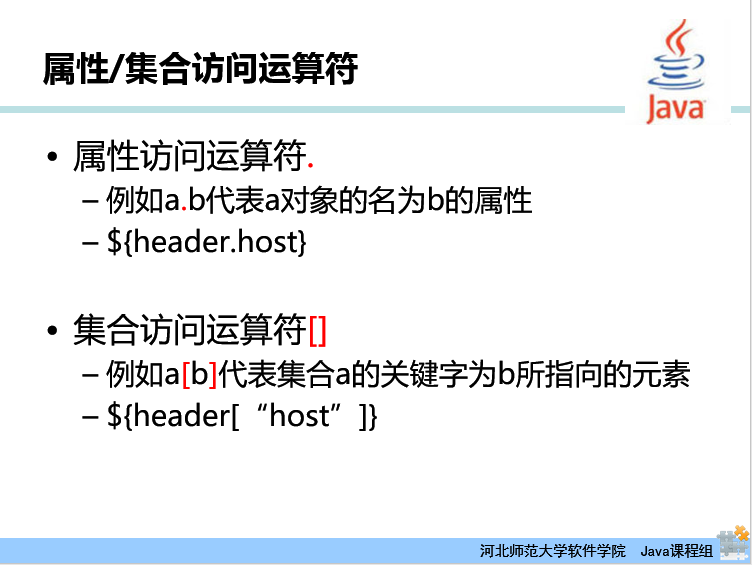
属性/集合访问运算符：主要作用是从Java的集合或者对象中获取集合元素或者对象某个属性；

算数运算符：主要作用是进行基本的算数运算；

关系运算符：主要作用是进行变量的关系运算；

逻辑运算符：主要作用是进行变量的一些基本逻辑运算。

接了下来我们首先看一下属性/集合运算符的使用。



属性访问运算符，主要作用：是访问一个对象的某个属性，也叫做.运算。例如：a.b代表a对象的名为b的属性，访问header对象的host属性可以使用${header.host}表达式。

集合范文运算符是[]，主要作用是：访问一个集合中的某个对象。例如：a[b]代表集合a的关键字为b所指向的元素，访问header集合中的关键字为host的值，表达式为${header[“host”]}。

属性访问运算符和集合运算符优势两者代表的意思是一样的，例如：${sessionScope.user.sex} 等价于${sessionScope.user[“sex”]}，都代表访问session作用域的user对象的sex属性。两种运算符也可混合使用，例如：${sessionScope.shoppingCart[0].price}代表session作用域的shoppingCart对象的第一个元素的价格。

但是需要注意的是当要存取的属性名称中包含一些特殊字符时（”.”或”-”等并非字母或数字的符号），一定要用[]，比如：${user.My-Name}应写为${user[“My-Name”]}。

还有如果想动态的获取值时，也要使用[]，例如：${user[data]}，data是一个变量时。

第二类EL运算符是算数运算符。



+ 运算，例如：${6+6} 显示12。注意：在EL表达式中的‘+’只有数学运算的功能，没有连接符的功能，它会试着把运算符两边的操作数转换为数值类型，进而进行数学加法运算，最后把结果输出。若出现${'a'+'b'}则会出现异常。

- 运算，例如：${4-3} 显示1。

\* 运算，例如：${4\*3} 显示12。

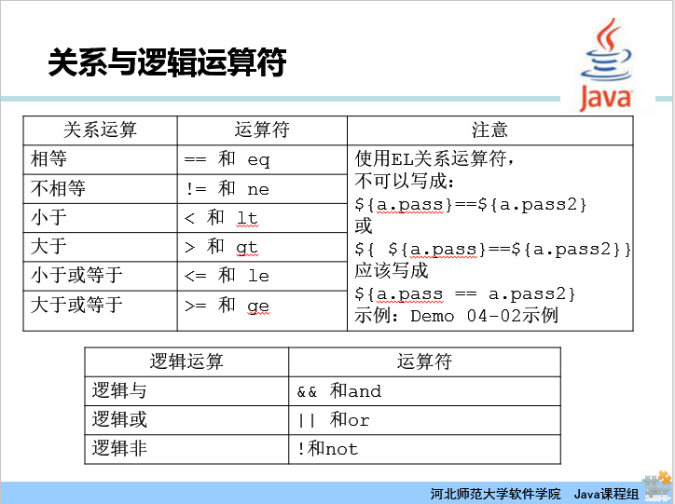
/ 运算，例如：${9/3}显示3。

% 运算，例如：${9%3}显示0。

Demo04-02 实现代码如下：

|  |
| --- |
| <h1>算术运算符示例</h1>  <ul>  <li>\${3 + 7}运算结果${3 + 7}</li>  <li>\${3 - 7}运算结果${3 - 7}</li>  <li>\${3 \* 7}运算结果${3 \* 7}</li>  <li>\${3 / 7}运算结果${3 / 7}</li>  <li>\${3 / 0}运算结果${3 / 0}</li>  <li>\${3 / 7}运算结果${3 / 7}</li>  <li>\${10 % 4}运算结果${10 % 4}</li>  </ul> |

接下来介绍关系运算符和逻辑运算符。



首先看一下关系运算符：

== 或者 eq， 例如：${4==4} 或者 ${4 eq 4 }

!= 或者 ne， 例如：${4!=3} 或者 ${4 ne 3 }

< 或者 lt， 例如：${4<9} 或者 ${4 lt 9 }

<= 或者 le， 例如：${9<=8} 或者 ${9 le 8 }

> 或者 gt， 例如：${8>9} 或者 ${8 gt 9 }

>= 或者 ge， 例如：${45>=9} 或者 ${45 ge 9 }

需要注意：EL关系运算符，不可以写成：${a.pass}==${a.pass2}或${ ${a.pass}==${a.pass2}}，而应该写成${a.pass == a.pass2}。这样才代表判断a变量的pass属性和pass2的属性值是否相等。

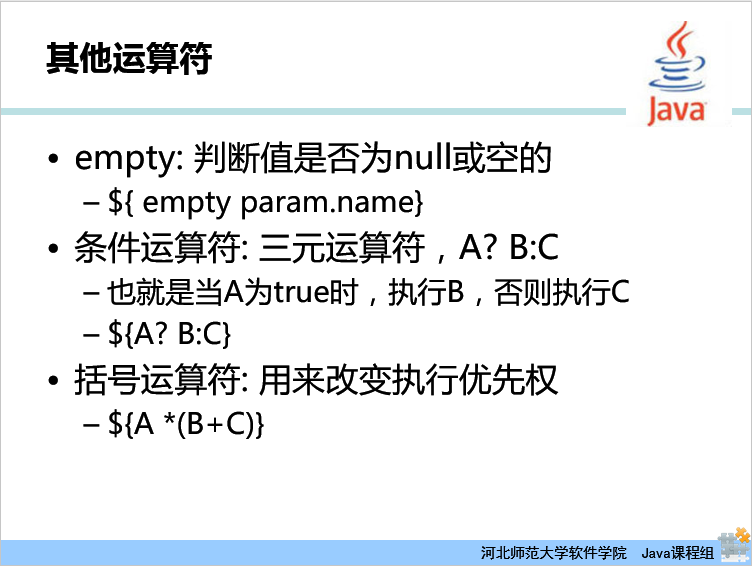
|  |
| --- |
| <h1>关系运算符</h1>  <ul>  <li>\${1 &lt;6}运算结果是:${1<6}</li>  <li>\${1 &gt;6}运算结果是:${1>6}</li>  <li>\${1 &lt;2}运算结果是:${1<2}</li>  <li>\${1 &gt;2}运算结果是:${1>2}</li>  <li>\${'a' &gt;'z'}运算结果是:${'a'>'z'}</li>  <li>\${'a' &lt;'z'}运算结果是:${'a'<'z'}</li>  </ul> |

接下来介绍逻辑运算符：

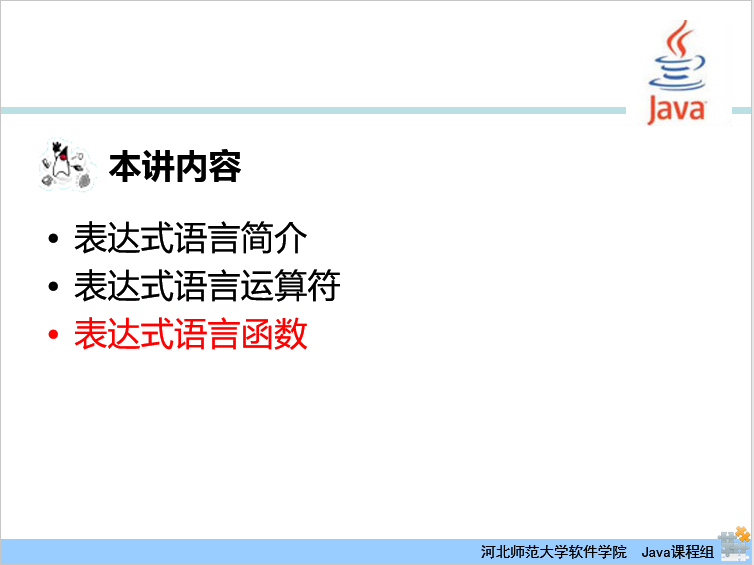
&& 或者 and， 例如：${false && false} 或者 ${false and false }

|| 或者 or， 例如：${true || false} 或者 ${true or false }

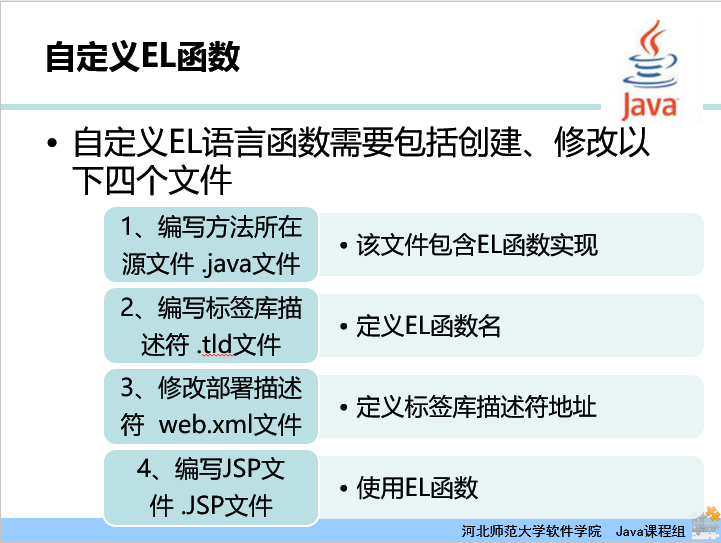
! 或者 not，例如：${!true}（相当于${false}） 或者 ${not true }



在EL中还有一些其他的特殊运算符，empty 判断EL表达式中的表达式是否为空，例如：${empty sessionScope.user} 是用来判断session作用域中的user对象是否为空。条件运算符? :， A? B:C就是当A为true时，执行B，否则执行C，例如：${3>2?'是':'不是'} 。还有括号运算符: 用来改变运算符的优先级。${A \*(B+C)} 先计算B+C的值，然后再乘以A。



当然如果在JSP页面显示的过程中，如果有一些公用的逻辑运算集合，也可以把固定的逻辑运算封装成表达式函数。表达式语言也具有函数的功能。



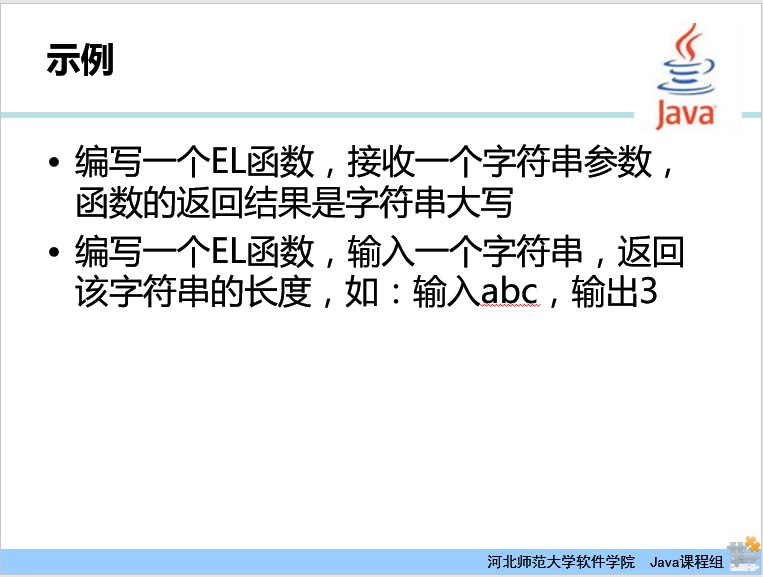
自定义EL函数的步骤是：

1.定义方法类对应的源文件Java文件，该文件包含了JSP页面中使用的Java方法。

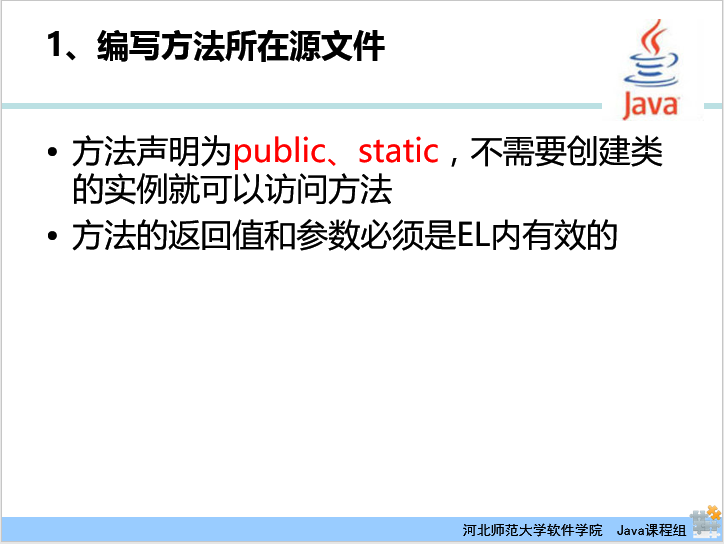
2.定义标签库描述符.tld文件，该文件定义Java方法对应的函数名称。

3.定义部署描述符web.xml文件，该问指定标签库描述符的地址。

4.编写.jsp文件，在JSP文件中调用标签库的函数名来调用Java文件中的方法。



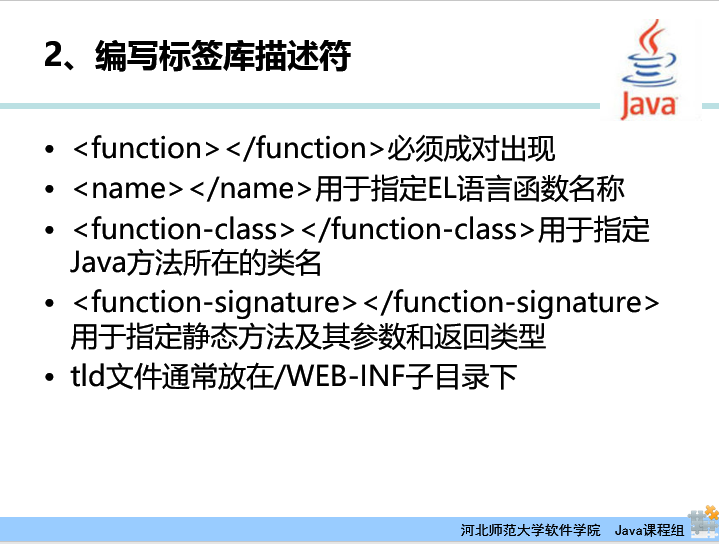
接下来完成一个示例：编写一个EL函数，接收一个字符串参数，函数的返回结果是字符串大写；编写一个EL函数，输入一个字符串，返回该字符串的长度，如：输入abc，输出3。



第一步 创建被EL调用的Java方法。

|  |
| --- |
| public class StrMethods {  public static String upper(String x) {  return x.toUpperCase();  }  public static int length(String x) {  return x.length();  }  } |

注意：方法声明为public、static，不需要创建类的实例就可以访问方法。方法的返回值和参数必须是EL内有效的。



第二步 创建标签库描述符

|  |
| --- |
| <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/j2eeweb-jsptaglibrary\_2\_0.xsd"  version="2.0">  <tlib-version>1.0</tlib-version>  <function>  <name>upper</name>  <function-class>com.StrMethods</function-class>  <function-signature>java.lang.String upper(java.lang.String)  </function-signature>  </function>  <function>  <name>length</name>  <function-class>com.StrMethods</function-class>  <function-signature>java.lang.int length(java.lang.String)  </function-signature>  </function>  </taglib> |

注意：

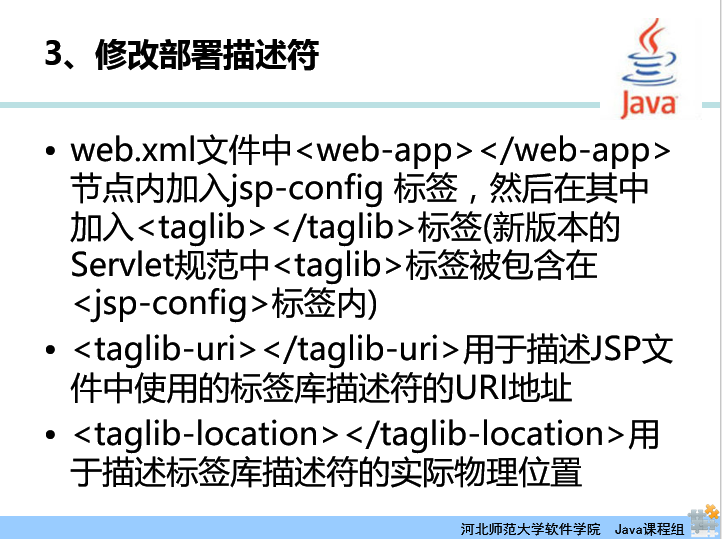
<function></function>必须成对出现

<name></name>用于指定EL语言函数名称

<function-class></function-class>用于指定Java方法所在的类名

<function-signature></function-signature>用于指定静态方法及其参数和返回类型

tld文件通常放在/WEB-INF子目录下



第三步 修改部署描述符

|  |
| --- |
| <jsp-config>  <taglib>  <!-- 配置标签的引用地址 JSP页面中引用时使用-->  <taglib-uri>/Functions</taglib-uri>  <!-- 配置标签的TLD文件地址 -->  <taglib-location>  /WEB-INF/Functions.tld  </taglib-location>  </taglib>  </jsp-config> |

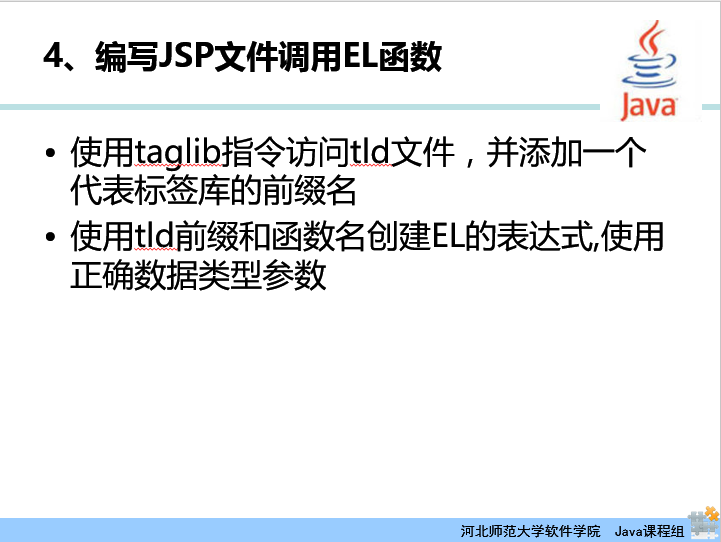
注意：

web.xml文件中<web-app></web-app>节点内加入jsp-config 标签，然后在其中加入<taglib></taglib>标签(新版本的Servlet规范中<taglib>标签被包含在<jsp-config>标签内)

<taglib-uri></taglib-uri>用于描述JSP文件中使用的标签库描述符的URI地址

<taglib-location></taglib-location>用于描述标签库描述符的实际物理位置

第四步 在JSP中访问定义的函数



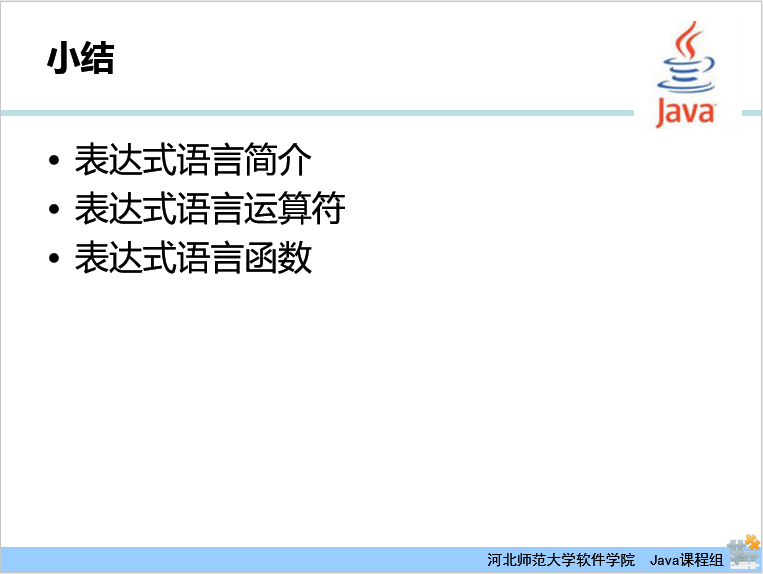
|  |
| --- |
| <%@ taglib uri="/Functions" prefix="el" %>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>EL 示例\_使用Param隐含对象接收数据</title>  </head>  <body bgcolor="#FFFFFF">  <div align="center">  <h1>欢迎您，${el:upper(param.userName)}</h1>  <h1>您的密码是 ${param.password}</h1>  </div> |

注意：使用taglib指令访问tld文件，并添加一个代表标签库的前缀名；使用tld前缀和函数名创建EL的表达式,使用正确数据类型参数。

EL函数部分的内容就讲解完毕，接下来进行总结。

## 课时小结

（10分钟）



首先介绍了EL表达式的发展历程及主要作用，然后通过一系列的例子，通过QTEL表达式获取变量获取，变量的基本运算，算数运算，逻辑运算，最后讲解了如何定义语言函数

表达式语言简介

表达式语言运算符

表达式语言函数

# 布置作业

（10分钟）

编写一个EL函数，接收两个int型参数，函数的返回结果是两参数的和

编写一个EL函数，输入一个字符串，返回该字符串的逆序字符排列，如：输入abc，输出cba