# 第一章 Java Web 开发技术概述

### B/S(Brower/Server)和 C/S(Client/Server)的区别

(1)概念：C/S是客户端和服务器，B/S是浏览器和服务器。  
(2)使用：C/S在客户端要安装软件的，B/S只要有IE就可以浏览。  
(3)扩展：C/S的扩展升级比较困难，B/S就很容易的。  
(4)安全：C/S的安全性好一些，B/S就没有C/S的高。  
(5)应用：C/S一般用在局域网，B/S一般用在广域网.

### 静态网页和动态网页

（1）静态网页：  
1）是指网页一旦制作完成，就不能随意更改；  
2）无法实现用户与服务器之间的交互；  
3）制作成本较高，制作周期长，更改困难；  
4）只适合于一些不需经常更改内容的网页。   
（2）动态网页：  
1）能根据用户的要求而动态地改变的页面；  
2）不需要维护人员经常手动更新；  
3）能根据不同的时间、不同的人产生不同的页面；  
4）目前绝大多数网站采用的都是动态网页技术。

### B/S结构的编程语言种类

##### 浏览器端编程语言

主要浏览器端编程语言  
•HTML（Hypertext Markup Language，超文本标记语言）  
•CSS（Cascading Style Sheets，层叠样式表单）  
•JavaScript语言

##### 服务器端编程语言

目前主要采用三种服务器端语言：  
•JSP（Java Server Pages）  
•ASP（Active Server Pages）  
•PHP（PHP：Hypertext Preprocessor）

### java web

Java web是以Java 语言作为脚本语言的,以JSP(Java server pages)网页为整个服务器端的Java库单元提供了一个接口来服务于HTTP的应用程序。是由原太阳微系统公司(sun Microsystems Inc)倡导, 多家公司参与建立的一种动态网页技术。  
它整合了已经存在的Java编程环境, 产生一个全新的网络程序设计语言。  
  
编程形式是将Java代码嵌入到HTML标记中并运行在服务器端。

（1）JSP的优点   
 一次编写，随处运行   
 有统一的技术标准   
 自由扩展JSP标签   
 执行性能高   
 开发工具多而强大   
（2） JSP程序的缺点   
 必须熟悉Java语言   
 难以调试

### 企业应用开发架构

##### 两层架构

两层架构包括用户接口和后台程序，显示逻辑层由html、jsp实现，通过jsp与数据库相连  
  
优点是开发过程比较简单，利用服务器端的程序直接访问数据库，部署起来比较方便。缺点是程序代码维护起来比较困难，程序执行的效率比较低，用户容量比较少。

##### 三层架构

在用户接口代码和数据库中间加入了一个附加的逻辑层：商务逻辑层  
  
三层架构基本解决了两层架构的缺点，将显示部分和逻辑流程控制分开，利用服务器应用程序实现显示部分，利用商务逻辑层实现程序的流程控制，分层使维护变的方便一些，而且执行效率也会有所提高，但是相对部署起来就比较困难一些。

##### n层架构

如果某个应用超过三个独立的代码层，那么这个应用叫做N应用，就不再叫四层或者五层等名称，而是统称为N层  
根据实际的需要，会进一步细化每一层，或者添加一些层，就形成了N层架构，和三层架构一样，组件化的设计使维护相对容易，但是部署相对困难。

### Java Web开发环境的配置

在JSP运行过程中，首先由客户端发出请求，Web服务器接收到请求后，Web服务器对JSP代码进行以下3个操作过程。   
第一步是代码转化（JSP引擎）   
第二步是编译（Java编译器）   
第三步是用Java虚拟机执行编译文件，通过Java虚拟机将执行结果返回给Web服务器，并最终返回给客户端。（Java虚拟机）

JDK由一个标准类库和一组建立、测试及建立文档的Java实用程序组成。其核心Java API（应用程序接口）是一些预定义的类库，开发人员需要用这些类来实现Java语言的各种功能。Java API包括一些重要的语言结构以及基本图形，网络和文件的输入和输出（简记为I/O）

JSP的执行必须同时具备3个条件：JSP引擎、Java编译器和Java虚拟机。   
   
JSP引擎：JSP引擎是运行在Web服务器中的一个组件，用于解析和执行JSP文件，将其转换为可以运行的Servlet代码。   
Java编译器：将Java源代码编译为字节码（.class 文件）的工具。   
Java虚拟机：加载并执行编译后的字节码，完成用户请求处理。   
   
一个典型的运行环境是：   
使用 Apache Tomcat 作为JSP引擎。   
使用 JDK自带的javac 作为Java编译器。   
使用 JVM 来执行字节码并生成响应页面。

### Java Web的执行过程

JSP的执行过程如下：   
1. 客户端通过Web浏览器向JSP服务器发出请求。   
2. JSP服务器检查是否已经存在JSP页面对应的Servlet源代码，若存在则继续下一步，否则转至（4）。   
3. JSP服务器检查JSP页面是否有更新修改，若存在更新修改则继续下一步，否则转至（5）。   
4. JSP服务器将JSP代码转译为Servlet的源代码。   
5. JSP服务器将Servlet源代码经编译后加载至内存执行。   
6. 将产生的结果返回至客户端。

# 第二章 HTML语言和CSS层叠样式

### 1 网页的工作机制

URL：统一资源定位符

http://127.0.0.1:8080/ch1/example.jsp   
 一般由三部分组成:   
 使用的协议，在Web网页中，使用HTTP协议。   
 存放该资源的主机IP地址或者域名（有时也包括端口号，如8080）   
 主机资源的具体地址（如目录和文件名）。 第一部分与第二部分之间用“://”隔开，第二部分与第三部分之间用“/”隔开，第二部分和第三部分有时可以省略。

HTTP：超文本传输协议

执行过程：

从浏览器与Web通信的过程看浏览器的功能：   
1）请求与Web服务器建立TCP连接；   
2）建立并发送HTTP请求   
3）接收并解析HTTP的响应   
4）在浏览器中展示HTML文档   
Web服务器的功能：   
1）接收来自浏览器的TCP连接请求   
2）接收并解析HTTP请求；   
3）创建并发送HTTP的响应

### 2 HTML语言

#### 2.1 页面结构及页面标记使用

（1）HTML文件结构

<html>   
 <head>...</head>   
 <body>...</body>   
</html>

（2）页面标记

种类：   
1）单标记 <元素名称>或 <元素名/>   
2）双标记<元素名称>要控制的元素</元素名称>

（3）语言字符集(Charsets)的信息

<meta http-equiv=“Content-Type” content=“text/html charset=UTF-8”>

<>中分割用空格，""中分割用分号；

（4）背景色彩和文字色彩

<body bgcolor=# text=# link=# alink=# vlink=#>   
bgcolor --- 背景色彩   
text --- 非可链接文字的色彩   
link --- 可链接文字的色彩   
alink --- 正被点击的可链接文字的色彩   
vlink --- 已经点击(访问)过的可链接文字的色彩

（5）连接

跳转到另外的页面  
<a herf="(链接地址)">链接</a>  
<a href="https://www.example.com" target="\_self">打开链接在当前窗口</a>  
<a href="https://www.example.com" target="\_blank">在新窗口中打开链接</a>  
<a href="https://www.example.com" target="myFrame">(自定义)在指定窗口或框架中打开链接</a>

跳转到页面中的另外一个地方（html5取消）  
<a href="#name"> ... </a>  
<a name="name"> ... </a>

（6）标尺线

<hr size=#>

#### 2.2 字体

（1）标题字体

<h1>...</h1> 1~6

（2）字体大小、字体类型、字体颜色的设计

font-size：字体大小；  
font- family ：字体类型（必须是系统字库中有的字体）；  
color:字体颜色  
  
<h1 style=“font-size:60px”>今天天气真好！</h1>  
<h1 style=“font-family:仿宋”>今天天气真好！</h1>   
<h1 style=“color:red”>今天天气真好！</h1>

（3）物理字体

<b>今天天气真好！</b>粗体  
<i>今天天气真好！</i> 斜体  
<u>今天天气真好！</u>下划线  
<sup>今天天气真好！</sup>上标，将文字显示为上标  
<sub>今天天气真好！</sub>下标，将文字显示为下标。  
<s>今天天气真好！</s> 删除线，该标签在指定的文本上画一条删除线

（4）逻辑字体：不指明字体如何显示，而是让Web浏览器自行决定显示方式

（5）组合使用

<i><b>今天</b>天气<u>真好！</u></i>

#### 2.3 页面布局

（1）行的控制

1.自然段<p>  
2.换行<br>  
3.不换行<nobr>

#### 2.4 图像

#### 2.5 表格

#### 2.6 会移动的文字

#### 2.7 多媒体页面

#### 2.8 页面布局的组织

#### 2.9 多窗口页面

#### 2.10 表单

### 3 CSS

#### 3.1 html标记

（1）以HTML标记作为选择符、包含选择符   
p{   
 font-size:30;   
 color:red   
 }   
h1{   
 text-align:center;   
 color:red   
 }   
（2）包含选择符   
指用空格符隔开的两个或多个单一选择符组成字符串，起作用是对某些具有包含关系的元素单独定义样式。   
div p {   
 color : red ;   
 font-size :12   
 } //只对div中的p标记进行设计   
table tr {   
 } //只对table中的tr有效

#### 3.2 class选择符

概念：使用HTML的属性CLASS设置的选择符。   
格式：   
（1）关联：在标记中关联   
 p.标志符{属性：值}   
 <p class=”标志符”>被修饰文字</p>   
（2）独立的CLASS选择符   
 在<style type=”text/css” >中定义   
   
 .标志符 { }   
 </style>   
 例子：   
 <style>   
 .st1 {   
 font-family:隶书;   
 font-size:12pt;   
 color:blue }   
 </style>   
   
 <p class="st1" >蓝色隶书文字 </p>   
 <h1 class="st1">蓝色隶书标题</h1>

#### 3.3 ID选择符

概念：用于定义某个特定的HTML标签，一个页面中只能有一个标记用此id号   
格式：<标记 id=”标志符”>   
 <style type=”text/css”>   
 #标志{属性：值}   
 </style>   
例：   
 <style type=”text/css”>   
 #stid { text-align：center;color:#afdf00 }   
 </style>   
 <div id="stid"> welcome to danqingbuilding</div>

#### 3.4 CSS设置方式

1.在标记中使用，用style属性引用。   
2.嵌入式：在头标记的<style type=”text/css”>中使用   
3.在外部文件：   
 方法：1）将样式保存为.css   
 2）用link标记包含：   
 <link rel=”stylesheet” href=”example.css” type=”text/css”>   
 一般默认type=”text/css”，不需要设置   
各种使用优先级：   
1）浏览器默认使用 2）外部样式 3）嵌入式 4）内联方式。（低——高）   
   
内联方式：<h1 style="color: red;">这是标题</h1>

### 4 HTML的头部标记（head）使用

作用：是头部标记的容器，一般表示对某个页面性质的设定。   
主要包含大标记有： <title>、<base>、<link>、<meta>、<script> 以及 <style>等。

<title> :双标记，用于设定页面窗口标题栏；   
例 <head> <title> This is a example </title></head>

<base>：一般作单标记，页面上的所有链接默认地址或目标；   
例1：<head> <base href=http://localhost:8080/工程名/ /> </head>   
 <img src=“img/1.gif” >   
 将打开图片地址为：http://localhost:8080/工程名/img/1.gif   
例2：<head> <base href=http://localhost:8080/ /> </head>   
 <a href=“register.html” >注册</a>   
 则超链接“注册”将在新开页面中显示内容，文件地址则为：   
 http://localhost:8080/register.html

3）<meta>：一般作单标记，用于设定页面的元信息，比如页面文字字符编码，页面关键字以及过期时间等。主要有两种类型的<meta>标签。   
 第一种<meta>标签的格式：   
 <meta name=“名称” content=“值”>   
 用法一：作用：声明页面的关键词，以利于搜索引挚的收录。   
 <meta name=“keywords” content=“软件开发,网站建设,网站开发”>   
 用法二：作用：声明页面的简单介绍，以利于搜索引挚的收录   
 <meta name=“description” content=“网页简介”>   
 用法三：作用：声明页面的作者，以利于搜索引挚的收录。   
 <meta name=“author” content=“作者”>   
   
 第二种<meta>标签的格式：   
  <meta http-equiv=“名称” content=“值”>   
  用法一：作用：设置页面内容格式及所用字符集。   
 <meta http-equiv=“Content-Type” content=“text/html;charset=UTF-8”>   
  用法二：作用：让浏览器定期刷新或定时转向。   
 <meta http-equiv=“refresh” content=“5;url=http://www.163.com/”>   
  用法三：作用：设置网页过期时间或立即过期。   
 <meta http-equiv=“Expires” content=“Mon,12 October 2023 00:20:00 GMT”>   
 网页在2023年10月12日星期一的00:20:00后过期。   
 <meta http-equiv=“Expires” content=“0”> 网页立即过期   
  用法四：清除缓存的设计.   
 <meta http-equiv=“Cache-Control” content=“no-cache”> ——清除缓存   
 <meta http-equiv=“pragma” content=“no-cache”>禁止浏览器从本地机的缓存中调阅页面内容。   
 若网页过期或不缓存，下次再浏览该网页，都将从服务器上重新下载最新页面。

# 第三章 JS编程基础

### 1 JS简介

#### 1.1 JS介绍

● JavaScript是一种与平台无关的解释性脚本语言，依赖于浏览器，而与操作系统无关，只要计算机能运行浏览器，而且该浏览器支持JavaScript，就可以执行JavaScript脚本。   
● JavaScript可以直接对用户或者客户的输入做出响应，而不需要经过Web服务器程序

#### 1.2 JS和Java的区别

1）JavaScript与Java都是与平台无关的语言，它们都具有很强的实用性。   
2）两者处理方式不同。Java程序必须经过编译，形成独立的字节码，然后在相应的Java虚拟机上执行，正由于Java虚拟机的存在，才使Java能够实现跨平台。JavaScript脚本是嵌入在html文本中，不需要编译，通过浏览器逐行解释执行。   
3）在Java程序中，如果需要某个变量，在使用之前必须进行声明（强变量模式），JavaScript不必事先声明就可以使用变量（弱变量模式）。   
4）Java采用面向对象的程序设计方法，编程时需要创建对象，而后编程。JavaScript是基于对象编程，它可以将html的标记、浏览器相关对象、自定义对象作为处理对象，基于它的属性和方法编程。   
5）JavaScript是一种嵌入语言，它通过在html文本中使用标记：<script>…</script>来插入JavaScript小程序。Java以类为单位建立独立程序文件   
6）JavaScript是一种脚本语言，学习起来比Java简单得多.

#### 1.3 JS的实现

完整的JavaScript编程包括三个组成部分：   
Script（核心部分）ECMA Script：是JavaScript的基础部分，包括数据类型、控制语句、保留字、函数、对象等。   
DOM（文档对象模型）：JavaScript 操作 html 文档的接口，提供了访问 html 文档的途径以及操作方法。   
BOM（浏览器对象模型）：提供了独立于内容而在浏览器窗口之间进行交互的对象和方法。

三者的关系与作用   
ECMAScript：提供 JavaScript 的语言核心能力，如变量、运算、函数、控制流等。   
DOM：提供操作网页内容的接口，让 JavaScript 能够动态修改 HTML 和 CSS。   
BOM：提供操作浏览器窗口和环境的接口，处理 URL、导航、弹窗等。   
三者协同工作：   
   
ECMAScript 提供语言基础。   
DOM 让 JavaScript 可以动态操作页面内容。   
BOM 让 JavaScript 能与浏览器交互。

### 2 JS基础

#### 2.1 标志符

##### JavaScript标志符命名规范：

必须以字母开头   
能以$和\_符号开头   
名称对大小写敏感   
长度没有特定要求

#### 2.2 数据类型与运算符

##### （1）常见的数据类型一共有7种：

数字类型：包括整数和浮点数。整数可以为正整数、0或者负整数，浮点数可以包括小数点，如8.33，或者“E”（也可为“e”） 如7E-2等。   
字符串类型：字符串数据应加上单引号或者双引号，例如“hello”   
对象类型：这是JavaScript的重要组成部分，将在后面的小节中介绍。var obj\_person={firstName:“Marry", lastName:“Eich"};   
布尔类型：可以为true和false两个值。   
数组类型：var name=[“zhangsan”,”lisi”,”wangwu”];   
Null（空）、Undefined（未定义）。数据类型可以typeof查看

##### （2）常用运算符

① 算数运算符:+、-、、/、%、、++、--。  
② 赋值运算符=、+=、-=、=、/=、%=、=。  
③ 关系：>、<、>=、<=、==、!=、===、（绝对等,值与类型均相等） 、！== （不绝对相等，值和类型有一个不相等,或两个都不相等）。  
④ 逻辑运算符：！、&&、||。

#### 2.3保留字

var、abstract、boolean、break、int 等。

#### 2.4常用语句

##### 函数定义语句

function 函数名称 (形参)  
{  
 函数执行部分  
 return 表达式  
}    
函数调用的格式:  
函数名称（实参）  
函数调用位置：  
1）在脚本中被调用  
2）通过事件驱动被调用  
3）在超链接点击被调用

##### 条件语句

if (条件)   
{ 语句1 }   
else   
{ 语句2 }

##### 分支语句

switch (expression)  
{   
 case label1: 语句1;  
 case label2: 语句2;  
 case label3: 语句3;   
 ……  
 default : 语句n; }  
如果表达式expression的值与任何一个label都不匹配，将执行default后面的语句n。

##### 循环语句

for语句：  
for (变量初始化; 条件; 更新变量)  
 { 语句 }  
   
for…in语句：  
for (变量 in 对象或数组)  
 { 语句 }  
   
while语句：  
while (条件)  
 { 语句 }   
   
do while语句：  
 do  
 { 语句 }   
 while(条件)；

#### 2.5事件

在客户端处理时需要通过表单控件的事件驱动机制驱动和客户端脚本来实现

##### 事件过程的调用方式

事件过程按其名称被识别，事件过程的命名规则是： on事件名，如Click事件过程名为onclick。事件过程的定义与调用有如下2种方式。

##### (1) 通过等标记块实现

在body标记块中用onload 等事件调用   
格式如 <body onload = 函数名（）>

<HTML>   
<HEAD><TITLE>调用事件过程</TITLE>   
<SCRIPT LANGUAGE=javascript>   
function mysub()   
{ aa = " onload事件调用"; alert(aa); }   
</SCRIPT>   
</HEAD>   
<BODY onload=mysub()> </BODY>   
</HTML>

##### (2) 在对象定义标记中调用事件过程

在对象定义标记中设置事件过程属性可以调用函数。

<HTML>   
 <HEAD><TITLE>调用事件过程</TITLE>   
 <SCRIPT LANGUAGE=javascript>   
 function mysub()   
{ aa = "这是按钮的单击事件“; alert(aa); }   
</SCRIPT>   
 </HEAD>   
 <BODY><H3>调用事件过程</H3><HR>   
 <INPUT type=button onclick=mysub() value=确定>   
 </BODY>   
</HTML>

##### (3) 在超链接中调用事件过程

在对象定义标记中设置事件过程属性可以调用函数。

<HTML>   
<HEAD><TITLE>调用事件过程</TITLE>   
<SCRIPT LANGUAGE=javascript>   
function mysub()   
{ aa = "这是按钮的单击事件“; alert(aa); }   
</SCRIPT>   
</HEAD>   
<BODY><H3>调用事件过程</H3><HR>   
<a href=“javascript:mysub();”> 调用函数</a>   
</BODY>   
</HTML>

##### 事件种类

* **鼠标事件：onMouseDown、onMouseUp、onMouseOut、OnClick、onMouseOver、onMouseMove。**
* **键盘事件：onKeydown、onKeyup、onKeypress。**
* **焦点事件：onFocus、onBlur。**
* **调整窗口尺寸事件：onresize。**
* **加载和卸载窗口事件：onLoad、unLoad。**
* 
* <html>  
  <head>  
  <script language=javascript>  
  function ss()  
  { if (f1.r1(0).checked==true)  
  alert(f1.r1(0).value);  
  else  
  alert(f1.r1(1).value); }  
  </script>  
  </head>  
  <body > <form name="f1">  
  <input type="text" name="t1"><br>  
  <input type="text" name="t2"><br>  
  <input type="radio" name=r1 value="男" checked>nan<br>  
  <input type="radio" name=r1 value="女">女<br>  
  <input type=button value="执行" onclick=ss() title="this is a  
  example">  
  </form>  
  </body>  
  </html>

#### 2.6对象

##### String对象:内部动态性

基本使用格式：objectName.属性/方法

###### **1.** **串对象的属性**

该对象常用的只有一个属性，即length。它表明了字符串中的字符个数，包括所有符号。

###### 2.**串对象的方法**

1）字符搜索：   
格式：indexOf [ charactor,fromIndex ]   
作用：从指定formIndtx位置开始搜索charactor第一次出现的位置。   
   
2）返回字符串的一部分子串：   
格式：substring(start,end) ; charAt(n)   
返回从start开始到end的字符; 返回第n个字符，n:0-length   
3）字符串大小写转换   
toLowerCase()：小写转换   
toUpperCase()：大写转换   
4）有关字符显示的控制方法   
big()：用大字体显示；   
italics()：斜体字显示   
bold()：粗体字显示   
blink()：字符闪烁显示   
small()：字符用小体字显示   
fixed()：固定高亮字显示   
fontcolor(color)：字体颜色方法；   
fontsize(size)：控制字体大小等

##### Date对象：动态性

提供一个有关日期和时间的对象

即必须使用new运算符创建一个实例。

Date对象没有提供直接访问的属性。只具有获取和设置日期和时间的方法。

var now=new Date() ; //将系统当前日期及时间作为对象now的初始值；   
1）获取日期的时间方法   
getYear()： 返回年数   
getMonth()：返回当月号数 //月份的号从0开始计数   
getDate()：返回当日号数   
getDay()：返回星期几   
getHours()：返回小时数   
getMintes()：返回分钟数   
getSeconds()：返回秒数   
getTime()：返回毫秒数 //日期起始值:1970年1月1日 00:00:00。   
2）设置日期和时间： 时间设定调用时不需要返回值。   
setYear()：设置年   
setDate()：设置当月号数   
setMonth()：设置当月份数   
setHours()：设置小时数   
setMintes()：设置分钟数   
setSeconds()：设置秒数   
setTime ()：设置毫秒数   
例：now.setDate(2022)； //将now的年份设定为2022年

##### Math对象：静态对象

提供除加、减、乘、除以外的一引些自述运算。如对数，平方根等

1.主要属性   
Math中提供了10个属性，它们是数学中经常用到的常数e、以10为底的自然对数LN10、以2为底的自然对数LN2、3.14159的PI、1/2的平方根SQRT1-2、2的平方根为SQRT2等。   
绝对值：abs(x)   
正弦余弦值：sin(x),cos(x)   
反正弦反余弦 :asin(x), acos(x)   
正切反正切：tan(x),atan(x)   
四舍五入取整：round(x)   
平方根：sqrt(x)   
对一个数进行舍入取整：floor(x)   
求自然对数：log(x)   
求e的指数：exp(x)   
基于几方次的值：pow(base,exponent)   
返回0~1之间的随机数：random()

##### 数组对象Array :动态性

提供对数组类型数据的操作。

即必须使用new运算符创建一个 实例

属性   
1）constructor 所建立对象的函数参考   
2）prototype 能够为对象加入的属性和方法   
3）length 获取数组元素的个数,即最大下标加1   
使用：可以采用new和Array对象来创建数组。   
基本语法：   
1)arrayObj=new Array() //说明此时创建的数组元素不固定   
2)arrayObj=new Array(size) //此时创建的数组的个数为size   
3)arrayObj=new Array(元素1，元素2，…,元素n)，存入数组的元素值及其个数   
方法：   
1）concat(array1,arrayn)：将两个或两个以上的数组值连接起来，合并后返回结果；   
2）join(string) ： 将数组中元素合并为字符串，string为分隔符。如省略参数则直接合并，不再分隔；   
3）pop()： 移除数组中的最后一个元素并返回该元素；   
4）push(value) ：在数组的末尾加上一个或多个元素，并且返回新的数   
组长度值；   
5）reverse()：颠倒数组中元素的顺序,反向排列；   
6）shift()：移除数组中的第一个元素并返回该元素；   
7）sort(compare Function) ：在未指定排序号的情况下,按照元素的字母顺序排列，如果不是字符串类型则转换成字符串再排序,返回排序后的数组；   
8）splice()：为数组删除并添加新的元素；   
9）toString()：将数组所有元素返回一个字符串,其间用逗号分隔；   
10）unshift(value) ：为数组的开始部分加上一个或多个元素，并且返回该数组的新长度；   
11）valueOf()：返回数组对象的原始值；

##### 全局函数

即JavaScript 的内部函数，可作用在JavaScript的所有内部对象上

1)parseInt   
作用：将字符串转换为整数   
格式: parseInt(“字符串”)；   
parseInt(“字符串”，“进制”)；   
2）parseFloat   
作用：将字符串转换为实数   
格式：parseFloat(“字符串”)   
3）String   
作用：将对象转换为字符串   
格式：String（object）   
4) eval   
作用：函数可计算某个字符串，并执行其中的的 JavaScript 代码   
格式：eval(“字符串”)

### 3 JS的DOM编程

JavaScript 能够改变页面中的所有 HTML 元素内容

JavaScript 能够改变页面中的所有 HTML 属性

JavaScript 能够改变页面中的所有 CSS 样式

JavaScript 能够对页面中的所有事件做出反应

#### 3.2 DOM的实现

通过 id 找到 HTML 元素   
var x=document.getElementById(“id名”);--------简单变量   
var array=document.getElementsByName(“name名”)； --数组   
var array= document.getElementsByTagName(”标记名”); --数组   
改变 HTML 内容   
document.getElementById(id).innerHTML=new HTML；   
改变 HTML 属性   
document.getElementById(id).attribute=new value；   
改变 HTML 样式   
document.getElementById(id).style.property=new style ；   
创建新的 HTML 元素   
1）var 元素变量名=document.createElement(“html标记");   
2）var 结点变量=document.createTextNode(“内容");   
3）元素变量.appendChild(结点变量);   
4）var 已有元素=document.getElementById(“元素id”);   
5）已有元素.appendChild(新元素变量名);   
删除已有的 HTML 元素   
在母标记中删除子节点的过程：   
1）var 母节点变量名=document.getElementById ("母标记id名");   
2）var 子元素变量=document.getElementById ("子标记id名");   
3）母节点变量名.removeChild(子元素变量);   
注：DOM只能删除已有母节点中的子节点

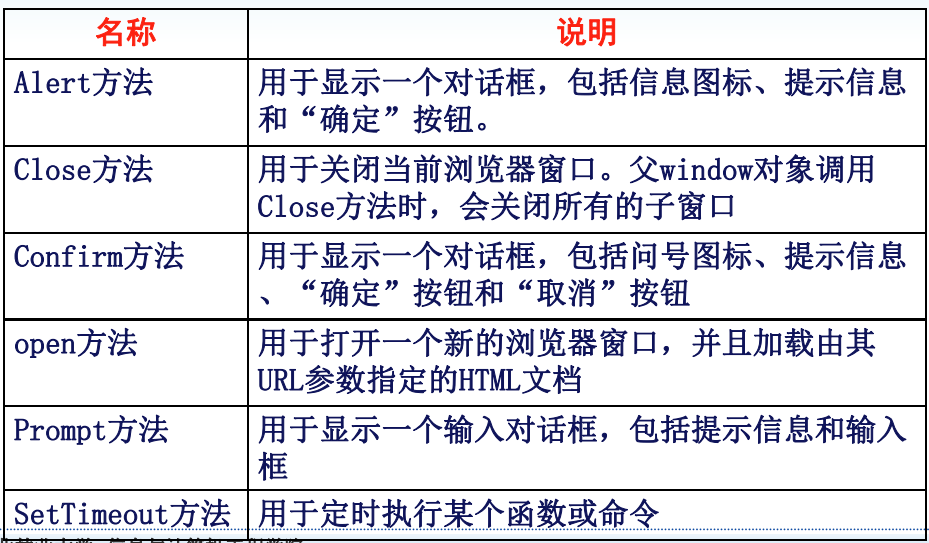
### 4 JS的BOM编程

#### 简介

JavaScript的BOM(Browser Object Model)编程技术提供了JavaScript与浏览器对话的方式，即可以通过访问浏览器对象实现动态程序设计的一种方式。  
常用浏览器对象有：window、document、navigator、location、history、screen、cookies等；其中最常用的是window对象，它也是其他对象的母对象。

#### window对象

所有浏览器都支持 window 对象。它表示浏览器窗口。  
所有 JavaScript 全局对象、函数以及变量均自动成为 window对象的成员。  
全局对象是 window 对象的属性。  
全局函数是 window 对象的方法。  
HTML DOM 的 document 也是 window 对象的属性之一



（1）alert方法  
<html>  
<head>  
<script type="text/javascript">  
function disp\_alert()  
{ alert("我是警告框！！") }  
</script>  
</head>  
<body>  
<input type="button" onclick="disp\_alert()" value="显示警告框" />  
</body>  
</html>

#### history对象

包含浏览器的历史

方法 history.back() - 与在浏览器点击后退按钮相同，返回上一URL;  
 history.forward() - 与在浏览器中点击按钮向前相同，前进下一URL;  
 history.go(n)-根据n的正负觉得前进或后退

#### navigator对象

对象包含有关访问者浏览器的信息

属性：<div id="example"></div>  
<script>   
txt = "<p>Browser CodeName: "+navigator.appCodeName +"</p>";   
txt+= "<p>Browser Name: " + navigator.appName +"</p>"; txt+="<p>Browser Version: " + navigator.appVersion +"</p>"; txt+="<p>CookiesEnabled: "+navigator.cookieEnabled +"</p>"; txt+= "<p>Platform:"+navigator.platform +"</p>";   
txt+= "<p>User-agent header: "+navigator.userAgent + "</p>"; txt+="<p>User-agent language:"+navigator.systemLanguage+"</p>";  
</script>

#### 5 正则表达式

正则表达式的检索实现包括定义模式及执行两个过程：  
1）定义 RegExp  
格式：var 模式对象=new RegExp（模式[,属性]）  
说明：模式对象是创建的正则表达式对象，用于匹配模式用；模式是待检索匹配的字符，即匹配的模式，写在两个“//” 之间。属性是指对模式匹配的范围或方式进行说明，(i:不区分大小写;g:全局范围匹配；m：多行匹配）  
例：var e=new RegExp(/[A-Z]/) :匹配模式为英文大写字母  
2）RegExp的执行  
RegExp有三个方法执行匹配，分别是test()、exec()、compile()  
.\* test("待查找字符串")  
作用：用于检索指定指定字符串中是否含有模式字符串；  
返回值：true| false;  
例：e.test(“abABC”):查找字符串”abABC”是否符合模式e.\* exec("待查找字符串")  
作用：用于检索指定字符串中的模式；  
返回值：找到的字符串或是null

# 第四章 Jquery基础

### 1.Jquery概述

#### 1.1 Jquery定义:

Jquery是一个JavaScript函数库，它是一个轻量级的库，通过少量的代码实现丰富的功能，包括HTML元素选取、操作、遍历和修改、JavaScript特效和动画、CSS操作、HTML事件操作以及AJAX异步请求等。

#### 1.2 Jquery功能:

Jquery库提供了多种功能，如HTML元素的选取和操作、DOM遍历和修改、JavaScript特效和动画效果、CSS操作、HTML事件操作以及AJAX异步请求等。

#### 1.3 Jquery版本和下载:

Jquery有两个版本，生产版本用于实际网站，开发版本用于测试和开发。Jquery可以从官方网站下载，也可以通过CDN如百度、新浪、谷歌和微软等服务器引用。

### 2.Jquery语法

#### 2.1 基础语法:

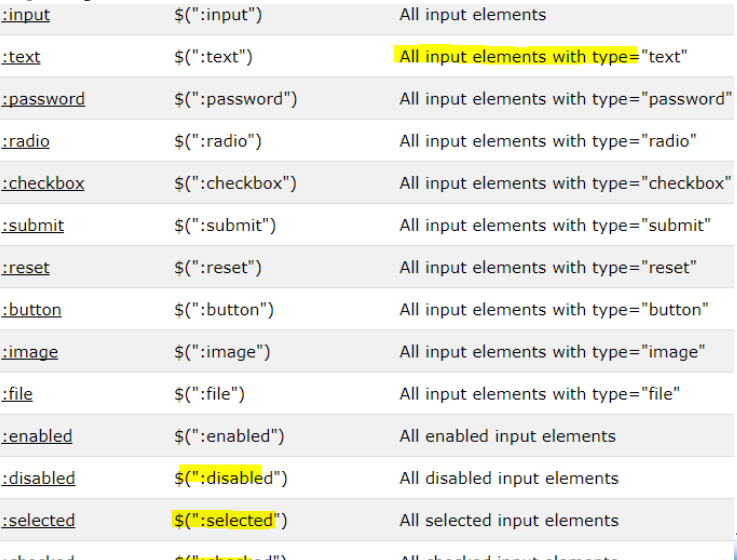
Jquery的基础语法是**$(selector).action()**，其中$是Jquery的标志，selector是选择器，用于选取元素，action()是对选取的元素执行的操作。

$ : Jquery指令的标志  
selector ：选择器，同CSS选择器（HTML标记、id、class）使用时要以字符串的形式出现。例：”div”,”#id”,”.class”;甚至可以是变量，字符串变量。  
action() : 执行对元素的操作（事件、方法、回调函数）例：hide()：隐藏选择器 click() : 单击选择器； val() : 获取控件值

#### 2.2 选择器用法:

$(this).hide()——隐藏当前元素  
$(“p”).hide()——隐藏所有HTML标记<p>的元素  
$(“.test”).hide()——隐藏所有class为test的元素  
$(“#test”).hide()——隐藏所有id是test的元素





获取name属性值为username的input元素  
$('input[name=username]')  
获取name属性值为courses，并且为被选中状态元素  
$('input[name=courses]:checked')  
当获取多个元素时，返回jquery对象，而非js数组  
可在事件回调函数中直接使用jquery对象  
$('input[name=courses]').click(function(){  
$(this).hide();  
});  
指定name属性值，指定元素中，被点击的元素对象

: 可以理解为种类的意思，如：p:first，p的种类为第一个。

[ ]很自然的可以理解为属性的意思，如：[href] 选取带有href 属性的元素。

### 3.Jquery事件

#### 3.1 事件处理:

Jquery是为事件处理特别设计的

click()：当HTML元素点击事件被触发时会调用一个函数；  
dblclick():当HTML元素双击事件被触发时会调用一个函数；  
mouseenter():当鼠标指针穿过元素时，会发生的事件；  
mouseleave()：当鼠标指针离开元素时，会发生的事件  
hover()：用于模拟光标悬停事件  
focus()：当元素获得焦点时，发生 focus 事件  
blur():丢失焦点

#### 3.2 文档准备事件:

Jquery的$(document).ready(function(){...})用于在文档加载完成后执行Jquery代码，也可以简写为$(function(){...})。

### 4.Jquery方法

#### 4.1 页面效果:

JQuery 方法用于具体效果的完成

Jquery提供了hide()、show()、toggle()、fadeIn()、fadeOut()、fadeToggle()和fadeTo()等方法来实现元素的隐藏、显示和淡入淡出效果。

（1）hide() 和 show()  
作用：隐藏和显示 HTML 元素  
格式：$(selector).hide(speed,callback);  
参数说明：speed:速度，可以为slow,fast,毫秒计 callback：之后执行的函数  
（2）toggle()  
作用：切换 hide() 和 show() 方法；隐后显，显后隐  
格式： ：$(selector).toggle(speed,callback);\* 说明：参数均可选。  
（3）Fading （）  
作用：实现元素的淡入淡出效果  
种类：  
1）fadeIn()：淡入已隐藏的元素  
2）fadeOut()：淡出可见元素  
3）fadeToggle()：  
4）fadeTo(）：允许渐变为给定的不透明度（0,1）  
格式： $(selector).fadeIn(speed,callback);

#### 4.2 获取/设置内容和属性:

Jquery提供了text()、html()、val()、css()和attr()等方法来获取或设置元素的文本内容、HTML内容、表单值、CSS样式和属性。

(1)获取或设置内容：  
text() - 设置或返回所选元素的文本内容  
html() - 设置或返回所选元素的内容（包括 HTML 标记）  
val() - 设置或返回表单字段的值  
格式：$(“选择器”).text()——获取  
$(“选择器”).text(参数)——设置为参数  
例： alert("值为: " + $("#test").val());  
<input type="text" id="test" value="Web程序">  
\*注：无参获取，有参设定对应选择器的值  
  
(2)获取或设置CSS样式：  
1）css(“属性名”) ----------获取匹配的css样式的属性；  
2）css(“属性名”，“属性值”)-------为匹配属性设定值  
作用：获取对应属性名的值  
例： $(“#runoob”).css(“color”,”red”);   
<div id=“runoob” >菜鸟教程</div>  
得到：菜鸟教程  
注：可以通过addClass(“样式的class”) 方式改变样式，即通过添加class的方式加样式。  
  
(3)获取或设置属性：  
1）attr(“属性名”) ----------获取匹配属性；  
2）attr(“属性名”，“属性值”)-------为匹配属性设定值  
作用：获取对应属性名的值  
例：  
alert($("#runoob").attr("href"));   
<a href="http://www.runoob.com" id="runoob">菜鸟教程</a></p>  
得到：http://www.runoob.com

#### 4.3 添加和删除元素:

Jquery提供了append()、prepend()、after()、before()、remove()和empty()等方法来添加和删除元素。

append() - 在被选元素的结尾插入内容  
prepend() - 在被选元素的开头插入内容  
after() - 在被选元素之后插入内容  
before() - 在被选元素之前插入内容  
格式：$(“选择器”).append(“添加元素1”[,”元素2”,...])  
说明：  
1)添加元素可以是HTML标记的  
2)可以添加多个元素  
  
remove() - 删除被选元素（及其子元素）  
empty() - 从被选元素中删除子元素  
格式：$(“选择器”).remove()  
说明：remove()方法可以有参数，是备选的元素  
格式：$(“选择器”).remove(“选择器”)  
  
对被选对象添加一个或多个事件处理程序方法  
1) bind( )  
格式：$(“选择器”).bind( “事件”,函数)  
说明：bind()方法在触发事件时，执行函数  
2) on( )   
格式同bind()方法。在Jquery1.7以后用on方法

### 5.Jquery尺寸及遍历

#### 5.1 尺寸方法:

Jquery提供了width()、height()、innerWidth()、innerHeight()、outerWidth()和outerHeight()等方法来获取或设置元素的尺寸。

1)jQuery 尺寸方法  
width() 方法设置或返回元素的宽度（不包括内边距、边框或外边距）。  
height() 方法设置或返回元素的高度（不包括内边距、边框或外边距）。  
innerWidth() 方法返回元素的宽度（包括内边距）。  
innerHeight() 方法返回元素的高度（包括内边距）。  
outerWidth() 方法返回元素的宽度（包括内边距和边框）。  
outerHeight() 方法返回元素的高度（包括内边距和边框）。

#### 5.2 遍历:

Jquery提供了遍历DOM树的方法，如parent()、children()、next()、prev()等，用于导航和查找DOM元素。

### 6.Jquery插件

#### 6.1 插件概述:

Jquery插件扩展了Jquery的功能，提供了如表单验证、密码验证、树形目录、Cookie管理和折叠菜单等额外功能。

#### 6.2 插件使用:

通过Jquery官网下载插件，然后通过标签引入相应的插件文件，即可使用插件提供的功能。

# 第五章Java Web

## JSP编程技术

### Java Web理解服务器端执行

Java Web页面必须通过**Web服务器**执行，然后通过HTTP进行访问。

**服务器端的Java代码被编译后解释执行。客户端则执行JavaScript代码。**  
通过程序5-1.jsp展示了服务器端和客户端的执行过程，

**服务器端执行Java代码获取服务器时间，客户端执行JavaScript代码获取浏览器时间。**  
服务器端执行的结果可以在客户端查看，但<%%>中的Java代码内容在客户端源代码中不可见。

#### 注释

注释分为两种，包括HTML注释和隐藏注释。  
1．HTML注释：在客户端显示一个注释。  
HTML注释的Java Web语法：  
<!-- comment [ <%= expression %> ] -->  
2．隐藏注释：写在JSP程序中，但是不发给客户。  
隐藏注释的Java Web语法：  
<%-- comment --%>

### Java Web页面结构

Java Web页面由四种元素组成：编译指令、操作指令、Java Web代码和EL表达式。

#### 编译指令

编译指令用于告诉JSP解释引擎在编译时需要执行的动作，如引入类、设置字符集和编译语言等。

page指令：定义与JSP页面相关的属性，并和JSP引擎进行通信。  
include指令：定义JSP编译时需要插入的资源。  
taglib指令：定义JSP页面可以调用的一个客户标记库。  
<%@ page language="java" %>  
<%@ include file=“hearder.htm"%>  
<%@ taglib uri=#prefix=“util" %>

##### page指令

page指令用于定义与JSP页面相关的属性，并与JSP引擎进行通信。一个JSP页面可以包含多个page指令，但每个属性只能定义一次。page指令可以运用于整个JSP文件。

page指令针对于当前页面，由<%@%>进行标记，常用的指令有8个：language; extends; import; errorPage; isErrorPage; contentType, isTreadSafe; session;  
  
（1）language="java"  
声明脚本语言的种类，暂时只能用“java”。  
（2）extends="package.class“  
Extends是当前JSP要继承的父类，一般无需设置。在默认情况下，JSP页面的默认父类是HttpJspBase。  
例如：当前JSP页面要继承mypackage包下的myclass类 ， 相 应 的 声明语句为 “<%@ page extends="mypackage.myclass"%>”。  
（3） import="{package.class | package.\* }, ..."  
<%@ page import="java.io.\*,java.util.Hashtable" %>  
或者 <%@ page import ="java.io.\*" %>  
<%@ page import ="java.util.Hashtable" %>  
需要引入的包，或是引入的类，是JDK当中的类，也可以是用户自定义的类。有些类在默认情况下已经被加入到当前JSP页面，而不需要特殊声明，包括四个类java.lang.\*;java.servlet.\*;java.servlet.jsp.\*;java.servlet.http.\*;。  
（4）<%@ page isErrorPage="true" %>  
isErrorPage用来设定当前的JSP页面是否作为传回错误页面的网页，默认值是“false”。如果设定为“true”，则JSP容器会在当前的页面中生成一个exception对象。  
（5）<%@ page errorPage=“/error.jsp" %>  
errorPage用来设定当JSP页面出现异常（Exception）时，所要转向的页面。如果没有设定，则JSP容器会用默认的当前网页来显示出错信息。例如：“<%@page errorPage="/error/error\_page.jsp"%>”  
（6）<%@ page contentType = "text/html" %>设置传回网页的文件格式及编码方式,一般使用“text/html;charset=GBK”。  
（7）isTreadSafe定义web服务器执行JSP程序的方式，默认“true”，代表多线程的方式执行jsp页面，“false”代表单线程的方式执行。  
（8）session当前jsp页面是否用到session对象。  
  
page指令的详细格式如下：  
<%@ page  
[ language="java" ] [ extends="package.class" ]  
[ import="{package.class | package.\*} , ... " ]  
[ session="true|false" ][ buffer="none| 8kb|sizekb" ] [ autoFlush="true|false"][ isThreadSafe="true|false" ] [ info="text" ][ errorPage="relativeURL" ]  
[ isErrorPage="true|false" ]  
[ contentType="{mimeType [ ; charset=characterSet ]   
text/html ; charset=ISO-8859-1}" ]  
[ pageEncoding="{characterSet | ISO-8859-1}" ]  
[ isELIgnored="true | false" ]  
%>

##### include指令

用于指定JSP文件编译时需要插入的资源，如文本、代码、HTML文件或JSP文件。

<%@include file="relativeURL"%>  
其中，relativeURL表示要包含的文件路径。如果路径以“/”开头，则表示该路径是参照JSP应用的上下关系路径，如果路径直接以目录名或文件名开头，则表示该路径是正在使用的JSP文件的当前路径。一旦JSP文件完成编译，该资源内容就不可变，要改变该资源内容，必须重新编译JSP文件。  
  
利用include指令，可以将一个复杂的JSP页面分为若干个部分，这样可以方便管理JSP页面。一个JSP页面一般可以分为三段：head（页头）、body（页体）和tail（页尾）。可以将一个JSP页面分为3个不同的JSP页面：head.jsp、body.jsp和tail.jsp，其中head.jsp表示页头，body.jsp表示页体，tail.jsp表示页尾，这样对于同一网站的不同JSP页面，可以直接利用include指令调用 head.jsp和tail.jsp,仅body.jsp不同。

##### taglib指令

用于定义JSP页面可以调用的自定义标签库。

#### 操作指令

主要是在Java Web页面被请求时动态执行的，可以根据某个条件动态跳转。

##### jsp:include指令

<jsp:include>允许在JSP页面中包含静态和动态页面。如果包含的是静态页面，则只是将静态页面的内容加入至JSP页面中，如果包含的是动态页面，则所包含的页面将会被JSP服务器编译执行。  
语法格式为：  
<jsp:include page=“test.htm”/>  
jsp:include指令必须以“/”结束，功能和include指令相同。  
  
<jsp:include>操作的格式如下：  
<jsp:include page="relativeURL|<%=expression%>" flush="true|false"/>  
page：表示所要包含的文件的相对URL，它可以是一个字符串，也可以是一个JSP表达式。  
flush：默认值为false，若该值为true则表示当缓冲区满时，缓冲区将被清空。  
  
<jsp:include>与<%@include%>指令功能上看不出任何区别。

##### jsp:forward指令

<jsp:forward>操作允许将当前的请求运行转发至另外一个静态的文件、JSP页面或含有与当前页面相同内容的Servlet。操作指令用于把当前的JSP页面转发到另一个页面上。  
<jsp:forward>的格式如下：  
<jsp:forward page="relativeURL|<%=expression%>" />  
<jsp:forward page="5-1.jsp"></jsp:forward>

##### jsp:param指令

<jsp:param>操作提供了“名称──值”信息，通常和<jsp:include>、<jsp:forward>一起使用，包含的页面或重定向的页面将看到新参数增加的原始request对象。可以按照“名字/值”的形式进行参数传递。该操作若独立于<jsp:include>、<jsp:forward>这些操作将没有任何作用。  
<jsp:param>操作的格式如下：  
<jsp:param name="paramName" value="paramValue"/>  
其中paramName表示参数名称，paramValue表示参数值。

##### jsp:plugin指令

<jsp:plugin>用来根据浏览器的类型，插入通过Java插件运行Java Applet所必需的Object或embed元素。如果需要的插件不存在，它会下载插件，然后执行Java组件。Java组件可以是一个applet或一个JavaBean  
<jsp:plugin>的格式如下：  
<jsp:plugin type="applet" codebase="dirname" code="MyApplet.class" width="60" height="80">   
 <jsp:fallback> Unable to initialize Java Plugin </jsp:fallback>   
</jsp:plugin>

jsp:useBean指令、jsp:setProperty指令、jsp:getProperty指令

#### Java Web代码

Java Web代码主要包括三种类型：声明、脚本代码和表达式。

Java Web代码主要包括三种：  
1)声明，<%! 变量、函数或方法 %>；  
2)脚本代码，<% Java 代码%> ；  
3)表达式，<%=Java 表达式%>。

声明用于定义变量和方法，以“<%! %>”标记开始和结束，声明的内容仅在当前页面有效。  
表达式用于输出Java表达式的值，以“<%= %>”标记开始和结束，表达式在执行时会被转换为字符串并显示在页面上。  
Scriptlet代码块用于编写任意合法的Java代码，以“<% %>”标记开始和结束，可以包含多个Scriptlet，按顺序执行。

#### 实例演示

程序5-2.jsp和5-3.jsp演示了错误页面的使用，通过page指令的errorPage属性指定错误页面。  
程序5-4.jsp展示了include指令的使用，通过<%@include file="3-1.jsp"%>包含其他JSP页面。  
程序5-5.jsp、5-6.jsp、5-7.jsp和5-8.jsp分别演示了jsp:include、jsp:forward、jsp:param指令的使用方法和效果。

## JSP内部对象

### 1.JSP内建对象简述

JSP规范中定义了9种内建对象，包括request、response、out、session、application、config、pageContext、page和exception。  
这些对象在JSP中无需额外代码即可自动使用，它们分别对应于Servlet API中的不同类。  
内建对象的使用极大地简化了JSP页面的开发，使得开发者可以专注于业务逻辑的实现。

JSP内部对象的方法实际上都源于Servlet API包中提供的各种类的成员方法。

Servlet API包主要包括  
① javax.servlet；  
② javax.servlet.http；  
③ javax.servlet.jsp。



### 2.out对象

out对象是JspWriter类的子类

用于将信息回送到客户端的浏览器。  
常用方法包括print()和println()，可以输出各种数据类型的内容。

clear()：清除缓冲区中的内容，不将数据发送至客户端。  
clearBuffer()：清除缓冲区中的内容，并将数据发送至客户端。  
close()：关闭输出流。  
flush()：输出缓冲区中的数据。  
getBufferSize()：获取缓冲区的大小。  
getRemaining()：获取缓冲区的剩余空间。  
newLine()：输出一个换行字符，换一行。  
print()：显示各种数据类型的内容。  
println()：分行显示各种数据类型的内容。  
out.println(”<h1>Hello,JSP</h1>“);

### 3.request对象（maybe重）

request对象用于获取客户端请求的信息，封装为HttpServletRequest接口。

请求信息的内容包括：请求的标头（Header）信息（如浏览器的版本名称、语言和编码方式等）、请求的方式（HTTP方法：如GET、POST和PUT，的method属性设定值）、请求的参数名称和参数值、客户端的主机名称等

#### 常用的request方法

1）getParameter(String strTextName) 获取表单提交的信息。  
String strName＝request.getParameter("name");  
2）getProtocol() 获取客户使用的协议。  
String strProtocol=request.getProtocol();  
3）getServletPath() 获取客户提交信息的页面。  
String strServlet=request.getServletPath();  
4）getMethod() 获取客户提交信息的方式，get|post。  
String strMethod = request.getMethod();  
5）getHeade() 获取HTTP头文件中的accept、accept-encoding和Host的值。  
String strHeader = request.getHeader("accept");  
6）getRemoteAddr() 获取客户的IP地址。  
String strIP = request.getRemoteAddr();  
7）getRemoteHost() 获取客户机的名称。  
String clientName = request.getRemoteHost();  
8）getServerName() 获取服务器名称。  
String serverName = request.getServerName();  
9）getServerPort() 获取服务器的端口号。  
int serverPort = request.getServerPort();  
10）getParameterNames() 获取客户端提交的所有参数的名字。  
Enumeration enum = request.getParameterNames();  
while(enum.hasMoreElements()){  
String s=(String)enum.nextElement();  
out.println(s);  
}

#### 获取请求参数

1）String getParameter(String paramName)：  
获取paramName请求参数的值。  
2）Map getParameterMap()：  
获取所有请求参数名和参数值所组成的Map对象。  
3）Enumeration getParameterNames()：  
获取所有请求参数名所组成的Enumeration对象。  
4） String[] getParameterValues(String name)：  
paramName请求参数的值，当该请求参数有多个值时，该方法将返回多个值所组成的数组。

#### 操作request范围的属性

1)setAttribute(String attName , Object attValue)：  
作用：将attValue设置成request范围的属性值。  
2)Object getAttribute(String attName)：  
作用：获取request范围的属性。  
解决汉字输出问题  
1）在执行获取请求参数前设置编码  
request.setCharacterEncoding(“汉字编码”)  
2）转换字符编码  
//获取原始的请求参数值  
String rawName = request.getParameter("name");  
//将请求参数值使用ISO-8859-1字符串分解成字节数组  
byte[] rawBytes = rawName.getBytes("ISO-8859-1");  
//将字节数组重新编码成字符串  
String name = new String(rawBytes , "gb2312");  
3）获取请求参数同时转换编码  
request.getParameter(“name”).getBytes ("ISO-8859-1");

### 4.response对象（maybe重）

response对象用于封装JSP产生的响应并发送至客户端，封装为HttpServletResponse接口。  
常用方法包括sendRedirect()进行页面跳转，setContentType()设置响应内容类型。  
response对象还支持动态设置ContentType响应和HTTP文件头响应。

#### 应用

相应生成图片  
ImageIO.write(BufferedImage,"bmp", response.getOutputStream());  
  
重定向页面  
response.sendRedirect(“URL”)  
  
HTTP文件头相应  
Response.setHeader(“Refresh”, “时间”)  
  
动态ContentType响应  
response.setContentType(“文档格式”;charset=编码方式)  
其中文档格式可以是text/plain(文本文件);application/x-msexcel(excel文件);application/msword(Word文件)  
  
设置清除页面缓冲区  
response.setHeader("Pragma", "no-cache")  
response.setHeader("Cache-Control", "no-cache");   
response.setDateHeader("Expires", -1);  
  
网页跳转例子  
response对象，最常用到的是sendRedirect()方法，<jsp:forward>只能在站内跳转，response.sendRedirect()可以在任意的网址跳转。  
  
动态contentType响应例子  
当请求jsp页面的时候，page指令设置页面的contentType属性是text/html，服务器端将静态页面发给客户端，可以通过设置contentType来实现其他输出格式。  
text/plain(文本文件)；application/x-msexcel(Excel  
文件)和application/msword（word文件）。  
  
HTTP文件头响应例子  
可以利用jsp改变客户端的响应。  
response.setHeader()。

### 5.application对象（maybe重）

application对象用于为多个应用程序保存信息，直至服务器关闭，封装为ServletContext接口。

#### application对象自定义属性

可以添加自定义属性，通过setAttribute()和getAttribute()方法进行操作。

可以为application对象添加属性，application对象最常用的方法有两个：  
Public void setAttribute（String key, Object obj）:  
将对象obj添加到application对象中，并添加索引关键字key；  
Public Object getAttribute（String key）：获取application对象还有key关键字的对象。  
注：任何对象都可以添加到application对象中，返回的时候需要强制转换为原来的类型  
  
Application对象不会因为某一个或是全部用户离开就消失，一旦建立application对象，它就会一直保存到网站关闭或是这个application对象被卸载。  
removeAttribute()方法  
形式：removeAttribute(String name1)  
说明：删除指定属性的值。使用此方法后，所操作了的属性的值为空值。

application对象常用于实现聊天室和网页计数器等应用。

### 6.session对象（maybe重）

session对象用于记录每个客户端的访问状态，封装为HttpSession接口。当客户端请求超过一个以上的JSP程序网页时，session对象提供有保存请求时期对象属性的方法，所保存的对象在请求过程中都是有效的。

session对象指的就是客户端与服务器端的一次会话。

#### getID()

当一个用户登录一个网站的时候，系统会对用户自动分配一个session，可以通过getID()方法获取session的唯一标识符。

#### 自定义属性

session对象支持自定义属性，常用方法包括setAttribute()、getAttribute()和removeAttribute()。

（1）getAttribute()方法  
形式：String getAttribute(String attribute1)  
说明：获得指定名字的属性，如果该属性不存在，则返回null。  
（2）setAttribute()方法  
形式：String setAttribute(String attribute1)  
说明：设定指定名字的属性值，并将其存储到session对象中。

### 7.PageContext对象

PageContext对象用于取得任何范围的参数，可以获取JSP页面的out、request、response、session、application等对象。  
通过PageContext可以进行请求的重新定向等操作。

### 8.config对象

config对象在Servlet初始化时使用，用于传递初始化参数及服务器信息。  
通过config对象可以获取Servlet配置信息。

### 9.page对象

page对象代表JSP页面本身，是正在运行的由JSP文件产生的类对象。  
page对象在JSP页面中使用，代表了当前的Servlet实例。

### 10.exception对象

exception对象用于显示异常信息，是java.lang.Throwable的一个实例。  
只有在包含isErrorPage="true"的页面中才可以被使用，用于捕获和处理JSP页面中的错误。

### 11.cookie的使用

Cookie对象由Web服务器端产生，保存在浏览器中，用于在浏览器中保存信息。  
创建Cookie的步骤包括创建Cookie对象、写入Cookie和读取Cookie。  
Cookie常用于实现用户登录状态的保持和个性化设置。

1.创建cookie  
格式：Cookie 对象名=new Cookie(“变量名”，值);  
2.写入Cookie中  
格式：response.addCookie(“对象名”);  
3.读取Cookie  
Cookie[] 数据变量名=request.getCookies();

### 12.错误解决方法

文档介绍了HTTP常见错误代码，如404、405和500等。  
通过编写显示错误的页面和设置web.xml可以定制错误信息。  
重点掌握response、request、session和application对象的基本应用。

# 第六章JDBC数据访问接口

### 1.数据库应用开发简介

数据库技术是信息管理的核心，涉及数据、数据库、数据库管理系统和数据库系统四个基本概念。  
数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度，较高的数据独立性和可为各种用户共享。  
数据库管理系统（DBMS）是计算机程序的集合，用于创建和维护数据库，如Oracle、SQL Server、DB2和Informix等。

### 2.SQL语言概述

SQL（Structured Query Language）是使用关系模型的数据库语言，用于和各类数据库连接，提供通用的数据管理和查询功能。  
SQL语言最初由IBM公司开发，实现了关系数据库中的信息检索，并被国际标准化组织确定为国际标准，目前执行的是1992年制定的SQL-92标准。  
SQL语言本身并不是完整的编程语言，需要与其他高级编程语言配合，才能实现应用程序对数据库的访问操作。

#### DQL

1. 基本句型一：（最简单的SELECT语句）  
SELECT 字段名 FROM 数据表  
例1. SELECT \* FROM grade  
功能说明：将grade表中的所有字段取出来。  
例2. SELECT 学号,姓名 FROM grade  
功能说明：将grade表中学号和姓名字段取出来。  
例3. SELECT学号,姓名,语文+数学+英语 as 总成绩 FROM grade  
功能说明：将grade表中的学号和姓名取出来，并将语文、数学和英语成绩相加产生虚拟列总成绩。  
  
2.基本句型二：使用条件查询  
SELECT字段名 FROM数据表 WHERE 筛选条件  
测试句型如下。  
例1. SELECT \* FROM grade WHERE数学>60  
功能说明：把所有数学成绩大于60分的记录选出来。  
例2. SELECT \* FROM grade WHERE数学=300 or 语文=300  
功能说明：把数学成绩等于300分或者语文成绩等于300分的人选出来。  
例3. Like子句基本格式一：“\_”匹配。  
功能说明：每个下划线匹配一个任意字符，注意只匹配一个字符。比如：姓名 like ‘\_敏’，匹配姓名以“敏”字结尾且字数等于二的所有数据记录，如：“张敏”。  
例4. Like子句基本格式二：“%”匹配。  
比如：姓名 Like '%敏%'，匹配姓名中出现“敏”的所有数据记录，如“周惠敏”，“于敏”、“敏大”、“敏二”等。比如要在数据库中查询姓江的人，只要利用一条SQL语句就可以了，SELECT \* FROM 数据库表 WHERE 姓名 Like '江%'。  
  
3.基本句型三：（进行排序）  
SELECT字段名 FROM 数据表 ORDER BY 字段名  
测试句型如下。  
（1）SELECT \* FROM grade ORDER BY数学 注：从低到高排序  
功能说明：从grade表中取出所有字段，并按数学成绩排序。  
（2）SELECT \* FROM grade ORDER BY数学,语文  
功能说明：从grade表中取出所有字段，并按数学成绩排序，如果数学成绩相同则按照语文成绩排序。  
（3）SELECT \* FROM grade ORDER BY数学 desc注：从高到低排序  
功能说明：从grade表中取出所有字段，并按数学成绩倒序。  
（4）SELECT top 5 \* FROM grade  
功能说明：从grade表中取出前五条记录的所有字段。

#### DML

（1）DELETE指令：删除数据记录。  
基本语法：DELETE FROM 数据表 WHERE 条件  
例：DELETE from grade WHERE数学=0   
功能说明：删除所有数学成绩为零的记录，如果没有WHERE子句，则删除所有记录。  
  
（2）UPDATE指令：更新数据记录。  
基本语法：UPDATE 数据表 SET 字段值=新值 WHERE条件  
例1：UPDATE grade SET 数学=数学+10 说明：将grade表中所有人的成绩加10分  
例2：UPDATE grade SET 数学=100 WHERE 姓名 like '%敏%'  
功能说明：将姓名中含有敏的人的数学成绩更新为100分  
  
（3）INSERT INTO指令：添加数据记录。  
基本格式1：INSERT INTO 数据表VALUES （字段新值）  
基本格式2：INSERT INTO 数据表（字段一，二，……）VALUES （字段新值）  
其中关键字两种格式的区别是：当values含有数据库表所有字段的值，并且顺序和数据库字段一致时，就可以省略数据库表后面的字段名称。  
例1：INSERT INTO grade(学号, 姓名, 数学) VALUES (1234,'周润发',70)例2：INSERT INTO grade VALUES (5678, '周润发',70,80,90)  
功能说明：该语句等价于：  
INSERT INTO grade(学号, 姓名, 语文, 数学, 英语) VALUES   
(5678, '周润发',70,80,90)

#### 聚合函数

（1）SUM函数，功能是算出某个字段的总值。  
例. SELECT SUM(数学) As Total FROM grade  
功能说明：求出所有学生数学成绩总和，这个数值的列名为Total。  
（2）AVG函数，功能是算出某个字段的平均值。  
例. SELECT AVG(数学) As Average FROM grade  
功能说明：求出所有学生数学成绩平均分，这个数值的列名为Average。  
（3）COUNT函数，功能是算出返回记录的行数。  
例. SELECT COUNT(\*) As Counts FROM grade  
功能说明：求出满足条件的记录总数。  
（4）MAX函数，功能是算出某个字段的最大值。  
例. SELECT MAX(数学) As First FROM grade  
功能说明：求出所有学生数学成绩的最高分，这个数值的列名为  
First。  
（5）MIN函数，功能是算出某个字段的总值。  
例. SELECT MIN(数学) As Last FROM grade  
功能说明：求出所有学生数学成绩的最低分，这个数值的列名为Last。

#### 分组查询

#### 交叉查询

### 3.JDBC的结构

JDBC（Java Database Connectivity）是Java语言中用于数据库连接的API，它定义了Java程序如何与数据库进行交互。

#### JDBC驱动数据库的四种类型

包括JDBC-ODBC桥加ODBC驱动程序、本地API、JDBC网络纯Java驱动程序和本地协议纯Java驱动程序。

（1）JDBC-ODBC桥加ODBC驱动程序：JDBC-ODBC桥产品利用ODBC驱动程序提供JDBC访问。在服务器上必须可以安装ODBC驱动程序。  
  
（2）本地API：这种类型的驱动程序把客户机API上的JDBC调用转换为Oracle、Sybase、Informix、DB2或其它DBMS的调用  
  
（3）JDBC网络纯Java驱动程序：这种驱动程序将JDBC转换为与DBMS无关的网络协议，之后这种协议又被某个服务器转换为一种DBMS协议。这种网络服务器中间件能够将它的纯Java客户机连接到多种不同的数据库上。  
  
（4）本地协议纯Java驱动程序：这种类型的驱动程序将JDBC调用直接转换为DBMS所使用的网络协议。这将允许从客户机机器上直接调用DBMS服务器，是Intranet访问的一个很实用的解决方法。  
  
第3类和第4类驱动程序将成为JDBC访问数据库的首选方法。第1类和  
第2类驱动程序在直接的纯Java驱动程序还没有上市前会作为过渡方案  
来使用。

#### 连接数据库的方法

包括找到数据库的物理实体、打开数据库（建立数据库的连接）、发送SQL语句和返回发送结果。

#### 数据驱动程序

加载Mysql的JDBC驱动程序  
Class.forName(“ com.mysql.jdbc.Driver”)  
   
Class.forName(" com.mysql.jdbc.Driver ");   
catch(ClassNotFoundException ce){  
System.out.println(ce.getMessage());  
conn=DriverManager.getConnection("jdbc:mysql://host:3306/dbName,”username”,“password”);

#### Connection对象

建立是建立与数据库之间的连接，也就是创建一个Connection的实例。DriverManager类的getConnection()方法将建立数据库的连接：  
public static Connection getConnection(String URL,String user, String password) throws SQLException  
在程序的最后，应该关闭Connection对象：  
public void close() throws SQLException

#### Statement对象

Statement对象用于将SQL语句发送到数据库中。  
存在3种Statement对象：  
（1）Statement：发送基本的sql语句。  
（2）PreparedStatement（从Statement继承而来）：发送带有参数的sql语句或基本sql语句  
（3）CallableStatement（从PreparedStatement继承而来）：调用数据库中的存储过程。  
  
Statement接口提供了两种执行SQL语句的常用方法：  
public ResultSet executeQuery(String sql) throws SQLException  
用于产生单个ResultSet的语句，例如SELECT语句。  
public int executeUpdate(String sql) throws SQLException  
用于执行INSERT、UPDATE或DELETE语句以及SQL DDL语句，例如CREATE TABLE和DROP TABLE。该方法返回一个整数，指示受影响的行数。

#### ResultSet对象

ResultSet包含符合SQL语句执行结果所有行，并且它通过一套get方  
法提供了对这些行中数据的访问，常用的get方法有：  
int getInt(int columnIndex)，取得当前行中第columnInde列的整数的值。  
int getInt(String columnName)，取得当前行中列名为columnName的整数的值。  
Date getDate(int columnIndex)，取得当前行中第columnIndex列的日期的值。  
Date getDate(String columnName)，取得当前行中列名为columnName的日期的值。  
public String getString(int columnIndex)，取得当前行中第columnIndex列的字符串的值。  
public String getString(String columnName)，取得当前中列名为columnName的字符串的值。

#### PreparedStatement对象

PreparedStatement对象表示预编译SQL语句，SQL语句经过预编译后存储在PrepareStatement对象当中，可以用来进行高效的多次执行，PreparedStatement对象是Conncention的preparedStatement()方法创建的；  
  
Public PreparedStatement preparedStatement(String   
sql) throws SQLException  
  
在生成PreparedStatement对象的字符串中可以通过”?”代表一个产生变化的参数，随后通过循环语句生成这一系列语句，方便SQL的生成。  
  
Public void setInt(int parameterindex,int x) throws SQLExecption  
可以将第parameterindex个参数赋值为x，通过addBatch()方法将其加到一个批次作业，通过executeBatch()执行所有加入批次的作业，  
最后应该关闭PreparedStatement对象。

### JDBC四种数据访问格式

格式一：执行基本SQL语句，适用于简单的数据查询和数据操作。

Class.forName("JDBC驱动程序");  
Connection conn=DriverManager.getConnection("相应  
JDBC驱动程序的连接串);  
Statement stmt=conn.createStatement();  
ResultSet rs=stmt.executeQuery("DQL语句");//如果是数据查  
询  
stmt.executeUpdate("DML语句");//如果是数据操作

格式二：实现分页，通过可滚动结果集和结果集并发性设置，实现灵活的行间移动和数据修改。

Class.forName("JDBC驱动程序");  
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);  
Statement stmt = connect.createStatement(结果集类型,结果集并发性);  
ResultSet rs=stmt.executeQuery("DQL语句");//如果是数据查询  
stmt.executeUpdate("DML语句");//如果是数据操作  
   
Statement createStatement(int resultSetType, int resultSetConcurrency)  
结果集类型有3种resultSetType ：  
ResultSet.TYPE\_FORWORD\_ONLY：指定ResultSet对象是不可滚动，这是默认值。  
ResultSet.TYPE\_SCOLL\_INSENSITIVE：指定ResultSet对象是可滚动的，但是对数据库中修改不敏感。  
ResultSet.TYPE\_SCOLL\_SENSITIVE：指定ResultSet对象是可  
滚动的，而且对数据库的修改敏感  
  
结果集并发性有2种resultSetConcurrency ：  
结果集的并发性（Concurrency）决定ResultSet对象是否可以修改数据库中的行。可以使用ResultSet类中定义的int常量来指定结果集的并发性。  
ResultSet.CONCUR\_READ\_ONLY，指定ResultSet对象不能修改数据库，默认值。  
ResultSet.CONCUR\_UPDATABLE，指定ResultSet对象可以修改数据库。  
   
在可滚动结构集中，可以使用多个方法灵活的在行之间移动：  
next()方法：移动到结果集下一行，没有下一行返回false;否则返回true；  
privious()方法：移动到前一行，没有前一行返回false;否则返回true；  
first()方法：移动到第一行，结果集中没有行返回false;否则返回true；  
last()方法：移动到最后一行，结果集中没有行返回false;否则返回true；  
beforeFirst()方法：移动到第一行前面位置；  
afterlast()方法：移动到最后一行后面的位置；  
absolute(int rowNumber)方法：移动到rowNumber指定的行；  
relative(int relativeRowNumber)方法：移动到相对于当前行的一行；  
   
在可滚动结构集中，如果不知道当前行的位置，以下方法检查：  
getRow()方法：返回当前行的行号（int值），没有当前行返回0；  
isFirst()方法：如果当前是第一行，返回true;否则返回false；  
isLast()方法：如果当前是最后一行，返回true；否则返回false；  
isBrforeFirst() 方法：如果当前位置在第一行之前，返回true；否则返回false；  
isAfterLast()方法：如果当前位置在最后一行之后，返回true；否则返回false；  
  
在使用可更新结果集的时候，可以对ResultSet对象本身进行添  
加、删除和修改：  
updateString()方法更新字符串列；  
updateDate()方法更新日期型列；  
updateInt()方法更新数字列。

格式三：执行带参数的SQL语句，使用PreparedStatement对象进行高效的多次执行。

Class.forName("JDBC驱动程序");  
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);  
PreparedStatement stmt = connect.PreparedStatement (带参数的SQL语句);  
pst.setInt(设置参数的值);  
ResultSet rs = pst.executeQuery(); //如果是数据查询  
stmt.executeUpdate(); //如果是数据操作

格式四：执行存储过程，通过CallableStatement对象调用数据库中的存储过程。

Class.forName("JDBC驱动程序");  
Connection conn=DriverManager.getConnection("相应JDBC驱动程序的连接串);  
CallableStatement cs = con.prepareCall("{call 存储过程名(?, ?)}")  
cs.setInt(1, 设置参数的值); //设置输入参数的值  
cs.execute (); //执行存储过程  
String strBack = cs.getString(2) //得到输出参数的值

### 使用JDBC访问数据库

读取Excel数据，利用Connection对象连接Excel文件，并执行SQL查询。  
实现事务管理，通过控制事务的提交和回滚，确保数据的一致性和完整性。  
动态合成SQL语句，根据用户输入动态构建SQL查询语句，提高应用的灵活性。  
实现分页显示，通过ResultSet的absolute()方法直接定位到某一条记录，实现分页效果。

### 使用JavaBean访问数据库

JavaBean是一种可重用的软件组件，通过封装数据库操作的方法，简化了数据库访问的代码。  
通过JDBC技术，JavaBean可以执行SQL查询和数据操纵语言（DML）操作，返回结果集或影响的行数。  
JavaBean的使用提高了代码的模块化和可维护性，使得数据库操作更加高效和安全。

Select 使用executeQuery()方法，返回rs对象；  
DML语句使用executeUpdate()方法，返回整数；  
所以可以在JavaBean中队select和DML提供处理的方法。

# 第七章 JavaBean及Servlet编程技术

### JavaBean概述

定义: JavaBean是一种可重复使用的Java类，可以作为软件组件在开发工具中直观操作。  
作用: JavaBean通过封装属性和方法，实现逻辑处理与界面设计的分离，提高开发效率和降低维护难度。  
结合JSP: 将JSP与JavaBean结合使用，JSP专注于界面设计，JavaBean处理逻辑，使得JSP页面更加清晰。

（1）可以实现代码的重复利用，因此可以缩短开发时间。  
（2）易编写，易维护、易使用。  
（3）可以在任何安装了Java运行环境的平台上使用，而不需要重新编译，为JSP的应用带来了更多的可扩展性。

### JavaBean的特点和特征

特点: JavaBean具有代码复用、易编写、易维护和易使用的特点，可在任何安装了Java运行环境的平台上使用。  
特征: JavaBean类的命名、属性获取和设置方法、构造函数和访问属性都有特定的规范，如属性名前缀为get或set，构造函数无参数等。

（1）如果成员变量的名字是xxx，为了修改或获取成员变量的值，类中应用以下两个方法：  
getXxx()：获取属性XXX；setXxx()：修改属性XXX  
（2）对于boolean类型的成员变量可以利用is代替set。  
（3）类中方法的访问属性必须是public的。  
（4）如果有构造方法，必须是public的，而且无参数（用于初始化工作）。

### JavaBean的创建和使用

#### 创建

创建JavaBean需要定义类、构造函数、属性和属性值的设定与读取方法。

创建一个标准JavaBean的基本语法如下。  
（1）定义包名称  
（2）定义JavaBean类  
JavaBean类定义的语法结构如下：  
public class 类名称{  
//类成员的定义  
//类方法的定义。  
//类属性的定义。 }  
其中需要强调的是，类名称与保存的文件名称必须一致，否则就会出  
现编译错误。同时，类必须声明为公有类，即public。  
（3）定义JavaBean类的构造函数  
JavaBean是Java类，JavaBean就有自己的构造函数，并且构造函数的名称必须与JavaBean类的名称一致。构造函数的主要作用是用来初始化，而且构造函数无参数输入，其语法结构如下：  
public JavaBean类名称  
{  
//初始化  
…….. }  
（4）定义JavaBean属性  
JavaBean是Java类，那么它不但有自己的构造函数，还有自己的属性。JavaBean属性的定义的语法结构如下：  
private 数据类型 属性名称；  
例如：private String maker;  
（5）设定JavaBean属性值的方法  
定义了JavaBean属性之后，还要对其属性值进行设定，其属性值设  
定的语法结构如下：  
public void set设定方法名称(数据类型 参数)  
{this.变量=参数;}  
例如：  
public void setMaker(String maker)  
{ this.maker="Benz"; }  
值得注意的是：JavaBean属性值的设定方法名称一般以set三个字母  
开头，后面跟上属性设定方法名称。  
（6）读取JavaBean属性值的方法  
上面讲述了如何设定JavaBean属性值，那么属性值设定好之后，如  
何读取呢？JavaBean属性值读取的语法结构如下：  
public void get读取方法名称()  
{ return this.变量＝参数; }  
例如：  
public void getMaker()  
{ return this.maker;}

#### 使用:

JavaBean在JSP中通过page指令的import标记引入，或使用jsp:useBean标记创建实例，并通过jsp:setProperty和jsp:getProperty标记进行属性的设置和获取。

JSP访问JavaBean，可以使用page指令的import标记引入页面中需  
要用到的java类。  
import="{package.class | package.\* }, ..."  
<%@ page import="java.io.\*,java.util.Hashtable" %>  
注意：仅仅是引入了一个类，并没有对这个类进行初始化，所以需要通构造函数进行类对象的初始化。  
创建类的对象:  
类名 对象名= new 类名（）用于初始化  
注意：在java声明类时，可以不定义构造函数，系统将为其生成默认的构造函数，与类名字相同，没有参数，也不完成任何操作。JSP 访问 JavaBean ， 也可以使用 <jsp:useBean> 标记。  
<jsp:useBean>标记是用于JavaBean对象的动作标记，当在JSP网页  
中使用它时，表示会产生一个JavaBean的实例。  
<jsp:useBean>标记有5个属性：id、scope、class、beanName和  
type，如：  
<jsp:useBean id="name" scope="page | request | session|application“ class="classname"/>  
<jsp:useBean id="pi" class=“nefu.radius" scope="page"/>  
编译好的JavaBean是一个Class文件，可以通过JAR命令将  
多个class文件打包成JAR。打包过程中，需要将class所在的目录一起打包，保存到根目录下，命令：删除class下的类文件，将JAR添加到lib下程序依然可以执行。

#### <jsp:getProperty>和<jsp:setProperty>

使用<jsp:getProperty>和<jsp:setProperty>标记对属性进行设置和获取。  
<jsp:getProperty>标记用来取得JavaBean的属性值。  
<jsp:getProperty name="beanname" property="propertyname">  
其中，beanname是JavaBean实例名，它是在JSP文本中前面使用<jsp:useBean>标记引入的。propertyname为JavaBean的属性名。  
<jsp:setProperty>标记  
用来设置JavaBean的属性值，它一共有如下3种形式：  
① <jsp:setProperty name="beanname" property="\*">  
② <jsp:setProperty name="beanname"   
property="propertyname" param="paramname">  
③ <jsp:setProperty name="beanname"   
property="propertyname" value="beanvalue">

### JavaBean存在的范围

Page: JavaBean实例的生命周期仅限于一个页面。  
Request: 实例与请求对象相关，可在包含的页面中访问。  
Session: 实例与用户会话相关，会话结束时实例失效。  
Application: 实例生命周期最长，服务器不重启则一直存在。

### 常见JavaBean使用方法

表单Bean: 用于处理表单数据，可将表单元素与Bean属性对应。  
页面Bean: 用于JSP页面保留数据，不涉及应用程序流。  
共享Bean: 在多个页面间共享数据，通过session或application范围实现。

### Servlet基础概述

定义: Servlet是Java Web程序，运行在Web容器上，如Tomcat。

（1）客户向服务器发送对页面的请求。  
（2）如果Servlet还没有装入，服务器就装入它。  
（3）服务器把请求信息送给Servlet，给每一个请求创建一个执行的新线程（Java语言的线程允许同时执行多个任务)。  
（4）Serlvet处理这个请求，生成一个响应并传递给服务器。  
（5）服务器把响应送回给客户。

生命周期: 包括初始化、执行请求和响应、终止三个阶段，由容器管理。

Servlet运行在Servlet容器中，其生命周期由容器来管理。  
Servlet的生命周期与javax.servlet.Servlet接口中  
init(),service()和destroy()方法相对应。  
Servlet的生命周期包含了3个阶段。

结构: Servlet类需要实现javax.servlet.Servlet接口，常用方法包括init()、service()和destroy()。

### Servlet的编写及部署

普通编辑器: 编写、编译、拷贝class文件到WEB-INF/classes目录、配置web.xml、重启服务。  
Eclipse: 建包、建类、编写内容、保存、拷贝servlet-api.jar到WEB-INF/lib、修改web.xml、重启服务。

### Servlet过滤器的使用

定义: Servlet过滤器是Java接口，位于客户端和处理程序之间，用于检查请求和响应。

Servlet过滤器是一中Java的接口，位于客户端和处理的程序之间，能够对请求和响应进行检查。 主要目的如下：  
1）在客户端的请求访问后端资源之前，拦截这些请求。  
2）在服务器的响应发送回客户端之前，处理这些响应  
主要包括：身份验证过滤器、统一字符编码过滤器、字符压缩及加密等。

实现方法: 实现javax.servlet.Filter接口，定义doFilter、init和destroy方法。

Servlet过滤器是一个实现了 javax.servlet.Filter 接口的 Java 类。  
javax.servlet.Filter 接口定义了三个方法：  
1） public void doFilter (ServletRequest,ServletResponse, FilterChain)  
该方法完成实际的过滤操作，当客户端请求的方法与过滤器设置相匹配的URL时，Servlet容器将先调用过滤器的doFilter方法。FilterChain实现用户访问后续过滤器。  
2） public void init(FilterConfig filterConfig)  
web 应用程序启动时，web 服务器将创建Filter 的实例对象，并调用其init方法，读取web.xml配置，完成对象的初始化功能，从而为后续的用户请求作好拦截的准备工作。filter对象只会创建一次，init方法只会执行一次。开发人员通过init方法的参数，获得代表当前filter配置信息的FilterConfig对象。  
3） public void destroy( )  
Servlet容器在销毁过滤器实例前调用该方法，在该方法中释放Servlet过滤器占用的资源。  
用户编写的Servlet过滤器由上述三个方法构成并改写。

配置: 在web.xml中配置和元素，指定过滤器类和URL模式。

### Servlet过滤器实例

LogFilter: 一个实现了Filter接口的Java类，用于输出请求信息和站点名称，展示了过滤器的基本结构和操作。