第 3 章　练功不站桩，等于瞎晃荡——Java Web 开发必备

大家都知道，一个好的武术家一般都是从桩功开始练的，站桩是练武的入门功夫，也就是基本功。没有基本功，纵然你拳法再漂亮，那也是花架子，战不了几个回合就会败下台来。正所谓：练功不站桩，等于瞎晃荡，就是这个道理。而对于 Java Web 开发这个道理同样适用，Java Web 也需要从基本功开始练起，没有基本功，做 Java Web 开发根本是空谈。

3.1　桩功之一：HTML 网页设计

如果说桩功是练武的基本功夫，那么 HTML 对于 Java Web 开发那就是桩功中的桩功。因为在 Java Web 开发中，哪一个界面都离不开 HTML。HTML 描述了要显示的页面的内容和结果，没有 HTML，Java Web 开发也就没有了显示的功能，也就不能显示给用户了。把 HTML 说得如此厉害，那么什么是 HTML？HTML 有什么特点？如何运用 HTML 做 Java Web 开发？接下来我们就开始来揭开 HTML 的神秘面纱吧！

3.1.1　什么是 HTML

HTML 是 HyperText Markup Language 的英文缩写，一般常翻译为超文本标记语言，是用于描述网页形式的一种语言。

HTML 是标准通用标记语言下的一个应用，也是一种规范，一种标准，它通过标记符号来标记要显示的网页中的各个部分。网页文件其实本身就是一种文本文件，它通过在文本文件中添加标记符，可以告诉浏览器如何显示其中的内容（比如：文字如何处理，画面如何安排，图片如何显示等）。浏览器按顺序阅读网页文件，然后根据标记符解释并显示其标记的内容。对书写出错的标记一般浏览器不会指出标记的错误，并且不停止其解释执行的过程，编制者只能通过显示效果来分析出错原因和出错部位。

注意：对于不同的浏览器，对同一标记符可能会有不完全相同的解释，因而可能会有不同的显示效果。所以有时可能出现显示效果与期望效果有些偏差，可能是由于浏览器的兼容性造成的。

HTML 之所以称为超文本标记语言，主要是因为文本中包含了所谓的“超级链接”点。超级链接就是一种 URL 指针，它定义了指向不同的网页或资源，通过（单击）它，可使浏

览器方便地获取新的资源或者跳转到新的网页，这也是 HTML 获得广泛应用的最重要的原因之一。网页的本质就是 HTML，通过结合使用其他的 Web 技术（如：脚本语言、CGI、组件等），就可以创造出功能强大的网页。因而，HTML 是 Web 编程的基础，也就是说万维网是建立在超文本基础之上的。

3.1.2　HTML 语言特点

HTML 文档制作不是很复杂，但功能强大，支持不同数据格式的文件镶入，这也是 HTML 盛行的原因之一。概括来讲其主要特点如下。

- 简易性，HTML 版本升级采用超集方式，从而更加灵活方便。
- 可扩展性，HTML 语言的广泛应用带来了加强功能，增加标识符等要求，HTML 采取子类元素的方式，为系统扩展带来保证。
- 平台无关性。虽然 PC 机大行其道，但使用 MAC 等其他机器的大有人在，HTML 可以使用在广泛的平台上。

3.1.3　HTML 文档的编写方法

用 HTML 编写的超文本文档(以后就称为 HTML 文档或者网页)是非常简单的。HTML 文档的编写方法主要有下面三种。

- 手工直接编写：比如记事本等，存成.htm 或者 .html 格式，这个比较适合初学者。
- 使用可视化 HTML 编辑器编写：比如 Frontpage、Dreamweaver 等，通过这些工具可以制作出非常漂亮的 HTML 网页来，并且这些工具集成了很多快捷方式。这个我们在以后的章节中会细致地讲解。
- 由 Web 服务器（或称 HTTP 服务器）产生： 这是一种实时动态地生成方法。

当 HTML 文档编写完成后，需要将 HTML 文档保存，对于 HTML 文档命名，特别提出需要注意以下几点。

- 后缀名是*.htm 或*.html。
- 无空格。
- 无特殊符号（例如&符号），只可以有下划线“_”，只可以为英文、数字。
- 区分大小写。
- 首页文件名默认为：index.htm 或 index.html。

3.1.4　HTML 文档结构

对于每个 HTML 文档，都有如下的结构。

```
<HTML>              <!--HTML 的开始标志-->
<HEAD>              <!--HTML 的头部信息开始标志-->
<title></title>     <!--标题标记符-->
<meta>              <!--在 HTML 文档中模拟 HTTP 协议的响应头报文-->
</HEAD>             <!--HTML 的头部信息结束标志-->
<BODY>              <!--HTML 的正文开始标志-->
```

```
HTML 文件的正文          <!--HTML 的正文内容-->
</BODY>                  <!--HTML 的正文结束标志-->
</HTML>                  <!--HTML 的文档结束标志-->
```

提示：在上文中出现的<!--文本内容-->是专门用来对 HTML 文档内容做注释用的，中间的文本内容就是注释内容，这种注释标志可以对单行进行注释，也可以对多行进行注释！

标记符<HTML>，说明该文件是用 HTML 来制作的，它是文件的开头标志；而</HTML>表示该 HTML 文件的结尾。<HTML></HTML>是成对出现的，它们是 HTML 文件的始标记和尾标记。

<HEAD></HEAD>，这两个标记符分别表示头部信息的开始和结尾。头部中包含的标记是页面的标题、序言和说明等内容，它本身不作为内容来显示，但影响网页显示的效果。头部中最常用的标记符是标题标记符<title>和<meta>，其中标题标记符用于定义网页的标题，它的内容显在网页窗口的标题栏中，网页标题可以被浏览器用作书签和收藏清单；而<meta>标记符是用来在 HTML 文档中模拟 HTTP 协议的响应头报文。

<BODY></BODY>，网页的正文，网页中显示的实际内容均包含在这两个正文标记符之间，分别标识正文的开始和结束。

3.1.5　HTML 元素属性

HTML 元素可以有自己的相关属性，每一个属性还可以由网页编制者赋予一定的值。元素属性出现在元素的< >内，并且和元素名之间有一个空格分隔；属性值用“ ”括起来。例如下面的代码。

```
<html>
     <head>
       <title>my first html page</title>   <!--网页的标题-->
     </head>
     <body>
        <p align="center">This is my first htmlpage!</p>
                                            <!--另起一段，在网页中居中显示-->
     </body>
</html>
```

可以在自己的电脑上新建一个文本文档，将上述代码输入，另存为 first.html，然后用浏览器将 HTLM 文档打开，可以看到如图 3-1 所示效果。

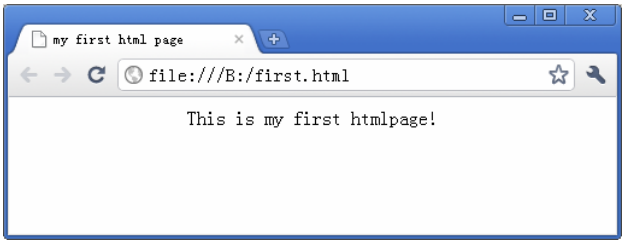

图 3-1　第一个 HTML 图

从图 3-1 中可以看到文字居中在浏览器中显示，这就是 的作用，表示意思是另起一段，并且居中显示。

1．<meta>标签中元素属性的使用

在 HTML 文档结构中提到了<meta>标签，可以看到<meta> 标签用于网页的<head>与</head>中，<meta>的属性有两种：name 和 http-equiv。

1）name 属性

name 属性主要用于描述网页，对应于 content，以便于搜索引擎机器人查找、分类（目前几乎所有的搜索引擎都使用网上机器人自动查找 meta 值来给网页分类）。这其中最重要的是 description 和 keywords，所以在开发网站时建议给每页加一个 meta 值。下面介绍一下比较常用的 name 属性。

```
<meta name="Generater" content="Java WEB ">
                                    <!--用于说明生成工具、开发工具等-->
<meta name="KeyWords" content=" JAVA WEB">
                                    <!--说明您的网页的关键词给搜索引擎等-->
<meta name="Description" content="JAVA WEB">
                                    <!--告诉搜索引擎您的站点的主要内容-->
<meta name="Author" content="longlyboyhe"> <!--告诉搜索引擎您的站点的制作者-->
<meta name="Robots" content= "all|none|index|noindex|follow|nofollow">
```

对最后一行的属性说明如下。

- 设定为 all：文件将被检索，且页面上的链接可以被查询。
- 设定为 none：文件将不被检索，且页面上的链接不可以被查询。
- 设定为 index：文件将被检索。
- 设定为 follow：页面上的链接可以被查询。
- 设定为 noindex：文件将不被检索，但页面上的链接可以被查询。
- 设定为 nofollow：文件将不被检索，页面上的链接可以被查询。

2）http-equiv 属性

http-equiv 主要用于回应给浏览器一些有用的信息，以帮助正确和精确地显示网页内容。下面介绍一下常用的 http-equiv 类型。

（1）Content-Type 和 Content-Language（显示字符集的设定）。

说明：设定页面使用的字符集，用以说明主页制作所使用的文字以及语言，浏览器会根据此来调用相应的字符集显示 page 内容，如下所示。

```
<!--设置页面文档类型为文本/网页类型，编码方式为 gb2312-->
<meta http-equiv="Content-Type" Content="text/html; Charset=gb2312">
 <!--设置语言代码为中文-->
<meta http-equiv="Content-Language" Content="zh-CN">
```

该 meta 标签定义了 HTML 页面所使用的字符集为 GB2132，就是国标汉字码。如果将其中的 charset=GB2312 替换成 BIG5，则该页面所用的字符集就是繁体中文 Big5 码；如果将其中的 charset=GB2312 替换成 UTF-8，则该页面所用的字符集就是“万国码”（可以在同一个网页中支持多个国家的编码方式）。

知识拓展：

我们经常会遇到这种情况，当我们浏览一些国外的站点时，浏览器会提示我们“该页面不能正常显示，如果想正常显示该页面，请下载 xx 语支持”。其实这个功能就是通过读取 HTML 页面 meta 标签的 Content-Type 属性而得知需要使用哪种字符集显示该页面的。如果系统里没有装相应的字符集，则 IE 就会提示下载该字符集。其他的语言也对应不同的 charset，像日文的字符集对应于 iso-2022-jp，韩文的字符集对应于 ks_c_5601。

常见 Charset 选项有：ISO-8859-1（英文）、BIG5、UTF-8、gb2312 等字符集；Content-Language 的 Content 还可以是 EN、FR 等语言代码。不过对于 Charset 的设置，建议以后采用 UTF-8，这样既可以做到支持中文，也便于与国际接轨。

（2）Refresh（刷新）。

说明：让网页多长时间（秒）刷新自己，或在多长时间后让网页自动链接到其他网页。例如：

```
<meta http-equiv="Refresh" Content="30">
<meta http-equiv="Refresh" Content="5; Url=http://www.javaweb.com">
```

这里的 30 是指让网页 30 秒刷新一下自己;5 是指停留 5 秒钟后自动刷新到 URL 网址。

（3）Expires（期限）。

说明：指定网页在缓存中的过期时间，一旦网页过期，必须到服务器上重新调阅。例如：

```
<meta http-equiv="Expires" Content="0">
<meta http-equiv="Expires" Content="Wed, 26 Feb 1997 08:21:57 GMT">
```

注意：必须使用 GMT 的时间格式，或直接设为 0（数字表示多少时间后过期）。

（4）Pragma（cach 模式）。

说明：禁止浏览器从本地机器的缓存中调阅页面内容。

```
<meta http-equiv="Pragma" Content="No-cach">
```

注意：网页不保存在缓存中，每次访问都刷新页面。这样设定，访问者将无法脱机浏览，以保证浏览网站的时长。当您希望访问者每次都刷新广告的图标或者页面，或每次都刷新您的计数器，就要禁止浏览器本地缓存了。

（5）Set-Cookie（cookie 设定）。

说明：浏览器访问某个页面时会将它存在缓存中，下次再次访问时就可从缓存中读取，以提高速度。当您希望访问者每次都刷新计数器，就要禁用缓存了。当保存的 cookie 达到设置的过期时限时，cookie 将被删除。例如：

```
<meta http-equiv="Set-Cookie" Content="cookievalue=xxx; expires=
Wednesday, 21-Oct-98 16:14:21 GMT; path=/**">
                              <!--设置 Cookie，网页过期时限，缓存路径-->
```

（6）Window-target（显示窗口的设定）。

说明：强制页面在当前窗口以独立页面显示。

```
<meta http-equiv="Widow-target" Content="_top">
```

Content 选项：可以为_blank、_top、_self、_parent，说明页面在当前窗口显示的位置。

注意：这个属性是用来防止别人在框架里调用您的页面。

（7）ontent-Script-Type（脚本相关）。

注意：指明页面中脚本的类型。

```
<meta http-equiv="Content-Script-Type" Content="text/javascript"> <!--页面中的脚本为 JavaScript-->
```

对于上面的脚本类型和 JavaScript，我们会在以后的章节中详细讲解，这里只要知道属性的相关作用就可以了。

（8）Page-Enter、Page-Exit 是页面被载入和调出时的一些特效。

```
<meta http-equiv="Page-Enter" Content="blendTrans(Duration=0.5)">
                                <!--持续 0.5 秒页面渐入-->
<meta http-equiv="Page-Exit" Content="blendTrans(Duration=0.5)">
                                <!--持续 0.5 秒页面渐出-->
```

blendTrans 是动态滤镜的一种，产生渐隐效果。另一种动态滤镜 RevealTrans 也可以用于页面进入与退出效果，如下：

```
<meta http-equiv="Page-Enter" Content="revealTrans(duration=0.5, transition=1)">
<meta http-equiv="Page-Exit" Content="revealTrans(duration=0.5, transition=0)">
```

Duration，表示滤镜特效的持续时间（单位：秒）。Transition，表示滤镜类型。使用哪种特效，由它的取值决定，取值范围为 0～23。当我们单击网页上的链接时，浏览器页面就会转到链接指向的新的页面，我们可在页面转换时加上过渡效果。

打开这个页面的原代码，在<head>与</head>插入代码：

```
<meta http-equiv="Page-Exit" content="revealTrans(Duration=3，Transition=5)">
```

这样这个过渡效果就完成了，很简单吧。现在我们来测试一下效果如何，打开这个页面，然后单击页面上的链接，页面在转到下一个页面的过程中，我们看到页面是从上到下慢慢地转换到第二个页面的。

我们现在再试一个效果，将那段代码换成：

```
<meta http-equiv="Page-Exit" content="revealTrans(Duration=1，Transition=14)">
```

我们看到页面是从中间向左右两端展开过渡的，而且速度上快了一点。原因在于 Duration 和 Transition 的值不同。Duration 的值为网页动态过渡的时间，单位为秒。Transition 是过渡方式，它的值为 0～23，分别对应 24 种过渡方式，如表 3-1 所示。

表 3-1　Transition的取值和对应效果表

取　值	特　效	取　值	特　效
0	矩形缩小	4	下到上刷新
1	矩形扩大	5	上到下刷新
2	圆形缩小	6	左到右刷新
3	圆形扩大	7	右到左刷新

续表

取　值	特　效	取　值	特　效
8	竖百叶窗	16	上下到中间
9	横百叶窗	17	右下到左上
10	错位横百叶窗	18	右上到左下
11	错位竖百叶窗	19	左上到右下
12	点扩散	20	左下到右上
13	左右到中间刷新	21	水平线状展开
14	中间到左右刷新	22	垂直线状展开
15	中间到上下	23	随机产生一种过渡方式

当 Transition 为 23 时，会随机产生 0～22 中的一个过渡效果。例如：

```
<meta http-equiv="Page-Exit" content="revealTrans(Duration=2，Transition=23)">
```

2．<body>标签中元素属性的使用

<body>元素表明是 HTML 文档的主体部分。在<body>与</body>之间，通常都会有很多其他元素，这些元素和元素属性构成 HTML 文档的主体部分。下面介绍一下<body>元素中常见元素属性。

（1）bgcolor。bgcolor 属性标志 HTML 文档的背景颜色。如：bgcolor="#CCFFCC"。HTML 对颜色的控制也有自己的语法。HTML 使用十六进制的 RGB 颜色值对颜色进行控制。十六进制的数码有：0，1，2，3，4，5，6，7，8，9，a，b，c，d，e，f。对于颜色值可以查看 RGB 颜色对照表。

（2）background。background 属性标志 HTML 文档的背景图片。如：background="images/bg.gif"。可以使用的图片格式为 GIF 和 JPG。

（3）bgproperties=fixed。bgproperties=fixed 使背景图片成水印效果，即图片不随着滚动条的滚动而滚动。

（4）text。text 属性标志 HTML 文档的正文文字颜色。如：text="#FF6666"。Text 元素定义的颜色将应用于整篇文档。

（5）超级链接颜色。link、vlink 和 alink 分别控制普通超级链接、访问过的超级链接以及当前活动超级链接颜色。

（6）leftmargin 和 topmargin。设置网页主体内容距离网页顶端和左端的距离，如：leftmargin="20" topmargin="30"。

由于本书主要讲 Java Web 开发入门，所以本节的目的主要是教大家怎么用和怎么读取 HTML 文档的，在本章后面的所有章节中都会不断地穿插 HTML 知识，只要在以后的学习中注意对 HTML 知识点的积累就可以了，没有必要一下记住所有的元素属性和使用方法。

如果那样，从本人学习的经验来讲，也是不可取的，我认为只有在用到的时候不断查询并积累才是一种好的学习方法。下面就列出一些常用的元素及其作用，至于如何在 HTML 文档中运用，在后面的章节中遇到了，会有所讲解，就不一一介绍了。

3．HTML界面元素

- <html></html>创建一个 HTML 文档。

- <head></head>设置文档标题和其他在网页中不显示的信息。
- <title></title>设置文档的标题。
- <h1></h1>最大的标题。
- <pre></pre>预先格式化文本。
- <u></u>下划线。
- 黑体字。
- 斜体字。
- <tt></tt>打字机风格的字体。
- <cite></cite>引用，通常是斜体。
- 强调文本（通常是斜体加黑体）。
- 加重文本（通常是斜体加黑体）。
- 设置字体大小从 1～7，颜色使用名字或 RGB 的十六进制值。
- <BASEFONT></BASEFONT>基准字体标记。
- <big></big>字体加大。
- <SMALL></SMALL>字体缩小。
- <STRIKE></STRIKE>加删除线。

4．HTML文字元素

- <CODE></CODE>程序代码。
- <KBD></KBD>键盘字。
- <SAMP></SAMP>范例。
- <VAR></VAR>变量。
- <BLOCKQUOTE></BLOCKQUOTE>向右缩排。
- <DFN></DFN>述语定义。
- <ADDRESS></ADDRESS>地址标记。
- 上标字。
- 下标字。
- <xmp>...</xmp>固定宽度字体（在文件中空白、换行、定位功能有效）。
- <plaintext>...</plaintext>固定宽度字体（不执行标记符号）。
- <listing>...</listing>固定宽度小字体。
- ...字体颜色。
- ...最小字体。
- ...无限增大。

5．HTML段落元素

- 创建一个段落。
- 将段落按左、中、右对齐。
-
定义新行。
- <blockquote></blockquote>从两边缩进文本。

- <dl></dl>定义列表。
- <dt>放在每个定义术语词前。
- <dd>放在每个定义之前。
- 创建一个标有数字的列表。
- 创建一个标有圆点的列表。
- 放在每个列表项之前，若在
之间则每个列表项加上一个数字， 若在
之间则每个列表项加上一个圆点。
- 用来排版大块 HTML 段落，也用于格式化表。
- <MENU>选项清单。
- <DIR>目录清单。
- <nobr></nobr>强行不换行。
- <hr size='9' width='80%' color='ff0000'>水平线（设定宽度）。
- 水平居中。

6．HTML链接元素

- 创建超文本链接。
- 创建自动发送电子邮件的链接。
- 创建位于文档内部的书签。
- 创建指向位于文档内部书签的链接。
- <BASE>文档中不能被该站点辨识的其他所有链接源的 URL。
- < LINK>定义一个链接和源之间的相互关系。

3.2　桩功之二：DIV+CSS 网页布局

上节主要介绍了 HTML 语言和它的特点，以及如何认识和简单地运用 HTML 语言制作网页。我们了解到，HTML 语言是一种超文本标记语言，它具有简单性、可扩展性以及与平台无关性等优点，主要用来显示网页的内容和结果。那么如何将网页的内容和结果更加美观、有条理、有层次地显示出来，给用户带来更好的体验呢？接下来的这一节中我们将介绍 CSS，它就可以实现这种效果。

3.2.1　什么是 CSS

CSS 是英语 Cascading Style Sheets 的缩写，中文译为层叠样式表，是用于控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。CSS 随着时间也在不断升级，目前最新版本为 CSS 3。相对于传统 HTML 的表现而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，拥有对网页对象盒模型的能力，并能够进行初步交互设计，是目前基于文本展示最优秀的表现设计语言。

3.2.2　Web 标准的构成和布局

一个标准的 Web 构成就是：表现、结构和行为这三个要素，如图 3-2 所示。

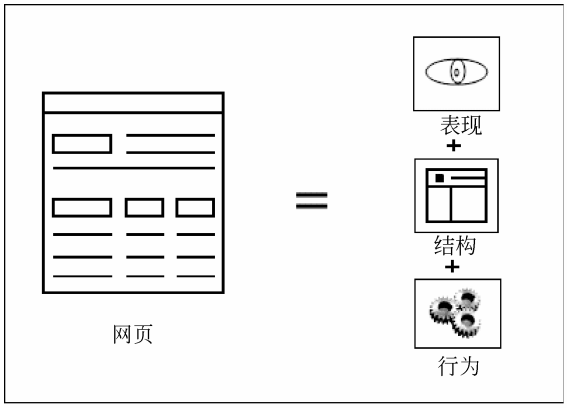

图 3-2　Web 标准构成

表现：用于对已经被结构化的信息进行显示上的修饰，包括版式、颜色、大小等。主要技术就是 CSS 样式表，目前用的最多的是版本 2.0，最新版本是 3.0。

结构：用来对网页中的信息进行整理与分类，常用的技术有：HTML、XHTML 和 XML。

行为：是指对整个文档内部的一个模型进行定义及交互行为的编写。主要技术有：DOM（文档对象模型）、JavaScript 脚本语言和 Ajax 等。

Web 标准的核心目的就是将表现、结构和内容分离开来，这样做的好处是：

- 高效率的开发与简单维护。
- 信息跨平台的可用性。
- 降低服务器成本；加快网页解析速度。
- 更良好的用户体验。

CSS 的出现，正好解决了这个问题，从 CSS 2.0 开始真正意义上实现了这个标准，将表现和内容才算是真正意义上的分开了。

3.2.3　传统布局与 CSS 布局

这里说的传统布局就是 Table 布局，它利用了 HTML 的 Table 元素所具有的零边框特性，因此可以知道 Table 布局的核心是：设计一个能满足版式要求的表格结构，将内容装入每个单元格中，间距及空格使用透明 gif 图片实现，最终的结构是一个复杂的表格（有时候会出现多次嵌套）。显然，这样不利于设计和修改。

图 3-3 是一个传统 Table 布局的示意图。

从图 3-3 可以看出，结构非常复杂，出现了多个表格的嵌套，并且出现了大量文件在网页上，如此势必导致浏览器对网页代码的解析非常慢。图 3-4 是一个 DIV+CSS 布局示意图。

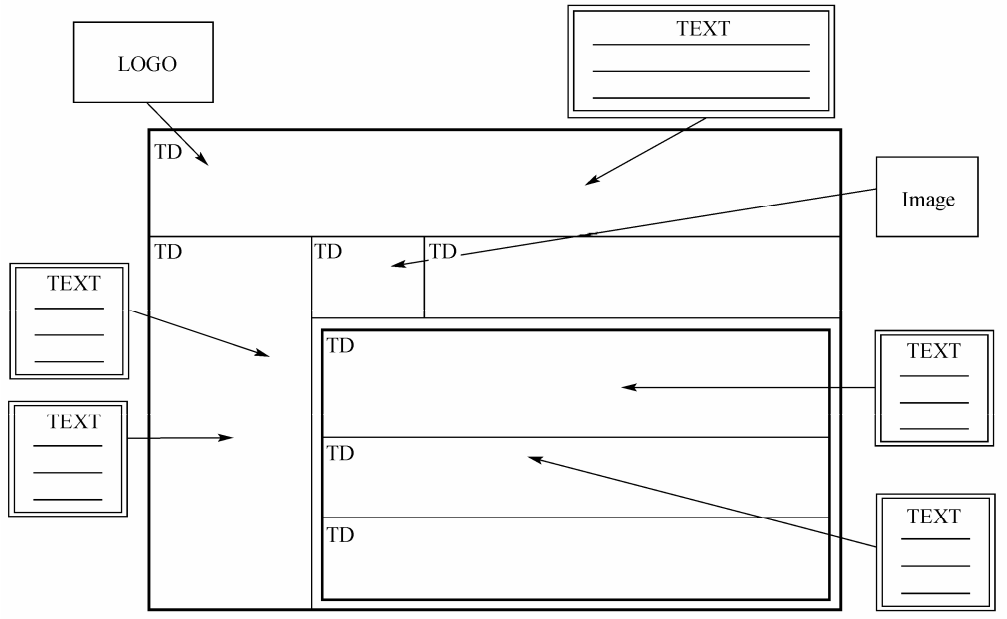

图 3-3　传统 Table 布局示意图

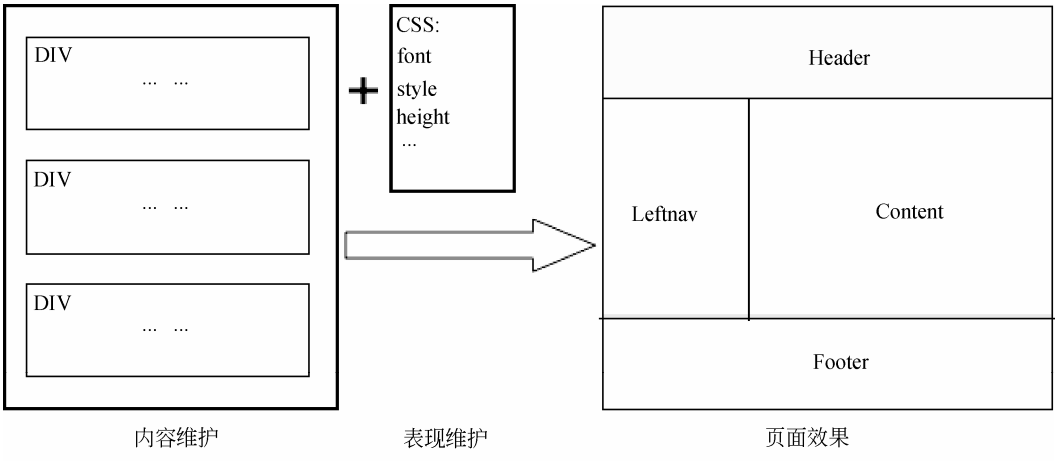

图 3-4　DIV+CSS 布局示意图

从图 3-4 中可以看到，CSS 单独地设置字体、格式、比重等网页元素的属性，而 DIV 单独对网页的结构布局做控制，这样就真正做到了表现与内容的完全分离，使代码的可读性提高了。并且这些样式还可以重复用到不同的页面，复用性增强了。

3.2.4　CSS 布局实例

对 Java Web 标准和 CSS 的布局特点了解后，下面我们来看一个 CSS 布局实例。首先在电脑上创建一个文本文档，然后打开，将下面的代码输入到文本文档。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http:
```

```
//www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS 实例测试</title>
<link href="css/style1.css" rel="stylesheet" type="text/css"  />
                                                    <!--引入 CSS 布局-->
</head>
<body>
<div id="container">                                <!--定义一个容器-->
<div id="header">头部</div>                         <!--定义头部-->
<div id="content">主体</div>                        <!--定义中间部分-->
<div id="footer">尾部</div>                         <!--定义尾部-->
</div>
</body>
</html>
```

将上述代码输入文本文档后，选择“文件”菜单下的“另存为”命令，命名为 css.html，会得到提示是否修改文件扩展名称，单击“是”，就保存成 html 文档了。接下来就是引入 CSS 样式了。

操作步骤和上面的一样，同样新建一个文本文档，输入如下代码。

```
* {
  //对整个布局的整体查询
    font-family: Arial,  Helvetica,  sans-serif,  "宋体";
    font-size: 12px;
    margin: 0px;
    text-align:center;
}
#container {
  //查询网页的 ID 为 container，并设置元素值
    width: 810px;
    margin:auto;
    background:#CCCCCC;
}
#header {
  //查询网页的 ID 为 header，并设置元素值
    height: 100px;
    width: 800px;
    padding:5px;
    background-color: #6699FF;
}
#content {
//查询网页的 ID 为 content，并设置元素值
    height: 400px;
    width: 800px;
    padding:5px;
    background-color#FF9900;
}
#footer {
//查询网页的 ID 为 foot，并设置元素值
    width: 800px;
    height: 50px;
    padding:5px;
  background-color: #6699FF;
}
```

然后单击“另存为”命令，保存为 style1.css，并在刚才的 html 文档同一个目录下面新建一个文件夹，命名为 css，用来专门存放 css 文件，然后将 style1.css 放到里面。

可能您看不懂代码，没有关系，我们可以先看一下效果，然后再解释代码到底是什么含义。接下来用浏览器打开 css.html，效果如图 3-5 所示。

图 3-5　CSS 实例效果

3.2.5　CSS 语法基础

通过上面的 CSS 实例代码，我只是想告诉大家 CSS 可以单独地作为文件直接被引用到网页中，对于代码中一些标签和属性可能大家还不知道是什么含义，为什么要这样写，这个不要着急，接下来将介绍 CSS 的语法基础，学过之后您再回过头来读上面的代码，就会明白到底要表达的是什么含义了。

CSS 语法结构：选择符{ 属性 1：值 1；属性 2：值 2…… } ，例如下面这段代码。

```
body {
//设置网页主要内容的字体和对齐方式
 font-size:12px;
 text-align:center;
}
```

这段代码中，font-size:12px 表示网页中 body 内的字体大小为 12px，text-align:center 表示文字居中显示。

参数说明：属性和属性值之间用冒号（:）隔开，定义多个属性时，属性之间用分号（;）

隔开。

1．CSS中常见的3种选择器

需要说明的是，在 CSS 中主要通过查询选择器来找到网页中需要设置样式的位置，常见的选择器有 3 种，分别为：标签选择器、类别选择器和 ID 选择器。下面分别介绍一下各自的特点和用法。

（1）标签选择器。指以网页中已有的标签名作为名称的选择器，几乎所有的 HTML 标签均可用作该类选择器。如：body{ }、p{ }、h1{ } 等等，都可以作为标签选择器。具体结构如图 3-6 所示。

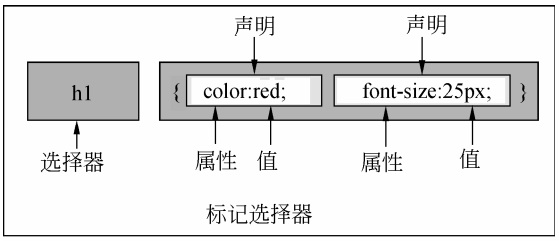

图 3-6　标签选择器

（2）类别选择器。属于用户自定义名称的选择器，可以在 HTML 标签中用 class=" "为相应标签指派样式。可理解为一类。

特点是：可以重复使用，在网页中只要定义的标签中的 class 类别相同，就可套用相同的属性和样式。具体结构如图 3-7 所示。

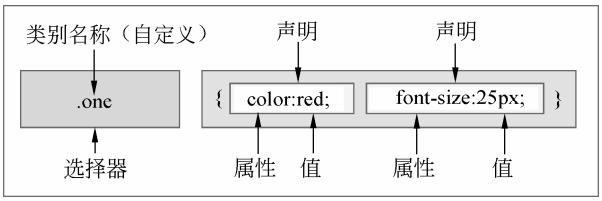

图 3-7　类别选择器

（3）ID 选择器。属于用户自定义名称的选择器，基于 DOM 文档对象模型原理出现的选择器，可以在 XHTML 标签中用 id=""为相应标签指派样式，可理解为一个标识。具体结构如图 3-8 所示。

特点是：在网页中，每个 ID 名称只能使用一次。

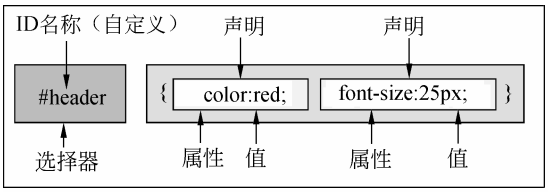

图 3-8　ID 选择器

2．CSS中的3种声明

我们从上面看到，选择器后面是声明，声明又包含属性和属性值。其中声明又分为：集体声明、嵌套声明和混合声明 3 种。

（1）集体声明：属于并列关系，对并列的所有的标签、ID 和类别都使用相同的属性和属性值。这时可以把相同属性和值的选择符组合起来书写，用逗号将选择符分开，这样可以减少样式重复定义。如下：

```
<style type="text/css" >
body，td，th，#header，.one{color:blue;font-size:12px;}
</style>
```

说明：这个样式表示所有列在前面的标签、ID 和类的字体颜色都是 blue，字体大小都是 12px。

（2）嵌套声明：属于从属关系，如下所示。

```
<style type="text/css" >
P h1{color:blue;font-size:14px;}
</style>
```

说明：属于段落元素标题 h1 中的字体颜色为 blue，字体大小为 4px。

（3）混合声明：属于并列及从属关系，如下所示。

```
<style type="text/css" >
P H1，#header ul{color:blue;font-size:12px;}
</style>
```

说明：属于段落元素中标题 h1 中的字体和属于 id 为 header 的 ul 的字体颜色都为 blue，大小都为 12px。

通过上述介绍之后，再回过头来看上一小节的 CSS 实例代码，是不是可以看懂到底是什么意思了。没错，在 style1.css 中定义了一个全局选择器。

```
* {
  //定义字体类型
    font-family: Arial， Helvetica， sans-serif， "宋体";
  //定义字体大小
    font-size: 12px;
  //定义页边距
    margin: 0px;
  //定义文字居中显示
    text-align:center;
}
```

在这个选择器中定义了字体类型（font-family）、字体大小（font-size:）、页边距（margin）和文字显示方式（text-align），往下看是 4 个 ID 选择器。

```
#container {
 //定义容器宽度、页边距和背景颜色
```

```
    width: 810px;
    margin:auto;
    background:#CCCCCC;
}
#header {
 //定义 header 容器的高、宽、边距和背景
    height: 100px;
    width: 800px;
    padding:5px;
    background-color: #6699FF;
}
#content {
 //定义内容容器 content 的高度、宽度、边距和背景颜色
    height: 400px;
    width: 800px;
    padding:5px;
    background-color#FF9900;
}
#footer {
  //定义底部容器的高度、宽度、边距和背景
    width: 800px;
    height: 50px;
    padding:5px;
  background-color: #6699FF;
}
```

这 4 个 ID 选择器中，在 id="container"的选择器中定义了它的宽、页边距和背景颜色，在 id 为 header、content 和 footer 的选择器中分别都定义了每个容器的高、宽、与边界的距离和背景颜色。

我们还可以看到在 HTML 文档中 CSS 文件是用下面这句话加载上的：

```
<link href="css/style1.css" rel="stylesheet" type="text/css"  />
```

此链接为外部样式表，您也许会问外部样式表是什么，如何在网页中运用 CSS 知识，不用着急，在接下来的小节中，将重点介绍如何应用 CSS 到网页中。

3.2.6　如何应用 CSS 到网页中

同样先来看如下代码。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
                              <!--说明：该段指定文档类型为 Transitional-->
<html xmlns="http://www.w3.org/1999/xhtml">
                              <!--说明：该句确定名字空间，在 xml 中用到-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
                              <!--说明：该句声明编码语言为简体中文-->
<title>测试页</title>
<link href="css/style1.css" rel="stylesheet" type="text/css" />
                              <!--此为链接（外部）样式表-->
<style type="text/css">
<!--CSS 定义显示字体大小
body，td，th {font-size: 12px;}
p{font-size: 24px;color:#FFFF00;}
```

```
-->
</style>
</head>
<body>
<div id="container">
<div id="header"> <p>头部</p> </div>
<div id="content">
<p style="color:#FF0000;font-size:36px">主体</p>
                                          <!--此为行内样式表 -->
</div>
<div id="footer"> <p>尾部</p> </div>
</div>
</body>
</html>
```

可以看到 CSS 编码可以多种方式灵活地应用到我们所设计的 XHTML 页面之中，选择方式可根据我们对设计的不同表现手段来制定，一般按 CSS 代码位置可分为以下 4 种。

（1）行内样式表：如果只有一个 HTML 标签需要设定样式，则可在该标签内加上属性。

```
<p style="color:#FF0000;font-size:36px">主题</p>  <!-- 此为行内样式表 -->
```

说明：只对段内“主题”一词设置颜色为#FF0000 和大小为 36px。

（2）内部样式表：在网页内部设置样式格式。

```
<style type="text/css">
<!--
body, td, th {font-size: 12px;}
p{font-size: 24px;color:#FFFF00;}
-->
</style>
```

说明：它的作用是全局定义 body 标签，td 标记和 th 标记内的字体全为 12px；段内 p 标签内的字体为 24px，颜色为#FFFF00。

注意：有些低版本的浏览器不能识别 style 标记，这意味着低版本的浏览器会忽略 style 标记里的内容，并把 style 标记里的内容以文本直接显示到页面上。为了避免这样的情况发生，我们用加 HTML 注释的方式（<!-- 注释 -->）隐藏内容而不让它显示。

（3）外部样式表。

```
<!-- 此为链接（外部）样式表 -->
<link href="css/style1.css" rel="stylesheet" type="text/css" media="all"/>
```

说明：浏览器从 style1.css 文件中以文档格式读出定义的样式表。rel="stylesheet"是指在页面中使用这个外部的样式表。type="text/css"是指文件的类型是样式表文本。href="mystyle.css"是文件所在的位置。media 是选择媒体类型，这些媒体包括：屏幕、纸张、语音合成设备以及盲文阅读设备等。

注意：一个外部样式表文件可以应用于多个页面。当改变这个样式表文件时，所有页面的样式都随之而改变。在制作大量相同样式页面的网站时，非常有用，不仅减少

了重复的工作量，而且有利于以后的修改、编辑，浏览时也减少了重复下载代码，所以建议在网页中使用这个样式引入。

（4）导入式样式表。

在<style></style>之间加入 @import url(css/style2.css)，这种方式称为导入式样式表。

```
<style type="text/css" media="all">@import url( css/style.css );</style>
```

注意：其中的"@import"命令用于输入样式表。而"@import"命令对 netscape 4.0 版本浏览器是无效的。也就是说，当您希望某些效果在 netscape 4.0 浏览器中隐藏，在 4.0 以上版本或其他浏览器中又显示的时候，可以采用"@import"命令调用样式表。

3.2.7　CSS 开发与调试环境

以上章节中我们主要用文本文档对代码进行编写，目的主要是想让大家自己动手把代码输入一遍，以加深初学者对 HTML 和 CSS 的认识。现在开始介绍一些常用的网页制作工具。

1．网页设计与制作的主要工具

其实对于网页的设计和制作，有很多软件可以使用，这些软件可以帮我们更好、更方便地制作和调试网页，下面就介绍 4 种主要的工具。

（1）FrontPage。其优点可以概括如下。

- 文件视图最优化。
- 自动缩略图。
- 广告横幅管理者。
- 层叠样式表单支持。

（2）Dreamweaver 8.0。其优点可以概括如下。

- 不生成冗余代码。
- 方便的工作模式。
- 强大的动态页面支持。
- 可准确对层进行定位。
- 操作简便。
- 与 Flash 和 Fireworks 的紧密结合。

（3）Flash MX Professional 8.0。其优点可以概括如下。

- 时间轴特效。
- 行为。
- 创作环境中的辅助支持。
- 更新的模板。
- 集成的帮助系统。
- 拼写检查器。

- 文档选项卡。
- 开始页。

（4）Fireworks MX 8.0。其优点可以概括如下。

- 是一款用来设计网页图形的应用程序。
- 是一个创建、编辑和优化网页图形的多功能应用程序。

2．Dreamweaver 8.0的安装和使用

在本书中我们主要以 Dreamweaver 8.0 作为网页设计工具，Dreamweaver 是一款专业的网页制作软件，相信对于它大家都不会太陌生。可以说，它是第一套针对专业网页设计师特别开发的视觉化网页开发工具，利用它可以轻而易举地制作出跨越平台限制和跨越浏览器限制的充满动感的网页。下面将重点介绍 Dreamweaver 8.0 是如何使用，以及如何利用 Dreamweaver 8.0 对网页 CSS 进行编写和调试。

1）Dreamweaver 8.0 的安装

首先下载 Dreamweaver 8.0，下载后双击安装包，就会弹出界面如图 3-9 所示。

图 3-9　Dreamweaver 安装欢迎界面

接下来就非常简单了，直接单击“下一步”按钮，进入许可证协议界面，选择“我接受该许可证协议中的条款”，然后单击“下一步”按钮，就进入安装路径的选择界面，默认直接安装到 C 盘，如图 3-10 所示。

当然您也可以根据自己的需要更改安装路径，更改后选择要创建的快捷方式，然后单击“下一步”按钮。此时看到很多候选的文件类型，选择您需要 Dreamweaver 编辑的文件类型，建议大家全部选择，这样就可以用 Dreamweaver 编辑各种文件了，选择好文件类型后单击“下一步”按钮，这时弹出窗口提示“已经做好安装程序的准备”，接下来直接单击“安装”按钮就可以了。安装需要几分钟，稍等几分钟不出错的话，就会看到安装完成界面，如图 3-11 所示。

图 3-10　Dreamweaver 设置安装路径界面

图 3-11　Dreamweaver 安装完成界面

2）Dreamweaver 8.0 的使用

其实上面这些步骤都非常简单，只要您会下载软件，会单击鼠标，直接按照提示单击“下一步”安装就可以了。这里需要提醒大家注意的是，在首次启动 Dreamweaver 8 时会出现一个“工作区设置”对话框，在对话框左侧是 Dreamweaver 8 的设计视图，右侧是 Dreamweaver 8 的代码视图。Dreamweaver 8 设计视图布局提供了一个将全部元素置于一个窗口中的集成布局。我们可以选择面向设计者的设计视图布局，也可以选择代码视图布局。这里我们先选择“设计器”，如图 3-12 所示。

单击“确定”按钮后，在 Dreamweaver 8 中首先将显示一个起始页，可以勾选这个窗口下面的“不再显示此对话框”来隐藏它。如图 3-13 所示。

图 3-13 中可以看到其中包括“打开最近项目”、“创建新项目”和“从范例创建”3 个方便实用的功能选择区，建议大家保留此界面。

我们选择“文件”→“新建”命令，就进入“新建文档”界面，如图 3-14 所示。

图 3-12　Dreamweaver 工作区设置界面

图 3-13　Dreamweaver 8 起始页

此时选择“基本页”→“HTML”选项，将“文档类型”设置为 XHTML 1.0 Transitional，然后单击“创建”按钮，就会创建一个 HTML 页面，并且 DTD 类型为 XHTML 1.0 Transitional。

这里需要说明一下，一个标准的 XHTML 文档，必须以 doctype 标签作为开始，doctype 用于定义文档类型。对于 XHTML 而言，可以选择以下 3 种不同的 XHTML 文档类型。

- ❑ Strict 类型：严格类型，文档中不允许使用任何表现样式的标识和属性。
- ❑ Frameset 类型：框架页类型，网页使用框架结构时，声明此类型。
- ❑ Transitional 类型：过渡类型，浏览器对 XHTML 的解析较为宽松。

图 3-14　新建文档界面

小经验：对于这 3 种类型，由于过渡类型对 XHTML 的解析比较宽松，对于其他两种都要求比较严格，所以建议刚开始时将网页的 DTD 设置为 Transitional 类型比较好。

当创建 HTML 文档之后，就可以对 HTML 文档进行设计了，在其中可以按照前面所讲的方法插入 CSS 样式和布局文件，自己动手加入标签和属性就可以了。当然也可以通过 Dreamweaver 图形化界面设置页面属性，如图 3-15 所示。

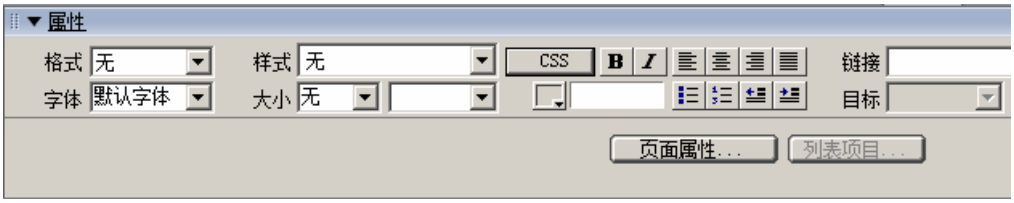

图 3-15　HTML 页面属性设置

在这里可以设置页面格式、加入 CSS 样式、字体类型、字体大小、字体颜色、链接文件和项目列表等。单击“页面属性”按钮就可以对页面属性的外观、链接、标题、编码和跟踪图像等类别进行设置，如图 3-16 所示。

通过上面的设置就可以制作出一个简单的 HTML 文档了。单击“应用”按钮之后就可以看到实现的具体效果了，如果是想要的效果，就可以单击“确定”按钮，得到自己想要的 HTML 文档，否则还可以继续对页面属性进行设置，直到调试出满意的页面效果。当然我们可以在主界面中选择代码，此时就可以看到实现该网页 HTML 的代码是什么样的了。一般我们使用拆分模式，那样既可以看到代码也可以随时看到界面效果。

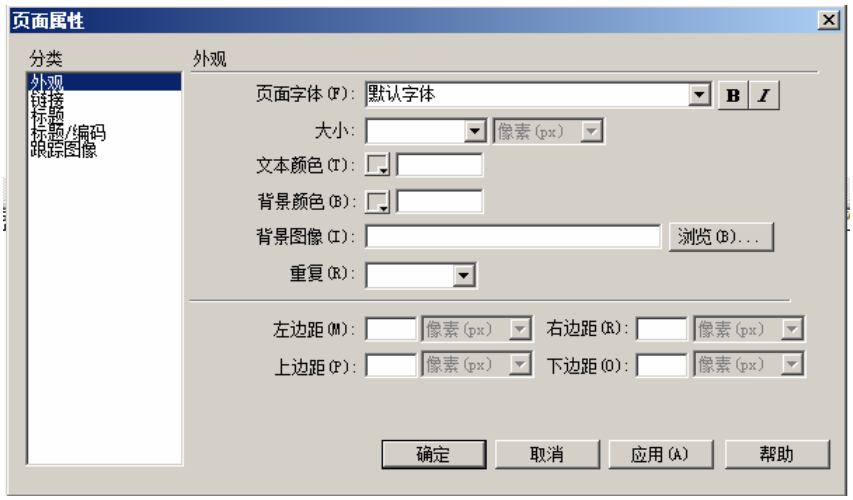

图 3-16　页面属性的具体分类设置

3.2.8　CSS 样式表

我们知道要想做出精美的网页一定要用到 CSS。CSS 样式表是一系列格式规则，它们控制网页内容的外观。CSS 样式使您可以控制许多仅使用 HTML 无法控制的属性。

在 Dreamweaver 中创建 CSS 时，可以选择“定义在该文档”和“新建样式表文件”这两种方式之一创建 CSS 样式表。“定义在该文档”只作用在当前文档；“新建样式表文件”创建出一个独立的外部 CSS 样式表文件，多个文档可以链接到外部 CSS 样式表文件。

1．外部CSS样式表

在创建 CSS 时，可以根据个人喜好，选择一种引入 CSS 的方式。如果希望用相同的样式控制多个文档的格式，使用“外部 CSS 样式表”是最简单的方法。

（1）在当前界面中选择“文本”→“CSS 样式表”→“新建”，打开对话框如图 3-17 所示。

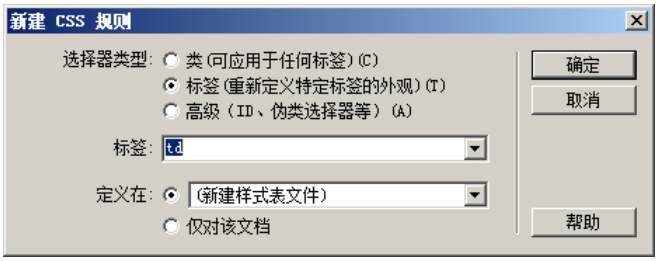

图 3-17　新建 CSS 规则

选中“标签”，填写标签名称，在“定义在”中选择“新建样式表文件”，单击“确定”按钮。

（2）页面跳转到“保存样式表文件为”对话框，这里没有建站点，所以选择文件来自

“文件系统”，保存在“B:\Java web 开发入门\第三章\CSS”目录下，文件名称定义为 out_css，其他为默认值，单击“保存”按钮，如图 3-18 所示。

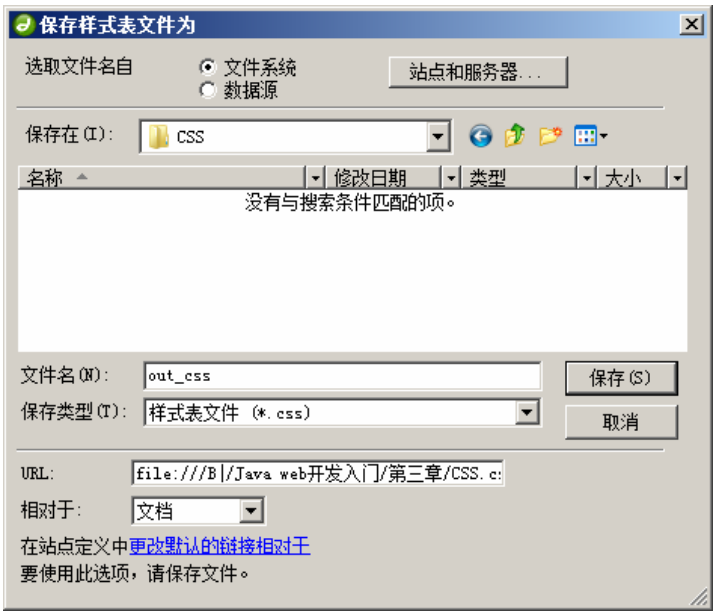

图 3-18　保存 CSS 文档

（3）对刚创建的标签定义规则，包括“类型”、“背景”、“区块”、“方框”、“边框”、“列表”、“定位”和“扩展”等分类。如图 3-19 所示定义了类型的字体类型、大小、颜色、大小写、样式和修饰等。

图 3-19　对标签的 CSS 规则定义

（4）单击“确定”按钮，这样就新生成了一个“外部 CSS 样式表”，名称为 out_css.css。如图 3-20 所示。

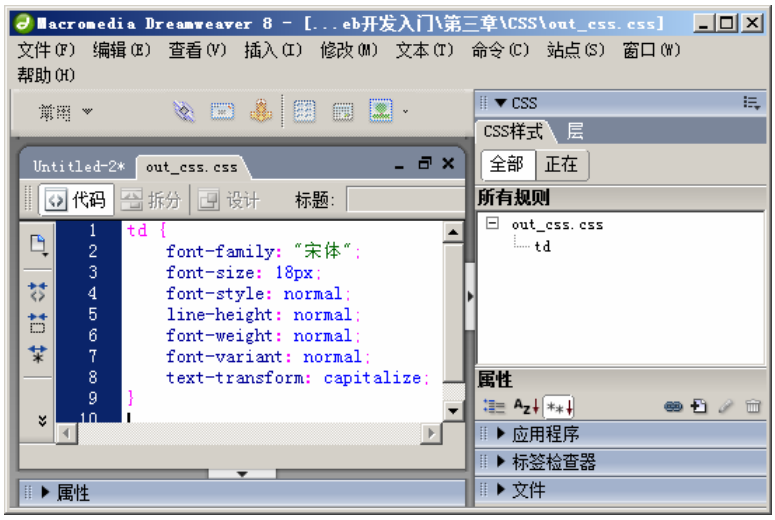

图 3-20　外部 CSS 样式表示例

说明：这个样式表是独立的，对于所有的网页都可以应用，引用方式是使用如下代码。

```
<link href="file:///B|/Java web 开发入门/第三章/CSS/out_css.css" rel=
"stylesheet" type="text/css">
```

知识积累：

使用外部 CSS 的优点是，只要修改外部的 CSS 样式表文件，所有链接到该样式表文件的文档格式都会自动发生改变。外部引入的简明步骤如下：

打开一个网页文档→打开 CSS 样式面板→单击“附加样式表”按钮→单击“浏览”按钮→选择需要的外部 CSS 样式表文件→单击“确定”按钮。

2．内部样式表（仅对该文档的CSS样式表）

如果喜欢步骤简单，或者只有一个页面需要应用某个 CSS 样式表，那就使用“仅对该文档”的 CSS 样式表，如图 3-21 所示。

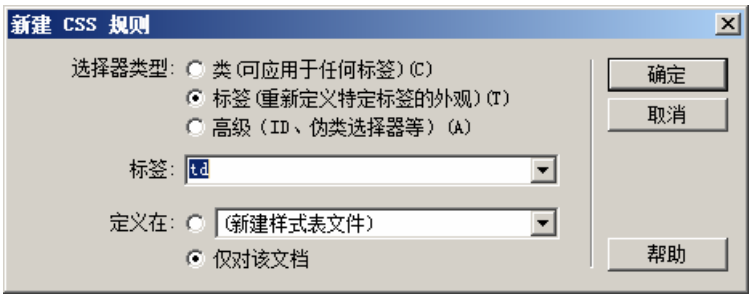

图 3-21　内部样式表新建规则

选择“仅对该文档”，单击“确定”按钮后，接下来和上面的步骤一样对标签定义规则，不同的是定义好 CSS 后，CSS 就直接出现在文档中间了。这种 CSS 样式的优点是创建好了就直接应用到当前文档了。

通过前面的介绍，CSS 的重要性就不多说了，当然如果要说清楚 CSS 就算写成一本大书也不够。本节的目的，是想通过本节的介绍，掌握使用 CSS 最基本和最重要的应用。至于 CSS 的其他知识，我们在接下来的章节中会不断讲到，只要大家用心积累，跟着我讲的走，其实 Java Web 入门就是这么简单。

3.3　桩功之三：JavaScript 功能

在上面的章节中我们主要讲到了 HTML 和 CSS 的有关知识，通过大家认真地学习，相信大家已经对 HTML 和 CSS 有些熟悉了吧！通过 HTML 和 CSS 已经可以制作出非常漂亮的界面了，但是这些界面缺少生命力，只是静态的，无法与用户做到很好的交流。

下面我们将学习如何将网页实现各种很好的动态效果，给用户更好的视觉享受和沟通效果，那就不得不讲 JavaScript 了。如果把 HTML 比作是武术的桩功，那 HTML 只能说是静静地站着的那种常规的桩功，而 JavaScript 却在常规桩功的基础上添加了难点，增加了重量，就像我们常见的头顶香炉、脚踩梅花等桩功。

所以 JavaScript 既是基础，也是难点，功能灵活，变幻莫测。说得这么神乎其神的，是不是真的很想知道 JavaScript 到底是什么，那我们就赶快开始本节的学习吧！

3.3.1　什么是 JavaScript

JavaScript 是由 Netscape 公司开发并随 Navigator 导航者一起发布的，介于 Java 与 HTML 之间的一种基于对象（Object）和事件驱动（Event Driven）并具有安全性能的脚本语言。

这样的解释是不是感觉就像没说似的，别说对于一个新手而言，就算是研究一两年的专业技术人员，对于什么是基于对象，什么是事件驱动，这些概念我想也未必可以解释和理解得那么清楚。所以这些对于我们新手而言刚开始没有必要了解，我们只有知道“JavaScript 是一种能让您的网页更加生动活泼的程序设计语言，也是目前网页设计中最容易学又最方便的语言”就可以了。至于那些概念我们以后就会慢慢解释的。

您可以利用 JavaScript 轻易地做出亲切的欢迎讯息、漂亮的数字钟、有广告效果的跑马灯及简易的选举，还可以显示浏览器停留的时间等等效果，让这些特殊效果提高网页的可观性和交互性。下面简单介绍一下 JavaScript 的语言特点。

1．JavaScript语言特点

JavaScript 的出现，使得信息和用户之间不仅只是一种显示和浏览的关系，而是实现了一种实时的、动态的、可交式的表达能力。从而基于 CGI 静态的 HTML 页面将被可提供动态实时信息，并对客户操作进行反应的 Web 页面所取代。JavaScript 脚本正是满足这种需求而产生的语言。它深受广泛用户的喜爱和欢迎。它是众多脚本语言中较为优秀的一种，通过嵌入在标准的 HTML 语言中实现功能，它的出现弥补了 HTML 语言的缺陷，是 Java 与 HTML 折衷的选择。概括起来，JavaScript 语言的基本特点有 6 个。

（1）脚本编写语言。JavaScript 是一种脚本语言，它采用小程序段的方式实现编程。像其他脚本语言一样，JavaScript 同样也是一种解释性语言，它提供了一个简单的开发过程。

它的基本结构形式与 C、C++、VB 和 Delphi 十分类似。但它不像这些语言一样，需

要先编译，而是在程序运行过程中被逐行地解释。它与 HTML 标识结合在一起，从而方便用户的使用操作。

（2）基于对象的语言。JavaScript 是一种基于对象的语言，同时也可以看作是面向对象的。这意味着它能运用自己已经创建的对象。因此，许多功能可以来自于脚本环境中对象的方法与脚本的相互作用。

注意：对于对象的概念如果不熟的话，可以复习一下我们第 1 章讲的 Java 基础知识，也可以自己在网上或者书店买本 Java 基础书籍学习一下。由于本书是 Java Web 开发，默认您已经对 Java 基础知识有所了解了，所以对 Java 基础讲的并不多。

（3）简单性。JavaScript 的简单性主要体现在：首先它是基于 Java 基本语句和控制流之上的，设计简单紧凑，只要您懂得 Java 语法就非常容易上手。其次它的变量类型是采用弱类型，并未使用严格的数据类型，所以设置变量就不用区分是什么类型了。

（4）安全性。JavaScript 的安全性主要体现在：首先不允许访问本地的硬盘，并不能将数据存入到服务器上。其次不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互。这样就能有效地防止数据的丢失。

（5）动态性。JavaScript 的动态性主要体现在：它可以直接对用户或客户输入做出响应，无须经过 Web 服务程序。它对用户的反映响应，是采用事件驱动的方式进行的。

知识扩展：

什么是事件驱动？在主页（Home Page）中执行了某种操作所产生的动作，就称为“事件（Event）”。比如按下鼠标、移动窗口、选择菜单等都可以视为事件。当事件发生后，可能会引起相应的事件响应。

（6）跨平台性。JavaScript 是依赖于浏览器本身的，与操作环境无关，只要能运行浏览器的计算机，并支持 JavaScript 的浏览器就可正确执行。从而实现了“一次编写，到处运行”的理念。

2．JavaScript执行原理

（1）客户端请求某个网页。即我们在上网时在地址栏中输入某个网址，浏览器接收到网址之后，向远程 Web 服务器提出请求。

（2）Web 服务器响应请求。Web 服务器找到请求的页面，并将整个页面包含 JavaScript 的脚本代码作为响应内容，发送回客户端机器。

（3）客户端浏览器解释并执行带脚本的代码。客户端浏览器打开回应的网页文件内容，从上往下逐行读取并显示其中的 html 或者脚本代码，脚本是从服务器端下载到客户端，然后在客户端进行的，即不占用服务器端的资源。因此通过客户端脚本，客户端分担了服务器的任务，大大地减轻了服务器的压力，从而间接地提升了服务器的性能。

实际上 JavaScript 最杰出之处在于可以用很小的程序做大量的事。无须有高性能的电脑，软件仅需一个字处理软件及一个浏览器，无须 Web 服务器通道，通过自己的电脑即可完成所有的事情。

综上所述，JavaScript 是一种新的描述语言，它可以被嵌入到 HTML 的文件之中。JavaScript 语言可以做到回应使用者的需求事件（如：form 的输入），而不用任何的网路来回传输资料，所以当一位使用者输入一项资料时，它不用经过传给服务器端（server）处

理，再传回来的过程，而直接可以被客户端（client）的应用程序所处理。

3.3.2　JavaScript 与 Java 的区别

在第 1 章中我们介绍了 Java 语言的特点，在上文又介绍了 JavaScript 脚本语言的特点，乍一看还真是很相似，那么我们可能会想它们有什么联系和区别的。其实 Java 和 JavaScript 语言虽然有很多联系，但是到底并不是一种语言，其目的和原理都不一样，Java 是一种比 JavaScript 更复杂许多的程序设计语言，而 JavaScript 则是相当容易了解的脚本语言。JavaScript 创作者可以不那么注重程序技巧，所以许多 Java 的特性在 JavaScript 中并不支持。主要异同如下。

（1）创造公司和开发目的。Java 是 SUN 公司推出的新一代面向对象的程序设计语言，特别适合于 Internet 应用程序开发；而 JavaScript 是 Netscape 公司的产品，其目的是为了扩展 Netscape Navigator 功能，而开发的一种可以嵌入 Web 页面中的基于对象和事件驱动的解释性语言。

（2）基于对象和面向对象。Java 是一种真正的面向对象的语言，即使是开发简单的程序，也必须设计对象。JavaScript 是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object Based）和事件驱动（Event Driver）的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用。

（3）解释和编译。两种语言在浏览器中所执行的方式不一样。Java 的源代码在传递到客户端执行之前，必须经过编译，因而客户端上必须具有相应平台上的仿真器或解释器，它可以通过编译器或解释器实现独立于某个特定的平台编译代码的束缚。JavaScript 是一种解释性端程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户端由浏览器解释执行。

（4）强变量和弱变量。两种语言所采取的变量是不一样的。Java 采用强类型变量检查，即所有变量在编译之前必须作声明，如下所示。

```
int x;          //定义一个整型变量
String y;       //定义一个字符串变量
char a;         //定义一个字符变量
float 1.5f      //定义一个浮点型变量
x=10;           //将整型变量 x 初始化为 10
y="abc";        //对字符串变量 y 进行初始化操作
```

其中 x=10 说明是一个普通整型数，y="abc"说明 y 是一个字符串类型。a 是一个字符类型，1.5f 是一个浮点类型数据。

JavaScript 中变量声明，采用弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型，如下所示。

```
Var x;          //定义一个变量 x
Var  y;         //定义一个变量 y
x=10;           //初始化 x 为 10
y="abc";        //初始化 y 为"abc"
```

前者说明 x 为数值型变量，而后者说明 y 为字符型变量。

（5）代码格式不一样。Java 是一种与 HTML 无关的格式，必须通过像 HTML 中引用

外媒体那样进行装载，其代码以字节代码的形式保存在独立的文档中。JavaScript 的代码是一种文本字符格式，可以直接嵌入 HTML 文档中，并且可动态装载。编写 HTML 文档就像编辑文本文件一样方便。

（6）嵌入方式不一样。在 HTML 文档中，两种编程语言的标识不同，JavaScript 使用<Script>　</Script>来标识，如下所示。

```
<html>
<head>
<Script Language ="JavaScript">
//编写 Javascript 功能效果
</Script>
</head>
<body>
网页主题部分
</body>
</html>
```

而 Java 使用<applet>　</applet>来标识。这里就不再举例说明了，读者可以回头查看 Java 基础。

（7）静态联编和动态联编。Java 采用静态联编，即 Java 的对象引用必须在编译时进行，以使编译器能够实现强类型检查。JavaScript 采用动态联编，即 JavaScript 的对象引用在运行时进行检查，如不经编译则无法实现对象引用的检查。

3.3.3　第一个 JavaScript 程序

我们在浏览网页时经常收到提示窗口，这个功能用 JavaScript 就可以实现。那么这个功能用 JavaScript 是怎么实现的呢？代码非常简单，如下所示。

```
<html>
<head>
<Script Language ="JavaScript">
//JavaScript 实现的功能
alert("这是第一个 JavaScript 程序!");
alert("Java web 开发入门就是这么简单!");
alert("今后我将带领大家学习 Java web 知识! ");
</Script>
</head>
</html>
```

保存成 html 网页，用浏览器打开，效果如图 3-22 所示。

单击“确定”按钮，就会显示下一个窗口给出提示警告。效果是不是很酷呀，这么几行简单的 JavaScript 就可以实现我们想要的提示警告窗口。

通过上面的 JavaScript 例子，我们会发现 JavaScript 在网页中使用如下代码表示。

```
<Script Language ="JavaScript">...</Script>
```

在标识<Script Language ="JavaScript">...</Script>之间就可加入 JavaScript 脚本。alert()是 JavaScript 的窗口对象方法，其功能是弹出一个具有“确定”按钮的对话框并显示 alert 后面括号中的字符串。通过<!--……//-->标识，若不认识 JavaScript 代码的浏览器，则所有在其中的标识均被忽略；若使用认识，则执行其结果。

图 3-22　第一个 JavaScript 程序效果

注意：使用注释是一个好的编程习惯，它既可以使其他人读懂你的语言，又可以防止由于浏览器不兼容引起的问题。

3.3.4　JavaScript 程序控制结构

JavaScript 脚本语言包括：控制语句、函数、对象、方法和属性等。从上面的实例可以看到，在代码中只使用了一个 alert()函数就实现提示警告的窗口功能。其实 JavaScript 还有很多功能函数，下面我们就详细地介绍一下 JavaScript 的程序的组织构造。

在第 1 章的 Java 基础部分我们简单介绍了一下什么叫控制语句以及一些简单的控制语句的使用语法。其实在任何一种语言中，包括 C、C++、Java 等都需要程序控制语句，有了这些控制语句才可以把各种对像、函数、方法和属性连接在一块，所以程序控制语句对于编程是必须的，它能使得整个程序逻辑清楚，使之可以顺利地按照一定的规则和方式执行并完成某项具体的功能。而对于 JavaScript 同样有程序控制语句，它的使用语法和 Java 差不多，JavaScript 使用这些控制语句组成了不同的程序结构，实现了各种功能效果。下面就对常见的 JavaScript 程序控制语句和结构进行介绍。

1．判断语句的使用

（1）if 条件语句，基本格式如下。

```
 if(表述式)
{
语句段 1;语句 2;......
}
else  {
语句段 3;语句 4;......
}
```

- 功能：若表达式为 true，则执行语句段 1 和语句 2；否则执行语句段 3 和语句 4。
- 说明：if-else 语句是 JavaScript 中最基本的控制语句，通过它可以改变语句的执

行顺序。表达式中必须使用关系语句来实现判断，它是作为一个布尔值来估算的。它将零和非零的数分别转化成 false 和 true。若 if 后的语句有多行，则必须使用花括号将其括起来。

（2）if 嵌套语句，基本格式如下。

```
if(表达式)语句 1;
 else(表达式)语句 2;
 else if(表达式)语句 3;……
 else 语句 4 ;
```

在这种情况下，每一级的布尔表达式都会被计算，若为真，则执行相应的语句；否则执行 else 后的语句。

（3）switch…case 语句，基本格式如下。

```
switch(变量名称)
{
    case 条件值 1：执行语句 1;break
    case 条件值 2：执行语句 2;break
    …
    default：执行语句 n
}
```

- break 语句作用：帮助跳出该判断语句，不再执行后面的语句。
- default 语句作用：表示默认执行语句，则当所有 case 值都不满足时则执行该语句。

功能：其执行过程是，当变量与 case 条件值相等时，则执行其后所有的语句，并且当碰上下一个 case 时也不再判断相等与否。

2．循环语句的使用

（1）for 循环语句，基本格式如下。

```
for(初始化;条件;增量)
{
语句 1;语句 2;语句 3;......
}
```

- 功能：实现条件循环，当条件成立时，执行语句集，否则跳出循环体。
- 说明：初始化参数告诉循环的开始位置，必须赋予变量的初值。
- 条件：是用于判断循环停止时的条件。若条件满足，则执行循环体，否则跳出。
- 增量：主要定义循环控制变量在每次循环时按什么方式变化。3 个主要语句之间，必须使用分号分隔。

（2）while 循环语句，基本格式 1 如下。

```
while(是否循环的条件)
 {
语句 1;语句 2;语句 3;......
}
```

基本格式 2 如下。

```
do   {
```

```
        条件为真时循环的代码
    } while(是否循环的条件)
```

该语句与 for 语句一样，当条件为真时，重复循环，否则退出循环。

提示：for 语句与 while 语句都是循环语句，使用 for 语句在处理有关数字时更易看懂，也较紧凑；而 while 循环对复杂的语句效果更特别。

3．break和continue语句

与C++和Java语言相同，使用break语句可使得循环从for或while语句中跳出，continue语句使得跳过循环内剩余的语句而进入下一次循环。

3.3.5　JavaScript 函数

函数就是一个功能集合，它为程序设计人员提供了一个丰常方便的能力。通常在进行一个复杂的程序设计时，总是根据所要完成的功能，将程序划分成多个相对独立的模块，每个模块编写成一个函数。从而，使各部分充分独立，任务单一，程序清晰，易懂、易读、易维护，这也是我们软件设计中所追求的“高内聚，低耦合”。JavaScript 函数可以封装那些在程序中可能要多次用到的模块，做到代码复用。并可作为事件驱动的结果而调用的程序，从而实现一个函数把它与事件驱动相关联。这也是与其他语言不一样的地方。

1．JavaScript函数定义

函数由关键字 Function 定义，包含函数名、参数表、函数体和返回值。
函数名：定义自己函数的名字。
参数表：是传递给函数使用或操作的值，其值可以是常量、变量或其他表达式。

```
Function 函数名 (参数,变元)
{
函数体;
Return 表达式;
}
```

说明：当调用函数时，所用变量或字面量均可作为变元传递。通过指定函数名（实参）来调用一个函数。必须使用 Return 将值返回。函数名对大小写是敏感的。

2．函数中的形式参数

在函数的定义中，我们看到函数名后有参数表，这些参数变量可以是一个或者是几个。那么怎样才能确定参数变量的个数呢？在 JavaScript 中可通过 arguments .Length 来检查参数的个数。例如下面代码。

```
Function function_Name(exp1, exp2, exp3, exp4)      //定义一个 4 个参数的函数
Number =function _Name . arguments .length;         //得到函数参数的个数
if (Number>1)                                  //如果个数大于 1，输出第二个参数
document.write(exp2);
```

```
if (Number>2)                          //如果个数大于 2，输出第三个参数
document.write(exp3);
if(Number>3)                           //如果个数大于 3，输出第四个参数
document.write(exp4);
```

3．常见的JavaScript函数

JavaScript 函数一共可分为 5 类：常规函数、数组函数、日期函数、数学函数和字符串函数。

（1）常规函数。JavaScript 常规函数包括以下 9 个。

- alert 函数：显示一个警告对话框，包括一个“确定”按钮。
- confirm 函数：显示一个确认对话框，包括“确定”和“取消”按钮。
- escape 函数：将字符转换成 Unicode 码。
- eval 函数：计算表达式的结果。
- isNaN 函数：测试是（true）否（false）不是一个数字。
- parseFloat 函数：将字符串转换成浮点数字形式。
- parseInt 函数：将符串转换成整数数字形式（可指定几进制）。
- unescape 函数：解码由 escape 函数编码的字符。
- prompt 函数：显示一个输入对话框，提示等待用户输入。例如下面这段代码。

```
<script language="javascript">
   <!--
   alert("输入错误");
   prompt("请输入您的姓名", "姓名");
   confirm("确定否！");
   //-->
</script>
```

（2）数组函数。JavaScript 数组函数包括以下 4 个函数。

- join 函数：转换并连接数组中的所有元素为一个字符串，如下所示。

```
function JoinDemo()
  {
        var a, b;
        a = new Array(0,1,2,3,4);
        b = a.join("-");   //分隔符
        return(b);         //返回的 b=="0-1-2-3-4"
 }
```

- length 函数：返回数组的长度，如下所示。

```
function LengthDemo()
{
 var a, l;
 a = new Array(0,1,2,3,4);
 l = a.length;
 return(l);         //l==5
}
```

- reverse 函数：将数组元素顺序颠倒，如下所示。

```
function ReverseDemo()
{
```

```
  var a, l;
  a = new Array(0,1,2,3,4);
  l = a.reverse();
  return(l);
}
```

- sort 函数：将数组元素重新排序，如下所示。

```
function SortDemo()
       {
         var a, l;
         a = new Array("X" ,"y" ,"d", "Z", "v","m","r");
         l = a.sort();
         return(l);
}
```

（3）日期函数。JavaScript 日期函数包括以下 20 个函数。

- getDate 函数：返回日期的“日”部分，值为 1～31，如下所示。

```
function DateDemo()
     {
       var d, s = "Today's date is: ";
       d = new Date();
       s += (d.getMonth() + 1) + "/";
       s += d.getDate() + "/";
       s += d.getYear();
       return(s);
     }
```

- getDay 函数：返回星期几，值为 0～6，其中 0 表示星期日，1 表示星期一，...，6 表示星期六。例如：

```
function DateDemo()
     {
       var d, day, x, s = "Today is: ";
       var x = new Array("Sunday", "Monday", "Tuesday");
       var x = x.concat("Wednesday","Thursday", "Friday");
       var x = x.concat("Saturday");
       d = new Date();
       day = d.getDay();
       return(s += x[day]);
     }
```

- getHouse 函数：返回日期的“小时”部分，值为 0～23。例如：

```
function TimeDemo()
     {
       var d, s = "The current local time is: ";
       var c = ":";
       d = new Date();
       s += d.getHours() + c;
       s += d.getMinutes() + c;
       s += d.getSeconds() + c;
       s += d.getMilliseconds();
       return(s);
     }
```

- getMinutes 函数：返回日期的“分钟”部分，值为 0～59。见上例。
- getMonth 函数：返回日期的“月”部分，值为 0～11。其中 0 表示 1 月，2 表示 3

月，...，11 表示 12 月。见前面的例子。

- getSeconds 函数：返回日期的“秒”部分，值为 0～59。见前面的例子。
- getTime 函数：返回系统时间。例如：

```
function GetTimeTest()
     {
       var d, s, t;
       var MinMilli = 1000 * 60;
       var HrMilli = MinMilli * 60;
       var DyMilli = HrMilli * 24;
       d = new Date();
       t = d.getTime();
       s = "It's been "
       s += Math.round(t / DyMilli) + " days since 1/1/70";
       return(s);
     }
```

- getTimezoneOffset 函数：返回此地区的时差（当地时间与 GMT 格林威治标准时间的地区时差），单位为分钟。例如：

```
function TZDemo()
     {
       var d, tz, s = "The current local time is ";
       d = new Date();
       tz = d.getTimezoneOffset();
       if (tz < 0)
       s += tz / 60 + " hours before GMT";
       else if (tz == 0)
       s += "GMT";
       else
       s += tz / 60 + " hours after GMT";
       return(s);
     }
```

- getYear 函数：返回日期的“年”部分。返回值以 1900 年为基数，例如 1999 年为 99。见前面的例子。
- parse 函数：返回从 1970 年 1 月 1 日零时整算起的毫秒数（当地时间）。例如：

```
function GetTimeTest(testdate)
     {
       var d, s, t;
       var MinMilli = 1000 * 60;
       var HrMilli = MinMilli * 60;
       var DyMilli = HrMilli * 24;
       d = new Date();
       t = Date.parse(testdate);
       s = "There are "
       s += Math.round(Math.abs(t / DyMilli)) + " days "
       s += "between " + testdate + " and 1/1/70";
       return(s);
     }
```

- setDate 函数：设定日期的“日”部分，值为 0～31。
- setHours 函数：设定日期的“小时”部分，值为 0～23。
- setMinutes 函数：设定日期的“分钟”部分，值为 0～59。

- setMonth 函数：设定日期的“月”部分，值为 0～11。其中 0 表示 1 月，…，11 表示 12 月。
- setSeconds 函数：设定日期的“秒”部分，值为 0～59。
- setTime 函数：设定时间。时间数值为 1970 年 1 月 1 日零时整算起的毫秒数。
- setYear 函数：设定日期的“年”部分。
- toGMTString 函数：转换日期成为字符串，为 GMT 格林威治标准时间。
- setLocaleString 函数：转换日期成为字符串，为当地时间。
- UTC 函数：返回从 1970 年 1 月 1 日零时整算起的毫秒数，以 GMT 格林威治标准时间计算。

（4）数学函数。JavaScript 数学函数其实就是 Math 对象，它包括属性和函数（或称方法）两部分。其中，属性主要有下列内容。

Math.e:e（自然对数）、Math.LN2（2 的自然对数）、Math.LN10（10 的自然对数）、Math.LOG2E（e 的对数，底数为 2）、Math.LOG10E（e 的对数，底数为 10）、Math.PI（π）、Math.SQRT1_2（1/2 的平方根值）、Math.SQRT2（2 的平方根值）。函数有以下 18 个。

- abs 函数：即 Math.abs（以下同），返回一个数字的绝对值。
- acos 函数：返回一个数字的反余弦值，结果为 0～π 弧度（radians）。
- asin 函数：返回一个数字的反正弦值，结果为–π/2～π/2 弧度。
- atan 函数：返回一个数字的反正切值，结果为–π/2～π/2 弧度。
- atan2 函数：返回一个坐标的极坐标角度值。
- ceil 函数：返回一个数字的最小整数值（大于或等于）。
- cos 函数：返回一个数字的余弦值，结果为–1～1。
- exp 函数：返回 e（自然对数）的乘方值。
- floor 函数：返回一个数字的最大整数值（小于或等于）。
- log 函数：自然对数函数，返回一个数字的自然对数（e）值。
- max 函数：返回两个数的最大值。
- min 函数：返回两个数的最小值。
- pow 函数：返回一个数字的乘方值。
- random 函数：返回一个 0～1 的随机数值。
- round 函数：返回一个数字的四舍五入值，类型是整数。
- sin 函数：返回一个数字的正弦值，结果为–1～1。
- sqrt 函数：返回一个数字的平方根值。
- tan 函数：返回一个数字的正切值。

（5）字符串函数。JavaScript 字符串函数完成对字符串的字体大小、颜色、长度设置和查找等操作，共包括以下 20 个函数。

- anchor 函数：产生一个链接点（anchor）以作超级链接用。anchor 函数设定链接点的名称，另一个函数 link 设定 URL 地址。
- big 函数：将字体加大一号。
- blink 函数：使字符串闪烁。
- bold 函数：使字体加粗。

- charAt 函数：返回字符串中指定的某个字符。
- fixed 函数：将字体设定为固定宽度。
- fontcolor 函数：设定字体颜色。
- fontsize 函数：设定字体大小。
- indexOf 函数：返回字符串中第一个查找到的下标 index，从左边开始查找。
- italics 函数：使字体成为斜体字。
- lastIndexOf 函数：返回字符串中第一个查找到的下标 index，从右边开始查找。
- length 函数：返回字符串的长度（不用带括号）。
- link 函数：产生一个超级链接，相当于设定的 URL 地址。
- small 函数：将字体减小一号。
- strike 函数：在文本的中间加一条横线。
- sub 函数：显示字符串为下标字（subscript）。
- substring 函数：返回字符串中指定的几个字符。
- sup 函数：显示字符串为上标字（superscript）。
- toLowerCase 函数：将字符串转换为小写。
- toUpperCase 函数：将字符串转换为大写。

由于章节篇幅有限，在这里就没有对每一个函数详细举例说明了，不过上面列出的函数只要读者知道它们的作用就行了，在后面的章节里我们还会涉及到，到时候再对遇到的函数做深度的讲解。

3.3.6　JavaScript 事件驱动与事件处理

在前面已经介绍过了，JavaScript 是基于对象（object-based）的语言，这与 Java 面向对象的语言不同。Java 是面向对象的语言，而基于对象的基本特征，就是采用事件驱动（event-driven）。它是在图形界面的环境下，使得一切输入变得简单化。通常鼠标、热键或者触摸的动作我们就称之为事件（Event），而由鼠标、热键或者触摸引发的一连串程序的动作，称之为事件驱动（Event Driver）。而对事件进行处理的程序或函数，我们称之为事件处理程序（Event Handler）。

1．事件驱动

JavaScript 事件驱动中的事件是通过鼠标、热键或者触摸的动作引发的。它主要有以下几个事件：

（1）单击事件 onClick。当用户单击鼠标按钮时，产生 onClick 事件。同时 onClick 指定的事件处理程序或代码将被调用执行。单击事件通常在下列基本对象中产生。

- button（按钮对象）。
- checkbox（复选框）或（检查列表框）。
- radio （单选钮）。
- reset buttons（重置按钮）。
- submit buttons（提交按钮）。

例如：可通过下列按钮激活 change()文件。

```
<Form>
<Input type="button" Value="  " >
</Form>
```

在 onClick 等号后，可以使用自己编写的函数作为事件处理程序，也可以使用 JavaScript 中内部的函数。还可以直接使用 JavaScript 的代码等，如下所示。

```
<Input type="button" value=" " onclick=alert("这是一个例子");
```

（2）onChange 改变事件。当利用 text 或 texturea 元素输入字符值改变时引发该事件，同时当在 select 表格项中一个选项状态改变后也会引发该事件，如下所示。

```
<Form>
<Input type="text" name="Test" value="Test" >
</Form>
```

（3）选中事件 onSelect。当 Text 或 Textarea 对象中的文字被加亮后，引发该事件。

（4）获得焦点事件 onFocus。当用户单击 Text 或 textarea 以及 select 对象时，产生该事件。此时该对象成为前台对象。

（5）失去焦点 onBlur。当 text 对象或 textarea 对象以及 select 对象不再拥有焦点而退到后台时，引发该文件，它与 onFocas 事件是一个对应的关系。

（6）载入文件 onLoad。当文档载入时，产生该事件。onLoad 的一个作用就是在首次载入一个文档时检测 cookie 的值，并用一个变量为其赋值，使它可以被源代码使用。

（7）卸载文件 onUnload。当 Web 页面退出时引发 onUnload 事件，并可更新 Cookie 的状态。

2．事件处理程序

在 JavaScript 中对象事件的处理通常由函数（Function）担任。其基本格式与函数全部一样，可将前面所介绍的所有函数作为事件处理程序，格式如下。

```
Function 事件处理名(参数表)
{
事件处理语句 1;
事件处理语句 2;
......
}
```

3.3.7　如何将 JavaScript 加入网页

在第一个 JavaScript 程序那一小节，我们已经见到 JavaScript 引入网页的一种方式，这种方式我们称为内部引入方式，这种方式只对当前界面有效，在其他界面不可以调用。还有一种引入网页的方式我们称之为外部导入，这种方式将 JavaScript 作成单独文档放在特定目录下的 js 文件夹下。好处是一次编写，就可以使用其中的方法或者函数等。

1．内部嵌入

这种方式，使 JavaScript 的脚本包括在 HTML 中，成为 HTML 文档的一部分。与 HTML

标识相结合，构成了一个功能强大的 Internet 网上编程语言。可以直接将 JavaScript 脚本加入文档，如下所示。

```
<Script Language ="JavaScript">
JavaScript 语言代码;
 </Script>
```

通过标识<Script>...</Script>指明 JavaScript 脚本源代码将放入其间。通过属性 Language="JavaScript"说明标识中是使用的何种语言，这里是 JavaScript 语言，表示在 JavaScript 中使用的语言。下面是将 JavaScript 脚本加入网页（Web 文档）例子中的 js_in.html 代码。

```
<HTML>
<Head>
<Script Language ="JavaScript">  <!--调用 JavaScript 输出文字-->
document. write("内部调用 JavaScript 加入网页");
document. close();          <!--关闭输入文字-->
</Script>
</Head>
</HTML>
```

用浏览器打开效果如图 3-23 所示。

图 3-23　内部调用 JavaScript 效果

说明：Document. write()是文档对象的输出函数，其功能是将括号中的字符或变量值输出到窗口；document. close()是将输出关闭。可将<Script>...</Script>标识放入<Head>…</Head>或<Body> ...</Body>之间。将 JavaScript 标识放在<Head>...</Head>头部之间，使之在主页和其余部分代码之前装载，从而可使代码的功能更强大；将 JavaScript 标识放置在<Body>... </Body>主体之间可以实现某些部分动态地创建文档。

2．外部引入

这种方式首先创建一个*.js 文件，把要实现的 JavaScript 语句保存到该文件，并存储在特定的 js 文件夹中（当有多个.js 文件时使用该文件夹管理所有的.js 文件，如果只有一个.js 文件的话，可以把该文件和调用该文件的网页放在同一目录下即可），这样使用相同效果或者相同功能的页面就可以使用同一个.js 文件，不仅便于代码的复用，减少了与网页的耦合，还使代码显得简单，可读性增强。

例如：在网页 js_out.html 中调用 js_out.js 文件，并显示提示。

js_out.html 网页内容如下。

```
<html>
< body>
<!--调用 test.js 文件-->
<script language="JavaScript" src="js_out.js">
< /script>
< /body>
< /html>
```

js_out.js 的文件内容如下。

```
alert("测试外部调用 JavaScript 语句！")
```

效果如图 3-24 所示。

图 3-24　外部调用 JavaScript 效果

3.3.8　JavaScript 对象的使用

所谓对象就是真实世界中的实体，对象与实体是一一对应的，也就是说现实世界中每一个实体都是一个对象，它是一种具体的概念。JavaScript 语言是基于对象的（Object-Based），而不是面向对象的（object-oriented）。之所以说它是一门基于对象的语言，主要是因为它没有提供抽象、继承、重载等有关面向对象语言的许多功能。而是把其他语言所创建的复杂对象统一起来，从而形成一个非常强大的对象系统。下面我们就来介绍一下 JavaScript 的对象系统。

1．JavaScript中的对象

JavaScript 中的对象是由属性（properties）和方法（methods）两个基本的元素构成的。前者是对象在实施其所需要行为的过程中，实现信息的装载单位，从而与变量相关联；后者是指对象能够按照设计者的意图而被执行，从而与特定的函数相联。

（1）JavaScript 对象的定义。其基本格式如下。

```
Function Object（属性表）      //定义一个函数
This.prop1=prop1               //初始化属性
```

```
This.prop2=prop2
...
This.meth=FunctionName1;    //初始化方法
This.meth=FunctionName2;
...
```

在一个对象的定义中，可以为该对象指明其属性和方法。通过属性和方法构成了一个对象。下面是一个关于 factory 对象的定义的例子。

```
Function factory(name，city，creatDate URL)
This.name=name
This.city=city
This.creatDate=New Date(creatDate)
This.URL=URL
```

其基本含义如下。

- Name：指定一个“单位”名称。
- City：“单位”所在城市。
- CreatDate：记载 factory 对象的更新日期。
- URL：该对象指向一个网址。

（2）创建对象实例，一旦对象定义完成后，就可以为该对象创建一个实例了。基本格式如下。

```
newObject=new Object();
```

其中 newobject 是新的对象，Object 是已经定义好的对象。例如下面的代码。

```
newCity=new city("北京市","朝阳区","January 05,2012 12:00:00","http://
www.beijing.com")
newUniversity=new university("北京大学","北京市","January 05 2012 12:00:00",
"htlp://www.beijing.CN")
```

2．常用对象的属性和方法

JavaScript 为我们提供了一些非常有用的常用内部对象和方法，用户不需要用脚本来实现这些功能。这正是基于对象编程的真正目的。

在 JavaScript 提供了 string（字符串）、math（数值计算）和 Date（日期）3 种对象以及其他一些相关的方法，从而为编程人员快速开发强大的脚本程序提供了非常有利的条件。

1）串对象

（1）串（string）对象的特点。内部静态性，访问 properties 和 methods 时，可使用（.）运算符实现。

基本使用格式：objectName.prop/methods。

说明：objectName 是对象名称，prop/methods 表示属性或者方法。这些在后面的方法和属性的引用中会具体讲解。

（2）串对象的属性。该对象只有一个属性，即 length。它表明了字符串中的字符个数，包括所有符号。例如下面定义一个串对象，返回字符串的长度，代码如下：

```
mytest="This is a JavaScript"   //定义一个串对象，并初始化
mystringlength=mytest.length    //返回字符串的长度
```

最后 mystringlength 返回 mytest 字串的长度为 20。

（3）串对象的方法。string 对象的方法共有 19 个。主要用于有关字符串在 Web 页面中的显示、设置字体大小和颜色、字符的搜索以及字符的大小写转换，其主要的 4 个方法如下。

① 锚点 anchor()，该方法用于创建 HTML 锚点。所谓锚点，可以理解为网页内的“超级链接”，可以让读者在网页内跳转方便阅读。通过下列格式访问。

```
string.anchor(anchorName)
```

② 有关字符显示的控制方法。Big 为显示 big 字体，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，fontsize(size)控制字体大小，fontcolor(color) 控制字体颜色。

③ 字符串大小写转换。toLowerCase()表示小写转换，toUpperCase()表示大写转换。下列把一个给定的字符串分别转换成大写和小写格式。

```
string=stringValue.toUpperCase     //将字符转换成大写字母
string=stringValue.toLowerCase     //将字符转换成小写字母
```

④ 字符搜索。例如：

```
indexOf[characto，fromIndex]      //从指定位置搜索 characto 第一次在出现的位置
```

上面的代码是从指定 froIndex 位置开始搜索字符 characto 第一次出现的位置，返回整数。下列代码返回字符串的一部分字符串：

```
substring(start，end)//截取从 start 开始到 end 结束的字符串，star 和 end 为 int 类型
```

从 start 开始到 end 的字符全部返回。

2）算术函数的 math 对象

（1）math 对象的功能：除加、减、乘、除 4 个标准算术运算外，math 对象还包含一些复杂的数学方法，用于完成算术运算无法实现的复杂计算，例如 cos()方法用于计算弧度参数的余弦值。静动性：静态对象。

（2）math 对象属性。math 中提供了 6 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(1/2)，2 的平方根为 SQRT2。

（3）math 对象主要方法如下。

- 绝对值：abs()。
- 正弦、余弦值：sin()、cos()。
- 反正弦、反余弦：asin()、acos()。
- 正切、反正切：tan()、atan()。
- 四舍五入：round()。
- 平方根：sqrt()。
- 基于几方次的值：Pow(base，exponent)。

3）日期及时间对象

（1）日期及时间对象的功能：提供一个有关日期和时间的对象。静动性：动态对象，

即必须使用 New 运算符创建一个实例，如下所示。

```
MyDate=New Date()
```

（2）日期及时间对象的属性：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。日期起始值：1770 年 1 月 1 日 00:00:00。

（3）获取日期和时间的方法如下。

- getYear()：返回年数。
- getMonth()：返回当月号数。
- 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。

（4）设置日期和时间。

- setYear()：设置年。
- setDate()：设置当日号数。
- 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。

3．JavaScript对象的操作语句

JavaScript 不是一个纯面向对象的语言，它设有提供面向对象语言的许多功能，因此 JavaScript 设计者把它称为“基于对象”而不是面向对象的语言。在 JavaScript 中提供了几个用于操作对象的语句、关键字及运算符。

（1）For...in 语句。格式如下。

```
For(对象属性名 in 已知对象名)
```

说明：该语句的功能是用于对已知对象的所有属性进行操作的控制循环。它是将一个已知对象的所有属性反复置给一个变量，而不是使用计数器来实现的。对于循环控制语句在前面已经讲过，可以复习查看。

使用该语句的优点就是无须知道对象中属性的个数即可进行操作。比使用之前我们介绍的 for 循环语句更加方便。用之前介绍的 for 循环语句显示数组中的内容，代码如下。

```
Function showData(object)
for (var X=0; X<30;X++)
document.write(object);
```

该函数是通过数组下标顺序值，来访问每个对象的属性。使用这种方式首先必须知道

数组的下标值，否则若超出范围，就会发生错误。而使 For...in 语句，则根本不需要知道对象属性的个数，如下所示。

```
Function showData(object)
for(var prop in object)
document.write(object[prop]);
```

使用该函数时，在循环体中，For 自动将对象中的属性取出来，直到最后为此。

（2）with 语句。在该语句体内，任何对变量的引用都被认为是这个对象的属性，以节省一些代码。格式如下。

```
with object
{
语句 1;语句 2;...
}
```

说明：*所有在 with 语句后的花括号中的语句，都是在后面 object 对象的作用域的。*

（3）this 关键字。this 是对当前的引用，在 JavaScript 中由于对象的引用是多层次、多方位的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，最后自己可能都不知道现在引用的是哪一个对象，为此 JavaScript 提供了一个用于将对象指定为当前对象的语句 this。

（4）New 运算符。虽然在 JavaScript 中对象的功能已经是非常强大的了。但更强大的是设计人员可以按照需求来创建自己的对象，以满足某一特定的要求。使用 New 运算符可以创建一个新的对象，其创建对象的基本格式如下。

```
newObject=new Object(Parameters table);
```

说明：*其中 newObject 是创建的新对象：Object 是已经存在的对象；Parameters table 是参数表；new 是 JavaScript 中的命令语句。*

下面创建一个日期新对象，代码如下。

```
newData=new Data()
birthday=new Data (December 30.2012)
```

之后就可使 newData、birthday 作为一个新的日期对象了。

4．引用对象的途径

一个对象要真正地被使用，可采用以下几种方式获得。

（1）引用 JavaScript 内部对象。

（2）由浏览器环境提供。

（3）创建新对象。

这就是说，一个对象在被引用之前，这个对象必须存在，否则引用将毫无意义，并会出现错误信息。从上面我们可以看出 JavaScript 引用对象可通过 3 种方式获取。要么创建新的对象，要么利用现存的对象。

5．对象属性的引用

概括起来说，对象属性的引用有以下几种方式，

（1）使用点（.）运算符。这种方式在前面的对象属性和方法中已经提及。基本格式如下所示。

```
university.Name="北京"
university.city="北京大学"
university.Date="2012"
```

其中 university 是一个已经存在的对象，Name、City 和 Date 是它的 3 个属性，并通过操作对其赋值。

（2）通过对象的下标实现引用。基本格式如下所示。

```
university[0]="北京"
university[1]="北京大学"
university[2]="2012"
```

（3）通过数组形式访问属性，可以使用循环控制语句获取其值。代码如下。

```
function showunievsity(object)
for (var j=0;j<2; j++)
document.write(object[j])
```

若采用 For...in 语句，则可以不知其属性的个数就可以实现，代码如下。

```
Function showmy(object)
for (var prop in this)
docament.write(this[prop]);
```

（4）通过字符串的形式实现。基本格式如下所示。

```
university["Name"]="云南"
university["City"]="昆明市"
university["Date"]="1999"
```

6．对象方法的引用

在 JavaScript 中对象方法的引用是非常简单的。我们在前面介绍对象的方法时也提及到了，基本格式如下。

```
ObjectName.methods()
```

说明：ObjectName 是对象名称，methods()是方法名。

实际上 methods()=FunctionName 方法实质上是一个函数。 如引用 university 对象中的 show()方法，则可使用 document.write (university.show())或 document.write(university)。

小试牛刀：引用 math 内部对象中 cos()的方法。

分析：由于 cos()方法在 math 内部对象中，所有使用 with(math)，这样就可以直接引用内部的 cos 方法了。

```
document.write(cos(35));
document.write(cos(80));
```

若不使用 with 则引用时相对要复杂些，如下所示。

```
document.write(math.cos(35))
document.write(math.sin(80))
```

由于篇幅有限，如果要将 JavaScript 讲得面面俱到、非常细致的话，恐怕我这本书只讲它都讲不完，因为 JavaScript 的知识还有很多，不过我们旨在入门，所以只要读者了解我上面讲的这些常用的基础知识就可以对 JavaScript 的常规方法和属性进行使用了。由于 JavaScript 是网页中的重要部分，所以我们在后面的每个章节中都会用到 JavaScript，只要大家用心跟着我讲的去学、去一点一点地积累，您就可以自己动手独立做 Java Web 开发了，Java Web 开发入门就是这么简单。

3.4　桩功之四：JSP 动态界面的设计

在前面我们已经把 HTML、CSS 和 JavaScript 的基础知识做了比较细致的讲解，通过前面的学习，相信大家已经可以做出非常漂亮、功能不错的 HTML 静态界面了，但是 HTML 界面毕竟是静态界面，由 Web 服务器向客户端发送。

如果用户要知道服务器的时间，还使用 HTML 静态页面，那么开发人员就要在服务器端不停地修改 HTML 页面中的时间，这是不是很麻烦呢？但是这个让 JSP 来做就非常简单，因为 JSP 是由 JSP 容器执行该页面的 Java 代码部分，然后实时生成 HTML 页面，所以通过调用 Java 函数实现时间的获取，就可以很轻易地实现该功能。既然那么厉害，就让我们来了解一下什么是 JSP。

3.4.1　什么是 JSP

JSP 是 Java Server Page 的缩写，是基于 Java 语言的一种 Web 应用开发技术，通俗一点说，它是一种实现普通静态 HTML 和动态 HTML 混合编码的技术。利用这一技术可以搭建一个安全、跨平台的，动态网站。

JSP 是在 Servlet 的基础之上产生的（Servlet 后面会讲到），用来显示页面。我们在刚开始学习 JSP 的时候，就可以把 JSP 理解成它实现了把 Java 语句写到 HTML 里面去。当然随着我们学习的深入，尤其在学习了 MVC 模式之后，就不是这样了，JSP 充当的是 View（视图层）的角色，也就是说 JSP 只是用来做显示的，而不应该包含业务逻辑。业务逻辑是放在 JavaBean 中的，也就是 Service 对象。

3.4.2　JSP 运行原理

JSP 是服务器端技术，在服务器端，JSP 引擎解释 JSP 代码，然后将结果以 HTML 或 XML 页面形式发送到客户端，在客户端的用户是看不到 JSP 代码的。具体原理解释如下：第一次请求 JSP 页面，JSP 页面将先转换成为一个 Java 文件（Servlet），编译后该 Java 文件生成对应的 class 文件，将其加载在内存，然后执行 class 文件完成响应；再次请求就直

接加载 class 文件完成响应，每次请求都会启动一个线程来负责。当第一次加载 JSP 页面时，因为要将 JSP 文件转换为 Servlet 类，所以响应速度较慢；当再次请求时，JSP 容器就会直接执行第一次请求时产生的 Servlet，而不会再重新转换 JSP 文件，所以其执行速度较快。JSP 运行原理如图 3-25 所示。

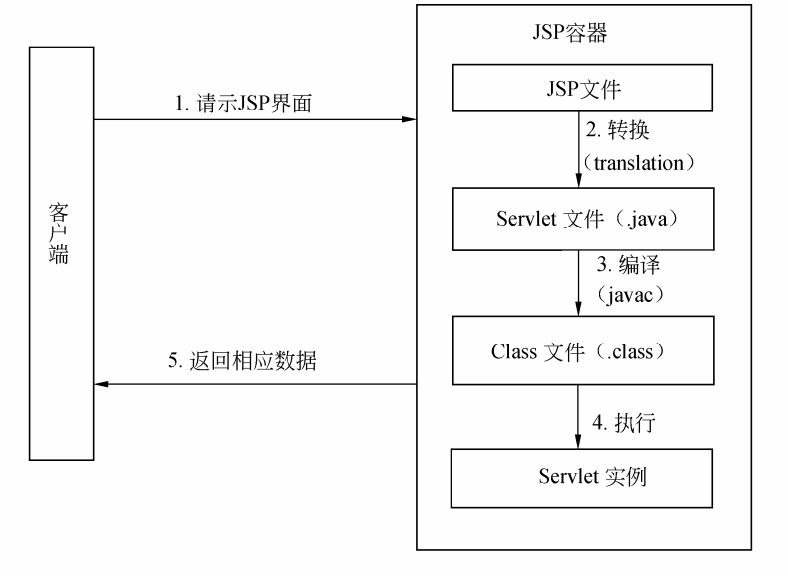

图 3-25　JSP 运行原理

3.4.3　JSP 语法

1．JSP的主要特点

- 把内容的生成和显示分离。
- 生成可重用的组件。
- 应用标记简化页面的开发。
- 具有 Java 的特点。

2．JSP页面组成

JSP 代码放在特定的标签中，然后嵌入到 HTML 代码中。开始标签、结束标签和元素内容三部分统称为 JSP 元素（Elements），这是 JSP 页面组成的主要部分。

JSP 元素可分成如下三种不同的类型。

- 脚本元素（Scripting）：规范 JSP 网页所使用的 Java 代码，包括：HTML 注释、隐藏注释、声明、表达式和脚本段。
- 指令元素（Directive）：是针对 JSP 引擎的，并不会直接产生任何看得见的输出。包括：inc1ude 指令、page 指令和 taglib 指令。
- 动作元素（Action）：利用 XML 语法格式的标记来控制 Servlet 引擎的行为。

JSP 页面组成如图 3-26 所示。

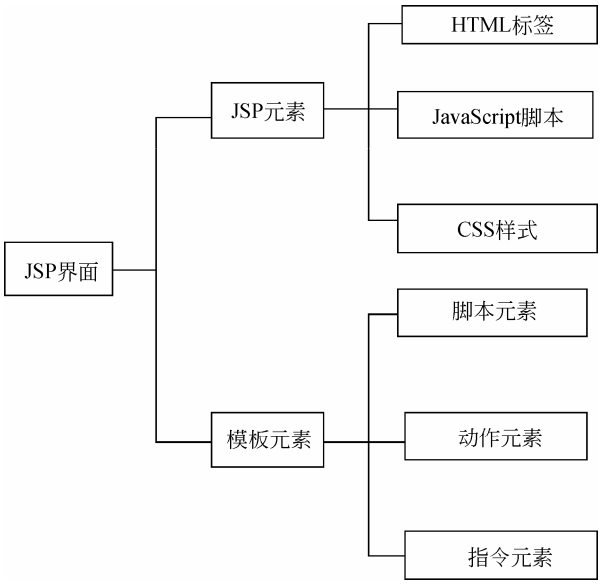

图 3-26　JSP 页面组成

3．JSP注释

注释增加了程序的可读性与可维护性，应该养成写注释的好习惯。JSP 文件的注释有两种：HTML 注释和隐藏注释。

HTML 注释：发送到客户端，但不在浏览器上显示，在客户端可通过查看源文件看到，基本语法如下。

```
<!--注释[<%=表达式%>]-->
```

隐藏注释：写在 JSP 程序代码中，不发送到客户端，基本语法如下。

```
<%--注释--%>
```

4．JSP变量和方法的声明

在 JSP 程序中需要对用到的变量和方法进行声明，声明的基本语法如下。

```
<%!声明;[声明;]%>
```

例如下面这段代码。

```
<%! int i=6;%>
<%! int a,b,c;double d=6.0;%>
<%! Date d=new Date(); %>
```

需要注意的事项如下。

- 声明必须以“；”结尾。
- 可以一次声明多个变量和方法，必须以“，”分开，以“；”号结尾。
- 声明的范围通常是 JSP 页面，但如果页面中使用 include 指令包含其他页面，范围应扩展到被包含的页面。
- 可以直接使用在<%@ page%>指令中包含进来已经声明了的变量和方法，不需要

重新进行声明。

- 一个声明仅在一个页面中有效。如果想每个页面都用到一些声明，最好把它们写成一个单独的文件，然后用＜%@include%>或＜jsp:inc1ude>动作包含进来。

5．JSP的表达式

JSP 的表达式是由变量、常量组成的算式，它将 JSP 生成的数值嵌入 HTML 页面，用来直接输出 Java 代码的值。

表达式的基本语法规则如下。

```
<%=表达式%>
```

需要注意的事项如下。

- 不能用一个分号“；”来作为表达式的结束符。
- “<%=”是一个完整的标记，中间不能有空格。
- 表达式元素包含任何在 Java 语言规范中有效的表达式。
- 表达式可以成为其他 JSP 元素的属性值。一个表达式可以由一个或多个表达式组成，按从左到右的顺序求值。

3.4.4　JSP 指令

JSP 指令是一些特殊的 JSP 语句，它是为 JSP 引擎而设计的，它们并不直接产生任何可见输出，只是告诉引擎如何处理其余的 JSP 页面，这些指令被括在“<%@　%>”标记中，常见指令有以下三种。

- page 指令。
- include 指令。
- taglib 指令：用来定义一个标记库以及标记的前缀。

1．page指令

page 指令称为页面指令，几乎在所有 JSP 页面顶部都会看到 page 指令。

（1）page 指令的语法规则如下。

```
<%@ page language="脚本语言"
         extends="继承的父类名称"
         import="导入的 java 包或类的名称"
         session="true/false"
         buffer="none/8kB/自定义缓冲区大小"
         autoflush="true/false"
         isThreadSafe=" true/false"
         info="页面信息"
         errorPage="发生错误时所转向的页面相对地址"
         isErrorPage="true/false"
         contentType="MIME 类型和字符集"
%>
```

（2）page 指令的常用属性如下。

- import：用来导入将要用到的一个或多个包/类。基本语法规则如下。

```
<%@ page import="java.util.Date"%>
<%@ page import="java.util.*"%>
```

- errorPage：这个属性值为一个 URL 路径指向的 JSP 网页，在指向的 JSP 网页中处理初始 JSP 网页上产生的错误；通常在指向的 JSP 网页上都会设置 isErrorPage=true。
- isErrorPage：这个属性的默认值为 false；isErrorPage 用来指定目前的 JSP 网页是否是另一个 JSP 网页的错误处理页，通常与 errorPage 属性配合使用。
- contentType：用来指定 JSP 网页输出到客户端时所用的 MIME 类型和字符集。默认 MIME 类型是 text/html，默认的字符集是 ISO-8859-1。如果想输出简体中文，字符集需要被设置为 gb2312。

需要注意的事项如下。

- 在一个页面中可以使用多个<%@ page%>指令，分别描述不同的属性。
- 每个属性只能用一次，但是 import 指令可以多次使用。
- <%@ page%>指令区分大小写。

（3）page 指令的用法。为了更好地掌握 page 指令的用法，下面我们举例说明，本例包括两个 JSP 页面文件：pageDir.jsp 和 errorPageDir.jsp。

pageDir.jsp 为主页面，在本页面中，通过 page 指令指定当页面发生错误时转向的错误处理页面。errorPageDir.jsp 为错误处理页面，在该页面显示相关信息提示用户访问出错。

pageDirec.jsp 代码如下。

```
<%@ page contentType="text/html; charset=gb2312 errorPage="errorPageDirec
.jsp"%>
    <html><head><title>page 指令示例</title></head>
        <body>
            <%
                int a = 10;
                int b = a / 0;
                out.println(b);
            %>
        </body>
</html>
```

errorPageDirec.jsp 代码如下。

```
<%@ page contentType="text/html; charset=gb2312" isErrorPage="true"%>
    <html><head><title>错误处理页面</title></head>
        <body>
            您访问的页面发生了错误！！
        </body>
    </html>
```

说明：errorPageDirec.jsp 文件的 page 指令 isErrorPage="true" 指定该文件为错误处理文件，只能通过其他页面发生错误而转向它来运行。

运行结果如图 3-27 所示。

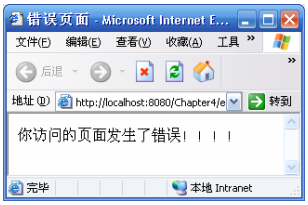

图 3-27　page 指令用法举例

2．include指令

有时候我们需要在 JSP 网页中插入其他的文件，插入文件

有两种方式：include 指令和 jsp:include 动作。

include 指令称为文件加载指令，可以将其他的文件插入 JSP 网页，被插入的文件可以是 JSP 文件、HTML 文件或者其他文本文件，但是必须保证插入后形成的新文件符合 JSP 页面的语法规则。

include 指令形式如下。

```
<%@ include file="相对地址"%>
```

说明：file 是 include 指令的属性，在 include 指令中只有一个属性：file。

下面用实例说明 include 指令的使用，该实例包含两个文件：systemTime.html 文件和 includeDirec.jsp 文件。systemTime.html 输出系统的日期和时间；includeDirec.jsp 中通过 include 指令将 systemTime.html 文件包含进来。

systemTime.html 代码如下。

```
<%=(new Date()).toLocaleString()%>
```

includeDirec.jsp 代码如下。

```
<%@ page contentType="text/html; charset=gb2312" import="java.util.*"%>
<html>
    <head>
        <title>include 指令实例</title>
    </head>
    <body>
        <center>
            现在的日期和时间是：
            <hr>
            <%@ include file="systemTime.html"%>
        </center>
    </body>
</html>
```

3．taglib指令

taglib 指令用来定义一个标记库以及标记的前缀，其语法规则如下。

```
<%@ taglib uri="URIToLibrary" prefix="标记前缀"%>
```

3.4.5　JSP 动作

JSP 动作元素用来控制 JSP 引擎的行为，可以动态插入文件、重用 JavaBean 组件（后面会讲）和导向另一个页面等。常见的 JSP 动作元素有如下几种。

- jsp:include 动作：在页面得到请求时包含一个文件。
- jsp:forward 动作：引导请求者进入新的页面。
- jsp:plugin 动作：连接客户端的 Applet 或 Bean 插件。
- jsp:useBean 动作：应用 JavaBean 组件。
- jsp:setProperty 动作：设置 JavaBean 的属性。
- jsp:getProperty 动作：获取 JavaBean 的属性并输出。

注意：JSP 动作元素的形式都是以 XML 为标准的，而 XML 中大小写是敏感的。因此 jsp:useBean 不等于 jsp:usebean，前者是标准的动作元素，而后者什么都不是，在实际使用时一定要注意。

1．jsp:include动作

jsp:include 动作在即将生成的页面上动态地插入文件，它在页面运行时才将文件插入，对被插入的文件进行处理，也就是说它是在页面产生时插入文件。

（1）其语法规则如下。

```
<jsp:include page="文件相对路径" flush="true"/>或
<jsp:include page="文件相对路径" flush="true">
       <jsp:param name="参数名 1" value="参数值 1"/>
       <jsp:param name="参数名 2" value="参数值 2"/>
    …
</jsp:include>
```

（2）include 指令和 jsp:include 动作的区别。前面介绍了 include 指令，我们知道 include 指令是静态的，是在 JSP 文件被转换成 Servlet 的时候引入文件，它把被插入文件插到当前位置后再进行编译。

jsp:include 动作是动态的，插入文件的时间是在页面被请求的时候。JSP 引擎不把插入文件和原 JSP 文件合并成一个新的 JSP 文件，而是在运行时把被插入文件包含进来。

注意：如果包含页面时需要传递参数，则只能使用 jsp:include 动作。

（3）jsp:include 动作实例。为了详细说明，来看看下面的示例。该实例包含 5 个文件，在 newBook.jsp 文件代码中插入了 4 个文件：newbook1.html、newbook2.html、newbook3.html 和 newbook4.html。这 4 个文件分别表示 4 本新书的信息。

newBook.jsp 文件代码如下。

```
<%@ page contentType="text/html; charset=GB2312"%>
<html>
<head><title>jsp:include 动作示例</title></head>
    <body>
        <p align="center">新书展示</p><hr>
        <!--新建表格，居中，表格线宽为 1，两行两列-->
        <table border="1" align="center">
            <tr>
                <td>
                    <! --包含 newbook1.html 页面-->
                  <jsp:include page="newbook1.html" flush="true" />
                </td>
                <td>
                  <! --包含 newbook2.html 页面-->
                  <jsp:include page="newbook2.html" flush="true" />
                </td>
            </tr>
            <tr>
                <td>
                  <! --包含 newbook3.html 页面-->
                  <jsp:include page="newbook3.html" flush="true" />
                </td>
```

```
                <td>
                  <! -- 包含 newbook4.html 页面  -->
                  <jsp:include page="newbook4.html" flush="true" />
                </td>
            </tr>
        </table>
</body>
</html>
```

2．jsp:forward动作

用于停止当前页面的执行，转向另一个 HTML 或 JSP 页面。在执行中 JSP 引擎不再处理当前页面剩下的内容，缓冲区被清空。在客户端看到的是原页面的地址，而实际显示的是另一个页面的内容。

（1）jsp:forward 动作的语法规则如下。

```
<jsp:forward page="文件名"/>
```

或者中间加入参数，格式如下。

```
<jsp:forward page="文件名">
    <jsp:param name="参数名 1" value="参数值 1"/>
    <jsp:param name="参数名 2" value="参数值 2"/>
   …
</jsp:forward>
```

（2）jsp:forward 示例说明。下面我们用常见的登录模块作为示例来具体说明该动作的使用。

当用户进入登录界面 login.jsp，输入用户名和密码，提交表单后，由文件 loginReceive.jsp 接收用户的输入，如果输入正确则转到文件 loginCorrect.html ，如果输入错误则转到 loginError.html。文件 login.jsp 的主要代码如下。

```
......
 <!--定义 form 表单，接受输入的姓名和密码-->
<form name="form1" method="post"
        action="loginReceive.jsp">
        姓名
         <! --接受输入用户名，类型为文本类型，大小为 12-->
        <input name="userName" type="text" size="12">
   <br>
         密码
         <!--接受输入密码，类型为密码，大小为 12 -->
        <input name="passWord" type="password" size="12">
   <br>
        <!--单击提交，跳转到 action 指向的 loginReceive.jsp -->
        <input type="submit" name="Submit" value="提交">
        <!--单击重置，触发 Submit2，类型为 reset，使已经输入的内容清空-->
        <input type="reset" name="Submit2" value="重置">
</form>
......
```

文件 loginReceive.jsp 主要代码如下。

```
......
<body>
```

```
       <!--获取用户名和密码-->
        <%
         String Name=request.getParameter("userName");
         String Pwd=request.getParameter("passWord");
         <!--如果用户名是 java 并且密码是 123456，跳转到 loginCorrect.html
         否则，跳转到 loginError.html
         -->
         if(Name.equals("java")&& Pwd.equals("123456"))
         {
       %>
        <jsp:forward page="loginCorrect.html" />
        <%}
         else
       %>
        <jsp:forward page="loginError.html" />
</body>......
```

说明：loginCorrect.html 文件显示登录成功信息，loginError.html 文件显示登录失败的信息。

3．jsp:plugin动作

jsp:plugin 动作的功能是将服务器端的 Java 小应用程序（Applet）或 JavaBean 组件下载到浏览器端去执行，相当于在客户端浏览器插入 Java 插件。

（1）jsp:plugin 动作的语法规则如下。

```
<jsp:plugin
          type="bean | applet"
          code="保存类的文件名"
          codebase="类路径"
          [name="对象名"]
          [archiv="相关文件路径"]
          [align="bottom | top | middle | left | right"]//对齐方式
          [height="displayPixels"]     //高度
          [width="displayPixels"]      //宽度
          [hspace="leftRightPixels"]   //水平间距
          [vspace="topBottomPixels"]   //垂直间距
          [jreversion="Java 环境版本"]
          [nspluginurl="供 NC 使用的 plugin 加载位置"]
          [iepluginurl="供 IE 使用的 plugin 加载位置"]>
    <jsp:params>
          <jsp:param name="参数 1"  value="参数值 1"/>
          <jsp:param name="参数 2"  value="参数值 2"/>
          ...
    </jsp:params>
    [<jsp:fallback>错误信息</jsp:fallback>]
</jsp:plugin>
```

（2）jsp:plugin 动作常用属性如下。

- type="bean | applet"：指定将被执行的插件对象的类型是 Bean 还是 Applet。
- code="保存类的文件名"：指定 Java 插件将要执行的字节码（Java Class）文件的名字，其后缀必须是.class。这个文件必须保存在由 codebase 属性指定的目录里。

- codebase="类路径"：说明将要被下载的 Java Class 文件的目录。
- name="对象名"：bean 或 applet 实例的名字。
- [<jsp:fallback>错误信息</jsp:fallback>]：一段文字，当 Java 插件不能启动时，这段文字向用户显示。如果插件能够启动而 Applet 或 Bean 不能执行，那么浏览器弹出一个错误信息。

（3）jsp:plugin 动作应用示例。在文件 plugin.jsp 中使用 jsp:plugin 动作下载名为 RollingMessage.java 的 Java 小程序。

文件 plugin.jsp 主要代码如下。

```
<body>
        <center>
            用&ltjsp:plugin&gt 加载 Applet
            <hr>    <br>
            <!--用 plugin 加载 applet -->
             <jsp:plugin type="applet"
                  code="RollingMessage.class" height="60"
                 width="550">
    </jsp:plugin>
        </center>
</body>
```

RollingMessage.java 程序需要先编译，形成字节码文件 RollingMessage.class，此 Applet 的功能是输出一行滚动显示的文字“欢迎学习“Java Web 入门很简单”！”

4．jsp:useBean动作

jsp:useBean 动作用来装载一个将要在 JSP 页面中使用的 JavaBean。它创建一个 JavaBean 实例并指定其名字和作用范围。

实际工程中常用 JavaBean 做组件开发，而在 JSP 中只需要声明并使用这个组件，这样可以较大限度地实现静态内容和动态内容的分离，这也是 JSP 的优点之一。

（1）jsp:useBean 语法规则。在 JSP 中实例化一个 bean 的最简单的方法如下。

```
<jsp:useBean id="bean 的名称" scope="有效范围" class="包名.类名" />
```

说明：其中 scope=“有效范围”，“有效范围”属性的取值有 4 种：page、 request、session 和 application，默认值是 page。

取不同值含义如下。

- page：该 JavaBean 只有在当前页面及当前页面所包含的静态页面有效。
- request：该 JavaBean 的有效范围是当前的客户请求。
- session：该 JavaBean 的有效范围是当前客户的会话期间。
- application：该 JavaBean 对所有具有相同 ServletContext 的页面都有效，即从创建开始，所有客户端共享这个 JavaBean，直至服务器关闭时才取消这个 bean。

还可以通过下面的形式实例化一个 JavaBean。

```
<jsp:useBean id="bean 的名称" scope="有效范围" class="包名.类名">
       实体
< /jsp:useBean >
```

注意：这种实例化形式下，只有当第一次实例化 bean 时才执行实体部分，如果是利用现有的 bean 实例则不执行实体部分。jsp:useBean 并非总是意味着创建一个新的 bean 实例。

（2）jsp:useBean 工作过程。JSP 引擎根据 useBean 中 id 属性指定的名字，在一个同步块中，查找内置对象 pageContext 中是否包含该 id 指定的名字和 scope 指定的作用域的对象。如果该对象存在，JSP 引擎就把这样一个对象分配给用户。如果不存在则创建新的 bean 实例。

5．jsp:setProperty动作

用来设置已经实例化的 bean 对象的属性。

（1）jsp:setProperty 动作的两种语法规则如下。

第一种是直接将属性值设置为字符串或表达式，形式如下。

```
<jsp:setProperty name="bean 的名称" property="bean 的属性名称" value= "属性值"/>
```

第二种方法用 request 的参数值来设置 JavaBean 的属性值，request 参数的名字和 JavaBean 属性的名字可以不同，其语法规则如下。

```
<jsp:setProperty name="bean 的名称"
                 property="bean 的属性名称"
                 param= "request 参数的名字"/>
```

（2）jsp:setProperty 动作的两种用法。

首先，可以在 jsp:useBean 元素的外面使用 jsp:setProperty。

```
<jsp:useBean id="myName" ... />
   ...
<jsp:setProperty name="myName"  property="someProperty" ... />
```

此时，不管 jsp:useBean 是找到了一个现有的 bean，还是新创建了一个 bean 实例，jsp:setProperty 都会执行。

第二种用法是把 jsp:setProperty 放入 jsp:useBean 元素的内部，如下所示。

```
<jsp:useBean id="myName" ... >
     ...
    <jsp:setProperty name="myName"
                property="someProperty" ... />
</jsp:useBean>
```

此时，jsp:setProperty 只有在新建 bean 实例时才会执行，如果是使用现有实例则不执行 jsp:setProperty。

（3）jsp:getProperty 动作。用来获取 beans 的属性值，将其转换成字符串，然后输出。其语法规则如下。

```
<jsp:getProperty name="bean 的名称"
                   property="bean 的属性名称"/>
```

注意：jsp:setProperty 动作和 jsp:getProperty 动作必须与 jsp:useBean 动作一起使用，不能单独使用。

3.5　桩功之五：Servlet 的认识和使用

在介绍 JSP 中提到 Servlet 是 JSP 的基础，那么这一节我们就来看看什么是 Servlet 以及 Servlet 在 Java Web 开发中是如何配置和使用的。

3.5.1　什么是 Servlet

Servlet 是一种服务器端的 Java 应用程序，具有独立于平台和协议的特性，可以生成动态的 Web 页面。它担当客户请求（Web 浏览器或其他 HTTP 客户程序）与服务器响应（HTTP 服务器上的数据库或应用程序）的中间层。Servlet 是位于 Web 服务器内部的服务器端的 Java 应用程序，与传统的从命令行启动的 Java 应用程序不同，Servlet 由 Web 服务器进行加载，该 Web 服务器必须包含支持 Servlet 的 Java 虚拟机。

Servlet=Server+Applet，意思是指 Servlet 为服务器端的小程序。这个词是在 Java applet 的环境中创造的，Java applet 是一种当作单独文件跟网页一起发送的小程序，它通常用于在服务器端运行，结果得到为用户进行运算或者根据用户交互作用定位图形等服务。

服务器上需要一些程序，常常是根据用户输入访问数据库的程序。这些通常是使用公共网关接口（CGI，Common Gateway Interface）应用程序完成的。然而，在服务器上运行 Java，这种程序可使用 Java 编程语言实现。在通信量大的服务器上，Java Servlet 的优点在于它们的执行速度更快于 CGI 程序。各个用户请求被激活成单个程序中的一个线程，而无需创建单独的进程，这意味着服务器端处理请求的系统开销将明显降低。

3.5.2　Servlet 的特点

与传统的 CGI 和许多其他类似 CGI 的技术相比，Java Servlet 具有更高的效率，更容易使用，功能更强大，具有更好的可移植性，更节省投资。在未来的技术发展过程中，Servlet 有可能彻底取代 CGI。

在传统的 CGI 中，每个请求都要启动一个新的进程，如果 CGI 程序本身的执行时间较短，启动进程所需要的开销很可能反而超过实际执行时间。而在 Servlet 中，每个请求由一个轻量级的 Java 线程处理（而不是重量级的操作系统进程）。

在传统 CGI 中，如果有 N 个并发的对同一 CGI 程序的请求，则该 CGI 程序的代码在内存中重复装载了 N 次；而对于 Servlet，处理请求的是 N 个线程，只需要一份 Servlet 类代码。在性能优化方面，Servlet 也比 CGI 有着更多的选择。

（1）方便。Servlet 提供了大量的实用工具例程，例如自动地解析和解码 HTML 表单数据、读取和设置 HTTP 头、处理 Cookie、跟踪会话状态等。

（2）功能强大。在 Servlet 中，许多使用传统 CGI 程序很难完成的任务都可以轻松地

完成。例如，Servlet 能够直接和 Web 服务器交互，而普通的 CGI 程序则不能。Servlet 还能够在各个程序之间共享数据，使得数据库连接池之类的功能很容易实现。

（3）可移植性好。Servlet 用 Java 编写，Servlet API 具有完善的标准。因此，很多的 Servlet 无需任何实质上的改动即可移植到 Apache 和 Microsoft IIS 上。几乎所有的主流服务器都直接或通过插件支持 Servlet。

（4）节省投资。不仅有许多廉价甚至免费的 Web 服务器可供个人或小规模网站使用，而且对于现有的服务器，如果它不支持 Servlet 的话，要加上这部分功能也往往是免费的（或只需要极少的投资）。

3.5.3　Servlet 的生命周期

Servlet 生命周期分为三个阶段，如下所示。

1．Servlet初始化阶段

在下列时刻 Servlet 容器装载 Servlet。

（1）Servlet 容器启动时自动装载某些 Servlet，实现它只需要在 web.XML 文件中的<Servlet></Servlet>之间添加如下代码。

```
<loadon-startup>1</loadon-startup>
```

（2）在 Servlet 容器启动后，客户首次向 Servlet 发送请求。

（3）Servlet 类文件被更新后，重新装载 Servlet。

Servlet 被装载后，Servlet 容器创建一个 Servlet 实例并且调用 Servlet 的 init()方法进行初始化。在 Servlet 的整个生命周期内，init()方法只被调用一次。

2．Servlet响应请求阶段

对于用户到达 Servlet 的请求，Servlet 容器会创建特定于这个请求的 ServletRequest 对象和 ServletResponse 对象，然后调用 Servlet 的 service 方法。service 方法从 ServletRequest 对象获得客户请求信息，处理该请求，并通过 ServletResponse 对象向客户返回响应信息。

对于 Tomcat 来说，它会将传递过来的参数放在一个 Hashtable 中，该 Hashtable 的定义如下。

```
private  Hashtable<String String[]>  paramHashStringArray = new  Hashtable
<String String[]>();
```

提示：这是一个 String→String[]的键值映射。HashMap 线程是不安全的，Hashtable 线程安全。

3．Servlet终止阶段

当 Web 应用被终止，或 Servlet 容器终止运行，或 Servlet 容器重新装载 Servlet 新实例时，Servlet 容器会先调用 Servlet 的 destroy()方法，在 destroy()方法中可以释放掉 Servlet 所占用的资源。

总的来说就是 Servlet 被服务器实例化后，容器运行其 init 方法，请求到达时运行其 service 方法，service 方法自动派遣运行与请求对应的 doXXX 方法（doGet、doPost 等），当服务器决定将实例销毁的时候调用其 destroy 方法。

与 CGI 的区别在于 Servlet 处于服务器进程中，它通过多线程方式运行其 service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁。而 CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于 Servlet。

3.5.4　Servlet 的配置

总的说来 Servlet 的配置包括 Servlet 的名字、Servlet 的类（如果是 JSP，就指定 JSP 文件）、初始化参数、启动装入的优先级、Servlet 的映射以及运行的安全设置。

一个完整的 Servlet 配置如下。

```
<servlet>
  <description>Study Servlet Config</description>
                                    <!--Servlet 有关部署描述信息-->
  <display-name>HelloWorld Config</display-name>
                                    <!--XML 编辑器显示的名字名称-->
  <servlet-name>HelloWorld</servlet-name>  <!--引入 Servlet 类名称-->
  <servlet-class>com.cn.javaweb.servlet.ch3.HelloWorldServlet</servle
  t-class>                          <!--引入 Servlet 类-->
  <!--初始化参数-->
  <init-param>
    <!--初始化参数连接驱动名称-->
    <param-name>jdbcDriver</param-name>
    <!--初始化参数值（驱动名称）-->
    <param-value>com.mysql.jdbc.Driver</param-value>
  </init-param>
  <init-param>
     <!--初始化参数 url-->
    <param-name>url</param-name>
     <!--初始化参数 url 名称-->
    <param-value>127.1.1.1</param-value>
  </init-param>
    <!-- 启动装入的优先级 -->
  <load-on-startup>30</load-on-startup>
</servlet>
 <!-- seervlet-mapping 元素将 URL 模式映射到某个 Servlet -->
<servlet-mapping>
    <!--定义 Servlet 名称，该名称在整个应用中是唯一的-->
  <servlet-name>HelloWorld</servlet-name>
    <!--配置访问路径，输入*/hello 访问-->
  <url-pattern>/hello</url-pattern>
</servlet-mapping>

<servlet-mapping>
  <!--配置 Servlet 名称-->
  <servlet-name>HelloWorld</servlet-name>
  <!--配置访问路径，输入 count/*访问-->
  <url-pattern>/count/*</url-pattern>
</servlet-mapping>
  <!--session-timeout 元素用来指定默认的会话超时时间间隔，以分钟为单位。该元素值必须为
```

```
  整数。如果 session-timeout 元素的值为零或负数，则表示会话将永远不会超时-->
  <session-config>
  <session-timeout>30</session-timeout>
</session-config>
```

1．配置Servlet的名字、类和其他杂项

在配置 Servlet 时，首先必须指定 Servlet 的名字和 Servlet 的类（如果是 JSP，必须指定 JSP 文件的位置）。另外，可以选择性地给 Servlet 增加一定的描述，并且指定它在部署时显示的名字，部署时显示的 icon。实例代码如下：

```
 <description>Study Servlet Config</description>   <!--Servlet 描述-->
 <display-name>HelloWorld Config</display-name>    <!--显示的名字名称-->
<servlet-name>HelloWorld</servlet-name>        <!--引入 Servlet 类名称-->
<servlet-class> com.cn.javaweb.servlet.ch3.HelloWorldServlet
</servlet-class>                                   <!--引入 Servlet 类-->
```

2．初始化参数

初始化参数配置后，在 Serlvet 中可以取得。

```
<!--初始化参数-->
<init-param>
     <!--初始化参数连接驱动名称-->
     <param-name>jdbcDriver</param-name>
     <!--初始化参数值（驱动名称）-->
     <param-value>com.mysql.jdbc.Driver</param-value>
 </init-param>
<init-param>
      <!--初始化参数 url-->
     <param-name>url</param-name>
     <!--初始化参数 url 名称-->
     <param-value>127.1.1.1</param-value>
</init-param>
```

3．启动装入的优先级

启动装入的优先级通过<load-on-startup></load-on-startup>来配置。

```
<load-on-startup>1</load-on-startup>
```

当启动 Web 容器时，用 load-on-startup 元素自动将 Servlet 加入内存。加载 Servlet 就意味着实例化这个 Servlet，并调用它的 init 方法。可以使用这个元素来避免第一个 Servlet 请求的响应因为 Servlet 载入内存所导致的任何延迟。

如果 load-on- startup 元素存在，而且也指定了 jsp-file 元素，则 JSP 文件会被重新编译成 Servlet，同时产生的 Servlet 也被载入内存。

load-on-startup 元素的内容可以为空或者是一个整数。这个值表示由 Web 容器载入内存的顺序。举个例子，如果有两个 Servlet 元素都含有 load-on-startup 子元素，则 load-on-startup 子元素值较小的 Servlet 将先被加载。如果 load-on- startup 子元素值为空或负值，则由 Web 容器决定什么时候加载 Servlet。如果两个 Servlet 的 load-on-startup 子元素值相同，则由 Web 容器决定先加载哪一个 Servlet。

4．Servlet的映射

可以给一个 Servlet 做多个映射，这样我们可以通过不同的方式来访问这个 Servlet。

```
 <servlet-mapping>
<servlet-name>HelloWorld</servlet-name>   <!--配置 Servlet 名称-->
<url-pattern>/hello</url-pattern>  <!--配置访问路径，输入*/hello 访问-->
</servlet-mapping

<servlet-mapping>
<servlet-name>HelloWorld</servlet-name>    <!--配置 Servlet 名称-->
<url-pattern>/count/*</url-pattern> <!--配置访问路径，输入 count/*访问-->
</servlet-mapping>
```

根据以上配置可以用下列 URL 来访问该 Servlet。在浏览器中输入下面两种 URL 都可以访问到该 Servlet。具体 URL 如下所示。

http://localhost/HibernateStudy/hello

或者

http://localhost/HibernateStudy/count/***

3.5.5　Servlet 使用

举例：HelloWorldServlet.java 代码如下。

```
package com.cn.javaweb.servlet.ch3;
  。。。
public class HelloWorldServlet extends HttpServlet {

    private String  driver;
    private String url;
    private String  password;
    private String user;
    @Override
    protected void doGet(HttpServletRequest req,  HttpServletResponse re
    sp)
            throws ServletException,  IOException {

        PrintWriter out = resp.getWriter();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>HelloWorld</title>");
        out.println("</head>");

        out.println("<body bgcolor=/"lightblue/">");
        out.println("<hr>");
        out.println("DRVER is " + jdbcDriver);
        out.println("URL   is " + url);
        out.println("</body>");
        out.println("</html>");
    }

    @Override
    protected void doPost(HttpServletRequest req,  HttpServletResponse r
```

```
    esp)
            throws ServletException,  IOException {
        doPost(req,  resp);
    }

    @Override
    public void init() throws ServletException {
        super.init();
        driver = getInitParameter("driver");
        url = getInitParameter("url");
        user = getInitParameter("user");
        password = getInitParameter("password");
    }
}
```

在浏览器中输入如下所示的 URL，然后按回车键或者单击前进按钮，结果画面上的信息就会在屏幕上显示出来。

http://localhost/HibernateStudy/hello

具体输出如下所示。

```
DRVER is com.mysql.jdbc.Driver  URL is 127.1.1.1
```

通过上面的实例我们看到，原来 Servlet 实现起来就是这么简单。不过可能我们对其中的一些代码还不知道到底什么意思，没有关系，通过下面的讲解，您就会很清楚了！

Servlet 定义了很多类接口，Servlet 的类接口可以从以下几个方面进行分类。

- Servlet 实现相关：定义了用于实现 Servlet 相关的类和方法。
- Servlet 配置相关：主要包括 ServletConfig 接口。
- Servlet 异常相关：Servlet API 有两个异常——ServletException 和 UnavailableException。
- 请求和响应相关：用于接收客户端的请求，并且做出对应的响应。
- 会话跟踪：用于跟踪和客户端的会话。
- Servlet 上下文：通过这个接口可以在多个 Web 应用程序中共享数据。
- Servlet 协作：主要是 RequestDispatcher 接口，用于进行视图派发。
- 过滤：Cookie 和 HttpUtils 类。

1．Servlet实现相关

（1）Servlet。Servlet 的声明如下。

```
public  interface  Servlet
```

说明：这个接口是所有 Servlet 必须直接或间接实现的接口。它定义了以下方法。

- init（ServletConfig config）：用于初始化 Servlet。
- destroy()：销毁 Servlet。
- getServletInfo：获得 Servlet 的信息。
- getServletConfig：获得 Servlet 配置相关信息。
- Service(ServletRequest req, ServletResponse res)：运行应用程序逻辑的入口点，它接收两个参数，ServletRequest 表示客户端请求信息，ServletResponse 表示对客户端

的响应。

（2）GenericServlet。GenericServlet 声明如下。

```
public abstract class GenericServlet implements Servlet,
ServletConfig,java.io.Serializable
```

说明：GenericServlet 提供了对 Servlet 接口的基本实现。它是一个抽象类，它的 service() 方法是一个抽象方法，GenericServlet 的派生类必须直接或间接实现这个方法。

（3）HttpServlet。HttpServlet 的声明如下。

```
public abstract class HttpServlet extends GenericServlet implements
java.io.Serializable
```

说明：HttpServlet 类是针对使用 Http 协议的 Web 服务器的 Servlet 类。HttpServlet 类通过执行 Servlet 接口，能够提供 Http 协议的功能。

HttpServlet 的子类必须实现以下方法中的一个。

- ❑ doGet：支持 Http Get 请求。
- ❑ doPost：支持 Http Post 请求。
- ❑ doPut：支持 Http Put 请求。
- ❑ doDelete：支持 Http Delete 请求。
- ❑ init 和 destroy：管理 Servlet 占用的资源。
- ❑ getServletInfo：获得 Servlet 自身的信息。

2．Servlet配置相关

javax.servlet.ServletConfig 接口代表了 Servlet 的配置，Servlet 的配置包括 Servlet 的名字、Servlet 的初始化参数和 Servlet 上下文。

下面代码示例表示了一个 Servlet 的配置。

```
<servlet>
  <servlet-name>HelloWorld</servlet-name>
  <servlet-class> com.cn.javaweb.servlet.ch3.HelloWorldServlet </servlet-
  class>
  <init-param>
    <param-name>encoding</param-name>      <!--配置编码方式 -->
    <param-value>UTF-8</param-value>       <!--配置编码方式为 UTF-8 -->
  </init-param>
</servlet>

<servlet-mapping>
<servlet-name>HelloWorld</servlet-name>    <!--配置 Servlet 名称-->
<url-pattern>/hello</url-pattern>    <!--配置访问路径，输入*/hello 访问-->
</servlet-mapping
```

说明：ServletConfig 定义了 Servlet 的配置相关参数。

声明：public interface ServletConfig。

主要方法如下。

- getInitParameter(String name)：返回特定名字的初始化参数。如上配置中 getInitParameter("encoding")，那么将返回“UTF-8”字符串。
- getInitParameterNames()：返回所有的初始化参数的名字。
- getServletContext()：返回 Servlet 的上下文对象的引用。

3．Servlet异常相关

（1）ServletException 异常。

声明：public class ServletException extends Exception。

说明：它包含几个构造方法和一个获得异常原因的方法，获得异常原因的方法为：getRootCause()。

（2）UnavailableException 异常。

声明：public class UnavailableException。

说明：当 Servlet 或者 Filter 暂时或永久不可用时，抛出该异常。

4．请求和响应相关

对于请求和响应相关的接口主要有以下几种。

- ServletRequest：代表了 Servlet 的请求，它是一个高层接口，HttpServletRequest 是它的子接口。
- ServletResponse：代表了 Servlet 的响应，它是一个高层接口，HttpServletResponse 是它的子接口。
- ServletInputStream：Servlet 的输入流。
- ServletOutputStream：Servlet 的输出流。
- ServletRequestWrapper：是 ServletRequest 的实现。
- ServletResponseWrapper：是 ServletResponse 的实现。
- HttpServletRequest：代表了 Http 的请求，继承了 ServletRequest 接口。
- HttpServletResponse：代表了 Http 的响应，继承了 ServletResponse 接口。
- HttpServletRequestWrapper：是 HttpServletRequest 的实现。
- HttpServletResponseWrapper：是 HttpServletResponse 的实现。

由于 HttpServletRequest 和 HttpServletResponse 接口是我们经常用到的，所以下面重点介绍 HttpServletRequest 和 HttpServletResponse 接口。

（1）HttpServletRequest。这个接口中最常用的方法就是获得请求中的参数，这个参数是客户端表单中的数据。此接口可以获取客户端传送的参数名称，也可以获取客户端正在使用的通信协议，可以获取远端主机名和其 IP 地址等。其中的一些重要方法如下。

- getCookies()：获得客户端发送的 Cookie。
- getSession()：返回和客户端关联的 Session，如果没有给客户端分配 Session，则返回 null。

- getSession(boolean create)：和上一方法类似，不同的是，如果没有给客户端分配，则创建一个新的 Session 并返回。
- getParameter(String name)：获得请求中名为 name 的参数的值，如果没有该参数，则返回 null。
- getParameterValues(String name)：获得请求中名为 name 的参数的值，这个值往往是 checkbox 或者 select 控件提交的，获得的值是一个 String 数组。

（2）HttpServletResponse。HttpServletResponse 的声明如下。

```
public interface HttpServlet Response extends ServletResponse
```

说明：它代表了对客户端请求的响应。在响应过程中一些常用方法如下。

- addCookie(Cookie cookie)：在响应中增加一个 Cookie。
- encodeURL(String url)：使用 url 和一个 SessionId 重写这个 URl。
- sendRedirect(String location)：把响应发送到另一个页面或者 Servlet 进行处理。
- setContentType(String type)：设置响应的 MIME 类型。
- setCharacterEncoding(String charset)：设置响应的字符编码类型。

5．会话跟踪

和会话跟踪有关的类和接口有：HttpSession。HttpSession 的声明如下。

```
public interface HttpSession
```

说明：这个接口被 Servlet 引擎用来实现 Http 客户端和 Http 会话之间的关联。这种关联可能在多处连接和请求中持续一段给定的时间。Session 用来在无状态的 Http 协议下越过多个请求页面来维持状态和识别用户。一个 Session 可以通过 Cookie 或重写 URl 来维持。

会话接口中的常用方法如下。

- getCreationTime()：返回创建 Session 的时间。
- getId()：返回分配给这个 Session 的标识符。一个 Http Session 的标识符是一个由服务器来建立和维持的唯一的字符串。
- getLastAccessedTime()：返回客户端最后一次发出与这个 Session 有关的请求的时间，如果这个 Session 是新建立的就返回–1。
- getMaxInactiveInterval()：返回一个秒数，这个秒数表示客户端在不发出请求时，Session 被 Servlet 引擎维持的最长时间。在这个时间之后，Session 可能被 Servlet 引擎终止。如果这个 Session 不会被终止，则返回–1。
- getValue(String name)：返回一个以给定名字绑定到 Session 上的对象。如果不存在这样的绑定，则返回空值。
- getValueNames()：以一个数组返回绑定到 Session 上的所有数据的名称。
- invalidate()：这个方法会终止这个 Session。所有绑定在 Session 上的数据都会被清除。
- isNew()：返回一个布尔值以判断这个 Session 是不是新的。如果一个 Session 已经被服务器建立但还没有收到相应的客户端的请求，这个 Session 将被认为是新的。

这意味着，这个客户端还没有加入会话或没有被会话公认。在它发出下一个请求时还不能返回适当的 Session 认证信息。当 Session 无效后，再调用这个方法会抛出一个 IllegalStateException 异常。

- putValue(String name, Object value)：以给定的名字绑定给定的对象到 Session 中。已存在的同名的绑定会被重置，这时会调用 HttpSessionBindingListerer 接口的 valueBound 方法。
- removeValue(String name)：取消绑定。如果未找到给定名字的绑定则什么也不做。这时会调用 HttpSessionBindingListerer 接口的 valueBound 方法。
- setMaxInactiveInterval(int interval)：设置一个秒数，表示客户端在不发出请求时，Session 被 Servlet 引擎维持的最长时间。

6．Servlet上下文

servlet 上下文相关的接口有 ServletContext。ServletContext 声明如下。

```
public  interface  ServletContext
```

注意：在服务器上使用 Session 对象来维持单个客户相关的状态，而当为多个用户的 Web 应用维持一个状态时，则应使用 Servlet 环境（Context）。

ServletContext 对象表示一组 Servlet 共享的资源，在 Servlet API 的 1.0 和 2.0 中，ServletContext 对象仅仅提供了访问有关 Servlet 环境信息的方法。例如提供了访问服务器名称、MIME 类型映射等方法和可以将信息写入服务器日志文件的 log()方法。

ServletContext 对象访问有关 Servlet 环境信息常用方法如下。

- getAttribute(String name)：获得 ServletContext 中名称为 name 的属性。
- getAttribute(String uripath)：返回给定的 uripath 的应用的 Servlet 上下文。
- removeAttribute(String name)：删除 ServletContext 中名称为 name 的属性。
- setAttribute(String name,Object obj)：在 ServletContext 中设置一个属性，名称为 name，值为 obj。

7．Servlet协作

Servlet 协作主要是 RequestDispatcher 接口，它可以把一个请求转发到另一个 Servlet。RequestDispatcher 的声明如下。

```
public  interface  RequestDispatcher
```

RequestDispatcher 接口包含两个方法，具体如下。

- forward(ServletRequest req, ServletResponse res)：把请求转发到服务器上的另一个资源（Servlet、Jsp 和 Html）。
- include(ServletRequest req, ServletResponse res)：把服务器上的另一个资源（Servlet、Jsp 和 Html）包含到响应中。

8．过滤

在 Web 应用中实施过滤是我们常使用的技术，通过过滤，可以对请求进行统一编码，对请求进行认证等。每个 Filter 可能只担任很少的任务，多个 Filter 可以互相协作，通过这种协作，可以完成一个复杂的功能。

（1）Filter。Filter 的声明如下。

```
public  interface  Filter
```

说明：它是 Filter 必须实现的接口，它包含以下方法。

- init(FilterConfig fc)：这个方法初始化 Filter。
- doFilter(ServletRequest req, ServletResponse res, FilterChain chain)：Filter 的业务方法就在这里实现。
- destroy()：释放 Filter 占用的资源。

（2）FilterChain。FilterChain 的声明如下。

```
public  interface  FilterChain
```

说明：它是代码的过滤链，通过这个接口把过滤的任务在不同的 Filter 之间转移。

它包含了一个方法：doFileter(ServletRequest req, ServletResponse res)，通过这个方法来调用下一个 Filter，如果没有下一个 Filter，那么将调用目标的资源。

（3）FilterConfig。FilterConfig 的声明如下。

```
public  interface  FilterConfig
```

代表了 Filter 的配置，和 Servlet 一样，Filter 也有一些配置信息，比如 Filter 的名字和初始化参数等。它包含了以下方法。

- getFilterName()：返回 Filter 的名字。
- getInitParameter(String name)：获得名字为 name 的初始化参数。
- getServletContext()：返回这个 Filter 所在的 Servlet 上下文对象。
- getInitParameterNames()：获得 Filter 配置中的所有初始化参数的名字。

3.6　桩功之六：JavaBean 的认知和使用

本节将带您认识 JavaBean，详细介绍 JavaBean 的特点、属性和方法。并通过具体实例向读者展示 JavaBean 的命名规则，让读者知晓什么是 JavaBean 以及如何使用 JavaBean 实现实体类。通过本节的学习，我们就可以掌握 JavaBean 的一些内部规则，了解使用 JavaBean 管理数据库问题。下面我们就来看一看什么是 JavaBean。

3.6.1　什么是 JavaBean

JavaBean 行业内通常称为 Java 豆，带点美哩口味，飘零着咖啡的味道。JavaBean 是基

于 Java 的组件模型，由属性、方法和事件 3 部分组成。在该模型中，JavaBean 可以被修改或与其他组件结合以生成新组件或完整的程序。它又是一种 Java 类，通过封装成为具有某种功能或者处理某个业务的对象。因此，可以通过嵌在 JSP 页面内的 Java 代码访问 Bean 及其属性。

为写成 JavaBean，类必须是具体的和公共的，并且具有无参数的构造器。JavaBean 通过提供符合一致性设计模式的公共方法将内部域暴露成员属性。在业内众所周知，属性名称符合这种模式，其他 Java 类可以通过自身机制发现和操作这些 JavaBean 属性。在计算机编程中代表 Java 构件（EJB 的构件），通常有 Session bean、Entity bean 和 MessageDrivenBean 三大类。

- Session bean：会话构件，是短暂的对象，运行在服务器上，并执行一些应用逻辑处理，它由客户端应用程序建立，其数据需要自己来管理。分为无状态和有状态两种。
- Entity bean：实体构件，是持久对象，可以被其他对象调用。在建立时指定一个唯一的标识，并允许客户程序根据实体 bean 标识来定位 beans 实例。多个实体可以并发访问实体 bean，事物间的协调由容器来完成。
- MessageDriven Bean：消息构件，是专门用来处理 JMS（Java Message System）消息的规范（EIB 2.0）。JMS 是一种与厂商无关的 API，用来访问消息收发系统，并提供了与厂商无关的访问方法，以此来访问消息收发服务。JMS 客户机可以用来发送消息而不必等待回应。

3.6.2　JavaBean 的特点

按照 Sun 公司的定义，JavaBean 是一个可重复使用的软件组件。实际上 JavaBean 是一种 Java 类，通过封装属性和方法成为具有某种功能或者处理某个业务的对象，简称 bean。由于 JavaBean 是基于 Java 语言的，因此 JavaBean 不依赖平台，具有以下特点。

- 可以实现代码的重复利用。
- 易编写、易维护、易使用。
- 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。

编写 JavaBean 就是编写一个 Java 的类，所以您只要会写类就能编写一个 bean，这个类创建的一个对象称作一个 bean。为了能让使用这个 bean 的应用程序构建工具（比如 JSP 引擎）知道这个 bean 的属性和方法，需要在类的方法命名上遵守以下规则。

- 如果类的成员变量的名字是 person，那么为了更改或获取成员变量的值，即更改或获取属性，在类中可以使用两个方法：getPerson()，用来获取属性 person；setPerson()，用来修改属性 person。
- 对于 boolean 类型的成员变量，即布尔逻辑类型的属性，允许使用 is 代替上面的 get 和 set。
- 类中方法的访问属性都必须是 public 的。

- 类中如果有构造方法，那么这个构造方法也是 public 的并且是无参数的。

3.6.3　JavaBean 的属性

JavaBean 的属性与一般 Java 程序中所指的属性，或者说与所有面向对象的程序设计语言中对象的属性是一个概念，在程序中的具体体现就是类中的变量。属性分为四类：即单值（Simple）、索引（Index）、关联（Bound）和约束（Constrained）属性。本小节将对这些属性进行详细说明。

1．单值属性

单值（Simple）属性是最普通的属性类型，该类属性只有一个单一的数据值，该数据值的数据类型可以是 Java 中的任意数据类型，包括类和接口等类型。

定义了属性，还需定义对应的访问方法，一般每个单值属性都伴随一对 get/set 方法。属性名与该属性相关的 get/set 方法名对应。例如如果有一个名为 dog 的属性，则会有 setDog 和 getDog 方法。

另外，布尔（Boolean）属性是一种特殊的单值属性，它只有两个允许值：true 和 false，如果有一个名为 dog 的布尔属性，则可以通过 isDog 方法访问。

2．索引属性

索引属性是指 JavaBean 中数组类型的成员变量。使用与该属性对应的 set/get 方法可取得数组的值。索引属性通过对应的访问方法设置或取得该属性中某个元素的值，也可以一次设置或取得整个属性的值。如果需要定义一批同类型的属性，使用单值属性就会显得非常烦琐，为解决此问题，JavaBean 中提供了索引（Indexed）属性。

3．关联属性

关联（Bound）属性是指当该种属性的值发生变化时，要通知其他的对象。每次属性值改变时，这种属性就触发一个 PropertyChange 事件。事件中封装了属性名、属性的原值和属性变化后的新值。这种事件传递到其他的 Beans，至于接收事件的 Beans 应做什么动作，由其自己定义。

4．约束属性

JavaBean 的属性如果改变时，相关的外部类对象首先要检查这个属性改变的合理性再决定是否接受这种改变，这样的 JavaBean 属性叫约束（Constrained）属性。当约束属性的改变被拒绝时，改变约束属性的方法产生一个约束属性改变异常（PropertyVetoException），通过这个异常处理，JavaBean 约束属性还原回原来的值，并为这个还原操作发送一个新的属性修改通知。

约束属性的改变可能会被拒绝，因此它的 set 方法（例如 setDog()）与一般其他 JavaBean 属性的 setDog 也有所不同。约束属性的方法如下。

```
public  void  setDog(dogType newDog) throws PropertyVetoException
```

3.6.4　JavaBean 的方法

在 JavaBean 中的函数和过程统称为方法，通过方法来改变和获取上面介绍的各种属性值。方法可以分为构造方法、访问方法和普通方法等。本小节将学习创建和使用这些方法。

1．构造方法

JavaBean 的构造方法就是对 JavaBean 的属性及其方法进行初始化，即对所定义的属性及方法设一个初始值，构造方法名要和 JavaBean 的类名相同。下面的代码定义的就是一个 JavaBean 及其构造方法。

```
public class TimeShow{
   //定义属性
   private int hour;
   private int minute;
  //构造方法，对属性进行初始化操作，其名字应该与 bean 的名字相同
  public TimeShow(){
  //初始化日期对象
Date now = new Date();
 //获取小时
hour = now.getHours();
//获取分钟
minute = now.getMinutes();
}
}
```

2．访问方法

访问方法就是对组件中定义的属性的访问，包括读和写两种访问方式。在定义了 Bean 的属性，并通过构造方法将其初始化后，要让其他程序访问 Bean 的这些属性，就必须为其创建访问方法。

读就是一种用于取出 Bean 属性的值的取值函数，即 getter；而写则是一种用于设置 Bean 属性的赋值函数，即 setter。以下列出的就是 Bean 属性访问方法的具体语法格式。

```
public void setPropertyName(PropertyType value);//给属性赋值，即写方法
public PropertyType getPropertyName();          //读取属性值，即读方法
```

3．普通方法

除了对属性的访问方法外，还可以在 Bean 创建一般方法来实现对函数的调用，只要将 Bean 中的一般方法定义成公有的方法，就可以供其他程序调用。例如下面代码是一个实现求阶乘函数的一般方法。

代码 Multiple.java。

```
package jsp.examples.mybean;
public class Multiple{
    public int Multi(int j){
       //定义整形变量，并初始化为 1
```

```
        int x = 1;
       //循环函数，输出 1*2*3*…j 的结果
        for (int i = 1; i <= j; ++i)
        { x = x * i; }
        return x;
    }
}
```

3.6.5　JavaBean 的使用

由于 JavaBean 是为了重复使用的程序段落，具有“Write once， run anywhere， reuse everywhere”，即“一次性编写，任何地方执行，所有地方可重用”的特点，所以可以为 JSP 平台提供一个简单的、紧凑的和优秀的问题解决方案，能大幅度提高系统的功能上限、加快执行速度，而且不需要牺牲系统的性能。同时，采用 JavaBean 技术可以使系统更易于维护，因此极大地提高了 JSP 的应用范围。本小节将详细学习如何在 JSP 中应用 Bean 组件。

前面章节介绍 JSP 时介绍了通过 JSP 标记中的<jsp:useBean>动作来调用 JavaBean，现在我们学习完了 JavaBean 的编写，再来复习一下这个标记。

```
<jsp:useBean id="beanId" scope="page|request|session|application" class=
"package.class" />
```

（1）首先我们通过标记中的 id 属性标记 Bean，以使 JSP 页面的其余部分可以正确地识别该 Bean。

（2）其次，使用 scope 属性来确定该 Bean 的使用范围。scope 属性所决定的使用范围，可以参考我们在上小节中所作的介绍。

（3）最后，class 属性通知 JSP 页面从何处查找 Bean，即找到 Bean 的.class 文件。在此我们必须同时指定 JavaBean 的包（package）名和类（class）名，即 class="package.class"，否则 JSP 引擎将无法找到相应的 Bean。

3.7　桩功之七：XML 配置

本节主要介绍 XML 的配置，将详细介绍什么是 XML，及 XML 文档的结构、内容编辑与解析等内容。

近年来，随着 Web 的应用越来越广泛和深入，人们渐渐觉得 HTML 不够用了，HTML 过于简单的语法严重地阻碍了用它来表现复杂的形式。尽管 HTML 推出了一个又一个新版本，已经有了脚本、表格、帧等表达功能，但始终满足不了不断增长的需求。另一方面，这几年来计算机技术的发展也十分迅速，已经可以实现比当初发明 HTML 时复杂得多的 Web 浏览器，所以开发一种新的 Web 页面语言是必要的，也是可能的。

有人建议直接使用 SGML 作为 Web 语言，这固然能解决 HTML 遇到的困难。但是 SGML 太庞大了，用户学、用不方便尚且不说，要全面实现 SGML 的浏览器就非常困难，于是自然会想到仅使用 SGML 的子集，使新的语言既方便使用又实现容易。正是在这种形

势下，Web 标准化组织 W3C 建议使用一种精简的 SGML 版本——XML 应运而生了。

3.7.1　XML 语言概述

XML 是一个精简的 SGML，它将 SGML 的丰富功能与 HTML 的易用性结合到 Web 的应用中。XML 保留了 SGML 的可扩展功能，这使 XML 从根本上有别于 HTML。XML 要比 HTML 强大得多，它不再是固定的标记，而允许定义数量不限的标记来描述文档中的资料，允许嵌套的信息结构。HTML 只是 Web 显示数据的通用方法，而 XML 提供了一个直接处理 Web 数据的通用方法。HTML 着重描述 Web 页面的显示格式，而 XML 着重描述 Web 页面的内容。

总之，XML 使用一个简单而灵活的标准格式，为基于 Web 的应用提供了一个描述数据和交换数据的有效手段。HTML 描述了显示全球数据的通用方法，而 XML 提供了直接处理全球数据的通用方法。XML 是一种元语言，可以定义其他的语言，并且它的标记是用户定义的，从理论上讲，其类型的数量可以是无限的。XML 的前景被人们看好，有人预言在 21 世纪 XML 将成为世人皆知的“世界语”。

3.7.2　XML 文档结构

XML 文档结构包含下面三个部分。

- 声明部分，声明该文档是一个 XML 文档。
- 定义部分，定义 XML 数据的类型以及所使用的 DTD（可选）。
- 内容部分，用 XML 标签和注释标注过的文档内容。

1．XML声明

XML 文档以 XML 声明开头，声明本文档是一个 XML 文档。一般书写格式如下。

```
<? xml version encoding standalone?>
```

定义符<?和?>表示这是一条给 XML 解析器的处理指令。虽然声明这条语句是可有可无的，但考虑到以后的兼容，建议读者还是写上为好。随着语言的进一步发展，以后的浏览器如果知道文档所用的 XML 版本的话，将是有好处的。

注意：XML 声明语句必须全部用小写。

在上面的声明中 version 表示 XML 的标准版本号，encoding 表示的是文档所用的编码，standalone 用来指定在 XML 文档被解析之前，是否使用外部或内部 DTD，它的值只能是 yes 或 no。如果为 no，表示使用外部 DTD；如果为 yes 表示使用内部 DTD；如果不使用 DTD，则不使用这个属性。

看下面的实际例子。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes">
```

表示 XML 版本为 1.0，文档所用编码方式为 UTF-8，使用内部 DTD。在 XML 声明之

后，紧接着是类型定义部分，定义 XML 文档中数据的类型。

2．文档定义类型（DTD）

（1）DTD 的作用。DTD 是用来定义 XML 文档内容的结构的，以便按统一的格式存储信息。DTD 规定了 XML 文档中可以出现哪些元素，这些元素是必须的还是可选的，这些元素有什么属性以及它们之间的相互位置关系等等。XML 允许用户为自己的应用程序定义专用的 DTD，这样用户就可以完成检查文档结构和内容的过程了。这一检查过程称为有效化，严格依从一个 DTD 的 XML 文档被称作有效文档。

（2）创建 DTD。创建 DTD 的过程与在数据库里创建数据表是类似的。在 DTD 中，用户定义用来表示数据的元素，然后规定数据的结构，并规定这个元素是可选的还是必须的，这就好比创建数据表的列；把数据存入 XML 文档，就好比添加数据表的记录。

XML 文档使用的元素可以在内部 DTD 中定义，也可以在外部 DTD 中定义。

内部 DTD， DTD 可以作为文档的一部分直接放到文档里面，这样的 DTD 只能用于包含它的这个文档，别的文档就不能使用了。创建内部 DTD 的语法如下。

```
<!DOCTYPE rootelement
[element and attribute declarations]
>
```

<!DOCTYPE 标识文档类型定义的开始，属性 rootelement 指明根元素名字。

外部 DTD，外部 DTD 是一个单独的文件，存放 XML 文档中可以使用的全部元素及属性的定义。您可以在多个文档中同时使用同一个 DTD，以便保持多个文档之间数据结构的一致性。在 XML 文档中引用外部 DTD 的语法如下。

```
<!DOCTYPE rootelement [PUBLIC|SYSTEM] "name-of-file">
```

其中，DOCTYPE 标识这是文档类型定义部分；rootelement 代表根元素；PUBLIC 表示这个 DTD 是存放在公用服务器上的；SYSTEM 表示这个 DTD 是存放在本地计算机系统上的；Name-of-file 是被引用的 DTD 文件的名称。

表 3-1 是 DTD 中使用的部分专用字符及其含义。

表 3-1　DTD中使用的部分专用字符及其含义

DTD 字符	含　义	举　例	描　述
,	指定顺序中的“与”	Firstname ， Lastname	Firstname 与 Lastname
|	“或”	Firstname | Lastname	Firstname 或 Lastname
?	可选项，只能出现一次	Lastname?	可以不出现 Lastname，但如果使用，则只能出现一次
()	用于组成元素	(Firstname | Lastname)，Address	一个 Firstname 或 Lastname 元素必须出现在 Address 元素之前
*	该元素可以不出现也可以出现多次	(Firstname | Lastname)*	可以以任何顺序出现任意个数 Firstname 或 Lastname 元素
+	该元素至少出现一次也可以出现多次	(Firstname +)	可以出现多个 Firstname 元素

（3）在 DTD 中定义元素。元素是 XML 文件的基本组成部分，每个元素都是用标签标识的一小段数据。标签包括了元素的名字和属性。XML 允许用于创建自己的元素集。因此，

元素名应该取得容易记忆，并且最好有一定的含义，让人一看到它，便对里面的数据有个大概的了解。XML 是大小写敏感的，所以要么全用大写，要么就一律用小写。定义元素的语法如下。

```
<!ELEMENT elementname content>
```

在 DTD 中，通过创建一个元素内容模型（element content model）来精确地规定一个元素中是否含有其他元素，可以出现多少次以及按什么顺序出现。如果元素中只包含别的元素，而不包含字符数据，我们就说它只含有元素内容。

（4）XML 中命名元素的规则如下。

- 元素名至少要含有一个字母（a～z 或 A～Z 中的一个）。
- 元素名可以用下划线（_）或冒号（:）开头。
- 第一个字符后面可以是一个或多个字母、数字、连字符、下划线或句号，但不能是空格和定位符（tab），至于标点符号只能使用连字符（–）和句点号（.）。

（5）元素类型。元素有空元素、自由元素和容器元素三种，如表 3-2 所示。

表 3-2　空元素、自由元素和容器元素

元 素 类 型	语　　法
空元素（Empty）	<!ELEMENT empty.element EMPTY>
自由元素（Unrestricted）	<!ELEMENT any.element ANY>
容器元素（Container）	<!ELEMENT TITLE(#PCDATA)>

PCDATA 表示 parsable character data，即可解析的字符数据。为了避免将这一关键字与普通的元素名混淆起来，在此关键字前加前缀字符#。

分析下面的标签结构。

```
<student>
<firstname> Java</firstname>
<lastname> Web</lastname>
<rollno> 49 </rollno>
<score> 70 </score>
</student>
```

要使上面的文档生效，必须创建一个 DTD，里面包含 student、firstname、lastname、rollno 和 score 等 5 个元素的定义。另外，还要规定这 5 个元素是必需的或可选的，以及规定顺序或任意排序，还有它们出现的次数。用户为这些规定编写元素定义，每个元素的定义可能不同。

譬如，如果 firstname 和 lastname 都是必需的元素，并且 firstname 要在 lastname 后面，那么 DTD 可以按如下这样编写。

```
<!ELEMENT student  (firstname，lastname)><!--元素内容-->
<!ELEMENT firstname (#PCDATA)><!--元素内容-->
<!ELEMENT lastname  (#CDATA)><!--元素内容-->
```

注意：数据类型#CDATA 表示元素包含字符型数据，解析器不解析这些数据，其中的标签是不作为标记的。数据类型#PCDATA 表示元素包含的数据将由解析器解析，其中的标签是被作为标记处理。

3.7.3　XML 文档内容编辑

XML 文档必须包含根元素。根元素是所有其他元素的父元素。XML 文档中的元素形成了一棵文档树。这棵树从根部开始，并扩展到树的最底端。

所有元素均可拥有子元素。

```
<root>
<child>
<subchild>.....</subchild>
</child>
</root>
```

父、子以及同胞等术语用于描述元素之间的关系。父元素拥有子元素。相同等级上的子元素我们称同胞（兄弟或姐妹）。关系如图 3-28 所示。

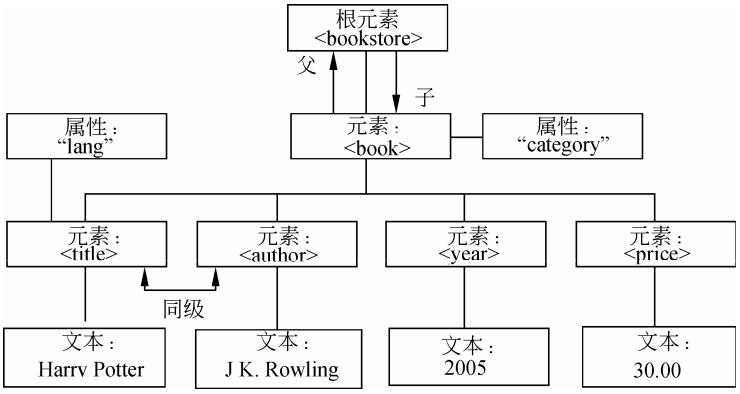

图 3-28　XML 元素关系

通过图 3-28 我们可以看出该关系图表示一本书，具体 XML 代码如下所示。

```
<bookstore> <!--根元素开始-->
<book category="COOKING"> <!--book 属性为类型-->
<title lang="en">Everyday Italian</title>  <!--book 名称-->
 <author>Giada De Laurentiis</author>  <!--图书作者-->
<year>2005</year>   <!--出版日期-->
<price>30.00</price>  <!--图书定价-->
</book>       <!--book 属性为类型结束-->
<book category="CHILDREN">
<title lang="en">Harry Potter</title>
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<book category="WEB">
<title lang="en">Learning XML</title>
<author>Erik T. Ray</author>
<year>2003</year>
<price>39.95</price>
</book>
</bookstore> <!--根元素结束-->
```

例子中的根元素是 <bookstore>。文档中的所有<book> 元素都被包含在 <bookstore> 中。<book> 元素有 4 个子元素：<title>、< author>、<year>和<price>。

3.7.4　XML 文档解析

在项目开发中需要用到 XML 技术，在针对 XML 文档的应用编程接口中，最主要的有 W3C 制定的 DOM（Document Object Method，文档对象模型）和由 David Megginson 领导的 SAX（SimpleAPI for XML，用于 XML 的简单 API）。这里对 XML 的 SAX 和 DOM 两种解析方式做下简单描述。

SAX 和 DOM 在实现过程中，分别侧重于不同的方面以满足不同的应用需求。DOM 为开发基于 XML 的应用系统提供了便利。它通过一种随机访问机制，使得应用程序利用该接口可以在任何时候访问 XML 文档中的任何一部分数据，也可以对 XML 文档中的数据进行插入、删除、修改和移动等操作。

在 DOM 中，文档的逻辑结构类似一棵树。文档、文档中的根、元素、元素内容、属性、属性值等都是以对象模型的形式表示的。DOM 的优点在于它在内存中保存文档的整个模型。这使得能以任何顺序访问 XML 元素。然而，对于大型文档来说，这样做可能不方便，因为它可能会用尽内存，或者当系统达到了它的极限时，机器的性能将会慢下来。

1．使用DOM解析XML文档

我们现在来看看 DOM 是如何解析 XML 的吧！同样的，我将从一个简单得不能再简单的例子来说明 DOM 是如何解析 XML 文档的，先让我们看看 XML 是什么内容吧。

（1）先建立一个 configure.xml，存放在 src 下的 xml 包下。

```
<?xml version="1.0" encoding="gbk"?>
<books>
<book email="****@126.com">
<name>java</name>
<price>30</price>
</book>
</books>
```

建立解析程序 DomParse.java，存放在 src 下的 com.domparse.xml 包下。

```
package com.domparse.xml;

。。。
import org.xml.sax.SAXException;
public class DomParse {
public DomParse() {

//得到 DOM 解析器的工厂实例
//javax.xml.parsers.DocumentBuilderFactory 类的实例就是我们要的解析器工厂
DocumentBuilderFactory domfac = DocumentBuilderFactory.newInstance();
try {
//通过 javax.xml.parsers.DocumentBuilderFactory 实例的静态方法
newDocumentBuilder()得到 DOM 解析器
DocumentBuilder dombuilder = domfac.newDocumentBuilder();

//把要解析的 XML 文档转化为输入流，以便 DOM 解析器解析它
```

```
InputStream is = new FileInputStream("src/xml/configure.xml");

//解析 XML 文档的输入流，得到一个 Document
//由 XML 文档的输入流得到一个 org.w3c.dom.Document 对象，以后的处理都是对 Document
  对象进行的
Document doc = dombuilder.parse(is);

//得到 XML 文档的根节点
//在 DOM 中只有根节点是一个 org.w3c.dom.Element 对象
Element root = doc.getDocumentElement();

//得到节点的子节点
//这是用一个 org.w3c.dom.NodeList 接口来存放它所有子节点的，还有一种轮循子节点的方法
NodeList books = root.getChildNodes();
if (books != null) {
for (int i = 0; i < books.getLength(); i++) {
Node book = books.item(i);
if (book.getNodeType() == Node.ELEMENT_NODE) {

//取得节点的属性值
//注意，节点的属性也是它的子节点。它的节点类型也是 Node.ELEMENT_NODE
String email = book.getAttributes().getNamedItem("email").getNodeValue(
);
System.out.println(email);

//轮循子节点
for (Node node = book.getFirstChild(); node != null; nodenode = node.ge
tNextSibling())
{
if (node.getNodeType() == Node.ELEMENT_NODE) {
if (node.getNodeName().equals("name")) {
String name = node.getNodeValue();
String name1 = node.getFirstChild()
.getNodeValue();
System.out.println(name);
System.out.println(name1);
}
if (node.getNodeName().equals("price")) {
String price = node.getFirstChild()
.getNodeValue();
System.out.println(price);
}
}
}
}
}
}
} catch (ParserConfigurationException e) {
e.printStackTrace();
} catch (FileNotFoundException e) {
e.printStackTrace();
} catch (SAXException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}

public static void main(String[] args) {
DomParse domp=new DomParse();
```

```
}
}
```

2．用SAX解析XML

SAX 提供了一种对 XML 文档进行顺序访问的模式，这是一种快速读写 XML 数据的方式。SAX 接口是事件驱动的，当使用 SAX 分析器对 XML 文档进行分析时，就会触发一系列事件，并激活相应的事件处理函数，从而完成对 XML 文档的访问。

SAX 处理 XML 的方式与 DOM 不同。SAX 解析器不是将 DOM 树解析和表现为输出，它是基于事件的，所以在 XML 被解析时，事件被发送给引擎。SAX 可以在文档的开始接收事件，也可以接收文档中的元素。使用这些事件可以构建一种结构。

因为 SAX 没有把 XML 文档完全地加载到内存中，所以需要的系统资源较少，是一个分析大型 XML 文档的高效 API。缺点是编写 SAX 比编写 DOM 复杂，这因为首先必须实现通知接口并维护状态，其次 SAX 不允许对文档进行随机访问，也没有提供像 DOM 那样的修改功能。

以下是一个用 SAX 解析 XML 实例，输出 XML 中的所有属性和标签值。

```
package com.saxparse.xml;
。。。
public class PraseXML extends DefaultHandler
{
 private Vector<String> tagName;
private Vector<String> tagValue;
private int step;
//开始解析 XML 文件
 public void startDocument() throws SAXException
 {
tagName = new Vector<String>();
tagValue = new Vector<String>();
step = 0;
}
//结束解析 XML 文件
public void endDocument() throws SAXException
{
for (int i = 0; i < tagName.size(); i++)
{
if (!tagName.get(i).equals("") || tagName.get(i) != null) {
System.out.println("节点名称: " + tagName.get(i));
System.out.println("节点值: " + tagValue.get(i));
}
}
}
/**
* 在解释到一个开始元素时会调用此方法，但是当元素有重复时可以自己写算法来区分
* 这些重复的元素 qName 是什么？ <name:page ll=""></name:page>这样写就会抛出
  SAXException 错误
* 通常情况下 qName 等于 localName
*/
public void startElement(String uri， String localName， String qName，
          Attributes attributes) throws SAXException
      {
        //节点名称
        tagName.add(qName);
```

```
        //循环输出属性
        for (int i = 0; i < attributes.getLength(); i++)
        {
          //获取属性名称
          System.out.println("属性名称：" + attributes.getQName(i));
          //获取属性值
          System.out.println("属性值："
              + attributes.getValue(attributes.getQName(i)));
        }

      }

      /**
       * 在遇到结束标签时调用此方法
       */
      public void endElement(String uri, String localName, String qName)
          throws SAXException
      {

        step = step + 1;
      }

      /**
       * 读取标签里的值，ch 用来存放某行的 xml 的字符数据，包括标签，初始大小是 2048，
       * 每解释到新的字符会把它添加到 char[]里。    * 注意，这个 char 字符会自己管理
         存储的字符，
       * 并不是每一行就会刷新一次 char，
       *start，length 是由 xml 的元素数据确定的，
       * 暂时找不到规律，以后看源代码
       *
       * 这里一个正标签、反标签都会被执行一次 characters，所以在反标签时不用获得其
       * 中的值
       */
      public void characters(char ch[], int start, int length)
          throws SAXException
      {
        //只要当前的标签组的长度一致，值就不赋，则反标签不被计划在内
        if (tagName.size() - 1 == tagValue.size())
        {
          tagValue.add(new String(ch, start, length));
        }
      }

      public static void main(String[] args)
      {
        String filename = "MyXml.xml";
        SAXParserFactory spf = SAXParserFactory.newInstance();
        try
        {
          SAXParser saxParser = spf.newSAXParser();
          saxParser.parse(new File(filename), new PraseXML());
        }
        catch (Exception e)
        {
          e.printStackTrace();
        }
      }
```

```
    public Vector getTagName()
    {
      return tagName;
    }

    public void setTagName(Vector tagName)
    {
      this.tagName = tagName;
    }

    public Vector getTagValue()
    {
      return tagValue;
    }

    public void setTagValue(Vector tagValue)
    {
      this.tagValue = tagValue;
    }
}
```

输出结果如下。

```
属性名称：personid
属性值：e01
属性名称：enable
属性值：true
属性名称：personid
属性值：e02
属性名称：enable
属性值：false
属性名称：personid
属性值：e03
属性名称：enable
属性值：true
节点名称：people
节点值：
节点名称：person
节点值：
节点名称：name
节点值：张三
节点名称：tel
节点值：5128
节点名称：email
节点值：txq512@sina.com
节点名称：person
节点值：
节点名称：name
节点值：meixin
节点名称：tel
节点值：5252525
节点名称：email
节点值：wnight88@sina.com
节点名称：person
节点值：
```

```
节点名称：name
节点值：yu
节点名称：tel
节点值：5389654
节点名称：email
节点值：yu@188.net
```

3．MyXml.xml文件内容

比较而言，DOM 和 SAX 各有自己的应用场合。DOM 适用于处理下面的问题：解析比较小的 XML 文件；需要对文档进行修改；需要随机对文档进行访问。SAX 适于处理下面的问题：对大型文档进行处理；只需要文档的部分内容；只需要从文档中得到特定信息。

```
<?xml version="1.0" encoding="UTF-8"?>
<!--在 people 节点中储存 3 个人物信息-->
<people>
<person personid="e01" enable="true">   <!--人物 id -->
   <name>张三</name>                     <!--人物姓名 -->
   <tel>5128</tel>                       <!--人物电话-->
   <email>txq512@sina.com</email>        <!--人物邮箱，下面相同，不再解释-->
  </person>

  <person personid="e02" enable="false">
    <name>meixin</name>
    <tel>5252525</tel>
    <email>wnight88@sina.com</email>
  </person>

  <person personid="e03" enable="true">
    <name>yu</name>
    <tel>5389654</tel>
    <email>yu@188.net</email>
  </person>
</people>
```

3.8　本 章 小 结

本书是意在将太极拳的精妙理论与 Java Web 开发的实际需求相结合，来达到用通俗易懂的武学理论来阐述 Java Web 开发知识的目的。本书是按照循序渐进的方式来介绍 Java Web 开发的，在准备篇章里，相信您已经了解了 Java Web 开发的内涵、前景以及如何学好 Java Web 开发，掌握了在动手 Java Web 开发之前的开发环境的搭建工作。让您从对从 Java Web 开发一知半解，到熟练地搭建 Java Web 开发的运行环境，并对 JDK、Tomcat 和 Eclipse 等 Java Web 开发的常用工具有了深刻的认识。

本章中，我们进入了本书的基础篇章，对 Java Web 开发的一些基础知识进行讲解。主要是对最基本的界面元素的介绍和使用，如 HTML、CSS、JavaScript、XML、JSP、Servlet 和 Javabean 等概念的介绍和使用。这些知识是 Java Web 开发需要的“桩功”，只有练好这些桩功，才有可能学好 Java Web 开发，所以这些对于一个 Java Web 开发者是必须掌握的知识。

HTML 是组织展示内容的标记语言，JavaScript 是客户端的脚本语言，CSS 是美化页面的样式表，这三种技术结合在一起构成了 Web 开发最基础的知识，所有的 Web 应用开发都是在这个基础之上进行的。

在本章的讲解中，对这三种技术的主要使用方法进行了比较细致的介绍，使读者可以迅速对 Web 开发的基础知识有一个宏观的清楚的认识，从而可以快速进入后面章节的学习，如果读者对这方面基础知识有更深一步了解的需要，就有必要参考相关的专题书籍。

本章在介绍完静态的网页知识之后，对 JSP、Servlet、JavaBean 和 XML 等的基本语法和使用等知识进行了系统的介绍，而且对于其中大部分的知识点都给出了具体示例，这些示例在具体的开发过程中都有很重要的参考价值，读者可以在这些示例程序的基础上进行尝试，试着修改其中的功能，只有这样才能对其运行原理有更深入的了解和体会，这就是学习程序语言的最基本、最有效的方法。

由于篇幅有限，还有很多知识点没有讲到，这些知识点将在后面的章节中介绍，所以请各位读者做好知识的不断复习和积累。