**本部分余下的内容会详细地介绍 HTTP 1.1中的状态码。这些状态码被分为五大类：**   
  
*100-199 用于指定客户端应相应的某些动作。   
200-299 用于表示请求成功。   
300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。   
400-499 用于指出客户端的错误。   
500-599 用于支持服务器错误。*   
  
HttpServletResponse中的常量代表关联不同标准消息的状态码。在servlet程序中，你会更多地用到这些常量的标识来使用状态码。例如：你一般会使用response.setStatus(response.SC\_NO\_CONTENT)而不是 response.setStatus(204)，因为后者不易理解而且容易导致错误。但是，你应当注意到服务器允许对消息轻微的改变，而客户端只注意状态码的数字值。所以服务器可能只返回 HTTP/1.1 200 而不是 HTTP/1.1 200 OK。   
  
100 (Continue/继续)  
如果服务器收到头信息中带有100-continue的请求，这是指客户端询问是否可以在后续的请求中发送附件。在这种情况下，服务器用100(SC\_CONTINUE)允许客户端继续或用417 (Expectation Failed)告诉客户端不同意接受附件。这个状态码是 HTTP 1.1中新加入的。   
  
101 (Switching Protocols/转换协议)  
101 (SC\_SWITCHING\_PROTOCOLS)状态码是指服务器将按照其上的头信息变为一个不同的协议。这是 HTTP 1.1中新加入的。   
  
200 (OK/正常)  
200 (SC\_OK)的意思是一切正常。一般用于相应GET和POST请求。这个状态码对servlet是缺省的；如果没有调用setStatus方法的话，就会得到200。   
  
201 (Created/已创建)  
201 (SC\_CREATED)表示服务器在请求的响应中建立了新文档；应在定位头信息中给出它的URL。  
  
202 (Accepted/接受)  
202 (SC\_ACCEPTED)告诉客户端请求正在被执行，但还没有处理完。   
  
203 (Non-Authoritative Information/非官方信息)  
状态码203 (SC\_NON\_AUTHORITATIVE\_INFORMATION)是表示文档被正常的返回，但是由于正在使用的是文档副本所以某些响应头信息可能不正确。这是 HTTP 1.1中新加入的。   
  
204 (No Content/无内容)  
在并没有新文档的情况下，204 (SC\_NO\_CONTENT)确保浏览器继续显示先前的文档。这各状态码对于用户周期性的重载某一页非常有用，并且你可以确定先前的页面是否已经更新。例如，某个servlet可能作如下操作：   
int pageVersion =Integer.parseInt(request.getParameter("pageVersion"));  
if (pageVersion >;= currentVersion) {  
   response.setStatus(response.SC\_NO\_CONTENT);  
} else {  
       // Create regular page  
}  
但是，这种方法对通过刷新响应头信息或等价的HTML标记自动重载的页面起作用，因为它会返回一个204状态码停止以后的重载。但基于JavaScript脚本的自动重载在这种情况下仍然需要能够起作用。可以阅读本书7.2 ( HTTP 1.1 Response Headers and Their Meaning/HTTP 1.1响应头信息以及他们的意义)部分的详细讨论。   
  
205 (Reset Content/重置内容)  
重置内容205 (SC\_RESET\_CONTENT)的意思是虽然没有新文档但浏览器要重置文档显示。这个状态码用于强迫浏览器清除表单域。这是 HTTP 1.1中新加入的。   
  
206 (Partial Content/局部内容)  
206 (SC\_PARTIAL\_CONTENT)是在服务器完成了一个包含Range头信息的局部请求时被发送的。这是 HTTP 1.1中新加入的。   
  
300 (Multiple Choices/多重选择)  
300 (SC\_MULTIPLE\_CHOICES)表示被请求的文档可以在多个地方找到，并将在返回的文档中列出来。如果服务器有首选设置，首选项将会被列于定位响应头信息中。   
  
301 (Moved Permanently)  
301 (SC\_MOVED\_PERMANENTLY)状态是指所请求的文档在别的地方；文档新的URL会在定位响应头信息中给出。浏览器会自动连接到新的URL。   
  
302 (Found/找到)  
与301有些类似，只是定位头信息中所给的URL应被理解为临时交换地址而不是永久的。注意：在 HTTP 1.0中，消息是临时移动(Moved Temporarily)的而不是被找到，因此HttpServletResponse中的常量是SC\_MOVED\_TEMPORARILY不是我们以为的SC\_FOUND。   
  
注意  
代表状态码302的常量是SC\_MOVED\_TEMPORARILY而不是SC\_FOUND。   
  
状态码302是非常有用的因为浏览器自动连接在定为响应头信息中给出的新URL。这非常有用，而且为此有一个专门的方法——sendRedirect。使用response.sendRedirect(url)比调用response.setStatus(response.SC\_MOVED\_TEMPORARILY)和response.setHeader("Location", url)多几个好处。首先，response.sendRedirect(url)方法明显要简单和容易。第二，servlet自动建立一页保存这一连接以提供给那些不能自动转向的浏览器显示。最后，在servlet 2.2版本（J2EE中的版本）中，sendRedirect能够处理相对路径，自动转换为绝对路径。但是你只能在2.1版本中使用绝对路径。   
  
如果你将用户转向到站点的另一页中，你要用 HttpServletResponse 中的 encodeURL 方法传送URL。这么做可预防不断使用基于URL重写的会话跟踪的情况。URL重写是一种在你的网站跟踪不使用 cookies 的用户的方法。这是通过在每一个URL尾部附加路径信息实现的，但是 servlet 会话跟踪API会自动的注意这些细节。会话跟踪在第九章讨论，并且养成使用 encodeURL 的习惯会使以后添加会话跟踪的功能更容易很多。   
  
核心技巧  
如果你将用户转向到你的站点的其他页面，用 response.sendRedirect(response.encodeURL(url)) 的方式事先计划好会话跟踪(session tracking)要比只是调用 response.sendRedirect(url) 好的多。   
  
这个状态码有时可以与301交换使用。例如，如果你错误的访问了[http://www.talentdigger.cn/home/link.php?url=aG9zdC9%2BdXNlcg%3D%3D](http://www.talentdigger.cn/home/link.php?url=aG9zdC9+dXNlcg==" \t "http://www.cnblogs.com/lxinxuan/archive/2009/10/22/_blank)（路径信息不完整），有些服务器就会回复301状态码而有些则回复302。从技术上说，如果最初的请求是GET浏览器只是被假定自动转向。如果想了解更多细节，请看状态码307的讨论。   
  
303 (See Other/参见其他信息)  
这个状态码和 301、302 相似，只是如果最初的请求是 POST，那么新文档（在定位头信息中给出）药用 GET 找回。这个状态码是新加入 HTTP 1.1中的。   
  
304 (Not Modified/为修正)  
当客户端有一个缓存的文档，通过提供一个 If-Modified-Since 头信息可指出客户端只希望文档在指定日期之后有所修改时才会重载此文档，用这种方式可以进行有条件的请求。304 (SC\_NOT\_MODIFIED)是指缓冲的版本已经被更新并且客户端应刷新文档。另外，服务器将返回请求的文档及状态码 200。servlet一般情况下不会直接设置这个状态码。它们会实现getLastModified方法并根据修正日期让默认服务方法处理有条件的请求。这个方法的例程已在2.8部分(An Example Using Servlet Initialization and Page Modification Dates/一个使用servlet初始化和页面修正日期的例子)给出。   
  
305 (Use Proxy/使用代理)  
305 (SC\_USE\_PROXY)表示所请求的文档要通过定位头信息中的代理服务器获得。这个状态码是新加入 HTTP 1.1中的。   
  
307 (Temporary Redirect/临时重定向)  
浏览器处理307状态的规则与302相同。307状态被加入到 HTTP 1.1中是由于许多浏览器在收到302响应时即使是原始消息为POST的情况下仍然执行了错误的转向。只有在收到303响应时才假定浏览器会在POST请求时重定向。添加这个新的状态码的目的很明确：在响应为303时按照GET和POST请求转向；而在307响应时则按照GET请求转向而不是POST请求。注意：由于某些原因在HttpServletResponse中还没有与这个状态对应的常量。该状态码是新加入HTTP 1.1中的。   
  
注意  
在 HttpServletResponse 中没有 SC\_TEMPORARY\_REDIRECT 常量，所以你只能显示的使用307状态码。   
  
400 (Bad Request/错误请求)  
400 (SC\_BAD\_REQUEST)指出客户端请求中的语法错误。   
  
401 (Unauthorized/未授权)  
401 (SC\_UNAUTHORIZED)表示客户端在授权头信息中没有有效的身份信息时访问受到密码保护的页面。这个响应必须包含一个WWW-Authenticate的授权信息头。例如，在本书4.5部分中的“Restricting Access to Web Pages./限制访问Web页。”   
  
403 (Forbidden/禁止)  
403 (SC\_FORBIDDEN)的意思是除非拥有授权否则服务器拒绝提供所请求的资源。这个状态经常会由于服务器上的损坏文件或目录许可而引起。   
  
404 (Not Found/未找到)  
404 (SC\_NOT\_FOUND)状态每个网络程序员可能都遇到过，他告诉客户端所给的地址无法找到任何资源。它是表示“没有所访问页面”的标准方式。这个状态码是常用的响应并且在HttpServletResponse类中有专门的方法实现它：sendError("message")。相对于setStatus使用sendError得好处是：服务器会自动生成一个错误页来显示错误信息。但是，Internet Explorer 5浏览器却默认忽略你发挥的错误页面并显示其自定义的错误提示页面，虽然微软这么做违反了 HTTP 规范。要关闭此功能，在工具菜单里，选择Internet选项，进入高级标签页，并确认“显示友好的 HTTP 错误信息”选项（在我的浏览器中是倒数第8各选项）没有被选。但是很少有用户知道此选项，因此这个特性被IE5隐藏了起来使用户无法看到你所返回给用户的信息。而其他主流浏览器及IE4都完全的显示服务器生成的错误提示页面。可以参考图6-3及6-4中的例子。   
  
核心警告  
默认情况下，IE5忽略服务端生成的错误提示页面。   
  
405 (Method Not Allowed/方法未允许)  
405 (SC\_METHOD\_NOT\_ALLOWED)指出请求方法(GET, POST, HEAD, PUT, DELETE, 等)对某些特定的资源不允许使用。该状态码是新加入 HTTP 1.1中的。   
  
406 (Not Acceptable/无法访问)  
406 (SC\_NOT\_ACCEPTABLE)表示请求资源的MIME类型与客户端中Accept头信息中指定的类型不一致。见本书7.2部分中的表7.1(HTTP 1.1 Response Headers and Their Meaning/HTTP 1.1响应头信息以及他们的意义)中对MIME类型的介绍。406是新加入 HTTP 1.1中的。   
  
407 (Proxy Authentication Required/代理服务器认证要求)  
407 (SC\_PROXY\_AUTHENTICATION\_REQUIRED)与401状态有些相似，只是这个状态用于代理服务器。该状态指出客户端必须通过代理服务器的认证。代理服务器返回一个Proxy-Authenticate响应头信息给客户端，这会引起客户端使用带有Proxy-Authorization请求的头信息重新连接。该状态码是新加入 HTTP 1.1中的。   
  
408 (Request Timeout/请求超时)  
408 (SC\_REQUEST\_TIMEOUT)是指服务端等待客户端发送请求的时间过长。该状态码是新加入 HTTP 1.1中的。   
  
409 (Conflict/冲突)  
该状态通常与PUT请求一同使用，409 (SC\_CONFLICT)状态常被用于试图上传版本不正确的文件时。该状态码是新加入 HTTP 1.1中的。   
  
410 (Gone/已经不存在)  
410 (SC\_GONE)告诉客户端所请求的文档已经不存在并且没有更新的地址。410状态不同于404，410是在指导文档已被移走的情况下使用，而404则用于未知原因的无法访问。该状态码是新加入 HTTP 1.1中的。   
  
411 (Length Required/需要数据长度)  
411 (SC\_LENGTH\_REQUIRED)表示服务器不能处理请求（假设为带有附件的POST请求），除非客户端发送Content-Length头信息指出发送给服务器的数据的大小。该状态是新加入 HTTP 1.1的。   
  
412 (Precondition Failed/先决条件错误)  
412 (SC\_PRECONDITION\_FAILED)状态指出请求头信息中的某些先决条件是错误的。该状态是新加入 HTTP 1.1的。   
  
413 (Request Entity Too Large/请求实体过大)  
413 (SC\_REQUEST\_ENTITY\_TOO\_LARGE)告诉客户端现在所请求的文档比服务器现在想要处理的要大。如果服务器认为能够过一段时间处理，则会包含一个Retry-After的响应头信息。该状态是新加入 HTTP 1.1的。   
  
414 (Request URI Too Long/请求URI过长)  
414 (SC\_REQUEST\_URI\_TOO\_LONG)状态用于在URI过长的情况时。这里所指的“URI”是指URL中主机、域名及端口号之后的内容。例如：在URL--http://www.y2k-disaster.com:8080/we/look/silly/now/中URI是指/we/look/silly/now/。该状态是新加入 HTTP 1.1的。   
  
415 (Unsupported Media Type/不支持的媒体格式)  
415 (SC\_UNSUPPORTED\_MEDIA\_TYPE)意味着请求所带的附件的格式类型服务器不知道如何处理。该状态是新加入 HTTP 1.1的。   
  
416 (Requested Range Not Satisfiable/请求范围无法满足)  
416表示客户端包含了一个服务器无法满足的Range头信息的请求。该状态是新加入 HTTP 1.1的。奇怪的是，在servlet 2.1版本API的HttpServletResponse中并没有相应的常量代表该状态。   
  
注意  
在servlet 2.1的规范中，类HttpServletResponse并没有SC\_REQUESTED\_RANGE\_NOT\_SATISFIABLE 这样的常量，所以你只能直接使用416。在servlet 2.2版本之后都包含了此常量。   
  
417 (Expectation Failed/期望失败)  
如果服务器得到一个带有100-continue值的Expect请求头信息，这是指客户端正在询问是否可以在后面的请求中发送附件。在这种情况下，服务器也会用该状态(417)告诉浏览器服务器不接收该附件或用100 (SC\_CONTINUE)状态告诉客户端可以继续发送附件。该状态是新加入 HTTP 1.1的。   
  
500 (Internal Server Error/内部服务器错误)  
500 (SC\_INTERNAL\_SERVER\_ERROR) 是常用的“服务器错误”状态。该状态经常由CGI程序引起也可能（但愿不会如此！）由无法正常运行的或返回头信息格式不正确的servlet引起。   
  
501 (Not Implemented/未实现)  
501 (SC\_NOT\_IMPLEMENTED)状态告诉客户端服务器不支持请求中要求的功能。例如，客户端执行了如PUT这样的服务器并不支持的命令。   
  
502 (Bad Gateway/错误的网关)  
502 (SC\_BAD\_GATEWAY)被用于充当代理或网关的服务器；该状态指出接收服务器接收到远端服务器的错误响应。   
  
503 (Service Unavailable/服务无法获得)  
状态码503 (SC\_SERVICE\_UNAVAILABLE)表示服务器由于在维护或已经超载而无法响应。例如，如果某些线程或数据库连接池已经没有空闲则servlet会返回这个头信息。服务器可提供一个Retry-After头信息告诉客户端什么时候可以在试一次。   
  
504 (Gateway Timeout/网关超时)  
该状态也用于充当代理或网关的服务器；它指出接收服务器没有从远端服务器得到及时的响应。该状态是新加入 HTTP 1.1的。   
  
505 (HTTP Version Not Supported/不支持的 HTTP 版本)  
505 (SC\_HTTP\_VERSION\_NOT\_SUPPORTED)状态码是说服务器并不支持在请求中所标明 HTTP 版本。该状态是新加入 HTTP 1.1的。