6.2 HTTP 1.1 Status Codes

In this section we describe the most important status codes available for use in servlets talking to HTTP 1.1 clients, along with the standard message associated with each code. A good understanding of these codes can dramatically increase the capabilities of your servlets, so you should at least skim the descriptions to see what options are at your disposal. You can come back for details when you are ready to use the capabilities.

The complete HTTP 1.1 specification is given in RFC 2616. In general, you can access RFCs online by going to http://www.rfc-editor.org/ and following the links to the latest RFC archive sites, but since this one came from the World Wide Web Consortium, you can just go to http://www.w3.org/Protocols/. Codes that are new in HTTP 1.1 are noted since some browsers support only HTTP 1.0. You should only send the new codes to clients that support HTTP 1.1, as verified by checking `request.getRequestProtocol`.

The rest of this section describes the specific status codes available in HTTP 1.1. These codes fall into five general categories:

- **100–199**

  Codes in the 100s are informational, indicating that the client should respond with some other action.

- **200–299**

  Values in the 200s signify that the request was successful.

- **300–399**

  Values in the 300s are used for files that have moved and usually include a `Location` header indicating the new address.

- **400–499**

  Values in the 400s indicate an error by the client.

- **500–599**

  Codes in the 500s signify an error by the server.

The constants in `HttpServletResponse` that represent the various codes are derived from the standard messages associated with the codes. In servlets, you usually refer to status codes only by means of these constants. For example, you would use `response.setStatus(response.SC_NO_CONTENT)` rather than `response.setStatus(204)`, since the latter is unclear to readers and is prone to typographical errors. However, you should note that servers are allowed to vary the messages slightly, and clients pay attention only to the numeric value. So, for example, you might see a server return a status line of `HTTP/1.1 200 Document Follows` instead of `HTTP/1.1 200 OK`.

100 (Continue)

If the server receives an `Expect` request header with a value of `100-continue`, it means that the client is asking if it can send an attached document in a follow-up

request. In such a case, the server should either respond with status 100 (`SC_CONTINUE`) to tell the client to go ahead or use 417 (`SC_EXPECTATION_FAILED`) to tell the browser it won't accept the document. This status code is new in HTTP 1.1.

200 (OK)

A value of 200 (`SC_OK`) means that everything is fine; the document follows for `GET` and `POST` requests. This status is the default for servlets; if you don't use `setStatus`, you'll get 200.

202 (Accepted)

A value of 202 (`SC_ACCEPTED`) tells the client that the request is being acted upon but processing is not yet complete.

204 (No Content)

A status code of 204 (`SC_NO_CONTENT`) stipulates that the browser should continue to display the previous document because no new document is available. This behavior is useful if the user periodically reloads a page by pressing the Reload button and you can determine that the previous page is already up-to-date.

205 (Reset Content)

A value of 205 (`SC_RESET_CONTENT`) means that there is no new document but the browser should reset the document view. Thus, this status code is used to instruct browsers to clear form fields. It is new in HTTP 1.1.

301 (Moved Permanently)

The 301 (`SC_MOVED_PERMANENTLY`) status indicates that the requested document is elsewhere; the new URL for the document is given in the `Location` response header. Browsers should automatically follow the link to the new URL.

302 (Found)

This value is similar to 301, except that in principle the URL given by the `Location` header should be interpreted as a temporary replacement, not a permanent one. In practice, most browsers treat 301 and 302 identically. Note: in HTTP 1.0, the message was `Moved Temporarily` instead of `Found`, and the constant in `HttpServletResponse` is `SC_MOVED_TEMPORARILY`, not the expected `SC_FOUND`.

Core Note

The constant representing 302 is `SC_MOVED_TEMPORARILY`, not `SC_FOUND`.

Status code 302 is useful because browsers automatically follow the reference to the new URL given in the `Location` response header. Note that the browser reconnects to the new URL immediately; no intermediate output is displayed. This behavior distinguishes *redirects* from *refreshes* where an intermediate page is temporarily displayed (see the next chapter for details on the `Refresh` header). With redirects, another site, not the servlet itself, generates the

results. So why use a servlet at all? Redirects are useful for the following tasks:

- **Computing destinations.** If you know the final destination for the user in advance, your hypertext link or HTML form could send the user directly there. But, if you need to look at the data before deciding where to obtain the necessary results, a redirection is useful. For example, you might want to send users to a standard site that gives information on stocks, but you need to look at the stock symbol before deciding whether to send them to the New York Stock Exchange, NASDAQ, or a non-U.S. site.

- **Tracking user behavior.** If you send users a page that contains a hypertext link to another site, you have no way to know if they actually click on the link. But perhaps this information is important in analyzing the usefulness of the different links you send them. So, instead of sending users the direct link, you can send them a link to your own site, where you can then record some information and then redirect them to the real site. For example, several search engines use this trick to determine which of the results they display are most popular.

- **Performing side effects.** What if you want to send users to a certain site but set a cookie on the user's browser first? No problem: return both a `Set-Cookie` response header (by means of `response.addCookie`—see Chapter 8) *and* a 302 status code (by means of `response.sendRedirect`).

The 302 status code is so useful, in fact, that there is a special method for it, `sendRedirect`. Using `response.sendRedirect(url)` has a couple of advantages over using `response.setStatus(response.SC_MOVED_TEMPORARILY)` and `response.setHeader ("Location", url)`. First, it is shorter and easier. Second, with `sendRedirect`, the servlet automatically builds a page containing the link to show to older browsers that don't automatically follow redirects. Finally, `sendRedirect` can handle relative URLs, automatically translating them into absolute ones.

Technically, browsers are supposed to automatically follow the redirection only if the original request was `GET`. For details, see the discussion of the 307 status code.

303 (See Other)

The 303 (`SC_SEE_OTHER`) status is similar to 301 and 302, except that if the original request was `POST`, the new document (given in the `Location` header) should be retrieved with `GET`. See status code 307. This code is new in HTTP 1.1.

304 (Not Modified)

When a client has a cached document, it can perform a conditional request by supplying an `If-Modified-Since` header to signify that it wants the document only if it has been changed since the specified date. A value of 304 (`SC_NOT_MODIFIED`) means that the cached version is up-to-date and the client should use it. Otherwise, the server should return the requested document with the normal (200) status code. Servlets normally should not set this status code directly. Instead, they should implement the `getLastModified` method and let the default `service` method handle conditional requests based upon this modification date. For an example, see the `LotteryNumbers` servlet in Section 3.6 (The Servlet Life Cycle).

307 (Temporary Redirect)

The rules for how a browser should handle a 307 status are identical to those for 302. The 307 value was added to HTTP 1.1 since many browsers erroneously follow the redirection on a 302 response even if the original message is a `POST`. Browsers

are supposed to follow the redirection of a POST request only when they receive a 303 resp status. This new status is intended to be unambiguously clear: follow redirected GET *and* P requests in the case of 303 responses; follow redirected GET *but not* POST requests in the c of 307 responses. This status code is new in HTTP 1.1.

400 (Bad Request)

A 400 (SC_BAD_REQUEST) status indicates bad syntax in the client request.

401 (Unauthorized)

A value of 401 (SC_UNAUTHORIZED) signifies that the client tried to access a password-prot page but that the request did not have proper identifying information in the Authorizatio header. The response must include a WWW-Authenticate header. For details, see the chapt programmatic Web application security in Volume 2 of this book.

403 (Forbidden)

A status code of 403 (SC_FORBIDDEN) means that the server refuses to supply the resource regardless of authorization. This status is often the result of bad file or directory permissio the server.

404 (Not Found)

The infamous 404 (SC_NOT_FOUND) status tells the client that no resource could be found a address. This value is the standard "no such page" response. It is such a common and use response that there is a special method for it in the HttpServletResponse class: sendErr ("message"). The advantage of sendError over setStatus is that with sendError, the se automatically generates an error page showing the error message. 404 errors need not me say "Sorry, the page cannot be found." Instead, they can give information on why the pag couldn't be found or supply search boxes or alternative places to look. The sites at www.microsoft.com and www.ibm.com have particularly good examples of useful error pag (to see them, just make up a nonexistent URL at either site). In fact, there is an entire site dedicated to the good, the bad, the ugly, and the bizarre in 404 error messages: http://www.plinko.net/404/. We find http://www.plinko.net/404/links.asp?type=cat&key= (amusing 404 error messages) particularly funny.

Unfortunately, however, the default behavior of Internet Explorer in version 5 and later is ignore the error page you send back and to display its own static (and relatively useless) e message, even though doing so explicitly contradicts the HTTP specification. To turn off thi setting, go to the Tools menu, select Internet Options, choose the Advanced tab, and mak sure the "Show friendly HTTP error messages" box is *not* checked. Regrettably, few users a aware of this setting, so this "feature" prevents most users of Internet Explorer from seein informative messages you return. Other major browsers and version 4 of Internet Explorer properly display server-generated error pages.

Core Warning

By default, Internet Explorer versions 5 and later improperly ignore server-generated error pages.

Fortunately, it is relatively uncommon for individual servlets to build their own 404 error p A more common approach is to set up error pages for an entire Web application; see Secti 2.11 (Web Applications: A Preview) for details.

405 (Method Not Allowed)

A 405 (`SC_METHOD_NOT_ALLOWED`) value signifies that the request method (`GET`, `POST`, `HEAD`, `PUT`, `DELETE`, etc.) was not allowed for this particular resource. This status code is new in HTTP 1.1.

415 (Unsupported Media Type)

A value of 415 (`SC_UNSUPPORTED_MEDIA_TYPE`) means that the request had an attached document of a type the server doesn't know how to handle. This status code is new in HTTP 1.1.

417 (Expectation Failed)

If the server receives an `Expect` request header with a value of `100-continue`, it means that the client is asking if it can send an attached document in a follow-up request. In such a case, the server should either respond with this status (417) to tell the browser it won't accept the document or use 100 (`SC_CONTINUE`) to tell the client to go ahead. This status code is new in HTTP 1.1.

500 (Internal Server Error)

500 (`SC_INTERNAL_SERVER_ERROR`) is the generic "server is confused" status code. It often results from CGI programs or (heaven forbid!) servlets that crash or return improperly formatted headers.

501 (Not Implemented)

The 501 (`SC_NOT_IMPLEMENTED`) status notifies the client that the server doesn't support the functionality to fulfill the request. It is used, for example, when the client issues a command like `PUT` that the server doesn't support.

503 (Service Unavailable)

A status code of 503 (`SC_SERVICE_UNAVAILABLE`) signifies that the server cannot respond because of maintenance or overloading. For example, a servlet might return this header if some thread or database connection pool is currently full. The server can supply a `Retry-After` header to tell the client when to try again.

505 (HTTP Version Not Supported)

The 505 (`SC_HTTP_VERSION_NOT_SUPPORTED`) code means that the server doesn't support the version of HTTP named in the request line. This status code is new in HTTP 1.1.