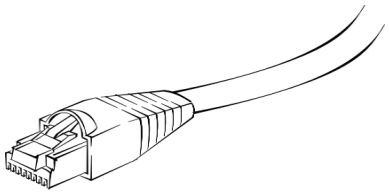
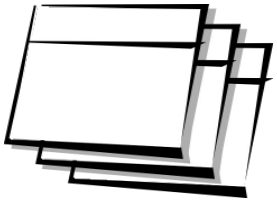
Códigos de Estado HTTP



Vladimir Zúñiga <http://vladimirzuniga.tk>

Al realizar una solicitud a un servidor de páginas web (como Apache o similares), el servidor muestra un código en respuesta. Este código, que es numerico y esta compuesto por 3 digitos, entrega información acerca del estado de la solicitud y dependiendo del numero con el que comience es el tipo o nivel de respuesta que nos esta entregando.



1xx: Respuestas informativas
2xx: Peticiones correctas
3xx: Redirecciones
4xx Errores del cliente
5xx Errores de servidor

Esta es la lista de los códigos de estado con su respectivo significado

Código de Estado	Mensaje Asociado	**Significado**
100	*Continue*	Continúa con petición parcial (nuevo en HTTP 1.1)
101	*Switching Protocols*	El servidor cumplirá con la cabecera Upgrade y cambiará a un protocolo diferente. (Nuevo en HTTP 1.1)
200	*OK*	Todo está bien; los documentos seguidos por peticiones GET y POST. Esto es por defecto para los Servlets, si no usamos setStatus, obtendremos esto.
201	*Created*	El servidor creo un documento; la cabecera Location indica la URL.
202	*Accepted*	La petición se está realizando, el proceso no se ha completado.
203	*Non-Authoritative Information*	El documento está siendo devuelto normalmente, pero algunas cabeceras de respuesta podrían ser incorrectas porque se está usando una copia del documento (Nuevo en HTTP 1.1)
204	*No Content*	No hay un documento nuevo; el navegador contínua mostrando el documento anterior. Esto es útil si el usuario recarga periódicamente una página y podemos determinar que la página anterior ya está actualizada. Sin embargo, esto no funciona para páginas que se recargan automáticamente mediante cabeceras de respuesta Refresh o su equivalente <META HTTP-EQUIV="Refresh" ...>,ya que al devolver este código de estado se pararán futuras recargas.
205	*Reset Content*	No hay documento nuevo, pero el navegador debería resetear el documento. Usado para forzar al navegador a borrar los contenidos de los campos de un formulario CGI (Nuevo en HTTP 1.1)
206	*Partial Content*	El cliente envía una petición parcial con una cabecera Range, y el servidor la ha completado. (Nuevo en HTTP 1.1)
300	*Multiple Choices*	El documento pedido se puede encontrar en varios sitios; serán listados en el documento devuelto. Si el servidor tiene una opción preferida, debería listarse en la cabecera de respuesta Location .
301	*Moved Permanently*	El documento pedido está en algún lugar, y la URL se da en la cabecera de respuesta Location. Los navegadores deberían seguir automáticamente el enlace a la nueva URL.

302	*Found*	Similar a 301, excepto que la nueva URL debería ser interpretada como reemplazada temporalmente, no permanentemente. Observa: el mensaje era "Moved Temporarily" en HTTP 1.0, y la constante en HttpServletResponse es SC_MOVED_TEMPORARILY, no SC_FOUND. Cabecera muy útil, ya que los navegadores siguen automáticamente el enlace a la nueva URL. Este código de estado es tan útil que hay un método especial para ella, sendRedirect. Usar response.sendRedirect(url) tiene unpar de ventajas sobre hacer response.setStatus(response.SC_MOVED_TEMPORARILY) y response.setHeader("Location", url). Primero, es más fácil. Segundo, con sendRedirect, el servlet automáticamente construye una página que contiene el enlace (para mostrar a los viejos navegadores que no siguen las redirecciones automáticamente). Finalmente, sendRedirect puede manejar URLs relativas, automáticamentes las traducen a absolutas. Observa que este código de estado es usado algunas veces de forma intercambiada con 301. Por ejemplo, si erróneamente pedimos http://host/~user (olvidando la última barra), algunos servidores enviarán 301 y otros 302. Técnicamente, se supone que los navegadores siguen automáticamente la redirección su la petición original era GET. Puedes ver la cabecera 307 para más detalles.
303	*See Other*	Igual que 301/302, excepto que si la petición original era POST, el documento redirigido (dado en la cabecera Location) debería ser recuperado mediante GET. (Nuevo en HTTP 1.1)
304	*Not Modified*	El cliente tiene un documento en el caché y realiza una petición condicional (normalmente suministrando una cabecera If-Modified-Since indicando que sólo quiere documentos más nuevos que la fecha especificada). El servidor quiere decirle al cliente que el viejo documento del caché todavía está en uso.
305	*Use Proxy*	El documento pedido debería recuperarse mediante el proxy listado en la cabecera Location. (Nuevo en HTTP 1.1)
307	*Temporary Redirect*	Es idéntica a 302 ("Found" o "Temporarily Moved"). Fue añádido a HTTP 1.1 ya que muchos navegadores siguen erróneamente la redirección de una respuesta 302 incluso si el mensaje original fue un POST, y sólo se debe seguir la redirección de una petición POST en respuestas 303. Esta respuesta es algo ambígua: sigue el redireccionamiento para peticiones GET y POST en el caso de respuestas 303, y en el caso de respuesta 307 sólo sigue la redirección de peticiones GET. Nota: por alguna razón no existe una constante en HttpServletResponse que corresponda con este código de estado. (Nuevo en HTTP 1.1)
400	*Bad Request*	Mala Síntaxis de la petición.
401	*Unauthorized*	El cliente intenta acceder a una página protegida por password sin las autorización apropiada. La respuesta debería incluir una cabecera WWW-Authenticate que el navegador debería usar para mostrar la caja de diálogo usuario/password, que viene de vuelta con la cabecera Authorization. Compruebe que las urls en su módulo de configuración aparecen con http:
403	*Forbidden*	El recurso no está disponible, si importar la autorización. Normalmente indica la falta permisos de fichero o directorios en el servidor.
404	*Not Found*	No se pudo encontrar el recurso en esa dirección. Esta la respuesta estándard "no such page". Es tan cómún y útil esta respuesta que hay un método especial para ella en HttpServletResponse: sendError(message). La ventaja de sendError sobre setStatus es que, con sendErr, el servidor genera automáticamente una página que muestra un mensaje de error.
405	*Method Not Allowed*	El método de la petición (GET, POST, HEAD, DELETE, PUT, TRACE, etc.) no estaba permitido para este recurso particular. (Nuevo en HTTP 1.1)
406	*Not Acceptable*	El recurso indicado genera un tipo MIME incompatible con el especificado por el cliente mediante su cabecera Accept. (Nuevo en HTTP 1.1)
407	*Proxy Authentication Required*	Similar a 401, pero el servidor proxy debería devolver una cabecera Proxy-Authenticate. (Nuevo en HTTP 1.1)

408	*Request Timeout*	El cliente tarda demasiado en envíar la petición. (Nuevo en HTTP 1.1)
409	*Conflict*	Usualmente asociado con peticiones PUT; usado para situaciones como la carga de una versión incorrecta de un fichero. (Nuevo en HTTP 1.1)
410	*Gone*	El documento se ha ido; no se conoce la dirección de reenvio. Difiere de la 404 en que se sabe que el documento se ha ido permanentemente, no sólo está indisponible por alguna razón desconocida como con 404. (Nuevo en HTTP 1.1)
411	*Length Required*	El servidor no puede procesar la petición a menos que el cliente envíe una cabecera Content-Length. (Nuevo en HTTP 1.1)
412	*Precondition Failed*	Alguna condición prévia especificada en la petición era falsa (Nuevo en HTTP 1.1)
413	*Request Entity Too Large*	El documento pedido es mayor que lo que el servidor quiere manejar ahora. Si el servidor cree que puede manejarlo más tarde, debería incluir una cabecera Retry-After. (Nuevo en HTTP 1.1)
414	*Request URI Too Long*	La URI es demsiado larga. (Nuevo en HTTP 1.1)
415	*Unsupported Media Type*	La petición está en un formato desconocido. (Nuevo en HTTP 1.1)
416	*Requested Range Not Satisfiable*	El cliente incluyó una cabecera Range no satisfactoria en la petición. (Nuevo en HTTP 1.1)
417	*Expectation Failed*	No se puede conseguir el valor de la cabecera Expect. (Nuevo en HTTP 1.1)
500	*Internal Server Error*	Mensaje genérico "server is confused". Normalmente es el resultado de programas CGI o servlets que se quedan colgados o retornan cabeceras mal formateadas.
501	*Not Implemented*	El servidor no soporta la funcionalidad de rellenar peticiones. Usado, por ejemplo, cuando el cliente envía comandos como PUT que el cliente no soporta.
502	*Bad Gateway*	Usado por servidores que actúan como proxies o gateways; indica que el servidor inicial obtuvo una mala respuesta desde el servidor remoto.
503	*Service Unavailable*	El servidor no puede responder debido a mentenimiento o sobrecarga. Por ejemplo, un servlet podría devolver esta cabecera si algún almacen de threads o de conexiones con bases de datos están llenos. El servidor puede suministrar una cabecera Retry-After.
504	*Gateway Timeout*	Usado por servidores que actúan como proxies o gateways; indica que el servidor inicial no obtuvo una respuesta a tiempo del servidor remoto. (Nuevo en HTTP 1.1)
505	*HTTP Version Not Supported*	El servidor no soporta la versión de HTTP indicada en la línea de petición. (Nuevo en HTTP 1.1)