



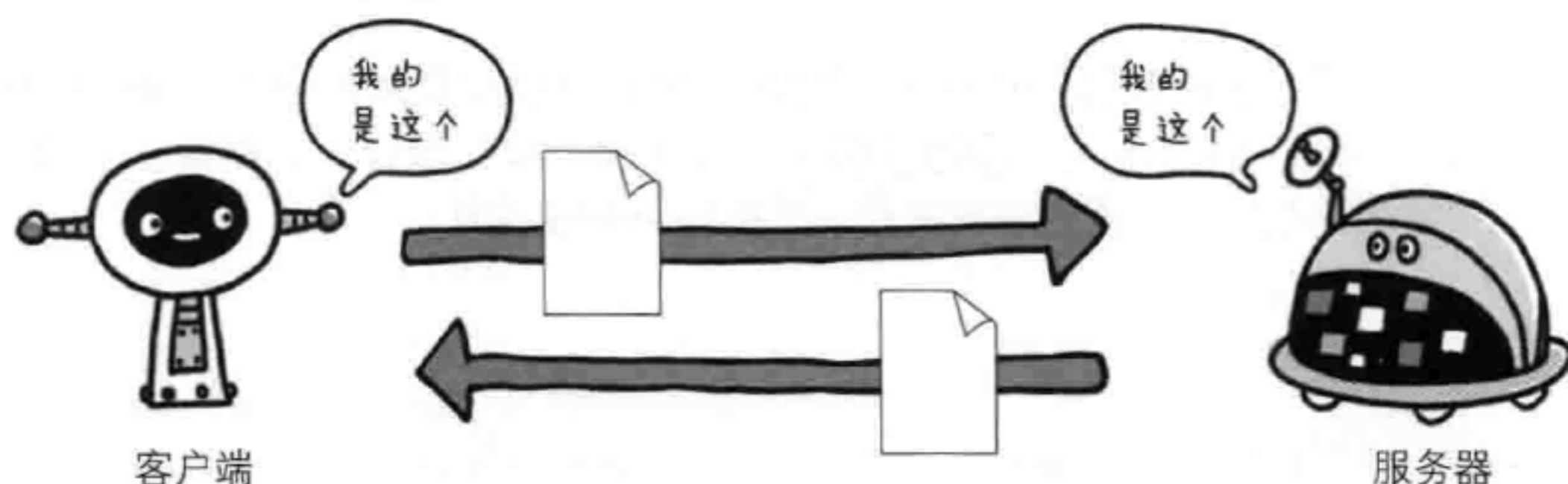
首部字段 WWW-Authenticate 用于 HTTP 访问认证。它会告知客户端适用于访问请求 URI 所指定资源的认证方案（Basic 或是 Digest）和带参数提示的质询（challenge）。状态码 401 Unauthorized 响应中，肯定带有首部字段 WWW-Authenticate。

上述示例中，realm 字段的字符串是为了辨别请求 URI 指定资源所受到的保护策略。有关该首部，请参阅本章之后的内容。

6.6 实体首部字段

实体首部字段是包含在请求报文和响应报文中的实体部分所使用的首部，用于补充内容的更新时间等与实体相关的信息。

126



图：在请求和响应两方的 HTTP 报文中都含有与实体相关的首部字段

6.6.1 Allow



Allow: GET, HEAD

首部字段 Allow 用于通知客户端能够支持 Request-URI 指定资源的所有 HTTP 方法。当服务器接收到不支持的 HTTP 方法时，会以状态码 405 Method Not Allowed 作为响应返回。与此同时，还会把所有能支持的 HTTP 方法写入首部字段 Allow 后返回。

6.6.2 Content-Encoding

```
Content-Encoding: gzip
```

首部字段 Content-Encoding 会告知客户端服务器对实体的主体部分选用的内容编码方式。内容编码是指在不丢失实体信息的前提下所进行的压缩。

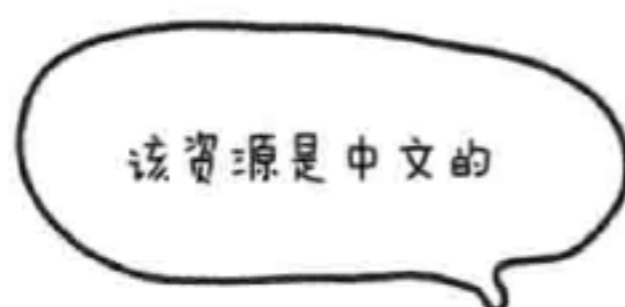


主要采用以下 4 种内容编码的方式。（各方式的说明请参考 6.4.3 节 Accept-Encoding 首部字段）。

- gzip
- compress
- deflate
- identity



6.6.3 Content-Language



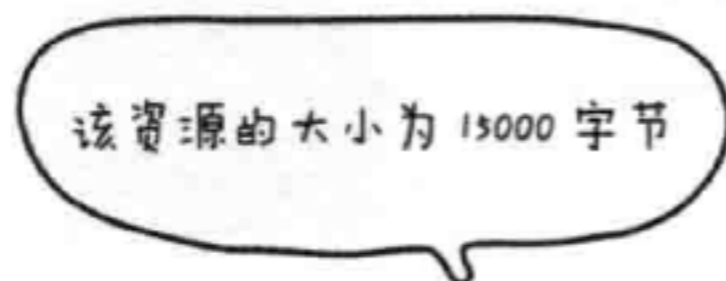
服务器

Content-Language: zh-CN

首部字段 Content-Language 会告知客户端，实体主体使用的自然语言（指中文或英文等语言）。

128

6.6.4 Content-Length



服务器

Content-Length: 15000

首部字段 Content-Length 表明了实体主体部分的大小（单位是字节）。对实体主体进行内容编码传输时，不能再使用 Content-Length 首部字段。由于实体主体大小的计算方法略微复杂，所以在此不再展开。读者若想一探究竟，可参考 RFC2616 的 4.4。

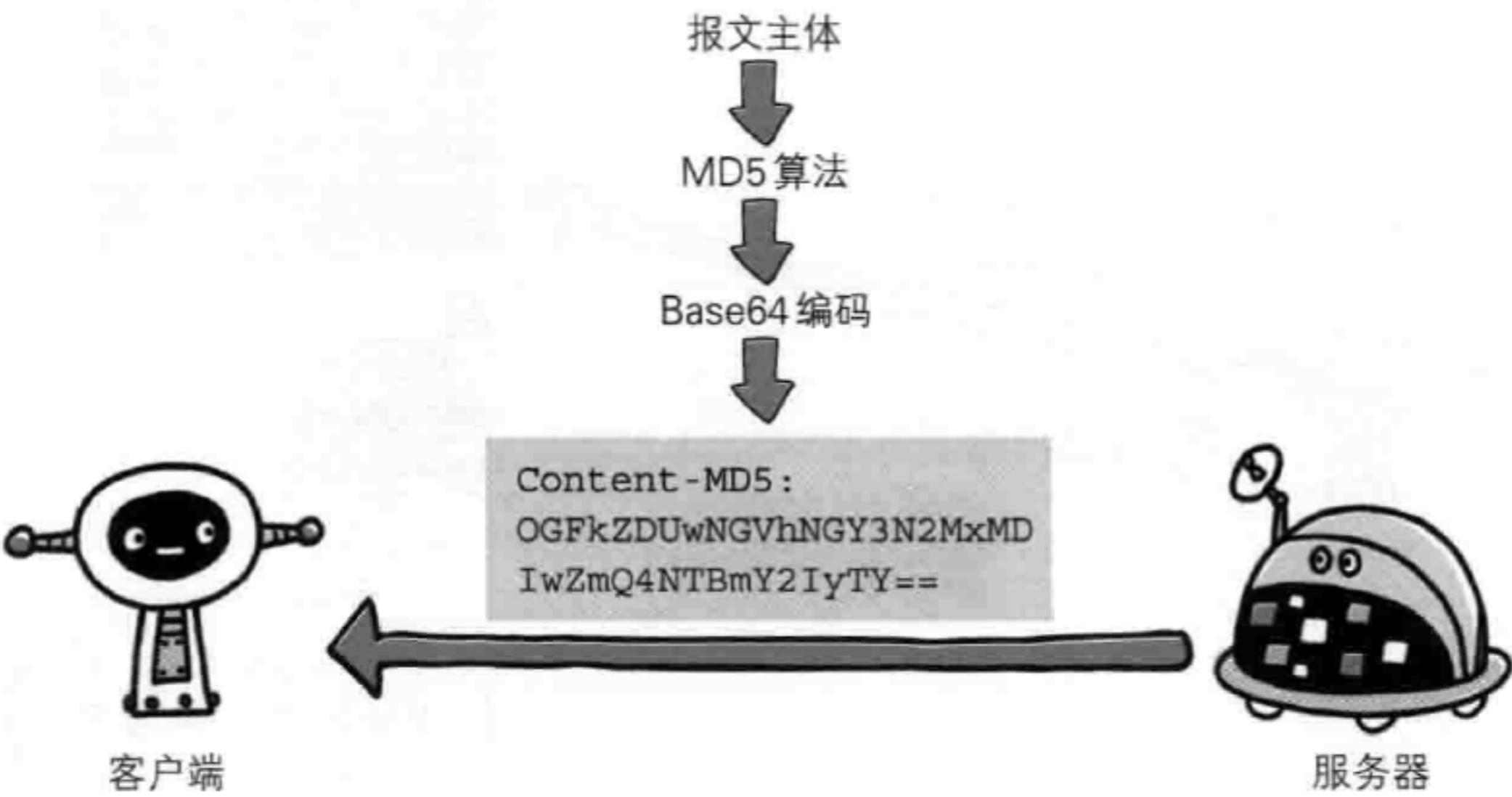
6.6.5 Content-Location

```
Content-Location: http://www.hackr.jp/index-ja.html
```

首部字段 Content-Location 给出与报文主体部分相对应的 URI。和首部字段 Location 不同，Content-Location 表示的是报文主体返回资源对应的 URI。

比如，对于使用首部字段 Accept-Language 的服务器驱动型请求，当返回的页面内容与实际请求的对象不同时，首部字段 Content-Location 内会写明 URI。（访问 `http://www.hackr.jp/` 返回的对象却是 `http://www.hackr.jp/index-ja.html` 等类似情况）

6.6.6 Content-MD5



图：客户端会对接收的报文主体执行相同的 MD5 算法，然后与首部字段 Content-MD5 的字段值比较

```
Content-MD5: OGFkZDUwNGVhNGY3N2MxMDIwZmQ4NTBmY2IyTY==
```



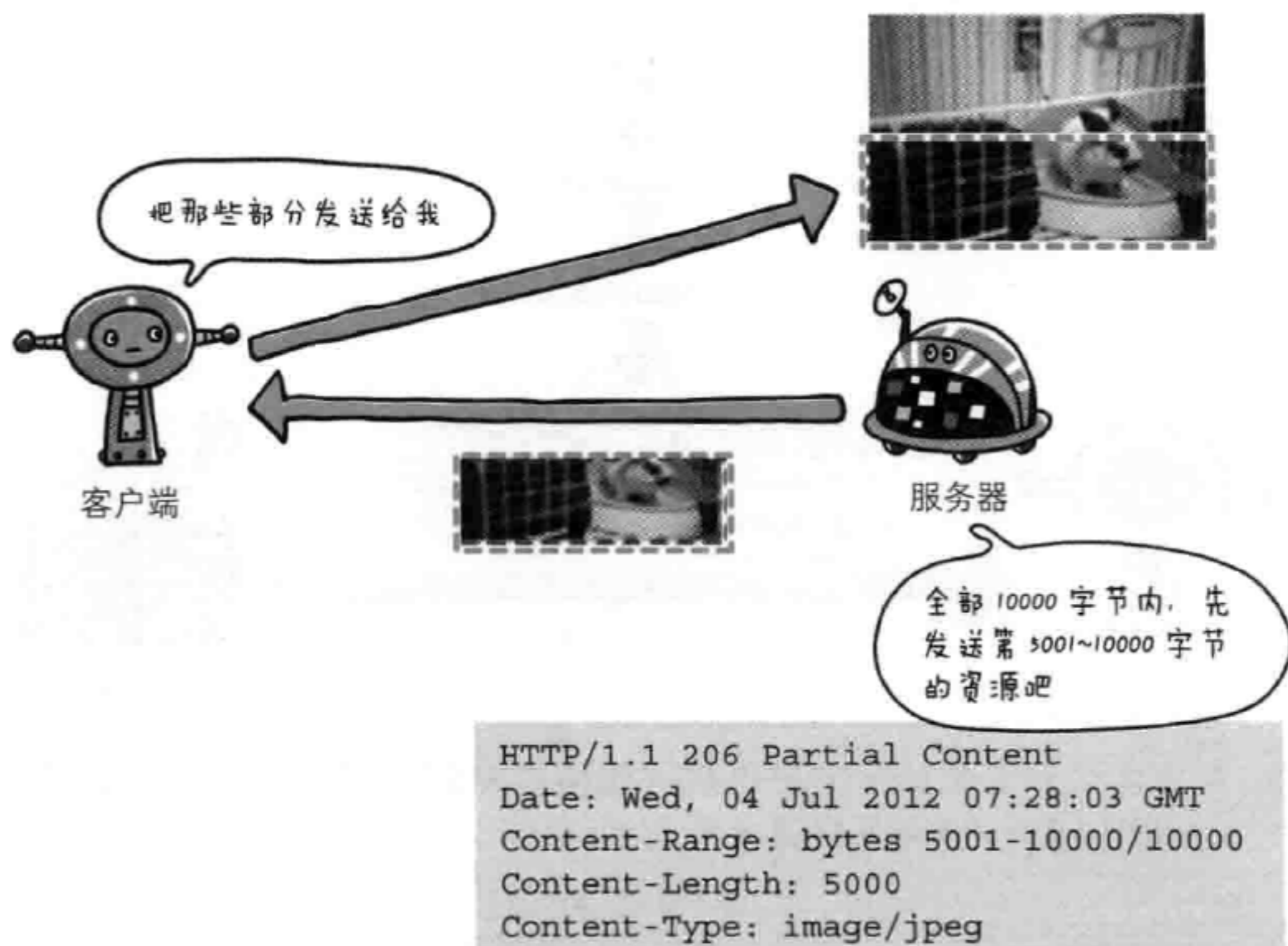
首部字段 Content-MD5 是一串由 MD5 算法生成的值，其目的在于检查报文主体在传输过程中是否保持完整，以及确认传输到达。

对报文主体执行 MD5 算法获得的 128 位二进制数，再通过 Base64 编码后将结果写入 Content-MD5 字段值。由于 HTTP 首部无法记录二进制值，所以要通过 Base64 编码处理。为确保报文的有效性，作为接收方的客户端会对报文主体再执行一次相同的 MD5 算法。计算出的值与字段值作比较后，即可判断出报文主体的准确性。

采用这种方法，对内容上的偶发性改变是无从查证的，也无法检测出恶意篡改。其中一个原因在于，内容如果能够被篡改，那么同时意味着 Content-MD5 也可重新计算然后被篡改。所以处在接收阶段的客户端是无法意识到报文主体以及首部字段 Content-MD5 是已经被篡改过的。

6.6.7 Content-Range

130



```
Content-Range: bytes 5001-10000/10000
```

针对范围请求，返回响应时使用的首部字段 `Content-Range`，能告知客户端作为响应返回的实体的哪个部分符合范围请求。字段值以字节为单位，表示当前发送部分及整个实体大小。

6.6.8 Content-Type

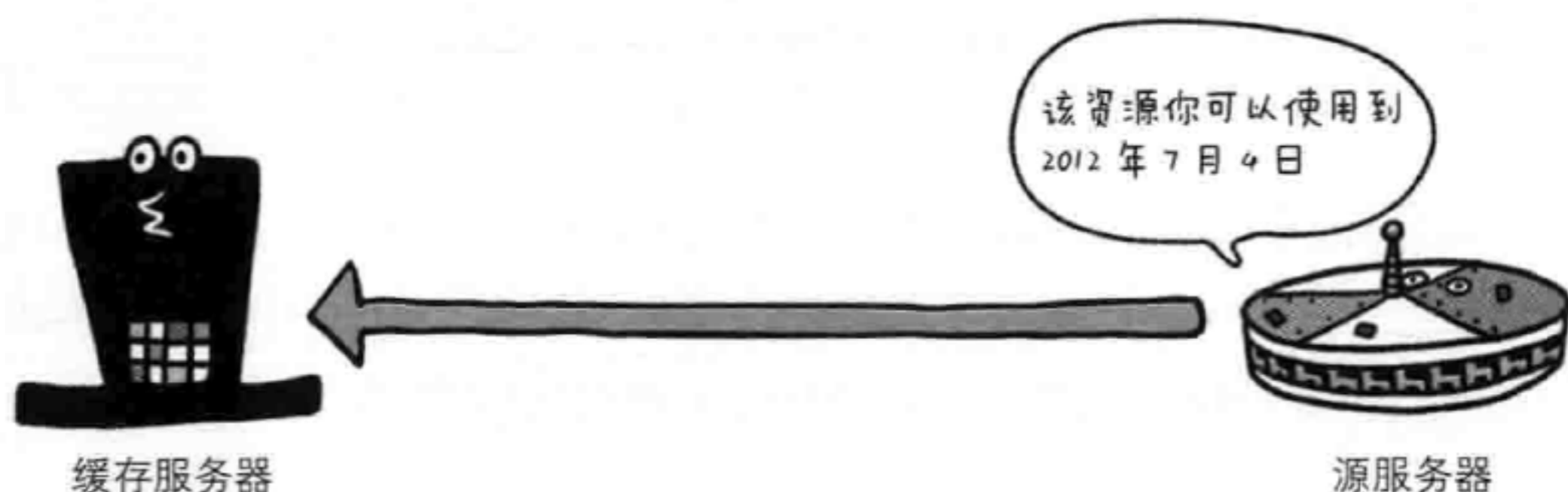
```
Content-Type: text/html; charset=UTF-8
```

首部字段 `Content-Type` 说明了实体主体内对象的媒体类型。和首部字段 `Accept` 一样，字段值用 `type/subtype` 形式赋值。

参数 `charset` 使用 `iso-8859-1` 或 `euc-jp` 等字符集进行赋值。

131

6.6.9 Expires



```
Expires: Wed, 04 Jul 2012 08:26:05 GMT
```

首部字段 `Expires` 会将资源失效的日期告知客户端。缓存服务器在接收到含有首部字段 `Expires` 的响应后，会以缓存来应答请求，在

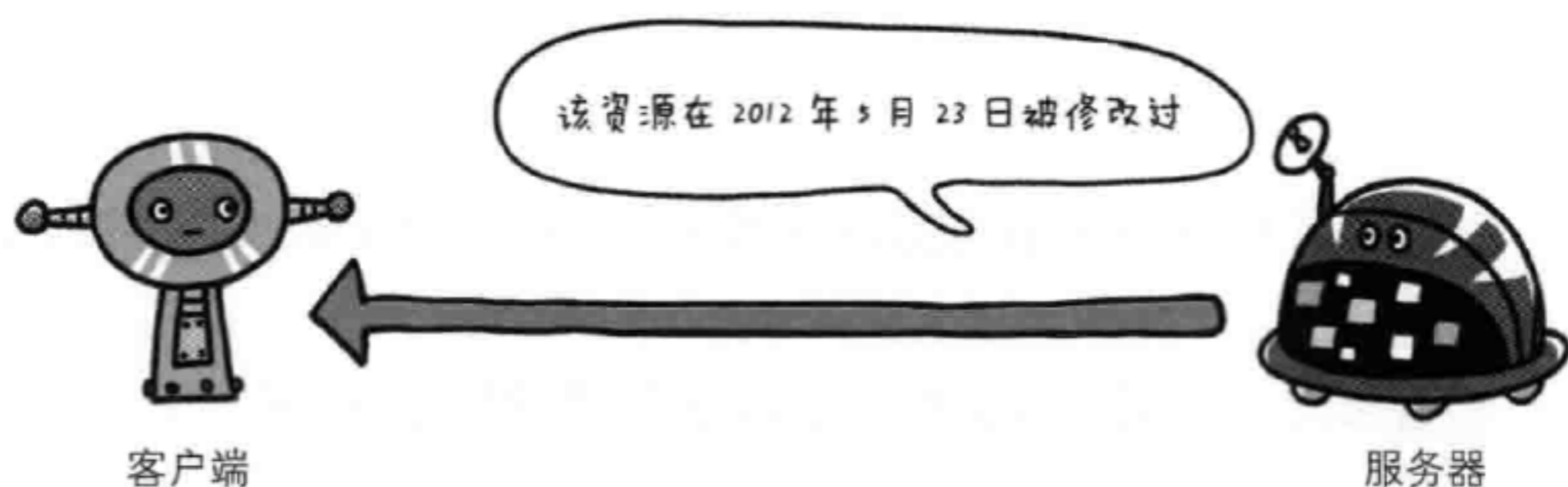


Expires 字段值指定的时间之前，响应的副本会一直被保存。当超过指定的时间后，缓存服务器在请求发送过来时，会转向源服务器请求资源。

源服务器不希望缓存服务器对资源缓存时，最好在 Expires 字段内写入与首部字段 Date 相同的时间值。

但是，当首部字段 Cache-Control 有指定 max-age 指令时，比起首部字段 Expires，会优先处理 max-age 指令。

6.6.10 Last-Modified



Last-Modified: Wed, 23 May 2012 09:59:55 GMT

首部字段 Last-Modified 指明资源最终修改的时间。一般来说，这个值就是 Request-URI 指定资源被修改的时间。但类似使用 CGI 脚本进行动态数据处理时，该值有可能会变成数据最终修改时的时间。

6.7 为 Cookie 服务的首部字段

管理服务器与客户端之间状态的 Cookie，虽然没有被编入标准化 HTTP/1.1 的 RFC2616 中，但在 Web 网站方面得到了广泛的应用。

Cookie 的工作机制是用户识别及状态管理。Web 网站为了管理用户的状态会通过 Web 浏览器，把一些数据临时写入用户的计算机内。接

着当用户访问该 Web 网站时，可通过通信方式取回之前发放的 Cookie。

调用 Cookie 时，由于可校验 Cookie 的有效期，以及发送方的域、路径、协议等信息，所以正规发布的 Cookie 内的数据不会因来自其他 Web 站点和攻击者的攻击而泄露。

至 2013 年 5 月，Cookie 的规格标准文档有以下 4 种。

由网景公司颁布的规格标准

网景通信公司设计并开发了 Cookie，并制定相关的规格标准。1994 年前后，Cookie 正式应用在网景浏览器中。目前最为普及的 Cookie 方式也是以此为基准的。

RFC2109

某企业尝试以独立技术对 Cookie 规格进行标准化统筹。原本的意图是想和网景公司制定的标准交互应用，可惜发生了微妙的差异。现在该标准已淡出了人们的视线。

RFC2965

为终结 Internet Explorer 浏览器与 Netscape Navigator 的标准差异而导致的浏览器战争，RFC2965 内定义了新的 HTTP 首部 Set-Cookie2 和 Cookie2。可事实上，它们几乎没怎么投入使用。

RFC6265

将网景公司制定的标准作为业界事实标准（De facto standard），重新定义 Cookie 标准后的产物。

目前使用最广泛的 Cookie 标准却不是 RFC 中定义的任何一个。而是在网景公司制定的标准上进行扩展后的产物。

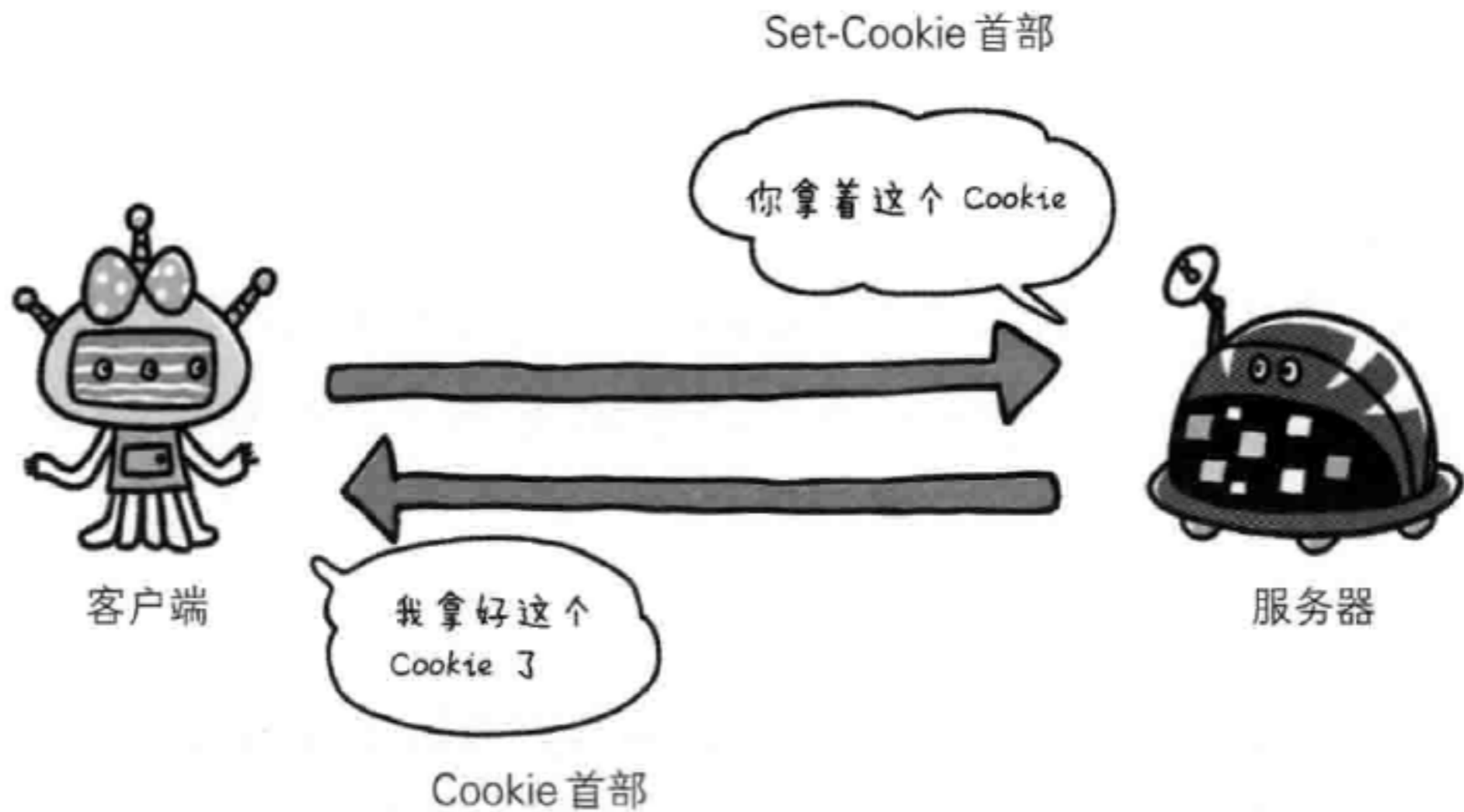
本节接下来就对目前使用最为广泛普及的标准进行说明。

下面的表格内列举了与 Cookie 有关的首部字段。



表 6-8: 为 Cookie 服务的首部字段

首部字段名	说明	首部类型
Set-Cookie	开始状态管理所使用的 Cookie 信息	响应首部字段
Cookie	服务器接收到的 Cookie 信息	请求首部字段



6.7.1 Set-Cookie

```
Set-Cookie: status=enable; expires=Tue, 05 Jul 2011 07:26:31 GMT; =>
path=/; domain=.hackr.jp;
```

当服务器准备开始管理客户端的状态时，会事先告知各种信息。
下面的表格列举了 Set-Cookie 的字段值。

表 6-9: Set-Cookie 字段的属性

属性	说明
NAME=VALUE	赋予 Cookie 的名称和其值（必需项）
expires=DATE	Cookie 的有效期（若不明确指定则默认为浏览器关闭前为止）
path=PATH	将服务器上的文件目录作为 Cookie 的适用对象（若不指定则默认为文档所在的文件目录）
domain= 域名	作为 Cookie 适用对象的域名（若不指定则默认为创建 Cookie 的服务器的域名）

(续)

属性	说明
Secure	仅在 HTTPS 安全通信时才会发送 Cookie
HttpOnly	加以限制，使 Cookie 不能被 JavaScript 脚本访问

expires 属性

Cookie 的 expires 属性指定浏览器可发送 Cookie 的有效期。

当省略 expires 属性时，其有效期仅限于维持浏览器会话（Session）时间段内。这通常限于浏览器应用程序被关闭之前。

另外，一旦 Cookie 从服务器端发送至客户端，服务器端就不存在可以显式删除 Cookie 的方法。但可通过覆盖已过期的 Cookie，实现对客户端 Cookie 的实质性删除操作。

path 属性

Cookie 的 path 属性可用于限制指定 Cookie 的发送范围的文件目录。不过另有办法可避开这项限制，看来对其作为安全机制的效果不能抱有期待。

domain 属性

通过 Cookie 的 domain 属性指定的域名可做到与结尾匹配一致。比如，当指定 example.com 后，除 example.com 以外，www.example.com 或 www2.example.com 等都可以发送 Cookie。

因此，除了针对具体指定的多个域名发送 Cookie 之外，不指定 domain 属性显得更安全。

secure 属性

Cookie 的 secure 属性用于限制 Web 页面仅在 HTTPS 安全连接时，才可以发送 Cookie。

发送 Cookie 时，指定 secure 属性的方法如下所示。



```
Set-Cookie: name=value; secure
```

以上例子仅当在 `https://www.example.com/` (HTTPS) 安全连接的情况下才会进行 Cookie 的回收。也就是说, 即使域名相同, `http://www.example.com/` (HTTP) 也不会发生 Cookie 回收行为。

当省略 `secure` 属性时, 不论 HTTP 还是 HTTPS, 都会对 Cookie 进行回收。

HttpOnly 属性

Cookie 的 `HttpOnly` 属性是 Cookie 的扩展功能, 它使 JavaScript 脚本无法获得 Cookie。其主要目的为防止跨站脚本攻击 (Cross-site scripting, XSS) 对 Cookie 的信息窃取。

发送指定 `HttpOnly` 属性的 Cookie 的方法如下所示。

```
Set-Cookie: name=value; HttpOnly
```

通过上述设置, 通常从 Web 页面内还可以对 Cookie 进行读取操作。但使用 JavaScript 的 `document.cookie` 就无法读取附加 `HttpOnly` 属性后的 Cookie 的内容了。因此, 也就无法在 XSS 中利用 JavaScript 劫持 Cookie 了。

虽然是独立的扩展功能, 但 Internet Explorer 6 SP1 以上版本等当下的主流浏览器都已经支持该扩展了。另外顺带一提, 该扩展并非是为了防止 XSS 而开发的。

6.7.2 Cookie

```
Cookie: status=enable
```

首部字段 Cookie 会告知服务器，当客户端想获得 HTTP 状态管理支持时，就会在请求中包含从服务器接收到的 Cookie。接收到多个 Cookie 时，同样可以以多个 Cookie 形式发送。

6.8 其他首部字段

HTTP 首部字段是可以自行扩展的。所以在 Web 服务器和浏览器的应用上，会出现各种非标准的首部字段。

接下来，我们就一些最为常用的首部字段进行说明。

- X-Frame-Options
- X-XSS-Protection
- DNT
- P3P

137

6.8.1 X-Frame-Options

```
X-Frame-Options: DENY
```

首部字段 X-Frame-Options 属于 HTTP 响应首部，用于控制网站内容在其他 Web 网站的 Frame 标签内的显示问题。其主要目的是为了防止点击劫持（clickjacking）攻击。

首部字段 X-Frame-Options 有以下两个可指定的字段值。

- DENY：拒绝
- SAMEORIGIN：仅同源域名下的页面（Top-level-browsing-context）匹配时许可。（比如，当指定 `http://hackr.jp/sample.html` 页面为 SAMEORIGIN 时，那么 `hackr.jp` 上所有页面的 frame 都被允许可加载该页面，而 `example.com` 等其他域名的页面就不行了）