图解 HTTP 二（简单的 HTTP 协议）

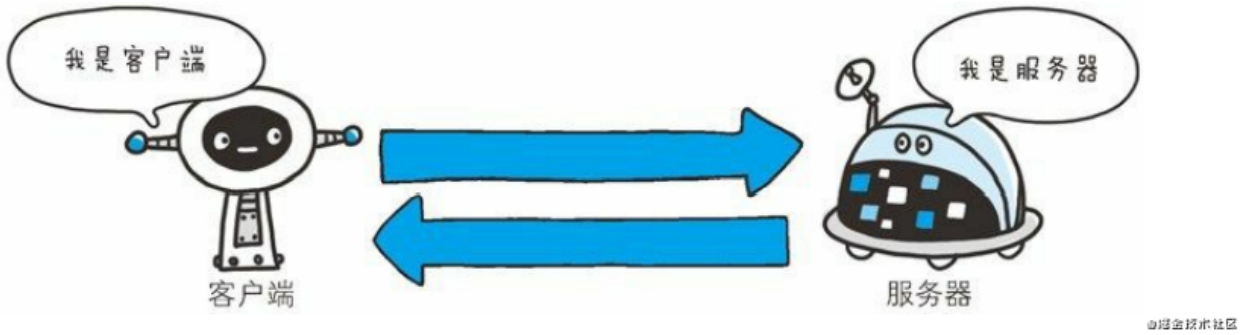
海底月是天上月，眼前人是心上人 —— 夏吉尔硕

起步：

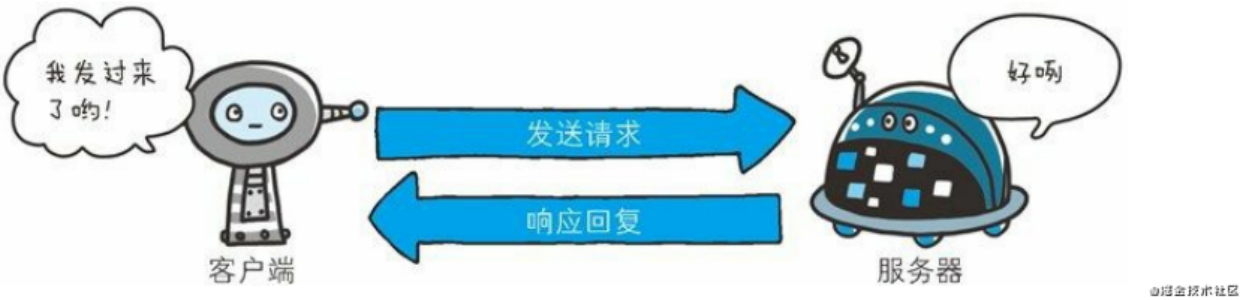
本章内容及进度根据《图解 HTTP》图灵系列丛书，加上自己搜索的互联网资料补充总结。

关于客户端和服务端

- 请求访问文本或图像等资源的一端称为客户端，而提供资源响应的一 端称为服务器端。



- 在应用 HTTP 协议通信时，在通信线路上必定一端是客户端，另一端是服务器端。
- 有时，两者的身份可能会互换，则使用 HTTP 协议能够明确区分两端的身份。



- **请求必定是客户端，响应的必定为服务端，如下图：**

- 相应报文构成

无状态协议——HTTP

- HTTP 属于无状态协议，即：不保存请求和响应之间的通信状态。
- 协议对发送过的请求或响应都不做持久化处理。
  - 对于 HTTP 协议来说，当有新请求发送时，就会产生对应的新响应。
  - 协议本身不会保留之前一切的请求和响应信息。
  - 原因：为了更快处理大量事务。
- 棘手的问题
  - 随着业务发展，无状态在实际应用中带来的问题越来越多
  - 为了保留状态信息，解决无状态协议带来的麻烦，引入`Cookie`技术
  - 在后边状态管理里介绍 `Cookie`

与服务器交流的 HTTP 方法

GET：获取资源

- `GET` 方法用以请求被 URI 识别的资源

- 指定的资源经服务器端进行解析后返回的响应内容

请求	GET /index.html HTTP/1.1 Host: www.hackr.jp If-Modified-Since: Thu, 12 Jul 2012 07:30:00 GMT
响应	仅返回2012年7 月12日7 点30分以后更新过的index.html页面资源。如果未有内容更新，则以状态码304 Not Modified作为响应返回

POST：传输实体主体

- GET 方法也可以进行实体传输，但一般只使用POST传输
- POST 方法主要也不是为了获取响应的主体内容

请求	POST /submit.cgi HTTP/1.1 Host: www.hackr.jp Content-Length: 1560（1560字节的数据）
响应	返回 submit.cgi 接收数据的处理结果

PUT：传输文件

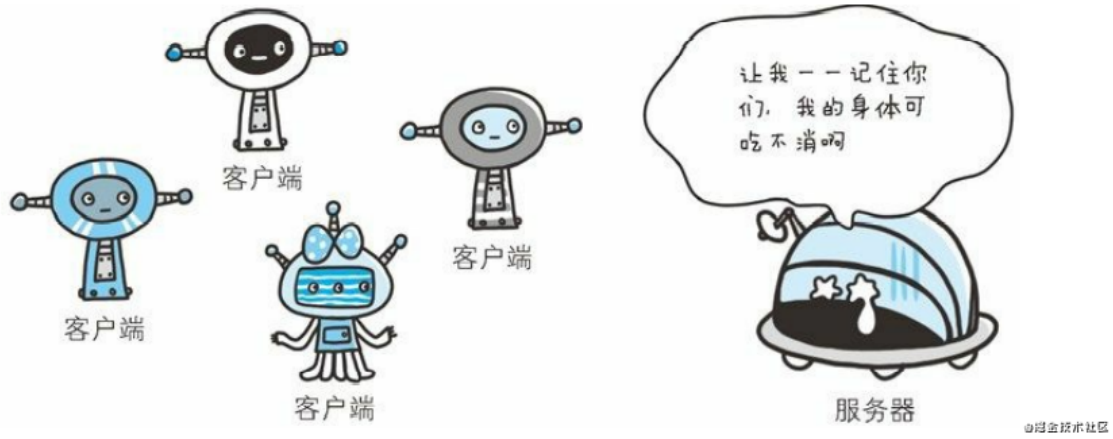
- PUT 方法，在请求报文的主体中包含文件内容，保存到请求 URI 指定位置
- 由于 HTTP/1.1 的PUT方法无验证机制，因此存在安全性问题
- 因此 PUT 方法一般不开放

HEAD：获得报文首部

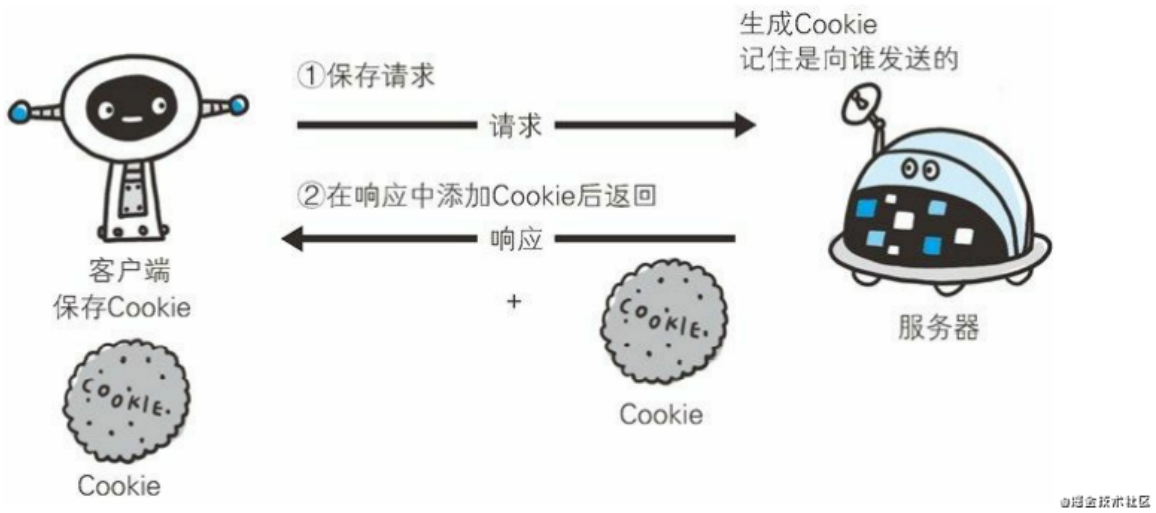
- HEAD 方法用于确认 URI 的有效性及资源更新的日期时间等
- HEAD 方法与 GET 相同，只是不返回报文主体部分

使用 Cookie 的状态管理

- 由于 HTTP 是无状态协议，不保存之前发送过的请求与响应状态，无法依据之前状态进行本次请求处理
- 因此会出现每次跳转新页面，都需要再次登录这种低效的事件
- 解决方案：
    - 每次请求报文中附加参数，管理登录状态
    - 但管理全部客户端状态又会产生很大的负担

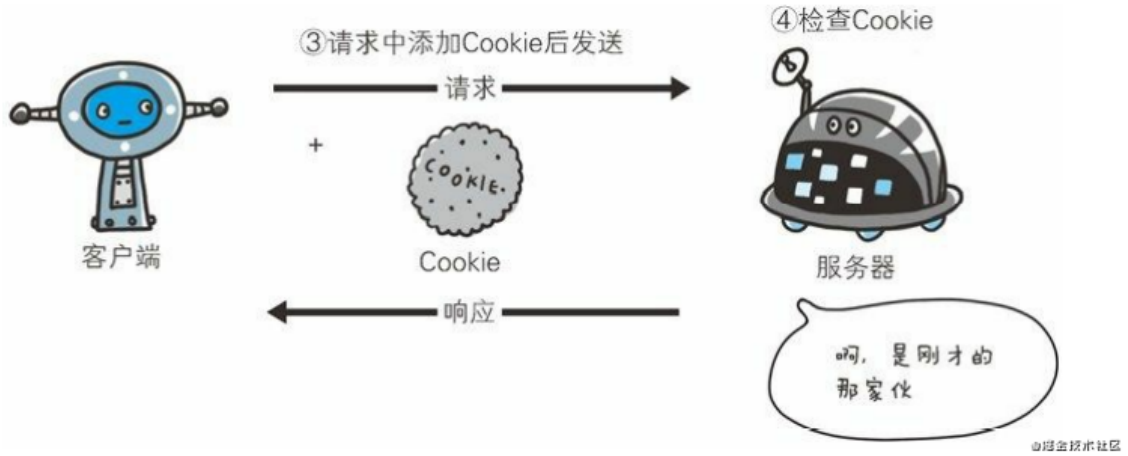


- 正是基于上方矛盾的考虑，所以引入了 `Cookie` 技术
    - 它会通过在请求和响应报文中写入 `Cookie` 信息来控制客户端状态
    - 通过发送响应报文中 `Set-Cookie` 的字段信息，通知 **客户端保存 `Cookie`**



    - 下次客户端向服务端发起请求时，自动在请求报文中加入 `Cookie` 值
    - 服务端接收客户端发送的 `Cookie` 后，会与服务器比对，找出之前状态信息

- 第 **2** 次以后（存有 **Cookie** 信息状态）的请求



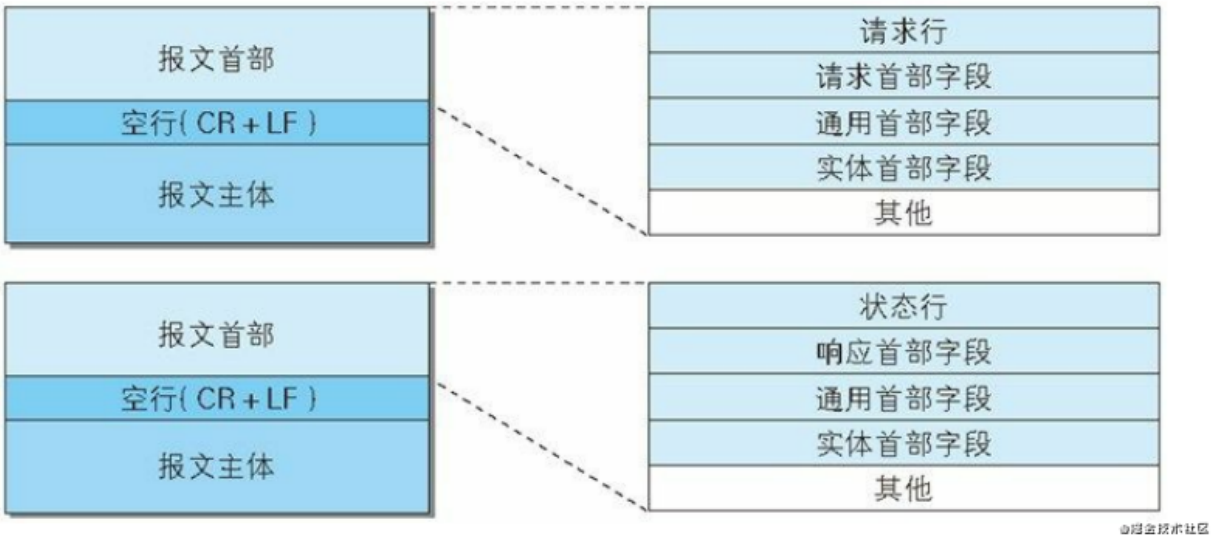
- 上图介绍了理论上的交互场景，具体发送报文如下

```
/* 1.请求报文（无cookie信息）*/
GET /reader/ HTTP/1.1
Host: hackr.jp

/* 2. 响应报文（服务器端生成 Cookie 信息）*/
HTTP/1.1 200 OK
Date: Thu, 12 Jul 2012 07:12:20 GMT
Server: Apache
<Set-Cookie: sid=1342077140226724; path=/; expires=Wed,
10-Oct-12 07:12:20 GMT>
Content-Type: text/plain; charset=UTF-8

/*3. 请求报文（自动发送保存着的 Cookie 信息）*/
GET /image/ HTTP/1.1
Host: hackr.jp
Cookie: sid=1342077140226724
```

报文中的 HTTP 信息

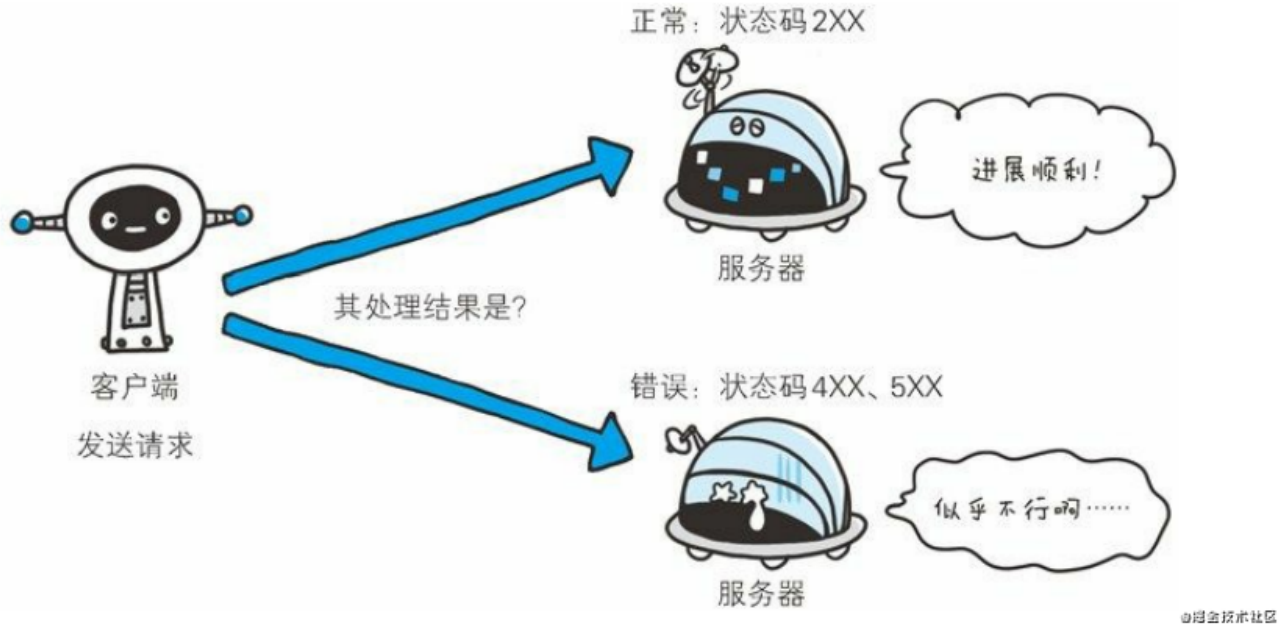
HTTP 报文

- HTTP协议交互的信息称为 HTTP 报文
  - 客户端：称为请求报文
  - 服务端：称为相应报文
- HTTP 报文内容分为，报文首部和报文主体（非必须）
- 请求报文与响应报文结构如下
  - 请求行：包含请求方法，请求 URI 和 HTTP 版本。
  - 状态行：包含响应结果的状态码，原因短语和 HTTP 版本。
  - 首部字段：包含请求和响应的各种条件和属性的各类首部。
    - 一般有：通用，请求，响应，实体四种首部。



HTTP 状态码

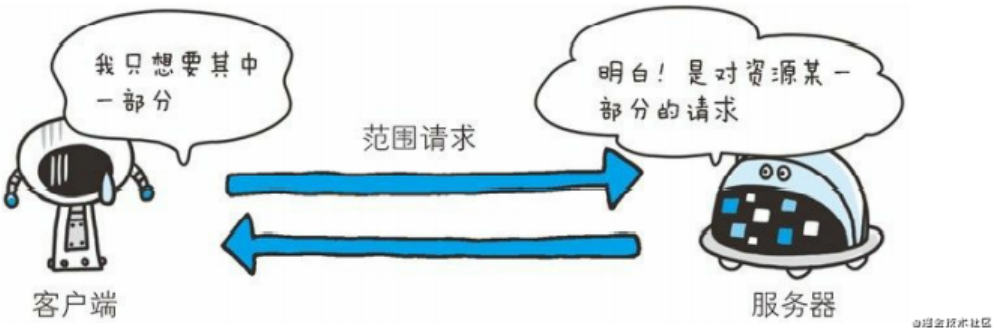
- 状态码表示客户端 HTTP 请求的结果、标记服务器端的处理是否正常、通知出现的错误等。



- 状态码的类别

	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

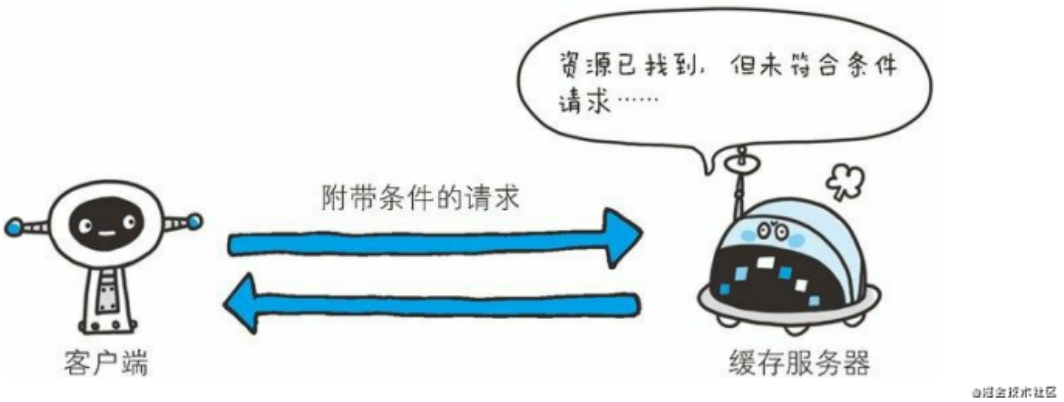
- 常用的状态码解析为
    - 200 OK ： 表示客户端请求在服务器端正常处理。
    - 204 No Content：
        - 表服务端对请求成功处理，但在返回的响应报文中不含实体的主体部分
        - 在只需客户端往服务器发送信息，而对客户端无需发送新信息下使用。
    - 206 Partial Content
        - 该状态码表示客户端进行了范围请求
        - 响应报文中包含由 Content-Range 指定范围的实体内容



    - 3XX 重定向：表明浏览器需要执行某些特殊的处理以正确处理请求。
    - 301 Moved Permanently（永久性重定向）
        - 表示请求的资源已被分配了新的 URI
        - 若把资源对应的 URI 保存为书签，则应按 Location 首部字段提示的 URI 重新保存
    - 302 Found（临时重定向）
        - 与 301 类似，只不过它表示临时重定向
        - 此时不会像 301 那样，去更新书签
    - 303 See Other
        - 表示请求对应的资源存在另一个 URI，应使用 GET 方法定向获取请求的资源。

> 当 301、302、303 响应状态码返回时，几乎所有的浏览器都会把 POST 改成 GET，并删除请求报文内的主体，之后请求会自动再次发送。

- 304 Not Modified
    - 表示客户端发送附带条件的请求时，服务器端允许请求访问资源，但未满足条件的情况。



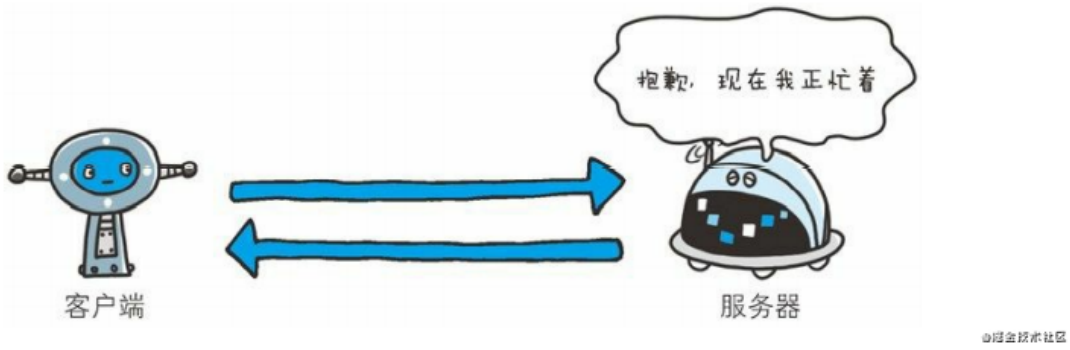
- 4XX 客户端错误

> 4XX 的响应结果表明客户端是发生错误的原因所在。

- 400 Bad Request：该状态码表示请求报文中存在语法错误
- 401 Unauthorized（未经授权）
  - 表示发送的请求需要有通过 HTTP 认证（BASIC 认证、DIGEST 认证）的认证信息
  - 若之前已进行过 1 次请求，则表示用户认证失败
- 403 Forbidden（禁止的）
  - 表明对请求资源的访问被服务器拒绝
- 404 Not Found（未找到）
  - 表明服务器上无法找到请求的资源
- 5XX 服务器错误

> 5XX 的响应结果表明服务器本身发生错误。

- 500 Internal Server Error（内部服务器错误）
  - 表明服务器端在执行请求时发生了错误
  - 也可能是 Web 应用存在的 bug 或某些临时的故障
- 503 Service Unavailable（暂停服务）
  - 表明服务器暂时处于超负载或进行停机维护，现在无法处理请求



** HTTP 首部字段**

4 种 HTTP 首部字段类型

- 通用首部字段（General Header Fields）
  - 请求报文和响应报文两方都会使用的首部。
- 请求首部字段（Request Header Fields）
  - 客户端向服务器端发送请求报文时使用的首部。补充请求的附加内容、响应优先级等。
- 响应首部字段（Response Header Fields）
  - 服务器端向客户端返回响应报文时使用的首部。补充响应的附加内容，要求客户端附加的信息。
- 实体首部字段（Entity Header Fields）
  - 针对请求报文和响应报文的实体部分使用的首部。补充资源内容更新时间等信息。

首部字段表

- 通用首部字段表

首部字段名	说明
Cache-Control	控制缓存的行为
Connection	逐跳首部、连接的管理
Date	创建报文的日期时间
Pragma	报文指令
Trailer	报文末端的首部一览
Transfer-Encoding	指定报文主体的传输编码方式
Via	代理服务器的相关信息
Warning	错误通知

- 请求首部字段（部分）

首部字段名	说明
Accept	用户代理可处理的媒体类型
Accept-Charset	优先的字符集
Accept-Language	优先的语言（自然语言）
Authorization	Web 认证信息
Expect	期待服务器的特定行为
From	用户的电子邮箱地址
Host	请求资源所在服务器
If-Match	比较实体标记（ETag）
If-Modified-Since	比较资源的更新时间
If-None-Match	比较实体标记（与 If-Match 相反）
If-Range	资源未更新时发送实体 Byte 的范围请求
Proxy-Authorization	代理服务器要求客户端的认证信息
Referer	对请求中 URI 的原始获取方
User-Agent HTTP	客户端程序的信息

- 响应首部字段（部分）

首部字段名	说明

首部字段名	说明
Accept-Ranges	是否接受字节范围请求
Age	推算资源创建经过时间
ETag	资源的匹配信息
Location	令客户端重定向至指定 URI
Proxy-Authenticate	代理服务器对客户端的认证信息
Retry-After	对再次发起请求的时机要求
Server HTTP	服务器的安装信息
Vary	代理服务器缓存的管理信息
WWW-Authenticate	服务器对客户端的认证信息

HTTP缓存—Cache-Control

- 通过指定首部字段 Cache-Control 的指令，操作缓存的工作机制。
- 指令的参数是可选的，多个指令之间通过“,”分隔。
- 缓存请求指令表

指令	参数	说明
no-cache	无	强制向源服务器再次验证
no-store	无	不缓存请求或响应的任何内容
max-age = [秒]	必需	响应的最大 Age 值
max-stale(= [秒])	可省略	接收已过期的响应
min-fresh = [秒]	必需	期望在指定时间内的响应仍有效
only-if-cached	无	从缓存获取资源

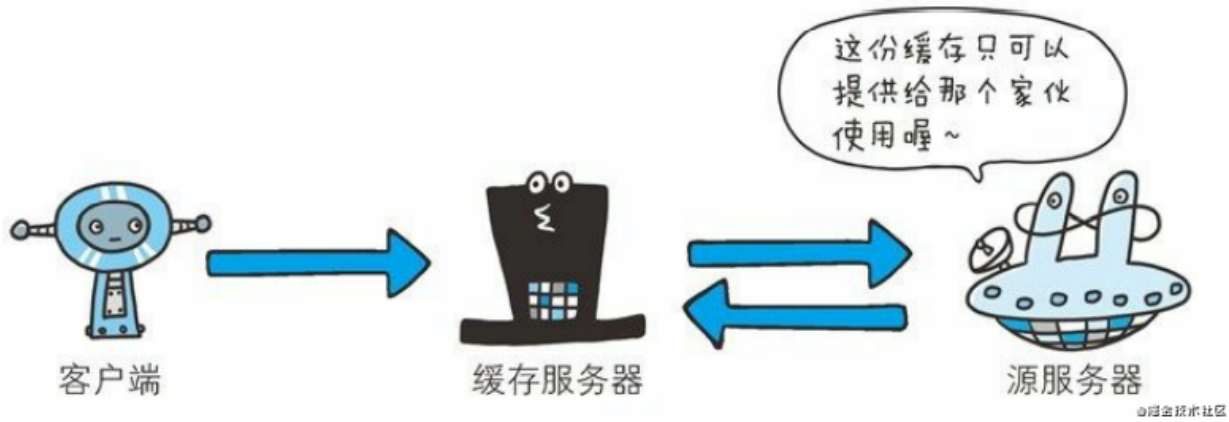
- 缓存响应指令

指令	参数	说明
public	无	可向任意方提供响应的缓存
private	可省略	仅向特定用户返回响应
no-cache	可省略	缓存前必须先确认其有效性
no-store	无	不缓存请求或响应的任何内容
no-transform	无	代理不可更改媒体类型
must-revalidate	无	可缓存但必须再向源服务器进行确认

- 当使用 `public` 指令时，则明确表明其他用户也可利用缓存。

  > Cache-Control: public

- 当使用 `private` 指令 响应只以特定的用户作为对象，与 `public` 指令行为相反。



总结：

HTTP 协议的请求和响应报文中必定包含 HTTP 首部，虽然普通人在实际使用中根本感受不到它，但对于开发者来说，掌握这些信息，不仅能学会请求报文发送时的各种配置，更便于提升在实际生产中 debug 的效率