HTTP

@M了个J
李明杰

https://github.com/CoderMJLee

https://space.bilibili.com/325538782

小码哥教育 SEEMYGO

实力IT教育 www.520it.com

小码哥教育 SEEMYGO

HTTP

- HTTP（Hyper Text Transfer Protocol），译为超文本传输协议
  - 是互联网中应用最广泛的应用层协议之一
  - 设计HTTP最初的目的是：提供一种发布和接收HTML页面的方法，由URI来标识具体的资源
  - 后面用HTTP来传递的数据格式不仅仅是HTML，应用非常广泛

- HTML（ Hyper Text Markup Language）：超文本标记语言
  - 用以编写网页

小码哥教育 SEEMYGO

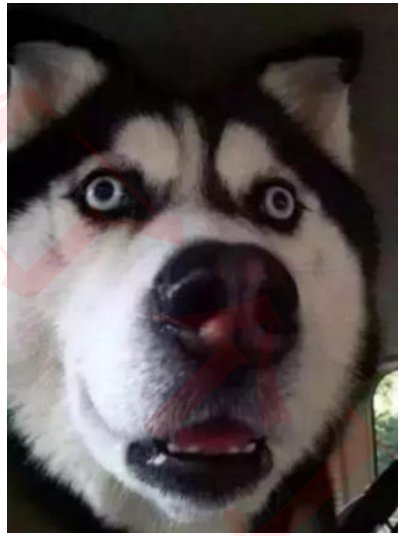
百度百科

http

本词条由"科普中国"科学百科词条编写与应用工作项目 审核 。

http是一个简单的请求-响应协议，它通常运行在TCP之上。它指定了客户端可能发送给服务器什么样的消息以及得到什么样的响应。请求和响应消息的头以ASCII码形式给出；而消息内容则具有一个类似MIME的格式。这个简单模型是早期Web成功的有功之臣，因为它使开发和部署非常地直截了当。

中文名	超文本传输协议	基　础	架构在TCP协议上
外文名	HTTP	适用浏览器	Firefox、Google chrome等
工作层	应用层	作　用	规定WWW服务器与浏览器之间信息传递规范



小码哥教育 SEEMYGO

维基百科

超文本传输协议 [编辑]

维基百科，自由的百科全书

超文本传输协议（英语：**HyperText Transfer Protocol**，缩写：**HTTP**）是一种用于分布式、协作式和超媒体信息系统的应用层协议[1]。HTTP是万维网的数据通信的基础。

设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。通过HTTP或者HTTPS协议请求的资源由统一资源标识符（Uniform Resource Identifiers，URI）来标识。

HTTP的发展是由蒂姆·伯纳斯-李于1989年在欧洲核子研究组织（CERN）所发起。HTTP的标准制定由万维网协会（World Wide Web Consortium，W3C）和互联网工程任务组（Internet Engineering Task Force，IETF）进行协调，最终发布了一系列的RFC，其中最著名的是1999年6月公布的RFC 2616，定义了HTTP协议中现今广泛使用的一个版本——HTTP 1.1。

2014年12月，互联网工程任务组（IETF）的Hypertext Transfer Protocol Bis（httpbis）工作小组将HTTP/2标准提议递交至IESG进行讨论[2]，于2015年2月17日被批准。[3] HTTP/2标准于2015年5月以RFC 7540正式发表，取代HTTP 1.1成为HTTP的实现标准。[4]

小码哥教育 SEEMYGO

版本

- 1991年，HTTP/0.9
  - 只支持GET请求方法获取文本数据（比如HTML文档），且不支持请求头、响应头等，无法向服务器传递太多信息
- 1996年，HTTP/1.0
  - 支持POST、HEAD等请求方法，支持请求头、响应头等，支持更多种数据类型（不再局限于文本数据）
  - 浏览器的每次请求都需要与服务器建立一个TCP连接，请求处理完成后立即断开TCP连接
- 1997年，HTTP/1.1（最经典、使用最广泛的版本）
  - 支持PUT、DELETE等请求方法
  - 采用持久连接（Connection: keep-alive），多个请求可以共用同一个TCP连接
- 2015年，HTTP/2.0
- 2018年，HTTP/3.0

小码哥教育 SEEMYGO

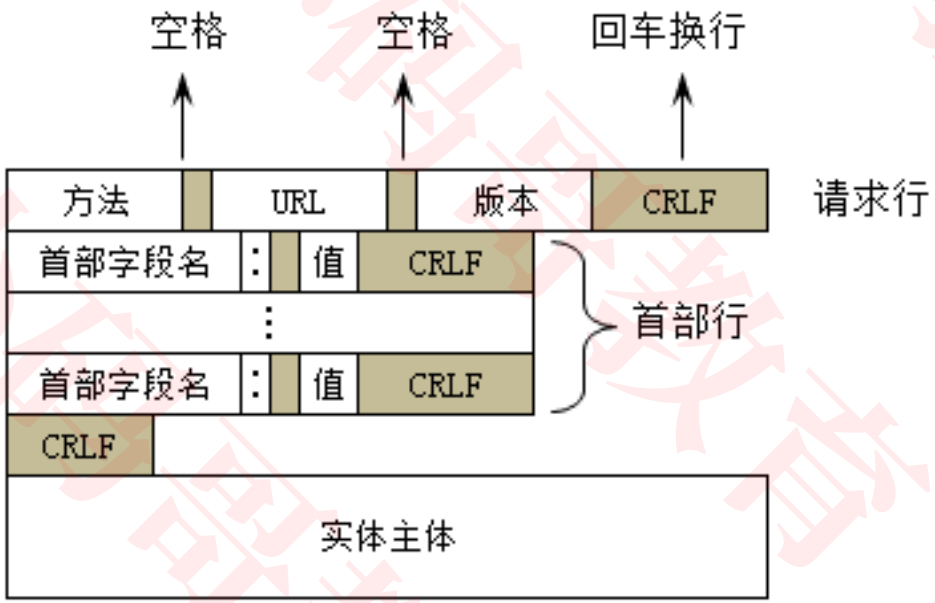
标准

- HTTP的标准
  - 由万维网协会（W3C）、互联网工程任务组（IETF）协调制定，最终发布了一系列的RFC

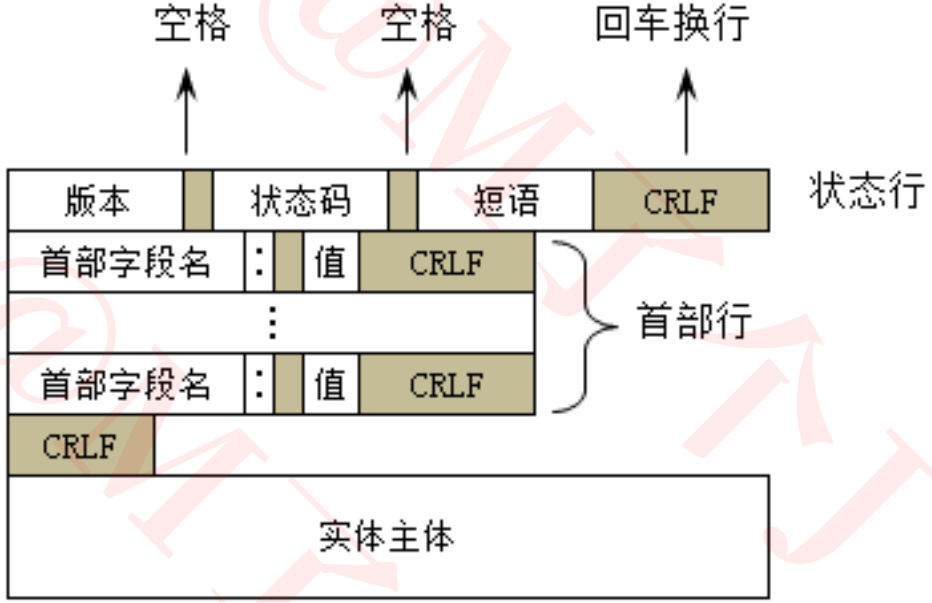
- RFC（Request For Comments，可以译为：请求意见稿）
  - HTTP/1.1最早是在1997年的RFC 2068中记录的
    - ✓ 该规范在1999年的RFC 2616中已作废
    - ✓ 2014年又由RFC 7230系列的RFC取代
  - HTTP/2标准于2015年5月以RFC 7540正式发表，取代HTTP/1.1成为HTTP的实现标准

- 中国的RFC
  - 1996年3月，清华大学提交的适应不同国家和地区中文编码的汉字统一传输标准被IETF通过为RFC 1922
  - 成为中国大陆第一个被认可为RFC文件的提交协议

报文格式



（a）请求报文



（b）响应报文

小码哥教育 SEEMYGO

ABNF

- ABNF（Augmented BNF）
  - 是BNF（Backus-Naur Form，译为：巴科斯-瑙尔范式）的修改、增强版
  - 在RFC 5234中表明：ABNF用作internet中通信协议的定义语言
  - ABNF是最严谨的HTTP报文格式描述形式，脱离ABNF谈论HTTP报文格式，往往都是片面、不严谨的

- 关于HTTP报文格式的定义
  - RFC 2616 4.HTTP Message（旧）
  - RFC 7230 3.Message Format（新）

小码哥教育 SEEMYGO

ABNF – 核心规则

规则	形式定义	意义
ALPHA	%x41-5A / %x61-7A	大写和小写ASCII字母（A-Z，a-z）
DIGIT	%x30-39	数字（0-9）
HEXDIG	DIGIT / "A" / "B" / "C" / "D" / "E" / "F"	十六进制数字（0-9，A-F，a-f）
DQUOTE	%x22	双引号
SP	%x20	空格
HTAB	%x09	横向制表符
WSP	SP / HTAB	空格或横向制表符
LWSP	*(WSP / CRLF WSP)	直线空白（晚于换行）
VCHAR	%x21-7E	可见（打印）字符
CHAR	%x01-7F	任何7-位US-ASCII字符，不包括NUL（%x00）
OCTET	%x00-FF	8位数据
CTL	%x00-1F / %x7F	控制字符
CR	%x0D	回车
LF	%x0A	换行
CRLF	CR LF	互联网标准换行（%x0D%x0A）
BIT	"0" / "1"	二进制数字

报文格式 – 整体

HTTP-message = start-line

*(header-field CRLF)

CRLF

[message-body]

start-line = request-line / status-line

/	任选一个
*	0个或多个。2*表示至少2个，3*6表示3到6个
( )	组成一个整体
[ ]	可选（可有可无）

报文格式 - request-line、status-line

request-line = method SP request-target SP HTTP-version CRLF

HTTP-version = HTTP-name "/" DIGIT "." DIGIT

HTTP-name = %x48.54.54.50 ; HTTP

```
GET /hello/ HTTP/1.1
```

status-line = HTTP-version SP status-code SP reason-phrase CRLF

status-code = 3DIGIT

reason-phrase = *(HTAB / SP / VCHAR / obs-text)

```
HTTP/1.1 200
HTTP/1.1 200 OK
```

报文格式 - header-filed、message-body

```
header-field = field-name ":" OWS field-value OWS

field-name = token

field-value = *( field-content / obs-fold )

OWS = *( SP / HTAB )


message-body = *OCTET
```

URL的编码

- URL中一旦出现了一些特殊字符（比如中文、空格），需要进行编码
  - 在浏览器地址栏输入URL时，是采用UTF-8进行编码

- 比如
  - 编码前：https://www.baidu.com/s?wd=百度
  - 编码后：https://www.baidu.com/s?wd=%E5%8D%8E%E4%B8%BA

Xshell + telnet

- 安装一个Xshell（安全终端模拟软件），在Xshell中使用telnet
  - 可以直接面向HTTP报文与服务器交互
  - 可以更清晰、直观地看到请求报文、响应报文的内容
  - 可以检验请求报文格式的正确与否

```
Xshell 6 (Build 0189)
Copyright (c) 2002 NetSarang Computer, Inc. All rights reserved.

Type `help' to learn how to use Xshell prompt.
[C:\~]$ telnet localhost 8080


Host 'localhost' resolved to ::1.
Connecting to ::1:8080...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
```

```
GET /hello/ HTTP/1.1
Host: localhost:8080

HTTP/1.1 200
Set-Cookie: JSESSIONID=985DEC26A52AB36
Content-Type: text/html;charset=UTF-8
Content-Length: 90
Date: Tue, 08 Dec 2020 08:10:14 GMT
```

请求方法

- RFC 7231, section 4: Request methods：描述了8种请求方法
  - GET、HEAD、POST、PUT、DELETE、CONNECT、OPTIONS、TRACE
- RFC 5789, section 2: Patch method：描述了PATCH方法
- GET：常用于读取的操作，请求参数直接拼接在URL的后面（浏览器对URL是有长度限制的）
- POST：常用于添加、修改、删除的操作，请求参数可以放到请求体中（没有大小限制）
- HEAD：请求得到与GET请求相同的响应，但没有响应体
  - 使用场景举例：在下载一个大文件前，先获取其大小，再决定是否要下载。以此可以节约带宽资源

请求方法

- OPTIONS：用于获取目的资源所支持的通信选项，比如服务器支持的请求方法
  - OPTIONS * HTTP/1.1
- PUT：用于对已存在的资源进行整体覆盖
- PATCH：用于对资源进行部分修改（资源不存在，会创建新的资源）
- DELETE：用于删除指定的资源
- TRACE：请求服务器回显其收到的请求信息，主要用于HTTP请求的测试或诊断
- CONNECT：可以开启一个客户端与所请求资源之间的双向沟通的通道，它可以用来创建隧道（tunnel）
  - 可以用来访问采用了 SSL (HTTPS) 协议的站点

头部字段（Header Field）

- 头部字段可以分为4种类型

- 请求头字段（Request Header Fields）
  - ✓有关要获取的资源或客户端本身信息的消息头

- 响应头字段（Response Header Fields）
  - ✓有关响应的补充信息，比如服务器本身（名称和版本等）的消息头

- 实体头字段（Entity Header Fields）
  - ✓有关实体主体的更多信息，比如主体长度（Content-Length）或其MIME类型

- 通用头字段（General Header Fields）
  - ✓同时适用于请求和响应消息，但与消息主体无关的消息头

小码哥教育 SEEMYGO

请求头字段

头字段名	说明	示例
User-Agent	浏览器的身份标识字符串	User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/21.0
Host	服务器的域名、端口号	Host: localhost:80
Date	发送该消息的日期和时间	Date: Tue, 15 Nov 1994 08:12:31 GMT
Referer	表示浏览器所访问的前一个页面 正是前一个页面的某个链接将浏览器带到了当前这个页面	Referer: https://www.baidu.com
Content-Type	请求体的类型	Content-Type: multipart/form-data
Content-Length	请求体的长度（字节为单位）	Content-Length: 348

小码哥教育 SEEMYGO

请求头字段

头字段名	说明	示例
Accept	能够接受的响应内容类型 (Content-Types)	Accept: text/plain
Accept-Charset	能够接受的字符集	Accept-Charset: GB2312,utf-8;q=0.7,*;q=0.7
Accept-Encoding	能够接受的编码方式列表	Accept-Encoding: gzip, deflate
Accept-Language	能够接受的响应内容的自然语言列表	Accept-Language: en-US

- q值越大，表示优先级越高
- 如果不指定q值，默认是1.0（1.0是最大值）

小码哥教育 SEEMYGO

请求头字段

头字段名	说明	示例
Range	仅请求某个实体的一部分。字节偏移以0开始	Range: bytes=500-999
Origin	发起一个针对跨域资源共享的请求	Origin: https://www.baidu.com
Cookie	之前由服务器通过Set-Cookie发送的Cookie	Cookie: $Version=1; Skin=new;
Connection	该浏览器想要优先使用的连接类型	Connection: keep-alive
Cache-Control	用来指定在这次的请求/响应链中的所有缓存机制都必须遵守的指令	Cache-Control: no-cache

小码哥教育 SEEMYGO

响应头字段

头字段名	说明	示例
Date	发送该消息的日期和时间	Date: Tue, 15 Nov 1994 08:12:31 GMT
Last-Modified	所请求的对象的最后修改日期	Last-Modified: Tue, 15 Nov 1994 12:45:26 GMT
Server	服务器的名字	Server: Apache/2.4.1 (Unix)
Expires	指定一个时间，超过该时间则认为此响应已经过期	Expires: Thu, 01 Dec 1994 16:00:00 GMT

小码哥教育 SEEMYGO

响应头字段

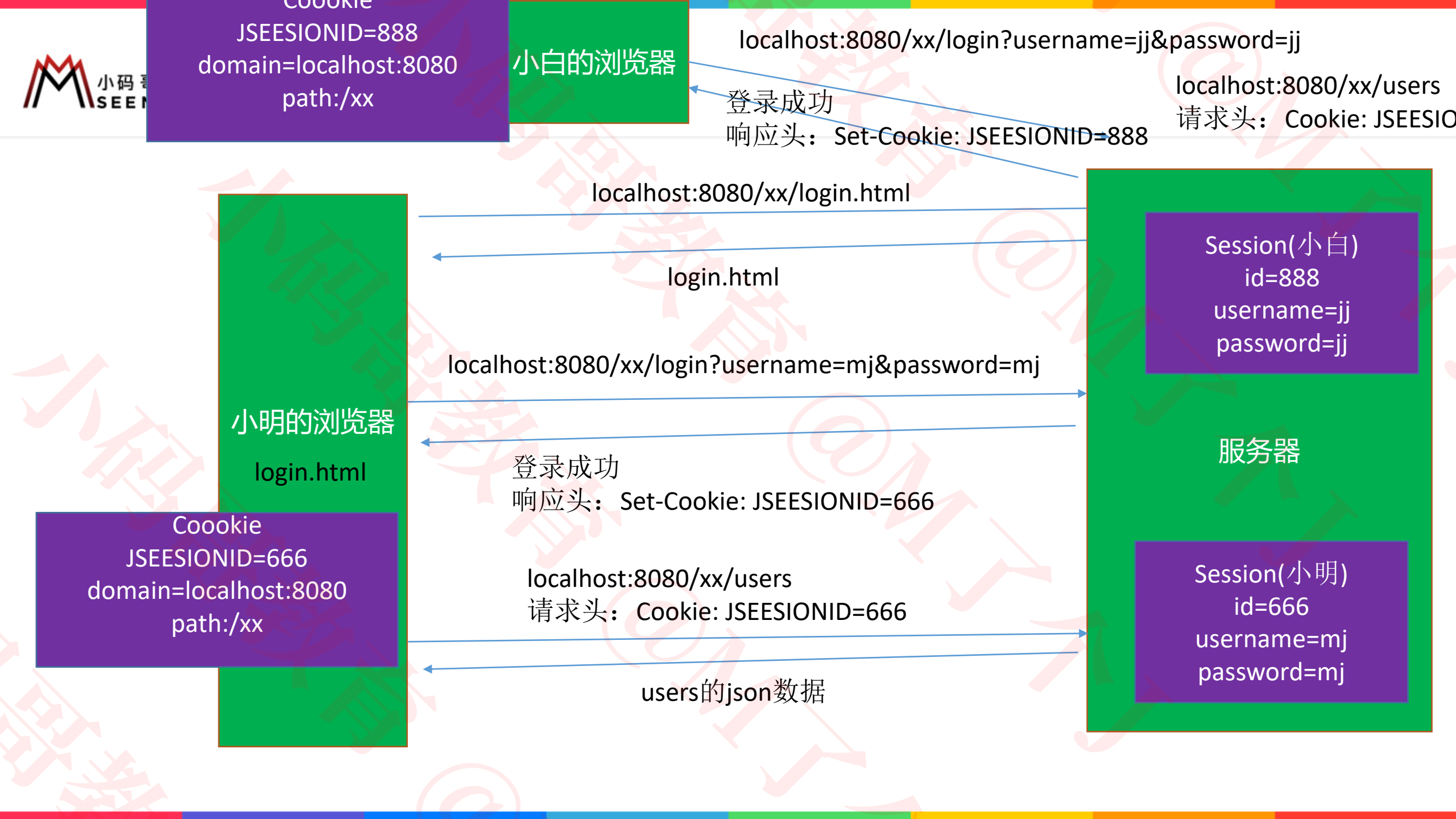
头字段名	说明	示例
Content-Type	响应体的类型	Content-Type: text/html; charset=utf-8
Content-Encoding	内容所使用的编码类型	Content-Encoding: gzip
Content-Length	响应体的长度（字节为单位）	Content-Length: 348
Content-Disposition	一个可以让客户端下载文件并建议文件名的头部	Content-Disposition: attachment; filename="fname.ext"
Accept-Ranges	服务器支持哪些种类的部分内容范围	Accept-Ranges: bytes
Content-Range	这条部分消息是属于完整消息的哪部分	Content-Range: bytes 21010-47021/47022

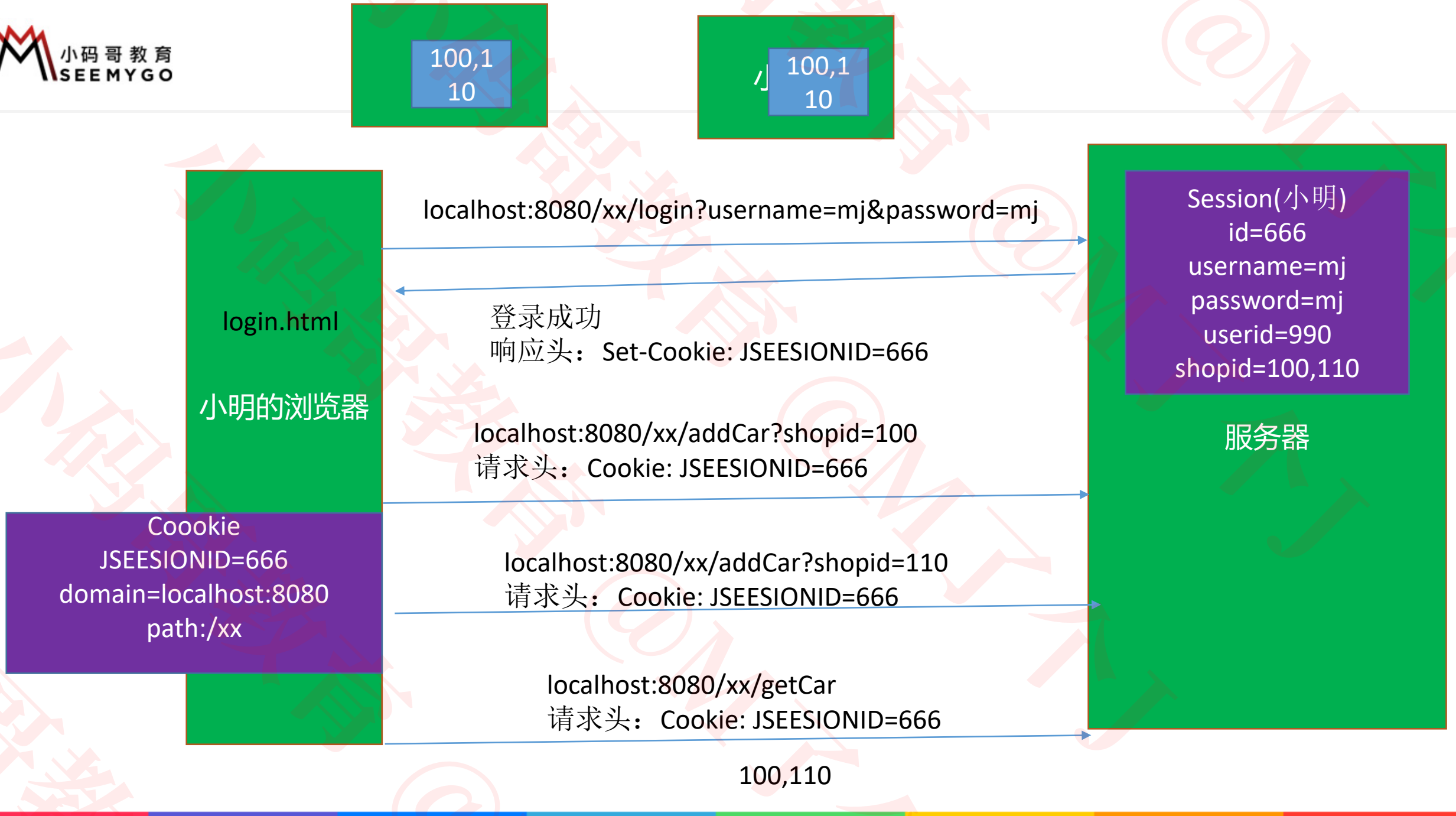
小码哥教育 SEEMYGO

响应头字段

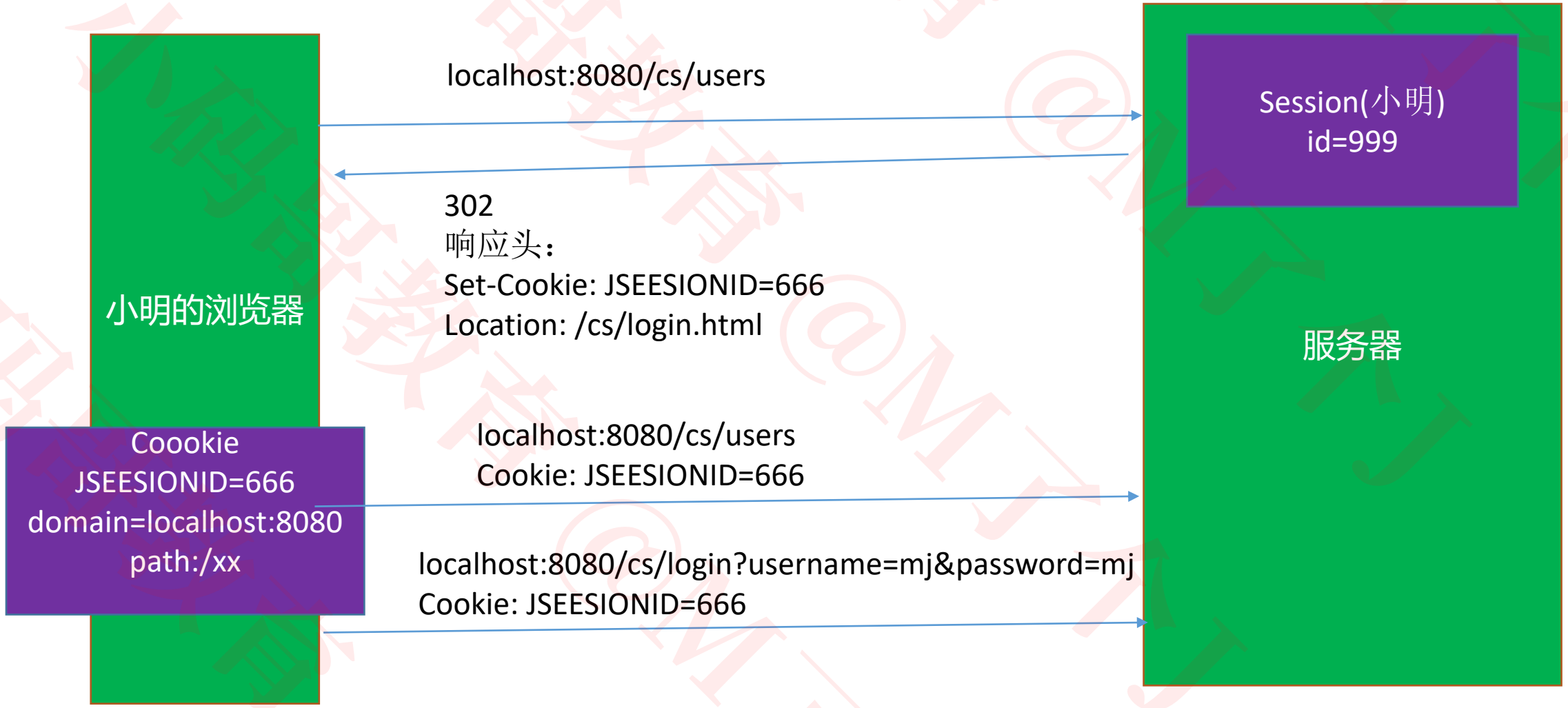
头字段名	说明	示例
Access-Control-Allow-Origin	指定哪些网站可参与到跨来源资源共享过程中	Access-Control-Allow-Origin: *
Location	用来进行重定向，或者在创建了某个新资源时使用	Location: http://www.w3.org
Set-Cookie	返回一个Cookie让客户端去保存	Set-Cookie: UserID=JohnDoe; Max-Age=3600; Version=1
Connection	针对该连接所预期的选项	Connection: close
Cache-Control	向从服务器直到客户端在内的所有缓存机制告知，它们是否可以缓存这个对象。单位为秒	Cache-Control: max-age=3600


01_HelloWorld F:\Dev\Java\Codes\Network\01_He
02_CORS F:\Dev\Java\Codes\Network\02_CORS
src
web
03_CORS_FrontEdge F:\Dev\Java\Codes\Network\
js
index.html
XmgResume
External Libraries
Scratches and Consoles
localhost:63342
页面服务器（Nginx）
Search Everywhere Double Shift
03_CORS_FrontEdge
index.html
Recent Files Ctrl+E
localhost:8080
浏览器
index.html
Drop files here to open
后台服务器（Tomcat）
02_CORS
UserServlet
Access-*: http://localhost:63342


Coookie
JSEESIONID=888
domain=localhost:8080
path:/xx
小白的浏览器
localhost:8080/xx/login?username=jj&password=jj
登录成功
响应头：Set-Cookie: JSEESIONID=888
localhost:8080/xx/users
请求头：Cookie: JSEESIO
localhost:8080/xx/login.html
login.html
Session(小白)
id=888
username=jj
password=jj
localhost:8080/xx/login?username=mj&password=mj
小明的浏览器
login.html
登录成功
响应头：Set-Cookie: JSEESIONID=666
服务器
Coookie
JSEESIONID=666
domain=localhost:8080
path:/xx
localhost:8080/xx/users
请求头：Cookie: JSEESIONID=666
Session(小明)
id=666
username=mj
password=mj
users的json数据


100,1
10
100,1
10
login.html
小明的浏览器
Coookie
JSEESIONID=666
domain=localhost:8080
path:/xx
localhost:8080/xx/login?username=mj&password=mj
登录成功
响应头：Set-Cookie: JSEESIONID=666
localhost:8080/xx/addCar?shopid=100
请求头：Cookie: JSEESIONID=666
localhost:8080/xx/addCar?shopid=110
请求头：Cookie: JSEESIONID=666
localhost:8080/xx/getCar
请求头：Cookie: JSEESIONID=666
100,110
Session(小明)
id=666
username=mj
password=mj
userid=990
shopid=100,110
服务器


login.html
小明的浏览器
Coookie
JSEESIONID=666
domain=localhost:8080
path:/xx
localhost:8080/xx/login?username=mj&password=mj
登录成功
响应头：Set-Cookie: JSEESIONID=666
localhost:8080/xx/users
Cookie: JESESSIONID=666
Session(小明)
id=666
username=mj
password=mj
服务器


小明的浏览器
Coookie
JSEESIONID=666
domain=localhost:8080
path:/xx
localhost:8080/cs/users
302
响应头：
Set-Cookie: JSEESIONID=666
Location: /cs/login.html
localhost:8080/cs/users
Cookie: JSEESIONID=666
localhost:8080/cs/login?username=mj&password=mj
Cookie: JSEESIONID=666
Session(小明)
id=999
服务器

状态码（Status Code）

- 在RFC 2616 10.Status Code Definitions规范中定义
  - 状态码指示HTTP请求是否已成功完成

- 状态码可以分为5类
  - 信息响应：100~199
  - 成功响应：200~299
  - 重定向：300~399
  - 客户端错误：400~499
  - 服务器错误：500~599

常见状态码

- 100 Continue
  - 请求的初始部分已经被服务器收到，并且没有被服务器拒绝。客户端应该继续发送剩余的请求，如果请求已经完成，就忽略这个响应
  - 允许客户端发送带请求体的请求前，判断服务器是否愿意接收请求（服务器通过请求头判断）
  - 在某些情况下，如果服务器在不看请求体就拒绝请求时，客户端就发送请求体是不恰当的或低效的
- 200 OK：请求成功
- 302 Found：请求的资源被暂时的移动到了由Location头部指定的URL上
- 304 Not Modified：说明无需再次传输请求的内容，也就是说可以使用缓存的内容

常见状态码

- 400 Bad Request：由于语法无效，服务器无法理解该请求
- 401 Unauthorized：由于缺乏目标资源要求的身份验证凭证
- 403 Forbidden：服务器端有能力处理该请求，但是拒绝授权访问
- 404 Not Found：服务器端无法找到所请求的资源
- 405 Method Not Allowed：服务器禁止了使用当前HTTP方法的请求
- 406 Not Acceptable：服务器端无法提供与Accept-Charset以及Accept-Language指定的值相匹配的响应
- 408 Request Timeout：服务器想要将没有在使用的连接关闭
  - 一些服务器会在空闲连接上发送此信息，即便是在客户端没有发送任何请求的情况下

常见状态码

- 500 Internal Server Error：所请求的服务器遇到意外的情况并阻止其执行请求

- 501 Not Implemented：请求的方法不被服务器支持，因此无法被处理
  - 服务器必须支持的方法（即不会返回这个状态码的方法）只有 GET 和 HEAD

- 502 Bad Gateway：作为网关或代理角色的服务器，从上游服务器（如tomcat）中接收到的响应是无效的

- 503 Service Unavailable：服务器尚未处于可以接受请求的状态
  - 通常造成这种情况的原因是由于服务器停机维护或者已超载

form提交 – 常用属性

- action：请求的URI

- method：请求方法（GET、POST）

- enctype：POST请求时，请求体的编码方式
  - application/x-www-form-urlencoded（默认值）
    - ✓ 用&分隔参数，用=分隔键和值，字符用URL编码方式进行编码
  - multipart/form-data
    - ✓ 文件上传时必须使用这种编码方式

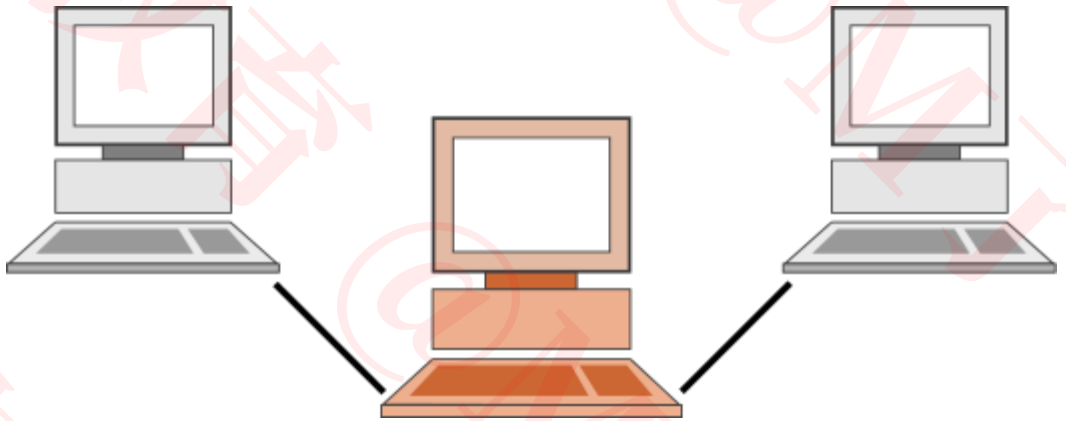
form提交 – multipart/form-data

- 参考RFC 1521
- 请求头
  - Content-Type: multipart/form-data; boundary=xxx

```
multipart-body := preamble 1*encapsulation close-delimiter epilogue

encapsulation := delimiter body-part CRLF

delimiter := "--" boundary CRLF ; taken from Content-Type field.
                                ; There must be no space between "--" and boundary.

close-delimiter := "--" boundary "--" CRLF ; Again, no space by "--"

preamble := discard-text   ;  to  be  ignored upon receipt.
epilogue := discard-text   ;  to  be  ignored upon receipt.
```

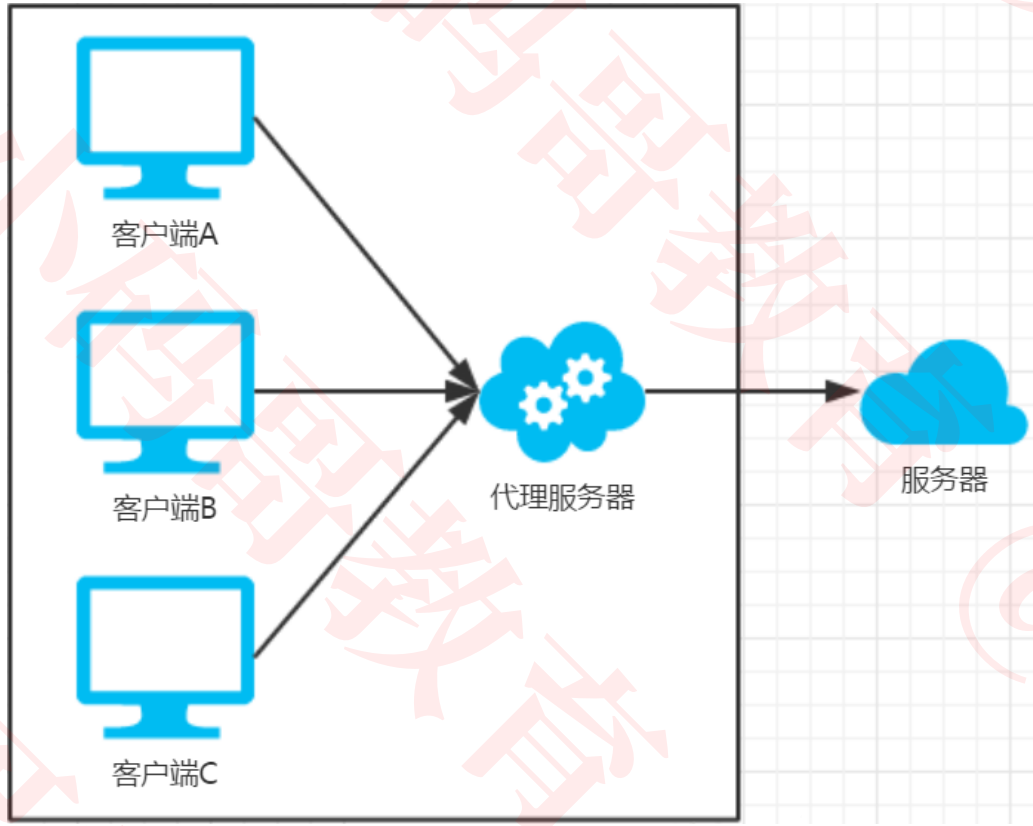
代理服务器（Proxy Server）

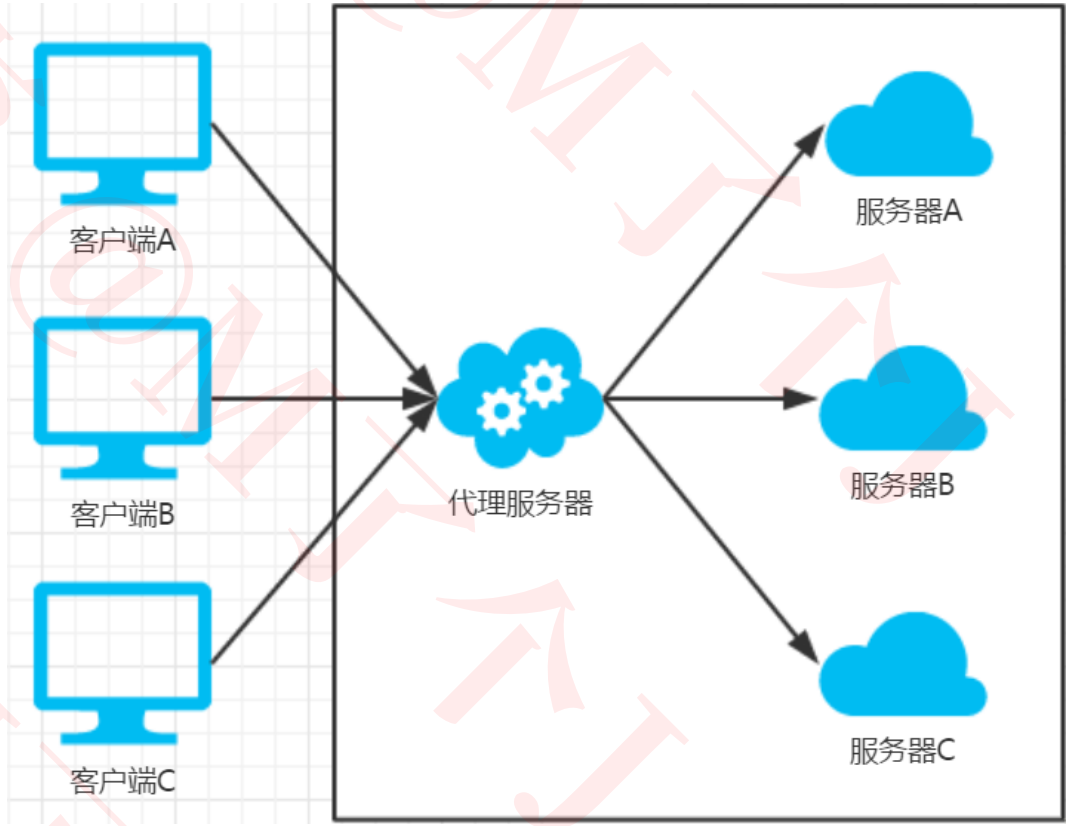
- 特点
  - 本身不生产内容
  - 处于中间位置转发上下游的请求和响应
    - ✓ 面向下游的客户端：它是服务器
    - ✓ 面向上游的服务器：它是客户端



正向代理、反向代理

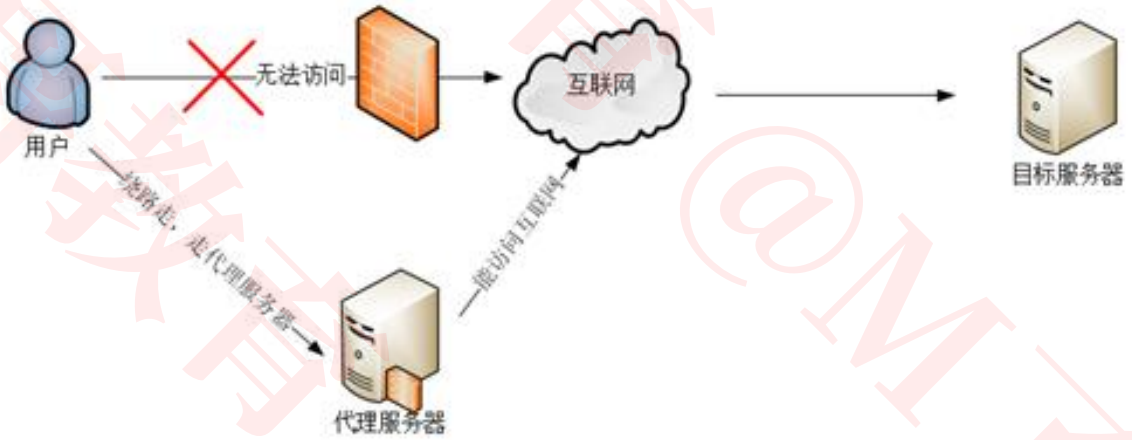
- 正向代理：代理的对象是客户端
- 反向代理：代理的对象是服务器





正向代理 – 作用

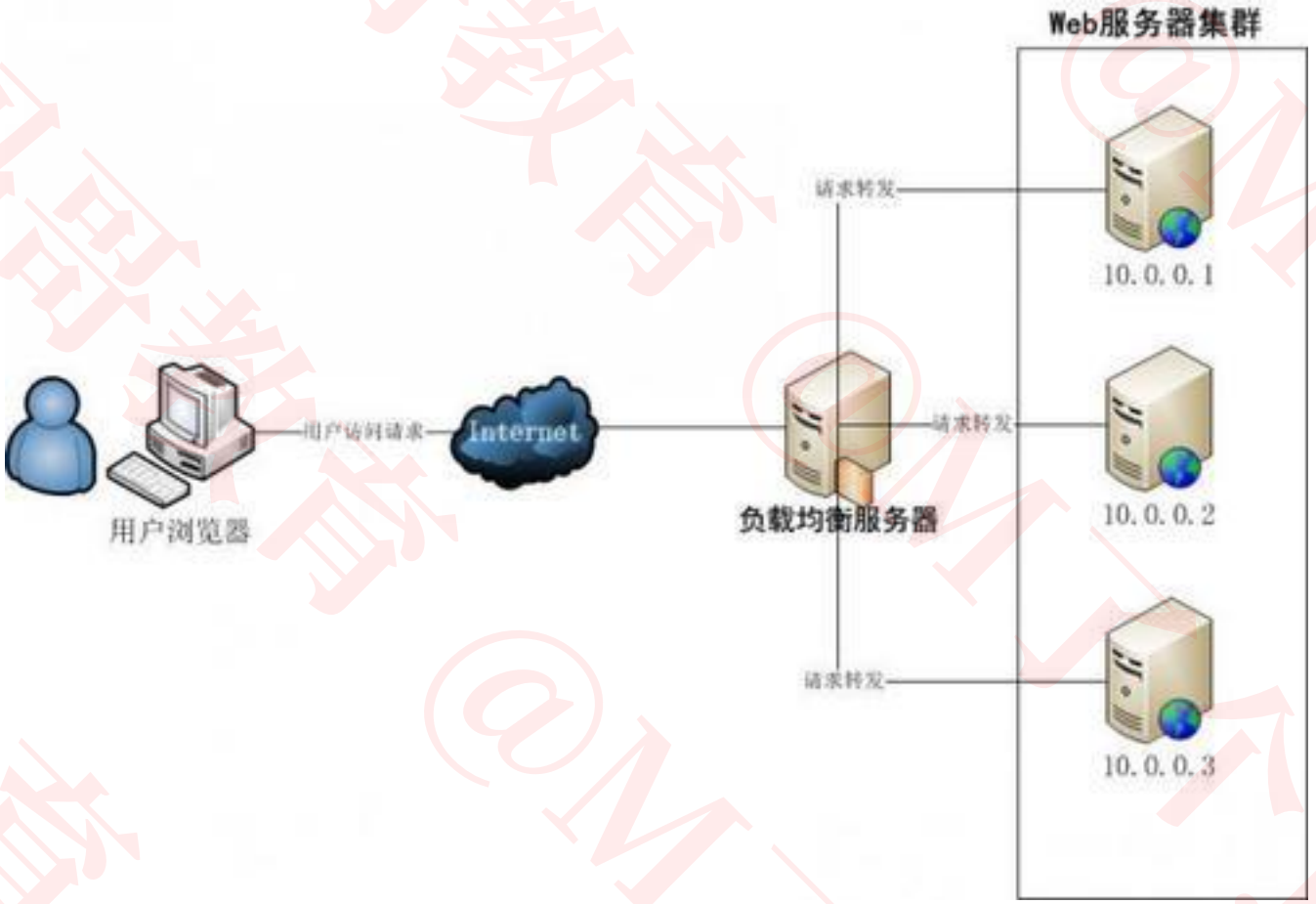
- 隐藏客户端身份
- 绕过防火墙（突破访问限制）
- Internet访问控制
- 数据过滤
- ……



- 一些免费的正向代理
  - https://ip.jiangxianli.com/
  - https://www.kuaidaili.com/free/inha/

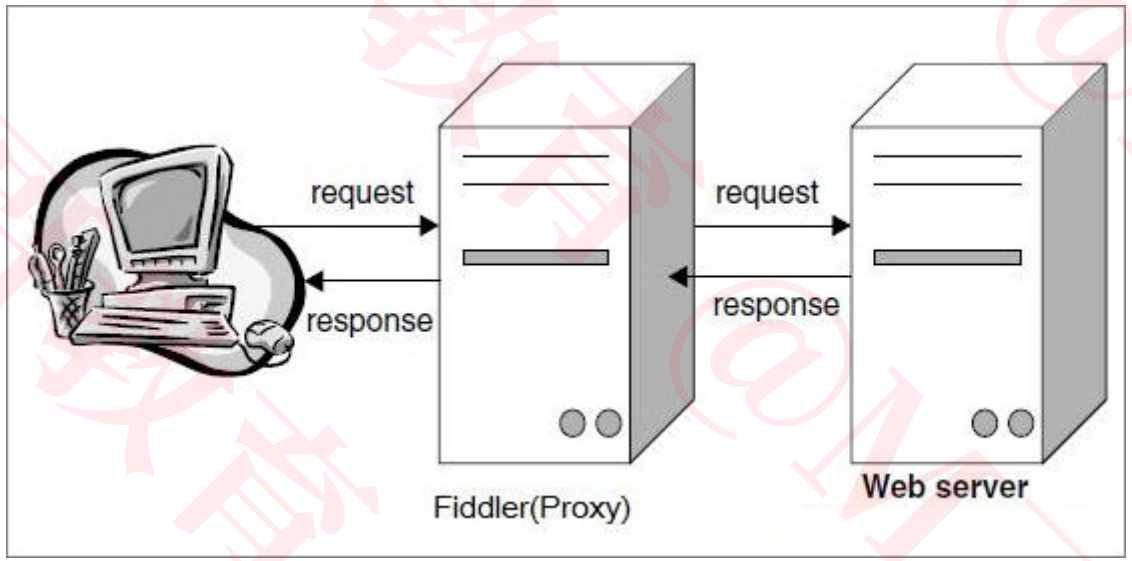
反向代理 – 作用

- 隐藏服务器身份
- 安全防护
- 负载均衡



抓包工具的原理

■ Fiddler、Charles等抓包工具的原理：在客户端启动了正向代理服务



■ 需要注意的是

□ Wireshark的原理是：通过底层驱动，拦截网卡上流过的数据

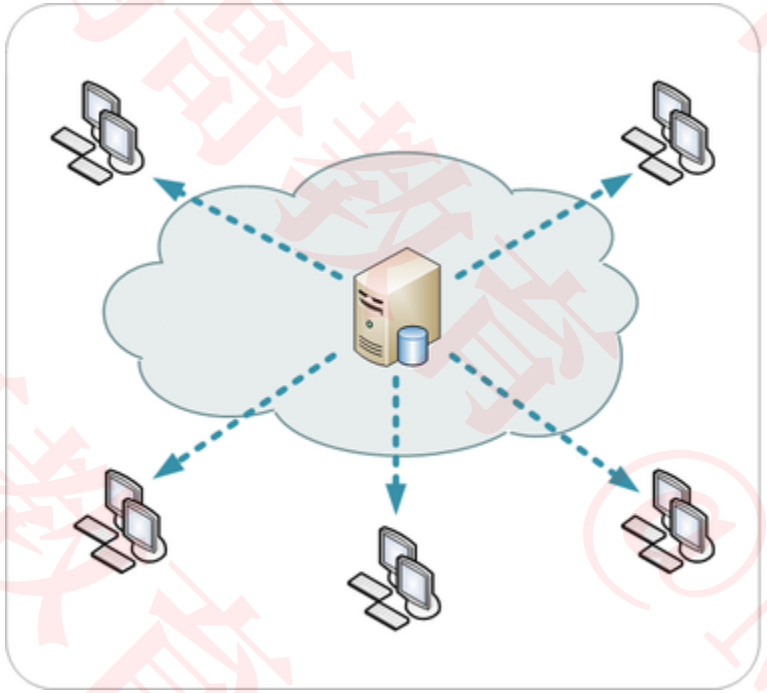
代理服务器 – 相关的头部字段

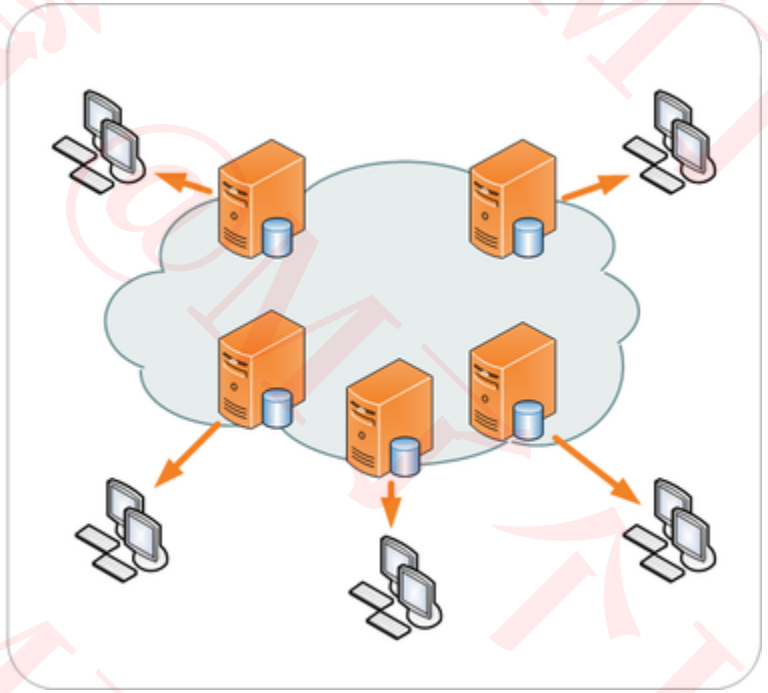
- Via：追加经过的每一台代理服务器的主机名（或域名）
- X-Forwarded-For：追加请求方的IP地址
- X-Real-IP：客户端的真实IP地址

- ①
  - X-Forwarded-For: 14.14.14.14
  - X-Real-IP: 14.14.14.14
  - Via: proxy1
- ②
  - X-Forwarded-For: 14.14.14.14, 220.11.11.11
  - X-Real-IP: 14.14.14.14
  - Via: proxy1, proxy2
- ③
  - Via: proxy2
- ④
  - Via: proxy2, proxy1

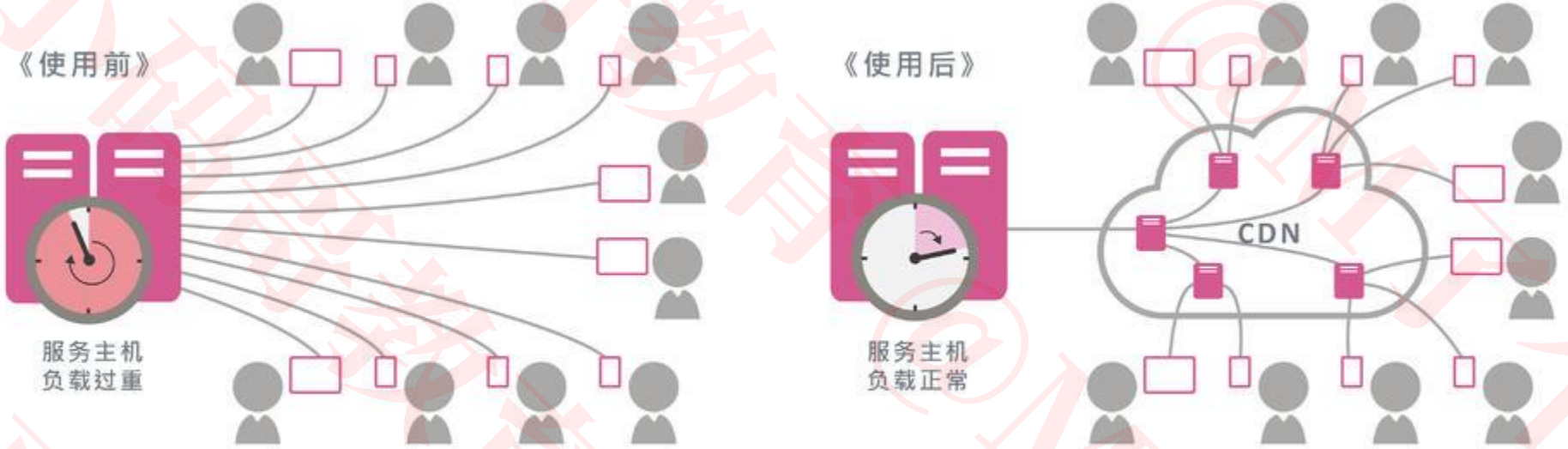
CDN

- CDN（**C**ontent **D**elivery **N**etwork或**C**ontent **D**istribution **N**etwork），译为：内容分发网络
  - 利用最靠近每位用户的服务器
  - 更快更可靠地将音乐、图片、视频等资源文件（一般是静态资源）传递给用户





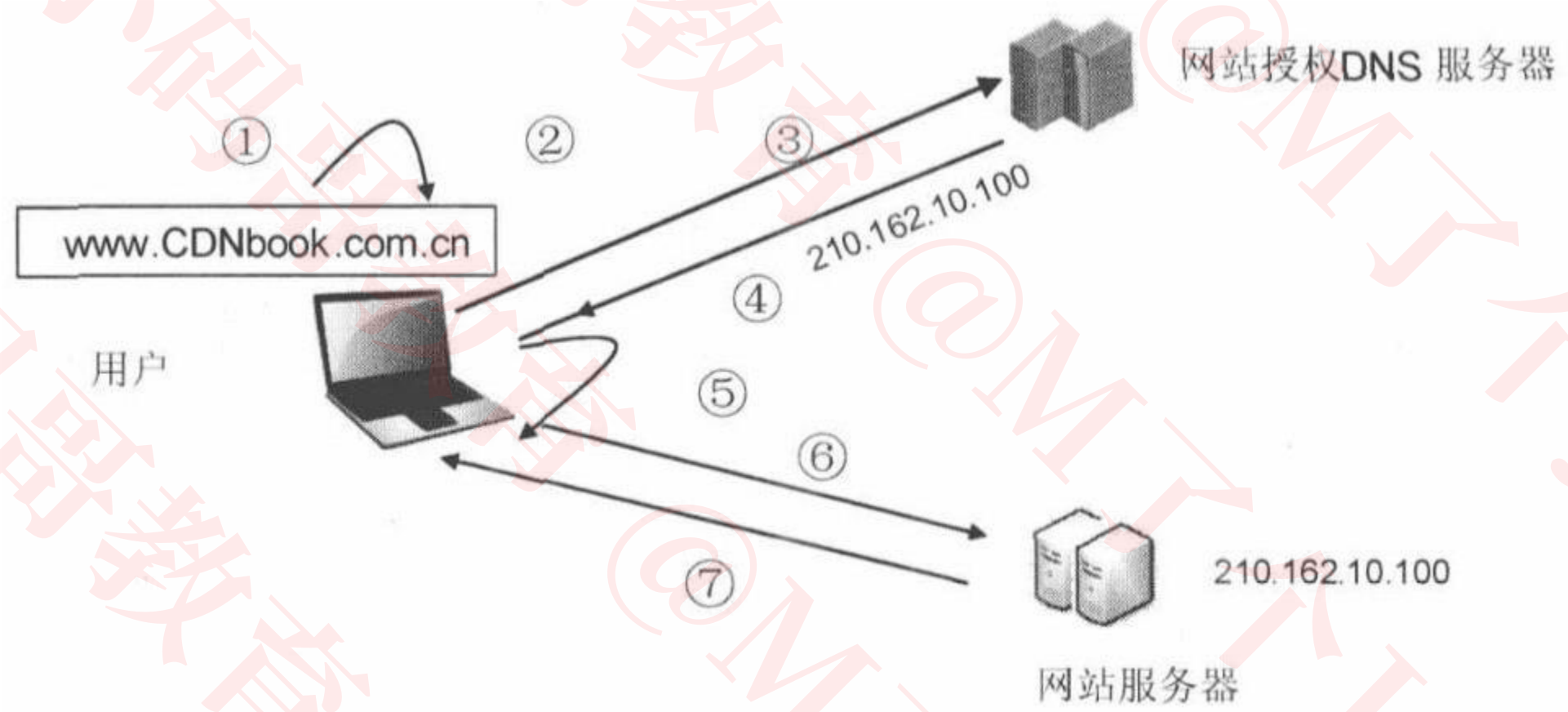
CDN – 使用CDN前后



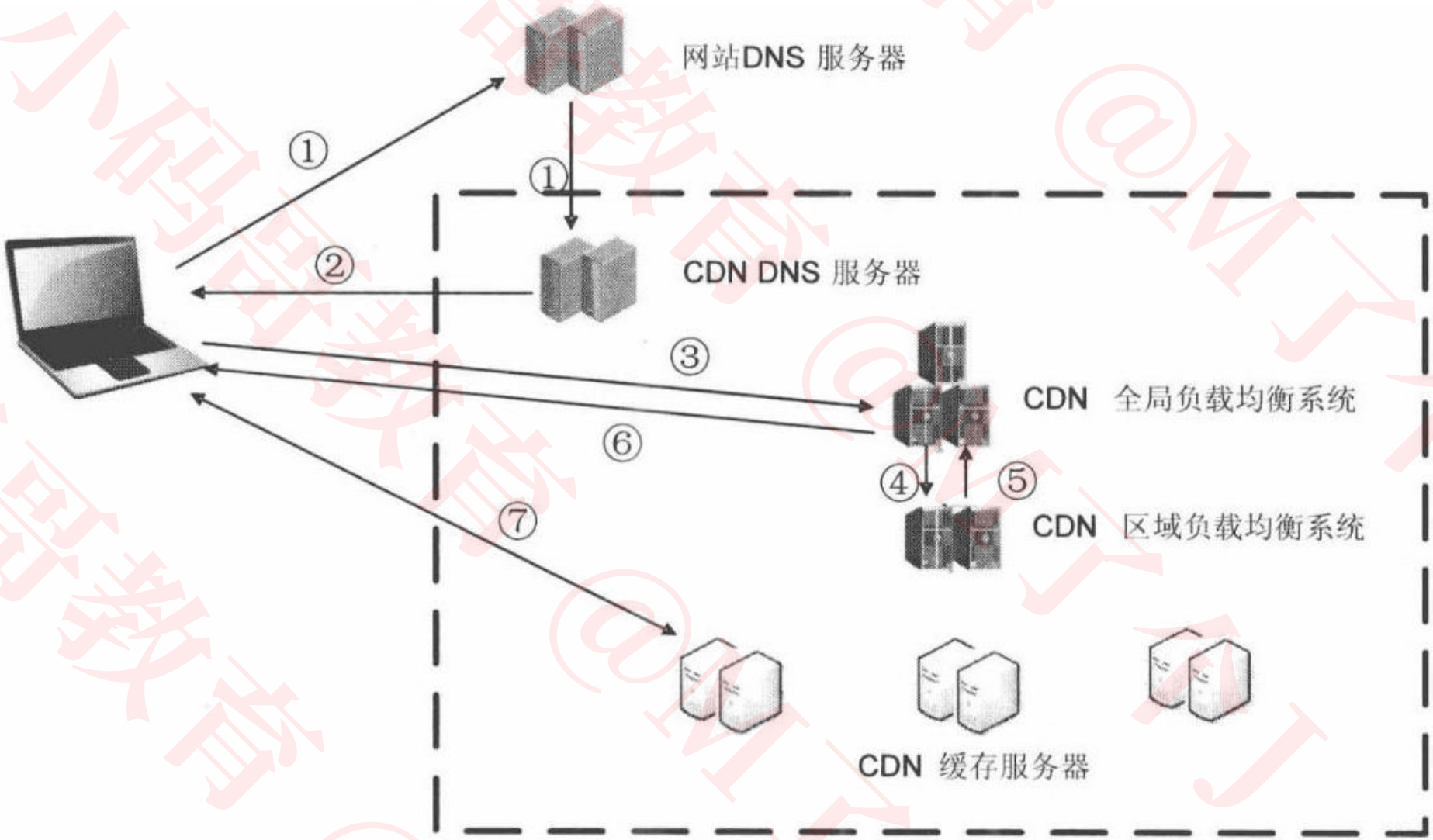
- CDN运营商在全国、乃至全球的各个大枢纽城市都建立了机房
  - 部署了大量拥有高存储高带宽的节点，构建了一个跨运营商、跨地域的专用网络

- 内容所有者向CDN运营商支付费用，CDN将其内容交付给最终用户

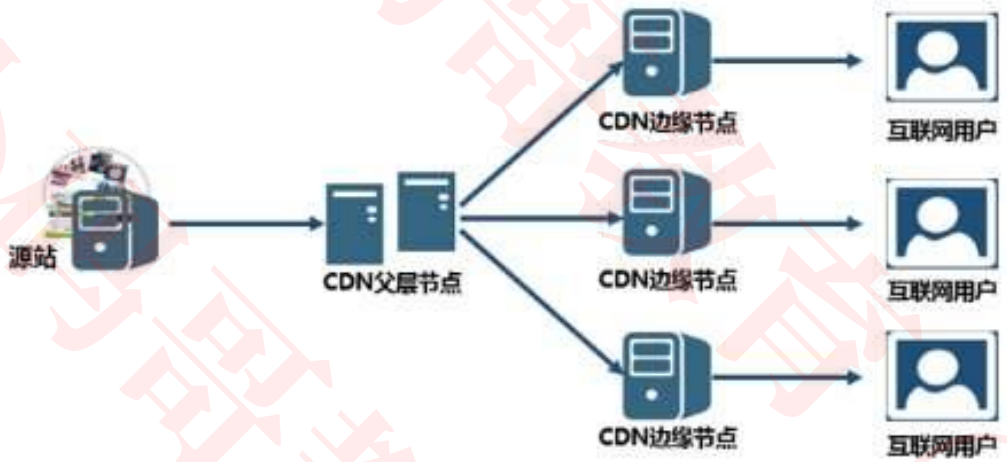
CDN – 使用CDN前

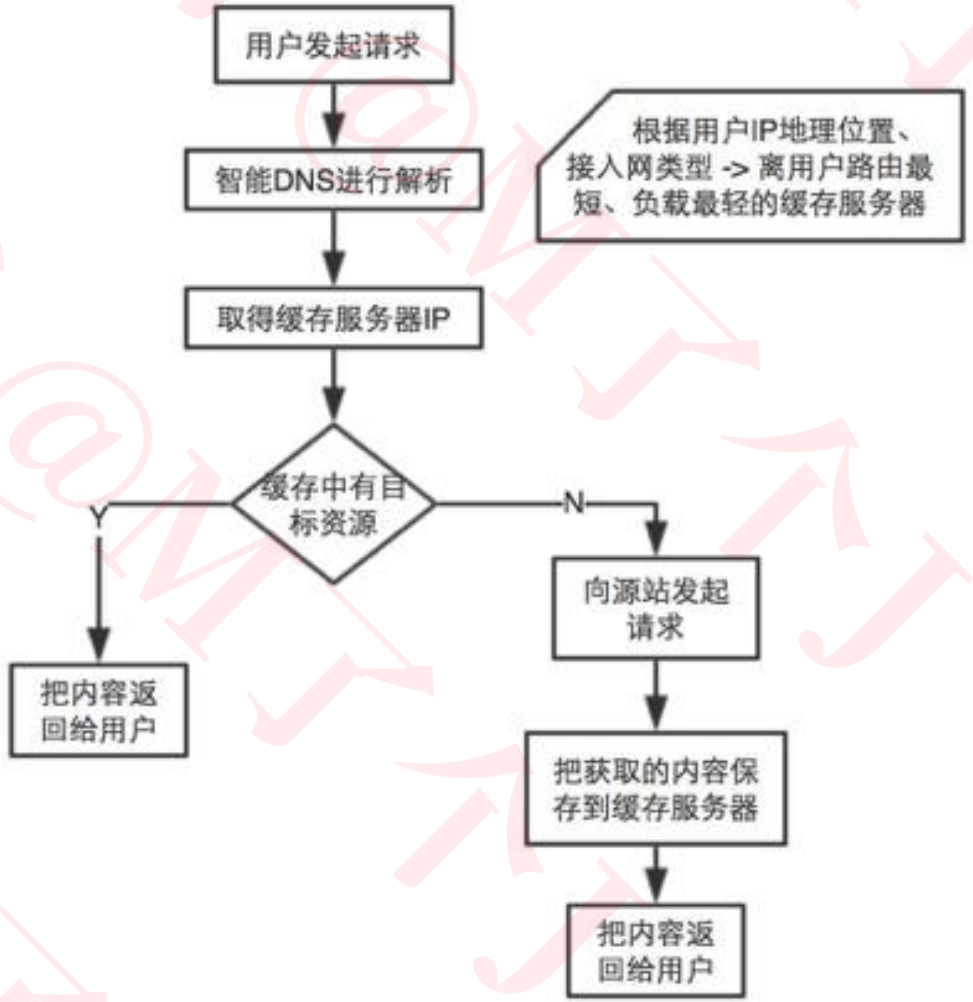

网站授权DNS 服务器
①
②
③
210.162.10.100
④
www.CDNbook.com.cn
用户
⑤
⑥
⑦
210.162.10.100
网站服务器

CDN – 使用CDN后


网站DNS 服务器
①
①
CDN DNS 服务器
②
③
CDN 全局负载均衡系统
⑥
④
⑤
CDN 区域负载均衡系统
⑦
CDN 缓存服务器

CDN – 使用CDN后


源站
CDN父层节点
CDN边缘节点
CDN边缘节点
CDN边缘节点
互联网用户
互联网用户
互联网用户


用户发起请求
智能DNS进行解析
根据用户IP地理位置、接入网类型 -> 离用户路由最短、负载最轻的缓存服务器
取得缓存服务器IP
缓存中有目标资源
Y
N
把内容返回给用户
向源站发起请求
把获取的内容保存到缓存服务器
把内容返回给用户

小码哥教育 SEEMYGO

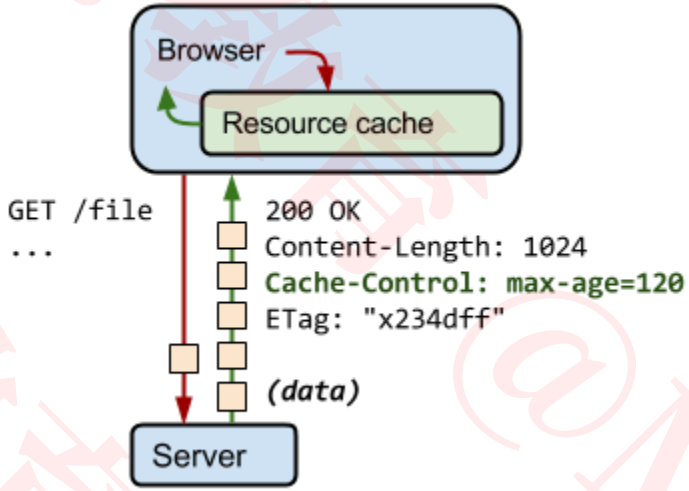
CDN – 使用举例

- 使用cdn引入jquery

```
<script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
    $(() => {
        $(document.body).css('background', '#f00')
    })
</script>
```

缓存（Cache）

- Cache的发音跟Cash（现金）一样： 美 /kæʃ/



- 实际上，HTTP的缓存机制远远比上图的流程要复杂

- 通常会缓存的情况是：GET请求 + 静态资源（比如HTML、CSS、JS、图片等）

- Ctrl + F5：可以强制刷新缓存

缓存 – 响应头

- Pragma：作用类似于Cache-Control，HTTP/1.0的产物

- Expires：缓存的过期时间（GMT格式时间），HTTP/1.0的产物

- Cache-Control：设置缓存策略
  - no-storage：不缓存数据到本地
  - public：允许用户、代理服务器缓存数据到本地
  - private：只允许用户缓存数据到本地
  - max-age：缓存的有效时间（多长时间不过期），单位秒
  - no-cache：每次需要发请求给服务器询问缓存是否有变化，再来决定如何使用缓存

- 优先级：Pragma > Cache-Control > Expires

缓存 – 响应头

- Last-Modified：资源的最后一次修改时间
- ETag：资源的唯一标识（根据文件内容计算出来的摘要值）
- 优先级：ETag > Last-Modified

缓存 – 请求头

- If-None-Match
  - 如果上一次的响应头中**有**ETag，就会将ETag的值作为请求头的值
  - 如果服务器发现资源的最新摘要值跟If-None-Match不匹配，就会返回新的资源（200 OK）
  - 否则，就不会返回资源的具体数据（304 Not Modified）

- If-Modified-Since
  - 如果上一次的响应头中**没有**ETag，有Last-Modified，就会将Last-Modified的值作为请求头的值
  - 如果服务器发现资源的最后一次修改时间晚于If-Modified-Since，就会返回新的资源（200 OK）
  - 否则，就不会返回资源的具体数据（304 Not Modified）

缓存 – Last-Modified vs ETag

- Last-Modified的缺陷
  - 只能精确到秒级别，如果资源在1秒内被修改了，客户端将无法获取最新的资源数据
  - 如果某些资源被修改了（最后一次修改时间发生了变化），但是内容并没有任何变化
  - ✓ 会导致相同数据重复传输，没有使用到缓存

- ETag可以办到
  - 只要资源的内容没有变化，就不会重复传输资源数据
  - 只要资源的内容发生了变化，就会返回最新的资源数据给客户端

小码哥教育 SEEMYGO

缓存的使用流程