15-海纳百川：HTTP的实体数据

你好，我是Chrono。

今天我要与你分享的话题是“海纳百川：HTTP的实体数据”。

这一讲是“进阶篇”的第一讲，从今天开始，我会用连续的8讲的篇幅来详细解析HTTP协议里的各种头字段，包括定义、功能、使用方式、注意事项等等。学完了这些课程，你就可以完全掌握HTTP协议。

在前面的“基础篇”里我们了解了HTTP报文的结构，知道一个HTTP报文是由“header+body”组成的。但那时我们主要研究的是header，没有涉及到body。所以，“进阶篇”的第一讲就从HTTP的body谈起。

数据类型与编码

在TCP/IP协议栈里，传输数据基本上都是“header+body”的格式。但TCP、UDP因为是传输层的协议，它们不会关心body数据是什么，只要把数据发送到对方就算是完成了任务。

而HTTP协议则不同，它是应用层的协议，数据到达之后工作只能说是完成了一半，还必须要告诉上层应用这是什么数据才行，否则上层应用就会“不知所措”。

你可以设想一下，假如HTTP没有告知数据类型的功能，服务器把“一大坨”数据发给了浏览器，浏览器看到的是一个“黑盒子”，这时候该怎么办呢？

当然，它可以“猜”。因为很多数据都是有固定格式的，所以通过检查数据的前几个字节也许就能知道这是个GIF图片、或者是个MP3音乐文件，但这种方式无疑十分低效，而且有很大几率会检查不出来文件类型。

幸运的是，早在HTTP协议诞生之前就已经有了针对这种问题的解决方案，不过它是用在电子邮件系统里的，让电子邮件可以发送ASCII码以外的任意数据，方案的名字叫做“**多用途互联网邮件扩展**”（Multipurpose Internet Mail Extensions），简称为MIME。

MIME是一个很大的标准规范，但HTTP只“顺手牵羊”取了其中的一部分，用来标记body的数据类型，这就是我们平常总能听到的“**MIME type**”。

MIME把数据分成了八大类，每个大类下再细分出多个子类，形式是“type/subtype”的字符串，巧得很，刚好也符合了HTTP明文的特点，所以能够很容易地纳入HTTP头字段里。

这里简单列举一下在HTTP里经常遇到的几个类别：

1. text：即文本格式的可读数据，我们最熟悉的应该就是text/html了，表示超文本文档，此外还有纯文本text/plain、样式表text/css等。
2. image：即图像文件，有image/gif、image/jpeg、image/png等。
3. audio/video：音频和视频数据，例如audio/mpeg、video/mp4等。
4. application：数据格式不固定，可能是文本也可能是二进制，必须由上层应用程序来解释。常见的有application/json，application/javascript、application/pdf等，另外，如果实在是不知道数据是什么类型，像刚才说的“黑盒”，就会是application/octet-stream，即不透明的二进制数据。

但仅有MIME type还不够，因为HTTP在传输时为了节约带宽，有时候还会压缩数据，为了不要让浏览器继

续“猜”，还需要有一个“Encoding type”，告诉数据是用的什么编码格式，这样对方才能正确解压缩，还原出原始的数据。

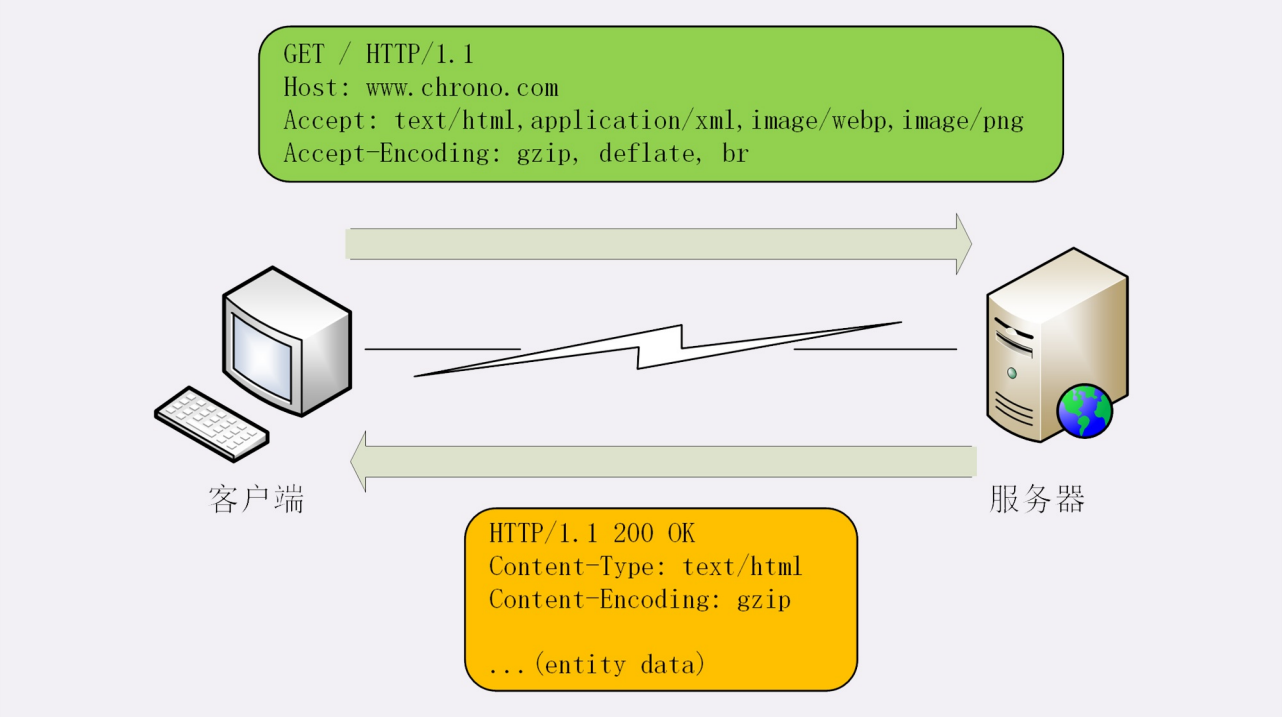
比起MIME type来说，Encoding type就少了很多，常用的只有下面三种：

1. gzip：GNU zip压缩格式，也是互联网上最流行的压缩格式；
2. deflate：zlib（deflate）压缩格式，流行程度仅次于gzip；
3. br：一种专门为HTTP优化的新压缩算法（Brotli）。

数据类型使用的头字段

有了MIME type和Encoding type，无论是浏览器还是服务器就都可以轻松识别出body的类型，也就能够正确处理数据了。

HTTP协议为此定义了两个Accept请求头字段和两个Content实体头字段，用于客户端和服务器进行“**内容协商**”。也就是说，客户端用Accept头告诉服务器希望接收什么样的数据，而服务器用Content头告诉客户端实际发送了什么样的数据。



Accept字段标记的是客户端可理解的MIME type，可以用“,”做分隔符列出多个类型，让服务器有更多的选择余地，例如下面的这个头：

```
Accept: text/html,application/xml,image/webp,image/png
```

这就是告诉服务器：“我能够看懂HTML、XML的文本，还有webp和png的图片，请给我这四类格式的数据”。

相应的，服务器会在响应报文里用头字段**Content-Type**告诉实体数据的真实类型：

```
Content-Type: text/html
Content-Type: image/png
```

这样浏览器看到报文里的类型是“text/html”就知道是HTML文件，会调用排版引擎渲染出页面，看到“image/png”就知道是一个PNG文件，就会在页面上显示出图像。

Accept-Encoding字段标记的是客户端支持的压缩格式，例如上面说的gzip、deflate等，同样也可以用“,”列出多个，服务器可以选择其中一种来压缩数据，实际使用的压缩格式放在响应头字段**Content-Encoding**里。

```
Accept-Encoding: gzip, deflate, br
Content-Encoding: gzip
```

不过这两个字段是可以省略的，如果请求报文里没有Accept-Encoding字段，就表示客户端不支持压缩数据；如果响应报文里没有Content-Encoding字段，就表示响应数据没有被压缩。

语言类型与编码

MIME type和Encoding type解决了计算机理解body数据的问题，但互联网遍布全球，不同国家不同地区的人使用了很多不同的语言，虽然都是text/html，但如何让浏览器显示出每个人都可理解可阅读的语言文字呢？

这实际上就是“国际化”的问题。HTTP采用了与数据类型相似的解决方案，又引入了两个概念：语言类型与字符集。

所谓的“**语言类型**”就是人类使用的自然语言，例如英语、汉语、日语等，而这些自然语言可能还有下属的地区性方言，所以在需要明确区分的时候也要使用“type-subtype”的形式，不过这里的格式与数据类型不同，**分隔符不是“/”，而是“-”**。

举几个例子：en表示任意的英语，en-US表示美式英语，en-GB表示英式英语，而zh-CN就表示我们最常使用的汉语。

关于自然语言的计算机处理还有一个更麻烦的东西叫做“字符集”。

在计算机发展的早期，各个国家和地区的人们“各自为政”，发明了许多字符编码方式来处理文字，比如英语世界用的ASCII、汉语世界用的GBK、BIG5，日语世界用的Shift_JIS等。同样的一段文字，用一种编码显示正常，换另一种编码后可能就会变得一团糟。

所以后来就出现了Unicode和UTF-8，把世界上所有的语言都容纳在一种编码方案里，UTF-8也成为了互联网上的标准字符集。

语言类型使用的头字段

同样的，HTTP协议也使用Accept请求头字段和Content实体头字段，用于客户端和服务器就语言与编码进行“**内容协商**”。

Accept-Language字段标记了客户端可理解的自然语言，也允许用“,”做分隔符列出多个类型，例如：

```
Accept-Language: zh-CN, zh, en
```

这个请求头会告诉服务器：“最好给我zh-CN的汉语文字，如果没有就用其他的汉语方言，如果还没有就给英文”。

相应的，服务器应该在响应报文里用头字段**Content-Language**告诉客户端实体数据使用的实际语言类型：

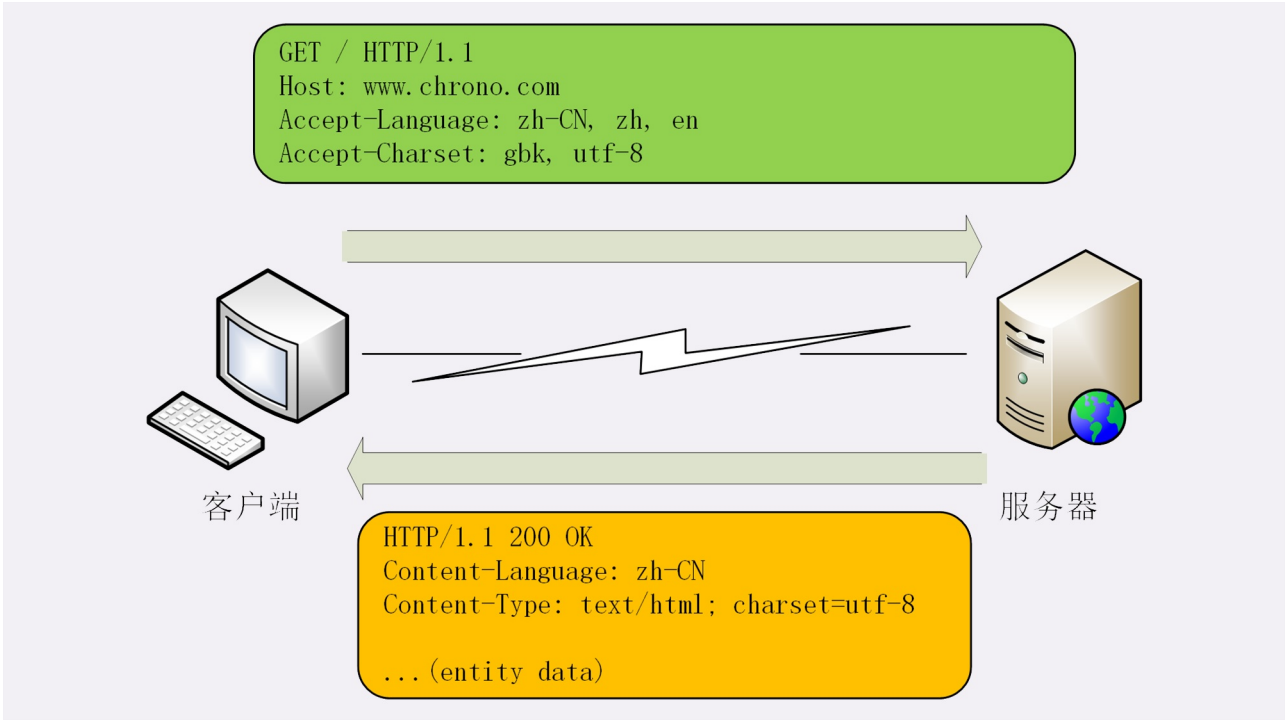
```
Content-Language: zh-CN
```

字符集在HTTP里使用的请求头字段是**Accept-Charset**，但响应头里却没有对应的Content-Charset，而是在**Content-Type**字段的数据类型后面用“charset=xxx”来表示，这点需要特别注意。

例如，浏览器请求GBK或UTF-8的字符集，然后服务器返回的是UTF-8编码，就是下面这样：

```
Accept-Charset: gbk, utf-8
Content-Type: text/html; charset=utf-8
```

不过现在的浏览器都支持多种字符集，通常不会发送Accept-Charset，而服务器也不会发送Content-Language，因为使用的语言完全可以由字符集推断出来，所以在请求头里一般只会有Accept-Language字段，响应头里只会有Content-Type字段。



内容协商的质量值

在HTTP协议里用Accept、Accept-Encoding、Accept-Language等请求头字段进行内容协商的时候，还可以用一种特殊的“q”参数表示权重来设定优先级，这里的“q”是“quality factor”的意思。

权重的最大值是1，最小值是0.01，默认值是1，如果值是0就表示拒绝。具体的形式是在数据类型或语言代码后面加一个“;”，然后是“q=value”。

这里要提醒的是“;”的用法，在大多数编程语言里“;”的断句语气要强于“,”，而在HTTP的内容协商里却恰好反了过来，“;”的意义是小于“,”的。

例如下面的Accept字段：

```
Accept: text/html,application/xml;q=0.9,*/*;q=0.8
```

它表示浏览器最希望使用的是HTML文件，权重是1，其次是XML文件，权重是0.9，最后是任意数据类型，权重是0.8。服务器收到请求头后，就会计算权重，再根据自己的实际情况优先输出HTML或者XML。

内容协商的结果

内容协商的过程是不透明的，每个Web服务器使用的算法都不一样。但有的时候，服务器会在响应头里多加一个**Vary**字段，记录服务器在内容协商时参考的请求头字段，给出一点信息，例如：

```
Vary: Accept-Encoding,User-Agent,Accept
```

这个Vary字段表示服务器依据了Accept-Encoding、User-Agent和Accept这三个头字段，然后决定了发回的

响应报文。

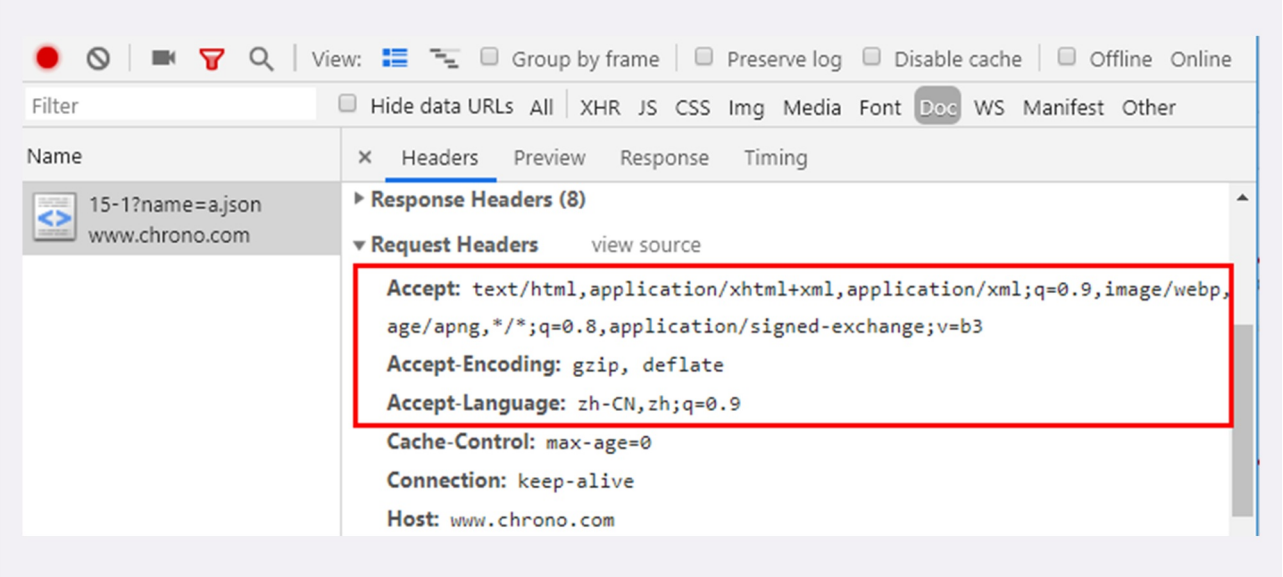
Vary字段可以认为是响应报文的一个特殊的“版本标记”。每当Accept等请求头变化时，Vary也会随着响应报文一起变化。也就是说，同一个URI可能会有多个不同的“版本”，主要用在传输链路中间的代理服务器实现缓存服务，这个之后讲“HTTP缓存”时还会再提到。

动手实验

上面讲完了理论部分，接下来就是实际动手操作了。可以用我们的实验环境，在www目录下有一个mime目录，里面预先存放了几个文件，可以用URI“/15-1?name=file”的形式访问，例如：

```
http://www.chrono.com/15-1?name=a.json
http://www.chrono.com/15-1?name=a.xml
```

在Chrome里打开开发者工具，就能够看到Accept和Content头：

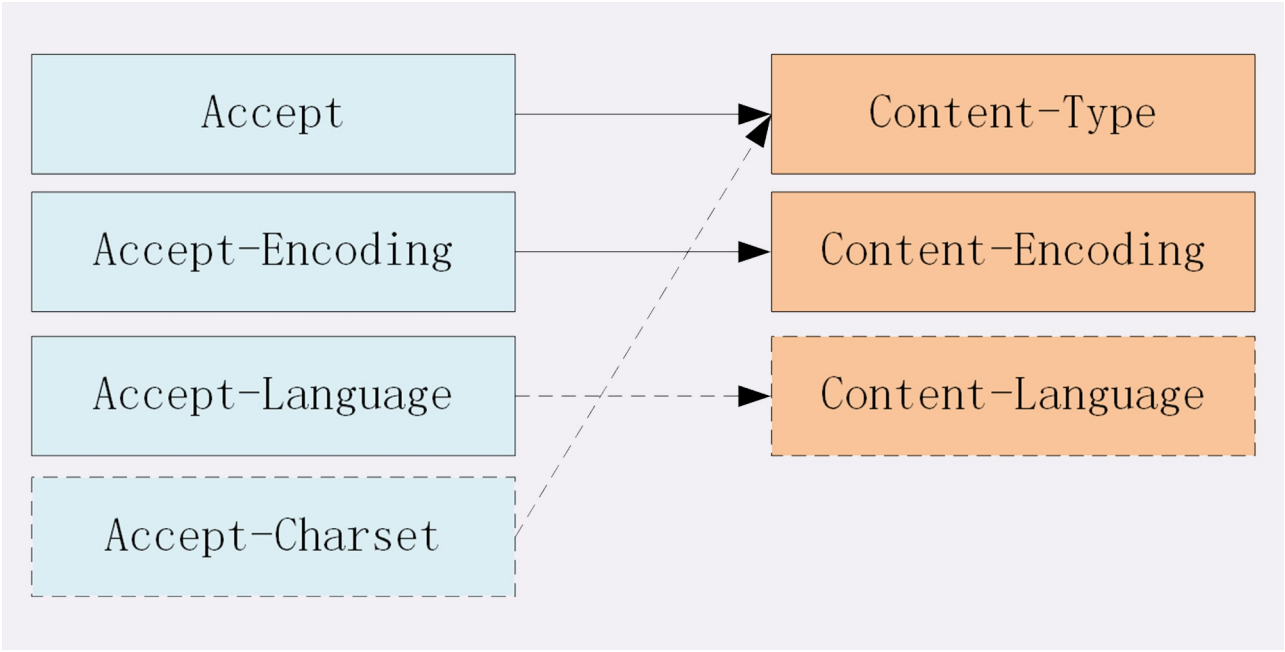


你也可以把任意的文件拷贝到mime目录下，比如压缩包、MP3、图片、视频等，再用Chrome访问，观察更多的MIME type。

有了这些经验后，你还可以离开实验环境，直接访问各大门户网站，看看真实网络世界里的HTTP报文是什么样子的。

小结

今天我们学习了HTTP里的数据类型和语言类型，在这里为今天的内容做个小结。



1. 数据类型表示实体数据的内容是什么，使用的是MIME type，相关的头字段是Accept和Content-Type；
2. 数据编码表示实体数据的压缩方式，相关的头字段是Accept-Encoding和Content-Encoding；
3. 语言类型表示实体数据的自然语言，相关的头字段是Accept-Language和Content-Language；
4. 字符集表示实体数据的编码方式，相关的头字段是Accept-Charset和Content-Type；
5. 客户端需要在请求头里使用Accept等头字段与服务器进行“内容协商”，要求服务器返回最合适的数据；
6. Accept等头字段可以用“,”顺序列出多个可能的选项，还可以用“;q=”参数来精确指定权重。

课下作业

1. 试着解释一下这个请求头“Accept-Encoding: gzip, deflate;q=1.0, *;q=0.5, br;q=0”，再模拟一下服务器的响应头。
2. 假设你要使用POST方法向服务器提交一些JSON格式的数据，里面包含有中文，请求头应该是什么样子的呢？
3. 试着用快递发货收货比喻一下MIME、Encoding等概念。

欢迎你把自己的答案写在留言区，与我和其他同学一起讨论。如果你觉得有所收获，欢迎你把文章分享给你的朋友。

精选留言：

- BellenHsin 2019-07-01 08:46:47
  这篇写的不错 [5赞]

  作者回复2019-07-01 09:00:53
  thanks。

- 彧豪 2019-07-01 13:04:36
  上周五和服务端做上传图片的时候遇到过这个content-type的问题，上传图片时候我这边需要设置content-type:"image/jpg"，然后传完了，我在预览的时候获取图片的地址，此时比如通过a标签的方式打开新标签预览该图片时才能成功预览，不然如果使用上传的js-sdk设置的默认类型：content-type:"octet-stream"，那么浏览器就不认识这个图片了，转而会下载这个文件(图片)，所以我是不是可以理解为content-type这字段在请求头，和响应头里都能使用？或者上传文件这个业务又不同于一般的请求操作呢？ [2赞]

  作者回复2019-07-01 15:19:29
  是的，看来是我没说清楚，导致有的同学误会了。

  content-type是实体字段，所以请求和响应里都可以用，作用是指明body数据的类型。

  正文里为了叙述方便，只在服务器的响应报文里出现了content-type，实际上它是个通用字段，如果要发post请求，就需要带上它。

- 苦行僧 2019-07-02 07:19:44
  现在很多小文件 比如图片 都往云存上放了 千万指定正确content-type 一旦指定错 批量修改太麻烦 而且会影响终端的解析 [1赞]

- Geek_f91853 2019-07-01 16:51:19
  1.含义是：我这个请求最希望服务器给我返回的编码方式是gzip和deflate，他们俩在我这是最优的地位，我不接受br的编码方式，如果还有其他的编码方式的话对我来说权重0.5。服务器可能的响应头是
  HTTP/1.1 200 OK
  Content-Encoding: gzip

2.请求头可能是
POST /serv/v1/user/auth HTTP/1.1
Content-Type: application/json
Accept-Language: zh-CN, zh
Accept-Charset: gbk, utf-8
3.MIME类比快递的话就是你要快递的物品（衣服，食物等），Encoding就是快递你这个物品的包装方式，如果是衣服可能就随意一点一个袋子，如果是食物担心腐烂给你放个冰袋进去
不知道回答的对不对，请老师指正 [1赞]

- Aviation 2019-07-01 11:12:58
这里面的, 优先级高于; 第一次理解了. 还有q值. 哈哈 [1赞]

作者回复2019-07-01 11:38:12
有收获就是好事。

- Geek_54edc1 2019-07-01 09:05:06
1.服务器优先按照gzip和deflate压缩，否则用其他压缩算法，但是不用brotli算法 [1赞]

作者回复2019-07-01 10:01:20
√

- 苦行僧 2019-07-02 07:16:38
content-type 千万不能填错 否则其他终端解析会存在问题

- レイン小雨 2019-07-01 23:57:25
真棒

- 1900 2019-07-01 17:14:19
“所以后来就出现了 Unicode 和 UTF-8，把世界上所有的语言都容纳在一种编码方案里，UTF-8 也成为了互联网上的标准字符集。”

这句话最后有点问题吧？Unicode才是字符集，应该是“遵循UTF-8字符编码方式的Unicode字符集也成为了互联网上的标准字符集”，是么？

作者回复2019-07-01 18:22:11
嗯，我说的时候不太准确。utf-8只是编码方案，Unicode是字符集。

- Geek_54edc1 2019-07-01 12:58:12
content-type: application/json; charset=gbk 如果有压缩用content-encoding指定下，使用的语言可以通过charset判断出来

作者回复2019-07-01 15:17:23
对，不过最好还是加上content-language。

- Geek_54edc1 2019-07-01 09:10:46
2. accept language: zh-CN accept: application/ json post json数据一般会有压缩，因此accept encoding: gzip

作者回复2019-07-01 10:03:17
不应该用accept头，而是应该用content-*头，因为accept是“希望”服务器返回什么样的数据，问题里

是客户端发出的数据，要告诉服务器是什么样的数据。

试着再改一下。

- -W.LI- 2019-07-01 09:08:26
  老师好!那accept是不是有两个语意
  1.客户端希望接受(支持)的数据类型
  2.我发送的数据就是这个类型的。请用这些方式解析?
  问题:accept指定text。实际传的数据是一个json这样的后台会用text解析。然后拿不到数据是么?在请求头里加content -type这些字段会起作用么?

  作者回复2019-07-01 10:00:48
  1.accept是你说的第一个意思，没有第二个意思。

  2.第二个意思应该用Content-Type

  3.看后台逻辑如何处理，数据是肯定可以拿到的，而且json也属于text。

  4.在请求头里可以加content-type字段，表示请求体的数据类型。

- -W.LI- 2019-07-01 08:57:29
  老师好!有个问题,之前遇到过一个发送ajax请求。前端忘记在content-type里面指定，application/json。后端接受数据失败。具体表现不太记得了好像都是null。后来前端加了content-type就好了。accept比较好理解就是发起请求放想要接受的内容。content-type是服务器，是响应类型的话。客户端在发送请求时压根就不知道啊，也不应该由客户端来设置。
  所以我想问的是，accept相关的都是请求头里面的数据
  content-type相关的都是响应头里的数据么?
  至于我前面正确的写法应该是在accept里面设置json类型。错写了content-type。框架做了兼容处理(在服务端看起来content-type起作用了)?
  谢谢老师

  作者回复2019-07-01 09:09:39
  客户端在发送请求的时候也有义务设置content-type，也应该是知道数据是什么类型的，你设置成json，服务器看到了就好处理。

  content-type是实体字段，请求响应里都可以出现。

  accept是告诉服务器，客户端支持什么类型，防止服务器发过来的数据不认识。