

## 13 | HTTP有哪些特点？

2019-06-26 Chrono

《透视HTTP协议》

课程介绍 >



讲述：Chrono

时长 09:03 大小 12.45M



通过“基础篇”前几讲的学习，你应该已经知道了 HTTP 协议的基本知识，了解它的报文结构，请求头、响应头以及内部的请求方法、URI 和状态码等细节。

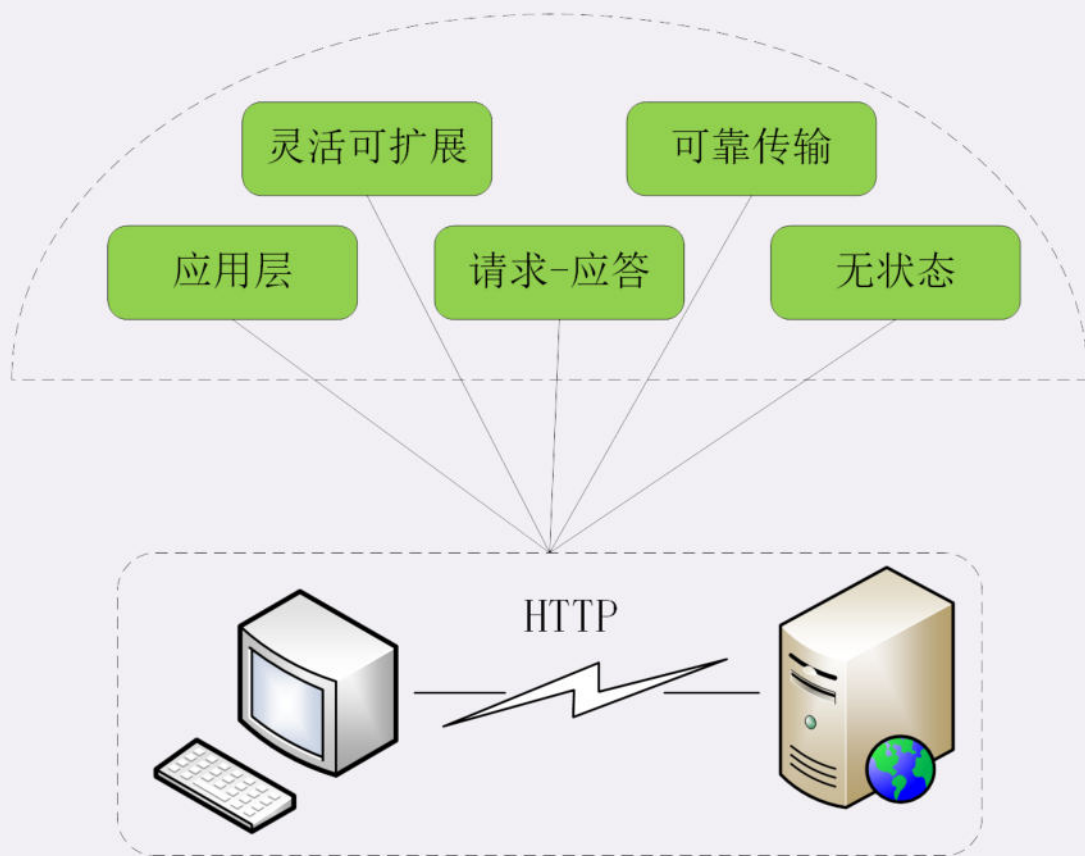
你会不会有种疑惑：“HTTP 协议好像也挺简单的啊，凭什么它就能统治互联网这么多年呢？”

所以接下来的这两讲，我会跟你聊聊 HTTP 协议的特点、优点和缺点。既要看到它好的一面，也要正视它不好的一面，只有全方位、多角度了解 HTTP，才能实现“扬长避短”，更好地利用 HTTP。

领资料

今天这节课主要说的是 HTTP 协议的特点，但不会讲它们的好坏，这些特点即有可能是优点，也有可能是缺点，你可以边听边思考。





## 灵活可扩展

首先，HTTP 协议是一个“灵活可扩展”的传输协议。

HTTP 协议最初诞生的时候就比较简单，本着开放的精神只规定了报文的基本格式，比如用空格分隔单词，用换行分隔字段，“header+body”等，报文里的各个组成部分都没有做严格的语法语义限制，可以由开发者任意定制。

所以，HTTP 协议就随着互联网的发展一同成长起来了。在这个过程中，HTTP 协议逐渐增加了请求方法、版本号、状态码、头字段等特性。而 body 也不再限于文本形式的 TXT 或 HTML，而是能够传输图片、音频视频等任意数据，这些都是源于它的“灵活可扩展”的特点。

而那些 RFC 文档，实际上也可以理解为是对已有扩展的“承认和标准化”，实现了“从实践中来，到实践中去”的良性循环。

也正是因为这个特点，HTTP 才能在三十年的历史长河中“屹立不倒”，从最初的低速实验网络发展到现在的遍布全球的高速互联网，始终保持着旺盛的生命力。

领资料



## 可靠传输

第二个特点，HTTP 协议是一个“可靠”的传输协议。

这个特点显而易见，因为 HTTP 协议是基于 TCP/IP 的，而 TCP 本身是一个“可靠”的传输协议，所以 HTTP 自然也就继承了这个特性，能够在请求方和应答方之间“可靠”地传输数据。

它的具体做法与 TCP/UDP 差不多，都是对实际传输的数据（entity）做了一层包装，加上一个头，然后调用 Socket API，通过 TCP/IP 协议栈发送或者接收。

不过我们必须正确地理解“可靠”的含义，HTTP 并不能 100% 保证数据一定能够发送到另一端，在网络繁忙、连接质量差等恶劣的环境下，也有可能收发失败。“可靠”只是向使用者提供了一个“承诺”，会在下层用多种手段“尽量”保证数据的完整送达。

当然，如果遇到光纤被意外挖断这样的极端情况，即使是神仙也不能发送成功。所以，“可靠”传输是指在网络基本正常的情况下数据收发必定成功，借用运维里的术语，大概就是“3 个 9”或者“4 个 9”的程度吧。

## 应用层协议

第三个特点，HTTP 协议是一个应用层的协议。

这个特点也是不言自明的，但却很重要。

在 TCP/IP 诞生后的几十年里，虽然出现了许多的应用层协议，但它们都仅关注很小的应用领域，局限在很少的应用场景。例如 FTP 只能传输文件、SMTP 只能发送邮件、SSH 只能远程登录等，在通用的数据传输方面“完全不能打”。

所以 HTTP 凭借着可携带任意头字段和实体数据的报文结构，以及连接控制、缓存代理等方便易用的特性，一出现就“技压群雄”，迅速成为了应用层里的“明星”协议。只要不太苛求性能，HTTP 几乎可以传递一切东西，满足各种需求，称得上是一个“万能”的协议。

套用一个网上流行的段子，HTTP 完全可以用开玩笑的口吻说：“不要误会，我不是针对 FTP，我是说在座的应用层各位，都是垃圾。”

## 请求 – 应答

领资料



第四个特点，HTTP 协议使用的是请求 – 应答通信模式。

这个请求 – 应答模式是 HTTP 协议最根本的通信模型，通俗来讲就是“一发一收”“有来有去”，就像是写代码时的函数调用，只要填好请求头里的字段，“调用”后就会收到答复。

请求 – 应答模式也明确了 HTTP 协议里通信双方的定位，永远是请求方先发起连接和请求，是主动的，而应答方只有在收到请求后才能答复，是被动的，如果没有请求时不会有任何动作。

当然，请求方和应答方的角色也不是绝对的，在浏览器 – 服务器的场景里，通常服务器都是应答方，但如果将它用作代理连接后端服务器，那么它就可能同时扮演请求方和应答方的角色。

HTTP 的请求 – 应答模式也恰好契合了传统的 C/S (Client/Server) 系统架构，请求方作为客户端、应答方作为服务器。所以，随着互联网的发展就出现了 B/S (Browser/Server) 架构，用轻量级的浏览器代替笨重的客户端应用，实现零维护的“瘦”客户端，而服务器则摒弃私有通信协议转而使用 HTTP 协议。

此外，请求 – 应答模式也完全符合 RPC (Remote Procedure Call) 的工作模式，可以把 HTTP 请求处理封装成远程函数调用，导致了 Webservice、RESTful 和 gRPC 等的出现。

## 无状态

第五个特点，HTTP 协议是无状态的。

这个所谓的“状态”应该怎么理解呢？

“状态”其实就是客户端或者服务器里保存的一些数据或者标志，记录了通信过程中的一些变化信息。

你一定知道，TCP 协议是有状态的，一开始处于 CLOSED 状态，连接成功后是 ESTABLISHED 状态，断开连接后是 FIN-WAIT 状态，最后又是 CLOSED 状态。

这些“状态”就需要 TCP 在内部用一些数据结构去维护，可以简单地想象成是个标志量，标记当前所处的状态，例如 0 是 CLOSED，2 是 ESTABLISHED 等等。

领资料



再来看 HTTP，那么对比一下 TCP 就看出来了，在整个协议里没有规定任何的“状态”，客户端和服务端永远是处在一种“**无知**”的状态。建立连接前两者互不知情，每次收发的报文也都是互相独立的，没有任何的联系。收发报文也不会对客户端或服务器产生任何影响，连接后也不会要求保存任何信息。

“无状态”形象地来说就是“没有记忆能力”。比如，浏览器发了一个请求，说“我是小明，请给我 A 文件。”，服务器收到报文后就会检查一下权限，看小明确实可以访问 A 文件，于是把文件发回给浏览器。接着浏览器还想要 B 文件，但服务器不会记录刚才的请求状态，不知道第二个请求和第一个请求是同一个浏览器发来的，所以浏览器必须还得重复一次自己的身份才行：“我是刚才的小明，请再给我 B 文件。”

我们可以再对比一下 UDP 协议，不过它是无连接也无状态的，顺序发包乱序收包，数据包发出去后就不管了，收到后也不会顺序整理。而 HTTP 是有连接无状态，顺序发包顺序收包，按照收发的顺序管理报文。

但不要忘了 HTTP 是“灵活可扩展”的，虽然标准里没有规定“状态”，但完全能够在协议的框架里给它“打个补丁”，增加这个特性。

## 其他特点

除了以上的五大特点，其实 HTTP 协议还可以列出非常多的特点，例如传输的实体数据可缓存可压缩、可分段获取数据、支持身份认证、支持国际化语言等。但这些并不能算是 HTTP 的基本特点，因为这都是由第一个“灵活可扩展”的特点所衍生出来的。

## 小结

1. HTTP 是灵活可扩展的，可以任意添加头字段实现任意功能；
2. HTTP 是可靠传输协议，基于 TCP/IP 协议“尽量”保证数据的送达；
3. HTTP 是应用层协议，比 FTP、SSH 等更通用功能更多，能够传输任意数据；
4. HTTP 使用了请求 – 应答模式，客户端主动发起请求，服务器被动回复请求；
5. HTTP 本质上是无状态的，每个请求都是互相独立、毫无关联的，协议不要求客户端或服务端记录请求相关的信息。

## 课下作业

领资料



1. 就如同开头我讲的那样，你能说一下今天列出的这些 HTTP 的特点中哪些是优点，哪些是缺点吗？
2. 不同的应用场合有不同的侧重方面，你觉得哪个特点对你来说是最重要的呢？

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。

领资料







## == 课外小贴士 ==

- 01 如果要 100% 保证数据收发成功就不能使用 HTTP 或者 TCP 协议了，而是要用各种消息中间件 (MQ)，如 RabbitMQ、ZeroMQ、Kafka 等。
- 02 以前 HTTP 协议还有一个“无连接”的特点，指的是协议不保持连接状态，每次请求应答后都会关闭连接，这就和 UDP 几乎一模一样了。但这样会很影响性能，在 HTTP/1.1 里就改成了总是默认启用 keepalive 长连接机制，所以现在的 HTTP 已经不再是“无连接”的了。
- 03 注意 HTTP 的“无状态”特点与响应头里的“状态码”是完全不相关的两个概念，状态码表示的是此次报文处理的结果，并不会导致服务器内部状态变化。

领资料



# 透视 HTTP 协议

深入理解 HTTP 协议本质与应用

罗剑锋

奇虎360技术专家


Nginx/OpenResty 开源项目贡献者



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

分享给需要的人，Ta订阅超级会员，你将得 **50** 元

Ta单独购买本课程，你将得 **20** 元

 生成海报并分享

 赞 16

 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 12 | 响应状态码该怎么用？

下一篇 14 | HTTP有哪些优点？又有哪些缺点？

领资料





# JVM + NIO + Spring

各大厂面试题及知识点详解

限时免费



## 精选留言 (47)

写留言



壹笙 漂泊

2019-06-26

课后题：

1、我觉得所谓的优点和缺点都是要区别场景来看待的，比如，在一些长连接场景中，需要保存上下文状态，那么无状态这一点就成为缺点甚至是致命缺点了。但是在客户端-服务端通信中，如果场景不需要保存上下文信息，那么无状态就可以减少一些网络资源消耗，也就是优点了。

2、可靠传输对我比较重要，减少很多查错的工作。。。

总结：

http特点：

灵活可扩展

可以扩展头字段实现功能

可靠传输

HTTP并不能100%保证数据一定能够发送到另一端，在网络繁忙、连接差等恶劣环境时，也有可能收发失败，可靠只是向使用者提供了一个承诺，会在下层用多种手段尽量保证数据的完整送达

应用层协议

请求-应答通信模式

客户端主动请求，服务端被动响应

无状态协议

领资料



状态：客户端或者服务器里保存的一些数据或者标志，记录了通信过程中的一些变化信息  
每个请求都是互相独立，毫无关联的，两端都不会记录请求相关的信息

作者回复：一直这么认真学习，值得表扬！！



👍 49



狼的诱惑

2019-07-31

老师好，有两点疑问

1、为什么说MQ的比HTTP是高可靠的

2、个人觉得http的无状态，完全符合可扩展，轻量级、易维护现代设计，属于优点，不算是缺点，为什么大家认为不支持有状态就算是缺点，如果http做成有状态，估计复杂度会非常高吧？

个人想法，还请老师指正

作者回复：

1.mq的设计目的就是消息传输，保证不丢失，为此用了很多的手段，比如消息队列、路由、存储等等。而http的目的不在于此，它的可靠性基于tcp，而tcp不能保证消息100%发送。

2.无状态有的时候是优点有的时候是缺点，看应用场景，在需要扩展的时候无状态就是优点，在需要会话保持的时候就是缺点。

3.但无状态可以很容易变成有状态，而反过来就很难，这就体现了http的灵活性。



👍 27



瑞

2019-06-29

http无状态不是特别理解，状态的意思就是一个事物因为外界因素导致这个事物的形态，使用等发生改变，这个就是有状态的吧，http作为一个协议，本质不承载任何东西，只是一个数据的传输通道，即使传输通道里的数据有状态改变，也不能称为是http状态改变，因为理解为无状态，可以这样理解吗？

作者回复：http协议里的“状态”是指没有规定通信双方需要记录通信过程的上下文信息，与日常生活的“状态”意思是不一样的。

每次http请求都是独立，无关的，不需要保留状态信息。

共 2 条评论 >

👍 13

领资料





钱

2020-03-28

1: 就如同开头我讲的那样, 你能说一下今天列出的这些 HTTP 的特点中哪些是优点, 哪些是缺点吗?

灵活可扩展是优点, 没有这一点, 她不可能一统江湖, 这也是她敢说“在坐的各位都是垃圾”的底气。

灵活可扩展是缺点, 她引入了一定的复杂度, 增加了一定的学习成本。

可靠传输协议是优点, 保证了数据的可靠性。

可靠传输协议是缺点, 必须先建连才行, 效率估计不如UDP。

应用层协议是优点, 靠近用户, 对用户优化, 方便使用。

应用层协议是缺点, 越往上处理的事情会越多, 通信效率会差一些。

请求应答模式是优点, 符合人类的对话方式容易理解, 一个请求得到响应在发生另外一个。

请求应答模式是缺点, 我请求一个资源, 发现不够再请求一次, 你一次回给我俩, 不用回两次这么简单的事就做不到。

无状态是优点, 无状态意味着易扩展, 也不需要保持状态信息, 节省空间。

无状态是缺点, 我想保持回话状态还必须自己去想办法。

好吧! 😊我编不下去了, 不过我觉得HTTP最大的优点就是灵活可扩展, 我们自己设计程序也希望做到这样, 不过真要如此, 复杂度会马上去, 不过HTTP确做到了灵活可扩展但是复杂度却没有增加太多。这一点太厉害了。

2: 不同的应用场合有不同的侧重方面, 你觉得哪个特点对你来说是最重要的呢?

看应用场景吧! 目前, 可靠传输对我而言至关重要, 效率可以放一放, 安全稳定第一。

作者回复:

1.说的很好, 没有绝对的优点, 也没有绝对的缺点。

2.http的可靠传输不能算是顶级的, 但搭配上灵活就无敌了。



👍 10



Celine

2020-03-27

Http的缺点, 无状态, 在需要身份信息验证的时候就显示为缺点, 用户登陆后每次请求都需要阐释一下身份, 不能保存登陆成功的状态, 不过现在可以在头字段里加入cookie或者token来解决这个问题;请求应答模式, 导致服务器端只能被动的接收而不能主动推送, 如果服务器资源状态变化的时候想要主动推送到客户端就需要借助其他协议来完成, 比如we socket

作者回复: 说的很好, 最后应该是笔误, WebSocket。



👍 10

领资料





火车日记  
2019-06-26

一个能打的都没有!!

缺点:明文传输



8



渴望做梦  
2019-08-21

老师，这个顺序发包和顺序收包不是很理解，是指第一个发送的请求一定会排在第一位处理吗？

作者回复: 是的，http这种请求-应答模式就是这样，必须第一个有回应后才能处理下一个。

共 2 条评论 >



7



尔冬橙  
2019-12-14

HTTP的长连接和无状态不矛盾么

作者回复: 不矛盾，长连接只是在连接层面节约了成本，每次的请求还是没有携带任何客户端的信息，服务器仍然不记录状态，每次请求都是独立的。

共 3 条评论 >



5



彩色的沙漠  
2019-06-26

请求应答模式也就注定了HTTP不适合用于IM场景？

作者回复: 是的，所以就出现了WebSocket。

共 2 条评论 >



5



哈德韦  
2020-10-07

尽量送达不是UDP的特征吗？只送出去不确认的，而TCP是保证送达，收不到确认会重试的，如果实在不行就会报错。HTTP基于它，应该也是保证送达吧？

作者回复: 可能是我说的不太准确，引起误会了。tcp是保证送达的，用“尽量”也许和udp有点混淆了，抱歉。



3





潇潇雨歇

2020-06-13

- 1、无状态是把双刃剑，设计成无状态能够减少资源的使用。但是如果需要保存用户身份，那么无状态就是缺点了。
- 2、可靠传输和请求应答吧，能确定成功发起请求和收到响应很重要。这也是平常开发的基础

作者回复: 学习得很努力啊。

共 2 条评论 >



2



一票

2019-06-26

终于明白为何是HTTP一统天下了，扬长避短灵活使用才是王道。

作者回复: √

共 2 条评论 >



2



Tintin

2020-06-02

1. 优点：可扩展（更能适应不确定的环境）、可靠（在可靠比时效性重要的场景下）；缺点：可靠（在时效性比可靠更重要的场景下）、请求应答模式（服务器无法主动推送数据给浏览器）、无状态

作者回复: 说的很好。



1



有米

2020-05-13

即使是当红MQ也没人敢打包票100%不丢

作者回复: 对，但可靠性99.9%和99.999%还是有差距的，所以需要根据具体场合选用恰当的产品。



1

领资料



凉人。

2020-03-17

cookie应该算是一种上下文么？

这里有点想不通



作者回复: cookie只是用来携带kv数据, 本身不具有特定含义, 可以做任何事情, 所以很适合用来标记用户身份, 但这只是它的用途之一。

共 2 条评论 >

👍 1



ddq432

2019-12-05

我有个问题, http 用 tcp 来尽量保持可靠请求, 客户端发送一次请求服务, 服务端会发送一个tcp的ack, 告诉客户端我收到消息了, 假如说这次发送的tcp下的ack信息, 客户端没收到, 会怎么办

作者回复: 这个属于tcp协议要解决的问题, 如果没有收到ack, 客户端就认为是丢包, 启用重发机制, 再发一次。

共 2 条评论 >

👍 1



李鑫磊

2019-11-28

老师, 特别困惑的是: HTTP、WebService、RPC、RESTful、gRPC、WebSocket 这几个概念之间的联系和区别, 纠结...

作者回复: 在开头几讲简单介绍了这些概念, 要搞清楚的是协议、调用接口、设计风格/规范这些更大范围的概念。



👍 1



Cris

2019-08-16

HTTP 完全可以用开玩笑的口吻说: “不要误会, 我不是针对 FTP, 我是说在座的应用层各位, 都是垃圾。” websocket表示不服

作者回复: WebSocket属于后来的“小字辈”了, 而且应用范围比http还是要差很多, 不过应该不算“垃圾”了, 笑。



👍 1



响雨

2019-07-04

优点: 灵活拓展性强

缺点: 明文传输

领资料





因时而异：无状态

作者回复: 对



1



风翱

2019-06-26

1、灵活可扩展是优点，促进其发展。 2和3也是优点。 4、是优点也是缺点，缺点是应答方不能做到主动反馈。 5、即是优点也是缺点，主要是看适用的场景，无状态，请求时需要携带更多的数据。

作者回复: 说的很好。



1

领资料

