Why is it said that "HTTP is a stateless protocol"?

Asked 8 years, 5 months ago Active 7 months ago Viewed 177k times

182

HTTP has HTTP Cookies. Cookies allow the server to track the user state, the number of connections, last connection, etc.

HTTP has persistent connections (Keep-Alive) where several requests can be sent from the same TCP Connection.

90

http stateless

Share Improve this question

Follow

edited Aug 18 '18 at 16:33

Rick
6,423 7 37 62

asked Nov 2 '12 at 17:21

Jose Nobile
2,835 2 20 28

3 Another area where I don't see "stateless-ness" is in Authorization - particularly Proxy-Authorization. It seems that it is stateful during the negotiation. For NTLM Authentication, the client needs to remember the type of Proxy-Authentication and the server needs to be stateful since there is a sequence to the NTLM Message Types. So I'm not sure I understand the answers. – Lindsay Morsillo Aug 1 '13 at 18:34

1 Should I now add HTTP/1.1? Because I think HTTP/2 has state. – Jose Nobile Mar 5 '16 at 3:26

4 **HTTP/2 is stateful. HTTP 1 is stateless.** Later additions intended for HTTP 1, like cookies, added state. Those additions are not apart of the "core" HTTP 1 specification. This is why HTTP 1 is said to be a stateless protocol although in practice it is not. HTTP/2 on the other hand was designed with stateful components baked in. No additions were required to satisfy the requirement of being labeled "stateful". – Zamicol Apr 12 '17 at 3:42

Also related: stackoverflow.com/questions/11067500/… stackoverflow.com/questions/36178447/… – Andrew Aug 23 '17 at 14:16

10 Answers

Active Oldest Votes

10 Answers Active Oldest Votes

136

Even though multiple requests can be sent over the same HTTP connection, the server does not attach any special meaning to their arriving over the same socket. That is solely a performance thing, intended to minimize the time/bandwidth that'd otherwise be spent reestablishing a connection for each request.

As far as HTTP is concerned, they are all still separate requests and must contain enough information on their own to fulfill the request. That is the essence of "statelessness". Requests will not be associated with each other absent some shared info the server knows about, which in most cases is a session ID in a cookie.

Share Improve this answer Follow

edited Nov 2 '12 at 17:37

answered Nov 2 '12 at 17:24

cHao

78.5k 19 134 168

1 What happens when the server remembers a session (server-side) and customizes user experience according to it? – NurShomik Feb 8 '16 at 16:46

3 @NurShomik: See stackoverflow.com/a/3521393/319403 for an explanation of how sessions typically work. – cHao Feb 9 '16 at 1:10

12 @Andrew: HTTP is not "built on" TCP, and TCP's state is not HTTP's. The two are entirely separate protocols at different layers in the stack. You could serve HTTP over named pipes if you wanted, or even by sending files around, if you got enough masochists to agree to do it, and it would work precisely because HTTP is transport-protocol-agnostic. At that level, it's all just requests and responses. That makes HTTP itself stateless, regardless of what state may be used/maintained/required by lower- or higher-level protocols. – cHao Aug 22 '17 at 17:26

@cHao Okay, I'll concede. **If** we define statelessness as "not *necessarily* needing to have state in order to operate" (see dimo414's answer below listing options for state within HTTP cited from Wikipedia), and **if** we view each protocol strictly by itself and not based upon the layers below it, then yes, I can agree that HTTP is "stateless". – Andrew Dec 24 '19 at 2:05

105

From Wikipedia:

> HTTP is a stateless protocol. A stateless protocol does not require the server to retain information or status about each user for the duration of multiple requests.
>
> But some web applications may have to track the user's progress from page to page, for example when a web server is required to customize the content of a web page for a user. Solutions for these cases include:
>
> - the use of HTTP cookies.
> - server side sessions,
> - hidden variables (when the current page contains a form), and
> - URL-rewriting using URI-encoded parameters, e.g., /index.php?session_id=some_unique_session_code.

What makes the protocol stateless is that the server is not *required* to track state over multiple

requests, not that it cannot do so if it wants to. This simplifies the contract between client and server, and in many cases (for instance serving up static data over a CDN) minimizes the amount of data that needs to be transferred. If servers were *required* to maintain the state of clients' visits the structure of issuing and responding to requests would be more complex. As it is, the simplicity of the model is one of its greatest features.

Share Improve this answer Follow

edited Jan 11 '16 at 19:16

answered Nov 2 '12 at 17:53

dimo414

42k 17 129 216

19

Because a stateless protocol does not require the server to retain session information or status about each communications partner for the duration of multiple requests.

HTTP is a stateless protocol, which means that the connection between the browser and the server is lost once the transaction ends.

Share Improve this answer Follow

answered Nov 2 '12 at 17:25

Rahul Tripathi

152k 27 229 296

3 But, HTTP can save information in server, using cookies. HTTP wih keep-alive don't close connection on each request. – Jose Nobile Nov 2 '12 at 17:29

3 Check out this article:- ecst.csuchico.edu/~amk/foo/advjava/notes/servlets/Cookies.html – Rahul Tripathi Nov 2 '12 at 17:31

19 Saving information on server does not mean that the connection is alive constantly. – srijan Nov 2 '12 at 17:32

1 @srijan Well, no. So? Nobody was claiming otherwise. – Mark Amery Nov 2 '15 at 16:57

13

HTTP is called as a `stateless protocol` because each request is executed independently, without any knowledge of the requests that were executed before it, which means once the transaction ends the connection between the browser and the server is also lost.

What makes the protocol `stateless` is that in its original design, **HTTP** is a relatively simple `file transfer protocol`:

1. make a request for a file named by a URL,
2. get the file in response,
3. disconnect.

There was no relationship maintained between one connection and another, even from the same client. This simplifies the contract between client and server, and in many cases minimizes the amount of data that needs to be transferred.

Share Improve this answer Follow

edited Mar 18 '20 at 16:43

community wiki

2 revs

Mobeen Sarwar

1 This is not true, especially HTTP/2 where request depend on other requests. Cookies, HTTPS, HTTP authentication, Web Storage, HTTP caching, HTTP stream identifiers, HTTP/2 header blocks, HTTP/2 frames, header compression, and opportunistic encryption are all stateful. – Zamicol Aug 3 '20 at 2:59

4

If protocol HTTP is given as State full protocol,browser window uses single connection to communicate with web server for multiple request given to web application.this gives chance to browser window to engage the connections between browser window and web servers for long time and to keep them in idle state for long time.This may create the situation of reaching to maximum connections of web server even though most of the connections in clients are idle.

Share Improve this answer Follow

answered Jan 21 '14 at 5:56

Rajasekhar reddy
51 1

1 HTTP already have keep-alive, this mean that server doesn't close the connection, and client can makes many request on the same connection. – Jose Nobile Apr 4 '14 at 23:26

3

HTTP is a connectionless and this is a direct result that HTTP is a stateless protocol. The server and client are aware of each other only during a current request. Afterwards, both of them forget about each other. Due to this nature of the protocol, neither the client nor the browser can retain information between different request across the web pages.

Share Improve this answer Follow

answered Apr 4 '14 at 6:29

user3496740
39 1

2

`HTTP` is stateless. `TCP` is stateful. There is no so-called `HTTP connection`, but only `HTTP request` and `HTTP response`. We don't need anything to be maintained to make another `HTTP request`. A connection header that is **"keep-alive"** means the `TCP` will be reused by the subsequent `HTTP` requests and responses, instead of disconnecting and re-establishing `TCP` connection all the time.

Share Improve this answer Follow

edited Mar 18 '20 at 22:38

Mobeen Sarwar
514 5 22

answered Jun 25 '18 at 12:59

Xing
31 2

HTTP/2 is stateful and does have a HTTP connection, called a stream. tools.ietf.org/html/rfc7540#section-5.1 – Zamicol Aug 4 '20 at 2:19

2

Modern HTTP is stateful. Old timey HTTP was stateless.

Before Netscape invented cookies and HTTPS in 1994 http could be considered stateless. As time progressed more and more stateful aspects were added for a myriad of reasons,

including performance and security.

While HTTP 1 originally sought out to be stateless many HTTP/2 components are the very definition of stateful. **HTTP/2 abandoned stateless goals**.

No reasonable person can read the HTTP/2 RFC and think it is stateless. The errant "HTTP is stateless" old time dogma is false and far from the current reality of stateful HTTP.

Here's a limited, and not exhaustive list, of stateful HTTP/1 and HTTP/2 components:

- Cookies, named "HTTP State Management Mechanism" by the RFC.
- HTTPS, which stores keys thus state.
- HTTP authentication requires state.
- Web Storage.
- HTTP caching is stateful.

- The very purpose of the stream identifier is state. It's even in the name of the RFC section, "Stream States".
- Header blocks, which establish stream identifiers, are stateful.
- Frames which reference stream identifiers are stateful.
- Header Compression, which the HTTP RFC explicitly says is stateful, is stateful.
- Opportunistic encryption is stateful.

Section 5.1 of the HTTP/2 RFC is a great example of stateful mechanisms defined by the HTTP/2 standard.

Is it safe for web applications to consider HTTP/2 as a stateless protocol?

HTTP/2 is a stateful protocol, but that doesn't mean your HTTP/2 application can't be stateless. You can choose to not use certain stateful features for stateless HTTP/2 applications by using only a subset of HTTP/2 features.

Cookies and some other stateful mechanisms, or less obvious stateful mechanisms, are later HTTP additions. HTTP 1 is said to be stateless although in practice we use standardized stateful mechanisms, like cookies, TLS, and caching. Unlike HTTP/1, HTTP/2 defines stateful components in its standard from the very beginning. A particular HTTP/2 application can use a subset of HTTP/2 features to maintain statelessness, but the protocol itself anticipate state to be the norm, not the exception.

Existing applications, even HTTP 1 applications, needing state will break if trying to use them statelessly. It can be impossible to log into some HTTP/1.1 websites if cookies are disabled, thus breaking the application. It may not be safe to assume that a particular HTTP 1 application does not use state. This is no different for HTTP/2.

Say it with me one last time:

HTTP/2 is a stateful protocol.

Share Improve this answer Follow

edited Sep 14 '20 at 22:56

answered Aug 3 '20 at 2:54

Zamicol
3,228 1 27 35

"HTTPS, which stores keys thus state". Mh, no has nothing ot do with HTTP. HTTPS = HTTP over TLS. The key "storage" happens in another protcol layer. en.wikipedia.org/wiki/HTTPS#Difference_from_HTTP – Sebi2020 Sep 13 '20 at 21:30

1 this is a reasonable answer. But why does nobody support this? – scottxiao Feb 9 at 5:25

1

What is stateless??

Once the request is made and the response is rendered back to the client the connection will be dropped or terminated. The server will forget all about the requester.

Why stateless??

The web chooses to go for the stateless protocol. It was a genius choice because the original goal of the web was to allow documents(web pages) to be served to extremely large no. of people using very basic hardware for the server.

Maintaining a long-running connection would have been extremely resource-intensive.

If the web were chosen the stateful protocol then the load on the server would have been increased to maintain the visitor's connection.

Share Improve this answer Follow

answered Feb 13 '20 at 16:51

chirag soni
594 5 16

1

I think somebody chose very unfortunate name for STATELESS concept and that's why the whole misunderstanding is caused. It's not about storing any kind of resources, but rather about the relationship between the client and the server.

Client: I'm keeping all the resources on my side and send you the "list" of all the important items which need to processed. Do your job.

Server: All right.. let me take the responsibility on filtering what's important to give you the proper response.

Which means that the server is the "slave" of the client and has to forget about his "master" after each request. Actually, STATELESS refers only to the state of the server.

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_1_3

Share Improve this answer Follow

answered Mar 16 '20 at 17:31

JacobTheKnitter
61 4