REliable, INtelligent & Scalable Systems

互联网开发技术

Web Application Development

第2课 HTTP协议

Episode Two

HTTP Protocol

Web Application Development

陈昊鹏

chen-hp@sjtu.edu.cn

HTTP Protocol （协议）

- HTTP functions as a request–response protocol in the client–server computing model.

HTTP协议是一种请求-响应式协议

  - A web browser, for example, may be the *client* and an application running on a computer hosting a website may be the *server*.
  - The client submits an HTTP *request* message to the server.
  - The server, which provides *resources* such as HTML files and other content, or performs other functions on behalf of the client, returns a *response* message to the client.
    - The response contains completion status information about the request and may also contain requested content in its message body.
  - A web browser is an example of a *user agent* (UA).
    - Other types of user agent include the indexing software used by search providers (web crawlers), voice browsers, mobile apps, and other software that accesses, consumes, or displays web content.

HTTP Protocol

- HTTP resources are identified and located on the network by Uniform Resource Locators (URLs), 统一资源定位符
  - using the Uniform Resource Identifiers (URI's) schemes *http* and *https*.
  - URIs and hyperlinks in HTML documents form inter-linked hypertext documents.

- Every HTTP URL conforms to the syntax of a generic URI. A generic URI is of the form:
  - scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]

    端口编号 设置参数，一般以键值对形式出现 书签

  - For example:
  - https://en.wikipedia.org/wiki/Uniform_Resource_Locator

Uniform Resource Locator

- **scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]**
  - **scheme**,
    - consisting of a sequence of characters beginning with a letter and followed by any combination of letters, digits, plus (+), period (.), or hyphen (-).
    - It is followed by a colon (:).
    - Examples of popular schemes include http, ftp, mailto, file, data, and irc.
  - Two slashes (//)
    - This is required by some schemes and not required by some others.
  - An **authority part**, comprising:
    - An optional authentication section of a user name and password, separated by a colon, followed by an at symbol (@)
  - A "**host**",
    - consisting of either a registered name (including but not limited to a hostname), or an IP address.
    - An optional port number, separated from the hostname by a colon

Uniform Resource Locator

REliable, INtelligent & Scalable Systems

- **scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]**
  - A **path**,
    - which contains data, usually organized in hierarchical form, that appears as a sequence of segments separated by slashes.
  - An optional **query**,
    - separated from the preceding part by a question mark (?), containing a query string of non-hierarchical data.
    - Its syntax is not well defined, but by convention is most often a sequence of attribute–value pairs separated by a delimiter.

Query delimiter	Example
Ampersand (&)	key1=value1&key2=value2
Semicolon (;)	key1=value1;key2=value2

  - An optional **fragment**,
    - separated from the preceding part by a hash (#). The fragment contains

A Request & A Response

- http://www.example.com/index.html

GET:一种表示要执行的
方法的标记
包括：
GET PUT DELETE POST

- **Client request**

```
GET / HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

此部分统称为 request
head（请求头）

Request Syntax

- A client sends *request messages* to the server, which consist of:
  - a **request line**, consisting of the case-sensitive request method, a space, the requested URL, another space, the protocol version, a carriage return, and a line feed, e.g.:

    GET /images/logo.png HTTP/1.1

  - zero or more request header fields (at least 1 or more headers in case of HTTP/1.1), each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:

    Host: www.example.com

    Accept-Language: en

  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

  - In the HTTP/1.1 protocol, all header fields except Host: hostname are optional.

Request Syntax

- A client sends *request messages* to the server, which consist of:
  - a **request line**, consisting of the case-sensitive request method, a space, the requested URL, another space, the protocol version, a carriage return, and a line feed, e.g.:

    GET /images/logo.png HTTP/1.1

  - zero or more request header fields (at least 1 or more headers in case of HTTP/1.1), each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:

    Host: www.example.com

    Accept-Language: en

  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

  - In the HTTP/1.1 protocol, all header fields except Host: hostname are optional.

Request methods

REliable, INtelligent & Scalable Systems

- HTTP defines methods to indicate the desired action to be performed on the identified resource.
  - GET
    - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect. (idempotent)
  - HEAD

    idempotent：幂等的，即重复运行多次的结果都是相同的

    - The HEAD method asks for a response identical to that of a GET request, but without the response body. (idempotent)
  - POST

    POST不是幂等的

    - The POST method requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI.
  - PUT
    - The PUT method requests that the enclosed entity be stored under the supplied URI. (idempotent)
  - DELETE
    - The DELETE method deletes the specified resource. (idempotent)

Request methods

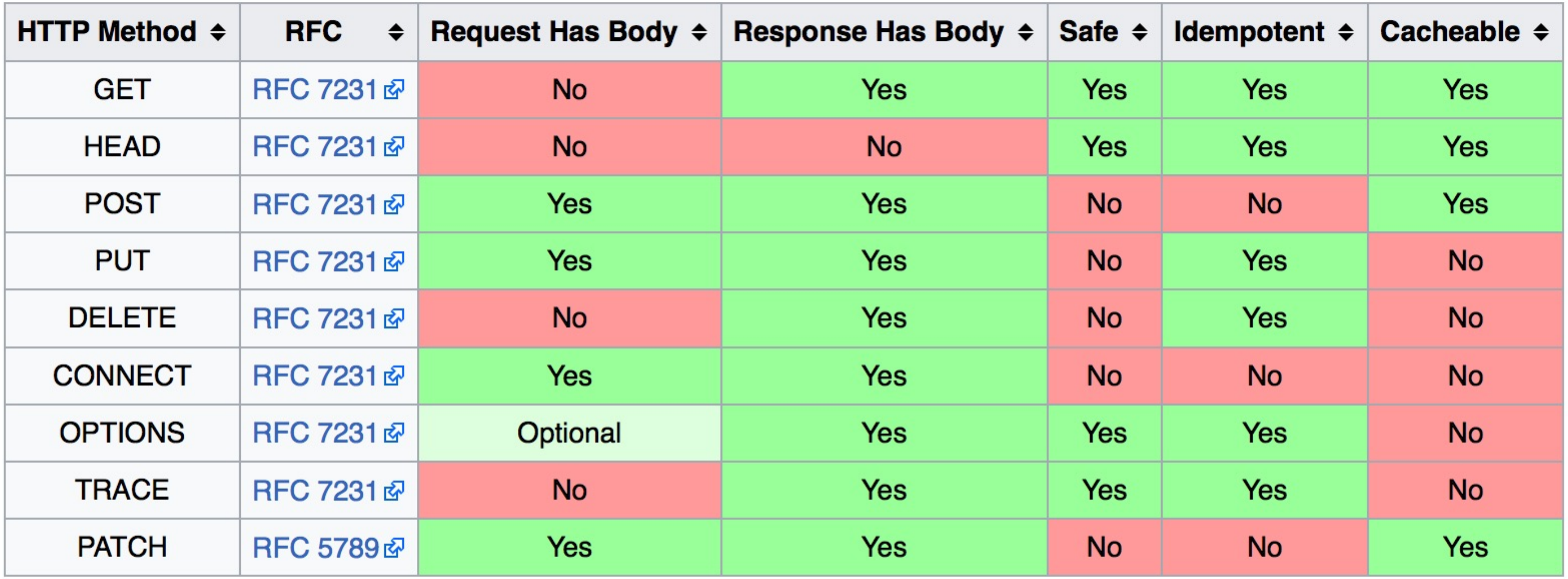
- HTTP defines methods to indicate the desired action to be performed on the identified resource.
  - TRACE
    - The TRACE method echoes the received request so that a client can see what (if any) changes or additions have been made by intermediate servers. (idempotent)
  - OPTIONS
    - The OPTIONS method returns the HTTP methods that the server supports for the specified URL. This can be used to check the functionality of a web server by requesting '*' instead of a specific resource. (idempotent)
  - CONNECT
    - The CONNECT method converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.
  - PATCH
    - The PATCH method applies partial modifications to a resource.
  - All general-purpose HTTP servers are required to implement at least the GET and HEAD methods, and, whenever possible, also the OPTIONS method.

Safe methods & Idempotent methods

- Some of the methods (for example, HEAD, GET, OPTIONS and TRACE) are, by convention, defined as *safe*
  - which means they are intended only for information retrieval and should not change the state of the server.

- Methods PUT and DELETE are defined to be idempotent,
  - meaning that multiple identical requests should have the same effect as a single request
  - or the response code it returns may be different on subsequent requests, the system state will be the same every time.
  - Methods GET, HEAD, OPTIONS and TRACE, being prescribed as safe, should also be idempotent, as HTTP is a stateless protocol

Safe methods & Idempotent methods



HTTP Method	RFC	Request Has Body	Response Has Body	Safe	Idempotent	Cacheable
GET	RFC 7231	No	Yes	Yes	Yes	Yes
HEAD	RFC 7231	No	No	Yes	Yes	Yes
POST	RFC 7231	Yes	Yes	No	No	Yes
PUT	RFC 7231	Yes	Yes	No	Yes	No
DELETE	RFC 7231	No	Yes	No	Yes	No
CONNECT	RFC 7231	Yes	Yes	No	No	No
OPTIONS	RFC 7231	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231	No	Yes	Yes	Yes	No
PATCH	RFC 5789	Yes	Yes	No	No	Yes

Request Header Fields

Name	Description	Example
Accept	Media type(s) that is/are acceptable for the response. See Content negotiation.	Accept: text/html
Accept-Charset	Character sets that are acceptable.	Accept-Charset: utf-8
Accept-Datetime	Acceptable version in time.	Accept-Datetime: Thu, 31 May 2007 20:35:00 GMT
Accept-Encoding	List of acceptable encodings. See HTTP compression.	Accept-Encoding: gzip, deflate
Accept-Language	List of acceptable human languages for response. See Content negotiation.	Accept-Language: en-US
Cache-Control	Used to specify directives that *must* be obeyed by all caching mechanisms along the request-response chain.	Cache-Control: no-cache
Connection	Control options for the current connection and list of hop-by-hop request fields.[14]Must not be used with HTTP/2.[15]	Connection: keep-aliveConnection: Upgrade
Content-Encoding	The type of encoding used on the data. See HTTP compression.	Content-Encoding: gzip
Content-Length	The length of the request body in octets (8-bit bytes).	Content-Length: 348
Content-Type	The Media type of the body of the request (used with POST and PUT requests).	Content-Type: application/x-www-form-urlencoded
Cookie	An HTTP cookie previously sent by the server with Set-Cookie (below).	Cookie: $Version=1; Skin=new;

A Request & A Response

REliable, INtelligent & Scalable Systems

- http://www.example.com/index.html

- **Server response**

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 155
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close
```

```
<html>
  <head>
    <title>An Example Page</title>
  </head>
  <body>
    <p>
      Hello World, this is a very simple HTML document.
    </p>
  </body>
</html>
```

Response Syntax

- A server sends *response messages* to the client, which consist of:[47]
  - a **status line**, consisting of the protocol version, a space, the response status code, another space, a possibly empty reason phrase, a carriage return and a line feed, e.g.:
    HTTP/1.1 200 OK
  - zero or more response header fields, each consisting of the case-insensitive field name, a colon, optional leading whitespace, the field value, an optional trailing whitespace and ending with a carriage return and a line feed, e.g.:
    Content-Type: text/html
  - an empty line, consisting of a carriage return and a line feed;
  - an optional message body.

Response Status

- In HTTP/1.0 and since,
  - the first line of the HTTP response is called the *status line* and includes a numeric *status code* (such as "404") and a textual *reason phrase* (such as "Not Found").

- The first digit of the status code defines its class:
  - 1XX (informational) The request was received, continuing process.
  - 2XX (successful) The request was successfully received, understood, and accepted.
  - 3XX (redirection) Further action needs to be taken in order to complete the request.
  - 4XX (client error) The request contains bad syntax or cannot be fulfilled.
  - 5XX (server error) The server failed to fulfill an apparently valid request.

Response Header Fields

Field name	Description	Example
Accept-CH	Requests HTTP Client Hints	Accept-CH: UA, Platform
Access-Control-Allow-Origin, Access-Control-Allow-Credentials, Access-Control-Expose-Headers, Access-Control-Max-Age, Access-Control-Allow-Methods, Access-Control-Allow-Headers[13]	Specifying which web sites can participate in cross-origin resource sharing	Access-Control-Allow-Origin: *
Age	The age the object has been in a proxy cache in seconds	Age: 12
Allow	Valid methods for a specified resource. To be used for a *405 Method not allowed*	Allow: GET, HEAD
Content-Encoding	The type of encoding used on the data. See HTTP compression.	Content-Encoding: gzip
Content-Language	The natural language or languages of the intended audience for the enclosed content[51]	Content-Language: da
Content-Length	The length of the response body in octets (8-bit bytes)	Content-Length: 348
Content-Location	An alternate location for the returned data	Content-Location: /index.htm
Content-Type	The MIME type of this content	Content-Type: text/html; charset=utf-8

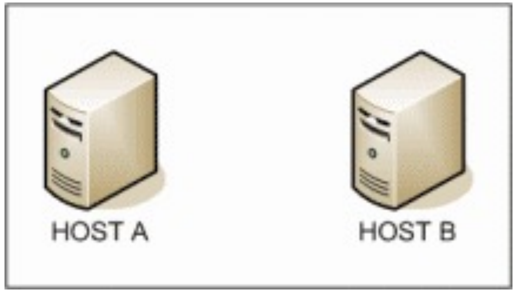
HTTP session

REliable, INtelligent & Scalable Systems

- An HTTP session is a sequence of network request-response transactions.
  - TCP 3-Way Handshake

HTTP是一种高层协议，一般通过三次握手来建立HTTP协议

第一次握手：A向B发送请求
第二次握手：B接收到A的消息，向A发送响应信息
第三次握手：A接收到B发来的接收到消息的响应信息，发出响应信息回应B的答复

EVENT	DIAGRAM
Host A **sends** a TCP **SYN**chronize packet to Host B Host B receives A's **SYN** Host B **sends** a **SYN**chronize-ACKnowledgement Host A receives B's **SYN-ACK** Host A **sends** **ACK**nowledge Host B receives **ACK**. ***TCP socket connection is ESTABLISHED.***	 TCP Three Way Handshake (SYN, SYN-ACK, ACK)

- *from: http://blog.csdn.net/wangshihui512/article/details/9051819*

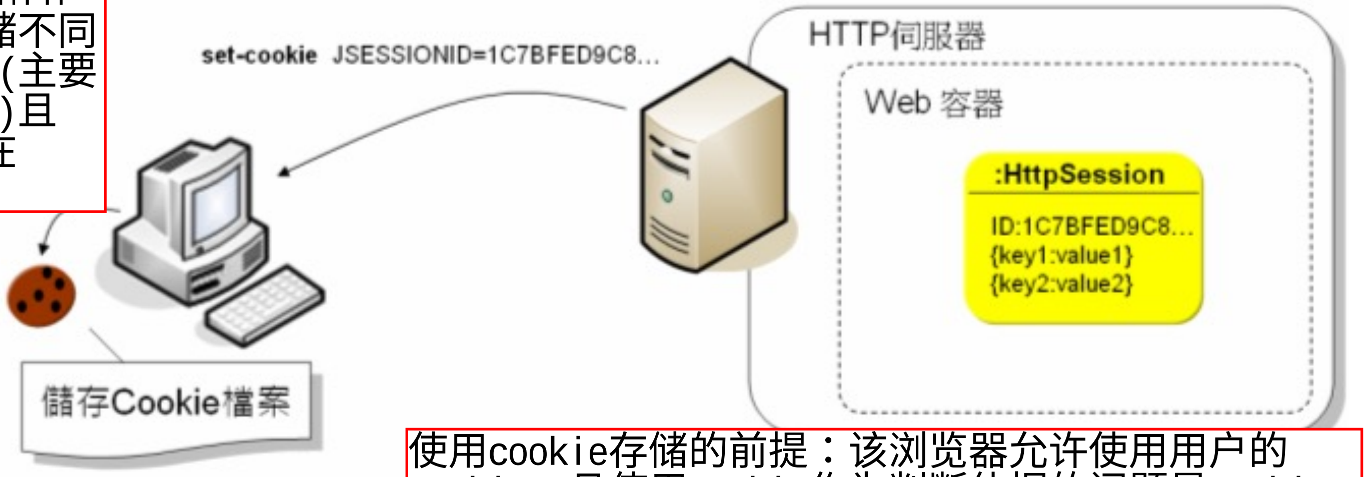
HTTP session state

- HTTP is a stateless protocol

HTTP协议不需要维护当前的状态

  – A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.

HTTP协议中用HTTP session来存储不同用户的ID值，(主要存的是键值对)且session存储在cookie中。



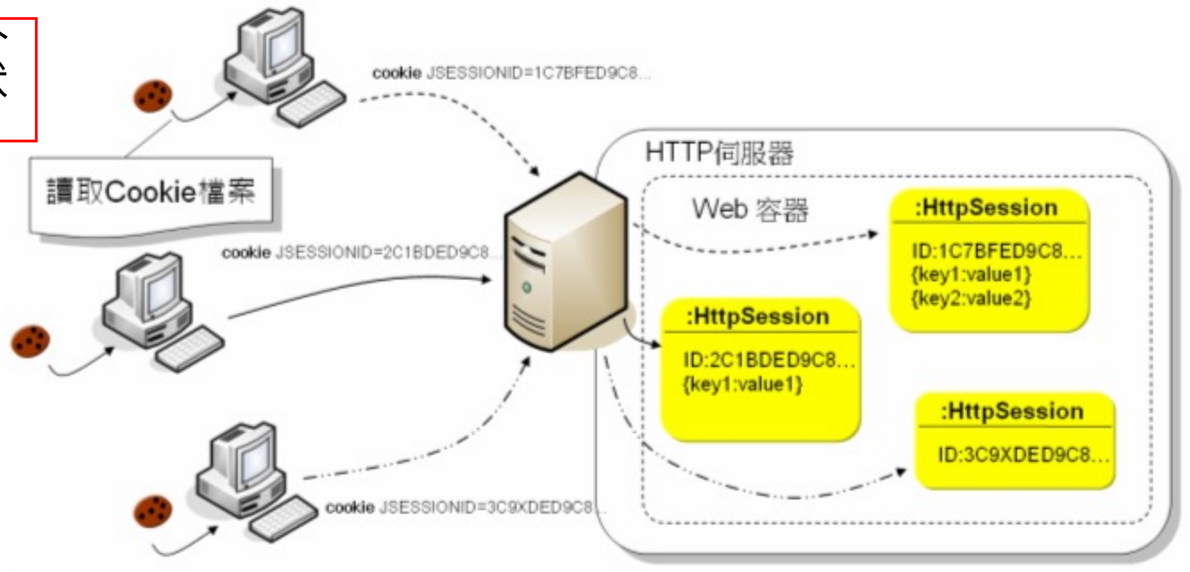
使用cookie存储的前提：该浏览器允许使用用户的cookie，且使用cookie作为判断依据的问题是cookie以纯文本保存，容易被修改

  – *from: https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html*

HTTP session state

- HTTP is a stateless protocol
  - A stateless protocol does not require the HTTP server to retain information or status about each user for the duration of multiple requests.

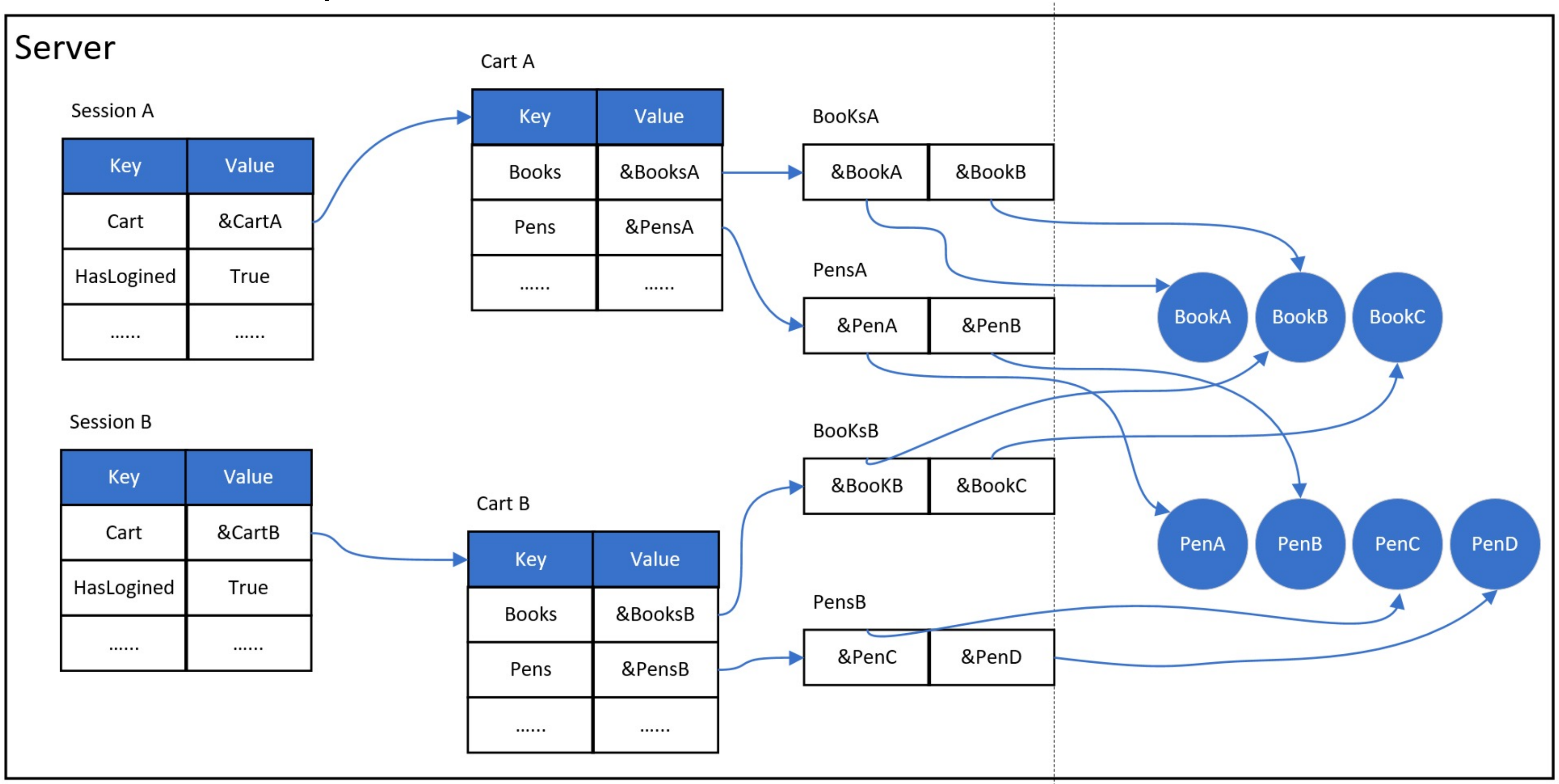
在这里引入第一个核心问题：会话状态维护



  - *from: https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html*

HTTP session state

- HTTP is a stateless protocol



References

- Hypertext Transfer Protocol
  - https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- Uniform Resource Locator
  - https://en.wikipedia.org/wiki/Uniform_Resource_Locator
- List of HTTP header fields
  - https://en.wikipedia.org/wiki/List_of_HTTP_header_fields#Request_fields
- HttpSession 原理
  - https://www.openhome.cc/Gossip/ServletJSP/BehindHttpSession.html

REin

- *Web开发技术*
- *Web Application Development*

Thank You!