Unit 4:
HTTP

By

Khushbu Kumari Sarraf

EMBA,BE

HTTP Protocol

- The Hypertext Transfer Protocol (HTTP) is a standard that defines how a web client talks to a server and how data is transferred from the server back to the client. HTTP works as a request-response protocol between a client and server. Although HTTP is usually thought of as a means of transferring HTML files and the pictures ,Microsoft Word documents, Windows *.exe* files, or anything else that embedded in them. A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.
- HTTP is the standard protocol for communication between web browsers and web servers. HTTP specifies how a client and server establish a connection, how the client requests data from the server, how the server responds to that request, and finally, how the connection is closed. HTTP connections use the TCP protocol for data transfer. For each request from client to server, there is a sequence of four steps:

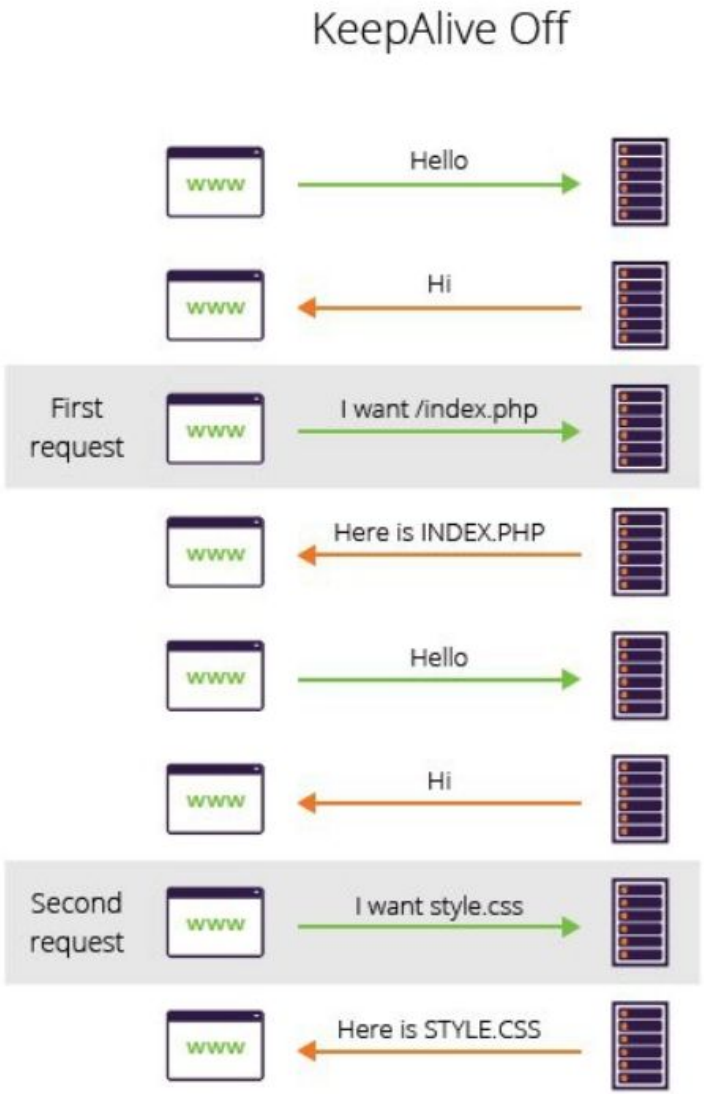
HTTP Protocol

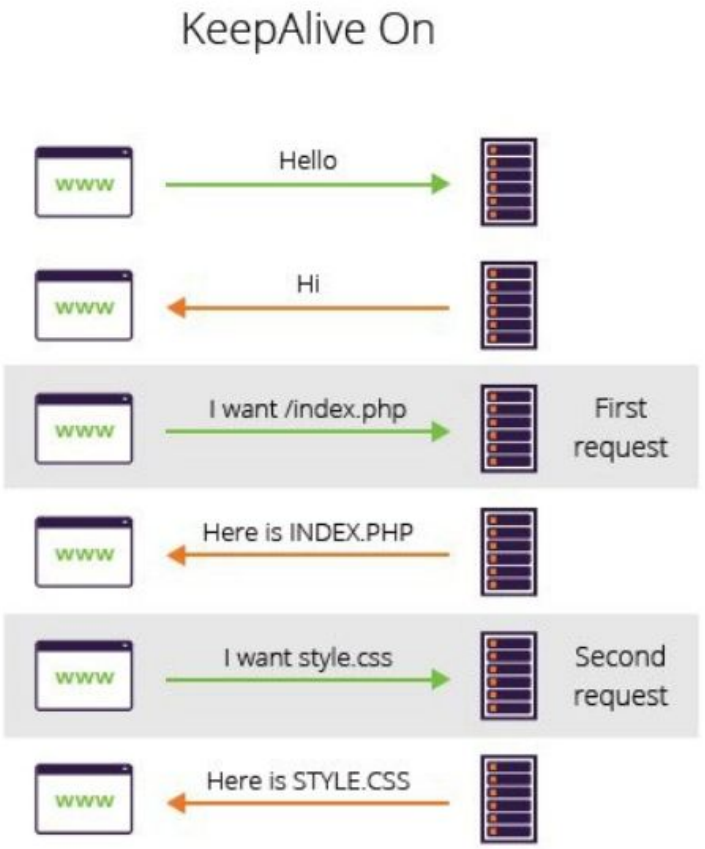
1. The client opens a TCP connection to the server on port 80, by default; other ports may be specified in the URL.
2. The client sends a message to the server requesting the resource at a specified path. The request includes a header, and optionally (depending on the nature of the request) a blank line followed by data for the request.
3. The server sends a response to the client. The response begins with a response code, followed by a header full of metadata, a blank line, and the requested document or an error message.
4. The server closes the connection.

HTTP Keep Alive

- HTTP persistent connections, also called HTTP keep-alive, or HTTP connection reuse, is the idea of using the same TCP connection to send and receive multiple HTTP requests/responses, as opposed to opening a new one for every single request/response pair. Using persistent connections is very important for improving HTTP performance.
- Establishing a TCP connection first requires a three-way handshake – a mutual exchange of SYN and ACK packets between a client and server before data can be transmitted. Using the keep-alive header means not having to constantly perform this process. This results in:
- **Network resource conservation** – It's less taxing on network resources to use a single connection per client.
- **Reduced network congestion** – Reducing the number of TCP connections between your servers and clients can lead to a drop in network congestion.
- **Decreased latency** – Reducing the number of three-way handshakes can lead to improved site latency. This is especially true with SSL/TLS connections, which require additional round-trips to encrypt and verify connections.

HTTP Keep Alive





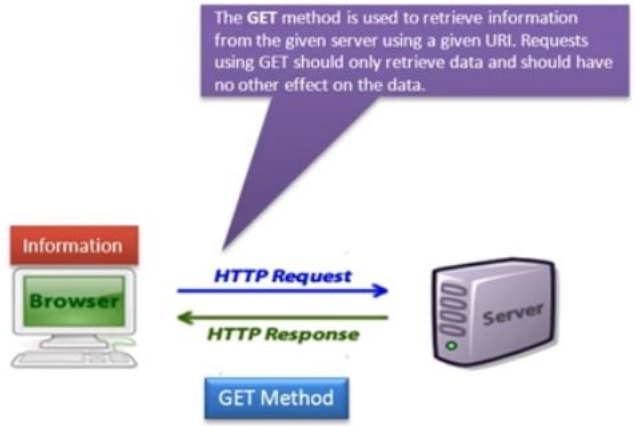
HTTP Methods

1. GET
2. HEAD
3. POST
4. PUT
5. DELETE
6. CONNECT
7. OPTIONS
8. TRACE

HTTP Methods

GET Method

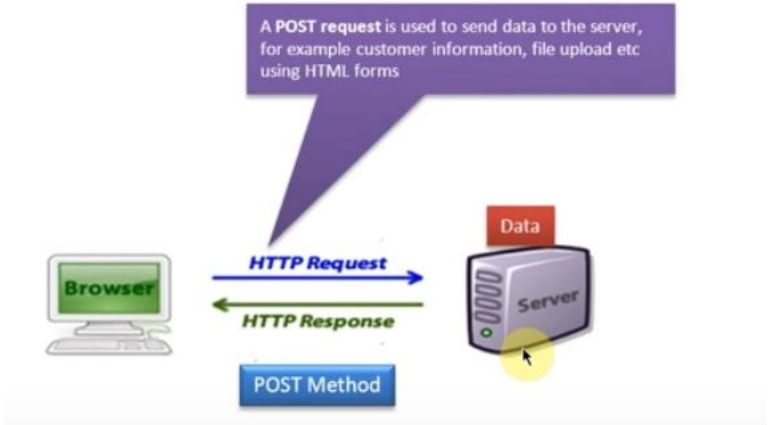
- A GET request retrieves data from a web server by specifying parameters in the URL portion of the request. This is the main method used for document retrieval.
- It is very easy to bookmark data using GET method.
- The length restriction of GET method is limited.
- You can use this method only to retrieve data from the address bar in the browser.
- This method enables you to easily store the data.



HTTP Methods

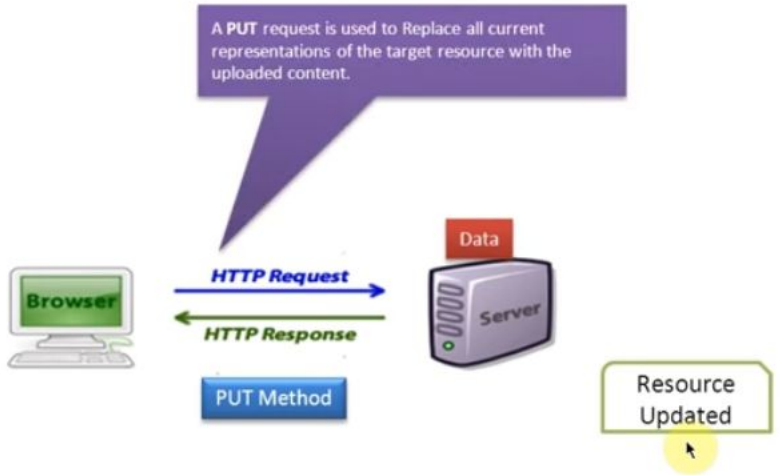
POST Method

- A POST request is used to send data to the server, for example, customer information, file upload, etc. using HTML forms.
- POST method request gets input from the request body and query string.
- Data passed using the POST method will not visible in query parameters in browser URL.
- Parameters of POST methods are not saved in browser history.
- There is no restriction in sending the length of data.
- It helps you to securely pass sensitive and confidential information like login details to server.



HTTP Methods

- **PUT Method**
  - Replaces all current representations of the target resource with the uploaded content.
  - PUT replaces the file or resource.
  - PUT responses are not cacheable.
  - Update a resource.



HTTP Methods

DELETE Method

- Removes all current representations of the target resource given by a URL.
- DELETE is quite easy to understand. It is used to *delete* a resource identified by filters or ID.

HTTP Methods

HEAD Method

- Head method is the same as GET but returns only HTTP headers and no document body. The HTTP HEAD request is used to check the availability, size, and last modification date of a resource without downloading it (as indicated by the Content-Length and Last-Modified headers).

CONNECT

- The http connect method starts two-way communications with the requested resource. It can be used to open a tunnel. The CONNECT method is used by the client to establish a network connection to a web server over HTTP.

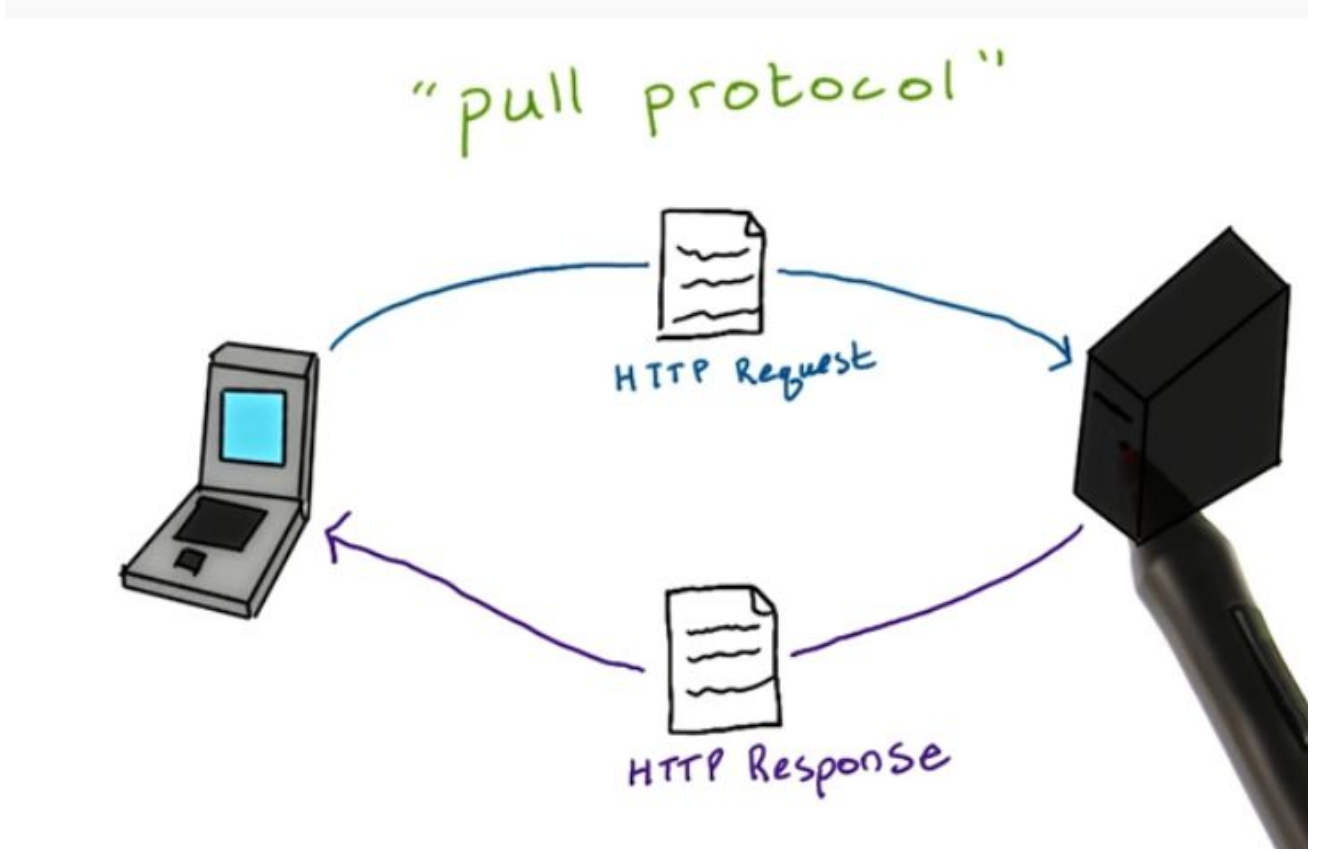
OPTIONS

- The OPTIONS method is used by the client to find out the HTTP methods and other options supported by a web server.

TRACE

- Performs a message loop-back test along the path to the target resource.

HTTP Request Body

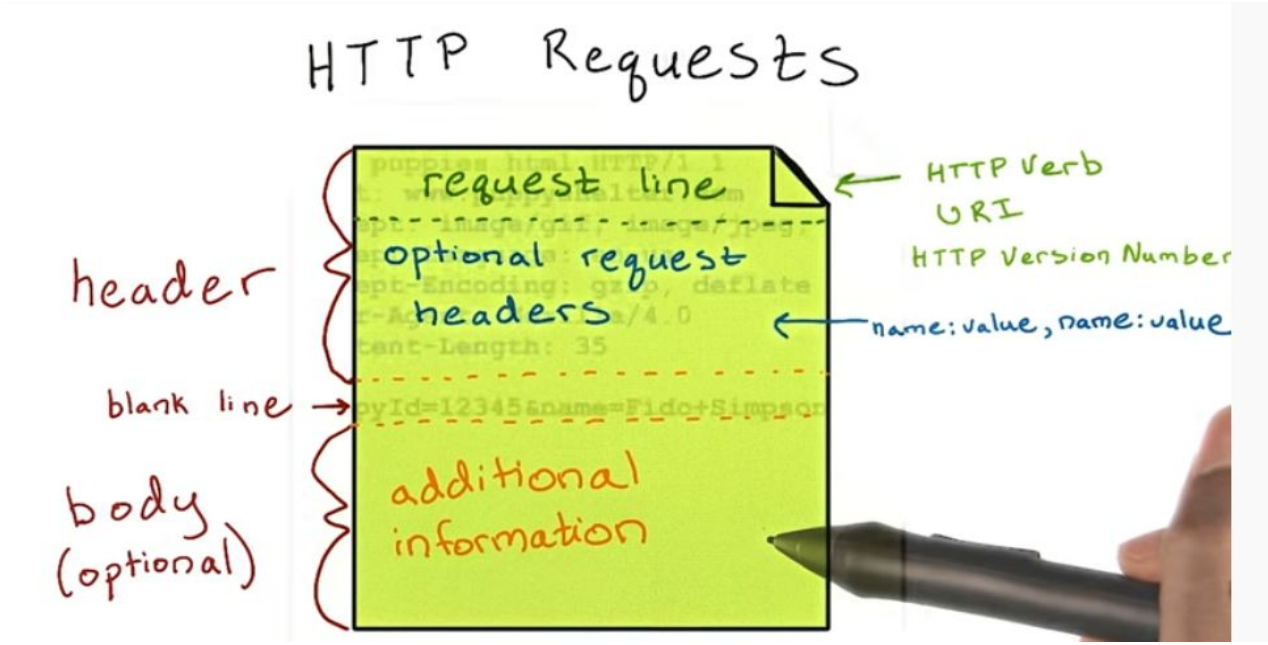


HTTP Request/Response Message

The request/response message consists of the following:

1. Request line, such as GET /logo.gif HTTP/1.1(Request Message) or Status line, such as HTTP/1.1 200 OK (Response Message)
2. Header
3. An empty line
4. Optionally HTTP message-body

HTTP Request/Respond Message



HTTP Request Body

Request Header Fields

The request-header fields allow the client to pass additional information about the request, and about the client itself, to the server. These fields act as request modifiers.

Examples of Request Message

```
GET /hello.htm HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

HTTP Request Message

The following example shows how to send form data to the server using request message body:

POST /cgi-bin/process.cgi HTTP/1.1 ==========Request Line

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.tutorialspoint.com

Content-Type: application/x-www-form-urlencoded

Content-Length: **length** Accept-Language: en-us Accept-Encoding: gzip, deflate

Connection: Keep-Alive

Additional Header Information

-------Blank Line

licenseID=string&content=string&/paramsXML=string ----------Message Body

HTTP Respond Message

An HTTP response for a request to fetch the **hello.htm** page from the web server.

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Length: 88 Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1>

</body>

</html>

Status Code

The Status-Code element is a 3-digit integer where first digit of the Status-Code defines the class of response and the last two digits do not have any categorization role. There are 5 values for the first digit:

S.N.	Code and Description
1	**1xx: Informational** It means the request was received and the process is continuing.
2	**2xx: Success** It means the action was successfully received, understood, and accepted.
3	**3xx: Redirection** It means further action must be taken in order to complete the request.
4	**4xx: Client Error** It means the request contains incorrect syntax or cannot be fulfilled.
5	**5xx: Server Error** It means the server failed to fulfill an apparently valid request.

Cookies

- Cookies are small files which are stored on a user's computer. They are used to hold a modest amount of data specific to a particular client and website and can be accessed either by the web server or by the client computer.
- When cookies were invented, they were basically little documents containing information about you and your preferences. For instance, when you select your language in which you want to view your website, the website would save the information in a document called a cookie on your computer, and the next time when you visit the website, it would be able to read a cookie saved earlier. That way the website could remember your language and let you view the website in your preferred language without having to select the language again.

Cookies

- Cookies are most commonly used to track website activity. When you visit some sites, the server gives you a cookie that acts as your identification card. Upon each return visit to that site, your browser passes that cookie back to the server. In this way, a web server can gather information about which web pages are used the most, and which pages are gathering the most repeat hits.
- Cookies are also used for online shopping. Online stores often use cookies that record any personal information you enter, as well as any items in your electronic shopping cart, so that you don't need to re-enter this information each time you visit the site.
- Servers can use cookies to provide personalized web pages. When you select preferences at a site that uses this option, the server places the information in a cookie. When you return, the server uses the information in the cookie to create a customized page for you.

CookieHandler

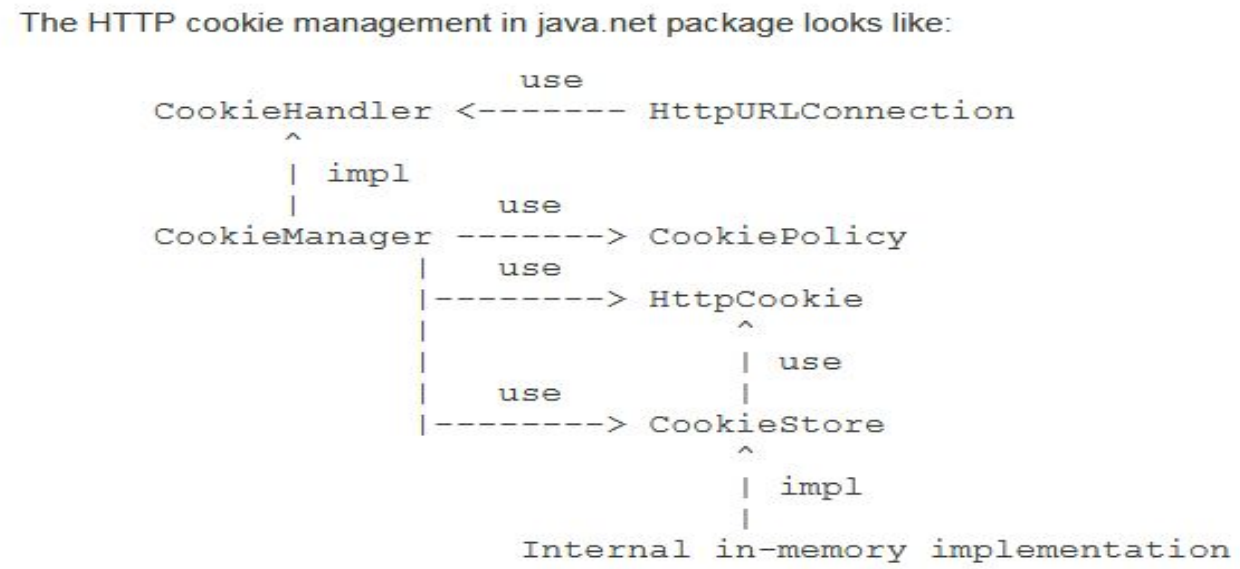
Provides an abstract base class for reading, writing, and deleting session cookies on an HTTP client. Consider, we are communicating with the server www.google.com, or any other URL that uses HTTP protocol, the URL object will be using an engine called the HTTP protocol handler. The HTTP protocol handler checks if there is a default CookieHandler instance in the system. If there is, it invokes it to take charge of state management.

CookieHandler Method

Methods	Action Performed
getDefault()	Used to retrieve the current CookieHandler installation.
setDefault(Cookie Handler)	Used to install a CookieHandler object on a system wide basis.
get(uri, requestHeaders)	Retrieves the cookies saved under the given URL and adds them to requestHeaders. It is called just before a request is made.
Put(uri, responseHeaders)	Saving any cookies to cookie store. These cookies are retrieved from response header of the Http response from the given URL. Its called every time a response is received.

CookieManager

- The CookieManager class provides a precise implementation of CookieHandler. This separates the storage of cookies from the policy surrounding accepting and rejecting cookies. A CookieManager is initialized with a CookieStore and a CookiePolicy. The CookieStore manages storage, and the CookiePolicy object makes policy decisions on cookie acceptance/rejection.



Cookie Manager

Constructor Detail

- Syntax:

  public CookieManager()

- Create a new cookie manager
- This constructor will create new cookie manager with default cookie store and accept policy. The effect is same as CookieManager(null, null).

Cookie Manager

Constructor Detail

- Syntax:

public CookieManager(CookieStore store, CookiePolicy cookiePolicy)

Create a new cookie manager with specified cookie store and cookie policy.

Parameters:

store - a CookieStore to be used by cookie manager. if null, cookie manager will use a default one, which is an in-memory CookieStore implmentation.

cookiePolicy - a CookiePolicy instance to be used by cookie manager as policy callback. if null, ACCEPT_ORIGINAL_SERVER will be used.

Cookie Manager

Method	Action Performed
getCookieStore()	This method will retrieve the current cookie store.
setCookiePolicy(CookiePolicy cookiePolicy)	This method will set the cookie policy of this cookie manager.
get(URI uri, Map<String, List<String>> requestHeaders)	This method will get all the applicable cookies from a cookie cache for the specified URI in the request header.
put(URI uri, Map<String, List<String>> responseHeaders)	This method will set all the applicable cookies

Cookie Manager

- getCookieStore() method retrieves the current cookie store and returns the cookie store currently used by the cookie manager.

  Syntax:

  public CookieStore getCookieStore() ;

Cookie Manager

setCookiePolicy() method sets the cookie policy of the cookie manager. An instance of CookieManager will have cookie policy ACCEPT_ORIGINAL_SERVER by default. This method can be called to set another cookie policy.

Syntax:

- public void setCookiePolicy(CookiePolicy cookiePolicy) ;

Cookie Manager

put() method sets all the applicable cookies, for example, response header fields that are named Set-Cookie2, present in the response headers into a cookie cache.

Syntax:

public void put(URI uri, Map<String,List<String>> responseHeaders)

Cookie Manager

- get() method gets all the applicable cookies from a cookie cache for the required URI within the request header. It is up to the implementation to require under consideration the URI and therefore the cookies attributes and security settings to work out which of them should be returned. HTTP's protocol implementers should confirm that this method is called after all request headers associated with choosing cookies are added and before the request is sent.
- Syntax:
- public Map<String,List<String>> get(URI uri, Map<String,List<String>> requestHeaders)

Cookie Manager

To store or return cookies, you need to enable it:

CookieManager cookiemanager = new CookieManager();

cookiehandler.setDefault(cookiemanager);

Only two line of code are required if you want to receive cookies from a site or send them back. In case you want to be more careful about cookies there are predefined policies which you can use:

CookiePolicy.ACCEPT_ALL all cookies are allowed

CookiePolicy.ACCEPT_NONE no cookies are allowed

CookiePolicy.ACCEPT_ORIGNAL_SERVER only first party cookies are allowed

HTTP Cookie

An **HTTP cookie** (web cookie, browser cookie) is a small piece of data that a server sends to a user's web browser. The browser may store the cookie and send it back to the same server with later requests. They are helpful for session management, user personalization, and tracking.

HTTP Cookie

- **Session management:**

Session management facilitates secure interactions between the user and some service or application. As the user continues to interact with the web application, they submit requests that help track the user's status during the session.

Examples: Logins, auto-filled form fields, shopping lists

- **User personalization:**

User personalization retains user preferences and settings for reapplication upon user login or application start. Settings are saved and synchronized with a central database to help users return to preferred settings used during their first application interaction.

Examples: User preferences, themes, language

- **Tracking:**

Tracking records and analyzes users' web browsing habits and finds the number and type of pages the user visits. Details include time spent on page and page sequence across a user's sessions. Due to the sensitive information behind tracking, users should be aware of vulnerabilities from visiting insecure web pages.

Example: Ad tracking