1/5/22

**A practical, Complete Tutorial on HTTP cookies**

<https://www.valentinog.com/blog/cookies/>

[**Cookies**](http://il.php.net/manual/en/features.cookies.php) and [**sessions**](http://il.php.net/manual/en/features.sessions.php) are both ways to preserve the application's state between different requests the browser makes.

Cookies

Cookies are small bits of data, (**maximum of 4KB** long), which hold data in a key=value pairs:

name=value; name2=value2

These are set either by [**JavaScript**](https://developer.mozilla.org/en/DOM/document.cookie), or via the server using an [**HTTP header**](http://il.php.net/manual/en/function.setcookie.php).

Cookies have an expiry datetime set, example using HTTP headers:

**Set-Cookie:** name2=value2**;** **Expires=**Wed, 19 Jun 2021 10:18:14 GMT

Which would cause the browser to set a cookie named name2 with a value of value2, which would expire in about 9 years.

Cookies are considered **highly insecure** because the user can easily manipulate their content.

**Cookies are usually used to preserve login state**, where a username and a special hash are sent from the browser, and the server checks them against the database to approve access.

**Cookies are also often used in sessions creation**.

Sessions

Sessions are slightly different. Each user gets a **session ID, which is sent back to the server for validation either by cookie**or by **GET variable**.

Sessions are usually short-lived, which makes them ideal in saving temporary state between applications. **Sessions also expire once the user closes the browser**.

Sessions are considered more secure than cookies because the variables themselves are kept on the **server**. Here's how it works:

1. Server opens a session (sets a cookie via HTTP header)
2. Server sets a session variable.
3. Client changes page
4. Client sends all cookies, along with the session ID from step 1.
5. Server reads session ID from cookie.
6. Server matches session ID from a list in a database (or memory etc).
7. Server finds a match, reads variables which are now available on $\_SESSION superglobal.

If PHP does not find a match, it will start a new session, and repeat the steps from 1-7.

You can store sensitive information on a session because it is kept on the server, but **be aware that the session ID can still be stolen** if the user, let's say, logged in over an insecure WiFi. (An attacker can sniff the cookies, and set it as its own, **he won't see the variables themselves, but the server will identify the attacker as the user**).

1. 2011

The concept is **storing persistent data across page loads** for a web visitor. Cookies store it directly on the client. Sessions use a cookie as a key of sorts, to associate with the data that is stored on the server side.

It is preferred to use sessions because the actual values are hidden from the client, and you control when the data expires and becomes invalid. If it was all based on cookies, a user (or hacker) could manipulate their cookie data and then play requests to your site.

**session ID is stored as a cookie on the client machine**, and is then matched up server-side with the session data. The server does not typically control sessions via IP address, rather through a cookie value.

Basic ideas to distinguish between those two.

**Session:**

1. IDU is stored on server (i.e. server-side)
2. Safer (because of 1)
3. Expiration can not be set, session variables will be expired when users close the browser. (nowadays it is stored for 24 minutes as default in php)

**Cookies:**

1. IDU is stored on web-browser (i.e. client-side)
2. Not very safe, since hackers can reach and get your information (because of 1)
3. Expiration can be set (see [setcookies()](http://php.net/manual/en/function.setcookie.php) for more information)

Session is preferred when you need **to store short-term information/values**, such as variables for calculating, measuring, querying etc.

Cookies is preferred when you need to **store long-term information/values**, such as user's account (so that even when they shutdown the computer for 2 days, their account will still be logged in). I can't think of many examples for cookies since it isn't adopted in most of the situations.

**SESSIONS ENDS WHEN USER CLOSES THEIR BROWSER**,

Sessions are stored in a file your php server will generate. To remember which file is for which user, php will **also set a cookie on the user's browser** that holds this session file id so in their next visit php will read this file and reload the session.

**Sessions are server-side files that contain user information, while Cookies are client-side files that contain user information. Sessions have a unique identifier that maps them to specific users. This identifier can be passed in the URL or saved into a session cookie**.

Most modern sites use the second approach, **saving the identifier in a Cookie instead of passing it in a URL** (which poses a security risk). You are probably using this approach without knowing it, and by deleting the cookies you effectively erase their matching sessions as you remove the unique session identifier contained in the cookies

Translating that to Web Servers: The server will store the pertinent information in the session object, and create a session ID which it will send back to the client in a cookie. When the client sends back the cookie, the server can simply look up the session object using the ID. So, if you delete the cookie, the session will be lost.

One other alternative is for the server to use URL rewriting to exchange the session id.

Suppose you had a link - www.myserver.com/myApp.jsp You could go through the page and rewrite every URL as www.myserver.com/myApp.jsp**?sessionID=asdf** or even www.myserver.com/asdf/myApp.jsp and exchange the identifier that way. This technique is handled by the web application container and is usually turned on by setting the configuration to use cookieless sessions.

Cookies and session both store information about the user (to make the HTTP request stateful) but the difference is that cookies store information on the client-side (browser) and sessions store information on the server-side. A cookie is limited in the sense that it stores information about limited users and only stores limited content for each user. A session is not limit in such a way.

1. 2019

The appropriate choice

**Sessions use a cookie!** Session data is stored on the server side, but a UID is stored on client side in a cookie. It allows the server to match a given user with the right session data. UID is protected and hard to hack, but not invulnerable. For sensitive actions (changing email or resetting password), do not rely on sessions neither cookies : ask for the user password to confirm the action.

**Sensitive data** should never be stored in cookies (emails, encrypted passwords, personal data ...). Keep in mind the data are stored on a foreign computer, and if the computer is not private (classroom or public computers) someone else can potentially read the cookies content.

**Remember-me data must be stored in cookies**, otherwise data will be lost when the user closes the browser. However, don't save password or user personal data in the 'remember-me' cookie. Store user data in database and link this data with an encrypted pair of ID / key stored in a cookie.

**Must persistent data remain when the user closes the browser ?**

* If the answer is **yes**, use **cookies**.
* If the answer is **no**, use **sessions**.

Stateless applications

* Web application servers are generally "stateless":
  + Each HTTP request is independent; server can't tell if 2 requests came from the same browser or user.
  + Web server applications maintain no information in memory from request to request (only information on disk survives from one request to another).
* Statelessness not always convenient for application developers: need to tie together a series of requests from the same user.

Browser cookies

* Cookie basics:
  + The first time a browser connects with a particular server, there are no cookies.
  + When the server responds it includes a Set-Cookie: header that defines a cookie.
  + Each cookie is just a name-value pair.
  + In the future whenever the browser connects with the same server, it includes a Cookie: header containing the name and value, which the server can use to connect related requests.
* What's in a cookie?
  + Name and data.
    - **Data size limited by browsers (typically < 4 KB).**
    - A server can define multiple cookies with different names, but browsers limit **the number of cookies per server (around 50)**.
  + Domain for this cookie: server, port (optional), URL prefix (optional). The cookie is only included in requests matching its domain.
  + Expiration date: browser can delete old cookies.

Sessions

* Cookies are used by the server to **implement *sessions***:
  + A pool of data related to an active connection (one browser instance).
* Typically the cookie for an application contains an identifier for a session.
* Web frameworks like Rails do most of the work of managing sessions and cookies:
  + Rails provides session, a hash-like object in which you can store anything you like
    - Data will be available in all future requests from the same browser.
  + Rails automatically checks for a session cookie at the start of each request:
    - Cookie exists? use it to find session data
    - No cookie? Create new session, new cookie
  + End of each request: save session data where it can be found by future requests.
* Managing session state:
  + Approach #1: just keep state in main memory
  + Approach #2: store session state in files on disk
  + Approach #3: store session state in a database
  + Most frameworks allow you to control session storage:
    - Provide an object that saves and restores session data.

Cookies vs Cache

Anyone who browses the internet has heard of cookies and cache. They are messages and memories of who you are, what you did online, what choices you made, how you can be recognized the next time you log in, and how that login can be speeded up. But while the terms are sometimes used interchangeably, they are not identical in form or function.

Let us see what cookies and cache mean.

[](https://kingpinbrowser.com/sponsorship/)

## What is a Cookie?

A browser cookie or simply a cookie is a message. This message is stored in a small file. This file is like your ID.

When you open your browser like Chrome, Safari, or Kingpin and visit a website, the site sends small files about you to your browser which then stores them on your device. This is the website’s way of remembering you (your device) and the actions you performed.

## What is a Cache?

It is an imprint or memory. A memory of some of the elements of sites you visited last. For example, if the site has a video, a cached video will be downloaded on your browser/device. This is the website’s way of quickening the webpage display.

What’s the difference between caches and cookies?  
Caches and cookies look and work differently. We look at the differences:

### 1. Data type

Caches are copies of website data while cookies are information capsules about client behavior. The client here is your machine, not you specifically or anyone else who uses the same device.

### 2. Identification

Cookies, also called HTTP cookies or web cookies are usually text files. Some examples would be your login data, browsing ID, location, your IP address, and the time you spent on them or the information you entered. All cookie files end in .txt and that’s how you identify them.

Caches do not have a single type of extensions. They may be found in the temporary folders on your device and the Settings section on your browser.

The cache includes scripts, graphics, images, animation, GIFs, and audio or video content of websites that you visited the last time. Because these elements are already downloaded on your laptop, the next time you load that site, it would load faster. It would consume less data and would be lighter. The cache is, after all, SRAM (Static Random Access Memory) and it is recalled in a twinkling.

### 3. Storage location and movement

Caches are stored on your device. So are cookies. But follow different types of routes. Site cookies can travel in two directions: from the server to browser and back the same way. HTTP cache travels only in a single direction – server to browser/device.

### 4. Validity

Cookies are erased automatically on some browsers after their expiration date but caches do not. They have to be physically removed.

### 5. Function

Cookies are related to tracking user behavior while caches are related to website loading speed, navigation, easy display, and plugins.

### 6. Size

The cached data vary widely in size. Each page you load sends some memory files over to your browser/device. Since they are copies of web content, they run into gigabytes. Cookies, on the other hand, are fairly tiny, taking three to four KBs on average. But they come in lots.

### 7. Usage

Marketers often use cookies to understand what users like on a site. Based on these files, they show relevant ads to users. So when you log out and come back, cookies are sent back to the server and the site has an Aha moment! It recognizes you and is glad to welcome you back. Even when you are on another site, for example, Facebook or Instagram, businesses still display ads you might like. All based on cookies. And cookies of different sites can interact.

Caches do no such thing. They are simply content to sit on your browser/device and ease the pressure on your server.

## What is the difference between cookies and history?

Browsing history is a register of the number of sites you’ve visited. It also shows the page titles and the time of the visit. Cookies show other facts like passwords or site preferences. Together with cache and auto-fill forms, they form browsing data.

## How cookies and caches work

It’s not all tech mumbo-jumbo. Let’s see how cookies and caches work on something almost all users do – online shopping.

You log into your account. You’re grocery shopping for the month. You look through the catalog and start putting items in your cart. Cookies remember your actions and all the images you checked were made into small bite-sized caches. So when you get back a couple of days later to pay, you find your cart is still loaded and your login is quicker.

## What does [clearing cookies and caches](https://kingpinbrowser.com/blog/best-browser-cleaners/) do?

Experts often suggest that browsing data should be cleared off at the end of a browsing session. [Cookies and caches are likely used to track users.](https://kingpinbrowser.com/blog/computer-monitored/) If you have online privacy concerns, you might want to consider deleting these files when you exit browsing. Saved settings might invite malware and get hackers interested. On the downside, sites may take longer to load and have to figure out your choices all over again.

## Why is it important to clear your cache?

Cache eats into your device storage. It latches on to the RAM and slows down your device. Now although sites load faster your laptop or mobile performs other functions at snail’s pace. Experts recommend clearing off cached or accumulate data regularly to boost up your machine.

Privacy issues may also arise due to the presence of third-party cookies. These files threaten your accounts with a security breach. Your personal and sensitive data is compromised, especially if you perform online transactions with your bank or click on unsuspecting links accidentally.

All major browsers like Firefox or Chrome offer users the choice of manually removing their entire browsing data. While they don’t automatically remove cookies from your device, you can manually visit the cookie folder and delete them.

[Privacy-focused browsers such as Kingpin](https://kingpinbrowser.com/) don’t store any browser history so you don’t have to manually clear it regularly. Also, the Kingpin browser erases cookies automatically as soon as the browsing session is over. If you’re a privacy-conscious user, try a secure browser that doesn’t store your browsing history or cookies.

# **What are Cookies?**

Cookies are small files that contain information useful to a web site — such as password, preferences, browser, IP Address, date and time of visit, etc. Every time the user loads the website, the browser sends the cookie back to the server to notify the website of the user’s previous activity.

**Cookies have a certain life span defined by their creators and it expires after the fixed time span.**

Cookies often track information like how frequently the user visits, what are the times of visits, what banners have been clicked on, what button clicked, user preferences, items in shopping cart, etc. This allows the site to present you with information customized to fit your needs.

Cookies are usually used to store information needed for shorter periods. Cookies was first introduced by Netscape. In those earlier stages cookies did not receive good acceptance, since rumors said it might hack your personal data. Later people realized that cookies are actually harmless, and now they are highly accepted.

# **What are Caches?**

A web cache (or HTTP cache) is an information technology for the temporary storage (caching) of web documents, such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. **Cache is just a collection of data downloaded to help display a web page.**

A web cache system stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met. A web cache system can refer either to an appliance, or to a software.

For example, when you open websites with large pictures and video’s, it might take some time to load the website. The web browser stores the site contents like the images, videos, audio etc. on your computer. So the next time you load the same website you will find it loading faster.

# **What’s The Difference Between Cache And Cookies**

Although cookies and cache are two ways to store data on client’s machine, but there are difference between cache and cookies and they serve different purposes.  
• Cookie is used to store information to track different characteristics related to user, while cache is used to make the loading of web pages faster.  
• Cookies stores information such as user preferences, while cache will keep resource files such as audio, video or flash files.  
• Typically, cookies expire after some time, but cache is kept in the client’s machine until they are removed manually by the user.

## What is the Browser Cache?

In general terms, “caching” something means temporarily storing it in a spot that makes for easier/faster retrieval.

There are a lot of different ways that you can cache your WordPress website, but we’re obviously focused on one specific implementation – the **browser cache**.

Let’s start at the beginning – your site has a lot of static files that won’t change visit-to-visit. One common example is your site’s logo. You probably won’t change your logo that often, and your logo is likely the same on all or most of your site.

So let’s say a visitor views three separate pages on your site – it would be a waste to force that visitor to download the exact same logo file three times in a row, right?

The browser cache lets you avoid that situation by storing these static files on a visitor’s local computer. So with the browser cache, it would work like this instead:

* **First visit**– the visitor’s browser downloads the logo file from your server and saves it locally in the browser cache.
* **Second and third visits**– the visitor’s browser loads the logo from the visitor’s local storage (the browser cache), instead of downloading the logo again.

And as a result, your website loads faster, which is a great thing.

To control how long a visitor’s browser stores certain types of files, you can set expiration dates. For example, you can tell a visitor’s browser “store all JPEG images for 4 months, but only store MP4 videos for 1 month”.

Additionally, if you do need to change a file before the expiration is up (like your logo), you can employ a technique called [**cache busting**](https://www.keycdn.com/support/what-is-cache-busting) to force visitors’ browsers to download the most recent version of a file.

Because browser caching is a great performance tip, WP Rocket automatically adds browser caching to your WordPress site as soon as you activate the WP Rocket plugin. You can [**learn more about this feature here**](https://docs.wp-rocket.me/article/80-browser-caching).

## What is a Cookie?

According to the dictionary, a cookie is a “small sweet cake, typically round, fl”…wait, that doesn’t sound right.

Let’s try again…

A cookie is a small file that’s stored on a visitor’s device and contains data specific to a particular client. “Client”, in this case, just means the device of the person who’s visiting.

Cookies help you store useful information about a visitor, like their login information/authentication so that they don’t have to manually log in each time, or the items in their shopping cart. Or, you can also use cookies to track and identify visitors, even between different website visits.

Cookies are only used to store text-based data, like IP addresses, session IDs, visit histories, etc. – you can’t use them to store an image, as the browser cache allows for.

Additionally, unlike a browser cache, which is a one-way transfer (from the server to the local cache), a client’s web browser will send the cookie to your web server on each visit – so the information can go from both:

* Server → client
* Client → server

There are two types of cookies:

* **Persistent cookie**– though this type of cookie has an expiration date, it stays on a visitor’s local machine while active and “persists” between visits to your website. It lets you identify a visitor even if they leave your site and come back.
* **Session cookie**– this type of cookie is stored in memory and never saved to your visitor’s local machine. It’s active during a visit, but as soon as your visitor closes their browser, a session cookie is permanently gone.

## Browser Cache vs Cookies: What Are They Used For?

At this point, you probably have some idea about the difference in use between the browser cache vs cookies, but let’s cover the point in a little more detail to make sure.

**Browser cache** lets you store certain static files – including images, videos, CSS/JavaScript, etc. This is a one-way relationship – those files don’t communicate back to your server once they’re stored. Additionally, the browser cache doesn’t identify specific users – it treats all users the same.

The browser cache helps you **speed up your site**and lessen the load on your server – that’s all it’s used for.

**Cookies** are small text-based files that let you track, identify, or otherwise store information that’s unique to each visitor. This is a two-way relationship, where your server is able to read the information from the cookie.

Cookies help you provide a better user experience, like recognizing that a user is logged-in or identifying a visitor to show them the items in their shopping cart at an eCommerce store.

Or, cookies can help you track and identify specific visitors, like setting a cookie so that you don’t display an email opt-in popup after a visitor has already subscribed to your email list.

Finally, you can even use cookies to help with off-site functionality. For example, setting a retargeting cookie so that you can display ads to your site’s visitors even when they’re on other sites.





           