Cookies

A cookie is a small piece of text data set by Web server that resided on the client’s machine.

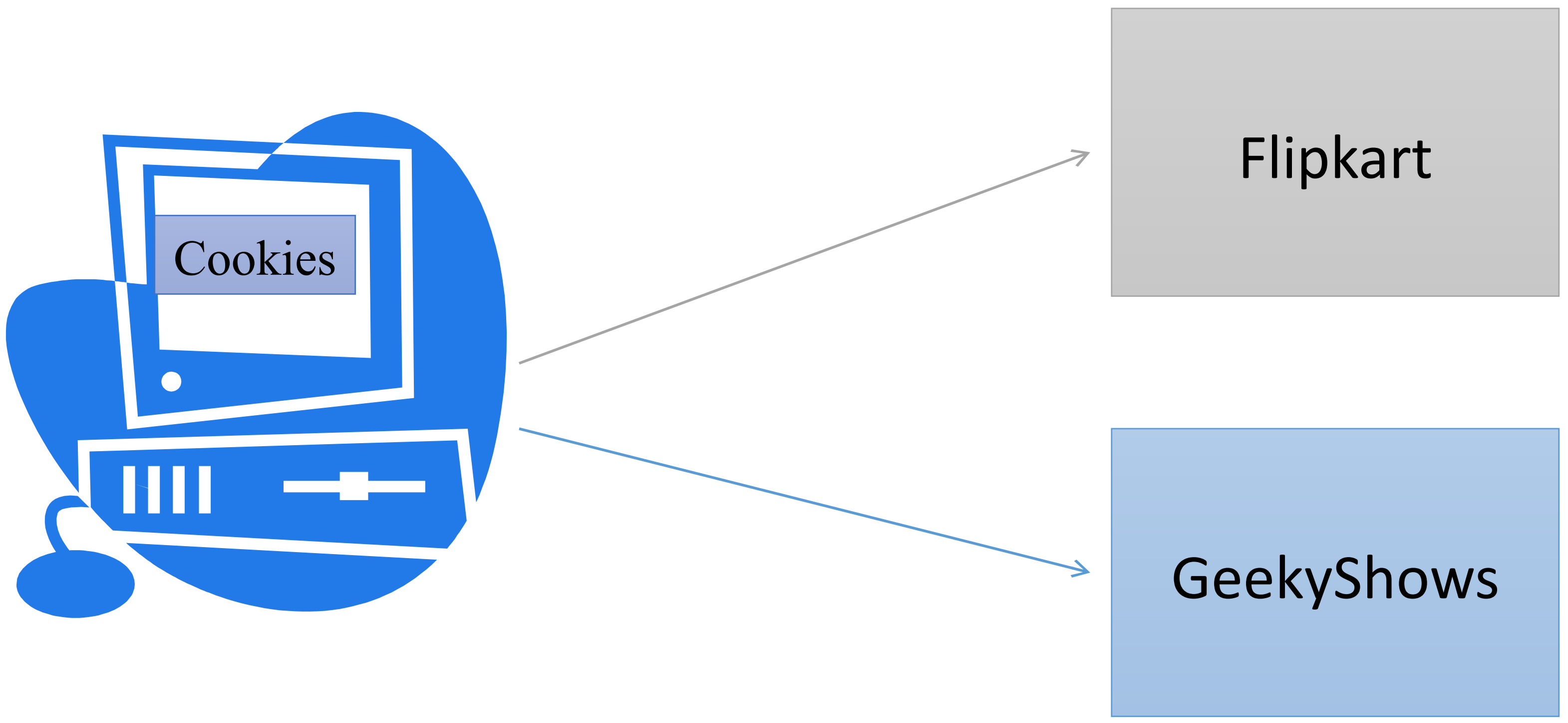
Once it’s been set, the client automatically returns the cookie to the web server with each request that it makes.

This allows the server to place value it wishes to ‘remember’ in the cookie, and have access to them when creating a response.

How Cookie Works

Note
No 12345
New Bedsheet
Room No. 45
Food Paneer

How Cookie Works



Django Cookies

HttpRequest.COOKIES - A dictionary containing all cookies. Keys and values are strings.

Creating Cookies

set_cookie() – set_cookie() is used to set/create/sent cookies.

Syntax: - HttpResponse.set_cookie(key, value='', max_age=None, expires=None, path='/', domain=None, secure=False, httponly=False, samesite=None)

Example: - set_cookie(“name”, “sonam”)

key − This is the name of the cookie.

value − This sets the value of cookie. This value is stored on the clients computer.

name and value are required to set cookie.

When you omit the optional cookie fields, the browser fills them in automatically with reasonable defaults.

Creating Cookies

max_age - It should be a number of seconds, or None (default) if the cookie should last only as long as the client's browser session. If expires is not specified, it will be calculated.

Example:- set_cookie("name", "sonam", max_age=60*60*24*10)   // 10 days

expires - It describes the time when cookie will be expire. It should either be a string in the format "Wdy, DD-Mon-YY HH:MM:SS GMT" or a datetime.datetime object in UTC. If expires is a datetime object, the max_age will be calculated.

Example:- set_cookie("name", "sonam", expires= datetime.utcnow() + timedelta(days=2))

path - Path can be / (root) or /mydir (directory).

Example: - set_cookie("name", "sonam", "/ ")

Example: - set_cookie("name", "sonam", "/home")

Creating Cookies

domain - Use domain if you want to set a cross-domain cookie. For example, domain="example.com" will set a cookie that is readable by the domains www.example.com, blog.example.com, etc. Otherwise, a cookie will only be readable by the domain that set it.

Example: - set_cookie (“name”, “sonam”, domain=“geekyshows.com”)

note.geekyshows.com

code.geekyshows.com

Example: - set_cookie (“name”, “sonam”, “code.geekyshows.com”)

secure - Cookie to only be transmitted over secure protocol as https. When set to TRUE , the cookie will only be set if a secure connection exists.

Example: - set_cookie(“name”, “sonam”, max_age=60*60*24*10, path=“/ ”, domain=“geekyshows.com”, secure=True)

<u>Creating Cookies</u>

httponly - Use httponly=True if you want to prevent client-side JavaScript from having access to the cookie.

HttpOnly is a flag included in a Set-Cookie HTTP response header. It’s part of the RFC 6265 standard for cookies and can be a useful way to mitigate the risk of a client-side script accessing the protected cookie data.

Example: - set_cookie(“name”, “sonam”, max_age=60*60*24*10, httponly=True)

samesite - Use samesite='Strict' or samesite='Lax' to tell the browser not to send this cookie when performing a cross-origin request. SameSite isn’t supported by all browsers, so it’s not a replacement for Django’s CSRF protection, but rather a defense in depth measure.

RFC 6265 states that user agents should support cookies of at least 4096 bytes. For many browsers this is also the maximum size. Django will not raise an exception if there’s an attempt to store a cookie of more than 4096 bytes, but many browsers will not set the cookie correctly.

Example: - set_cookie(“name”, “sonam”, samesite=‘’Strict)

Reading/Accessing Cookie

HttpRequest.COOKIES - A dictionary containing all cookies. Keys and values are strings.

Syntax:- request.COOKIES[‘key’];

Example: - request.COOKIES[‘name’];

Syntax:- request.COOKIES.get(‘key’, default)

Example: - request.COOKIES.get(‘name’)

Example: - request.COOKIES.get(‘name’, “Guest”)

Replace/Append Cookies

When we assign a new value to cookie, the current cookie are not replaced. The new cookie is parsed and its name-value pair is appended to the list. The exception is when you assign a new cookie with the same name (and same domain and path, if they exist) as a cookie that already exists. In this case the old value is replaced with the new.

Examples:-

- set_cookie("name", "sonam")
  set_cookie("name", "rahul")

  Replace

- set_cookie("name", "sonam")
  set_cookie("lname", "jha")

  Append

Deleting Cookies

HttpResponse.delete_cookie(key, path='/', domain=None) – This method is used to delete the cookie based on given key with same domain and path, if they were set, otherwise the cookie may not be deleted.

Example: - delete_cookie('name')

Creating Signed Cookies

HttpResponse.set_signed_cookie(key, value, salt='', max_age=None, expires=None, path='/', domain=None, secure=False, httponly=False, samesite=None) –

It is similar to set_cookie(), but cryptographic signing the cookie before setting it. Use in conjunction with HttpRequest.get_signed_cookie().

You can use the optional salt argument for added key strength, but you will need to remember to pass it to the corresponding HttpRequest.get_signed_cookie() call.

Getting Signed Cookies

HttpRequest.get_signed_cookie(key, default=RAISE_ERROR, salt='', max_age=None) –

It returns a cookie value for a signed cookie, or raises a django.core.signing.BadSignature exception if the signature is no longer valid.

If you provide the default argument the exception will be suppressed and that default value will be returned instead.

The optional salt argument can be used to provide extra protection against brute force attacks on your secret key. If supplied, the max_age argument will be checked against the signed timestamp attached to the cookie value to ensure the cookie is not older than max_age seconds.

Cookies Security Issues

- Can misuse Client Details
- Can track User
- Client Can Delete Cookies
- Client can Manipulate Cookies

Cookies Limitation

- Each cookie can contain 4096 bytes Data
- Cookies can be stored in Browser and server
- It is sent with each request