Session Management with Cookies

Malintha Adikari

1

Cookies

Cookies

- HTTP cookie (web cookie, Internet cookie, browser cookie) is a small piece of data sent from a website and stored on the user's computer by the web browser.
- Cookies were designed to be a reliable mechanism for websites to remember stateful information or to record the user's browsing activity.
- Can also be used to remember arbitrary pieces of information that the user previously entered into form fields such as names, addresses, passwords, etc.

Cookies

- Usually chosen and first sent by the web server, and stored by the web browser.
- The browser then sends them back to the server with every request, introducing states (memory of previous events) into stateless HTTP transactions.
- Each retrieval of a web page would be an isolated event, largely unrelated to all other page views made by the user on the website without cookies.
- Can also be set by the client using a scripting language such as JavaScript.

Authentication Cookie

- The most common method used by web servers to know whether the user is logged in or not and which account they are logged in with.
- Without such a mechanism, the site would not know whether to send a page containing sensitive information, or require the user to authenticate themselves by logging in.
- The security of an authentication cookie depends on the security of the issuing website and the user's web browser, and on whether the cookie data is encrypted.

Session Cookie

- A session cookie (in-memory cookie or transient cookie) exists only in temporary memory while the user navigates the website.
- Web browsers normally delete session cookies when the user closes the browser.
- Session cookies do not have an expiration date assigned to them

Persistent Cookie

- Persistent cookie expires at a specific date or after a specific length of time.
- For the cookie's entire lifespan, its information will be transmitted to the server every time the user visits the website
- Referred to as tracking cookies because they can be used by advertisers to record information about a user's web browsing habits over an extended period of time.
- These cookies are reset if the expiration time is reached or the user manually deletes the cookie.

Cookie Structure

A cookie consists of the following components:

1. Name
2. Value
3. Zero or more attributes (name/value pairs). Attributes store information such as the cookie’s expiration, domain, and flags (ex: `Secure`, `HttpOnly`).

Uses of Cookies

Session management

- Cookies were introduced to provide a way for users to record items they want to purchase as they navigate throughout a website.
- The server sends a cookie to the client that contains a unique session identifier.
- Cookies are sent to the server with every request the client makes to keep track of which user is assigned to which shopping cart
- Session identifier will be sent back to the server every time the user visits a new page on the website

Uses of Cookies

Session management

- Another popular use of cookies is for logging into websites.
- When the user visits a website's login page, the web server sends the client a cookie containing a unique session identifier.
- When the user successfully logs in, the server remembers that that particular session identifier has been authenticated, and grants the user access to its services..
- Session cookies also help to improve page load times, since the amount of information in a session cookie is small and requires little bandwidth

Uses of Cookies

Personalization

- Cookies can be used to remember information about the user in order to show relevant content to that user over time.
- Ex: Web server send a cookie containing the username last used to log into a website so that it may be filled in automatically the next time the user logs in.

Uses of Cookies

Personalization

- Users select their preferences by entering them in a web form and submitting the form to the server.
- The server encodes the preferences in a cookie and sends the cookie back to the browser.
- Every time, the server can personalize the page according to the user's preferences.
- Ex: Google search engine once used cookies to allow users to decide how many search results per page they wanted to see.

Uses of Cookies

Tracking

- Tracking cookies are used to track users' web browsing habits.
- This can also be done to some extent by using the IP address of the computer requesting the page or the referer field of the HTTP request header, but cookies allow for greater precision.

Uses of Cookies

Tracking

1. If the user requests a page of the site, but the request contains no cookie, the server presumes that this is the first page visited by the user. So the server creates a unique identifier and sends it as a cookie back to the browser together with the requested page.
2. From this point on, the cookie will automatically be sent by the browser to the server every time a new page from the site is requested. The server sends the page as usual, but also stores the URL of the requested page, the date/time of the request, and the cookie in a log file.

By analyzing this log file, it is then possible to find out which pages the user has visited, in what sequence, and for how long.

Requirements

The cookie specifications require that browsers meet the following requirements in order to support cookies:

- Can support cookies as large as 4,096 bytes in size.
- Can support at least 50 cookies per domain.
- Can support at least 3,000 cookies in total.

Setting Cookies

- Cookies are set using the Set-Cookie HTTP header, sent in an HTTP response from the web server.
- This header instructs the web browser to store the cookie and send it back in future requests to the server.
- As an example, the browser sends its first request for the homepage of the `www.example.org` website:

```
GET /index.html HTTP/1.1
Host: www.example.org
...
```

Setting Cookies

The server responds with two Set-Cookie headers:

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT
...
```

The server's HTTP response contains the contents of the website's homepage an and instructs the browser to set two cookies. .

Setting Cookies

```
HTTP/1.0 200 OK
Content-type: text/html
Set-Cookie: theme=light
Set-Cookie: sessionToken=abc123; Expires=Wed, 09 Jun 2021 10:18:14 GMT
```

- The first, "theme", is considered to be a *session cookie*, since it does not have an `Expires` or `Max-Age` attribute.
- The second, "sessionToken" is a *persistent cookie*, since it contains an `Expires` attribute, which instructs the browser to delete the cookie at a specific date and time.

Setting Cookies

- Next, the browser sends another request to visit the `spec.html` page on the website.
- This request contains a `Cookie` HTTP header, which contains the two cookies that the server instructed the browser to set:

```
GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: theme=light; sessionToken=abc123
...
```

- This way, the server knows that this request is related to the previous one.

Setting Cookies

- The value of a cookie can be modified by the server by including a Set-Cookie header in response to a page request.
- The browser then replaces the old value with the new value.
- The value of a cookie may consist of any printable ASCII character
- The name of a cookie excludes the same characters, as well as `=`, since that is the delimiter between the name and value.
- In addition to a name and value, cookies can also have one or more attributes.
- Cookie attributes are used by browsers to determine when to delete a cookie, block a cookie or whether to send a cookie to the server.

Cookies Attributes

Domain and path

- The `Domain` and `Path` attributes define the scope of the cookie.
- They essentially tell the browser what website the cookie belongs to.
- For security reasons, cookies can only be set on the current resource's top domain and its sub domains, and not for another domain and its sub domains.
- For example, the website `example.org` cannot set a cookie that has a domain of `foo.com`
- If a cookie's `Domain` and `Path` attributes are not specified by the server, they default to the domain and path of the resource that was requested.

Cookies Attributes

Domain and path

- Ex: The HTTP request was sent to a webpage within the docs.foo.com subdomain:

```
HTTP/1.0 200 OK
Set-Cookie: LSID=DQAAAK…Eaem_vYg; Path=/accounts; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
Set-Cookie: HSID=AYQEVn…DKrdst; Domain=.foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; HttpOnly
Set-Cookie: SSID=Ap4P…GTEq; Domain=foo.com; Path=/; Expires=Wed, 13 Jan 2021 22:23:01 GMT; Secure; HttpOnly
…
```

- LSID, has no Domain attribute, and has a Path attribute set to /accounts, tells the browser to use the cookie only when requesting pages contained in docs.foo.com/accounts (the domain is derived from the request domain).
- The other two cookies, HSID and SSID, would be used when the browser requests any subdomain in .foo.com on any path

Cookies Attributes

Expires and Max-Age

The `Expires` attribute defines a specific date and time for when the browser should delete the cookie.

The date and time are specified in the form `Wdy, DD Mon YYYY HH:MM:SS GMT`, or in the form `Wdy, DD Mon YY HH:MM:SS GMT`

Alternatively, the `Max-Age` attribute can be used to set the cookie's expiration as an interval of seconds in the future, relative to the time the browser received the cookie.

Cookies Attributes

Expires and Max-Age

Ex: three `Set-Cookie` headers that were received from a website after a user logged in:

```
HTTP/1.0 200 OK
Set-Cookie: lu=Rg3vHJZnehYLjVg; Expires=Tue, 15 Jan 2013 21:47:38 GMT; Path=/; Domain=.example.com; HttpOnly
Set-Cookie: made_write_conn=1295214458; Path=/; Domain=.example.com
Set-Cookie: reg_fb_gate=deleted; Expires=Thu, 01 Jan 1970 00:00:01 GMT; Path=/; Domain=.example.com; HttpOnly
```

- The first cookie, `lu`, is set to expire sometime on 15 January 2013. It will be used by the client browser until that time.
- The second cookie, `made_write_conn`, does not have an expiration date, making it a session cookie. It will be deleted after the user closes their browser.
- The third cookie, `reg_fb_gate`, has its value changed to "deleted", with an expiration time in the past.

Cookies Attributes

Secure and HttpOnly

- The Secure and HttpOnly attributes do not have associated values.
- Presence attribute names indicates that their behaviors should be enabled.
- The Secure attribute is to keep cookie communication limited to encrypted transmission, directing browsers to use cookies only via secure/encrypted connections.
- The HttpOnly attribute directs browsers not to expose cookies through channels other than HTTP (and HTTPS) requests.
- The cookie cannot be accessed via client-side scripting languages, and therefore cannot be stolen easily via cross-site scripting

2

Session

Session Management

- HTTP protocol and Web Servers are stateless, for web server every request is a new request to process and they can’t identify if it’s coming from client that has been sending request previously
- But we should know who the client is and process the request accordingly.
- Ex: Shopping cart application should know who is sending the request to add an item and in which cart the item has to be added

Session Management

- Session is a conversational state between client and server and it can consists of multiple request and response between client and server.
- Only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response.

Session Management

Providing unique identifier in request and response.

User Authentication –

- Common way where we can provide authentication credentials from the login page and then pass the authentication information between server and client to maintain the session.
- Not very effective method because it wont work if the same user is logged in from different browsers.

Session Management

Providing unique identifier in request and response.

HTML Hidden Field –

- Create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user and keep track of the session.
- This method can’t be used with links because it needs the form to be submitted every time request is made from client to server with the hidden field.
- Not secure because we can get the hidden field value from the HTML source and use it to hack the session.

Session Management

Providing unique identifier in request and response.

URL Rewriting –

- Can append a session identifier parameter with every request and response to keep track of the session.
- Very tedious because we need to keep track of this parameter in every response and make sure it’s not clashing with other parameters.

Session Management

Providing unique identifier in request and response.

Cookies –

- When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session.
- We can maintain a session with cookies but if the client disables the cookies, then it won’t work.

Session Management

Providing unique identifier in request and response.

Session Management API –

Built on top of above methods for session tracking. Some of the major disadvantages of all the above methods are:

- Most of the time we don’t want to only track the session, we have to store some data into the session that we can use in future requests.
- All the above methods are not complete in themselves, all of them won’t work in a particular scenario. So we need a solution that can utilize these methods of session tracking to provide session management in all cases.

Demo

1. Session Management with Cookies

   http://localhost:8080/ServletCookieApp/

2. Session Management with HTTP Session Interface

   http://localhost:8080/ServletHTTPSessionApp/

3. Session Management with URL Rewriting

   http://localhost:8080/ServletSessionURLRewriteApp/

Thanks!

Any questions?