COM 1001

Introduction to Software Engineering

Professor Phil McMinn

Cookies

HTTP Cookies

Although databases provide a persistent datastore, we cannot use them, by themselves, to track individual users of a web application.

HTTP cookies provide this facility.

Cookies are are pieces of information that a web application can store on a user's computer.

Cookies can be used to "remember" information about a specific user, such as the pages they visited, when, what data they typed into a form last time they encountered it, and so on. Cookies can be used in a variety of different ways.

Two Types of HTTP Cookie

There are two types of cookie: **session** and **persistent**.

Session cookies are destroyed when the user closes their browser. The web application therefore has no memory about a user next time they use it.

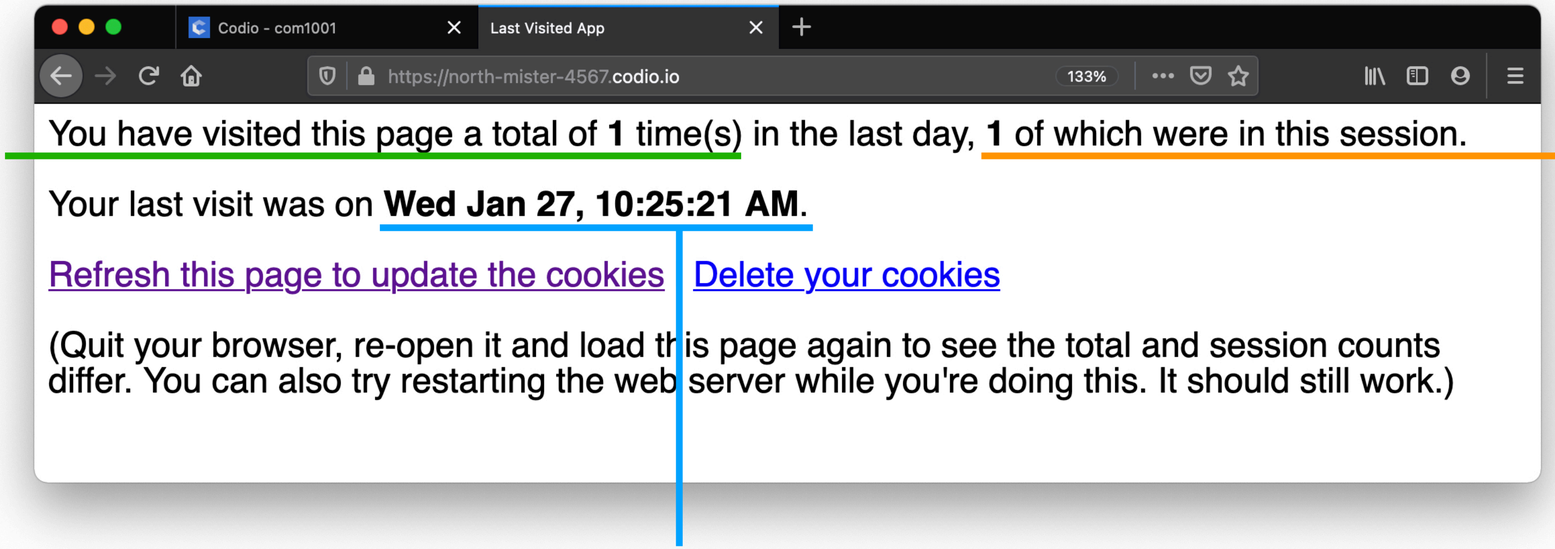
Persistent cookies expire in a predetermined time that is stored in the cookie itself. Persistent cookies can be used to remember information between visits to a web application.

Cookies in Action

Run the “Last Visited” example in the GitHub repository (in the `cookies/last_visited` directory).

Try refreshing the page, closing down your browser and re-opening it.

This information is managed by a **persistent cookie** set to expire after a day.



This information is managed by a **session cookie**. The number will reset to zero if we close and reopen the browser.

We can store any information (so long as it’s a string!) in a cookie.

Reading Cookies

How do we know which cookies we can read? Because we know which ones we've written to the browser (later in the code). But we could also interrogate the `cookies` hash to see what keys it has.

```ruby
get "/" do
  # read persistent cookies
  last_visited_persistent = request.cookies["last_visited_persistent"]
  num_visits_persistent = request.cookies.fetch("num_visits_persistent", 0).to_i

  # read session cookies
  num_visits_session = request.cookies.fetch("num_visits_session", 0).to_i

  ...
```

`cookies/last_visited/controllers/last_visited.rb`
(part 1/3)

Cookies are sent by the browser as part of the HTTP request. So we need to read them from the `cookies` hash in the `request` object. **Each cookie has a key in the has that is a string.**

Cookie values are also strings, but we can cast them to other types (here an integer) where appropriate.

There is no difference to how we access persistent and session cookies, this is just how the code is organised.

By using `fetch` on the hash, we ensure we have a value (`0`) even if the cookie is not set.

Writing Cookies

```
...

# set persistent cookies
response.set_cookie("last_visited_persistent",
                    { value: Time.now,
                      expires: Time.now + ONE_DAY_IN_SECONDS })
response.set_cookie("num_visits_persistent",
                    { value: num_visits_persistent + 1,
                      expires: Time.now + ONE_DAY_IN_SECONDS })

# set session cookies
response.set_cookie("num_visits_session", num_visits_session + 1)

...
```

cookies/last_visited/controllers/last_visited.rb
(part 2/3)

We send cookies back to the browser via the `response` object, using the `set_cookie` method. The first parameter is the cookie name.

For persistent cookies, the second parameter is a hash containing the value and expiry time (to the second)

For session cookies, we just use the data as the second parameter

The final part of this block (not shown) formats the cookie data for presentation in the page. Check out the code yourself.

Deleting Cookies

```ruby
get "/delete" do
  response.delete_cookie("last_visited_persistent")
  response.delete_cookie("num_visits_persistent")
  response.delete_cookie("num_visits_session")
  erb :delete
end
```

`cookies/last_visited/controllers/last_visited.rb`
(part 3/3)

We ask the browser to remove cookies via the `response` object again. The method we need is `delete_cookie`.

Practical Example: Remembering Search Terms

The search on the football players app forgets the search term when we move to an add or edit page, then return to the search. We can remedy this with a cookie:

```
get "/search" do
  @club_search = params.fetch("club_search",
                              request.cookies.fetch("club_search", "")).strip
  response.set_cookie("club_search", @club_search)

  @players = if @club_search.empty?
               Player.all
             else
               Player.where(Sequel.like(:club, "%#{@club_search}%"))
             end

  erb :search
end
```

cookies/football_players/controllers/football_players.rb

If "`club_search`" cannot be found in params (i.e., submitted as a user input), we then look to see if it's been set in a cookie.

(If it's in neither, we just use the empty string.)

Whatever value for `@club_search` is being used is saved in a cookie. This ensures the value will be remembered by the above code next time the user returns to the page.

Since we've not set an expiry date, this will be a session cookie – it will be forgotten after the user leaves the application.

Concerns about Cookies

Cookies cannot carry virus or install malware on a user's computer.

However, cookies are a way that web applications can use to compile long-term records of individual browsing histories (so-called "**tracking cookies**").

This is sometimes done legitimately to optimise a website design according to common usage patterns.

However, the privacy concern prompted EU law makers to take action, meaning that websites potentially need to get user-consent for storing cookies, or at least explain how they are used by the website.

You can find out more at **https://www.privacypolicies.com/blog/eu-cookie-law/**