COMP 3015

Cookies, sessions, intro to scalability

Today

- A couple language features
- Unix Time
- HTTP cookies
- Sessions
- Intro to scalability

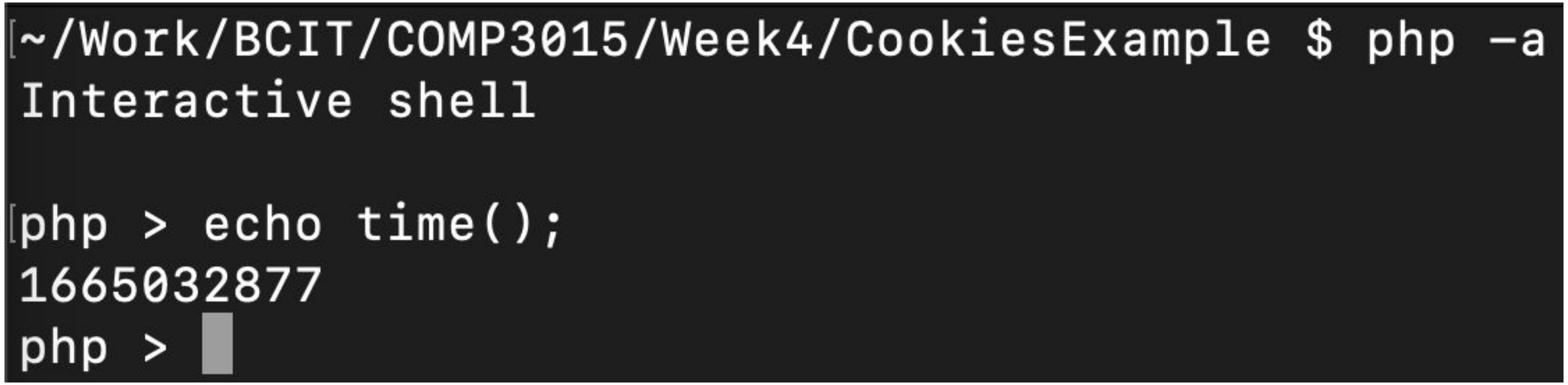
A Couple Language Features

See **in-class-examples/LangFeatures**

Unix Time

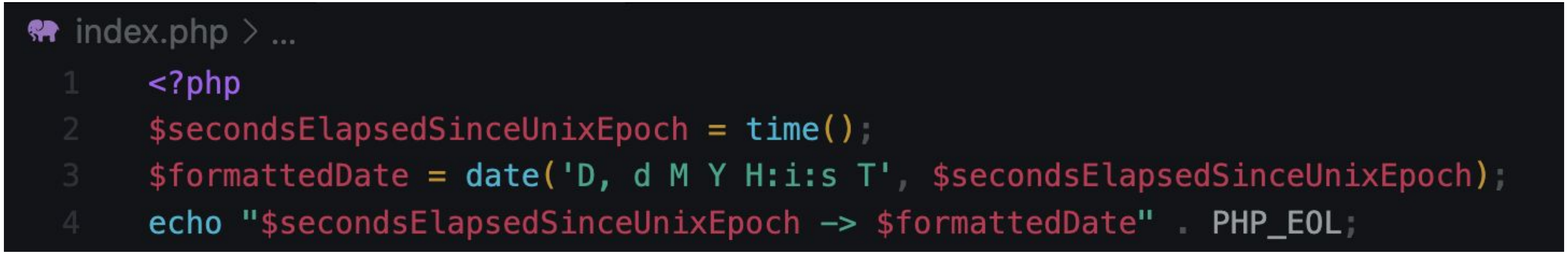
Unix Epoch Time

The number of seconds since January 1st, 1970, 00:00:00 Coordinated Universal Time (UTC). PHP has time() which returns this value.



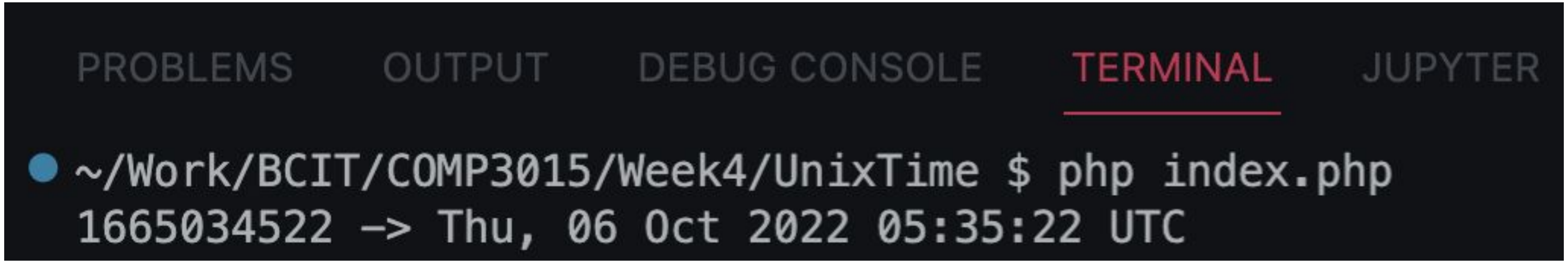
```
~/Work/BCIT/COMP3015/Week4/CookiesExample $ php -a
Interactive shell

php > echo time();
1665032877
php > 
```

Unix Epoch Time (continued)



index.php > ...

```php
<?php
$secondsElapsedSinceUnixEpoch = time();
$formattedDate = date('D, d M Y H:i:s T', $secondsElapsedSinceUnixEpoch);
echo "$secondsElapsedSinceUnixEpoch -> $formattedDate" . PHP_EOL;
```



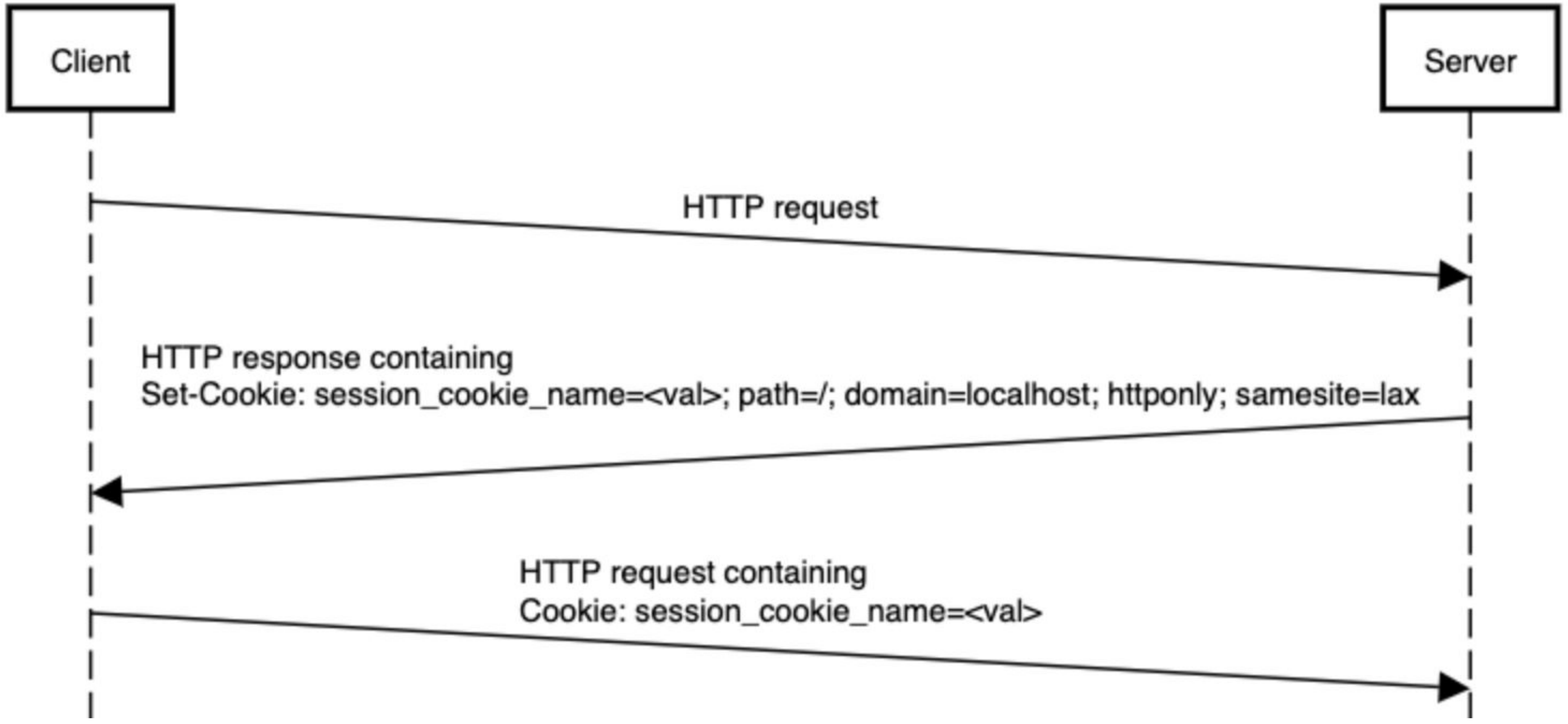
```
~/Work/BCIT/COMP3015/Week4/UnixTime $ php index.php
1665034522 -> Thu, 06 Oct 2022 05:35:22 UTC
```

Unix Epoch Time (continued)

Moving forwards and backwards can be done by adding and subtracting integers from the Unix timestamp.

See **in-class-examples/UnixTime**

🍪🍪🍪🍪 Cookies! 🍪🍪🍪🍪


Client
Server
HTTP request
HTTP response containing
Set-Cookie: session_cookie_name=<val>; path=/; domain=localhost; httponly; samesite=lax
HTTP request containing
Cookie: session_cookie_name=<val>

HTTP Cookies

- HTTP cookies are used to store state in the browser (client side storage)
  - Once cookies are sent to the client, it’s possible for them to be modified
    - Aside: In some cases, cookie contents will be encrypted and signed (e.g. using AES-GCM) to prevent modification without the server knowing – think authentication related cookies

- In our case, cookies will be sent from the server to the client
  - In PHP we have the **setcookie** function, which will cause the **Set-Cookie** response header to be set

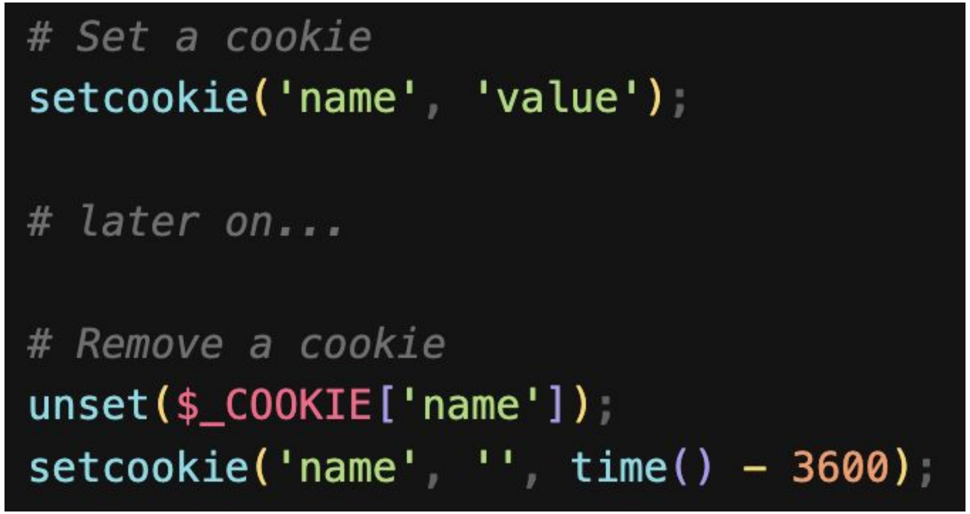
- The browser may choose to store it and send it back to the server on subsequent requests
  - Users can block cookies. In this case, the browser will not store the cookie

HTTP Cookies

The Cookie and Set-Cookie HTTP headers are described in the HTTP State Management Mechanism RFC: **https://www.rfc-editor.org/rfc/rfc6265**

- Learning to read RFCs is a good skill to have
- Lots of instances in building software where something isn’t working and you’ll need to ensure you’re following the protocol (be it HTTP, TCP, WebSockets, etc.)

HTTP Cookies and PHP



```
# Set a cookie
setcookie('name', 'value');

# later on...

# Remove a cookie
unset($_COOKIE['name']);
setcookie('name', '', time() - 3600);
```

setcookie will set a “Set-Cookie” HTTP response header.

Call **setcookie** with the same name, and a time in the past in order to ask the browser to remove it.

HTTP Cookies (continued): Attributes to know

Secure: Only sent to the server using HTTPS (encrypted version of HTTP). Note: on localhost it can still be sent over HTTP.

HttpOnly: Inaccessible to the JavaScript document.cookie API. The cookie will only be sent to the server using the **Cookie** request header

- This helps mitigate XSS attacks such as the one we saw last week

HTTP Cookies (continued)

Storing state in browsers: small theme example on D2L

CookiesExample.zip

Add a button to clear the cookie!

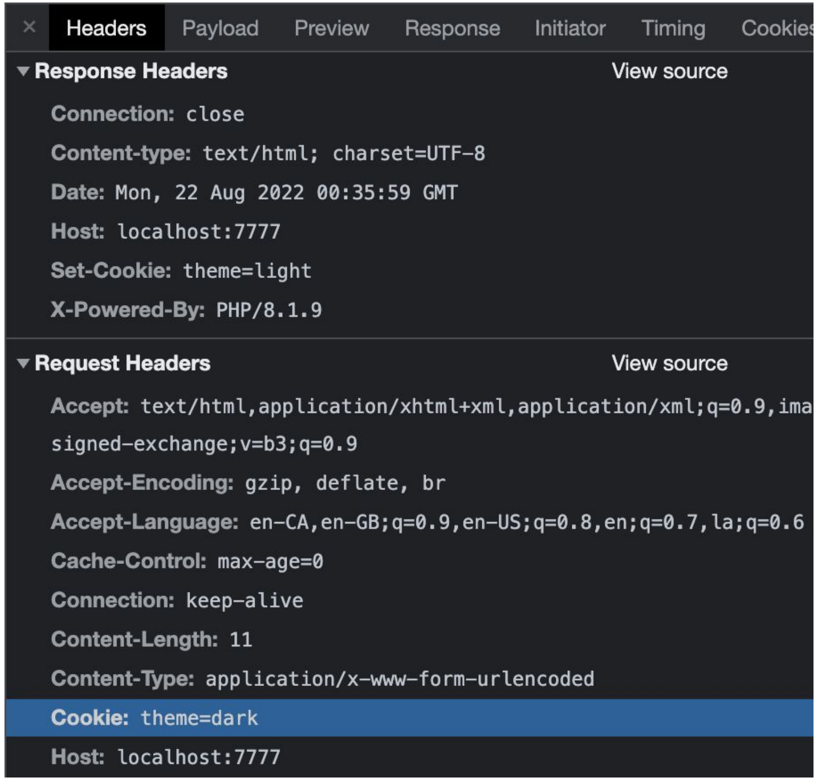
HTTP Cookies: Notes on the demo/example

Try running the demo and inspecting the request and response headers.

You'll be able to see (when submitting the form) that the **Cookie** request header is used to send data from the client to the server. The value "theme=dark" is the content of the cookie on in the browser.

The **Set-Cookie** response header which the server sends the client asks the browser to store a new cookie, or update an existing cookie.

Study this for quiz/exam purposes.



```
▼ Response Headers                                   View source
   Connection: close
   Content-type: text/html; charset=UTF-8
   Date: Mon, 22 Aug 2022 00:35:59 GMT
   Host: localhost:7777
   Set-Cookie: theme=light
   X-Powered-By: PHP/8.1.9
▼ Request Headers                                    View source
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,ima
   signed-exchange;v=b3;q=0.9
   Accept-Encoding: gzip, deflate, br
   Accept-Language: en-CA,en-GB;q=0.9,en-US;q=0.8,en;q=0.7,la;q=0.6
   Cache-Control: max-age=0
   Connection: keep-alive
   Content-Length: 11
   Content-Type: application/x-www-form-urlencoded
   Cookie: theme=dark
   Host: localhost:7777
```

Cookies and subdomains

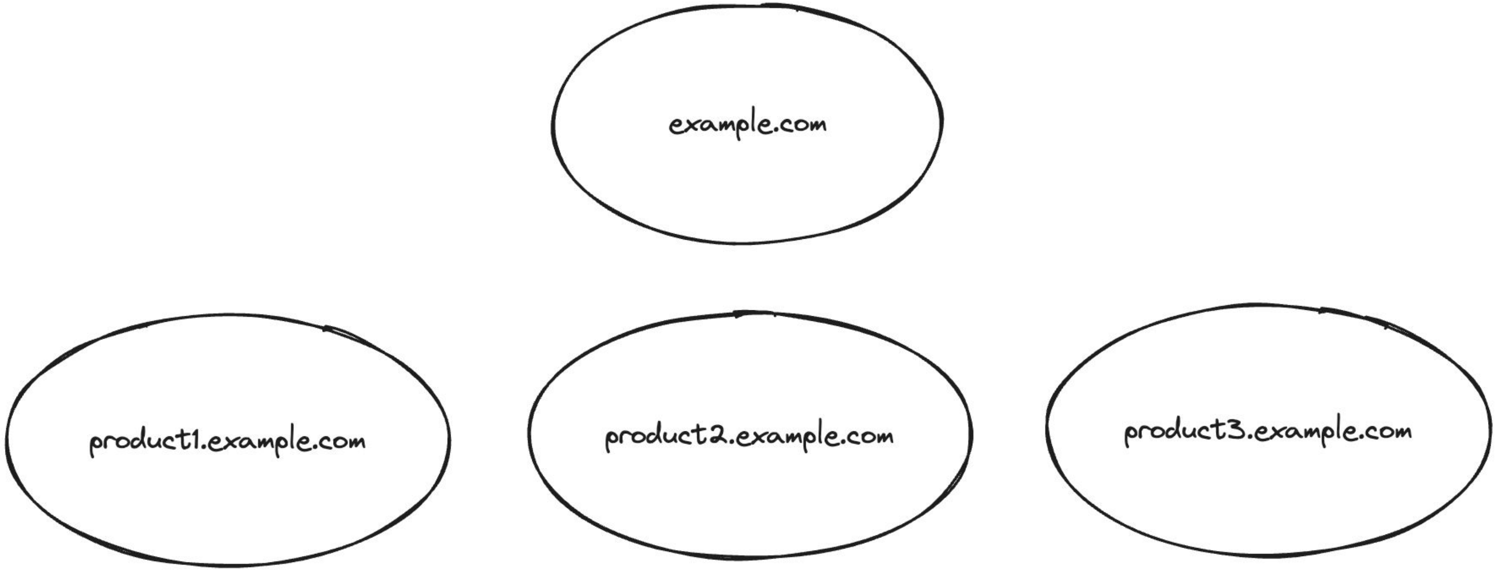
- A common pattern for web applications is to use subdomains on a per-product/subscription basis.

- For example, a sales site might be at **example.com**, and various products/services offered at subdomains of the sales site: **product1.example.com**, **product2.example.com**, etc.

- Real example: Amazon has amazon.com, aws.amazon.com, alexa.amazon.com, www.amazon.com, etc.

Cookies and subdomains

Some cookies should be shared between the parent domain and all subdomains. For example, any cookie used for authentication or preferences.



Cookies and subdomains

- We need to be able to share cookies with the parent domain, and subdomains

- In order to achieve this, we can set the “domain” attribute of the Set-Cookie response header value to “**.example.com**”.

    - Note that the leading dot character shouldn’t be needed according to the HTTP State Management RFC: RFC 6265, it is still often included to ensure backwards compatibility with any browsers implementing RFC 2109.

    - It’s generally a good idea to include the leading dot - doesn’t hurt to!

Sessions

Intro to Sessions

Cookies are used to store data on the client side.

Sessions are used to store data on the server side.

A server has many clients though, so we need a way to determine what data belongs to which user. Cookies are used for this.

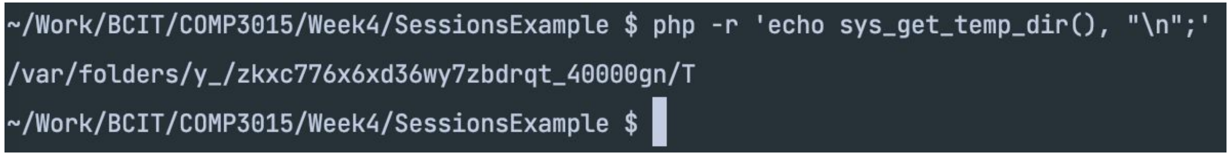
Sessions (cont.)

- Session data is stored on the server. By default, this is in the temporary files directory of your OS.

- We can store session data in a number of other places including in Redis, or a database.

- For the purposes of this course we will just use files.

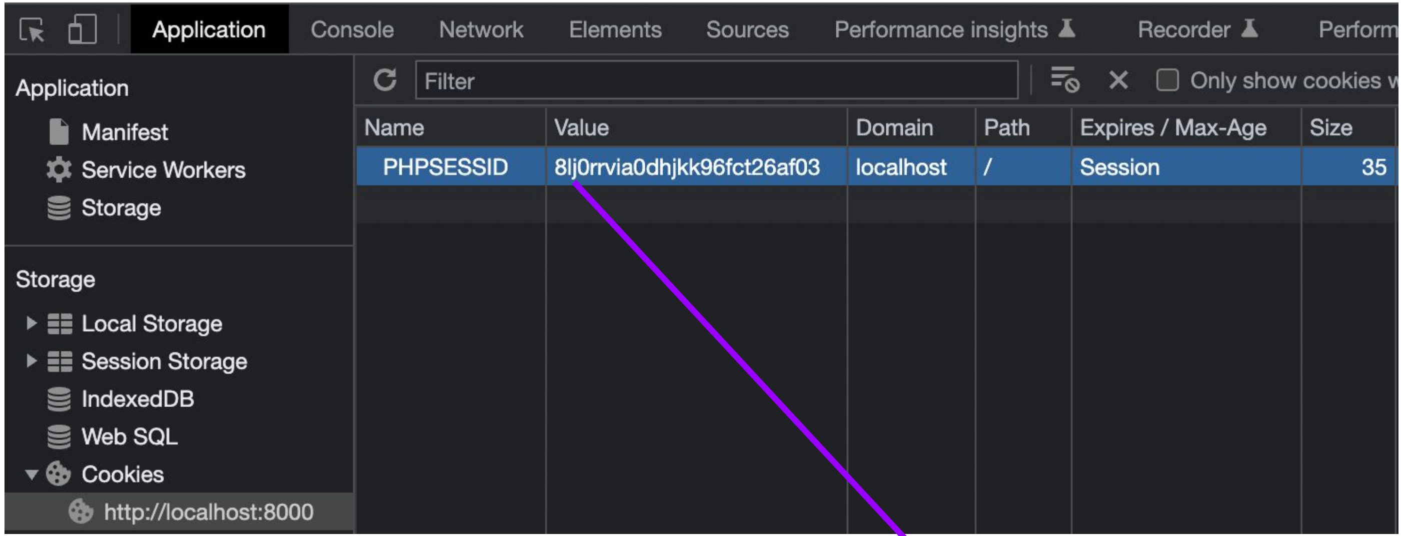
- Find the temp directory where session data is stored:
    - **php -r 'echo sys_get_temp_dir(), "\n";'**



```
~/Work/BCIT/COMP3015/Week4/SessionsExample $ php -r 'echo sys_get_temp_dir(), "\n";'
/var/folders/y_/zkxc776x6xd36wy7zbdrqt_40000gn/T
~/Work/BCIT/COMP3015/Week4/SessionsExample $
```

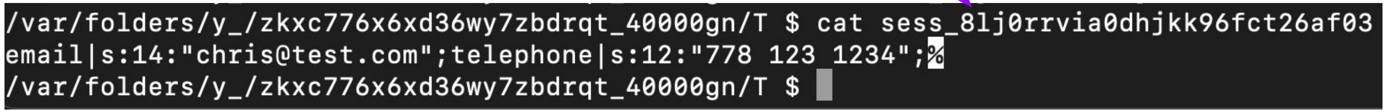
Sessions (cont.)

Session cookie on the client:



Name	Value	Domain	Path	Expires / Max-Age	Size
PHPSESSID	8lj0rrvia0dhjkk96fct26af03	localhost	/	Session	35

Session data on the server (in the temp directory):



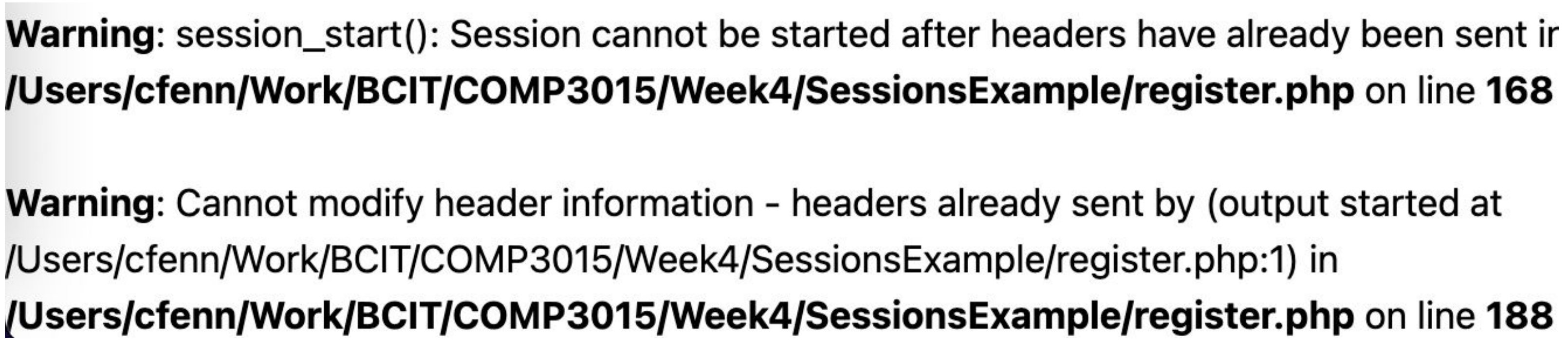
```
/var/folders/y_/zkxc776x6xd36wy7zbdrqt_40000gn/T $ cat sess_8lj0rrvia0dhjkk96fct26af03
email|s:14:"chris@test.com";telephone|s:12:"778 123 1234";%
/var/folders/y_/zkxc776x6xd36wy7zbdrqt_40000gn/T $
```

Sessions (cont.)

Session data can be managed by using the **$_SESSION** array superglobal

Before using the **$_SESSION** array we need to call **session_start()**

Sessions (cont.): Addressing a common error



```
Warning: session_start(): Session cannot be started after headers have already been sent ir
/Users/cfenn/Work/BCIT/COMP3015/Week4/SessionsExample/register.php on line 168

Warning: Cannot modify header information - headers already sent by (output started at
/Users/cfenn/Work/BCIT/COMP3015/Week4/SessionsExample/register.php:1) in
/Users/cfenn/Work/BCIT/COMP3015/Week4/SessionsExample/register.php on line 188
```

Sessions (cont.): Addressing a common error

Why does this happen? What does it mean?

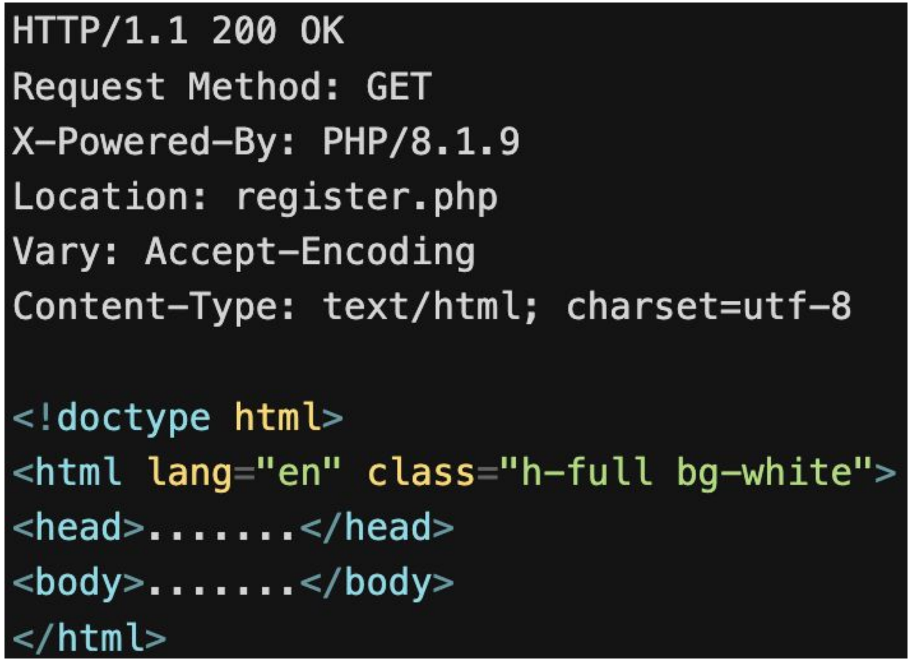
- PHP scripts generate HTML, but also need to generate and pass the HTTP response headers to the web server.

- On the first occurrence of output in the file (e.g. echo, print, HTML tags), the headers (e.g. from calls to functions such as header(), setcookie(...), session_start(), etc.) are set at this point as they've been sent to the webserver.
    - This is why the error message reads "...headers have already been sent"

- At this stage, additional output can be sent, but not additional headers.

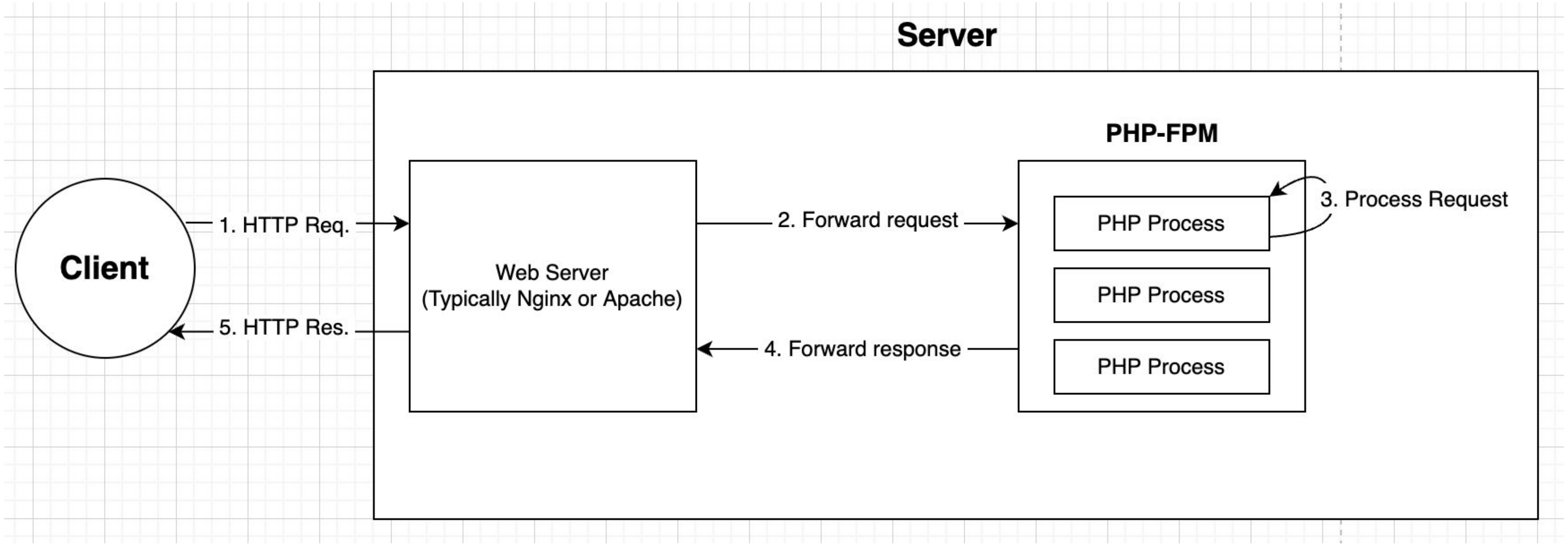
Sessions (cont.) Addressing a common error

When the PHP process communicates with the web server, the content needs to be after the response headers:



```
HTTP/1.1 200 OK
Request Method: GET
X-Powered-By: PHP/8.1.9
Location: register.php
Vary: Accept-Encoding
Content-Type: text/html; charset=utf-8

<!doctype html>
<html lang="en" class="h-full bg-white">
<head>.......</head>
<body>.......</body>
</html>
```

Sessions (cont.): Addressing a common error



Key takeaway: functions that modify headers must be called before output is sent from PHP to the web server.

Sessions (cont.)

Flash error messages and remember form data using sessions for the lab:

lab-3-starter-kit.zip

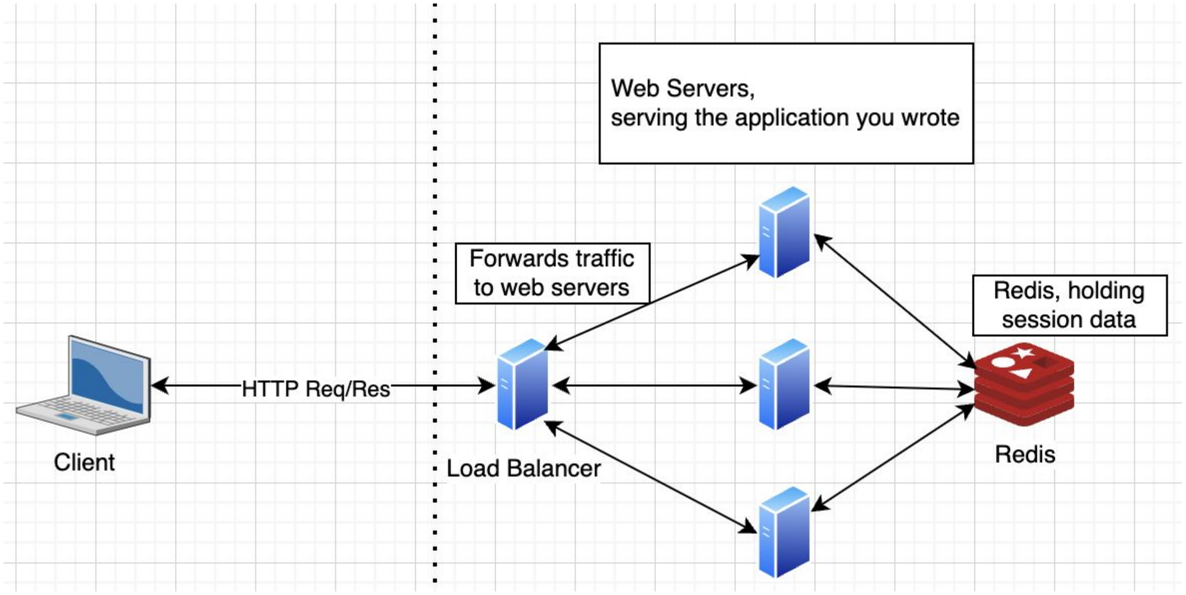
Scalability

Scalability

- How we can build systems that support lots of users at the same time?

- **Vertical scaling** (scaling up)
  - Add more resources to an existing system
  - Increase RAM, compute power (eg. upgrade CPUs)
- **Horizontal scaling** (scaling out)
  - Add additional machines
  - Machines will now need to be load balanced
    - One machine acts as the entry point to your system, and directs traffic to web servers
  - How can we add additional machines when we start to store session data on each web application server?

Horizontal Scalability and Session Data



By default session data is stored in files on the web server.

For scalable systems we can move the storage of session data to another server.

This approach allows for "horizontal scalability"

Lab 3 is on D2L

Smaller lab considering assignment 1 is due next week

Review (this content will be on quiz #2)

- What PHP function is used to set a cookie?
- What HTTP header is used to ask the browser to store a cookie?
- How do we ask the browser to remove a cookie?
- Are HTTP cookies sent back to the server on subsequent requests?
- What is an “HttpOnly” cookie?
- What is a “secure” cookie?
- What is the difference between cookies and sessions?
    - Think client-server architecture
- How does the server know which session (and associated data) belongs to which client?
    - This is probably one of the most important questions to be able to answer

TODO

- Assignment 1 due next week

- Lab 3 due next week
  - Submit to D2L as usual
  - Smaller lab due to assignment 1

- Review cookies, sessions, intro to scalability

- Take home quiz on various HTTP topics (GET/POST requests, Cookies), Sessions