PHP

HTTP Methods, Sessions, and JSON

COS216
AVINASH SINGH
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF PRETORIA

PHP – HTTP METHODS

- GET request

  http://satoshi.com/api.php?currency=bitcoin

- POST request

  http://satoshi.com/api.php

  Body: currency=bitcoin

- HTTP requests can be handled on the server side in PHP

PHP – HTTP METHODS

- GET request

http://satoshi.com/api.php?currency=bitcoin

```php
<?php
        $data = [
                "bitcoin" => [ "symbol" => "BTC", "price" => 10452 ],
                "ethereum" => [ "symbol" => "ETH", "price" => 843 ]
        ];

        $currency = $data[$_GET["currency"]];
        echo $currency["symbol"] . ": " . $currency["price"];
?>
```

PHP – HTTP METHODS

- POST request

http://satoshi.com/api.php

Body: currency=bitcoin

```php
<?php
        $data = [
                "bitcoin" => [ "symbol" => "BTC", "price" => 10452 ],
                "ethereum" => [ "symbol" => "ETH", "price" => 843 ]
        ];

        $currency = $data[$_POST["currency"]];
        echo $currency["symbol"] . ": " . $currency["price"];
?>
```

PHP – HTTP METHODS

- Check if parameter was set

```php
<?php
	$data = [
		"bitcoin" => [ "symbol" => "BTC", "price" => 10452 ],
		"ethereum" => [ "symbol" => "ETH", "price" => 843 ]
	];

	if(isset($_GET["currency"]))
		echo $data[$_GET["currency"]]["symbol"];
	else if(isset($_POST["currency"]))
		echo $data[$_POST["currency"]]["symbol"];
	else
		echo "No currency parameter specified";
?>
```

PHP – HTTP METHODS

- It is always a good idea to provide access to your API/server functions via both GET and POST
- So end users can either make a GET or a POST request and both work
- Although passwords (and sensitive information) should never be send via GET, sometimes the app/library does not allow for POST requests
  - In certain frameworks (eg: Qt/QML on Android) you can embed a GET URL directly inside the graphical interface
  - For instance, you can provide a URL to a profile picture which is then displayed in the Android app as an image (retrieving the picture might require a login)
  - In that case a GET URL can be specified containing the username/password
  - E.g: *https://u12345678:pA$$w0rD@domain.co.za/images/logo.png*

PHP – HEADERS

- Return a custom HTTP response header
- Set a HTTP code for your custom code (not found, unauthorized, internal error)
- PHP returns 200 by default, expect if some error occurred

```php
<?php
        if($loggedIn)
        {
                header("HTTP/1.1 200 OK");
                echo "Everything went fine";
        }
        else
        {
                header("HTTP/1.1 401 Unauthorized");
        }
?>
```

PHP – HEADERS

- Return a fixed content type (HTML, XML, JSON, or binaries like images/videos)
- By default, text/html is returned

```
<?php
        header("Content-Type: application/json");

        echo '{
                "bitcoin" : { "symbol" : "BTC", "price" : 10452 },
                "ethereum" : { "symbol" : "ETH", "price" : 843 }
        }';
?>
```

PHP – JSON

- Encode PHP objects and arrays as a JSON string

```php
<?php
        $data = [
                "bitcoin" => [ "symbol" => "BTC", "price" => 10452 ],
                "ethereum" => [ "symbol" => "ETH", "price" => 843 ]
        ];

        header("HTTP/1.1 200 OK");
        header("Content-Type: application/json");
        echo json_encode($data);
?>
```

PHP – JSON

- Decode a JSON string to a PHP object or array

```php
<?php
	$login = json_decode($_POST["login_json_data"]);
	if(validLogin($login))
	{
		header("HTTP/1.1 200 OK");
		echo "Your are logged in";
	}
	else
	{
		header("HTTP/1.1 401 Unauthorized");
		echo "Your login credentials are invalid";
	}
?>
```

PHP – PARAMETERS

- Hence there are different ways in which to send data to the server

- Use the URL path to specify parameters
- Limited in what can be sent
- Requires some extra coding in Apache and PHP
- Other server frameworks (like NodeJS) already support this by default

http://satoshi.com/api/val1/val2

PHP – PARAMETERS

- Hence there are different ways in which to send data to the server

- Use separate HTTP parameters for each value to be send to the server

http://satoshi.com/api.php?param1=val1&param2=val2

PHP – PARAMETERS

- Hence there are different ways in which to send data to the server

- Encode all the data as a single JSON object and send the JSON as a single HTTP parameter to the server
- Note the JSON object is URL encoded here, since some characters (like :) are reserved in URLs (https://www.w3schools.com/tags/ref_urlencode.asp)

http://satoshi.com/api.php?data=%7B%22param1%22%3A%22val1%22%2C%22param2%22%3A%22val2%22%7D

PHP – GLOBALS

- Super global variables can be accessed from anywhere in the trail of executing scripts (including functions, classes, and other scripts)
- Can be used to store configurations or keep track of some global information
- Note it does not have an underscore (like $_GET and $_POST)

```php
<?php
	$GLOBALS["myvar"] = "access from anywhere";

	function execute()
	{
		echo $GLOBALS["myvar"];
	}

	execute();
?>
```

PHP – GLOBALS

- Common practice is to create a separate script (eg: config.php) with your server configuration, database details, etc
- This script can then be included anywhere and the globals used

```php
<?php
	$GLOBALS["config"] = [
		"database" => [
			"host" : "127.0.0.1",
			"name" : "crypto_prices",
			"user" : "satoshi",
			"pass" : "df^A_h%u&A12*"
		]
	];

	connectToDatabase($GLOBALS["config"]["database"]);
?>
```

PHP – STATE

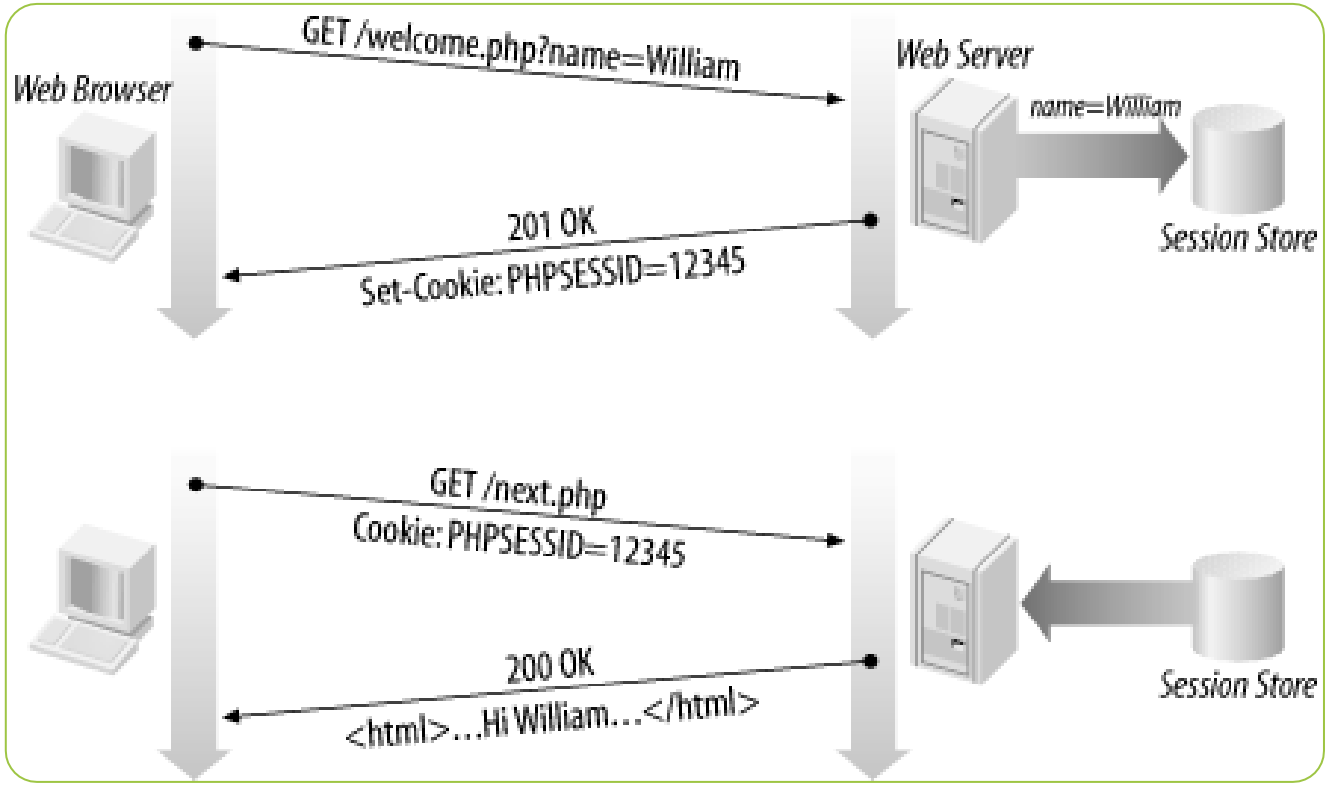
- REST and protocols, on which most websites are built, are stateless
- Hence, no state (info or variables) are stored between different requests
- Sometimes states is required between different request
  - Eg: Login request, and all requests thereafter should have the login state, user and pass

- Two approaches to store information
  - On the client side: Using cookies or DOM storage
  - On the server side: Either temporary (sessions), or permanent storage (database, files)

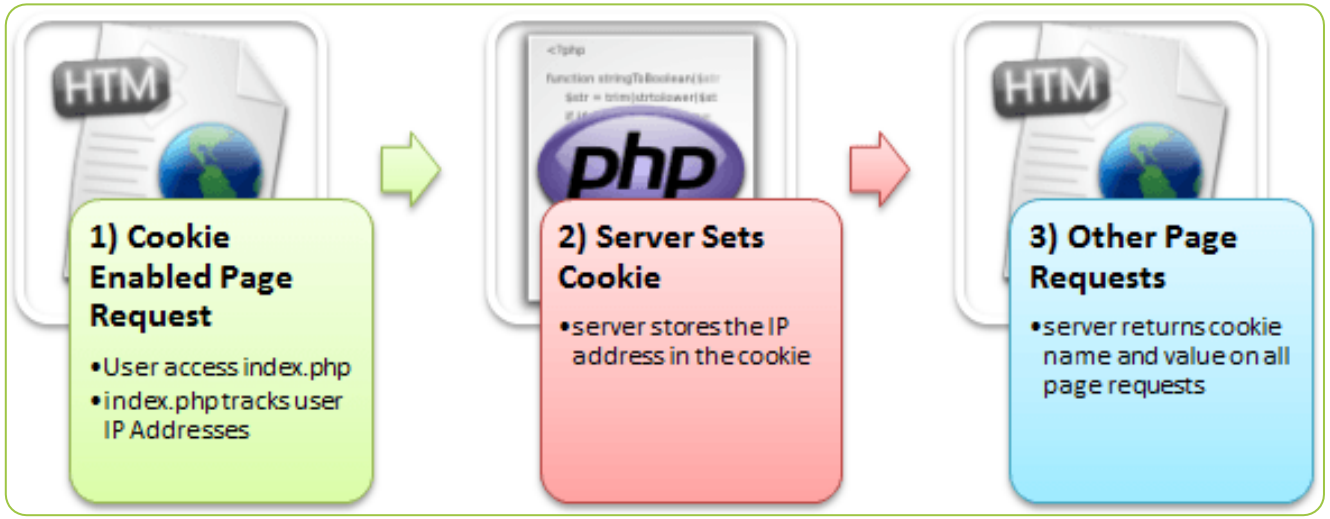
PHP – SESSIONS

- Sessions store information over different requests
- The general sequence:
  1. A user makes a request to the server (eg: login)
  2. The server evaluates the request and if successful creates a new session (eg: valid login)
  3. PHP creates a unique session ID
  4. PHP saves the session ID and the session data to a file on the server
  5. The session ID is returned to the client
  6. The client browser saves the session ID locally (typically as a cookie)
  7. With every subsequent request, the client appends the session ID to the call
  8. The server receives the subsequent requests with the session ID
  9. PHP will check if the session ID is valid and load the session data into memory

PHP – SESSIONS



PHP – SESSIONS



PHP – SESSIONS

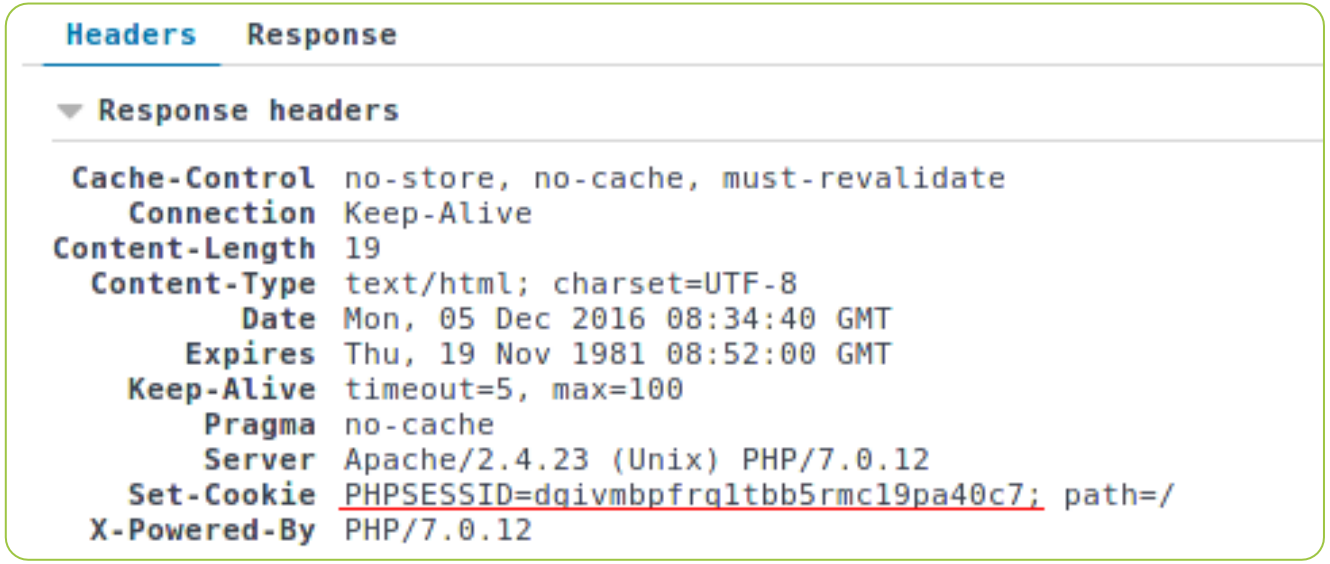
Creating the session ID, reading and writing session files, and managing the session memory is all done for you by PHP

Saving and loading the session cookie on the client side is also done for you by the browser

Web developers only have to check and update session variables

PHP – SESSIONS

- A typical PHP session ID returned in the server response
- Note the "Set-Cookie" header, which will tell the browser to save the data locally as a cookie (for later use)



PHP – SESSIONS

- The session should always be started at the beginning of the execution
  - Do not start a thread somewhere late in your code, PHP will throw an error
  - Sessions have to be started for creating new sessions, or using existing once
  - Starting a session tells PHP to load the necessary sessions into memory
  - If a session does not exists, PHP will create a new one

```php
<?php
        session_start();
?>
```

PHP – SESSIONS

- Create/update the session variables (eg: logging in)

```php
<?php
	session_start();

	if(validLogin($_POST["user"], $_POST["pass"]))
	{
		$_SESSION["loggedin"] = true;
		$_SESSION["user"] = $_POST["user"];
	}
?>
```

PHP – SESSIONS

- Remove/clear session variables (eg: when logging out)

```php
<?php
        session_start();

        // Check if session variable was previously set
        $loggedIn = isset($_SESSION["user"])

        // Remove a single session variable
        if($loggedIn) unset($_SESSION["user"]);

        // Remove all session variables
        session_unset();

        // Destroy the session, its ID and corresponding file
        session_destroy();
?>
```

PHP – SESSIONS

- Login:

http://satoshi.com/api.php

Body: action=login&user=satoshi&pass=U8Yh0Pok

- Logout:

http://satoshi.com/api.php

Body: action=logout

PHP – SESSIONS

```php
<?php
	session_start();
	if($_POST["action"] == "login")
	{
		if(isset($_SESSION["loggedin"]))
			echo "Already logged in";
		else if(validLogin($_POST["user"], $_POST["pass"]))
		{
			$_SESSION["loggedin"] = true;
			$_SESSION["user"] = $_POST["user"];
			echo "Logged in";
		}
		else
			echo "Invalid login details";
	}
	else if($_POST["action"] == "logout"){ // ... }
?>
```

PHP – SESSIONS

```php
<?php
	session_start();
	if($_POST["action"] == "login") { // ... }
	else if($_POST["action"] == "logout")
	{
		if(isset($_SESSION["loggedin"]))
		{
			session_unset();
			session_destroy();
			echo "Logged out";
		}
		else
			echo "Not logged in";
	}
?>
```

PHP – SESSIONS

- Login using JSON:

http://satoshi.com/api.php

Body: {“action”:“login”,“user”:“satoshi”,“pass”:“U8Yh0Pok”}

- Logout using JSON:

http://satoshi.com/api.php

Body: {“action”:“logout”}

PHP – SESSIONS

Remember to do error checking

```php
<?php
	session_start();
	// Takes raw data from the request
	$json = file_get_contents('php://input’);
	// Converts it into a PHP object
	$data = json_decode($json, true);
	if($data["action"] == "login")
	{
		if(isset($_SESSION["loggedin"]))
			echo "Already logged in";
		else if(validLogin($data["user"], $data["pass"]))
		{
			$_SESSION["loggedin"] = true;
			$_SESSION["user"] = $data["user"];
			echo "Logged in";
		}
		else
			echo "Invalid login details";
	}
	else if($data["action"] == "logout"){ // ... }
?>
```

PHP – SESSIONS

```php
<?php
        // ...
        else if($data["action"] == "logout")
        {
                if(isset($_SESSION["loggedin"]))
                {
                        session_unset();
                        session_destroy();
                        echo "Logged out";
                }
                else
                        echo "Not logged in";
        }
?>
```

$_SESSION