Apache Configuration File

- httpd.conf → text file with main Apache configuration settings
- # (hash) represent comment/disabled option
- Location
  - PC location → C:/xampp/apache/conf/httpd.conf
  - Mac location → /Applications/XAMPP/etc/httpd.conf
- Changing any option in the file requires server rebooting
- httpd.conf defines options to customize the HTTP server
  - **ServerName** → name and port the server uses to identify itself
    - Example: ServerName localhost:80
  - **Listen** → to bind Apache to specific IP address/port
    - Example: Listen 8080 (binding to port 8080)
  - **DocumentRoot** → directory where documents reside
    - Example: DocumentRoot "C:/xampp/htdocs”
  - Make sure you update Listen and ServerName when changing ports

Apache Configuration File

- Additional httpd.conf configuration options
  - **LoadModule** → allows the use of the functionality of a module that was built as DSO (Dynamic Shared Object)
    - **Example:** LoadModule php5_module "c:/php/php5apache2_2.dll“
    - Several modules, some of which are disabled by default (remove # to enable)
    - In XAMPP the php5 module is loaded in file included by httpd.conf. The file included is at: xampp/conf/extra/ and it is named httpd-xampp.conf
  - **ServerAdmin** → E-mail to which server problems should be sent
    - Example: ServerAdmin staff@notreal.here.gone
  - **ErrorLog** → Error log file location
    - Example: ErrorLog logs/error.log
  - **ErrorDocument** → Customizable Error Responses
    - #ErrorDocument 500 "The server made a mistake“
    - #ErrorDocument 404 /missing.html
  - **AddType** → to map file extensions to specific content type
    - Syntax: *AddType MIME-type extension [extension]*
    - Example: AddType application/x-httpd-php .php .html
- A note about Apache
  - You can issue commands via httpd.exe (part of the apache installation and found in apache/bin)
  - httpd.exe –h provides a complete listing of commands

PHP Configuration File

- php.ini → configuration file controlling many aspects of PHP's behavior
  - Mac location: /Applications/XAMPP/etc
  - PC location: C:/xampp/php/php.ini
- ; (semicolon) represents comment/disabled option
- Any change in the configuration file requires server rebooting
- Directives are specified using the syntax
  - directive = value
- Boolean flags
  - Can be turned on with 1, On, True or Yes
  - Can be turned off with 0, Off, False or No
- Use the phpInfo() function to see PHP configuration Information
  - **Example:** phpConfInfo.php

PHP Configuration File

- Example of directives
  - precision → significant digits displayed in floating point numbers
    - **Example: precision = 11**
  - max_execution_time → maximum execution time of each script in seconds
    - **Example: max_execution_time = 60**
  - display_errors → prints out errors as part of the output. It is discouraged to have it on
    - **Example: display_errors = On**
  - file_uploads → to enable/disable HTTP file uploads
    - **Example: file_uploads = On**
  - upload_tmp_dir → temporary location for HTTP uploaded files
  - upload_max_filesize → maximum size of HTTP uploaded files
  - extension → to automatically load an extension
    - **Example: extension = php_mysql.dll**
    - Location of the extension should be specified using extension_dir
  - sendmail_from → sender e-mail address (me@not.real.gone)
  - session.use_cookies → whether to use cookies
  - session.name → cookie name
    - **Example: session.name = PHPSESSID**

Cookies

- Cookie → small piece of information stored either in the browser's memory or as a small file in the hard drive
- Cookie →contains a name/value pair
- Setting a cookie → associating a value with a name
- Getting a cookie →getting the value associated with a name
- Browsers put a limit on the number of cookies a domain can set
- Size of each cookie → Around 4096 bytes
- To meet specifications a browse should support
  - At least 300 cookies total
  - At least 20 cookers per host or domain name
  - Source
    - http://webdesign.about.com/od/cookies/f/cookies-per-domain-limit.htm

PHP Support for Cookies

- setcookie() → Allows you to create a cookie. It takes the following parameters
  - **name** (string) →cookie's name
  - **value** (string) → cookies' value
  - **expiration** (int) → when cookie expires
    - Value of 0 → default value and specifies cookie should last until browser is closed
    - Time stamp representing expiration time
  - **path** (string) → Defines which sites or subareas of the web server can set the cookie.
    - By default any page within the web root folder will be able to set and read the cookie
    - Make sure you include a trailing / in the path
  - **domain** (string) → controls which site can access (read or set) cookies associated with the browser. The default is not to check the domain
  - **secure** (int) → if 1 cookie will be sent only over a secure connection (e.g., https). It defaults to 0
- Deleting a cookie
  - Call setcookie with the exact same arguments used when the cookie was set except the value should be an empty string
- Reading cookies
  - Important: cookies are read on the next request of the page by the browser
  - You can access cookies values through $_COOKIE superglobal
  - Global array $HTTP_COOKIE_VARS

PHP Support for Cookies

- **Example:** cookie.php
  - Before running this example make sure cookies are accepted
    - In Chrome:
      - Select Chrome menu (=), **Settings**, **Show advanced settings**
      - In Privacy section select **Content settings…**
      - You will then see the Cookies option
      - Select **All cookies and site data…** to see cookies. Enter in the search box "localhost" to see cookies set
  - Run the example two times
- About cookies
  - setcookie() sends HTTP header information, which you cannot do after some output has been sent to the browser. **Set cookies before any output is generated**
  - setcookie does not guarantee the cookie will be set in the browser. It just tries to set it by sending header information setting the cookie
- You can also see cookie information via "Developer Tools"
  - Select Chrome menu(=), **More tools**, **Developer tools**, **Resources**, **Cookies**

Sessions

- **Session** → time period during which a person views a number of different web pages in a browser and then quits
- **What would you like**
  - To keep track of information throughout the session. For example, keeping track of color preferences, usernames, data selection, etc.
- **What is the problem?**
  - http (the protocol that makes possible the communication between browsers and web servers) is stateless
  - Stateless → every page request is independent
- **Solution**
  - PHP Sessions

PHP Session Support

- PHP **provides**
  - Session tracking by detecting when two script invocations belong to the same session
  - Storing information (e.g., variables) associated with a session
- Methods to propagate a session id:
  - Cookies
  - URL parameter
  - Hidden form field
- If cookies are not supported then session information (session ID) can be passed through URL parameter or a hidden form field
- A session is identified via an ID

PHP Session Support

- **session_start()** function
  - Searches for a session in progress (looks for session id) or starts a new one if one in progress is not found
  - Starting a new session creates a new session ID
  - If a session in progress is found, registered session variables are restored (and made available through $_SESSION superglobal)
  - You can avoid using session_start() in every script (and let the script look for a session) by setting the php.ini variable session.auto start to 1
- How to store values for use in other scripts?
  - Assign them to superglobal $_SESSION
- **Example:** sessionSetVars.php, sessionGetVars.php
- **Example:** *courseEvaluation.html* (uses verify.php/confirmation.php)
  - See the cookies after running the example
- **Let's run the previous example but with cookies disabled**

Where is Session Information Stored?

- In files in the server (one file per session)
- The file is named after the session id
- You can use the following code snippet to see the directory where session information is stored

```
echo "Session save path".session_save_path();
```

- **Example:** sessionInfo.php
- Let’s open one of these files

Dealing with Disabled Cookies via Get/URL

- You can rely on a PHP constant (SID) available when a session is active. Using that constant you could pass the session information via the URL
- Notice that the SID constant is of the form *name/value* where value is the actual session id and name is the variable name (e.g., PHPSESSID)
- Alternative via GET
  - You must pass a GET argument (session_name=session_id) in your links
  - Make sure cookies are disabled before running the example
  - **Example:** *courseEvaluationNoCookiesGet.html (uses* verifyNoCookiesGet.php, confirmation.php (same as before))
  - **Look at the URL**
    - You will see PHPSESSID as parameter
    - What's the security problem with this approach?

Dealing with Disabled Cookies via Post/Hidden Field

- You can rely on the session_id() function to retrieve a session id
- You can pass the session id value as a hidden field
- You can "restore" the session by retrieving the hidden field value and using the session id function to set the session id
- **Example**
  - Make sure cookies are disabled
  - **Example:** *courseEvaluationNoCookiesPost.html* (uses verifyNoCookiesPost.php, confirmationPost.php (NOT the same file we have been using))
  - Look at the URL

http headers

- You can see information exchanged between user agent and server by using apps similar to Chrome's **Advanced REST Client**
  - https://chrome.google.com/webstore/search/Advanced%20REST%20Client?hl=en-US&utm_source=ARC
- You can see how cookies are transmitted
  - Try http://localhost/cmsc389NSummer2015/ConfigSessionsCode/cookie.php