Web Design and Programming

Week 10
Peeraya Sripian

4 July 2024

Course schedule

Week	Date	Topic
1	4/18	Intro to WWW, Intro to HTML
2	4/25	CSS Fundamental
	5/2	Holiday (GW)
3	5/9	CSS and Bootstrap
4	5/16	Work on midterm project
5	**COIL 22 MAY 18:00-19:30 (counted as 1 class, replacing 23 May)**	
6	5/30	Midterm project presentation week
7	6/6	PHP fundamentals + Installation XAMPP
8	6/13	PHP fundamentals 2 + Intro of Final project
9	6/20	mySQL fundamentals
10	6/27	Assessing MySQL using PHP, MVC pattern
11	7/4	Cookies, sessions, and authentication **+ Proposal of final project**
12	7/11	Javascript and PHP validation
13	7/18	Final project development
14	7/25	Final project presentation

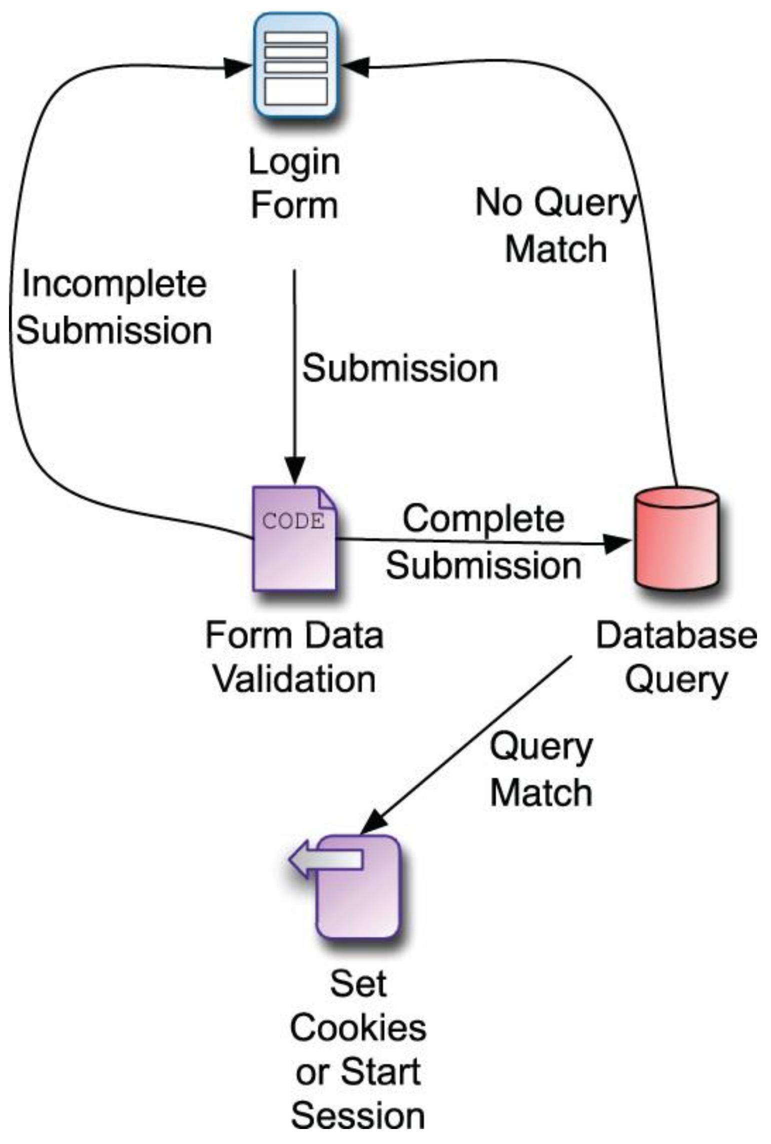
Cookies and sessions

Making a Login Page

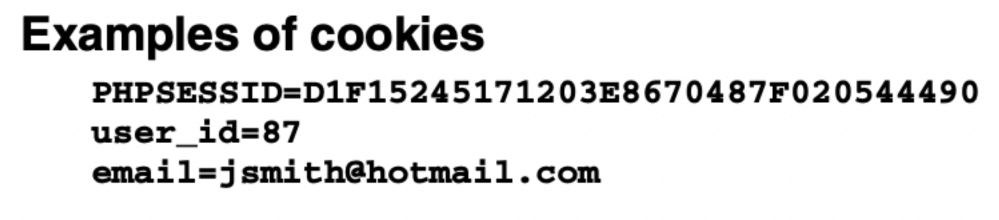
The login process.

- A form for submitting the login information
- A validation routine that confirms the necessary information was submitted
- A database query that compares the submitted information against the stored information
- Cookies or sessions to store data that reflects a successful login

Subsequent pages can check if the user is logged in to restrict access or enable additional features.
Logging out involves clearing session data, deleting the session on the server, and removing the session cookie from the user's browser.



Cookies



Examples of cookies

```
PHPSESSID=D1F15245171203E8670487F020544490
user_id=87
email=jsmith@hotmail.com
```

- A cookie is a name/value pair that is stored in a browser.
  - On server: a web application creates a cookie → browsers
  - On client: browser saves the cookie → server every time it access a page
- By default, cookies only last until the user closes his or her web browser.
  - Or until a specified expiration date of cookie.
- Can be disabled by the user.
- Browsers generally accept only 20 cookies from each site and 300 cookies total. In addition, they can limit each cookie to 4 kilobytes.
- A cookie can be associated with one or more subdomain names.

Typical uses for cookies

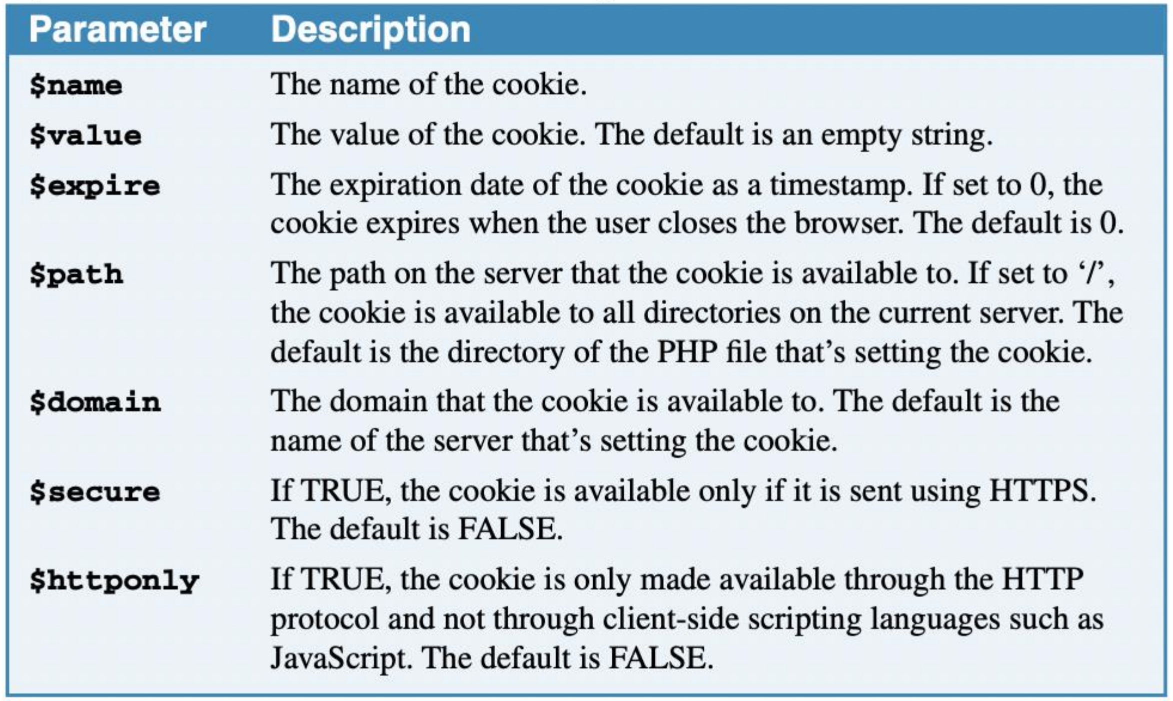
- To **allow users to skip login and registration forms** that gather data like username, password, address, or credit card data.
- To **customize pages** that display information like weather reports, sports scores, and stock quotations.
- To focus **advertising like banner ads** that target the user’s interests.

How to set and get a cookie

The syntax of the setcookie() function

```
setcookie($name, $value, $expire, $path, $domain, $secure, $httponly)
```

The parameters of the setcookie() function



Parameter	Description
`$name`	The name of the cookie.
`$value`	The value of the cookie. The default is an empty string.
`$expire`	The expiration date of the cookie as a timestamp. If set to 0, the cookie expires when the user closes the browser. The default is 0.
`$path`	The path on the server that the cookie is available to. If set to '/', the cookie is available to all directories on the current server. The default is the directory of the PHP file that's setting the cookie.
`$domain`	The domain that the cookie is available to. The default is the name of the server that's setting the cookie.
`$secure`	If TRUE, the cookie is available only if it is sent using HTTPS. The default is FALSE.
`$httponly`	If TRUE, the cookie is only made available through the HTTP protocol and not through client-side scripting languages such as JavaScript. The default is FALSE.

- `setcookie()` function. must be called before any HTML output is sent from your application.
- A session cookie expires when the user closes the browser, or until an expiration date (persistent cookie)
- Once a cookie has been set, you can get it the next time the browser requests a page.
  - `filter_input()` function can be used to get from superglobal `$_COOKIE` variable.
  - `$_COOKIE` is associative array
- To delete a cookie from a browser, set the value to an empty string and the expiration date to a time in the past. Any remaining parameters must have the same values as when the cookie was originally created.

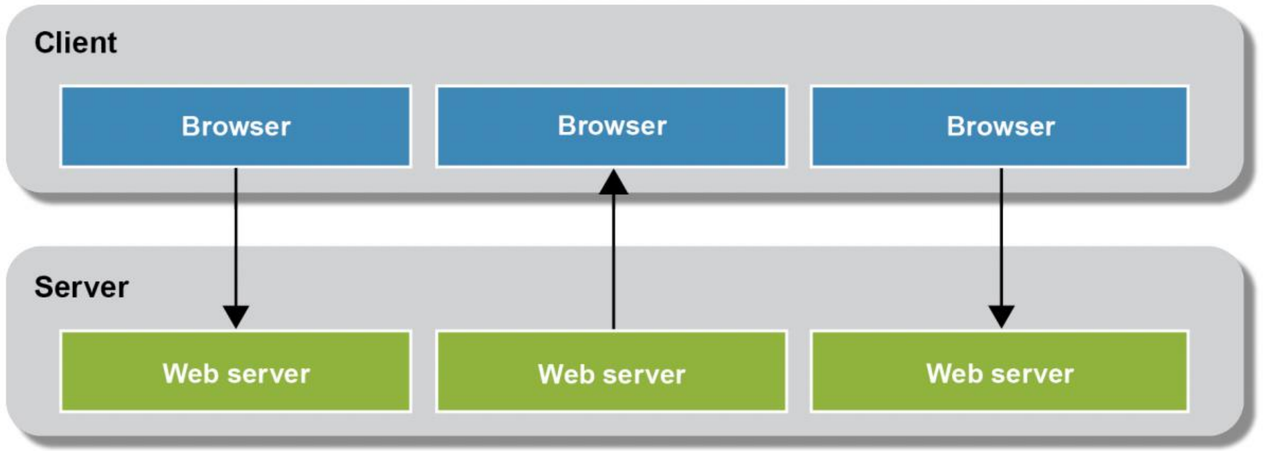
How to start a session

- A session starts when a browser makes its first request to a page that includes the `session_start()` function.
- By default, a session uses a session cookie to associate a browser with the data for its session. However, you can use the `session_set_cookie_params()` function to customize the cookie for the session.
- In the `session_set_cookie_params()` function, the `$lifetime` parameter is the only required parameter. The other parameters are optional.
- The `session_set_cookie_params()` function must be called before the `session_start()` function.

How to work with sessions

- HTTP is a stateless protocol.
  - Once a browser makes a request, it drops the connection to the server.
  - To maintain state, a web application uses session tracking.
- By default, PHP uses a cookie to store a session ID in each browser.
  - Then, the browser passes the cookie to the server with each request.
- To provide session tracking when cookies are disabled in the browser, you can use URL encoding to store the session ID in the URL for each page of an application.
  - Not recommended

Why session tracking is difficult with HTTP

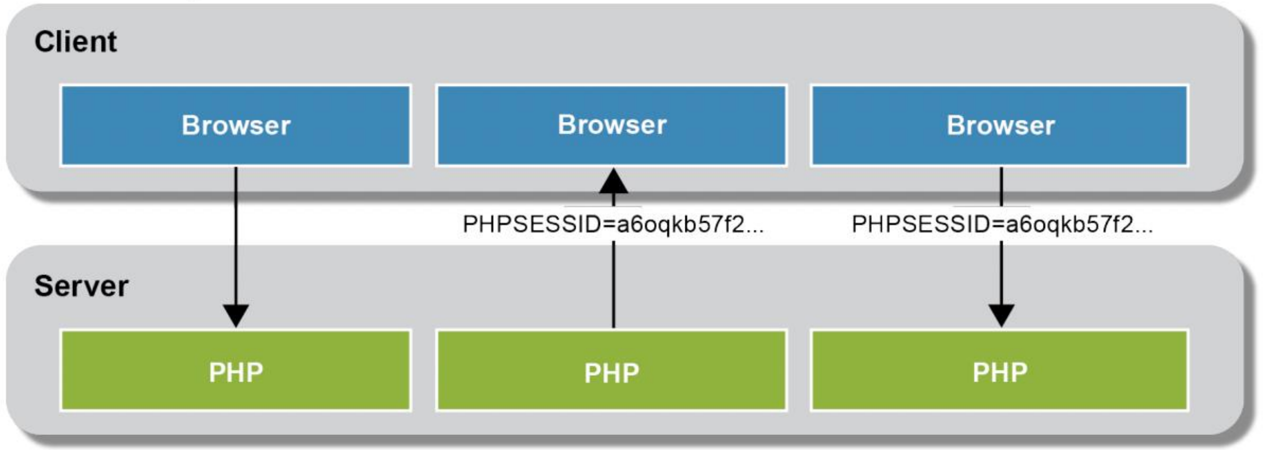


First HTTP Request:
The browser requests a page.

First HTTP Response:
The server returns the requested page and drops the connection.

Following HTTP Requests:
The browser requests a page. The web server has no way to associate the browser with its previous request.

How PHP keeps track of sessions



First HTTP Request:
The browser requests a PHP page. PHP creates a session and assigns it an ID.

First HTTP Response:
The server returns the requested page and the ID for the session as a cookie.

Following HTTP Requests:
The browser requests a PHP page and sends the session ID cookie. PHP uses the session ID to associate the browser with its session.

How to start a session

Function	Description
`session_start()`	Starts a new session or resumes a previous session. Returns TRUE if successful and FALSE otherwise. This function must be called before the page sends any HTML output to your application.

Start a session with the default cookie parameters

```
session_start();
```

Set cookie parameters

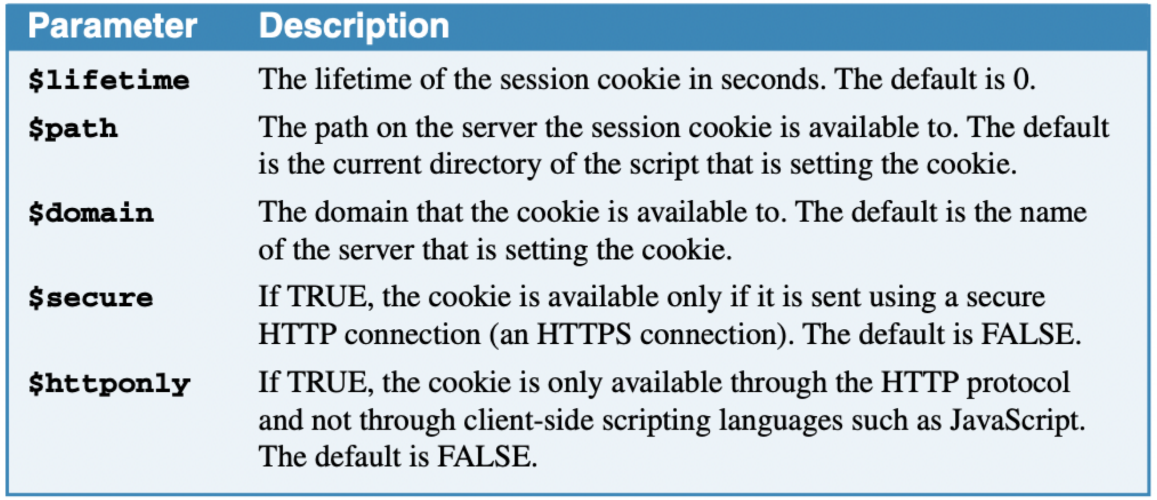
The syntax of the session_set_cookie_params() function

```
session_set_cookie_params($lifetime, $path, $domain, $secure, $httponly)
```

Start a session with custom cookie parameters

```
$lifetime = 60 * 60 * 24 * 365; // 1 year in seconds
session_set_cookie_params($lifetime, '/’);
session_start();
```

The parameters of the session_set_cookie_params() function



Parameter	Description
`$lifetime`	The lifetime of the session cookie in seconds. The default is 0.
`$path`	The path on the server the session cookie is available to. The default is the current directory of the script that is setting the cookie.
`$domain`	The domain that the cookie is available to. The default is the name of the server that is setting the cookie.
`$secure`	If TRUE, the cookie is available only if it is sent using a secure HTTP connection (an HTTPS connection). The default is FALSE.
`$httponly`	If TRUE, the cookie is only available through the HTTP protocol and not through client-side scripting languages such as JavaScript. The default is FALSE.

Set and get session variables

- Once you start a session, you can use the superglobal `$_SESSION` variable to set and get the user's data for a session.
  - This variable is an associative array.
- If necessary, you can use the `isset()` function to test if an element already exists in the `$_SESSION` array.
- You can use the `unset()` function to remove an element from the `$_SESSION` array.
  - However, don't use the unset() function on the `$_SESSION` array itself as it can cause unpredictable results.
- You can set the `$_SESSION` array to an empty array to remove its contents.

How to set and get scalar variables

Set a variable in a session

```
$_SESSION['product_code'] = 'MBT-1753’;
```

Get a variable from a session

```
$product_code = $_SESSION['product_code’];
```

How to set and get arrays

Set an array in a session

```
if (!isset($_SESSION['cart'])) {
  $_SESSION['cart'] = array();
}
```

Add an element to an array that’s stored in a session

```
$_SESSION['cart']['key1'] = 'value1’;
$_SESSION['cart']['key2'] = 'value2';
```

Get and use an array that’s stored in a session

```
$cart = $_SESSION['cart’];
foreach ($cart as $item) {
  echo '<li>' . $item . '</li>';
}
```

How to remove variables from a session

Remove a session variable

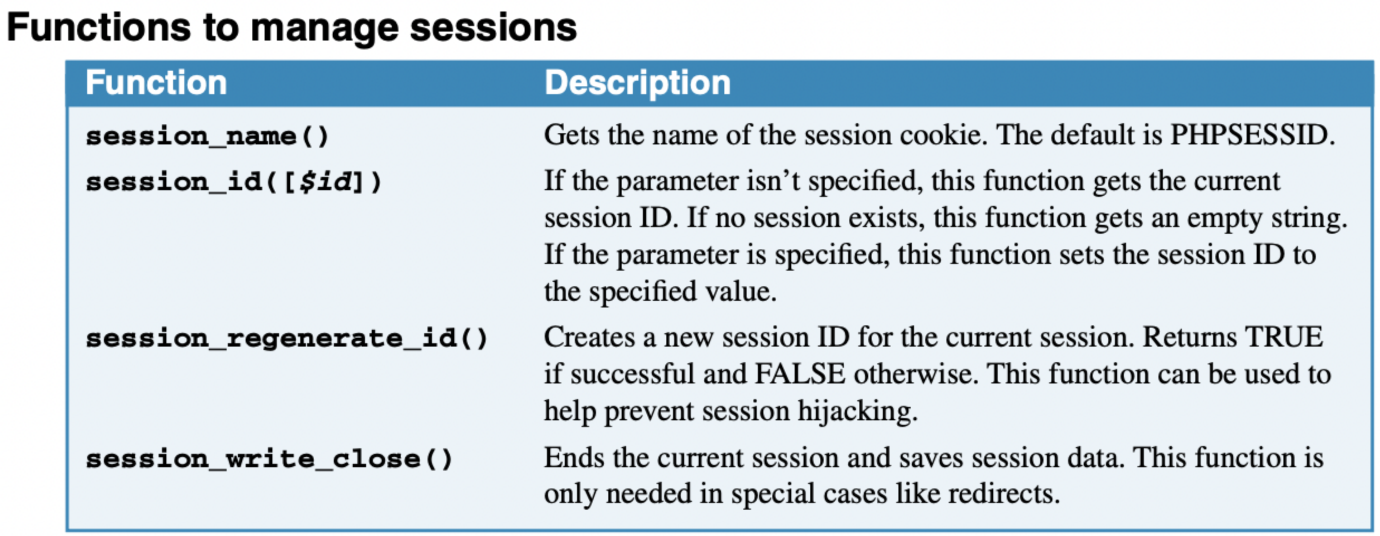
```
unset($_SESSION['cart']);
```

Remove all session variables

```
$_SESSION = array();
```

Managing a session

- Typically you won’t need these function, but they can be useful



Functions to manage sessions

Function	Description
`session_name()`	Gets the name of the session cookie. The default is PHPSESSID.
`session_id([$id])`	If the parameter isn’t specified, this function gets the current session ID. If no session exists, this function gets an empty string. If the parameter is specified, this function sets the session ID to the specified value.
`session_regenerate_id()`	Creates a new session ID for the current session. Returns TRUE if successful and FALSE otherwise. This function can be used to help prevent session hijacking.
`session_write_close()`	Ends the current session and saves session data. This function is only needed in special cases like redirects.

Get the name of the session cookie

```
$name = session_name();
```

Get the value of the session ID

```
$id = session_id();
```

Set the session ID

```
session_id('abc123');
```

How to end a session

- A session ends when the user closes the browser, when a specified amount of time elapses without a request, or when the code calls the `session_destroy()` function.
- To remove all data associated with the session from the client and the server, you can clear the session data from memory, call the `session_destroy()` function, and use the `setcookie()` function to delete the session cookie.
- The `session_name()` function gets the name of the session cookie. By default, the session cookie has a name of “`PHPSESSID`”.
- The `session_get_cookie_params()` function gets an associative array that contains all of the parameters for the session cookie.

End a session

Function	Description
`session_destroy()`	Ends a session. Returns TRUE if successful and FALSE otherwise.

End a session

```
$_SESSION = array();          // Clear session data from memory
session_destroy();            // Clean up the session ID
```

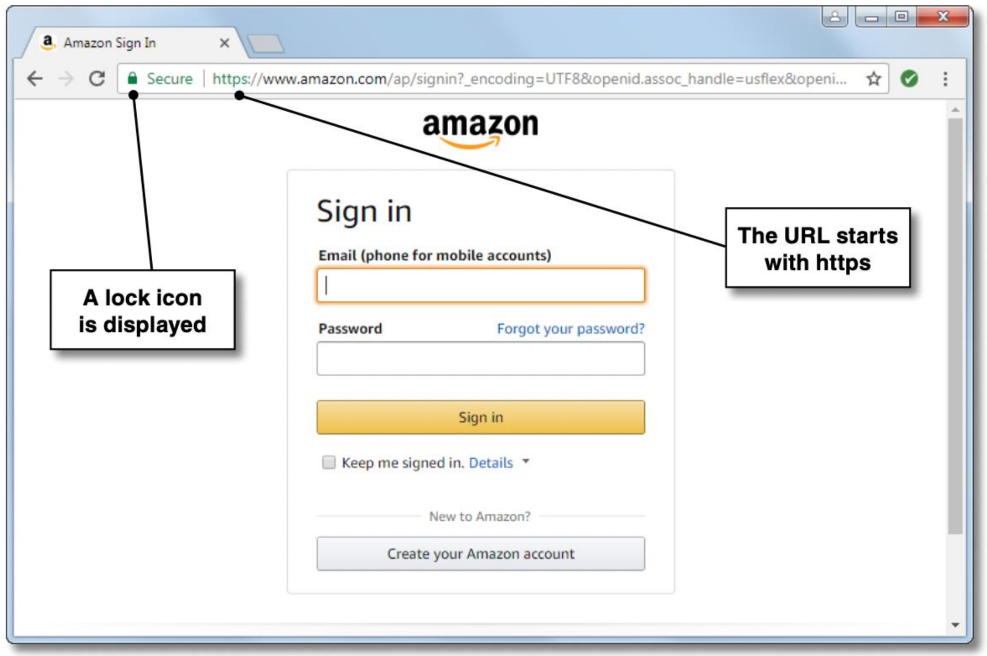
Delete the session cookie from the browser

```
$name = session_name();      // Get name of session cookie
$expire = strtotime('-1 year’); // Create expire date in the past
$params = session_get_cookie_params(); // Get session params
$path = $params['path’];
$domain = $params['domain’];
$secure = $params['secure’];
$httponly = $params['httponly'];
setcookie($name, '', $expire, $path, $domain, $secure, $httponly);
```

Creating secure website

Secure connections

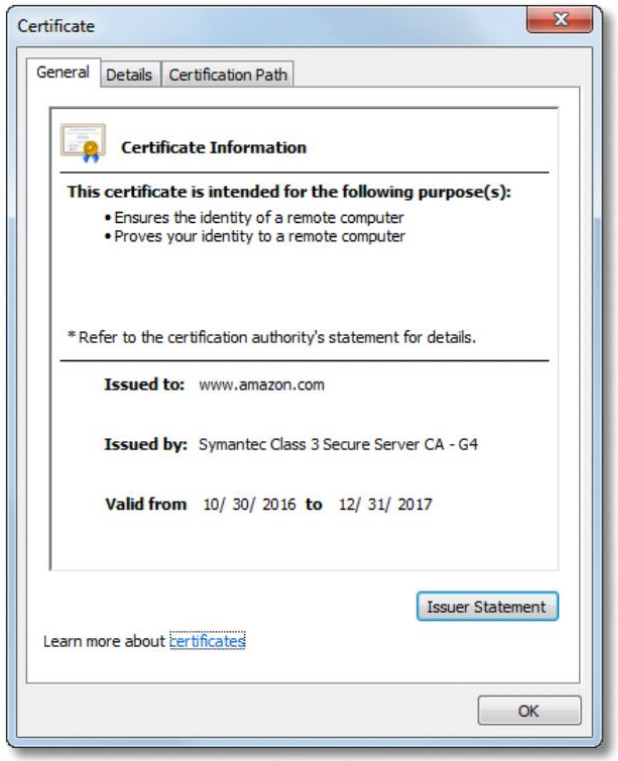
- https is secure connection, generally, a small lock icon will appears to the left of URL
- With a secure connection, all data is encrypted before it is transferred between the client and server
- SSL (Secure Sockets Layer)
    - older Internet protocol that transfer data on secure connection
- TLS (Transport Layer Security)
    - Newer protocol for working with secure connection
- Although there are slight differences between SSL and TLS, the protocol remains substantially the same. As a result, they are sometimes referred to as the TLS/SSL protocol or these terms are used interchangeably.


Amazon Sign In
Secure
https://www.amazon.com/ap/signin?_encoding=UTF8&openid.assoc_handle=usflex&openi...
amazon
Sign in
Email (phone for mobile accounts)
Password
Forgot your password?
Sign in
Keep me signed in. Details
New to Amazon?
Create your Amazon account
A lock icon is displayed
The URL starts with https

SSL authentication

- **Authentication**: the process of determining whether a server or client is who and what it claims to be.
- When a browser makes an initial attempt to communicate with a server over a secure connection, the server authenticates itself by providing a digital secure certificate.
- If the digital secure certificate is registered with the browser, the browser won't display the certificate by default. However, the user still has the option to view the certificate.
- In some rare cases, the server may request that a client authenticate itself by presenting its own digital secure certificate.

A digital secure certificate



Certificate Information

This certificate is intended for the following purpose(s):
- Ensures the identity of a remote computer
- Proves your identity to a remote computer

* Refer to the certification authority's statement for details.

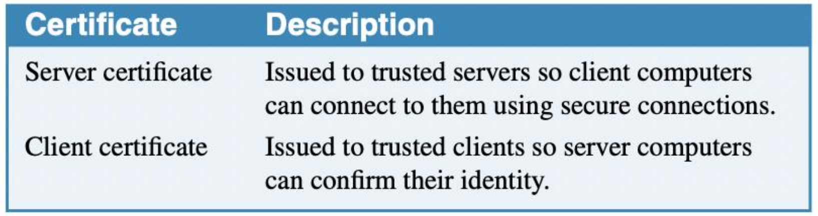
Issued to: www.amazon.com

Issued by: Symantec Class 3 Secure Server CA - G4

Valid from 10/ 30/ 2016 to 12/ 31/ 2017

Learn more about certificates

Types of digital secure certificates



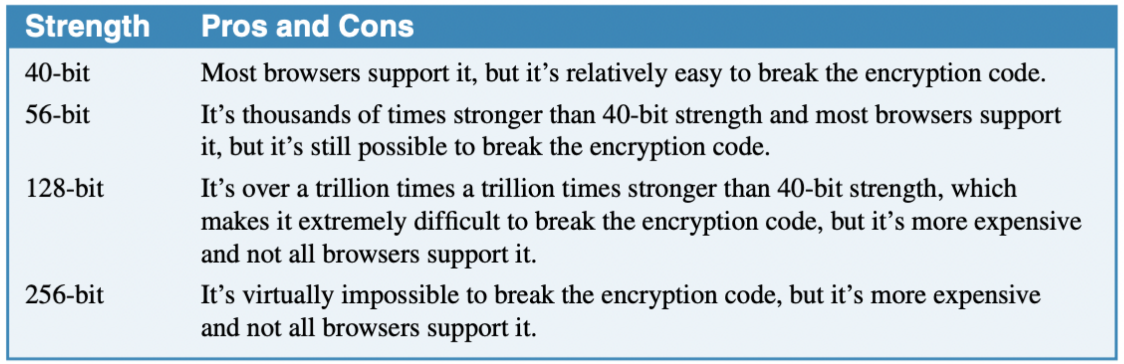
Certificate	Description
Server certificate	Issued to trusted servers so client computers can connect to them using secure connections.
Client certificate	Issued to trusted clients so server computers can confirm their identity.

A digital secure certificate

Common certificate authorities that issue digital secure certificates

- `www.symantec.com/ssl-sem-page`
- `www.godaddy.com/ssl`
- `www.globalsign.com`
- `www.startcom.org`
- `www.comodo.com`

SSL strengths



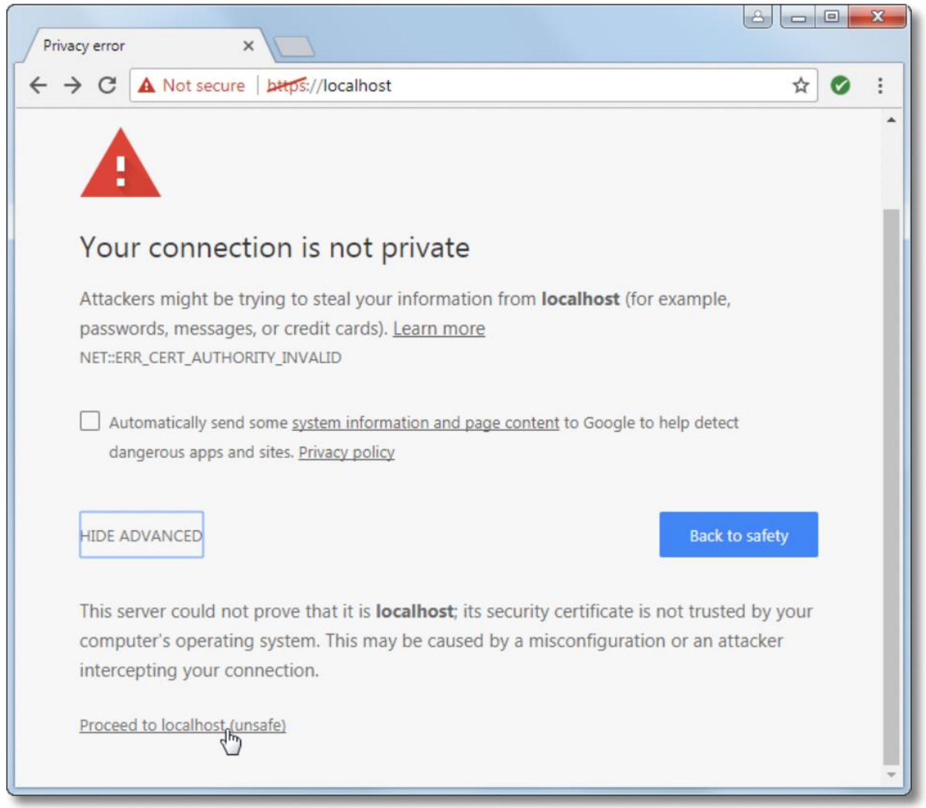
Strength	Pros and Cons
40-bit	Most browsers support it, but it's relatively easy to break the encryption code.
56-bit	It's thousands of times stronger than 40-bit strength and most browsers support it, but it's still possible to break the encryption code.
128-bit	It's over a trillion times a trillion times stronger than 40-bit strength, which makes it extremely difficult to break the encryption code, but it's more expensive and not all browsers support it.
256-bit	It's virtually impossible to break the encryption code, but it's more expensive and not all browsers support it.

- To use SSL, you need to purchase a digital secure certificate from a *trusted certification authority* (CA)
- CA issues and manages security credentials, it check with registration authority to verify information provided by the requestor
- SSL strength- the certificate strength
- Use 128-bit or better strength for payment processing or personal info collection

Requesting a secure connection

- To request a secure connection, use an absolute URL that starts with https.
- Once you establish a secure connection, you can use relative URLs to continue using the secure connection.
- To return to a regular HTTP connection after using a secure connection, use an absolute URL that starts with http.

A warning page for the security certificate



Your connection is not private

Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

☐ Automatically send some system information and page content to Google to help detect dangerous apps and sites. Privacy policy

This server could not prove that it is **localhost**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to localhost (unsafe)

Authentication

Types of authentication

- Form-based authentication
  - Allows the developer to code a login form that gets the username and password.
  - Allows the developer to only request the username and password once per session.
  - By default, it **doesn’t encrypt** the username and password before sending them to the server.
- Basic authentication
  - Causes the browser to **display a dialog box** that gets the username and password.
  - Requires the browser to send the username and password for every protected page.
  - By default, it **doesn’t encrypt** the username and password before sending them to the server.
- Digest authentication
  - Causes the browser to **display a dialog box** that gets the username and password.
  - **Encrypts** the username and password before sending them to the server.
  - Not as secure as using a secure connection
  - Not use as often as other types

Authentication

- *hash function* accepts a variable-size string, uses an algorithm to encrypt it and return a fixed-size string known as a hash.
  - One-way functions that are only designed to encrypt, not decrypt.
  - The original string is sometimes called the *message*, and the hash is sometimes called the *digest*.
- *salt:* some random data that's added to the password before the password is hashed.
  - Makes the hash stronger and more difficult to decrypt.
- *cipher:* a type of algorithm that's used in cryptography.
- *bcrypt* algorithm: a hashing algorithm based on the *Blowfish* cipher.
- Best practice to use the `password_hash()` and `password_verify()` functions to hash passwords and to verify whether the hashes are valid.

Storing and validate a password

Functions for working with passwords

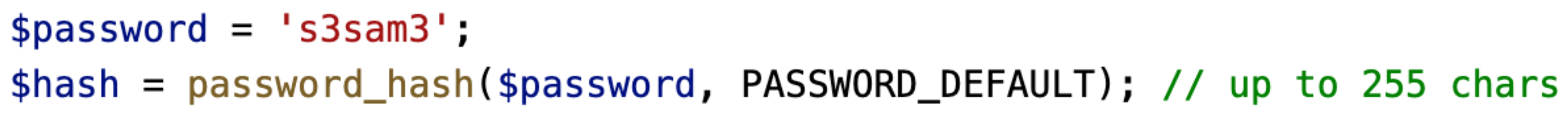
Function	Description
`password_hash($password, $algorithm)`	Creates a new hash of the password using a strong salt and a strong one-way encryption algorithm.
`password_verify($password, $hash)`	Returns TRUE if the specified password matches the specified hash.

Constants for setting the algorithm for password

Constant	Description
`PASSWORD_BCRYPT`	Uses the bcrypt algorithm to create a hash that's 60 characters long.
`PASSWORD_DEFAULT`	Uses the default algorithm of the **password_hash()** function. With PHP 5.5 and 7.1, the default algorithm is **bcrypt**. However, this default may change as newer and stronger algorithms become available. This may cause the number of characters in the hash to increase beyond the current 60 characters.

Password

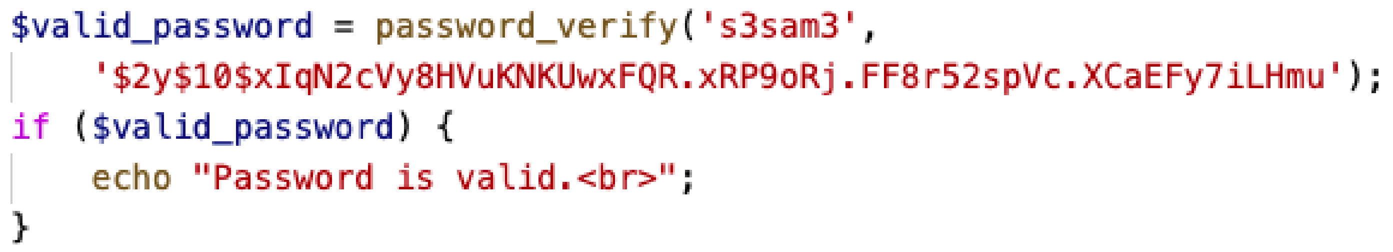
Code that hashes a password using the default algorithm



```php
$password = 's3sam3';
$hash = password_hash($password, PASSWORD_DEFAULT); // up to 255 chars
```

Create a hash that is 60 characters long

Code that verifies whether a password is valid



```php
$valid_password = password_verify('s3sam3',
    '$2y$10$xIqN2cVy8HVuKNKUwxFQR.xRP9oRj.FF8r52spVc.XCaEFy7iLHmu');
if ($valid_password) {
    echo "Password is valid.<br>";
}
```

Demo: import `sqlfordemo.sql` to your **localhost** (xampp)
*will delete database my_guitar_shop2

Store password

- The SQL script for creating a table for storing usernames and passwords

```
CREATE TABLE administrators (
  adminID            INT            NOT NULL   AUTO_INCREMENT,
  emailAddress       VARCHAR(255)   NOT NULL,
  password           VARCHAR(255)   NOT NULL,
  firstName          VARCHAR(255)   NOT NULL,
  lastName           VARCHAR(255)   NOT NULL,
  PRIMARY KEY (adminID)
);
```

Can store password even if hashed

```
INSERT INTO administrators (adminID, emailAddress, password, firstName, lastName) VALUES
(1, 'admin@myguitarshop.com',
'$2y$10$lHqybsUxtrV/y6j6WfG3.utNzpVTkNCm/neRFPnaaQiBWOJVIIEiq', 'Admin', 'User'),
(2, 'joel@murach.com',
'$2y$10$.imVkbsvI2XTC13bMONdUOllyhddj/IhYZBGU87nqZ1j8ebXPezre', 'Joel', 'Murach'),
(3, 'mike@murach.com',
'$2y$10$21KIM2059gSrnAQWV.5Ciufzo9sNqONmmzIhE8qvd/IDaeQvHG1Eq', 'Mike', 'Murach');
```

Demo: run application admin in **localhost**

admin_db.php

Work with administrators table

```php
<?php
function add_admin($email, $password) {
    global $db;
    $hash = password_hash($password, PASSWORD_DEFAULT);
    $query = 'INSERT INTO administrators (emailAddress, password)
              VALUES (:email, :password)';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->bindValue(':password', $hash);
    $statement->execute();
    $statement->closeCursor();
}

function is_valid_admin_login($email, $password) {
    global $db;
    $query = 'SELECT password FROM administrators
              WHERE emailAddress = :email';
    $statement = $db->prepare($query);
    $statement->bindValue(':email', $email);
    $statement->execute();
    $row = $statement->fetch();
    $statement->closeCursor();
    if ($row && is_array($row)) {
        $hash = $row['password'];
        return password_verify($password, $hash);
    }
    return false;
}
?>
```

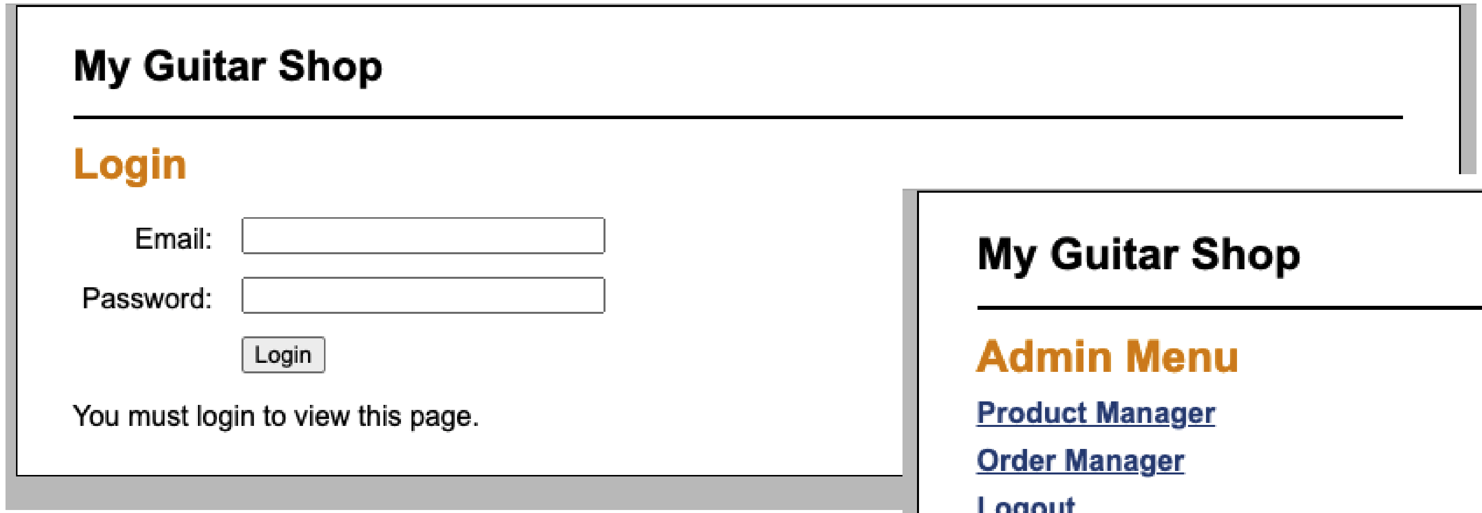
Encrypt the password

Use $hash for password

If the user has already enter username and password information, compares the password supplied by the user with the encrypted password stored in the database

Form-based authentication

- Code a form within an HTML or PHP file that gets the username and password
- When the user click Login button, the username and password are sent to the server



My Guitar Shop

Login

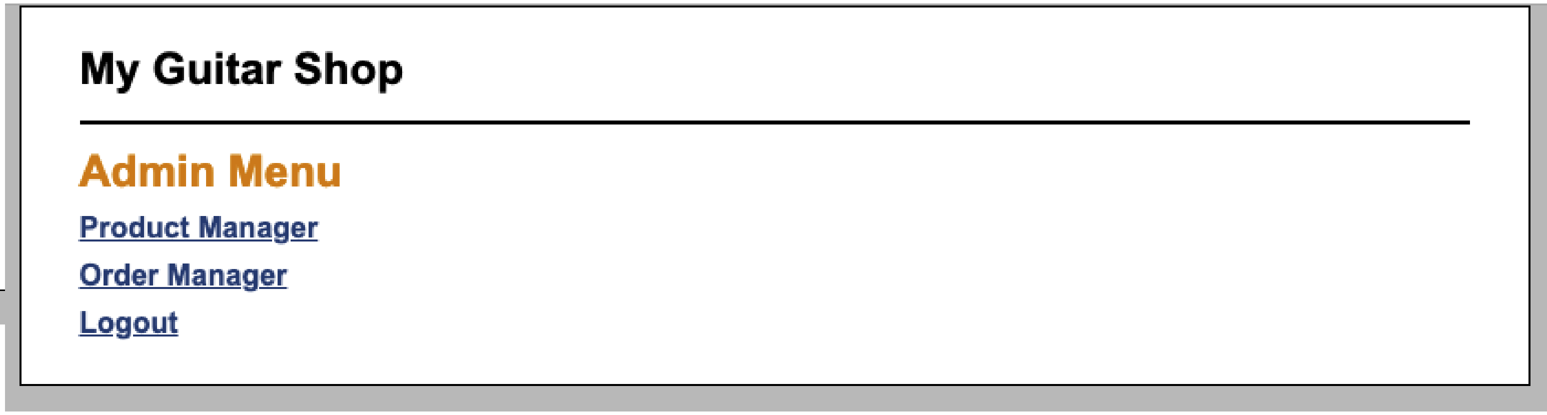
Email:

Password:

Login

You must login to view this page.

When login is successful



My Guitar Shop

Admin Menu

Product Manager

Order Manager

Logout

Proposal of your final project

Each group to join zoom and share your screen using Zoom

End of class 11

Homework 8