Notes:

* In oop separation of concern is so much important the same model or class related code should be placed in the same class or model.
* We make a function if the same functionality or same code need again and again even if the code is one line block.
* Every complexity should be placed in the object or class or model.
* Always sanitize user input like query string etc.

Keep in mind

* Suppose we want to echo the value of the input field to avoid the retyping if form submission is failed
* Then we assign to the value field variable in the else statement of the

POST variable otherwise it gives the message of undefined variable or other unwanted errors.

Web Server

* A web server is simply a computer program that dispenses web pages as they are requested.

Return True or False in a Function

* Return means stop the process now return the instructed thing.

Array Functions:

* Shifting and unshifting works at the beginning and pops and push works at the last.

Dates & Times

* Epoch time/Unix timestamp
* Is a way to track time as a running total of seconds?
* Coordinated Universal Time (UTC)
* Zero is January 1, 1970
* 32bit = December 10902 to January 2037
* 64bit = Billions of years
* Make a Unix timestamp
* Time() // current time
* Mktime($hr,$min,$sec,$mo,$day,$yr) //return in seconds
* Strtotime($any\_string)
* Format a Unix Timestamp
* Date($format, $timestamp)
* Strftime($format, $timestamp)

References

* References in PHP is to access the same variable content by different names.
* Note that in PHP, variable name and variable content are different, so the same content can have different names.
* References allow variables which allow same content of the others variable.
* Alias to the variable.

Static

* When we work with :: that means we work with something like static
* When we use this i.e. we work with instance.

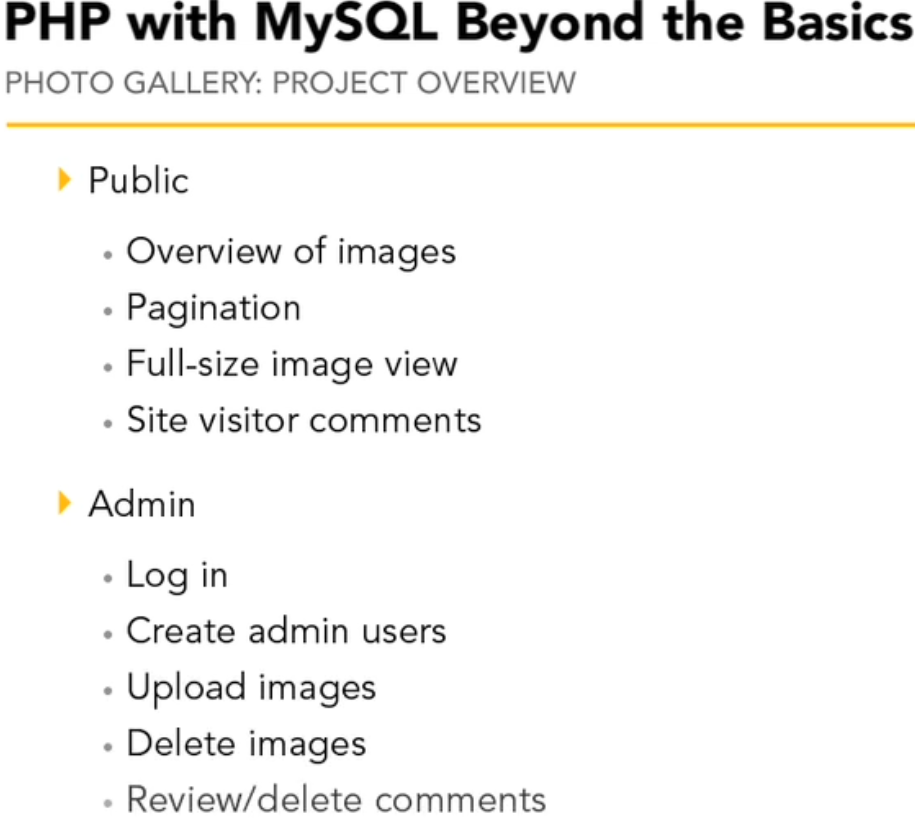
Constructors & Destructors

* Both are special methods automatically call when an object is created or destroyed.

Comparing Objects

|  |  |  |
| --- | --- | --- |
|  | = = | = = = |
| References | Yes | yes |
| Instances with matching attributes | Yes | no |
| Instances with different attributes | No | no |

OOP project overview:



Objects:

* User
* Photograph
* Comment
* Database
* Session
* Pagination

Some Useful Functions:

* addslashes — Quote string with slashes
* Returns a string with backslashes before characters that need to be escaped. These characters are single quote ('), double quote ("), backslash (\) and NULL (the NULL byte).
* stripslashes — Un-quotes a quoted string
* get\_object\_vars ($object) -- This function gets the properties of the given object.
* Returns an associative array of defined object properties for the specified object. If a property has not been assigned a value, it will be returned with a NULL value.
* get\_class\_vars() will expose default public variables (or private/protected if called within the class)
* get\_object\_vars() will expose the current public variables (or private/protected if called within the class object)
* get\_objcet\_vars() return all of the attributes of the class.
* property\_exists — Checks if the object or class has a property
* This function checks if the given property exists in the specified class
* As opposed with isset(), property\_exists() returns TRUE even if the property has the value NULL.
* strip\_tags()
* The strip\_tags() function strips a string from HTML, XML, and PHP tags.
* Note: HTML comments are always stripped. This cannot be changed with the allow parameter.
* Note: This function is binary-safe.
* We can also specify tags that should be allowed, for example <div><b><p>.
* array\_shift()
* The array\_shift() function removes the first element from an array, and returns the value of the removed element.
* Note: If the keys are numeric, all elements will get new keys, starting from 0 and increases by 1
* Example

<?php

$a=array(0=>"red",1=>"green",2=>"blue");

echo array\_shift($a);

print\_r ($a);

?>

* <?php

$a=array("a"=>"red","b"=>"green","c"=>"blue");

echo array\_shift($a)."<br>";

print\_r ($a);

?>

* Returns the value of the removed element from an array, or NULL if the array is empty.

Setter & Getter

* Sets something and get something.

Autoload:

* Standalone function access outside of an object.
* The \_\_autoload() function will internally search out the class and load its definition.
* Each class file should be named the same as the class definition itself. eg: a class definition file named fax would have the filename fax.class.php
* Autoload function giving the safeness if class file is not included unwillingly.
* Autoload function execute independently outside of the object not inside the object.
* Autoload - The undeclared object safetyness.
* Many developers writing object-oriented applications create one PHP source file per class definition. One of the biggest annoyances is having to write a long list of needed includes at the beginning of each script (one for each class).
* In PHP 5, this is no longer necessary. You may define an \_\_autoload() function which is automatically called in case you are trying to use a class/interface which hasn't been defined yet. By calling this function the scripting engine is given a last chance to load the class before PHP fails with an error.

Session:

* Session is going to be maintained in server as file.
* A session is a way to store information (in variables) to be used across multiple pages.
* Session variables hold information about one single user, and are available to all pages in one application
* If you need a permanent storage, you may want to store the data in a database.
* The session\_start() function must be the very first thing in your document. Before any HTML tags.
* Session variables are not passed individually to each new page, instead they are retrieved from the session we open at the beginning of each page (session\_start()).
* When you work with an application, you open it, do some changes, and then you close it. This is much like a Session. The computer knows who you are. It knows when you start the application and when you end. But on the internet there is one problem: the web server does not know who you are or what you do, because the HTTP address doesn't maintain state.
* Session variables solve this problem by storing user information to be used across multiple pages (e.g. username, favorite color, etc). By default, session variables last until the user closes the browser.
* When you call session\_start(), PHP will check to see whether the visitor sent a session cookie - if it did, PHP will load the session data. Otherwise, PHP will create a new session file on the server, and send an ID back to the visitor to associate the visitor with the new file. Because each visitor has their own data locked away in their unique session file, you need to call session\_start() before you try to read session variables - failing to do so will mean that you simply will not have access to their data. Furthermore, as session\_start() needs to send the reference cookie to the user's computer, you need to have it before the body of your web page - even before any spaces.

Magic Quotes:

* Created to help protect newbie programmers from writing bad form processing code.
* Magic quotes would automatically escape risky form data that might be used for SQL Injection with a backslash \.
* When on, all ' (single-quote), " (double quote), \ (backslash) and NULL characters are escaped with a backslash automatically.
* get\_magic\_quotes\_gpc — gets the current configuration setting of magic\_quotes\_gpc.

Static Binding

* Avoid static methods

$user = User::find\_by\_id(1);

* We use instance method instead

$user = new User();

$user = $User->find\_by\_id(1);

Deduce the class name

* get\_class()
* Even if we use get\_class unfortunately in this case returns DatabaseObject

Late Static Bindings

* Is better approach as compare to static bindings
* get\_called\_class() identify which class actually called.
* Which will the class name and make instance of the class.
* Simple way use

Static::instead of self::

* Push all of the common methods in the common class
* Inheritance those class into our sub class.
* In such way reuse same method over and over by different classes
* Load core objects before use a class.

Upload File

* Upload\_tmp\_dir – upload file temporarily in this directory
* We put the file in this directory then grab and put out from their works with those files afterwards.
* Maximum size for any post request 8M by default maximum uploaded file size – 2M
* Memory limit – how long memory each script allowed to use. By default memory\_limit 128 M.
* Upload Folders – to upload for all of our uploaded files.
* max\_execution\_time – how long PHP expands trying to accomplish this task such as 30 seconds.
* All uploaded files will be found in the $\_FILES superglobal variable not in the $\_POST variable.
* Basename - Returns trailing name component of path
* The basename() function returns the filename from a path.
* move\_uploaded\_file — Moves an uploaded file to a new location
* Generally to upload a file follow to steps
* Store the file in permanent location
* Entry the file record in database like file-name, type, size, caption etc.

Abstracting

* All of the common methods and properties should be pushed in the common model or class.
* self::$table\_name that means abstracting database table name

Session Message

* Session hang on a value and keep them around for us.
* Check\_message() checks whether previously set a message or not.
* When instantiating the session class then message attributes get the values from the session this will allow us set the session message as soon as in the message.
* Remember We create the instance automatically just below in our session class
* So that new instances created it will assign session value to the message variable.

Functionality of Session message in our project

* We populate the session message in photo\_upload.php
* When the page is redirected to list\_photos.php then the message is set in the session variable.
* After redirecting happen in the list\_photos.php then check\_message automatically called because object is called of session class as soon as session page is loaded.
* Then the attribute message gets the value from the session variable after that session is destroyed.
* In that case our message property get the same value when redirection occurred.

Pagination

* Suppose we have one thousand photograph in a database
* We don’t need to see the all photographs once
* Instead we want subset of the list to the user a single page
* Like a single page might have 10 photographs
* And the page links to the both next and previous page

Pagination requires two sets of information

* The subset of records on the current page
* Information about the complete set of records

To jump any arbitrary page

Keep track of three variables

* Current page ($current\_page)
* Records per page ($per\_page)
* Total record count ($total\_count)

LIMIT, OFFSET, AND COUNT

LIMIT

* Maximum records to return
* $per\_page

OFFSET

* Records to skip before returning records
* $per\_page\*($current\_page – 1)

OFFSET Examples

* $per\_page = 10
* $offset = $per\_page\*($current\_page-1)

|  |  |  |
| --- | --- | --- |
| $current\_page | $offset | Records |
| 1 | 0 | 1-10 |
| 2 | 10 | 11-20 |
| 3 | 20 | 21-30 |
| 4 | 30 | 31-40 |
| 5 | 40 | 41-50 |

SQL for Offset

* SELECT \* FROM photographs LIMIT $per\_page OFFSET $offset;

COUNT

* Is very straightforward which is returned how many records are in the database?
* SELECT COUNT(\*) FROM photographs;

Sending Emails

php.ini Mail Settings

4 important things for sending a mail

* SMTP(Simple Mail Transfer Protocol)
* smtp\_port
* sendmail\_from
* sendmail\_path

sendmail

* Real simple application that doesn’t have graphical interface that’s sends out email; php use sendmail to get email from the computer that sends out to the internet.
* Generally it commented in beginning because of pc will spam or virus affected by hacker using sendmail.

Sending Email in PHP

* mail($to, $subject, $message, $headers);

Templating in PHP

* Set variables/include PHP file
* Load file/search and replace

SMTP Overview

* Simple Mail Transfer Protocol (SMTP) is an Internet standard for electronic mail (email) transmission.
* SMTP by default uses TCP port 25.
* The protocol for mail submission is the same, but uses port 587.
* SMTP is part of the application layer of the TCP/IP protocol. Using a process called "store and forward," SMTP moves your email on and across networks. It works closely with something called the Mail Transfer Agent (MTA) to send your communication to the right computer and email inbox.
* Although electronic mail servers and other mail transfer agents use SMTP to send and receive mail messages, user-level client mail applications typically use SMTP only for sending messages to a mail server for relaying. For receiving messages, client applications usually use either POP3 or IMAP.
* Simple Mail Transfer Protocol – nice and simple way to send email across on the internet in fact it’s backbone of the email to the internet basically it grabs the email and identify where the email is next send and pass it along to another SMTP server same things finally our isp this belongs this user account and then put the message into our inbox.
* Essential SMTP Info
* Server/Host
* Port
* Authentication
* Username
* Password
* SMTP: SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, which let the user save messages in a server mailbox and download them periodically from the server. In other words, users typically use a program that uses SMTP for sending e-mail and either POP3 or IMAP for receiving e-mail. On Unix-based systems, sendmail is the most widely-used SMTP server for e-mail. A commercial package, Sendmail, includes a POP3 server. Microsoft Exchange includes an SMTP server and can also be set up to include POP3 support.
* POP3 (Post Office Protocol 3)

POP3 (Post Office Protocol 3) is the most recent version of a standard protocol for receiving e-mail. POP3 is a client/server protocol in which e-mail is received and held for you by your Internet server. Periodically, you (or your client e-mail receiver) check your mail-box on the server and download any mail, probably using POP3. This standard protocol is built into most popular e-mail products, such as Eudora and Outlook Express. It's also built into the Netscape and Microsoft Internet Explorer browsers.

* IMAP (Internet Message Access Protocol)

IMAP (Internet Message Access Protocol) is a standard protocol for accessing e-mail from your local server. IMAP (the latest version is IMAP Version 4) is a client/server protocol in which e-mail is received and held for you by your Internet server. You (or your e-mail client) can view just the heading and the sender of the letter and then decide whether to download the mail. You can also create and manipulate multiple folders or mailboxes on the server, delete messages, or search for certain parts or an entire note. IMAP requires continual access to the server during the time that you are working with your mail.

* A less sophisticated protocol is Post Office Protocol 3 (POP3). With POP3, your mail is saved for you in a single mailbox on the server. When you read your mail, all of it is immediately downloaded to your computer and, except when previously arranged, no longer maintained on the server.
* IMAP can be thought of as a remote file server. POP3 can be thought of as a "store-and-forward" service.
* POP3 and IMAP deal with the receiving of e-mail from your local server and are not to be confused with Simple Mail Transfer Protocol (SMTP), a protocol used for exchanging e-mail between points on the Internet. Typically, SMTP is used for sending only and POP3 or IMAP are used to read e-mail

Sending Email with PHPMailer

* Download PHP-Mailer from <http://phpmailer.worxware.com/>