Title: Conversations
Subtitle: PHP Streamlining Ideas
Second Subtitle: PHP Streamlining Ideas

Author: wizanda
Date: 1213379803
URL: https://www.wizanda.com/modules/newbb/viewtopic.php?topic_id=14

Here are a few ideas I've noticed that make a big difference to the speed of PHP code and the server overall....

Blank Space:
The biggest is blank space, indent, etc; this is all read by a computer as processed, so remove all indents and blank space, its not necessary and PHP....Don't believe this? Try removing just the indents from any file and server and see the difference in reading time...Especially in IE which se then FF....

Escaping Info Correctly:
There are two main ways to escape a line of code or instructions:
// Escapes the line it's on..
/* */ Escapes an Area of code or info..
Many people when including their own head or GPL data, make a common line individually (by using // on each line), this means when its read, it c line and has something that must be processed for each line, before the It is far better to have /* at the start of the head, and */ to close it at th only has one escape to read for multiple lines, this also means if someth the closing line */}, it doesn't vanish, like it can when accidental added

Including files:
Many people have no ideas about the differences between include or requ we have include_once or require_once as standard for most operations in
Include_once:
This performs the operation of including the file, at the time the item is
Require_once:
This must be included before the item can be performed.

So if you have functions that you know exists in other functions or class operation to perform, then these need requiring; else the operation can r collected from the include at the time of the operation (which of course w waits for the info it needs), which obviously is to late, as that operation have all constants, functions and classes before it can be operated.

The only time not to use _once is if an item will need multiple copies of 95% of the time, it is far safer to use _once, encase of multiple includin somehow lead around in a full circle.

Includes are there when items happen, so a header/footer is loaded at th
contents behind the header should be ready when it is asked for.

If !not defined CONSTANTS
This is something you would think would work and is often seen to escap
file/constant is loaded...
The problem with that though, is a CONSTANT is as the name suggests s
changing that means....
There is also numerous PHP tests/operations that were designed to do th
method_exists, function_exists, file_exists with these your asking for the
and so therefore taking less time to complete.

As mentioned previously we have include_once or require_once, this will
is there once over the whole system; else what happens if a file is asked
within both locations the constant won't be defined, as it is per file, yet
require_once will be per site.

The other major problem is these switches don't work as well as they cou
not exactly the same thing...
If !defined('CONSTANT'); = if not defined constant?
Define ('CONSTANT', ''); = constant equals this, so the question is still
been answered, it's been told it equals something...
Defined ('CONSTANT'); = that tells the original question; yes it is define
further investigation, to then understand as what or if it even is defined

Approximated Files
In most web development, if something comes from the same folder it doe
root of the server first; you can simply place it as a file name and extens
There are times in PHP this might not work, yet 95% of the time it works
of collecting the file is from different sources.
This saves vast amounts of resources in some cases, especially where pe
(dirname(_file_)).'/here.php' which can be just put as 'here.php', there a
where we try and collect a global constant, and go to the root path and t
to it...

Converting File Address or Not
Using constants to place words is what it's meant for, as its something c
of a file or directory, its a good practise as a name can change only onc
can be given in an address, if these are also maintained as defined ques
constant alive in the system and adds security if they are not defined ex
need to have a security code though to work for none inclusion of files v

Variables for server address though can be dangerous, as each time that
definition of the variables must be tested, as they are variables.
Basically this is saying keep it simple..
If a file is one up folder '../' same folder 'name.ext' you don't need to be
your approach to file locations, the reason for saying use this method is

folder name; it isn't inflicted by a million variables or even a constant cl
complete beginners, unless configured and defined by a system for them
Lastly and most importantly is, it takes far less time to go up one or two
what each item for the file location is.

Sometimes these paths are needed, for instances Tag's, where items can
and so need the exact folder name, yet not always the root path as it has
inline with the files asked for..
The more you can shorten a path, the quicker reaching it will be...Think
asking your self when you go on journey, you simplify it first...Not add e
if they are still there on your journey you make regularly....Imagine if yo
house, just to check if you have a house, to see if you have a bathroom

Server Address with Spaces
Though space has been mentioned, it is really noticeable if you leave sp
addresses of files you're trying to collect.
For instance:

```
PATH . '/modules' . $some["config"] . 'index.php'
```

You're basically asking it to convert all those spaces into code, there is
spaces, other then to slow your PHP code down.

```
PATH.'/modules'.$some["config"].'index.php'
```

To be continued.....