php.iniの話

builderscon tokyo 2016
2016/12/3

自己紹介

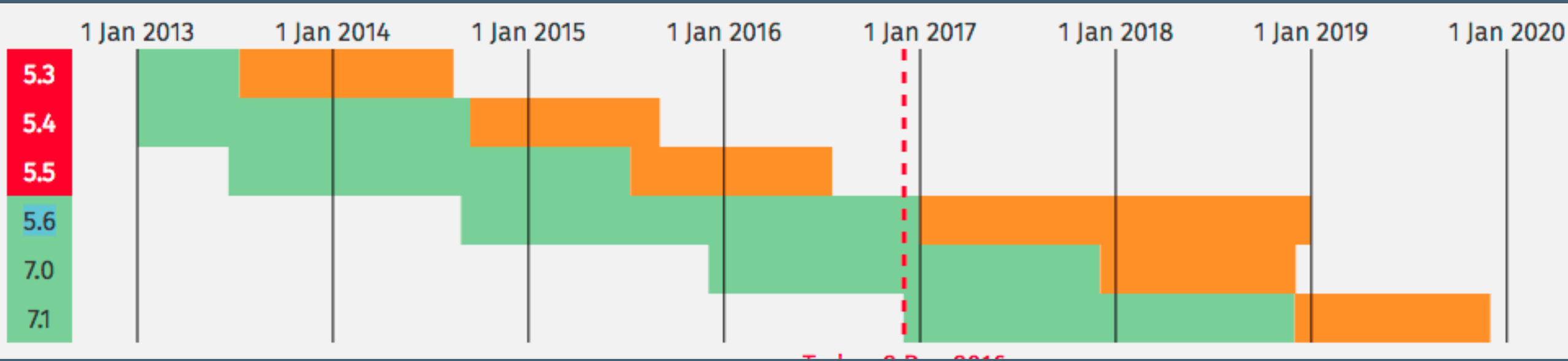
uzulla

そんなことより

PHP7.1.0

release 

- Nullable types
- Void return type
- Iterable pseudo-type
- Class constant visiblity modifiers
- Square bracket syntax for list() and the ability to specify keys in list()
- Catching multiple exceptions types


1 Jan 2013
1 Jan 2014
1 Jan 2015
1 Jan 2016
1 Jan 2017
1 Jan 2018
1 Jan 2019
1 Jan 2020
5.3
5.4
5.5
5.6
7.0
7.1

話しを戻します

えっ、php.iniの話を一時間？

MATSUO Masaru @localdisk 2016/10/17

php.iniで60分か。すごいな / php.iniについて知る - builderscon tokyo 2016 builderscon.io/builderscon/to...

1 4

「皆さんPHPを知っています」

» もはや知らないとは言わさない

» なので、PHPがどういうものなのかというのは省略

php.iniをしっていますか？

- php実行時の様々なスイッチ
- 内部の文字コードとか...
- 日付計算時のタイムゾーンとか...

文字コード関連での例

```
php > echo strlen("あいうえお");
15
```

» strlenはマルチバイト非対応関数

» UTF-8では 1 文字が3バイトなので、5文字が15バイトとして計算されている

勿論PHPはマルチバイトの文字列も読み書きできます。

```
php > echo mb_strlen("あいうえお");
5
```

» マルチバイト対応関数(mb関数)を使う事で正しく処理ができる

» ただし、世の中はUTF-8だけではない

» UTF-16LE, SJIS, EUC-JP …

mb関数は、扱う文字コードを設定から判断している

```
php > var_dump( ini_get("mbstring.internal_encoding"));
string(5) "UTF-8"
```

» だから、さきほどmb_strlenが正しく数えられた

» mbstring.internal_encodingというキーに、
"UTF-8"という値がはいっています。
これがphp.iniの設定です。

» こういう設定が、（環境によるが）ゆうに200個以上ある

設定を、UTF-8から変えてみる

```
php > echo mb_strlen("あいうえお");
5
php > ini_set("mbstring.internal_encoding", "SJIS-win");
php > echo mb_strlen("あいうえお");
8
```

>> ということで、内部処理をSJIS-win(cp932)にすると...

>> その場からエラーもなく、文字数が正しくカウントできなくなる。

「なんで言語にこんなスイッチがあるの？コードにかけばいいのでは？」

» （それも（ある程度）できますが）

» 差異をphp.iniに集め、コードは修正無く動かしたいという希望と、

» （実際、これはかなり成功している）

» 過去の互換性を保つための努力なのです。

» （たとえば昔はUTF-8なんて使われてなかったし）

まとめ

- PHPは設定ファイルがある
- 中には大量の設定スイッチがはいっている
- それは、キーと値の集合である

php.iniの歴史

php.iniの歴史

- php.iniはphp3から登場
  - 1998年の事
- 現在の最新はphp7.1.0
  - 今は2016年
- 18年の月日が流れています。

PHP3.0のphp.ini-dist（抜粋）

```
[PHP_3]
engine           =   On  ; enable PHP 3.0 parser
short_open_tag   =   On  ; allow the <? tag.  othe〜
precision        =   14  ; number of significant d〜
y2k_compliance   =   Off ; whether to be year 2000〜
safe_mode        =   Off
safe_mode_exec_dir   =
max_execution_time = 30      ; Maximum execution ti〜
memory_limit = 8388608       ; Maximum amount of me〜
error_reporting =   7
```

PHP7のini（抜粋

```
[PHP]
engine = On
short_open_tag = Off
precision = 14
serialize_precision = 17
zend.enable_gc = On
expose_php = On
max_execution_time = 30
max_input_time = 60
memory_limit = 128M
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
```

変わってないですね

- 皆さんが親しみある項目ばかり
  - safe_modeやy2k_compliance（懐かしい）とかは消えた
  - memory_limitのデフォルトとかは無論(?)増えた
  - error_reportingは定数化
- php3 213行 => php7 1933行
  - （コメントが特に増えた）

- 「php.iniの知識は20年使える💪」（だから？
  - たぶん php8,9,10も！（あるのか？
- 或いはPHP2020とかでもつかえるだろう
- 「(おそらく)20年以上使えるphp.iniを覚えるのはバリュー!(?)」
- 「なんとHHVMでもつかえる技術✌」
  - (.hdfというコンフィグは無かった事にされつつある)

ということで

» 皆さんの意欲は湧いてきました？🤗

» （意欲が湧いたら真面目に話し、意欲がなさそうなら…😿）

基礎知識

基礎知識

- phpはインタプリタ言語です
- ウェブアプリケーション開発に使われる前提です
- 今日はphpを
  - **実行エンジン**
  - **SAPI**
- で、わけて考えましょう

PHPの実行環境って色々ある

- Apache+mod_php
- Apache+CGI
- nginx+FastCGI
- IIS+FastCGI
- builtin server
- (もっとある)

PHPの凄い所

» これらバラバラの実行環境で

» **「同一コードのWordPressが動く」**

» というレベルの互換性がある

» つまり…

PHPは、コード修正無しに

12億

のサイトで動く！

「30億のデバイスで動くJavaには負けた（）」

（雑な事をいいました）

- 数字の理屈の出典
- Netcraft曰く、ネットのサイト数は約15億、
  - https://news.netcraft.com/archives/2016/11/22/november-2016-web-server-survey.html
- w3techs曰く、PHPシェアは82%
  - https://w3techs.com/
- （15億の80%が12億）

大分数字は盛りましたが

- 実際、PHPの99%の実行環境で
- あのWordPressやらが
- PHPのコード修正不要でうごく！
- **これこそPHP!!**

>> 「絶対だな？例外なし？」「ウッ…全部試した訳では無い…」

>> 「少なくとも、CLIは別…」「嘘つきだ！！」

（一部のみなさん）
「えっ、それって凄いの？」

» 「じゃあ、Perlで、mod_perlとCGIとPlackで1コードで同等に動作するウェブアプリの書いてよ！！！！」

» 「じゃあ、Rubyで（ｒｙ」

» 「じゃあ、じゃｂ（ｒｙ」

» 「（ｒｙ」

» （スーパーエンジニアなら書けるかもしれないけど、相当変なコードになる）

環境の差を埋めてるのがSAPI

- SAPIが、環境とPHPエンジンの間でとりなしている
- SAPIのお陰で、php（で書かれたプログラム）は
  - echoすればブラウザに出る
  - エラーがエラーログにでる
  - $_POSTや$_FILEでパラメタがとれる
  - 無心でセッションが使える（$_COOKIE等、httpヘッダ）
- ...といったことが普遍的に扱える

SAPIとは？

- Server API の略
- mod_php、CGI、FastCGI、CLI、等々
- 組み合わせるhttpdや環境に合わせて選ぶ
  - apacheならmod_php,CGI,FastCGI
  - nginxならFastCGI
  - IISならCGI,FastCGI

それぞれで結構違う

- 同じ「PHP」だけど、SAPI毎に違うバイナリだったりする
  - mod_phpは`libphp7.so`、CGIは`php-cgi`、CLIは`php`
  - （FastCGIとCGIみたいな例外もあるけど）

- httpdと同一プロセスでうごく （mod_php）
- 標準入出力でやりとりする （CGI,CLI）
- FastCGIプロトコルでやり取りする （FastCGI）
- httpでやり取りする ...のはなんと無い
  - （Builtin serverは開発用です）

>> (例外もあります)

>> (「PHP自体」がhttpdになるやつとか

>> (reactPHP

>> (Swoole

>> (AppServer.IO

>> (等

>> (でも、見た事ないでしょ？)

で、本題のphp.iniとSAPI、どう関係が？

- 色々をSAPIが吸収して、phpコードには互換性がある
- しかし、php.iniはそうもいかないのであった
  - サーバー構成が全然違う
  - 当然、設定の仕方も結構違う
  - 勿論、設定できる項目が多少違う
  - つまり、php.iniを知るにはSAPIを意識する必要がある

具体的には？

- 後述します
  - php.iniの場所が違う
  - 皆大好き.htaccessがなかったり
  - デフォルト値が変わる
  - 等

途中まとめ

» 様々な実行環境があるが、phpはどこでも同様に動くぞ！👍

» 差異はSAPI等が吸収している💪

» しかし、php.ini（やその周辺）はその限りではない😣

» （とはいえ、メジャーな環境の情報はググればすぐでてきます。
大人気PHPだもん）

» （"PHP による hello world 入門"という記事がとても良いです
http://tech.respect-pal.jp/php-helloworld/）

php.iniを見る

```
[PHP]
;;;;;;;;;;;;;;;;;;;
; About php.ini   ;
;;;;;;;;;;;;;;;;;;;
; PHP's initialization file, generally called php.ini, is responsible for
; configuring many of the aspects of PHP's behavior.
略

engine = On
expose_php = On
max_execution_time = 30
max_input_time = 60
```

どこにあるのか🔎

>> /etc 以下を探す

>> CLIで php --ini

>> phpinfo() の「Loaded Configuration File」と
「Additional .ini files parsed」

>> php_ini_loaded_file() と
php_ini_scanned_files();の返値

注意

- ✅「調べたい環境の、調べたいファイルでしらべろ！」
- ⚠️「CLIのphpと、貴方が調べたい環境は本当に同じphpか？」
  - まるっきり別のファイルなんて事がザラにある😨
  - SAPIでphp.iniの場所が指定できる
  - 他にもあるけど後述
- ⚠️非text/htmlなAPI等はphpinfo()は面倒だぞ（後述）

必然的に

» php_ini_loaded_file() と php_ini_scanned_files();の返値が、
安全だとおもわれます（個人的意見です）

» これを返値で取得し、どこかのファイルにでも書き出してください

» （🤔「面倒、注意すればphp --iniやphpinfoでいいのでは？」
😝「はい、ぼくもよくそれで調べます」）

調べたい所にコード挿入

```
$info = php_ini_loaded_file().",\n";
$info .= php_ini_scanned_files();
error_log($info);  // ファイルに出力
```

以下結果

```
/Users/uzulla/.phpenv/versions/7.0.1/etc/php.ini,
/Users/uzulla/.phpenv/versions/7.0.1/etc/conf.d/xdebug.ini,
/Users/uzulla/.phpenv/versions/7.0.1/etc/conf.d/my_special.ini
```

tips: デバッグログをどこかに出すなら...

- `file_put_contents('/tmp/info.txt', $info);`
  - 定番、楽ではある
  - 二回走ったら上書きされちゃう...（Appendする？）
  - `/tmp` に見つからない事も
    - 最近は private tmp というものがあってな...

>> phpにはerror_log()という便利関数がある

>> error_log($info);

>> 規定のエラーログにでます

>> （たとえば、apacheのerror_logに出る）

>> error_log($info, $filename); とファイル名指定もできます

>> 「長いと切り捨てられるので、その場合はphp.iniで log_errors_max_lenをふやしましょう」「うーん」

所で…

» php.iniって一つしかないイメージなんだけど、なんで複数あるの？

```
/Users/uzulla/.phpenv/versions/7.0.1/etc/php.ini,
/Users/uzulla/.phpenv/versions/7.0.1/etc/conf.d/xdebug.ini,
/Users/uzulla/.phpenv/versions/7.0.1/etc/conf.d/my_special.ini
```

- **Config file scan directory**とよばれ、最近よくつかわれています
  - `php --ini` 等でしらべられます(Scan for additional .ini files)
- 有るディレクトリ以下の `*.ini` が全部読まれます、`/etc/php.ini` と同等
- 「インフラストラクチャーアズコードにぴったり👍」
- （php.iniの設定を書き換えるのに、sedとかもうしたくないからね...）
- なお、同一の設定記述があった場合、上書きされます

まとめ

- php.iniの場所が皆さんわかりましたね
- 繰り返しになるのですが、かならず

  ✅「調べたいファイルの、調べたい行で調べる」

  事を忘れないでくださいね。

php.iniを書く

- php.iniはテキストの設定ファイルです
- 見たら解る程度のシンプルなものです
- どういう項目があるかは、php.netをみよう（雑）

フォーマットについて

```
; comment here
[hoge]
key = value
key2 = "this is value2"
key3 = On
key4 = This is value4 ; クオートは実は不要
```

- シンプルな、キーと値ペア
- ;の後はコメントです
- [〜]これは無視されます、**何を書いても意味ない**です。
  - そう「⚠️sectionに意味は無い」んです（以外と知られてない）
  - 目印でしかない
- (hoge[] = fugeという書式があるが、見かけない)

型は(実質)二種類だけ

- String
- Boolean
  - On/Off, true/false, yes/no,none
  - ⚠️booleanは丁寧にあつかいましょう

- (integer)
  - （ってあるけど、実質Stringでは…）

同じキーがあれば、上書きです

```
mbstring.strict_detection = On
mbstring.strict_detection = Off

// Offになります
```

>> 重複はエラーになりません

- ただし、順序が意味を持つ場合もあります
  - 例：mbstring.languageとmbstring.internal_encoding
  - 例：default_charsetとinternal_encoding
  - （省略されたとき、ある値を暗黙でデフォルトとして使う、等）

突然php.iniクイズ！

以下の行、どれが「エラー」！？

```
k1 = 1
k2 = True
k3 = On
k4 = "On"
k5 = text contain
new line.
k6 = text
```

正解は、「エラー」にはなりません！！

- 「エラーでおこられないけど」k5の値は"text contain"となる
- A=B 形式以外は読み飛ばされる
- ⚠️つまり、ミスってもきづかないぞ（注意！）
- ⚠️Booleanに誤ったテキストをあてても無視されます
  - よくある例: hoge = enable とか…

第二問

以下の行、どれが「エラー」！？

```
日本人 = 日本語

# hoge = 1

1

echo

💢

⭕=❌

[]

]
```

正解は、「エラーになりません」！！！！

» ⚠️繰り返しになりますが、ミスっても気づけないぞ

エラーになるのは以下くらいです

- 行の先頭に =
- 同一行に] で閉じられていない [

ところで、独自のキーは入れられるのか？

» 自分のアプリの変数とか、トークンとか

» 不可能です、無視されます。

» そういう便利環境変数ではない

環境変数を読み込めるよ

>> mysqli.default_user = ${MYSQL_DEFAULT_USER}

>> つかったことはほぼ無いな...🤔

>> （PHP世界では、あまり環境変数はつかわれない(主観)）

エラーにならないの怖い？lintしたい？

» iniをパースする関数はあります

» parse_ini_file, parse_ini_string

» http://php.net/manual/ja/function.parse-ini-file.php

```
php > var_dump(parse_ini_file('dummy.ini'));
array(9) {
  ["k1"]=>
  string(1) "1"
略
```

» ただし、「php.ini」として正しいかではない

» ⚠️php.iniとして正しいかlintするツールは無い

» （つらい）

» 方法は後述しますが、設定したら必ず確認しましょう

反映するには...

- php 実行環境を再起動
  - apache
  - php-fpm
  - 等

反映されないんだけど？😡

- ちゃんと再起動した？CLIとmod_phpは別ですよ
- nginxを再起動してない？(fpmを再起動しましょう)
- php.iniの場所、間違えてない？
- 独立した`phpinfo()`では反映されてない？
- 反映が後々上書きされていない？（後述）
- 一応ログも見よう

まとめ

- php.iniはテキストの設定ファイル
- 文法ミスや記入ミスはスルーされるぞ
- 設定したら、すぐ確認
- 反映には再起動

php.iniを調べる

» 🤔「さっき話したのでは？」

» 😐「php.iniファイル(設定)と、php.ini(現実)は違う」

どこでも設定できて便利なphpの設定場所事例

- php.ini や scan dirの.ini
- httpd.conf や nginx.conf
- .htaccess や .user.ini
- ユーザープログラム内
- 「ある関数(等)を実行すると、暗黙に変わる」
- 環境変数、Windowsのレジストリ...etcetc

» つまり、php.ini（ファイル）をいくら見ても無駄ということだ！

ワハハハ！👻

» ワハハではない💀

なお

- 「どこで設定されるか」を全部調べる事は困難
- そもそも、どこで設定されたのかが役立つ事も少ない
  - ⚠️変わっているということは、どこかで意図があってやっている
  - ⚠️上位でかえちゃうとロクなことにならない

おもいだしましょう

» ✅「調べたい環境の、調べたいファイルでしらべろ！」

» と、いうことで、皆さんはphp.ini（理想）を見た上で、php.ini（現実）も見る必要があるのだった…。

調べるための関数

>> phpinfo > 省略

>> ini_get > よくつかう

>> ini_get_all > 私はよくつかう

>> get_cfg_var > 目的が違う

始まりと終わりの地、phpinfo()

```
<?php
phpinfo();
```

» すべてがここにある

» 基本中の基本

変数にキャプチャはできないので...

```
ob_start();
phpinfo();
$info = ob_get_clean();
file_put_contents('/tmp/phpinfo.html', $info);
```

>> しんどい、ので、微妙。

ini_get()

>> ini_get (string $varname)

>> 現在の設定を一つ取得できる

>> ⚠️失敗時にはFalseが返る

>> ⚠️返値は「String」である

「失敗時にfalseを返します」

» ミエミエの罠

» ⚠️PHPの自動変換で、falseと空文字をミスる

» よくあるミスが以下

```
if( ini_get('display_error') ){
    die('display_errorは消しましょう');
}
```

```
if( ini_get('display_errors') ){
    die('display_errorsは消しましょう');
}
```

>> （このコードがどうなのか、というのは他所に置くとして）

>> ただしくはdisplay_errors（個人的によくあるTypo）

>> キーを間違えると、かならずfalseになる

>> 「まあ、===やればいいですよね！」（訓練された人間の発想）

「ini_getはStringで返してきます」

>> そうですか

>> みてみましょう

```
// php.ini（ファイル）では Off だと…
short_open_tag = Off

// このように空文字が返ってくる
php > var_dump(ini_get('short_open_tag'));
string(0) ""
```

>> php.ini 記述のままではない（例外もある）

>> 変わるのは良いが、初見殺しです

>> なぜ On/Off、true/false、1/0 等でないのか…

>> 「自動変換のPHPです、慣れましょう🤗」

```
php > var_dump( ini_get('upload_max_filesize'));
string(4) "2M"
```

» upload_max_filesize は記述がそのままでてきます

» upload_max_filesize の指定には、K,M,Gなどの短縮記法が使えて人が読みやすくできる。

  » http://php.net/manual/ja/faq.using.php#faq.using.shorthandbytes

» しかしこれによって引き起こされる悲劇！()

```
php > echo ini_get('upload_max_filesize');
2M // 2Mbyte = 2*1024*1024

if( 1*1024*1024 > ini_get('upload_max_filesize') )
{ die("plz more!!!"); } //-> plz more!! // あれれれ？

php > echo (int)"2M"; // "2M"を数値へキャスト
2
```

» ⚠️皆さんご存じ、"2M" が評価で 2 になる安定の自動変換

» ちなみに、短縮記法をバイトに変換する関数はない。なんでや…

» 各自書くか、秘伝のたれをググりましょう(腐ってないか確認も)

話を戻して

>> ini_get()の話でしたね

ini_get_all()

>> ini_get_all([str $extension [,bool $details]])

>> 現在の設定を、全部を取得できる

>> 引数は(null, false)がおすすめ

>> 返値はキーと値の配列

```
php > var_dump(ini_get_all());
array(233) {
  ["allow_url_fopen"]=> // キー名
  array(3) {
    ["global_value"]=> // グローバルの値（？）
    string(1) "1"
    ["local_value"]=> // 現在の値
    string(1) "1"
    ["access"]=> // アクセスレベル
    int(4)
  }
```

null, false 指定例

```
php > var_dump( ini_get_all(null,false));
array(233) {
  ["allow_url_fopen"]=>
  string(1) "1"
  ["allow_url_include"]=>
  string(0) ""
```

勿論こうすれば単体の値もとれます

```
var_dump( ini_get_all(null, false)['display_errors'] );
string(0) ""
```

» ⚠️ところが、ini_getとは挙動が異なる

```
php > var_dump(ini_get('upload_tmp_dir'));
string(0) ""

php > var_dump(ini_get_all(null, false)['upload_tmp_dir']);
NULL  // 突然のNULL！！
```

>> 「名前が似てるだけで、二つ関数の返値が同じと誰が言った？」

>> 本来 upload_tmp_dir は、(php.net曰く)デフォルトがNULL

>> ini_getのほうが正しく(?)ない

>> 型を意識したくなりますね！

つまり、値は取れるが要注意

- 今回はvar_dumpをつかってますが、返値がないので書き出しづらい😫
  - phpinfoと同様、obつかえば取れる
  - print_rでは返値に出来るのに...

みんな大好きprint_rは...

```
php > echo print_r("",1); // なにもでない
php > echo print_r(null,1); // なにもでない
php > echo print_r(true,1); // trueは1になる…
1
```

» このようにprint_rは型をださない

» し型ないので、別の道具が必要

余談：ことある毎にこの不満を言う私

- var_dumpは、何故変数にキャプチャできないのだろう...
  - みんな思うでしょ...思わない？
- 某識者「obつかおうよ」私「なんでや！obさわりたくないし、めんどいやろ！」
- 別の識者「深遠なる理由があるのだろう（作ってる人に聞け）」私「たしかに...」

meanwhile in php conference 2015...

- PHP作者のラスムスがきていた
  - 「歯ブラシ」で有名な人
- ヘタな英語でこの件を直訴質問
- 作者「それはそういうものだ、obをつかえ😐」私「はい...😞」
- PHPに救いはなかった
- 私に食い下がる英語力もなかった(fin...)

余談終わり

>> var_dumpの希望は潰えましたので、他の手法の紹介

serialize()

```
php > echo serialize(0);          // i:0;
php > echo serialize("false"); // s:5:"false";
php > echo serialize(false);    // b:0;
php > echo serialize(null);     // N;
```

» 一文字目で型がわかって便利！👍

» i:int, s:string, b:bool, N:null

» 皆、PHPのシリアライズ形式を読むようになります（よね？）

json_encode

```
php > echo json_encode("");      // ""
php > echo json_encode(null);    // null
php > echo json_encode(true);    // true
php > echo json_encode("true"); // "true"
php > echo json_encode(1);       // 1
php > echo json_encode("1");     // "1"
```

» 型も、ちゃんとみればわかる👍

個々でなく、全部の設定を見たい時は

```
php > echo json_encode(ini_get_all(null,false), JSON_PRETTY_PRINT);
{
    "allow_url_fopen": "1",
    "allow_url_include": "",
    "arg_separator.input": "&",
```

>> 結果をjson_encodeで、JSON_PRETTY_PRINT

>> 比較にも便利です（重要）👍

jsonとdiffで雑に比較する例👍

```
$ diff 56.json 7.json
26c25
<     "date.timezone": "Asia\/Tokyo",
---
>     "date.timezone": "",
42c41
<     "error_reporting": "-1",
---
>     "error_reporting": "22527",
```

» 各自ツールを適当に

tipsおわり

- ✅普通は json_encode() が一番では？
  - 機械的処理にも向いている
- 他にvar_export()などもつかえます、ほぼjsonに近しい見やすさ
- serialize() を人力で読むのには、数時間は経験がいる

get_cfg_var

» 初期のphp.iniの設定を取得できる

» まあ、使わない

» ini_get_allの$detail=trueでまかなえる

中休み

- ini_get/ini_get_allで現在の設定を確認
- ✅型や、短縮記法の危険性をきちんと意識、把握しよう
- ⚠️特にini_getの返値がStringなのは注意せよ
- 👍ini_get_allをjson_encodeなどで整形すると一覧性高いし、jsonだから比較に便利

php.iniに 設定する

» 確認したら次は設定ですよね

» 「もうやったのでは？」「その件ではない」

» php.ini(リアル)の設定は「実行時に変更できる」

```
<a href="<?php
 ini_set('display_errors', 0);
 some_broken_func();
 ini_set('display_errors', 1);
 ?>">死</a>
```

設定につかう関数

- ini_set()
  - ほんとうによくつかう
  - ini_alter()というAliasがあるが、見かけた事はない
- ini_restore()
  - 使ったことない

ini_set

>> `ini_set ( string $varname , string $newvalue )`

>> キーと値をセットします

>> 値はStringです（注意）

>> ⚠返値は「変更前の値」です（注意）

>> ⚠失敗時はFALSEが返ります（注意）

>> ⚠設定できないものもあります(後述)

値はStringです

- 悪い予感しかしない😰
- 実際悪い事に、罠がある

```
php > ini_set('mbstring.strict_detection', 'On');
php > var_dump(ini_get('mbstring.strict_detection'));
string(2) "On"
php > var_dump(mb_get_info()['strict_detection']);
string(3) "Off" <-- !!??
```

» mbstring.strict_detectionで確認

» bool型なので、php.iniでは"On"を指定する

» しかしini_set は、"On"を正しく受け付けません！😰

色々な値をini_set経由でboolにいれると...

- Onになる
  - true, 1, -1
- Offになる
  - false, "true", "false", 0, "On", "Off"

うーんこの😇

資料をみてみましょう

» **php.ini**では true/false,on/off, yes/no, none と指定する

http://php.net/manual/ja/configuration.file.php

```
; 論理値は、次のいずれかで指定します
;    true, on, yes
; または false, off, no, none
```

» しかし、"true"をいれると、オフになる...のが...

» まちがえなければどうということはない🤖

っていうかね、

» mbstring.strict_detectionは

» Booleanとかいてあるのに

» (php.net曰く)デフォルトは"0"なんだよ

» うぐぐぐ…Booleanとは…

そもそも、ini_getすると

» （前述もしましたけど）

» Booleanで、Offのときにini_getすると""（空文字）がかえってくるので、Offなんてなかったんや…。（有ります）

他にも闇が

>> デフォルトNULLのmbstring.substitute_character

>> ini_setでNULLをいれても""になるよぉ...

>> （ただ、""≒NULLであり、困ったことはない...）

```
php > ini_set('mbstring.substitute_character', null);
php > var_dump(ini_get_all()['mbstring.substitute_character']);
string(0) ""
```

個人の感想です

- （bool相手は）1と0を使うとよい
- 0/1の指定はphp.iniでも使えます
  - php.netには書いて無いけど....。
- "On"、”Off”のことはわすれよう...
- （個人の感想です）

中休みまとめ

- `ini_set`でphp.iniの設定を変えられる
- ✅引数はstrだが、相手がboolの場合1/0が無難（個人の感想です）
- 正しい人は、php.iniと`ini_set`で使い分けてください🤖

アクセスレベル

- 先程の通り、php.iniの設定は実行時も変更できる
- ただし、ini_setでその場で変更できないものもある
- その制限を「アクセスレベル」という
- ４段階になっている

アクセスレベルは四種

- **PHP_INI_ALL** =>どこでも設定可能、多くがコレ
- **PHP_INI_USER** =>ほぼ存在しない
- **PHP_INI_PERDIR** =>.htaccess,.user.ini,php.ini,httpd.conf(等)
- **PHP_INI_SYSTEM** =>php.ini、httpd.conf(等)

>> PHP_INI_ALL

  >> どこでも

>> PHP_INI_PERDIR

  >> .htaccess、.user.iniが利用できれば可能

>> PHP_INI_SYSTEM

  >> サーバー管理者のみが変更可能

なぜ全てがPHP_INI_ALLではないのか

- ユーザプログラム実行前に必要な情報
- セキュリティ的な理由など
- （よほどの事がなければ、ALLです、開放的です）

» 所で、「アクセスレベル」って名称が正式か不明（？）

» レベルが記載された以下のURLなどには特に名称がない...

» http://php.net/manual/ja/configuration.changes.modes.php

» ここでつかわれているので、多分正式名称だとおもいます...

» http://php.net/manual/ja/function.ini-get-all.php

よくあるハマり、以下はPHP_INI_ALLではない

- **PHP_INI_SYSTEM**
  - sendmail_path
  - max_file_uploads
  - upload_tmp_dir

- **PHP_INI_PERDIR**
  - max_input_vars、max_input_time
  - auto_append_file、auto_prepend_file
  - post_max_size
  - session.use_trans_sid
  - short_open_tag
  - upload_max_filesize

まとめ

- php.iniには４種のアクセスレベルがある
- ただ、管理者が制限したそうな、memory_limitとか、max_execution_timeみたいなのもALLである
- セキュリティ等より、インタプリタなどの動作上の都合がメインの区分けっぽい
- 「php.iniって権威がなくない…？ザルでは？」
- もっとも、それを防ぐ手段もある(後述)

SAPI毎に独特な php.iniの設定方法

- すでに述べたように、SAPIというものがある
- それぞれで、少しだけ独自の設定方法がある
- php.iniの設定を制限する方法がある
- いくつかをご紹介

CLI

例：

```
/etc/php.ini
/etc/php/conf.d/*.ini
```

CLI

- /etc/php.ini等
- あるいは-c /path/to/php.ini php.iniをパスで指定
- あるいは/etc/php-cli.ini の設置
  - 本来のphp.iniのdirに、php-{SAPI名}.iniがあるとphp.iniに優先される

CLIは独特さがある

- CLIはいくつか挙動が違う（相手がTERMなので）
  - phpinfo出力がtxtモードになったり
  - 実行時間など各種リミットが外れたり
- CLIは手軽で色々確認できるが、デフォルト値が変わるので、テストに使う時は注意しましょう
- cliは`-d memory_limit=-1`などとCLIオプションで指定可能

apache+mod_php

例：

```
/etc/php.ini
/etc/php/conf.d/*.ini
/etc/apache/httpd.conf
/etc/apache/conf.d/some.conf
/var/www/html/.htaccess
/var/www/html/abc/.htaccess
```

apache+mod_php

- /etc/php.ini等
- あるいはPHPIniDir 指定でphp.iniの場所を指定
- httpd.confや.htaccess
  - いくつか追加のディレクティブが利用可能に

mod_phpの追加ディレクティブ

- `php_value key value` とStringの設定が可能
- `php_admin_value` 同上だが、ユーザが設定上書きできなくなる
- `php_flag key on` とBoolの設定が可能
- `php_admin_flag` 同上だが、ユーザが設定上書きできなくなる

mod_phpの追加ディレクティブ例

```
php_admin_value memory_limit 128M
php_admin_value max_execution_time 10
php_flag display_errors off
```

コピペしようとしてよくある罠

- (httpd.confや、特に.htaccessにおいて...)
- 「php_value(等)入れたらエラー」
  - そのサーバはmod_php入ってないのでは？
  - その環境はmod_phpではく、CGI/FastCGIでは？

- 「反映されないぞ？」
  - mod_phpは入っているが、前述の通りCGI/FastCGIでPHPをつかっている
  - アクセスレベルを無視している
  - 残念、定数は使えないのだ（例:E_NOTICE）
  - 再起動をわすれている

CGI

例：

```
/etc/php.ini
/etc/php/conf.d/*.ini
/var/web/html/.user.ini
```

CGI

- /etc/php.iniなど
- .user.ini
  - .htaccessのphp_value等の代用、記法はphp.iniと同じ
  - 同一DirからDocRootまでの間に設置する
  - 一度読むとデフォルトで５分キャッシュされます
  - publicに置く=漏洩に注意！（.htaccessみたいに403にしよう）

nginx+FastCGI

例：

/etc/php.ini

/etc/php/conf.d/*.ini

/etc/php/php-fpm.conf

/etc/php/php-fpm.d/*.conf

/etc/nginx/nginx.conf

/var/web/html/.user.ini

nginx+FastCGI

- 基本CGIと同様（php.ini、.user.ini）
- nginxのfast_cgi_paramで追加指定可能
  - fastcgi_param PHP_VALUE "memory_limit=-1;
    max_execute_time=-1";
  - 同様に、PHP_ADMIN_VALUEもある
- /etc/php-fpm.conf(プール設定ファイル、次ページ)

プール設定ファイル

```
php_flag[display_errors] = off
php_admin_value[error_log] = /var/log/fpm-php.www.log
php_admin_flag[log_errors] = on
php_admin_value[memory_limit] = 32M
```

>> また新たな記法がうまれた…😇

「あれ、Windowsの話は？」

>> はい

>> GUIなどで、設定したり、レジストリ(!)などが、あります。

>> レジストリはつかわないで、php.iniや.user.iniをつかいましょう

>> 最近のIISは普通FastCGIらしいので、FastCGIの資料をみてください

まとめ

- php.iniの後に、各SAPIによる設定ができる
- 各SAPIで設定ファイルや手法がちがう
- 設定できるキー名などは同じ
- ⚠️ただし、phpの定数は利用できない（注意）
- ✅`php_admin_*`で、ユーザーに制約をつくれる

バージョン間差異

兎に角変わる、マイナーで変わる

- PHPはマイナーバージョンアップ（x.y.zのy）でもすっごく変わる
  - （semverではないので、意味が違うが）
- zくらいなら、大体大丈夫...大体ね...
- つまりどこが変わっても信用しづらいってことだな、ガッハッハ
- （ガッハッハではない）

いにしえの実行環境

- PHP5.1、5.2、5.3あたりは7とは相当ちがったりする
- 「その時代を生きてきた俺たち」にはよいけど、そうでもない人はつらそう
- php.netの付録をよみましょう
- 実機で心行くまで試しましょう
  - テスト用に古いphpをビルドするのにつかれたら、古いLinuxディストリをDLするとよいです(真顔)

デフォルト変更を確認する

» php.netには「付録」という「これこそ本編」みたいな情報がある

» そこをちゃんとチェックすれば、大体大丈夫

» `php -n -a`でiniをロードせずに`ini_get_all`を動かして、差を見る

» 差があったら、適切に埋めるiniを書く

```
// リモートと手元を確認する有名テク。だが、前述したように安易にcliで確認するのはお勧めしない。
diff <(php -r 'phpinfo();') <(~/.phpenv/versions/5.6.9/bin/php -r 'phpinfo();')
diff <(php -r 'phpinfo();') <(ssh remote 'php -r "phpinfo();"')
diff <(ssh remote1 'php -r "phpinfo();"') <(ssh remote2 'php -r "phpinfo();"')
```

上がり続けるバージョン、大変

- 昨今はPHPでもガンガンバージョンを上げていくスタイル
- バージョン上げる前にテストしましょうね〜
- phpも、phpenvとかで複数バージョン管理しやすくなりましたから
  - （入れやすいとは言っていない）
- ディストリの標準、特にRHやCentOSはのんびりしてるので、ゆるふわ派はそれで...
  - （verは上がらないけど、ある程度パッチは降ってくるので...）

tips 沢山のphp.iniを抱えた俺たちはどうすれば

- 毎回php.iniをエディタでいじるのは大変なので...
- php.iniはさわらず、必要な設定をかいたiniをconf.dにコピーし、上書きすると楽
- 私は、手元のphpenv相手はツールをかいてどうにかしてます
- https://github.com/uzulla/setmyphpini.php

tips でも、mod_phpはどうすればいいのよ

- php-buildやphpbrewがある現代でも、mod_phpは一手間
- ちゃんとapacheで確認するしかない...
- 私はapacheをbuiltin serverみたいにサッと立てるツールを書いて、それでやってます
- https://github.com/uzulla/apachehere
- （libphpX.soつくるのは、依然としてダルい）

まあ、カッコイイ会社は

- CIとかをちゃんとくんでやってるんじゃろ？
  - 実際、サーバーを自由にできるなら、よさそう
- 突然サーバーのftpアカウントがメールされてくるような、野良のPHPerはそうもいかなくてつらい（愚痴）
- がんばろう…😇

ちょっと休憩

質問ございますか？

» ということで、php.iniとその愉快な仲間達のお話でした

» ググるとでてくるし、php.netにだいたいのってます（当然だが）

» ✅「その行でのphp.ini(リアル)しか信用するな、必ず確認せよ」

» ✅「SAPIで設定手法が変わる」

» 言いたい事はこの二行です

ここから先は

» php.iniで設定できる各項目についてのお話...

» 「つまりここまでは基礎知識だったんだよ！！！」Ω

» ΩΩΩ 「な、なんだってーｒｙ」

» 「PHPむずかし杉内？」「たし蟹」

お品書き

ファイルアップロード、mbstring、セッション、assert、db、curl、セキュリティ、メール、日付、エラーとログ周り、リソース制限

エラーとログ周り

まず言いたいのは

>> ググって出てくる「こうやったらエラーがきえました！」

>>> の8割くらいは「エラーがみえなくなった」だけである

>> 🙅ダメ絶対！！！！

```
log_errors = On ; そもそもエラーログを取るか
error_reporting = E_ALL & ~E_DEPRECATED & ~E_STRICT
display_errors = Off ; 画面にエラーを出すか
display_startup_errors = Off ; PHPの起動シーケンスにおいて発生したエラーを画面に出すか
log_errors_max_len = 1024 ; エラーログの最長（切り捨て
error_log = ; 出力先、省略時SAPIへ
html_errors = On ; SAPIへ出力時、エラー文字列をhtml化するか
```

- phpのエラーはかわりつつある
  - php<=5.6 例外とエラーが別
  - php>=7 例外とエラーが「似てる」
    - Error（内部的には例外なのだが、ユーザーは作れない）
    - Exception（例外＆ユーザー例外）
  - 例外もキャッチしきれなければエラーとなり、それはログに出る

» （php7以前は、エラーがThrowableの皮を被っていないだけで、
php.iniにおける扱い方としてはそこまでかわりません）

エラーには種類がある

- 16個ある（多すぎない？）
  - NOTICEもエラーです
  - E_NOTICE,E_ERROR,E_DEPRECATEDなど
  - http://php.net/manual/ja/errorfunc.constants.php

- error_reporting にスイッチ指定する
  - 表現は数値だが、定数のビット演算で指定できる
  - PHPerが唯一ビット演算をする場です（？）
  - 例 E_ALL & ~E_DEPRECATED & ~E_STRICT
    - = 22527 、覚えられるか！(なのでbit演算)
    - 全部のエラーから、非推奨エラーと、Strictエラーを抜いてる
    - 定数なので*.iniやini_set以外ではこの記法はつかえない
  - 「何も考えず、-1 にしておけばオッケー」（個人の感想です）

>> っていうか、-1 以外に設定する人は、わかっているのかないのか

>> 最初にかいたけど、見えなくしてるだけでは？

>> E_DEPRECATEDは、現場によっては「わかる〜仕方ないよね〜」

>> E_NOTICEをグローバルに落とす人は楽観的で、人生楽しそう()

>> わかってやってるならいいとおもいます

>> 「PHPは自制心がもとめられる言語」

- log_errors_max_len
  - ガンガンダンプする人はのばしましょう
- display_errors、display_startup_errors
  - 開発時ばかりはOnのほうがはかどる事は多いです
  - PHPは些細なエラーは、そのまますすんじゃうので、displayしないとエラーログ見ないかぎり無言。
  - 「エラーはエラー！Noticeでもとまってほしい！」
    - （止めるテクもあります）

- error_log
  - 未指定時はSAPIのエラーログへ、指定すると好きなファイルやSYSLOGにエラーを出せます。
  - Apacheのエラーログは膨大すぎて見づらいとか、深遠な理由でエラーログが翌日にならないと見れない、等の場合に便利()
  - 書き込めない場合、SAPIのログにフォールバックされます
  - うっかり `ini_set('error_log', 'php.log');`とかやって、公開ディレクトリに出力しないように気を付けてくださいね、フルパス指定になるように書こう！

>> ⚠️error_logは、php.iniで指定しないほうがいい

>> ⚠️Builtin serverやCLIで画面にエラーでてこなくなります(罠)

>> SAPI側のphp_value等で設定するのが良いでしょう

>> (あるいは、局所ini_setがよいかと)

```
php > echo ini_get("error_log");
/tmp/php_errors.log
php > echo $a; // 未定義変数を触っているので、エラーがでるはずだが出ない
php > ^D

$ tail /tmp/php_errors.log
[XXXX] PHP Notice:  Undefined variable: a in php shell code on line 1
```

>> ログローテーションとか、無い。

```
// init.dで、環境変数指定で、後は再起動で…！？
export D_OF_WEEK=`date +%w`
rm -f /path/to/php_errorlog.$D_OF_WEEK
// php.ini
error_log = php_errorlog.${D_OF_WEEK}
```

- （ジョークです）
- 素直にlogrotate設定して再起動したほうが枕が高いです
- パイプできないので、 rotatelogsとかは使えぬ

まとめ

>> error_reporting = -1 が最強👍

>> （前述の通り整数なので）確認しやすいし

>> 「しかし現実世界は辛く苦しい😇」

>> E_NOTICEが落としてあったら、解りやすい危険フラグ

>> <s>不可避なコードは、そこでだけerror_reportingを変えたり@をつけよう、やっぱりPHPは便利</s>

日付

date.timezone まわり

```
date.timezone = "Asia/Tokyo"
;date.default_latitude = 31.7667
;date.default_longitude = 35.2333
;date.sunrise_zenith = 90.583333
;date.sunset_zenith = 90.583333
```

date.timezone

- ✅兎に角設定しよう
- "Asia/Tokyo"
- 設定しないと、皆大好き`strtotime`等でWarnが出る
  - ...のはPHP5.6まで、7からUTCがデフォルトになった
  - ⚠️むしろ罠になったのでは？

date.timezoneの変遷...

- date.timezoneはWarnを無視すれば、UTC(ドキュメントではGMT)
- 過去、TZ 環境変数を読んでいたが、5.4から参照しなくなった
- 曰く「タイムゾーンの判定時に、OSから得られる情報に頼らないようになりました。 推測に基づくタイムゾーンは信頼できないからです」
- TZは確かにすごく重要だよね！でもそれなら必須のままでよかったのでは...

余談：謎の緯度経度

```
;date.default_latitude = 31.7667
;date.default_longitude = 35.2333
;date.sunrise_zenith = 90.583333
;date.sunset_zenith = 90.583333
```

» 「date_sunrise() と date_sunset() でのみ使用されます。」

» なんと適切に設定することで日の出と日の入りを計算できます！

» php.iniに持つ必要があるのだろうか…深遠なる理由がありそう！

mbstring 等

色々あるけど、大抵これでよい✅

```
default_charset = "UTF-8" ; Content-Type のデフォルト
internal_encoding = "UTF-8"
[mbstring]
mbstring.language = Japanese
mbstring.internal_encoding = "UTF-8"
mbstring.strict_detection = On
```

余談(？)「default_charsetとは一体...」

```
; Use of this INI entry is deprecated, use global internal_encoding instead.
; internal/script encoding.
; Some encoding cannot work as internal encoding. (e.g. SJIS, BIG5, ISO-2022-*)
; If empty, default_charset or internal_encoding or iconv.internal_encoding is used.
; The precedence is: default_charset < internal_encoding < iconv.internal_encoding
;mbstring.internal_encoding =
```

>> 「mbstring.internal_encodingとか時代遅れ、
時代はdefault_charset一箇所でおｋ!!!」

>> 私「へーそうなんだ」

>> 「（default_charsetは） mbstring.internal_encoding が未設定の場合のデフォルトとしても用いられます」

>> へー、すごい、楽でいいね！

>> が！残念！

>> SJIS-winは、入れても反映されない！（コメントにも書いてある）

>> eucJP-winもダメっぽい！

>> UTF-8なら、そもそもデフォルトやろがい

>> 「...ま、昔ながらに設定しますわ、信用一番」

そもそも...

» 「デフォルト値」ということは、

» 実行時にdefault_charsetを変更しても、
mbstring.internal_encodingなどにも反映されるわけではなさげ

» （php.iniを動的にいじる際に比較的よくあるタイプの罠です）

» このように、PHPにはしばしば変な期待をいだかせられ、裏切られることが良くあります。

» ということで余談終わり

追加で...

- `mbstring.substitute_character`
  - 文字列のエンコーディングを変換した際に、変換できなかった文字を「ゲタ(〓)」等の特定文字に置換することで、変換漏れ等をわかりやすくできる。
  - SJIS,EUC,JIS共存時代の遺物感ある
  - UTF-8の時代はもういらないのでは
  - （と、思いたいけどまだまだcp932を使う事はあるね...）

以下は危険なので利用しないこと🙅‍♀️

```
;mbstring.http_input =
;mbstring.http_output =
;mbstring.encoding_translation = Off
;mbstring.func_overload = 0
;mbstring.http_output_conv_mimetype=
```

» 間違っても「便利！」と思ってはいけない

まとめ

» ✅デフォルトはUTF-8になったけど、自覚の為に指定しましょう。

» 自動変換系、関数オーバーロードはダメ絶対

» コメントやphp.netと実機の食い違いは、素直に実機優先しましょう

» 他にも、コメントにある

`http://php.net/internal-encoding`
などのURLが404だったりするのです

» 「php.net最高！」とかいいますが、そんなもんなのです

リソース制限

リソース制限

>> PHPerが初めてググる事になるのがここらへんでは

>> `max_execution_time = 30`

>> `memory_limit = 128M`

>> （実際は、もっともっとへらしてもよい）

安全弁

- PHPはどんなにアホなコードをかいても、ここらへんを闇雲に広げなければ（比較的）安全なのです
- 「PHPはゆるふわコードを赦す🙏」
  - （条件式をいつもノリで書いて、無限ループするの見てから直すタイプの人とか）
  - （2GBくらいあるログファイルの先頭1行をだすために、全部を`file_get_contents`したりする人とか）

勘違いされがちな事

- max_execition_timeはCPU時間以外、特に**I/O時間はノーカウント**なので注意
- 「（時計を見て）あと10秒で強制終了」**ではない。**
- ⚠️たとえば**DBマターな高負荷では全然終了しない**
- 普通のウェブアプリで何秒もCPU時間食うなんてありえない
- （勘違いして、Apache+mod_phpでガンガンプロセス数を上げる、偽スケールアップ病に陥る=>悪化が加速）

» （それよりも前に、ブラウザや NAT の TCPセッションが切れたり）

» （似たような「カウントの仕方を誤解」はセッション周りなどにも多い）

「PHPはゆるふわコードを赦す」が...

» 「何も考えずに便利だとかいいつつ、

`memory_limit = 1024M` と `MaxClients 512`

とかコピペで設定するのはゆるさん👊」

» 「PHPは自制心がもとめられる言語」

- memory_limitと、fpmならプロセス数もほどほどにしよう
  - （fpmのpm.max_children）
  - 「ほどほどとは？」「多くてもコア数の10倍こえてたら、一度検証してもよいのでは」(個人の感想です)
  - 「mod_phpは？」「前段にnginx置くとよいのでは()」
- 自信がない？isuconってやつの過去問でためしてみるといいよ！💪
  仮にわずかにスループット上がっても、メモリ枯渇でのswapやOOM Killerの恐怖とは釣り合わないよ

スタックまわり

- pcre.backtrack_limit=100000
- pcre.recursion_limit=100000
- 複雑な正規表現、あるいはデカイデータを処理するとこれにかかることがあります
- 何か異常なことをやっていない？
  - しかたなく大きくしましょう

ファイルアップロード

```
file_uploads = On
upload_tmp_dir =
upload_max_filesize = 2M
max_file_uploads = 20
post_max_size = 8M
```

» PHPはファイルアップロードをどの環境でもちゃんと扱える、
すげえ！！

» （当たり前に聞こえるけど、結構面倒なんやで…（老害））

- upload_max_filesizeはほどほどに
  - ファイルサイズがデカイと攻撃にもなりうる
  - 最近は回線速いし、Diskの小さいインスタンスだとヤラれる
- post_max_sizeの変更を忘れがち
  - post_max_size > upload_max_filesize
  - （複数アップロード数も考慮しましょう）

- 2GB以上のアップロードを扱う場合は…
  - 「異常？大丈夫？」「以上です、大丈夫です」
  - 64bit環境にしましょう
  - php>=5.6にしましょう
  - nginx/apacheの設定（制限）も確認しましょう
    （非常によくあるハマリ所）
  - 経路にあるリバースプロキシも

- 「なんかmemory_limitに当たった！ふやそう！😋」
  - 巨大ファイルを変数にロードしてはダメ（当たり前）
  - file_get_contentsを使うな
  - fread等をつかってください
  - Generatorをつかってください
  - Stream Wrapperをつくってください

- upload_tmp_dir
  - 省略時、systemのtmpをつかいます
  - 環境やアプリ内容によっては、安全な所に変えよう

まとめ

» 小さいファイルサイズなら簡単！PHP最高！

» 大きいファイルサイズだと罠が沢山！それでもPHP最高！

セキュリティ関連

- PHP自体はデフォルトでもそれなりに大丈夫
- 12億のサイト()が全部やられたりはしてない
- 「わりと」ね…HTTPOXYとかあったけど
- 有名なOSSがヤバい💀
- 「EOLなバージョンのPHPはあげてこ！」（ポジショントークです）

>> max_input_time

>> 「スクリプトが POST、GET などの入力を パースする最大の時間を、秒単位で指定します。 これは、サーバー側で PHP が起動してからスクリプトの実行を開始するまでの時間です。」

>> よくわからんね？正直俺もよくわからん

>> 大抵これはもっとへらせます

>> 「これセキュリティなの？」「うーん…いれる所が…」

>> `max_input_nesting_level = 64`

>> PHPは`hoge[hage]`みたいなパラメタが自動的に配列として扱える、
その最大深さである。

>> デフォルトの64は、つまり64次元変数まで許可

>> そんなに恐ろしいプログラムがあるのだろうか…😰
と納得させるパワーがPHPにはある

>> 精々5とかでは？

- `max_input_vars = 1000`
- POSTやGETのクエリパラメタの最大数
- 1000を多いとみるか少ないとみるか、人によって様々でしょう、どうですか？
- 1000でたりないことはある（真顔）
- 主にDoS対策がベース（ハッシュコリジョン攻撃とか）
- 超えると？WARNがでて、切り捨てられて、実行は進行する（注意）

このあたり、絞っていくのは正しいのだが

- 攻撃の第一波目くらいまではやらなくてもいいんでは
- ✅PHPでのKISSとは、できることならデフォルト設定でつかうことである（要出典）
- しかし、いつか対応するために知っておきましょう

>> open_basedir = /var/www/html:/tmp

>> phpコード上からは指定されたpathの外を読み書きできないように

>> 「安全そうだ！これはPHP界のSELinuxか！😍」

>> （つまり、すぐにオフにされます）

>> （しかも、PHP_INI_ALLです）

>> 複数指定時はPathを:でつなぐが、Winの場合は;でつなぐ

>> 一文字の代わりに、長いPATH_SEPARATOR定数をつかえば解決

>> allow_url_fopen = On

>> $html = file_get_contents('https://example.com/');

>> を許可するか

>> 許可しましょう😊

>> （これができなかったらなぜPHPをつかっているのか疑問を抱いてしまう...）

>> allow_url_include = Off

>> require ('http://example.com/super_lib.php');

>> を許可するか。

>> 「ヤバない？😨」「ネットからコードDLとか引くわ😫」

>> curl https://hoge/installer.sh | bash
ってやったことがない人はそう言っても良い

>> でもまぁ、無いわー

>> `sql.safe_mode = Off`

>> 「名前からしてつよそう！💪」

>> しかし「PHPでいうところのsafe_mode」という意味であり、別にsafeではない👻

>> 「オンにすると、デフォルト値が指定されているデータベース接続関数は、 引数で指定された値よりもデフォルト値を優先して使用します。 」

>> （phpにおける「safe mode」とは、ユーザーのポカ（や悪意）を多少邪魔するという意味です）

>> expose_php = On

>> X-Powered-By: PHP/7.0.9 などを隠す

>> まあオフにしたい人はオフにしたほうがよろしい

>> （ヘッダーにPHPってかいてあるとうれしいから否定しづらい）

>> 副作用として、「イースターエッグ」がつかえなくなります


localhost:8080/some.php?=PHPE9568F34-D428-11d2-A769-00AA001ACF42
php

» まあ、PHP5.5からはもともと無い

» （しかし、apacheではうごくけど、builtinserverではうごかんのよねこれ。）

```
disable_functions =
disable_classes =
```

» 関数やクラスを禁止できる

» 本日散々つかったini_setなどを殺せる

» 「なお、echoは関数ではないので禁止できない、これ豆な」

メール

最初に書いておくが...

- メール関連の設定をいじる場合、`mail()`とか、`mb_send_mail()`とかの挙動を変えたいのだろうとおもう
- ⚠️やめておこう⚠️
- ✅良いメール送信ライブラリをつかおう！
- するとだね、設定のほとんどが要らなくなるんだな...
- 「そもそも、若者はMailgunとかSESとかつかうんじゃろ？」

>> `sendmail_path = "/usr/sbin/sendmail -t -i"`

>> せいぜい、sendmailのPathが変わってる場合に変更

>> 通常、このpathにsendmailがあるので変更する必要はない

tips: sendmail_pathに自作のプログラムを指定すると便利

```
#!/usr/bin/perl
my $out_file_name = '/tmp/mailout';

open(my $fh, ">>", $out_file_name) or die $!;
while(<STDIN>){
    print $fh $_;
}
```

>> 「Perlやんけ！」

>> 「ばれたか」

» (他にも項目はあるけど、本当にいじって良いことないので省略)

セッション

- PHP最高便利機能の一つ、セッション
- 「PHPはフレームワークだ！」
- 言語（？）にセッション機能があるので、本当に楽
- これだけで60分話せる(?)

PHP以外の言語の方々へ、PHPのセッションは...

» 適切に発番されたセッションIDだけがCookieに保存、送信され、

» アクセス時には、自動的にSIDにヒモ付いたキーを元に、

» (自作もできる)セッションストレージハンドラからデータをひきだし、$_SESSIONへデシリアライズ。

» 終了時には、自動的に逆方向で$_SESSIONをシリアライズして保存。

» セッションIDの再割り当ても軽々！

» ...と、いったものが言語（環境）で用意されております。

- PHPのセッション機能は大きく二つの機能にわけられます
  - セッションIDの管理
  - ストレージ

- セッションID(SID)の管理
  - 何でやり取りするか
    - URL
    - Cookie
  - IDの生成、管理
    - シード
    - 長さ（強度）
  - セキュリティ強化の設定

» もはやテンプレ、SessionでCookieを使う各種設定

```
session.use_cookies = 1 ; SIDをCookieから読めるように
session.use_only_cookies = 1 ; いまどきURLに埋め込まないので、On
session.name = PHPSESSID ; Cookieキー名
session.cookie_secure = 1 ; httpsでのみセッションのCookieをやりとり
session.cookie_domain = ; 未指定で現在のドメインになるので、通常不要
session.cookie_path = / ; セッションクッキーのPath
session.cookie_httponly = 1 ; SIDをJSから見えなくする
session.use_strict_mode = 1 ; SIDを注入させない
```

`session.cookie_lifetime = 0 ;` セッションクッキーのExpireを指定

- 0はブラウザを閉じるまで有効
  - （現代で「ブラウザを閉じる」とは...？）
- sessionを維持するには、適切に設定する
  - 「今からN秒」
  - 別途、ストレージのlifetimeの設定もいる（後述）
  - 「長さかくあるべし」の議論はここではしない

» ガラケー時代は終わった、URLにSID関連はOnにしない

» セキュリティ的に、ロクなことになりません

session.use_trans_sid = 0 ; URLのセッションIDを受け入れるか？

session.referer_check = ; その場合、受け入れるドメインを固定

セッションIDの生成手段関係

» セキュリティにコダワリがあるなら変えてもよいのでは

» 変更すると、現在の全セッションが消えます（当たり前だが...）

session.hash_function = 1 ; セッションのランダム文字の長さ ; 0:md5か1:sha1か指定できる、けど7.1で消えました

session.entropy_length = 32 ; PHP 7.1で消えました

session.entropy_file = /dev/urandom ; PHP 7.1で消えました

なお、php7.1はこのようにシンプルになった

» php.iniのデフォルトだと後方互換性のために26文字になっている
が、もっとのばしたほうがいいぞということらしい。

```
; Shorter length than default is supported only for compatibility reason.
; Users should use 32 or more chars.
; Default Value: 32
; Development Value: 26
; Production Value: 26

session.sid_length = 26
```

session.auto_start = 0 ; 自動的にセッションを開始するかどうか

>> 大抵のコードでは自前でsession_start()していますので、
session.auto_startはオフでよいです

>> 不要な時はうごかないので負荷もさがります

```
session.cache_limiter = nocache ; キャッシュさせない
session.cache_expire = 180
```

» セッションがついたレスポンスをどうキャッシュさせるか

» なにかの都合や負荷の事を他所にすれば、デフォルトで大丈夫

- セッションストレージ（ハンドラ）
  - 通常はfile
  - memcachedとかの拡張をいれると、増える
  - KVSみたいなクラスを実装すると、自作できる
  - 複数台のサーバにするなどで変更する

session.save_handler = files ; デフォルトのfileストレージを使う

session.save_path = "/tmp" ; fileストレージの設定で、どこに情報を保存するか

- session.save_pathにはセッション情報が保存された大量のファイルができる
  - 邪魔だとか、Cronで掃除されないようにとか、nfsで共有したいとか(古)、必要があれば変更する
    - （パーミッションに注意！）
  - 「PHP以外で、セッションファイルを自前で読み書きする」などといった、強まったアプリを書くならこのpathを確認する

>> $_SESSIONをシリアライズするハンドラを指定

>> 「変更するの？強いね〜」

```
session.serialize_handler = php ; $_SESSIONを何でシリアライズするか
; 他にphp_serializeなどがある
;session.lazy_write = On ; 7から、更新がある場合のみ書き込む事で性能向上
```

>> 解ってる人だけがいじる項目です

>> セッションの有効期間関連

>> gc_maxlifetime、デフォルトの24分は短いような、長いような...

```
session.gc_maxlifetime = 1440 ; セッション有効期間、秒
; アクセス毎にgc_probability/gc_divisorのサイコロを振り、
; 確率的にsessionのGC要求が行われる
session.gc_probability = 1
session.gc_divisor = 1000
```

セッション有効期間の決め方

- session.cookie_lifetimeで、SID(Cookie)の寿命を適切に長くしましょう
- session.gc_maxlifetimeで、ストアされた情報が破棄されるまでの期間を延ばしましょう
- どっちが切れてもセッション（に保存された情報は）消えます
- 「何故別なの？」「SID管理と、ストレージは別の概念なんで」

「（セッション）キレてな〜い」

» ストレージ側のGCは、サイコロ任せ、丁度に消えることはない

» 「何分でセッションが切れるのか保証してください」「面倒な...」

» その場合、ExpireをPHPに任せるなら諦めて、$_SESSIONの中に時刻付きの情報をいれましょう

; なお、このようなイカサマサイコロは愚考です

```
session.gc_probability = 1
session.gc_divisor = 1
```

» 「じゃあgc_divisorを増やすと軽いのか！？」って思うかもしれませんが、いつかはGCが走って、それはそれで重い処理になります。

» （ISUCONならいいかもねw）

>> 幻の機能、upload_progress

```
;session.upload_progress.enabled = On
;session.upload_progress.cleanup = On
;session.upload_progress.prefix = "upload_progress_"
;session.upload_progress.name = "PHP_SESSION_UPLOAD_PROGRESS"
;session.upload_progress.freq =  "1%"
;session.upload_progress.min_freq = "1"
```

>> なんと「ファイルアップロードのプログレスバー」を出せる

>> アップロード中に別のリクエストを飛ばして、セッションを見ると、数値が取れる

>> 真面目につかったことはない

`session.hash_bits_per_character`

» お馴染みではないでしょう

session.hash_function	"0"	PHP_INI_ALL	PHP 5.0.0 から利用可能。PHP 7.1.0 で削除されました。
session.hash_bits_per_character	"4"	PHP_INI_ALL	PHP 5.0.0 から利用可能。PHP 7.1.0 で削除されました。
session.upload_progress.enabled	"1"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.upload_progress.cleanup	"1"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.upload_progress.prefix	"upload_progress_"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.upload_progress.name	"PHP_SESSION_UPLOAD_PROGRESS"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.upload_progress.freq	"1%"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.upload_progress.min_freq	"1"	PHP_INI_PERDIR	PHP 5.4.0 から利用可能
session.lazy_write	"1"	PHP_INI_ALL	PHP 7.0.0 から利用可能
url_rewriter.tags	"a=href,area=href,frame=src,form="	PHP_INI_ALL	PHP 4.0.4 から利用可能。PHP 7.1.0 以降は、セッションではこの項目を使いません。
session.hash_function	"0"	PHP_INI_ALL	PHP 5.0.0 から利用可能。PHP 7.1.0 で削除されました。
session.hash_bits_per_character	"4"	PHP_INI_ALL	PHP 5.0.0 から利用可能。PHP 7.1.0 で削除されました。
session.entropy_file	""	PHP_INI_ALL	PHP 7.1.0 で削除されました。

» 「なんと！表に二回登場します！」

» きっと重要だからなんでしょう！

» （つかったことない）

» 「また、PHP 7.1.0 で削除されました。」

» 「え？php.iniファイルに残ってるんだけど…」（ini_getはできませんでした）

» （兎に角まあ、そんなあつかいだということですね）

まとめ

- セッション周りは7.1でちょこちょこ仕様が変わりました
- デフォルトでつかってる人には関係ないけど、色々拡張したりしてる強い人は注意が必要
- 「昨日すぐにビルドしたけど、7.1とかいつ仕事で使えるんだろう…」「さあ…」

Assert

- 実は昔からある
- 7からプリプロセスで殺せるようになった
- 式評価ができるようになった
- 開発環境だけで使われるものです
  - なにかこれでロジックを書いてはいけない
- 「ここから普及するのかな…？」

```
; -1 コンパイル時無視、0 実行しない、1 実行する
zend.assertions = -1
;assert.exception = On ; fail時例外(？)スロー

; これ以下は下位互換性のため残っているが、つかうなと
;assert.active = On
;assert.warning = On
;assert.bail = Off
;assert.callback = 0
;assert.quiet_eval = 0
```

- スローというが、\Errorの子なので、\Exceptionではキャッチされない
  - PHPで投げられる例外は、Throwableという親から、ErrorとExceptionにわかれており、過去のコードはExceptionを親としていたので「普通」はにぎりつぶされないぞ
  - 7から例外とエラーがどちらも（基本的には）「Throw」になった。
  - キャッチはできる
  - ただし、Errorはユーザークラスで通常継承できない

curl

curl.cainfo=/path/to/cacert.pem

» SSL certificate problem: unable to get local issuer certificateみたいなエラーが出たら対応

» 証明書は検証してこそ、CURLOPT_SSL_VERIFYPEERをfalseにするとかダメ絶対

» cacert.pemはhttps://curl.haxx.se/ca/cacert.pemなどから入手

DB

はっきりいって、php.iniでDBとかの設定なんてせえへんので省略

まとめにはいっていきます！

沢山あるぞ！今日話せなかったこと

- cgi.force_redirectとかの話
- urlrewriterの話
- filterの話
- realpth_cacheとかの話
- zend.enable_gcの神話
- report_memleaksの期待と絶望

本トーク全体のまとめ

ああ！php.ini面倒くさい！

- 「php.iniなんてなくしてしまえばいい！🔫」
  - 前述もしたけど、なくてもうごきます
- ただし、phpのバージョンアップでデフォルトは変わる
  - 「マイナーで大変更が入る事で定評のあるPHP」
  - エンコーディングのデフォルトがUTF-8に
  - datetime.zoneデフォルトがUTCに
- 「把握し続けるくらいなら、php.iniを書いた方がマシでは」

» 「たれでいいじゃん🤗」

» まあ、現実としてそれでもいいんだけど、ハマった時つらい

» 特に、よくわからん他人のPHP環境はよくわからん

» そういう所は「なんか色々変えたら偶然動いた！」→「秘伝のタレ化」しており…

» 「これもういらないでしょ」「消さないでください！」

» 「この.htaccessが置けなければ死」「こちとらnginx」

» 等の問答が発生

- 「php.iniに設定があると管理が大変...😞」と思う人も多い
- 本末転倒っぽいけど、「全部コードに埋め込み」
  - 実は、これはこれで解決策でもある
  - （PHP_INI_SYSTEMなどは残るが...）
  - 私も、ini_setを多用する
  - しかし、自分以外が触るかもしれないんだぞ
  - CLIでバッチ回すときにハマるぞ

「やはりphpは悪い言語！抹殺する！😡🔪」

- php.iniが（も？）ややこしくても、PHPを嫌いにならないでください頼む
- 普段から使っていれば、慣れていきますので…
- 本日の資料は普段使いの範囲を結構カバーしているので…がんばって…
- 聞いてくれてもいいのよ？

php.iniマスターになるには

- php.netを熟読する（が、座学で信用するな）
- php.iniを熟読する（が、座学で信用するな）
- `ini_get_all`の値を全部見ていく
- 実地でハマって覚える
- 20年は持つphp.iniだし、あと10年は使われそう(？)

やっていきましょう 

完

質問ありますか？