PHP7 新特性

1. 关于 PHP7 新特性手册

这个手册由禅道项目管理软件团队（www.zentao.net）翻译自 PHP 源代码里面的升级声明。
英文的原文请大家参考：https://github.com/php/php-src/blob/php-7.0.0alpha1/UPGRADING
翻译如果有疏漏的地方，请大家以英文为准。

感谢鸟哥对 PHP7项目的推进和付出！
欢迎访问 PHP7中文网站（www.php7.site）了解更多 PHP7的内容。

2. PHP7 非兼容性改动

2.1 语言修改

2.1.1 变量处理机制修改

PHP7版本对变量解析机制做了调整，调整如下：

一、间接变量、属性和方法引用都按照从左到右的顺序进行解释：

```
$$foo['bar']['baz']      // interpreted as ($$foo)['bar']['baz']
$foo->$bar['baz']        // interpreted as ($foo->$bar)['baz']
$foo->$bar['baz']()      // interpreted as ($foo->$bar)['baz']()
Foo::$bar['baz']()       // interpreted as (Foo::$bar)['baz']()
```

如果想改变解释的顺序，可以使用大括号：

```
${$foo['bar']['baz']}
$foo->{$bar['baz']}
$foo->{$bar['baz']}()
Foo::{$bar['baz']}()
```

二、global 关键字现在只能引用简单变量

```
global $$foo->bar;      // 这种写法不支持。
global ${$foo->bar};    // 需用大括号来达到效果。
```

三、用括号把变量或者函数括起来没有用了

```
function getArray() { return [1, 2, 3]; }
$last = array_pop(getArray());      // Strict Standards: Only variables should be passed by reference
$last = array_pop((getArray()));    // Strict Standards: Only variables should be passed by reference
```

注意第二句的调用，是用圆括号包了起来，但还是报这个严格错误。之前版本的 PHP 是不会报这个错误的。

四、引用赋值时自动创建的数组元素或者对象属性顺序和以前不同了。

```
$array = [];
$array["a"] =& $array["b"];
$array["b"] = 1;
var_dump($array);

PHP7产生的数组：["a" => 1, "b" => 1]
PHP5产生的数组：["b" => 1, "a" => 1]
```

参考资料：

https://wiki.php.net/rfc/uniform_variable_syntax
https://wiki.php.net/rfc/abstract_syntax_tree

2.1.2 list()修改

一、list()不再按照相反的顺序赋值

```
list($array[], $array[], $array[]) = [1, 2, 3];
var_dump($array);
上面的代码会返回一个数组：$array == [1, 2, 3] 而不是之前的 [3, 2, 1]

注意：只是赋值的顺序发生变化，赋的值还是和原来一样的。
List($a, $b, $c) = [1, 2, 3];
// $a = 1; $b = 2; $c = 3;
和原来的行为还是一样的。
```

二、空的 list()赋值不再允许。

```
list() = $a;
list(,,) = $a;
list($x, list(), $y) = $a;
上面的这些代码运行起来会报错了。
```

三、list()不在支持字符串拆分功能

```
$string = "xy";
list($x, $y) = $string;
```

这段代码最终的结果是：$x == null and $y == null (不会有提示)
PHP5运行的结果是： $x == "x" and $y == "y".

四、除此之外，list()现在也适用于数组对象：

```
list($a, $b) = (object) new ArrayObject([0, 1]);
PHP7结果：$a == 0 and $b == 1.
```

```
PHP5结果：$a == null and $b == null.
```

参考资料：
https://wiki.php.net/rfc/abstract_syntax_tree#changes_to_list
https://wiki.php.net/rfc/fix_list_behavior_inconsistency

2.1.3 foreach 修改

一、foreach()循环对数组内部指针不再起作用。

```
$array = [0, 1, 2];
foreach ($array as &$val)
{
    var_dump(current($array));
}
```

PHP7运行的结果会打印三次 int(0)，也就是说数组的内部指针并没有改变。
之前运行的结果会打印 int(1), int(2)和 bool(false)

二、按照值进行循环的时候，foreach 是对该数组的拷贝操作

foreach 按照值进行循环的时候(by-value)，foreach 是对该数组的一个拷贝进行操作。这样在循环过程中对数组做的修改是不会影响循环行为的。

```
$array = [0, 1, 2];
$ref =& $array; // Necessary to trigger the old behavior
foreach ($array as $val) {
    var_dump($val);
    unset($array[1]);
}
```

上面的代码虽然在循环中把数组的第二个元素 unset 掉，但 PHP7还是会把三个元素打印出来：(0 1 2)
之前老版本的 PHP 会把1跳过，只打印(0 2).

三、按照引用进行循环的时候，对数组的修改会影响循环。

如果在循环的时候是引用的方式，对数组的修改会影响循环行为。不过 PHP7版本优化了很多场景下面位置的维护。比如在循环时往数组中追加元素。

```
$array = [0];
foreach ($array as &$val) {
    var_dump($val);
    $array[1] = 1;
}
```

上面的代码中追加的元素也会参与循环，这样 PHP7会打印"int(0) int(1)"，老版本只会打印"int(0)"。

四、对简单对象 plain (non-Traversable) 的循环。

对简单对象的循环，不管是按照值循环还是按引用循环，和按照引用对数组循环的行为是一样的。不过对位置的管理会更加精确。

五、对迭代对象(Traversable objects)对象行为和之前一致。

编者按：stackoverflow 上面的解释：Traversable object is one that implements Iterator or IteratorAggregate interface。如果一个对象实现了 iterator 或者 IteratorAggregate 接口，即可称之为迭代对象。

参考：Relevant RFC: https://wiki.php.net/rfc/php7_foreach

2.1.4 参数处理机制修改

一、重复参数命名不再支持。

重复的参数命名不再支持。比如下面的代码执行的时候会报错：

```
public function foo($a, $b, $unused, $unused) {
        // ...
}
```

编者按：应该没有人这样用吧。

二、**func_get_arg 和 func_get_args()调整**

func_get_arg()和 func_get_args()这两个方法返回参数当前的值，而不是传入时的值。当前的值有可能会被修改

```
function foo($x)
{
    $x++;
    var_dump(func_get_arg(0));
}
foo(1);
```

上面的代码会打印 2， 而不是 1。如果想打印原始的值，调用的顺序调整下即可。

三、同样在打印异常回溯信息的时候也是显示修改后的值。

```
function foo($x)
{
    $x = 42;
    throw new Exception;
}
foo("string");
```

PHP7的运行结果：Stack trace:

```
#0 file.php(4): foo(42)
#1 {main}
```

PHP5的运行结果：Stack trace:

```
#0 file.php(4): foo('string')
#1 {main}
```

这个调整不会影响代码的行为，不过在调试的时候需要注意这个变化。

其他和参数有关的函数都是同样的调整，比如 debug_backtrace() .

参考：Relevant RFC: https://wiki.php.net/phpng

2.1.5 整型处理机制修改

一、无效八进制数字会报编译错误

无效的八进制数字（包含大于7的数字）会报编译错误，比如下面的代码会报错：

```
$i = 0781; // 8 is not a valid octal digit!
```

老版本的 PHP 会把无效的数字忽略。

二、位移负的位置会产生异常

```
 var_dump(1 >> -1);
 // ArithmeticError: Bit shift by negative number
```

三、左位移如果超出位数返回0

```
var_dump(1 << 64); // int(0)
```

老版本的 PHP 运行结果和 cpu 架构有关系。比如 x86会返回1。

四、右位移超出会返回0或者-1.

```
var_dump(1 >> 64);      // int(0)
var_dump(-1 >> 64);   // int(-1)
```

参考：Relevant RFC: https://wiki.php.net/rfc/integer_semantics

2.1.6 字符串处理机制修改

一、含有十六进制字符的字符串不再视为数字

含有十六进制字符的字符串不再视为数字，也不再区别对待。比如下面的代码：

```
var_dump("0x123" == "291");          // bool(false)          (previously true)
var_dump(is_numeric("0x123"));      // bool(false)          (previously true)
var_dump("0xe" + "0x1");             // int(0)               (previously 16)
var_dump(substr("foo", "0x1"));      // string(3) "foo"      (previously "oo")
// Notice: A non well formed numeric value encountered
```

可以使用 filter_var 函数来检查一个字符串是否包含十六进制字符或者是否可以转成一个整型

```
$str = "0xffff";
$int = filter_var($str, FILTER_VALIDATE_INT, FILTER_FLAG_ALLOW_HEX);
if (false === $int) {
	throw new Exception("Invalid integer!");
}
var_dump($int); // int(65535)
```

二、\u{后面如果包含非法字符会报错

双引号和 heredocs 语法里面增加了 unicode 码点转义语法，“\u{”后面必须是 utf-8字符。如果是非 utf-8字符，会报错：

```
$str = "\u{xyz}"; // Fatal error: Invalid UTF-8 codepoint escape sequence
```

可以通过对第一个\进行转义来避免这种错误。

```
$str = "\\u{xyz}"; // Works fine
“\u”后面如果没有{，则没有影响：
$str = "\u202e"; // Works fine
```

参考资料：
https://wiki.php.net/rfc/remove_hex_support_in_numeric_strings
https://wiki.php.net/rfc/unicode_escape

2.1.7 错误处理机制修改

一、现在有两个异常类：Exception and Error.

PHP7现在有两个异常类，Exception and Error。这两个类都实现了一个新的接口：Throwable。在您的异常处理代码中，类型暗示可能需要调整下。

二、一些致命错误和可恢复致命错误改为抛出 Error 对象。

有一些致命错误和可恢复致命错误现在改为报出 Error 对象。Error 对象是和 Exception 独立的，它们无法被常规的 try/catch 扑获。编者按：需要注册错误处理函数，请参考下面的 RFC。

对于这些已经转为异常的可恢复致命错误，已经无法通过 error handler 静默的忽略掉。尤其是无法忽略类型暗示错误。

三、语法错误会抛出一个 ParseError 对象

语法错误会抛出一个 ParseError 对象，该对象继承自 Error 对象。之前处理 eval()的时候，对于潜在可能错误的代码除了检查返回值或者 error_get_last()之外，还应该捕获 ParseError 对象。

四、内部对象的构造方法如果失败的时候总会抛出异常

内部对象的构造方法如果失败的时候总会报出异常。之前的有一些构造方法会返回 NULL 或者一个无法使用的对象。

五、一些 E_STRICT 错误的级别调整了。

六、参考资料

https://wiki.php.net/rfc/engine_exceptions_for_php7
https://wiki.php.net/rfc/throwable-interface
https://wiki.php.net/rfc/internal_constructor_behaviour
https://wiki.php.net/rfc/reclassify_e_strict

2.1.8 其他语言层面的修改

一、在非兼容$this 语境中以静态方式调用非静态方法将不再支持。

在非兼容$this 语境中以静态方式调用非静态方法将不再支持。在这种场景下面，$this 不会被定义，但调用还可以调用，但会有一个警告提示：

```
class A {
    public function test() { var_dump($this); }
}
// Note: Does NOT extend A

class B {
    public function callNonStaticMethodOfA() { A::test(); }
}
(new B)->callNonStaticMethodOfA();
// Deprecated: Non-static method A::test() should not be called statically
// Notice: Undefined variable $this
NULL
```

注意这种情况适用于在非兼容语境中调用。上面代码的例子中 class B 和 class A 没有关系，所以调用的时候$this 是没有定义的。

但如果 class B 是从 class A 继承的话，该调用是合法的。

二、下面的这些保留字不能用作类名、接口名和 trait 名。

```
bool
int
float
string
null
false
true
```

下面这些关键字已经被留作将来使用，目前可以使用，不会报错，但不建议。

```
resourceobject
```

```
mixed
numeric
```

三、yield 语法调整

在表达式里面使用 yield 语法结构的时候，不再需要括号了。它现在是一个右关联的操作符，优先级介于"print"和"=>"操作符。在某些场景下面行为和之前会不一致。

```
echo yield -1;
echo (yield) - 1;   // 之前的语法解释行为
echo yield (-1);     // 现在的语法解释行为

yield $foo or die;
yield ($foo or die);   // 之前的语法解释行为
(yield $foo) or die;   // 现在的语法解释行为
```

可以通过括号来避免歧义。

备注：关于 yield，大家可以参考鸟哥的这篇文章：http://www.laruence.com/2012/08/30/2738.html

四、其他的一些调整.

移除了 ASP 格式的支持和脚本语法的支持：<% 和 <script language=php>
Removed support for assigning the result of new by reference.（很晦涩，大家有合适的翻译希望告诉我:)）
移除了在非兼容$this 语境中对非静态方法的作用域调用。参考资料：https://wiki.php.net/rfc/incompat_ctx.
http://www.laruence.com/2012/06/14/2628.html
ini 文件里面不再支持#开头的注释，使用;。
$HTTP_RAW_POST_DATA 变量被移除，使用 php://input 来代替。https://wiki.php.net/rfc/remove_alternative_php_tags

2.2 标准库修改

substr()方法在边界切分的时候会返回一个空字符串，不再返回 FALSE.
call_user_method() and call_user_method_array() 被删除了。
当一个输出缓冲是在另外一个输出缓冲处理器创建时，ob_start()返回 E_RECOVERABLE_ERROR 错误，不再返回 E_RROR 错误。
优化了内置的排序算法，对相等元素的排序可能和之前不同。
Fpm-fcgi 移除了 dl()函数。
Setcookie() 如果设置一个没有名字的 cookie，会产生一个 WARNING 错误，不再发送空的 set-cookie 头信息。

2.3 其他修改

- CURL 模块：禁止禁用 CURLOPT_SAFE_UPLOAD 选项，通过 curl 上传文件必须使用 curl_file/CURLFILE 接口。
- DATE 模块：mktime()和 gmmktime()函数移除了$is_dst parameter 参数

- DBA 模块：dba_delete() 如果在 inifile 里面没有找到 key 的时候会返回 false。
- GMP 模块：必须用 libgmp 4.2版本以上。gmp_setbit() and gmp_clrbit()如果传入的 index 为负数的话，会返回 false。
- Intl 模块：移除了别名函数 datefmt_set_timezone_id() 和 IntlDateFormatter::setTimeZoneID()，用 datefmt_set_timezone()

和 IntlDateFormatter::setTimeZone()

- libxml 模块：新增 ibxml 2.9.0引入的 LIBXML_BIGLINES 选项，并在错误报告中增加了行号> 16-bit 的支持。
- Mcrypt 模块：.移除了 mcrypt_generic_end() mcrypt_ecb(), mcrypt_cbc(), mcrypt_cfb() 和 mcrypt_ofb()
- Opcache：移除了 opcache.load_comments 配置项，现在注释加载总是被激活的。
- OpenSSL：移除了"rsa_key_size"、"CN_match" 、"SNI_server_name" 选项。
- PCRE：移除了 /e (PREG_REPLACE_EVAL) 修饰符的支持，使用 preg_replace_callback()来代替。
- PDO_pgsql：删除了 PGSQL_ATTR_DISABLE_NATIVE_PREPARED_STATEMENT 选项。
- Standard：删除了 setlocale()函数里面对字符串类型的支持，使用 LC_* 常量。删除了 set_magic_quotes_runtime() magic_quotes_runtime().
- JSON：json_decode()会拒绝与 RFC 7159不兼容的数字格式。json_decode 第一个参数是空值的时候会返回 json 语法错误。
- Stream：删除别名函数 set_socket_blocking()
- XSL：删除 xsl.security_prefs 选项。
- session
  - session_start()可以接受所有的 INI 设置，可以用数组的方式传入，比如：['cache_limiter'=>'private']
  - save handler 接受 validate_sid(), update_timestamp() ，可用来检查 sid 是否存在，更新 session 数据的时间戳。
  - 增加了 SessionUpdateTimestampHandlerInterface，这个接口里面定义了 validateSid(), updateTimestamp()方法。
  - session.lazy_write(default=On) 配置项可以允许只有 session 数据有变化时才写数据。

3. PHP7 新增功能

一、核心

- 增加了 group use 语法声明。RFC: https://wiki.php.net/rfc/group_use_declarations
- 增加了 null 合并运算符??。RFC: https://wiki.php.net/rfc/isset_ternary
- 64位 PHP7字符串长度可以超过2^31次方字节。
- 增加了 Closure::call()方法。
- 双引号字符串和 heredocs 里面支持使用\u{xxxxx}来声明 unicode 字符。
- define()可以把一个数组定义为常量。
- 增加了合并比较运算符<=>。RFC: https://wiki.php.net/rfc/combined-comparison-operator
- 增加了 yield from 操作符。https://wiki.php.net/rfc/generator-delegation
- 关键词在特定的场景中也可以使用了。RFC: https://wiki.php.net/rfc/context_sensitive_lexer
- 增加了标量类型声明功能。RFC: https://wiki.php.net/rfc/scalar_type_hints_v5

增加接口为用户层提供安全方便的随机数生成器。RFC: https://wiki.php.net/rfc/easy_userland_csprng

二、Opcache 模块

- 增加了基于文件的二级 opcode 缓存机制。可以在 php.ini 文件中设置 opcache.file_cache=<DIR>。当服务重启或者 SHM 重置的时候，使用二级文件缓存机制可以提高性能。
- 也可以设置 opcache.file_cache_only=1来限定只使用文件缓存。
- 可以设置 opcache.file_cache_consistency_checks=0参数来加快加载速度。
- 可以设置 opcache.huge_code_pages=0/1 来决定是否将 PHP code pages 放到 huage pages 里面。http://www.laruence.com/2015/10/02/3069.html
- windows 版本增加了 opcache.file_cache_fallback=1 配置项。

三、OpenSSL 模块

增加了"alpn_protocols"选项。

四、反射

- 增加了 ReflectionGenerator 类，用于 yield from Traces, current file/line 等等。
- 增加了 ReflectionType 类，更好的支持新的返回值和标量声明功能。

五、流

windows 版本增加了块读取的选项。可以通过传递 array("pipe" => array("blocking" => true))参数来激活。

4. PHP7SAPI 模块修改

FPM:

- 修复了 bug#65933。
- Listen = port 会监听所有的 ip 地址，包括 IPv6和 IPv4。

5. PHP7 弃用功能

核心:

- PHP4风格的构造函数将被弃用。(和类名同名的方法视为构造方法，这是 PHP4的语法。)
- 静态调用非静态方法将被弃用。

OpenSSL
capture_session_meta 选项将被弃用，可以调用 stream_get_meta_data()获得。

6. PHP7 修改的函数

- parse_ini_file()和 parse_ini_string()的 scanner_mode 参数增加了 INI_SCANNER_TYPED 选项。
- unserialize()增加了第二个参数，可以用来指定接受的类列表。RFC: https://wiki.php.net/rfc/secure_unserialize
- proc_open()打开的最大限制之前是写死的16，现在这个限制被移除了，最大数量取决于 PHP 可用的内存。windows 版本增加了选项"blocking_pipes"，可用来指定是否强制以块的方式读取。
- array_column():The function now supports an array of objects as well as two-dimensional arrays
- stream_context_create()windows 下面可以接收 array("pipe" => array("blocking" => <boolean>))参数。
- dirname()增加了可选项$levels，可以用来指定目录的层级。dirname(dirname($foo)) => dirname($foo, 2);
- debug_zval_dump()打印的时候，使用 int 代替 long，使用 float 代替 double。

7. PHP7 新增函数

- GMP 模块新增了 gmp_random_seed()函数。
- PCRE 增加了 preg_replace_callback_array 方法。RFC: https://wiki.php.net/rfc/preg_replace_callback_array
- 增加了 intdiv()函数。
- 增加了 error_clear_last()函数来重置错误状态。
- 增加了 ZipArchive::setComapressionIndex()和 ZipArchive::setCompressionName()来设置压缩方法。
- 增加了 deflate_init(), deflate_add(), inflate_init(), inflate_add()。

8. PHP7 新增类和接口

- ReflectionGenerator
- ReflectionType

9. PHP7 移除的扩展和 SAPI

删除了：

- sapi/aolserver
- sapi/apache
- sapi/apache_hooks
- sapi/apache2filter
- sapi/caudium
- sapi/continuity
- sapi/isapi
- sapi/milter
- sapi/nsapi
- sapi/phttpd
- sapi/pi3web
- sapi/roxen
- sapi/thttpd
- sapi/tux
- sapi/webjames
- ext/mssql
- ext/mysql
- ext/sybase_ct
- ext/ereg

参考资料：

https://wiki.php.net/rfc/removal_of_dead_sapis_and_exts
https://wiki.php.net/rfc/remove_deprecated_functionality_in_php7

10. PHP7 其他对扩展的修改

- Mhash 不再是一个扩展，使用前需要用 function_exists("mhash") 来检查方法是否存在。
- GD 库依赖 libwebp，增加对 WebP 的支持。
- Openssl 最小支持的 openssl 版本是0.9.8.

11. PHP7 新增常量

一、核心

增加了 PHP_INT_MIN 常量。

二、PCRE

增加了 PREG_JIT_STACKLIMIT_ERROR 常量。

三、Zlib

. ZLIB_NO_FLUSH
. ZLIB_PARTIAL_FLUSH
. ZLIB_SYNC_FLUSH
. ZLIB_FULL_FLUSH
. ZLIB_BLOCK
. ZLIB_FINISH

四、GD

T1Lib 的支持被移除。下面的方法和资源也被移除了：

Functions:
- imagepsbbox()
- imagepsencodefont()
- imagepsextendedfont()
- imagepsfreefont()
- imagepsloadfont()
- imagepsslantfont()
- imagepstext()

Resources:
- 'gd PS font'
- 'gd PS encoding'

12. PHP7INI 配置文件修改

- 核心：移除了 asp 标签格式的支持。
- 移除了 always_populate_raw_post_data 选项。

13. PHP7windows 支持

一、核心：

64位版本 PHP7原生支持64位整数。
64位版本增加大文件支持。
支持 getrusage()。

二、ftp

ftp 扩展始终以共享的方式提供。
SSL 的支持不再依赖 openssl 扩展，只依赖 openssl 库。如果编译的时候加入了 openssl 的支持，ftp_ssl_connect 会自动激活。

三、imap

静态编译 ext/imap 被禁用。

四、odbc

odbc 模块始终以共享的方式提供。

14. PHP7 其他修改

- NaN 和 Infinity 转为整型的时候，始终为0。
- Instead of being undefined and platform-dependent, NaN and Infinity will always be zero when cast to integer.
- Calling a method on a non-object 现在会抛出一个可以扑获的错误，不再是致命错误。
  https://wiki.php.net/rfc/catchable-call-to-member-of-non-object
- zend_parse_parameters 的错误信息始终使用 integer, float，不再使用 long 和 double 。
- ignore_user_abort 设为真的话，输出缓存会继续工作，即使链接中断。
- Zend 扩展接口使用 zend_extension.op_array_persist_calc()和 zend_extensions.op_array_persist() 。
- 引入了 zend_internal_function.reserved[]数组。

关于我们

青岛易软天创网络科技有限公司成立于2010年，是一家年轻的互联网软件公司。公司以**为天下企业提供专业的管理工具**为使命，坚持开源开放，先后开发了禅道项目管理、蝉知企业门户、然之协同等多款企业管理软件，帮助国内外成千上万的公司改善了其管理运营流程。

易软天创的基础开发语言是 PHP，为了解决自身需求，我们开发了自己的 zentaoPHP 框架和前端 HTML5框架 ZUI。
为了解决 apache, php, mysql 等环境部署问题，我们还编译了 Linux 一键安装包：ZBOX。
我们还专门针对国内开源现状发布了自己的中文授权协议：ZPL。

关注我们