PHP之道

极客学院出版

前言

目前网络上充斥着大量的陈旧信息，让PHP新手误入歧途，传播着错误的实践和糟糕的代码，这必须得到纠正。PHP之道的目标就是搜集PHP最佳实践、编码规范和网络上的权威学习指南，给PHP学习者提供一个易于阅读，快速查找的入口。

> 致谢
>
> 内容撰写：http://wulijun.github.io/php-the-right-way/

更新日期	更新内容
2015-05-29	PHP 之道

目录

1 入门指南

使用当前稳定版本 (5.5)

如果你刚开始学习PHP，请使用最新稳定版本 PHP 5.5 (http://php.net/downloads.php) . PHP近年来有了巨大的改进，增加了许多强大的 新特性。不要让低版本的PHP如5.2的缺陷误导你，这些新特性是对旧版本的重要改进。如果 你想查找一个函数及其用法，可以去官方PHP手册php.net (http://php.net/manual/en/) 查找。

内置的Web服务器

有了它，你可以不用安装和配置功能齐全的Web服务器，就可以开始学习PHP(要求PHP 5.4+版本)。要启动内置Web服务器，需要从你的命令行终端进入项目的Web根目录，执行下面的命令:

```
> php -S localhost:8000
```

- 了解更多内置的命令行Web服务器 (http://php.net/manual/en/features.commandline.webserver.php)

Mac 安装

OSX系统会预装PHP，只是版本比最新稳定版低一点。 目前Lion下是PHP 5.3.6，Mountain Lion下是5.3.10，Mavericks下是5.4.17.

要更新OSX中的PHP，你可以通过那些Mac包管理器 (http://php.net/manual/en/install.macosx.packages.php) 来安装，推荐使用php-osx by Liip (http://php-osx.liip.ch/) 。

另外一种方式是自己编译 (http://php.net/manual/en/install.macosx.compile.php) ，不过要确认已经安装Xcode或"Command Line Tools for Xcode" (https://idmsa.apple.com/IDMSWebAuth/login?appIdKey=891bd3417a7776362562d2197f89480a8547b108fd934911bcbea0110d07f757&baseURL=https://developer.apple.com/&path=%2F%2Fdownloads%2F&rv=1) ，它们可以 从Apple的Mac Developer Center下载。

如果想安装包含了PHP、Apache和MySQL的一键安装包，可以试试MAMP (https://www.mamp.info/en/downloads/) 或XAMPP (https://www.apachefriends.org/index.html) ，里面包含了相应的图形管理工具。

Windows 安装

Windows下有多种方式来安装PHP，你可以下载二进制安装包 (http://windows.php.net/) 。

若只是本地开发和学习，可以直接使用PHP 5.4+内置的Web服务器，还能省去配置服务器的麻烦。如果你喜欢包含PHP、Apache和MySQL的 一键安装包，可以下载Web Platform Installer (http://www.microsoft.com/web/downloads/platform.aspx) 、Zend Server CE (http://www.zend.com/en/products/server-ce) 、XAMPP (https://www.apachefriends.org/index.html) 或WAMP (http://www.wampserver.com/) ，它们可以帮你快速搭建出PHP运行环境。 不过这些工具和你产品的正式运行环境会有一些差别，特别是你在Windows下开发，而代码最终部署在Linux服务器上的时候。

如果你需要把产品部署在Windows上，那么IIS7将给你最稳定和性能最佳的环境，你可以使用phpmanager (http://phpmanager.codeplex.com/) (IIS7下的PHP 管理插件)来配置和管理PHP。IIS7已经内置FastCGI，你只需把PHP配置为它的处理器即可。更多详情可以参考dedicated area on iis.net (http://php.iis.net/) 。

Vagrant

如果你在开发应用和发布应用的时候采用了不同的环境，那么在正式使用时，应用可能出现许多奇怪的BUG。如果你是在开发团队里工作，那么保证各位的开发环境和所有的库文件都是最新的并且处在同一版本，会是件更麻烦的事。 如果你在Windows平台开发并准备部署到Linux（或其他非Windows的平台）上，或者你是在开发团队里工作，那你应该考虑用个虚拟机。这虽然听起来挺麻烦，但是 Vagrant (https://www.vagrantup.com/) 这个程序可以辅助你用几步就创建一个简单的虚拟机。 接下来，你可以手动配置这些基础的环境，或者你可以找个部署软件来替你完成这些事情，比如说Puppet (https://puppetlabs.com/) 或Chef (https://www.chef.io/) 。部署个基础环境，能很好地保证大家的开发环境建立的方式都大致相似，而且还能省去你维护那些复杂的"安装命令"列表的麻烦。 你也可以轻易地毁掉现有的基础环境后再做一个新的出来，这样你就能有一个全新的环境。

Vagrant (https://www.vagrantup.com/) 会创建一些共享文件夹，用来给你在主机和虚拟机之间共享代码用。也就是说，你可以在主机上写好程序，然后在虚拟机中运行。

小助手

如果你想获取一些Vagrant的使用帮助的话，可以参考下面的三个服务：

- Rove (http://rove.io/) : 通过Chef来提供常用的Vagrant构建，其中包含了PHP选项。
- Puphpet (https://puphpet.com/) : 简单的图形界面来设置虚拟机的PHP开发环境，**专注于PHP开发**，不仅可以配置本地的虚拟机，还可以部署到云服务上，它的底层是通过Puppet实现。
- Protobox (http://getprotobox.com/) : 构建在Vagrant上的服务层，提供Web界面设置虚拟机的Web开发环境，通过单个YAML文档控制虚拟机的所有配置。

2

代码风格指南

PHP社区百花齐放，拥有大量的函数库、框架和组件。PHP开发者通常会在自己的项目中使用若干个外部库，因而PHP代码遵循或尽量接近 同一个代码风格就非常重要，可以让开发者方便地把多个代码库集成在自己的项目中。

[框架互操作组href="http://www.php-fig.org/)(即PHP标准组)发布了一系列推荐风格。其中有部分是关于代码风格的，即PSR-0 (https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md") ，[PSR-1href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md")，[PSR-2href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md")和[PSR-4href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md")。 不要让这些名称所混淆，这些推荐仅是一些被其它项目所遵循的规则，如Drupal, Zend, Symfony, CakePHP, phpBB, AWS SDK, FuelPHP, Lithium等，你可以把这些规则用在自己的项目中，或者继续使用你自己的风格。

通常情况下，你的PHP代码应该遵循其中一项或多项标准，从而其他开发者可以方便地阅读和使用你的代码。这些标准都是在前一个标准 上附加新的规则，所以使用PSR-1就同时要求遵循PSR-0，但可以不遵循PSR-2。

- [阅读PSR-0href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md")
- [阅读PSR-1href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-1-basic-coding-standard.md")
- [阅读PSR-2href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-2-coding-style-guide.md")
- [阅读PSR-4href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md")
- Read about PEAR Coding Standards (http://pear.php.net/manual/en/standards.php)
- Read about Symfony Coding Standards (http://symfony.com/doc/current/contributing/code/standards.html)

可以使用PHP_CodeSniffer (http://pear.php.net/package/PHP_CodeSniffer/) 来检查代码是否符合这些标准，文本编辑器插件Sublime Text 2 (https://github.com/benmatselby/sublime-phpcs) 还能 提供实时检查。如果不符合规范，可以使用Fabien Potencier提供的工 具PHP Coding Standards Fixer (http://cs.sensiolabs.org/) 自动修复，不用自己手工修复。

变量名和代码结构建议使用英文符号编写，注释则可以使用各种语言，没有限制。

3

语言亮点

编程范式

PHP是一个灵活的动态语言，支持多种编程范式。这些年来一直在不断的进化，重要的里程碑包括PHP 5.0 (2004)增加完善的 面向对象模型、PHP 5.3 (2009)增加匿名函数和命名空间和PHP 5.4 (2012)增加traits.

面向对象编程

PHP具有完整的面向对象编程特性，如类、抽象类、接口、继承、构造函数、克隆和异常等。

- 学习PHP面向对象编程 (http://php.net/manual/en/language.oop5.php)
- 学习Traits (http://php.net/traits)

函数式编程

PHP支持第一类函数(first-class function)，即函数可以赋值给变量，包括用户自定义的函数和内置函数，然后动态调用它。 函数可以作为参数传递给其他函数(即高阶函数)，也可以作为函数返回值返回。

PHP支持函数递归调用，即函数自己调用自己，不过在实际的PHP代码中，我们更喜欢用迭代来代替递归。

2009年发布的PHP 5.3开始引入支持闭包的匿名函数。

PHP 5.4支持把闭包绑定到对象作用域，并改善其可调用性，从而可以在大部分场景中使用匿名函数替代普通函数。

- 学习更多PHP函数式编程 (http://wulijun.github.io/php-the-right-way/pages/Functional-Programming.html)
- 学习匿名函数 (http://php.net/manual/en/functions.anonymous.php)
- Read about the Closure class (http://php.net/manual/en/class.closure.php)
- More details in the Closures RFC (https://wiki.php.net/rfc/closures)
- Read about Callables (http://php.net/manual/en/language.types.callable.php)
- 学习动态调用函数call_user_func_array (http://php.net/manual/en/function.call-user-func-array.php)

元编程

PHP通过反射API和魔术方法机制，支持多种方式的元编程。开发者通过魔术方法，如 `__get()`，`__set()`，`__clone()`，`__toString()`，`__invoke()` 等，可以改变类的行为。Ruby开发者经常说PHP没有 `method_missing` 方法，实际上通过 `__call()` 和 `__callStatic()` 就可以 完成同样的功能。

命名空间

如前所述，PHP社区的众多开发者已经开发了大量的代码。这意味着一个函数库中的PHP代码可能使用了另外一个库中相同的类名，如果它们共享一个命名空间，则会产生冲突导致异常。

*命名空间*解决了这个问题。如PHP手册里描述的那样，命名空间类似于操作系统中的目录，两个同名文件可以共存于不同的目录。同理，同名的PHP类可以在不同的PHP命名空间下共存，就这么简单。

因而把代码放在自己的命名空间下就显得非常必要，这样其他人就可以放心的使用这些代码，而无需担心与其他函数库的命名冲突。

[PSR-0href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md") 里提供了命名空间的推荐使用方式, 它试图提供一个标准的文件、类和命名空间的使用惯例，从而让代码做到即插即用。

2013年12月，PHP-FIG发布了新的自动加载标准：[PSR-4href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md")，将来可能会替换旧的PSR-0标准。PSR-4要求PHP 5.3版本以上，而目前很多项目用的都是PHP5.2， 因此当前两个标准都可用，但是对于新应用或者包的话，应优先考虑PSR-4.

- 了解更多命名空间
- 了解更多PSR-0
- 了解更多PSR-4

标准PHP库

标准PHP库(SPL)和PHP一起发布，提供了一组类和接口，包括了常用的数据结构如栈，队列和堆等，以及遍历这些数据结构的迭代器， 或者你还可以自己实现SPL接口。

- 学习更多SPL

命令行接口

PHP的主要目的是开发Web应用，不过它的命令行脚本接口(CLI)也非常有用。PHP命令行编程可以帮你完成自动化的任务，如测试，部署和 应用管理。

CLI PHP编程非常强大，可以直接调用你自己的app代码而无需创建Web图像界面，需要注意的是不要把CLI PHP脚本放在公开的web目录下!

在命令行下运行PHP:

```
> php -i
```

选项 `-i` 将会打印PHP配置，类似于[`phpinfo`][33]函数。

选项 `-a` 提供交互式shell，和ruby的IRB或python的交互式shell相似，此外还有很多其他有用的[命令行选项][34]。

接下来写一个简单的"Hello, $name" CLI程序，先创建名为 `hello.php` 的脚本：

```
<?php
if($argc != 2) {
    echo "Usage: php hello.php [name].\n";
    exit(1);
}
$name = $argv[1];
echo "Hello, $name\n";
```

PHP会在脚本运行时根据参数创建两个特殊的变量，$argc (http://php.net/manual/en/reserved.variables.argc.php) 是一个整数，表示参数个数，$argv (http://php.net/manual/en/reserved.variables.argv.php) 是一个数组变量，包含每个参数的值， 它的第一个元素一直是PHP脚本的名字，如本例中为hello.php。

命令运行失败时，可以通过exit()表达式返回一个非0整数来通知shell，常用的exit返回码可以查看列表 (http://www.gsp.com/cgi-bin/man.cgi?section=3&topic=sysexits)

运行上面的脚本，在命令行输入：

```
> php hello.php
Usage: php hello.php [name]
> php hello.php world
Hello, world
```

学习如何在命令行运行PHP (http://php.net/manual/en/features.commandline.php)

学习如何在Windows环境下运行PHP命令行程序 (http://php.net/manual/en/install.windows.commandline.php)

XDebug

调试器是软件开发过程中非常重要的一个工具，通过它，可以跟踪代码的执行过程，查看堆栈信息。XDebug是一个PHP调试器，可以集成在常见的IDE中，提供设置断点、 查看堆栈信息等功能，还可以和PHPUnit、KCacheGrind等工具配合，执行代码覆盖率测试和性能调优。

如果你现在还没有使用调试器，仅仅依靠var_dump/print_r调试的话，XDebug就是你的最佳选择。

安装XDebug (http://xdebug.org/docs/install) 有点复杂，其中一个重要功能"远程调试"——如果你在本地开发代码，然后在虚拟机或其他主机中测试，那么它对你就非常有用。

通常，你需要修改Apache虚拟主机或者.htaccess配置文件，增加：

```
php_value xdebug.remote_host=192.168.?.?
php_value xdebug.remote_port=9000
```

"remote host"和"remote port"对应本机IDE的监听地址和端口，然后设置IDE为"等待连接"模式，打开URL:

```
http://your-website.example.com/index.php?XDEBUG_SESSION_START=1
```

这样IDE就会监控脚本的执行，允许用户设置断点和查看内存中的变量值。

图形化调试器使得单步调试、查看变量和现场运行代码变得异常简单，很多IDE都自带或者通过第三方插件支持xdebug的调试，如Mac平台下的开源软件MacGDBp。

了解更多XDebug (http://xdebug.org/docs/) 了解更多MacGDBp (https://www.bluestatic.org/software/macgdbp/)

4

依赖管理

如今有大量的PHP函数库、框架和组件可供选择，一个项目中可能会使用其中的若干——这就是项目的依赖。到目前为止，PHP还没有有效的 项目依赖管理方案。即使你手工的管理它们，你还不得不处理它们的自动加载问题。

目前主要有两个PHP包管理系统：Composer和PEAR，哪个适合你呢？答案是两个都需要。

- 管理单个项目的依赖时使用Composer
- 管理整个系统的PHP依赖时使用PEAR

通常情况下，Composer包只在你项目中明确指定时才可用，而PEAR包在所有的PHP项目中可用。尽管PEAR听起来似乎更简单，但是根据每个 项目制定方案可能更合适。

Composer and Packagist

Composer是一个**出色**的PHP依赖管理器，把项目的依赖列在 `composer.json` 文件中，然后通过一些简单的命令，Composer就会 自动的帮你下载这些依赖，并配置好自动加载路径。

现在已经有很多PHP库支持Composer，可以在项目中使用它们，具体列表可以点击查看 (https://packagist.org/)，这是官方支持的Composer兼容的PHP库。

如何安装Composer

Composer可以安装在本地(在当前工作目录，不推荐这种方式)，也可以安装在系统中(如/usr/local/bin)。假设你要在本地安装，在 项目的根目录执行：

```
curl -s https://getcomposer.org/installer | php
```

它会下载 `composer.phar` (PHP二进制文档)，然后你就可以用 `php` 运行它来完成项目依赖的管理。 **请注意:**如果 你通过管道直接把下载的代码传给PHP解释器，请先在线阅读代码以确保该代码是安全的。

如何手动安装Composer

手动安装composer有点麻烦，不过很多开发者可能更喜欢这种安装方式。使用交互式安装程序，它会检查你安装的PHP:

- PHP版本满足要求
- `.phar` 文件可以正确执行
- 相关目录的权限设置正确

- 没有加载某些不兼容的扩展
- 相应的 php.ini 设置正确

而手动安装则需要你自己做这些事情，你必须自己权衡利弊，以决定是否手动安装。下面是手动获取Composer的方法:

```
curl -s https://getcomposer.org/composer.phar -o $HOME/local/bin/composer
chmod +x $HOME/local/bin/composer
```

目录 $HOME/local/bin (或你自己选择其它目录)应该在你的 $PATH 环境变量中，从而可以直接运行 composer 命令。

这样文档中描述的运行Composer的命令 php composer.phar install ，就可以用如下命令替代：

```
composer install
```

下面默认你已经在PATH路径下安装Composer。

如何定义和安装依赖

Composer通过文件 composer.json 跟踪项目的依赖。这个文件可以手工维护，也可以通过Composer管理，命令 php composer require 用于添加项目的依赖，如果项目下还没有 composer.json 文件，则会自动创建一个。下面是一个依赖[Twig][40]例子，在项目的根目录执行：

```
composer require twig/twig:~1.8
```

或者通过 composer init 命令也可以一步步地引导你创建项目所需的 composer.json 文件。无论使用哪种方式创建了 composer.json 文件后，就可以通过Composer下载和安装项目依赖到目录 vendors/ :

```
composer install
```

最后在应用的PHP入口文件添加下面代码，告诉PHP使用Composer自动加载器加载项目的依赖库：

```
<?php
require 'vendor/autoload.php';
```

现在你就可以使用项目依赖的库了，它们会在需要的时候自动加载。

5

Coding Practices

基础知识

PHP是一个伟大的语言，可以让各个层次的程序员都能够快速高效地完成编码任务。虽然如此，我们还是经常会因为临时救急或者 坏习惯而忽视了PHP的基础。为了解决这个问题，这部分专门给开发者回顾一下PHP的基础编码实践。

- 继续阅读基础知识 (http://wulijun.github.io/php-the-right-way/pages/The-Basics.html)

日期和时间

PHP使用DateTime类完成读取、设置、比较和计算日期与时间。虽然PHP中有很多日期和时间处理相关的函数，但是DateTime类提供了 完善的面向对象接口完成各项常见操作，而且还能处理时区，这里不作深入介绍。

要使用DateTime，可以用工厂方法 `createFromFormat()` 把原始的日期时间字符串转换为DateTime对象，或直接用 `new DateTime` 获得当前日期和时间的DateTime对象。用 `format()` 方法可以把DateTime对象转换成字符串输出。

```
<?php
$raw = '22. 11. 1968';
$start = \DateTime::createFromFormat('d. m. Y', $raw);

echo "Start date: " . $start->format('m/d/Y') . "\n";
```

DateTime计算时间时通常需要DateInterval类，如 `add()` 和 `sub()` 方法，都是将DateInterval作为参数。尽量避免直接用 时间戳表示时间，夏令时和时区会让时间戳产生歧义，使用间隔日期更为妥当。计算两个日期的差值使用 `diff()` 方法，返回 DateInterval对象，输出显示也很方便。

```
  <?php
// create a copy of $start and add one month and 6 days
$end = clone $start;
$end->add(new \DateInterval('P1M6D'));

$diff = $end->diff($start);
echo "Difference: " . $diff->format('%m month, %d days (total: %a days)') . "\n";
// Difference: 1 month, 6 days (total: 37 days)
```

DateTime对象之间可以直接比较:

```
<?php
if($start < $end) {
```

```
    echo "Start is before end!\n";
}
```

最后一个例子是DatePeriod类的用法，它用于循环事项(recurring events)的迭代。它的构造函数参数为：start和end，均为 DateTime对象,以及返回事项的间隔周期。

```
<?php
// output all thursdays between $start and $end
$periodInterval = \DateInterval::createFromDateString('first thursday');
$periodIterator = new \DatePeriod($start, $periodInterval, $end, \DatePeriod::EXCLUDE_START_DATE);
foreach($periodIterator as $date)
{
    // output each date in the period
    echo $date->format('m/d/Y') . " ";
}
```

了解更多DateTime (http://php.net/manual/en/book.datetime.php) 学习更多日期格式化知识 (http://php.net/manual/en/function.date.php) (accepted date format string options)

设计模式

在代码和项目中使用常见模式是有好处的，可以让代码更易于管理，同时也便于其他开发者理解你的项目。

如果你的项目使用了框架，那么在代码和项目结构上，都会遵循框架的约束，自然也就继承了框架中的各种模式， 这时你所需要考虑的是让上层代码也能够遵循最合适的模式。反之，如果没有使用框架，那么就需要你自己选择 适用于当前项目类型和规模的最佳模式了。

了解更多设计模式 (http://wulijun.github.io/php-the-right-way/pages/Design-Patterns.html)

6

数据库

通常PHP代码使用数据库来持久化存储数据，并有多种方式去连接和操作数据库。在*PHP 5.1.0*之前，推荐的方式有mysql (http://php.net/manual/en/mysql.php) 、 mysqli (http://php.net/mysqli) 和pgsql (http://php.net/pgsql) 等。

如果应用只是使用一个数据库的话，原生驱动就工作的非常好，否则使用MySQL的同时，还需要使用MSSQL或Oracle数据库的话，那么 就没有办法只使用一个原生驱动了，只能分别学习各个数据库驱动的API，这非常令人生厌。

另外需要注意，mysql这个原生驱动已经不在活跃开发状态了，从PHP 5.4.0开始被标记为不推荐使用，意味着将来版本如PHP 5.6可能会 移除这个扩展。如果你正在使用 `mysql_connect()` 和 `mysql_query()` ，那么将来可能要重写部分代码，所以最好用mysqli或PDO来 代替。**如果你正在开发新项目，请不要用mysql扩展，尝试用MySQLi扩展 (http://php.net/mysqli) 或PDO来替代**

PHP: 选择MySQL API (http://php.net/manual/en/mysqlinfo.api.choosing.php)

PDO

PDO是数据库连接抽象库，从PHP 5.1.0开始提供，提供多种数据库的统一的操作接口。PDO不会转化你的SQL查询或者模拟缺失特性； 它只是提供统一的API去连接不同的数据库而已。

更重要的是， `PDO` 允许你绑定SQL查询语句中的变量，而无需担心SQL注入问题，这主要通过PDO statements和变量绑定来实现。

假设PHP脚本接收一个数字ID作为查询参数，从数据库取回一条记录。下面是一种错误的做法：

```
<?php
$pdo = new PDO('sqlite:users.db');
$pdo->query("SELECT name FROM users WHERE id = " . $_GET['id']); // <-- NO!
```

这是非常糟糕的代码，直接在SQL中插入一个原始输入变量，导致潜在的SQL注入风险。假如黑客构造URL： `http://domain.com/?id=1%3BDELETE+FROM+users` 来传入恶意参数id，则 `$_GET['id']` 的变量值为 `1;DELETE FROM users` ， 这将删除数据表中的所有用户！因此，你应该使用PDO的绑定参数功能来处理ID输入参数。

```
prepare('SELECT name FROM users WHERE id = :id');
$stmt->bindParam(':id', $_GET['id'], PDO::PARAM_INT); // <-- Automatically sanitized by PDO
$stmt->execute();
```

这才是正确的代码，在PDO statement中绑定一个参数，使得查询被发给数据库之前，对输入参数进行转义，防止SQL注入攻击。

学习PDO (http://php.net/manual/en/book.pdo.php)

另外一个要注意的问题是，如果数据库连接没有隐式地关闭，那么数据库连接数可能会超过数据库服务器的限制而连接失败，这种 错误在其他编程语言中比较常见。PDO对象在销毁的时候会隐式的关闭数据库连接，只要你把指向它的引用全部删除即可，如设置 为NULL。如果没有，PHP也会在脚本结束时关闭所有非持久化的数据库连接。

了解更多PDO连接 (http://php.net/manual/en/pdo.connections.php)

抽象层

很多框架都提供了自己的数据库抽象层，有的是基于PDO，有的不是。它们通过PHP方法来包装实际的查询，能够模拟出只存在于 某些数据库系统的特性，给你一个真正的数据库抽象层。这么做会带来一些性能的损失，但是在一个需要支持MySQL、PostgreSQL 和SQLite的应用中，这个损失相对于由此带来的代码一致性而言是可以接受的。

有些抽象层遵循[PSR-0href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-0.md")或[PSR-4href="https://github.com/php-fig/fig-standards/blob/master/accepted/PSR-4-autoloader.md")命名空间标准，可以集成在任意的应用中：

Back to Top

异常

异常是大部分流行语言的标准特性，但是PHP开发者却不太重视。其他语言如 Ruby极度倚赖异常，在任何错误发生的时候，如HTTP请求失败 、DB查询错误，甚至图片资源未找到，都会抛出一个异常，以及时提示那里发生了一个错误。

PHP则对此很宽松，如调用 `file_get_contents()` 失败，只是返回 `FALSE` 并提示一个warning信息而已。很多老的PHP框架，如 CodeIgniter会返回false，然后在自己的日志里记录一个消息，开发者需要使用如 `$this->upload->get_error()` 的方式来查看发生了什么 错误。这么做需要你自己检查是否有错误，并需要根据不同类调用不同的方法来获取错误消息，而不能让错误明显的显示出来。

这种做法的另外一个弊端是当类自动在屏幕打印一个错误，然后退出进程，阻止了其他开发者动态处理该错误的机会。而异常则是让开发者知道 发生了错误，并让他们选择如何处理：

```
<?php
$email = new Fuel\Email;
$email->subject('My Subject');
$email->body('How the heck are you?');
```

```
$email->to('guy@example.com', 'Some Guy');

try
{
   $email->send();
}
catch(Fuel\Email\ValidationFailedException $e)
{
   // The validation failed
}
catch(Fuel\Email\SendingFailedException $e)
{
   // The driver could not send the email
}
finally
{
   // Executed regardless of whether an exception has been thrown, and before normal execution resumes
}
```

SPL异常

默认的异常类Exception包含的上下文信息很少，对于debug不方便，常见的做法是创建更具体的子类：

```
<?php
class ValidationException extends Exception {}
```

这使得你可以包含多个catch子句来处理不同的异常，但是这又会导致创建*很多的*自定义异常类，可以用SPL中的异常类来缓解这个问题 SPL扩展 (http://wulijun.github.io/#standard_php_library) .

如使用__call()魔术方法，对不存在的方法调用抛出一个throw new BadFunctionCallException;，既避免了抛出含义模糊的 Exception异常，也省去了自定义异常类的麻烦。

- 学习更多Exceptions (http://php.net/manual/en/language.exceptions.php)
- 了解跟多SPL Exceptions (http://php.net/manual/en/spl.exceptions.php)
- PHP中的异常嵌套 (http://www.brandonsavage.net/exceptional-php-nesting-exceptions-in-php/)
- PHP 5.3异常最佳实践 (http://ralphschindler.com/2010/09/15/exception-best-practices-in-php-5-3)

7

安全

Web应用安全

总有一些人会千方百计的想着破坏你的Web应用，提前想办法加强自己的Web应用的安全性非常重要。幸运的是， The Open Web Application Security Project (https://www.owasp.org/index.php/Main_Page) (OWASP) 已经提供详尽的已知安全问题列表和防范对策。 每个关注Web安全的开发者都应该仔细阅读该列表。

阅读OWASP安全指南 (https://www.owasp.org/index.php/Guide_Table_of_Contents)

数据过滤

永远不要在PHP代码中信任外部输入，在使用之前一定要先过滤和验证，filter_var和filter_input函数可以帮助过滤文本和 验证文本格式(如邮箱地址)。

外部输入可以是：$_GET和$_POST表单输入数据、$_SERVER超级变量中的某些值和通过fopen('php://input', 'r')获取的 HTTP请求体。要记住外部输入不仅仅是用户提交的表单数据，还包括上传和下载的文件、session变量、cookie数据和第三方Web服务提供的 数据等。

当外部数据被存储合并之后，下次读取时，它们仍然算是外部输入，每次在代码中处理的时候，需要问自己是否已经正确过滤，是否可以 信任它们。

数据需要根据不同用处，进行不同的*过滤*，如果把未经过滤的数据输出到HTML页面，它可以在你的网站里执行HTML和JavaScript！即通常 说的跨站脚本攻击(XSS)。避免XSS的一个策略就是使用strip_tags函数过滤外部输入的所有HTML标签，或者使用htmlentities/ htmlspecialchars转义其中的HTML实体。

另外一个例子是传给命令行命令的选项，这可能非常危险(通常不是一个好主意)，不过你可以用内置的escapeshellarg函数过滤命令行的 参数。

最后一个例子是根据外部输入来从文件系统中加载文件的操作，可以通过修改路径中的文件名实施攻击。你需要过滤输入中的”/”，“../”, null bytes, 或其他特殊字符，以防止加载不能公开的包含敏感数据的文件。

- 学习更多数据过滤 (http://php.net/manual/en/book.filter.php)
- 了解更多filter_var (http://php.net/manual/en/function.filter-var.php)
- 了解更多filter_input (http://php.net/manual/en/function.filter-input.php)
- 学习更多null字节处理 (http://php.net/manual/en/security.filesystem.nullbytes.php)

数据清洗

数据清洗就是删除或转义外部输入中的非法或不安全字符。比如把外部数据输出到HTML或插入到SQL语句之前，需要先清洗外部输入。 当你通过PDO绑定数据时，它会替你转义输入数据。

有时候需要允许外部数据包含安全的HTML标签，并输出到HTML页面中。这个比较难处理，可以考虑使用其他更严格的格式，如 或BBCode，实在不能的话，可以使用HTML Purifier (http://htmlpurifier.org/) 库来进行数据清洗。

了解更多数据清洗过滤器 (http://php.net/manual/en/filter.filters.sanitize.php)

数据验证

数据验证外部输入就是你预期的，如你在处理注册表单时，需要验证email地址、电话号码和年龄等数据

了解更多数据验证过滤器 (http://php.net/manual/en/filter.filters.validate.php)

配置文件

在创建应用的配置文件时，请遵循下面的业界最佳实践：

- 配置文件保存在Web不能直接访问和上传的目录中。
- 如果配置文件只能放在文档根目录时，请使用.php作为文件名后缀，这样即使直接访问该配置文件，也不会输出配置信息。
- 配置文件内容应该加密，或者对文件设置访问权限。

注册全局变量

提示:从PHP 5.4.0开始，register_globals配置已经删除，不再生效。保留这个配置，只是提示依赖该配置的应用进行升级。

启用register_globals配置后，$_POST, $_GET和$_REQUEST中的变量自动注册为全局变量，使得应用很难辨别变量的确切来源，从而产生安全漏洞。

例如：$_GET['foo']将注册变量$foo，这会覆盖程序中未声明的同名变量。如果你使用PHP 5.4.0之前的版本，请确保已经把register_globals设置为off。

Register_globals in the PHP manual (http://php.net/manual/en/security.globals.php)

错误提示

错误日志可以帮助追查应用的Bug，但是也会暴露应用的结构信息而产生安全问题，为此，需要在开发环境和线上环境设置不同的配置，防止 敏感信息的泄漏。

开发环境

要在开发环境显示错误提示，需要在php.ini中配置以下配置项:

```
display_errors = On
display_startup_errors = On
error_reporting = -1
log_errors = On
```

来自php.net: (http://php.net/manual/en/function.error-reporting.php)

> -1表示显示各种错误，包括将来增加的新错误类型，和PHP 5.4中的E_ALL行为相同。

E_STRICT错误级别在5.3.0版本引入，不在E_ALL中，不过5.4.0版本开始，E_ALL包含E_STRICT级别的错误。所以在5.3版本中，要显示 所有错误，需要把error_reporting设置为-1或者E_ALL | E_STRICT。

各PHP版本显示所有错误的配置

```
< 5.3 -1 or E_ALL
  5.3 -1 or E_ALL | E_STRICT
> 5.3 -1 or E_ALL
```

线上环境

要在线上环境隐藏错误提示，需要在 `php.ini` 中配置以下配置项:

```
display_errors = Off
display_startup_errors = Off
error_reporting = E_ALL
log_errors = On
```

这样设置后，线上错误会记录到Web服务器的错误日志中，而不是直接显示给用户。如果想了解更多错误提示相关的设置，请参考手册:

- error_reporting (http://php.net/manual/en/errorfunc.configuration.php#ini.error-reporting)
- display_errors (http://php.net/manual/en/errorfunc.configuration.php#ini.display-errors)
- display_startup_errors (http://php.net/manual/en/errorfunc.configuration.php#ini.display-startup-errors)
- log_errors (http://php.net/manual/en/errorfunc.configuration.php#ini.log-errors)

8

测试

为PHP代码编写自动化测试被认为是一个最佳实践，可以帮助你构建出高质量的应用。自动化测试可以帮助你确认没有因为重构或添加 新功能而破坏原有功能，所以应该重视自动化测试。

PHP有多种类型的测试工具和框架可以使用，具体方法各有区别——但是它们的目标都是避免手工测试，满足大型QA组织的需求，保证最新的 更改没有破坏已有功能。

测试驱动开发

Wikipedia (http://en.wikipedia.org/wiki/Test-driven_development) 的定义：

> 测试驱动开发(TDD)是以非常短的开发周期，不断进行迭代的软件开发流程：首先开发者针对改进或新功能编写失败的自动化测试用例，然后编写代码使测试用例通过， 最后重构代码，让代码满足可接受的标准。Kent Beck，该技术的创建者或者说重新发现者，在2003年声明TDD鼓励简单的设计和提振信心。

目前对应用有多种类型的测试：

单元测试

单元测试是从编写开始，贯穿于整个开发周期的一种用于保证函数、类和方法的行为与预期一致的编程方法。通过检查各个函数和方法的输入和输出值，你可以保证它们 内部逻辑已经正确执行；通过依赖注入、编写mock类和stubs，你可以验证依赖是否已经正确处理，提高测试覆盖率。

在编写一个类或函数的时候，应该为它的每一个行为创建一个单元测试，至少你要保证它收到错误参数时能够触发错误，而参数正确时能正常工作。这可以帮你在后面 修改类或函数的时候，确认已有功能仍然正常工作。PHP中var_dump()的功能与此类似，但是它是无法用于创建应用的。

单元测试的另外一个用武之地是在给开源项目贡献代码时，如果你编写一个测试，证明代码存在bug，然后修复代码，让测试通过，这样该补丁被接受的概率要高很多。 如果你的项目接受人家的补丁，你应该把单元测试作为项目的一项要求。

PHPUnit (https://phpunit.de/) 是PHP应用的单元测试框架的业界标准，其他几个可选框架是：

- atoum (https://github.com/atoum/atoum)
- Enhance PHP (https://github.com/Enhance-PHP/Enhance-PHP)
- PUnit (http://www.ibm.com/developerworks/cn/java/j-lo-punit/)
- SimpleTest (http://simpletest.org/)

集成测试

Wikipedia (http://en.wikipedia.org/wiki/Integration_testing) 的定义:

> 集成测试(也称集成与测试，缩写为I&T)是把各个独立模块集成在一起，作为一个整体进行测试的软件测试阶段，它处于单元测试和验收测试之间。集成测试把已经 做过单元测试的模块集成在一块，然后运行集成测试用例，最终输出一个可以进行系统测试的系统。

很多单元测试工具同时也可以用于集成测试，并且原理也是相通的。

功能测试

有时也称为验收测试，使用工具创建自动化的测试用例，然后在真实的系统上运行，这一点与单元测试验证单个模块的正确性和集成测试验证模块间交互的正确性是有 区别的，这些工具通常使用真实的数据集来模拟真实用户的使用行为来验证系统的正确性。

功能测试工具

- Selenium
- Mink (http://mink.behat.org/en/latest/)
- Codeception (http://codeception.com/) is a full-stack testing framework that includes acceptance testing tools
- Storyplayer (http://datasift.github.io/storyplayer/) is a full-stack testing framework that includes su pport for creating and destroying test environments on demand

行为驱动开发

行为驱动开发(BDD)有两种方式：SpecBDD和StoryBDD。SpecBDD关注代码的技术行为，而StoryBDD关注业务、特性和交互，这两种方式都有对应的PHP框架。

采用StoryBDD，开发者编写人类可读的故事来描述应用的行为，然后这些故事可以作为应用的测试用例。PHP中用于StoryBDD编程的框架是Behat，从Ruby 的[Cucumber][67]项目演化而来，实现了Gherkin DSL来描述特性行为。

采用SpecBDD，开发者编写规格说明来描述实际代码的行为，与测试一个函数或方法不同，规格描述了一个函数或方法应该具有的行为。PHP中的PHPSpec框 架提供该编程方式的支持，它也是从Ruby的[RSpec projec t][68]演化而来。

BDD链接

- Behat (http://docs.behat.org/en/v2.5/) , the StoryBDD framework for PHP, inspired by Ruby’s Cucumber (https://cucumber.io/) project;
- PHPSpec (http://www.phpspec.net/en/latest/) , the SpecBDD framework for PHP, inspired by Ruby’s RSpec (http://rspec.info/) project;
- Codeception (https://github.com/codeception/codeception) is a full-stack testing framework that uses BDD principles.
- Behat (http://docs.behat.org/en/v2.5/) , the StoryBDD framework for PHP, inspired by Ruby's [Cucumber][67] project;
- [PHPSpec][70], the SpecBDD framework for PHP, inspired by Ruby's [RSpec][68] project;
- [Codeception][71] is a full-stack testing framework that uses BDD principles.

测试辅助工具

除了测试驱动和行为驱动开发框架，还有大量的通用框架和函数库，可以在各种开发方法下使用。

工具链接

Selenium is a browser automation tool which can be integrated with PHPUnit (https://phpunit.de/manual/current/en/selenium.html) Mockery (https://github.com/padraic/mockery) is a Mock Object Framework which can be integrated with PHPUnit (https://phpunit.de/) or PHPSpec (http://www.phpspec.net/en/latest/) Prophecy (https://github.com/phpspec/prophecy) is a highly opinionated yet very powerful and flexible PHP object mocking framework. It’s integrated with PHPSpec (http://www.phpspec.net/en/latest/) and can be used with PHPUnit (https://phpunit.de/) .

9

服务器和部署

部署PHP应用到线上Web服务器的方式有很多种。

平台即服务(PaaS)

PaaS提供运行PHP Web应用所需的系统和网络环境，对PHP应用和框架只需要做少量的配置即可。

现在PaaS已经成为部署、托管和扩展各种规模的PHP应用的流行方式，可以在 resources (http://wulijun.github.io/php-the-right-way/#resources) 部分查看PHP PaaS "平台即服务"供应商列表。

虚拟或独立主机

如果你愿意或想学习系统管理，那么虚拟或独立主机可以让你完全控制自己的运行环境。

nginx和PHP-FPM

PHP通过内置的FastCGI进程管理器(FPM)，可以非常高效地和轻量级的高性能Web服务器[nginx][72]进行通信。 nginx比Apache消耗更少的内存，能更好的处理并发请求，这在内存限制较多的虚拟主机环境中尤为重要。

- 阅读更多nginx (http://nginx.org/)
- 阅读更多PHP-FPM (http://php.net/manual/en/install.fpm.php)
- 学习如何配置安全的nginx和PHP-FPM (https://nealpoole.com/blog/2011/04/setting-up-php-fastcgi-and-nginx-dont-trust-the-tutorials-check-your-configuration/)

Apache和PHP

PHP和Apache是一个老搭档，历史悠久。Apache有很强的可配置性和大量的[扩展模块][73]， 是共享主机中常见的Web服务器，完美支持各种PHP框架和开源应用(如WordPress)。可惜的是，默认情况下，Apache比nginx更耗资源，并发处理能力不强。

Apache有多种方式运行PHP，最常见简单的方式是使用mod_php5的[prefork MPM][74]方式， 缺点是它对内存的利用效率不高，如果你不想深入学习服务器的管理，那么这种最简单的方式就是你的最佳选择了。注意，如果你使用mod_php5，最好使用 prefork MPM方式。

如果你想追求高性能和高稳定性，那么也可以为Apache选择与nginx类似的FPM系统[worker MPM][75]或 [event MPM][76]，它们分别使用mod_fastcgi和mod_fcgid模块。FPM方式可以更高效的利用内存，运行 速度更快，但是配置也相对复杂一些。

- 阅读更多Apache(http://httpd.apache.org/)
- 深入学习多进程模块(http://httpd.apache.org/docs/2.4/mod/mpm_common.html)
- 阅读更多mod_fastcgi(http://www.fastcgi.com/mod_fastcgi/docs/mod_fastcgi.html)
- 阅读更多mod_fcgid(http://httpd.apache.org/mod_fcgid/)

共享主机

PHP非常流行，很少有服务器没有安装PHP的，因而有很多共享主机，不过需要注意服务器上的PHP是否是最新稳定 版本。共享主机允许多个开发者把自己的网站部署在上面，这样的好处是费用非常便宜，坏处是你不知道将和哪些 网站共享主机，因此需要仔细考虑机器负载和安全问题。如果项目预算允许的话，避免使用共享主机是上策。

10

缓存

PHP自身效率很高，但是执行创建远程连接、加载文件等操作时容易出现瓶颈，幸运的是，我们有很多工具来加速这部分操作，或减少 这些耗时操作的执行次数。

字节码缓存

在一个PHP文件被执行时，它先被编译为字节码(也称opcode)，然后这些字节码被执行。如果文件没有修改，那么字节码也会保持不变， 这意味着编译这一步白白浪费了CPU资源。

这就是引入字节码缓存的原因，通过把字节码保存在内存中来消除冗余的编译，重用它们完成后续的调用。配置字节码缓存非常简单， 而且可以极大地提高应用的执行效率，没有理由不使用字节码缓存。

PHP 5.5开始内置字节码缓存组件[OPcache][84]，老版本的PHP可以使用第三方的字节码缓存组件， 流行的字节码缓存方案有：

- APC (http://php.net/manual/en/book.apc.php) (PHP 5.4 and earlier)
- XCache (http://xcache.lighttpd.net/)
- Zend Optimizer+ (http://www.zend.com/en/products/server) (part of Zend Server package)
- WinCache (http://www.iis.net/downloads/microsoft/wincache-extension) (extension for MS Windows Server)

对象缓存

很多时候，在代码中缓存对象可以带来很大的收益，例如获取代价很大的数据和查询结果很少变化的数据库调用。我们可以使用对象 缓存系统缓存这些数据，大大加快后续的同类访问请求。如果你在取得这些数据之后，把它们缓存在系统中，在后续对这些数据的请求 中，就可以直接使用缓存中的对象，这么做可以很大的提示系统性能，减少服务器的负载。

很多流行的字节码缓存方案也允许你缓存自定义数据，因此我们更应该充分利用对象缓存功能。APCu、XCache和WinCache都提供API， 让你把数据缓存在他们的内存cache中。

使用最多的内存对象缓存系统是APCu和memcached，APCu是很好的一个对象缓存方案，它提供了简单的API来让你把对象存储在内存中，而且 配置和使用都非常容易，它的一个缺点是只能在本机使用。Memcached则是另外一种方式，它是一个单独的服务，可以通过网络访问，这 意味着可以在一个地方写入数据，然后在不同的系统中访问这份数据。

Note that when running PHP as a (Fast-)CGI application inside your webserver, every PHP process will have its own cache, i.e. APCu data is not shared between your worker processes. In these cases, you might want to consider using memcached instead, as it's not tied to the PHP processes.

在单机性能上，APCu通常比Memcached更高，如果你不需要多台服务器或者其他Memcached的高级功能，APCu可能是你的最佳选择。

APCu的示例:

```
<?php
// check if there is data saved as 'expensive_data' in cache
$data = apc_fetch('expensive_data');
if ($data === false) {
    // data is not in cache; save result of expensive call for later use
    apc_add('expensive_data', $data = get_expensive_data());
}

print_r($data);
```

Note that prior to PHP 5.5, APC provides both an object cache and a bytecode cache. APCu is a project to bring APC’s object cache to PHP 5.5+, since PHP now has a built-in bytecode cache (OPcache).

学习更多对象缓存系统:

APCu (https://github.com/krakjoe/apcu) APC Functions (http://php.net/manual/en/ref.apc.php) Memcached (http://memcached.org/) Redis (http://redis.io/) XCache APIs (http://xcache.lighttpd.net/wiki/XcacheApi) WinCache Functions (http://php.net/manual/en/ref.wincache.php)

11

资源

框架

大量的PHP开发者使用框架，而不是重复发明轮子来创建自己的Web应用。框架抽象出底层通用的业务逻辑，给使用者了提供简单易用的接口。

不是所有的项目都需要框架，有时候原生的PHP就能满足需求，但是需要框架的时候，有三种类型的框架可供选择：

- 微框架
- 全能(Full-Stack)框架
- 组件框架

微框架仅是一个包装器(Wrapper)，尽量快地把HTTP请求路由到回调函数、控制器或方法上，有些框架也会提供一些函数库，如基本的数据库 操作。微框架主要用于构建远程HTTP服务。

全能框架则是在微框架的功能之上提供了更多的功能特性，如ORM，验证组件等。

组件框架则是一组独立功能库的集合，多个基于组件的框架集合在一起，甚至可以用作微框架或者全能框架。

- PHP常用框架列表 (https://github.com/codeguy/php-the-right-way/wiki/Frameworks)

组件

如前所述，组件是另外一种创建、实现和发布开源代码的方式，当前社区存在很多组件库，最主要的两个：

- Packagist
- PEAR

这两个库都有用于安装和升级的命令行工具，已经在依赖管理部分讲述.

还有基于组件的框架，你可以使用其中的组件，它们相互之间依赖很少，或完全独立，如FuelPHP (https://github.com/fuelphp/validation) 验证包， 就可以脱离FuelPHP框架而独立使用。这些项目就相当于一个可重用的组件库：