“Head First Design Patterns” (2004)

Notes open for creative commons use @ developer blog: https://unfoldkyle.com, github: SmilingStallman, email: kmiskell@protonmail.com

Language Features

The New PHP
-With additions of features like namespaces, closures, traits, etc., PHP has advanced far in encapsulation and allows moving away from massive monolithic apps.

-For version control, check out Vagrant. For provision tools, look into Ansible, Chef, and Puppet.

-Composer allows for easy dependency management

-PSR,by PHP-FIG community working group has become widely accepted PHP coding standard

-PHPUnit as powerful unit testing tool

-PHP FastCGI process manager

-PHP engine – parses, interprets, and executes PHP code.

-Addition of HipHop Virtual Machine PHP engine, built by Facebook. Exists as alt to older Zend engine. FB also built Hack, white is built on top of core PHP, as a PHP version of Typescript of sorts, with static typing, new data structures, etc.. Both use JIT (Just In Time) compiler for increased speed.

-New features make PHP much more suitable for developing command line tools

Namespace Basics
-Organize PHP files into a virtual hierarchy similar to standard filesystem directory structures

-Each namespace has it’s own global scope. Used in packages to prevent name clashes across files and packages, as well as package uploaders. Vendor + package specific namespaces.

-Namespaces allow import of code from any other package, whether created by another user (ex. *Laravel)* or created in house, providing a sandboxed execution environment, to fully prevent name clashes, as well as organize code, to make easy for re-use, etc..

-Sub-namespaces prevent clashes inside same main namespace by further breaking namespace down

-Declare namespace as first line in PHP file via syntax:
namespace MyVendor\MyNamespace;

-After deceleration of namespace, all code under this declaration now exists in defined namespace. Any other file that also declares that same namespace also exists in same namespace as other same namespace files.

-A namespace acts as an encapsulation platform for grouping and organizing PHP classes together, just how directories organize and group files.

-Recommended to store PHP files in directory structure matching namespace hierarchy

-To access methods, etc. in *namespaceXFile.php* from *namespaceYFile.php*, must first import *Y* into *X* via standard *require_once*, *include*, etc. import.s

-Once import *Y* into *X*, can access *Y* namespace members in multiple way:

```
//file1.php
namespace SmilingStallman\Main\Sub;
function print_me(){ echo 'file1'; }

//file2.php
namespace SmilingStallman\Main;
include 'file1.php';
function print_me(){ echo 'file2'; }

//a) unqualified name – similar to referencing file via relative filename
print_me()                  //prints 'file2', referencing print_me() in current \Main namespace

//b) qualified name – similar to relative file path, where call in relation to current namespace
Sub\print_me();             //prints 'file1'. Note that do not call with \Main, as already in \Main

//c) fully qualified name – similar to absolute (full) file path, where call by full namespace hierarchy
\SmilingStallman\Main\Sub\print_me()            //prints 'file1'
```

Namespace Imports and Aliases

-PHP *use* is an operator for bringing something from another namespace into current namespace

-Syntax:
```
use \Main\Sub\someClass;            //import with fully qualified name
use \Main\Sub\someFunction;
```

-Note that PHP also uses *use* to inherit traits in classes and inherit variables in closures. This is just bad design, though, so don't get the uses of *use* confused.

-Put *use* statements at top of file, just under *namespace* declaration. *use* must be called in global scope only.

-Since this brings *Y* into *X*, this could again result in name clashes, hence *use* is often used with the *as*

-Syntax:
```
use function \Main\Sub\SomeFunction as AnotherFunction;
AnotherFunction();
```

-Even with *as* name clashes can still occur, so comprehended to give unique *as* alias via:

```
use \Main\Sub\SomeClass as SubSomeClass;        //prepend file/class name
```

-Can import classes, const, and functions, each with own syntax

```
use \Main\Sub\someClass;s
new someClass();s

use function Sub\someFunction;
someFunction();

use const Sub\someConst;
echo someConst;
```

-Each of above can also be combine with *as* aliasing

Multiple Namespaces

Can declare multiple namespaces in one file, but recommended against, as results in multi classes per file, etc.

<?php
namespace Foo{ ……. }
namespace Bar{ ……. }

Global Namespace
-When don't declare namespace for file, still exists in namespace, the default PHP *global* namespace

Autoloading
-Namespaces provide basis for autoloading, as to be discussed later in further detail