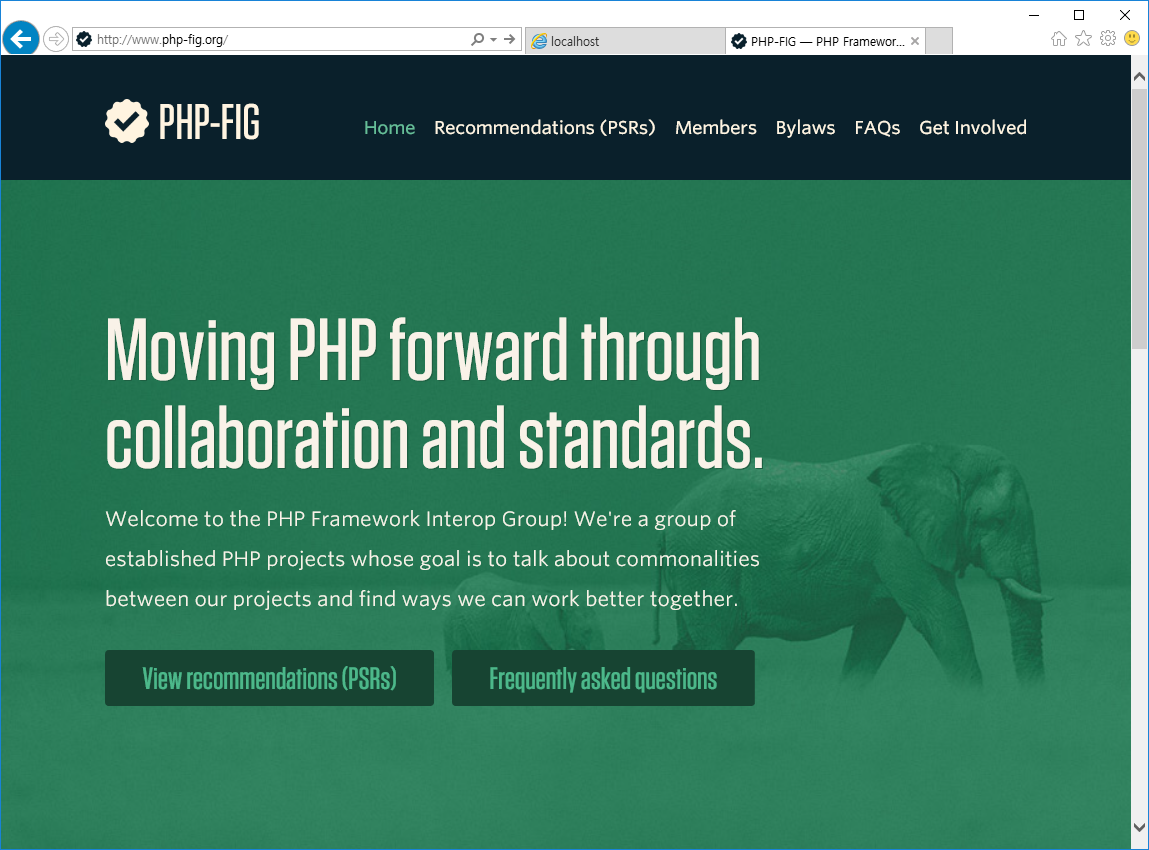
jinyPHP 1권 - 04. PSR

**04 PSR**

PHP는 자유로운 스타일의 언어입니다. 또한 프로그램 소스 코드는 개발자 성향과 개발 방법에 따라서 코드 작성 표기가 자유롭습니다.

PHP는 오픈소스, 프레임워크 등 공개로 인하여 여러 개발자들이 소스를 공유하고, 함께 개발하고 있습니다. 또한 이러한 공유와 나눔으로 PHP는 많은 발전을 하고 있습니다.

PSR은 PHP Standard Recommendation의 약자로 PHP-FIG에서는 권장하는 PHP의 코드 스타일입니다. PHP-FIG에서 PSR이라는 코드 스타일을 권장하는 이유는 더 많은 사람들이 PHP 소스를 공유하고 함께 개발하기 위해서입니다.



**공식 사이트** <http://www.php-fig.org>

PSR 권고안은 함께 만들어가는 권고안입니다. 지금도 새로운 아이디어가 제안되고 많은 개발자들의 의견을 받아들이고 있습니다. 이런 의견들은 초안 → 리뷰 → 수용 세 가지 상태로 구분됩니다.

PSR 권고안은 PSR + 번호 형태로 발표합니다. 현재 공식 사이트에서 수용된 권고안은 다음과 같습니다.

* PSR-1: 기본 표준코딩
* PSR-2: 코딩 스타일
* PSR-3: 로거 인터페이스
* PSR-4: 오토로딩
* PSR-6: 캐시 인터페이스
* PSR-7: HTTP 메시지 인터페이스
* PSR-13: 하이퍼미디어 링크
* PSR-16: 간단한 캐시

이외 리뷰 중인 제안과 다수의 초안 제안 등이 있습니다. 아직 수용되지 않은 초안과 리뷰 단계의 제안은 웹 사이트를 방문하면 확인할 수 있습니다.

PSR은 다수의 PHP 프레임워크에서 적용되고 있습니다. 또한 이 책에서도 PSR 코딩 스타일 규격을 따라서 학습자에게도 향후 PSR 스타일의 코딩과 소스 공유 시 이해가 쉽도록 설명하려고 합니다.

**4.1 PSR-1 표준 코딩**

PSR-1은 기본적인 PHP의 표준 코딩 가이드입니다. PHP 태그, 인코딩 및 이름 규칙 등을 서로 규약하여 호환성을 증대합니다.

**4.1.1 PHP 파일**

**시작 태그:** PHP의 시작과 종료 태그입니다. PHP는 다양한 태그 약자가 존재합니다. PSR-1에서는 **<?php ?>** 와 <?= ?> 만 사용하기를 권고합니다.

**인코딩:** PHP 소스 텍스트 파일의 인코딩을 규약합니다. 보통 텍스트 에디터에는 다양한 문서 인코딩 방식을 지원하지만 PSR-1 에서는 문서 인코딩 포맷을 **UTF-8**로 사용할 것을 권고합니다.

UTF-8은 그냥 UTF-8 방식과 BOM(byte order Mark)이 적용된 두 가지 방식이 있습니다. PSR-1에서는 BOM이 적용되지 않은 그냥 UTF-8을 권고합니다.

**4.1.2 사이드 이펙트**

코드를 작성할 때는 사이드 이펙트(side effect)가 적게 발생되도록 주의합니다. 사이드 이펙트는 다양한 곳에서 발생할 수 있습니다. 한 개의 PHP 파일 안에서도 클래스, 함수, 상수 등 새로운 심볼을 선언할 때, 또한 이것들이 다른 사이드 이펙트의 원인이 되어서는 안 됩니다.

‘사이드 이펙트’의 의미는 클래스, 함수, 상수를 직접적인 관련 없이 로직이 실행되는 것입니다. 사이드 이펙트 적용될 수도 있지만 그 사용을 구체적으로 제한하지는 않습니다(이를테면, 출력 발생, **require** 또는 **include 사용**, 외부 서비스 연결, ini 설정 변경, 방출 오류 또는 예외, 글로벌 또는 정적변수 변경, 파일을 읽거나 저장 등).

다음은 선언과 사이드 이펙트 관련된 파일의 예 입니다.

|  |
| --- |
| <?php  // 사이드 이펙트: ini 설정을 변경  ini\_set('error\_reporting', E\_ALL);  // 사이드 이펙트: 파일I/O 등을 이용하여 파일을 읽음  include "file.php";  // 사이드 이펙트: 화면 I/O 로 출력을 발생  echo "<html>\n";  // 선언  function foo()  {  // 함수 본문 내용...  } |

다음은 사이드 이펙트가 적게 발생하는 스타일입니다.

|  |
| --- |
| <?php  // 선언  function foo()  {  // 함수 본문 내용...  }  // conditional declaration is \*not\* a side effect  if (! function\_exists('bar')) {  function bar()  {  // 함수 본문 내용...  }  } |

**4.1.3 네임스페이스와 클래스**

네임스페이스와 클래스는 다음 장에서 설명하지만 네임스페이스와 클래스는 PSR 오토로딩 권장을 따라서 사용해야 합니다.

클래스는 PHP 파일 하나에 한 개씩 작성하는 것을 권장합니다. 또한 네임스페이스는 이름은 최소한 한 개 이상의 레벨로 작성합니다. 1개 이상을 사용해야 하는 이유는 최상의 레벨은 벤더 이름으로 사용하기 때문입니다.

PHP 5.3 이상에서는 네임스페이스를 사용하는 것을 적극 권장합니다.

|  |
| --- |
| <?php // PHP 5.3 and later: namespace Vendor\Model;  class Foo { } |

PHP 5.2 이전 버전에서는 “\_”를 통해 벤더, 모델, 클래스명 등으로 구분할 것을 권장합니다.

|  |
| --- |
| <?php // PHP 5.2.x and earlier: class Vendor\_Model\_Foo { } |

**4.1.4 클래스 상수, 프로퍼티 및 매서드**

클래스, 인터페이스, 트레이트 등의 이름을 작명할 때 다음과 같은 작명 방식을 권장합니다.

**클래스명:** 클래스 이름은 낙타 표기법(camel case)인 StudlyCaps 스타일로 작성합니다. 클래스 이름의 첫 글자는 대문자로 표기하는 타이틀 케이스 형태로 작성을 권고합니다.

예) StudyCaffe, JinyPhp

**상수:** 상수는 대문자와 밑줄 “\_”로 구분하여 사용합니다.

|  |
| --- |
| <?php namespace Vendor\Model;  class Foo {  const VERSION = '1.0';  const DATE\_APPROVED = '2012-06-01'; } |

**메서드:** 메서드 이름은 낙타 표기법 으로 작성합니다. 메서드의 시작 문자는 소문자를 사용합니다.

예) camelCase (), loveStudy()

**프로퍼티:** 프로퍼티(변수)는 낙타 표기법(StudlyCaps, camelCase, under\_score)을 구분하지는 않습니다. 코드상에서 일관성 있게 사용하면 됩니다.

**4.2 PSR-2 코딩 스타일**

PSR-2는 PSR-1의 확장 규칙입니다. 또한 일관적인 코드 스타일과 관련된 권고안입니다.

PSR-2의 일관적인 코드 스타일은 여러 사람이 한 명이 작성한 것과 같이 보이는 효과를 주기 위함입니다. 여러 개발자와 공동 작업을 하거나 작성된 코드가 공개가 될 때 일관성은 매우 중요합니다.

**4.2.1 들여쓰기**

들여쓰기는 코드의 가독성을 위해서 매우 중요한 코딩 스타일입니다. 들여쓰기는 에디터마다 적용하는 스타일이 다르고, 개발자들 사이에서도 공백 수, 탭 키 등 다양한 방식으로 들여쓰기를 합니다.

PSR-2에서는 일관된 들여쓰기를 위해서 tab 대신에 **공백 문자 4**개를 기본으로 들여쓰기를 합니다. 사용하는 에디터마다 tab 키의 크기는 각기 다를 수 있습니다. 미리 에디터 환경설정에서 tab의 크기를 지정해 놓으면 편리합니다.

**4.2.2 파일**

PHP 파일은 각 줄의 끝에는 유닉스 개행 문자(Unix LF)를 사용해야 합니다. 또한 파일의 맨 마지막 줄에는 공백의 한 줄을 포함합니다.

그리고 PHP 파일의 **종료 태그 ?>를 생략**합니다.?>를 생각하게 되면 예상치 않은 출력 오류를 방지할 수도 있습니다.

**4.2.3 줄**

한 줄에 수많은 코드를 작성한다면 코드의 줄이 너무 길어 읽기가 불편할 것입니다. 그래서 PSR-2에서는 각 줄의 길이를 80자 이내 작성을 권고합니다. 만일 80자를 넘더라도 최대 120자를 넘지 않도록 하고 있습니다.

또한 각 줄의 마지막에는 공백 문자가 들어가지 않도록 합니다.

**4.2.4 키워드**

PSR-2에서 PHP의 키워드는 소문자 사용을 권고합니다. 또한 상수도 true, false, null 등 **소문자로 사용할 것을 권고**합니다. 대문자로 TRUE, FALSE, NULL로 쓰고 있다면 PSR-2 스타일로 바꾸도록 합니다.

**4.2.5 네임스페이스**

네임스페이스 키워드와 use 키워드 다음에는 **비어 있는 한 줄**을 추가합니다.

한 줄을 추가함으로써 문장을 쉽게 구분할 수 있고 가독성을 높입니다.

|  |
| --- |
| <?php namespace Vendor\Package;  use FooClass; use BarClass as Bar; use OtherVendor\OtherPackage\BazClass;  // ... additional PHP code … |

**4.2.5 클래스**

클래스, 인터페이스, 트레이트도 이와 같은 스타일 코딩 권고를 따릅니다.

extends와 implements 키워드는 클래스명과 함께 한 줄에 위치해야 합니다. 그리고 본문을 감싸고 있는 중괄호 **{** 시작은 class 키워드와 같은 위치의 **다음 줄**에 위치합니다.

|  |
| --- |
| <?php  namespace Vendor\Package;  use FooClass;  use BarClass as Bar;  use OtherVendor\OtherPackage\BazClass;  class ClassName extends ParentClass implements \ArrayAccess, \Countable  **{**  // constants, properties, methods  **}** |

**프로퍼티**

모든 프로퍼티는 접근 권한 속성을 앞에 선언해야 합니다. 접근 속성은 public, protected, private 세 가지가 있습니다. var와 같은 속성 키워드는 사용하지 않습니다.

프로퍼티 이름은 밑줄 “\_”과 같은 개인적인 접두사를 사용하지 않습니다. 또한 메서드 이름 다음은 공백 문자를 삽입하지 않습니다.

|  |
| --- |
| <?php  namespace Vendor\Package;  class ClassName  {  public $foo = null;  } |

**매서드**

모든 메서드는 접근 권한 속성을 앞에 선언해야 합니다. 접근 속성은 public, protected, private 세 가지가 있습니다.

메서드 이름은 밑줄 “\_”과 같은 개인적인 접두사를 사용하지 않습니다. 또한 메서드 이름 다음은 공백 문자를 삽입하지 안습니다.

메서드의 내용을 담고 있는 중괄호는 다음 줄 자신의 위치를 시작으로 그 다음 줄에 닫는 중괄호를 작성합니다. 여는 괄호 뒤에 공백이 있으면 안 되며 닫는 괄호 앞에 공백이 있어서는 안 됩니다.

메서드 선언은 다음과 같습니다. 괄호, 쉼표, 공백 및 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php namespace Vendor\Package;  class ClassName {  public function fooBarBaz($arg1, &$arg2, $arg3 = [])  {  // method body  } } |

**메서드 인자**

메서드 인자 뒤에는 공백이 들어가면 안 됩니다. 여러 개의 인자를 전달할 경우 인자명 뒤에 쉼표를 삽입을 하고 **공백을 하나 추가**합니다.

|  |
| --- |
| <?php namespace Vendor\Package;  class ClassName {  public function foo($arg1, &$arg2, $arg3 = [])  {  // method body  } } |

만일 매개변수 인자가 많아서 여러 줄로 구성해야 할 때도 있습니다. 이런 경우에는 **들여쓰기**를 적용합니다.

**매개변수는 한 줄에 하나씩**만 작성합니다. 닫는 괄호와 여는 중괄호는 한 줄의 공백을 사용하여 각각의 줄에 함께 있어야 합니다.

|  |
| --- |
| <?php namespace Vendor\Package;  class ClassName {  public function aVeryLongMethodName(  ClassTypeHint $arg1,  &$arg2,  array $arg3 = []  ) {  // method body  } } |

**abstract, final, static**

abstract와 final 은 가시성 키워드 앞에 선언돼야 합니다. static은 가시성 키워드 다음에 선언해야 합니다.

|  |
| --- |
| <?php namespace Vendor\Package;  abstract class ClassName {  protected static $foo;   abstract protected function zim();   final public static function bar()  {  // method body  } } |

**메서드와 함수의 호출**

메서드와 함수를 호출할 때 이름 뒤 소괄호 시작 사이에는 공백이 없어야 합니다. 또한 소괄호 시작과 인수명 사이, 소괄호가 끝나는 앞에도 공백이 없어야 합니다.

또한 여러 인수가 전달될 때 구분자 쉼표 앞에도 공백이 없어야 합니다.

|  |
| --- |
| <?php bar(); $foo->bar($arg1); Foo::bar($arg2, $arg3); |

인수가 여러 개일 때는 다음 줄에 시작하여 들여쓰기를 할 수 있습니다. 여러 줄로 인수를 작성할 때 첫 번째 항목은 다음 줄에 있어야 하며 한 줄에 하나의 인수만 있어야 합니다.

|  |
| --- |
| <?php $foo->bar(  $longArgument,  $longerArgument,  $muchLongerArgument ); |

**4.2.6 제어 구조**

모든 제어 구조문 키워드 다음에는 하나의 공백을 포함합니다. if, elsif, else, switch, while, do wile, for, foreach, try, catch입니다.

제어문의 본문을 담는 중괄호는 앞뒤에 공백을 포함하지 않습니다. 또한 중괄호 안에는 들여쓰기를 권장합니다.

**PSR-2 코딩 스타일: 조건문**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php if ($expr1) {  // if body } elseif ($expr2) {  // elseif body } else {  // else body; } |

조건문의 거짓 조건 else if 처럼 두 개의 키워드 말고 else if처럼 하나의 단어로 공백 없이 붙여서 사용합니다.

**PSR-2 코딩 스타일: 스위치, 케이스**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호 배치에 유의합니다.

스위치의 케이스 문장은 들여쓰기를 적용합니다. 브레이크 키워드는 한 단계 더 들여쓰기합니다.

|  |
| --- |
| <?php switch ($expr) {  case 0:  echo 'First case, with a break';  break;  case 1:  echo 'Second case, which falls through';  // no break  case 2:  case 3:  case 4:  echo 'Third case, return instead of break';  return;  default:  echo 'Default case';  break; } |

**PSR-2 코딩 스타일: while**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php while ($expr) {  // structure body } |

**PSR-2 코딩 스타일: do~while**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백 및 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php do {  // structure body; } while ($expr); |

**PSR-2 코딩 스타일 : for**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php for ($i = 0; $i < 10; $i++) {  // for body } |

**PSR-2 코딩 스타일: foreach**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php foreach ($iterable as $key => $value) {  // foreach body } |

**PSR-2 코딩 스타일: try~catch**

다음은 PSR-2 권고의 코딩 스타일입니다. 괄호, 공백, 중괄호의 배치에 유의합니다.

|  |
| --- |
| <?php try {  // try body } catch (FirstExceptionType $e) {  // catch body } catch (OtherExceptionType $e) {  // catch body } |