|  |
| --- |
| CMSKorea 코딩스타일  코딩스타일 |

이 문서의 저작권은 ㈜씨엠에스코리아에게 있으며 무단 복제 및 배포를 금합니다.

**개정이력**

|  |  |  |
| --- | --- | --- |
| 개정차수 | 개정일 | 주 요 내 용 |
| 1 | 2022.12.20 | 최초 문서 작성 |

목차

[1 Overview 4](#_Toc122617478)

[2 Basic 4](#_Toc122617479)

[3 Files 4](#_Toc122617480)

[4 Namespace and Class Names 5](#_Toc122617481)

[5 Classes, Properties, And Methods 5](#_Toc122617482)

[5.1 Extends and Implements 6](#_Toc122617483)

[5.2 Properties 6](#_Toc122617484)

[5.3 Methods 6](#_Toc122617485)

[5.4 Method Arguments 6](#_Toc122617486)

[5.5 Abstract, final and static 7](#_Toc122617487)

[5.6 Method and Function Calls 7](#_Toc122617488)

[6 Lines 8](#_Toc122617489)

[7 Indenting 8](#_Toc122617490)

[8 Keywords and True/False/Null 8](#_Toc122617491)

[9 Control Structures 8](#_Toc122617492)

[9.1 Basic 8](#_Toc122617493)

[9.2 If, elseif, else 8](#_Toc122617494)

[9.3 Switch, case 9](#_Toc122617495)

[9.4 While, do whie 9](#_Toc122617496)

[9.5 For 10](#_Toc122617497)

[9.6 Foreach 10](#_Toc122617498)

[9.7 Try, catch 10](#_Toc122617499)

[10 Operators 10](#_Toc122617500)

[10.1 Basic 10](#_Toc122617501)

[10.2 Unary operators 11](#_Toc122617502)

[10.3 Binary operators 11](#_Toc122617503)

[10.4 Ternary operators 11](#_Toc122617504)

**이 문서에서 설명하는 내용은 (주)씨엠에스코리아의 지적 자산입니다.**

1. **Overview**

이 문서는 ㈜씨엠에스코리아에서 사용하는 프로그램 코딩 스타일을 정의한 문서입니다.

㈜씨엠에스코리아에서는 PHP 권장하는 PSR-1, PSR-2, PSR-4 스타일을 기본으로 사용합니다.

그리고 PHP 프로그래밍 표시법은 카멜표기법(Camel case)을 사용합니다.

1. **Basic**

파일의 시작과 끝은 <?php 와 ?>를 사용합니다.

네임스페이스와 클래스는 반드시 “autoloading” PSR [PSR-4] 규칙을 따라야 합니다.

클래스 상수는 모두 대문자로 사용하며 띄어쓰기는 밑줄 구분 기호로 선언해야 합니다.

매서드 이름은 반드시 카멜표기법을 사용해야합니다.

1. **Files**

PHP 코드는 긴 태그를 사용해야합니다. <?php ?>

PHP 코드로만 이루어진 파일에서는 닫는 태그 ?>를 생략할 수 있습니다.

|  |
| --- |
| <?php  // declaration  Function foo() {  // function body  }  // conditional declaration is …  If (!function\_exists(‘bar’)) {  Function bar() {  // function body  }  } |

1. **Namespace and Class Names**

파일 경로에서 클래스를 자동로드하기 위한 스펙을 설명합니다.

네임스페이스와 클래스 이름은 PSR-4 규격에 따라서 “오토로딩”을 지원하도록 만들어야합니다.

㈜씨엠에스코리아에서는 PHP 5.2 이하에서 만들어진 코드규칙을 적용하여 선언합니다.

|  |
| --- |
| \<NamespaceName>(\<SubNamespaceNames>)\*\<ClassName>  Was  | - Model  | - Foo.php  예) class Was\_Model\_Foo {  } |

1. **Classes, Properties, And Methods**

클래스는 객체의 상태를 나타내는 필드와 객체의 행동을 나타내는 메소드로 구성됩니다.

클래스는 Property 선언 후 Method를 선언합니다.

클래스의 여는 중괄호는 클래스 선언 마지막에 1칸의 공백을 주고 선언해야 합니다.

클래스의 닫는 중괄호는 본문 뒤의 다음 줄에 선언해야 합니다.

|  |
| --- |
| class Foo {  protected $\_property;  public function method() {  // method body…  }  } |

* 1. Extends and Implements

Extends 및 implements 키워드는 클래스 이름과 같은 줄에 선언해야 합니다.

|  |
| --- |
| require\_once ‘ParentClass.php’;  require\_once ‘ArrayAccess.php’;  require\_once ‘Countable.php’;  class Foo extends ParentClass implements ArrayAccess, Countable {  } |

* 1. Properties

모든 속성에서 가시성을 반드시 선언해야 합니다. var 키워드는 속성을 선언하는데 사용할 수 없습니다.

속성을 사용처에 맞는 단어로 선언해야 합니다. 카멜표기법을 사용합니다. 첫 글자는 무조건 소문자 입니다.

* 1. Methods

모든 메소드에서 가시성을 선언해야 합니다. 메소드 이름은 카멜표기법을 사용하여 선언합니다.

* 1. Method Arguments

인수 목록에서 각 쉼표 앞에 공백이 있으면 안되며 각 쉼표 뒤에 하나의 공백이 있어야 합니다. 디폴트 값을 가진 메소드 인수는 인수 목록의 끝에 와야 합니다.

|  |
| --- |
| class Foo {  protected $\_property;  public function method($arg1, &$arg2, $arg3 = []) {  // method body…  }  } |

* 1. Abstract, final and static

Abstract, final 선언은 가시성 선언 앞에 위치해야 합니다.

Static 선언은 가시성 뒤에 위치 합니다.

|  |
| --- |
| class Foo {  protected static $\_property;    abstract protected function ball();  final public static function bar() {  // method body…  }  } |

* 1. **Method and Function Calls**

메서드나 함수 호출을 할 때 메서드 또는 함수 이름과 여는 괄호 사이에는 공백이 없어야 합니다. 또 한 인수 목록에서 각 쉼표 앞에 공백이 있으면 안되며 각 쉼표 뒤에 하나의 공백이 있어야 합니다.

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

인수 목록은 여러 줄에 걸쳐 나뉘어 질 수 있으며, 이때 각 줄은 한 번 들여쓰기가 필요 합니다. 이렇게 할 때 한 줄에 하나의 인수만 있어야 합니다.

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

1. **Lines**

행은 120자를 기준으로 작성을 합니다. 이보다 긴 행은 여러 행으로 나눠서 작성해야 합니다.

줄 끝에는 공백 문자가 없어야 합니다.

가독성을 높이고 관련 코드 블록을 나타내기 위해 빈 줄을 추가 할 수 있습니다.

한 줄에는 하나 이상의 문장이 있어서는 안됩니다.

1. **Indenting**

코드는 반드시 4칸의 들여쓰기를 사용해야 합니다. 탭을 사용하지 않습니다.

탭 키의 옵션을 에디터에서 4칸의 스페이스로 변경합니다.

1. **Keywords and True/False/Null**

PHP 키워드는 소문자를 사용합니다. PHP 상수 true, false, null 은 소문자로 사용합니다.

Boolean 대신 bool, integer 대시 int 등 짧은 형태의 타입 키워드를 사용합니다.

1. **Control Structures**
   1. Basic

제어 구조 키워드는 다음에 하나의 공백이 있어야 합니다.

여닫는 괄호 앞뒤에 공백이 있으면 안됩니다.

닫는 괄호와 여는 중괄호 사이에 하나의 공백이 있어야 합니다.

구조체는 한 번 들여 쓰기 되어야 합니다.

닫는 중괄호는 몸체 뒤에 다음 줄에 있어야 합니다.

* 1. If, elseif, else

If 구조는 다음과 같습니다. 모든 제어 키워드가 단일 단어처럼 보이도록 elseif 대신 else if 라고 규정합니다.

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

* 1. Switch, case

스위치 구조는 다음과 같습니다. case 문은 스위치에서 한 번 들여 쓰기 되어야 합니다. Break 키워드 (또는 다른 종료 키워드) 는 본문과 동일한 수준에서 들여 쓰기 되어야 합니다.

|  |
| --- |
| <?php  switch ($expr) {  case 0:  echo ‘first case, with a break;  break;  case 1:  echo ‘Second case, which falls through’;  case 2:  case 3:  case 4:  echo ‘Third case, return instead of break’;  return;  default:  echo ‘Default case’;  break;  } |

* 1. While, do whie

while, do while 문의 구조는 아래와 같습니다.

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

* 1. For

for 문은 다음과 같습니다.

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

* 1. Foreach

foreach 문은 다음과 같습니다.

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

* 1. Try, catch

try catch 문은 다음과 같습니다.

|  |
| --- |
| <?php  try {  // try body  } catch (FirstException $e) {  // catch body  } catch (OtherException $e {  // catch body  } |

1. **Operators**
   1. **Basic**

연산자의 스타일 규칙은 피연산자 수 별로 그룹화됩니다.

연산자 주변에 공백이 허용되며, 가독성을 위해 여러 공백이 있을 수 있습니다.

* 1. **Unary operators**

단항 연산자

증가/감소 연산자, 논리 not 연산자, 변환 및 부정 연산자는 연산자와 피연산자 사이에 공백이 없어야 합니다.

|  |
| --- |
| <?php  $value++;  --$value;  $a = !$b;  $c = +$d;  $e = -$f; |

타입 캐스팅 연산자는 괄호 안에 공백이 없어야 합니다.

|  |
| --- |
| <?php  $intValue = (int) $input; |

* 1. **Binary operators**

이진 연산자

모든 이진 arithmetic, comparison, assignment, bitwise, logical, 문자열 및 type 연산자의 앞뒤에는 최소한 하나 이상의 공백이 있어야 합니다.

|  |
| --- |
| <?php  if ($a === $b) {  $foo = $bar ?? $a ?? $b;  } elseif ($a > $b) {  $foo = $a + $b \* $c;  } elseif ($a <= $c) {  $foo += $bar % $a & $b \*\* $c;  } |

* 1. **Ternary operators**

삼항 연산자

조건부 연산자 ? 및 : 문자 앞뒤에는 하나 이상의 공백이 와야합니다.

|  |
| --- |
| <?php  $variable = $foo ? ‘foo’ : ‘bar’; |

조건부 연산자의 중간 피연산자가 생략되면 연산자는 다른 이진 연산자와 동알한 스타일 규칙을 적용합니다.

|  |
| --- |
| <?php  $variable = $foo ?: ‘bar’; |