* Не использовать табуляцию в коде: все отступы должны делаться с помощью двух пробелов;

<?php

**class** sfFoo

{

**public** **function** bar()

{

sfCoffee::make();

}

}

* Не добавлять пробелы после открывающей скобки и перед закрывающей скобкой;

<?php

**if** ($myVar == getRequestValue($name)) // правильно

**if** ( $myVar == getRequestValue($name) ) // неправильно

* Использовать «Верблюжий регистр»(CamelCase) написания имен переменных, функций и методов:
  + Слова пишутся слитно;
  + Кроме первого, каждое новое слово начинается с большой буквы;

Примеры:

* Неправильно:
  + function GetStudents()
  + function get\_students()
  + function Get\_Students()
* Правильно:
  + function getStudents()
  + function isStudent()
  + function takeFirstStudent()
* Названия переменных, методов и функций должны иметь осмысленное значение. Запрещено называть переменные транслитом;
* Открывающая и закрывающая фигурные скобки должны располагаться всегда на одной вертикальной линии;
* Всегда использовать фигурные скобки для тела выражения, независимо от количества выражений;
* У каждого метода или члена класса должен быть явно казан модификатор доступности: private, protected, public;
* Не заканчивать код библиотечных файлов закрывающим тегом ?>. Потому что это не необходимо и может создавать трудности при выводе;
* В теле функций, перед оператором возврата результата функции return, всегда надо оставлять пустую строку. Это необходимо для лучшей читаемости;

Пример:

<?php

**function** makeCoffee()

{

**if** (**false** !== isSleeping() && **false** !== hasEnoughCaffeineForToday())

{

canMakeCoffee();

**return** 1;

}

**else**

{

cantMakeCoffee();

}

**return** **null**;

}

* Все однолинейные комментарии должны располагаться на отдельной строке и между символом комментария // и текстом комментария необходимо устанавливать ровно один пробел;

Пример:

<?php

// пробел перед текстом комментария

* Избегать внедрения выражений в строку, вместо этого использовать конкатенацию:

Пример:

<?php

$exampleString = 'кодировать';

$newString = "**$exampleString** круто!"; // плохой стиль

$newString = $exampleString.' круто!'; // хороший стиль

$newString = sprintf('%s is awesome', $exampleString); // для сообщения об ошибках и строк, с множеством замен

* Использовать в нижнем регистре предоставляемые PHP типизированные константы: false, true, null. Так же и для array(). Для пользовательских констант наоборот, использовать верхний регистр букв;

Пример:

**define('MY\_CONSTANT', 'foo/bar')**

Но лучше объявлять константы внутри класса:

Пример:

<?php

**class** sfCoffee

{

**const** COFFEE\_NAME = 'Руссианно';

}

var\_dump(sfCoffee::COFFEE\_NAME);

* Для проверки переменной на null не использовать встроенную функцию is\_null();

Пример:

<?php

**if** (**null** !== $coffee)

{

**echo** 'Ура! У меня есть кофе';

}

* При сравнении переменной с литералом, помещать литерал всегда вначале, использовать сравнение с проверкой типов там, где это применимо;

Пример:

<?php

**if** (1 === $variable)

* Использовать PHP контроль типов для описания сигнатур методов и функций;

Пример:

<?php

**public** **function** checkCoffee(sfCoffee $coffee)

{

// тело функции

}

* Все функции и моды должны иметь описания в формате [phpdoc](https://www.phpdoc.org/about);

Пример:

<?php

/\*\*

\* Метод проверяет, что указанный объект sfCoffee удовлетворяет всем требованиям к кофе

\*

\* @param sfCoffee $coffee Объект типа sfCoffee

\*

\* @return boolean Результат проверки

\*/

**public** **function** checkCoffee(sfCoffee $coffee)

{

// тело функции

}

* Все public методы и функции должны иметь проверку своих входных аргументов на корректность;
* Служебные методы классов не должны быть открытыми и иметь модификатор public;
* Классы должны корректно отрабатывать независимо от других модулей, т.е. если методу переданы некорректные данные, то должна быть вызвана ошибка. Метод не должен обрабатывать некорректные данные и возвращать неверный результат;
* Каждому объекту из предметной области должен строго соответствовать один объект в коде. Не допускается множество объектов одной физической сущности;
* Текст исключений должен быть понятно оформлены, чтобы не требовалось для определения ошибки смотреть исходный код;
* Код обязательно должен быть снабжен комментариями достаточно подробными, чтобы программист, не знакомый с языком программирования PHP понял логику работы;
* Размер одного метода не должен превышать 100 строк;
* Переменная в любой момент времени должна иметь только один тип, не допускается использование переменных с разными типами под одним именем в одном методе;
* Вложенность блоков в методах и функциях не превышает 4 уровня;
* В PHP коде (за исключением phpTemplate) запрещены вставки HTML, JavaScript. Все вставки должны производиться через специальные шаблоны;
* Работа с БД должна строиться на концепции как можно меньшего числа обращений к БД. Запрещается использовать запросы в цикле без явных на это оснований;
* Желательно, чтобы поиск по БД производился по ключам;
* Транзакции в БД должны производиться со строгим порядком изменения таблиц, чтобы не возникал deadlock.