

# Стандарты оформления кода

1. Правильное именование переменных и функций.
  1. Имена глобальных переменных и констант должны вводиться в верхнем регистре
  2. Префикс переменной определяет её тип. Для переменной типа string следует использовать префикс «s», например \$sInput. Для bool - «b». Для объектов - это «o», например, \$oBook, для массива - «a». При этом, если у нас есть массив с данными одного типа, то помимо префикса массива, нужно указывать еще и тип элементов массива, например, \$aoBooks. Кстати, предусмотрен префикс и для смешанных типов mixed - «m».
  3. Переменные, определенные в классе, как private всегда носят префикс «\_», например, \$\_oController
  4. Временные переменные и итераторы следует называть коротко. Например, i, j, k, l, m, n. Для строковых и числовых временных переменных рационально использование названий без префикса, например \$name, \$pass, \$counter.
  5. Переменные, использующие несколько слов в названии, пишутся слитно, без знака «\_». Например, \$oBookComment. Но если речь идет о константе или глобальной переменной, то здесь целесообразно использование знака «\_». Например, \$CACHE\_DIR.
  6. Для имен методов и функций следует применять те же правила, что и для переменных. Начинаться имена методов и функций должны с символов в нижнем регистре.
  7. При создании модели объекта с неопределенной выборкой, например, \$oShopItems = Core\_Entity::factory('shop\_item') имя переменной должно быть во множественном числе с содержанием префикса o - object. При уточнении выборки и получении массива объектов \$aoShopItems = \$oShopItems->findAll() префикс должен быть ao - array of objects. При создании модели конкретного объекта имя переменной должно быть в единственном числе, например, \$oShopItem = Core\_Entity::factory('shop\_item', 25).
2. Использование пустого пространства и фигурных скобок.
  1. Чтобы усилить логическую структуру кода, следует использовать переносы строк - пустое пространство. Не бойтесь ставить переносы строк, давайте коду как можно больше свободы, но без излишеств - двойные переносы строк делать не следует.
    1. Следует разбивать блоки инициализации ряда переменных по логическому смыслу, например

```
$name = "testname";  
$login = "testlogin";  
$pass = "testpass";  
$oSite = Core_Entity::factory('site', $site_id);  
$oUser = Core_Entity::factory('user', $user_id);  
$oController = new Core_Controller($oUser);
```

Первый блок с переменными содержит какие-то предустановленные значения. Второй - создание объектов. Третий - контроллера. Следует записать так:

```
$name = "testname";  
$login = "testlogin";
```

```

$pass = "testpass";

$oSite = Core_Entity::factory('site', $site_id);
$oUser = Core_Entity::factory('user', $user_id);

$oController = new Core_Controller($oUser);

```

- Управляющие конструкции – условия и циклы следует отделять переносами. Например,

```

$bonus = ;

/* Вычисляем сумму бонусов товаров */
foreach ($aoShopOrderItems as $oShopOrderItem)
{
    $bonus += $this->getBonus($oShopOrderItem->shop_item_id);
}

return $bonus;

```

- Фигурные скобки всегда следует отделять переносом строки.
- Отступы
  - Отступы внутри управляющих конструкций должны присутствовать обязательно.
  - Также отступы и переносы строк следует использовать в сложных условиях и в сложных конструкциях, подразумевающих создание вложенных объектов.

```

if (isset(Core_Page::instance()->object)
    && strpos(
        get_class(Core_Page::instance()->object),
        'Informationssystem_Controller_Show'
    ) !== false
)
{
    $Informationssystem_Controller_Show
    ->addEntity(
        Core::factory('Core_Xml_Entity')
        ->name('current_group_id')
    ->value(intval(Core_Page::instance()->object->group))
    )->addEntity(
        Core::factory('Core_Xml_Entity')
        ->name('current_item_id')
    ->value(intval(Core_Page::instance()->object->item))
    )
);
}

```

- Не используйте оператор echo для вывода HTML-кода. Следует использовать закрывающие и открывающие PHP-теги: `<?php ?>`
- Вставки кода должны быть окружены копиями с комментарием. Пример

```
/**
```

```
* В этой строке должен быть комментарий о том, что делает код внутри комментариев.  
*  
* @author Kuts Artem, KAD Systems (©) 2017  
* @date 18-11-2017  
* Начало >>  
*/  
  
/**  
* << Конец  
* @author Kuts Artem, KAD Systems (©) 2017  
* @date 18-11-2017  
*/
```

5. Параметры методов класса указываются в порядке: входные значения, изменяемые значения, выходные значения.