

Руководство разработчика команды «KAD :: Systems»

Добро пожаловать в нашу команду! Мы занимаемся разработкой сайтов на системе управления HostCMS 6.x, а также их доработкой и технической поддержкой. Все взаимодействия мы производим внутри нашей системы для работы с клиентами, которая показала себя очень эффективно. Данная система носит название «KAD :: Outsource». Она является основным инструментом для связи всех участников процесса разработки. Регулярно мы обновляем её и дорабатываем новый функционал. Данное руководство состоит из двух разделов. Первый из них – это правила и стандарты, принятые в нашей команде. Второй – практические советы и рекомендации.

Правила и стандарты

Правила оформления кода

Независимо от уровня профессионализма или отличительных особенностей синтаксиса, очень просто написать неряшливый или малопонятный код. Трудный для чтения код сложен в обслуживании и отладке. Скверный стиль кодирования ассоциируется с недостатком профессионализма.

В нашей команде мы используем единый стиль программирования. Помимо удобочитаемости такого кода, красивый код говорит о нашем профессионализме. Если вы ставите над кодом копирайты, то есть подписываетесь именем нашей команды, то ваш код должен быть на уровне. Иначе вы ставите под угрозу репутацию самой команды и каждого из её участников. Основные правила 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.
 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);
```

2. 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;
```

2. Фигурные скобки всегда следует отделять переносом строки.
3. Отступы
 1. Отступы внутри управляющих конструкций должны присутствовать

обязательно.

2. 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)
    )
    );
}
```

3. Не используйте оператор echo для вывода html кода. Следует использовать закрывающие и открывающие php теги: <?php ?>

Правила интеграции верстки

Вынесены в документ "[Стандарты интеграции](#)"

Порядок и правила обращения к техподдержке

1. Поприветствовать. Вежливость не будет лишней.
2. Подробно разъяснить суть вопроса, приложить ссылки, скриншоты.
3. Приложить необходимые доступы. Обычно хватает доступов к админке сайта. Перед выдачей доступов создаем пользователя с данными support support, установив флажок «Привилегированный».
4. Выдачу данных доступа по FTP нужно согласовать с менеджером проектов или тимлидом.
5. После решения проблемы поблагодарить техподдержку. Удалить созданного пользователя из системы управления.