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

```php
$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:

```php
$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,

```php
$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),
            'Informationsystem_Controller_Show'
        ) !== false
    )
{
       $Informationsystem_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.

Практические советы и рекомендации

Создание нового сайта

Первым делом нужно сконфигурировать систему управления. Для этого:

1. Добавляем новый сайт
2. Переносим группу с пользователем в новый сайт запросом update user_groups set site_id = 2 where id = 3
3. Переносим статичные документы с ошибками запросами:
    1. update documents set site_id=2 where id IN (2,4,5,6)
    2. update document_dirs set site_id=2 where id=1
4. Создаем группу констант с названием "Конфигурация"
5. Перемещаем все константы в эту группу запросом update constants set constant_dir_id=1
6. Изменяем качество jpeg на максимум(100), константа JPG_QUALITY
7. Переходим на созданный сайт и удаляем демо сайт
8. Отключаем модули "Типограф" и "Услуги раскрутки сайта"
9. Запрещаем индексацию сайта. Для этого в настройках сайта переходим к вкладке robots.txt и удаляем «admin», должно остаться только «disallow: /»
10. Перейти к модулю «Сайты» и добавить домен для текущего сайта.
11. Переходим к модулю "XSL - шаблоны" и создаем группу с названием домена сайта. Устанавливаем её порядок сортировки "100", чтобы она попала в самый верх.
12. Добавляем макет: “Основной макет сайта”, в нем еще 2 макета: “Основной” и “Для главной”.
13. Создаем меню сайта. Меню создается в модуле “Структура”. Добавляем меню с названием “Основное” и “Нет” – для разделов, которые не должны попасть в меню, но должны попасть в карту сайта.
14. В структуру добавляем:
    1. Главную страницу, документ “Главная страница”.
    2. Поиск, если модуль поддерживается редакцией, ТДС, макет “Основной”
    3. Google sitemap с путем "sitemap", ТДС, макет “Основной”.
    4. Карту сайта с путем "map", ТДС, макет “Основной”.
    5. Страницу ошибки 404, документ “Ошибка 404”. Прописать страницу ошибки можно в настройках сайта(модуль "Сайты").
    6. Страницу “Сайт отключен”, документ “Сайт отключен”. Прописываем её также в настройках сайта.
15. Очистить корзину.

Затем нужно создать элементы структуры. Их берем из меню сайта, которое можно увидеть в верстке.

1. Элементы структуры, которые еще не задействованы необходимо именовать с префиксом "~"
2. Раздел интернет-магазина всегда должен иметь путь "shop" а корзина "shop/cart"

Все XSL шаблоны будут именоваться с префиксом, состоящик из 2 ключевых символов домена или названия сайта. Например, для tritan.ru это "tt"; для hasttings.ru это "ht".

Интеграция верстки

Интеграция – процесс внедрения верстки в движок сайта. После интеграции элементы сайта станут динамическими и будут управляться из админки. Важно правильно разметить верстку для интеграции – не все элементы сайта должны управляться. Например, копирайты никто и никогда не редактирует – они останутся в коде макета. Первым делом необходимо создать макеты сайта. Макеты в HostCMS могут быть бесконечно вложенными. Для любого сайта будет существовать 2 макета – «Основной макет» и «Главная». Перед переходом к дальнейшим

действиям необходимо скопировать все папки верстки с картинками, стилями и скриптами в папку сайта, а также файлы “kad.php” и “template.php” из библиотеки Kad в папку modules/kad. Далее:

1. Переходим к модулю «Макеты сайта»
2. Создаем макет с названием «Основной макет»
3. Помещаем в код макета код главной страницы верстки сайта. Предварительно во вставляемом коде нужно скорректировать пути к скриптам, стилям и картинкам.
4. Шапку приводим к такому виду, как в руководстве по интеграции HostCMS, но скрипты и стили макета должны остаться подключены. Строку с вызовом метода showCss необходимо подключить после всех стилей верстки. Они должны быть перекрывающими. В дальнейшем при недочетах стилей верстки мы будем вносить перекрывающие стили в поле во вкладке «CSS» при редактировании макета.
5. Следующим шагом будет вывод меню сайта в шапке и футере. Это делается с помощью функции `Kad::showMenu([XSL-шаблон], [id_меню])`.
6. Далее необходимо найти в верстке статичные элементы макета, которые редко, но могут изменяться пользователем. Это могут быть телефоны в шапке и футере или же рекламные баннеры, соц. иконки и т.д. Многострочные блоки и баннеры необходимо вынести в статичные документы(Модуль «Страницы и документы»). Для этого создаем документ, в который помещаем код блока. Сохраняем, запоминаем id. В макете в том месте, где мы этот блок кода вырезали, помещаем код `Kad::showDocument([id_документа])`.
7. Статичные блоки, указанные в п.6, но представленные в макете всего одной строкой без тегов, выносятся в константы. Это может быть адрес или телефоны. Если телефон имеет сложное оформление, необходимо использовать функции `Kad::getPhoneBody` и `Kad::getPhoneCode`.
8. Слайдеры, новости, статьи или другие информационные системы в макете выводятся с помощью функции Kad::showInfosystem
9. Группы каталога товаров или товары выводятся с помощью функции `Kad::showCatalog`
10. Если требуется вывод новинок или распродаж, то следует использовать `Kad::showFilteredCatalog`
11. После этого на основе других страниц сайта нам необходимо выделить внутреннюю часть контента. В «Основном макете» должен остаться только тот код, который един для всех страниц сайта.
12. Создаем вложенный в «Основной макет» макет и называем «Главная»
13. Вложенную часть из «Основного макета» вырезаем и помещаем в только что созданный макет.
14. В «Основной макет» на месте вырезанного кода вставляем `<?php Kad::execute(); ?>` - это функция, которая подключит вложенный макет.

Разработка и внедрение наблюдателей

Наблюдатели(они же хуки) позволяют вносить дополнительную логику при срабатывании стандартных функций и методов ядра HostCMS при этом не изменяя исходные файлы. (…)

Создание многоязычных сайтов

Структура. В структуре создаем разделы с названиями языков, например, ru, en, az. Внутри каждого раздела создаем элементы структуры в соответствии с пунктами меню.

Информационные системы и товары. В каждой информационной системе создаем разделы с названиями языков, как в структуре. В каждом разделе будут созданы элементы для своего языка. ТДС для инфосистем и товаров необходимо скорректировать так, чтобы она определяла язык который необходимо вывести. Страницы и документы. Страницы и документы создаются в разделах с названием языков. В качестве идентификатора для выбора конкретного документа используем поле “Комментарий”. Макеты. В макетах для меню и страниц создаются специальные условия для определения языка и переключения нужного документа в зависимости от языка.