Мир изменился. Я чувствую это в воде,

я чувствую это в земле, ощущаю в воздухе…

*Джон Р. Толкиен «Властелин Колец».*

**ИНСТРУКЦИЯ ПО РАБОТЕ С CMS.**

**ОБЩЕЕ ОПИСАНИЕ ОСНОВНОГО ФУНКЦИОНАЛА CMS**

CMS создан с использованием языка PHP и поддерживает работу с базами данных MySQL и Amazon

**Описание встроенных механизмов CMS**

**Прямая адресация страниц**

Предусмотрено создание страниц с прямой адресацией (e.g. [www.site.ru/Shop/mp3players](http://www.site.ru/Shop/mp3players)) в том числе с использованием кириллицы и других языков, поддерживаемых UTF-8 и заключающих в себе как HTML/Javascript контент, так и введённый администратором сайта программный код на языке php.

Варианты адресации:

1. Механизм загрузки контента, жёстко привязанного к текущему url

например, по двум ссылкам [www.site.ru/О\_нас/Контакты](http://www.site.ru/О_нас/Контакты) и [www.site.ru/О\_нас/Схема\_проезда](http://www.site.ru/О_нас/Схема_проезда) будут загружены два независимых друг от друга варианта содержимого или два разных php скрипта

1. Механизм загрузки контента с дополнительной обработкой данных в зависимости текущего url .

В этом случае какой-то адрес используется как статический раздел с изменяемым окончанием url, например:

[www.site.ru/wiki/Генерал\_Корнилов](http://www.site.ru/wiki/Генерал_Корнилов) в данном примере CMS обратится к разделу **/wiki**, и загрузит контент из таблицы, имеющий метку-заголовок «Генерал Корнилов»; при переходе же на страницу [www.site.ru/wiki/Теория\_Ховинда](http://www.site.ru/wiki/Теория_Ховинда) CMS обратится к **тому же разделу /wiki** и загрузит контент, имеющий метку-заголовок «Теория Ховинда»

1. Механизм загрузки контента, с поиском в таблицах, указанных в настройках. В этом случае контент зависит от различных частей url. Возможно задействование собственных SQL запросов.

Пример:

По адресу [www.site.ru/Panasonic/TX-L19X5](http://www.site.ru/Panasonic/TX-L19X5) скрипт загрузит данные из таблицы appliance, а по адресу [www.site.ru/Менеджеры/Комаров\_Геннадий](http://www.site.ru/Менеджеры/Комаров_Геннадий) - данные из таблицы personal. Количество настраиваемых вариантов загрузки не ограничено.

**Обработка полей HTML форм (GET/POST)**

CMS содержит следующие таблицы для управления данными из форм:

* таблицу со списком форм
* таблицу со списком валидационных и вспомогательных действий
* таблицу со списком акций, которые должны быть применены к данным в формах.

Валидационные действия прикрепляются к определённой форме, в записи о валидации указываются имена переменных, тип валидации, имя таблицы для проверки и сообщение об ошибке, выдаваемое пользователю.

Предусмотрены следующие типы валидации GET/POST переменных:

* Проверка на наличие /отсутствие переменной с определённым именем/индексом
* Проверка на равенство или отличие пары переменных
* Проверка на наличие или отсутствие переменной с данным значением в указанной таблице
* Проверка длинны значения указанной переменной (не короче/не длиннее определённого числа знаков)
* Проверка валидности написания email адреса
* Проверка значения на наличие только: латинских букв, цифр, символов тире и подчёркивания
* Проверка значения на соответствие указанному регулярному выражению

Предусмотрены следующие типы вспомогательных действий (содержатся в той же таблице, что и валидаторы):

* Стандартные процедуры login и logout
* Сохраниение HTML тегов (по умолчанию удаляются с помощью strip\_tags)
* Зашифровка MD5
* Замена кавычек на спецсимволы
* Преобразование HTML в спецсимволы

В случае несоответствия данных одному или нескольким валидаторам, будет произведён возврат пользователя обратно к заполнению формы с автоматическим возвратом данных в форму (поля формы будут заполнены) и выдачей пользователю списка сообщений об ошибке.

В случае отсутствия сообщений об ошибке, будут загружен список действий, прикреплённых к данной форме.

Действия существуют следующих типов:

* Добавление данных в указанную таблицу
* Удаление данных из указанной таблицы
* Изменение данных в указанной таблице.
* Выполнение php скрипта.

Таблица может быть в БД MySQL илбо в роли таблицы может выступать домен Amazon

**Прочие встроенные механизмы:**

**Механизм загрузки данных из таблиц в массив.** Загрузка осуществляется одной строкой программного кода, не считая кода указания параметров. Возможные параметры (не обязательные):

* Тип источника MySQL или Amazon (по умолчанию – MySQL)
* Список полей (по умолчанию – все)
* Цифровые или строковые индексы массива (по умолчанию - строковые)
* Фильтр
* Сортировка
* Группировка
* Связь с другой таблицей (JOIN)
* Диапазон загрузки (загружаются записи от n-ой до k-ой)
* SQL запрос

**Механизм редиректа** (в таблице содержится список «старых» и «новых» адресов. В случае, если текущий адрес страницы соответствует какой-то из записей таблицы по полю «старый» адрес, будет произведён 301-ый редирект на «новый» адрес.

**Механизм загрузки php ini set**. Таблицы содержит формируемый администратором сайта список директив php-ini-set, применяемый при каждой загрузке страницы.

**Механизм формирования xml из массива.** (например, для работы с Маркет.Яндексом)

**Механизм автоматического формирования контента** по метками в HTML коде. Предусмотрен следующий формат меток для контента:

1. [>marker<] – такая метка будет заменена контентом с меткой «marker».
2. [{label}] – такая метка будет заменена url-ом раздела, имеющего метку label.

О том как пользоваться этим механизмом – позднее.

**Инструкция по запуску CMS**

Дря запуска CMS нужно распаковать архив в папку, указанную в настройках вебсервера как docroot. Для настройки соединения с БД MySQL нужно отредактировать файл /mods/mod\_defines.php; в строках

define("my\_host", "адрес mysql сервера");

define("my\_user", "логин");

define("my\_password", "пароль");

define("my\_db", "база данных");

нужно заменить хост, логин, пароль и имя базы данных на свои. Также, если есть акаунт на Amazon, необходимо указать свой key и secret key для соединения с сервером amazon.

define("am\_key", "key");

define("am\_secret", "secret key");

Если соединение с БД произошло успешно, то зайдя на главную страницу, Вы увидите базовый дизайн, назначенный по умолчанию.

Все внутренние настройки CMS производятся в кабинете администратора (на данный момент - 16 апреля 2013 г. Кабинет проработан лишь частично.). Адрес кабинета администратора по умолчанию - /engine, логин: admin, пароль: 123111

Интерфейс кабинета администратора имеет следующие разделы:

**Управление разделами сайта**: (добавление/ удаление разделов, изменение их параметров)

Каждый раздел сайта имеет следующие параметры:

1. Адрес (URL), например, «О компании». При назначении кириллического адреса, его можно указывать в незакодированном виде (перекодировка автоматизирована), Также не нужно заменять пробелы символами подчёркивания, это также сделает за Вас CMS.

Примечание: Адрес может быть составным (например «О нас/Контакты»).

1. Уникальную метку (например, «about»), которая является идентификатором раздела.
2. Php файл, прикреплённый к данному разделу (необязательный параметр).
3. Скрипт на языках php, javascript, или просто HTML контент, привязанный к данному разделу (необязательный параметр)
4. Шаблон HTML (выбор из БД)
5. Параметр раширяемого адреса (может или не может в адресе после /about быть указана дополнительная часть адреса, например /about/contacts). В случае, если расширение адреса не указано, по адресу /about/contacts будет выдаваться ошибка 404.

Примечания: GET переменные (/about?var=value) расширением не считаются и ошибки не выдают; Нужно иметь в виду, что могут существовать два независимых раздела с адресами, например /about и /about/contacts. Этого лучше избегать для предотвращения путаницы.

1. Доступность раздела (общедоступен/недоступен/настраиваемый доступ)
2. Индексация раздела (для неиндексируемых будет добавляться метатег NOINDEX)
3. Заголовок h1 (необязательный параметр).
4. Заголовок title (необязательный параметр).
5. Ключевые слова страницы keywords (необязательный параметр)
6. Описание страницыdescription (необязательный параметр)

**Управление скриптами и контентом:** (добавление/удаление/редактирование скриптов php, javascript, а также просто HTML контента)

CMS содержит встроенный редактор кода на базе Edit Area. В редакторе можно обрабатывать новые либо уже существующие в базе данных скрипты.

**Управление формами.**

Идентификаторы форм содержатся в таблице sys\_forms.

**Поле nam** содержит имя формы**.**

**Форма должна содержать элемент с таким же именем, передаваемый через get или post.** Это может быть кнопка submit или просто элемент типа hidden. При использовании элемента input type="image" в качестве кнопки отправки данных формы следует помнить 2 вещи:

1. Элемент input type="image" с именем **name не создаёт get/post переменной с именем name**. Он создаёт две get/post переменных – **name\_x** и **name\_y**.
2. Данный из такой формы можно отправить без нажатия на кнопку-картинку input type="image". Этом можно сделать нажав enter в текстовом моле формы. В этом случае переменная get/post с именем кнопки картинки не будет отправлена вообще.

Отсутствие get/post переменной с именем равным записи в поле nam таблицы sys\_forms приведёт к тому, что данные из формы не будут обработаны.

**Поле nam** содержит имя формы,

**Поле val** содержит метку раздела, в котором должна быть заполнена форма. Данные будут приниматься к обработке только при условии, что пришли с адреса, соответствующего данной метке. Если поле содержит ноль, то данные принимаются из любого раздела.

**Поле forms** содержит сам html код формы. Поле можно оставить пустым и формировать html код формы динамически.

**Поле comeback** содержит метку раздела, куда следует вернуться после успешной обработки формы. Если полке пусто, возврат осуществляется назад к адресу, содержащему форму.

Валидационные действия содержатся в таблице sys\_datacheck

**Поле formname** содержит имя формы.

**Поле label** содержит метку раздела.

**Поле type1** содержит тип первой переменной: GET/POST (или пусто – если это просто переменная)

**Поле var** содержит имя первой переменной.

**Поле type2** содержит тип первой переменной.

**Поле param** содержит имя второй переменной.

**Поле tab** содержит название таблицы с данными для проверки

**Поле errmsg** содержит текст сообщения об ошибке при отрицательном результате ошибки.

**Поле kind** – содержит тип проверки или вспомогательного действия.

|  |  |
| --- | --- |
| **Содержимое поля** | **Тип действия** |
| 1 | Проверка значения переменной на наличие значения |
| 2 | Проверка равенства двух переменных |
| 3 | Проверка отличия двух переменных |
| 4 | Проверка значения переменной на существование в таблице |
| 5 | Проверка значения переменной на отсутствие в таблице |
| 6 | Проверка длины переменной (не меньше чем) |
| 7 | Проверка длины переменной (не больше чем) |
| 8 | Валидация емайла |
| 9 | Проверка значения переменной на наличие только латинских букв, цифр, тире и знака подчёркивания |
| 10 | Процедура логин |
| 11 | Процедура логаута |
| 12 | Процедура dataput (в разработке) |
| 13 | Предотвращает автоматическое удаление тегов |
| 14 | Шифровка переменной (md5) |
| 15 | Проверка значения переменной на соответствие регулярному выражению |
| 16 | Замена кавычек на спецсимволы |
| 17 | htmlspecialchars |

Действия, которые должны быть совершены с данными после успешной проверки, содержатся в таблице sys\_actions

**Поле formname** содержит имя формы.

**Поле var** содержит имя переменной

**Поле type1** содержит тип переменной

**Поле tab** содержит имя таблицы, с которой связано действие.

**Поле source** содержит тип таблицы (1-Amazon, пусто – MySQL)

**Поле req** содержит SQL запрос, если таковой нужно применить

**Поле phpcode** программный код php, если таковой нужно выполнить

**Поле okmess** содержит сообщение, выдаваемое пользователю после действия

Форма должна направлять данные системному скрипту, обрабатывающему GET и POST переменные (по умолчанию метка post).

Пример использования.

Нам нужно добавить нового пользователя в таблицу sys\_users. Запись должна содержать следующие записи: имя, фамилия, логин, пароль, емайл (обязательные поля) а также отчество и дату рождения (необязательные поля).

В каком-либо разделе сайта размещаем форму. К примеру, пусть это будет раздел с меткой newuser и адресом /новый\_пользователь. Таблица sys\_chpu должна содержать запись с полями label и address равными «newuser» и «новый пользователь» соответственно.

Пусть форма у нас имеет следующий html код:

<form action='/post' method='post' name='user'>

\* Фамилия <input name='surname' type='text' value='' size='32' maxlength='32'>

\* Имя <input name='firstname' type='text' value='' size='32' maxlength='32'>

Отчество <input name='middlename' type='text' value='' size='32' maxlength='32'> <br />

Дата рождения <input name='birthday' type='text' value='' size='10' maxlength='10' id='datepicker'>

\* Email <input name='email' type='text' value='' size='32' maxlength='64'>

\* Логин <input name='login' type='text' value='' size='16' maxlength='16'>

\* Пароль <input name='pass' type='text' value='' size='16' maxlength='16'>

\* Подтвердите пароль <input name='pass2' type='text' value='' size='16' maxlength='16'>

<input name='**adduser**' type='submit' value='Готово' /></form>

Форма может содержаться в статическом HTML коде страницы, может быть сгенерирована программно или загружена из таблицы sys\_forms – это не важно. Но она обязательно должна передавать идентификатор. В нашем случае это имя кнопки отправки формы – «**adduser**».

**Имена элементов, которые мы будем заносить в базу должны соотвествовать названиям полей таблицы, с которой данные будут взаимодействовать.** Т.е. если поле, содержащее фамилию пользователя в таблице sys\_users, называется surname, то текстовое поле формы, куда будет введена фамилия, должно называться также.

В таблицу sys\_forms добавляем следующие данные.

В поле val добавляем метку раздела, где размещена форма – newuser

В поле nam добавляем идентификатор формы – adduser

В поле forms, если нам это нужно, добавляем сам HTML код формы

Далее вводим в таблицу sys\_datacheck данные для проверки заполнения формы

Вводим следующие записи в таблицу sys\_datacheck:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **formname** | **label** | **type1** | **var** | **kind** | **type2** | **param** | **tab** | **errmsg** |  |
| adduser | newuser | POST | surname | 1 |  |  |  | Не указана фамилия | Проверка заполнено ли поле surname |
| adduser | newuser | POST | firstname | 1 |  |  |  | Не указано имя | Проверка заполнено ли поле firstname |
| adduser | newuser | POST | login | 1 |  |  |  | Не указан логин | Проверка заполнено ли поле login |
| adduser | newuser | POST | pass | 1 |  |  |  | Не указан пароль | Проверка заполнено ли поле pass |
| adduser | newuser | POST | pass | 2 | POST | Pass2 |  | Пароли не совпадают | Проверка идентичности двух паролей |
| adduser | newuser | POST | email | 8 |  |  |  | Указан некорректный емайл | Валидация емайла |
| adduser | newuser | POST | login | 6 |  | 4 |  | Логин не должен быть короче 4х символов | Проверка длины login |
| adduser | newuser | POST | login | 7 |  | 15 |  | Логин не должен быть длиннее 15ти символов | Проверка длины login |
| adduser | newuser | POST | login | 9 |  |  |  | Логин может содержать только латинские буквы, цифры, тире и нижнее подч. | Проверка login на лат. Буквы/цифры |
| adduser | newuser | POST | login | 5 |  |  | sys\_users | Такой логин уже есть | Проверка логина на наличие в базе |
| adduser | newuser | POST | email | 5 |  |  | sys\_users | Такой емайл уже есть | Проверка майла на наличие в базе |

После отправки формы будет проведена проверка пришли ли данные с адреса, имеющего метку newuser и есть ли среди POST переменных метка данной формы – adduser.

Затем будет проведена проверка по всем указанным в таблице параметрам, и если какие-то из условий не будут соблюдены, пользователь будет возвращён в форму и ему будут показано сообщение, состоящее из объединённых сообщений об ошибке, соответствующих условиям проверки, выявившим невалидные данные.

Если все условия, указанные в таблице, соблюдены, данные пройдут обработку действиями, указанными в таблице sys\_actions.

Вводим следующую запись в таблицу sys\_actions:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **formname** | **var** | **act** | **type1** | **tab** | **source** | **req** | **phpcode** | **okmess** |
| adduser |  | add | POST | sys\_users |  |  |  | Пользователь добавлен. |

Эта запись сообщает, механизму обработки форм, что нужно добавить все POST переменные, полученные из формы adduser, **чьи индексы совпадают с именами полей таблицы sys\_users** в эту же таблицу (sys\_users), после чего на экран будет выведено сообщение «Пользователь добавлен».

Использование методов базового класса

На данный момент – (16 апреля 2013 г.) большая часть методов объединена в одни класс ($my) .

Каждой таблице, независимо от того, есть ли у неё ключевые поля, назначается «заглавное» поле, которое будет идентификатором строк загружаемой таблицы. Записи о том, какок поле каждой таблицы назначено индентификационным, содержатся в отдельной таблице sys\_keyfields.

|  |  |
| --- | --- |
| **tab** | **keyfield** |
| sys\_users | login |
| sys\_forms | nam |
| goods | partnumber |

И т.д.

Из этих записей следует, что идентификатором записей в таблице sys\_users является поле login, а в таблице goods – поле partnumber.

Синтаксис загрузки таблиц

Таблица загружается функцией fetch. Синтаксис:

$my->fetch(‘имя таблицы’);

Предположим, у нас есть таблица goods со следующими записями:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **partnumber** | **name** | **vendor** | **price** | **stock** |
| TX-P(R)42ST50 | Телевизор Panasonic TX-P(R)42ST50 | Panasonic | 22560 | 55 |
| TX-P(R)50XT50 | Телевизор Panasonic TX-P(R)50XT50 | Panasonic | 33500 | 70 |
| 32CS560 | Телевизор LG 32CS560 | LG | 18400 | 112 |
| 42CS560 | Телевизор LG 42CS560 | LG | 23000 | 75 |
| KDL-40EX653 | Телевизор Sony KDL-40EX653 | Sony | 24890 | 32 |
| KDL-22EX553 | Телевизор Sony KDL-22EX553 | Sony | 14500 | 97 |

Загружаем её:

$my->fetch(‘goods’);

После этого в классе появится свойство (массив) $my->tabdata[‘goods’].

Элементы этого массива будут содержать следующие значения (указано выборочно):

$my->tabdata[‘goods’] [‘32CS560’][‘name’] => ‘Телевизор LG 32CS560’

$my->tabdata[‘goods’] [‘TX-P(R)42ST50’][‘price’] => 22560

$my->tabdata[‘goods’] [‘KDL-40EX653’][‘vendor’] => ‘Sony’

$my->tabdata[‘goods’] [‘42CS560’][‘stock’] => ‘75’

Если в качестве идентификатора неудобно использовать значения идентификационного поля, то можно загрузить таблицу в массив с особыми параметрами, в том числе указать, что идентификаторы нужны числовые. Это делается с помощью свойства mod

Синтаксис:

$my->mod[‘index’] = 1; //любое значение, отличное от нуля

$my->fetch(‘goods’);

Теперь элементы массива будут содержать следующие значения (указано выборочно):

$my->tabdata[‘goods’] [0][‘name’] => ‘Телевизор Panasonic TX-P(R)42ST50’

$my->tabdata[‘goods’] [1][‘name’] => ‘Телевизор Panasonic TX-P(R)50XT50’

$my->tabdata[‘goods’] [2][‘name’] => ‘Телевизор LG 32CS560’

$my->tabdata[‘goods’] [0][‘stock’] => 55

$my->tabdata[‘goods’] [5][‘price’] => 14500

Также свойство mod кроме параметра index может иметь следующие параметры:

‘fields’ - список полей таблицы через запятую

‘filter’ – например, $my->mod[‘filter’] = ‘price<25000’;

‘order’ – указывается поле для сортировки

‘group’ – указывается поле для группировки

‘join’ – задаётся связь между таблицами

‘range’ – Диапазон загрузки. Пример: $my->mod[‘range’] = ‘3:8’; (загружаются записи от 3-ей до 8-ой)

Кроме этого, есть ещё 2 свойства, важных для загрузки таблиц:

1. $my->req – может содержать SQL запрос, тогда в свойство tabdata загружаются результаты его выполнения, независимо от параметров mod
2. $my->source – источник данных. (0 – MySQL; 1 – Amazon;)

Пример:

$my->mod=array(

‘filter’=>’vendor=”LG”’,

‘order’=>’price DESC’);

$my->fetch(‘goods’);

Результат будет эквивалентен выполнению запроса:

SELECT \* FROM ‘goods’ WHERE ’vendor=”LG”ORDER BY price DESC