Vedro System :: Документация

версия 0.9.7

От автора

Лучше всего читать эту документацию в редакторе gedit с темой Oblivion $)
Vedro System и данная документация используют кодировку UTF-8.
На моём сервере уже установлена Vedro System. К ней, в ознакомительном режиме, можно обратиться по следующему адресу *http://91.198.212.70:81/* Логин/пароль для доступа к сайту — root/pass. Настройка DEBUG в конфигурационном файле системы — включена. В ознакомительном режиме установлены все прилагающиеся по умолчанию модули, а также ещё один модуль, работающий с достаточно большой базой данных — статистике игрового сервера. Прошу не обращать внимание на иногда всплывающие "крякозяблы" — это связанно с тем, что ознакомительная версия установлена на ОС FreeBSD, имеющей некоторые проблемы с кодировкой UTF-8.

Системой Vedro System заинтересовалась одна компания, занимающаяся услугами предоставления доступа в Интернет. И сейчас Vedro System активно используется этой компанией. Посмотреть на то, что именно предоставляет Vedro System можно на скриншотах представленных в директории */docs/screenshots*.
По последним данным "Ведром" заинтересовалось ещё несколько человек. От которых было много положительных отзывов.)

Прошу не критиковать меня за язык изложения. Старался как мог.)

О программе

Vedro System - Web CMS с удобными компонентами для работы с БД.

Vedro System была разработана на операционной системе Ubuntu 8.04 Hardy Heron — Linux.
Во время разработки просматривалась в браузере Mozilla Firefox 3.0.
Написана на скриптовом языке программирования PHP 5.2.4.
В качестве Web-сервера использовался Apache 2.2.8.
В качестве сервера БД использовался MySQL 5.0.51a.

Тестирование производилось на FreeBSD 6.1 / Apache 2.2.2 / PHP 5.2.1 / MySQL 4.1.21, а также на FreeBSD 5.3 / Apache 1.3.33 / PHP 5.2.6 / MySQL 4.1.10 (в последнем случае, машина используется для отображения ознакомительной версии Vedro System, ссылка на которую приведена выше).
Тестирование в браузерах IE 6, Opera 9 прошло успешно.

При написании системы не использовались никакие сторонние PHP / JavaScript скрипты. Всё было написано мной (за исключением JavaScript подсвечивания строк, о чём упоминается в исходниках).

Чтобы установить Vedro System, ознакомьтесь с приложенным файлом */docs/install.txt*.

Общее описание работы системы

Система работает через "ненайденные" документы. Т.е. формируются ссылки на несуществующие страницы. Эти ссылки перехватываются скриптом `/404.php`. Скрипт `404.php` распознаёт к какому модулю хочет обратиться пользователь и через системный класс `/classes/page.php` формирует и выводит страницу.

Vedro System — модульная система. Она не работает без установленных в неё модулей. Все модули устанавливаются через командную строку, скриптом `/mod`. Для получения справки по возможностям скрипта наберите в командной строке "`./mod -h`".
При установке модуля необходимо указать путь, до директории, в которой этот модуль находится. В данной директории должен "лежать" конфигурационный файл — `config.xml`. В качестве пути до модуля можно (не проверено!) использовать URL адреса.

Структура `config.xml`:

```
<?xml version="1.0" encoding="utf-8">
<data>
    <name>module_name</name>
    <kernel>core.php</kernel>
    <menu name="название в меню" full_name="полное название"
```

```
script_name="script0.php">
            <item name="Элемент1" full_name="Полное название элемента">
                <item  name="Элемент11" full_name="Полное название элемента"
script_name="script11.php" pos="100" />
                <item  name="Элемент12" script_name="script12.php" pos="200" />
            </item>
            <item  name="Элемент2" script_name="script2.php" />
            <item  name="Элемент3" script_name="script3.php">
                <item  name="Элемент31" script_name="script31.php" />
                <item  name="Элемент32" full_name="Название элемента"
script_name="script32.php" />
                <item  name="Элемент34" script_name="script34.php" pos="400"/>
                <item  name="Элемент33" script_name="script33.php" pos="300" />
            </item>
          </menu>
          <depend>
            <module name="module1" />
            <module name="parent_module" />
          </depend>
      </data>
```

Описание элементов `config.xml`:

- `name` — системное название модуля
- `kernel` — (можно не указывать) скрипт ядра модуля, который будут подключать зависимые модули (непроверенная возможность!)
- `menu` — (можно не указывать, если есть `kernel`) описывает меню подключаемого модуля
  - `name` — название главного элемента меню для данного модуля
  - `full_name` — (можно не указывать) полное название отображаемое пользователю, при выборе данного элемента меню
  - `script_name` — (можно не указывать, но обязательно должен быть у какого-нибудь конечного элемента меню) скрипт, который работает по выбору элемента меню. В качестве этого параметра можно указывать URL другого хоста, по 80 порту (при указании URL не работает преобразователь ссылок - об этом указывает предупреждающее сообщение. Либо же в качестве этого параметра можно использовать .xml файл определённого вида (шаблона)):

```
<?xml version="1.0" encoding="utf-8"?>
<data>
      <template>triple-bottom</template>
      <zone id="1">
            <script src="users_connections_list.php" />
            <html>
                  <br />
```

```
                  <br />
            </html>
      </zone>
      <zone id="3">
            <script src="user_list.php" />
            <!--html><h3>zone3_1</h3></html-->
      </zone>
      <zone id="2">
            <script src="group_list.php" />
            <!--html><h3>zone2_1</h3></html-->
      </zone>
      <zone id="2">
            <html><h3>zone2_2</h3></html>
      </zone>
      <zone id="3">
            <!--html><h3>zone3_2</h3></html-->
      </zone>
</data>
```

Здесь,

- `template` — указывает на название файла, хранящегося в директории `/templates`; этот файл описывает то, как будут отображаться нижеописанные зоны
- `zone` — описывает что будет отображаться в блоке - одной из зон
- `script` — ссылается на один из скриптов модуля; также может ссылаться на подобный .xml файл или URL
- `html` — блок содержащий любой html код

Как и подгружаемые скрипты, так и html код могут содержать любой отображаемый пользователю конечный html (включая <head>, <body> блоки), который в последствии переформировывается классом `/classes/page.php` и приводится в "потребный" вид

Пример подобного .xml файла можно найти в */modules/system_admin/info.xml*

  - `item` (может быть сколько угодно, в том числе и вложенных друг в друга) - элемент меню
    - `name` - название элемента меню
    - `full_name` - (можно не указывать) полное название отображаемое пользователю, при выборе данного элемента меню
    - `script_name` - (можно не указывать, но обязательно должен быть у какого-нибудь конечного элемента) скрипт, который работает по выбору элемента меню; см. выше
    - `pos` - (можно не указывать) положение относительно других элементов меню, у которых указан pos (чем больше значение, тем правее будет элемент меню)
- `depend` - блок зависимости от других модулей (непроверенная возможность!)

- module
  - name - системное имя модуля, от которого должен зависеть текущий (у этого модуля (от которого зависит) должен быть описан kernel)

Пример конфигурационного файла модуля можно посмотреть в */modules/demo/config.xml*.
Если вы изменяете конфигурационный файл, то необходимо переустановить модуль командой

```
./mod -e путь_до/переустанавливаемого_модуля/
```

при переустановке, информация с которой работает модуль не потеряется.

Возможность описания модулей, от которых зависит устанавливаемый модуль — не проверена!

Все отображаемые пользователю страницы (за исключением /login.php) формируются системным классом /classes/page.php. Этот класс производит нужную разметку страницы на основе файлов-шаблонов, хранящихся в директории /templates; настройку переменных окружения; подключает необходимые модули; а также заменяет все пути и ссылки, отображаемые пользователю выбранным модулем, на "валидные", для работы системы. Другими словами этот класс показывает страницы генерируемые модулем, выводя при этом соответствующее меню.
Когда пользователь входит в систему — создаётся сессия, куда сохраняются параметры отображения страницы. По выходу из системы сессия сохраняется до следующего входа пользователя (директория /users).
Класс /classes/page.php использует системный класс /classes/menu.php, для отображения меню. Этот класс кеширует отображаемое пользователю меню, сохраняя кеш в директорию /cache. Кеш изменяется, когда пользователь выбирает другой пункт меню, и отчищается, спустя то количество часов, которое указанно в конфигурационном файле системы.

Список переменных окружения:

$db — соединение с основной БД;

$sys_user — пользователь вошедший в систему;

$path — путь до модуля;

$inc_path — путь до модуля, который подставляется в директиву include/require(_once);

$url_path — изменённый (псевдо-) путь до модуля используемый в "рисовании" ссылок (если не использовать данную переменную окружения, ссылки и пути автоматически заменяются классом /classes/page.php).

Чтобы в модулях/функциях модулей использовать переменные окружения необходимо использовать строку

```
include('global.php');
```

Файл `global.php` содержит объявление вышеописанных переменных окружения, как глобальных.

Скрипты на которые ссылается конфигурационный `config.xml` могут использовать предоставляемую системой коллекцию классов, для работы с различными базами данных и пр. Также в скриптах доступен список переменных окружения, предоставляемых классом `/classes/page.php`.
Ниже следует описание коллекции системных классов.

Описание системных классов

Этот раздел содержит информацию о некоторых системных классах. Кратко описывается назначение класса, а также его методы, с помощью которых производится дополнительная настройка интерфейса пользователя.

`classes/table.php` — компонента отображения таблицы, по SQL запросу, содержащей различные элементы управления. Элементы управления представляют собой:

- элемент поиска — позволяет искать информацию, отображаемую таблицей; поиск производится выбором столбца, по которому требуется выполнить некоторое условие поиска; выбирая различные столбцы, автоматически изменяется возможность выбора типа поиска по столбцу, т.к. столбцы могут содержать различные сущности; поиск можно производить сразу по нескольким столбцам;
- элемент выбора столбцов — позволяет выбрать какие столбцы отображаются таблицей, а какие нет; удобно использовать, когда таблице нужно выводить много информации, по многим столбцам, но отображать все столбцы одновременно — не имеет смысла;
- элемент переключения страниц — позволяет устанавливать лимит, по сколько записей показывает таблица, и соответственно переключаться по страницам, на которые делится таблица, если количество отображаемых записей меньше, общего количества записей.

Описание работы:

Компонента отображает пользователю таблицу с интуитивно понятным интерфейсом просмотра информации хранящейся в базе данных.
Все производимые изменения с интерфейсом таблицы сохраняются в сессии пользователя, и при выходе пользователя из системы сохраняются в файл сессии пользователя.

Объявление: `$table = new Table(&$db_connect, $table_name, $sql_query, $array_of_cols, $bad_lines, $width);`

Описание параметров:

`$db_connect` — ссылка на объект соединения с базой данных (передаётся по ссылке, здесь имеется ввиду ссылка в понятии языка программирования)

`$table_name` — уникальное для системы имя таблицы

`$sql_query` — SELECT запрос в базу данных, по которому рисуется таблица. Может производиться как из одной, так и из нескольких таблиц БД, содержать условия на выборку, количество выборки и сортировку

`$array_of_cols` — (можно не указывать) массив столбцов — в качестве ключей массива указываются либо столбцы получаемые из запроса, либо столбцы какого либо другого типа из коллекции классов (см. пример)

`$bad_lines` — массив описанных определённым образом строк таблицы, которые будут отображаться другим цветом; в качестве ключа используется id столбца, по которому будет производиться сравнение

`$width` — (можно не указывать) ширина таблицы

Описание доступных методов:

`$table->Show();` — нарисовать таблицу

`$table->Show_PageSwitch(bool);` — показывать/не показывать переключатель страниц (по умолчанию `true`)

`$table->Show_SQLCols(bool);` — показывать/не показывать все столбцы из SQL запроса (по умолчанию `false`)

`$table->Show_AllCols(bool);` — показывать переключатель отображаемых столбцов (по умолчанию `false`)

`$table->Show_Bottom(bool);` — показывать ли в нижней части таблицы переключатели страниц и столбцов (по умолчанию `false`)

`$table->Show_AutoBottom(bool);` — автоматическое включение предыдущей опции при количестве строк в таблице больше 50 (по умолчанию `true`)

`$table->Show_Search($col_id);` — включает отображение элемента таблицы, с помощью которого можно производить поиск по информации, отображаемой таблицей; `$col_id` - ключ (текущего) массива столбцов, указывающий на столбец, по которому будет выбран поиск по умолчанию (по умолчанию не отображается)

`$table->SetWidth($width);` — установить ширину таблицы

`$table->SetTheme($theme_pref);` — указать префикс в названиях стилей (CSS), используемых в отображении таблицы

`$table->GetName();` — получить имя таблицы

Пример работы с компонентой можно найти в файле */modules/ipwork/external_ip_list.php*. Или же на ознакомительном сайте *http://91.198.212.70:81/battle/pvpgn/*

---

classes/table_column.php — коллекция столбцов, с которой работает класс /classes/table.php

Классы:

Column - базовый класс в иерархии классов столбцов, обеспечивающий самый обычный вывод информации в ячейки столбца

```
"login" => new Column($name, $visible=true, $align='center', $width=0)
```

здесь,

$name — название столбца, которое будет выводиться пользователю

$visible — отображается ли по умолчанию данный столбец

$align — выравнивание текста в ячейках столбца

$width — ширина столбца

Используемые открытые методы:

```
$columns["login"]->SetDBTableName($db_tablename);
```
— используется, если запрос на выборку выбирает из нескольких таблиц, и может появиться неоднозначность

```
$columns["login"]->SetLink($url, $url_param);
```
— отображает в каждой ячейке столбца ссылку

здесь,

$url — URL который будет открыт по нажатию на ссылку

$url_param — параметр, который будет передан по URL; в качестве параметра указывается id столбца — значение этого столбца в текущей строке будет передано

Column_Number — столбец, предназначенный для отображения числовой информации

```
"id" => new Column_Number($name, $precision=2, $visible=true, $align='center', $width=false)
```

здесь, $precision - число знаков после запятой

Column_IP — столбец, предназначенный для отображения ip адресов, хранящихся в базе данных в виде длинного целого

```
"ip" => new Column_IP($name, $visible=true, $align='center', $width=0)
```

Column_Time — столбец, предназначенный для отображения даты/времени, хранящейся в базе данных в UNIX формате (число секунд от 1 января 1970 года)

```
"date_register" => new Column_Time($name, $visible=true, $align='center', $width=false, $format='d.m.Y H:i:s')
```

здесь, $format — формат отображения даты/времени

Column_Bool — столбец, предназначенный для отображения логической пары (0|1)

```
"is_deleted" => new Column_Bool($name, $yes_str='Да', $no_str='Нет', $visible=true, $align='center', $width=false)
```

Column_Array — столбец, предназначенный для отображения массива, ключи которого в качестве id хранятся в базе

```
"server" => new Column_Array($name, $values, $visible=true, $align='center', $width=false)
```

здесь, $values — тот самый массив

Используемые методы:

`$columns['server']->AsValue();` - установить значение для сравнения (если задаётся $badlines у таблицы), равное значению массива, на которое указывает ключ, берущийся из БД.

Column_List — столбец, предназначенный для отображения значений по id хранящимся в базе, из другой таблицы БД

```
"location" => new Column_List(&$db, $sql, $name, $visible=true, $align='center', $width=false)
```

здесь,

$db — соединение с базой данных

$sql — SELECT запрос в базу данных на формирование массива подставляемых значений

Column_Button — столбец, предназначенный для отображения кнопок, работающих в качестве ссылок

```
"delete" => new Column_Button($name, $url, $url_param, $value, $visible=true, $align='center', $width=false)
```

здесь,

$url — URL который будет открыт по нажатию на кнопку

$url_param — параметр, который будет передан по URL; в качестве параметра указывается id столбца — значение этого столбца в текущей строке будет передано

$value — текст на кнопке

Column_Custom — столбец, предназначенный для отображения какого-либо html, будь то JavaScript или что-нибудь в этом роде)

```
"ping_ajax" => new Column_Custom($name, $html, $visible=true, $align='center', $width=false)
```

здесь, $html — тот самый html, в текст которого можно вставлять значения столбцов текущей строки по шаблону "[[column_id]]"

Column_UserFunc — столбец, предназначенный для отображения результата работы функции

```
"ping" => new Column_UserFunc($name, $func, $args, $visible=true, $align='center', $width=false)
```

здесь,

$func — название пользовательской функции

$args — массив параметров, передающихся в функцию в порядке перечисления

Иерархию классов столбцов можно посмотреть визуально — */docs/schemas/table_conlumn.jpg*

`classes/table_row.php` — класс обеспечивающий выделение строк в таблице красным цветом

Объявление: `"is_deleted" => new Row($compare, $value);`

здесь,

`$compare` — тип сравнения (может быть самым разным: `'='`, `'!='`, `'<>'`, `'>='`, `'gt'`, `'lt'` и т.д.)

`$value` — значение с которым будет сравниваться значение столбца

В качестве ключа массива, параметра `$bad_lines`, следует указывать ключ массива столбцов, по столбцу которого будет производиться сравнение.

`classes/formelementlist.php` — абстрактный базовый класс отображения форм, с помощью которых, пользователю, можно изменять информацию в базе данных. Форма передаёт информацию в саму себя через `$_GET` переменную.

Используемые открытые методы:

`AsString();` — возвращает отображаемую форму, в виде html строки

`SetWidth($width);` — задаёт ширину отображаемой формы

`SetUserFunc($func_name);` — задаёт функцию, которая будет вызываться до обработки формой полученной информации; если задаваемая функция возвращает `false`, то класс формы не обрабатывает передаваемую информацию

`Show();` — отображает форму

Иерархию классов форм можно посмотреть визуально — */docs/schemas/forms.jpg*

`classes/form.php` — компонента для отображения форм, с помощью которых, пользователю, можно изменять информацию в базе данных.

Объявление: `$fr = new Form($elements=array(), $submit_text='', $method='GET', $id='');`
здесь,

`$elements` — массив элементов, отображаемых формой

`$submit_text` — текст отображаемый на кнопке "отправить"

`$id` — id формы, который может понадобиться, для работы JavaScript-а

Используемые открытые методы:

`$fr->AddButton($b_html_str);` — добавляет возле кнопки отправить ещё одну кнопку; кнопку эту необходим передавать в виде текста в формате html

`$fr->SetGoBack(bool);` — если включен этот параметр, страница будет автоматически возвращаться на страницу, с который произошёл вызов данной формы (по умолчанию `true`)

Пример работы с компонентой можно посмотреть в:

/modules/system_admin/user_edit.php

/modules/system_admin/user_add.php

`classes/formchecklist.php` — компонента для отображения флажков указания зависимости, типа "многое ко многим"

Объявление: `$formcheckboxs = new FormCheckList(&$db, $list_sql, $join_sql, $inserted=false);`

здесь,

`$db` — соединение с базой данных

`$list_sql` — SELECT запрос на список сущностей, с которыми необходимо связать "некую другую сущность"

`$join_sql` — SELECT запрос на получение id вышеуказанных сущностей, которые уже принадлежат "некой другой сущности"

`bool $inserted` — указывает на то добавляются ли абсолютно новые записи (`true`), или же изменяются уже существующие (`false`)

Пример работы с компонентой можно посмотреть в:

/modules/system_admin/user_edit.php

/modules/system_admin/user_add.php

`classes/formupdatesql.php` — компонента для отображения формы изменения полей таблицы

Объявление: `$formcheckboxs = new FormUpdateSQL(&$db, $sql, $elements=array());`

здесь,

`$db` — соединение с базой данных

`$sql` — SELECT запрос на выборку полей определённой записи

`$elements` — массив элементов, отображаемых формой

Пример работы с компонентой можно посмотреть в:

/modules/system_admin/user_edit.php

`classes/forminsertsql.php` — компонента для отображения формы заполнения полей таблицы.

Объявление: `$formcheckboxs = new FormInsertSQL(&$db, $sql, $elements=array());`
здесь,

`$db` — соединение с базой данных

`$sql` — INSERT запрос на заполнение полей таблицы базы данных с пустым значением параметра запроса "VALUES()"

`$elements` — массив элементов, отображаемых формой

Пример работы с компонентой можно посмотреть в:

/modules/system_admin/user_add.php

`classes/form_elements.php` — коллекция элементов отображаемых компонентой формы

Классы:

`FBlock` — элемент отображения в форму какого-либо html блока

`"editform" => new FBlock($html)`

Используемые открытые методы:

`AddConstent($content);` — добавляет в блок дополнительную информацию в виде html

`AsString();` — возвращает строку содержащую отображаемый html

`Show();` — отображает элемент

`FBaseElement` — абстрактный базовый класс элементов формы

`Конструктор($name, $value='', $id='');`
здесь,

`$name` — название элемента формы

`$value` — значение элемента

`$id` — id элемента, для дальнейшего возможного использования JavaScript-ом

Используемые открытые методы:

`SetName($name);` — устанавливает название элемента

`GetName();` — возвращает название элемента

`SetValue($value);` — устанавливает значение элемента; при установке значения через этот метод, свойство элемента "изменён" не изменяется

`GetValue();` — возвращает значение элемента

`bool Changed();` — изменён ли элемент?

`AsString();` — возвращает строку содержащую отображаемый html

`Show();` — отображает элемент

`FHidden` — класс описывающий скрытый элемент формы

`FElement` — абстрактный базовый класс, наследуемый от `FBaseElement` и наследуемый ниже описанными элементами

Конструктор(`$name, $desc='', $value='', $id='');`

здесь,

`$desc` — описание элемента формы

`FCheckbox` — класс элемента отображения флажка

`FSelect` — класс элемента выпадающего списка

`"location" => new FSelect(&$db, $sql, $name, $desc='', $value=0, $id='', $user_func='')`

здесь,

`$db` — соединение с базой данных

`$sql` — SELECT запрос на выборку массива из таблицы БД, ключи которого будут являться значением обновляемого поля в изменяемой таблице БД

`$user_func` — пользовательская функция, изменяющая отображаемые пользователю элементы списка, получающая в качестве параметра id элемента списка, полученного из запроса на выборку

`FInput` — класс текстого элемента формы

`FPassword` — класс текстого элемента, в котором все символы заменяются "звёздочками"

`FIP` — класс элемента содержащего IP адрес

Иерархию классов элементов форм можно посмотреть визуально - */docs/schemas/form_elements.jpg*

`classes/sql.php` — класс для работы с SQL запросами. Позволяет некоторым образом проверять синтаксис запроса (запрос разбирается на составляющие, если определённая составляющая не найдена — генерируется исключение, перехватив которое можно перестроить начальный запрос, и т.п.)

Объявление: `$sql = new SelectSQL($sql_query);` или `$sql = new InsertSQL();`
Описание методов:

`$sql->SQL();` — получить исходный или уже перестроенный запрос

`$sql->Table();` — получить название таблицы, из/в которую идёт запрос

`$sql->Columns();` — получить столбцы, по которым идёт запрос

`$sql->FirstTable();` — получить первую таблицу, по которой идёт выборка

`bool $sql->FromOneTable();` — выборка идёт из одной таблицы или из нескольких

`$sql->WhereString();` — получить строку, по условию которой идёт выборка

`$sql->AddWhere($compare, $addstring);` — позволяет расширить условие выборки

здесь,

`$compare` — `( 'and' | 'or' )`

`$addstring` — добавляемое условие

`$sql->OrderArray();` — возвращает массив с двумя элементами

`"column" => 'название столбца по которому сортируется'`

`"type" => ( 'asc' | 'desc' )`

`$sql->OrderString();` — получить строку, по которой идёт сортировка

`$sql->SetOrderString();` - задать строку, по которой будет происходить сортировка

`$sql->LimitArray();` — вовзращает массив с двумя элементами

`"from" => с какой записи начинать выборку`

`"limit" => по сколько записей показывает`

`$sql->LimitString();` — получить строку, по которой идёт выборка

`$sql->SetLimitString();` — задать строку, по которой идёт выборка

`classes/db_mysql.php` — класс работы с базой данных MySQL. На основе этого класса можно написать другой интерфейс, для работы какой-либо другой базой данных (например Oracle).

Объявление: `$db = new DB_MySQL(host_name, user_name, user_password, db_name);`
Описание методов:

`$db->SelectDB(db_name);` — сменить базу

`$db->Query("SQL QUERY");` — выполнить запрос и вернуть результат (аналог mysql_query())

`$db->Query_Fetch("SQL QUERY");` — выполняет запрос и возвращает массив строк (со столбцами) выполненного запроса

`$db->Query_Fetch_Result(query_result);` — возвращает массив строк (со столбцами) по уже выполнному запросу

`$db->Query_Fetch_Row("SQL QUERY");` — выполняет запрос и возвращает массив только одной, самой первой строки; индексация массива начинается с нуля, ключи - цифровые (имеет смысл, во избежание ошибки, в запросе указывать "`LIMIT 0,1`")

`$db->Query_Fetch_Assoc("SQL QUERY");` — выполняет запрос и возвращает массив только

одной, самой первой строки; ключи массива имеют такое же название, как и столбцы в базе (имеет смысл, во избежание ошибки, в запросе указывать "LIMIT 0,1")

`$db->Query_Fetch_Array("SQL QUERY");` — выполняет запрос и возвращает массив только одной, самой первой строки; ключи массива либо цифровые, либо имеют такое же название, как и столбцы в базе (имеет смысл, во избежание ошибки, в запросе указывать "LIMIT 0,1")

`$db->Insert_ID();` — возвращает id последней добавленной строки, запросом "INSERT ..."

`$db->GetDB();` — возвращает название базы данных с которым в данный момент есть соединение

`$db->QueryNums();` — возвращает количество выполненных запросов, с начала инициализации объекта

`classes/system_user.php` - класс системных пользователей

Объявление: `$user = new System_user(&$db_connect, $user_id=0);`
Описание параметров:

`$db_connect` — ссылка на объект соединения с базой данных

`$user_id` — id пользователя в базе данных

Описание методов:

`$user->Current();` — задать объекту параметры вошедшего в систему пользователя

`$user->Reset(user_id);` — сбрасывает значение объекта, присваивая ему новое значение, по id пользователя в базе данных

`$user->IsGroup($group_id);` — возвращает `boolean` значение принадлежит ли пользователь к группе, чей id передан в качестве параметра

Доступные для чтения свойства: `id, login, name, family, daddy, mobile_telephone, groups, is_deleted, full_name, fio, groups_SQL`.

`classes/module_tool.php` — класс управления модулями

Объявление: `$module = new Module_Tool(&$db_connect, $module_id=0);`
Описание параметров:

`$db_connect` — ссылка на объект соединения с базой данных

`$module_id` — id модуля в базе данных

Описание методов:

`$module->Reset($module_id);` — сбрасывает значение объекта, присваивая ему новое

значение, по $module_id модуля в базе данных

$module->InitWithPath(path); — инициализирует модуль по пути, передаваемого в качестве параметра

$module->Install(menu_id); — ($menu_id - необязательный параметр) устанавливает инициализированный модуль, засовывая его в меню, $menu_id которого передано в качестве параметра

$module->Remove(); — удаляет установленный модуль

$module->Reinstall(); — переустанавливает модуль

$module->PrintAllModules(); — выводит на экран список установленных модулей

Debug

Если вы написали модуль, установили в систему, после чего перегружаете страницу, а там ничего нет, просто белая страница, без ошибок, без нечего... то можно заглянуть в лог ошибок PHP (по умолчанию он в логи Apache сбрасывает). А ещё можно закомментировать строку в /classes/page.php — первую в описании метода Show(). Строка — ob_start(...);. После комментирования обновляете страницу, и если в php.ini (конфигурационный файл PHP) настроено, чтобы PHP выводил ошибки в браузер, то Вы увидите ошибки...
Ошибки могут быть очень странные. Могут ссылаться на системные классы, но (!) опыт подсказывает, что скорее всего ошибка в вашем модуле. Может быть неправильно объявляете объект, или неправильно используйте метод. Пожалуйста, ещё раз внимательно проверьте, правильно ли написан Ваш модуль, и если там действительно всё нормально, прошу сообщить об ошибке мне на e-mail <altermn@gmail.com>.

Бывает так, что в своём модуле, вы обращаетесь к той же базе данных (тот же хост, тот же пользователь), и при таком обращении к базе вместо страницы — ´белая пустотаª. Причём если переходить на страничку модуля, с другой странички — всё хорошо отображает, но при перезагрузке страницы — ´опять белая пустотаª. В таком случае, если у вас, всё-таки, используется та же база данных, но уже другая ´databaseª (прошу прощения за каламбур), рекомендую проблему решать так:

```
include('global.php');
$old_db = $db->GetDB();
$db->SelectDB(имя_нужной_database);
...
// работаем с соединением к БД
...
$db->SelectDB($old_db);
```

Пояснение. Подключаем переменные окружения и используем переменную окружения — $db — подключение к БД Vedro System. Запоминаем, на какую database ´смотрелоª это соединение.

Перенаправляем соединение на другую database. А когда работа с нужной database закончится — возвращаемся к ранее используемой.

Послесловие

Впечатления от написания системы:
PHP5 — вполне достойный язык программирования, поддерживающий ООП. В нём, конечно, нет всех плюсов ООП, что даёт например C++, но тем не менее подобные системы пишутся многими PHP программистами. Что говорит о зрелости языка.
Это моя третья по счёту CMS. Разрабатывая её, я постарался избежать тех недочётов и проблем, с которыми столкнулся при написании предыдущих систем.

Недочёты системы:
Неполная поддержка открываемых интернет страниц, на которые ссылается конфигурационный файл модуля.
Отсутствие Web-интерфейса для управления модулями.
Отсутствие интерфейса редактирования меню.
Отсутствие возможности распределения прав пользователей и групп.
Невозможность задать условие выделения строк в таблице, по одному столбцу, чтобы несколько условий было.

Благодарности:
Выражаю огромную благодарность своей жене Аверчук Валентине за моральную поддержку, во время написания системы. Также благодарю Черных Павла, Хован Андрея, Павлова Александра, Чуркина Алексея, Башлакова Ивана, Миназова Максима, Дьячкову Татьяну, Настенко Олега и Шакурова Рината за тестирование системы. Спасибо им за то что выслушивали все технические тонкости, о которых мне приходилось им рассказывать, для лучшего понимания и написания элементов системы.