о фреймворке

PHP-фреймворк PaGen предназначен для создания веб-приложений разного уровня сложности. Его главной особенностью является модель работы с контроллерами, которая позволяет сохранять страницы в базе данных для дальнейшей обработки модератором сайта.

Фреймворк [без подключения к БД] работает достаточно быстро (более чем в 10 раз быстрее по сравнению с Kohana 3) и использует совсем немного памяти [для шаблона по умолчанию используется не более 200 Кб памяти; для сравнения «Hello world!» на Kohana 3 использует 1,4 Мб, а блог на стандартном шаблоне WordPress – 25 Мб].

Версия 0.9 для реализации AJAX использовала библиотеку JsHttpRequest, версия 1.0 полностью адаптирована под jQuery (установлена версия 1.11).

установка

Фреймворк устанавливается с помощью файла install.html, следуя инструкциям и заполняя необходимые поля. Во время инсталляции определяются параметры подключения к базе данных (можно задать использование фреймворка без БД), языки сайта (по умолчанию определяется 3 языка), а также информацию администратора.

параметры конфигурации

Параметры, заданные пользователем во время установки можно изменять в файле pagen\_config.php (все, кроме установленных языков).

Класс config содержит следующие контанты:

PREFIX – префикс для таблиц в базе данных

LANG – язык сайта по умолчанию

TITLE – название сайта, используется в шаблоне

DB\_SERVER – сервер БД

DB\_USER – пользователь БД

DB\_PASS – пароль пользователя БД

DB\_NAME – имя БД

EMAIL – e-mail администратора сайта

ADMIN – логин администратора сайта

PASS – пароль администратора БД, закодированный алгоритмом RandKey

DB – обозначает наличие подключения к БД

TEMPLATE – название каталога, в котором находится шаблон сайта

Кроме того, он содержит 2 статических метода (по умолчанию они закомментированы, вызываются в файле index.php):

checkIP () – проверяет, не содержится ли IP-адрес пользователя в файле system/ban.txt, и, если результат положительный, не загружает сайт для пользователя (IP-адреса вносятся в ручном режиме, с обоих сторон ограничивается вертикальной чертой).

toLog () – содержит логи посещаемости ресурса.

маршрутизация

PaGen, прежде всего, предназначен для работы с базой данных – там задаются настройки страниц сайта.

Если подключение к базе данных отсутствует (задается в файле конфигурации), вызывается контроллер.

Контроллер соответствует заданному URL. Поиск продолжается до тех пор, пока не найден файл контроллера.

К примеру, если URL сайта //mysite.com/pagen, фреймворк будет искать контроллер по адресу /controllers/pagen.php.

Если URL сайта //mysite.com/frameworks/pagen/download, фреймворк будет искать контроллер по адресу /controllers/frameworks.php, если файл не найден – /controllers/frameworks/pagen.php, далее – /controllers/frameworks/pagen/download.php, далее – /controllers/frameworks/pagen/download/index.php.

Если первый файл найден, поиск останавливается, а оставшаяся часть URL передается в качестве аргументов. К примеру, если найден файл /controllers/frameworks.php, контроллеру передается массив [“pagen”, ”download”], если найден файл /controllers/frameworks/pagen/download.php или /controllers/frameworks/pagen/download/index.php, передается пустой массив с аргументами (подробнее в разделе «Контроллер»).

В случае ошибки (если ни один с файлов не найден), возвращается страница 404. Исходя с выше изложенной системы фреймворка, запрещено в одной директории содержать директорию и файл с одинаковыми названиями. Контроллеры должны иметь расширение \*.php.

Если веб-приложение подключается к базе данных, то данные о странице сначала анализируются в соответствии с таблицей `\*pages` в базе данных (\* - это префикс таблиц в БД, если он задан). Если соответствующая запись не найдена, вызывается контроллер по описанной выше схеме. Если она найдена, выполняется ее анализ в соответствии с полем (`cpurl`).

Если поле `static` равно 0, вызывается контроллер в соответствии с полем ‘url`. Если оно пустое, контроллер вызывается в соответствии с выше описанной схемой.

Если поле `static` равно 1, данные сначала загружаются в массив Site::$ThisPage и Site::$Content, а потом вызывается контроллер IndexController (/classes/IndexController.php), который передает указанному в базе отображению (или виду по-умолчанию) данные страницы.

Массив Site::$ThisPage содержит следующие элементы (по ключу):

Id – идентификатор страницы

url – адрес контроллера (может быть пустым)

cpurl – первый элемент URL (фраза перед первым слешем) [название – калька аббревиатуры ЧПУ]

title – заголовок страницы

hor\_menu – позиция страницы в горизонтальном меню

show – отображать ли страницу при генерации горизонтального меню

static – является ли контент страницы статическим

template – какой вид использовать (по-умолчанию – index, может быть пустым)

child – дочерние страницы

meta\_k – мета-тег с ключевыми словами

meta\_d – мета-тег с описанием страницы

Контент страницы содержится в переменной Site::$Content.

Если поле `child` не пустое, то перед получением полных данных страницы, происходит определенный анализ.

Это поле может содержать 3 типа данных: ассоциативные идентификаторы `cpurl` страницы (child\_page – [1]), ассоциативные идентификаторы с преобразованиями (child\_page=banana – [2]), массив ассоциативных идентификаторов (child\_page;table;info150 – [3]), массив ассоциативных идентификаторов с преобразованиями (child\_page=banana;table;info150=monkey – [4]).

Таким образом, если эти параметры заданы у страницы с ассоциативным идентификатором my\_page, то в первом варианте введенный URL /my\_page/child\_page попробует загрузить страницу с `cpurl`, которое равняется child\_page. [1]

Если у child\_page поле дочерних страниц не пустое, тогда выполняются те же действия, пока поле `child` не окажется пустым. Не стоит прописывать в него ассоциативный идентификатор этой же или родительской страницы – в процессе генерации страницы валидность введенной информации не проверяется.

URL /my\_page/banana загрузит страницу с `cpurl`, которое равняется child\_page. [2]

URL /my\_page/child\_page загрузит страницу с `cpurl`, которое равняется child\_page. URL /my\_page/table загрузит страницу с `cpurl`, которое равняется table. URL /my\_page/info150 загрузит страницу с `cpurl`, которое равняется info150. [3]

URL /my\_page/banana загрузит страницу с `cpurl`, которое равняется child\_page. URL /my\_page/table загрузит страницу с `cpurl`, которое равняется table. URL /my\_page/monkey загрузит страницу с `cpurl`, которое равняется info150. [4]

Эта информация может задаваться вручную или с помощью администраторской панели.

Мета теги и контент вызываются в соответствии с полем `id` с таблиц с соответствующим названием (примечание: по-умолчанию, связи в БД отстутствуют). Заголовок страницы определяется в соответствии с полем `titles` таблицы `\*pages`, который является идентификатором записи в таблице `\*titles`.

описания классов

eModel – класс для расширения моделей, представляет собой конструктор запросов

|  |  |  |
| --- | --- | --- |
| № | Метод | Описание |
|  | setDefault() | Устанавливает все поля по-умолчанию. Используется, если дополнительно необходимо выполнить еще один запрос, не создавая новый экземпляр модели. Метод не обнуляет таблицу и буфер объекта. |
|  | bufferQuery($isBuffered = true) | Устанавливает буферизацию запросов. То есть запросы сохраняются в свойстве объекта для последующего выполнения буфера |
|  | clearBuffer() | Очищает буфер запросов |
|  | renderBuffer($transaction = true, $union = false) | Выполняет буферизированные запросы. Первый параметр отвечает за выполнение запросов в трансакции, а второй – за объединение результатов выборки |
|  | setFields(array $myFields) | Устанавливает массив полей для выборки результата. Если для поля нужно задать таблицу, она задается в ключе элемента массива |
|  | addJoin(array $joinRules) | Объединяет таблицы конструкцией INNER JOIN. Например,  $this->addJoin ([[‘users’ => ‘id’, ‘users\_info’ => ‘user\_id’]])  Эквивалентно конструкции  INNER JOIN ‘users\_info’ ON ‘users’.’id’ = ‘users\_info’. ‘user\_id’  Обратите внимание, что используется двухмерный массив, то есть можно использовать более одного объединения. |
|  | addCond($field, $value, $sign = '=', $table = NULL) | Добавляет условие выборки. Первый параметр – поле для условия, второй – его значение, третий – условие (=, >, <, >=, <=), четвертый – привязка поля к таблице. |
|  | setUnion($myUnion) | Тип объединения условий выборки. По-умолчанию стоит AND. |
|  | setData($myData) | Устанавливает массив данных |
|  | getData() | Извлекает массив данных |
|  | setTable($myTable) | Устанавливает таблицу, с которой будет работать конструктор. Можно задать массив таблиц. Если массив ассоциативный, то ключ устанавливается как as.  К примеру, код $thid->setTable([‘a’ => ‘table1’, ‘b’ => ‘table2’]); будет устанавливать FROM `table1` as `a`, `table2` as `b`. |
|  | setOrder($myField, $myType = 0) | Устанавливает порядок сортировки. Первый параметр устанавливает поле, по которому будет производиться сортировка, а второй – тип сортировки (по-умолчанию используются параметры БД, вызов метода без второго параметра устанавливает сортировку с конца). Значение 0 или ‘DESC’ устанавливает сортировку с конца, 1 или ‘ASC’ – с начала. |
|  | setLimits($myFrom, $myLimit = '') | Устанавливает LIMIT OFFSET, ROWS |
|  | setSQL($sql) | Устанавливает SQL-код для WHERE. Не экранируется. |
|  | setResultType($myAssoc) | Если установлено true, результат будет выводиться в ассоциативном массиве (значение по-умолчанию) |
|  | getCount() | Получить количество записей |
|  | create() | Добавляет запись в базу данных |
|  | readById($id) | Извлекает запись с указанным идентификатором в виде одномерного массива. Получить извлеченные данные можно методом getData() |
|  | readFirst($order = 'id') | Извлекает первую запись в виде одномерного массива, сортируя записи по идентификатору. |
|  | readLast($order = 'id') | Извлекает последнюю запись в виде одномерного массива, сортируя записи по идентификатору. |
|  | readBy($field, $value) | Извлекает запись, соответствующую параметру, в виде одномерного массива. |
|  | read($poly = false) | Прочитать запись с указанными параметрами. Аргумент отвечает за тип возвращаемого массива. Если значение аргумента истинно, возвращается двухмерный массив со списком записей, иначе – одномерный. |
|  | update() | Обновляет записи |
|  | deleteById($id) |  |
|  | deleteBy($field, $value) |  |
|  | delete() |  |
|  | search($search = array()) |  |
|  | eSearch($titles = array(), $metas = array(), $content = array()) |  |

пользователи

мультиязычность

контроллер

Пример контроллера:

class **test** extends eController {

public function run () {

//метод run () запускается сразу после создания экземпляра контроллера

$this->loadModel ();

//загружаем стандартную модель

$m = new m\_test ('blog');

//создаем объект модели, мы можем задать имя необходимой таблицы как параметр конструктора

$data = array ();

//инициализируем массив данных для вида

$data = $m->returnTitles ();

//возвращаем с модели заголовки и мета-теги

$m->add ();

//добавляем новую запись – вся логика и детали зашиты в модель – в демке мы этим обойдемся

$m->setDefault ();

//обнуляем поля

$data ['content'] = $m->show ();

//наполняем массив с данными определенным контентом

$data = $this->getLocals ($data);

//получаем переменные $url (адрес сайта), $SiteTitle (название сайта), $SiteLang (язык сайта), $setLangUrl (ссылка на изменение языка)

$this->loadModel ('\_design');

$data['\_d'] = new \_design();

$files = array(

'hor\_menu' => 'blocks'.DIRSEP.'hor\_menu',

'sign\_in\_form' => 'blocks'.DIRSEP.'sign\_in\_form'

);

$this->view = View::factory ($data, $files, $this->word);

}

}

вид

работа с моделью

операция отображения данных

function test\_show () {

$this->setFields(array ('id', 'title', 'text'));

$this->setOrder('id', 'DESC');

$this->readLast ();

return $this->getData ();

}

Данный код возвратит массив значений последней записи

операция добавления данных

операция изменения данных

операция удаления данных

администрирование страниц

randkey