ADFM

Общая информация

ADFM - система управления контентом, написанная специально для разработчиков студии Вебтолк.

Система выполнена в виде пакета Laravel wtolk/adfm .

Основные задачи которые решает система ADFM:

- быстрая разработка типовых сайтов
- Легкая верстка шаблонов
- Простота изучения большинством программистов понимающих PHP, MVC паттерн, и основы ООП.

Установка

~~Текущую версию дистрибутива можно скачать здесь:~~
~~http://adfm.wtolk.ru/adfm.zip~~

Для корректной работы необходима версия PHP >= 7.4
Для простой и быстрой установки, необходимо скачать фаил установщика из репозитория https://github.com/wtolk/adfm_installer/blob/master/dist/adfm_installer.phar и создать конфиг по примеру https://github.com/wtolk/adfm_installer/blob/master/dist/config.yaml

и запустить его командой

```
php adfm_installer.phar
```

Данный установщик скачает свежую версию Laravel в папку {{config:path}}, установит пакеты {{ config:packages }} и задаст конфиг подключения к базе {{ config:db }}

Ручная установка

Устанавливаем свежий ларавел `composer create-project laravel/laravel`

Устанавливаем пакет с Adfm `composer require wtolk/adfm --no-cache`

Вспомогательные пакеты

```
composer require barryvdh/laravel-debugbar
composer require barryvdh/laravel-ide-helper
```

Публикуем файлы пакетов `php artisan vendor:publish`

Применяем миграции `php artisan migrate`

Создаем root пользователя

```
php artisan adfm:user test test@test.ru test
```

Файлы шаблонов

Существуют следующие страницы шаблонов :

Общий шаблон - layout.blade.php

Главная страница - index.blade.php

Обычная страница - page.blade.php

Функции и переменные используемые в шаблонах

Вывести все переменные, передаваемые в шаблон :

```
{{ dd(get_defined_vars()) }}
```

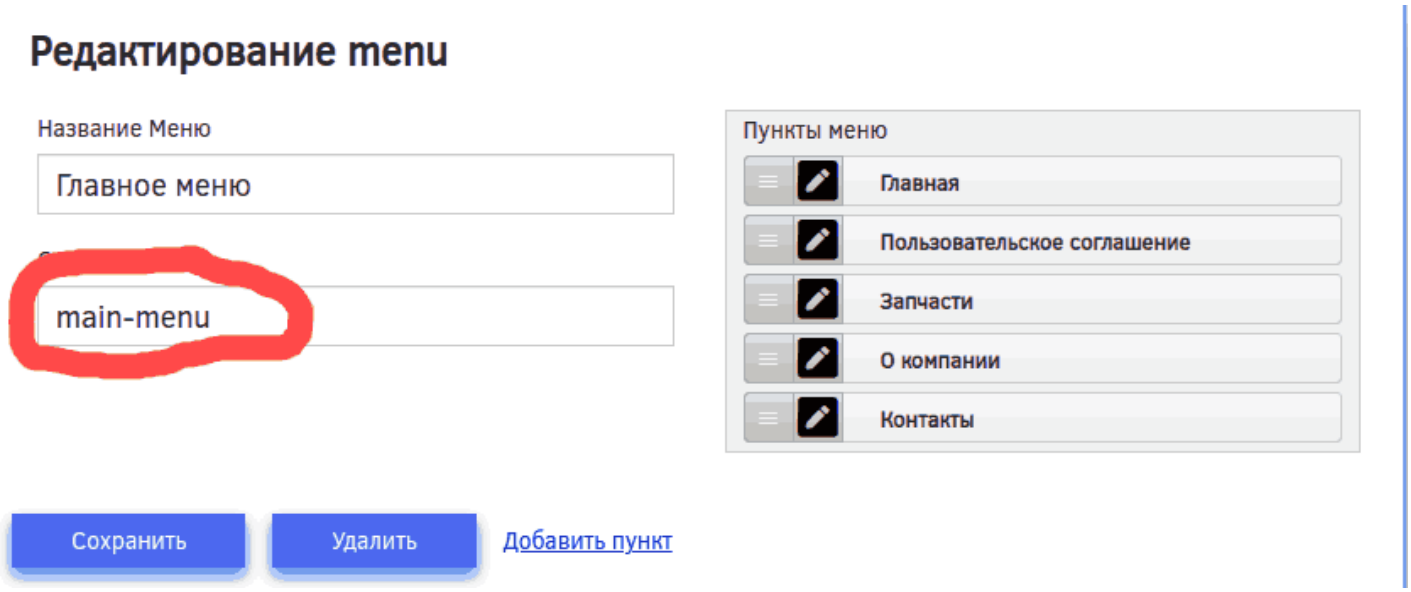
Переменные Page

```
{{$page->title}} - название страницы
{!! $page->content !!} - текст из визуального редактора
{{$page->files}} - прикрепленные файлы через поле MultiUpload

Пример вывода файлов

        @if(count($page->files) > 0)
            <div class="files">
                <h4 class="h3">Прикрепленные файлы</h4>
                <ul>
                    @foreach($page->files as $file)
                        <li><a href="{!! $file->url !!}">{{$file-
>original_name}}</a></li>
                    @endforeach
                </ul>
            </div>
        @endif
```

Вывести меню в шаблоне :

```
@php($links = \App\Models\Adfm\Menu::getData('main-menu'))
@foreach($links[0] as $el)
<a href="{{$el->link}}">{{$el->title}}</a>
@endforeach
```

Идентификатор меню можно посмотреть в админке в разделе меню (Выделено красным) :



Пример из реальной жизни :

```
<div id="mobile-menu" class="d-md-none">
    <ul class="list-group">
        @php($links = \App\Models\Adfm\Menu::getData('main-menu'))
        <li class="list-group-item">
            <span id="close-mobile-menu" class="close">&times;</span>
        </li>
        @foreach($links[0] as $el)
            <li class="list-group-item">
                <a href="{{$el->link}}">{{$el->title}}</a>
            </li>
        @endforeach
    </ul>
</div>
```

(УСТАРЕЛО) Переменные передаваемые в шаблон

Общая информация Переменные, которые передаются в шаблон, можно посомтреть с помощью выражения

```
{% autoescape false %} {{dump()}} {% endautoescape %}
```

Переменные Page

{{page.id}} => Уникальный идентификатор страницы

{{page.title}} => Заголовок страницы

{{page.alias}} => Синоним страницы

{{page.content}} => html контент страницы, созданный через WYSISWYG редактор текста

{{page.create_date}} => Время создания страницы в UNIX-времени

{{page.type}} => идентификатор типа страницы (0 - обычная страницы, 1 - новость, 2 галерея)

{{page.metaTitle}} => Метатег title - генерируется из заголовка страницы. В большинстве случаев они совпадают

{{page.metaDescription}} => Метатег description - генерируется из первых символов контента страницы. Хотя может быть задан отдельно.

(УСТАРЕЛО) Модуль каталог товаров

С помощью данного модуля можно создавать страницы товаров, и организовывать их по категориям.
Лежит в папке /app/modules/product_catalog.

Адреса и шаблоны страниц

- /categories - страница всех категорий (category-list-page.html.twig)
- /category/{id} - страница одной категории (category-page.html.twig)
- /product/{id} - страница товара (product-page.html.twig)

твиг функции

`getCategories()` - возвращает список категорий верхнего уровня. Пример использования :

```
{% set categories = getCategories() %}
{% for category in categories %}
    <div class="col-md-3">
        <a href="/category/{{ category.id }}">{{ category.title }}</a>
        {% for item in category.children %} # дочерние категории
            <div>{{ item.title }}</div>
        {% endfor %}
    </div>
{% endfor %}
```

свойства моделей

Категория

- {{category.id}} - идентификатор категории
- {{category.parent_id}} - идентификатор родительской категории
- {{category.position}} - порядковый номер в списке. Этот параметр нужен, для сортировки категорий в таком порядке, как они заданы в админке
- {{category.title}} - имя категории
- {{category.children}} - дочерние подкатегории.
- {{category.parent}} - родительская категория.
- {{category.image}} - Картинка категории. Для вывода пути картинки написать {{category.image.url}}
- {{category.created_at}} - дата создания категории
- {{category.deleted_at}} - дата удаления категории

товар

- {{product.id}} - идентификатор товара
- {{product.article}} - артикул товара (необязателен для заполнения). Нужен для синхронизации с 1с или прайсом
- {{product.title}} - название товара
- {{product.alias}} - синоним товара
- {{product.description }} - описание товара. Что бы вывести описание без тегов пишем {{product.description|raw }}
- {{product.price}} - цена товара.
- {{product.image}} - Картинка товара. Для вывода пути картинки написать {{product.image.url}}
- {{product.categories}} - Массив категорий товара. Для вывода всех категорий пишем {% for category in product.categories %}
- {{product.created_at}} - дата создания товара
- {{product.updated_at}} - дата обновления товара

(УСТАРЕЛО) модуль Новости

Список новостей и специальный блок для вывода анонсов

Лежит в папке /app/modules/news.

Адреса и шаблоны страниц

- /news - страница всех новостей (news.html.twig)
- /news/{id} - страница одной новости (news-page.html.twig)

твиг функции

`getLatestNews($count=4)` - возвращает список последних новостей (по умолчанию 4). Пример использования :

```
{% set news = getLatestNews(9) %} # 9 последних новостей
    {% for item in news %}
        <li><a href="/news/{{ item.id }}">{{ item.title }}</a></li>
    {% endfor %}
```

(УСТАРЕЛО) Обратная связь

Модуль для формы обратной связи . Для корректной работы, нужно задать правильные настройки в файле /app/config/mail.yaml
Лежит в папке /app/modules/contactform.

Адреса и шаблоны страниц

- /api/feedback/send - на этот адрес нужно отправить результаты формы. Можно ajax
- Шаблон формы находится в файле form.html

твиг функции

`getContactForm()` - рендерит форму обратной связи в этом месте

```
{{ getContactForm() }}
```

свойства моделей

Новость

- {{news.id}} - идентификатор новости
- {{news.title}} - имя новости
- {{news.content}} - Полный текст новости. Что бы вывести описание без тегов пишем {{news.content|raw}}
- {{news.image}} - Картинка новости. Для вывода пути картинки написать {{news.image.url}}

- {{news.created_at}} - дата создания новости
- {{news.updated_at}} - дата обновления новости

Основы

Система состоит из нескольких пакетов Laravel:
https://github.com/wtolk/adfm - Содержит общие классы , контроллеры и так далее
https://github.com/wtolk/crud - Содержит классы для работы с админкой и генератор моделей, контроллеров и экранов из таблицы бд.

Посмотреть админку можно по адресу /admin/pages . Далее нужно будет ввести логин и пароль заранее созданного пользователя. Как создать пользователя, можно посомтреть на странице с информацией о установке

Экран админки

```
<?php

namespace App\Http\Controllers\Admin\Screens;

use Wtolk\Crud\Form\Checkbox;
use Wtolk\Crud\Form\Column;
use Wtolk\Crud\Form\DateTime;
use Wtolk\Crud\Form\File;
use Wtolk\Crud\Form\Link;
use Wtolk\Crud\Form\MultiFile;
use Wtolk\Crud\Form\Summernote;
use Wtolk\Crud\Form\TableField;
use Wtolk\Crud\FormPresenter;
use App\Models\Adfm\Page;
use Wtolk\Crud\Form\Input;
use Wtolk\Crud\Form\Button;

class PageScreen
{
    public $form;
    public $request;

    public function __construct()
    {
        $this->form = new FormPresenter();
        $this->request = request();
    }

    public static function index()
    {
        $screen = new self();
        $screen->form->template('table-list'); // Объявляем шаблон страницы с таблицей
      □$screen->form->source([
            'pages'  => Page::filter(request()->input('filter'))->paginate(50) // Задаем
модели для экрана
```

```
        ]);
        $screen->form->title = 'Страницы'; // Заголовок экрана
        $screen->form->addField(
            TableField::make('title', 'Название страницы')
                ->link(function ($model) {
                    echo Link::make($model->ru_title)->route('adfm.pages.edit', ['id' =>
$model->id])
                        ->render();
                })
        );
        $screen->form->addField(TableField::make('created_at', 'Дата создания'));
        $screen->form->addField(
            TableField::make('', '')
                ->link(function ($model) {
                    echo Link::make('Удалить')->route('adfm.pages.destroy', ['id' => $model-
>id])->render();
                })
        );
        $screen->form->addField(
            TableField::make('', '')
                ->link(function ($model) {
                    echo Link::make('Просмотр')->route('adfm.show.page', ['slug' => $model-
>slug])->render();
                })
        );
        $screen->form->filters(self::getFilters()); // Объявляем поля по которым будем
фильтровать таблицу

        $screen->form->buttons([ // Кнопки на экране
            Link::make('Добавить')->class('button')->icon('note')-
>route('adfm.pages.create')
        ]);
        $screen->form->build();
        $screen->form->render();
    }

    public static function create()
    {
        $screen = new self();
        $screen->form->isModelExists = false;
```

```
        $screen->form->template('form-edit')->source([
            'page'  => new Page()
        ]);
        $screen->form->title = 'Создание страницы';
        $screen->form->route = route('adfm.pages.store');
        $screen->form->columns = self::getFields(); // В свойство поля передаем метод, в
котором возвращаем список полей
        $screen->form->buttons([
            Button::make('Сохранить')->icon('save')->route('adfm.pages.update')-
>submit(),
        ]);
        $screen->form->build();
        $screen->form->render();
    }

    public static function edit()
    {
        $screen = new self();
        $screen->form->isModelExists = true;
        $screen->form->template('form-edit')->source([
                'page'  => Page::findOrFail($screen->request->route('id'))
        ]);
        $screen->form->title = 'Редактирование страницы';
        $screen->form->route = route('adfm.pages.update', $screen->form->source['page']-
>id);
        $screen->form->columns = self::getFields();
        $screen->form->buttons([
            Button::make('Сохранить')->icon('save')->route('adfm.pages.update')-
>submit(),
            Button::make('Удалить')->icon('trash')->route('adfm.pages.destroy')-
>canSee($screen->form->isModelExists),
            Link::make('Добавить в меню')->icon('trash')
                ->route('adfm.menuitems.createFromModel', [
                    'model_name'  => 'Page',
                    'model_id'  => $screen->request->route('id'),
                    'menu_id'  => '0',
                ])->canSee($screen->form->isModelExists)
        ]);
        $screen->form->build();
        $screen->form->render();
```

```
    }

    public static function getFilters() {
        return [
            Input::make('filter.title:like')->title('Заголовок страницы')->setFilter(),
            Input::make('filter.content:like')->title('Текст страницы')->setFilter(),
        ];
    }

    public static function getFields() {
        return [
            Column::make([
                Input::make('page.ru_title')
                    ->title('Заголовок на русском')
                    ->required(),
                Input::make('page.title')
                    ->title('Заголовок на английском')
                    ->required(),
                Summernote::make('page.ru_content')->title('Содержимое на русском')->devMode($dev_mode),
                Summernote::make('page.content')->title('Содержимое на английском')->devMode($dev_mode),
                MultiFile::make('page.files')->title('Прикрепленные документы')
            ]),
            Column::make([
                Input::make('page.slug')
                    ->title('Вид в адресной строке'),

                Input::make('page.meta.title')
                    ->title('TITLE (мета-тег)'),

                Input::make('page.meta.description')
                    ->title('Description (мета-тег)'),

            ])->class('col col-md-4')
        ];
    }
}
```

ImageCache

Вспомогательный класс, который делает копии изображений заданного размера, что бы гарантировать что изображение которое загрузит пользователь будет соответствовать необходимым размерам.

Как пользоваться :

```
{!! ImageCache::get($file, ['w' => 300, 'h' => 200, 'fit' => 'crop']) !!}
```

Где,
$file - объект типа `App\Models\Adfm\File`
'w' => ширина фото,
'h' => высота фото,
'fit' => тип обрезки изображения, доступные значения : `crop-top-left`, `crop-top`, `crop-top-right`, `crop-left`, `crop-center`, `crop-right`, `crop-bottom-left`, `crop-bottom` `crop-bottom-right` Значение crop это то же самое что и crop-center

Данное объявление вернет тег <img> с адресом на обрезанную картинку. Для картинки можно задать следующие атрибуты title, alt, class, id с помощью соответсвующих методов :

```
{!! ImageCache::get($file, ['w' => 300, 'h' => 200, 'fit' => 'crop'])->title('заголовок') !!}
{!! ImageCache::get($file, ['w' => 300, 'h' => 200, 'fit' => 'crop'])->alt('альтернативная подпись') !!}
{!! ImageCache::get($file, ['w' => 300, 'h' => 200, 'fit' => 'crop'])->className('img-fluid') !!}
{!! ImageCache::get($file, ['w' => 300, 'h' => 200, 'fit' => 'crop'])->id('logo') !!}
```

Примеры из реальной жизни :

Выводим товары на странице категории

```
        @foreach($products as $product)
            <div class="col col-6 col-md-4 card">
                @if($product->images[0])
```

```
{!! ImageCache::get($product->images[0], ['w' => 265, 'h' => 265, 'fit' => 'crop'])
!!}
                @endif
                <div class="title">{{$product->title}}</div>
                <div class="price">{{$product->price}} ₽</div>
            </div>
        @endforeach
```

на строке 3 проверяем, есть ли у товара хотя бы одно изображение, если да, то показываем обрезанную копию.