## Документация Qm framework

[Главная страница](#_Главная_страница)

[Полная документация](#_Полная_документация)

[Контролеры](#_Контролеры)

[Модели](#_Модели)

[Представления или Виды](#_Представления_и_Виды)

[Структуры-модули](#_Структуры_-_модули)

[Скачать](#_Скачать)

[Примеры приложений](#_Примеры_приложений)

[Быстрый старт](#_Быстрый_старт)

[Шаг первый – Установка и настройка Фреймворка](#_1._Установка_и)

[Шаг второй – Анатомия и ход выполнения](#_2._Анатомия_и)

Шаг третий – Генератор кода "QmGen"

Шаг четвертый – Создание M.V.C. вручную

Шаг пятый – Работа с DB

Шаг шестой – расширения системы

# Главная страница

**Быстрый, простой MVC PHP Framework.**

Quick Minimalism - это PHP-Фреймворк для разработки разнообразных веб-приложений, создания сайтов или чего вам вздумается. Quick Minimalism предоставляет готовые инструменты для некоторый стандартных действий, таких как авторизации пользователей, легкая система маршрутизации, Возможность легко достучатся до любых частей представлений, легкой перенаправление на разный layout`ы, применение разных тем. Фреймворк возник не спонтанно, как иногда бывает, но складывал систему постепенно как легкий аналог «могучим» фреймворками, с минимумом кода и с широкими возможностями в расширении и простоте использования, стараясь применять стандартные подходы к структуре построения принципов и цепочек методов взаимодействия между собой.

Практически система отличается от других легких Фреймворков тем что не имеет строгих правил, она легка, быстра и отвечает всем требованиям для построения приложений простой и средней сложности.

Quick Minimalism постоянно развивается, основные приоритеты системы делаются на скорости и простоте кода.

Системные требования.

PHP 5.4 +

Apache

Стандартные расширения PHP для классов-расширений системы

Версия 0.2.0

Скачать ZIP

Скачать с репозитория

Лицензия

Пример приложения

Что внутри.

[ структура ]

# Полная документация

**ХХХ.**

ХХХ.ххх.

# Контролеры

**Логика приложения.**

Контролеры выполняют роль логической составной вмещая в себе все узлы действий программы. По принципам MVC.

Контролер в приложении наследуется от базового контролера:

Class ControllerIndex extends Controller{}

Для более гибких возможностей нужно использовать промежуточные контролеры которые наследуют базовый класс Controller, а их наследуют уже контролеры приложения:

Class BaseSiteController extends Controller{  
 //code…  
}

Class ControllerIndex extends BaseSiteController {}

Класс контролер BaseSiteController может содержать некие общие для всех контролеров выполняемые методы, параметры, загрузки скриптов и стилей или иного, например части sidebar которые в контролерах приложения могут менять их.

**Базовый контролер.**

Класс Controller имеет несколько основных методов и несколько дополнительных, перейдем к описанию их:

ХХХ.ххх.

# Модели

**Описание Каталога Models.**

Модель предназначена для формирования структуры запросов для ваших приложений, принцип и философия соблюдается, так что обойдемся без объяснения элементарных вещей.

**Внутри.**

Стандартно для работы с базой данных используется средства PHP объект PDO, он вложен в две обертки и абстрактный класс Model.

Первая обертка PDO это SafePDO - безопасное соединение и запросы, предотвращает изнанку данных при ошибках.

Вторая обертка SimplePDO обеспечивает подключение к базе данных и несколько методов для элементарных конструкций SQL запросов,

Создает соединение, которое и используется в наших моделях приложения.

Методы и свойства класса SimplePDO:

public $dbh; // Содержит объект соединение к БД

public $sth; // Содержит временные запросы внутренней работы класса

// Подключение модели и доступ к соединению с контролера

$model = $this->model("myModel");

$result = $model->db->exec($sql)

// Доступ к соединению с модели

$result = $this->db->exec($sql);

// Стандартный метод через обертку, принимает стандартный sql запрос

exec($sql)

// Использует стандартный метод execute() через обертку, принимает sql запрос,

// или если указан второй параметр происходит выполнение через метод prepare()

// возвращает экземпляр объекта.

query($sql, array $data=null)

// Выбирает строку с БД, используется после query()->row();

// Выбирает типы: assoc, class, obj

row($type="assoc")

// Выбирает несколько строк с БД, используется после query()->row();

// Выбирает типы: assoc, class, obj

all($type="assoc")

// Закрывает содинение

close()

// Обертка для запроса INSERT...

insert($table, array $dataColumn, array $dataValue)

// Обертка для запроса UPDATE...

update($table, array $dataColumn, array $dataValue, $where)

Подробное описание в комментариях кода.

Соединение с базой данных.

Чтобы создать соединение с базой данных нужно прописать настройки в конфиг-файле configuration.php активного приложения:

// Использование драйвера mysql

"db" => array(

"driver" => "mysql", // имя драйвера

"class" => "SimplePDO", // класс который используется

"host" => "localhost", // с названий все понятно должно быть

"dbName" => "db\_blog", // ...

"user" => "root", // ...

"password" => "", // ...

),

// Использование драйвера sqlite

"db" => array(

"driver" => "sqlite",

"class" => "SimplePDO",

"path" => dirname(\_\_DIR\_\_)."/lib/DATABASE/QmDataBase.sqlite", // путь к файлу базы данных

),

**Использование.**

Создание запросов происходит стандартными методами PDO, он достаточно прост и безопасен, все тонкости работы с PDO расписаны на официальном сайте php.net.

В Фреймворке модели наследуются от родительского класса Model который также, как и обертка SimplePDO содержит несколько простых конструкций для расширений SQL запросов. Опишу их:

Свойство "public $db" содержит объект соединения с БД обращение к все методам SimplePDO происходят через него.

// Подключение модели и доступ к соединению с контролера

$model = $this->model("myModel");

$result = $model->getAll($params);

// Доступ к соединению с модели

$result = $this->getAll($params);

// Выберает все записи с указаной таблицы.

getAll($tbl, $data=null)

// Выберает все с указаной таблицы по id

getById($tbl, $id, $data=null)

// Выберает одну запись с указаной таблицы по названию колонки

getByAttr($tbl, $attr, $attrVal, $data=null)

// Выберает все записи с указаной таблицы по названию колонки

getAllByAttr($tbl, $attr, $attrVal, $data=null)

Каждый метод имеет не обезательный аргумент $data он может быть строкой или массивом параметров

название колонок которые выбераються по умолчанию происходит выборка SELECT \* ...

Например:

// Выберает колонки "title, content, author" с таблицы "table" если колонка "column" имеет значение "column\_value"

$this->getByAttr("table", "column", "column\_value", "title, content, author");

// То же что и выше

$this->getByAttr("table", "column", "column\_value", array(

"title",

"content",

"author"

));

# Представления и Виды

**ХХХ.**

ХХХ.ххх.

# Структуры - модули

**ХХХ.**

ХХХ.ххх.

# Примеры приложений

**ХХХ.**

ХХХ.ххх.

# Скачать

**ХХХ.**

ХХХ.ххх.

# Быстрый старт

**ХХХ.**

ХХХ.ххх.

# 1. Установка и настройка системы

**ХХХ.**

ХХХ.ххх.

# 2. Анатомия и ход выполнения

**ХХХ.**

ХХХ.ххх.