# Консольные команды

Консольные команды необходимо вводить из корневой папки проекта через передачи параметров в файл console

## Создать контроллер

*php console mc my* – Создаст myController

## Создать модель

*php console mm my* – Создаст myModel

## Создать компонент

*php console mcom my* – Создаст компонент my.php

## Создать метод для api

*php console mmet my* – Создаст метод my.php c вашим выбором функции-рендера

*php console mmet my auto* – Создаст метод my.php c автоматическим выбором функции-рендера

## TODO Создать преконтроллер

*php console mpre my* – Создаст преконтроллер my.php

## Создать новый модуль

*php console nm my 1.0* – Создаст новый модуль my версии 1.0

## TODO Создать новый проект

*php console new project 1.0* – Создаст новый проект project на версии ядра 1.0

## TODO Создать класс

*php console class ClassName* – Создаст новый класс ClassName

# Загрузчик файлов

Вам не надо загружать файлы самим из $\_FILES, достаточно сделать:

FL::*loadFiles*([  
 'folder' => 'video',  
 'params' => [  
 'loader\_id' => $system['user']->data['id'],  
 'test\_params' => 'test'  
 ]  
]);

# Кроп

Если вы хотите обрезать фотографию и получить путь до нее, то можно сделать это 2 способами:

1 параметр – id файла

2 параметр – высота

3 параметр – ширина

4 параметр – строго вписать файл в размер или нет, если нет, то произойдет ресайз фото до данных размеров

5 параметров можно передать желаемое качество фотографии (до 100)

$crop = Crop::*getById*(21, 550, 100, **true**);

$crop = Crop::*getById*(21, 550, 100, **false**);

// Удалить весь кроп

Crop::*clearAllCrop*();

# Кеширование

## Вызов модели с кешированием

Кешируем вызов модели с классов testModel метода getData(…$params):

// Закешируется на 30 секунд по сессии  
$data = Cache:: *requireModel* ('testModel', 30);  
  
// Закешируется на 5 минут по сессии  
$data = Cache:: *requireModel* ('testModel');  
  
// Закешируется на 10 минут по сессии  
Cache::*$time\_sec\_cache* = 600; // Задаем время по умолчанию  
$data = Cache:: *requireModel* ('testModel');  
  
// Закешируется на 11 минут по сессии  
$data = Cache::*requireModel*('testModel', 660, $key, $params);

## Прямое кеширование

// Задаем тип кеширования

Cache::$type = ‘File’;

// Задать тип кеширования  
Cache::set($key, $data)

// Закешируется на 5 минут по умолчанию  
Cache::set($key, $data)

// Взять кеш  
Cache::get($key)

// Проверить кеш  
Cache::has($key)

// Сбросить весь кеш

Cache::clear();

// Сбросить весь кеш по ключу

Cache::delete($key);

# Генератор форм

Генерация select:

Пример:

## Генерация полей

$params['form\_auth'] = Form::*generateField*('  
s {test}   
o%%.test%%{ttt}%%.ttt%%.tetfsfsf%%@vaaaal   
o%%.test%%{ttt}%%.ttt%%.tetfsfsf%%@vaaaal   
o%%.test%%{ttt}%%.ttt%%.tetfsfsf%%@vaaaal   
o%%.test%%{ttt}%%.ttt%%.tetfsfsf%%@vaaaal   
');

<select value="test">

<option class="test ttt tetfsfsf" value="ttt">vaaaal</option>

<option class="test ttt tetfsfsf" value="ttt">vaaaal</option>

<option class="test ttt tetfsfsf" value="ttt">vaaaal</option>

<option class="test ttt tetfsfsf" value="ttt">vaaaal</option>

</select>

Form::*generateField*('c {test} @test #test55')

<input type="checkbox" id="test55" value="test"><label for="test55">test</label>

## Основные операции

{} – value

.test – class

.test.test2.test3 – class many (class=”test test2 test3”)

#example – id (id=”example”)

() – placeholder

! - required

[data] – attr

[data=test,test2,test3] = attr with value (data=”test test2 test3”)

# Роутер

Получить полный путь из url’ов:

$SYS->ROUTER->getFullPath()

## Режимы роута

// Задаем простой режим проверки роутов по регулярным выражением  
$SYS->ROUTER->mode\_check\_rout = 'REGEXP\_LITE';

// Задаем режим проверки роутов по регулярным выражением  
$SYS->ROUTER->mode\_check\_rout = 'REGEXP;

// Задаем режим проверки роутов по первому url  
$SYS->ROUTER->mode\_check\_rout = 'FIRST\_URL;

## Преконтроллеры

// Добавляем преконтроллеры  
$SYS->ROUTER->addPrecontrollers(  
 [

// Группа преконтроллеров checkAuth с одним преконтроллером checkAuth  
 'checkAuth',  
 'checkAccess' => ['checkAuthCabinet', 'checkAccess']  
 ]  
);

## Добавление роутов

Добавление роутов осуществляется в файле routes.php в папке /site

$SYS->ROUTER->add('root', 'auth', 'logIn', ['checkAuth']);  
$SYS->ROUTER->add('log-in', 'auth', 'logIn', ['checkAuth']);

$SYS->ROUTER->add('sign-up', 'auth', 'signUp', ['checkAuth']);

// Будет вызван по умолчанию экшн runAction в контроллере catalog  
$SYS->ROUTER->add('cabinet', 'catalog', **null**, ['checkAccess']);  
$SYS->ROUTER->add('catalogs', 'catalog', **null**, ['checkAccess']);

// При роуте 404 вызываем контроллер error runAction  
$SYS->ROUTER->add('404', 'error');

### Режим REGEXP\_LITE

Для его включение измените параметр в начале файла routes.php:

// Задаем простой режим проверки роутов по регулярным выражением  
$SYS->ROUTER->mode\_check\_rout = 'REGEXP\_LITE';

Режим позволяет проверять роуты по регулярным выражениям.

$SYS->ROUTER->add('catalogs/{int}/{int}', 'catalog', null, ['checkAccess']);

$SYS->ROUTER->add('catalogs/{int}/{str}', 'catalog', null, ['checkAccess']);

$SYS->ROUTER->add('catalogs/{int}/{str}', 'catalog');

## Методы

### add()

Метод добавляет роут для проверки для подключения view.

View хранятся в папке site/view.

Используется шаблонизатор twig.

# Логирование

# Работа с базой данных