**controllers**

**SiteController**

class SiteController extends AppController {

//контроллер для траниц в папке views->site

actionAbout() {...}

actionContact() {...}

}

Основной контроллер мы можем переопределить…

**AppController**

use yii\web\Controller;

class AppController extends Controller {

//тут свои действия для всех контроллеров

}

и уже от нашего, переопределенного, наследовать остальные:

**PostController**

class PostController extends AppController {

//контроллер для cтраниц в папке views->post

actionIndex() {

return $this->render('index', ['var' => $hello]); *//index - название вида, который надо обработать. ‘var – переменная передается в вид «index.php»*

}

actionTest() {...}

}

**views (виды)**

**layouts (шаблон – повторяющаяся часть на всех страницах. Хедар, футер и т.д.)**

**main.php**

**site (виды страниц. Для каждой страницы свой файл)**

**about.php**

**contact.php**

**…**

**post**

**index.php**

<p><?= $var ?></p>

**test.php**

**…**

**Альтернативный способ передачи переменых из контролерра в вид**

**В адресной строке мы можем передать параметр в обработчик в контролере в виде:**

http://yii2.mini/web/index.php?r=post/index**&name=Dima**

**PostController**

class PostController extends AppController {

//контроллер для cтраниц в папке views->post

actionIndex($**name = NULL**) {

$hello = “Hello”;

return $this->render('index', compact('hello', ‘**name’**)); *//index - название вида, который надо обработать. ‘hello – переменная передается в вид «index.php»*

}

**views (виды)**

**post**

**index.php**

<p><?= $hello ?></p>

<p><?= $**name** ?></p>

**Включаем и настраиваем ЧПУ**

В корневой папке сайта создаем .htaccess (перенаправляем все запросы в папку web)

RewriteEngine on  
RewriteRule ^(.+)?$ /web/$1

В папке web создаем .htaccess (если не прямое обращение к файлу или папке, то подставляем в запрос index.php)

RewriteBase /  
  
RewriteCond %{REQUEST\_FILENAME} !-f  
RewriteCond %{REQUEST\_FILENAME} !-d  
  
RewriteRule . index.php

Для **включения функции** **ЧПУ**, файле проекта **config -> web.php** расскомментируем следующие строки

'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [  
 ],  
],

Для того, чтобы **избавиться от \web\** в адресной строке, необходимо в этом же файле, после строки

'cookieValidationKey' => '\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*',

дописать

'baseUrl' => '',

**Изменить базовый путь к входному файлу**

В файле **config -> web.php** после строки

'bootstrap' => ['log'],

дописать

'defaultRoute' => 'post/index',

**Перенаправление урлов** (убрать лишние уровни из адреса)

В файле **config -> web.php** в правила менеджера урлов дописываем (подходит для индивидуальных перенаправлений)

'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [

**'test' => 'post/test',**

**'hello' => 'post/hello'**

],

**Перенаправление** используя регулярное выражение (заменяет два предыдущих):

'urlManager' => [  
 'enablePrettyUrl' => true,  
 'showScriptName' => false,  
 'rules' => [

'<action>' => 'post/<action>'

],

**Настройки БД**

config -> db.php

**Модели**

**Название модели = название таблицы в БД**

Для работы с БД нужно подключить класс

use yii\db\ActiveRecord;

и наследуем от него класс контроллера

class Post extends ActiveRecord{

}

По умолчанию активрекорд будет использовать таблицу в БД с там же именем, как и класс.

Можно задать имя таблицы вручную:

public static function tableName(){  
 return 'post';  
}

Модель необходимо подключить в контроллер где мы ее используем

use app\models\Post;

3 этапа использования класса

class Post extends ActiveRecord{

1. Создаем новый объект запроса (ActiveRecord::find())
2. Настройка этого объекта (настройка выборки)
3. Выборка данных (->all() – выбрать все)

$posts = Post::find()->all;

Выбираем необходимые столбцы

$posts = Post::find()->select('id, title, excerpt')->all();

Генерируем урл для перехода на материал

<a href=**"**<?= \yii\helpers\Url::to(['post/view', 'id' => $post->id]) ?>**"**></a>

Генерирует ссылку вида site**/post/view?id=3**

Переформатируем в формат site**/post/3**

В файле проекта **config -> web.php,** в urlManagere, поле ruls дописываем правило

*//'<action>' => 'post/<action>',*'post/<id:\d+>' => 'post/view'

post – контроллер

<id: > - GET параметр

=> куда отправляем

!Проверка останавливается на первом совпадении с шаблоном. По этому, необходимо более общие правила писать ниже, а детальные – выше!

**ПАГИНАЦИЯ**

В классе - class PostController extends AppController

$query = Post::find()->select('id, title, excerpt')->orderBy('id DESC');  
$pages = new Pagination(['totalCount' => $query->count(), 'pageSize' => 2]);  
$posts = $query->offset($pages->offset)->limit($pages->limit)->all();

В index.php

<?= \yii\widgets\LinkPager::widget(['pagination' => $pages]) ?>

Удаления из строки «http://site/post/index?page=1&per-page=1» лишних тега «per-page» и ГЕТ-параметра “page” на главной странице:

$pages = new Pagination(['totalCount' => $query->count(), 'pageSize' => 2, **'pageSizeParam' => false, 'forcePageParam' => false**]);

При пагинации, убираем index из строки, форматируем строку к виду /page/2

'page/<page:\d+>' => 'post/index'

При пагинации убираем index из строки

'/' => 'post/index'

Получить GET параметра ‘id’

$id = \Yii::*$app*->request->get('id');

Генерация исключения, вывод 404 ошибки

if(empty($post)) throw new \yii\web\HttpException(404, "Такой страницы нет");