Бесплатная электронная книга

УЧУСЬ

yii2

Free unaffiliated eBook created from
Stack Overflow contributors.

#yii2

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: yii2

It is an unofficial and free yii2 ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official yii2.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с yii2

замечания

Yii представляет собой общую структуру веб-программирования, что означает, что ее можно использовать для разработки всех видов веб-приложений с использованием PHP. Благодаря своей компонентной архитектуре и сложной поддержке кеширования она особенно подходит для разработки крупномасштабных приложений, таких как порталы, форумы, системы управления контентом (CMS), проекты электронной коммерции, веб-службы RESTful и т. Д.

Версии

Версия	Дата выхода
2.0.12	2017-06-05
2.0.11	2017-02-01
2.0.10	2016-10-20
2.0.9	2016-07-11
2.0.8	2016-04-28
2.0.7	2016-02-14
2.0.6	2015-08-06
2.0.5	2015-07-11
2.0.4	2015-05-10
2.0.3	2015-03-01
2.0.2	2015-01-11
2.0.1	2014-12-07
2.0.0	2014-10-12

Examples

Установка или настройка

Yii2 можно установить двумя способами. Они есть

1. Установка через Composer
2. Установка из архивного файла

Установка через Composer

Установка композитора

Если у вас еще нет Composer, вы можете сделать это, выполнив инструкции на getcomposer.org . В Linux и Mac OS X вы выполните следующие команды:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /usr/local/bin/composer
```

Для Windows просто загрузите и установите composer-setup.exe Возможно, вам придется настроить токен доступа к API github, чтобы преодолеть ограничение скорости Github API.

Установка Yii

С установленным Composer вы можете установить Yii, выполнив следующие команды в папке, доступной в Интернете:

```
composer global require "fxp/composer-asset-plugin:^1.2.0"
composer create-project --prefer-dist yiisoft/yii2-app-basic basic
```

затем выполните следующую команду для установки Yii2 с базовым шаблоном.

```
composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-basic basic
```

Чтобы установить Yii2 с расширенным запуском шаблона

```
composer create-project --prefer-dist --stability=dev yiisoft/yii2-app-advanced advanced
cd advanced
php init
```

После этого создайте новую базу данных и настройте конфигурацию компонентов ['db'] в общем / config / main-local.php соответственно. затем выполните следующую команду:

```
php yii migrate
```

Установка из архивного файла

1. Загрузите файл архива из Yii-download
2. Распакуйте загруженный файл в папку, доступную в Интернете.
3. Измените файл config / web.php, введя секретный ключ для элемента конфигурации cookieValidationKey

   Вы можете добавить любой тип ключа, который вы хотите:

   ```
   'cookieValidationKey' => '',

   For example : xyctuyvibonp

   'cookieValidationKey' => 'xyctuyvibonp',
   ```

```
 //insert a secret key in the following (if it is empty) - this is required by cookie
validation
'cookieValidationKey' => 'enter your secret key here',
```

Установить Yii2 в ubuntu

Сначала нам нужно установить композитор. Шаги по установке композитора Install Composer.

```
curl -sS https://getcomposer.org/installer | php
```

Теперь измените каталог:

```
sudo mv composer.phar /usr/local/bin/composer
```

Проверьте, работает ли композитор

```
composer
```

Теперь установлен Composer.

Существует два способа установки Yii2.

1.Установка из файла архива

Получите zip-файл из-под ссылки.

Разархивируйте его в каталог назначения, например `/var/www/html` .

https://github.com/yiisoft/yii2/releases/download/2.0.8/yii-advanced-app-2.0.8.tgz

Переместитесь в «расширенную» папку. Переместите вручную или введите команду ниже.

```
cd advanced
```

Выполните команду ниже.

```
php init
```

2.Установка через Composer

Для установки через композитор требуется токен аутентификации github. Для токена вам нужно зарегистрироваться в GitHub.

После регистрации вы можете создать свой токен:

Шаги для создания токена

1. В правом верхнем углу любой страницы щелкните фотографию своего профиля и нажмите «Настройки».
2. На боковой панели пользовательских настроек выберите Личные листы доступа.
3. Нажмите «Создать новый токен».
4. Дайте вашему маркеру описательное имя.
5. Выберите области, которые вы хотите предоставить этому токену.
6. Нажмите «Создать токен».
7. Скопируйте токен в буфер обмена. Из соображений безопасности после перехода с этой страницы никто не сможет снова увидеть токен.

Ссылка: https://help.github.com/articles/creating-an-access-token-for-command-line-use/

После генерации токена скопируйте его

Изменить каталог

```
cd /var/www/html/
```

Выполнить команду ниже

```
composer config -g github-oauth.github.com <AuthToken>
```

пример:

```
composer config -g github-oauth.github.com f1eefb8f188c22dd6467f1883cb2615c194d1ce1
```

Установить yii2

```
composer create-project --prefer-dist yiisoft/yii2-app-advanced advanced
```

Переместитесь в «расширенную» папку. Переместите вручную или введите команду ниже.

```
cd advanced
```

Выполните команду ниже.

```
php init
```

Это сделано!

Теперь вы можете проверить это.

HTTP: // локальный / продвинутый / интерфейс / веб

а также

HTTP: // локальный / продвинутый / бэкенд / веб

Прочитайте Начало работы с yii2 онлайн: https://riptutorial.com/ru/yii2/topic/788/начало-работы-с-yii2

глава 2: Pjax

Examples

Шаг 1 Добавить структуру

В views \ site \ form-submission.php

```
<?php Pjax::begin(['id'=>'id-pjax']); ?>
<?= Html::beginForm(['site/form-submission'], 'post', ['data-pjax' => '', 'class' => 'form-
inline']); ?>
    <?= Html::input('text', 'string', Yii::$app->request->post('string'), ['class' => 'form-
control']) ?>
    <?= Html::submitButton('Hash String', ['class' => 'btn btn-lg btn-primary', 'name' =>
'hash-button']) ?>
<?= Html::endForm() ?>
<h3><?= $stringHash ?></h3>
<?php Pjax::end(); ?>
```

Шаг 2 Код на стороне сервера

```
public function actionFormSubmission()
{
    $security = new Security();
    $string = Yii::$app->request->post('string');
    $stringHash = '';
    if (!is_null($string)) {
        $stringHash = $security->generatePasswordHash($string);
    }
    return $this->render('form-submission', [
        'stringHash' => $stringHash,
    ]);
}
```

как использовать pjax

Добавьте эту строку в начало вашего просмотра.

```
<?php
use yii\widgets\Pjax;
?>
```

Добавьте следующие две строки вокруг содержимого, для которого требуется частичное обновление.

```
<?php Pjax::begin(['id'=>'id-pjax']); ?>
Content that needs to be updated
<?php Pjax::end(); ?>
```

перезагрузить pjax

```
$.pjax.reload({container: '#id-pjax'});
```

использовать аргумент таймаута в pjax

```
<?php Pjax::begin(['id'=>'id-pjax', 'timeout' => false]); ?>
```

вы можете указать целочисленное значение для аргумента тайм-аута, для которого будет ожидаемое количество миллисекунд (его значение по умолчанию - 1000). Если время выполнения на сервере больше указанного значения таймаута, будет запущена полная загрузка страницы.

По умолчанию pjax отправит форму с использованием метода GET. Вы можете изменить способ отправки формы на POST, как в следующем примере

```
<?php Pjax::begin(['id'=>'id-pjax', 'timeout' => false, 'clientOptions' => ['method' =>
'POST']]); ?>
```

Расширенное использование Pjax

Yii Framework 2.0 поставляется со встроенной поддержкой Pjax , библиотеки JavaScript, которая уменьшает время загрузки страницы. Он выполняет это, только обновляя часть страницы, которая изменилась через Ajax, что может привести к значительной экономии, если на ваших страницах есть много других активов. Некоторые из наших проектов используют эту функциональность, и мы хотели поделиться некоторыми извлеченными уроками.

Проблема : Страница 1 - простая статическая страница, содержащая несколько элементов. Страница 2 включает ActiveForm, а также другие виджеты. Ресурсы JavaScript для ActiveForm необходимо загружать для запуска встроенного JavaScript, но поскольку они не включали эти активы, при попытке выполнить строку activeform столкнулся с ошибкой JavaScript: «Uncaught TypeError: undefined не является функция».

Решение . Включите активы ActiveForm в пакет общих ресурсов, который будет загружен по всем страницам, гарантируя, что на любой странице входа будут доступны правильные сценарии.

```
class AppAsset extends AssetBundle
{
    ...
    public $depends = [
        'yii\widgets\ActiveFormAsset',
        'yii\validators\ValidationAsset',
    ];
```

```
    ...
}
```

Проблема : в том же примере, приведенном выше, Page 1 включает несколько виджетов (NavBar и т. Д.). Page 2 включает в себя те же виджеты и еще несколько (ActiveForm и т. Д.). При загрузке страницы через Pjax выполнялся какой-то пользовательский встроенный JavaScript, но встроенный скрипт, размещенный виджетами ActiveForm, не работал, поскольку код проверки не работал. В отладке мы обнаружили, что функция init ActiveForm была запущена, но переменная 'this', похоже, не соответствовала ActiveForm. Фактически это соответствовало div NavBar. Исследуя идентификаторы div, мы увидели, что ActiveForm ожидал иметь идентификатор # w1, но NavBar уже был назначен этот идентификатор на странице 1, поскольку это был первый виджет, встречающийся на этой странице.

Решение . Не полагайтесь на Yii, чтобы автоматически генерировать идентификаторы виджета для вас. Вместо этого всегда указывайте идентификатор при создании виджета, чтобы поддерживать контроль над этими идентификаторами.

Проблема : запрос Pjax отменялся ровно через 1000 мс после начала запроса.

Решение . Увеличьте настройку тайм-аута Pjax. Он по умолчанию составляет 1 секунду, что должно быть приемлемым для производственных сайтов. Однако при разработке, используя xdebug, время загрузки нашей страницы регулярно превышает этот предел.

Проблема : веб-приложение реализует шаблон Post-Redirect-Get (PRG) . Pjax перезагружает всю страницу, а не только перенаправление.

Решение . Это предполагаемое поведение Pjax. Перенаправление не служит своей цели при использовании Pjax, поэтому вы можете определить, является ли запрос Pjax, и если да, то визуализируйте контент вместо перенаправления. Пример может выглядеть так:

```
$endURL = "main/endpoint";
if (Yii::$app->request->isPjax) {
    return $this->run($endURL);
} else {
    return $this->redirect([$endURL]);
}
```

Каков ваш опыт работы с Pjax и Yii? Комментарий ниже, если вы нашли какие-либо ошибки или получили лучшие решения, чем наши!

Прочитайте Pjax онлайн: https://riptutorial.com/ru/yii2/topic/4554/pjax

глава 3: Yii2 ActiveForm

Examples

Поля формы в Yii2

Отображение базового примера страницы просмотра в Yii2 для новых учеников

Это базовые классы, которые необходимо добавить для создания формы с помощью yii2 ActiveForm

```
<?php

Use yii\helpers\Html;
Use yii\widgets\ActiveForm;
```

Строка «Ниже» запустит тег формы для нашей формы ниже, показывая, что пример показывает, как указать идентификатор формы и как применять любые классы для формы.

```
$form =ActiveForm::begin([   'id'=> 'login-form',   'options'=> ['class' => 'form-
horizontal'],]) ?>
```

Здесь $ model Укажите, какое поле таблицы базы данных мы хотим связать с формой, которая хранится здесь в этой переменной, которая была передана из соответствующего контроллера.

```
   <?= $form->field($model, 'username') ?>
   <?= $form->field($model, 'password')->passwordInput() ?>
```

«имя пользователя» и «пароль» - это имя поля таблицы, с которым будет привязано наше значение.

Здесь, в приведенном ниже коде, мы помещаем кнопку отправки для отправки формы и применяем «Логин» в качестве текстового элемента Button и базовых классов css.

```
<div class="form-group">
      <div class="col-lg-offset-1 col-lg-11">
          <?= Html::submitButton('Login', ['class' => 'btn btn-primary']) ?>
      </div>
  </div>
```

Здесь, в нижнем коде, мы закрываем форму

```
<?php ActiveForm::end() ?>
```

Создать поле пароля:

```
<?= $form->field($model, 'password')->passwordInput() ?>
```

Создать TextField:

```
<?= $form->field($model, 'username') ?>
```

Создать скрытое поле формы:

```
echo $form->field($model, 'hidden1')->hiddenInput()->label(false);
```

Создание раскрывающегося списка:

```
<?php echo $form->field($model, 'name')
->dropdownList(
Stud::find()->select(['name'])
->indexBy('name')->column(),
['prompt'=>'Select no']); ?>
```

Выпадающий список с идентификатором и именем

```
<?= $form->field($model, 'name')->dropDownList(
            ArrayHelper::map(Stud::find()->all(), 'no', 'name'),['prompt' => 'Select Car
Name']
 ) ?>
```

Создать FileUploader:

```
echo $form->field($model, 'imagepath')->fileInput();
```

Добавление метки места и индивидуальной метки

```
 <?= $form->field($model, 'username')->textInput()->hint('Please enter your  name')-
>label('Name') ?>
```

Проверка ActiveForm

Вы можете включить / отключить проверки ajax и клиента в активной форме.

```
$form = ActiveForm::begin([
    'id' => 'signup-form',
    'enableClientValidation' => true,
    'enableAjaxValidation' => true,
    'validationUrl' => Url::to('signup'),
]);
```

1. `enableClientValidation` по умолчанию включен в ActiveForm. Если вы не нуждаетесь в

проверке клиента в форме, мы можем отключить, присвоив значение false.

2. `enableAjaxValidation` по умолчанию отключен в ActiveForm. Если вы хотите включить его, мы должны добавить вручную в ActiveForm, как указано выше.
3. `validationUrl` - если вы хотите сохранить все кодирование проверки в отдельном действии контроллера для этой формы, мы можем настроить активную форму с помощью `validationUrl` . Если мы не установили это, оно примет значение действия формы.

Вышеуказанные два аргумента повлияют на всю форму. Если вы хотите проверить валидацию ajax только для определенного поля в форме, вы можете добавить `enableAjaxValidation` для этого конкретного поля. Он будет работать только для этой области не всей формы.

Например, в регистрационной форме вы хотите проверить, что имя пользователя уже существует. Проверка на момент ввода пользователя в форму. вы можете использовать этот аргумент `enableAjaxValidation` для этого поля.

```
echo $form->field($model, 'username', ['enableAjaxValidation' => true]);
```

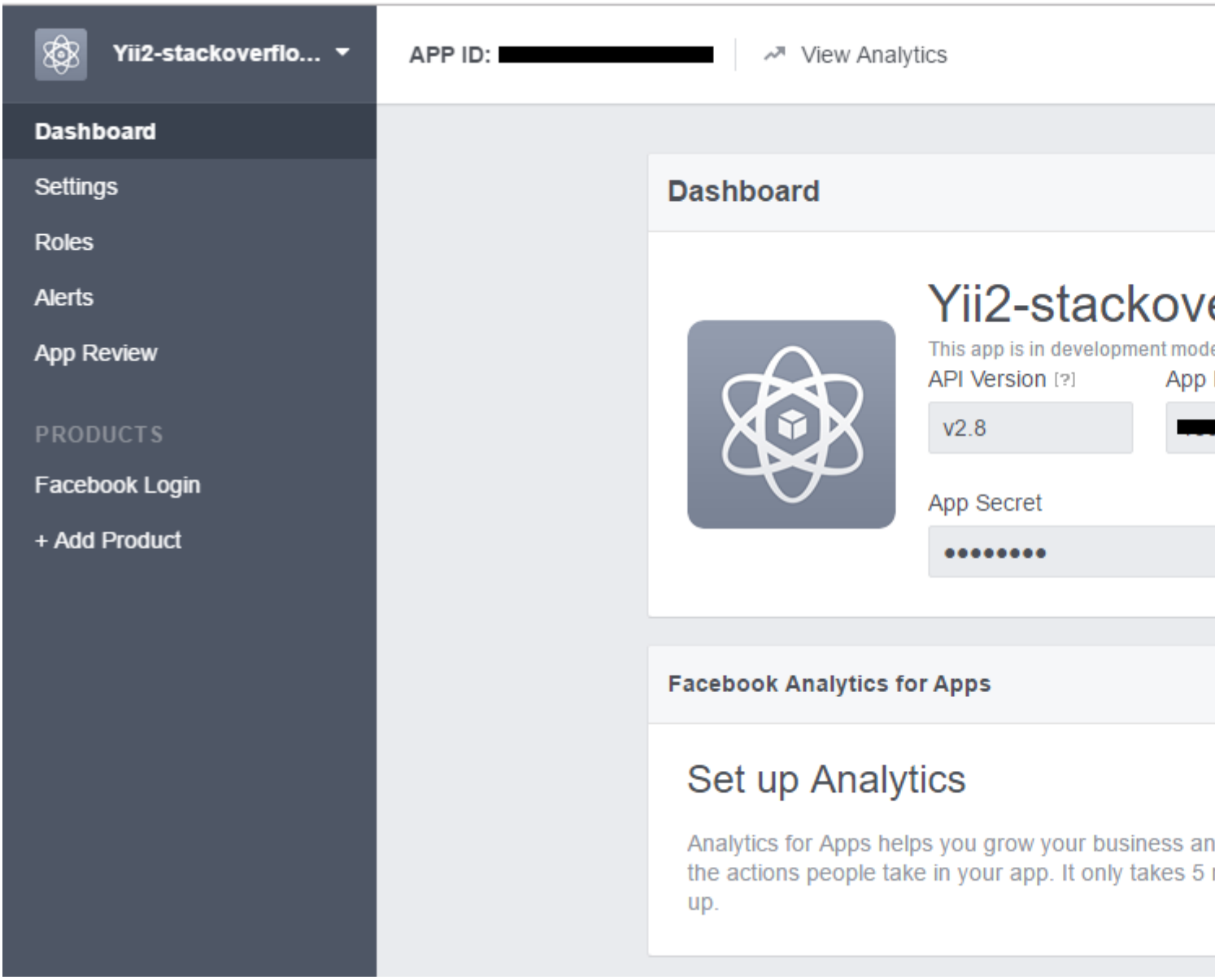
Прочитайте Yii2 ActiveForm онлайн: https://riptutorial.com/ru/yii2/topic/6807/yii2-activeform

глава 4: Yii2 OAuth2 - Ex: потребительский facebook OAuth2

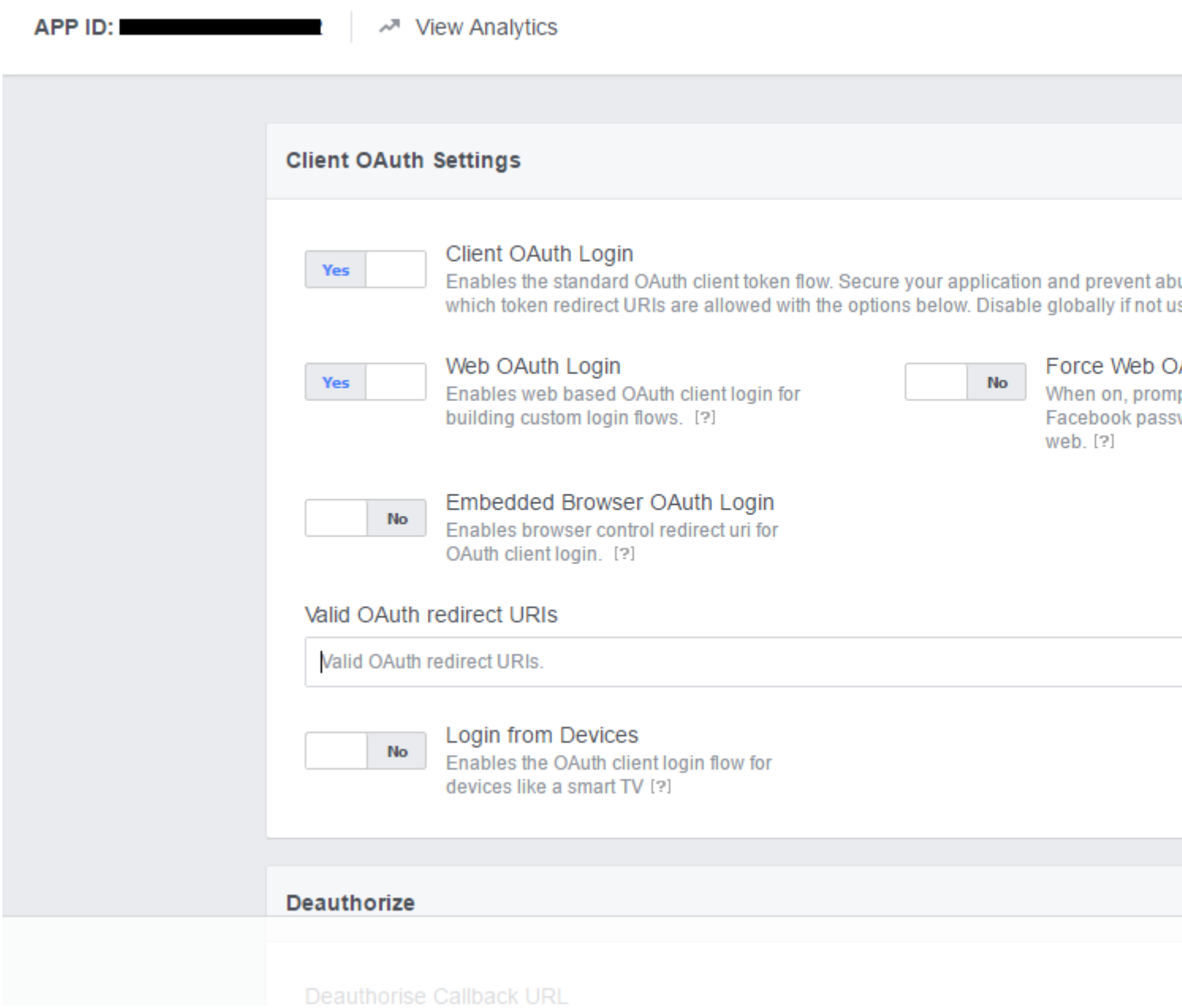
Examples

Создайте приложение для разработчика facebook

Перейдите на страницу https://developers.facebook.com/ и создайте приложение.



Нажмите `Add product` и выберите `Facebook Login`



Установить yii2-authclient

Перед установкой этого расширения вы должны установить yii2-app. В этом примере я использую шаблон yii2-basic. Руководство по установке здесь .

Бежать

```
composer require --prefer-dist yiisoft/yii2-authclient
```

или добавить

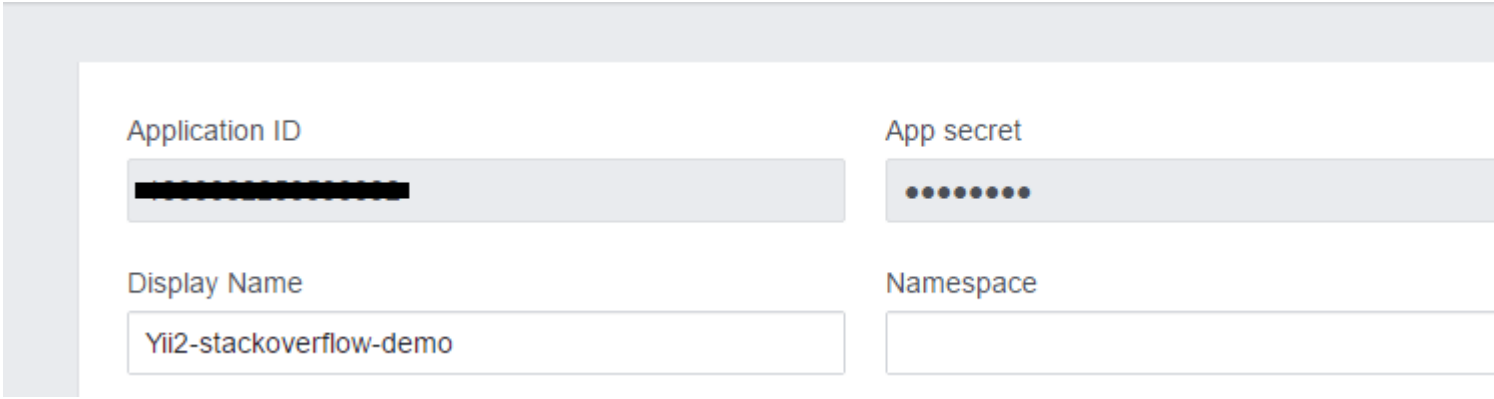
```
"yiisoft/yii2-authclient": "~2.1.0"
```

к разделу `require` вашего `composer.json` .

Добавьте конфигурацию `authClientCollection` в свои `components` конфигурации:

```
return [
    'components' => [
        'authClientCollection' => [
            'class' => 'yii\authclient\Collection',
            'clients' => [
                'facebook' => [
                    'class' => 'yii\authclient\clients\Facebook',
                    'clientId' => 'facebook_client_id',
                    'clientSecret' => 'facebook_client_secret',
                ],
            ],
        ]
    ],
    // ...
];
```

`facebook_client_id` - это идентификатор приложения, а `facebook_client_secret` - секрет приложения.



Добавить действие auth и настроить обратный вызов

1. Кнопка «Добавить» `Login as facebook account` в окно входа в систему:

Отредактируйте `site/login.php` в папке views, добавьте строку тезисов к содержимому входа в страницу:

```
<?= yii\authclient\widgets\AuthChoice::widget([
     'baseAuthUrl' => ['site/auth'],
     'popupMode' => false,
]) ?>
```

Выше мы установили, что действие `auth` в `SiteController` будет обрабатывать поток OAuth2.

Теперь мы его создаем.

```
class SiteController extends Controller
{
    public function actions()
    {
        return [
```

```
            'auth' => [
                'class' => 'yii\authclient\AuthAction',
                'successCallback' => [$this, 'onAuthSuccess'],
            ],
        ];
    }

    public function onAuthSuccess($client)
    {
        // do many stuff here, save user info to your app database
    }
}
```

Мы используем `yii\authclient\AuthAction` для создания URL- `yii\authclient\AuthAction` и перенаправления на страницу входа в facebook.

Функция `onAuthSuccess` используется для получения информации о пользователе, войдите в свое приложение.

Добавьте redirect_url в настройку приложения facebook

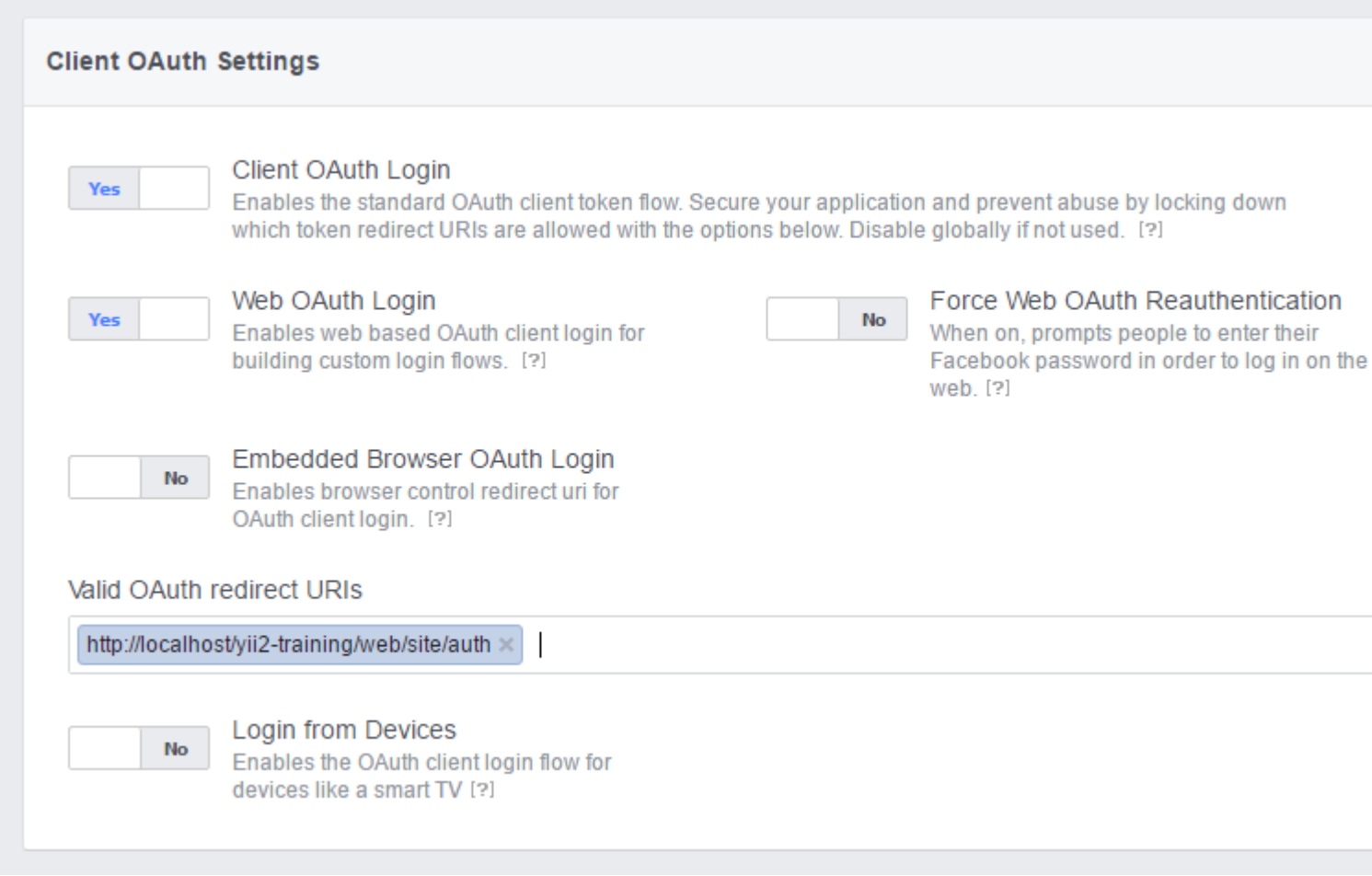
Если вы включите prettyUrl в своем yii2-приложении, ваша redirect_uri будет:

```
http://<base_url>/web/site/auth
```

И отключить довольно URL:

```
http://<base_url>/web/index.php?r=site%2Fauth
```

Пример:



Пример для функции onAuthSuccess

```
/**
 * @param $client ClientInterface
 */
public function onAuthSuccess($client)
{
    //Get user info
    /** @var array $attributes */
    $attributes = $client->getUserAttributes();
    $email = ArrayHelper::getValue($attributes, 'email'); //email info
    $id = ArrayHelper::getValue($attributes, 'id'); // id facebook user
    $name = ArrayHelper::getValue($attributes, 'name'); // name facebook account

    //Login user
    //For demo, I will login with admin/admin default account
    $admin = User::findByUsername('admin');
    Yii::$app->user->login($admin);
}
```

Прочитайте Yii2 OAuth2 - Ex: потребительский facebook OAuth2 онлайн:
https://riptutorial.com/ru/yii2/topic/7428/yii2-oauth2---ex--потребительский-facebook-oauth2

глава 5: Активная запись

замечания

AR идеально подходит, когда вам нужно удалить, обновить или создать одну или несколько записей последовательно. Его поддержка грязных атрибутов (сохранение только того, что было действительно изменено) приводит к оптимизированным операторам UPDATE, которые значительно увеличивают нагрузку на базу данных и уменьшают шансы на различные конфликты, связанные с одновременным редактированием одной записи несколькими лицами.

Если у вас нет сложной логики в вашем приложении, и поэтому она не требует абстрагирования сущностей, AR лучше всего подходит для удаления, обновления и создания.

AR также подходит для простых запросов, результатом чего является менее 100 записей на странице. Это не так эффективно, как работа с массивами, создаваемыми конструктором запросов или asArray (), но с большим удовольствием работать.

AR не рекомендуется для сложных запросов. Обычно это связано с агрегацией или преобразованием данных, поэтому возвращаемое не соответствует модели AR. В этом случае предпочтительнее использовать построитель запросов.

То же самое касается импорта и экспорта. Лучше использовать построитель запросов из-за большого количества данных и, возможно, сложных запросов.

Examples

Найти все записи

```
Post::find()->all();
// SELECT * FROM post
```

или сокращенное

(Возвращает экземпляр активной модели записи с помощью первичного ключа или массива значений столбца.)

```
Post::findAll(condition);
```

возвращает массив экземпляров ActiveRecord.

Найти все с помощью условия

```
$model = User::find()
        ->where(['id' => $id])
        ->andWhere('status = :status', [':status' => $status])
        ->all();
```

Найти все с orderBy

```
$model = User::find()
         ->orderBy(['id'=>SORT_DESC])
        ->all();
Or

$model = User::find()
         ->orderBy(['id'=>SORT_ASC])
        ->all();
```

Где пункт

ОПЕРАТОРЫ

```
$postsGreaterThan = Post::find()->where(['>', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at > '2016-01-25'

$postsLessThan = Post::find()->where(['<', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at < '2016-01-25'

$postsNotEqual = Post::find()->where(['<>', 'created_at', '2016-01-25'])->all();
// SELECT * FROM post WHERE created_at <> '2016-01-25'
```

В

```
$postsInArray = Post::find()->where(['id' => [1,2,3]])->all();
// SELECT * FROM post WHERE id IN (1,2,3)
```

МЕЖДУ

```
$postsInBetween = Post::find()
->where(['between', 'date', "2015-06-21", "2015-06-27" ])
->all();
```

НОЛЬ

```
$postsWithNullTitle = Post::find()->where(['title' => null]);
// SELECT * FROM post WHERE title IS NULL
```

А ТАКЖЕ

```
$postsAND = Post::find()->where(['title' => null, 'body' => null]);
// SELECT * FROM post WHERE title IS NULL AND body IS NULL
```

ИЛИ ЖЕ

```
$postsAND = Post::find()->where(['OR', 'title IS NULL', 'body IS NULL']);
// SELECT * FROM post WHERE title IS NULL OR body IS NULL
```

НЕ

```
 $postsNotEqual = Post::find()->where(['NOT', ['created_at'=>'2016-01-25']])->all();
// SELECT * FROM post WHERE created_at IS NOT '2016-01-25'
```

НЕЗАВИСЛЕННЫЕ КЛАССЫ

```
$postsNestedWhere = Post::find()->andWhere([
    'or',
    ['title' => null],
    ['body' => null]
])->orWhere([
    'and',
    ['not', ['title' => null]],
    ['body' => null]
]);
// SELECT * FROM post WHERE (title IS NULL OR body IS NULL) OR (title IS NOT NULL AND body IS
NULL)
```

КАК ОПЕРАТОР с фильтром.

Например, в поисковом фильтре вы хотите отфильтровать сообщение, обжигая заголовок или описание, опубликованные в настоящее время вошедшим в систему пользователем.

```
$title = 'test';
$description = 'test';
```

i) иFilterWhere ()

```
$postLIKE = Post::find()->where(['user_id' => Yii::$app->user->getId()])-
>andfilterWhere(['or', ['title' => $title, 'description' => $description]])->all();
//SELECT * FROM post WHERE user_id = 2 AND ((`title` LIKE '%test%') OR (`description` LIKE
'%test%'))
```

ii) илиFilterWhere ()

```
$postLIKE = Post::find()->where(['user_id' => Yii::$app->user->getId()])->orFilterWhere(['or',
['title' => $title, 'description' => $description]])->all();
//SELECT * FROM post WHERE user_id = 2 OR ((`title` LIKE '%test%') OR (`description` LIKE
'%test%'))
```

iii) filterWhere ()

```
$postLIKE = Post::find()->filterWhere(['AND', ['title' => $title, 'description' =>
$description]])->andWhere(['user_id' => Yii::$app->user->getId()])->all();
//SELECT * FROM post WHERE ((`title` LIKE '%test%') AND (`description` LIKE '%test%')) AND
user_id = 2
```

Примечание. При использовании filterWhere () мы должны вызывать все andwhere () или orWhere () после фильтраWhere () в противном случае все, где будут удалены условия, кроме filterWhere ()

Создайте класс ActiveRecord с знаками полей на основе событий

```
    <?php
    namespace models;

    use yii\db\ActiveRecord;
    use yii\behaviors\TimestampBehavior;

    class Post extends ActiveRecord
    {
        public static function tableName()
        {
            return 'post';
        }

        public function rules() {
            return [
                [['created_at', 'updated_at'], 'safe'],
            ];
        }
        public function behaviors() {
            parent::behaviors();

            return [
              'timestamp' => [
                'class' => TimestampBehavior::className(),
                'attributes' => [
                  ActiveRecord::EVENT_BEFORE_INSERT => ['created_at', 'updated_at'],
                  ActiveRecord::EVENT_BEFORE_UPDATE => ['updated_at']
                ],
                'value' => date('Y-m-d H:i:s'),
              ]
            ];
          }
    }
```

Или это можно использовать

```
public function beforeSave($insert)
{
    if($this->isNewRecord){
        //When create
    }else{
         //When update
    }

    return parent::beforeSave($insert);
}

  public function afterSave($insert, $changedAttributes )
{
    if($insert){
        //When create
```

```
    }else{
         //When update
    }
    return parent::afterSave($insert, $changedAttributes);
}
```

Найти одну запись

```
$customer = Customer::findOne(10);
```

или же

```
$customer = Customer::find()->where(['id' => 10])->one();
```

или же

```
$customer = Customer::find()->select('name,age')->where(['id' => 10])->one();
```

или же

```
$customer = Customer::findOne(['age' => 30, 'status' => 1]);
```

или же

```
$customer = Customer::find()->where(['age' => 30, 'status' => 1])->one();
```

Поиск по одному запросу

Найдите одиночную запись на основе идентификатора.

```
$model = User::findOne($id);
```

Выберите один столбец на основе идентификатора.

```
$model = User::findOne($id)->name;
```

Извлеките одну запись из базы данных на основе условия.

```
$model = User::find()->one();  // give first record

$model = User::find()->where(['id' => 2])->one(); // give single record based on id
```

Выберите запись одиночных колонок из базы данных на основе условия.

```
$model = User::find()->select('name,email_id')->where(['id' => 1])->one();
```

ИЛИ ЖЕ

```
$model = User::find()->select(['id','name','email_id'])->where(['id' => 1])->one();
```

Сортировать по

```
$model = User::find()->select(['id','name','email_id'])->orderBy(['id' => SORT_DESC])->one();

OR

$model = User::find()->select(['id','name','email_id'])->orderBy(['id' => SORT_ASC])->one();
```

Активные записи с подзапросами

Пример: клиенты, которые могут создать сообщение. Каждый клиент может создавать несколько сообщений. Как только клиент создаст сообщение, сообщение будет просмотрено администраторами. Теперь нам нужно получить список клиентов, у которых есть все активные сообщения, используя подзапрос.

Примечание. Если у клиента 5 сообщений, среди 5 должностей, если у него есть хотя бы один неактивный, мы должны исключить этого клиента из списка клиентов.

```
$subQuery = Post::find()->select(['customer_id'])->where(['status' => 2]); //fetch the
customers whos posts are inactive - subquery
$query = Customer::find()->where(['NOT IN', 'id', $subQuery])->all(); //Exclude the customers
whos posts are inactive by using subquery
```

Прочитайте Активная запись онлайн: https://riptutorial.com/ru/yii2/topic/1516/активная-запись

глава 6: Восстановительный API

Examples

Начните с отдыха api

У нас есть таблица, которая включает в себя страны, поэтому мы создаем модель, которая называется моделью списка стран

```
<?php

namespace app\models;

use Yii;

/**
 * This is the model class for table "countrylist".
 *
 * @property integer $id
 * @property string $iso
 * @property string $name
 * @property string $nicename
 * @property string $iso3
 * @property integer $numcode
 * @property integer $phonecode
 */
class Countrylist extends \yii\db\ActiveRecord
{
    /**
     * @inheritdoc
     */
    public static function tableName()
    {
        return 'countrylist';
    }

    /**
     * @inheritdoc
     */
    public function rules()
    {
        return [
            [['iso', 'name', 'nicename', 'phonecode'], 'required'],
            [['numcode', 'phonecode'], 'integer'],
            [['iso'], 'string', 'max' => 2],
            [['name', 'nicename'], 'string', 'max' => 80],
            [['iso3'], 'string', 'max' => 3]
        ];
    }

    /**
     * @inheritdoc
     */
    public function attributeLabels()
    {
```

```
        return [
            'id' => 'ID',
            'iso' => 'Iso',
            'name' => 'Name',
            'nicename' => 'Nicename',
            'iso3' => 'Iso3',
            'numcode' => 'Numcode',
            'phonecode' => 'Phonecode',
        ];
    }
}
```

и я создаю для этого веб-сервис для отдыха, мы создаем контроллер для restapi и устанавливаем переменную modelClass для нашей модели.

```
 <?php
    namespace app\controllers;
    use yii\rest\ActiveController;
    use Yii;
    class CountrylistController extends ActiveController
    {
      public $modelClass='app\models\Countrylist';
    }
?>
```

для использования restapi нам нужны довольно URL-адреса и
мы добавляем это правило для довольно URL-адреса

```
'urlManager' => [
   'class' => 'yii\web\UrlManager',
   'enablePrettyUrl' => true,
   'showScriptName' => false,
   'rules' => [
       ['class'=>'yii\rest\UrlRule','controller'=>'countrylist']
       ],
    ],
```

после этого мы можем получить доступ к тестированию нашего отдыха api в качестве примера

http: // localhost / countrylist дает нам список графств.

Как переопределить действия по умолчанию для отдыха api Yii2

В качестве примера вы хотите отключить разбиение на страницы в своем действии по умолчанию и получить все результаты в индексе. Как вы можете это сделать? Это просто. Вы должны переопределить действие индекса в своем контроллере следующим образом:

```
public function actions() {
    $actions = parent::actions();
    unset($actions['index']);
    return $actions;
}
```

```
public function actionIndex() {
    $activeData = new ActiveDataProvider([
        'query' => \common\models\Yourmodel::find(),
        'pagination' => false
    ]);
    return $activeData;
}
```

Переопределить Content-Type для определенного действия

Случай использования: только одно действие, которое должно возвращать простой (текстовый) контент как есть:

```
public function actionAsXML()
{
    $this->layout = false;
    Yii::$app->response->format = Response::FORMAT_XML;

    return ['aaa' => [1, 2, 3, 4]];;
}
```

Предварительно определенные форматы ответов:

- FORMAT_HTML
- FORMAT_XML
- FORMAT_JSON
- FORMAT_JSONP
- FORMAT_RAW

Нет никакого типа mime для `text/plain` из коробки, используйте это вместо:

```
public function actionPlainText()
{
    $this->layout = false;
    Yii::$app->response->format = Response::FORMAT_RAW;
    Yii::$app->response->headers->add('Content-Type', 'text/plain');

    return $this->render('plain-text'); // outputs template as plain text
}
```

Прочитайте Восстановительный API онлайн: https://riptutorial.com/ru/yii2/topic/6102/восстановительный-api

глава 7: Загрузка файлов

Examples

Как это сделать

Загрузка файлов

Загрузка файлов в Yii обычно выполняется с помощью [[yii \ web \ UploadedFile]], который инкапсулирует каждый загруженный файл в качестве объекта `UploadedFile` . В сочетании с [[yii \ widgets \ ActiveForm]] и моделями вы можете легко реализовать безопасный механизм загрузки файлов.

Создание моделей

Подобно работе с текстовыми вводами, чтобы загрузить один файл, вы должны создать класс модели и использовать атрибут модели для хранения загруженного экземпляра файла. Вы также должны объявить правило проверки для проверки загрузки файла. Например,

```
namespace app\models;

use yii\base\Model;
use yii\web\UploadedFile;

class UploadForm extends Model
{
    /**
     * @var UploadedFile
     */
    public $imageFile;

    public function rules()
    {
        return [
            [['imageFile'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg'],
        ];
    }

    public function upload()
    {
        if ($this->validate()) {
            $this->imageFile->saveAs('uploads/' . $this->imageFile->baseName . '.' . $this-
>imageFile->extension);
            return true;
        } else {
            return false;
        }
    }
```

```
}
```

В приведенном выше `imageFile` атрибут `imageFile` используется для хранения экземпляра загруженного файла. Он связан с правилом проверки `file` который использует [[yii \ validators \ FileValidator]] для обеспечения загрузки файла с расширением `png` или `jpg` . Метод `upload()` выполнит проверку и сохранит загруженный файл на сервере.

`file` валидатор позволяет проверять расширения файлов, размер, тип MIME и т.д. Пожалуйста , обратитесь к разделу Основные валидаторов для более подробной информации.

> Совет. Если вы загружаете изображение, вы можете вместо этого использовать средство проверки `image` . Валидатор `image` реализуется с помощью [[yii \ validators \ ImageValidator]], который проверяет, получил ли атрибут допустимое изображение, которое затем может быть сохранено или обработано с помощью расширения Imagine .

Ввод файла изображения

Затем создайте входной файл в представлении:

```
<?php
use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['options' => ['enctype' => 'multipart/form-data']]) ?>

    <?= $form->field($model, 'imageFile')->fileInput() ?>

    <button>Submit</button>

<?php ActiveForm::end() ?>
```

Важно помнить, что вы добавляете опцию `enctype` в форму, чтобы файл мог быть правильно загружен. `fileInput()` будет отображать `<input type="file">` который позволит пользователям выбирать файл для загрузки.

> Совет. Начиная с версии 2.0.8, [[yii \ web \ widgets \ ActiveField :: fileInput | fileInput]] `enctype` добавляет параметр `enctype` в форму, когда используется поле ввода файла.

Проводка

Теперь в действии контроллера напишите код, чтобы подключить модель и представление для реализации загрузки файлов:

```
namespace app\controllers;
```

```
use Yii;
use yii\web\Controller;
use app\models\UploadForm;
use yii\web\UploadedFile;

class SiteController extends Controller
{
    public function actionUpload()
    {
        $model = new UploadForm();

        if (Yii::$app->request->isPost) {
            $model->imageFile = UploadedFile::getInstance($model, 'imageFile');
            if ($model->upload()) {
                // file is uploaded successfully
                return;
            }
        }

        return $this->render('upload', ['model' => $model]);
    }
}
```

В приведенном выше коде при отправке формы вызывается метод [[yii \ web \ UploadedFile :: getInstance ()]] для представления загруженного файла в качестве экземпляра `UploadedFile` . Затем мы полагаемся на проверку модели, чтобы убедиться, что загруженный файл является допустимым и сохранить файл на сервере.

Загрузка нескольких файлов

Вы также можете загружать сразу несколько файлов с некоторыми корректировками кода, указанного в предыдущих подразделах.

Сначала вы должны настроить класс модели, добавив параметр `maxFiles` в правило проверки `file` чтобы ограничить максимальное количество файлов, разрешенных для загрузки. Установка `maxFiles` в `0` означает, что количество файлов, которые могут быть загружены одновременно, не ограничено. Максимальное количество файлов, которые могут быть загружены одновременно, также ограничено директивой PHP `max_file_uploads` , которая по умолчанию равна 20. Метод `upload()` также должен быть обновлен для сохранения загруженных файлов по одному.

```
namespace app\models;

use yii\base\Model;
use yii\web\UploadedFile;

class UploadForm extends Model
{
    /**
     * @var UploadedFile[]
     */
    public $imageFiles;
```

```
    public function rules()
    {
        return [
            [['imageFiles'], 'file', 'skipOnEmpty' => false, 'extensions' => 'png, jpg',
'maxFiles' => 4],
        ];
    }

    public function upload()
    {
        if ($this->validate()) {
            foreach ($this->imageFiles as $file) {
                $file->saveAs('uploads/' . $file->baseName . '.' . $file->extension);
            }
            return true;
        } else {
            return false;
        }
    }
}
```

В файле просмотра вы должны добавить `multiple` вариантов в `fileInput()` чтобы поле загрузки файла могло получать несколько файлов:

```
<?php
use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['options' => ['enctype' => 'multipart/form-data']]) ?>

    <?= $form->field($model, 'imageFiles[]')->fileInput(['multiple' => true, 'accept' =>
'image/*']) ?>

    <button>Submit</button>

<?php ActiveForm::end() ?>
```

И, наконец, в действии контроллера вы должны вызвать `UploadedFile::getInstances()` вместо `UploadedFile::getInstance()` чтобы назначить массив экземпляров `UploadedFile` для `UploadForm::imageFiles` .

```
namespace app\controllers;

use Yii;
use yii\web\Controller;
use app\models\UploadForm;
use yii\web\UploadedFile;

class SiteController extends Controller
{
    public function actionUpload()
    {
        $model = new UploadForm();

        if (Yii::$app->request->isPost) {
            $model->imageFiles = UploadedFile::getInstances($model, 'imageFiles');
```

```
            if ($model->upload()) {
                // file is uploaded successfully
                return;
            }
        }

        return $this->render('upload', ['model' => $model]);
    }
}
```

Прочитайте Загрузка файлов онлайн: https://riptutorial.com/ru/yii2/topic/2221/загрузка-файлов

глава 8: Запрос Ajax

Examples

Отправка формы Ajax

Просмотреть файл:

```
<?php
use yii;
use yii\bootstrap\ActiveForm;
use yii\helpers\Html;
?>

<?php
$form = ActiveForm::begin([
    'action' => ['comments/ajax-comment'],
    'options' => [
        'class' => 'comment-form'
    ]
]);
?>
    <?= $form->field($model, 'comment'); ?>

    <?= Html::submitButton("Submit", ['class' => "btn"]); ?>

<?php ActiveForm::end(); ?>
```

Javascript:

```
 jQuery(document).ready(function($) {
       $(".comment-form").submit(function(event) {
            event.preventDefault(); // stopping submitting
            var data = $(this).serializeArray();
            var url = $(this).attr('action');
            $.ajax({
                url: url,
                type: 'post',
                dataType: 'json',
                data: data
            })
            .done(function(response) {
                if (response.data.success == true) {
                    alert("Wow you commented");
                }
            })
            .fail(function() {
                console.log("error");
            });

        });
    });
```

Действие контроллера:

```
public function actionAjaxComment()
{
    $model = new Comments();
    if (Yii::$app->request->isAjax) {
        Yii::$app->response->format = \yii\web\Response::FORMAT_JSON;

        if ($model->load(Yii::$app->requset->post()) && $model->save()) {
            return [
                'data' => [
                    'success' => true,
                    'model' => $model,
                    'message' => 'Model has been saved.',
                ],
                'code' => 0,
            ];
        } else {
            return [
                'data' => [
                    'success' => false,
                    'model' => null,
                    'message' => 'An error occured.',
                ],
                'code' => 1, // Some semantic codes that you know them for yourself
            ];
        }
    }
}
```

Просмотр рендера Ajax

`Controller::renderAjax()` может использоваться для ответа на запрос Ajax. Этот метод аналогичен renderPartial (), за исключением того, что он будет вводить результат рендеринга с помощью сценариев и файлов JS / CSS, которые регистрируются в представлении

Предположим, что у нас есть форма входа в файл вида:

```
<?php
use yii\helpers\Html;
use yii\bootstrap\ActiveForm;

\yii\bootstrap\BootstrapAsset::register($this);

<div class="site-login">

    <?php $form = ActiveForm::begin(); ?>

        <?= $form->field($model, 'username')->textInput() ?>

        <?= $form->field($model, 'password')->passwordInput() ?>

        <?= Html::submitButton('Login',['class' => 'btn btn-primary btn-block']) ?>

    <?php ActiveForm::end(); ?>
</div>
```

Когда мы используем `renderPartial()` в действии контроллера:

```
public function actionLogin()
{
    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        return $this->goBack();
    }
    return $this->renderPartial('login', [
        'model' => $model,
    ]);
}
```

Результат действия:

```
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
</div>
```

Когда мы используем `renderAjax()` в действии контроллера:

```
    ...
    return $this->renderAjax('login', [
        'model' => $model,
    ]);
    ...
```

Вывод действия (JS, CSS):

```
<link href="/assets/f1759119/css/bootstrap.css" rel="stylesheet">
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
```

```
</div>
<script src="/assets/13aa7b5d/jquery.js"></script>
<script src="/assets/302a2946/yii.js"></script>
<script src="/assets/302a2946/yii.validation.js"></script>
<script src="/assets/302a2946/yii.activeForm.js"></script>
```

Если мы хотим исключить некоторые активы из представления (чтобы предотвратить дублирование):

```
...
Yii::$app->assetManager->bundles = [
    'yii\bootstrap\BootstrapAsset' => false,
];
return $this->renderAjax('login', [
    'model' => $model,
]);
...
```

Результат действия (нет bootstrap.css):

```
<div class="site-login">
    <form id="w0" action="/site/login" method="post" role="form">
        <div class="form-group field-loginform-username required">
            <label class="control-label" for="loginform-username">Имя пользователя</label>
            <input type="text" id="loginform-username" class="form-control"
name="LoginForm[username]">
        </div>
        <div class="form-group field-loginform-password required">
            <label class="control-label" for="loginform-password">Пароль</label>
            <input type="password" id="loginform-password" class="form-control"
name="LoginForm[password]">
        </div>
        <button type="submit" class="btn btn-primary btn-block">Login</button>
    </form>
</div>
<script src="/assets/13aa7b5d/jquery.js"></script>
<script src="/assets/302a2946/yii.js"></script>
<script src="/assets/302a2946/yii.validation.js"></script>
<script src="/assets/302a2946/yii.activeForm.js"></script>
```

Прочитайте Запрос Ajax онлайн: https://riptutorial.com/ru/yii2/topic/2944/запрос-ajax

глава 9: Календарь YII2 JQuery для текстового поля

Examples

Добавить календарь jquery для текстового поля с max как текущая дата

Если мы хотим отобразить календарь jquery для конечного пользователя, который может выбрать максимальную дату в качестве текущей даты в календаре. Следующий код будет полезен для этого сценария.

```
<?php
use yii\jui\DatePicker;
use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....
<?= $form->field($model, 'date_of_birth')->widget(DatePicker::classname(), ['dateFormat' =>
'php:M d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'maxDate' => '+0d']]) ?>
.....
<?php ActiveForm::end(); ?>
```

Добавить календарь jquery для текстового поля с указанием даты min

Для некоторых форм вы хотите отображать дни из будущего / прошлых дней, а в другие дни необходимо отключить, то этот сценарий поможет.

```
<?php
use yii\jui\DatePicker;
use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....

<?php
$day = '+5d'; //if you want to display +5 days from current date means for future days.
#(or)
$day = '-5d'; //if you want to display -5 days from current date means older days.
?>
<?= $form->field($model, 'date_of_birth')->widget(DatePicker::classname(), ['dateFormat' =>
'php:M d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'maxDate' => $day]]) ?>
.....
<?php ActiveForm::end(); ?>
```

Добавьте календарь jquery с даты и до даты

Если вы хотите иметь календарь с даты и на сегодняшний день, а также на календарные дни, всегда будет больше, чем от поля даты, то ниже сценария поможет.

```
<?php $form = ActiveForm::begin(['id' => 'profile-form']); ?>
.....
<?= $form->field($model, 'from_date')->widget(DatePicker::classname(), ['dateFormat' => 'php:M
d, Y', 'options' => ['readonly' => true], 'clientOptions' => [ 'changeMonth' => true,
'changeYear' => true, 'yearRange' => '1980:'.date('Y'), 'onSelect' => new
yii\web\JsExpression('function(selected) { var dt = new Date(selected);
dt.setDate(dt.getDate() + 1); $("#filter-date-to").datepicker("option", "minDate", dt); }')]])
?>

<?= $form->field($model, 'to_date')->widget(DatePicker::classname(), ['dateFormat' => 'php:M
d, Y', 'options' => ['readonly' => true, 'id' => 'filter-date-to'], 'clientOptions' => [
'changeMonth' => true, 'changeYear' => true, 'yearRange' => '1980:'.date('Y')]]) ?>
.....
<?php ActiveForm::end(); ?>
```

Прочитайте Календарь YII2 JQuery для текстового поля онлайн:
https://riptutorial.com/ru/yii2/topic/6366/календарь-yii2-jquery-для-текстового-поля

глава 10: Компоненты

Examples

Создание и использование компонентов приложения

Шаги по созданию компонента:

- Создайте папку с именами `components` в корневой папке проекта
- Создайте свой компонент внутри папки компонентов, например: `MyComponent.php`

```
namespace app\components;

use Yii;
use yii\base\Component;
use yii\base\InvalidConfigException;


class MyComponent extends Component
{
    public function demo()
    {
        return "welcome";
    }
}
```

- Зарегистрируйте свой компонент внутри файла `config/web.php`

```
components' => [
    'mycomponent' => [
        'class' => 'app\components\MyComponent',
    ],
]
```

Теперь вы можете использовать свой компонентный метод:

```
namespace app\controllers;

use Yii;

class DemoController extends \yii\web\Controller
{
    public function actionTest()
    {
        echo Yii::$app->mycomponent->demo();
    }
}
```

Выпадающий список с использованием функции компонента

Создать функцию в MyComponent.php

```
    namespace app\components;

    use Yii;
    use yii\base\Component;
    use yii\base\InvalidConfigException;
    use yii\helpers\Url;
    use yii\helpers\ArrayHelper;

    use app\models\User;

        class MyComponent extends Component
        {

          // Function return list of  id & user Names,used for dropdownlist
          public function getUserID()
          {
            $code = User::find()->select('id,name')
            ->where(['is_deleted'=>'n'])
            ->all();

            $result = ArrayHelper::map($code, 'id', 'name');
            if($result)
                return $result;
            else
                return ["null"=>"No User"];
          }
        }
```

-> Регистрация компонента в web.php

```
components' => [
        'mycomponent' => [
             'class' => 'app\components\MyComponent',
            ],
         ]
```

-> используйте его в своем представлении

```
<?= $form->field($model, 'user_id')->dropDownList(Yii::$app->mycomponent->getUserID())?>
```

Прочитайте Компоненты онлайн: https://riptutorial.com/ru/yii2/topic/2217/компоненты

глава 11: Маршрутизация и URL-адреса

замечания

Все URL-адреса должны быть созданы с помощью помощника `yii\helpers\Url` это очень помогает вам, если вы решите изменить правила URL-адреса в urlManager.

Examples

Создание URL-адресов

Helper yii \ helpers \ Url предоставляет набор статических методов для управления URL-адресами. Этот помощник может использоваться в коде представлений / контроллеров.

URL-адрес маршрута:

```
echo Url::to(['post/index']);
```

URL-адрес маршрута с параметрами:

```
echo Url::to(['post/view', 'id' => 100]);
```

привязанный URL:

```
echo Url::to(['post/view', 'id' => 100, '#' => 'content']);
```

абсолютный URL:

```
echo Url::to(['post/index'], true);
```

абсолютный URL-адрес с использованием схемы https:

```
echo Url::to(['post/index'], 'https');
```

Примечание . Маршрут, переданный в метод `Url::to()` чувствителен к контексту. Он может использовать текущий модуль и текущий контроллер. Например, предположим, что текущий модуль является `admin` а текущий контроллер - `post` :

относительный маршрут с идентификатором действия (не содержит никаких косых черт):

```
echo Url::to(['index']);    // -->> '/index.php?r=admin%2Fpost%2Findex'
```

относительный маршрут (не имеет косой черты):

```
echo Url::to(['post/index']);    // -->> '/index.php?r=admin%2Fpost%2Findex'
```

абсолютный маршрут (начинается с косой черты):

```
echo Url::to(['/post/index']);    // -->> '/index.php?r=post%2Findex'
```

текущий запрошенный URL:

```
echo Url::to();
echo Url::to(['']);
```

Чтобы создать URL, основанный на **текущем маршруте,** и **параметры GET** используют Url :: current () .

Предположим, что `$_GET = ['id' => 10, 'page' => 7]` , текущий маршрут - `post/view` .

текущий URL:

```
echo Url::current();    // -->> '/index.php?r=post%2Fview&id=10&page=7'
```

текущий URL без параметра `page` :

```
echo Url::current(['page' => null]);  // -->> '/index.php?r=post%2Fview&id=10'
```

текущий URL с измененным параметром `page` :

```
echo Url::current(['page' => 12]);    // -->> '/index.php?r=post%2Fview&id=10&page=12'
```

Прочитайте Маршрутизация и URL-адреса онлайн: https://riptutorial.com/ru/yii2/topic/5510/маршрутизация-и-url-адреса

глава 12: Миграции баз данных

Examples

Создание мигрантов

```
yii migrate/create <name>
```

Необходимый аргумент имени дает краткое описание новой миграции. Например, если миграция связана с созданием новой таблицы с именем news, вы можете использовать имя create_news_table и запустить следующую команду

```
yii migrate/create create_news_table
```

Пример файла миграции

```
<?php

use yii\db\Migration;

class m150101_185401_create_news_table extends Migration
{
public function up()
{

}

public function down()
{
    echo "m101129_185401_create_news_table cannot be reverted.\n";

    return false;
}

/*
// Use safeUp/safeDown to run migration code within a transaction
public function safeUp()
{
}

public function safeDown()
{
}
*/
}
```

Падение таблицы

```
public function up()
{
```

```
    $this->dropTable('post');
}
```

Создайте поля таблицы сразу

```
yii migrate/create create_post_table --fields="title:string,body:text"
```

Формирует:

```
/**
 * Handles the creation for table `post`.
 */
class m150811_220037_create_post_table extends Migration
{
/**
 * @inheritdoc
 */
public function up()
{
    $this->createTable('post', [
        'id' => $this->primaryKey(),
        'title' => $this->string(),
        'body' => $this->text(),
    ]);
}

/**
 * @inheritdoc
 */
public function down()
{
    $this->dropTable('post');
}
}
```

Создать таблицу

```
public function up()
{
    $this->createTable('post', [
        'id' => $this->primaryKey()
    ]);
}
```

Drop / Rename / Alter Column

```
public function up()
{
    $this->dropColumn('post', 'position');

    $this->renameColumn('post', 'owner_id', 'user_id');

    $this->alterColumn('post', 'updated', $this->timestamp()->notNull()->defaultValue('0000-
00-00 00:00:00'));
```

```
}
```

Добавить колонку

```
public function up()
{
    $this->addColumn('post', 'position', $this->integer());
}
```

Возвращение мигрантов

```
yii migrate/down     # revert the most recently applied migration
yii migrate/down 3   # revert the most 3 recently applied migrations
```

Транзакционные миграции

```
public function safeUp()
{
    $this->createTable('news', [
        'id' => $this->primaryKey(),
        'title' => $this->string()->notNull(),
        'content' => $this->text(),
    ]);

    $this->insert('news', [
        'title' => 'test 1',
        'content' => 'content 1',
    ]);
}

public function safeDown()
{
    $this->delete('news', ['id' => 1]);
    $this->dropTable('news');
}
```

Еще более простой способ реализации транзакционных миграций - установить код миграции в `safeUp()` и `safeDown()` . Эти два метода отличаются от `up()` и `down()` тем, что они неявно заключены в транзакции. В результате, если какая-либо операция в этих методах выходит из строя, все предыдущие операции будут автоматически откатываться.

Перенос нескольких баз данных

По умолчанию миграции применяются к той же базе данных, которая указана компонентом приложения db. Если вы хотите, чтобы их применяли к другой базе данных, вы можете указать опцию командной строки db, как показано ниже:

```
yii migrate --db=db2
```

Повторная миграция

```
yii migrate/redo        # redo the last applied migration
yii migrate/redo 3      # redo the last 3 applied migrations
```

Миграция листинга

```
yii migrate/history     # showing the last 10 applied migrations
yii migrate/history 5   # showing the last 5 applied migrations
yii migrate/history all # showing all applied migrations

yii migrate/new         # showing the first 10 new migrations
yii migrate/new 5       # showing the first 5 new migrations
yii migrate/new all     # showing all new migrations
```

Изменение истории миграции

```
yii migrate/mark 150101_185401                      # using timestamp to specify the migration
yii migrate/mark "2015-01-01 18:54:01"              # using a string that can be parsed by
strtotime()
yii migrate/mark m150101_185401_create_news_table   # using full name
yii migrate/mark 1392853618                         # using UNIX timestamp
```

Применение миграции

```
yii migrate
```

Эта команда перечислит все миграции, которые пока не были применены. Если вы подтвердите, что хотите применить эти миграции, он будет запускать метод up () или safeUp () в каждом новом классе миграции один за другим в порядке их значений метки времени. Если какая-либо миграция завершится неудачей, команда завершит работу без применения остальных миграций.

```
yii migrate 3
yii migrate/to 150101_185401                      # using timestamp to specify the migration
yii migrate/to "2015-01-01 18:54:01"              # using a string that can be parsed by
strtotime()
yii migrate/to m150101_185401_create_news_table   # using full name
yii migrate/to 1392853618                         # using UNIX timestamp
```

Прочитайте Миграции баз данных онлайн: https://riptutorial.com/ru/yii2/topic/1929/миграции-баз-данных

глава 13: Печенье

замечания

Куки-файлы являются частью HTTP-запроса, поэтому неплохо сделать и то, и другое в контроллере, ответственность за который точно связана с запросом и ответом.

Examples

Настройка файла cookie

Чтобы установить файл cookie, т. Е. Создать его и запланировать отправку в браузер, вам нужно создать новый экземпляр класса `\yii\web\Cookie` и добавить его в коллекцию файлов ответов:

```
$cookie = new Cookie([
    'name' => 'cookie_monster',
    'value' => 'Me want cookie!',
    'expire' => time() + 86400 * 365,
]);
\Yii::$app->getResponse()->getCookies()->add($cookie);
```

В приведенном выше примере мы передаем параметры конструктору класса cookie. Они в основном такие же, как и с встроенной функцией setcookie PHP:

- `name` - имя файла cookie.
- `value` - значение cookie. Убедитесь, что это строка. Браузеры обычно недовольны двоичными данными в файлах cookie.
- `domain` - домен, для которого вы настраиваете cookie.
- `expire` - отметка времени unix, указывающая время, когда файл cookie должен быть автоматически удален.
- `path` - путь на сервере, на котором будет доступен cookie.
- `secure` - если `true` , cookie будет установлен только в том случае, если используется HTTPS.
- `httpOnly` - если `true` , cookie будет недоступен через JavaScript.

Чтение файла cookie

Чтобы прочитать файл cookie, используйте следующий код:

```
$value = \Yii::$app->getRequest()->getCookies()->getValue('my_cookie');
```

Примечание. Этот код позволяет читать файлы cookie, которые были установлены с

использованием компонента cookie (поскольку он по умолчанию определяет все файлы cookie). Поэтому, если вы добавляете / обновляете cookie с использованием JS-кода, вы не можете его прочитать с помощью этого метода (по умолчанию).

Куки для субдоменов

Из-за соображений безопасности куки-файлы по умолчанию доступны только в том же домене, из которого они были установлены. Например, если вы установили cookie на домен `example.com` , вы не можете получить его в домене `www.example.com` . Поэтому, если вы планируете использовать субдомены (например, admin.example.com, profile.example.com), вам необходимо явно указать `domain` :

```
$cookie = new Cookie([
    'name' => 'cookie_monster',
    'value' => 'Me want cookie everywhere!',
    'expire' => time() + 86400 * 365,
    'domain' => '.example.com' // <<<=== HERE
]);
\Yii::$app->getResponse()->getCookies()->add($cookie);
```

Теперь cookie можно прочитать из всех поддоменов `example.com` .

Кросс-субдоменная аутентификация и файлы cookie идентичности

В случае автологина или «помнить меня» cookie применяются те же причуды, что и в случае куки-субдомена. Но на этот раз вам нужно настроить пользовательский компонент, установив массив `identityCookie` в желаемую конфигурацию cookie.

Откройте файл конфигурации приложения и добавьте параметры `identityCookie` в конфигурацию пользовательских компонентов:

```
$config = [
    // ...
    'components' => [
        // ...
        'user' => [
            'class' => 'yii\web\User',
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
            'loginUrl' => '/user/login',
            'identityCookie' => [ // <---- here!
                'name' => '_identity',
                'httpOnly' => true,
                'domain' => '.example.com',
            ],
        ],
        'request' => [
            'cookieValidationKey' => 'your_validation_key'
        ],
        'session' => [
            'cookieParams' => [
                'domain' => '.example.com',
```

```
                'httpOnly' => true,
            ],
        ],

    ],
];
```

Обратите внимание, что `cookieValidationKey` должен быть одинаковым для всех поддоменов.

Обратите внимание, что вам необходимо настроить свойство `session::cookieParams` чтобы иметь samedomain как ваш `user::identityCookie` чтобы обеспечить работу `login` и `logout` для всех поддоменов. Такое поведение лучше поясняется в следующем разделе.

Параметры cookie сеанса

Параметры cookie сеанса важны как в том случае, если вам нужно поддерживать сеанс при получении от одного субдомена к другому, либо, если наоборот, вы размещаете backend-приложение под `/admin` URL и хотите обрабатывать сеанс отдельно.

```
$config = [
    // ...
    'components' => [
        // ...
        'session' => [
            'name' => 'admin_session',
            'cookieParams' => [
                'httpOnly' => true,
                'path' => '/admin',
            ],
        ],
    ],
];
```

Прочитайте Печенье онлайн: https://riptutorial.com/ru/yii2/topic/2945/печенье

глава 14: Пользовательские проверки

Вступление

У Yii2 есть встроенные валидаторы, которые можно использовать при решении связанных с программированием или общих проверках на щенки. Когда вам нужно создать новую проверку бизнес-логики, вам нужно создать свои собственные валидаторы.

Examples

Типы проверок

Давайте сначала рассмотрим основные типы пользовательских валидаторов:

1. Встроенный валидатор
2. Автономный валидатор

Inline Validator : это тип валидатора, который мы создаем внутри класса, который в основном метод, который мы определяем, как и другие методы, но с дополнительными параметрами, которые передаются Yii2.

```
....
public function ValidateMyBusiness($attr, $params){
    // adding an error here means our validation is failed.
    if ($this->{$attr} > 1100) {
        $this->addError($attr, "Some error occured");
    }
}
...
// calling above validator is simple as below:
public function rules(){
  return [
     ['money', 'validateMyBusiness', 'params' => ['targetAccount' => $this->account]];
  ]
}

# params array will be passed to our inline parameter as a second argument.
```

Прочитайте Пользовательские проверки онлайн: https://riptutorial.com/ru/yii2/topic/9187/пользовательские-проверки

глава 15: Проверка

Examples

Проверка уникального значения из базы данных в Yii2

Мало кто знает, что сообщение об ошибке не отображается, если существующее значение вводится в текстовое поле.

Итак, для примера я *не разрешаю* пользователю вводить *существующий адрес электронной почты* .

signup.php

(Страница, где вы хотите зарегистрировать нового пользователя без существующего идентификатора электронной почты)

1. Удалите `use yii\bootstrap\ActiveForm;` (если представить)
2. Добавьте `use yii\widgets\ActiveForm;`
3. Добавьте `'enableAjaxValidation' => true` (В этом поле, где вы хотите, чтобы пользователь не вводил существующий идентификатор электронной почты.)

```
<?php
use yii\bootstrap\ActiveForm;
use yii\widgets\ActiveForm;
?>

<?= $form->field($modelUser, 'email',['enableAjaxValidation' => true])
    ->textInput(['class'=>'form-control','placeholder'=>'Email']); ?>
```

контроллер

Добавьте эти строки сверху `use yii\web\Response;use yii\widgets\ActiveForm;`

```
<?php
use yii\web\Response;
use yii\widgets\ActiveForm;

  .
  .// Your code
  .

  public function actionSignup() {

    $modelUser = new User();

    //Add This For Ajax Email Exist Validation
    if(Yii::$app->request->isAjax && $modelUser->load(Yii::$app->request->post())){
      Yii::$app->response->format = Response::FORMAT_JSON;
      return ActiveForm::validate($modelUser);
```

```
    }
    else if ($model->load(Yii::$app->request->post())) {

    }

  }
?>
```

модель

```
[['email'],'unique','message'=>'Email already exist. Please try another one.'],
```

Проверка уникальной ценности из базы данных: уникальная проверка

У некоторых людей возникают проблемы с отображением сообщений об ошибках, если вводится существующее значение. Например, я не разрешаю пользователям регистрироваться с существующим электронным письмом.

Посмотреть

```
<?php
.....................

    <?= $form->field($modelUser, 'email')->textInput(['class'=>'form-
control','placeholder'=>'Email']) ?>
.....................
```

контроллер

```
<?php
use yii\web\Response; // important lines
use yii\widgets\ActiveForm; // important lines

.
.// Your code
.

public function actionSignup()
{

    $modelUser = new User();

    //Add This For Ajax Validation
    if(Yii::$app->request->isAjax && $modelUser->load(Yii::$app->request->post())){
        Yii::$app->response->format = Response::FORMAT_JSON;
        return ActiveForm::validate($modelUser);
    }
    if ($modelUser->load(Yii::$app->request->post()) && $modelUser->save()) {
        return $this->redirect(['someplace nice']);
    }
    return $this->render('update', [
        'modelUser' => $modelUser,
    ]);
}
```

модель

```
public function rules()
{
    return [
        ..............
        ['email', 'unique', 'message'=>'Email already exist. Please try another one.'],
        ...........
    ]
}
```

Отключить сообщение об ошибке проверки по фокусу / клавише вверх

По умолчанию появится следующее сообщение об ошибке `textbox` в `<div class="help-block"></div>` на *KeyUp* или *после нажатия кнопки отправки* , если какое - либо ограничение проверки не выполняется.

Иногда мы хотим, чтобы сообщение было `onKeyup` . `onKeyup` Не было подтверждения на событие `onKeyup` .

Давайте проверим `yii2/widgets/ActiveForm.php` :

```
<?php

namespace yii\widgets;

use Yii;
use yii\base\InvalidCallException;
use yii\base\Widget;
use yii\base\Model;
use yii\helpers\ArrayHelper;
use yii\helpers\Url;
use yii\helpers\Html;
use yii\helpers\Json;

class ActiveForm extends Widget
{
  public $action = '';
  public $method = 'post';
  public $options = [];
  .
  .
  .
  public $validateOnSubmit = true;
  public $validateOnChange = true;
  public $validateOnBlur = true;
  public $validateOnType = false;

  .
  .
  .

}
```

Там мы видим, что для `$validateOnBlur` по умолчанию установлено значение `true` . Изменение

файлов фреймов очень плохо, поэтому мы должны переопределить его при использовании формы:

```
<?php $form = ActiveForm::begin([ 'id' => 'register-form','validateOnBlur' => false]); ?>
```

Сценарий проверки

С помощью сценария вы можете выполнять валидацию в разных ситуациях

Определение сценария в классе модели

```
    class User extends \yii\db\ActiveRecord
    {
      public static function tableName()
      {
          return 'user_master';
      }

    // define validation in rule() function
  public function rules()
  {
    return [
      [['email_id'],'email'],
      [['first_name',],'required','on'=>['create','update']],  // create scenario
      [['email_id',],'required','on'=> ['admin','create','update','forgotpassword']],
      [['mobile',],'required','on'=>['admin','create','update']],
    ];
  }
}
```

Использовать сценарий в контроллере

```
public function actionCreate()
{
  $model = new User();
  $model->scenario="create";  // use create scenario, create scenario validaion applied in
this model


}
public function actionUpdate()
{
    $model = new User();
    $model->scenario="update";  // use update scenario, update scenario validaion applied in
this model
}
```

Проверить массив

Начиная с версии Yii2 версии 2.0.4, для каждого элемента массива используется параметр EveryValidator.

```
[
```

```
    // ... other rules
    ['userIDs', 'each', 'rule' => ['integer']],
]
```

Часть `['integer']` может быть любым другим объектом валидатора, который предлагает Yii2, и может содержать конкретные аргументы для валидатора. Например: `['integer', 'min' => 1337]` . Если идентификаторы пользователя не содержат массив, проверка правильности не будет выполнена.

Если вы просто хотите узнать, содержит ли атрибут массив без проверки содержимого, вы можете написать собственный валидатор.

```
[
    ['myAttr', function($attribute, $params) {
        if (!is_array($this->$attribute)) {
            $this->addError($attribute, "$attribute isn't an array!");
        }
    }]
]
```

Прочитайте Проверка онлайн: https://riptutorial.com/ru/yii2/topic/839/проверка

глава 16: Работа с базами данных

Examples

Использование построителя запросов Yii2

Yii2 предоставляет эффективные способы извлечения данных из базы данных. ***Рассмотрим*** пример простой таблицы сотрудников с полями ***emp_id, emp_name и emp_salary*** . Чтобы получить имена сотрудников и их зарплаты, мы используем запрос.

```
select emp_name,emp_salary from employee
```

Чтобы сгенерировать вышеуказанный запрос в Yii2, существует множество методов. Один из методов заключается в использовании объекта ***yii \ db \ Quer*** *y* .

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();

$rows=$query->select(['emp_name','emp_salary']) //specify required columns in an array
             ->from('employee') //specify table name
             ->all(); //returns an array of rows with each row being an associative array of
name-value pairs.
```

Мы можем использовать цикл foreach для циклического прохождения каждой пары имя-значение в массиве ***$ rows*** .

```
foreach ($rows as $row) {
    echo "Employee Name: ".$row['emp_name'].",Employee Salary: ".$row['emp_salary']."<br>";
}
```

Это приведет к выводу

> Имя сотрудника: Kiran, Employee Зарплата: 25000
>
> Имя сотрудника: Midhun, Employee Зарплата: 50000
>
> Имя сотрудника: Jishnu, Employee Зарплата: 20000
>
> Имя сотрудника: Ajith, Employee Зарплата: 25000
>
> Имя сотрудника: Akshay, Employee Зарплата: 750000

Другие примеры

Предположим, нам нужно найти имя сотрудников, чья зарплата равна 25000. Мы можем написать запрос в sql как

```
select emp_name from employee where salary=25000
```

В Yii2 код для генерации вышеуказанного запроса

```
$query=new \yii\db\Query();

$rows=$query->select(['emp_name'])
            ->from('employee')
            ->where(['emp_salary'=>25000]) //specify the condition as an associative array
where key is column name
            ->all();
```

Если нам нужно найти имена сотрудников, чья зарплата больше 25000, мы можем написать код в Yii2 как

```
  $rows=$query->select(['emp_name'])
        ->from('employee')
        ->where(['>','emp_salary', 25000])
//Here first element in the above array specify relational operator used, second element
specify the table name and third the value itself.
        ->all();
```

Проверка состояния с использованием where ()

Несколько условий могут быть записаны с использованием метода **where (),** как показано ниже.

```
// Creates a new \yii\db\Query() object
$query = new \yii\db\Query();
$rows = $query->select(['emp_name','emp_salary'])
        ->from('employee')
        ->where(['emp_name' => 'Kiran', 'emp_salary' => 25000]) // Specify multiple conditions
        ->one(); // Returns the first row of the result
```

Вышеприведенный код получит сотрудника с именем **kiran** и зарплатой **25000** . Если несколько сотрудников удовлетворяют вышеуказанному условию, вызов ***one ()*** гарантирует получение только первого результата. Чтобы получить все результаты, вы должны использовать ***all ()*** .

Обратите внимание: если вы используете ***all (),*** результат всегда будет массивом; Даже если есть только один или нулевой результат. Этот массив содержит все результаты как массивы или пуст, когда никакие записи не совпадают. Вызов ***one ()*** вернет результирующий массив напрямую или false, если запрос ничего не возвращает.

Ниже приведен эквивалентный код в sql.

```
select emp_name, emp_salary from employee where emp_name = 'Kiran' and emp_salary = 25000
limit 1;
```

Ниже приведен альтернативный способ написания вышеуказанного запроса в Yii2.

```
$rows = $query->select(['emp_name', 'emp_salary'])
    ->from('employee')
    ->where(['emp_name' => 'Kiran'])
    ->andWhere(['emp_salary' => 25000])
    ->one();
```

Дополнительный набор условий можно задать с помощью **andWhere** . Это будет полезно, если нам нужно добавить дополнительную проверку условий в запрос позже.

Еще один способ указать несколько условий - использовать **формат оператора метода where ().** Вышеприведенный запрос также может быть записан как указано ниже.

```
 $rows = $query->select(['emp_name','emp_salary'])
    ->from('employee')
    ->where(['and', 'emp_name="kiran"', 'emp_salary=25000'])
    ->one();
```

Здесь мы указываем оператор ' **и** ' как первый элемент в массиве. Точно так же мы можем также использовать « **или** », « **между** », « **не между** », « **in** », « **not in** », « **like** », « **like** », « **not like** », « **или не нравится** », « **существует** », , ' **не существует** ', ' **>** ', ' **<=** ' и т. д. как операторы.

Примеры использования 'in' и 'like'

Предположим, нам нужно найти сотрудников, имеющих зарплаты **20000, 25000 и 50000** . В обычном sql мы будем писать запрос как

```
select * from employee where salary in (20000,25000,50000)
```

В Yii2 мы можем написать это, как показано ниже.

```
$rows = $query->from('employee')
        ->where(['emp_salary' => [20000,25000,50000]])
        ->all();
```

Другой способ указать одно и то же условие:

```
$rows = $query->from('employee')
    ->where(['in', 'emp_salary', [20000,25000,50000]]) // Making use of operator format of
where() method
    ->all();
```

Аналогично « **не в** » можно указать вместо « **в** », если мы хотим, чтобы все сотрудники не имели зарплаты 20000, 25000 и 50000.

Теперь давайте рассмотрим некоторые примеры использования « **like** » внутри where (). Предположим, нам нужно найти всех сотрудников, имеющих строку « **гопал** » от их имени.

Имена могут быть venugopal, rajagopal, gopalakrishnan и т. Д. Запрос sql приведен ниже.

```
select * from employee where emp_name like '%gopal%'
```

В Yii2 мы будем писать это как

```
 $rows = $query->from('employee')
        ->where(['like', 'emp_name', 'gopal']) // Making use of operator format of where()
method
        ->all();
```

Если нам нужно найти всех сотрудников, у которых есть строка « **gopal** » и « **nair** » от их имени. Мы можем написать как

```
   $rows = $query->from('employee')
        ->where(['like', 'emp_name', ['gopal','nair']]) // Making use of operator format of
where() method
        ->all();
```

Это будет оцениваться как

> выберите * у сотрудника, где emp_name, например «% gopal%» и «% nair%»,

Аналогичным образом мы можем использовать « **не похоже** », чтобы указать всем сотрудникам, не имеющим в своих именах строку « **гопал** » и « **нир** ».

Использование orderBy ()

Метод orderBy () указывает фрагмент ORDER BY SQL-запроса. Например, рассмотрим нашу таблицу сотрудников с полями **emp_id, emp_first_name, emp_last_name и emp_salary.** Предположим, нам нужно заказать результат, увеличив порядок зарплат сотрудников. Мы можем сделать это в sql, как указано ниже.

```
Select * from employee order by emp_salary
```

В yii2 мы можем построить запрос, как указано ниже

```
 //creates a new \yii\db\Query() object
    $query=new \yii\db\Query();

    $rows= $query->from('employee')->orderBy([
    'emp_salary' => SORT_ASC //specify sort order ASC for ascending DESC for descending
    ])->all();
```

Если нам нужно заказать сотрудников с их именем в порядке возрастания, а затем их зарплату в порядке убывания, мы можем написать это в обычном sql следующим образом.

```
Select * from employee order by emp_first_name ASC, emp_salary DESC
```

Эквивалентный sql можно построить с помощью yii2 следующим образом

```
//creates a new \yii\db\Query() object
    $query=new \yii\db\Query();

    $rows= $query->from('employee')->orderBy([
    'emp_first_name' => SORT_ASC
    'emp_salary' => SORT_DESC
    ])->all();
```

Вы также можете указать ORDER BY, используя строку, точно так же, как и при написании необработанных SQL-операторов. Например, вышеупомянутый запрос также может быть сгенерирован, как указано ниже.

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();
$rows=$query->from('employee')->orderBy('emp_first_name ASC, emp_salary DESC')->all();
```

Вы можете вызвать addOrderBy (), чтобы добавить дополнительные столбцы в фрагмент ORDER BY. Например

```
//creates a new \yii\db\Query() object
$query=new \yii\db\Query();
$rows=$query->from('employee')->orderBy('emp_first_name ASC')
    ->addOrderBy('emp_salary DESC')->all();
```

Прочитайте Работа с базами данных онлайн: https://riptutorial.com/ru/yii2/topic/4167/работа-с-базами-данных

глава 17: Расширенный шаблон проекта

Examples

Развертывание в среде совместного размещения

Развертывание расширенного шаблона проекта на общий хостинг немного сложнее, чем базовый, поскольку он имеет два веб-ресурса, которые не поддерживают веб-серверы с общим хостингом. Нам нужно будет отрегулировать структуру каталогов, поэтому URL-адрес интерфейса будет http: //site.local, а URL-адрес бэкэнд будет http: //site.local/admin .

Перенос сценариев ввода в один веб-корень

Прежде всего нам нужен каталог webroot. Создайте новый каталог и назовите его в соответствии с вашим именем веб-сайта хостинга, например, www или public_html или тому подобное. Затем создайте следующую структуру, где www - это созданный вами веб-сайт хостинга webroot:

```
www
    admin
backend
common
console
environments
frontend
...
```

> **www** будет нашим интерфейсом, поэтому переместите содержимое **интерфейса / веб-страницы** в него. Переместите содержимое **backend / web** в **www / admin** . В каждом случае вам нужно будет настроить пути в **index.php** и **index-test.php** .

Настройка сеансов и файлов cookie

Первоначально бэкэнд и интерфейсы предназначены для работы в разных доменах. Когда мы переместим все это в один домен, интерфейс и бэкэнд будут использовать одни и те же файлы cookie, создавая столкновение. Чтобы исправить это, настройте **backend / config / main.php** бэкэнд-приложение следующим образом:

```
'components' => [
    'request' => [
        'csrfParam' => '_csrf-backend',
        'csrfCookie' => [
            'httpOnly' => true,
            'path' => '/admin',
        ],
```

```
    ],
    'user' => [
        'identityClass' => 'common\models\User',
        'enableAutoLogin' => true,
        'identityCookie' => [
            'name' => '_identity-backend',
            'path' => '/admin',
            'httpOnly' => true,
        ],
    ],
    'session' => [
        // this is the name of the session cookie used for login on the backend
        'name' => 'advanced-backend',
        'cookieParams' => [
            'path' => '/admin',
        ],
    ],
],
```

Надеемся, что это поможет пользователям с общим хостингом развернуть расширенное приложение.

> кредиты: https://github.com/yiisoft/yii2-app-advanced/blob/master/docs/guide/topic-shared-hosting.md

Совместное использование загруженных файлов между интерфейсом и бэкэнд с помощью символических ссылок

Таким образом, вы загрузили свои файлы в папку say `/backend/web/uploads/` и вы хотите, чтобы эти загрузки отображались также на интерфейсе. Самый простой вариант - создать символическую ссылку в интерфейсе, которая ссылается на бэкэнд:

```
ln -s /path/to/backend/web/uploads/ /path/to/frontend/web/uploads
```

В ваших представлениях теперь можно использовать относительные ссылки на файлы:

```
<img src='/uploads/<?= $model->image?>' alt='My Image goes here'>
<a href='/uploads/<?= $model->filename?>' target='_blank'>Download File</a>
```

Убедитесь, что ваш веб-сервер позволяет использовать символические ссылки.

Прочитайте Расширенный шаблон проекта онлайн: https://riptutorial.com/ru/yii2/topic/944/расширенный-шаблон-проекта

глава 18: сессия

Examples

Сессия в yii2

импортировать класс сеанса

```
use  yii\web\Session;
```

Создать сеанс

```
$session = Yii::$app->session;
$session->open(); // open a session
$session->close();  // close a session
```

Сохраните значение в переменной сеанса.

```
$session = Yii::$app->session;

$session->set('name', 'stack');
OR
$session['name'] = 'stack';
OR
$_SESSION['name'] = 'stack';
```

Получите значение из переменной сеанса.

```
$name = $session->get('name');
OR
$name = $session['name'];
```

Удалить переменную сеанса

```
$session->remove('name');
OR
unset($session['name']);
OR
unset($_SESSION['name']);

$session->destroy(); // destroy all session
```

Удалить все переменные сеанса

```
$session->removeAll();
```

Проверка переменной сеанса

```
$session->has('name')
OR
isset($session['name'])
//both function return boolean value [true or false]
```

Вспышка сессии

Установка сеансовой вспышки

```
$session = Yii::$app->session;
$session->setFlash('error', 'Error in login');
```

Получить сеанс flash

```
echo $session->getFlash('error');
```

Проверка всплывающей подсказки

```
$result = $session->hasFlash('error');
```

Удаление сеанса

```
$session->removeFlash('error');
```

Удалить все временные переменные сеанса

```
$session->removeAllFlashes();
```

Непосредственно использовать переменную сеанса

Установить и получить переменную сеанса

```
\Yii::$app->session->set('name','stack');
\Yii::$app->session->get('name');
```

Вспышка сессии

```
\Yii::$app->getSession()->setFlash('flash_msg','Message');
\Yii::$app->getSession()->getFlash('flash_msg');
```

Создание и редактирование переменных сеанса, которые являются массивами

Сохраните переменную сеанса как переменную.

```
$session = Yii::$app->session;

$sess = $session['keys'];
```

Затем создайте или обновите требуемое значение массива

```
$sess['first'] = 'abc';
```

И, наконец, сохранить переменную сеанса

```
$session['keys'] = $sess
```

Помните URL для повторного просмотра позже.

Случай использования: помните, что текущий URL-адрес возвращается после добавления новой записи в другой (связанный) контроллер, например, создайте новый контакт, чтобы добавить к редактируемому счету.

InvoiceController / actionUpdate:

```
Url::remember(Url::current(), 'returnInvoice');
```

ContactController / actionCreate:

```
if ($model->save()) {
    $return = Url::previous('returnInvoice');
    if ($return) {
        return $this->redirect($return);
    }
    // ...
}
```

После того, как вы закончите, вы можете сбросить сохраненный URL:

InvoiceController / actionUpdate:

```
if ($model->save()) {
    Url::remember(null, 'returnInvoice');
    // ...
}
```

Имя ключа - `returnInvoice` в этом примере - необязательно.

Прочитайте сессия онлайн: https://riptutorial.com/ru/yii2/topic/3584/сессия

глава 19: тестирование

Examples

Настройка тестовой среды

Установить код:

```
composer global status
composer global require "codeception/codeception=~2.0.0" "codeception/specify=*"
"codeception/verify=*"
```

Установите Faker:

```
cd /var/www/yii                                    // Path to your application
composer require --dev yiisoft/yii2-faker:*
```

Создайте базу данных, которая будет использоваться только для тестов. Вы можете дублировать существующую базу данных или применять миграции:

```
cd tests
codeception/bin/yii migrate
```

Отрегулируйте конфигурацию `components['db']` в `tests/codeception/config/config-local.php` .

Добавьте каталог `/var/www/yii/vendor/bin` на ваш путь.

Просмотрите все файлы конфигурации и `.yml` .

Запустите веб-сервер, например:

```
php -S localhost:8080
```

Запустите тесты:

```
codecept run
```

Дополнительная информация:

- http://www.yiiframework.com/doc-2.0/guide-test-environment-setup.html
- http://codeception.com/install
- https://github.com/yiisoft/yii2-app-basic/tree/master/tests
- https://github.com/yiisoft/yii2-app-advanced/tree/master/tests

Примечание. Эти инструкции действительны для версии Yii2 версии 2.0.9. В версии 2.0.10

в соответствии с Sam Dark часть тестирования будет переопределена (и инструкции должны быть обновлены). Версия 2.0.10 должна быть выпущена 11 сентября 2016 года: https://github.com/yiisoft/yii2/milestones

Как насмехаться с ActiveRecord

Если вы хотите высмеять AR, который не пытается подключиться к базе данных, вы можете сделать это следующим образом (если используете PHPUnit):

```
$post = $this->getMockBuilder('\app\model\Post')
    ->setMethods(['save', 'attributes'])
    ->getMock();
$post->method('save')->willReturn(true);
$post->method('attributes')->willReturn([
    'id',
    'status',
    'title',
    'description',
    'text'
]);
```

Ловушка заключается в том, что нам нужно переопределить метод attributes (), поскольку по умолчанию ActiveRecord получает список атрибутов из схемы базы данных, которую мы пытаемся избежать.

Прочитайте тестирование онлайн: https://riptutorial.com/ru/yii2/topic/2226/тестирование

глава 20: Управление активами

Синтаксис

- заложенные активы будут загружены до этих активов в указанном порядке
    - public $ зависит = ['yii \ web \ YiiAsset', 'yii \ bootstrap \ BootstrapAsset', 'yii \ bootstrap \ BootstrapPluginAsset', 'cinghie \ fontawesome \ FontAwesomeAsset',];

замечания

этот пример основан на расширенном шаблоне https://github.com/yiisoft/yii2-app-advanced

Свойством cinghie в этом примере является пакет активов для adminLTE https://github.com/cinghie/yii2-admin-lte

Examples

Это часть файла макета

```
<?php
/* this example is based on the advanced template
 * This file is located in
 * backend/views/layouts/main.php
 */

use yii\helpers\Html;

// here the asset is registered
use cinghie\adminlte\AdminLTEAsset;
AdminLTEAsset::register($this);


?>
<?php $this->beginPage() ?>
<!DOCTYPE html>
<html lang="<?= Yii::$app->language ?>">
<head>
    <meta charset="<?= Yii::$app->charset ?>">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <?= Html::csrfMetaTags() ?>
    <title><?= Html::encode($this->title) ?></title>
    <?php $this->head() ?>
</head>

 <body class="hold-transition skin-blue sidebar-mini">

<?php $this->beginBody() ?>

<?= $content ?>

<?php $this->endBody() ?>
```

```
</body>
</html>
<?php $this->endPage() ?>
```

Это файл активов

```
<?php

/**
* This file is the Asset Bundle File located in
*  vendor/cinghie/yii2-admin-lte/AdminLTEAsset.php
* @copyright Copyright &copy; Gogodigital Srls
* @company Gogodigital Srls - Wide ICT Solutions
* @website http://www.gogodigital.it
* @github https://github.com/cinghie/yii2-admin-lte
* @license GNU GENERAL PUBLIC LICENSE VERSION 3
* @package yii2-AdminLTE
* @version 1.3.10
*/

namespace cinghie\adminlte;

use yii\web\AssetBundle;

/**
 * Class yii2-AdminLTEAsset
 * @package cinghie\adminlte
 */
class AdminLTEAsset extends AssetBundle
{

    /**
     * @inherit
     */
    public $sourcePath = '@bower/';

    /**
     * @inherit
     */
    public $css = [
        'ionicons/css/ionicons.css',
        'admin-lte/dist/css/AdminLTE.css',
        'admin-lte/dist/css/skins/_all-skins.css'
    ];

    /**
     * @inherit
     */
    public $js = [
        'admin-lte/dist/js/app.js'
    ];

    /**
     * @inherit
     */
    public $depends = [
        'yii\web\YiiAsset',
        'yii\bootstrap\BootstrapAsset',
```

```
        'yii\bootstrap\BootstrapPluginAsset',
    'cinghie\fontawesome\FontAwesomeAsset',
    ];

}
```

Сгенерированный HTML с автоматически загружаемыми активами

```
<!DOCTYPE html>
<html lang="en-EN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="csrf-param" content="_csrf">
    <meta name="csrf-token"
content="M01tTVZLdlBlBQEqGyYcYHc5PwI1CRknfB1bBiAaPTNBfyk0Ehg8EQ==">
    <title>Profil</title>
    <link href="/assets/f3e48cde/css/bootstrap.css?v=1473788138" rel="stylesheet">
<link href="/assets/24e44190/css/font-awesome.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/ionicons/css/ionicons.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/admin-lte/dist/css/AdminLTE.css?v=1473866258" rel="stylesheet">
<link href="/assets/fa4335a5/admin-lte/dist/css/skins/_all-skins.css?v=1473866258"
rel="stylesheet"></head>
<body class="hold-transition skin-blue sidebar-mini">

....

</script><script src="/assets/69b4ffbe/jquery.js?v=1473788138"></script>
<script src="/assets/6aa8a809/yii.js?v=1473788138"></script>
<script src="/assets/f3e48cde/js/bootstrap.js?v=1473788138"></script>
<script src="/assets/fa4335a5/admin-lte/dist/js/app.js?v=1473866258"></script>

</body>
</html>
```

Прочитайте Управление активами онлайн: https://riptutorial.com/ru/yii2/topic/6900/управление-активами

глава 21: Установка расширения вручную

Examples

Установка расширения без композитора

> Примечание: настоятельно рекомендуется использовать Composer. В приведенной ниже инструкции в основном используется композитор для вас.

- Загрузите файл расширения архива необходимой версии из Github
- Открыть `composer.json`
- Найдите раздел автозагрузки `PSR-4` и запомните его, например, `kmit/select2`
- Извлеките файлы в соответствующую папку в папке поставщика, например, `vendor/kmit/select2`
- Добавьте следующий код к `vendor/composer/autoload_psr4.php`

```
'kmit\\select2\\' => array($vendorDir . '/kmit/select2'),
```

- Добавьте следующий код для `vendor/yiisoft/extensions.php` :

```
'kmit/select2'(name of extension from composer.json file of extension) =>
    array (
        'name' => 'kmit/select2',
        'version' => '1.0.0.0',
        'alias' => array (
            '@vendor/kmit/select2'(path of extension folder alias) => $vendorDir .
'/kmit/select2' (path of extension folder),
        ),
    ),
```

Видеоролик

Прочитайте Установка расширения вручную онлайн: https://riptutorial.com/ru/yii2/topic/2224/установка-расширения-вручную

кредиты

S. No	Главы	Contributors
1	Начало работы с yii2	Bibek Lekhak, Community, Farcaller, jagsler, Mohan Rex, Muhammad Shahzad, Pasha Rumkin, Sam Dark, urmaul, Yasar Arafath, Yasin Patel, Yatin Mistry
2	Pjax	gmc, Manikandan S, Muaaz Rafi, Shaig Khaligli, yafater, Yasin Patel
3	Yii2 ActiveForm	Hina Vaja, Manikandan S, particleflux
4	Yii2 OAuth2 - Ex: потребительский facebook OAuth2	ThanhPV
5	Активная запись	Insane Skull, Manikandan S, Michael St Clair, Mike Artemiev, particleflux, saada, Sam Dark, Yasar Arafath, Yasin Patel
6	Восстановительный API	jagsler, jlapoutre, yafater, Yasin Patel
7	Загрузка файлов	Sam Dark
8	Запрос Ajax	Anton Rybalko, Ilyas karim, meysam
9	Календарь YII2 JQuery для текстового поля	Manikandan S
10	Компоненты	Sam Dark, Yasin Patel
11	Маршрутизация и URL-адреса	IStranger, Ярослав Гойса
12	Миграции баз данных	Ali MasudianPour, jlapoutre, Sam Dark
13	Печенье	IStranger, Sam Dark
14	Пользовательские проверки	Ejaz Karim
15	Проверка	jagsler, Mihai P., Nana Partykar, Sam Dark, Yasin Patel

16	Работа с базами данных	jagsler, Kiran Muralee
17	Расширенный шаблон проекта	mnoronha, Mohan Rex, Salem Ouerdani, Sam Dark, topher
18	сессия	Brett, Goke Obasa, jlapoutre, MikelG, Yasin Patel
19	тестирование	Antonín Slejška, Bizley, Sam Dark, XzAeRo
20	Управление активами	Thomas Rohde
21	Установка расширения вручную	Insane Skull, mnoronha, Sam Dark, vishuB