Создание REST server

В следующем рецепте мы используем пример, который иллюстрирует, как можно создать и настроить RESTful API с минимальными усилиями по кодированию. Этот рецепт будет повторно использован в других рецептах в этой главе.

Подготовка

1. Создайте новое приложение с помощью диспетчера пакетов Composer, как описано в официальном руководстве по адресу  
   <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>.   
   По русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation>.
2. Создайте миграцию для создания таблицы статей с помощью следующей команды:

./yii migrate/create create\_film\_table

1. Затем обновите только что созданный метод миграции с помощью следующего кода:

public function up()

{

$tableOptions = null;

if ($this->db->driverName === 'mysql') {

$tableOptions = 'CHARACTER SET utf8 COLLATE  
utf8\_general\_ci ENGINE=InnoDB';

}

$this->createTable('{{%film}}',

[

'id' => $this->primaryKey(),

'title' => $this->string(64)->notNull(),

'release\_year' => $this->integer(4)->notNull(),

], $tableOptions);

$this->batchInsert('{{%film}}',

['id','title','release\_year'],

[

[1, 'Interstellar', 2014],

[2, "Harry Potter and the Philosopher's Stone",2001],

[3, 'Back to the Future', 1985],

[4, 'Blade Runner', 1982],

[5, 'Dallas Buyers Club', 2013],

]);

}

Обновление метод со следующим кодом:

public function down()

{

$this->dropTable('film');

}

1. Запустите созданную миграцию create\_film\_table.
2. Создать модель Film модулем GII.
3. Настройте сервер приложений на использование чистых URL-адресов. Если вы используете Apache с mod\_rewrite и allowoverride включен, то вы должны добавить следующие строки в файл .htaccess в каталоге @web:

Options +FollowSymLinks  
IndexIgnore \*/\*

RewriteEngine on

* if a directory or a file exists, use it directly  
  RewriteCond %{REQUEST\_FILENAME} !-f  
  RewriteCond %{REQUEST\_FILENAME} !-d
* otherwise forward it to index.php

RewriteRule . index.php

Как это сделать...

1. Создать контроллер, @app/controller/FilmController.php, со следующим кодом:

<?php

namespace app\controllers;

use yii\rest\ActiveController;

class FilmController extends ActiveController  
{

public $modelClass = app\models\Film';

}

Обновите @app/config/web.php. Добавьте следующую конфигурацию компонента urlManager:

'urlManager' => [

'enablePrettyUrl' => true,

'enableStrictParsing' => true,

'showScriptName' => false,

'rules' => [

['class' => 'yii\rest\UrlRule', 'controller' => 'films'],

],

],

1. Перенастройте компонент запроса в @app/config/web.php:

'request' => [

'cookieValidationKey' => 'mySecretKey',

'parsers' => [

'application/json' => 'yii\web\JsonParser',

],

]

Как это работает…

Мы раширяем \yii\rest\ActiveController, чтобы создать наш собственный контроллер, затем для созданных контроллер, свойства modelClass был установлен. Класс \yii\rest\ActiveController реализует общий набор действий для поддержки RESTful-доступа к ActiveRecord.

С вышеуказанным минимальным количеством усилий вы уже закончили создание RESTful API для доступа к данным Film.

Созданные API включают:

* get /films: В этом списке все фильмы страница за страницей
* head /films: Это показывает обзорную информацию о списке фильмов
* post /films: Это создает новый фильм
* get /films/5: Это возвращает детали фильма 5
* head /films/5: Это показывает обзорную информацию фильма 5
* patch /films/5 and put /films/5: Это обновление фильм 5
* delete /films/5: Это удаляет фильм 5
* options /films: Это показывает Поддерживаемые команды относительно конечной точки /films
* options /films/5: Это показывает Поддерживаемые команды относительно конечной точки /films/5

Это работает так, потому что \yii\rest\ActiveController поддерживает следующие действия:

* index: Здесь перечислены модели
* view: Возвращает сведения о модели
* create: Это создает новую модель
* update: Это обновляет существующую модель
* delete: Это приведет к удалению существующей модели
* options: Возвращает методы HTTP

И есть также метод verbs (), который определяет разрешенные методы запроса для каждого действия.

Чтобы убедиться, что наш RESTful API работает корректно, отправим несколько запросов.

Начнем с запроса GET. Запустите это в консоли:

curl -i -H "Accept:application/json" "<http://yii-book.app/films>"

Вы получите следующий вывод:

HTTP/1.1 200 OK

Date: Wed, 23 Sep 2015 17:46:35 GMT  
Server: Apache  
X-Powered-By: PHP/5.5.23  
X-Pagination-Total-Count: 5  
X-Pagination-Page-Count: 1  
X-Pagination-Current-Page: 1  
X-Pagination-Per-Page: 20

Link: <<http://yii-book.app/films?page=1>>; rel=self  
Content-Length: 301

Content-Type: application/json; charset=UTF-8

[{"id":1,"title":"Interstellar","release\_year":2014},{"id":2,"title":"Harry Potter and  
the Philosopher's Stone","release\_year":2001},{"id":3,"title":"Back to the  
Future","release\_year":1985},{"id":4,"title":"Blade Runner","release\_year":1982},  
{"id":5,"title":"Dallas Buyers Club","release\_year":2013}]

Давайте отправим запрос POST. Запустите это в консоли:

curl -i -H "Accept:application/json" -X POST -d title="New film" -d release\_year=2015  
"<http://yii-book.app/films>"

Вы получите следующий вывод:

HTTP/1.1 201 Created

Date: Wed, 23 Sep 2015 17:48:06 GMT

Server: Apache

X-Powered-By: PHP/5.5.23

Location: <http://yii-book.app/films/6>

Content-Length: 49

Content-Type: application/json; charset=UTF-8  
{"title":"New film","release\_year":"2015", "id":6}

Давайте создадим фильм. Запустите в консоли:

curl -i -H "Accept:application/json" "<http://yii-book.app/films/6>"

Вы получите следующий вывод:

HTTP/1.1 200 OK

Date: Wed, 23 Sep 2015 17:48:36 GMT  
Server: Apache  
X-Powered-By: PHP/5.5.23  
Content-Length: 47

Content-Type: application/json; charset=UTF-8  
{"id":6, "title":"New film", "release\_year":2015}

Давайте отправим запрос на удаление. Запустите это в консоли:

curl -i -H "Accept:application/json" -X DELETE "<http://yii-book.app/films/6>"

И вы получите следующий результат:

HTTP/1.1 204 No Content

Date: Wed, 23 Sep 2015 17:48:55 GMT

Server: Apache

X-Powered-By: PHP/5.5.23

Content-Length: 0

Content-Type: application/json; charset=UTF-8

Кое что еще...

Теперь мы рассмотрим согласование контента и настройку правила rest URL:

Согласование содержания

Вы также можете легко отформатировать ответ поведением согласования Контента.

Например, можно поместить этот код в контроллер, и все данные будут возвращены в формате XML.

Ознакомьтесь с полным списком форматов в документации.

use yii\web\Response;  
public function behaviors()

{

$behaviors = parent::behaviors();

$behaviors['contentNegotiator']['formats']['application/xml']= Response::FORMAT\_XML;  
return $behaviors;

}

Запустите это в консоли:

curl -i -H "Accept:application/xml" "<http://yii-book.app/films>"

Вы получите следующий вывод:

HTTP/1.1 200 OK

Date: Wed, 23 Sep 2015 18:02:47 GMT  
Server: Apache

X-Powered-By: PHP/5.5.23  
X-Pagination-Total-Count: 5  
X-Pagination-Page-Count: 1  
X-Pagination-Current-Page: 1  
X-Pagination-Per-Page: 20

Link: <<http://yii-book.app/films?page=1>>; rel=self  
Content-Length: 516

Content-Type: application/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>

<response>

<item>

<id>1</id>

<title>Interstellar</title>

<release\_year>2014

</release\_year>

</item>

<item>

<id>2</id>

<title>Harry Potter and the Philosopher's Stone</title>

<release\_year>2001

</release\_year>

</item>

<item>

<id>3</id>

<title>Back to the Future</title>

<release\_year>1985

</release\_year>

</item>

<item>

<id>4</id>

<title>Blade Runner</title>

<release\_year>1982

</release\_year>

</item>

<item>

<id>5</id>

<title>Dallas Buyers Club</title>

<release\_year>2013

</release\_year>

</item>

</response>

Настройка остальных правил

Вы должны помнить, что идентификатор контроллера, по умолчанию, определяется во множественном числе. Это происходит потому, что в yii\rest\UrlRule автоматически в форму множественного идентификаторов контроллера. Вы можете просто отключить это, установив yii\rest\UrlRule::$pluralize в false:

'urlManager' => [

//..

'rules' => [

[

'class' => 'yii\rest\UrlRule',

'controller' => 'film'

'pluralize' => false

],

],

//..

]

Если вы хотите указать как ID контроллера должен присутствовать в модели, вы можете добавить Пользовательское имя массива как значение ключа пары, где массив ключ-ID контроллера и массив значений-это фактический контроллер ID. Например:

'urlManager' => [

//. .

'rules' => [

[

'class' => 'yii\rest\UrlRule',

'controller' => ['super-films' => 'film']

],

],

//..

]

Смотрите так же

Для получения дополнительной информации, обратитесь к следующему адресу:

* [http://www.viiframework.com/doc-2.0/guide-rest-quick-starr.hrml](http://www.yiiframework.com/doc-2.0/guide-rest-quick-start.html)
  1. по русски http://yiiframework.domain-na.me/doc/guide/2.0/ru/rest-quick-start
* <http://www.yiiframework.com/doc-2.0/yii-rest-urlrule.html>
* <http://www.yiiframework.com/doc-2.0/guide-rest-response-formatting.html>
  1. по русски http://yiiframework.domain-na.me/doc/guide/2.0/ru/rest-response-formatting
* <http://budiirawan.com/setup-restful-api-yii2/>