Ограничение скорости

Чтобы предотвратить злоупотребления, следует рассмотреть возможность добавления ограничения скорости к API. Например, можно ограничить использование API каждым пользователем максимум пятью вызовами API в течение одной минуты. Если в течение указанного периода времени От пользователя получено слишком много запросов, необходимо вернуть ответ с кодом состояния 429 (слишком много запросов).

Подготовка

Повторите все шаги, описанные в рецепте Создание рецепта REST-сервера подготовка и как это сделать... .

1. Создайте миграцию для создания таблицы разрешений пользователя с помощью следующей команды:

./yii migrate/create create\_user\_allowance\_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 COLLATEutf8\_general\_ci ENGINE=InnoDB';

}

$this->createTable('{{%user\_allowance}}', [

'user\_id' => $this->primaryKey(),

'allowed\_number\_requests' => $this->integer(10)->notNull(), 'last\_check\_time' => $this->integer(10)->notNull()

], $tableOptions);

}

1. Обновление метод Down() следующим кодом:

public function down()

{

$this->dropTable('{{%user\_allowance}}');

}

1. Запустите созданную миграцию create\_film\_table.
2. Создайте модель UserAllowance модулем GII.

Как это сделать...

Во-первых, вы должны обновить @app/controllers/FilmController.php следующим кодом:

<?php

namespace app\controllers;

use yii\rest\ActiveController;

use yii\filters\RateLimiter;

use yii\filters\auth\QueryParamAuth;

class FilmController extends ActiveController  
{

public $modelClass = 'app\models\Film';

public function behaviors()

{

$behaviors = parent::behaviors();

$behaviors['authenticator'] = [

'class' => QueryParamAuth::className(),

];

$behaviors['rateLimiter'] = [

'class' => RateLimiter::className(),

'enableRateLimitHeaders' => true  
];

return $behaviors;

}

}

Чтобы включить ограничение скорости, класс пользовательской модели должен реализовать Yii\filters\RateLimitInterface и требует реализации трех методов: getRateLimit (), loadAllowance () и saveAllowance(). Необходимо добавить их с константами RATE\_LIMIT\_NUMBER и RATE\_LIMIT\_RESET:

<?php

namespace app\models;

class User extends \yii\base\Object implements \yii\web\IdentityInterface, \yii\filters\RateLimitInterface  
{

public $id;  
public $username;  
public $password;  
public $authKey;  
public $accessToken;

const RATE\_LIMIT\_NUMBER = 5;  
const RATE\_LIMIT\_RESET = 60;

// it means that user allowed only 5 requests per one minute  
public function getRateLimit($request, $action)

{

return [self::RATE\_LIMIT\_NUMBER,self::RATE\_LIMIT\_RESET];

}

public function loadAllowance($request, $action)

{

$userAllowance = UserAllowance::findOne($this->id);  
return $userAllowance ?

[$userAllowance->allowed\_number\_requests,$userAllowance->last\_check\_time] :  
$this->getRateLimit($request, $action);

}

public function saveAllowance($request, $action,$allowance, $timestamp)

{

$userAllowance = ($allowanceModel =UserAllowance::findOne($this->id)) ?  
$allowanceModel : new UserAllowance();

$userAllowance->user\_id = $this->id;

$userAllowance->last\_check\_time = $timestamp;  
$userAllowance->allowed\_number\_requests =$allowance;

$userAllowance->save();

}

// other User model methods

}

Как это работает...

Как только класс identity реализует требуемый интерфейс, Yii автоматически использует [ [yii\filters\RateLimiter]], настроенный как фильтр действий для [[yii\rest\controller]] для выполнения проверки ограничения скорости. Мы также добавили поведение 'authenticator' с классом QueryParamAuth. Таким образом, теперь мы можем аутентифицироваться только с помощью маркера доступа, переданного через параметр запроса. Можно добавить любой метод проверки подлинности, описанный в официальном руководстве в разделе веб-службы RESTful.

Давайте объясним наши методы. Их довольно легко понять.

getRateLimit (): возвращает максимальное количество разрешенных запросов и период времени (например, [100, 600] означает, что может быть не более 100 вызовов API в течение 600 секунд)

loadAllowance(): возвращает количество разрешенных оставшихся запросов и соответствующую метку времени UNIX при последней проверке ограничения скорости

saveAllowance (): сохраняет как количество оставшихся запросов, так и текущую метку времени UNIX

Мы храним наши данные в базе данных MySQL. Для повышения производительности можно использовать базу данных NoSQL или другую систему хранения с большим временем получения и загрузки данных.

Теперь давайте попробуем проверить функцию ограничения скорости. Запустите это в консоли:

curl -i "<http://yii-book.app/films?access-token=100-token>"

Вы получите следующий вывод:

HTTP/1.1 200 OK

Date: Thu, 24 Sep 2015 01:35:51 GMT  
Server: Apache  
X-Powered-By: PHP/5.5.23

Set-Cookie: PHPSESSID=495a928978cc732bee853b83f521eba2; path=/; HttpOnly  
Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

X-Rate-Limit-Limit: 5

X-Rate-Limit-Remaining: 4

X-Rate-Limit-Reset: 0

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?access-token=100-token&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 узнаем о возвращаемых заголовках. Когда ограничение частоты запросов включено, по умолчанию каждый ответ будет возващаться со следующими http-заголовками, содержащими информацию о текущих ограничениях:

X-Rate-Limit-Limit: Это Максимальное число запросов, разрешенных в течение определенного периода времени

X-Rate-Limit-Remaining: Это количество оставшихся запросов в текущем периоде времени

X-Rate-Limit-Reset: Это количество секунд ожидания для получения максимального количества разрешенных запросов

Таким образом, теперь попробуйте превысить предел, запросить следующий URL-адрес более пяти раз в минуту, и вы увидите TooManyRequestsHttpExeption:

HTTP/1.1 429 Too Many Requests  
Date: Thu, 24 Sep 2015 01:37:24 GMT  
Server: Apache  
X-Powered-By: PHP/5.5.23

Set-Cookie: PHPSESSID=bb630ca8a641ef92bd210c0a936e3149; path=/; HttpOnly  
Expires: Thu, 19 Nov 1981 08:52:00 GMT

Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

X-Rate-Limit-Limit: 5

X-Rate-Limit-Remaining: 0

X-Rate-Limit-Reset: 60

Content-Length: 131

Content-Type: application/json; charset=UTF-8  
{"name":"Too Many Requests","message":"Rate limit

exceeded.","code":0,"status":429,"type":"yii\\web\\TooManyRequestsHttpException"}

Смотрите так же

Дополнительные сведения см. по следующим URL-адресам:

* [https://en.wikipedia.org/wiki/Leaky bucket](https://en.wikipedia.org/wiki/Leaky_bucket)
* <http://www.yiiframework.com/doc-2.0/guide-rest-rate-limiting.html>
  1. по русски http://yiiframework.domain-na.me/doc/guide/2.0/ru/rest-rate-limiting
* [httpy/www.yiiframework.com/doc-2.0/yii-filters-ratelimiter.html](http://www.yiiframework.com/doc-2.0/yii-filters-ratelimiter.html)