官方文档：

https://jwt-auth.readthedocs.io/en/develop/quick-start/

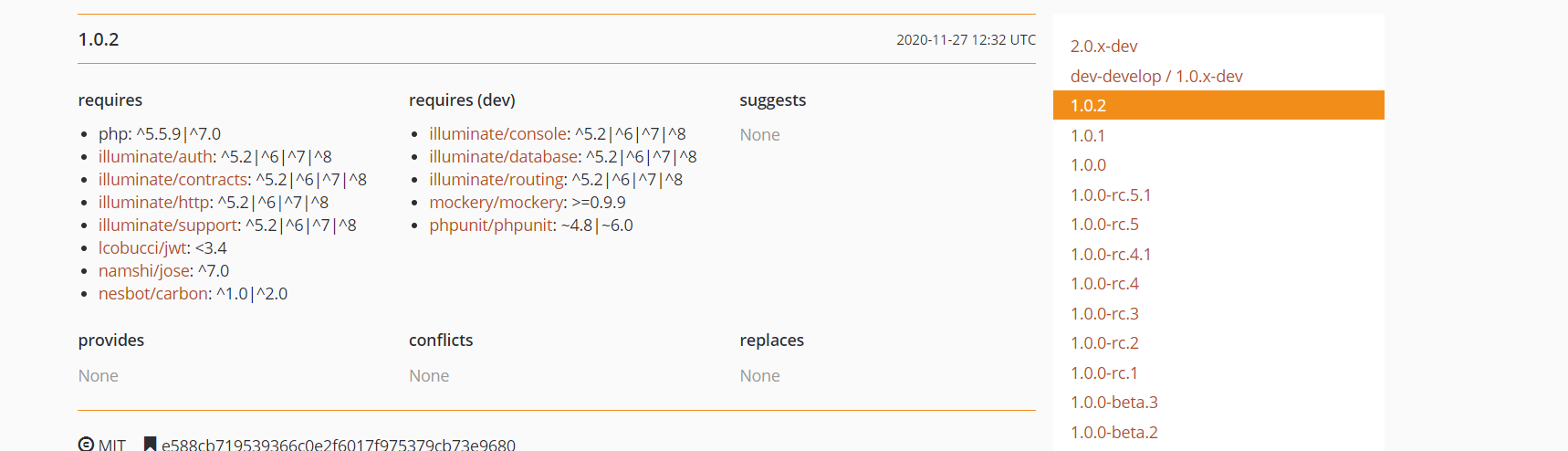
参考：

<https://blog.csdn.net/yqszkl/article/details/106658526>

## 安装与配置

### 1.安装

composer require tymon/jwt-auth:1.0.0-rc.4.1



第二遍安装过程中 [lcobucci /](https://packagist.org/packages/lcobucci/) jwt包出错，我重新安装了对应的版本 3.3.1具体的要求镜像仓库中有写要求

需要安装  [lcobucci /](https://packagist.org/packages/lcobucci/) jwt包

composer require lcobucci/jwt:3.3.1

composer config -g repo.packagist composer <https://packagist.org>

重置镜像

### 2.发布配置文件

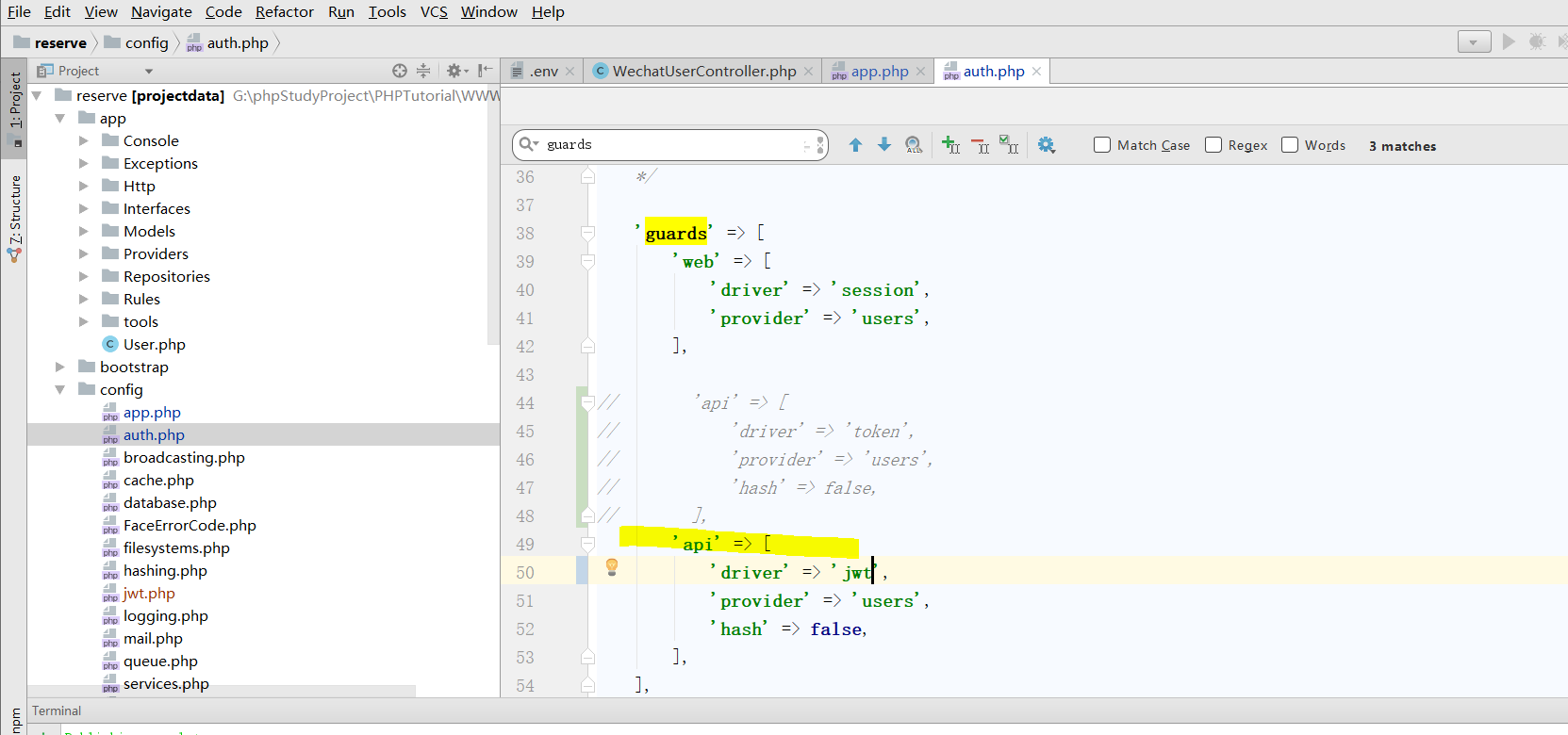
php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

这条命令会在 config 下增加一个 jwt.php 的配置文件

### 3. 生成加密秘钥

php artisan jwt:secret

### 4.修改/config/auth.php



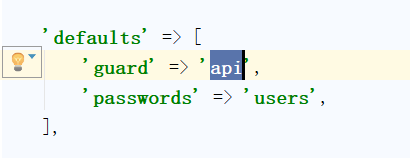
### 5.修改用户模型

**<?php  
  
namespace** App;  
  
**use** Tymon\JWTAuth\Contracts\JWTSubject;  
**use** Illuminate\Notifications\Notifiable;  
**use** Illuminate\Foundation\Auth\User **as** Authenticatable;  
  
**class** User **extends** Authenticatable **implements** JWTSubject  
{  
 **use** Notifiable;  
 **protected $table**=**"users"**;  
 **public $timestamps**=**false**;  
 **protected $fillable**=[**'id'**,**'name'**,**'password'**,**'created\_at'**,**'updated\_at'**];  
 *// Rest omitted for brevity  
  
 /\*\*  
 \* Get the identifier that will be stored in the subject claim of the JWT.  
 \*  
 \** ***@return*** *mixed  
 \*/* **public function** getJWTIdentifier()  
 {  
 **return** $this->getKey();  
 }  
  
 */\*\*  
 \* Return a key value array, containing any custom claims to be added to the JWT.  
 \*  
 \** ***@return*** *array  
 \*/* **public function** getJWTCustomClaims()  
 {  
 **return** [];  
  
 }  
}

### 6.controller使用jwt

auth(**'api'**)

auth（）



如果/config/auth.php

Guard默认设置成api可以直接用auth()

如果没有则用auth(‘api’)

*\* auth(‘api’)->attempt($credentials)密码校验的时候 会使用Hash::check()的校验方式，索引生成密码的时候要用  
\* Hash::make()*

<?php

namespace App\Http\Controllers;

use App\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Tymon\JWTAuth\Facades\JWTAuth;

use Log;

class AuthController extends Controller

{

/\*\*

\* Create a new AuthController instance.

\* 要求附带email和password（数据来源users表）

\*

\* @return void

\*/

public function \_\_construct()

{

// 这里额外注意了：官方文档样例中只除外了『login』

// 这样的结果是，token 只能在有效期以内进行刷新，过期无法刷新

// 如果把 refresh 也放进去，token 即使过期但仍在刷新期以内也可刷新

// 不过刷新一次作废

//$this->middleware('auth:api', ['except' => ['login']]);

// 另外关于上面的中间件，官方文档写的是『auth:api』

// 但是我推荐用 『jwt.auth』，效果是一样的，但是有更加丰富的报错信息返回

// auth:api auth指的是中间件 App\Http\Kernel中$routeMiddleware定义的。而后面 :api 是路由参数，指定了要使用哪个看守器，可以看到下面 api 对应的看守器就是 jwt 的看守器。

// 并且你可以直接使用 auth ，这样就相当于使用 defaults 中指定的看守器，即 session。

// Lumen 默认用的就是 api 那个，所以你直接用 auth 作为 api 路由的中间件完全没问题

// Laravel 中指定了两个看守器，而且默认的并不是 api，所以你必须得用 auth:api 作为路由的中间件

}

/\*\*

\* Get a JWT via given credentials.

\*

\* @return \Illuminate\Http\JsonResponse

\*/

public function login()

{

$credentials = request(['phone', 'password']);

/\*

\* 这里创建token有三种方式

\* 1. 基于账密参数

\*\*/

$token = auth('api')->attempt($credentials);

/\* 2. 基于 users 模型返回的实例

$user = User::where([

['phone', $credentials['phone'], ['password', $credentials['password']]]

])->first();

$token = auth('api')->login($user);

\*/

/\* 3. 基于 users 模型中的主键 id

$token = auth('api')->tokenById($user->id);

\*/

if (!$token) {

return response()->json(['error' => 'Unauthorized'], 401);

}

return response()->json([

'access\_token' => $token,

'token\_type' => 'bearer',

'expires\_in' => auth('api')->factory()->getTTL() \* 60,

]);

// return $this->respondWithToken($token);

}

/\*\*

\* Get the authenticated User.

\*

\* @return \Illuminate\Http\JsonResponse

\*/

public function me()

{

$user = auth('api')->user();

$r['user'] = $user;

return response()->json($r);

}

/\*\*

\* Log the user out (Invalidate the token).

\*

\* @return \Illuminate\Http\JsonResponse

\*/

public function logout(Request $request)

{

$token = $request->header("Authorization");

try {

JWTAuth::invalidate(JWTAuth::getToken());

return response()->json([

"status" => "success",

"message"=> "User successfully logged out."

]);

} catch (JWTException $e) {

// something went wrong whilst attempting to encode the token

return response()->json([

"status" => "error",

"message" => "Failed to logout, please try again."

], 500);

}

//auth('api')->logout(); 使用该方法，注销无效。

return response()->json(['message' => 'Successfully logged out']);

}

/\*\*

\* Refresh a token.

\* 刷新token，如果开启黑名单，以前的token便会失效。

\* 值得注意的是用上面的getToken再获取一次Token并不算做刷新，两次获得的Token是并行的，即两个都可用。

\* @return \Illuminate\Http\JsonResponse

\*/

public function refresh()

{

return $this->respondWithToken(auth('api')->refresh());

}

/\*\*

\* Get the token array structure.

\*

\* @param string $token

\*

\* @return \Illuminate\Http\JsonResponse

\*/

protected function respondWithToken($token)

{

return response()->json([

'access\_token' => $token,

'token\_type' => 'bearer',

'expires\_in' => auth('api')->factory()->getTTL() \* 60

]);

}

}

### 7.添加路由

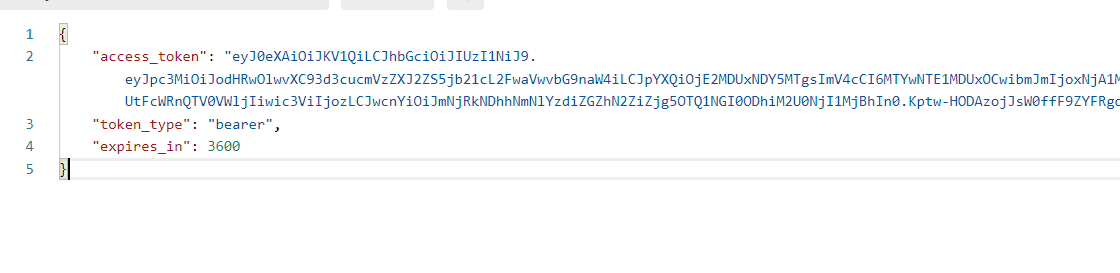
Route::*post*(**'create'**, **'AuthController@create'**);  
Route::*post*(**'login'**, **'AuthController@login'**);  
Route::*group*([**'middleware'**=>**'auth:api'**],**function**(){  
  
  
 Route::*post*(**'logout'**, **'AuthController@logout'**);  
 Route::*post*(**'refresh'**, **'AuthController@refresh'**);  
 Route::*post*(**'me'**, **'AuthController@me'**);  
  
});

### 8.jwt.php配置参数

*//用于加密生成token的秘钥***'secret'** => env(**'JWT\_SECRET'**),

*//用于加密生成token的秘钥***'secret'** => env(**'JWT\_SECRET'**),

*/\*\*  
 \* 刷新时间--在此时间内以旧token换取新token  
 \* token有效时间为60分钟 刷新时间为20160(一周)  
 \* 在60分钟以内可以用旧token换区新tokrn  
 \* 但是超过 60 分钟是不可以的，然后你可以一直循环获取，直到总时间超过 20160 分钟，不能再获取。  
 \*/***'refresh\_ttl'** => env(**'JWT\_REFRESH\_TTL'**, 20160),

登录成功之后访问其他路由 携带上token 添加请求头Accept：application/json.

### 9.token过期后不重新登录而是刷新token

在没有设置刷新token的情况下,如果**ttl**设置的token时间到期了 则需重新登录

自定义中间件JwtRefreshToken并继承BaseMiddleware

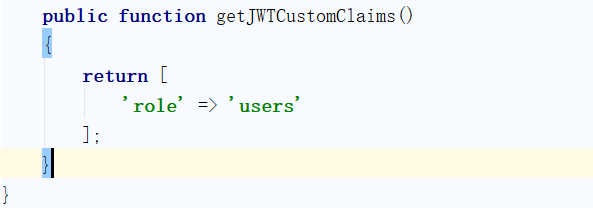
当用户token过期时 后端将token刷新传递给前端 前端修改token值

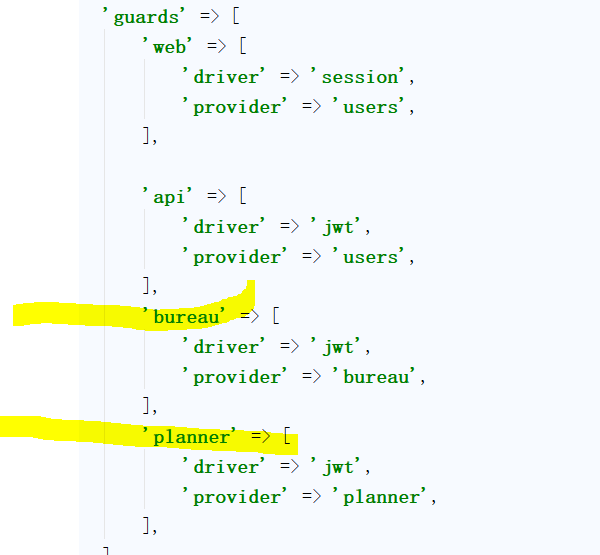
### 10.多表的jwt

参考：<https://learnku.com/articles/30342>

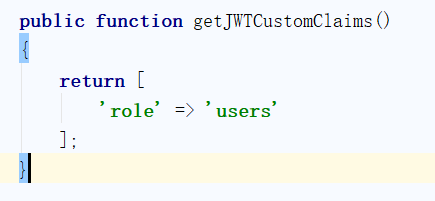
https://www.cnblogs.com/it-3327/p/11765336.html

#### 1.配置多个守卫





#### 2.模型层给token加标识



#### 3.中间件判断



#### 4.路由定义

