常言道「物以类聚，人以群分」，同样，在日常开发中，我们通常会将具有某些共同特征的路由进行分组，这些特征包括是否需要认证、是否具有共同的路由前缀或者子域名、以及是否具有相同的控制器命名空间等，显然，对路由按照共同特征进行分组后可以避免重复为某些路由定义相同的路由特征，让代码更加简洁，可读性和可维护性更好。

所谓路由分组，其实就是通过 Route::group 将几个路由聚合到一起，然后给它们应用对应的共享特征：

Route::group([], function () {

Route::get('hello', function () { return 'Hello'; });

Route::get('world', function () { return 'World'; });

});

由于没有应用任何共享特征（第一个参数是空数组），所以这样的路由分组其实并没有什么意义，下面我们来介绍一些常见的共享特征设置。

**中间件**

我们使用路由分组最常见的场景恐怕就是为一组路由应用共同的中间件了，关于中间件可以参考[官方文档](https://laravelacademy.org/post/9539.html)，后面也会有单独章节来讲解，使用中间件可以对 HTTP 请求进行过滤或重定向，比如以认证中间件（别名auth）为例，如果用户已经认证可以进行后续处理，否则将会把用户重定向到登录页面。

下面我们就来创建一个包含 dashboard 和 account 的路由分组，这两个路由都需要认证，所以我们可以通过 Route::middleware 为其设置共同的中间件 auth 并以此对其进行分组：

Route::middleware('auth')->group(function () {

Route::get('dashboard', function () {

return view('dashboard');

});

Route::get('account', function () {

return view('account');

});

});

如果是多个中间件，可以通过数组方式传递参数，比如 ['auth', 'another']，以上是 Laravel 5.5+ 提供的新语法，在此之前的版本，需要这么调用：

Route::group(['middleware' => 'auth'], function () {

Route::get('dashboard', function () {

return view('dashboard');

});

Route::get('account', function () {

return view('account');

});

});

当然，链式调用只是语法糖，底层最终还是下面 Route::group 这种定义实现的，感兴趣的同学可以去看下源码是如何实现的： vendor/laravel/framework/src/Illuminate/Routing/RouteRegistrar.php，下面路径前缀、子域名和命名空间的链式调用原理也是一样，以后我们都用链式调用来定义，因为这样代码可读性更好。

**路由路径前缀**

如果某些路由拥有共同的路径前缀，例如，所有 API 路由都以 /api 前缀开头，我们可以使用 Route::prefix 为这个分组路由指定路径前缀并对其进行分组：

Route::prefix('api')->group(function () {

Route::get('/', function () {

// 处理 /api 路由

})->name('api.index');

Route::get('users', function () {

// 处理 /api/users 路由

})->name('api.users');

});

**子域名路由**

子域名路由和路由路径前缀一样，不过是通过子域名而非路径前缀对分组路由进行约束，子域名路由有两个使用场景，一个是为应用子系统设置不同的子域名：

Route::domain('admin.blog.test')->group(function () {

Route::get('/', function () {

// 处理 http://admin.blog.test 路由

});

});

另一个是通过参数方式设置子域名，适用于网站拥有多租户的场景（比如天猫，顶级知名商家拥有自己独立的子域名，如 https://xiaomi.tmall.com）：

Route::domain('{account}.blog.test')->group(function () {

Route::get('/', function ($account) {

//

});

Route::get('user/{id}', function ($account, $id) {

//

});

});

这种情况下，$account 永远是所有分组路由的第一个路由参数。

**子命名空间**

以控制器方式定义路由的时候，当我们没有显式指定控制器的命名空间时，默认的命名空间是 App\Http\Controllers（在 app/Providers/RouteServiceProvider.php 中设置），如果某些控制器位于这个命名空间下的子命名空间中，该如何设置分组规则呢？我们可以通过 Route::namespace 为同一子命名空间下的分组路由设置共同的子命名空间：

Route::get('/', 'Controller@index');

Route::namespace('Admin')->group(function() {

// App\Http\Controllers\Admin\AdminController

Route::get('/admin', 'AdminController@index');

});

**路由命名前缀**

除了通过上述共同特征对路由进行分组外，对于某一类资源路由，比如用户，往往拥有相同的路由命名前缀，如 user.，我们还可以基于这一特征对路由进行分组，使用 Route::name 方法即可实现：

// 路由命名+路径前缀

Route::name('user.')->prefix('user')->group(function () {

Route::get('{id?}', function ($id = 1) {

// 处理 /user/{id} 路由，路由命名为 user.show

return route('user.show');

})->name('show');

Route::get('posts', function () {

// 处理 /user/posts 路由，路由命名为 user.posts

})->name('posts');

});

在这个示例中，我们通过链式调用的方式为该路由分组应用了路由命名前缀和路由路径前缀两个共享特征，我们还可以组合调用上述所有五个特征，调用方法参考上面这种链式调用，从而组合出更加复杂的分组规则。

SUMMARY：路由分组也就是将路由按照一定的规则分类到一起

后面两个不是很理解——2019-8-8 15:15:20

https://laravelacademy.org/post/9612.html