——高性能PHP应用开发之

第12讲 用户认证/授权和中间件

目录

CONTENTS

1 用户认证

2 用户授权

3 发送邮件

4 中间件

目录

CONTENTS

1 用户认证

2 用户授权

3 发送邮件

4 中间件

用户认证基本流程

- Laravel自带支持用户认证机制，支持的基本功能有：
  - ✓ 开启用户认证：php artisan **make:auth**
  - ✓ 用户注册：App\Controllers\Auth\RegisterController
  - ✓ 用户登录：App\Controllers\Auth\LoginController
  - ✓ 忘记密码和重置密码：App\Controllers\Auth\ForgotPasswordController 和 App\Controllers\Auth\ResetPasswordController
  - ✓ 用户注销：logout
- 配置文件：config/auth.php

```
'users' => [
    'driver' => 'eloquent',
    'model' => App\User::class,
],
```

用户注册

- 控制器：RegisterController
  - ✓ 使用 trait Illuminate\Foundation\Auth\RegistersUsers
  - ✓ 跳转地址：$redirectTo，注册成功后的跳转路径
  - ✓ 注册信息校验： validator（）

```php
protected function validator(array $data)
{
    return Validator::make($data, [
        'name' => 'required|max:255',
        'email' => 'required|email|max:255|unique:users',
        'password' => 'required|min:6|confirmed',
    ]);
}
```

- 路由： Illuminate\Routing\Router::auth( )

```php
// Registration Routes...
$this->get('register', 'Auth\RegisterController@showRegistrationForm')
    register');
$this->post('register', 'Auth\RegisterController@register');
```

- 视图：views/auth/register.blade.php

用户登录

- 控制器：LoginController
  - ✓ 使用 trait Illuminate\Foundation\Auth\AuthenticatesUsers
  - ✓ 跳转地址：$redirectTo，登录成功后的跳转路径
  - ✓ 校验字段：默认为 email 和 密码（必填），可以添加 username( )方法，修改 email字段为数据表中其它字段
  - ✓ 用户注销方法：logout( )
- 路由： Illuminate\Routing\Router::auth( )

```php
// Authentication Routes...
$this->get('login', 'Auth\LoginController@showLoginForm')->name('login');
$this->post('login', 'Auth\LoginController@login');
$this->post('logout', 'Auth\LoginController@logout')->name('logout');
```

- 视图：views/auth/login.blade.php

```php
public function username()
{
    return 'name';
}
```

忘记和重置密码

- 控制器：ForgotPasswordController、ResetPasswordController
  - ✓ 跳转地址：$redirectTo，重置密码成功后的跳转路径
  - ✓ 注意：提前配置好电子邮件
- 路由： Illuminate\Routing\Router::auth( )

```
// Password Reset Routes...
$this->get('password/reset', 'Auth\ForgotPasswordController@showLinkRequestForm');
$this->post('password/email', 'Auth\ForgotPasswordController@sendResetLinkEmail');
$this->get('password/reset/{token}', 'Auth\ResetPasswordController@showResetForm');
$this->post('password/reset', 'Auth\ResetPasswordController@reset');
```

- 视图：views/auth/passwords/*.blade.php

用户认证后的处理

- 获取认证用户：$user = Auth::user( );
  - ✓ 返回为 App\User 模型类对象，可以使用任何Eloquent ORM方法
  - ✓ 可以为 App\User 模型类添加 一对一 关联关系，从而方便获取用户的详细信息

```php
public function memberinfo()
{
    return $this->hasOne('App\Model\MemberInfo');
}
```

```php
$user = Auth::user();
dump($user->email);
dump($user->memberinfo->phone);
```

- 判断当前用户是否通过认证：Auth::check( )

```php
if (Auth::check()) {
    // The user is logged in...
}
```

- 路由中间件：
  - ✓ 在路由定义中，指明使用的中间件
  - ✓ 在控制器类定义中，指明使用的中间件

手动认证用户

- 认证用户登录：Auth::attempt(认证参数数组)

✓ 认证参数数组，将对应数据表中的数据；若完全匹配，则认证成功，否则失败。

✓ 认证成功后，可以使用 ResponseFactory 对象的 **intended**( )方法返回原始请求地址。

```php
if (Auth::attempt(['email' => $email, 'password' => $password])) {
    // Authentication passed...
    return redirect()->intended('dashboard');
}
```

- 记住用户：Auth::attempt(认证参数数组 , true)

```php
if (Auth::attempt(['email' => $email, 'password' => $password], true)) {
    // the user is being remembered...
}
```

- 用户注销：Auth::logout( );

目录

CONTENTS

1 用户认证

2 用户授权

3 发送邮件

4 中间件

用户授权

除了提供开箱即用的认证服务之外，Laravel 还提供了一个简单的方式来管理授权逻辑以便控制对资源的访问权限。和认证一样，在Laravel中实现授权很简单，主要有两种方式：gates和policies。

- Gate网关形式
  - ✓ 定义能力（用户-资源对应关系）
  - ✓ 校验用户是否具有某个能力
- Policies策略形式
  - ✓ 定义策略类（包含各种能力）
  - ✓ Gate校验

用户授权—Gate能力

■ 定义能力：

✓ 在 app/Providers/AuthServiceProvider 类里定义能力

✓ 使用 Illuminate\Auth\Access\Gate 类定义相应的能力

```php
public function boot(\Illuminate\Contracts\Auth\Access\Gate $gate)
{
    $this->registerPolicies($gate);

    $gate->define('update-post', function($user, $post) {
        return $user->id === $post->user_id;
    });
}
```

✓ 基于类方法定义能力

```php
$gate->define('update-post', 'Class@method');
```

用户授权—Gate能力

- 检查能力：

✓ Gate门面提供的 allows( ) 和 denies( ) 方法

```php
public function update(Request $request, Post $post)
{
    // 校验当前授权用户是否具有该能力
    if (Gate::denies('user-post', $post)) {
        // 不具备该能力
    }
}
```

✓ 检查指定用户是否具有某能力：forUser( )

```php
if (Gate::forUser($user)->allows('update-post', $post)) {
  //
}
```

用户授权—Gate能力

■ 检查能力：

✓ 直接通过用户模型对象检查能力：can( )、cannot( )

```php
public function update(Request $request, Post $post)
{
    // 通过用户模型对象检查能力
    if ($request->user()->cannot('user-post', $post)) {
        // 不具备该能力
    }
}
```

✓ Blade模板中使用：@can指令

```
<a href="/posts/{{$post->id}}">查看</a>
@can('update-post', $post)
<a href="/posts/{{$post->id}}/edit">编辑</a>
@endcan
```

用户授权—Policy策略

- 定义策略：php artisan make:policy 策略名称 --model=模型类名
  - ✓ 生成的policy位于app/Policies目录下
  - ✓ 默认生成一个空的策略类，--model参数针对CURD操作的策略类
- 注册策略：AuthServiceProvider类中注册
  - ✓ $policies 属性：存放所有实体与策略间的映射对
- 校验授权：
  - ✓ Gate门面方式：Gate::allows( )、Gate::denies( )
  - ✓ 指定用户：Gate::forUser( )->allows( )、Gate::forUser( )->denies( )
  - ✓ 授权用户方式：Auth::user( )->can( )、Auth::user( )->cannot( )
  - ✓ 控制器方法：$this->authorize( )
  - ✓ Blade模板：@can 指令

```php
protected $policies = [
    'App\Model' => 'App\Policies\ModelPolicy',
    Post::class => PostPolicy::class,
];
```

目录

CONTENTS

1 用户认证

2 用户授权

3 发送邮件

4 中间件

配置文件

Laravel中发送电子邮件非常简单，只需要在 config/mail.php 配置文件（或 .env文件）中设置必要的配置项，即可使用 Illuminate\Mail\Mailer 类提供的方法发送邮件。

■ 常用配置项：

- ✓ MAIL_DRIVER：邮件发送引擎，默认为 smtp
- ✓ MAIL_HOST：邮件服务器地址，若使用smtp，则为smtp发送服务器地址
- ✓ MAIL_PORT：邮件发送协议端口号
- ✓ MAIL_USERNAME：发件人邮箱地址
- ✓ MAIL_PASSWORD：发件人邮箱设置的客户端授权密码（注意：不是邮箱密码）
- ✓ MAIL_ENCRYPTION：是否使用SSL连接
- ✓ MAIL_FROM_ADDRESS：邮件发件人邮箱地址，必须与 MAIL_USERNAME 相同
- ✓ MAIL_FROM_NAME：邮件发件人显示的名称

163邮箱和QQ邮箱SMTP配置

- 163邮箱配置：

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.163.com
MAIL_PORT=25
MAIL_USERNAME=hbynlsl@163.com
MAIL_PASSWORD=laravel123456
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=hbynlsl@163.com
```

- QQ邮箱配置：

```
MAIL_DRIVER=smtp
MAIL_HOST=smtp.qq.com
MAIL_PORT=465
MAIL_USERNAME=475167303@qq.com
MAIL_PASSWORD=afbgojwmoiohcbbg
MAIL_ENCRYPTION=ssl
MAIL_FROM_ADDRESS=475167303@qq.com
```

发送邮件

- Illuminate\Mail\Mailer 类提供的发送邮件方法：

✓ 设置收件人：$message- >to( )

✓ 设置抄送人：$message- >bcc( )

✓ 邮件主题：$message->subject( )

✓ 添加附件：$message->attach( )

✓ 发送邮件：

```php
// 发送邮件
Mail::raw("邮件发送成功！", function($message) {
    $message->subject('测试邮件');
    $message->to('hbynlsl@163.com');
    $message->bcc('hbynlsl@hotmail.com');
});
```

- 发送纯文本邮件：**Mail::raw(邮件内容, 闭包函数)**
- 发送视图邮件：**Mail::send(邮件视图文件, 视图变量, 闭包函数)**

```php
Mail::send('emails.register', ['userName'=>'张三'], function($message) {
    $message->subject('注册邮件');
    $message->to('hbynlsl@163.com');
});
```

目录

CONTENTS

1 用户认证

2 用户授权

3 发送邮件

4 中间件

中间件

■ 中间件：middleware，用于用户认证、权限校验、日志文件……

✓ 使用流程：

① 定义中间件：php artisan make:middleware 中间件名称

② 注册中间件：全局注册、路由注册、中间件组注册、控制器注册

✓ 中间件使用的注意事项：

① 中间件位于 请求之前 或 请求之后

② handle( ) 方法 和 terminate( )方法

```php
public function handle($request, Closure $next)
{
    DB::enableQueryLog();
    return $next($request);
}

public function terminate()
{
    dd(DB::getQueryLog());
}
```

使用中间件

某些场合（如用户认证）需要在控制器执行之前使用中间件处理，为控制器添加中间件有两种方法。一是定义路由时，指明中间件；一是直接在控制器的构造方法中指定中间件。

■ 定义路由时指明中间件

```
// 定义路由时，指明控制器所采用的中间件
Route::get('profile', 'UserController@show')->middleware('auth');
```

■ 控制器构造方法中指明中间件

```
class UserController extends BaseController {
    public function __construct() {
        $this->middleware('auth');
        $this->middleware('log')->only('index');
        $this->middleware('subscribed')->except('store');
    }
}
```

验证

- 验证使用流程：
  - ✓ 编写验证逻辑并在控制器中验证：validate( )方法
  - ✓ 显示错误信息：在视图中使用 $errors 变量
    - $errors 为 Illuminate\Support\MessageBag 实例对象
- 创建表单请求验证类： php artisan make:request 验证类名
- 验证规则

感谢聆听！

THANK YOU FOR YOUR ATTENTION