Laravel 知识点

## 基础问题

1. 何为软删除,以及实现原理：

①并没有进行数据库的物理删除，只是在逻辑上进行删除;

②通过deleted\_at字段来实现逻辑删除，这个字段记录着逻辑删除的时间;

③一旦deleted\_at为非空，则表明此模型已被“软删除”，当查询时候，也会被自动从 查询结果中排除;

④I.如果想查询出已被“软删除”的模型，则需要使用withTrashed()方法II.如需恢 复已被“软删除”的模型，则需要使用restore()方法;

1. 如何对数据库查询做全局限制

①max\_user\_connections限制用户最大连接数

②max\_connection限制最大连接数

③DB::connection()->disableQueryLog();在Lavaral中禁掉查询日志

④读写分离:

'read' => array(

            array('host' => '192.168.xx.yy','port' => 3306),

            array('host' => '192.168.xx.zz','port' => 3306),

        ),

        'write' => array(

            'host' => '192.168.xx.xx', 'port'=> 3306

        ),

1. 使用Laravel自带的用户认证的好处,能获得哪些功能

①I. validator注册新用户的验证规则; II注册成功可以用create()直接插入数据库一条记 录;

②I. Auth::user()或者通过$request->user获取已登录用户;II. Auth::check() 可以判断某个用户是否登录

③只有登录用户才能访问给定路由

④Illuminate\Foundation\Auth\ThrottlesLogins trait登录失败次数限制，默认5次，锁定60秒;

1. 何为中间件,有哪些使用场景

对于一个Web应用来说，再一个请求真正处理前，我们可能会对请求做各种各样的判 断，然后才可以让它传递到更深层次中。会使得代码越来越难以维护，耦合度会增大。 而中间件可以解决这一问题，可以将这些判断独立出来做成中间件，以过滤请求.另外， 再请求处理之后，我们也许会紧接着执行一些操作，这些也可以用中间件来完成

场景:①表单验证②访问控制③注册后发邮件

1. 如何实现数据库多语言

写入数据库时每个语言都写一条记录，用一个列名区分语种，读取的时候根据url选择不同的语言

1. 如何在多个视图间共享数据

这些数据要放到AppServiceProvider里面，打开app/Provider,把要共享给所有视图的数` 据放到它的boot这个方法里面,用一个 view(),接着再用一个 share,把要共享的数据放 到这个方法里面,第一个参数是数据的名字,比如用一个 site\_name,第二个参数是具体的 值

1. 同样的表单数据,save多次,laravel会如何处理

①如果这个模型已经存在，Laravel只是用where 查询id去更新这个记录,也就只是 updated\_at这个字段会改变;

②如果是一个新模型，Laravel会插入数据库并生成id.

1. 如果有多种用户(PC前台/微信前台/后台),如何设计用户表,一张还是多张,各有哪些优缺点

①只用一张表，用login\_terminal字段（登录类型）来区分是哪个终端登录的，用user\_type 区分是前台还是后台用户，这样只需要用一套登录，注册逻辑，和用户信息session 的验证，用权限去控制用户的行为即可;

②用多张表，将前台，后台，pc端，微信端，移动端分开，这样的好处是根据各自不 同的业务需求设计表和字段、索引，能优化数据库

1. 如何使用laravel的队列

①1. 实现生产者

[PHP](http://lib.csdn.net/base/php" \o "PHP知识库" \t "http://blog.csdn.net/hel12he/article/details/_blank) artisan make:command PushMessage –queued

执行完后，可以在app目录下看到 app\Commands\PushMessage.php，新建一个 app\Commands\Command.php，PushMessage.php的handle方法里写上要执行的内容， 在控制器里将这个生产者生成的内容推进队列里边：Queue::later(60, new PushMessage());

② 队列数据表保存的地方

打开 config/queue.php，更改默认队列驱动为数据库，php artisan queue:table   
 php artisan migrate，就是在数据库中创建一个队列表，用来保存任务

③队列的消费者

当它监听到有要执行的任务时，就会自动按照指定条件执行。

php artisan queue:listen

## 进阶问题

1. 如何将日志名修改为 xxx.log.20170203

①在 config/app.php中的对应项设置

'log' => 'Daily',

②Laravel log日志是写死在Illuminate\Foundation\Bootstrap\ConfigureLogging里,

I . 继承他，在configureSingleHandler中写上:

$log->useFiles($app->storagePath().'/logs/xxx.log');

II.在App\Console\Kernel 、 App\Http\Kernel这两个地方我们要替换掉预设的 ConfigureLogging，换成我们自己写的类

1. 多个环境下如何设置多套evn配置

在.env中写入当前环境的名称，，比如现在有本地开发和线上两个环境 清空.env文件，只第一行写入当前运行的环境，比如 local 然后新建 .local.env 和 .production.env 分别写入本地和线上的配置项 将以下代码加入bootstrap/app.php 中

$env = $app->detectEnvironment(function () {

$environmentPath = \_\_DIR\_\_ . '/../.env';

$setEnv = trim(file\_get\_contents($environmentPath));

if(file\_exists($environmentPath)) {

putenv("APP\_ENV=$setEnv");

if(getenv('APP\_ENV') && file\_exists(\_\_DIR\_\_ . '/../.' . getenv('APP\_ENV') . '.env')) {

Dotenv::load(\_\_DIR\_\_ . '/../', '.' . getenv('APP\_ENV') . '.env');

}

}

});

1. 如何自定义异常,生产环境如果出现问题如何查看异常

①//自定义的异常类，继承了PHP的异常基类Exception

class customException extends Exception

 {

 public function errorMessage()

  {

  //error message

  $errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()

  .': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';

  return $errorMsg;

  }

 }

Laravel在App\Exceptions\[Handler](http://laravelacademy.org/tags/handler" \o "View all posts in Handler" \t "http://www.tuicool.com/articles/_blank)统一进行异常处理的，App\Exceptions\[Handler](http://laravelacademy.org/tags/handler" \o "View all posts in Handler" \t "http://www.tuicool.com/articles/_blank)的 父 类是Illuminate\Foundation\Exceptions\Handler。如果需求高了，可以try catch 后 再 根据情况再抛更详细的异常。

②生产环境如果出现异常，找到对应的异常名字，看看是哪里抛出的，然后去laravel.log 里面找到相应时间节点，看看前后代码执行是否有相关信息，找到错误原因

1. 如何进行模块化开发

①采用模块式开发，即每一个模块为一个子目录，子目录下有自己的MVC

/app/

  /modules/

     /article/

        /controllers/

        /models/

        /views/

    /video/

        /controllers/

        /models/

        /views/

②Pingpong/sky 扩展实现 Laravel 模块化开发

I .安装

composer快捷安装：

composer require "pingpong/sky:~2.1"

composer.json修改

{

"require": {

"pingpong/modules": "~2.0"

}

}

composer install

II.配置文件中添加服务

配置文件：config/app.php

Pingpong\Modules\ModulesServiceProvider::class,  
'Module' => Pingpong\Modules\Facades\Module::class,

发布扩展信息

php artisan vendor:publish

III.添加自动加载

修改文件：composer.json

{

"autoload": {

"psr-4": {

"App\\": "app/",

"Modules\\": "modules/"

}

}

}

IV.生成测试模块

php artisan module:make Blog

1. 如果用户可以用户名密码登录,也可以微信登录,也可以email登录,以后也可以接入更多登录形式,如何设计用户表

①用户基础信息表 users  
 id  
 nickname  
 avatar

②用户授权信息表 user\_auths  
 id  
 user\_id  
 identity\_type 登录类型（手机号 邮箱 用户名）或第三方应用名称（微信 微博等）  
 identifier 标识（手机号 邮箱 用户名或第三方应用的唯一标识）  
 credential 密码凭证（站内的保存密码，站外的不保存或保存token）

1. 上线前需要进行哪些优化

①关闭debug

打开.env文件，把debug设置为false.

②缓存路由和配置

php artisan route:cache

php artisan config:cache

③Laravel优化命令

php artisan optimize

④composer优化

sudo composer dump-autoload optimize

⑤artisan view:clear 清理 所有在 storage/framework/views/ 下的文件.