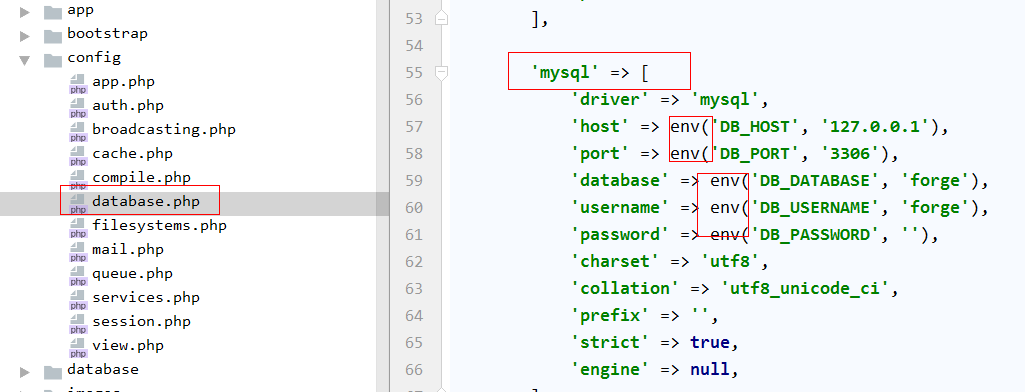
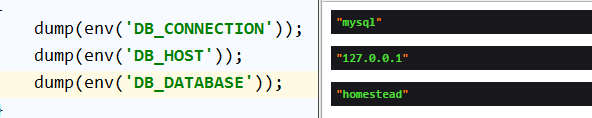
# 数据库

## 数据库的配置

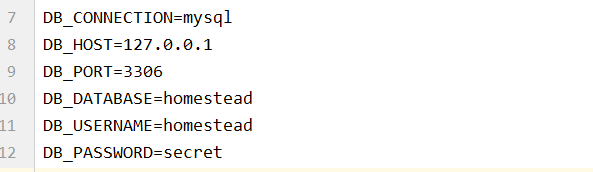
1. config/database.php文件中是所以关于数据的配置。



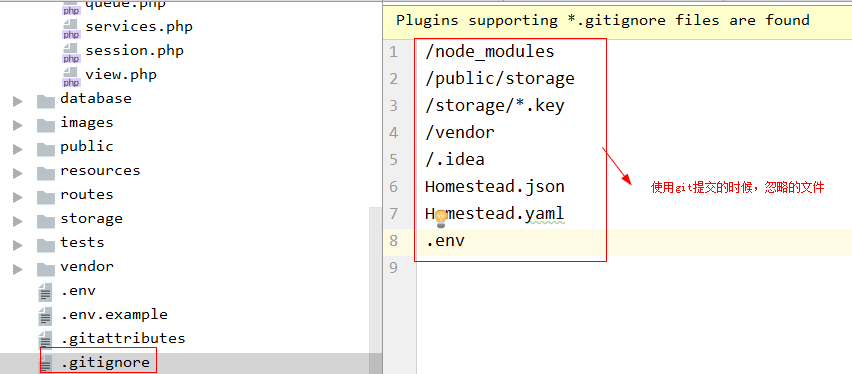
1. 发现配置文件中大部分的都是适应env()的函数，这个函数用户读取.env文件的配置信息



在修改数据的配置的，只需要去修改.env文件中的配置项

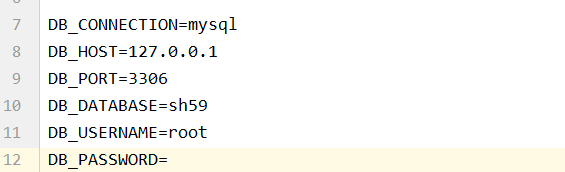


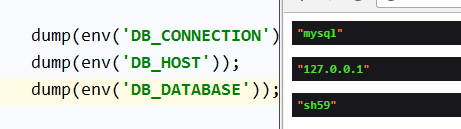
3、在使用git提交的时候，可以将不需要提交的文件忽略。



1. 本地的数据库配置

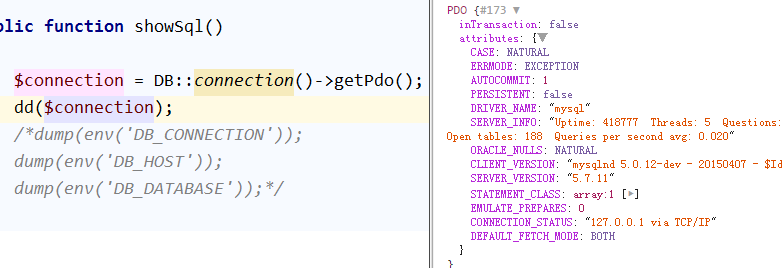
修改.env文件即可





## 测试数据库连接

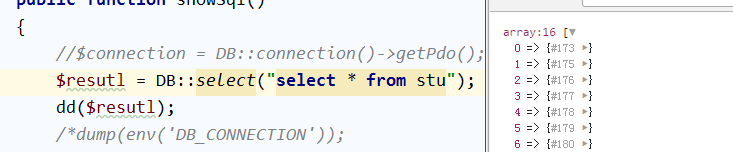
使用DB门面，一个类文件



## 执行原生的sql语句

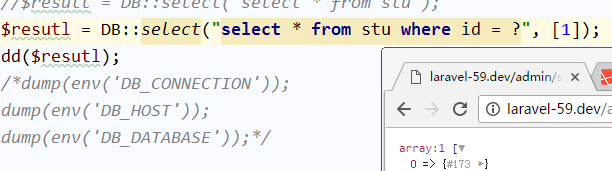
### 查询

1. 查询一下所以的数据

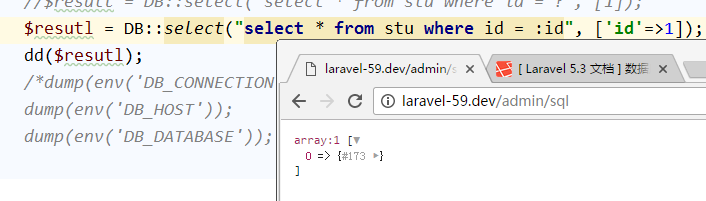


1. 使用预处理查询数据

使用?



使用:id的方法



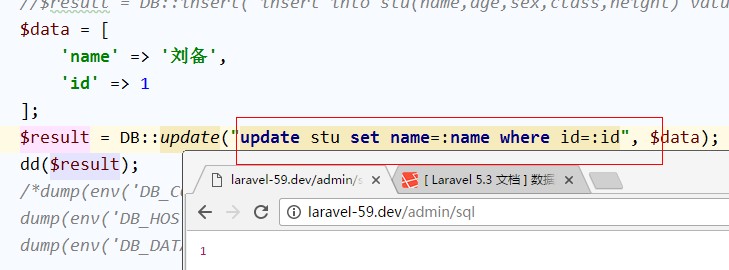
### 插入

使用预处理的方式完成数据的添加



### 修改

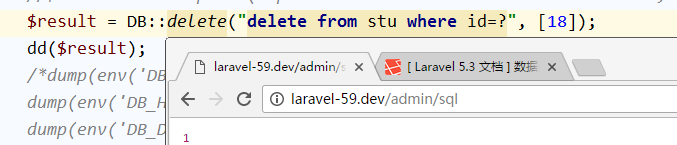
执行成功返回的是影响的行数





### 删除

执行成功返回的是影响的行数。





## 监听器

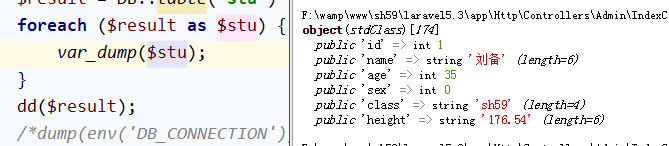
将执行的sql



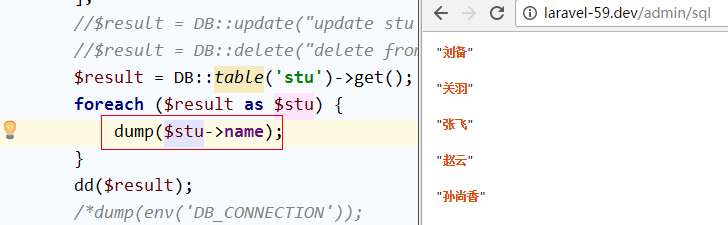
## 查询构建器

### 获取结果集

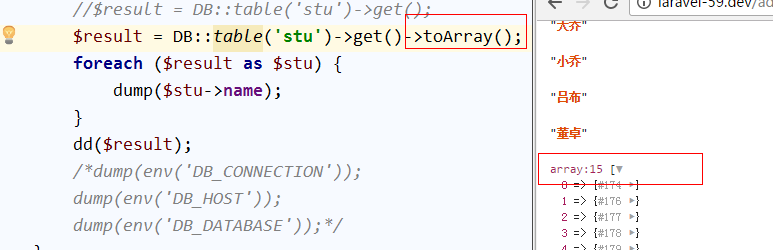
1. 查询所有的数据，使用get方法，返回的是对象



使用对象调用里面的字段名，得到数据

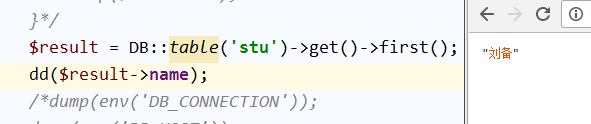


1. 可以将返回的集合转换为数组

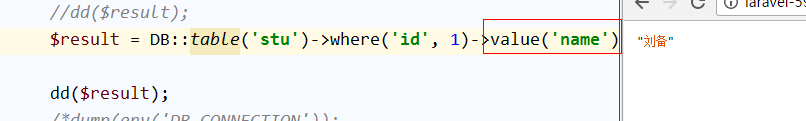


1. 获取一个记录/一个字段的值

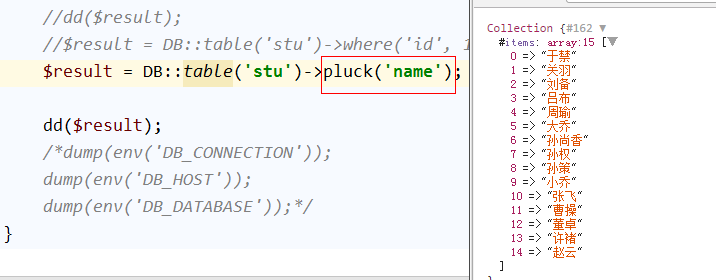
获取结果集中第一条记录



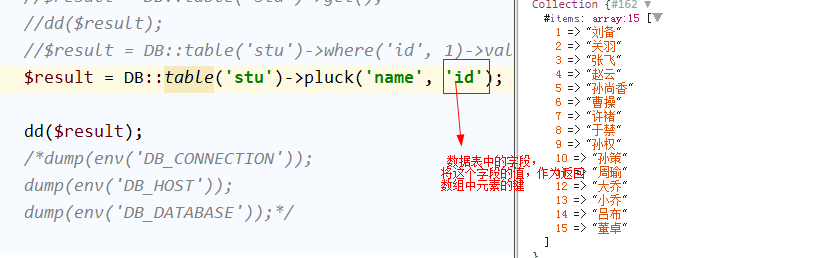
获取第一个记录条中指定的字段的值



获取一个字段的所有的值

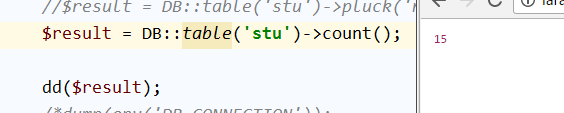


可以指定某个字段作为元素的键

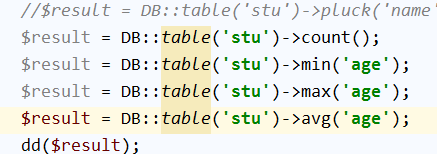


### 聚合函数

数据中的count，max，min，avg..

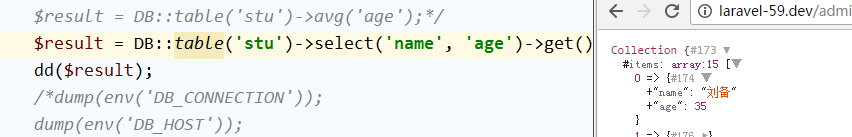


其他的聚合函数的使用

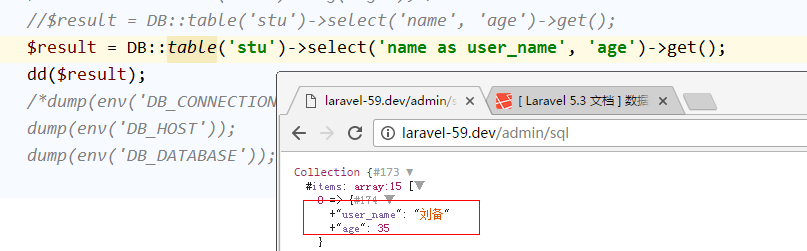


### 查询指定字段

1、使用select(),可以指定查询的字段

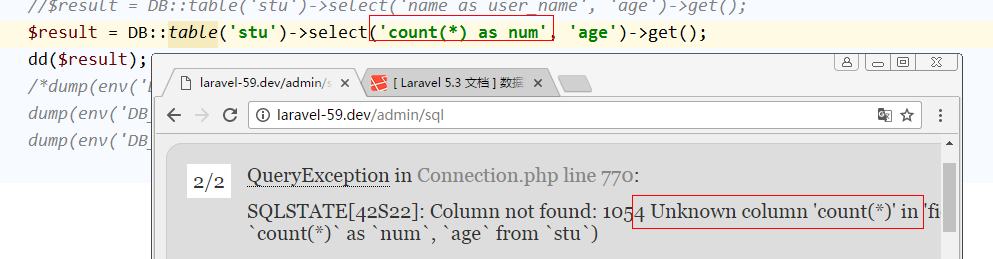


1. 可以为字段去一个别名



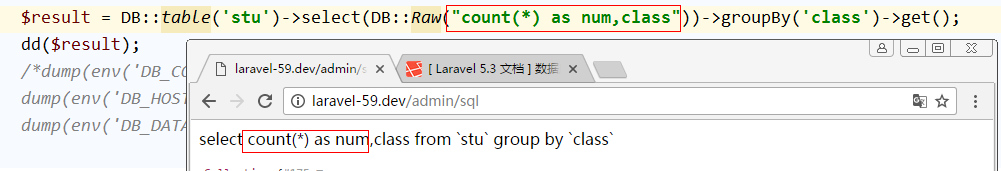
1. 统计写一下有多少学生。

Select count(\* ) as num from stu



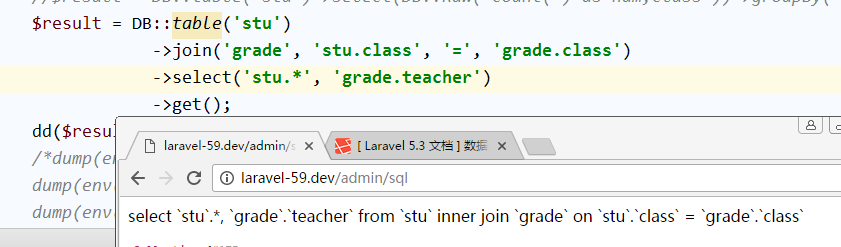
直接将count(\*)当成了一个字段

Raw(“count(\*) as num”);

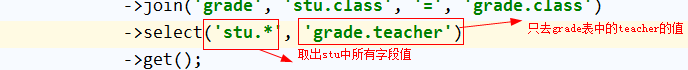


### 连接查询

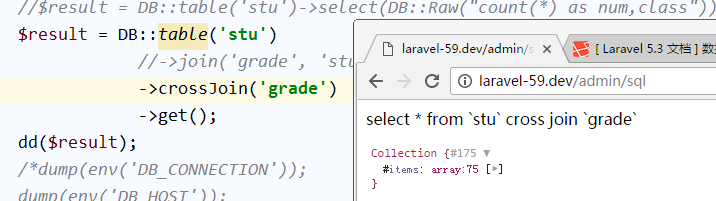
1. 内连接join方法



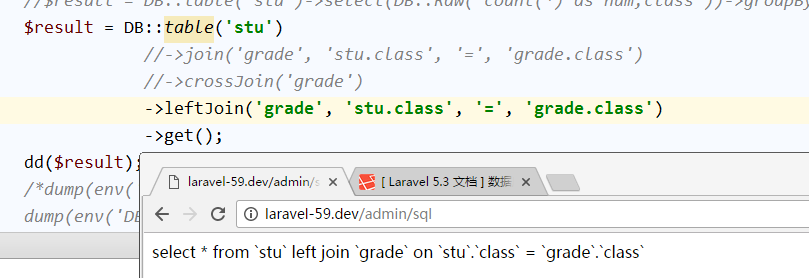
从两个表取出数据，如果两个表中，的字段重复，转为结果集的时候，后面的字段的值，会将前面的相同的字段的值覆盖。



1. 交叉连接

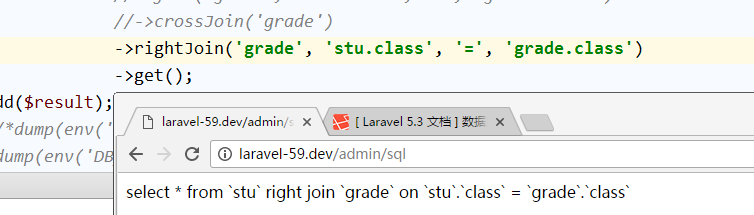


1. 左连接



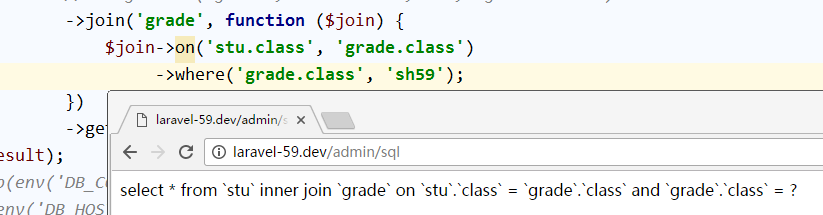
左表中所以的记录都被查询出来，如果右表中没有对应的数据，使用null表示

1. 右连接



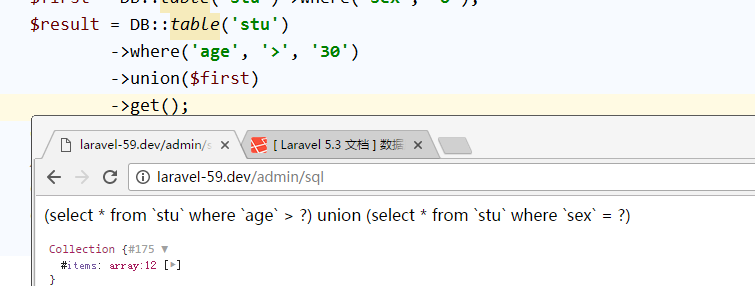
将右表中所以的记录查询出来，如果左边中没有对应的数据，使用null表示。

1. 高级的连接查询



### 联合查询

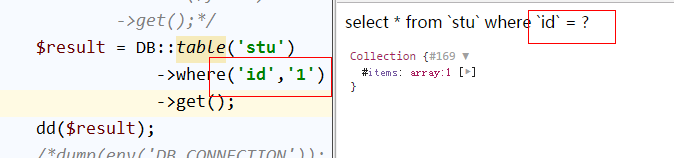
Union：先将根据某一个添加查询，再将这个查询联合到另外查询语句。

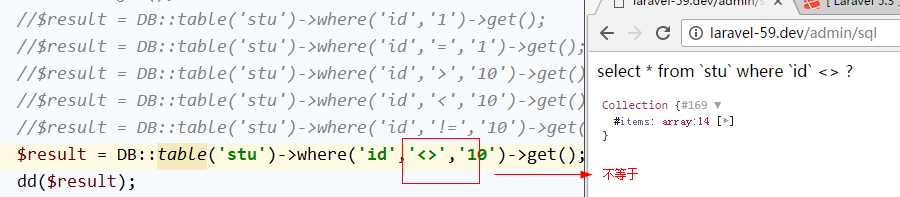


### 子句

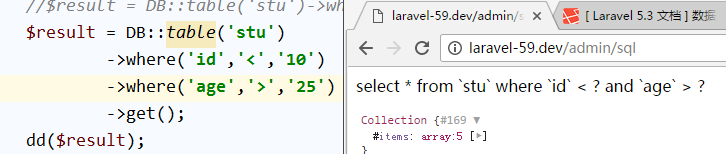
#### 简单的where子句

1. 根据比较运算符进行查询的

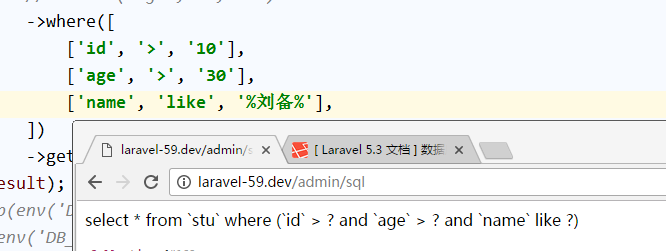
  
如果没有传递第二个参数，表示等于运算符的操作



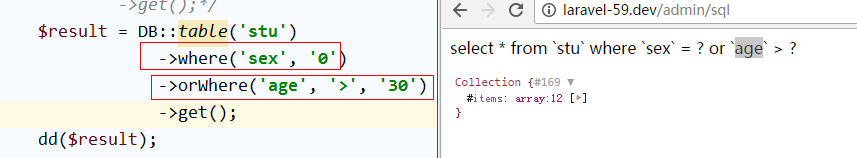
1. 根据where完成逻辑与的操作



1. 可以传递多个参查询提件



#### 逻辑或的操作



#### 其他的where子句

whereBetween/whereNotBetween

wherein/whereNotIn

whereNull/whereNotNull

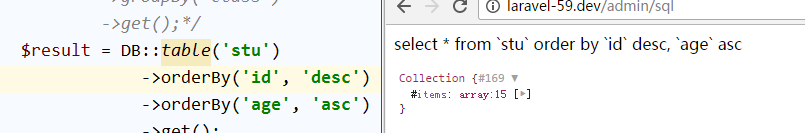
#### 分组，排序，限定

1. groupBy()，按照某一字段进行分组查询



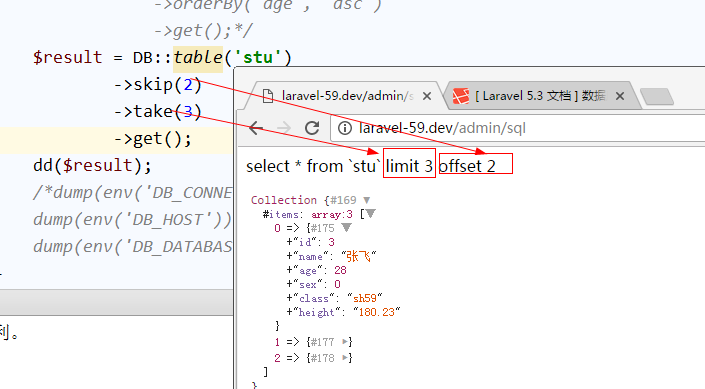
1. orderBy()按照某字段进行排序

第一个参数：字段，第二个参数：排序的方式(desc,asc)

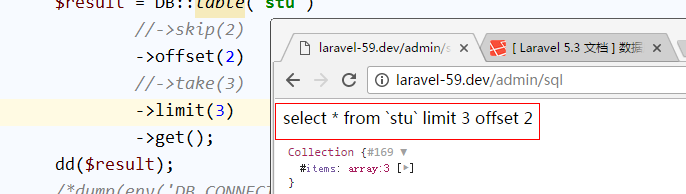


可以按照多个字段进行排序。

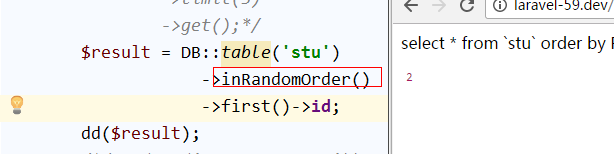
1. limit，使用skip(),设置的是起始位置，take()，取多少条



使用 offset和limit

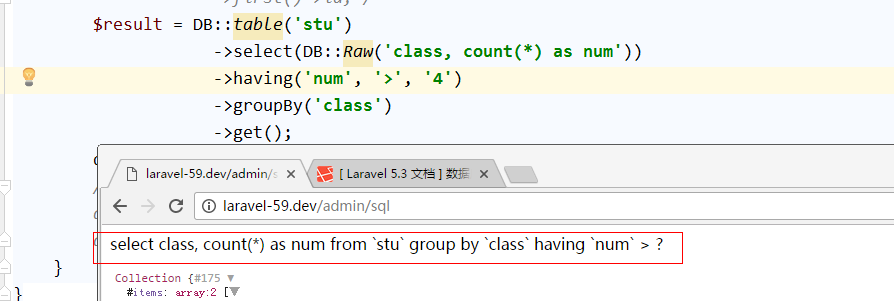


1. inRandomOrder:随机排序

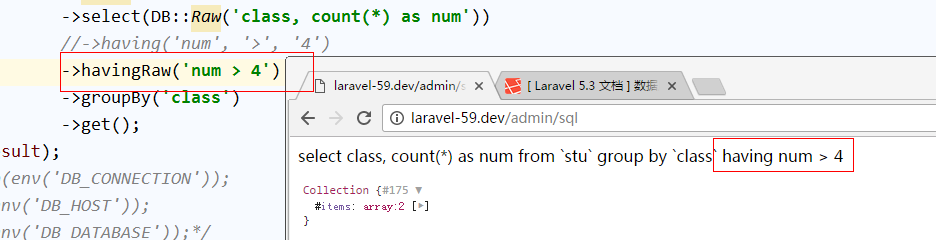


#### Having子句

1. 将查询的结果再一次进行过滤

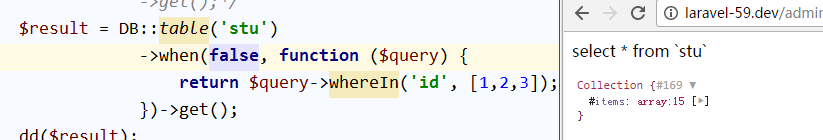


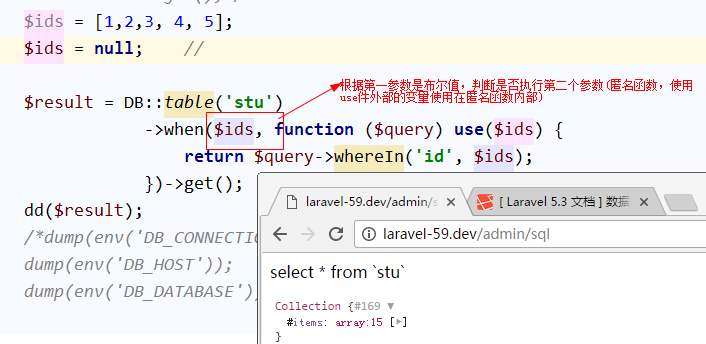
1. havingRaw():执行原生的sql语句



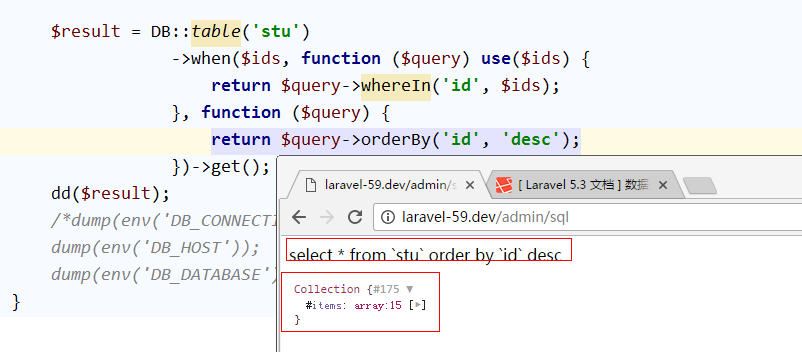
### 条件子句

Php中业务逻辑的条件。

1. 使用when的方法，完成条件(php中的true或者false)子句  
   



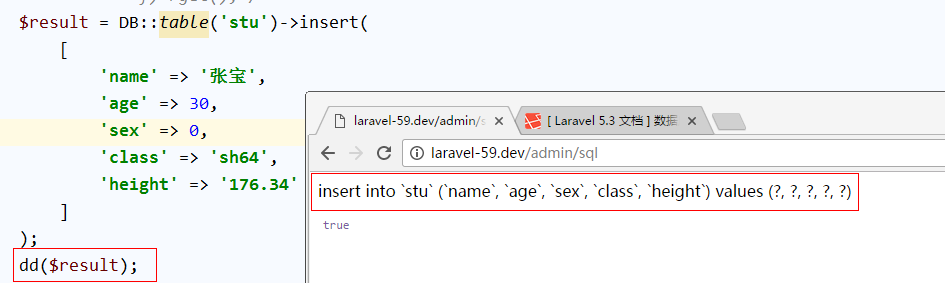
1. 当第一个参数的值为false，执行第三个参数



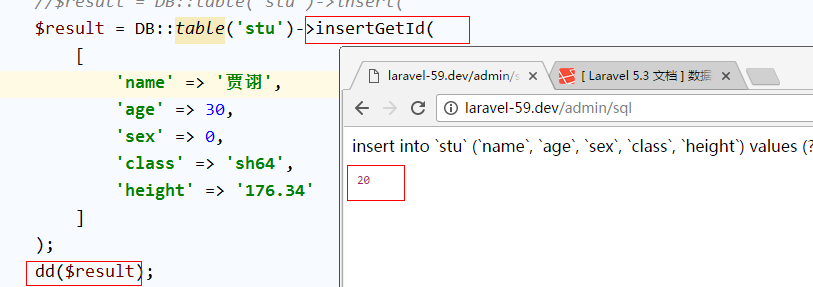
也是一个匿名函数

### 插入

1. 传递一个数据向数据库中添加一条记录



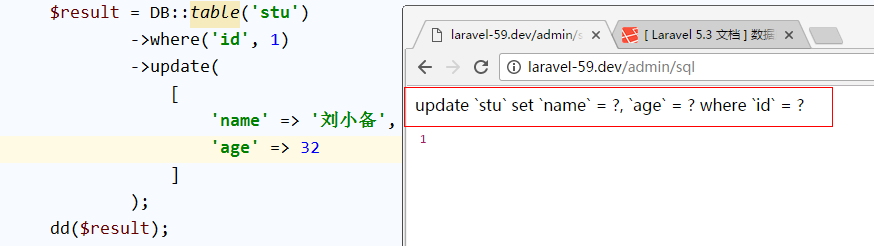
1. 获取当前插入的记录的id





### 修改

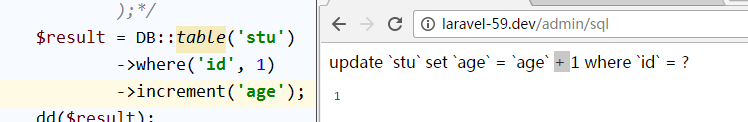
1. 根据指定的一个用户的id修改当前用于的值



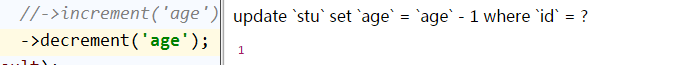


1. 可以直接增加或者减少数值。

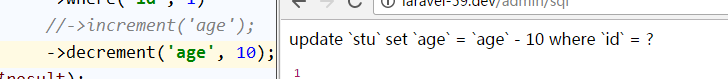
增加数量

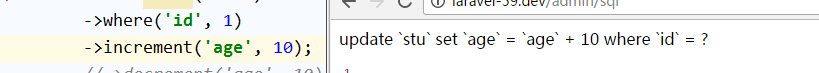


减少数量



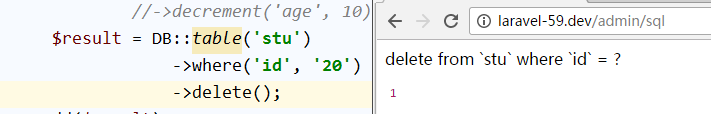
可以指定步长





### 删除

1. 调用delete()，方法



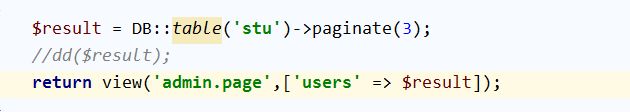


1. truncate(),清空表。

## 分页

### 基本使用

1. laravel提供基于boostrap前端css框架的一个分页的html

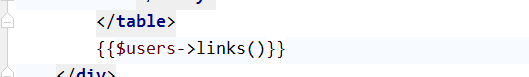




默认传递的是page，作为分页的参数



1. 在视图里面去调用一个links();就可以完成分页的html

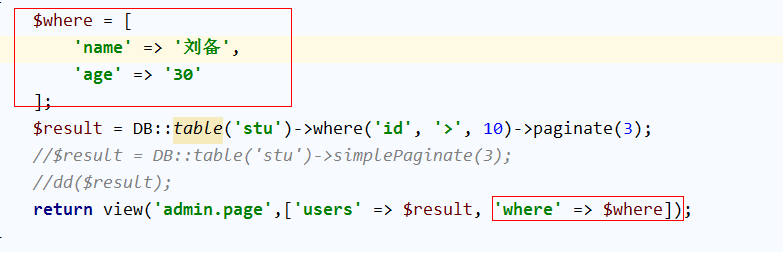




3、简单分页实现





1. 传递查询的参数。  
   

在视图里面使用appends方法即可。



1. 手动的设置，url



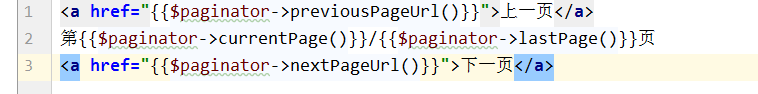
SetPath()



### 自定义分页视图

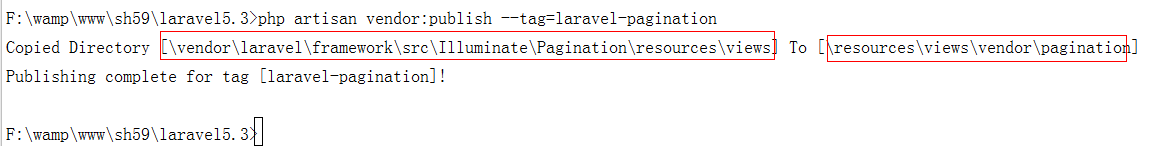
1. 创建一个视图

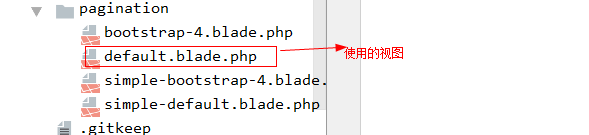




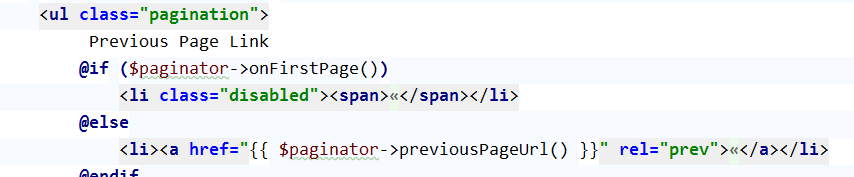
1. 导出laravel中提供的分页视图

使用artisan命令





1. 将laravel提供的视图，复制到用户自定义的视图里面



视图文件

### 分页中常用的方法

1. 是否存在分页

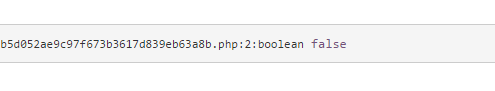




1. 判断是否是第一页

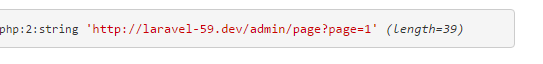


如果是返回true，否则返回false



1. 上一页的url





1. 所有分页的url是一个数组





数组中键是当前的页码，值是对应的url

1. 判断是否还有更多页



1. 下一页的url



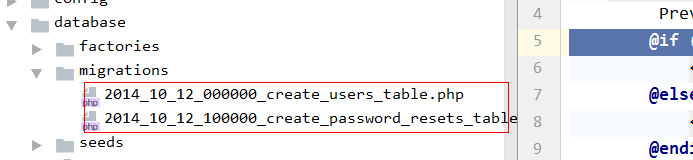


## 迁移

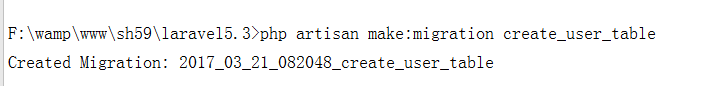
可以使用laravel中的操作，去生成一个数据表

### 定义迁移

1. 迁移的目录

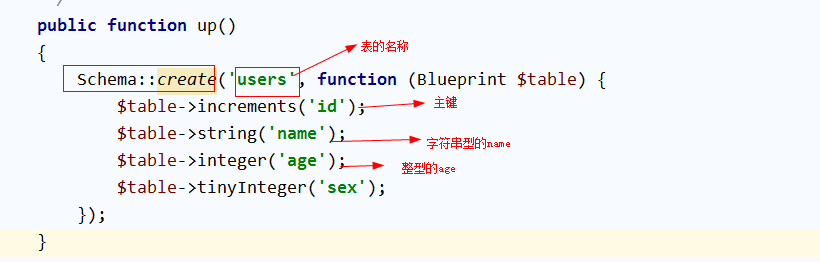


1. 使用artisan的命令

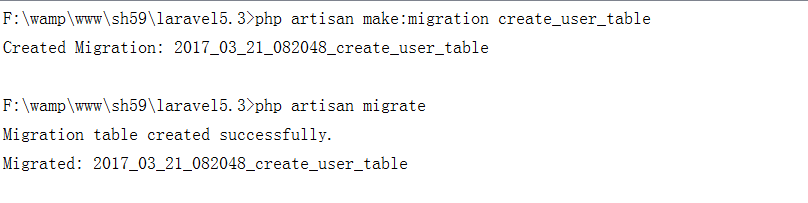




### 编辑迁移文件



### 运行迁移



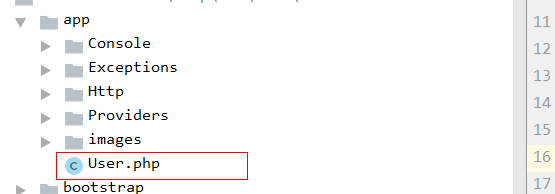


# Eloquent ORM

Laravel将一张表当初一个模型。操作的不一定使用是DB，采用的是模型。

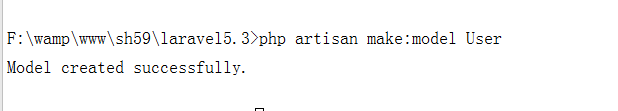
## 创建模型

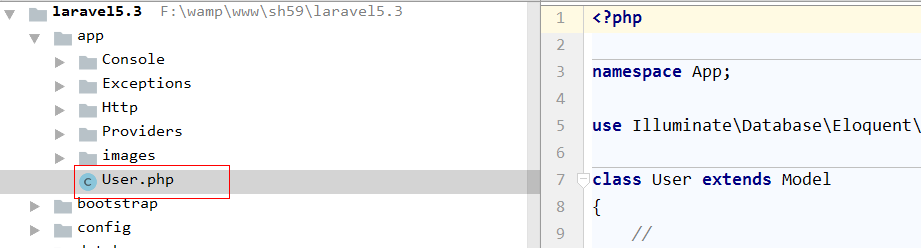
1. 默认是在app目录下创建的



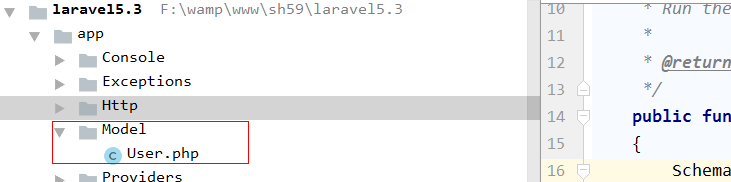
1. 创建一个模型

使用artisan命令



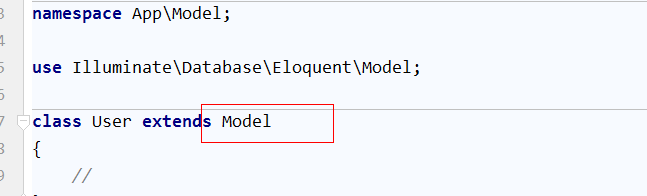


1. 需要在某一个模型的目录中

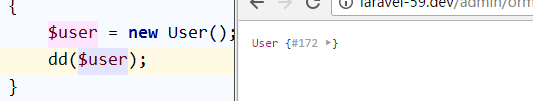


## 简单使用

1、自定义的模型会继承laravel中的基本模型类model

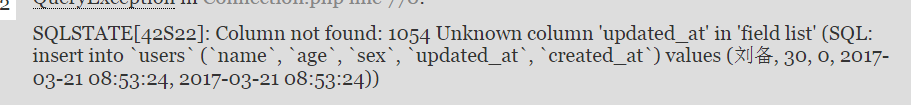


1. 得到模型类的实例



1. 操作user表，想表中添加数据

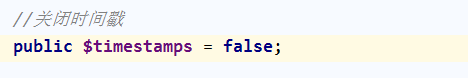
出现了以下的错误

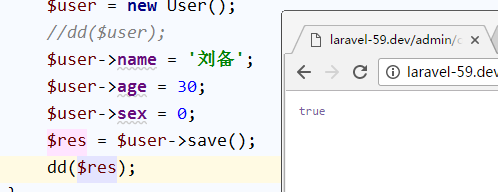


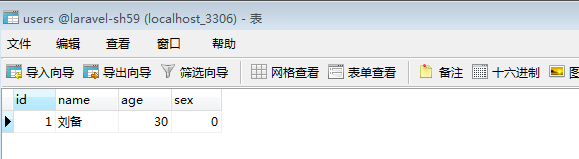
Laravel会自动在sql添加两个字段的，update\_at,create\_at.保存字段数据的修改时间和创建时间。

禁用时间戳

在模型中定义

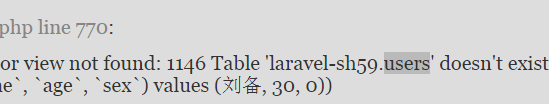


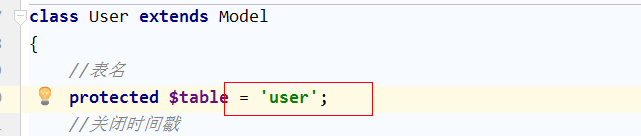


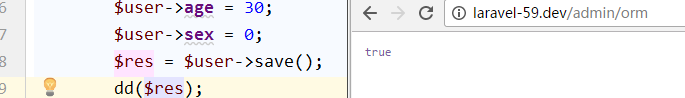


自定义表名

Laravel会自动根据模型去查询数据库中的表。User –>users.，默认加上s。





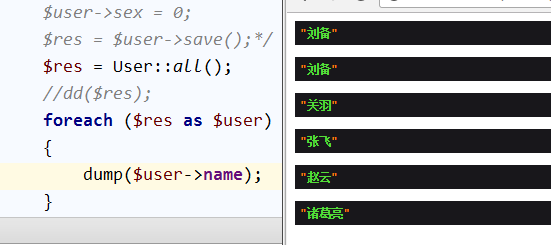




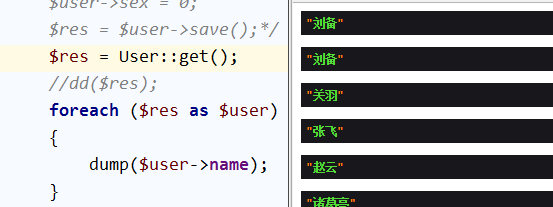
## 获取模型

将数据表中的所有的记录，都当成是一个模型来进行操作。

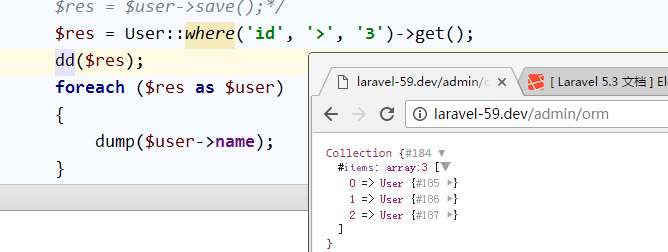
1. 获取所有的模型(记录)



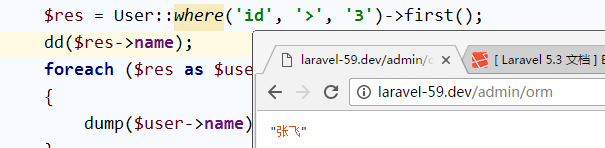
2、使用get方法也可以获取模型



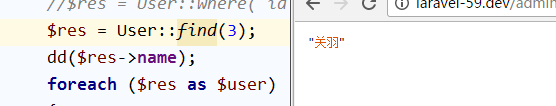
1. 根据条件获取模型



1. 得到第一个模型



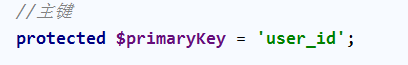
1. 使用find的方法根据主键得到模型



Laravel默认将id作为表的主键。如果一个表的主键不是id。

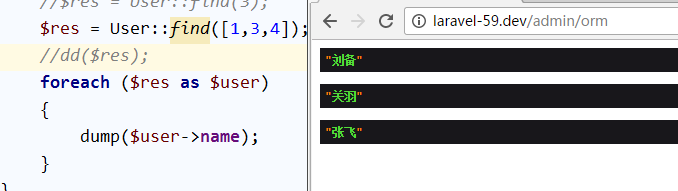


在模型中指定主键

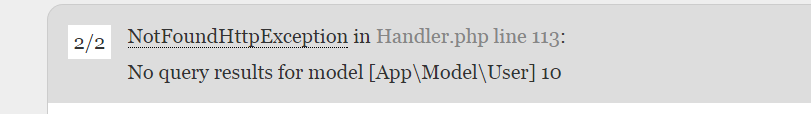


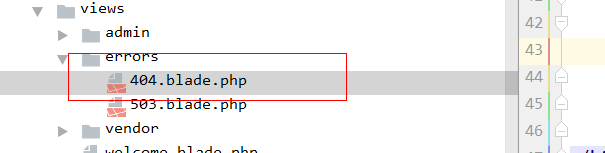


1. 使用find方法，根据主键获取多个模型



1. 使用findOrFail获取模型

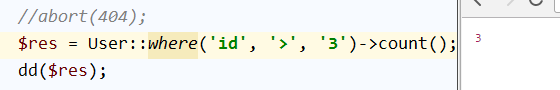




使用abort方法，自动跳转到404页面。

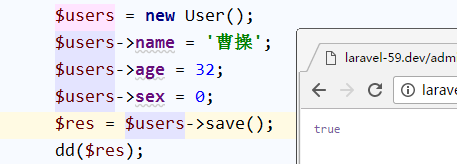


## 获取聚合



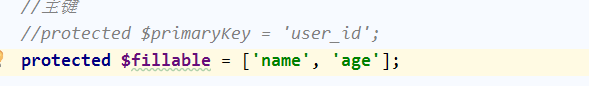
## 插入操作

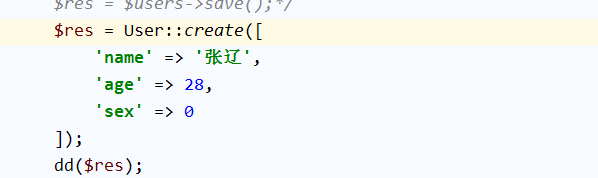
1、获取的是一个表模型，执行save方法



1. crate方法，完成数据的添加

在使用create的时候，需要在模型中设置一个白名单。





设置黑名单，不允许被插入的值

