**Laravel 03**

1. 视图
2. 什么是视图？

>>>视图是整个应用程序的显示页面，包含HTML，将控制器与演示逻辑分开

1. 视图所在目录？

>>>resources\views

1. 创建视图

>>>视图文件使用.blade.php为文件的拓展名

1. 解析模板

>>>view(‘所在目录.文件名称’)

1. 携带数据到模板

>>>view(‘所在目录.文件名称’,[‘id’=>1])

1. 模板引擎blade
2. 模板默认位置

>>>resources\views

1. 模板使用变量

>>>{{ $name }}

1. 模板使用函数

>>>{{ time() }}

1. 设置默认值

>>>{{ $name or ‘李四’ }}

1. 显示html代码

>>>{!! $html !!}

1. 引入子视图

>>>@include(‘header’)

1. 模板继承

>>>1.主模板占位符

\*\*\*1.@yield(‘名字’)

\*\*\*2.@section(‘名字’)

@show

>>>2.子模板内容

\*\*\*1.继承 @extends(‘目录名/文件名’)

\*\*\*2.赋值01 @section(‘名字’,’内容’)

\*\*\*3.赋值02 @section(‘名字’)

内容

@endsection

1. 流程控制

>>>格式

@if(条件)

内容

@elseif(条件)

内容

@else

内容

@endif

1. 循环控制

>>>for

@for($i=0;$i<9;$i++)

内容

@endfor

>>>foreach

@foreach($info as $k=>$v)

内容

@endforeach

1. 数据库操作
2. 数据库连接配置

>>>1.快速配置 .env文件

>>>2.基本配置 config\database.php

1. 数据库基本操作

>>>1.查询 DB::select()

>>>2.插入 DB::insert()

>>>3.更新 DB::update()

>>>4.删除 DB::delete()

>>>5.事务回滚

\*\*\*1.开启事务回滚 DB::beginTransaction()

\*\*\*2.回滚 DB::rollBack()

\*\*\*3.提交 DB::commit()

1. 数据库构造器

>>>1.插入

\*\*\*1.单条 DB::table(‘表名’)->insert(一维数组);

\*\*\*2.多条 DB::table(‘表名’)->isnert(二维数组);

\*\*\*3.获取id插入 $id = DB::table(‘表名’)->insertGetId(数组);

>>>2.更新

\*\*\*DB::table(‘表名’)->where(‘id’,’=’,1)->update(数组);

>>>3.删除

\*\*\*DB::table(‘表名’)->where(‘id’,’=’,1)->delete();

>>>4.查询

\*\*\*1.查询所有 DB::table(‘表名’)->get();

\*\*\*2.查询单条 DB::table(‘表名’)->first()

\*\*\*3.查询单条指定字段 DB::table(‘表名’)->value(‘指定字段’)

\*\*\*4.查询多条指定字段 DB::table(‘表名’)->lists(‘指定字段’)

>>>5.连贯操作

\*\*\*1.设置字段--指定字段查询

DB::table(‘表名’)->select(‘name’,’email as u\_email’)->get()

\*\*\*2.条件 where orWhere whereBetween

DB::table(‘表名’)

->where()

->get()

\*\*\*3.排序

DB::table(‘表名’)->orderBy(‘name’,’desc’)->get();

\*\*\*4.分页

DB::table(‘表名’)->skip(10)->take(5)->get()

\*\*\*5.连接表

DB::table(‘goods’)

->leftJoin(‘types’,’goods.type\_id’,’=’,’types.id’)

->get()

\*\*\*6.计算

总数Count() 最大值max() 最小值min()

1. 模型
2. 模型位置

>>>app\任意位置

1. 模型创建

>>>Php artisan make:model 目录\模型名

1. 模型使用

>>>引入DB类操作即可

1. 控制器使用模型

>>>Use 导入模型类

>>>实例化类

>>>调用成员方法

1. 设置自定义函数
2. app目录下手动创建任意文件夹 如Common
3. 在Common目录下创建自定义函数
4. 在composer.json中autoload下添加路径



1. Composer dump-auto使用
2. 控制器内直接调用方法名
3. 调试工具
4. debugbar安装

>>>1.composer require barryvdh/laravel-debugbar

>>>2.在config/app.php里面的providers添加

Barryvdh\Debugbar\ServiceProvider::class,

1. Chrome插件postman