随笔

运行Vagrant

vagrant init -- 初始化vagrant

vagrant up -- 启动vagrant

vagrant halt -- 关闭vagrant

vagrant ssh -- 通过SSH登录vagrant(先启动vagrant)

vagrant provision -- 重新应用更改vagrant配置

vagrant destroy -- 删除vagrant

vagrant provision && vagrant reload -- 重新应用更改vagrant配置 并是重启虚拟机使更改生效。

一.

创建并运行一个 Laravel 项目；

1.切换镜像加速

composer config -g repo.packagist composer https://mirrors.aliyun.com/composer/

2.install ( "8.\*" 代表版本号);

<!-- composer create-project laravel/laravel Laravel --prefer-dist "8.\*" -->

composer create-project laravel/laravel Laravel --prefer-dist "5.5"

GIT 使用

1.配置全局用户

git config --global user.name "Your Name"

git config --global user.email your@example.com

2.设置默认推送方式 simple,避免push时未指定分支报错

git config --global push.default simple

3.git 命令

git init

git log

git checkout -f #丢弃当前所有修改

git clean -df #删除本地所有修改

git checkout . #清空本地修改(当前分支)

4.绑定github仓

git remote add origin 'github仓地址'

git push -u origin master

5.创建分支，合并分支，删除分支

git checkout -b aaa #创建指定分支

git merge aaa #合并分支aaa

git branch -d aaa #删除分支aaa

git reset --soft HEAD^ #撤销commit

heroku

heroku login -i

heroku keys:add

heroku create #创建应用

heroku rename <appname> #重命名

php artisan key:generate --show

heroku config:set APP\_KEY=<your\_app\_key> #设置key

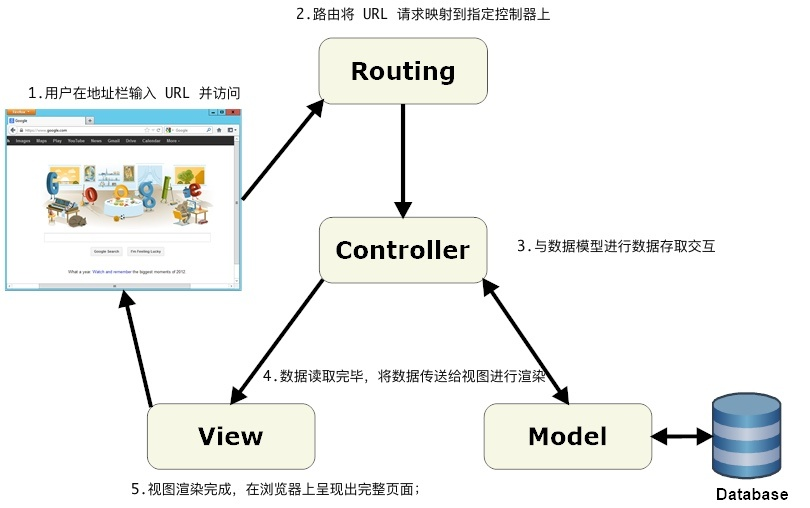
heroku open

     git remote rm heroku #删除源

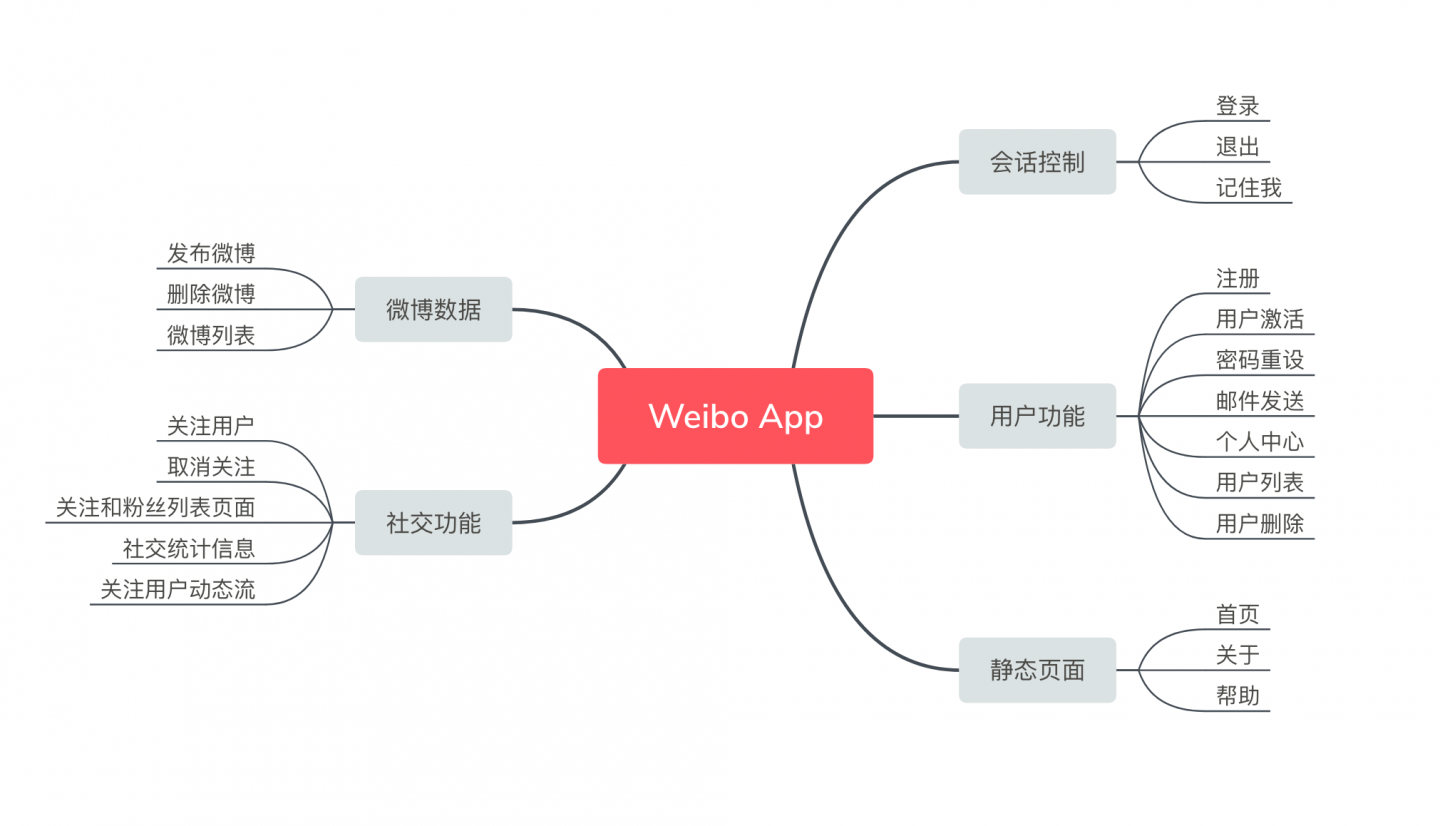
1. 网页访问的完整过程 (如图)  
    1.浏览器输入URL访问  
    2.路由将url映射到指定控制器  
    3.控制其接受请求后开始处理,如需动态数据渲染，则控制器开始从模型中读取数据

4.数据读取完毕，传输给视图进行渲染

5.视图渲染完成，呈现到界面



项目功能图



1. 四 （页面构建，页面优化）

NPM    命令

        npm run dev #启动

        npm run watch-poll #检测并自动编译

SASS

sass 语法

        可写变量 $a : blue

        可嵌套写 div{

            a{

                color:$a;

            }

            b{

            }

        }

        可应用父选择器 &

        a{

        }

        a:hover {

            color: blue;

        }

        a{

            color:white;

            &:hover{

                color: blue;

            }

        }

Laravel前端缓存问题

mix.js('resources/js/app.js', 'public/js')

    .sass('resources/sass/app.scss', 'public/css').version();

五 用户模型

    数据库迁移

        php artisan migrate #迁移

        php artisan migrate:rollback #回滚

用户模型  
 php artisan make:model Models/User

用户注册

    CSRF 保护

        {{ csrf\_field() }}

    old() #显示form表单提交失败时的旧数据

    bcrypt() # 加密

    heroku上使用PostgreSQL

        heroku addons:add heroku-postgresql:hobby-dev

        heroku config #查看配置

        heroku config:set IS\_IN\_HEROKU=true #新增配置项查看是否在heroku环境

        heroku pg:reset DATABASE #重置数据库

        heroku run php artisan migrate #数据库迁移

会话管理

    {{ method\_field('DELETE') }}

    method\_field()创建隐藏域 模拟delete 等请求

    登录

        Auth::check()  bool #验证用户是否登录

        Auth::attempt(array $credentials,bool $remember) #reuqest请求数据 remember（记住我） 验证

    验证

        validate() #验证器

        $this->validate($request, ['name' => 'in:field1,field2']);

用户CURD

    intended() #友好的跳转方式

    {!! $users->render() !!} #laravel分页集成

    trait 申明的类 可以被子类继承多个

邮件发送

    生成令牌

        Eloquent模型触发的一个事件。事件是 Laravel提供的监听器

        boot(){

            static::creating(){

            }

        }

微博

    显示微博

        $status->created\_at->diffForHumans() #可读性更佳的日期展示形式。

批量赋值假数据时需要设置白名单

粉丝关系

添加关联数据

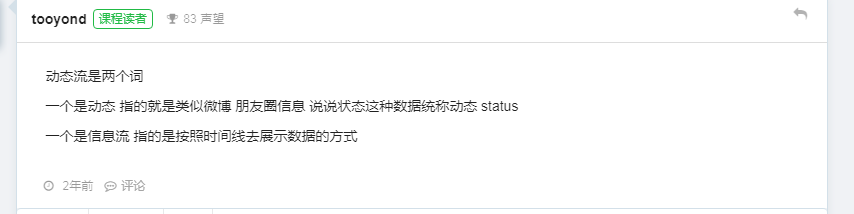
        sync([],false); #已存在时则不添加

    删除关联数据

        detach([]);

动态流

<https://learnku.com/laravel/t/9827/definition-of-dynamic-flow-where-is-the-feed-method-called>



    使用预加载 with()来避免N+1 查询问题;

