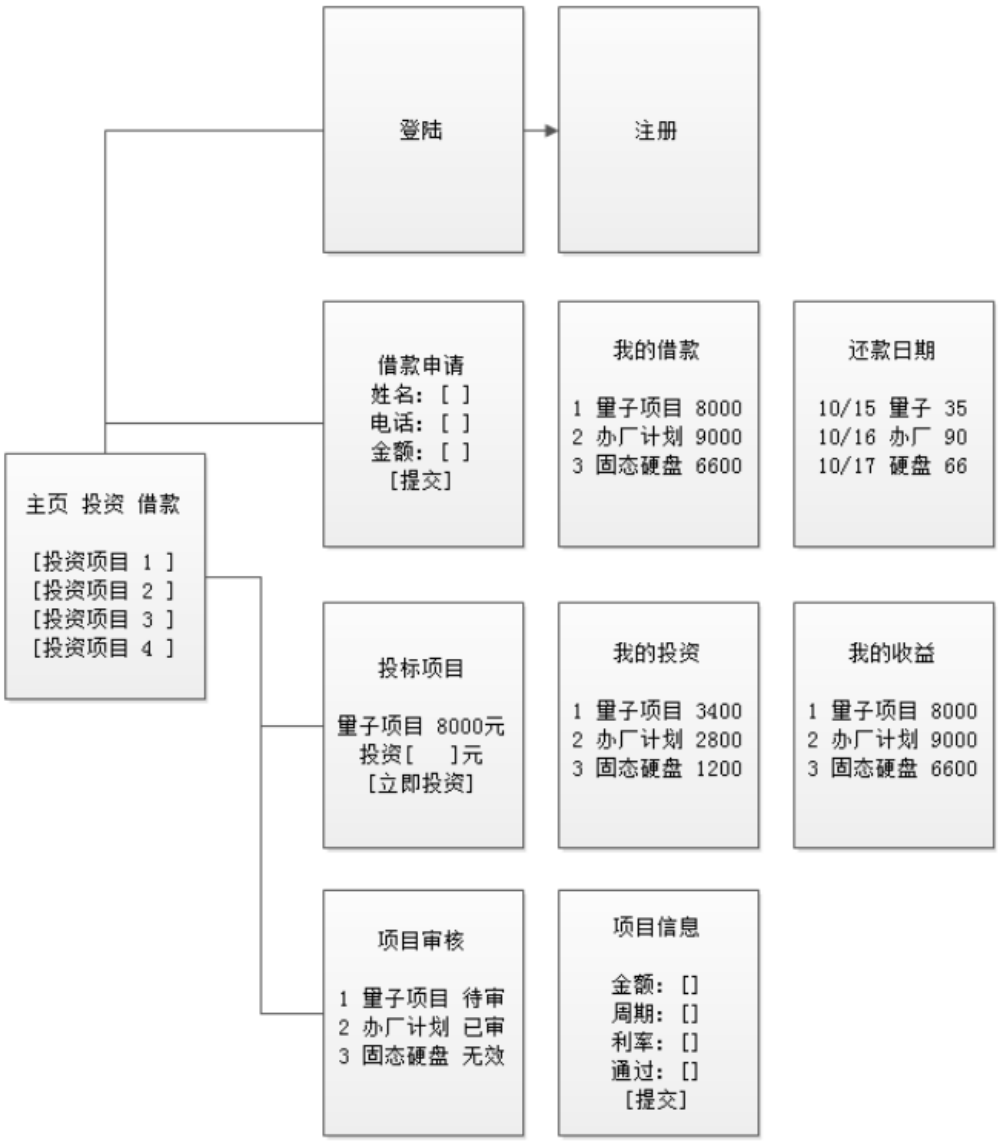
Laravel 5.5 入门教程

By IT崖柏图
Mail 973714522@qq.com
出自 布尔教育PHP高端教育培训

14章 p2p网贷项目开发

14.1 功能分析

p2p , e 租宝 , 人人贷 .
商业模式:
在豪华地段,租最豪华办公室,招模特做前台,一水的 170,大长腿.
注册资本,怎么着也得一个亿吧,反正不用真实出资.
再到敬老院雇一重病老头做法人代表(关键!).
市场宣传,承诺高额回报,利率 15% 起,我说的是月利息!
没客户怎么办?在央视黄金时段砸广告.线下销售,电话销售,狂拉单子.
付不出利息怎么办?没事,用下一个客户的本钱付上个客户的利息.
当客户本金积累到10亿时,法人代表及时的死掉.
我们的钱哪去了?鬼知道 !
基本功能:



14.2 准备模板

把点点贷模板解压放在/resources/view 下，
把解压出的image , css 目录移动到/public/css 目录下
写一个简单路由器测试：
Route::get('test/index' , function(){return view('/index');});
并把index.html 重命名为 index.blade.php
发现页面可以输出，但变形了，这是因为 css 文件路径不对 .
浏览器f12 打开控制台，以common.css 为例：
xx.com/test/common.css 404
这是因为当前 URL 是xx.com/test/index , 而源码中
所以形成了对xx.com/test/css/common.css 的请求 .
我们把所有模板的 css,image 路径，都换成相对于网站的根目录/ 就可以避免此问题 .
批量替换所有模板css/ --> /css/ , image/ --> /image/
再次刷新,页面已正常显示.

14.3 表分析

** 用户表 : users **

字段	类型	说明
uid	primary	key 主键
name	string	用户名
email	string	电子邮箱
mobile	string(11)	手机号
password	string(60)	密码
regtime	integer	注册时间
lastlogin	integer	上次登陆时间
rember_token	rememberToken()	记录用户 cookie

借款表 :projects 用到时再分析
借款附属表 :atts
投资表 :bids
收益表 :grows
还款表 :hks
流水表 :logs

14.4 迁移文件

创建迁移命令：
php artisan make:migration create_users_table --create=users

但是，系统已经准备好了，只需要我们修改一下;

```
// users 表
public function up()
{
    Schema::create('users', function (Blueprint $table) {
        $table->increments('uid');
        $table->string('name');
        $table->string('email')->unique(); //加入唯一索引
        $table->string('mobile',11);
        $table->string('password', 60);
        $table->rememberToken();
```

```
    $table->timestamps();
     });
}
```

修改完成后，执行迁移操作：
php artisan migrate

15 章 认证与授权(用户注册和登陆)

15.1 准备工作

laravel 自带了用户认证与授权类 , 可以方便的帮我们完成认证与授权.
主要用到: `App\Http\Controllers\Auth\LoginController及RegisterController 以及users 表`

laravel 要求users 表至少有如下字段 :

- name 字段
- email 字段
- password 60 个长度
- remember_token 100 个长度

由于我们的uesrs 表主键叫uid,且没有created_at,update_at 这两个时间字段,
因此,按 Model 的约定,做如下说明.

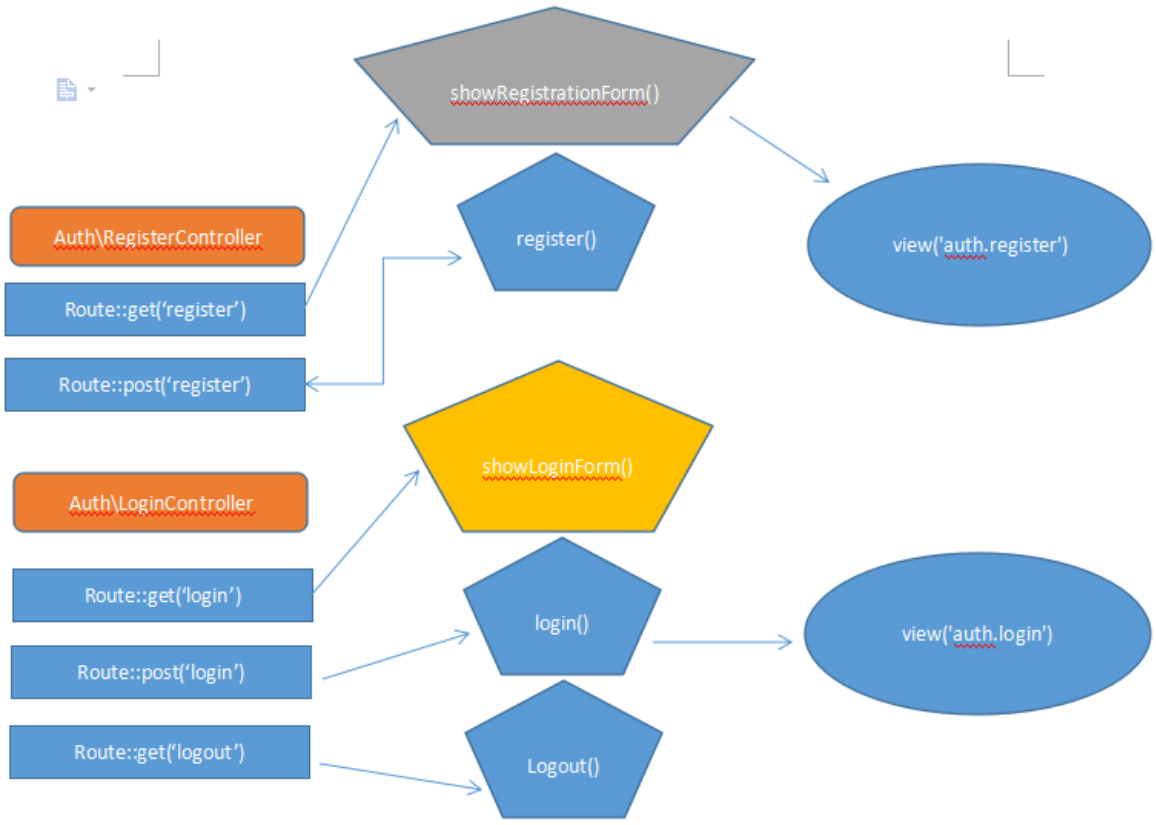
```
protected $primaryKey = 'uid';
public $timestamps = false;
```

两个重要文件：

```
/vendor/laravel/framework/src/Illuminate/Foundation/Auth/RegistersUsers.php
/vendor/laravel/framework/src/Illuminate/Foundation/Auth/AuthenticatesUsers.php
```

> 由于使用laravel自带的用户认证类,那么使用人家的东西呢就需要遵循人家设置的规则
> 我们使用`Auth`文件夹下的`RegisterController`注册控制器类以及`LoginController`下的登录控制器类
> 默认模板文件是放置在`views\auth\`文件夹下,对应的方法我们需要观察以下这两个文件同事结合图片使用

```
/vendor/laravel/framework/src/Illuminate/Foundation/Auth/RegistersUsers.php
/vendor/laravel/framework/src/Illuminate/Foundation/Auth/AuthenticatesUsers.php
```



`LoginController`和`RegisterController` 类有几个默认的属性如果需要,你可以 手动修改 这几个属性值).

自定义跳转路径

> 当一个用户成功进行登录认证后，默认将会跳转到/home，你可以通过在 `LoginController`， `RegisterController` 和 `ResetPasswordController`中定义 `redirectTo` 属性来自定义登录认证成功之后的跳转路径：

```
protected $redirectTo = '/home';
```

如果重定向路径需要自定义生成逻辑可以定义一个 `redirectTo` 方法来取代 `redirectTo` 属性：

```
protected function redirectTo()
{
    return '/path';
}
```

`redirectTo`方法优先级大于`redirectTo` 属性。

自定义用户名

> 默认情况下，`Laravel` 使用 `email` 字段进行认证，如果你想要自定义认证字段，可以在 `LoginController` 中定义 `username` 方法：

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

**退出默认跳转到 / **

15.2 RegisterController类工作原理

> 准备路由和模板

```
Route::get('register','Auth\RegisterController@showRegistrationForm');
Route::post('register','Auth\RegisterController@register');
```

> 在 `/resources/view` 下建立`auth` 目录，
> 把`register.html` 和`login.html` 放入`auth` 目录下，
> 并重命名为`register.blade.php` 和`login.blade.php` .
> 修改 `register.blade.php` 文件中的`POST`提交地址，表单中，添加 `token`验证；

```
<form method="POST" action="">
    {{ csrf_field() }}
```

注意：用户名邮箱不能重复，密码不能少于6位

> 此时提交，报SQL语句有错： 缺少时间字段，所以，别忘了你和Model的约定
> 添加：

```
protected $primaryKey = 'uid'; //指定主见
```

> 再次提交仍然报错，但是，数据已经添加到数据库，只是没有相应的home路由
> 只需要在路由中添加即可：

```
Route::get('home',function(){
 return 'welcome to center';
});
```

> `Laravel`注册成功默认是登录状态,所以需要先退出
> 准备退出路由

```
Route::get('logout','Auth\LoginController@logout');
```

查看数据库，注册用户的手机号为空，追踪Auth\RegisterController 类的源码：

```
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'password' => bcrypt($data['password']),
    ]);
}
```

> 发现在创建用户时，没有传递mobile 字段 .
> 修改如下：

```
protected function create(array $data)
{
    return User::create([
        'name' => $data['name'],
        'email' => $data['email'],
        'mobile' => $data['mobile'],
        'password' => bcrypt($data['password']),
    ]);
}
```

> 再次提交,仍然收不到`mobile`字段,再次查看`Model User.php`的文件,有如下属性:
> `protected $fillable = ['name', 'email' , 'password'];`
> 这个属性代表,外界对User Model 批量赋值时,Model 接收的字段.
> 修改如下：

> `protected $fillable = ['name', 'email' , 'mobile', 'password'];`
> 再次注册,此时数据库有mobile 数据了.

15.3 用户登陆 LoginController

> 添加路由：

```
Route::get('login','Auth\LoginController@showLoginForm');
Route::post('login','Auth\LoginController@login');
```

首先解决token错误

```
<form method="POST" action=" ">
{{ csrf_field() }}
```

> 用户名密码输入正确， 点击登陆没有反应， 也不报错；
> 在laravel中,如果表单校验失败,会自动传递一个 $errors 变量到view中
> 因此,我们可以在登陆模板中直接打印$errors来查看错误
> 模板中输出错误信息 {{print_r($errors)}}

> 再次登陆 , 会看到错误信息
> laravel 框架默认用email + password 字段来检测用户并登陆.
> 两种解决方案如下：

```
//修改模板中的表单名称
<input type="text" class="input_1 pos_u gray_border" placeholder="用户名"
id="username" name="email">

//在LoginController中添加属性,声明使用name配合password 登陆 回顾 13.1 章节
public function username(){
 return 'name';
}
```

> 此时已登录成功

15.4路由分组

路由分组：

```
//指定命名空间
Route::group(
    ['namespace'=>'Auth'],
    function(){
  Route::get('register','RegisterController@showRegistrationForm');
  Route::post('register','RegisterController@register');
  Route::get('login','LoginController@showLoginForm');
  Route::post('login','LoginController@login');
  Route::get('logout','LoginController@logout');
    }
);

//指定路由前缀

Route::group(
 ['prefix'=>'article'],function(){
 //你只需要在访问 'artcile/add'
 Route::get('add','ArticleController@add');
});

//我们还可以使用这种方法
//访问 article/create 响应 Admin\ArticleController@create
Route::prefix('article')->namespace('Admin')->group(function(

 Route::get('create','ArticleController@create');
```

```
));
```

15.5 手动登陆

如果有自己独特的独特登陆逻辑,可以在AuthController中手动重写登陆.

```
//引入Auth类
use Auth;

/*
* 此处省略很多原有代码……
*/

public function postLogin(){
    if( Auth::check()){ //判断是否已经登陆
        return redirect('/');
    }
    // Auth::attempt(['name'=>$_POST['name'],'password'=>$_POST['password']]) //实现登陆
    if(!Auth::attempt(['name'=>$_POST['name'],'password'=>$_POST['password']])){
        return redirect('/');
    }
    else {
        return redirect('home');
    }
}
```

退出：

```
Auth::logout();
```

> 主要感受laravel的验证机制，不建议写手动登陆代码，没必要；

16 章 借款功能

16.1 借款表及迁移文件

** 借款表 :projects **

字段	类型	说明
pid	primary key	主键
uid	integer	用户uid
name	string	用户名
money	integer	贷款金额
mobile	string(11)	手机号
title	string	项目名称
rate	tinyint	年利率 (百分比)
hrange	tinyint	还款期限,月为单位
status	tinyint	-1不通过，0 审核中 ,1 招标中 ,2 还款中 ,3 结束
recive	integer	已招标金额
pubtime	integer	项目发布时间

** 借款附属信息表 :atts **

字段	类型	说明
aid	primary key	主键
uid	integer	用户 uid
pid	integer	项目 pid
title	string	项目名称
realname	string	真实姓名
age	tinyint	年龄
gender	enum	(' 男 ',' 女 ') 性别
salary	tinyint	收入,千为单位
jobcity	string	工作城市
udesc	string	用户描述

迁移命令：

`php artisan make:migration create_table_projects --create=projects`

`php artisan make:migration create_table_atts --create=atts`

```
// projects 表 迁移文件
public function up()
{
    Schema::create('projects', function (Blueprint $table) {
        $table->increments('pid');
        $table->integer('uid');
        $table->string('name',10);
        $table->integer('money');
        $table->string('mobile',11);
        $table->string('title',50);
        $table->tinyinteger('rate');
        $table->tinyinteger('hrange');
        $table->tinyinteger('status'); # 0 审核中 ,1 招标中 ,2 还款中 ,3 结束
        $table->integer('recive');
        $table->integer('pubtime');
    });
}


// atts 表 迁移文件
public function up()
{
    Schema::create('atts', function (Blueprint $table) {
        $table->increments('aid');
        $table->integer('uid');
        $table->integer('pid');
        $table->string('title');
        $table->string('realname');
        $table->tinyinteger('age');
        $table->enum('gender',['男','女']);
        $table->tinyinteger('salary');
        $table->string('jobcity');
        $table->string('udesc');
    });
}
```

执行迁移文件： `php artisan migrate`

16.2 准备路由器和Controller

添加路由：

```
Route::get('jie','ProController@jie');
Route::post('jie','ProController@jiePost');
```

命令行创建控制器：php artisan make:controller ProController

```
public function jie(){
        return view('woyaojiekuan'); //修改模板，文件名，form表单地址，token等
}
public function jiePost(){
    return '我要借钱';
}
```

16.3 生成 Model

> 贷款申请牵涉到 2 张表,projects,atts; 使用命令，生成对应的 Model：php artisan make:model Pro php artisan make:model Att`

生成以后，别忘了你和Model的约定：

```
class Pro extends Model
{
 protected $table = 'projects';
    protected $primaryKey = 'pid';
    public $timestamps = false;
}

class Att extends Model
{
    protected $primaryKey = 'aid';
    public $timestamps = false;
//
}
```

16.4 借款方法

```
public function jiePost(Request $req){
    $pro = new Pro();
    $att = new Att();

    //主表
    $pro->money = intval($req->money) * 100;
    $pro->mobile = $req->mobile;
    $pro->pubtime = time();
    $pro->save();
    //上边new 完以后，对象对应的就是表，成为表对象，此时对象为空
    //下面又给表对象中的添加数据，最后执行save
    //此时的 pro 这个表对象，对应的就是刚刚执行完插入的那一行数据；

    //附表
    $att->age = $req->age;
    $att->pid = $pro->pid;  //此时可以得到表对象中的任何字段值；
    $att->save();
    return 'ok';
}
```

> 查看数据库发现，发布的projects表信息没有用户的uid,name信息.
> 我们需要从登陆信息中拿到这两条信息,并写入Project中.

16.5 获取登陆用户信息

laravel 中,获取当前登陆用户的对象,可以用如下方法:

```
$user = Auth::user();
$user = $request->user();
```

通过$user 实例，可以拿到当前登陆用户的信息 .
因此：

```
public function jiePost(Request $request) {
    //$user = Auth::user();
    $user = $request->user();
    ...
    $pro->uid = $user->uid;
    $pro->name= $user->name;
    ...

    $att->uid = $pro->uid;
    ...
}
```

17 章 项目审核

17.1 审核列表页

路由器 :

```
Route::get('prolist','CheckController@prolist');
```

生成控制器：php artisan make:controller CheckController

控制器 :

```
// file /app/Http/Controllers/CheckController.php
use App\Project;

public function prolist()
{
    $prolist = Pro::orderBy('pid','desc')->get();
    return view('prolist',['prolist'=>$prolist]);
}
```

view:

```
@forelse ($prolist as $p)
<tr>
    <td class="f">{{$p->pid}}</td>
    <td>{{date( 'Y/m/d H:i' , $p->pubtime)}}</td>
    <td class="tr"><span>{{$p->name}}</span></td>
    <td class="tr">{{$p->money/100}}</td>
    <td class="tr">{{$p->mobile}}</td>
    <td class="">
    @if ($p->status == 0)
    待审核
    @elseif ($p->status == 1)
    招标中
    @elseif ($p->status == 2)
    还款中
    @elseif ($p->status == 3)
    已结束
    @endif
    </td>
    <td class="l"><a href="{{ url('check' , [$p->pid]) }}"> 审核 </a></td>
</tr>
@empty
<div class="wujilu" id="errorMsg"> 暂无记录 </div>
```

```
@endforelse
```

17.2 项目 审核

> 路由： `Route::get('check/{pid}','CheckController@getCheck');`
> 控制器如下

```
// 审核项目 , 主要是修改 projects 表和 atts 表
public function check($pid) {
    $pro = Pro::find($pid);
    $att = Att::where('pid' , $pid)->first();
    if(empty($pro)) {
        return redirect('/prolist');
    }
    return view('shenhe' , ['pro'=>$pro , 'att'=>$att]);
}
```

view:

```
<form action="" method="post">
    {{ csrf_field()}}
    <input type="hidden" name="pid" value="{{$pro->pid}}">
    <tr>
        <td>真实姓名:</td>
        <td><input type="text" name="realname"></td>
    </tr>
    <tr>
        <td>借款金额:</td>
        <td><input type="text" name="money" value="{{$pro->money / 100}}"></td>
    </tr>
</form>
```

> 添加路由：Route::post('check/{pid}','CheckController@checkPost');
> 控制器中代码如下:

```
public function checkPost(Request $req,$pid){
    $pid = $req->pid;
    $pro = Pro::find($pid);
    $att = Att::where('pid' , $pid)->first();

    if(empty($pro)) {
        return redirect('/prolist');
    }

    $pro->title = $req->title; //项目名称
    $pro->hrange = $req->hrange; //借款期限
    $pro->rate = $req->rate; // 百分比
    $pro->status = $req->status; //审核结果

    $att->realname = $req->realname; //真实姓名
    $att->gender = $req->gender; //性别
    $att->age = $req->age; //年龄
    $att->udesc = $req->udesc; //借款人基本信息

    if( $pro->save() && $att->save() ) {
        return redirect('/prolist');
    } else {
        return 'error';
    }
}
```