**公司晚会选票系统**

**架构设计文档**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 日期 | 概要 | 开发人员 |
| Demo | 2018/03/08 | Demo版本发布 | 鲍雅娜 |
|  |  |  |  |

文档编写 鲍雅娜2018/03/08

1. **前言**
   1. **背景和需求**

|  |  |
| --- | --- |
| 场景 | 某公司使用剧院举办晚会 |
| 客户 | 该公司全体员工 |
| 需求 | 1. 员工通过系统购买晚会票 2. 员工每次可购买1-5张票 3. 座位由系统随机分配 |
| 剧院 | 1. 剧院分为A,B,C,D四个区域 2. 剧院所有区域座位相同 3. 第一排座位数为50 4. 最后一排座位数为100 5. 座位隔排递增2 |

* 1. **开发语言和搭建环境**

|  |  |
| --- | --- |
| 开发语言 | PHP7.1 |
| 架构模板 | MVC模板 |
| 开发框架 | Laravel5.6 |
| 前端开发语言 | HTML，CSS |
| 前端开发框架 | Bootstrap |
| 数据库语言 | MySQL |
| 运行环境 | 推荐XAMMP或者XNMMP环境 |

1. **MVC与Laravel**
   1. **MVC介绍**

MVC（Model, View, Controller），意思为模板，视图与控制器，是一种设计创建WEB应用程序的模式。

- Model（模型）为应用程序的核心，用来进行数据库的操作和各种核心算法的输入

- View（视图）显示数据，提交数据到控制器，用户UI界面

- Controller（控制器）对视图和模型双方提交来的数据进行处理，并进行交互处理

MVC模式将WEB应用程序的开发变得更加直观化和清晰化，不同的开发人员可以使用不同的开发视图，整个项目的架构也更加清晰明了。

* 1. **Laravel介绍**

Laravel是一款基于MVC设计模式的PHP框架，提供了一系列便捷的API接口，使PHP的开发更为快速优雅。Laravel的文档也非常的详细，可以让开发人员能够快速的查阅，便于开发人员接触和学习。此项目使用最新版本的Laravel5.6进行开发。

1. **设计架构**
   1. **系统流程图**
   2. **系统MVC进程图**
2. **设计要点**
   1. **计算座位，给出每一个座位唯一的座位号**

根据已给的信息，剧场分为A,B,C,D四个信息，每个区域的第一排座位都是50个，最后一排座位都是100个，每隔一排增加两个座位，将奇数排和偶数排看做两个单独的等差数列，根据等差数列公式可以计算出从50增加到100所需要的项数，因为此项目场景已给出第一排和最后一排，所以直接设为常量，进阶开发可以将第一排和最后一排定位变量，除开用户界面可以新建管理界面，使用到任何同样模式的不同座位数的剧场之中。

* + - 1. 计算50-100所需要的排数

private static function getExtensionRow()

{

try {

$first = 50;

$last = 100;

$d = 2;

$row = ($last - $first)/$d+1;

return $row;

} catch (\Exception $e) {

Log::alert('TheaterSeats::buildAllSeats : '.$e);

}

return null;

}

* + - 1. 根据排数计算出每一排的座位数

private static function buildSeatsInOneRow($row)

{

try {

$seats = array();

$first = 50;

$last = $first+($row-1)\*2;

for($i=1; $i<=$last; $i++) {

$seats[$i] = $i;

}

return $seats;

} catch (\Exception $e) {

Log::alert('TheaterSeats::buildAllSeats : '.$e);

}

return null;

}

* + - 1. 建立一个排数-座位号的数组，此为一个区域的所有座位号

public static function buildAllSeats()

{

try {

$row = self::getExtensionRow();

$odd\_seats = array();

$even\_seats = array();

/\*\*

\* 得到所有的奇数列数组

\*/

for($i=1; $i<=$row;$i++) {

$odd\_seats[$i] = self::buildSeatsInOneRow($i);

}

/\*\*

\* 得到所有的欧数列数组，

\* 最后一排为100，则座位数增加到100后不再增加，

\* 偶数排比奇数排少一排

\*/

for($i=1; $i<=$row-1;$i++) {

$even\_seats[$i] = self::buildSeatsInOneRow($i);

}

/\*\*

\* 合并奇数列和偶数列并根据座位的数量重新排列

\*/

$seats = array\_merge($odd\_seats ,$even\_seats);

usort($seats,array('self','mySort'));

return $seats;

} catch (\Exception $e) {

Log::alert('TheaterSeats::buildAllSeats : '.$e);

}

return null;

}

* 1. **用户被随机分配座位**

场景只设定了用户一次可购买的票数为1-5张，并没有用户购买票数的总数设定，因为晚会票需要购买并不是免费赠送的，所以并不考虑限制每一个用户购买晚会票的数量，但是此项目限制了每一个邮箱只能购买20张晚会票，所以用户若是想要购买超过20张晚会票则需要使用不同的邮箱，因为晚会票并不实名制，所以没有对用户姓名进行身份验证，用户姓名只是作为一个数据进行传递，留做进阶开发使用。

public static function randSelectSeats($user, $email,$number)

{

try {

if($number > 0) {

$results = new static();

/\*\*

\* 随机选取区域和排数

\*/

$row = $results->from($results->area\_table)->inRandomOrder()->first();

/\*\*

\* 根据区域和排数随机选取座位号

\*/

$seat = $results->from($results->seats\_table)

->where($results->seats\_table . '.row', '=', $row->row)

->inRandomOrder()

->first();

/\*\*

\* 将seats的格式转换为与user\_seats表中user\_seats的格式一样，便于对比

\*/

$seats = $row->area . ',' . $row->row . ',' . $seat->seat;

/\*\*

\* 确认所选择的的座位是不是已经存在数据库中

\*/

$validate = self::validateSeats($seats);

switch ($validate) {

/\*\*

\* 若是选择的座位不在user\_seats中，

\* 将座位号与用户信息写入到user\_seats表中，

\* 用recursion进行下一个座位的选择

\*/

case 1:

self::insertIntoUserSeats($user, $email, $seats);

self::randSelectSeats($user, $email, $number - 1);

break;

/\*\*

\* 若是选择的座位已经存在于user\_seats表中，

\* recursion重新开始选择座位

\*/

case 2:

self::randSelectSeats($user, $email, $number);

break;

default:

Log::alert('UserSeats::randSelectSeats : validate is wrong');

}

}

} catch (\Exception $e) {

Log::alert('UserSeats::randSelectSeats : '.$e);

}

}

* 1. **每个座位只能被选择一次，不能重复选择**

剧场的座位数量固定，但是每个座位只能对应一个人，所以程序必须进行validation以防止用户随机被分配座位时分配到了重复的座位，因为用户已选择的座位已经全部在数据库内，所以只需要进行数据库的查找，若是座位已经储存在数据库中，丢弃此选择重新开始选择，若是数据库中找不到座位号，选择座位并储存到数据库中。

* 1. **剧场有固定座位，所以此系统最大可接受的人数固定**

因为剧场的座位时固定的，并且公司员工数量也是固定的，假设此系统架设在公司的内部网络上，公司外部人员无法访问，所以不需要考虑大数据流量问题，数据库设置可以直接将剧场的所有座位建表插入数据，程序第一次运行时候讲座位插入到表中，所以控制器的BuildAllSeats方程只需要运行一次。

* 1. **此系统使用频率少**

此系统为公司内部举办晚会的购票系统，初次发布并不考虑面向外部使用问题，系统搭建在公司内网上，非本公司员工无法访问公司内网，所以不进行身份验证。并且考虑到公司并不是每天都会使用系统，所以此系统主要考虑的是简便性，能够达成员工购票的基本目的，架构使用简单，花费人力物力少，能够尽快被开发出来以供在晚会前让所有员工通过此系统完成购票。

* 1. **支付方式**

此系统暂定是在国内，所以包含了国内主流的支付方式，微信支付，支付宝支付和银行卡支付，支付方式开发需要对接支付的API，Demo系统暂不进行API的连接。

支付逻辑：

* 支付宝和微信，支付页面跳转到支付宝微信支付，支付成功返回消息到控制器，用户购买晚会票成功，随机分配座位并调用模型方程插入数据库。
* 银行卡支付，用户输入银行卡信息选择支付，调用银联支付API进行支付，支付成功返回消息到控制器，控制器随机分配座位并调用模型方程插入数据库。
* 支付失败，若用户支付失败，返回失败信息，控制器不调用插入数据库方程，直接返回原页面并显示“支付失败”等消息。

1. **Model**
   1. **TheaterSeats**

/app/Models/TheaterSeats.php

此模型主要用于建立剧场所有的座位号，根据等差数列计算出剧场座位的排数和每一排的座位数，建立数组，并插入到数据库中，此模型中所有方程只需要在程序初始化的时候运行一次即可。

* 1. **UserSeats**

/app/Models/UserSeats.php

此模型主要分为两项：用户选择座位和根据邮箱从数据库读取所有数据并列印出来。

* 用户购买晚会票，被随机分配座位，最大可以一次购买5张晚会票，并且不能够选择已被选择的座位，此处考虑了两种实现方式：
  + 第一种是根据用户选择数量随机从数据库中选择座位并建立数组，再使用Iteration方式将数组中每一个座位号与已选择座位表中的座位号进行对比，如果已存在，抛弃此值，重新选择，如果不存在插入到表中，
  + 第二种方式是使用Recursion每次随机选择一个座位，跟已选择座位表中座位对比，如果不存在，插入到表中，recursion次数-1，如果存在，以原来的次数开始recursion。
  + 因为已知用户最多一次只能够购买5张晚会票，并且用户数和座位数都是固定的，所以此处为了代码开发简便使用了Recursion。

1. **View**
   1. **主页（index）**

/resources/views/index.blade.php

Laravel框架有自己的视图结构，全部使用blade.php后缀，index便是用户访问此系统所看到的第一个页面。



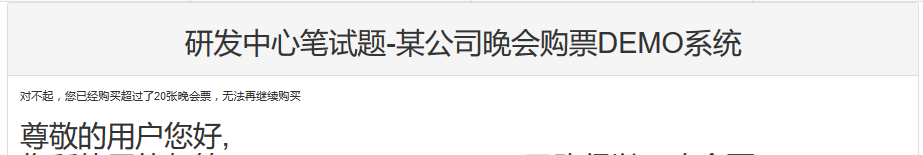
用户可再次输入姓名邮箱和选择购买票的数量，选择支付方式进入下一步，或者选择查询晚会票进入查询页面。此页面会向控制器使用POST提交四个属性，姓名，邮箱，票数和支付方式，控制器接收到这四个Request之后调用Model中的方程进行处理，若购买成功，用户则会跳转到查询页面。

* 1. **用户查询已买座位页面（show）**

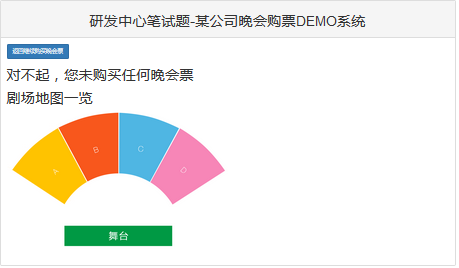
/resources/views/show.blade.php



若用户成功购买晚会票或者是此邮箱已经购买过晚会票，当转入到查询页面后，用户会看到以上信息，查询页面根据邮箱选择此邮箱下的所有座位，并列印出来展示在页面方便用户查询读取，同时页面也插入了简单的剧场地图以便用户对剧场有个直观的了解。若是用户还想要购买更多的晚会票，则可以直接点击按钮返回到购票页面，若此邮箱所购买的票已经超过20张，用户会看到如下信息：



如果用户所输入的晚会票并没有购买任何一张晚会票，而直接点击查询界面会看到如下信息，此时用户可以直接点击返回购买晚会票。



1. **Controller**
   1. **UserSeatsController**

/app/Http/Controllers/UserSeatsController

此项目目前的控制器只需要进行两件事情：

* 显示主页并接收主页上用户提交的信息判断是调用Model中的座位随机选择方程还是跳转到查询页面
* 查询页面根据用户提交的邮箱查询数据，并将所有座位放入到数组返回给视图页面。

1. **数据库结构表**