影院管理系统设计说明书

完成日期：2019/12/29

目录

[1. 编写目的： 2](#_Toc28612400)

[2. 背景 2](#_Toc28612401)

[3. 参考资料 3](#_Toc28612402)

[3.1 程序（子系统）系统的组织结构 3](#_Toc28612403)

[3.2 界面设计要求 4](#_Toc28612404)

[3.3 本软件系统中各程序(子系统)的设计说明 10](#_Toc28612405)

[3.3.1 Model、Database、Config设计说明 10](#_Toc28612406)

[3.3.2 DAO的设计说明 12](#_Toc28612407)

[3.3.3 Service的设计说明 13](#_Toc28612408)

[3.3.4 OrderService的设计说明 21](#_Toc28612409)

[3.3.5 Controller的设计说明 25](#_Toc28612410)

[3.3.6 DatabasePage的设计说明 27](#_Toc28612411)

[3.3.7 InsertMoviePage的设计说明 30](#_Toc28612412)

[3.3.8 InsertSchedulePage的设计说明 31](#_Toc28612413)

[3.3.9 StatisticPage的设计说明 32](#_Toc28612414)

[3.3.10 UserMainWindow的设计说明 33](#_Toc28612415)

[3.3.11 UserHomePage的设计说明 35](#_Toc28612416)

[3.3.12 MovieListPage的设计说明 35](#_Toc28612417)

[3.3.13 SchedulePage的设计说明 36](#_Toc28612418)

[3.3.14 SitePage的设计说明 37](#_Toc28612419)

[3.3.15 PaymentPage的设计说明 39](#_Toc28612420)

[3.3.16 ResultPage的设计说明 41](#_Toc28612421)

# 编写目的：

本系统分管理端和客户端两个子系统，管理端帮助影院管理员管理电影数据、档期数据、订单数据，客户端为顾客提供购票服务，

管理端：

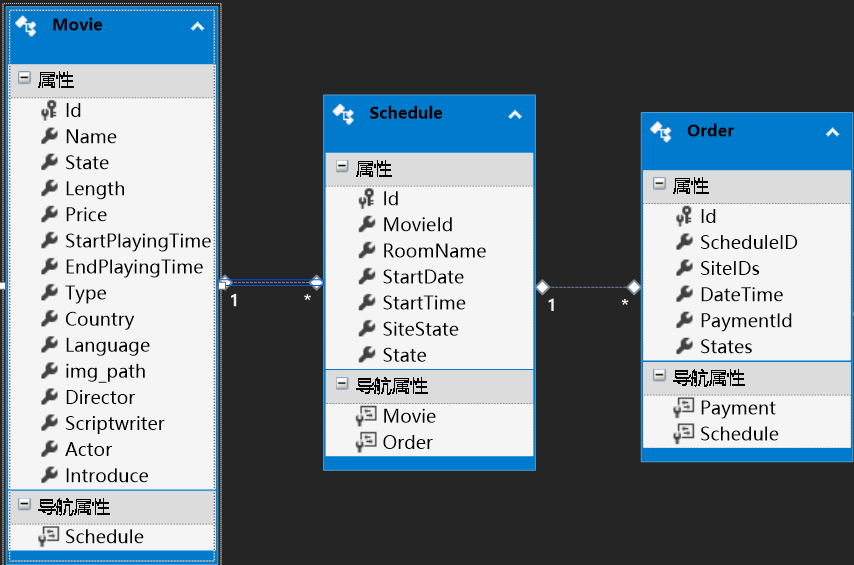
* 查看电影，档期，订单数据
* 导入电影
* 自动排制档期
* 销售统计

客户端：

* 查看正在上映的电影
* 查看电影对应的档期
* 选择座位
* 付款

数据库结构

3个实体类（项目中还有一个Payment，但没有使用），电影，档期，订单。一个电影对应多个档期，一个档期（一场电影）又对应多个订单，一个订单对应多个座位



# 背景

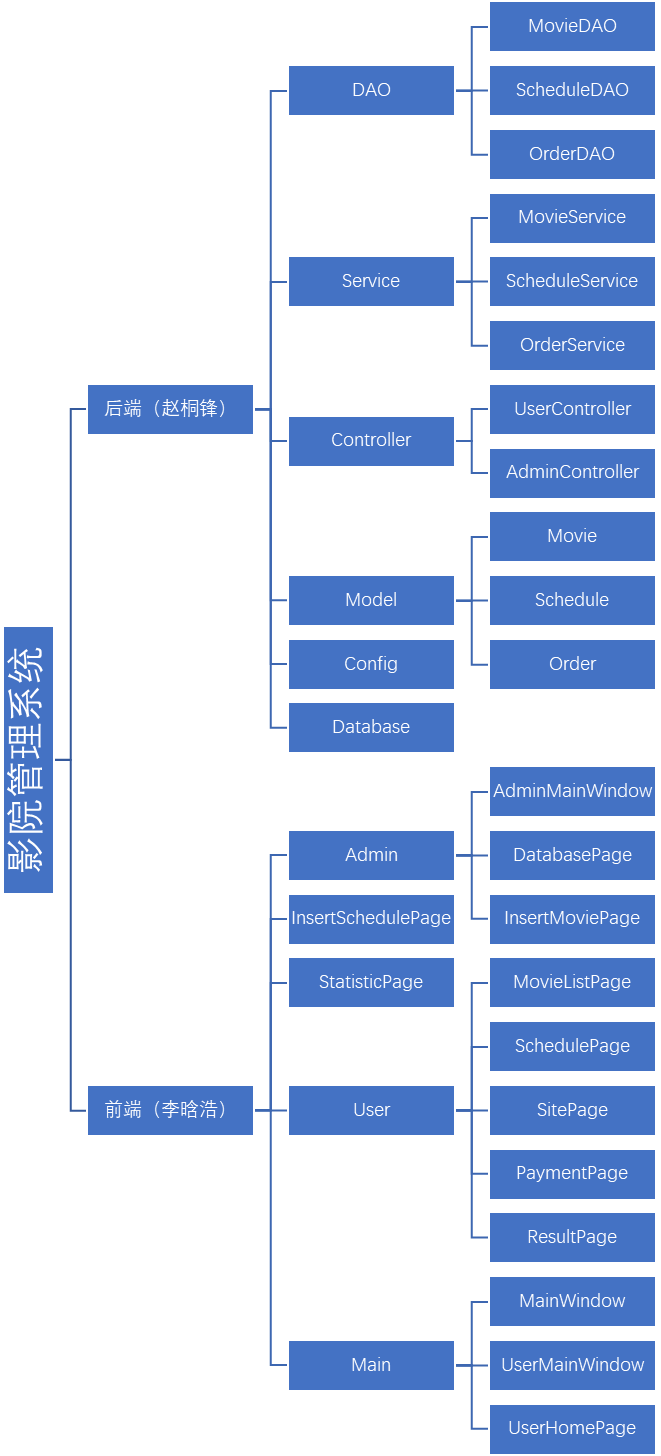
软件系统的名称： 智慧影院管理系统

本项目的任务提出者：

本项目的任务开发者：

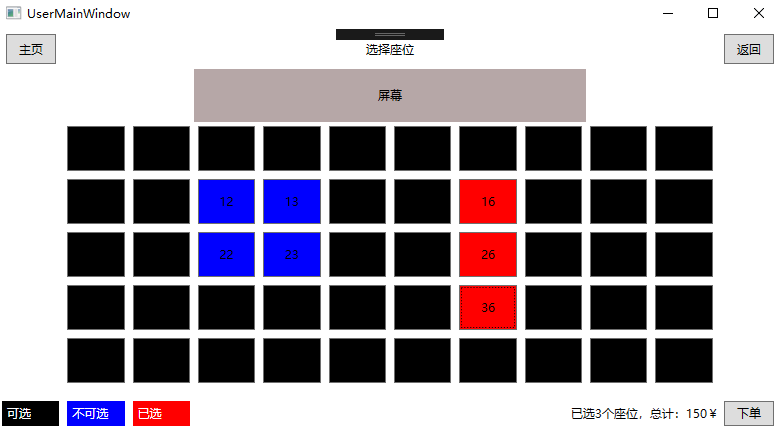
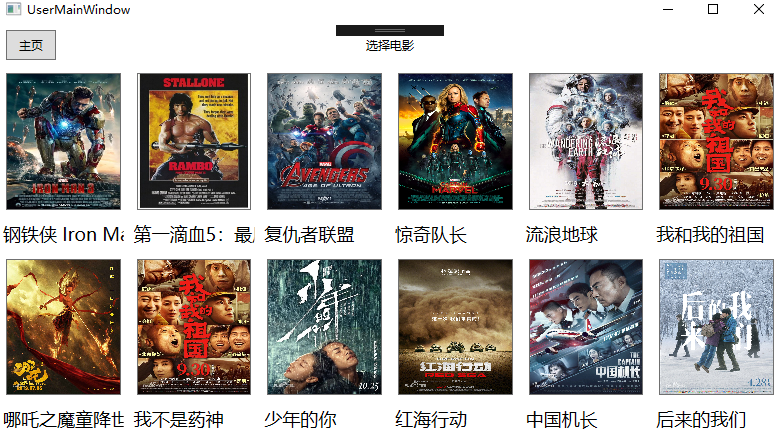
# 参考资料

## 程序（子系统）系统的组织结构



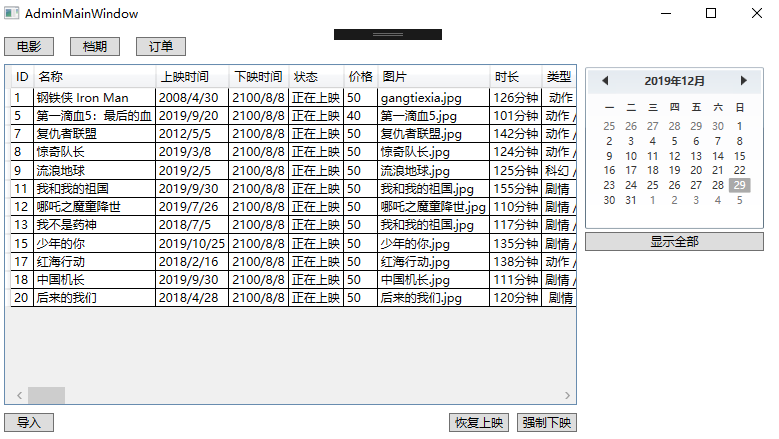
## 界面设计要求

客户端

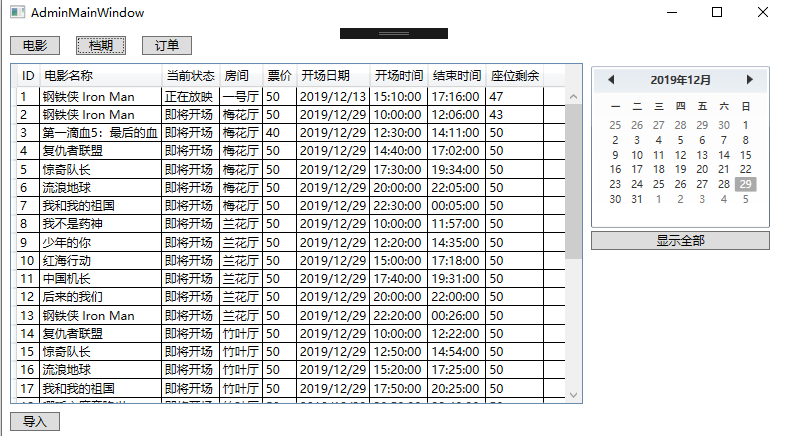


管理端

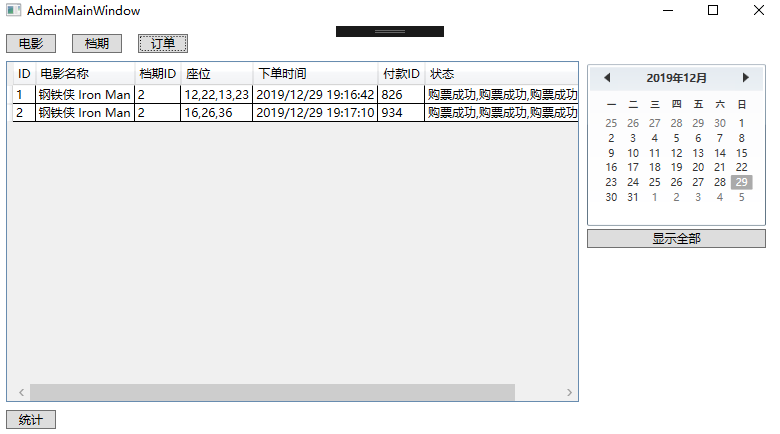
电影信息页面



档期信息页面



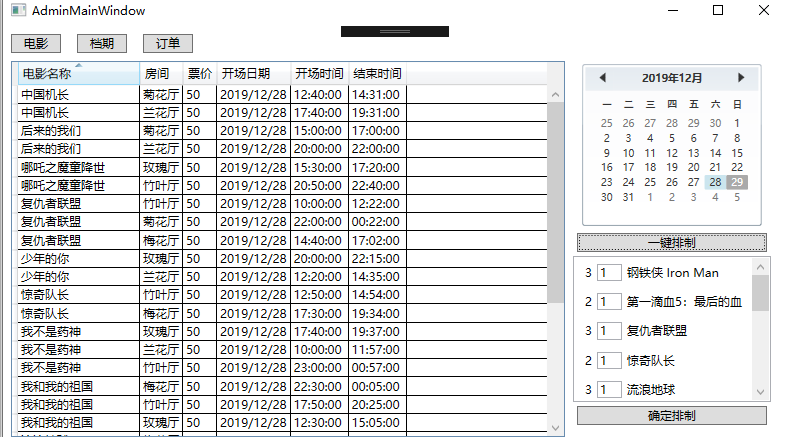
订单信息页面



导入和修改电影信息页面



排制档期页面



统计页面



## 本软件系统中各程序(子系统)的设计说明

整体简介：

本系统完全按照MVC的架构来设计并实现，前后端完全分离，后端部分，DAO用于和数据库直接交互，Service继承DAO封装所有业务逻辑，由Controller提供调用。

前端部分主要使用Frame和Page进行跳转，通过事件和委托调用父窗口函数。。

后端：

### Model、Database、Config设计说明

简介：

Model使用实体框架根据数据库自动生成。三者之间的关联关系和数据库完全一致。

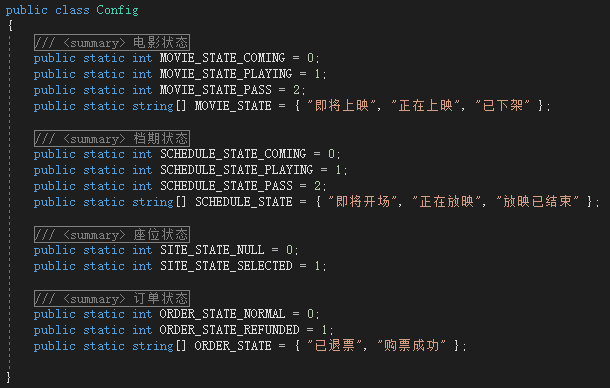
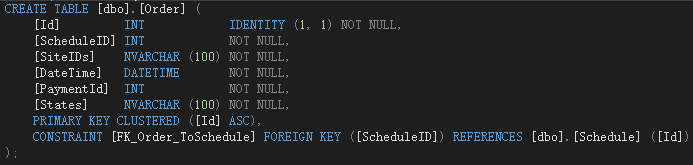
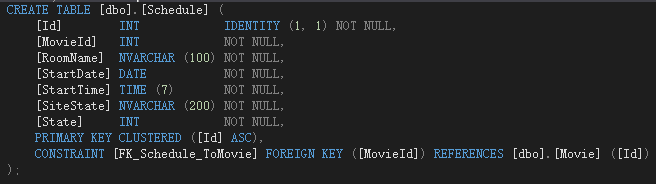
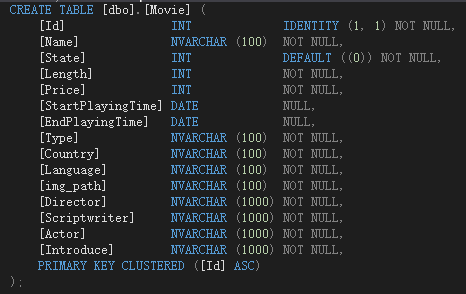
Model有三个，Movie（电影），Schedule（档期），Order（订单）。

Movie有很多基本属性，这里不做详细说明，要特别说明State状态字段，int类型，用于标记电影的上映状态，有三个枚举值，0，1，2分别对应的电影未上映，正在上映，已下架。

同样，档期也有三个属性，即将开场，正在放映，已结束，除此之外，档期还有一个SiteState属性，String类型，01字符串，用于记录座位情况，0表示未选，1表示已选。（比如50个空座位就有50个0）

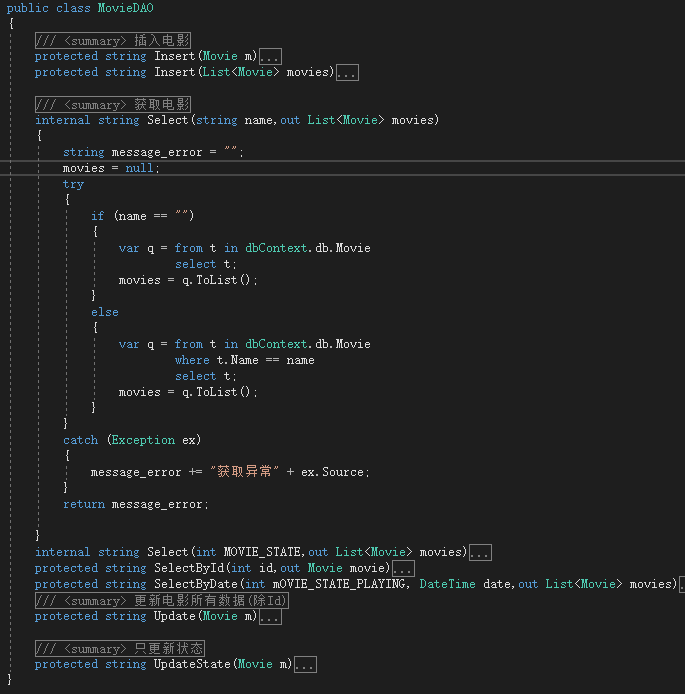
Order的State略有不同，是由逗号分隔的01字符串，1表示正常，0表示已退票

Config文件用于记录所有的枚举值，以便程序调用



### DAO的设计说明

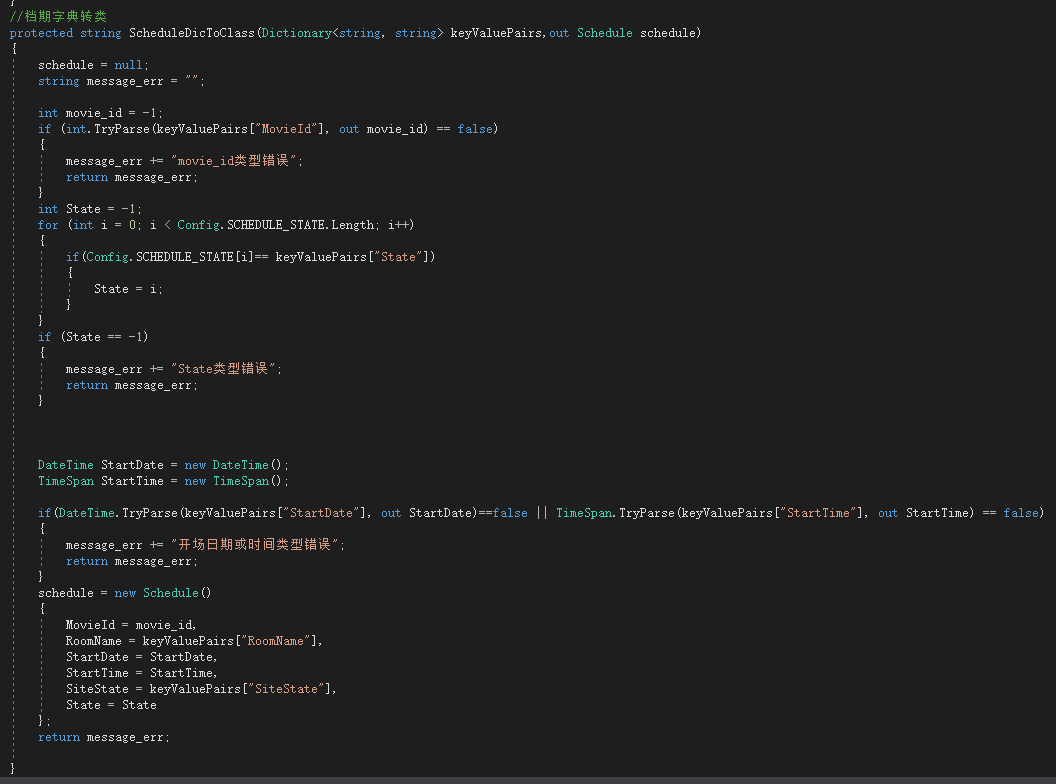
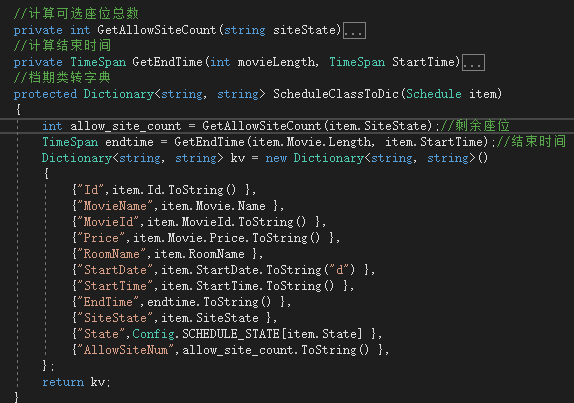
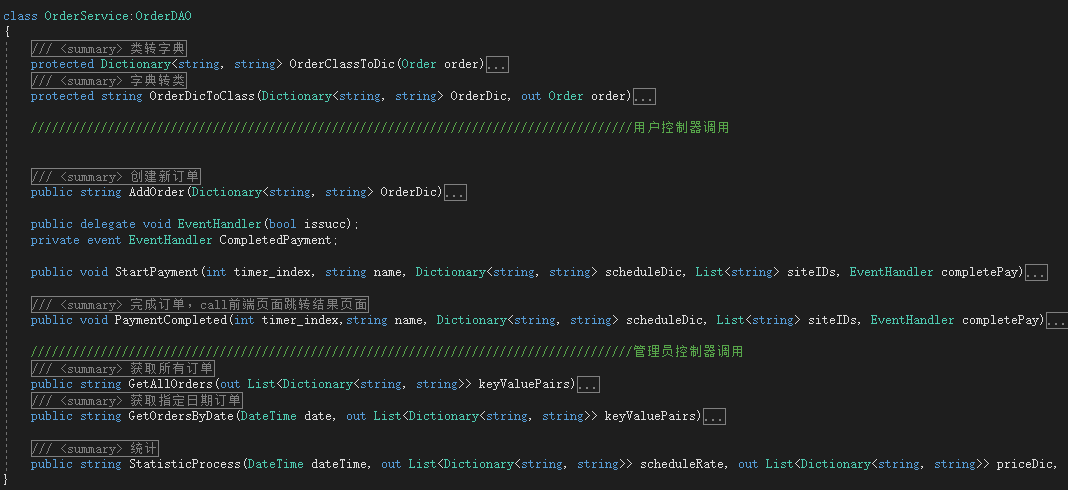
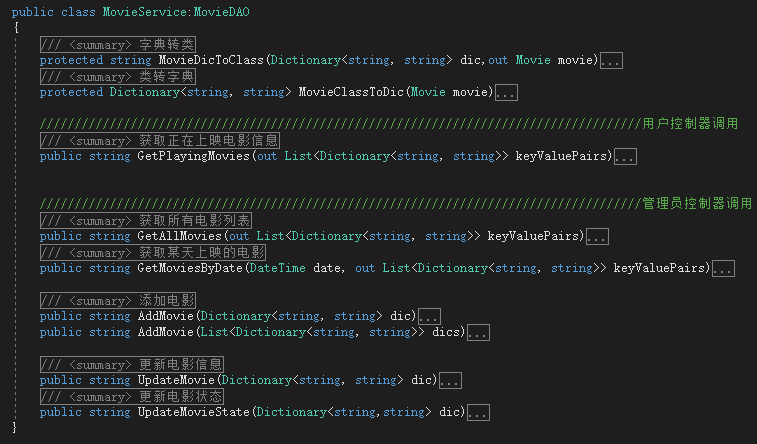
通过上下文使用LinkQ语句直接对数据库进行访问，三个DAO封装了对各自表进行增删查改的基本操作，使用了一些函数重载，用于不同的查询，使用protected属性限定方法只允许子类访问，不能直接访问DAO



### Service的设计说明

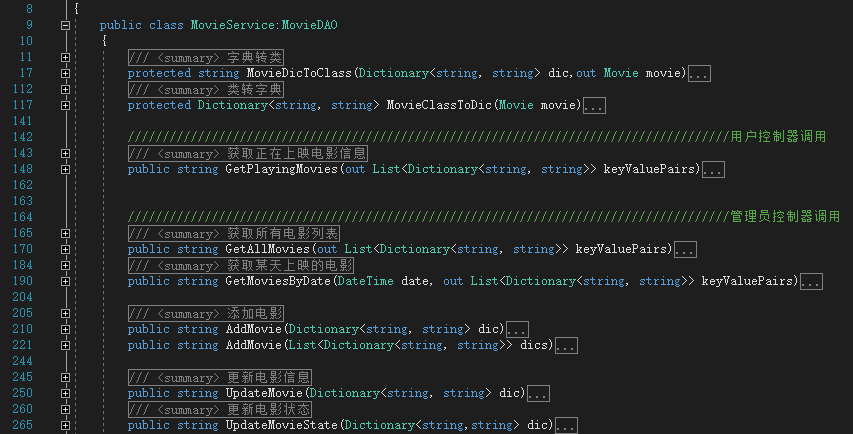
简介：

每一个Service都有一对序列化和反序列化Model的函数，在这个过程中要对数据库中的一些值进行处理，用于收发前端的数据；所有增删查改的基本思路都是，前端发送字典列表到后端，后端进行检测并反序列化，调用DAO函数进行基本的增删改查，或者反过来，从DAO调用函数获取Model列表，序列化之后，返回前端。

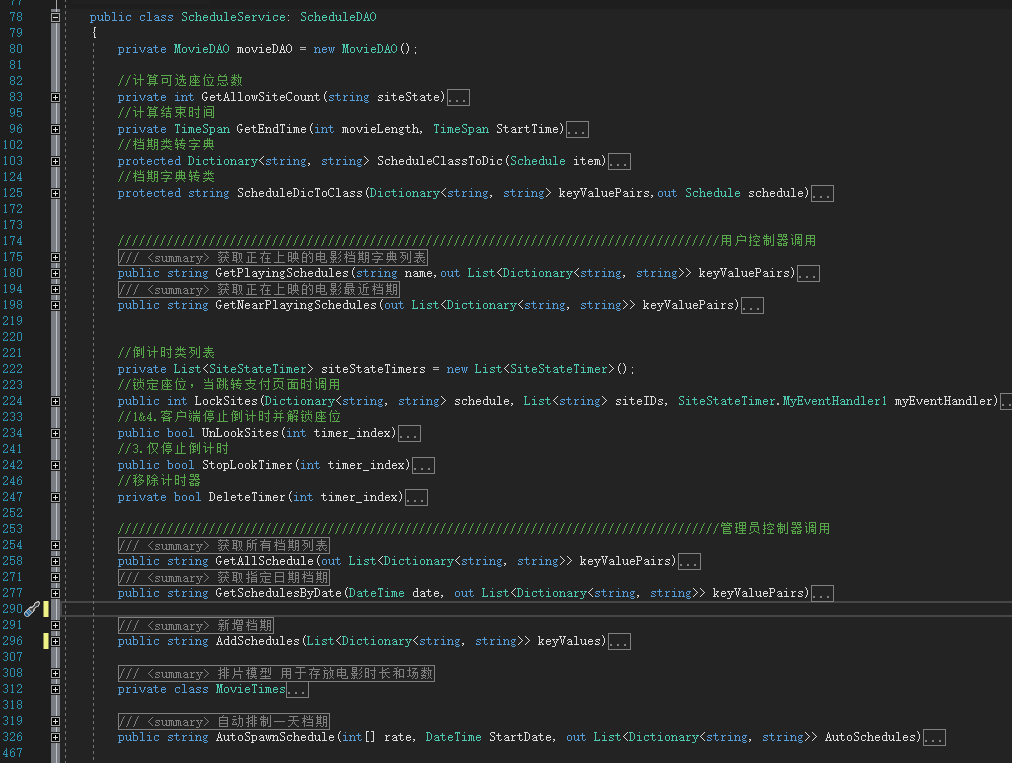


MovieService

除了增删查改，基本上没有其它特别的操作



ScheduleService

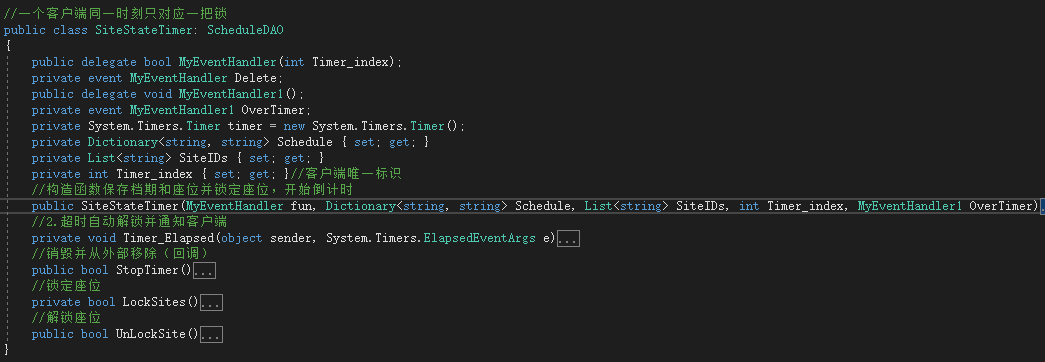


锁定订单

输入 档期字典，选择的座位，超时事件函数指针

输出 线程id（列表下标）

当前端跳转支付页面之后会调用后端的LockSites函数，这个函数会实例化一个自定义Timer类（主要记录的档期和选择的座位，构造函数中启动计时器，保存前端的超时事件委托，停止计时器函数，解锁座位函数），超时会自动解锁座位并通知前端。因为当计时器开始运行之后，前端也有可能触发停止计时器事件，所以将这个计时器存到一个列表中，返回下标给前端，当有多个前端的时候就需要一个id来关闭各自的计时器，这个UUID便是返回的下标。



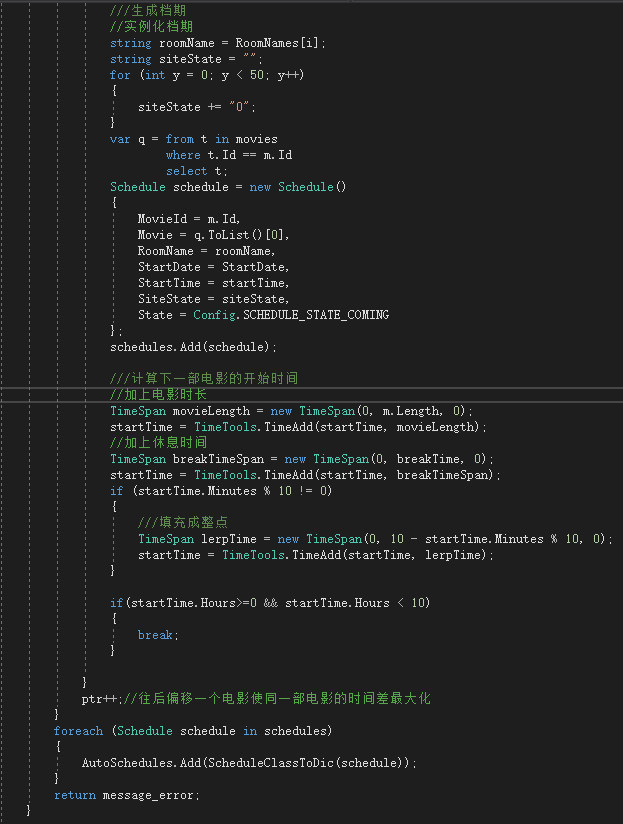
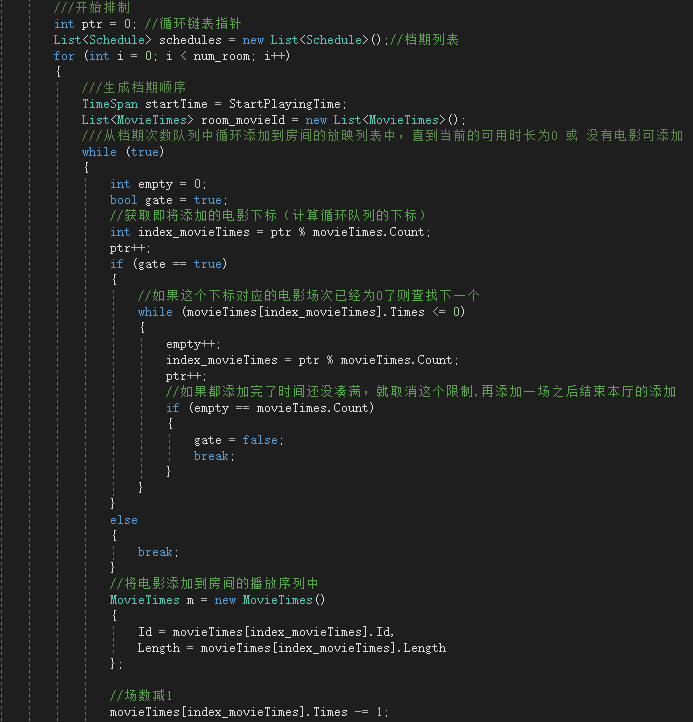
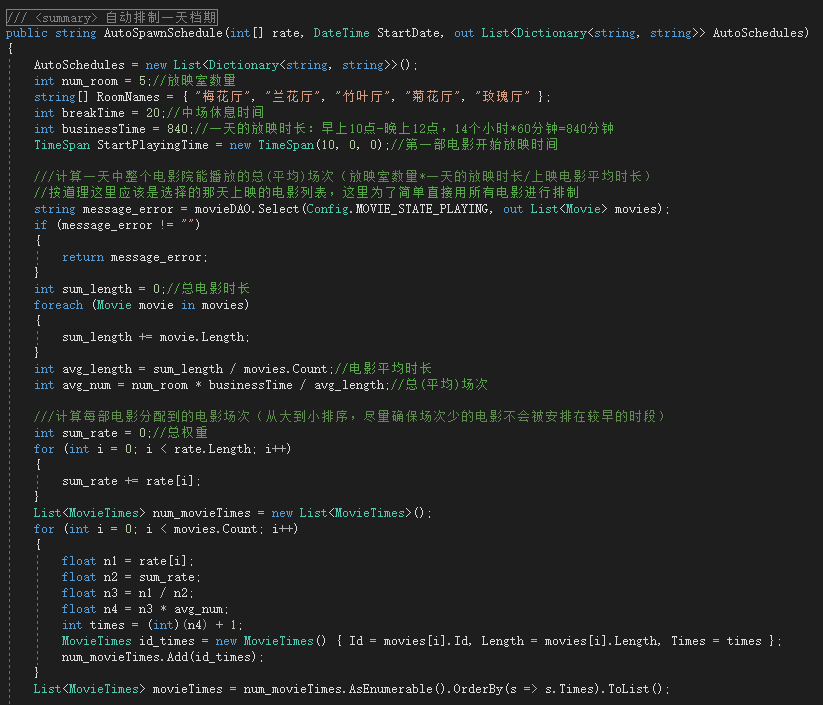
自动排制一天档期算法思路

需求：

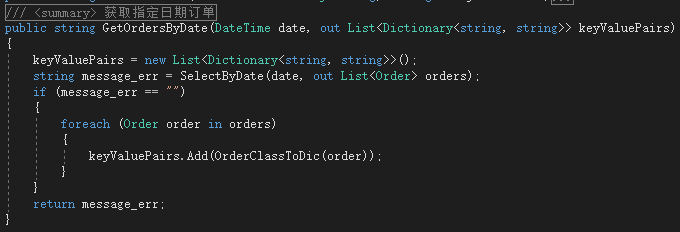
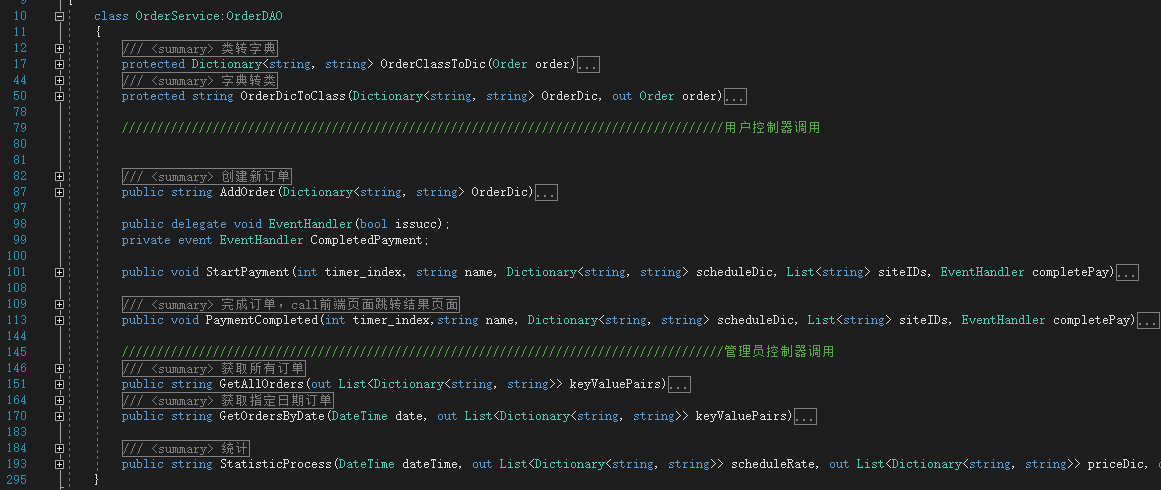
* 输入：电影热度（权重），日期
* 输出：一天中整个影院若干个放映室的档期

要求：

* 档期数量最小为2
* 档期越少间隔越长，
* 档期越多越均匀
* 档期整时开场
* 已知：
  1. 电影播放权重
  2. 各个电影时长
  3. 放映室数量5个
  4. 打扫卫生20分钟
  5. 开始时间段10点到24点
* 计算：
  1. 一天中整个电影院能播放的**总(平均)场次**
  2. 每部电影分配到的**电影场次列表**（从大到小，记录Id，时长，场数）
* New 档期列表 //保存最终排制的档期
* Int ptr //循环链表指针
* Foreach room
  1. 开场时间= 10:00
  2. While(开场时间 < 24:00 && **总(平均)场次**！= 0)
     + 电影Id = **电影场次列表**[ptr%电影数量].Id（剩余场次>0）
     + **电影场次列表**[ptr%电影数量].场数-1
     + **总(平均)场次--**
     + New 档期(电影，房间名称，开场时间，座位状态)
     + 添加档期到档期列表
     + 开场时间 =开场时间+电影时长+打扫卫生时间+顺延整时
     + ptr++
  3. ptr++ //确保同一电影不在同一时间开场
* 一次性将档期列表插入数据库

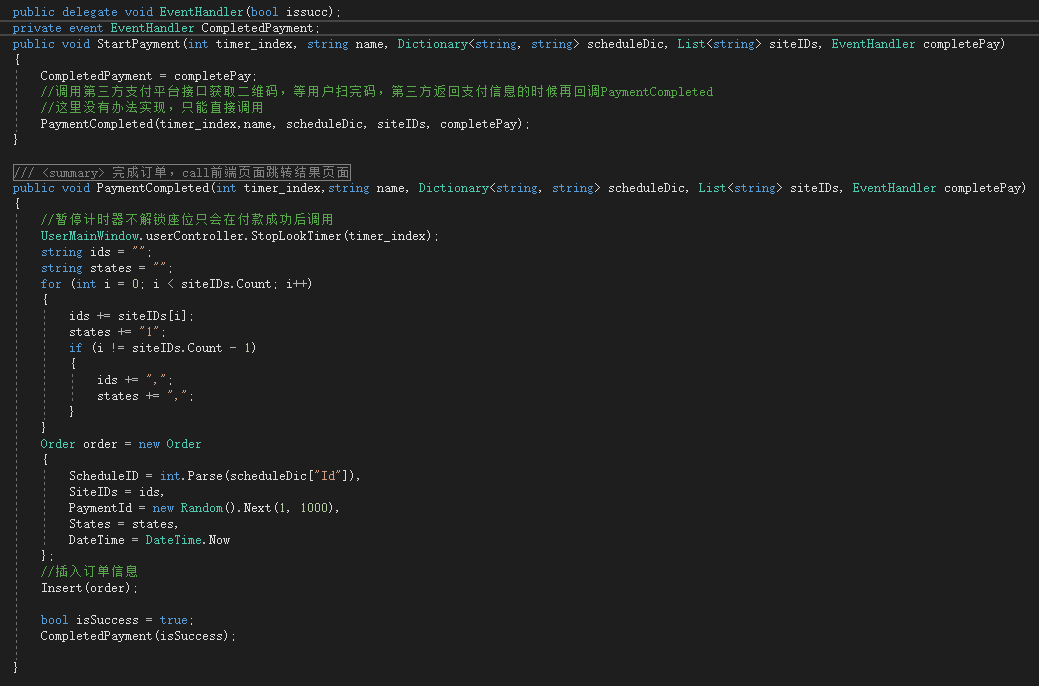


### OrderService的设计说明



支付模块

当用户在前端选完电影、档期、位置之后，在Payment页面调用一个StartPayment函数，将数据传递到后端，理论上说应该是想第三方支付平台请求一个二维码，然后返回给前端，前端进行显示，用户扫码付款，付完款之后，第三方支付平台再返回一个支付结果，再调用OrderService里面的OnPaymentCompleted函数，但是我们没办法实现，所以就直接调用了支付完成函数，支付完成函数将会实例化新的订单并调用DAO插入数据库，并且会调用StopLockTimer函数取消解锁订单的倒计时，

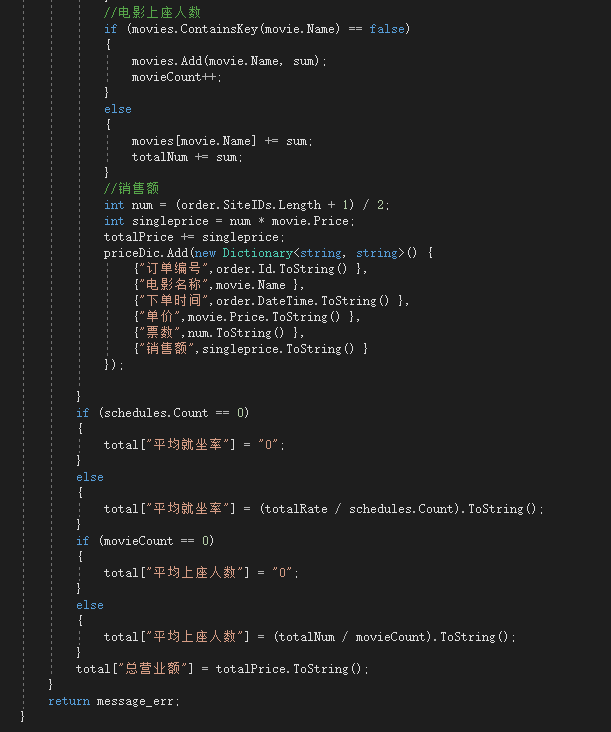


统计模块

输入 日期

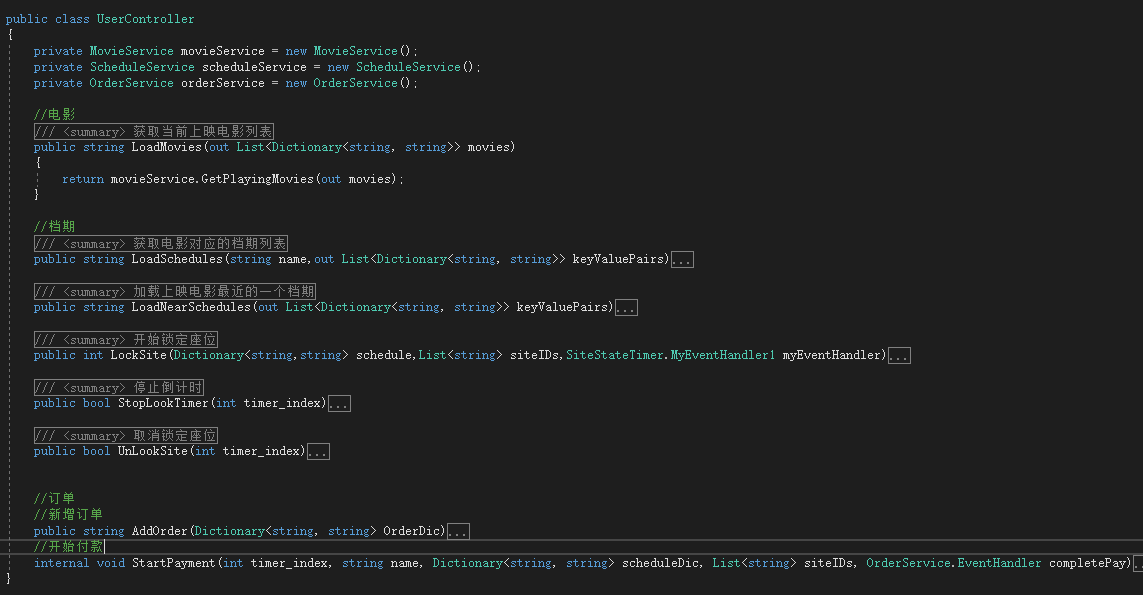
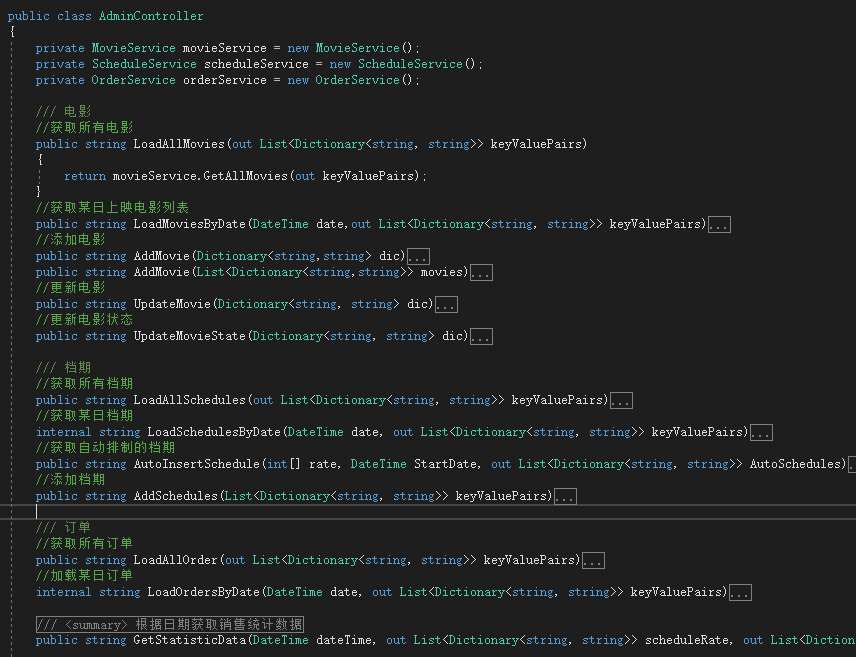
输出 统计信息

功能：统计一天中各个档期的就座率、电影的观影人数总数，以及销售金额



### Controller的设计说明

控制器内部没有任何业务逻辑，只是传递Service中的函数，当然对于管理端和客户端来说，不同的用户对应，我们呈现的数据可见度不同，所以分为两个控制器，分别封装各自要调用的函数。这两个类是主窗口的静态成员变量，方便子页面调用。



前端

页面设计说明：

简介：两个子系统相对独立，数据库共享

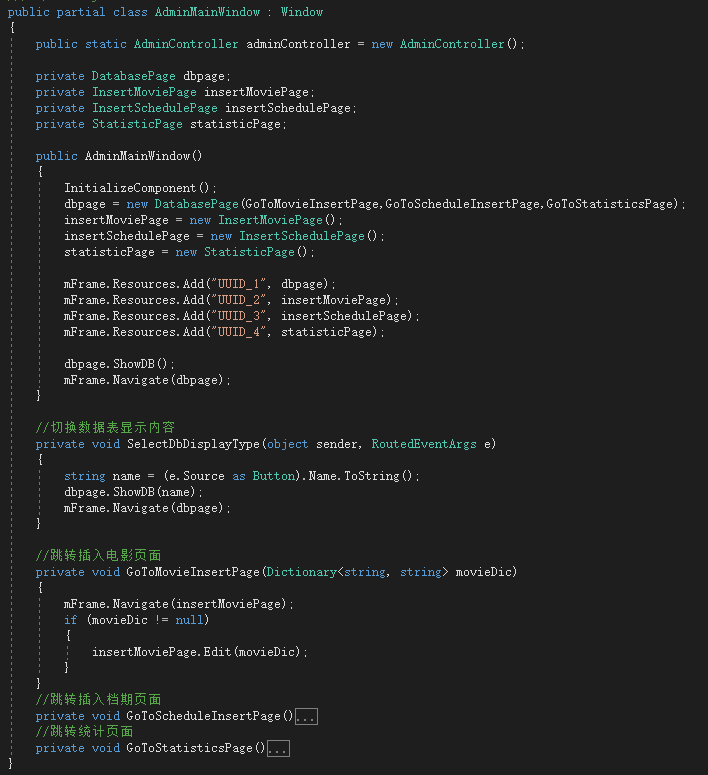
主页面

页面的跳转，两个子系统都只有一个主窗口，里面放一个Frame，其它页面都是放在Page里面的。同时主窗口中有通往各个page的函数，在实例化每个page的时候进行绑定，如果Page想要调用父窗口的函数，则通过**委托和事件**来执行的。

**由于所有业务逻辑都在后端，前端只负责一些显示和输入的工作**

**管理端**

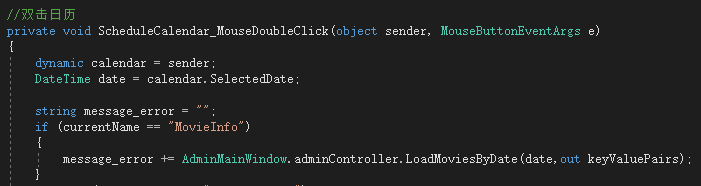
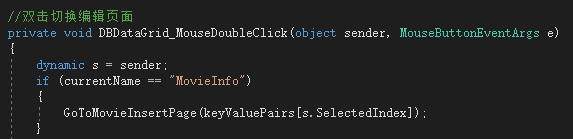
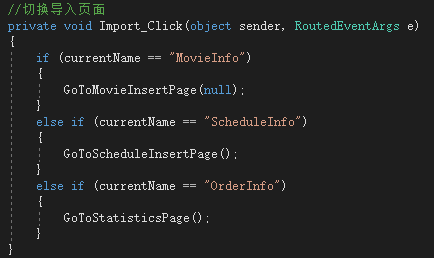
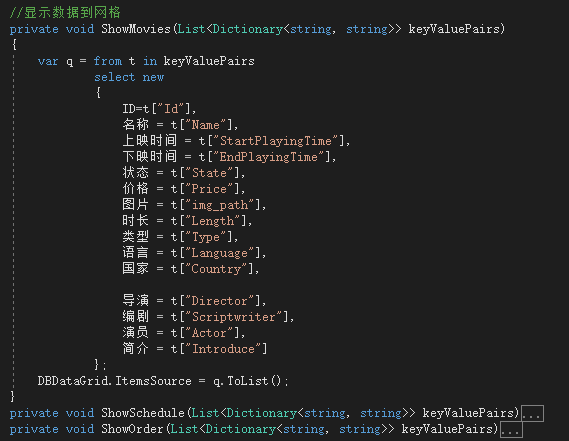
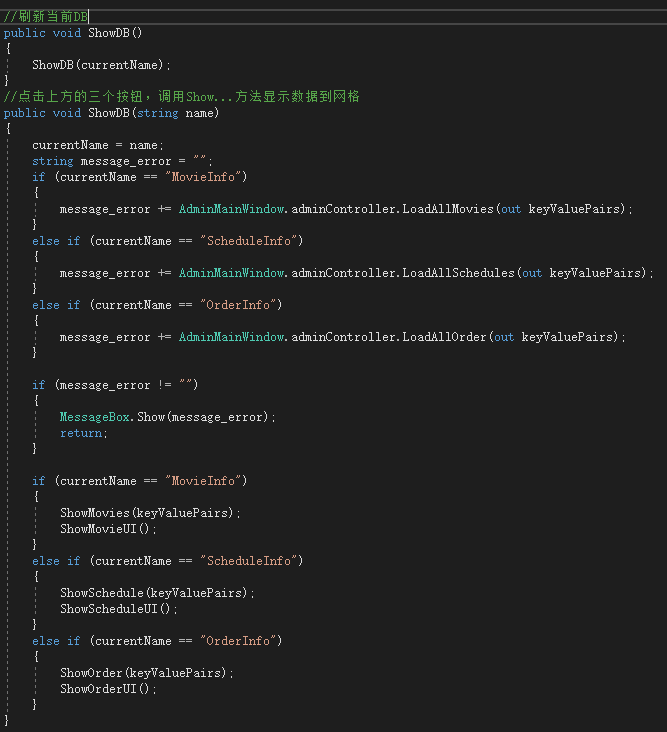
主窗口



### DatabasePage的设计说明

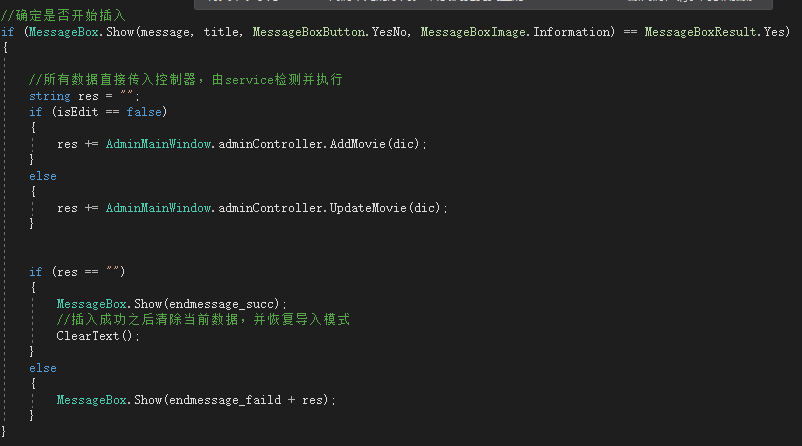
通过AdminController从后端Service中获取数据，电影、档期、订单共用一个dataGrid，通过点击的按钮来确定类型，如果页面显示的是电影，则可以双击跳转编辑页面，强制下架和恢复上映（因为电影的上下影都是通过时间控制的，这里只是以防万一），点击日历可以选择当天的上映的电影、档期、订单。

左下方的按钮是导入或者统计，点击之后会根据当前页面进行跳转



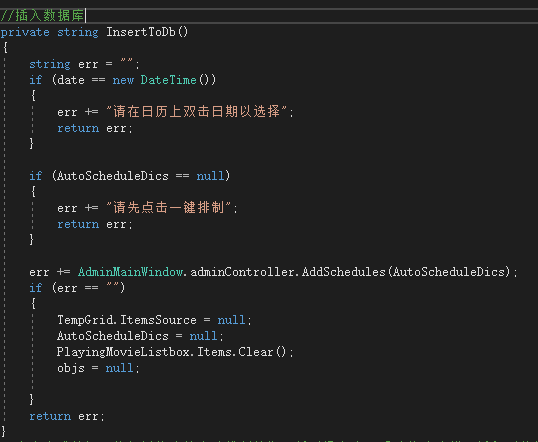
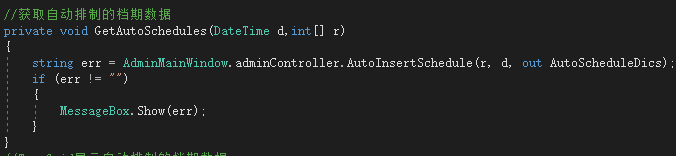
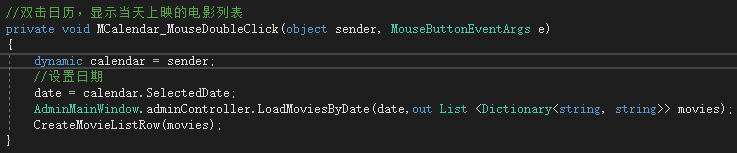
### InsertMoviePage的设计说明

给出电影的属性和输入框，可以手动输入也可以从文件导入。调用后端插入或更新函数



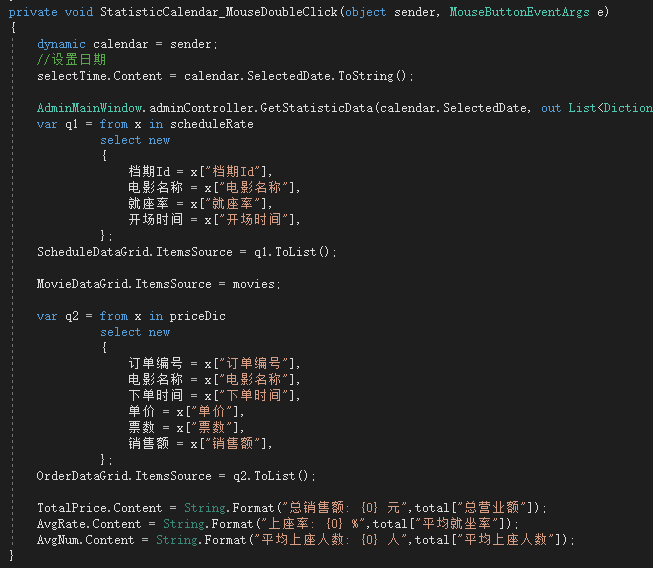
### InsertSchedulePage的设计说明

首先点击日历选择日期，页面中会显示当天上映的电影名称和输入框，管理员输入权重，点击一键排制，左侧DataGrid会显示排制结果，点击确定导入，档期被导入数据库



### StatisticPage的设计说明

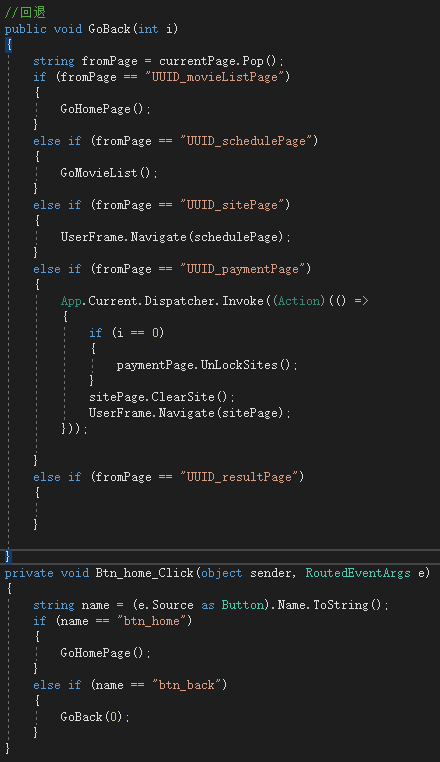
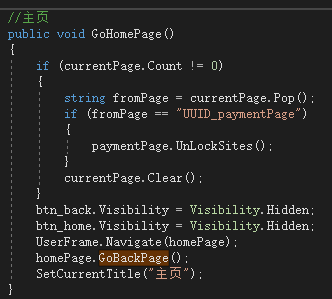
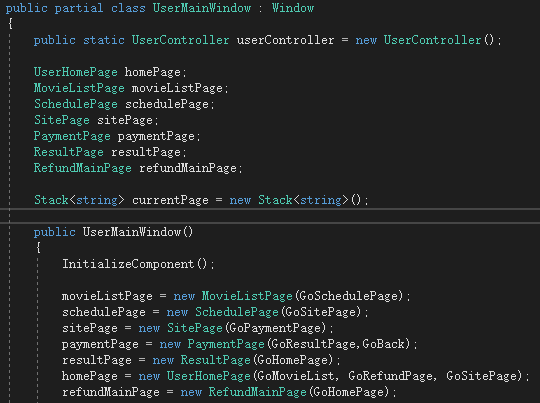
由于统计函数在OrderService中已经编写好了的，这里直接调用获取到数据，直接显示就可以了，左边页面是将三个DataGrid放到Listbox里面



**客户端**

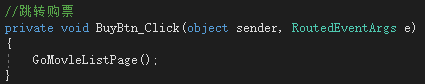
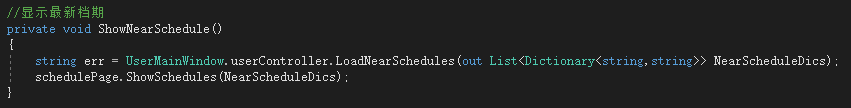
### UserMainWindow的设计说明

页面基本框架，左上方“主页”按钮，右上方“返回”按钮，下面Frame。使用的自定义的“返回”按钮，没有直接调用导航的返回函数的原因是，在某些页面进行返回的时候，还是需要调用一些事件，解决方法是将页面放入栈中，顺序跳转的时候进栈，返回时出栈。



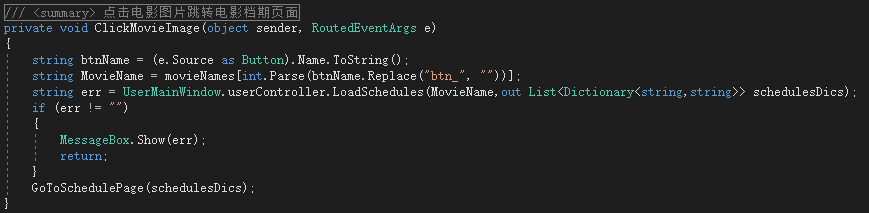
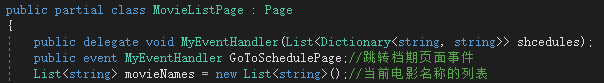
### UserHomePage的设计说明

从后端获取到NearlyScheduyle并显示，购票按钮，跳转MovieListPage



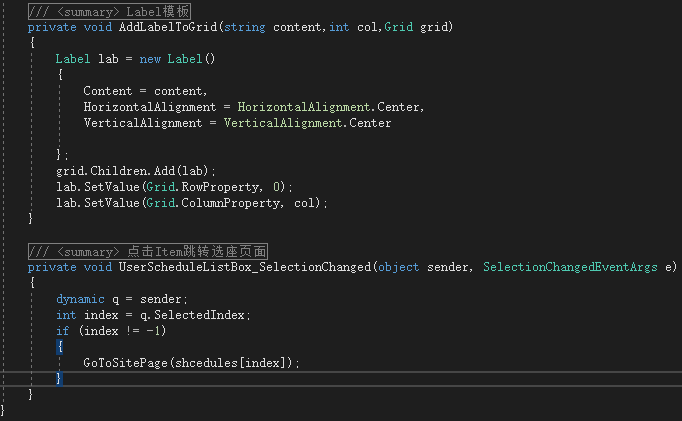
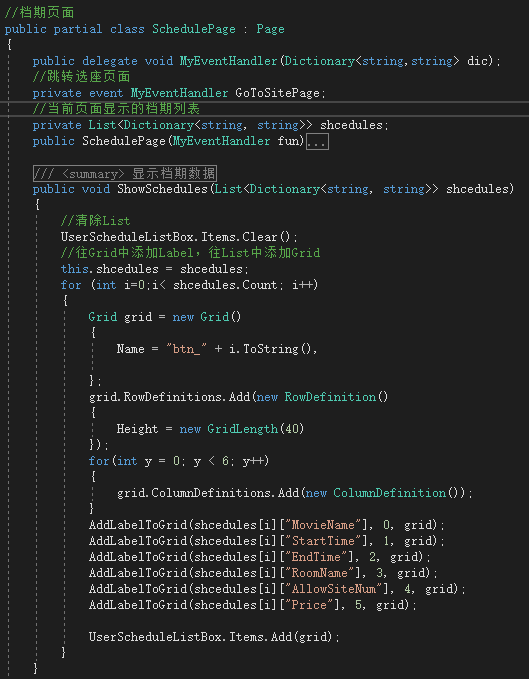
### MovieListPage的设计说明

从后端拿到正在上映的电影，动态设置Grid进行显示，绑定点击按钮，跳转SchedulePage，从后端获取电影对应的档期，传给SchedulePage



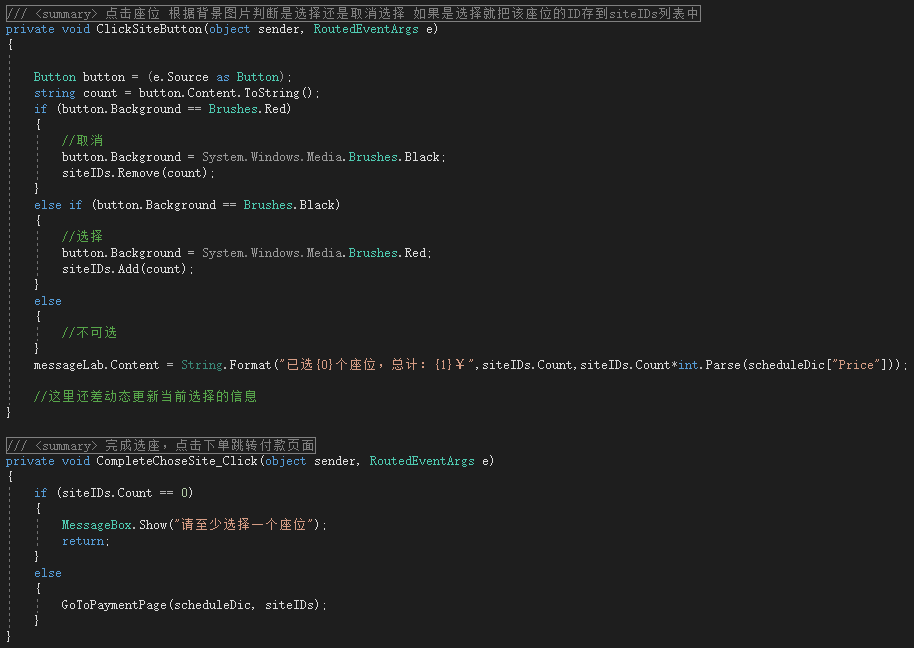
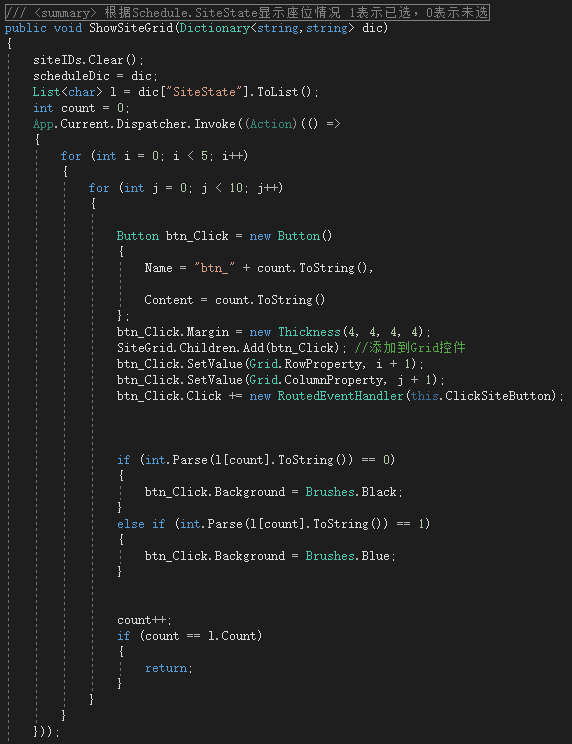
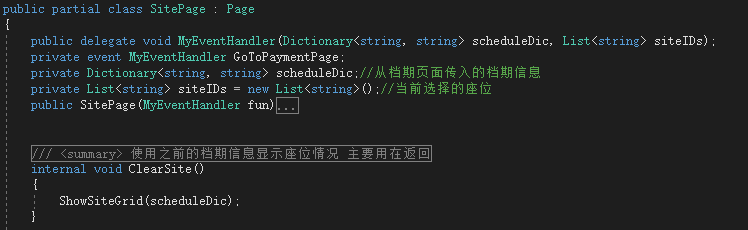
### SchedulePage的设计说明

显示档期数据，点击跳转SitePage，传递Schedule字典



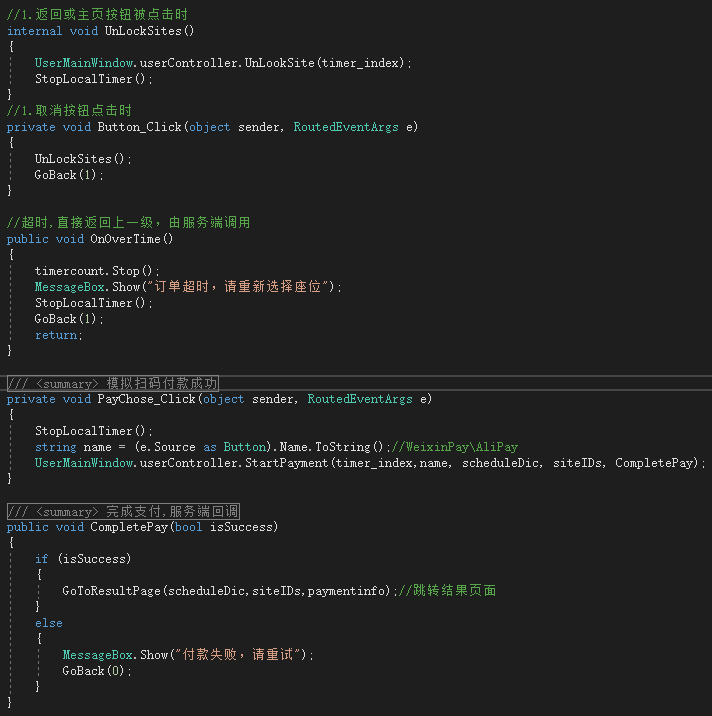
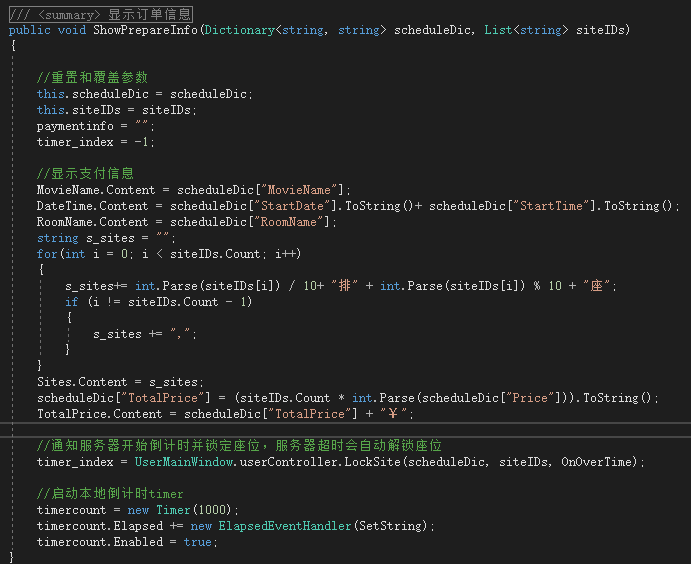
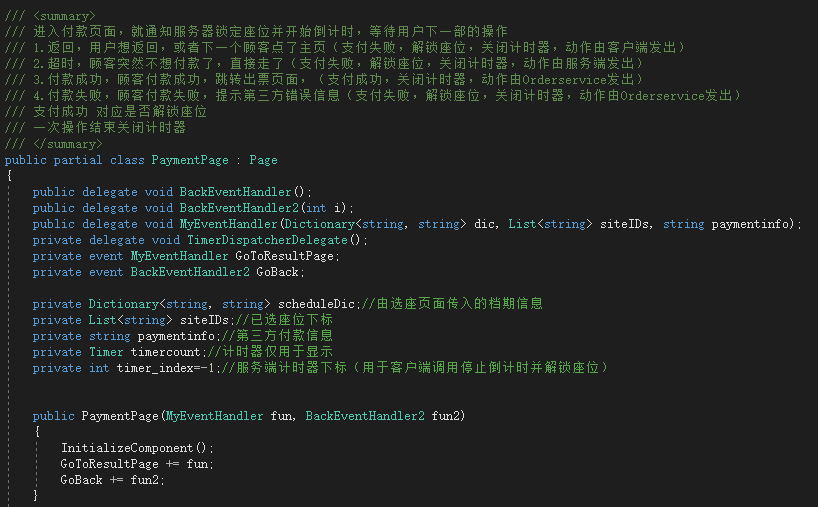
### SitePage的设计说明

根据Schedule字典中的SiteState，用网格渲染出座位状态，1表是已选，0表示可选，每次点击网格的时候，判断是状态进行设置颜色，将选择的座位添加到列表，用于确认下单时传递给PaymentPage



### PaymentPage的设计说明

PaymentPage调用后端的LockSite函数锁定座位，并等待付款，付款成功后，后端会调用OnPaymentCompleted函数显示结果，如果成功跳转ResultPage，如果失败则放回选座页面



### ResultPage的设计说明

仅显示购票结果