

酒店预订系统

总结报告



南京大学软院学院

Leftovers

2017-1-1

## 项目概况

酒店预订系统就是为满足xx创业公司创业发展需求开发的，它包括一个数据集中服务器和多个客户端。数据集中服务器将所有的数据储存起来进行维护。用户通过客户端完成日常任务，客户端与数据集中服务器采取实时通信的方式完成数据交换。

## 团队介绍

|  |  |
| --- | --- |
| 郭浩滨 | 151250048 |
| 林志和 | 151250094 |
| 吴游杰 | 151250161 |
| 吴志成 | 151250163 |

## 项目时间表

### 9.17-10.2 需求开发阶段

当时第一次面对文档都有点蒙蔽，没有提前做了了解的准备工作，第一次设计出来的需求文档有26个用例，方了，后来写了改，改了写，终于控制在20个，自己看着也比较合理，然后就是什么需求规格说明文档、测试套件、概念总图，现在还是觉得写出一份充分符合需求又简洁严谨的需求文档真的不容易……

### 10.3-10.17 体系结构设计阶段

这个时期疯狂得了解各种Vo、Po、Bl、Ds概念，画各种开发包图、设计方案，感觉最难搞、变更性最大的还是分模块和各模块之间的关系问题，比如user、personnel要不要分在一起，member、credit要不要独立开来。那时候这方面的东西都还比较懵懂，然后看到前辈的文档，吓懵了，十几个模块，然后看了自己的五个模块，心情复杂，最后当然还是采取了自己的方案。虽然最后五个模块其中一个模块还是做了改动，最后再解决循环依赖啊什么的问题。至于逻辑层和数据层的接口规范，由于我们一开始我们分工就比较明确，所以就主要还是我们两个搞逻辑和数据的来搞，当然，最后与代码一比对，差距还是有的，但总的设计思想还在哈哈哈哈。

### 10.18-10.24 人机交互设计阶段

上回说到，逻辑层和数据层的接口由我和大佬来搞，当然，这个文档的锅就甩到搞前端的 背上了。对于人机交互原型图，只能说我们too young too simple，以为只是把组件拖好就搞定了（借鉴前辈的文档发现他们也是这样= =！）这就导致我们很多细节没有考虑，比如配色与布局等等，导致构造阶段的时候还不得不边写代码边设计，效率低下。人机交互设计没有做好也导致了功能覆盖的缺失：有部分功能直到写界面才发现没有接口，导致重复加接口的问题。还有一个问题就是在构造阶段想在界面搞多一些花样的时候，特别容易把代码写乱。。。如果能重来，一定要把原型图做好。

### 10.25-11.14 详细设计阶段

这次详细设计阶段获得了很理想的分数，有点没想到，是不是要得瑟一下，文档写到后来，发现还是用例文档和体系结构文档的确定用例、确定模块的阶段最难熬，可能要设计出一种方案和作品的过程都像是浴火重生。（趁机夸小组一波）类的设计虽然参考的是学长的设计，但具体应用到我们自己的文档和设计，还是自己思考的。有自己的东西和细节在里面。写完详细设计文档，有点兴奋，又有点担心，因为终于来到重头戏阶段啦！

### 11.15-12.19 构造、测试阶段

万事开头难，刚开始的时候我们并不是很清楚一个项目要从何做起，比较不安，两三天后还是硬着头皮上了，然后就是一天一天的学习过程，还好JDBC和数据库数据有一点点的基础，前端程序员也有些web前端经验，果断fxml+css，搞成web风格，搞了一两周，项目转成maven，jenkins，结合junit，在开发过程中，自己一步步实现自己构想的功能和自认为创新、不错的实现方式真的很有成就感，界面的一个小动画、小效果都会很激动。比较遗憾的应该是构造阶段要结束的时候打包遇到的问题，导致最终客户端成品交不了。但是，但是，我们做出来的东西真的很棒。

## 项目成果展示

详见PPT

## 项目亮点

我们前端界面采用了web风格，只能说CSS真的是好东西，总体视觉效果及体验都还是不错的，可以吹。逻辑和数据层方面，我们的逻辑很稳，写完一个方法就会先自己进行单元测试，再通过界面端的操作测试。数据层用了许多工厂类，把耦合控制在一个较为正常的水平。

### UI的亮点：

1. 框架结构化：

保证流程的完整、清晰

1. 封装数据检查：

为不同的数据设置枚举类型，集中进行检查，方便管理。

1. 采用fxml：

将ui的数据和操作分离，方便调节界面的参数。每一个fxml对应一个controller，让ui代码与逻辑控制代码分离，可维护性高。

1. 采用统一的命名风格:

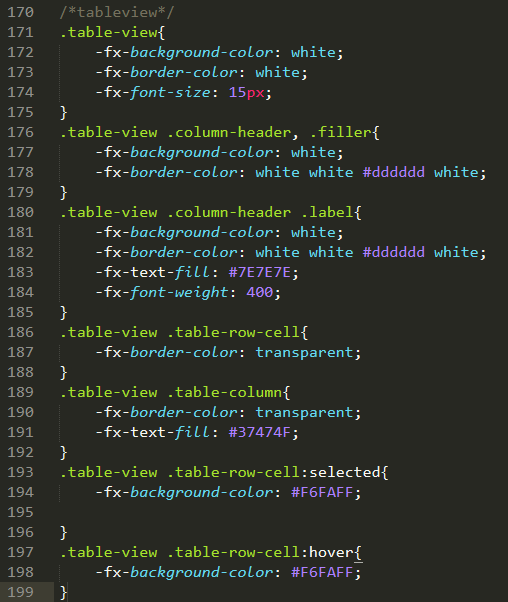
后缀名 Pane /Controller（控制器）/主界面（ComScene），建立精神模型，便于团队合作开发

1. 建立util包，使用自定义组件：

将自定义标签，按钮，提示框等等抽象出来封装，维护性高，减少代码量。

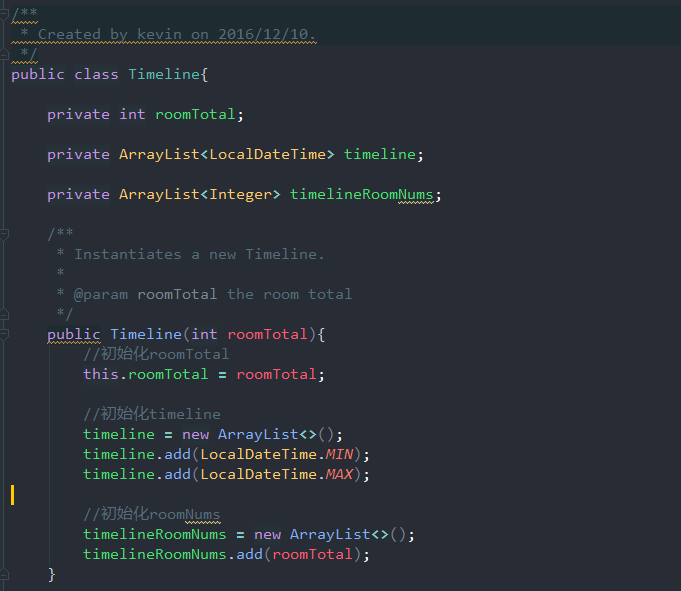
1. 采用CSS：

用CSS修改界面组件样式，省去P图的麻烦，而且可维护性极高，谁用谁知道！



### Bl的亮点：

1. 对网络异常进行集中处理，使得网络异常能够较为简单的处理完成
2. 委托式编程
3. 工厂模式+反射处理促销策略设计
4. 自己设计的Timeline类



### Data的亮点：

1. 提取父类，集中管理，父类以及helper承担起数据库大部分功能，使得子类的代码较为简单，甚至于仅涉及到调用构造器和赋值，不存在任何逻辑，代码可修改性大大得到提高，另一方面，将子类会用到的成员变量全部声明在父类，虽有一次声明，到处使用的嫌疑，毕竟还是方便和已修改（前提是父类足够稳定，本次的数据层符合这个特点，全过程中父类几乎无修改）



1. 数据集中，所有的数据库数据全都放在一个map中集中处理，便于修改



1. 数据库语句统一生成：利用可变参数列表，实现了只需要推入参数，就可以自动常用生成常用数据库语句，这样，在执行数据库语句时非常方便，可以统一使用，仅需要像上图一样把表给加入map中就可以了



1. 把数据库提取的数据get为Object，再通过List和PO的转换工厂，完成控制耦合，减少重复代码



## 经验

1. 使用google，stackoverflow解决了很多技术上的问题，如javafx，数据库操作，数据处理等

2. 数据库操作等代码可以多写泛型的、较为抽象的方法，可以提高代码质量

3. ui设计提前做好，图标等素材要统一并备份。图标可在阿里icon上找，配色可在flatuicolors等网站上借鉴

4. 不必畏惧bug，遇到bug不应该捶胸顿足，多利用断点查找错误，如果是技术上的难题直接StackOverflow（ps：很多bug都是因为简单地复制了一两行代码）

5. 做好方法的防御，因为方法会在各种意想不到的情况下被调用

7. 时间安排和团队分工、奖惩机制得先确定好

8. 可以参考一些优秀的框架

## 教训

1. 做好风险评估，提前规避未来的风险，这都是打包血淋林的教训呀

2. 写文档的时候要考虑的远一点，体现出真正的设计，不要等到失去了才懂得珍惜（等到写代码的时候才发现文档参考作用有限）

3. 开会要运用一些方法，把效率提高一点，不要钻在一个点上出不来，或者讨论着讨论着就飞到别的地方去了

## 团队分工

郭浩滨：队长，负责数据层、数据库及持久化对象

林志和：负责逻辑层代码编写

吴志成：负责界面设计及UI层代码编写

吴游杰：负责界面设计及UI层代码编写

## 展望

不管怎样，这个学期下来，已经很满意啦！已经很棒啦！以后我们会更棒，挑战会更多，路会更坎坷，不过，管他的。希望接下来做事情、谈事情思路都能更清晰，追求能够更高。嗯，半年的结果在这，之后会更精彩。Leftovers，我们是被抛弃的烟尘，还是最终留下来的强者，未来会给出答案。

Leftovers小组

2017年1月1日