武汉轻工大学

UML 课程设计

题　　目：ER-OA 系统

专　　业：软件工程

班　　级：软件工程 1704

学　　号：1905059014

姓　　名：毛俊峰

2020 年 1 月 3 日

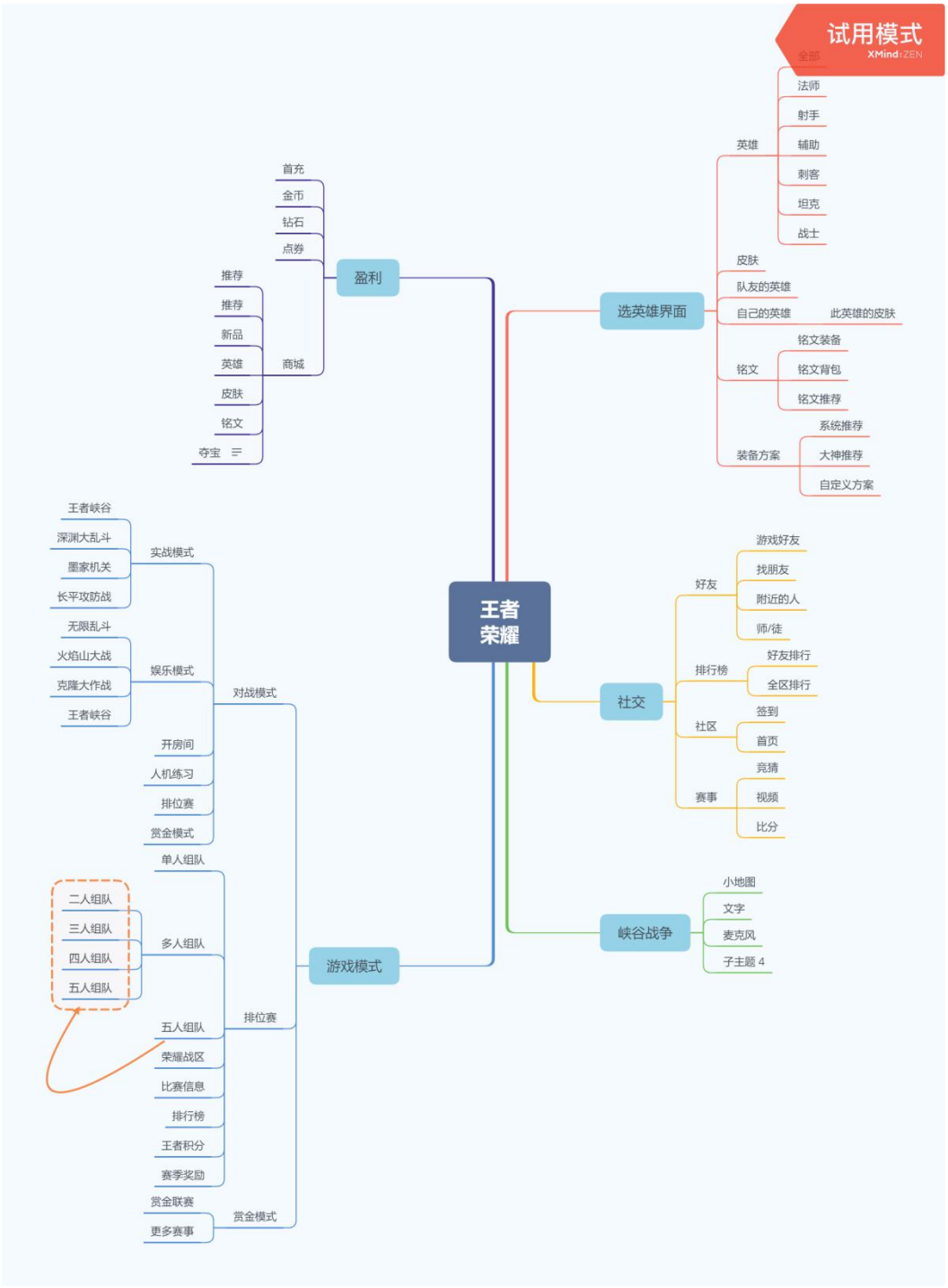
Day1:学习 Xmind 软件

学习内容：学习了思维导图的基本画法，了解了思维导图在工作中的所处的重要位置。

学习成果：通过所学到的知识画出了人力资源系统和王者荣耀的思维导图，巧用思维导图和联系，解锁思维导图的新姿势。

人力资源系统思维导图：

王者荣耀思维导图：


试用模式
XMind:ZEN
王者荣耀
盈利
首充
金币
钻石
点券
商城
推荐
推荐
新品
英雄
皮肤
铭文
夺宝
选英雄界面
英雄
全部
法师
射手
辅助
刺客
坦克
战士
皮肤
队友的英雄
自己的英雄
此英雄的皮肤
铭文
铭文装备
铭文背包
铭文推荐
装备方案
系统推荐
大神推荐
自定义方案
社交
好友
游戏好友
找朋友
附近的人
师/徒
排行榜
好友排行
全区排行
社区
签到
首页
赛事
竞猜
视频
比分
峡谷战争
小地图
文字
麦克风
子主题 4
游戏模式
对战模式
实战模式
王者峡谷
深渊大乱斗
墨家机关
长平攻防战
娱乐模式
无限乱斗
火焰山大战
克隆大作战
王者峡谷
开房间
人机练习
排位赛
赏金模式
排位赛
单人组队
多人组队
二人组队
三人组队
四人组队
五人组队
五人组队
荣耀战区
比赛信息
排行榜
王者积分
赛季奖励
赏金模式
赏金联赛
更多赛事

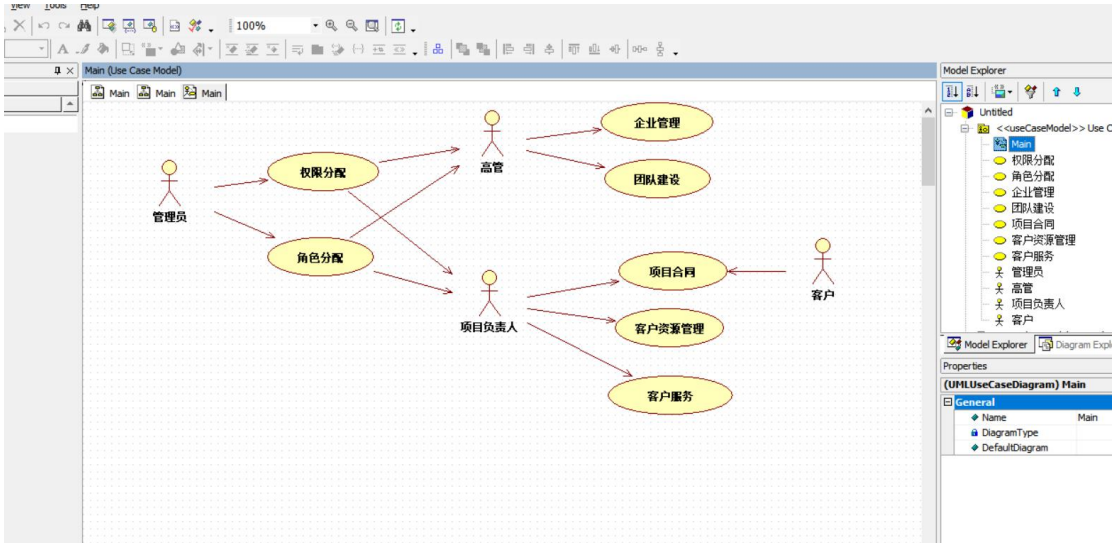
JavaShiro 框架思维导图

小结： 通过今天的学习让我知道了 Xmind 的高效以及它的可视化不是盖的，很容易就能让他人直接明确了解自己的想法和用意，并且很容易扩展，效率也很高，很适用于未来的工作之中。

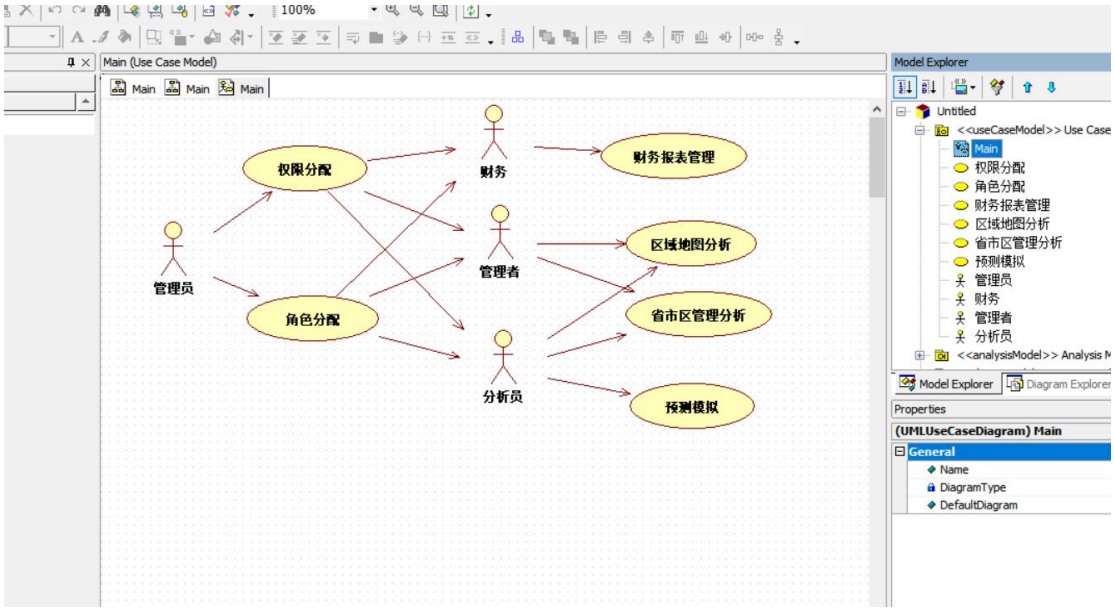
564Day2:UML 图（基于 Ehr 背景）

公司打卡流程图

5465546 用例图



DBI 用例图



小结：今天学习了 UML 图的画法，虽然今年已经上了 UML 课程，但是这样基于某个背景下再次深入了解要这样画，不仅要知道要知其然还要知其所以然，知道每个用例所处的意义所在，不冗余也不漏掉，当然也巩固了之前所学的 UML 知识，达到强化学习的目的！

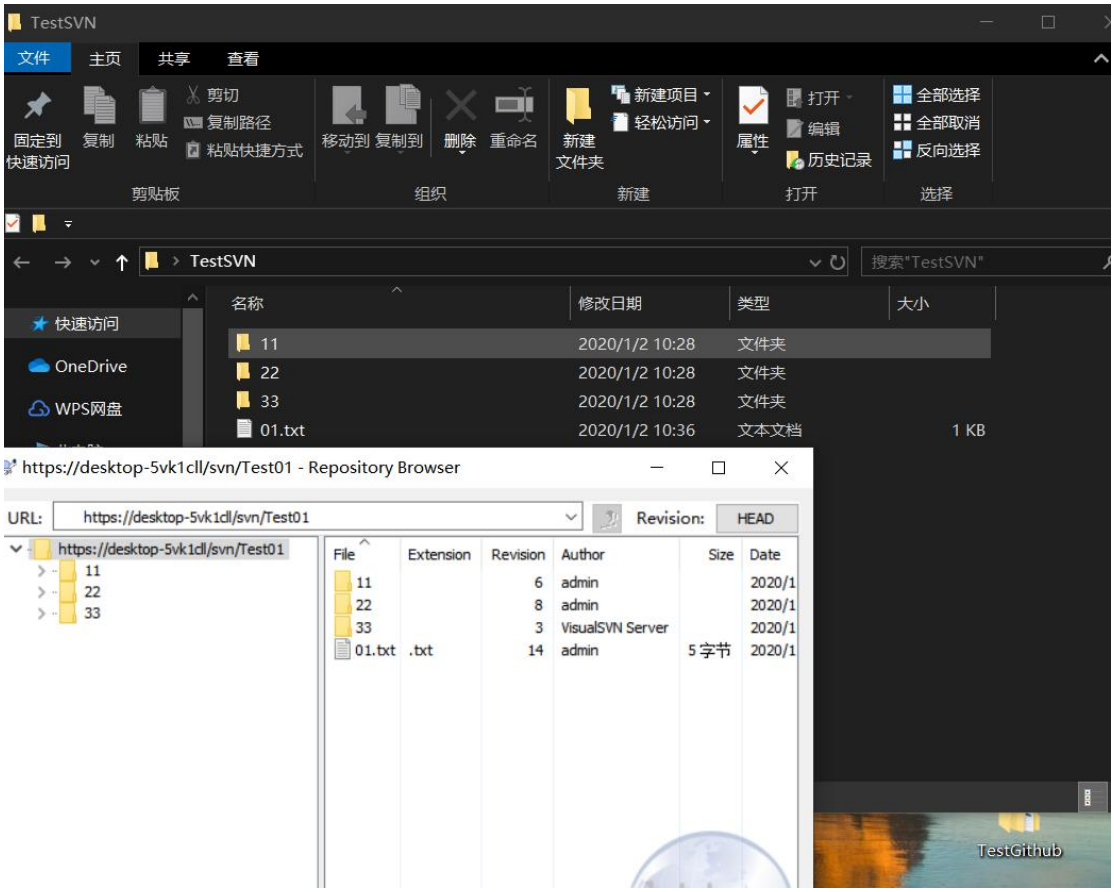
Day3:Svn 与 Git

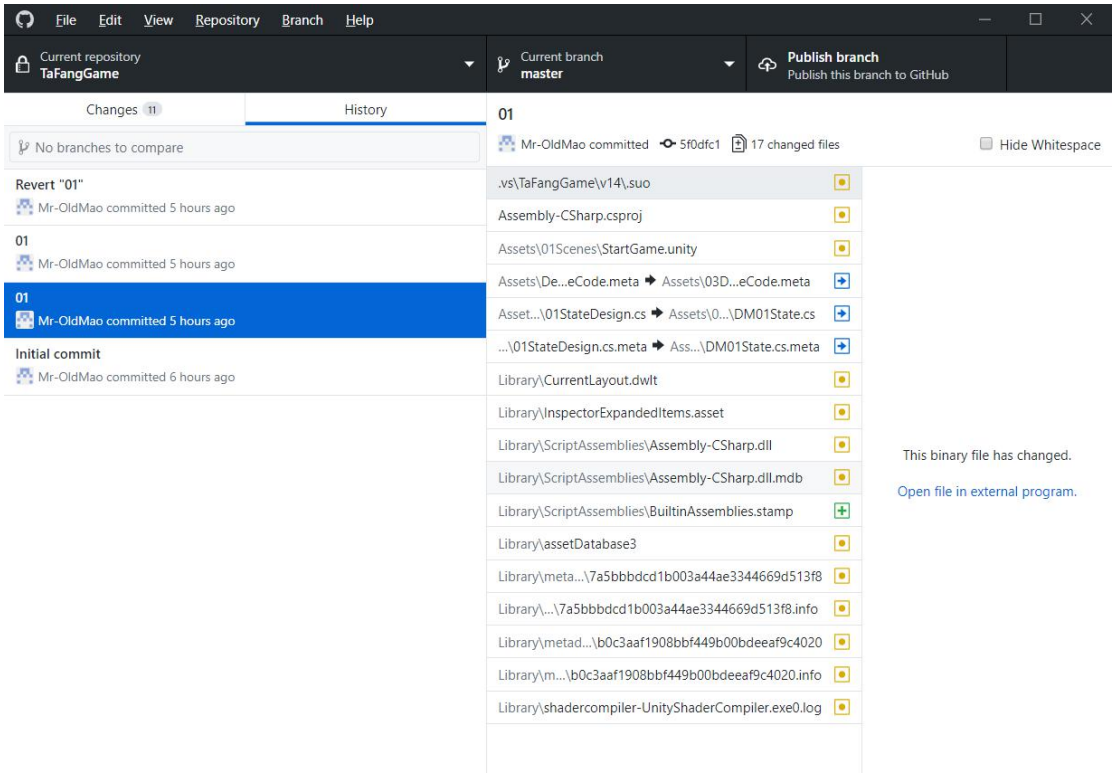
学习内容：

svn 和 git 的基本上传，下载项目操作，版本的回滚，资源的冲突的解决方法，git 和 svn 的利弊。

浅谈 svn 和 git：

通过今天老师所讲的 svn 和 git 的常用用法，以及自己之前所了解到的这两款工具，谈下自己的看法，首先毋容置疑这两款软件对于版本控制都很强大，但是也个有千秋，比如最直观的上手难度上看，应该是 svn 应该更快一些，但是上手难度对于程序员来说只是时间的问题，且此因素相对于性能，效率也不是关键。管理的方式，前者 svn 使用的是统一管理，比较依赖于服务器，当项目过大，或者参与此项目的开发人员过多可能会使服务器或数据库压力过大，后者 git 主要是针对分布式开发，所以说公共服务器压力和数据库都不会太大。Svn 对于方便快捷，上手速度具有很大优势，但 Git 对于解决冲突以及开源、离线方面更胜一筹，如果非让我在其中做出选择我也许会更偏向 git 吧。


TestSVN
文件
主页
共享
查看
固定到快速访问
复制
粘贴
剪切
复制路径
粘贴快捷方式
移动到
复制到
删除
重命名
新建文件夹
新建项目
轻松访问
属性
打开
编辑
历史记录
全部选择
全部取消
反向选择
剪贴板
组织
新建
打开
选择
TestSVN
搜索"TestSVN"
快速访问
OneDrive
WPS网盘
名称
修改日期
类型
大小
11
2020/1/2 10:28
文件夹
22
2020/1/2 10:28
文件夹
33
2020/1/2 10:28
文件夹
01.txt
2020/1/2 10:36
文本文档
1 KB
https://desktop-5vk1cll/svn/Test01 - Repository Browser
URL:
https://desktop-5vk1cll/svn/Test01
Revision:
HEAD
https://desktop-5vk1cll/svn/Test01
11
22
33
File
Extension
Revision
Author
Size
Date
11
6
admin
2020/1
22
8
admin
2020/1
33
3
VisualSVN Server
2020/1
01.txt
.txt
14
admin
5 字节
2020/1
TestGithub


File
Edit
View
Repository
Branch
Help
Current repository
TaFangGame
Current branch
master
Publish branch
Publish this branch to GitHub
Changes 11
History
No branches to compare
Revert "01"
Mr-OldMao committed 5 hours ago
01
Mr-OldMao committed 5 hours ago
01
Mr-OldMao committed 5 hours ago
Initial commit
Mr-OldMao committed 6 hours ago
01
Mr-OldMao committed
5f0dfc1
17 changed files
Hide Whitespace
.vs\TaFangGame\v14\.suo
Assembly-CSharp.csproj
Assets\01Scenes\StartGame.unity
Assets\De...eCode.meta
Assets\03D...eCode.meta
Asset...\01StateDesign.cs
Assets\0...\DM01State.cs
...\01StateDesign.cs.meta
Ass...\DM01State.cs.meta
Library\CurrentLayout.dwlt
Library\InspectorExpandedItems.asset
Library\ScriptAssemblies\Assembly-CSharp.dll
Library\ScriptAssemblies\Assembly-CSharp.dll.mdb
Library\ScriptAssemblies\BuiltinAssemblies.stamp
Library\assetDatabase3
Library\meta...\7a5bbbdcd1b003a44ae3344669d513f8
Library\...\7a5bbbdcd1b003a44ae3344669d513f8.info
Library\metad...\b0c3aaf1908bbf449b00bdeeaf9c4020
Library\m...\b0c3aaf1908bbf449b00bdeeaf9c4020.info
Library\shadercompiler-UnityShaderCompiler.exe0.log
This binary file has changed.
Open file in external program.

收获：

今天可能是这周收获最大的一天，课后我又专门研究了 github 与 vs 基于 unity3d 的版本控制，有了自己的仓库，虽然还有在其中还有一些问题（比如说 CRLF 换行问题跟自己电脑所处的操作系统有关等），但相信在放假前应该可以解决完。

Day4:优化 mysql

一：数据库优化：

1. 选取最适用的字段属性

MySQL 可以很好的支持大数据量的存取，但是一般说来，数据库中的表越小，在它上面执行的查询也就会越快。因此，在创建表的时候，为了获得更好的性能，我们可以将表中字段的宽度设得尽可能小。

2. 使用连接（JOIN）来代替子查询(Sub-Queries)

MySQL 从 4.1 开始支持 SQL 的子查询。这个技术可以使用 SELECT 语句来创建一个单列的查询结果，然后把这个结果作为过滤条件用在另一个查询中。

3. 使用联合(UNION)来代替手动创建的临时表

MySQL 从 4.0 的版本开始支持 union 查询,它可以把需要使用临时表的两条或更多的 select 查询合并的一个查询中。

4. 事务

尽管我们可以使用子查询（Sub-Queries）、连接（JOIN）和联合（UNION）来创建各种各样的查询,但不是所有的数据库操作都可以只用一条或少数几条 SQL 语句就可以完成的。更多的时候是需要用到一系列的语句来完成某种工作。但是在这种情况下，当这个语句块中的某一条语句运行出错的时候，整个语句块的操作就会变得不确定起来。

5. 锁定表

尽管事务是维护数据库完整性的一个非常好的方法，但却因为它的独占性，有时

会影响数据库的性能，尤其是在很大的应用系统中。由于在事务执行的过程中，数据库将会被锁定，因此其它的用户请求只能暂时等待直到该事务结束。

6. 使用外键

锁定表的方法可以维护数据的完整性，但是它却不能保证数据的关联性。这个时候我们就可以使用外键。

7. 使用索引

索引是提高数据库性能的常用方法，它可以令数据库服务器以比没有索引快得多的速度检索特定的行，尤其是在查询语句当中包含有 MAX(),MIN()和 ORDERBY 这些命令的时候，性能提高更为明显。

8. 优化的查询语句

绝大多数情况下，使用索引可以提高查询的速度，但如果 SQL 语句使用不恰当的话，索引将无法发挥它应有的作用。

二：优化方向：

1. SQL 以及索引的优化

首先要根据需求写出结构良好的 SQL，然后根据 SQL 在表中建立有效的索引。但是如果索引太多，不但会影响写入的效率，对查询也有一定的影响。

2. 合理的数据库是设计

根据数据库三范式来进行表结构的设计。设计表结构时，就需要考虑如何设计才能更有效的查询。

数据库三范式：

第一范式：数据表中每个字段都必须是不可拆分的最小单元，也就是确保每一列的原子性；

第二范式：满足一范式后，表中每一列必须有唯一性，都必须依赖于主键；

第三范式：满足二范式后，表中的每一列只与主键直接相关而不是间接相关(外键也是直接相关)，字段没有冗余。

3. 硬件优化

更快的 IO、更多的内存。一般来说内存越大，对于数据库的操作越好。但是 CPU 多就不一定了，因为他并不会用到太多的 CPU 数量，有很多的查询都是单 CPU。另外使用高的 IO（SSD、RAID），但是 IO 并不能减少数据库锁的机制。所以说如果查询缓慢是因为数据库内部的一些锁引起的，那么硬件优化就没有什么意义。

三：优化方案

代码优化：之所以把代码放到第一位，是因为这一点最容易引起技术人员的忽视。很多技术人员拿到一个性能优化的需求以后，言必称缓存、异步、JVM 等。实际上，第一步就应该是分析相关的代码，找出相应的瓶颈，再来考虑具体的优化策略。有一些性能问题，完全是由于代码写的不合理，通过直接修改一下代码就能解决问题的，比如 for 循环次数过多、作了很多无谓的条件判断、相同逻辑重复多次等。

总结

通过这周的实训让我懂了得了很多所不知道的知识，比如关系数据库的优化方向的知识，还有一些之前懵懵懂懂的 svn 和 git，通过这周老师的代入，当我能深入的了解到这方面的知识，着实让我受益匪浅，当然也很感激老师的教导。期待下一次课程设计。