**西北师范大学计算机学院**

课程设计论文

|  |  |
| --- | --- |
| 课程名称： | WEB应用开发技术 |
| 设计题目： | 学生信息管理系统 |
| 专业班级： | 16级软件工程(1)班 |
| 姓名： | 李朝阳 |
| 学号： | 201671020109 |
| 指导教师： | 李晓红 |
| 设计时间： | 2018-06-25 |

基于SSM的学生信息管理系统

# 问题定义

如果不知道问题是什么就试图解决这个问题，显然是盲目的，只会白白浪费时间和金钱，最终得出的结果很可能是毫无意义的。尽管确切地定义问题的必要性是十分明显的，但是在实践中它却可能是最容易被忽视的一个步骤。

虽然生活中学校的教务管理网站确实有大量的问题，比如并发性没做好（在线人数达到5、600就会卡顿），界面丑陋，教室借阅功能不完善，不能直接导入公式计算学生的绩点、综测（很多班都是在excel里自己输公式算的，规则都不一定符合学校要求）。然而挑刺容易，要我自己做一个教务网站，涉及到学生管理，教师管理，选课排课管理，教材管理，财务管理，成绩录入查询管理，教室管理等一系列问题时，我也不一定会做的比现有的系统好。这也是为什么年年都有同学要做这样一个网站，但年年都没有成功的原因吧。毕竟教务处那个管理系统听说学校也花了好几百万呢。

因此本文只做一个简化版的学生信息管理系统。涉及到班级管理，学生管理，课程管理以及成绩管理。

另外还有一个问题，本文采用了软件工程中软件生命周期的八个阶段来描述系统。夹杂了面向对象分析与设计的一些内容。目的是为了锻炼所学。使开发出的系统可理解性更好。在（虽然这里并不存在的）需求变更中不至于大量重构代码。用归一化的方法简化设计。

## 问题性质

本系统要解决的显然不是什么实际问题。只是对所学知识的一个综合运用。

## 工程目标和规模

由于时间关系，本系统实现以上几个模块后可以运行即可。

# 可行性研究

## 可行性分析

显然现有技术可以达到本系统需求。

## 系统运行环境要求

### 操作系统：

#### Windows7

#### Windows10

### 开发工具：

#### IDEA intellij 2016.3.4 专业版 试用

#### apache-tomcat-9.0.8-windows

#### Lombok插件

#### apache-maven-3.5.3

### 数据库：

#### MySQL 5.5 +

### 技术：

#### Spring+SpringMVC+Mybatis

#### 动态网页技术JSP

#### Bootstrap前端 + Bootstrap Table

### PS:如果环境搭建方面有问题，请2018.7.11后联系邮箱1092460929@qq.com

# 需求分析

这个阶段的任务仍然不是具体地解决问题，而是准确地确定“为了解决这个问题，目标系统必须做什么”，主要是确定目标系统必须具备哪些功能。

用户了解他们所面对的问题，知道必须做什么，但是通常不能完整准确地表达出他们的要求，更不知道怎样利用计算机解决他们的问题；软件开发人员知道怎样使用软件实现人们的要求，但是对特定用户的具体要求并不完全清楚。因此系统分析员在需求分析阶段必须和用户密切配合，充分交流信息，以得出经过用户确认的系统逻辑模型。通常用数据流图、数据字典和简要的[算法](http://lib.csdn.net/base/datastructure" \t "_blank" \o "算法与数据结构知识库)描述表示系统的逻辑模型。

在需求分析阶段确定的系统逻辑模型是以后设计和实现目标系统的基础，因此必须准确完整地体现用户的要求。系统分析员通常都是计算机软件专家，技术专家一般都喜欢很快着手进行具体设计，然而，一旦分析员开始谈论程序设计的细节，就会脱离用户，使他们不能继续提出他们的要求和建议。较件工程使用的结构分析设计的方法为每个阶段都规定了特定的结束标准，需求分析阶段必须提供完整准确的系统逻辑模型，经过用户确认之后才能进入下一个阶段，这就可以有效地防止和克服急于着手进行具体设计的倾向。

## 主要功能

学生信息与成绩查询。要求：

### 班级学习信息查询：实现按班级查询学生的基本信息；

### 学生详细信息查询：可按学生学号查询学生详细信息等功能

### 学生成绩查询：实现学生可以按学期或针对某一门课程来查询该课程的成绩等功能；

### 系统管理：实现用户管理和退出系统及修改密码等功能；

### 学生管理：实现学生资料的添加和修改，所在系统资料的添加和修改、所在学院资料的添加和修改；

### 课程管理模块：实现对课程资料的修改、添加、删除等功能；

### 成绩管理：实现学生成绩的浏览、修改、添加、删除等功能；

### 班级资料管理：实现班级的修改、添加、删除等功能；

### 退出系统功能

## 数据库设计

### 概念结构设计

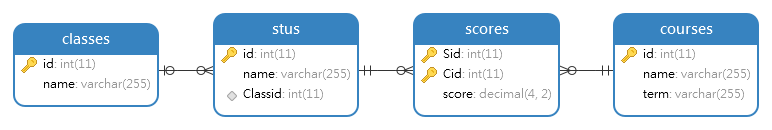


图3.1 系统ER模型

### 逻辑结构设计

stus和classes表之间有一对多的关系，在stus里加入外键。

stus和courses表之间有多对多的关系，抽取成scores表。

### 物理结构设计

#### Classes

CREATE TABLE `classes` (`id` int(11) NOT NULL AUTO\_INCREMENT,`name` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,PRIMARY KEY (`id`) )ENGINE = InnoDBAUTO\_INCREMENT = 1AVG\_ROW\_LENGTH = 0DEFAULT CHARACTER SET = utf8COLLATE = utf8\_general\_ciKEY\_BLOCK\_SIZE = 0MAX\_ROWS = 0MIN\_ROWS = 0ROW\_FORMAT = Dynamic;

#### Courses

CREATE TABLE `courses` (`id` int(11) NOT NULL AUTO\_INCREMENT,`name` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '课程名',`term` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL COMMENT '学期',PRIMARY KEY (`id`) ,INDEX `id` (`id` ASC) USING BTREE)ENGINE = InnoDBAUTO\_INCREMENT = 1AVG\_ROW\_LENGTH = 0DEFAULT CHARACTER SET = utf8COLLATE = utf8\_general\_ciKEY\_BLOCK\_SIZE = 0MAX\_ROWS = 0MIN\_ROWS = 0ROW\_FORMAT = Dynamic;

#### Scores

CREATE TABLE `scores` (`Sid` int(11) NOT NULL,`Cid` int(11) NOT NULL COMMENT '课程id',`score` decimal(4,2) NULL DEFAULT NULL,PRIMARY KEY (`Sid`, `Cid`) ,INDEX `Cid` (`Cid` ASC) USING BTREE)ENGINE = InnoDBAUTO\_INCREMENT = 0AVG\_ROW\_LENGTH = 0DEFAULT CHARACTER SET = utf8COLLATE = utf8\_general\_ciKEY\_BLOCK\_SIZE = 0MAX\_ROWS = 0MIN\_ROWS = 0ROW\_FORMAT = Dynamic;

#### Stus

CREATE TABLE `stus` (`id` int(11) NOT NULL AUTO\_INCREMENT,`name` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL,`Classid` int(11) NULL DEFAULT NULL,PRIMARY KEY (`id`) ,INDEX `Classid` (`Classid` ASC) USING BTREE)ENGINE = InnoDBAUTO\_INCREMENT = 1AVG\_ROW\_LENGTH = 0DEFAULT CHARACTER SET = utf8COLLATE = utf8\_general\_ciKEY\_BLOCK\_SIZE = 0MAX\_ROWS = 0MIN\_ROWS = 0ROW\_FORMAT = Dynamic;

#### 外键

ALTER TABLE `scores` ADD CONSTRAINT `scores\_ibfk\_1` FOREIGN KEY (`Sid`) REFERENCES `stus` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT;ALTER TABLE `scores` ADD CONSTRAINT `scores\_ibfk\_2` FOREIGN KEY (`Cid`) REFERENCES `courses` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT;ALTER TABLE `stus` ADD CONSTRAINT `stus\_ibfk\_1` FOREIGN KEY (`Classid`) REFERENCES `classes` (`id`) ON DELETE RESTRICT ON UPDATE RESTRICT;

## OOA

### 系统用况图



图3.2学生用况图



图3.3 管理员用况图

### 类图



图3.4 类图

### 顺序图

## 数据流图

# 总体设计

## HIPO图（层次图加输入／处理／输出图）

图4.1学生信息管理系统HIPO图

## OOD

### 问题域设计

学生自助查询服务和教师管理中，都涉及到对分数Scores表的查询。然而scores表里只有Sid,Cid,Score三个字段，mybatis自动逆向工程的代码也不足以支撑多对多的查询。难以提供前端所需要的数据（包含学生姓名，课程名称等）。

因此，为了解决这个问题，在符合设计模式六大原则中的合成复用原则的前提下。本系统决定在Java Pojo中的Score加入新的字段。

#### Java Pojo中的Score加入新的字段

1. @Data
2. @NoArgsConstructor@AllArgsConstructor
3. @Component
4. **public** **class** Score **extends** ScoreKey {
5. **private** BigDecimal score;
6. **private** Course course;
7. **private** Stu stu;
8. }

#### 接着实现Dao层ScoreMapper从数据库中提取数据

1. //贼恶心，不如写XML文件 参考文档
2. //https://blog.csdn.net/qq\_32786873/article/details/78297551
3. //    @Select("<script>"+
4. //            "select s.Sid ,  t.name stuName, t.pwd , s.Cid, c.name as courseName, c.term ,s.score from stus as t ,scores as s, courses as c where t.id=s.sid and c.id=s.cid"
5. //            +"<if test='startNo!=null and pageSize != null '>"
6. //            +""
7. //            + "</if>"
8. //            +"</script>")
9. @Results({
10. @Result(property = "score", column = "score"),
11. @Result(property = "stu.id", column = "Sid"),
12. @Result(property = "stu.name", column = "stuName"),
13. @Result(property = "stu.pwd", column = "pwd"),
14. @Result(property = "stu.academy", column = "academy"),
15. @Result(property = "stu.classid", column = "classid"),
16. @Result(property = "course.id", column = "Cid"),
17. @Result(property = "course.name", column = "courseName"),
18. @Result(property = "course.term", column = "term")
19. //            @Result(property = "orderItems", javaType = List.class, column = "id",
20. //                    many = @Many(select = "com.how2java.mapper.OrderItemMapper.listByOrder"))
21. })
22. @SelectProvider(type=ScoreDynaSqlProvider.**class**, method="listInnorJoin")
23. List<Score> selectAllDataRightJoinStuAndCourse(Integer sid,  Integer cid, String cterm);

#### 生成动态SQL

* 1. **public** **class** ScoreDynaSqlProvider {
  2. //https://blog.csdn.net/owen\_william/article/details/51815506
  3. **public** String listRightJoin(Map<String, Object> map) {
  4. //        0  1997
  5. //        1  null
  6. //        param1  1997
  7. //        param2  null
  8. //        Set<Map.Entry<String, Object>> entries = map.entrySet();
  9. //        for(Map.Entry<String, Object> i : entries){
  10. //            System.out.println(i.getKey()+"  "+i.getValue());
  11. //        }
  12. Object sid = map.get("0");
  13. Object cid =  map.get("1");
  14. Object cterm =  map.get("2");
  15. //        System.out.println(sid);
  16. SQL sql = **new** SQL()
  17. .SELECT("s.sid sid, c.id cid, c.name as courseName, c.term ,s.score")
  18. .FROM("scores s")
  19. .RIGHT\_OUTER\_JOIN("courses as c on s.cid = c.id");
  20. //        System.out.println("listInnorJoin:  "  +sid);
  21. **if**(sid != **null**){
  22. //            System.out.println("amuse");
  23. sql = sql.WHERE("sid="+sid).OR().WHERE("sid is null");
  25. }
  26. **if**(cid != **null**){
  27. sql = sql.WHERE("cid="+cid);
  28. }
  29. **if**(cterm!=**null**){
  30. sql = sql.WHERE("c.term like '%"+cterm+"%'");
  31. }
  33. String s =sql
  34. .toString();
  35. System.out.println(s);
  36. **return** s;
  37. }
  38. }

### 人机交互设计

#### 初始登录界面：



图4.2 初始登录界面图

#### 查询信息界面

图4.3 查询信息界面图

#### 修改信息界面



图4.4 修改信息界面图

## 数据流图

学生选课数据流图

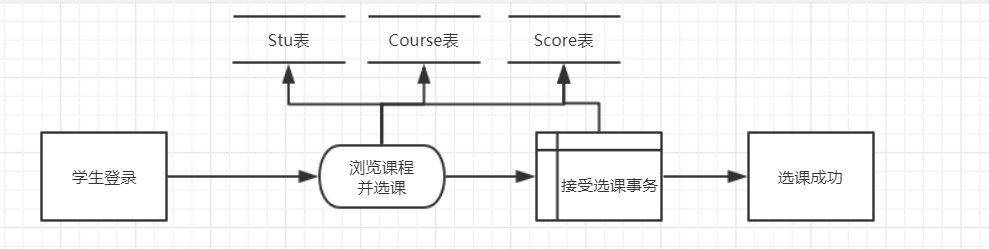


图4.4学生选课数据流图

这个阶段必须回答的关键问题是：“概括地说，应该如何解决这个问题？”

首先，应该考虑几种可能的解决方案。列如，目标系统的一些主要功能是用计算机自动完成还是用人工完成；如果使用计算机，那么是使用批处理方式还是人机交互方式；信息存储使用传统的文件系统还是[数据库](http://lib.csdn.net/base/mysql)……。通常至少应该考虑下述几类可能的方案：

低成本的解决方案。系统只能完成最必要的工作，不能多做一点额处的工作。

中等成本的解决方案。这样的系统不仅能够很好地完成预定的任务，使用起来很方便，而且可能还具有用户没有具体指定的某些功能和特点。虽然用户没有提出这些具体要求，但是系统分析员根据自己的知识和经验断定，这些附加的能力在实践中将证明是很有价值的。

高成本的“十全十美”的系统。这样的系统具有用户可能希望有的所有功能和特点。

系统分析员应该使用系统流程图或其他工具描述每种可能的系统，估计每种方案的成本和效益，还应该在充分权衡各种方案的利弊的基础上,推荐一个较好的系统(最佳方案),并且制定实现所推荐的系统的详细计划。如果用户接受分析员推荐的系统，则可以着手完成本阶段的另一项主要工作。

上面的工作确定了解决问题的策略以及目标系统需要哪些程序，但是，怎样设计这些程序呢？结构设计的一条基本原理就是程序应该模块化，也就是一个大程序应该由许多规模适中的模块按合理的层次结构组织而成。总体设计阶段的第二项主要任务就是设计软件的结构，也就是确定程序由哪些模块组成以及模块间的关系。通常用层次图或结构图描绘软件的结构。

# 详细设计

## PAD图

教师查询修改学生成绩的PAD图：

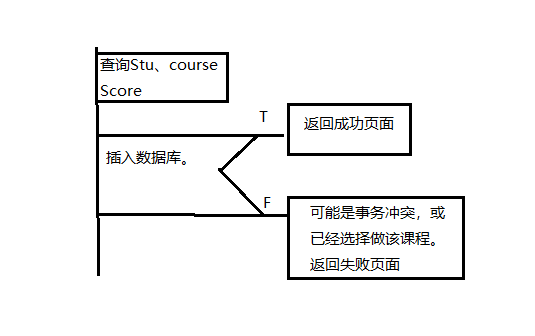


图5.1 教师查询修改学生成绩PAD图

总体设计阶段以比较抽象概括的方式提出了解决问题的办法。详细设计阶段的任务就是把解法具体化，也就是回答下面这个关键问题：“应该怎样具体地实现这个系统呢？”

这个阶段的任务还不是编写程序，而是设计出程序的详细规格说明。这种规格说明的作用很类似于其他工程领域中工程师经常使用的工程蓝图，它们应该包含必要的细节，程序员可以根据它们写出实际的程序代码。

通常用或PDL语言（过程设计语言）描述详细设计的结果。

# 编码和单元测试

## MyBatis逆向工程

生成java里的pojo、mapper、xml配置文件

## 数据准备

由于只是为了白盒测试，Random生成测试数据。

### Score

1. **public** **class** ScoreDaoTest {
2. @Autowired
3. **private** ScoreMapper scoreMapper;
4. **private** String[] term = {"大一上", "大一下", "大二上", "大二下"};
5. **private** String[] Coursename = {"线性代数", "高数", "java程序设计", "算法设计与分析"};
6. **private** Random random=**new** Random();
7. @Test
8. **public** **void** batchInsert(){
9. //        for (int i = 84; i < 164; i++) {
10. //            for (int j = 1; j <= 4; j++) {
11. //                Score s=new Score(i, j, new BigDecimal(random.nextFloat()\*100));
12. //                System.out.println(scoreMapper.insert(s));
13. //            }
14. //        }
15. **for** (**int** j = 1; j <= 4; j++) {
16. Score s=**new** Score(1997, j, **new** BigDecimal(random.nextFloat()\*100));
17. System.out.println(scoreMapper.insert(s));
18. }
19. }
20. }

### Stu

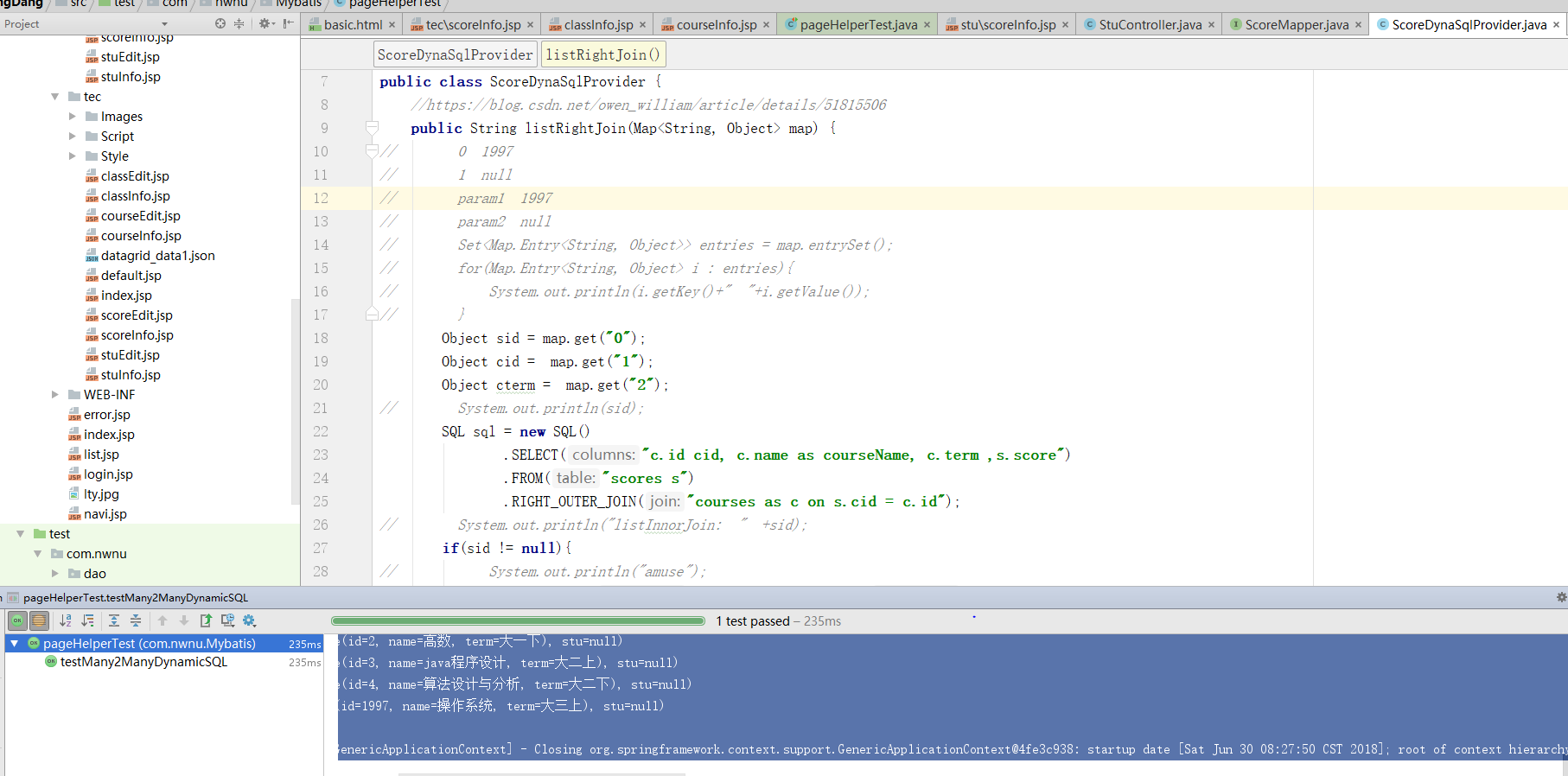
1. **public** **class** StuDaoTest {
2. @Autowired
3. **private** StuMapper stuMapper;
5. **private** String[] names = {"张三", "李四", "王五","王二麻子"};
6. **private** Random random=**new** Random();
7. @Test
8. **public** **void** batchInsert(){
9. **for** (**int** i = 0; i < 80; i++) {
10. Stu s=**new** Stu();
11. s.setName(names[i%4]+i);
12. s.setPwd("1234");
14. s.setClassid(random.nextInt(2)+1);
15. System.out.println(stuMapper.insert(s));
16. }
17. }
18. }

### Class**与**Course不在赘述。自行参阅源码

## ScoreDaoTest

代码：

1. @RunWith(SpringJUnit4ClassRunner.**class**)
2. @ContextConfiguration("classpath:app.xml")
3. **public** **class** pageHelperTest {
4. @Autowired
5. **private** StuMapper stuMapper;
6. @Autowired
7. **private** ScoreMapper scoreMapper;
8. @Test
9. **public** **void** testList() {
10. PageHelper.startPage(0, 5);
11. System.out.println(stuMapper);
12. List<Stu> cs=stuMapper.selectByExample(**new** StuExample());
13. **for** (Stu c : cs) {
14. System.out.println(c);
15. }
16. System.out.println(cs.size());
17. }
18. @Test
19. **public** **void** testMany2ManyDynamicSQL() {
20. List<Score> scores = scoreMapper.selectAllDataJoinStuAndCourse(1997, **null**, "大一");
21. **for** (Score s : scores) {
22. System.out.println(s);
23. }
24. System.out.println(scores.size());
25. }
26. @Test
27. **public** **void** testAnnotation(){
28. List<Score> scores = scoreMapper.selectAllDataJoinStuAndCourse(**null**, **null**, **null**);
29. **for** (Score s : scores) {
30. System.out.println(s);
31. }
32. System.out.println(scores.size());
33. }
35. }

测试结果：

## 集成各个模块，通过Controller进行Modle和View的整合。

这个阶段的关键任务是写出正确的容易理解、容易维护的程序模块。

程序员应该根据目标系统的性质和实际环境，选取一种适当的高级程序设计语言（必要时用汇编语言），把说细设计的结果翻译成用选定的语言书写的程序，并且仔细测试编写出的每一个模块。

# 综合测试

这个阶段的关键任务是通过各种类型的测试（及相应的调试）使软件达到预定的要求。

## 系统事务支持测试。

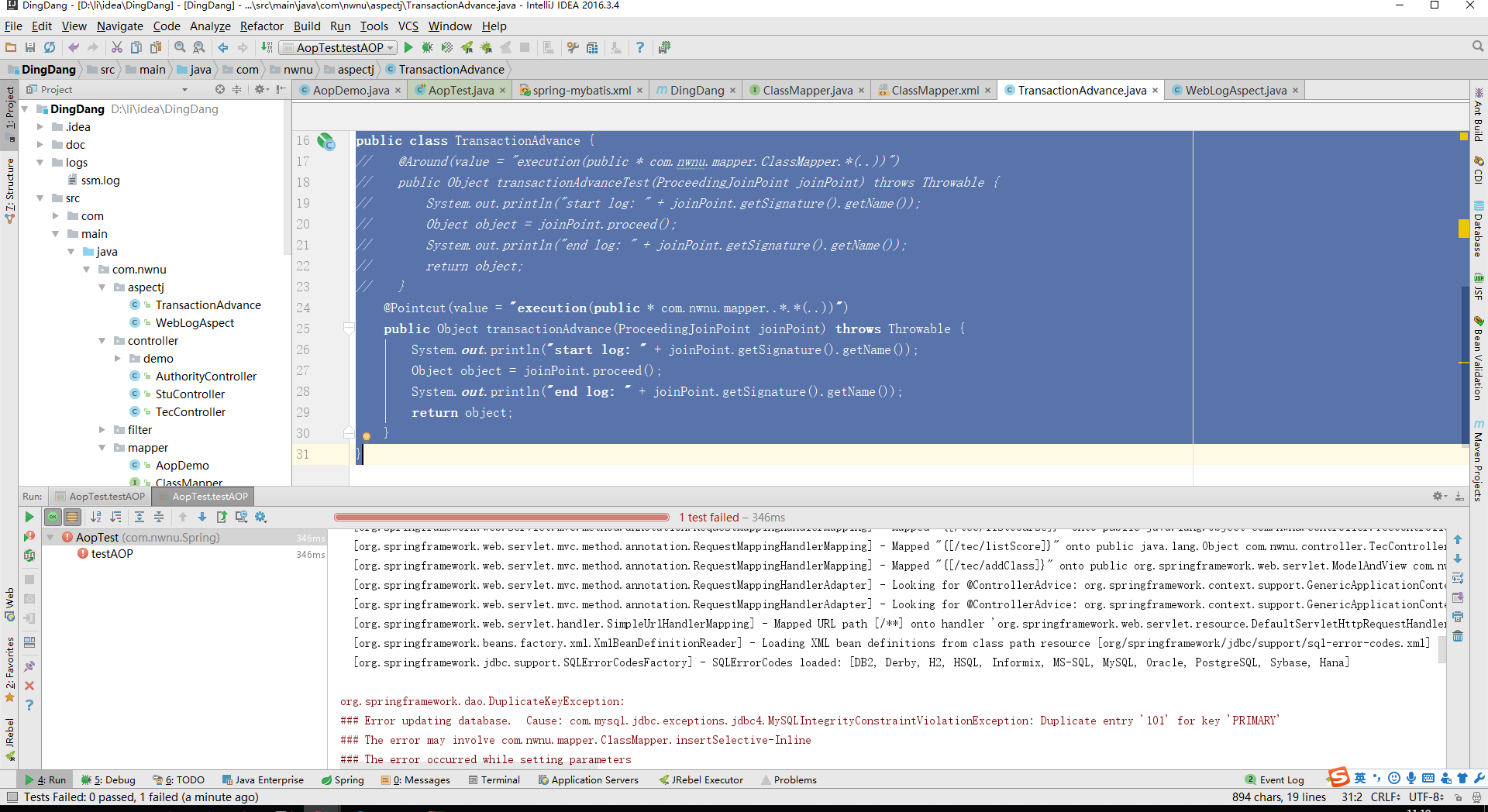
### 加入AOP支持类

1. @Aspect
2. @Component
3. @Transactional//没有用？所以别人都加到Service上？
4. **public** **class** TransactionAdvance {
5. //    @Around(value = "execution(public \* com.nwnu.mapper.ClassMapper.\*(..))")
6. //    public Object transactionAdvanceTest(ProceedingJoinPoint joinPoint) throws Throwable {
7. //        System.out.println("start log: " + joinPoint.getSignature().getName());
8. //        Object object = joinPoint.proceed();
9. //        System.out.println("end log: " + joinPoint.getSignature().getName());
10. //        return object;
11. //    }
12. @Pointcut(value = "execution(public \* com.nwnu.mapper..\*.\*(..))")
13. **public** Object transactionAdvance(ProceedingJoinPoint joinPoint) **throws** Throwable {
14. System.out.println("start log: " + joinPoint.getSignature().getName());
15. Object object = joinPoint.proceed();
16. System.out.println("end log: " + joinPoint.getSignature().getName());
17. **return** object;
18. }
19. }

### 测试代码

1. @RunWith(SpringJUnit4ClassRunner.**class**)
2. @ContextConfiguration("classpath:app.xml")
3. **public** **class** AopTest {
4. @Autowired
5. **private** AopDemo aopDemo;
6. @Autowired
7. **private** ClassMapper classMapper;
9. @Test
10. //    @Transactional
11. **public** **void** testAOP() **throws** SQLException, ClassNotFoundException {
12. //        aopDemo.doSth();
13. //        int insert2 = classMapper.insertSelective(new Class(101, "宇宙超级无敌软件一班"));
14. **int** insert = classMapper.insertSelective(**new** Class(**null**, "宇宙超级无敌软件一班"));
15. //        System.out.println(insert);
17. List<Class> classes = classMapper.selectByExample(**new** ClassExample());
18. classes.stream().forEach(System.out::println);
19. //        stuMapper.countByExample(new StuExample());
20. }
21. }
22. @Aspect
23. @Component
24. @Transactional//没有用？所以别人都加到Service上？
25. **public** **class** TransactionAdvance {
26. //    @Around(value = "execution(public \* com.nwnu.mapper.ClassMapper.\*(..))")
27. //    public Object transactionAdvanceTest(ProceedingJoinPoint joinPoint) throws Throwable {
28. //        System.out.println("start log: " + joinPoint.getSignature().getName());
29. //        Object object = joinPoint.proceed();
30. //        System.out.println("end log: " + joinPoint.getSignature().getName());
31. //        return object;
32. //    }
33. @Pointcut(value = "execution(public \* com.nwnu.mapper..\*.\*(..))")
34. **public** Object transactionAdvance(ProceedingJoinPoint joinPoint) **throws** Throwable {
35. System.out.println("start log: " + joinPoint.getSignature().getName());
36. Object object = joinPoint.proceed();
37. System.out.println("end log: " + joinPoint.getSignature().getName());
38. **return** object;
39. }
40. }

### 效果图



比如两个管理员同时对一个学生的同一门课程的成绩进行修改

## 集成测试

根据设计的软件结构，把经过单元测试检验的模块按某种选定的策略装配起来，在装配过程中对程序进行必要的测试。

## 压力测试

## 验收测试

所谓是所谓验收测试则是按照规格说明书的规定（通常在需求分析阶段确定），由用户（或在用户积极参加下）对目标系统进行验收。这个就由老师来完成了。

必要时还可以再通过现场测试或平行运行等方法对目标系统进一步测试检验。

为了使用户能够积极参加验收测试，并且在系统投入生产性运行以后能够正确有效地使用这个系统，通常需要以正式的或非正式的方式对用户进行培训。

# 软件维护

通过各种必要的维护活动使系统持久地满足用户的需要

## 改正性维护

也就是诊断和改正在使用过程中发现的软件错误；

## 适应性维护

即修改软件以适应环境的变化；

## 完善性维护

即根据用户的要求改进或扩充软件使它更完善；

## 预防性维护

即修改软件为将来的维护活动预先做准备。

虽然没有把维护阶段进一步划分成更小的阶段，但是实际上每一项维护活动都应该经过提出维护要求（或报告问题），分析维护要求，提出维护要求，提出维护方案，审批维护方案，确定维护计划，修改软件设计，修改程序，测试程序，复查验收等一系列步骤，因此实质上是经历了一次压缩和简化了的软件定义和开发的全过程。

都应该经过提出维护要求（或报告问题），分析维护要求，提出维护要求，提出维护方案，审批维护方案，确定维护计划，修改软件设计，修改程序，测试程序，复查验收等一系列步骤，因此实质上是经历了一次压缩和简化了的软件定义和开发的全过程。

# 参考文献

[1] ：软件工程的几个步骤 <https://blog.csdn.net/birthmarkqiqi/article/details/53433375>

[2] ：张海藩、牟永敏编著.《软件工程导论（第6版）》.清华大学出版社 2013

[3] ：麻志毅编著.《面向对象分析与设计.第2版》. 机械工业出版社

[4] ：王珊、萨师煊编著.《数据库系统概论（第5版）》. 高等教育出版社

[5] ：设计模式。[百度百科](https://baike.baidu.com/item/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/1212549?fr=aladdin#3_2)、[Wikepeda](https://en.wikipedia.org/wiki/Software_design_pattern)

# 致谢

岁月如歌，光阴似箭，大学的头两年生活即将结束。经历了两年网页设计课程的学习与思考，也经历了蓝桥杯的失利，立项的种种困难，想学各种技术的渴望和坚持不下去的挫败，我深深体会到了写作论文时的那份宁静与思考。回首这两年的求学历程，对那些引导我、帮忙我、激励我的人，我心中充满了感激。

首先要感谢C语言乃至计算机的启蒙老师张志昌教授。大一一入学时，我们还是什么都不懂的小白，张老师耐心又不失可爱（暴躁）的教我们编程的种种规范，代码要缩进，命名要驼峰，ijk表循环……之后也给我们介绍了进一步学习的方向，POJ等。能师从这位老师，我为自己感到庆幸。在此谨向张志昌老师表示我最诚挚的敬意和感谢！

还要感谢中关村的梁云亮老师。梁云亮老师与张志昌不同，注重实践中教学，“最好的学习方式就是照抄”。虽说一些同学这太过刻板，学过后就忘。但如果不是他手把手的教我们怎么用eclipse，我怕是真要一个一个的敲“import java.util.\*；”这句可以自动导入的代码，之后也是他无私的教了我们SSM的配置与运用。谨在此表示衷心的感谢。

以上两个老师是编程方面对我益处最大的，但郝老师也在生活寄予了我很多帮助，不光给我提供了3D实验室的环境。迷茫时就去找郝老师聊一聊也成为了我们实验室的传统。

还要感谢李晓红老师。里老师在论文的写作中给予了许多指导与推荐，谨在此表示衷心的感谢！

同时，我要感谢所有教导过我、关心过我的老师。你们为我的学业倾注了超多心血，你们为人师表的风范令我敬仰，严谨治学的态度令我敬佩。还要感谢感谢牛老师、马老师、蒋老师、陈老师和计算机学院的其他各位老师对我的一贯帮忙！

特别感谢我女朋友，二十一载终于等到你啦，久违久违。两个人在一起的感觉真的不一样呢。春朝，梦蝶。人生不满百，常怀千岁忧。夫天地者万物之逆旅也，光阴者百代之过客也。而浮生若梦，为欢几何？关于人生，太多先贤给过论述了。我有时也会迷茫，从来都没有初心又何谈不忘？只是认识你之后，我越发觉当下之可贵。未来不迎，过往不恋。每天进步一点点，为了更好的明天。共勉之。

感谢一向关心与支持我的同学和朋友们！我的朋友，感谢你们的鼓励和帮忙。还要感谢的是我寝室的室友。我们朝夕相处，共同进步，感谢你们给予我的所有关心和帮忙。同窗之谊，我将终生难忘！

最后要感谢我生活学习了两年的母校——nwnu，母校给了我一个宽阔的学习平台，让我不断吸取新知，充实自己。