南京大学

计算机科学与技术系

软件工程实验报告

实验名称： 软件分析与测试

学　　号： 171240501

姓　　名： 匡舒磊

指导教师： 王林章

实验地点： 基础实验楼乙 208

实验时间： 2019.11.21

一、 实验名称

软件分析与测试

二、 实验目的

掌握单元测试

三、 实验要求

1. 掌握单元测试，针对自己开发的网站编写单元测试用例
2. 完成实验报告

四、 实验环境

1. 软件：IntelliJ Idea
2. 硬件：笔记本

五、 实验内容

1. 针对自己开发的网站的代码编写单元测试用例
2. 执行测试用例查看结果
3. 完成实验报告

六、 实验步骤

1. 编写 HelloService 单元测试代码

   各测试点的测试作用如代码中注释所说，分别测试了以下的情况：

   插入用户：插入之前不存在的用户，插入一个 ID 已经存在的用户

   查看用户列表：在增删前后查看

   根据 id 查看用户：用户存在，用户不存在

   更新用户信息：正常更新，更新一个不存在的用户

   删除用户信息：删除一个存在的用户，删除一个不存在的用户

```java
@Test
public void testUsers_1() throws Exception {
    //检测初始getUserList是否正常
    List<HelloUser> users = helloService.getUserList();
    assertFalse( message: "User not null", condition: users == null);
    assertNotNull(users);
    assertEquals(users.size(), actual: 2);
    assertEquals(users.get(0).getName(), actual: "鸡肉工厂");
    //检测InsertUser与GetOne在输入正常的情况下是否正常
    Map<String, String> newusr = new HashMap<String, String>();
    newusr.put("id","003");
    newusr.put("name","stupid factory");
    newusr.put("password","abcde");
    helloService.InsertUser(newusr);
    HelloUser newadd = helloService.getOne( id: "003");
    assertEquals(newadd.getName(), actual: "stupid factory");
    assertEquals(newadd.getPassword(), actual: "abcde");
    //检测UpdateByID与getUsrList是否正常
    newusr.replace("password","edcba");
    helloService.UpdateByID(newusr);
    List<HelloUser> usrs2 =  helloService.getUserList();
    assertEquals(usrs2.size(), actual: 3);
    assertEquals(usrs2.get(2).getPassword(), actual: "edcba");
    //检测DeleteByID函数与getOne在ID不存在的情况下是否返回空值
    helloService.DeleteByID("003");
    HelloUser deletedUsr = helloService.getOne( id: "003");
    assertEquals(deletedUsr.getName(), actual: null);
    //检测删除不存在的ID是否生效
    List<HelloUser> Deleteusrs = helloService.getUserList();
    assertEquals(Deleteusrs.size(), actual: 2);
    helloService.DeleteByID("010");
    List<HelloUser> Deleteusrs2 = helloService.getUserList();
    assertEquals(Deleteusrs2.size(), actual: 2);
}


 // 检测插入相同ID用户时是否抛出异常
 @Test (expected = Exception.class)
 public void testInsertException() throws Exception{
     Map<String,String> params=new HashMap<>();
     params.put("id","001");
     params.put("name","interestingFactory");
     params.put("password","ispassword");
     helloService.InsertUser(params);
     fail("插入相同userID的用户没有抛出异常");
 }

 //检测更新不存在用户时是否抛出异常
 @Test (expected = Exception.class)
 public void testUpdateException() throws Exception{
     Map<String,String> params=new HashMap<>();
     params.put("id","005");
     params.put("name","interestingFactory");
     params.put("password","ispassword");
     helloService.UpdateByID(params);
     fail("更新不存在的用户没有抛出异常");
 }
```

2. 编写 AppcheckService 单元测试代码

在下面的代码中测试了以下的情况：

插入新材料：插入之前不存在的材料，插入一个 ID 已经存在的材料

查看全部材料列表：在增删前后查看

根据用户 id 查看其材料列表：在增删前后查看

根据 id 查看材料：材料存在，材料不存在

更新材料：正常更新，更新一个不存在的材料

删除材料：删除一个存在的材料，删除一个不存在的材料

```
public void testApps() throws Exception{
    //检测初始getAPPList是否正常
    List<Appcheck> apps1 = appcheckService.getAppList();
    assertFalse( message: "app not null", condition: apps1 == null);
    assertNotNull(apps1);
    assertEquals(apps1.size(), actual: 4);
    assertEquals(apps1.get(1).getPname(), actual: "打工APP");
    //检测初始getAppByUsr是否正常
    List<Appcheck> apps001 = appcheckService.getAppByUsr( uid: "001");
    assertFalse( message: "app not null", condition: apps001 == null);
    assertNotNull(apps001);
    assertEquals(apps001.size(), actual: 3);
    assertEquals(apps001.get(2).getPname(), actual: "工业鸡切割APP");
    List<Appcheck> apps005 = appcheckService.getAppByUsr( uid: "005");
    assertEquals(apps005.size(), actual: 0);
    //检测GetOneApp在检测不存在的提交材料的时候是否返回空
    Appcheck unexistedApp = appcheckService.getOneApp( pid: "888888");
    assertEquals(unexistedApp.getPname(), actual: null);
    //检测InsertUser与GetOne在输入正常的情况下是否正常
    Map<String, String> newapp = new HashMap<String, String>();
    newapp.put("pid","888888");
    newapp.put("uid","005");
    newapp.put("pname","LOLapp");
    newapp.put("pcate","生产类");
    newapp.put("status","待审核");
    appcheckService.InsertApp(newapp);
    Appcheck testNew = appcheckService.getOneApp( pid: "888888");
    assertEquals(testNew.getPname(), actual: "LOLapp");
    assertEquals(testNew.getPcateinfo(), actual: null);
    //检测UpdateByID与getAppByUsr是否正常
    newapp.put("pcateinfo","nice cate");
    newapp.replace("pname","DotaAPP");
    appcheckService.UpdateByID(newapp);
    apps005 = appcheckService.getAppByUsr( uid: "005");
    assertEquals(apps005.size(), actual: 1);
    assertEquals(apps005.get(0).getPcateinfo(), actual: "nice cate");
    assertEquals(apps005.get(0).getPname(), actual: "DotaAPP");
    //检测DeleteByID函数
    appcheckService.DeleteByID( pid: "888888");
    List<Appcheck> appList2 = appcheckService.getAppList();
    assertEquals(appList2.size(), actual: 4);
    //检测删除不存在的ID是否生效
    appcheckService.DeleteByID( pid: "66666");
    List<Appcheck> appList3 = appcheckService.getAppList();
    assertEquals(appList3.size(), actual: 4);
}
```

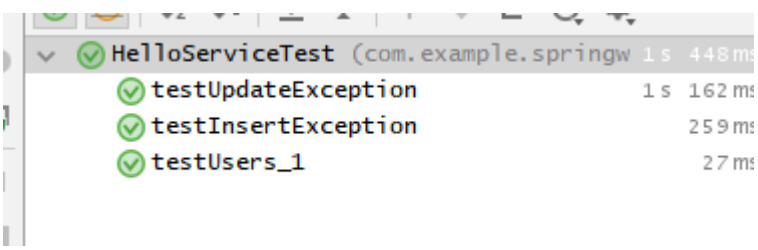
```java
@Test (expected = Exception.class)
public void testInsertException() throws Exception{
    Appcheck test = appcheckService.getOneApp( pid: "11254");
    assertEquals(test.getPname(), actual: "量子波动鸡增长APP");
    Map<String,String> app = new HashMap<>();
    app.put("pid","11254");
    app.put("pname","量子鸡APP");
    app.put("pcate","bb类");
    app.put("pcateinfo","这是一个分类说明");
    app.put("psecurity","I");
    app.put("psecuinfo","it is a secu info");
    app.put("padvanced","using liangzi is advanced");
    app.put("pcheckinfo","检查好了");
    app.put("uid","001");
    app.put("status","审核中");
    app.put("rate","66");
    appcheckService.InsertApp(app);
    fail("插入相同ID的APP材料没有抛出异常");
}

@Test (expected = Exception.class)
public void testUpdateException() {
    Map<String,String> app = new HashMap<>();
    app.put("pid","77777");
    app.put("pname","不存在的APP");
    appcheckService.UpdateByID(app);
    fail("更新不存在的材料没有抛出异常");
}
```

七、 实验结果

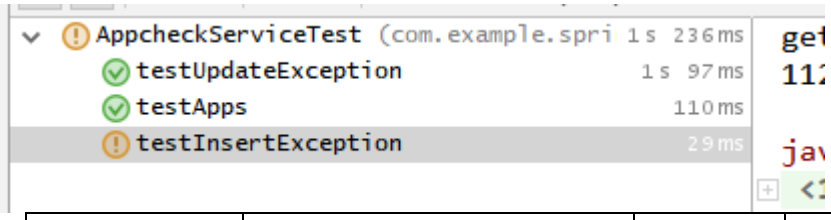
1. HelloService 测试结果

测试方法	功能描述	测试用例	成功	失败
InsertUser	插入新用户	2	1	1
UpdateByID	根据用户 ID 更新信息	2	1	1
getUserList	获取全部用户列表	4	4	0
DeleteByID	根据 ID 删除用户	2	2	0
getOne	根据 ID 获取用户信息	2	2	0



虽然测试样例显示通过了，但是在此处我们应用的代码为`@Test (expected = Exception.class)`，也就是说在这种测试情况下，会抛出异常，但是我们在代码处没有进行异常处理，故而此处姑且认为是测试失败了

2. AppcheckService 测试



测试方法	功能描述	测试用例	成功	失败
InsertApp	插入新材料	2	1	1
UpdateByID	根据材料 ID 更新信息	2	1	1
getAppList	获取全部材料	4	4	0
DeleteByID	根据 ID 删除材料	2	2	0
getOneApp	根据材料 ID 获取材料信息	2	2	0
GetAppByUsr	根据用户 ID 获取其全部材料	2	2	0

对于 InsertApp 失败的原因是，要求所有材料的 ID 是无法重复的，但是在加入相同 ID 的材料的时候，没有对其进行异常的抛出而是直接进行了 insert。

对于 update 失败的原因认为是其会产生异常，但是我们没有对其进行异常处理，故而姑且认为是失败的。

八、 结果分析与结论

本次实验做实验与写报告共花费了 3 小时。从本次实验中不难发现单元测试的一个最大的好处就在于不需要对其他模块的正确性进行处理而只需要考虑本模块的正确性。但是需要考虑到测试用例需要覆盖尽可能多的情况。但是让我疑惑的是，在本次实验中一些错误的测试我在 controller 里面进行了限制保证了其不会发生，那么在能保证的情况下我是否还需要进行测试其实是一件值得商榷的事情。

实验评分： _____

指导教师签字： _____

年　月　日