交大交交——交大二手交易信息平台

测试计划

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <14/04/2020> | <1.0> | 初始化测试计划文档 | 谭骁 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 背景 4

1.3 范围 4

2. 资源 8

2.1 角色 8

2.2 测试模型 9

2.3 测试日志 9

2.4 缺陷报告 9

测试计划

# 简介

本测试报告是针对交大交交二手交易APP的测试文档。交大交交二手交易APP能让交大内的同学在平台上进行二手交易信息的发布。本次测试针对面向用户输出的四个模块进行测试。本次测试为黑盒测试，采用了等价类和决策表测试的方法，所有测试用例按服务名存储在四个文件中，本次测试采用goconvey作为测试框架，gomod管理工程的依赖，由于测试仅包含四个模块，针对相应服务中存在的依赖，使用mock的方式进行处理，默认在测试该服务时，其他服务的正确性，使用VsCode编写和运行测试用例。

## 目的

交大交交 的这一“测试计划”文档有助于实现以下目标：  
本文档针对交大二手交易信息平台“交大交交”（简称交大交交）的黑盒测试，记录相关测试内容，测试方法和测试环境。

* 交大交交后端基于go-micro微服务架构，前端基于react-native框架。前后端使用基于HTTP协议的Restful API来进行通信、数据传递，微服务间使用RPC协议进行通信。一共拥有Auth，Avatar，BuyInfo，Content，File，Message，SellInfo，Tag，User 9个微服务。
* 列出推荐的测试需求（高层次）。
* 本次测试主要使用的方法为弱健壮等价类测试，基于决策表测试。
* 本次测试需要在测试环境部署该项目，需要4台4核8G内存的服务器，需要测试服务器1台4和8G服务器。
* 项目最终将交付项目计划，测试计划书，需求规约书。

## 背景

交大交交APP开发的目标是成为上海交通大学师生在进行二手交易时会首选的一款APP，软件定位是面向上海交通大学全体师生的最普及的二手交易APP。

二手交易是指消费者将购买于企业或代理商、零售商后使用一段时间，或者未使用过的商品，作为一种新的商品再进行交易流通的行为。大学生这一群体，生活区域相近，消费心理相似，购买的需求商品具有极高的重合度，有着极高的二手交易的潜在需求。譬如在开学季，大学生会寻求以低价购入相应的教材，在一个学期的使用后重新以二手书的形式重新售出；在毕业季，毕业生的大部分书籍、生活用品、自行车都可以以二手交易的形式售出，否则也只能当做废品丢弃或回收。

综上，鉴于二手交易在交大师生群体中存在极大的需求空间，以及考虑到当下以二手交易群为代表的二手交易途径的种种弊端，通过现代化技术手段优化传统群发盲寻的二手群交易模式，，开发一款实用、易用的二手交易信息平台APP，克服二手交易群现有的使用问题，以满足交大师生对二手交易更便捷体验的需求，是十分有必要的。

## 范围

测试范围为Content，Message，SellInfo，File四个微服务，涉及服务之间的调用以及服务内部的处理，选择这四个微服务的理由为：

下表列出了制定*测试计划*所用的文档，并标明了文档的可用性：

测试策略

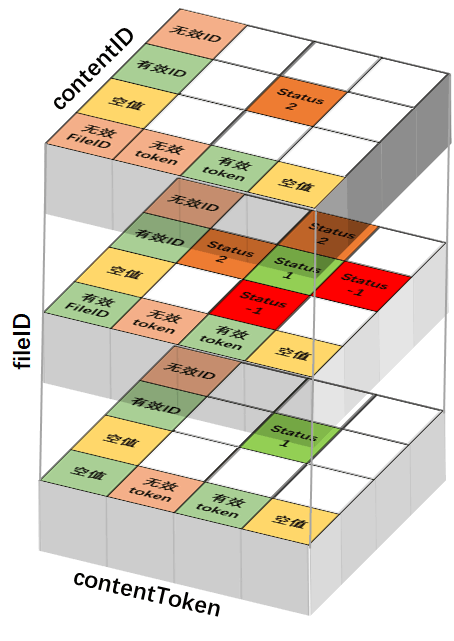
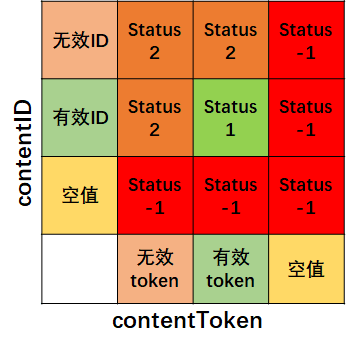
* Content

采用弱健壮等价类测试方法，因为Content中相应数据的内容和类型数据较多，大部分数据也没有明显的物理意义，最终决定使用弱健壮等价类的测试方法，用相对少的测试用例就能够进行比较完整的输入数据覆盖，解决了不能穷举测试的问题。

Content中共计6个待测接口，其中Check反复被调用，重要性程度高，使用强健壮等价类测试；Query接口十分简单，也可以当作强健壮等价类测试；由于弱一般等价类测试和强一般等价类测试均不考虑非法输入，不适用于此类接口，故其他四个接口Create, CreateTag, Delete, Update均使用弱健壮等价类测试。最终确定的测试用例数量为46个。

下面以Content.Check和Content.Delete为例分别说明强健壮等价类和弱健壮等价类的用例生成。如图所示：

* 1. Content.Check输入变量有两个contentID，contentToken，每个变量有一种有效值，两种无效值“值无效”和“空值”，采用强健壮等价类测试共需要（1+2）\*（1+2）=9个测试用例；
  2. Content.Delete输入变量有四个contentID，contentToken，fileID，前两者有一种有效值，两种无效值“值无效”和“空值”，fileID有两种有效值“正常值”和“空值”，一种无效值。综合以上，输出正常的用例应该有各个变量类别的最大数，即2个，检测错误的用例数本来应该是变量数3\*2=6个，但由于fileID只有一种无效条件，即不存在两种无效边界，所以实际上的错误用例数为5个。采用弱健壮等价类测试总计7个测试用例。



* Message

采用决策表来进行测试，因该模块中有关Message的操作都是依据输入的不同组合决定，代码层面的特点是if else语句较多，且判定条件都是有关于表单填写内容输入，因此选用决策表方法。最终确定的测试用例数量为19个。

基于决策表的决策方法是所有功能测试方法中最严谨的，因为它考虑到了测试用例整体的逻辑严密性。因为在实现过程中，与聊天消息相关的Message模块的逻辑相比而言更为复杂， 因为我们选择对Message模块进行基于决策表的测试方法。

此处我们以Message.Find(fromUser int32, toUser int32, way int32, limit uint32, offset uint32)为例，这个函数提供了从fromUser到toUser的聊天记录。

针对变量，我们设置了4个二值条件和一个多值条件：

1. C1：fromUser是否为0
2. C2：toUser是否为0
3. C3：fromUser与toUser是否相等
4. C4：way的值为1还是2
5. C5：limit的取值（）

与之对应，我们根据返回结果，设置4个不同但可以同时做的动作：

1. A1：返回-1
2. A2：返回1
3. A3：Limit修改为最大值
4. A4：将查询消息设置为已读

通过条件与动作的相互组合关系，我们可以给出相应的决策表，进而给出基于决策表的测试用例（具体见附件）。

* SellInfo

采用了健壮性边界值测试和弱健壮等价类测试结合的方式进行测试，对于一些变量例如商品ID，有明确物理或数值含义的，采用了健壮性边界值测试，其余变量则是组织了弱健壮等价类测试。最终确定的测试用例数量为40个。

相对于等价类测试，边界值测试的试用条件相对而言更加苛刻，它要求变量具有较为明确的物理或数值含义，但是本次测试软件的大部分接口的参数包含字符串类型、列表类型等难以确定边界的变量类型。

因此我们挑选SellInfo模块下的SellInfo.Update(sellInfoID int, status int)作为健壮性边界值测试的目标。变量sellInfoID是一个自增整型变量，从1开始自增，上边界是数据库中的sellInfo的数量；变量status表示待调整出售物品的目标状态，取值范围为{1, 2, 3, 4, 5}。两个变量均有明确的数值含义以及边界范围，适适用于边界值测试。

在测试中，我们在数据库中预存5个出售物品，则两个变量的具体情况如下表所示：

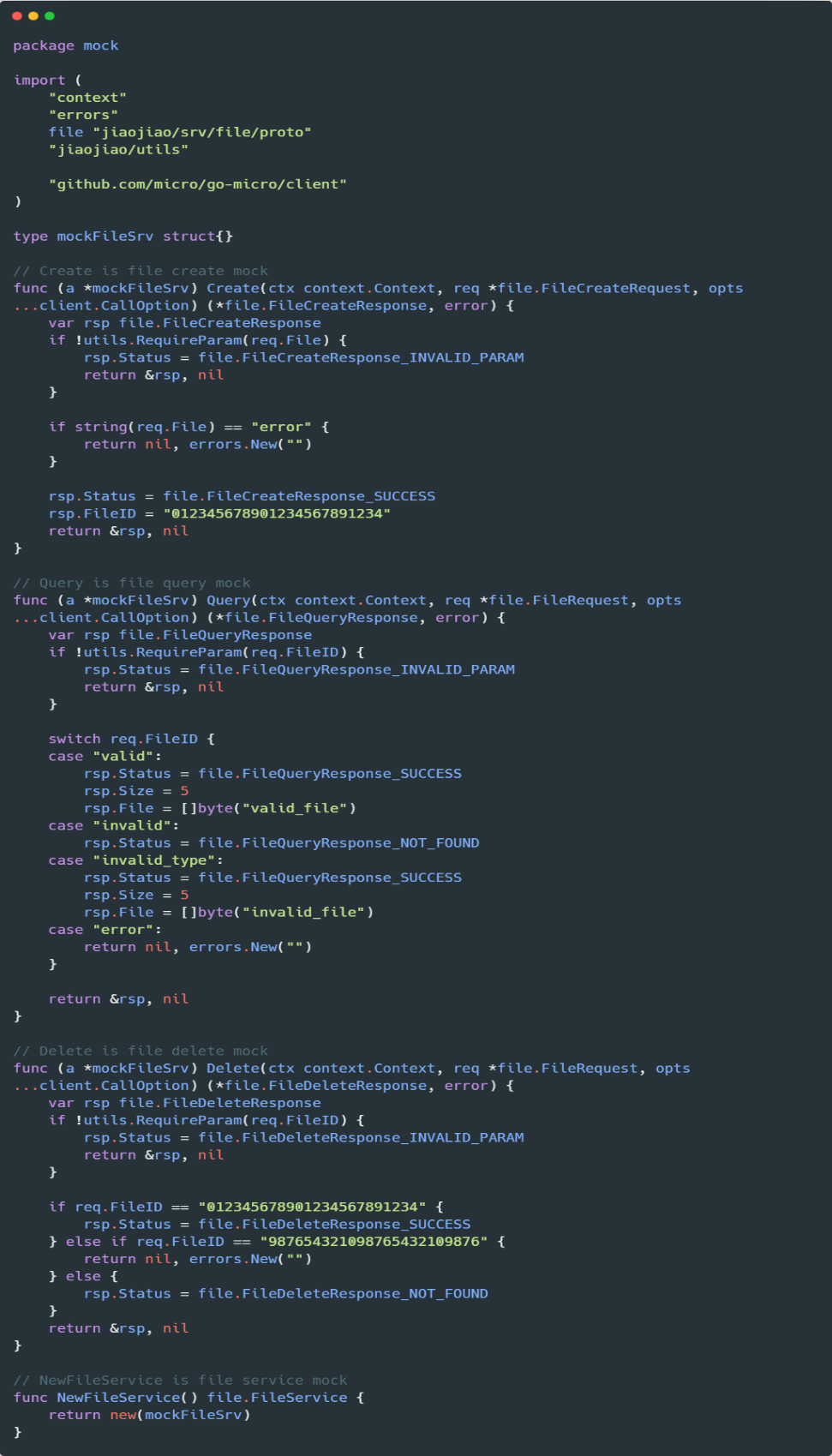
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 变量 | Min- | Min | Min+ | Valid | Max- | Max | Max+ |
| SellInfoID | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Status | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

因此，我们可以生成6n+1=13个测试用例（具体见附件）。

* File

采用弱健壮等价类测试方法，选择的理由与Content类似。最终确定的测试用例数量为10个。

由于各个模块之间存在通信，为便于测试，减少不必要的影响因素，我们在测试模块A时，会对有调用关系的模块采用mock的方式来参与测试，默认mock模块的完全正确。



File模块的mock

# 资源

* 服务器资源

使用4个4核8G内存服务器进行部署，1个4核8G服务器进行测试。

* 技能资源

goconvey, go-micro, Gin

## 角色

下表列出了在此项目的人员配备方面所作的各种假定。

|  |  |  |
| --- | --- | --- |
| 人力资源 | | |
| 角色 | 推荐的最少资源  （所分配的专职角色数量） | 具体职责或注释 |
| 测试经理，  测试项目经理 | 1 | 进行管理监督。  职责：  提供技术指导  获取适当的资源  提供管理报告 |
| 测试设计员 | 3 | 确定测试用例、确定测试用例的优先级并实施测试用例。  职责：  生成测试计划  生成测试模型  评估测试工作的有效性 |
| 测试员 | 3 | 执行测试。  职责：  执行测试  记录结果  从错误中恢复  记录变更请求 |
| 测试系统管理员 | 1 | 确保测试环境和资产得到管理和维护。  职责：  管理测试系统  授予和管理角色对测试系统的访问权 |
| 数据库管理员 | 1 | 确保测试数据（数据库）环境和资产得到管理和维护。  职责：  管理测试数据（数据库） |
| 设计员 | 3 | 确定并定义测试类的操作、属性和关联。  职责：  确定并定义测试类  确定并定义测试包 |
| 实施员 | 3 | 实施测试类和测试包，并对它们进行单元测试。  职责：  创建在测试模型中实施的测试类和测试包 |

系统

下表列出了测试项目所需的系统资源。

|  |  |
| --- | --- |
| 系统资源 | |
| 资源 | 名称/类型 |
| 数据库服务器 | Centos 7 |
| —网络或子网 | 202.120.40.8 |
| —服务器名服务器名 | jiaojiao-1，jiaojiao-2 |
| —数据库名 | jiaojiao |
| 客户端测试 PC | Windows 10 |
| —包括特殊的配置需求 | 无 |

项目里程碑

|  |  |  |  |
| --- | --- | --- | --- |
| **里程碑任务** | **工作量** | **开始日期** | **结束日期** |
| 制定测试计划 | 10 | 2020年4月4日 | 2020年4月7日 |
| 设计测试 | 20 | 2020年4月8日 | 2020年4月9日 |
| 实施测试 | 20 | 2020年4月9日 | 2020年4月15日 |
| 执行测试 | 10 | 2020年4月15日 | 2020年4月15日 |
| 评估测试 | 10 | 2020年4月16日 | 2020年4月19日 |

可交付工件

* 交付的文件如下：

1. 《测试计划》

2. 《需求规格说明书》

3. 《测试报告》

交付时间为2020年4月19日

* 交付的工具如下：

1. golang测试代码

交付时间为2020年4月19日

## 测试模型

详见测试报告

## 测试日志

详见测试报告

## 缺陷报告

详见测试报告