**黑we购需求调研规格说明书**

[1.引言 2](#_Toc470099873)

[1.1编写目的 2](#_Toc470099874)

[1.2定义 2](#_Toc470099875)

[1.3参考资料 2](#_Toc470099876)

[2.任务概述 3](#_Toc470099877)

[2.1目标 3](#_Toc470099878)

[2.1.1 项目开发背景 3](#_Toc470099879)

[2.1.2 项目目标 3](#_Toc470099880)

[2.1.3 作用范围 3](#_Toc470099881)

[2.2用户的特点 3](#_Toc470099882)

[2.3假定和约束 4](#_Toc470099883)

[3.需求规定 5](#_Toc470099884)

[3.1 系统功能模块 5](#_Toc470099885)

[3.2 各功能模块的详细描述 5](#_Toc470099886)

[3.2.1 账号模块 5](#_Toc470099887)

[3.2.2好友模块 6](#_Toc470099888)

[3.2.3 游戏模块 6](#_Toc470099889)

[3.2.4 商品模块 6](#_Toc470099890)

[3.2.5 订单模块 7](#_Toc470099891)

[3.3对性能的规定 7](#_Toc470099892)

[3.3.1 精度 7](#_Toc470099893)

[3.3.2 时间特性要求 7](#_Toc470099894)

[3.4数据管理能力要求 8](#_Toc470099895)

[3.5故障处理要求 8](#_Toc470099896)

[3.6其他专门要求 8](#_Toc470099897)

[4.可行性分析 9](#_Toc470099898)

[4.1技术可行性 9](#_Toc470099899)

[4.2经济可行性 9](#_Toc470099900)

[4.3政策可行性 10](#_Toc470099901)

[5.运行环境规定 10](#_Toc470099902)

[5.1设备 10](#_Toc470099903)

[5.2支持软件 10](#_Toc470099904)

[5.3接口 10](#_Toc470099905)

# 1.引言

## 1.1编写目的

编写该需求调研规格说明书，是为了供需求分析人员进行项目需求分析时使用，本文档可以作为项目验收标准之一，可以作为软件维护的参考资料。保证软件开发的质量、需求的完整与可追溯性，编写此文档。通过此文档，以保证业务需求提出者与需求分析人员、开发人员、测试人员及其也相关利益人对需求达成共识。

## 1.2定义

需求：用户解决问题或达到目标所需的条件或功能；系统或系统部件要满足合同、标准，规范或其它正式规定文档所需具有的条件或权能；一种反映前面两个所述条件或权能的文档说明。它包括功能性需求及非功能性需求，非功能性需求对设计和实现提出了限制，比如性能要求，质量标准，或者设计限制。

需求调研：需求调研对于一个应用软件开发来说，是一个系统开发的开始阶段，它的输出"软件需求分析报告"是设计阶段的输入，需求调研的质量对于一个应用软件来说，是一个极其重要的阶段，它的质量在一定程度上来说决定了一个软件的交付结果。怎样从客户中听取用户需求、分析用户需求就成为调研人员最重要的任务。

软件维护：软件维护(Software maintenance)是一个软件工程名词，是指在软件产品发布后，因修正错误、提升性能或其他属性而进行的软件修改。

## 1.3参考资料

[1]《软件工程》 作者:（英）萨默维尔　著，程成　等译 出版社:机械工业出版社 出版时间:2011年05月

[2]《软件项目管理》 作者:（英）休斯，（英）考特莱尔 著，廖彬山，周卫华 译 出版社:机械工业出版社 出版时间:2010年09月

[3] GB/T 11457 软件工程术语

[4] GB/T 12505 计算机软件配置管理计划规范

[5]《JavaWeb设计实用教程——基于Eclipse环境的应用开发》 作者:孔昊　主编出版社:机械工业出版社 出版时间:2012年04月

# 2.任务概述

## 2.1目标

### 2.1.1 项目开发背景

作为一个资深的游戏玩家，你是否有过这样的感受，身边的朋友跟你玩的不是一个游戏，不在一个频道，难以体会到一起玩游戏的快乐？游戏里的好友不在线，一个人怎么下副本？辛辛苦苦打造的装备线下交易居然被骗了？刚刚申请的账号才花费了不久就不想玩了，白白废了那么多钱？现在，是时候改变这种现状了！

随着游戏在年轻人甚至中年人之间受欢迎的程度不断提高，人们花在游戏上面的时间越来越多。现今各种直播平台、公众明星对电子游戏的宣传，使得游戏越来越深入人心，以游戏为娱乐的主要方式更加被人接受。游戏虽然只是虚拟数据，但游戏玩家在游戏中结识的人可以是真实的人，这些人也许能成为他的好朋友。越来越多的玩家快餐式的游戏方式，一时兴起想要下的区，玩的号很可能兴致会很快消失,而又碍于消费或者舍不得，需要一个交易平台来给那些想要卖号或者买号的玩家。物品交易不想换成游戏里的游戏币，想要换点零花钱。这些问题，都可以通过一个网站来改变，改变你的游戏交易方式，改变你的游戏行为。

### 2.1.2 项目目标

我们希望对那些热衷于在游戏中交朋友的用户提供一个交换信息和交流讨论的平台，用户可以添加自己喜欢的游戏，平台会提供推荐好友，用户可以添加好友，聊天，开黑，一应俱全。另外，我们也希望能为那些想要进行游戏物品，账号交易的用户提供一个可交易的平台，让玩家可以找到自己想要的大区账号，而且不玩的时候也可以将账号出售，无心理压力购买账号，形成帐号保值。玩家也可以将游戏中的游戏币换成零花钱花花。

### 2.1.3 作用范围

**系统适用单位：**各游戏官网平台

**系统用户类型：**在各种各样的游戏中遨游的玩家

## 2.2用户的特点

本软件的用户类型有两种，分别为游戏中交朋友一起玩游戏的人，游戏中想要交易物品，账号的人。

第一种用户的特点是喜欢交朋友，可能觉得游戏的乐趣就是在于一起玩，一个人的游戏没有意思。

第二种用户的特点是喜欢玩很多游戏，可能喜欢体验游戏的新，一旦游戏区服之类的步入后期就不喜欢玩，可能觉得游戏里的人越多越好，或者喜欢通过游戏赚点零花钱的，而游戏就是一种消遣的赚钱方式，不在于钱多少，赚钱可能有自豪感。

## 2.3假定和约束

软件开发小组提供相应的开发阶段的文档，用户提供相适应的行业标准，使软件开发与典型实例考核相符合。

操作员与用户要按照操作规程运行本系统，不得进行恶意破坏性操作。

用户必须提供相关运行软件有效的数据库接口标准，并在改动的过程中及时通知本软件开发商，以保证从中正确读取预决算参数，进行成本预算。

**开发期限:96天**

根据以往的项目开发经验，结合开发团队的具体情况，拟定了项目的开发进度安排：

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段 | 开始日期 | 结束日期 | 工作时长（工作日/天） |
| 开发任务及进度计划 | 2016-9-20 | 2016-9-25 | 5 |
| 可行性分析 | 2016-9-25 | 2016-9-30 | 5 |
| 需求分析 | 2016-9-30 | 2016-10-5 | 5 |
| 概要设计 | 2016-10-5 | 2016-10-10 | 5 |
| 模块设计 | 2016-10-10 | 2016-10-15 | 5 |
| 数据库设计 | 2016-10-15 | 2016-10-21 | 6 |
| 界面设计 | 2016-10-21 | 2016-10-31 | 10 |
| 账号管理模块 | 2016-10-31 | 2016-11-5 | 5 |
| 好友管理模块 | 2016-11-5 | 2016-11-10 | 5 |
| 游戏管理模块 | 2016-11-10 | 2016-11-15 | 5 |
| 商品管理模块 | 2016-11-15 | 2016-11-25 | 10 |
| 订单管理模块 | 2016-11-25 | 2016-11-30 | 5 |
| 项目整合 | 2016-11-30 | 2016-12-5 | 5 |
| 项目测试 | 2016-12-5 | 2016-12-15 | 10 |
| 项目试运行 | 2016-12-15 | 2016-12-20 | 5 |

表 1开发进度计划安排表

# 3.需求规定

## 3.1 系统功能模块

我们根据系统参与角色来划分了系统功能模块。

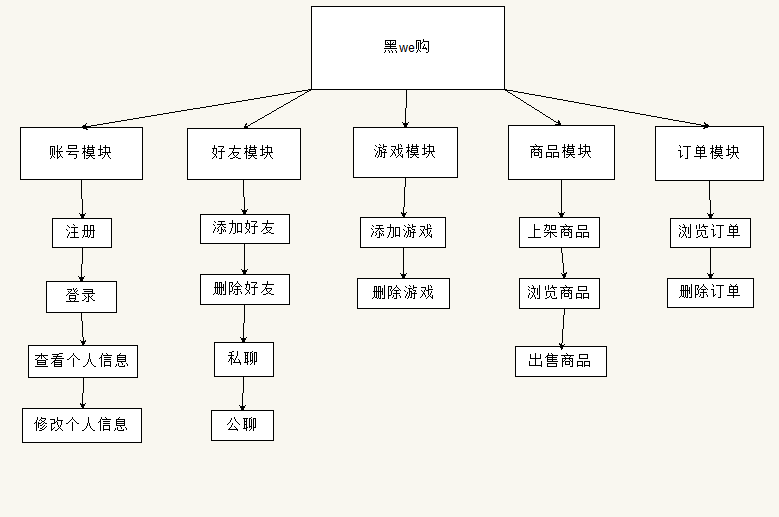


图 2 系统功能模块

## 3.2 各功能模块的详细描述

经过分析该网站的主要功能基本模块应该包括下面几个：

（1）账号模块

（2）好友模块

（3）游戏模块

（4）商品模块

（5）订单模块

### 3.2.1 账号模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 账号模块 |
| 用例描述 | 用户注册登录账号管理账号，查看信息 |
| 参与者 | 所有用户 |
| 前置条件 | 无 |
| 后置条件 | 登陆成功 |
| 正常时间流 | 1.新用户注册账号，注册成功跳转登录界面  2.已有账号的用户输入正确的用户名密码登陆  3.用户进入个人主页管理自己的个人信息 |
| 异常事件流 | 1.用户名密码错误  2.注册输入无效的信息 |

表 3 账号模块描述

### 3.2.2好友模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 好友模块 |
| 用例描述 | 用户添加好友和好友和非好友进行聊天 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入好友系统  2.查看自己的好友和系统推荐好友  3.选择添加和删除好友  4.选择公聊或者私聊 |
| 异常事件流 | 1.添加的好友不存在  2.聊天好友不存在 |

表 4 好友模块描述

### 3.2.3 游戏模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 游戏模块 |
| 用例描述 | 用户管理自己的游戏 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入游戏  2.查看自己的游戏  3.选择添加和删除游戏 |
| 异常事件流 | 无 |

表 5 游戏模块描述

### 3.2.4 商品模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 商品模块 |
| 用例描述 | 用户出售购买浏览商品 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入交易系统  2.浏览商品信息  3.选择上架或购买商品 |
| 异常事件流 | 1.商品不存在或已被购买  2. 金额不够支付 |

表 6 商品模块描述

### 3.2.5 订单模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 订单模块 |
| 用例描述 | 用户查看个人订单管理订单 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入订单系统  2.查看自己的订单  3.选择删除订单 |
| 异常事件流 | 无 |

表 7 订单模块描述

## 3.3对性能的规定

### 3.3.1 精度

系统数据的精度要求：

日期格式要求精确到秒；

数据库的所有记录实时更新；

用户接受发送信息和购买出售实时性，不超过5秒完成数据传输和数据库更新；

数据字段由系统开况自行设置；

用户金额精确到百分位。

### 3.3.2 时间特性要求

1. 呈现时间：网站呈现时间跟浏览器和操作系统相关，故不作描述。

2. 数据的传输：完成数据的传输和转换能够达到及时响应，包括各种聊天信息以及表单的提交数据库，不超过5秒。当然，数据的传输时间与互联网带宽相关。

3. 系统处理时间：系统得到请求后对请求进行处理并将结果返回。将系统的处理请求的时间缩到最短暂，采用JAVA语言编写网站后端，不超过5秒。

## 3.4数据管理能力要求

1. 定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储；

2. 对几个月未登录的账户发送通知，采取长期不登陆注销该账户，清除该账户的所有数据，减轻数据库压力；

3. 数据库的安全性就是指数据库中数据的保护措施，一般包括的登陆的身份验证管理、数据库的使用权限管理和数据库中对象的使用权限管理三种安全性保护措施，更改数据库不同数据的安全级别；

4. 对数据库的事物处理多次进行检查和测试，确保不会造成错误。

## 3.5故障处理要求

系统可能遇到的方面：

1．网站同时在线的人数过多，同时进行同一操作时造成网站卡死，系统崩溃；

2．服务器数据库故障，造成数据错误或者无法正常使用；

3. 网站处理异常，存在未知BUG。

针对以上几以下解决方案：

1．定期维护，防止病毒入侵。

2．加强服务器以及网站的性能测试，增强网站的安全性；

3. 一旦出现故障，尽快修复保证网站正常运行；

4. 加大软件测试的强度，尽量减少网站的出错；

5. 定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

## 3.6其他专门要求

本网站用MVC架构模式。MVC的目的是增加代码的重用率，减少数据表达，数据描述和应用操作的耦合度。同时也使得软件可维护性可修复性，可扩展性，灵活性以及封装性大大提高。本网站中将数据库，模型层，处理层，web层和用户界面层分开。后台的开发语言是JAVA程序设计语言，前台的开发语言主要是HTML，CSS，JAVASCRIPT。本网站是B/S系统，运行环境是浏览器。项目完成后要提供用户使用手册，要求用户清楚户籍业务流程，使用之前要严格测试。

# 4.可行性分析

## 4.1技术可行性

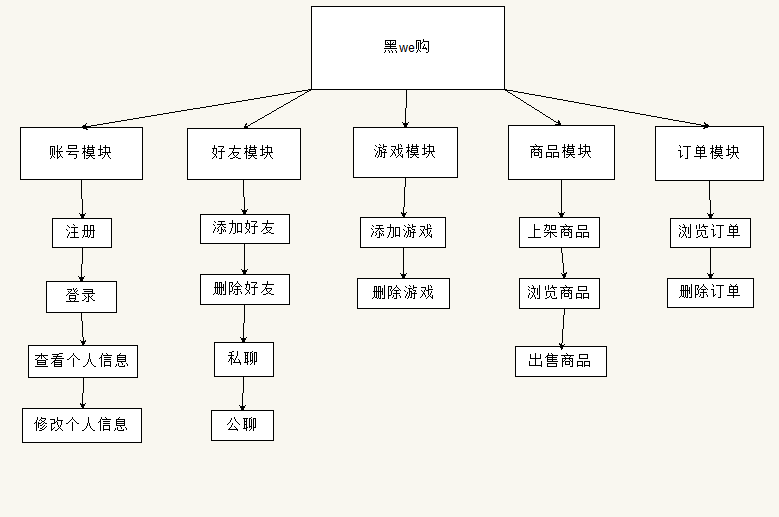


图 8 模块图

我们要做的黑we购网站，基于JAVAEE和Apache Tomcat服务器。综合各负责人调查所言，该项目需要很多方面的技术支持。最直观影响用户体验的是用户界面，美工需要有一定美学基础，UI 设计基础，一个简洁美观的界面在短期内是可以制作出来的。前端开发中,网站的界面设计主要是HTML，CSS，JAVASCRIPT等知识。当然如果界面需要美观效果不同，可能一些基本工具是需要的。后端的语言主要是Java，像JAVAEE高级部分的内容和MVC模式和javaweb开发中的一些框架也是需要学习的内容。另外像数据库的部分，高级的数据库部分像事物这种保护用户安全性的知识，另外就是提供交易功能的话，像热门的支付宝或者银联的接口API也是需要我们去研究的。本网站没有特别难的技术难点，像用户界面之类的技术已经很成熟了，我们只需要去学会调用别人已经写好的成品。比如好友模块中的聊天功能的实现，基于socket的编程，这就需要我们对http协议的了解。基于信息高速发展的今天,各种技术网课爆满屏幕,资料丰富,技术已不是最大的难题, 关键是和时间竞争。

## 4.2经济可行性

本网站涉及技术难点不是很多，经济方面的可行性主要表现在不需要额外支付其他费用，支付宝接口和银联接口可能需要按交易额一定比例支付费用，当然这都不是初期应该想的内容。另外，云端服务器搭建的费用，现在云服务器也比较便宜。

## 4.3政策可行性

随着游戏的崛起，各式各样的游戏平台也应运而生，在法律方面和政策方面没有限制。

# 5.运行环境规定

## 5.1设备

能够安装基础浏览器的设备。

## 5.2支持软件

本软件采用的是B/S结构，对软件并没有特殊要求，客户端是浏览器。

## 5.3接口

无

黑we购模块设计方案

[1.引言 12](#_Toc470100028)

[1.1 编写目的 12](#_Toc470100029)

[1.2 定义 12](#_Toc470100031)

[1.3 参考资料 12](#_Toc470100032)

[2.系统程序模块图 14](#_Toc470100033)

[3.模块的标准流程 15](#_Toc470100034)

[3.1 账号模块 15](#_Toc470100035)

[3.2 好友模块 16](#_Toc470100036)

[3.3 游戏模块 17](#_Toc470100037)

[3.4 商品模块 18](#_Toc470100038)

[3.5 订单模块 19](#_Toc470100039)

[4.功能需求与程序模块的关系 19](#_Toc470100040)

[5.功能模块构件图 20](#_Toc470100041)

[5.1账号管理子系统构件图描述 20](#_Toc470100042)

[5.2 好友子系统构件图描述 21](#_Toc470100043)

[5.3 游戏系统构件图描述 22](#_Toc470100044)

[5.4 商品子系统构件图描述 22](#_Toc470100045)

[5.5 订单子系统构件图描述 23](#_Toc470100046)

# 1.引言

## 1.1 编写目的

本文档是本次系统的模块设计说明，旨在更加全面、清晰、准确并且文档化的对本系统的模块设计方案作出说明。本文档的预期读者是系统的软件开发商。

①设计系统方案

软件结构设计，首先要把复杂的功能进一步分解成简单的功能，遵循模块划分独立性原则（即做到模块功能单一，模块与外部联系很弱，仅有数据联系），使划分过的模块的功能对大多数程序员而言都是易懂的。功能的分解导致对数据流图的进一步细化，并选用相应图形工具来描述。

②选取一组合理的方案分析员在通过问题定义、可行性研究和需求分析后，产生了一系列可供选择的方案，从中选取低成本、中成本、高成本三种方案，必要时再进一步征求用户意见。并准备好系统流程图，系统的物理元素清单，成本效益分析，实现系统的进度计划。

③推荐最佳实施方案综合分析对比各种合理方案的利弊，推荐一个最佳的方案，并为最佳方案制定详细的实现计划。

## 1.2 定义

模块：在程序设计中，为完成某一功能所需的一段程序或子程序；或指能由编译程序、装配程序等处理的独立程序单位；或指大型软件系统的一部分。

模块的耦合度：模块的藕合度(Coupling)是一个软件结构内不同模块之间联系程度的度量，软件设计应追求尽可能隅合松散的系统。

三层架构：三层架构(3-tier architecture) 通常意义上的三层架构就是将整个业务应用划分为:表现层(Presentation layer)、业务逻辑层(Business Logic Layer)、数据访问层(Data access layer)。区分层次的目的即为了"高内聚低耦合"的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。微软推荐的分层式结构一般分为三层，从下至上分别为:数据访问层、业务逻辑层(又或称为领域层)、表示层。

表现层：俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。

业务逻辑层：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。

数据层：该层所做事务直接操作数据库，针对数据的增添、删除、修改、查找等。

## 1.3 参考资料

[1]《软件工程》 作者:（英）萨默维尔　著，程成　等译 出版社:机械工业出版社 出版时间:2011年05月

[2]《软件项目管理》 作者:（英）休斯，（英）考特莱尔 著，廖彬山，周卫华 译 出版社:机械工业出版社 出版时间:2010年09月

[3] GB/T 11457 软件工程术语

[4] GB/T 12505 计算机软件配置管理计划规范

[5]《JavaWeb设计实用教程——基于Eclipse环境的应用开发》 作者:孔昊　主编出版社:机械工业出版社 出版时间:2012年04月

# 2.系统程序模块图

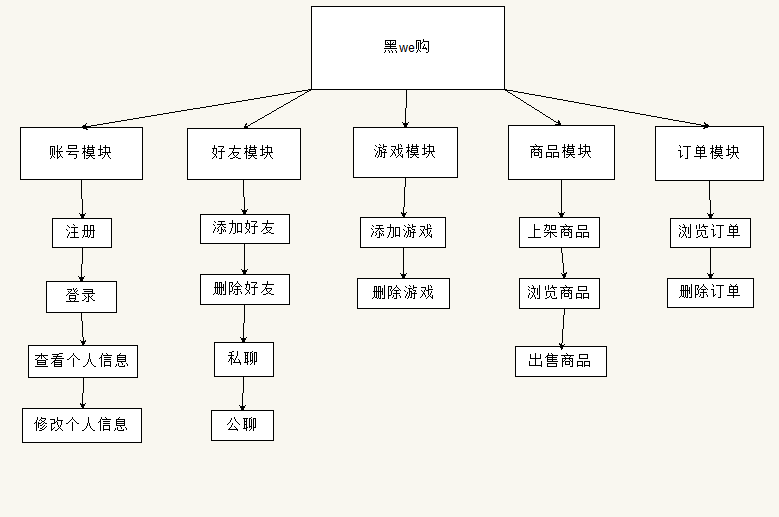


图 9 系统程序模块图

# 3.模块的标准流程

## 3.1 账号模块

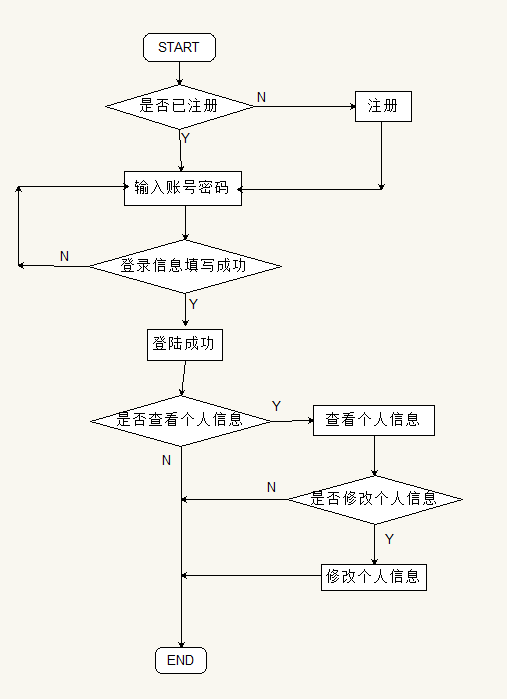


图 10 账号模块标准流程

## 3.2 好友模块

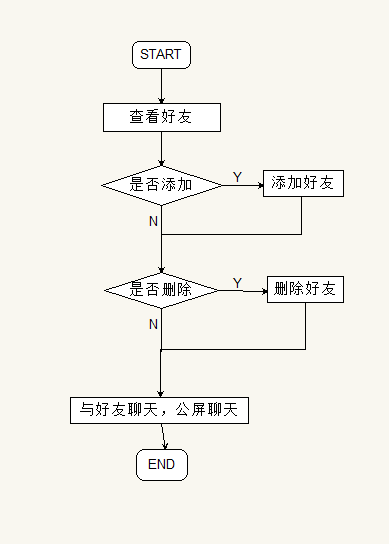


图 11 好友模块标准流程

## 3.3 游戏模块

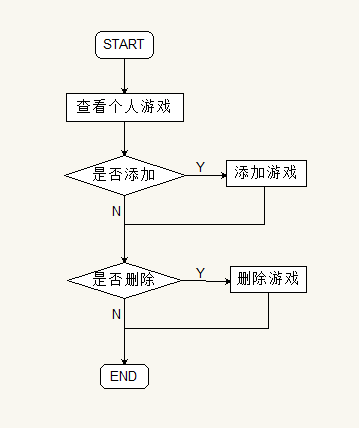


图 12 游戏模块标准流程

## 3.4 商品模块

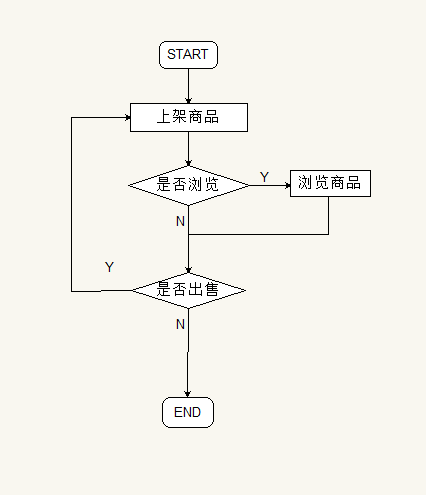


图 13 商品模块标准流程

## 3.5 订单模块

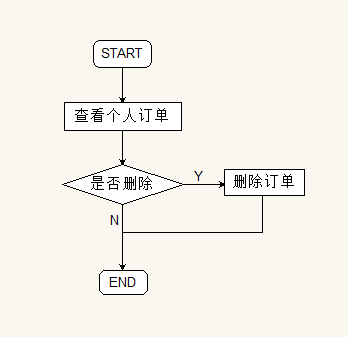


图 14 订单模块标准流程

# 4.功能需求与程序模块的关系

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能需求 | 账号管理子系统 | 好友子系统 | 游戏子系统 | 商品子系统 | 订单子系统 |
| 注册 | √ |  |  |  |  |
| 登录 | √ |  |  |  |  |
| 查看个人信息 | √ |  |  |  |  |
| 修改个人信息 | √ |  |  |  |  |
| 添加好友 |  | √ |  |  |  |
| 删除好友 |  | √ |  |  |  |
| 私聊 |  | √ |  |  |  |
| 公聊 |  | √ |  |  |  |
| 添加游戏 |  |  | √ |  |  |
| 删除游戏 |  |  | √ |  |  |
| 上架商品 |  |  |  | √ |  |
| 浏览商品 |  |  |  | √ |  |
| 购买商品 |  |  |  | √ |  |
| 浏览订单 |  |  |  |  | √ |
| 删除订单 |  |  |  |  | √ |

表 15 功能需求与程序模块的关系

# 5.功能模块构件图

## 5.1账号管理子系统构件图描述

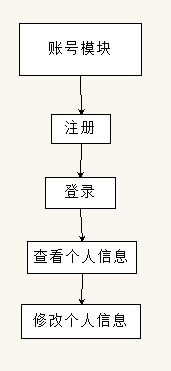


图 16账号管理子系统构件图

|  |  |
| --- | --- |
| 构件名 | 构件描述 |
| 注册 | 用户根据页面提示注册个人账号 |
| 登录 | 用户根据页面提示登录个人账号，进入主界面后才能进行其他个人账号操作 |
| 查看个人信息 | 用户登录界面后查看自己的个人信息 |
| 修改个人信息 | 用户查看个人信息后如果不满意可以对个人信息进行修改 |

表 17账号管理子系统构件图描述

## 5.2 好友子系统构件图描述

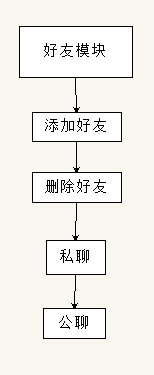


图 18好友子系统构件图

|  |  |
| --- | --- |
| 构件名 | 构件描述 |
| 添加好友 | 用户根据提供的推荐好友或者自己搜索添加个人好友 |
| 删除好友 | 用户在个人好友列表里根据个人意愿删除好友 |
| 私聊 | 用户选择自己的好友或者非好友发送个人，只有自己和对方能知道 |
| 公聊 | 用户选择公平聊天，对一个区域的所有用户发送，当前区域的所有用户均能看到 |

表 19好友子系统构件图描述

## 5.3 游戏系统构件图描述

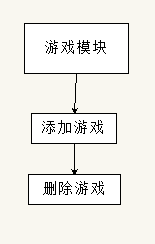


图 20游戏系统构件图

|  |  |
| --- | --- |
| 构件名 | 构件描述 |
| 添加游戏 | 用户根据提供的推荐游戏或者自己搜索添加个人游戏 |
| 删除游戏 | 用户在个人游戏列表里根据个人意愿删除游戏 |

表 21游戏系统构件图描述

## 5.4 商品子系统构件图描述

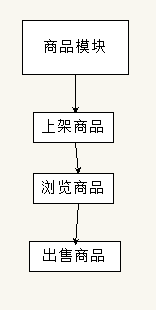


图 22商品子系统构件图

|  |  |
| --- | --- |
| 构件名 | 构件描述 |
| 上架商品 | 用户根据个人意愿选择出售游戏里的物品或者账号 |
| 浏览商品 | 用户选择查看商品列表，那些被人出售的物品或者账号 |
| 购买商品 | 用户浏览商品具体信息后，选择购买心仪的物品或者账号 |

表 23商品子系统构件图描述

## 5.5 订单子系统构件图描述

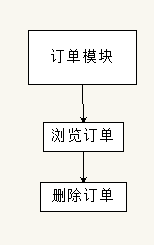


图 24订单子系统构件图

|  |  |
| --- | --- |
| 构件名 | 构件描述 |
| 浏览订单 | 用户选择查看订单列表包括出售中的商品订单，已经出售的商品订单和购买过的商品订单 |
| 删除订单 | 用户在个人订单列表里根据个人意愿删除订单 |

表 25订单子系统构件图描述

黑we购数据库设计文档

[1. 引言 25](#_Toc470100206)

[1.1 编写目的 25](#_Toc470100207)

[1.2 定义 25](#_Toc470100208)

[1.3 参考资料 25](#_Toc470100209)

[2. 数据库设计 26](#_Toc470100210)

[2.1 数据库环境说明 26](#_Toc470100211)

[2.2 数据库的命名规则 26](#_Toc470100212)

[2.3 逻辑设计 26](#_Toc470100213)

[2.3.1 实体图 26](#_Toc470100214)

[2.3.2 E-R图 29](#_Toc470100215)

[2.4 物理设计 29](#_Toc470100216)

[2.4.1 表汇总 29](#_Toc470100217)

[2.4.2表的详细信息 30](#_Toc470100218)

[2.5 安全性设计 36](#_Toc470100219)

[2.5.1 防止用户直接操作数据库的方法 36](#_Toc470100220)

[2.5.2 角色与权限 37](#_Toc470100221)

[2.6 数据库管理与维护说明 37](#_Toc470100222)

# 1. 引言

## 1.1 编写目的

本文档是黑we购系统概要设计文档的组成部分，编写数据库设计文档的目的是：明确数据库的表名、字段名等数据信息，用来指导后期的数据库脚本的开发，本文档遵循《数据库设计和开发规范》。本文档的读者对象是需求人员、系统设计人员、开发人员、测试人员。

## 1.2 定义

## 1.3 参考资料

[1]《软件工程》 作者:（英）萨默维尔　著，程成　等译 出版社:机械工业出版社 出版时间:2011年05月

[2] GB/T 11457 软件工程术语

[3] GB/T 12505 计算机软件配置管理计划规范

[4]《JavaWeb设计实用教程——基于Eclipse环境的应用开发》 作者:孔昊　主编出版社:机械工业出版社 出版时间:2012年04月

# 2. 数据库设计

## 2.1 数据库环境说明

数据库实例：HereWeGo

数据库系统：MySQL 5.5

数据库部署环境：

软件环境：无

硬件环境：windows 2007

数据库设计工具：powerDesigner、SQLyog

数据库存放位置：F:\MySQL

说明：本实例是用于黑we购网站的数据存储。

## 2.2 数据库的命名规则

（1）本数据库设计完全按照《数据库设计规范》命名。

（2） 不规范命名：myOrder表，Order为MySQL数据库关键字，无法命名为Order。

## 2.3 逻辑设计

## 2.3.1 实体图

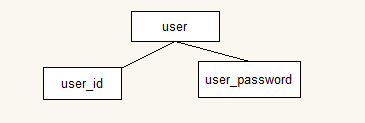


图 26 user实体图

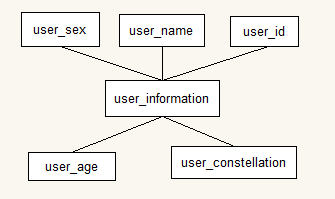


图 27 user\_information实体图

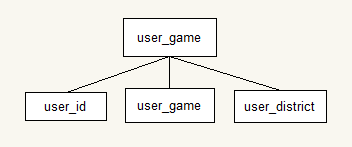


图 28 user\_game实体图

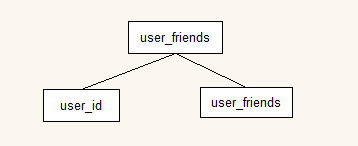


图 29 user\_friends实体图

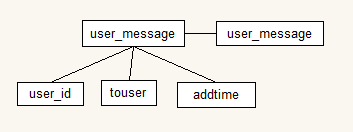


图 30 user\_message实体图

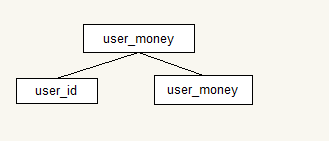


图 31 user\_money实体图

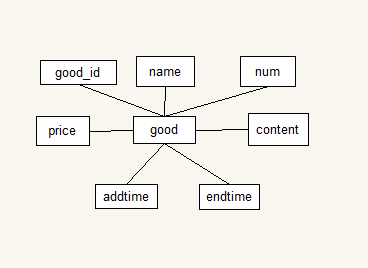


图 32 good实体图

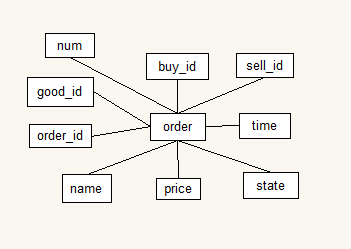


图 33 myorder实体图

### 2.3.2 E-R图

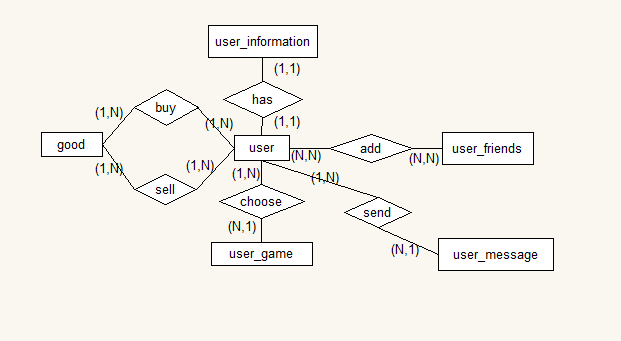


图 34 黑we购数据库E-R图

上图是数据库中每张表之间的关系：

每个用户都有一个属于他自己的信息；

每个用户都可以发送多条信息；

每个用户都可以选择多个游戏；

每个用户可以选择和多个用户成为好友；

每个用户可以上架多个商品；

每个商品只能被一个用户购买。

## 2.4 物理设计

### 2.4.1 表汇总

|  |  |
| --- | --- |
| 表名 | 功能说明 |
| user | 用户表，用于存储用户用户名和密码信息。 |
| user\_information | 用户信息表，用于存储用户个人信息，包括昵称，性别等信息 |
| user\_game | 用户游戏表，用于存储用户个人游戏。 |
| user\_friends | 用户好友表，用于存储用户与用户之间的好友关系。 |
| user\_money | 用户账户表，用于存储当前用户所属账户账户信息。 |
| user\_message | 用户消息表，用于存储用户与用户之间的聊天具体信息。 |
| good | 商品表，用户存储用户上架商品的信息。 |
| myorder | 订单表，用于存储用户订单信息，包括未出售商品的订单，已出售商品的订单， |

表 35 表汇总表

### 2.4.2表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id | | | | | |
| 其他排序字段 | | user\_password | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [Y] | [高] | 无 | 主键 |
| 2 | [user\_password] | [varchar] | [N] | [N] | [高] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `user` (  `user\_id` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_password` VARCHAR(16) CHARACTER SET utf8 NOT NULL,  PRIMARY KEY (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [82] | | | | | |
| 补充说明 | | 用于存储用户用户名和密码信息。 | | | | | |

表 36 user表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user\_information | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id | | | | | |
| 其他排序字段 | | user\_name、user\_sex、user\_age、user\_constellation | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [Y] | [高] | 无 | 主键、user表外键 |
| 2 | [user\_ name] | [varchar] | [N] | [N] | [高] | 无 | 无 |
| 3 | [user\_ sex] | [varchar] | [Y] | [N] | [低] | 无 | 无 |
| 4 | [user\_ age] | [int] | [Y] | [N] | [低] | 无 | 无 |
| 5 | [user\_constellation] | [varchar] | [Y] | [N] | [低] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `user\_information` (  `user\_id` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_name` VARCHAR(16) CHARACTER SET utf8 DEFAULT NULL,  `user\_sex` VARCHAR(20) CHARACTER SET utf8 DEFAULT NULL,  `user\_age` INT(20) DEFAULT NULL,  `user\_constellation` VARCHAR(20) CHARACTER SET utf8 DEFAULT NULL,  PRIMARY KEY (`user\_id`),  CONSTRAINT `foreign(user\_id)` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [82] | | | | | |
| 补充说明 | | 用于存储用户个人信息，包括昵称，性别等信息 | | | | | |

表 37 user\_information表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user\_game | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id | | | | | |
| 其他排序字段 | | user\_ game、user\_ district | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [Y] | [高] | 无 | 主键、user表外键 |
| 2 | [user\_ game] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| 3 | [user\_district] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `user\_game` (  `user\_id` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_game` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_district` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  PRIMARY KEY (`user\_id`,`user\_game`,`user\_district`),  CONSTRAINT `user\_game\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [123] | | | | | |
| 补充说明 | | 用于存储用户个人游戏。 | | | | | |

表 38 user\_game表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user\_friends | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id 、user\_friend | | | | | |
| 其他排序字段 | | 无 | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [N] | [高] | 无 | 主键、user表外键 |
| 2 | [user\_ friend] | [varchar] | [N] | [N] | [低] | 无 | 主键、user表外键 |
| MySQL脚本 | | CREATE TABLE `user\_friends` (  `user\_id` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_friend` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  PRIMARY KEY (`user\_id`,`user\_friend`),  KEY `user\_friend` (`user\_friend`),  CONSTRAINT `user\_friends\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),  CONSTRAINT `user\_friends\_ibfk\_2` FOREIGN KEY (`user\_friend`) REFERENCES `user` (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [52] | | | | | |
| 补充说明 | | 用于存储用户与用户之间的好友关系。 | | | | | |

表 39 user\_friends表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user\_money | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id | | | | | |
| 其他排序字段 | | user\_money | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [Y] | [高] | 无 | 主键、user表外键 |
| 2 | [user\_ money] | [float] | [N] | [N] | [低] | 0 | 无 |
| MySQL脚本 | | CREATE TABLE `user\_money` (  `user\_id` VARCHAR(20) COLLATE utf8\_unicode\_ci NOT NULL,  `user\_money` FLOAT DEFAULT '0',  PRIMARY KEY (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [82] | | | | | |
| 补充说明 | | 用于存储当前用户所属账户账户信息。 | | | | | |

表 40 user\_money表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | user\_message | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | user\_id | | | | | |
| 其他排序字段 | | touser、user\_message、user\_add\_time | | | | | |
| 索引字段 | | user\_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [user\_id] | [varchar] | [N] | [Y] | [高] | 无 | 主键 |
| 2 | [touser] | [varchar] | [N] | [N] | [低] | 无 | user表外键 |
| 3 | [user\_message] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| 4 | [user\_add\_time] | [datetime] | [N] | [N] | [低] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `user\_message` (  `user\_id` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `touser` VARCHAR(20) CHARACTER SET utf8 NOT NULL,  `user\_message` VARCHAR(128) CHARACTER SET utf8 NOT NULL,  `user\_add\_time` DATETIME NOT NULL,  PRIMARY KEY (`user\_id`,`user\_message`,`user\_add\_time`,`touser`),  KEY `touser` (`touser`),  CONSTRAINT `user\_message\_ibfk\_1` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`),  CONSTRAINT `user\_message\_ibfk\_2` FOREIGN KEY (`touser`) REFERENCES `user` (`user\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [32] | | | | | |
| 补充说明 | | 用于存储用户与用户之间的聊天具体信息。 | | | | | |

表 41 user\_message表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | good | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | good\_id | | | | | |
| 其他排序字段 | | name 、num、price、content、addtime、endtime | | | | | |
| 索引字段 | | good \_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [good\_id] | [int] | [N] | [Y] | [高] | 无 | 主键 |
| 2 | [name] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| 3 | [num] | [int] | [N] | [N] | [低] | 无 | 无 |
| 4 | [price] | [float] | [N] | [N] | [低] | 无 | 无 |
| 5 | [content] | [varchar] | [Y] | [N] | [低] | 无 | 无 |
| 6 | [addtime] | [datetime] | [N] | [N] | [低] | 无 | 无 |
| 7 | [endtime] | [datetime] | [N] | [N] | [低] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `good` (  `good\_id` INT(11) NOT NULL,  `name` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `num` INT(11) DEFAULT NULL,  `price` FLOAT DEFAULT NULL,  `content` VARCHAR(100) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `addtime` DATETIME DEFAULT NULL,  `endtime` DATETIME DEFAULT NULL,  PRIMARY KEY (`good\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [47] | | | | | |
| 补充说明 | | 用于存储用户上架商品信息。 | | | | | |

表 42 good表的详细信息

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 表名 | | myorder | | | | | |
| 数据库用户 | | herewego | | | | | |
| 主键 | | order\_id | | | | | |
| 其他排序字段 | | good\_id、name 、num、buy\_id、sell\_id、price、 time、state | | | | | |
| 索引字段 | | order \_id | | | | | |
| 序号 | 字段名称 | 数据类型（精度范围） | 允许为空Y/N | 唯一Y/N | 区别度 | 默认值 | 约束条件/说明 |
| 1 | [order\_id] | [int] | [N] | [Y] | [高] | 无 | 主键 |
| 2 | [good\_id] | [int] | [N] | [Y] | [高] | 无 | good表外键 |
| 3 | [name] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| 4 | [num] | [int] | [N] | [N] | [低] | 无 | 无 |
| 5 | [buy\_id] | [varchar] | [N] | [N] | [低] | 无 | user表外键 |
| 6 | [sell\_id] | [varchar] | [N] | [N] | [低] | 无 | user表外键 |
| 7 | [price] | [float] | [N] | [N] | [低] | 无 | 无 |
| 8 | [time] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| 9 | [state] | [varchar] | [N] | [N] | [低] | 无 | 无 |
| MySQL脚本 | | CREATE TABLE `myorder` (  `order\_id` INT(11) NOT NULL,  `good\_id` INT(11) NOT NULL,  `name` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `num` INT(11) DEFAULT NULL,  `buy\_id` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `sell\_id` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `price` FLOAT DEFAULT NULL,  `time` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  `state` VARCHAR(20) COLLATE utf8\_unicode\_ci DEFAULT NULL,  PRIMARY KEY (`order\_id`),  UNIQUE KEY `1` (`good\_id`),  KEY `2` (`buy\_id`),  KEY `3` (`sell\_id`)  ) ENGINE=INNODB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci; | | | | | |
| 记录数 | | [66] | | | | | |
| 补充说明 | | 用于存储用户订单信息，包括未出售商品的订单，已出售商品的订单， | | | | | |

表 43 myorder表的详细信息

## 2.5 安全性设计

（1）应用级的用户帐号密码不能与数据库相同，防止用户直接操作数据库。

（2）用户只能用帐号登陆到应用软件，通过应用软件访问数据库，而没有其它途径操作数据库。

### 2.5.1 防止用户直接操作数据库的方法

用户只能用帐号登陆到应用软件，通过应用软件访问数据库，而没有其他途径操作数据库。

（1）普通用户通常没有修改服务器端数据库的权限，只有管理员才拥有对数据库操作的权限。

（2）普通用户在申请注册或修改信息时，若请求通过，则数据库会自动添加或修改数据。

（3）普通用户在修改个人信息时，若请求通过，则数据库会自动添加或修改数据。

（4）普通用户在添加删除好友和游戏时，若请求通过，则数据库会自动添加或修改数据。

（5）普通用户在操作网站上的选项进行商品和订单修改时，若请求通过，则数据库会自动添加或修改数据。

（6）普通用户无法访问数据库中除了自己之外的其他数据。

### 2.5.2 角色与权限

|  |  |  |
| --- | --- | --- |
| 角色 | 可以访问的表与列 | 操作权限 |
| 用户 | user | 增加 |
| user\_information | 增加，修改 |
| user\_game | 增加，修改 |
| user\_friends | 增加，删除 |
| user\_money | 增加 |
| user\_message | 增加 |
| good | 增加，删除 |
| myorder | 增加，删除 |
| 管理员 | 所有 | 所有 |

表 44 数据库角色与权限表

## 2.6 数据库管理与维护说明

（1）数据库不因庞大的数据而造成不能处理或者数据缺失的情况。

（2）数据资源管理系统标准统一。

（3）要求能进行数据库的建立、数据库的调优、数据库的重组、数据库的重构、数据库的安全管控、报错问题的分析和汇总和处理、数据库数据的日常备份。

（4）定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

（5）当数据库的内容发生改变时，将会自动备份数据。

（6）数据库的安全性就是指数据库中数据的保护措施，一般包括的登陆的身份验证管理、数据库的使用权限管理和数据库中对象的使用权限管理三种安全性保护措施，更改数据库不同数据的安全级别；

（7）对数据库的事物处理多次进行检查和测试，确保不会造成错误。

黑we购软件概要设计方案

[1.引言 40](#_Toc470100362)

[1.1编写目的 40](#_Toc470100363)

[1.2定义 40](#_Toc470100364)

[1.3参考资料 40](#_Toc470100365)

[2.总体设计 42](#_Toc470100366)

[2.1需求规定 42](#_Toc470100367)

[2.1.1系统功能 42](#_Toc470100368)

[2.1.2系统性能 42](#_Toc470100369)

[2.1.3输入输出要求 43](#_Toc470100370)

[2.1.4数据管理能力要求 43](#_Toc470100371)

[2.1.5故障处理要求 44](#_Toc470100372)

[2.1.6其他专门要求 44](#_Toc470100373)

[2.2运行环境 44](#_Toc470100374)

[2.2.1 硬件环境 44](#_Toc470100375)

[2.2.2 系统运行环境 44](#_Toc470100376)

[2.2.3接口 44](#_Toc470100377)

[2.2.4控制 44](#_Toc470100378)

[2.3基本设计概念和处理流程 45](#_Toc470100379)

[2.3.1总体流程描述 45](#_Toc470100380)

[2.3.2模块的标准流程 45](#_Toc470100381)

[2.3.3安全控制 45](#_Toc470100382)

[2.3.4权限控制 45](#_Toc470100383)

[2.3.5工作流管理 46](#_Toc470100384)

[2.3.6软件质量约束 46](#_Toc470100385)

[2.3.7设计策略 46](#_Toc470100386)

[2.3.8主要数据结构和算法 47](#_Toc470100387)

[2.4系统程序模块结构 54](#_Toc470100388)

[2.5功能需求与程序模块的关系 54](#_Toc470100389)

[2.6人工处理过程 55](#_Toc470100390)

[2.7尚未解决的问题 55](#_Toc470100391)

[3.接口设计 56](#_Toc470100392)

[3.1用户接口 56](#_Toc470100393)

[3.2外部接口 56](#_Toc470100394)

[3.2.1软件接口 56](#_Toc470100395)

[3.2.2 硬件接口 56](#_Toc470100396)

[3.3内部接口 56](#_Toc470100397)

[4.运行控制 57](#_Toc470100398)

[4.1运行模块组合 57](#_Toc470100399)

[4.2运行控制 61](#_Toc470100400)

[4.3运行时间 61](#_Toc470100401)

[5.系统出错处理设计 61](#_Toc470100402)

[5.1出错信息 61](#_Toc470100403)

[5.2补救措施 61](#_Toc470100404)

[5.3系统维护设计 61](#_Toc470100405)

# 1.引言

## 1.1编写目的

本详细设计说明书是在前期的需求说明书和概要设计的基础上编写，黑we购软件概要设计方案说明书的读者对象为项目负责人、开发黑we购软件的主体人员以及系统二次开发人员。

编写此文档的目的是：

（1）详细、准确和全面地定义黑we购软件的概要设计，以利于指导该软件后续的开发工作；

（2）本文档所描述的概要设计作为该项目最终验收的标准和依据；

（3）给工作人员提供一定的维护依据，方便在以后的维修工作中提供方便。

## 1.2定义

三层架构：三层架构(3-tier architecture) 通常意义上的三层架构就是将整个业务应用划分为:表现层(Presentation layer)、业务逻辑层(Business Logic Layer)、数据访问层(Data access layer)。区分层次的目的即为了"高内聚低耦合"的思想。在软件体系架构设计中，分层式结构是最常见，也是最重要的一种结构。微软推荐的分层式结构一般分为三层，从下至上分别为:数据访问层、业务逻辑层(又或称为领域层)、表示层。

表现层：俗讲就是展现给用户的界面，即用户在使用一个系统的时候他的所见所得。

业务逻辑层：针对具体问题的操作，也可以说是对数据层的操作，对数据业务逻辑处理。

数据层：该层所做事务直接操作数据库，针对数据的增添、删除、修改、查找等。

工作流管理：工作流管理是一个被业界广泛应用并迅速发展的技术，它的主要特点是使处理过程自动化，使人以及各种应用工具相互之间协调工作，以完成某项工作。其目的是为了合适的人或软件在恰当的时间执行正确的工作。

主要的软件需求：包括三个不同的层次：业务需求，用户需求和功能需求，也包括非功能需求。业务需求反映了组织机构或客户对系统、产品高层次的目标要求，它们在项目视图与范围文档中给予说明。用户需求文档描述了用户使用产品必须要完成的任务，功能需求定义开发人员必须实现的软件功能，使用户能完成它们的任务，从而满足业务需求。

## 1.3参考资料

[1]《软件工程》 作者:（英）萨默维尔　著，程成　等译 出版社:机械工业出版社 出版时间:2011年05月

[2]《软件项目管理》 作者:（英）休斯，（英）考特莱尔 著，廖彬山，周卫华 译 出版社:机械工业出版社 出版时间:2010年09月

[3] GB/T 11457 软件工程术语

[4] GB/T 12505 计算机软件配置管理计划规范

[5]《云南省依法治省信息综合管理平台需求调研规格说明书》

[6]《JavaWeb设计实用教程——基于Eclipse环境的应用开发》 作者:孔昊　主编出版社:机械工业出版社 出版时间:2012年04月

# 2.总体设计

## 2.1需求规定

在充分理解《软件需求说明书》和《需求调研规格说明书》的基础上，充分了解针对于广大游戏用户的需求，来最大限度满足这些游戏用户对于网上和游戏中交朋友或者对游戏中的物品和账号进行交易的需求。

### 2.1.1系统功能

黑we购包括五个部分：账号模块，好友模块，游戏模块，商品模块和订单模块。

系统的结构框图如下：

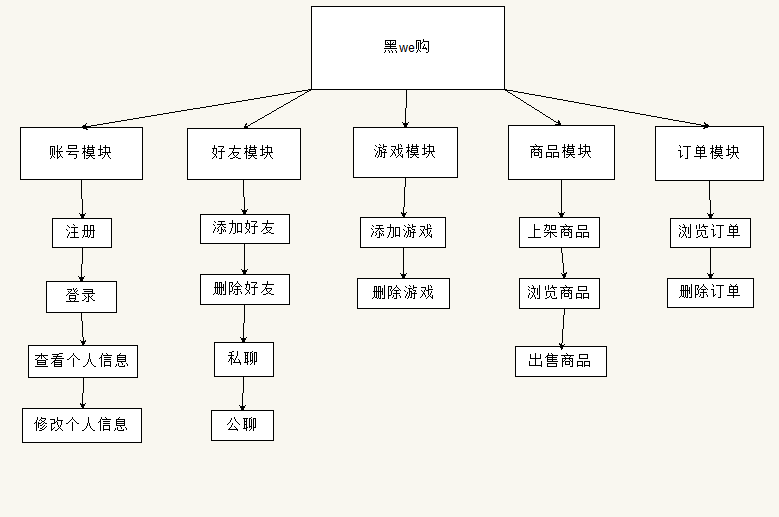


图 45 系统结构框图

### 2.1.2系统性能

针对黑we购网站的使用特性以及用户的特性，系统强调稳定性和安全性，包括账号的安全性和交易的安全性，保证网站不会轻易崩溃，能够承受数量大的用户同时访问，并且在系统出现崩溃的时候能够及时修复。

#### 2.1.2.1可维护性

本系统采用MVC模式开发，各层次代码分离，在代码上添加大量的注释，提高了软件的可理解性、可测试性和可修改性，从而提高了系统的可维护性。

#### 2.1.2.2可扩展性

系统在接口设计上尽量标准化，在版本更新时以实现和旧版本系统的无缝连接。

#### 2.1.2.3安全性原则

（1）数据库安全性

1. 定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储；

2. 对几个月未登录的账户发送通知，采取长期不登陆注销该账户，清除该账户的所有数据，减轻数据库压力；

3. 数据库的安全性就是指数据库中数据的保护措施，一般包括的登陆的身份验证管理、数据库的使用权限管理和数据库中对象的使用权限管理三种安全性保护措施，更改数据库不同数据的安全级别；

4. 对数据库的事物处理多次进行检查和测试，确保不会造成错误。

（2）交易安全性

提供一个支付平台，来保障双方的安全。先付款到网站相应的支付平台上，卖家看到显示才会发货，此时货款不属于买家，也不属于卖家，当买家收货觉得没问题之后，需要确认付款，此时支付平台才会把货款转入卖家帐户。而对卖家来说，他发了货，有发货凭证，买家也无法提出任何不合理的退款要求，而即使买家不确认付款操作，系统在一定时间后也会自动划账给卖家。要使用这种付款方式，需要一张银行卡就可以，或者邮局汇款，或者使用支付宝账户也可以。

#### 2.1.2.4稳定性

（1）定期维护，防止病毒入侵。

（2）一旦出现故障，尽快修复保证网站正常运行；

（3）定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

#### 2.1.2.5数据完备性

（1）定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

（2）当数据库的内容发生改变时，将会自动备份数据。

### 2.1.3输入输出要求

输入：一般为使用人员键盘输入。

输出：一般为屏幕输出。

### 2.1.4数据管理能力要求

（1）数据库不因庞大的数据而造成不能处理或者数据缺失的情况。

（2）数据资源管理系统标准统一。

（3）要求能进行数据库的建立、数据库的调优、数据库的重组、数据库的重构、数据库的安全管控、报错问题的分析和汇总和处理、数据库数据的日常备份。

### 2.1.5故障处理要求

#### 2.1.5.1硬件故障

硬件故障为浏览器故障或者操作系统故障，造成硬件故障与网站无关。

#### 2.1.5.2软件故障

（1）定期维护，防止病毒入侵。

（2）一旦出现故障，尽快修复保证网站正常运行；

（3）定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

### 2.1.6其他专门要求

数据库维护人员和网站维护人员应分开，不让同一批人次负责这两个方面的维护。

## 2.2运行环境

### 2.2.1 硬件环境

服务器端：

安装有MySql5.5以上版本的计算机

客户端：

安装有各种浏览器的计算机。

### 2.2.2 系统运行环境

服务器端：

数据库：MySql5.5以上版本

客户端：

各种浏览器

### 2.2.3接口

硬件接口：

无硬件接口要求

软件接口：

操作系统：Windows2007以上

浏览器：各种浏览器均可以

### 2.2.4控制

运行环境的控制将在用户手册中指出，提示用户使用相应的操作系统或者浏览器进行本系统的访问。

## 2.3基本设计概念和处理流程

### 2.3.1总体流程描述

1. 用户注册后选择登录。

2. 用户选择个人信息进行查看和修改。

3. 用户选择好友系统进行查看好友和添加删除好友，用户选择对好友或者非好友发送私聊或者公聊。

4. 用户选择游戏系统进行查看游戏和添加删除游戏。

5. 用户选择交易系统进行查看商品列表和上架或者购买商品。

6. 用户选择订单系统进行查看订单列表和删除订单。

7. 用户点击注销退出账号。

8. 用户关闭网页退出系统。

### 2.3.2模块的标准流程

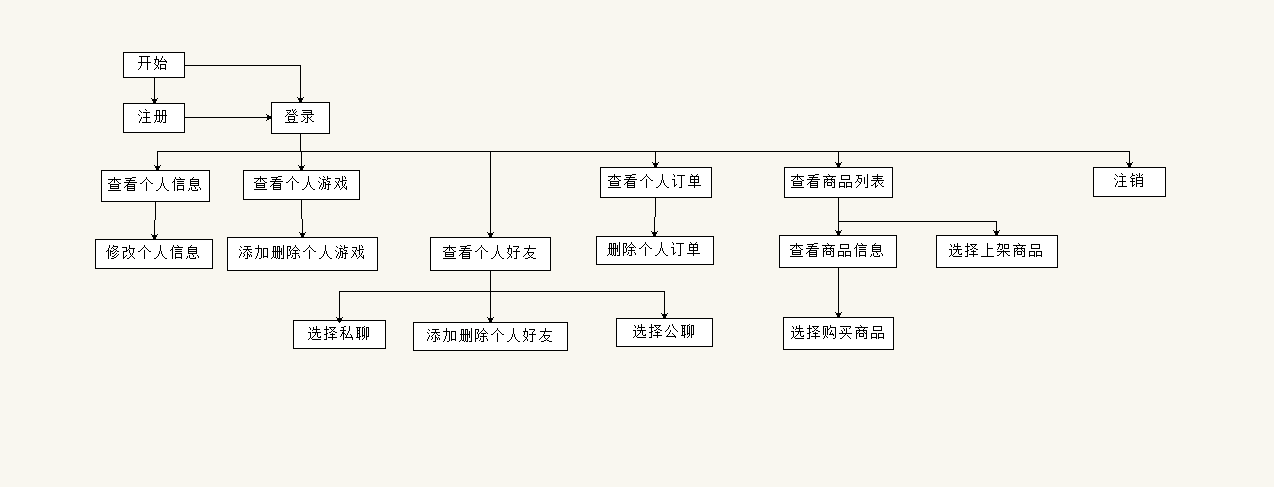


图 46 模块标准流程图

### 2.3.3安全控制

（1）数据库中的事物需要进行多次修改和验证，确保事物的准确性。

（2）提供一个支付平台，来保障双方的安全。先付款到网站相应的支付平台上，卖家看到显示才会发货，此时货款不属于买家，也不属于卖家，当买家收货觉得没问题之后，需要确认付款，此时支付平台才会把货款转入卖家帐户。而对卖家来说，他发了货，有发货凭证，买家也无法提出任何不合理的退款要求，而即使买家不确认付款操作，系统在一定时间后也会自动划账给卖家。要使用这种付款方式，需要一张银行卡就可以，或者邮局汇款，或者使用支付宝账户也可以。

### 2.3.4权限控制

1. 普通用户通常没有修改服务器端数据库的权限，只有管理员才拥有对数据库操作的权限。

2. 普通用户在申请注册或修改信息时，若请求通过，则数据库会自动添加或修改数据。

3. 普通用户在修改个人信息时，若请求通过，则数据库会自动添加或修改数据。

4. 普通用户在添加删除好友和游戏时，若请求通过，则数据库会自动添加或修改数据。

5. 普通用户在操作网站上的选项进行商品和订单修改时，若请求通过，则数据库会自动添加或修改数据。

6. 普通用户无法访问数据库中除了自己之外的其他数据。

### 2.3.5工作流管理

无

### 2.3.6软件质量约束

正确性：在系统运行中只有在用户输入正确的信息才能得出正确的结论，如果输入错误的信息就会有对话框提示用户输入错误。

健壮性：本系统除开发人员外其他人不得擅自修改本系统，因此在用户使用系统的过程中不会出现恶意毁坏系统的现象。

效率（性能）：本系统在用户填写完必要的信息之后操作人员只需通过点击按钮就可完成相应功能，操作很简便，通过简单的操作就可完成复杂的任务。

可靠性：除开发人员外其他人不得擅自修改本系统，以确保程序具有可靠性。

易用性：当输入完所需采购物品的数据后进行保存，以防系统崩溃等因素造成数据的丢失。

系统处理完数据自动对数据进行储存，以防重复处理同一数据。

清晰性：在本系统的运行过程中出现的所有对话框都会明确提醒用户，根据自己想要达到的要求合理选择相应功能按键

安全性：后台管理除了管理员外其他人不能随意批准，一经批准就不能随便修改，只有供应商可以查询库存情况。在登录的时候已经对不同身份的人进行了识别，拥有不同权限的人可进行不同的操作，权限设置保证了系统的安全性

兼容性：本系统具有很强的兼容性。

可移植性：本软件的运行不依赖于操作系统，而是依赖于Java Web的运行环境。所以不管移植到什么操作系统上，只要有Java Web的运行环境本软件就能运行。

### 2.3.7设计策略

（1）扩展策略：本系统是一个单独的管理系统，与其它的系统可以同时使用不受影响。

（2）折衷策略：在面对“时-空”效率发生冲突这一问题时，我们将尽可能的提高空间利用率，因为系统要和多个消费者进行通信，所以，相比较而言，我们更注重时间利用率。

### 2.3.8主要数据结构和算法

#### 2.3.8.1 连接数据库

JDBC与MySQL数据库的连接:

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**public** **class** DbUtil {

**private** String dbUrl = "jdbc:mysql://localhost:3306/herewego" ; // 数据库名字未写

**private** String dbUserName = "root" ;

**private** String dbPassword = "987654321+." ;

**private** String jdbcName = "com.mysql.jdbc.Driver" ;

**public** Connection getCon() **throws** Exception {

Class.*forName*(jdbcName) ;

Connection con = DriverManager.*getConnection*(dbUrl, dbUserName, dbPassword) ;

**return** con ;

}

**public** **void** closeCon (Connection con) **throws** Exception {

**if** (con != **null** ) {

con.close();

}

}

}

#### 2.3.8.2 servlet处理表单数据

获取表单里面的id值并处理:

**import** java.io.IOException;

**import** java.sql.Connection;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** com.herewego.dao.UserDao;

**import** com.herewego.model.Game;

**import** com.herewego.model.User;

**import** com.herewego.util.DbUtil;

//注册

**public** **class** RegisterServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L ;

DbUtil dbUtil = **new** DbUtil() ;

UserDao userDao = **new** UserDao() ;

@Override

**protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

**this**.doPost(req, resp);

}

//复写doPost方法

@Override

**protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {

//从表单提交中获取用户基本信息

String userName = req.getParameter("userName") ;

String password = req.getParameter("password") ;

String nickName = req.getParameter("nickName") ;

String sex = req.getParameter("sex") ;

**int** age = Integer.*parseInt*( req.getParameter("age") );

String constellation = req.getParameter("constellation") ;

String gameName = req.getParameter("game") ;

String district = req.getParameter("district") ;

Connection con = **null** ;

**try** {

//封装成两个对象

User user = **new** User(userName, password, nickName, sex, age, constellation) ;

Game game = **new** Game( gameName , district) ;

con = dbUtil.getCon() ;

**boolean** registerUser = userDao.register(con, user , game) ;

//返回值为true表示该用户不存在，注册成功，赋值success为注册成功提醒用户

**if** ( registerUser ) {

req.setAttribute("success", "注册成功");

req.getRequestDispatcher("register.jsp").forward(req, resp);

}

//返回值为false表示该用户存在，注册失败，赋值userFalse为用户名已经被占用提醒用户

**else** {

req.setAttribute("userfalse", "注册失败");

req.getRequestDispatcher("register.jsp").forward(req, resp);

}

} **catch** (Exception e) {

e.printStackTrace();

}**finally** {

**try** {

dbUtil.closeCon(con);

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

}

#### 2.3.8.3 servlet实现界面跳转

重定向和请求转发:

req.getRequestDispatcher("index.jsp").forward(req, resp);

}**else** {

//不为空则跳转页面至主页

HttpSession session = req.getSession() ;

session.setAttribute("currentUser", currentUser);

resp.sendRedirect("main.jsp");

#### 2.3.8.4 model封装

将类封装起来便于与数据库的操作:

**public** **class** Order {

**private** **int** order\_id ;

**private** **int** good\_id ;

**private** String name ;

**private** **int** number ;

**private** String buy\_id ;

**private** String sell\_id ;

**private** **float** price ;

**private** String time ;

**private** String state ;

**public** Order(**int** order\_id, **int** good\_id, String name, **int** number, String buy\_id, String sell\_id, **float** price,

String time, String state) {

**super**();

**this**.order\_id = order\_id;

**this**.good\_id = good\_id;

**this**.name = name;

**this**.number = number;

**this**.buy\_id = buy\_id;

**this**.sell\_id = sell\_id;

**this**.price = price;

**this**.time = time;

**this**.state = state;

}

**public** Order() {

// **TODO** Auto-generated constructor stub

}

**public** **int** getOrder\_id() {

**return** order\_id;

}

**public** **void** setOrder\_id(**int** order\_id) {

**this**.order\_id = order\_id;

}

**public** **int** getGood\_id() {

**return** good\_id;

}

**public** **void** setGood\_id(**int** good\_id) {

**this**.good\_id = good\_id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getNumber() {

**return** number;

}

**public** **void** setNumber(**int** number) {

**this**.number = number;

}

**public** String getBuy\_id() {

**return** buy\_id;

}

**public** **void** setBuy\_id(String buy\_id) {

**this**.buy\_id = buy\_id;

}

**public** String getSell\_id() {

**return** sell\_id;

}

**public** **void** setSell\_id(String sell\_id) {

**this**.sell\_id = sell\_id;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) {

**this**.price = price;

}

**public** String getTime() {

**return** time;

}

**public** **void** setTime(String time) {

**this**.time = time;

}

**public** String getState() {

**return** state;

}

**public** **void** setState(String state) {

**this**.state = state;

}

}

#### 2.3.8.5 jsp界面程序

利用jsp实现写动态页面的功能,动态html。

<table width=*"1817"* border=*"1"* bordercolor=*"black"* cellspacing=*"0"*>

<tr>

<td width=*"904"* align=*"center"*>

</div> <section class=*"container"*>

<div class=*"login"*>

<h1>我的好友</h1>

<%

DbUtil dbUtil = **new** DbUtil() ;

UserDao userDao = **new** UserDao() ;

Connection con = dbUtil.getCon() ;

User currentUser = (User)session.getAttribute("currentUser") ;

ArrayList<String> strFriend = userDao.getFriends(con, currentUser ) ;

**for** (**int** i = 0 ; i < strFriend.size() ; i++ ) {

out.println("<p>") ;

out.println(" <font face='楷体\_gb2312' color='#dd00dd' size = 'xx-large' > ") ;

out.println(strFriend.get(i));

out.println("</font>") ;

out.println("</p>") ;

out.println("<p><img src='zczindex/image/shy.png' width='30' height='30' /> ") ;

out.println("<img src='zczindex/image/lt.png' width='30' height='30' /></p> ") ;

}

%>

## 2.4系统程序模块结构

## 

图 47 系统程序模块结构图

## 2.5功能需求与程序模块的关系

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 功能需求 | 账号管理子系统 | 好友子系统 | 游戏子系统 | 商品子系统 | 订单子系统 |
| 注册 | √ |  |  |  |  |
| 登录 | √ |  |  |  |  |
| 查看个人信息 | √ |  |  |  |  |
| 修改个人信息 | √ |  |  |  |  |
| 添加好友 |  | √ |  |  |  |
| 删除好友 |  | √ |  |  |  |
| 私聊 |  | √ |  |  |  |
| 公聊 |  | √ |  |  |  |
| 添加游戏 |  |  | √ |  |  |
| 删除游戏 |  |  | √ |  |  |
| 上架商品 |  |  |  | √ |  |
| 浏览商品 |  |  |  | √ |  |
| 购买商品 |  |  |  | √ |  |
| 浏览订单 |  |  |  |  | √ |
| 删除订单 |  |  |  |  | √ |

表 48 功能需求与程序模块的关系

## 2.6人工处理过程

本系统运行中，需要以下的人工处理过程：

（1）用户首次使用网站的账号的注册。

（2）服务器数据库的数据恢复

（3）系统的维护与升级

## 2.7尚未解决的问题

（1） 尚有一些隐藏的BUG需要在使用过程中进一步发现，在后续维护过程中逐渐更新这些BUG。

（2）由于支付宝或者银联的接口支持需要支付宝方面的条件，比如说后续需要网站的营业证书才能使用支付宝接口，或者是支付宝接口使用后交易成功需要提交一部分所得额给支付宝。

（3）界面的美观问题，一个良好的美观界面能给用户一种舒服的感受。

# 3.接口设计

## 3.1用户接口

1. 本系统不支持命令行操作。

2. 系统提供用户接口均以用户图形化界面展示，本系统是B/S系统，用户只需要按照网页上的操作进行点击就可以实现对应操作。

3. 用户用户名或者密码有误，系统提示用户“登录失败，用户名或密码错误”。

4. 用户删除了不存在的个人游戏或者个人好友，系统提示用户“不存在该好友”或者“不存在该游戏”。

5. 用户查看了不存在商品列表的商品，系统提示用户“不存在该商品”。

6. 用户删除了不存在订单列表的订单，系统提示用户“不存在该订单”。

7. 用户使用浏览器进行操作，接口为浏览器。

## 3.2外部接口

### 3.2.1软件接口

OA服务器：

操作系统：Windows2007以上

应用软件：MySQL5.5以上

客户机：

操作系统：Windows2007以上

浏览器：各种浏览器均可以

|  |  |  |  |
| --- | --- | --- | --- |
|  | **名称** | **版本** | **语种** |
| 操作系统 | Windows | 2007 | 简体中文 |
| 数据库平台 | MySQL | 5.5 | 简体中文 |
| 应用平台 | Eclipse | JavaEE | English |

表 49 软件接口表

### 3.2.2 硬件接口

无

## 3.3内部接口

每个子系统都是相互联系的，只有完成某一操作才能做下一操作，但是在具体实现过程中，彼此相互独立，可分别编码。

系统采用以下接口：

数据库连接接口：系统按照MySQL的接口连接方式链接上MySQL数据库接口。

支付宝接口：系统按照支付宝的接口连接方式链接上支付宝的接口。

# 4.运行控制

## 4.1运行模块组合

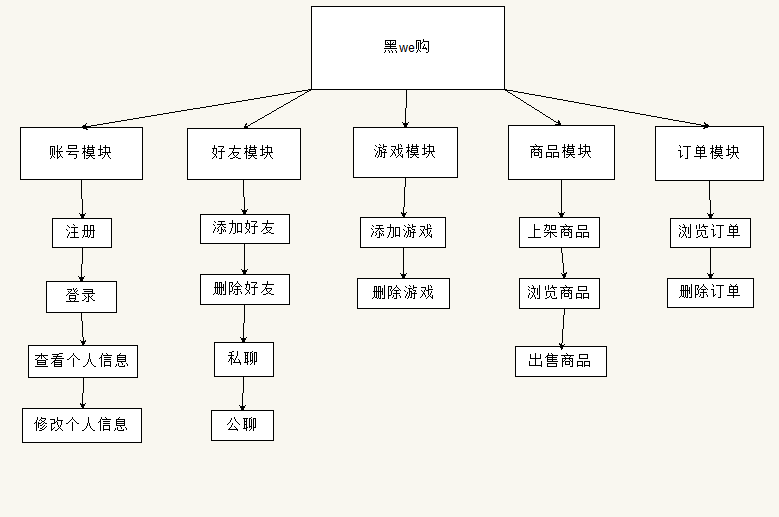


图 50 运行模块组合图

经过分析该网站的主要功能应该包括下面几个：

（1）注册账号

（2）登录账号

（3）查看个人信息

（4）修改个人信息

（5）添加好友

（6）删除好友

（7）私信聊天

（8）公屏聊天

（9）添加游戏

（10）删除游戏

（11）上架商品

（12）浏览商品

（13）购买商品

（14）浏览订单

（15）删除订单

|  |  |
| --- | --- |
| 功能描述： | 注册账号 |
| 输入： | 注册账号必须的各种信息 |
| 处理： | 查询数据库，生成数据库数据 |
| 输出： | 提示注册账号成功或失败 |

表 51 注册账号描述

|  |  |
| --- | --- |
| 功能描述： | 登录账号 |
| 输入： | 登录账号必须的用户名和密码 |
| 处理： | 查询数据库，判断是否存在账号 |
| 输出： | 登陆成功或失败 |

表 52登录账号描述

|  |  |
| --- | --- |
| 功能描述： | 查看个人信息 |
| 输入： | 提交查看个人信息申请 |
| 处理： | 查询数据库，显示在界面上 |
| 输出： | 个人信息 |

表 53 查看个人信息描述

|  |  |
| --- | --- |
| 功能描述： | 修改个人信息 |
| 输入： | 提交修改个人信息申请，并且输入修改信息的各项数据 |
| 处理： | 查询数据库，修改数据库 |
| 输出： | 提示修改个人信息成功，修改个人信息 |

表 54 修改个人信息描述

|  |  |
| --- | --- |
| 功能描述： | 添加好友 |
| 输入： | 选择要添加的好友 |
| 处理： | 查询数据库，生成数据库数据 |
| 输出： | 提示添加好友成功或失败 |

表 55 添加好友描述

|  |  |
| --- | --- |
| 功能描述： | 删除好友 |
| 输入： | 选择要删除的好友 |
| 处理： | 查询数据库，修改数据库数据 |
| 输出： | 提示删除好友成功或失败 |

表 56 删除好友描述

|  |  |
| --- | --- |
| 功能描述： | 私信聊天 |
| 输入： | 选择想要私信聊天的好友和内容 |
| 处理： | 发送http协议 |
| 输出： | 提示发送成功，对方接收 |

表 57 私信聊天描述

|  |  |
| --- | --- |
| 功能描述： | 公屏聊天 |
| 输入： | 输入想要聊天的内容 |
| 处理： | 发送http协议 |
| 输出： | 提示发送成功，所有用户接收 |

表 58 公屏聊天描述

|  |  |
| --- | --- |
| 功能描述： | 添加游戏 |
| 输入： | 选择要添加的游戏 |
| 处理： | 查询数据库，生成数据库数据 |
| 输出： | 提示添加游戏成功或失败 |

表 59 添加游戏描述

|  |  |
| --- | --- |
| 功能描述： | 删除游戏 |
| 输入： | 选择要添加的游戏 |
| 处理： | 查询数据库，修改数据库数据 |
| 输出： | 提示添加游戏成功或失败 |

表 60 删除游戏描述

|  |  |
| --- | --- |
| 功能描述： | 上架商品 |
| 输入： | 输入上架商品所需的各种信息 |
| 处理： | 查询数据库，生成数据库数据 |
| 输出： | 提示上架商品成功 |

表 61 上架商品描述

|  |  |
| --- | --- |
| 功能描述： | 浏览商品 |
| 输入： | 选择要浏览商品的编号 |
| 处理： | 查询数据库，返回界面显示 |
| 输出： | 商品信息 |

表 62 浏览商品描述

|  |  |
| --- | --- |
| 功能描述： | 购买商品 |
| 输入： | 选择要购买商品的编号 |
| 处理： | 查询数据库，生成数据库数据 |
| 输出： | 提示购买商品成功 |

表 63 购买商品描述

|  |  |
| --- | --- |
| 功能描述： | 浏览订单 |
| 输入： | 选择要浏览订单的编号 |
| 处理： | 查询数据库，返回界面显示 |
| 输出： | 订单信息 |

表 64 浏览订单描述

|  |  |
| --- | --- |
| 功能描述： | 删除订单 |
| 输入： | 选择要删除订单的编号 |
| 处理： | 查询数据库，修改数据库数据 |
| 输出： | 提示删除订单成功 |

表 65 删除订单描述

|  |  |  |
| --- | --- | --- |
| 硬件环境： | 运行Windows系统的计算机 | |
| 软件环境： | 操作系统： | Windows系统 |
| 数据库： | MySQL |

## 4.2运行控制

图表 66 运行控制

## 4.3运行时间

性能效率：响应时间小于5秒。

# 5.系统出错处理设计

## 5.1出错信息

系统出错信息整理如下所示：

1．网站同时在线的人数过多，同时进行同一操作时造成网站卡死，系统崩溃；

2．服务器数据库故障，造成数据错误或者无法正常使用；

3. 网站处理异常，存在未知BUG。

## 5.2补救措施

为防止系统运行错误：

（1）定期维护，防止病毒入侵。

（2）加强服务器以及网站的性能测试，增强网站的安全性；

（3）一旦出现故障，尽快修复保证网站正常运行；

（4）加大软件测试的强度，尽量减少网站的出错；

（5）定期对数据库的数据进行备份，清空缓存，将不同的数据分开存储。

## 5.3系统维护设计

系统没有外加维护模块，维护工作包括数据库的基本维护和系统运行环境维护。

定期检查网站中存在的BUG，获取用户的及时反馈。

检查服务器的操作系统（安装系统补丁，扫描病毒，检查端口，关闭不必要的服务等）。

系统的维护人员分开，错开开发人员，避免个人意愿的维护出错。

**黑we购软件测试报告**

[1.测试概述 63](#_Toc470100809)

[1.1 编写目的 63](#_Toc470100810)

[1.2 项目背景和简介 63](#_Toc470100811)

[1.3 项目的产品规格说明书（Specification） 64](#_Toc470100812)

[1.3.1 功能性需求 64](#_Toc470100813)

[1.3.2 非功能性需求 66](#_Toc470100814)

[1.4 项目的设计文档 67](#_Toc470100815)

[1.4.1 架构设计 67](#_Toc470100816)

[1.4.2 模块设计 68](#_Toc470100817)

[1.5 测试范围 72](#_Toc470100818)

[1.6 测试环境与配置 73](#_Toc470100819)

[2.测试用例设计 74](#_Toc470100820)

[2.1 单元测试 74](#_Toc470100821)

[2.1.1 代码审查 74](#_Toc470100822)

[2.1.2单元测试的用例设计 85](#_Toc470100823)

[2.2 集成测试的用例设计 102](#_Toc470100824)

[2.2.1 测试场景 102](#_Toc470100825)

[2.2.2 测试方法 104](#_Toc470100826)

[2.2.3 测试用例 105](#_Toc470100827)

[2.2.4 测试过程与测试结果 106](#_Toc470100828)

[2.3 非功能测试的用例设计 121](#_Toc470100829)

[2.4 自动化测试工具非功能测试 122](#_Toc470100830)

[3.测试总结 125](#_Toc470100831)

[3.1 测试用例执行结果 125](#_Toc470100832)

[3.2 测试问题解决 125](#_Toc470100833)

[3.3 测试结果分析 126](#_Toc470100834)

[3.3.1 单元测试结果分析 126](#_Toc470100835)

[3.3.2 集成测试结果分析 126](#_Toc470100836)

[3.3.3 非功能测试结果分析 126](#_Toc470100837)

[4.综合评价 127](#_Toc470100838)

[4.1软件能力 127](#_Toc470100839)

[4.2建议 127](#_Toc470100840)

1.测试概述

## 1.1 编写目的

本报告为专业实训课的大作业黑we购网站的测试报告，目的在于总结测试阶段的测试以及分析测试结果，找出系统存在的BUG，以期系统符合专业实训课的大作业需求。

## 1.2 项目背景和简介

（1）本项目的题目是黑we购网站；

（2）网站的主要背景：

①随着游戏在年轻人甚至中年人之间受欢迎的程度不断提高，人们花在游戏上面的时间越来越多。

②现今各种直播平台、公众明星对电子游戏的宣传，使得游戏越来越深入人心，以游戏为娱乐的主要方式更加被人接受。

③游戏虽然只是虚拟数据，但游戏玩家在游戏中结识的人可以是真实的人，这些人也许能成为他的好朋友。

④很多小伙伴想要入坑或者想要脱坑，苦于游戏没有平台来进行交易，或者交易不方便，线下交易又很容易上当受骗。

（3）网站的主要目的：

这个网站主要就提供给那些在游戏中还是形单影只的人一个交友平台，毕竟游戏还是要和他人一起玩才会有趣。提供一个社交平台，并为那些想要交易的小伙伴提供一个交易的平台。

（4）后台的开发语言是JAVA程序设计语言，前台的开发语言主要是HTML，CSS，JAVASCRIPT。

（5）集成开发环境为Eclipse 和 Dreamweaver；

（6）本网站是B/S系统，运行环境是浏览器。

## 1.3 项目的产品规格说明书（Specification）

### 1.3.1 功能性需求

#### 1.3.1.1 账号模块

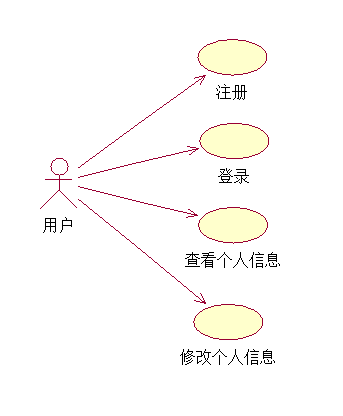


图 67 账号模块

#### 1.3.1.2 好友模块

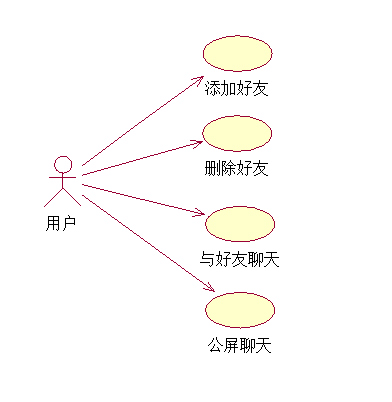


图 68 好友模块

#### 1.3.1.3 游戏模块

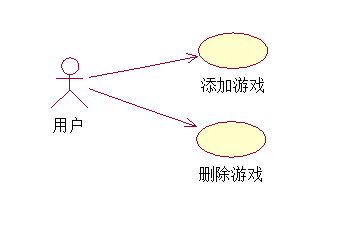


图 69 游戏模块

#### 1.3.1.4 商品模块

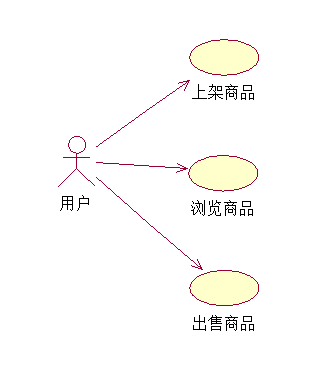


图 70 商品模块

#### 1.3.1.5 订单模块

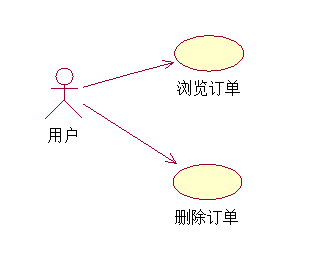


图 71 订单模块

### 1.3.2 非功能性需求

#### 1.3.2.1时间特性

1) 系统响应时间< 1s

2) 界面更新处理时间< 5s

3) 数据转换与传输时间< 1s

#### 1.3.2.2硬件环境需求

操作系统： windows10/windows8/windows7/windowsXP/linux 等通用操作系统均可运行。

#### 1.3.2.3产品质量需求

兼容性：系统开发框架是java 开源框架，可兼容大多数开源浏览器

## 1.4 项目的设计文档

### 1.4.1 架构设计

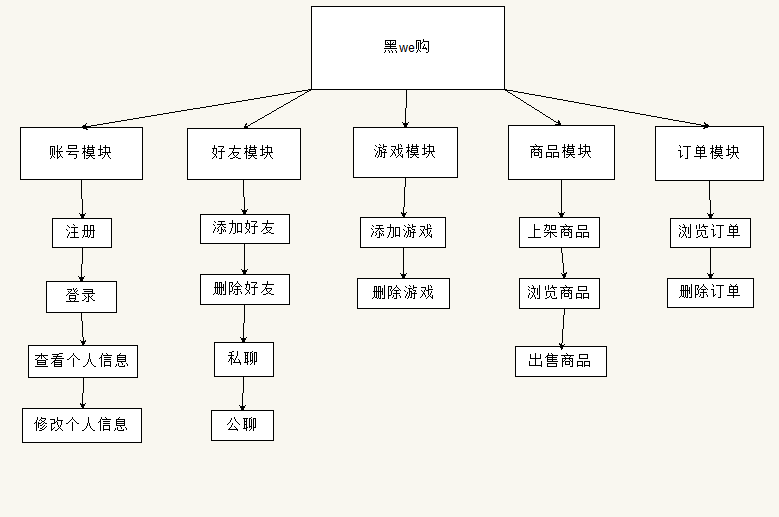


图 72 架构设计

### 1.4.2 模块设计

#### 1.4.2.1 账号模块

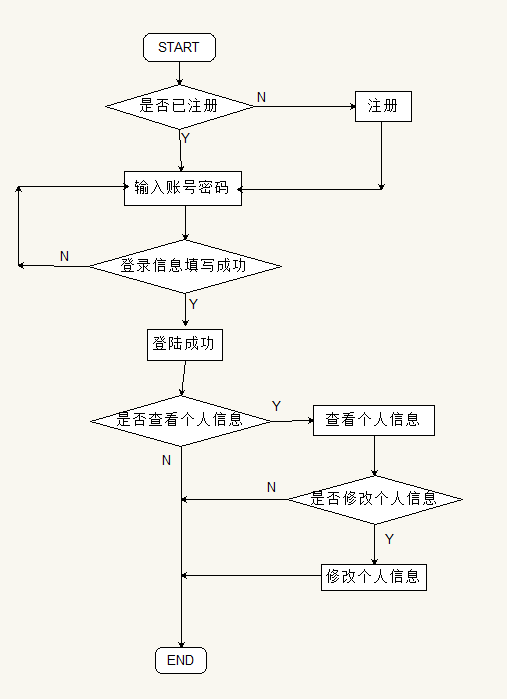


图 73 账号模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 账号模块 |
| 用例描述 | 用户注册登录账号管理账号，查看信息 |
| 参与者 | 所有用户 |
| 前置条件 | 无 |
| 后置条件 | 登陆成功 |
| 正常时间流 | 1.新用户注册账号，注册成功跳转登录界面  2.已有账号的用户输入正确的用户名密码登陆  3.用户进入个人主页管理自己的个人信息 |
| 异常事件流 | 1.用户名密码错误  2.注册输入无效的信息 |

表 74 账号模块用例描述

#### 1.4.2.2 好友模块

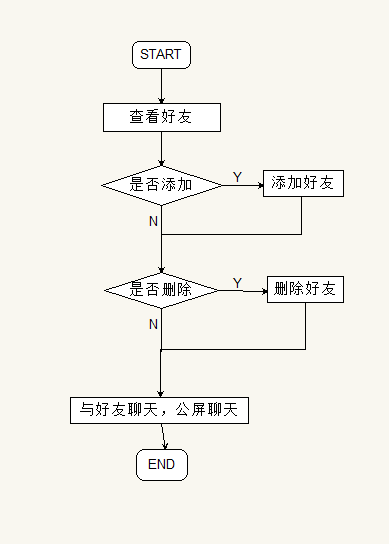


图 75 好友模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 好友模块 |
| 用例描述 | 用户添加好友和好友和非好友进行聊天 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入好友系统  2.查看自己的好友和系统推荐好友  3.选择添加和删除好友  4.选择公聊或者私聊 |
| 异常事件流 | 1.添加的好友不存在  2.聊天好友不存在 |

表 76 好友模块用例描述

#### 1.4.2.3 游戏模块

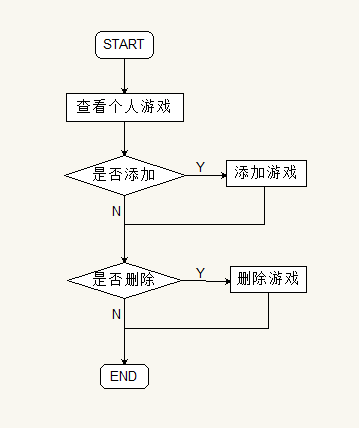


图 77 游戏模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 游戏模块 |
| 用例描述 | 用户管理自己的游戏 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入游戏  2.查看自己的游戏  3.选择添加和删除游戏 |
| 异常事件流 | 无 |

表 78 游戏模块用例描述

#### 1.4.2.4 商品模块

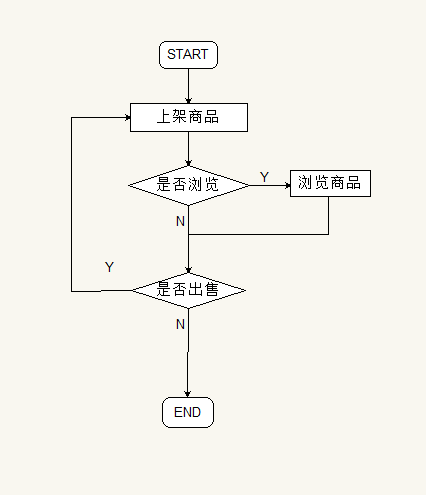


图 79 商品模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 商品模块 |
| 用例描述 | 用户出售购买浏览商品 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入交易系统  2.浏览商品信息  3.选择上架或购买商品 |
| 异常事件流 | 1.商品不存在或已被购买  2. 金额不够支付 |

表 80 商品模块用例描述

#### 1.4.2.5 订单模块

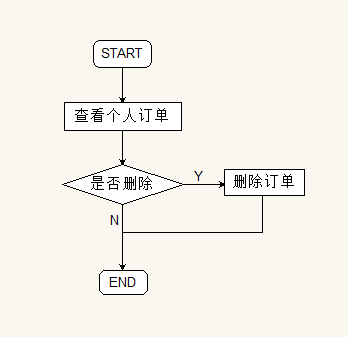


图 81 订单模块

|  |  |
| --- | --- |
| 描述项 | 说明 |
| 用例名称 | 订单模块 |
| 用例描述 | 用户查看个人订单管理订单 |
| 参与者 | 所有用户 |
| 前置条件 | 登陆成功 |
| 后置条件 | 无 |
| 正常时间流 | 1.登陆成功后进入订单系统  2.查看自己的订单  3.选择删除订单 |
| 异常事件流 | 无 |

表 82订单模块用例描述

## 1.5 测试范围

测试主要依据产品规格说明书和设计文档对系统进行测试，包括单元测试、集成测试、功能测试和系统测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试类型 | 测试内容 | 测试目的 | 测试方法和工具 |
| 单元测试 | 登录功能测试 获取商品列表功能测试 获取推荐好友功能测试 添加好友功能测试 | 结合代码测试，测试各个功能的具体代码，具体测试每个功能代码的分支是否都可以正常运行。 | 使用Eclipse，junit写测试代码，运行检测各个类。 |
| 集成测试 | 注册登录模块 用户信息模块 好友模块 游戏模块 聊天模块 商品模块 订单模块 | 将各个已测试过的模块组合进行集成测试，以发现模块之间的接口错误等 | 自底向上 |
| 非功能性测试 | 对网站进行大数据量测  试、负载测试、压力测  试 | 测试网站性能稳定性和性能瓶颈 | Apache JMeter3.1性能测试工具和 badboy脚本录制工具 |

表 83 各测试测试范围

## 1.6 测试环境与配置

|  |  |
| --- | --- |
| 资源名称/类型 | 配 置 |
| 测试PC | 主频2.20GHz，硬盘512G，内存8G |
| 应用服务器 | Apache Tomcat 7.0.69服务器 |
| 数据库管理系统 | MySql 5.5 |
| 集成开发环境 | Eclipse |
| 代码检测工具 | PMD或CheckStyle，SourceMonitor |
| 负载性能测试工具 | Apache Jmeter3.1 |

表 84 测试环境与配置

2.测试用例设计

## 2.1 单元测试

### 2.1.1 代码审查

#### 2.1.1.1 使用PMD进行代码检查

（1）PMD是一种分析Java代码错误的工具。与其他分析工具不同的是，PMD通过静态分析获知代码过错。也就是说，在不运行Java程序的情形下检查问题。PMD附带了许多可以直接应用的规则，利用这些规则可以找出Java源程序的许多问题，例如没有用到的变量、过剩的变量创建操作、空的catch块， 等等。此外，用户还可以自定义规则，检查Java代码是否合乎某些特定的编码标准。例如，你可以编写一个规则，要求PMD找出所有创立Thread和 Socket对象的操作。

（2）PMD，用于检测JAVA源文件中的潜在问题，主要包括：

① 潜在的bug：空的try/catch/finally/switch语句

② 未使用的代码：未使用的局部变量、参数、私有方法等

③ 可选的代码：String/StringBuffer的滥用

④ 复杂的表达式：不必须的if语句、可以使用while循环完成的for循环

⑤ 重复的代码：拷贝/粘贴代码意味着拷贝/粘贴bugs

⑥ 循环体创建新对象：尽量不要再for或while循环体内实例化一个新对象

⑦ 资源关闭：Connect，Result，Statement等使用之后确保关闭掉

1. PMD包含的十六个规则集：

①基本（rulesets/basic.xml）—— 规则的一个基本合集，可能大多数开发人员都不认同它： catch 块不该为空，无论何时重写 equals()，都要重写 hashCode()，等等。

② 命名（rulesets/naming.xml）—— 对标准 Java 命令规范的测试：变量名称不应太短；方法名称不应过长；类名称应当以小写字母开头；方法和字段名应当以小写字母开头，等等。

③ 未使用的代码（rulesets/unusedcode.xml）—— 查找从未使用的私有字段和本地变量、执行不到的语句、从未调用的私有方法，等等。

④ 设计（rulesets/design.xml）—— 检查各种设计良好的原则，例如： switch 语句应当有 default 块，应当避免深度嵌套的 if 块，不应当给参数重新赋值，不应该对 double 值进行相等比较。

⑤ 导入语句（rulesets/imports.xml）—— 检查 import 语句的问题，比如同一个类被导入两次或者被导入 java.lang 的类中。

⑥ JUnit 测试（rulesets/junit.xml）—— 查找测试用例和测试方法的特定问题，例如方法名称的正确拼写，以及 suite() 方法是不是 static 和 public。

⑦ 字符串（rulesets/string.xml）—— 找出处理字符串时遇到的常见问题，例如重复的字符串标量，调用 String 构造函数，对 String 变量调用 toString() 方法。

⑧ 括号（rulesets/braces.xml）—— 检查 for、 if、 while 和 else 语句是否使用了括号。

⑨ 代码尺寸（rulesets/codesize.xml）—— 测试过长的方法、有太多方法的类以及重构方面的类似问题。

⑩Javabean（rulesets/javabeans.xml）—— 查看 JavaBean 组件是否违反 JavaBean 编码规范，比如没有序列化的 bean 类。

⑪ 终结函数（finalizer）—— 因为在 Java 语言中， finalize() 方法不是那么普遍（我上次编写这个代码也经是好多年前的事了），所以它们的使用规则虽然很详细，但是人们对它们相对不是很熟悉。这类检查查找 finalize() 方法的各种问题，例如空的终结函数，调用其他方法的 finalize() 方法，对 finalize() 的显式调用，等等。

⑫ 克隆（rulesets/clone.xml）—— 用于 clone() 方法的新规则。凡是重写 clone() 方法的类都必须实现 Cloneable， clone() 方法应该调用 super.clone()，而 clone() 方法应该声明抛出 CloneNotSupportedException 异常，即使实际上没有抛出异常，也要如此。

⑬ 耦合（rulesets/coupling.xml）—— 查找类之间过度耦合的迹象，比如导入内容太多；在超类型或接口就已经够用的时候使用子类的类型；类中的字段、变量和返回类型过多等。

⑭ 严格的异常（rulesets/strictexception.xml）—— 针对异常的测试：不应该声明该方法而抛出 java.lang.Exception 异常，不应当将异常用于流控制，不应该捕获 Throwable，等等。

⑮ 有争议的（rulesets/controversial.xml）—— PMD 的有些规则是有能力的 Java 程序员可以接受的。但还是有一些争议。这个规则集包含一些更有问题的检验，其中包括把 null 赋值给变量、方法中有多个返回点，以及从 sun 包导入等。

⑯ 日志（rulesets/logging-java.xml）—— 查找 java.util.logging.Logger 的不当使用，包括非终状态（nonfinal）、非静态的记录器，以及在一个类中有多个记录器。

（4） PMD的错误级别：

PMD priority 1 -> severity error， priority high

PMD priority 2 -> severity error， priority normal

PMD priority 3 -> severity warning， priority high

PMD priority 4 -> severity warning， priority normal

PMD priority 5 -> severity information， priority normal

即：

PMD的优先级1 - 严重的错误，优先高

PMD的优先级2 - 严重的错误，优先正常

PMD的优先级3 - 严重警告，优先高

PMD的优先级4 - 严重警告，优先正常

PMD的优先级5 - 严重的信息，优先正常

即使是PMD priority 1，不意味着代码出错，而是代码不规范。如：

static final的变量非全部大写。

static final sysLoger --- static final SYSLOGER

（5） 运行PMD检查结果如下

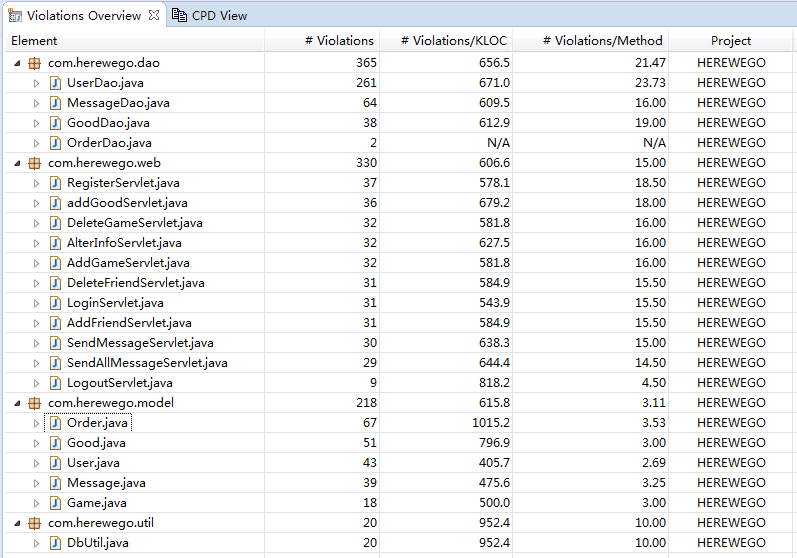


图 85 PMD检查结果

上图中各列名意义如下表：

|  |  |
| --- | --- |
| #Violations | 代码不规范数（处） |
| #Violations/KLOC | 不规范数/代码行数（单位：处/千） |
| #Violations/Method | 不规范数/方法数（单位：处/个） |

表 86 PMD列的意义

不规则处左侧有箭头标示：

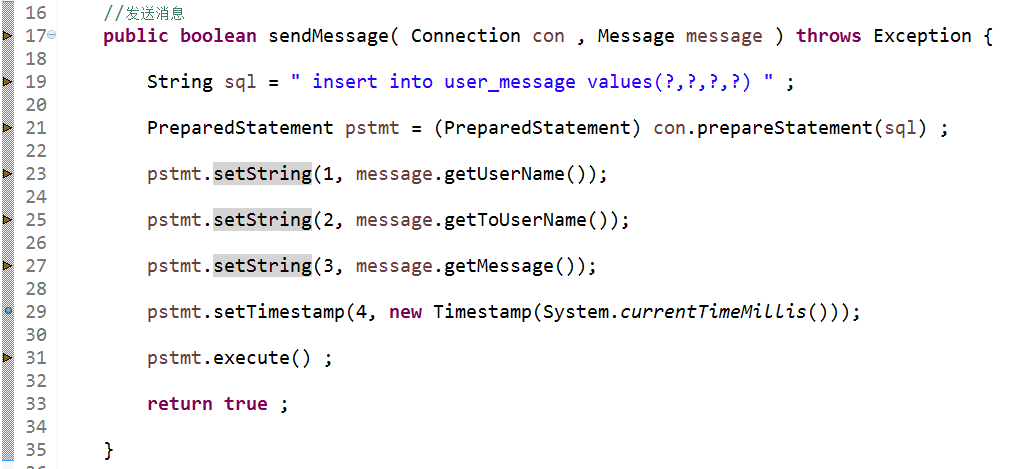


图 87 不规则处展示

导出的不规则信息统计文本：

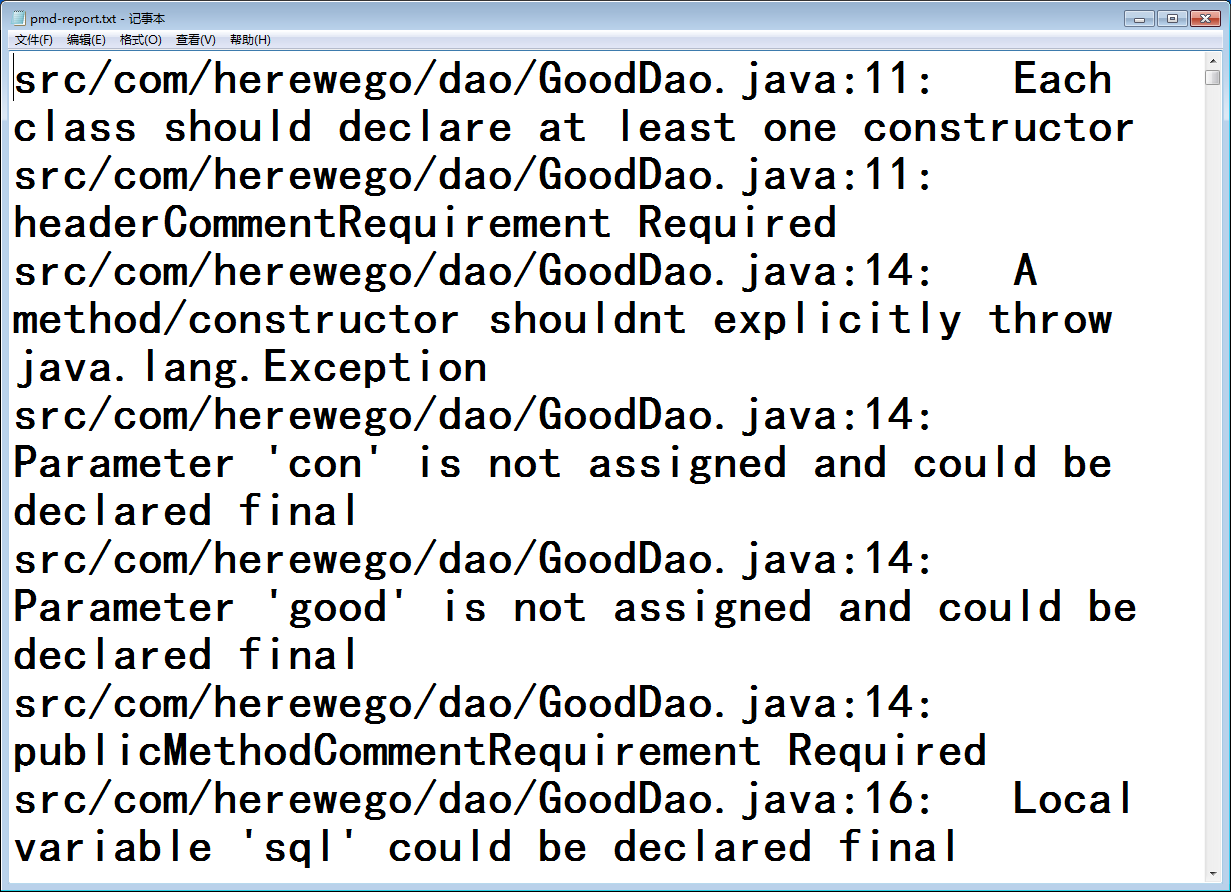


图 88 导出统计文本

CPD视图(显示各个类之间相同代码行数以及具体内容)：

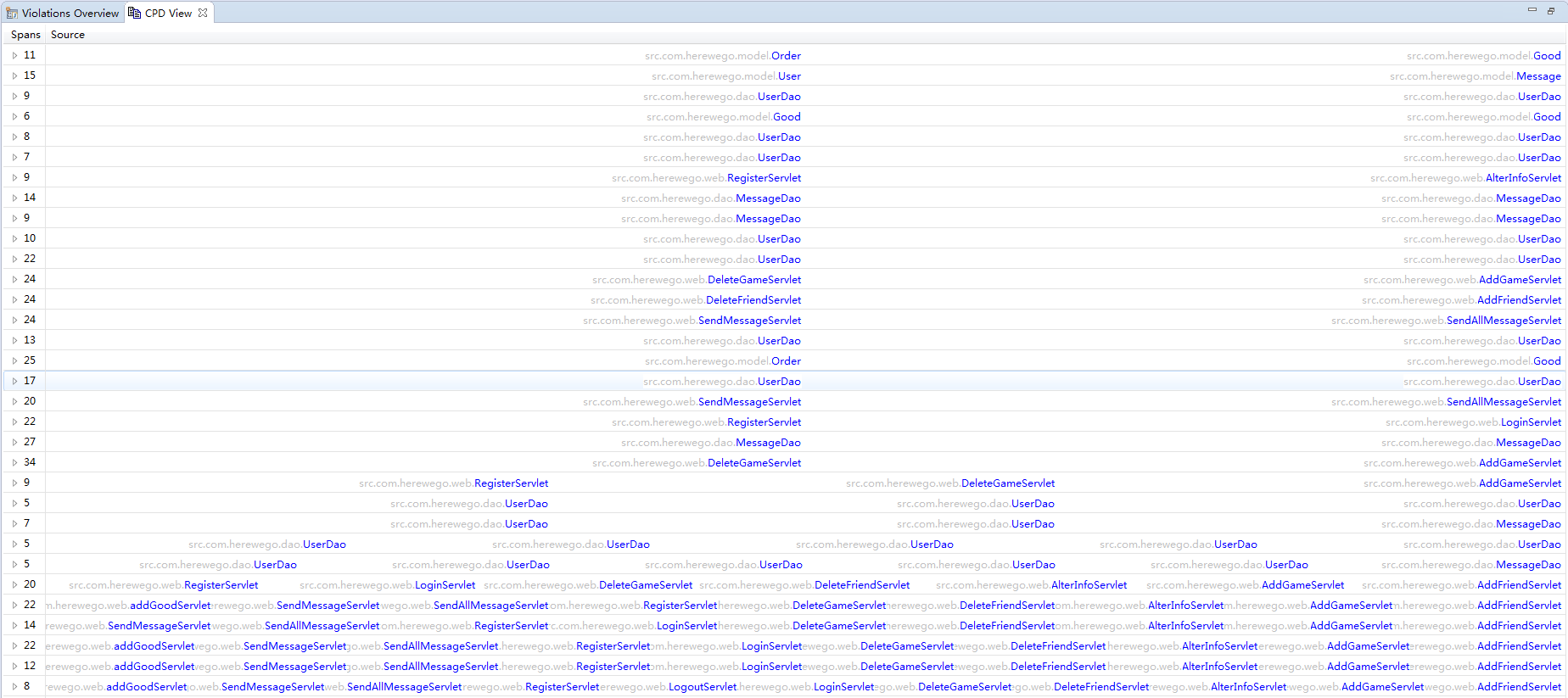


图 89 CPD视图

导出的CPD视图文本：

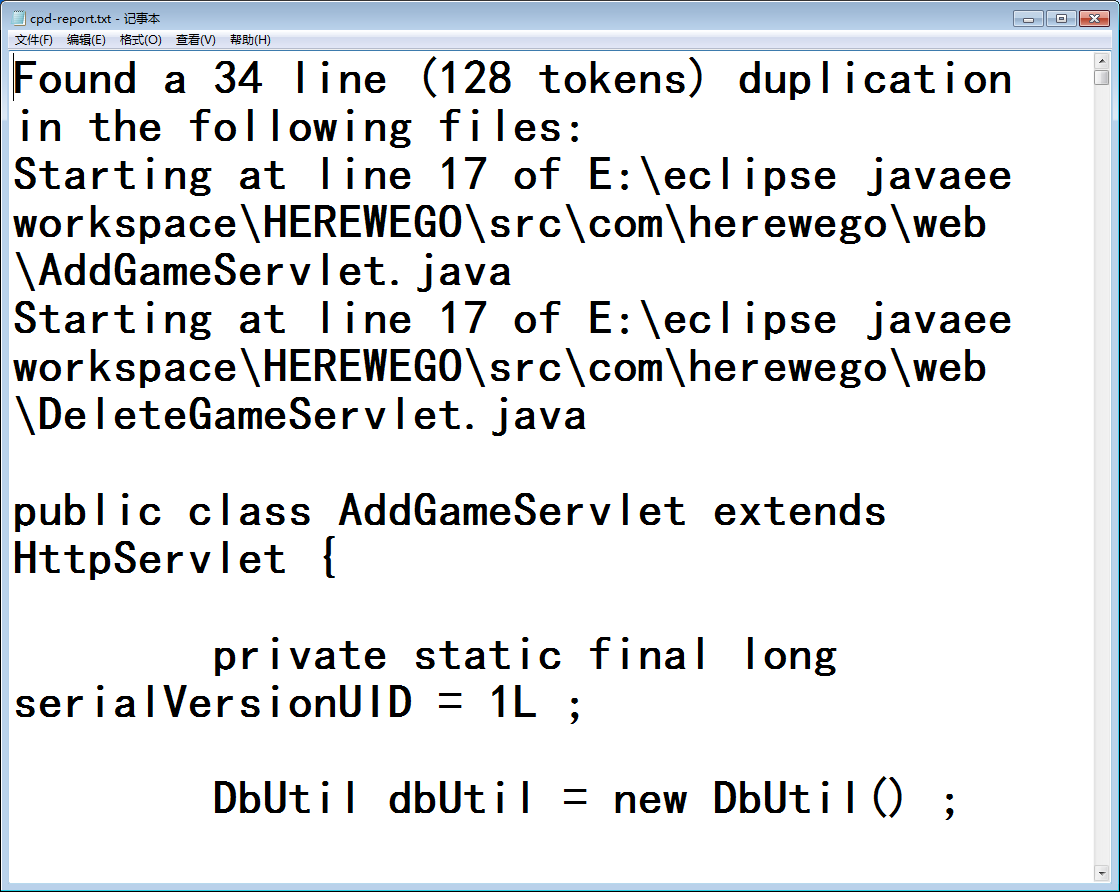


图 90 导出CPD

部分错误显示：

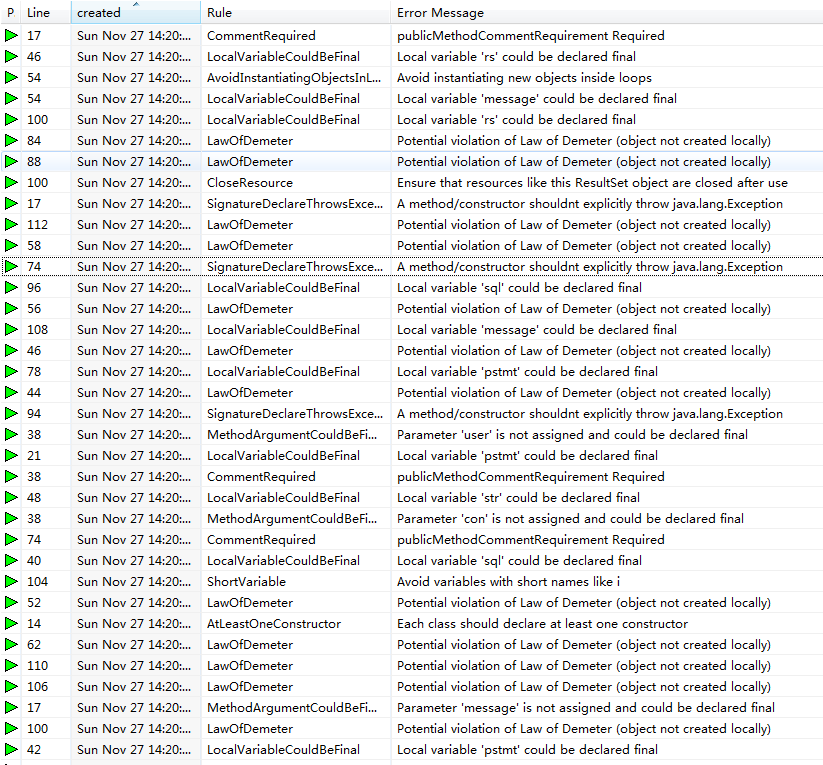


图 91 部分错误显示

（6）分析PMD以及CPD检查代码结果，可以看到各个类之间的相似代码很多，可能用于处理相同功能模块的代码复用性不高，可以提取出来封装成一个模块，每个类负责的功能太多，不符合编程的思想，应该尽肯能减少类的耦合。代码规范比如避免使用过短的变量命名，因为不太知道变量的含义，还有部分变量应定义为final，每个类至少拥有一个构造函数等等。需要对那些代码不规范过多的包或者类进行仔细检查，注意自己的代码书写。

#### 2.1.1.2 使用checkstyle进行代码检查

（1） CheckStyle是SourceForge下的一个项目，提供了一个帮助JAVA开发人员遵守某些编码规范的工具。它能够自动化代码规范检查过程，从而使得开发人员从这项重要，但是枯燥的任务中解脱出来。

（2） CheckStyle检验的主要内容

① Javadoc注释

1. 命名约定
2. 标题
3. Import语句
4. 体积大小
5. 空白
6. 修饰符
7. 块
8. 代码问题
9. 类设计

⑪ 混合检查(包括一些有用的比如非必须的System.out和printstackTrace)

从上面可以看出，CheckStyle提供了大部分功能都是对于代码规范的检查，而没有提供象PMD和Jalopy那么多的增强代码质量和修改代码的功能。但是，对于团队开发，尤其是强调代码规范的公司来说，它的功能已经足够强大。

（3） checkstyle基本设置

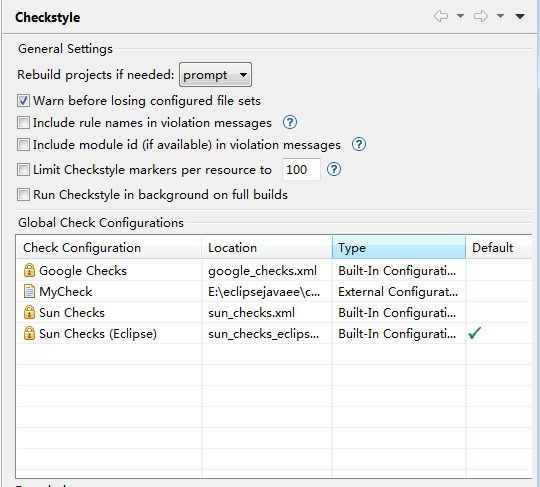


图 92 checkstyle设置

Checkstyle 部分配置详细信息：



图 93 部分配置

（4）以获取商品列表函数为例，显示代码审查结果如下：

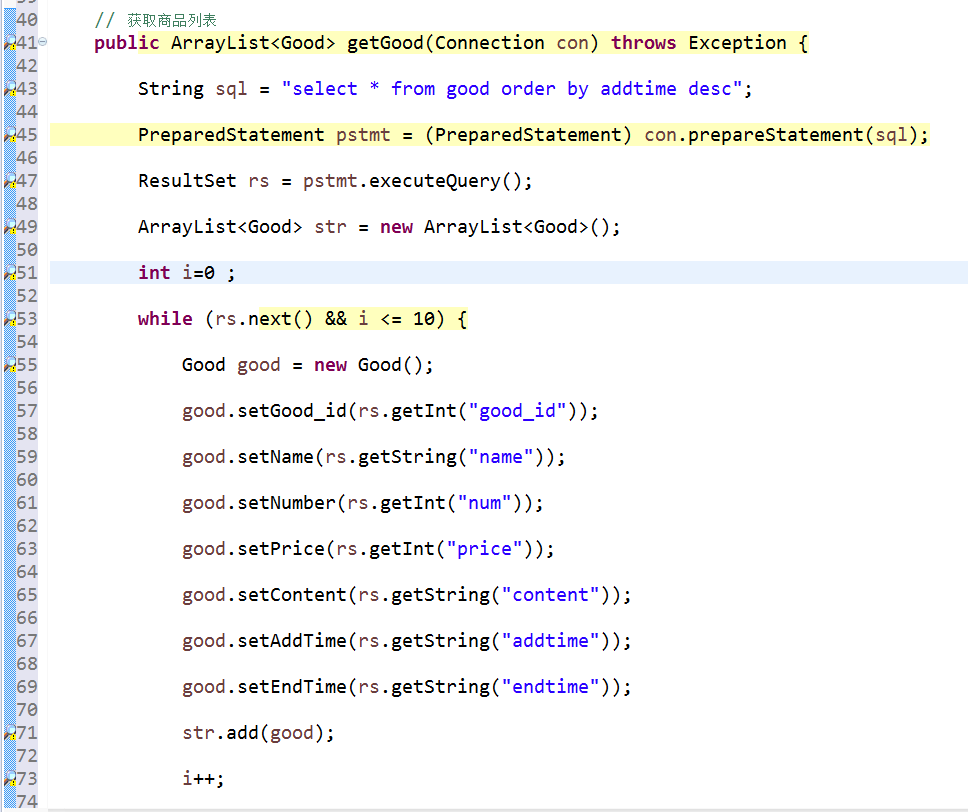


图 94 checkstyle代码审查结果

不规范代码详细信息如下：

Multiple markers at this line

- 参数： con 应定义为 final 的。

- 缺少 Javadoc 。

- '，' 前不应有空格。

- 参数： good 应定义为 final 的。

- '(' 后不应有空格。

- 参数： con 应定义为 final 的。

- ')' 前不应有空格。

- 本行字符数 85个，最多：80个。

- ';' 前不应有空格。

- '=' 前应有空格。

- '(' 后不应有空格。

- '<=' 前应有空格。

- ')' 前不应有空格。

- '10' 是一个魔术数字（直接常数）。

- '<=' 后应有空格。

（5） 针对checkstyle 所显示的内容分析，代码规范是一个程序员或者一个公司的自己的规范，我们可以自己设定自己的规范，写一个checkstyle.xml配置文件，应该增强注释的编写，养成良好的注释习惯。

#### 2.1.1.3 使用SourceMonitor进行代码检查

（1）SourceMonitor是一款免费的软件，运行在Windows平台下。它可对多种语言写就的代码进行度量，包括C、C++、C#、Java、VB、Delphi和HTML，并且针对不同的语言，输出不同的代码度量值。像其他代码度量工具一样，SourceMonitor只关注代码，并为编码人员提供及时的反馈，它不是一款项目管理工具，不关注项目实施中从功能分析到设计编码，再到测试这整个过程。

（2） SourceMonitor 统计结果如下

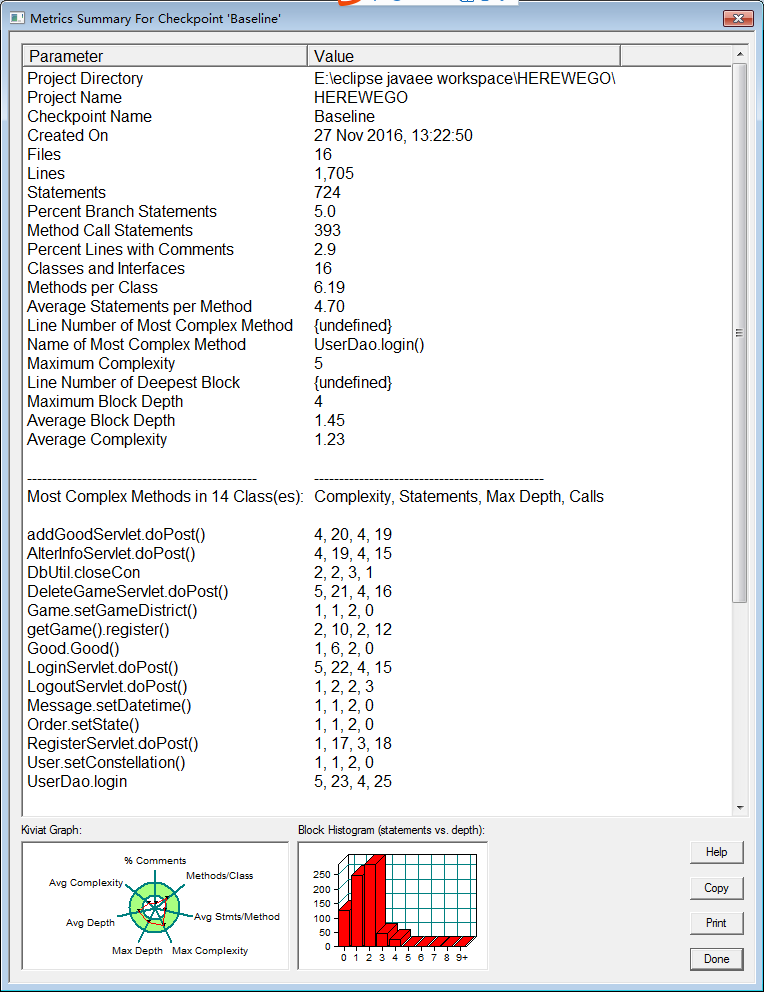


图 95 SourceMonitor 统计结果

上图为总的文件视图，显示了包括文件总数，总行数，代码行数，分支率，注释率，类和接口总数，最复杂方法，每类平均方法数，最大复杂度，最大块深度，平均快深度等各种信息显示，分析可以得出：

①注释率过低，程序员没有良好的写注释习惯，导致寥寥无几的注释，不利于后续维护人员或者自身代码维护。

1. 代码分之比率较小，复杂度最高的方法也没有多高，项目整体复杂度不高

③项目类和代码块的平均深度较低，方法功能解耦度高。

代码整体度量图：

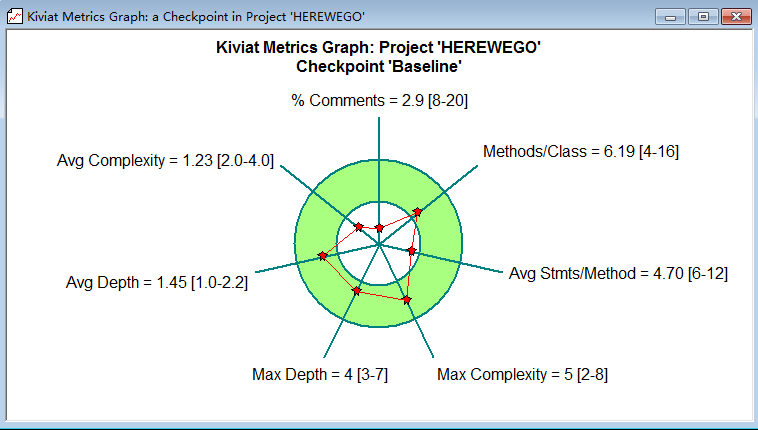


图 96代码整体度量图

单个java类的统计结果：

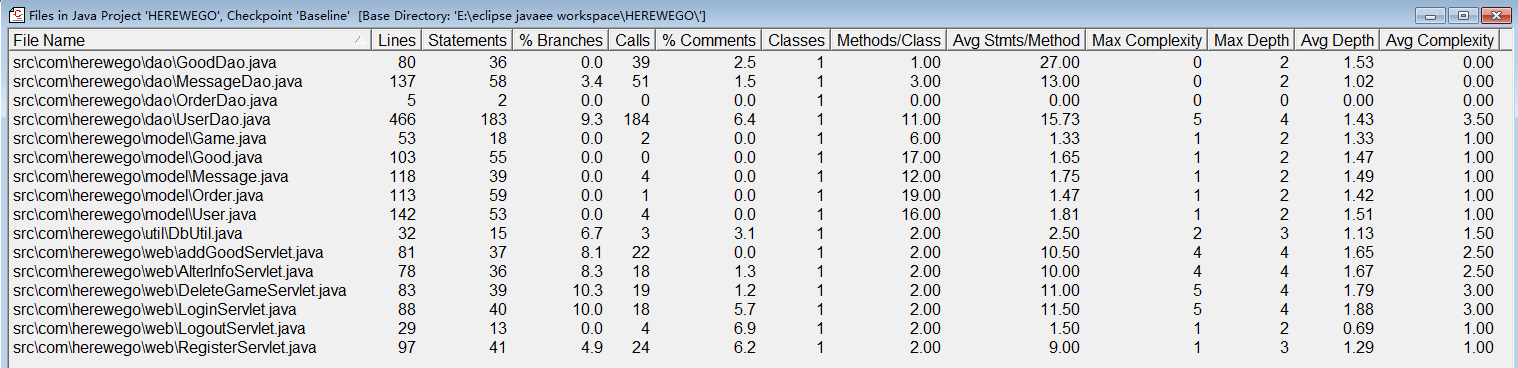


图 97单个java类的统计结果

方法复杂度统计结果：

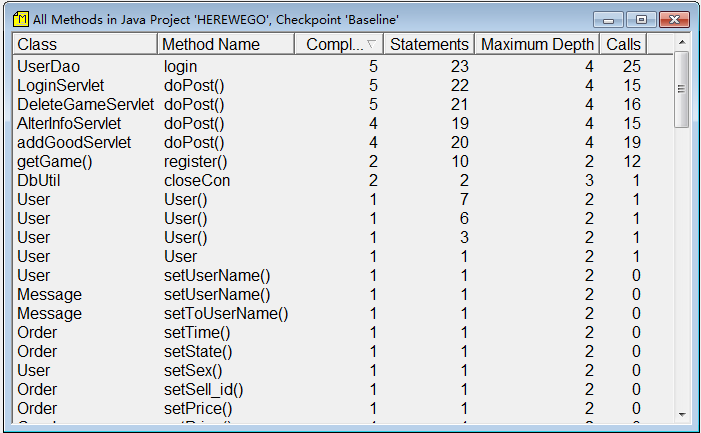


图 98方法复杂度统计结果

（3）据以上图表显示，某些方法与方法之间的调用过多，容易发生错误，测试人员应该对这些方法进行着重测试，类的总体复杂度不高，但是严重缺乏注释。

### 2.1.2单元测试的用例设计

#### 2.1.2.1 登录功能测试

所测试的源代码：

//登录功能

**public** User login ( Connection con ， User user ) **throws** Exception {

User resultUser = **null** ;

//查询是否存在对应的用户，密码

String sql = "select \* from user where user\_id = ? and user\_password = ? " ;

PreparedStatement pstmt = (PreparedStatement) con.prepareStatement(sql) ;

pstmt.setString(1， user.getUserName());

pstmt.setString(2， user.getPassword());

ResultSet rs = pstmt.executeQuery() ;

//从user\_information获取该用户的信息

String sql2 = "select \* from user\_information where user\_id = ? " ;

PreparedStatement pstmt2 = (PreparedStatement) con.prepareStatement(sql2) ;

pstmt2.setString(1， user.getUserName()) ;

ResultSet rs2 = pstmt2.executeQuery() ;

**if** (rs.next()) {

resultUser = **new** User() ;

resultUser.setUserName(rs.getString("user\_id"));

resultUser.setPassword(rs.getString("user\_password"));

**if** (rs2.next()) {

resultUser.setNickName(rs2.getString("user\_name"));

resultUser.setSex(rs2.getString("user\_sex"));

//获取得到的字符串转化为数值

resultUser.setAge(Integer.*parseInt*(rs2.getString("user\_age")));

resultUser.setConstellation(rs2.getString("user\_constellation"));

}

}

//如果输入用户名，密码一项为空则返回null

**if** (user.getUserName() == "" ) {

resultUser = **null** ;

}

**if** (user.getPassword() == "" ) {

resultUser = **null** ;

}

**return** resultUser ;

}

流程图：

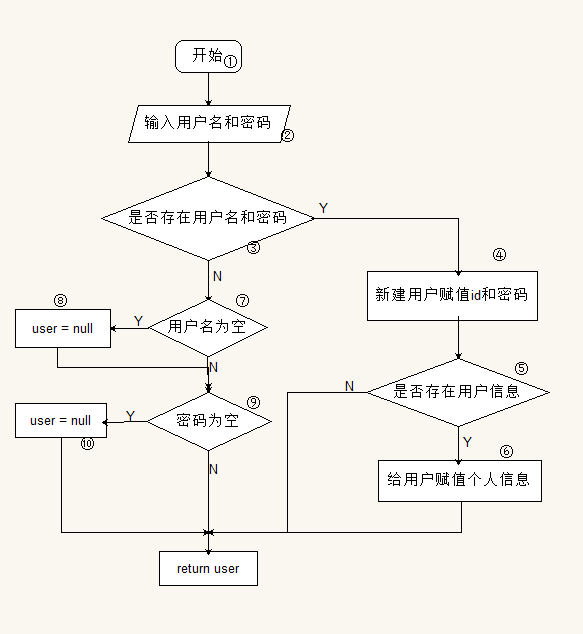


图 99 注册功能流程图

语句覆盖：

用例： 用户名：zcz 密码：123456

覆盖语句： ① ② ③ ④ ⑤ ⑥

条件： ③ = Y ， ⑤ = Y

路径： ① ② ③ ④ ⑤ ⑥

用例： 用户名： 密码：

覆盖语句： ① ② ③ ⑦ ⑧ ⑨ ⑩

条件： ③ = N ， ⑤ = N ， ⑦ = N ， ⑨ = N

路径： ① ② ③ ⑦ ⑧ ⑨ ⑩

分支覆盖 条件覆盖 判定条件覆盖：

用例： 用户名：zcz 密码：123456

覆盖语句： ① ② ③ ④ ⑤ ⑥

条件： ③ = Y ， ⑤ = Y

路径： ① ② ③ ④ ⑤ ⑥

用例： 用户名： 密码：

覆盖语句： ① ② ③ ⑦ ⑧ ⑨ ⑩

条件： ③ = N ， ⑤ = N ， ⑦ = Y ， ⑨ = Y

路径： ① ② ③ ⑦ ⑧ ⑨ ⑩

用例： 用户名：DFH 密码：12546

覆盖语句： ① ② ③ ⑦ ⑨

条件： ③ = N ， ⑤ = N ， ⑦ = N ， ⑨ = N

路径： ① ② ③ ⑦ ⑨

组合覆盖 路径覆盖：

用例： 用户名：zcz 密码：123456

覆盖语句： ① ② ③ ④ ⑤ ⑥

条件： ③ = Y ， ⑤ = Y

路径： ① ② ③ ④ ⑤ ⑥

用例： 用户名：zcz 密码：123456

覆盖语句： ① ② ③ ④ ⑤

条件： ③ = Y ， ⑤ = N

路径： ① ② ③ ④ ⑤

用例： 用户名： 密码：

覆盖语句： ① ② ③ ⑦ ⑧ ⑨ ⑩

条件： ③ = N ， ⑤ = N ， ⑦ = Y ， ⑨ = Y

路径： ① ② ③ ⑦ ⑧ ⑨ ⑩

用例： 用户名：DFH 密码：12546

覆盖语句： ① ② ③ ⑦ ⑨

条件： ③ = N ， ⑤ = N ， ⑦ = N ， ⑨ = N

路径： ① ② ③ ⑦ ⑨

用例： 用户名：gkkfdg 密码：

覆盖语句： ① ② ③ ⑦ ⑨ ⑩

条件： ③ = N ， ⑤ = N ， ⑦ = N ， ⑨ = Y

路径： ① ② ③ ⑦ ⑨ ⑩

用例： 用户名： 密码： gkkfdg

覆盖语句： ① ② ③ ⑦ ⑧ ⑨

条件： ③ = N ， ⑤ = N ， ⑦ = Y ， ⑨ = N

路径： ① ② ③ ⑦ ⑧ ⑨

综合测试用例：

用例： 用户名：zcz 密码：123456

用例： 用户名： 密码：

用例： 用户名：DFH 密码：12546

用例： 用户名：gkkfdg 密码：

用例： 用户名： 密码： gkkfdg

JUint测试代码：

**import** **static** org.junit.Assert.\*;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** com.herewego.dao.UserDao;

**import** com.herewego.model.User;

**public** **class** MyTest01 {

**private** Connection con ;

**private** String dbUrl = "jdbc：mysql：//localhost：3306/herewego" ;

**private** String dbUserName = "root" ;

**private** String dbPassword = "987654321+." ;

**private** String jdbcName = "com.mysql.jdbc.Driver" ;

**private** UserDao userDao = **new** UserDao();

**private** User user = **new** User() ;

**private** User current ;

@Before

**public** **void** setUp() **throws** Exception {

Class.*forName*(jdbcName) ;

con = DriverManager.*getConnection*(dbUrl， dbUserName， dbPassword) ;

}

@After

**public** **void** tearDown() **throws** Exception {

con = **null** ;

}

@Test

**public** **void** testLogin() **throws** Exception {

user.setUserName("zcz");

user.setPassword("123456");

current = userDao.login(con， user) ;

*assertEquals*( "zcz" ， current.getUserName() );

*assertEquals*( "123456" ， current.getPassword() );

user.setUserName("");

user.setPassword("");

current = userDao.login(con， user) ;

*assertEquals*(**null**， current);

user.setUserName("DFH");

user.setPassword("12546");

current = userDao.login(con， user) ;

*assertEquals*(**null**， current);

user.setUserName("gkkfdg");

user.setPassword("");

current = userDao.login(con， user) ;

*assertEquals*(**null**， current);

user.setUserName("");

user.setPassword("gkkfdg");

current = userDao.login(con， user) ;

*assertEquals*(**null**， current);

}

}

JUint测试结果：

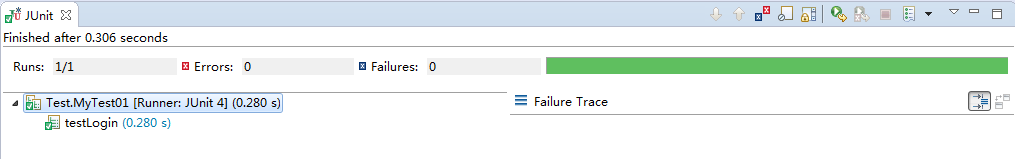


图 100 单元测试一截图

#### 2.1.2.2 获取商品列表功能测试

所测试的源代码：

// 获取商品列表

**public** ArrayList<Good> getGood(Connection con) **throws** Exception {

String sql = "select \* from good order by addtime desc";

PreparedStatement pstmt = (PreparedStatement) con.prepareStatement(sql);

ResultSet rs = pstmt.executeQuery();

ArrayList<Good> str = **new** ArrayList<Good>();

**int** i=0 ;

**while** (rs.next() && i <= 10) {

Good good = **new** Good();

good.setGood\_id(rs.getInt("good\_id"));

good.setName(rs.getString("name"));

good.setNumber(rs.getInt("num"));

good.setPrice(rs.getInt("price"));

good.setContent(rs.getString("content"));

good.setAddTime(rs.getString("addtime"));

good.setEndTime(rs.getString("endtime"));

str.add(good);

i++;

}

**return** str;

}

}

流程图：

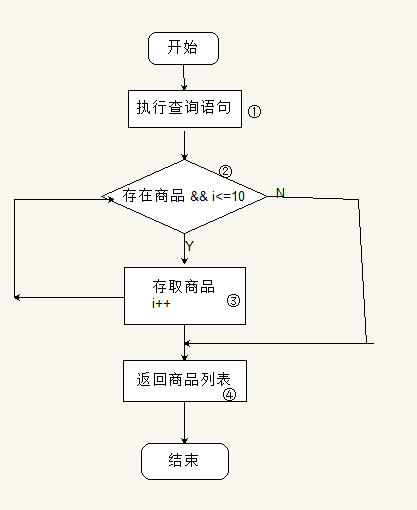


图 101 获取商品列表流程图

语句覆盖：

用例： good表存在数据

覆盖语句： ① ② ③ ④

条件： ② = Y

路径： ① ② ③ ④

分支覆盖 条件覆盖 判定条件覆盖 组合覆盖 路径覆盖：

用例： good表存在数据大于十条

覆盖语句： ① ② ③ ④

条件： ② = Y

路径： ① ② ③ ④

综合测试用例：

用例： good表存在数据大于十条

JUint测试代码：

**import** **static** org.junit.Assert.\*;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.util.ArrayList;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** com.herewego.dao.GoodDao;

**import** com.herewego.model.Good;

**public** **class** MyTest02 {

**private** Connection con ;

**private** String dbUrl = "jdbc：mysql：//localhost：3306/herewego" ;

**private** String dbUserName = "root" ;

**private** String dbPassword = "987654321+." ;

**private** String jdbcName = "com.mysql.jdbc.Driver" ;

**private** GoodDao goodDao = **new** GoodDao();

**private** Good good ;

**private** ArrayList<Good> arr ;

@Before

**public** **void** setUp() **throws** Exception {

Class.*forName*(jdbcName) ;

con = DriverManager.*getConnection*(dbUrl， dbUserName， dbPassword) ;

}

@After

**public** **void** tearDown() **throws** Exception {

con = **null** ;

}

@Test

**public** **void** testGetGood() **throws** Exception {

arr = goodDao.getGood(con) ;

good = arr.get(0) ;

*assertEquals*(2， good.getGood\_id());

*assertEquals*("剑"， good.getName());

*assertEquals*(1， good.getNumber());

*assertEquals*("无"， good.getContent() );

}

}

JUint测试结果：

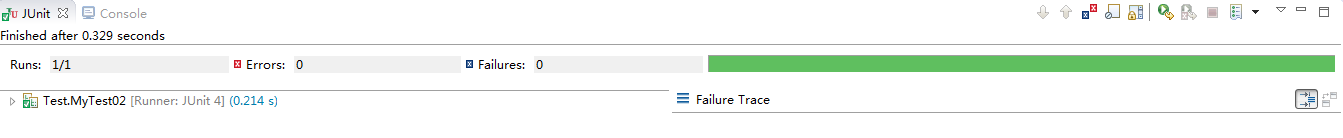


图 102 单元测试二截图

#### 2.1.2.3 获取推荐好友功能测试

所测试的源代码：

//获取推荐好友

**public** ArrayList<String> getRecFriends(Connection con， User user) **throws** Exception {

String sql = "select \* from user\_game where user\_id = ? " ;

PreparedStatement pstmt = (PreparedStatement) con.prepareStatement(sql) ;

pstmt.setString(1， user.getUserName());

ResultSet rs = pstmt.executeQuery() ;

**if** ( rs.next() ) {

Game game = **new** Game() ;

game.setGameName(rs.getString("user\_game"));

game.setGameDistrict(rs.getString("user\_district"));

String sql2 = "select \* from user\_game where user\_game = ? and user\_district = ? " ;

PreparedStatement pstmt2 = (PreparedStatement) con.prepareStatement(sql2) ;

pstmt2.setString(1， game.getGameName());

pstmt2.setString(2， game.getGameDistrict());

ResultSet rs2 = pstmt2.executeQuery() ;

ArrayList<String> str = **new** ArrayList<String>() ;

**int** i = 0 ;

**while** ( rs2.next() && i <= 5 ) {

str.add(rs2.getString("user\_id")) ;

i ++ ;

}

str.remove(user.getUserName()) ;

**return** str ;

} **else** {

**return** **null** ;

}

}

}

流程图：

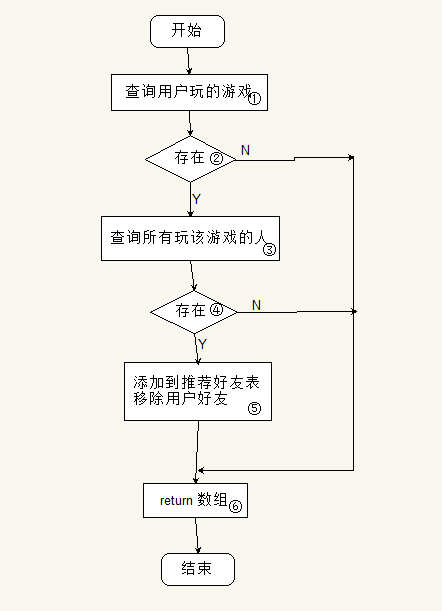


图 103 获取推荐好友流程图

语句覆盖：

用例： 用户游戏不为空 ， 玩该游戏的人不为空

覆盖语句： ① ② ③ ④ ⑤ ⑥

条件： ② = Y ， ④ = Y

路径： ① ② ③ ④ ⑤ ⑥

分支覆盖 条件覆盖 判定条件覆盖 路径覆盖 组合覆盖：

用例： 用户游戏不为空 ， 玩该游戏的人不为空

覆盖语句： ① ② ③ ④ ⑤ ⑥

条件： ② = Y ， ④ = Y

路径： ① ② ③ ④ ⑤ ⑥

用例： 用户游戏为空 ， 玩该游戏的人不为空

覆盖语句： ① ② ⑥

条件： ② = N ， ④ = Y

路径： ① ② ⑥

用例： 用户游戏不为空 ， 玩该游戏的人为空

覆盖语句： ① ② ③ ④ ⑥

条件： ② = Y ， ④ = N

路径： ① ② ③ ④ ⑥

用例： 用户游戏为空 ， 玩该游戏的人为空

覆盖语句： ① ② ⑥

条件： ② = N ， ④ = N

路径： ① ② ⑥

综合测试用例：

用例： 用户游戏不为空 ， 玩该游戏的人不为空

用例： 用户游戏为空 ， 玩该游戏的人不为空

用例： 用户游戏不为空 ， 玩该游戏的人为空

用例： 用户游戏为空 ， 玩该游戏的人为空

JUint测试代码：

**import** **static** org.junit.Assert.\*;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.util.ArrayList;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** com.herewego.dao.UserDao;

**import** com.herewego.model.User;

**public** **class** MyTest03 {

**private** Connection con ;

**private** String dbUrl = "jdbc：mysql：//localhost：3306/herewego" ;

**private** String dbUserName = "root" ;

**private** String dbPassword = "987654321+." ;

**private** String jdbcName = "com.mysql.jdbc.Driver" ;

**private** UserDao userDao = **new** UserDao();

**private** User user = **new** User() ;

**private** ArrayList<String> arr ;

@Before

**public** **void** setUp() **throws** Exception {

Class.*forName*(jdbcName) ;

con = DriverManager.*getConnection*(dbUrl， dbUserName， dbPassword) ;

}

@After

**public** **void** tearDown() **throws** Exception {

con = **null** ;

}

@Test

**public** **void** testGetRecFriends() **throws** Exception {

//用户游戏不为空 ， 玩该游戏的人不为空

user.setUserName("zcz");

arr = userDao.getRecFriends(con， user) ;

*assertEquals*("yhx"， arr.get(0));

//用户游戏为空 ， 玩该游戏的人不为空

user.setUserName("all");

arr = userDao.getRecFriends(con， user) ;

*assertNull*( arr );

//用户游戏不为空 ， 玩该游戏的人为空

user.setUserName("11111");

arr = userDao.getRecFriends(con， user) ;

*assertEquals*( 0 ， arr.size() );

//用户游戏为空 ， 玩该游戏的为空

user.setUserName("all");

arr = userDao.getRecFriends(con， user) ;

*assertNull*( arr );

}

}

JUint测试结果：

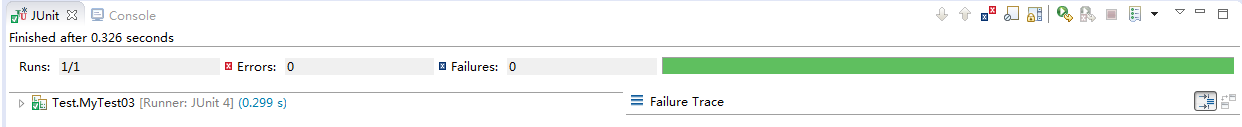


图 104 单元测试三截图

#### 2.1.2.4 添加好友功能测试

所测试的源代码：

//添加好友

**public** **boolean** addFriend ( Connection con ， User user ， User targetUser ) **throws** Exception {

String sql = "select \* from user\_friends where user\_id = ? and user\_friend = ? " ;

PreparedStatement pstmt = (PreparedStatement) con.prepareStatement(sql) ;

pstmt.setString(1， user.getUserName());

pstmt.setString(2， targetUser.getUserName());

ResultSet rs = pstmt.executeQuery() ;

**if** (!rs.next()) {

String insertsql = " insert into user\_friends values(?，?) " ;

PreparedStatement registerpstmt = (PreparedStatement) con.prepareStatement(insertsql) ;

registerpstmt.setString(1， user.getUserName());

registerpstmt.setString(2， targetUser.getUserName());

registerpstmt.execute() ;

**return** **true** ;

}

**else**

**return** **false** ;

}

流程图：

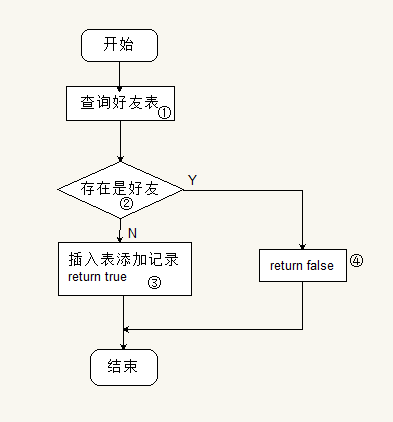


图 105 添加好友流程图

语句覆盖 分支覆盖 条件覆盖 判定条件覆盖 路径覆盖 组合覆盖：

用例： 好友表存在该记录

覆盖语句： ① ② ③

条件： ② = Y

路径： ① ② ③

用例： 好友表不存在该记录

覆盖语句： ① ② ④

条件： ② = N

路径： ① ② ④

综合测试用例：

用例： 好友表存在该记录

用例： 好友表不存在该记录

JUint测试代码：

**import** **static** org.junit.Assert.\*;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** com.herewego.dao.UserDao;

**import** com.herewego.model.User;

**public** **class** MyTest04 {

**private** Connection con ;

**private** String dbUrl = "jdbc：mysql：//localhost：3306/herewego" ;

**private** String dbUserName = "root" ;

**private** String dbPassword = "987654321+." ;

**private** String jdbcName = "com.mysql.jdbc.Driver" ;

**private** UserDao userDao = **new** UserDao();

**private** User user = **new** User() ;

**private** User targetUser = **new** User() ;

**private** **boolean** b ;

@Before

**public** **void** setUp() **throws** Exception {

Class.*forName*(jdbcName) ;

con = DriverManager.*getConnection*(dbUrl， dbUserName， dbPassword) ;

}

@After

**public** **void** tearDown() **throws** Exception {

con = **null** ;

}

@Test

**public** **void** testAddFriends() **throws** Exception {

//好友表存在该记录

user.setUserName("zcz");

targetUser.setUserName("11111");

b = userDao.addFriend(con， user， targetUser) ;

*assertFalse*(b);

//好友表不存在该记录

user.setUserName("zcz");

targetUser.setUserName("yhx");

b = userDao.addFriend(con， user， targetUser) ;

*assertTrue*(b);

}

}

JUint测试结果：

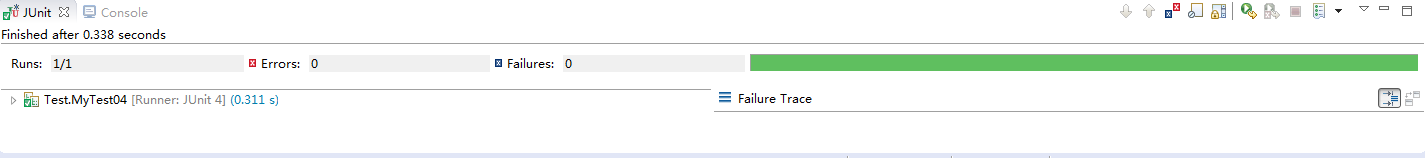


图 106 单元测试四截图

## 2.2 集成测试的用例设计

### 2.2.1 测试场景

集成测试主要测试以下模块：用户注册模块，用户登录模块，用户修改信息模块，用户添加好友模块，用户删除好友模块，用户私聊模块，用户公聊模块，用户添加游戏模块，用户删除游戏模块，用户上架商品模块，用户购买商品模块

|  |  |
| --- | --- |
| 编号 | 模块 |
| A | 用户注册模块 |
| B | 用户登录模块 |
| C | 用户修改信息模块 |
| D | 用户添加好友模块 |
| E | 用户删除好友模块 |
| F | 用户私聊模块 |
| G | 用户公聊模块 |
| H | 用户添加游戏模块 |
| I | 用户删除游戏模块 |
| J | 用户上架商品模块 |
| K | 用户购买商品模块 |
| L | 用户查看订单模块 |
| M | 用户删除订单模块 |

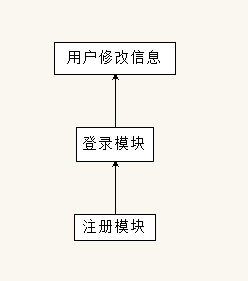
表 107 集成测试编号

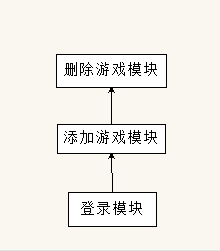
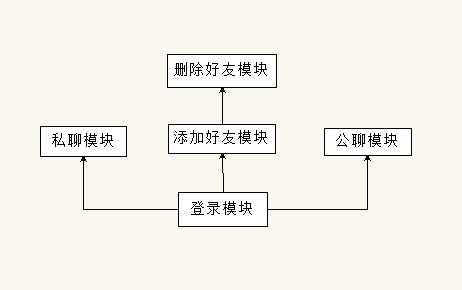
集成测试的方式为集成前端web页面进行模块间的组合测试。

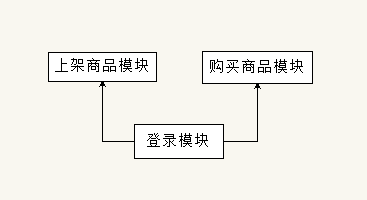
输入：

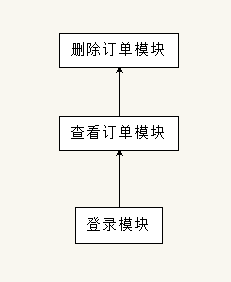
1. 软件代码
2. 测试计划
3. 概要设计说明书

所测试的模块结构如下图：









图组108 模块结构图

### 2.2.2 测试方法

本网站采用自底向上的集成测试方法。自底向上的集成（Bottom-Up Integration）方式是最常使用的方法。其他集成方法都或多或少地继承、吸收了这种集成方式的思想。自底向上集成方式从程序模块结构中最底层的模块开始组装和测试。因为模块是自底向上进行组装的，对于一个给定层次的模块，它的子模块（包括子模块的所有下属模块）事前已经完成组装并经过测试，所以不再需要编制桩模块（一种能模拟真实模块，给待测模块提供调用接口或数据的测试用软件模块）。自底向上集成测试的步骤大致如下：

步骤一： 按照概要设计规格说明，明确有哪些被测模块。在熟悉被测模块性质的基础上对被测模块进行分层，在同一层次上的测试可以并行进行，然后排出测试活动的先后关系，制定测试进度计划。

步骤二： 在步骤一的基础上，按时间线序关系，将软件单元集成为模块，并测试在集成过程中出现的问题。这里，可能需要测试人员开发一些驱动模块来驱动集成活动中形成的被测模块。对于比较大的模块，可以先将其中的某几个软件单元集成为子模块，然后再集成为一个较大的模块。

步骤三： 将各软件模块集成为子系统（或分系统）。检测各自子系统是否能正常工作。同样，可能需要测试人员开发少量的驱动模块来驱动被测子系统。

步骤四： 将各子系统集成为最终用户系统，测试是否存在各分系统能否在最终用户系统中正常工作。

自底向上测试：是从程序的底层模块开始测试。

* 1. I/O操作可以提前测试，更好提交测试用例。
  2. 测试后比较容易观察输出。
  3. 需要开发驱动模块。
  4. 直到最后一个模块提交，程序才能完整的系统测试。

### 2.2.3 测试用例

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 模块 | 测试步骤 | 期望的结果 | 说明 |
| Integration Testing 01 | 1. B | 1.用户注册帐号  2.用户使用注册的帐号密码进行登录 | 用户登陆成功 | 将登录注册模块结合起来测试，保证用户帐号服务的可用性 |
| Integration Testing 02 | B，C | 1.用户登录账号  2.用户查看个人信息  3.用户对个人信息进行修改 | 用户信息被成功修改，刷新后显示新的信息 | 对用户友善的信息修改功能，确保用户可以对自己的信息完善以及修改 |
| Integration Testing 03 | B，D，E | 1.用户登录账号  2.用户查看个人好友  3.用户添加与删除好友 | 能够添加已经存在的用户，对不存在的用户友好提示，对已经是好友的用户能够删除 | 确保用户拥有自己的好友系统，能够对好友进行管理 |
| Integration Testing 04 | B，F，G | 1.用户登录账号  2.用户对好友和非好友发送私聊  3.用户对当前所有用户公屏聊天 | 发送消息成功，对方能够收到信息 | 确保用户能够对其他用户进行交互，增进与用户之间的关系 |
| Integration Testing 05 | B，H，I | 1.用户登录账号  2.用户查看个人游戏  3.用户添加与删除游戏 | 增加删除游戏成功，对没添加的游戏删除友好提示 | 确保用户能够接收相同兴趣爱好的推荐好友，共同话题更能激起网站的功能 |
| Integration Testing 06 | B，J，K | 1.用户登录账号  2.用户选择出售商品  3.用户查看那些已经出售商品的详细信息  4.用户购买中意的商品 | 上架商品成功，显示信息无误，能够购买那些中意的商品成为自己的物品 | 提供良好的物品交易平台，优化线下交易的弊端，用户与用户之间的交互 |
| Integration Testing 07 | B，L，M | 1.用户登录账号  2.用户选择查看订单列表  3.用户查看订单详细信息  4.用户删除个人订单 | 查看订单成功，能够显示用户购买过商品的订单，用户出售过商品的订单和出售中商品的订单，并且用户能对订单进行删除 | 提供良好的物品交易订单显示，提高对用户的友好性，提供用户期望的信息显示 |

表 109 集成测试测试用例表

### 2.2.4 测试过程与测试结果

Integration Testing 01 ：

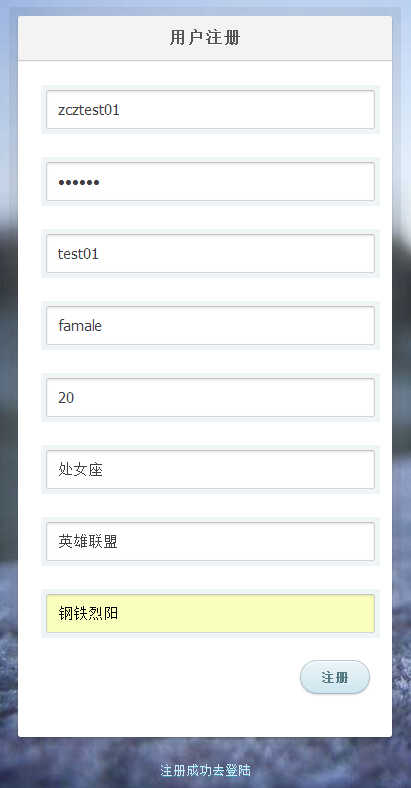


图 110 注册信息填写



图 111 注册成功



图 112 登录信息填写



图 113 登陆成功，显示用户信息

结果分析：达到预期结果，测试通过。

Integration Testing 02 ：

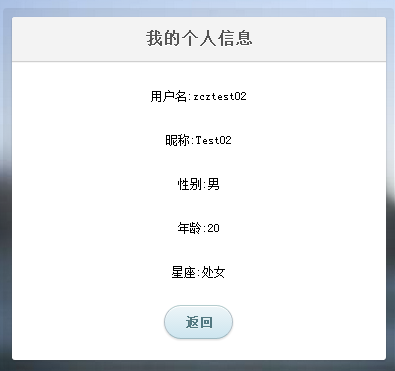


图 114 个人信息显示



图 115 修改信息填写



图 116 修改成功显示

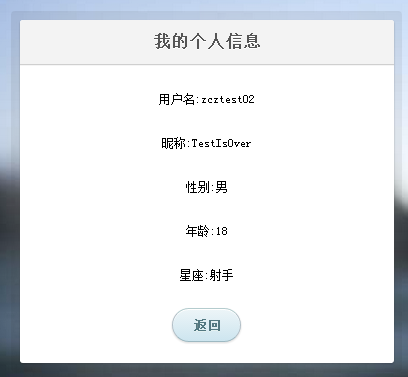


图 117 修改成功新的信息显示

结果分析：达到预期结果，测试通过。

Integration Testing 03 ：

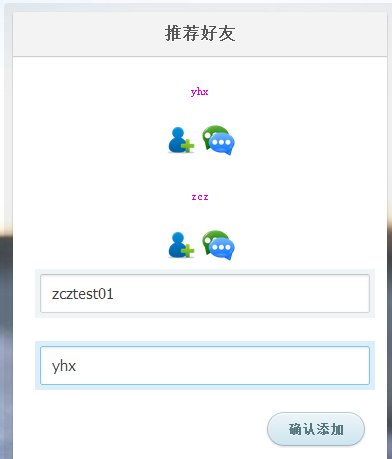


图 118 添加好友填写



图 119 添加成功显示



图 120 删除好友填写



图 121 删除成功显示

结果分析：达到预期结果，测试通过。

Integration Testing 04 ：



图 122 发送私聊填写



图 123 发送私聊成功显示



图 124 发送公聊填写



图 125 发送公聊成功显示



图 126 私聊显示



图 127 公聊显示

结果分析：达到预期结果，测试通过。

Integration Testing 05 ：



图 128 添加游戏填写



图 129 添加成功显示



图 130 游戏显示

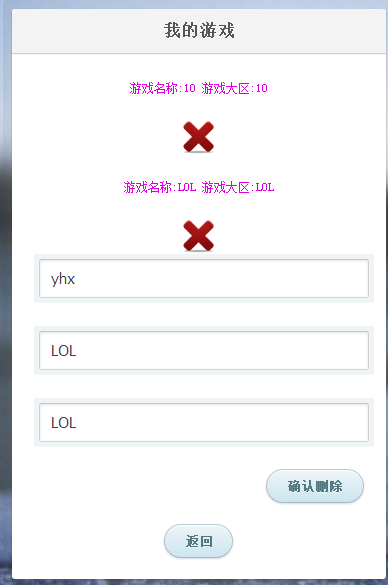


图 131 删除游戏填写



图 132 删除游戏成功显示



图 133 异常删除显示

结果分析：达到预期结果，测试通过。

Integration Testing 06 ：

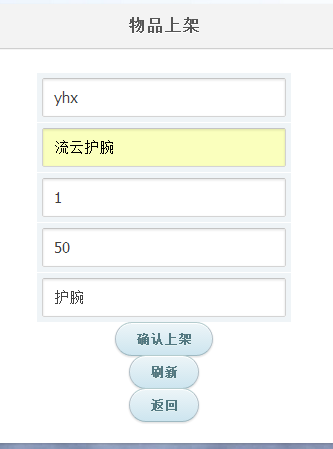


图 134 上架商品填写



图 135 上架成功显示



图 136 商品列表显示



图 137 异常查看显示



图 138 正常查看商品信息

结果分析：达到预期结果，测试通过。

Integration Testing 07 ：



图 139 各种订单显示



图 140 订单详细信息显示

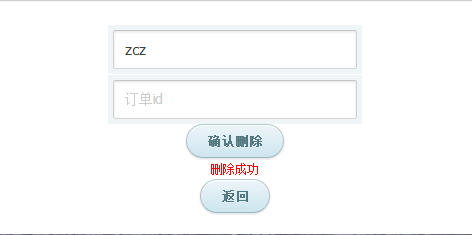


图 141 删除订单

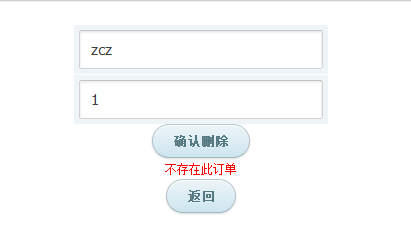


图 142 异常删除

结果分析：达到预期结果，测试通过。

## 2.3 非功能测试的用例设计

（1）测试场景

验证程序在承受某种负载或压力下是否能够正常运行，找出程序安全运行的临界值

（2）压力/负载分类与代号

CPU速度CS，CPU使用量CU，磁盘空间DS，物理内存PM

虚拟内存使用量VM，登录用户的数量UN，传输带宽BW，传输数据的大小DN

（3）测试用例

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 编号 | CS | CU | DS | PM | VM | UN | BW | DN | 应该产生的行为 | 结果 | 说明 |
| Test01 | Intel(R) Xeon(R) CPU E5-2430 0 @ 2.20GHz | 20% | 40GB | 512MB | 20% | 30 | 1M | 512M | 正常 | 运行正常 | 阿里云ECS |
| Test02 | Intel(R) Core(TM) i7-3632QM CPU @ 2.20GHz | 87% | 512GB | 8GB | 20% | 100 | 20M | 10M | 正常 | 运行正常 | 个人PC电脑 |
| Test03 | AMD A10-7850K FM2+3.7GHz | 50% | 256GB | 4GB | 30% | 60 | 5M | 10M | 正常 | 运行正常 | 个人PC电脑 |

表 143 非功能测试测试用例

## 2.4 自动化测试工具非功能测试

（1）测试工具

Badboy脚本录制工具和Apache jmeter性能测试工具

（2）测试过程：



图 144 使用badboy录制脚本

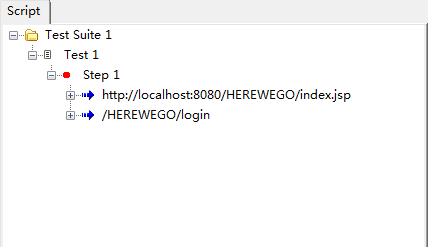


图 145 录制的脚本

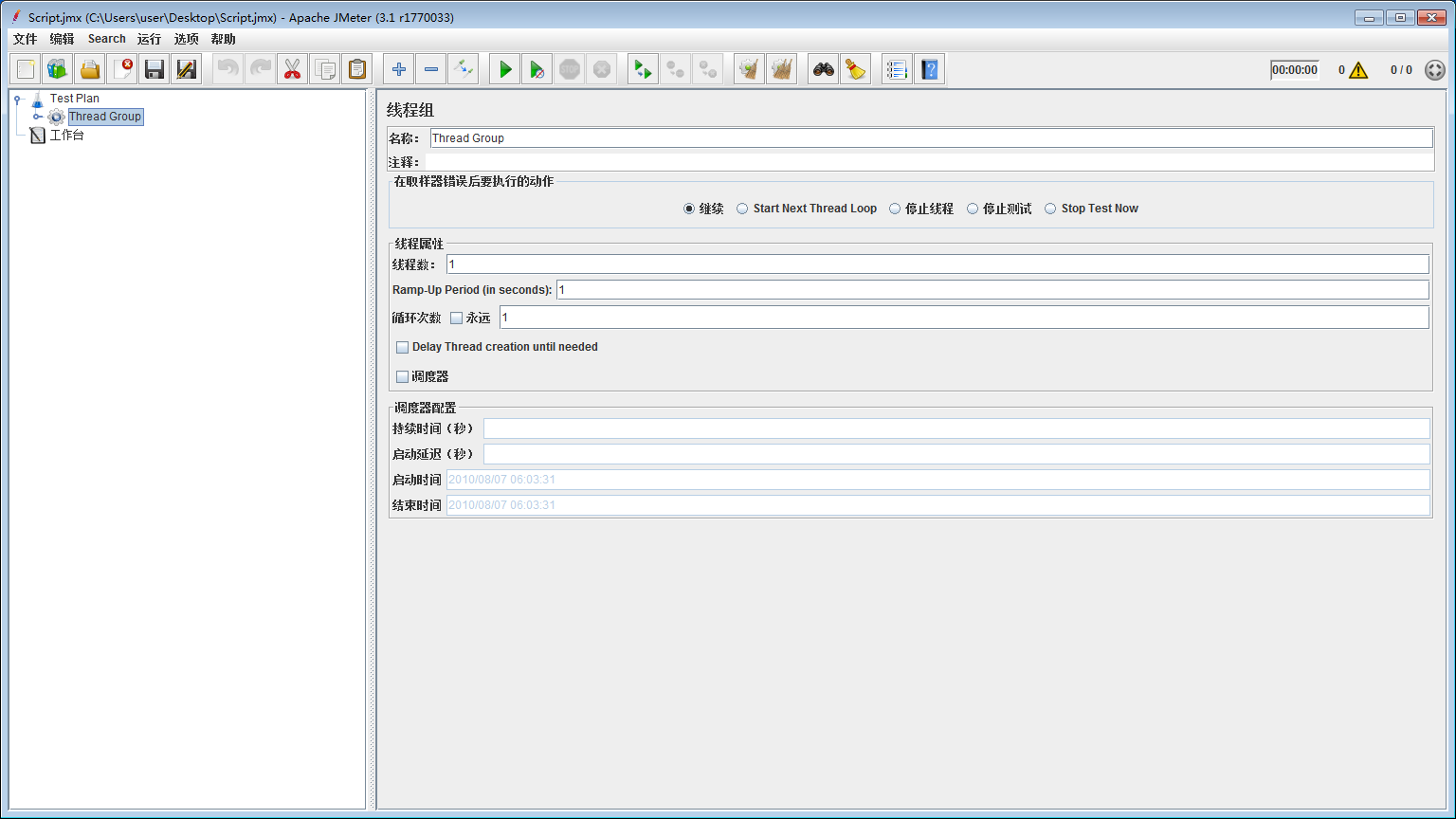


图 146 jmeter测试

（3）名词解释：

线程数——设置测试开启的总线程的数量

Ramp-Up Period（in seconds）——完成此次测试的总时间

循环次数——每个线程请求的次数

样本数目——模拟请求的总次数（=线程数\*循环次数）

最新样本——最近的一次请求的时间开销，单位ms

平均————运行到当前线程的平均请求时长，等于总的运行时间除以发送到服务器的请求数目

偏离————服务器响应变化的时间，数值越小服务器越稳定

吞吐量———服务器在每分钟内处理的请求数量，数值越大效率越高

中值————有一半的服务器响应时间低于该值而另一半高于该值

（4）首页测试结果及其分析：

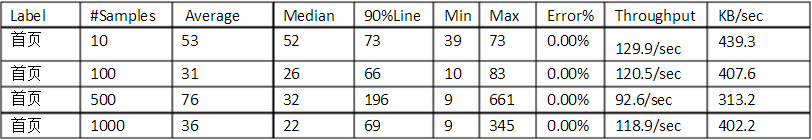


表 147 首页测试结果

分析：#samples模拟1000个用户时的压力测试，平均响应时间是0.036秒，百分之九十的用户响应时间为0.069秒，没有无法响应的请求。

3.测试总结

## 3.1 测试用例执行结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 测试用例标识符 | 测试用例名称 | 状态 | 测试结果 | 备注 |
| 单元测试 | | | | |
| Testcase001 | 登录功能 | 已执行 | 测试通过 |  |
| Testcase002 | 获取商品列表功能 | 已执行 | 测试通过 |  |
| Testcase003 | 获取推荐好友 | 已执行 | 测试通过 |  |
| Testcase004 | 添加好友 | 已执行 | 测试通过 |  |
| 集成测试 | | | | |
| Integration Testing 01 | 注册登录模块 | 已执行 | 测试通过 |  |
| Integration Testing 02 | 用户信息模块 | 已执行 | 测试通过 |  |
| Integration Testing 03 | 好友模块 | 已执行 | 测试通过 |  |
| Integration Testing 04 | 聊天模块 | 已执行 | 测试通过 |  |
| Integration Testing 05 | 游戏模块 | 已执行 | 测试通过 |  |
| Integration Testing 06 | 商品模块 | 已执行 | 测试通过 |  |
| Integration Testing 07 | 订单模块 | 已执行 | 测试通过 |  |
| 非功能性测试 | | | | |
| Test01 | 大数据量测试 | 已执行 | 测试通过 | 性能测试 |
| Test02 | 负载测试 | 已执行 | 测试通过 |
| Test03 | 压力测试 | 已执行 | 测试通过 |

表 148 测试用例执行结果

## 3.2 测试问题解决

|  |  |  |
| --- | --- | --- |
| 测试用例标识符 | 错误或问题描述 | 错误或问题状态 |
| 单元测试 | | |
| Testcase001 | 用户名密码为空报错 | 已解决，提供友善提示，不提交表单 |
| 集成测试 | | |
| Integration Testing 06 | 商品编号问题，自增结果不对 | 已解决，重新自增，数据库自增，服务端不提交编号 |
| 性能测试 | | |

表 149 测试问题解决

## 3.3 测试结果分析

### 3.3.1 单元测试结果分析

单元测试Testcase001失败，用户名密码的准确性关系到网站的安全以及用户的信息安全，需要对这些字段进行多次的判断操作。

### 3.3.2 集成测试结果分析

每个用户以及订单都应该有唯一的编号，而这些编号应该是所有用户共用的，不应该对这个编号的随机性以及自增性在应用端做出干涉，应该由服务器或者数据库服务器自生成。

### 3.3.3 非功能测试结果分析

所有测试用例以及自动化测试工具均达到预期效果。

4.综合评价

## 4.1软件能力

经过单元测试、集成测试、功能测试和非功能测试后，黑we购网站已达到专业实训课的大作业要求和目标。

经过本项目的所有成员的通力合作，黑we购网站项目如期完成并达到交付的目标，该网站能够基本实现《软件需求规格说明书》中所规定的功能要求，即能够满足用户进行注册，登录，修改信息，添加删除好友，添加删除个人游戏，向好友和非好友发送私聊和公聊，上架商品进行交易并购买商品和个人订单的一系列功能，部分功能的实现有些小的问题和偏差，如关于用户信息太基本，用户难以找到一个个人主页的感觉，但这些问题并不影响系统的整体功能，总的来说，这个网站达到了预期要求，测试结果良好，可以进行初期版本的交付。

## 4.2建议

（1）用户需要有良好的信息提示来认知不同的错误显示，所以应该对不同的用户操作错误做出对应的提示，而不应该是所有错误给出一个同样的提示，这明显对用户很不友好。比如说用户用户名太长，当用户提交表单时，应该对用户提示用户名过长，应该缩短用户名的长度，用户用户名出现了不规则的字符，当用户提交表单时，应该对用户提示用户名不允许使用特殊字符，应该删除特殊字符，修改用户名，而这些提示不应该只是一个简单的用户名错误之类，用户可能不清楚具体的错误在哪，不停地修改就会烦躁。或者直接在界面层对这些信息进行处理，当某个部分不符合规范时，用户对当前表单移除焦点，直接在后面做出提示，此时对应的按钮变为不可选的，这样也就更符合用户的心理。

（2）一个优秀的网站应该考虑到安全性这一个大问题，特别是这种拥有个人账户或者需要进行交易的平台，安全性显得尤为重要，避免由于安全性不足而造成的用户信息泄露或者更严重的财产泄露，应该对每个模块都加入相应的安全机制。

（3）对于拥有账户的网站来说，像找回密码这种功能是必不可少的，如果用户忘记了密码，而缺乏这种机制是很可怕的，所以应该加入这种功能。

（4）用户的公聊私聊显得很无聊，一个没有任何功能的信息发送框对用户来说显得枯燥，那如果一个完善的插件就可以完成呢。比如说用户对字体颜色，大小的修改，甚至是表情包的使用等等都可以加入到这种元素中，让聊天不在仅仅是通知信息功能。

（5）想想假如一个网站一个人都没有，一个人在上面是多无聊的一件事请，那就不谈黑we购了，用户也许想要了解当前网站上有多少用户在线，那是一件多幸福的事情，找人聊天，找人交易也就更方便了，别让用户呆了半天才发现原来所有人都在打游戏，并没有在找好友聊天。