软件复用课程设计

Client-Server 应用程序

项目管理文档

小组： Team10

成员： 谢志杰 1352975

计鹏玥 1352914

王思尧 1352896

谢明玥 1352937

徐锦程 1353012

文档变更记录

|  |  |  |
| --- | --- | --- |
| 更改人 | 日期 | 更改内容 |
| 王思尧 | 2016年3月28日 | 创建文档 |
| 计鹏玥，谢明玥 | 2016年4月12日 | 添加项目计划 |
| 谢明玥 | 2016年5月3日 | 添加项目计划 |
| 计鹏玥 | 2016年5月10日 | 第八次上机实践内容 |

[文档变更记录 2](#_Toc4682)

[1．项目简介 4](#_Toc13826)

[1.1功能要求 4](#_Toc9796)

[1.2开发环境 5](#_Toc28140)

[2． 项目计划 5](#_Toc3479)

[3．项目分工 7](#_Toc19167)

[4．项目完成情况 8](#_Toc32155)

[5．项目总结 11](#_Toc10074)

[5.1交付产品 11](#_Toc7990)

[5.2总结 11](#_Toc26501)

[5.3改进方向 12](#_Toc4449)

# 1．项目简介

## 1.1功能要求

项目要求实现一个client+server的通信项目，可以实现多个客户端与一个服务端的链接，即服务器可以向多个客户端发送广播消息，客户端也可以向其他客户端发送消息，服务端/客户端保存所有收到的消息到文件，可以将每天的输出文件归档称为压缩包并且将每周的7个归档文件重新生成一个压缩包，以及对于同组成员， 消息可以只向同组成员广播。本项目在实际应用上类似于聊天室，能够实现信息能在多个客户端之间共享。

客户端在工作过程中，可以循环不停输入想要向服务器发送的消息，并且启动一个线程，专门用来监听服务器端发来的消息，并可以将消息打印输到控制台上。服务器端在工作过程中，启动一个监听何时需要向客户端发送消息的线程，每次接受客户端连接请求时都启一个线程进行处理，并且将客户端信息存放到公共集合中。当客户端发送消息时，服务器端将消息顺序存入队列中，当需要输出时，从队列中取出广播到各客户端处。

将所有的可复用构件标识出来，其中开发的可复用构件包括以下：

* 配置管理（CM）

a.从文件中读取读取参数配置

b.提供查询接口

c.动态加载

* 性能管理（PM）

a.接收应用程序的性能指标（指标名称，指标数值）

b.每分钟自动生成性能报告（对每个指标求和）

c.性能报告输出到单独的性能文件，文件名包括性能报告的时间

* 许可证（License）

a.每收到一个请求，计数加一

b.根据已经收到的消息数量和预设的数值，判断是否可以继续提供服务 · Throughput

· Capacity

* 保存消息到文件（File）

a.文件格式不限

b.文件路径可配置

c. 根据制定时间周期获取文件压缩包

* 文件压缩/解压处理（utils）

a.压缩格式不限

b.压缩包路径可配置

c.对每天的消息文件可进行压缩

d.对每周的消息文件可进行压缩

e.对每周的压缩文件进行解压

f.对每天的压缩文件进行解压

* 消息处理（MessageUtils）
* 验证（Authentication）
* 数据库（Database）

## 1.2开发环境

GitHub - 版本管理工具

Java - 开发语言

JUnit - 单元测试框架

RabbitMQ - 中间件

# 项目计划

|  |  |  |
| --- | --- | --- |
| 起始时间 | 结束时间 | 计划 |
| 2016年3月16日 | 2016年3月18日 | 项目组在GitHub上建立repository，初步完成分工，同时学习相关技术，例如socket通信等等。 |
| 2016年3月19日 | 2016年3月23日 | 各成员分别完成界面类、IO操作类、客户端、服务端的编写。此时程序已经具备多客户端与服务端通信的基本功能，部分要求还没有实现。发送接收消息尚基于命令行，界面类还未加入。 |
| 2016年3月24日 | 2016年3月26日 | 加入界面类，使得系统的输入输出基于界面。同时完善客户端与服务端的细节，实现需求规约里的所有要求。 |
| 2016年3月27日 | 2016年3月29日 | 使用JUnit对系统进行测试，同时开始相关文档的撰写。 |
| 2016年3月30日 | 2016年4月2日 | 开发可复用构件Logger、License、CM，并在server端添加对中间件RabbitMQ的使用，同时更新了相关文档。 |
| 2016年4月3日 | 2016年4月5日 | 在server和client端使用开发的可复用构件，并对可复用构件进行改进，同时继续更新相关文档。 |
| 2016年4月6日 | 2016年4月8日 | 添加可复用构件Authentication、PM，同时完善复用文档。 |
| 2016年4月9日 | 2016年4月10日 | 继续对已经开发的可复用构件进行改进，并将其生成JAR包，且完善了项目文档。 |
| 2016年4月13日 | 2016年4月14日 | 添加DEMO，继续改进可复用构件的接口，在client中进行使用，并对所有可复用构件进行了测试。 |
| 2016年4月19日 | 2016年4月20日 | 选择可复用构件CM、PM、License，并提出改进意见，在此基础上重构server端的代码。 |
| 2016年4月20日 | 2016年4月22日 | 整理测试用例，撰写测试文档。 |
| 2016年4月22日 | 2016年4月26日 | 开发File和PackerUtils构件,在客户端进行应用，并更新服务端代码。 |
| 2016年4月26日 | 2016年4月27日 | Client端功能扩展文档初步撰写。 |
| 2016年4月27日 | 2016年4月30日 | 对File构件进行测试 |
| 2016年4月30日 | 2016年5月2日 | 对File构件进行修改并提交使用的demo，添加解压缩功能，并对每周的文件进行重新压缩。 |
| 2016年5月2日 | 2016年5月3日 | Serve端使用rabbitMQ exchange发布消息。 |
| 2016年5月4日 | 2016年5月10日 | 对输出文件进行时间、大小、参数等方面的限制和配置，并实现加密功能。 |

# 3．项目分工

|  |  |
| --- | --- |
| 组员 | 分工 |
| 谢志杰 | 负责服务端的编写和测试以及项目的统筹安排。 |
| 开发大部分的可复用构件，并对它们进行了测试，以及生成相应的JAR包。 |
| 说明对PM和License构件的选择原因及意见，并对server端进行重构。 |
| 对server端的功能扩展进行说明。 |
| 计鹏玥 ，谢明玥 | 负责客户端的初步开发和测试。 |
| 对所有文档进行撰写，并随着项目进度不断进行完善。 |
| 王思尧 | 完成界面的设计以及相关java类的编写。 |
| 管理文档和测试文档的初步撰写。 |
| 部分构件的开发和修改，以及demo。 |
| 对client端的扩展功能进行说明。 |
| 徐锦程 | 完成文件IO相关操作类的编写，并进行复用文档和程序文档的初步撰写。 |
| 根据新添加的功能点对server端进行修改。 |
| 开发、修改、测试CM和SaveToFile构件，并将其添加到客户端。 |
| 说明对CM构件的选择原因及意见。 |

# 4．项目完成情况

|  |  |  |
| --- | --- | --- |
| 时间 | 人员 | 完成内容 |
| 3.16~3.18 | 全员 | 项目组在GitHub上建立repository，初步完成分工。 |
| 3.19~3.23 | 谢志杰 | Server端基本完成。 |
| 3.24~3.24 | 徐锦程 | GUI完成，等待客户端完成后开始集成。 |
| 3.25~3.26 | 计鹏玥，谢明玥 | 客户端原型完成。 |
| 3.26~3.28 | 计鹏玥，谢明玥 | 客户端重构，漏洞修复。 |
| 3.26~3.28 | 谢志杰 | 服务端重构，模块化。 |
| 3.28~3.29 | 计鹏玥，谢明玥 | 对客户端进行测试，并进行重构。 |
| 3.28~3.30 | 王思尧，徐锦程 | 开始文档的撰写。 |
| 3.30~3.31 | 谢志杰 | 将server端使用的功能模块根据课程要求独立成为License以及PM构件。 |
| 3.31~4.2 | 谢志杰 | 将server端原本使用的我们自己开发的消息队列移除，改为使用RabbitMQ，且MessageUtils构件模块也进行了相应的调整。 |
| 4.1~4.3 | 计鹏玥，谢明玥 | 根据新添加的可复用构件来撰写可复用文档。 |
| 4.2~4.6 | 谢志杰 | 完成server端代码的重构，并将License和PM完全整合。 |
| 4.3~4.7 | 徐锦程 | 初步开发CM可复用构件，并重构client。 |
| 4.6~4.7 | 谢志杰 | 开始对Authentication和DataSource进行重构。 |
| 4.8 | 谢志杰 | 整合生成PM可复用构件。 |
| 4.8~4.9 | 计鹏玥，谢明玥 | 将全部构件都添加到可服用文档中，撰写程序文档，并对接口进行描述。 |
| 4.9 | 王思尧 | 对Authentication构件以及客服端进行修改。 |
| 4.9 | 计鹏玥，谢明玥 | 将所有文档进行规范化。 |
| 4.10 | 谢志杰 | 将可复用构件生成JAR包。 |
| 4.13~4.21 | 徐锦程 | 测试并修改CM构件以及修改配置文件，并对该构件生成JAR包。 |
| 4.13 | 谢志杰 | 对PM、License、MessageUtils构件进行测试，修改API，并对这些构件生成demo。 |
| 4.14 | 计鹏玥 | 完善复用文档，程序文档，管理文档。 |
| 4.19 | 徐锦程 | 提交对CM的改进意见。 |
| 4.20~4.22 | 谢志杰 | 提交构件选择文档，以及PM,License构件改进意见，重构Server端。 |
| 4.20 | 王思尧 | 一些构件测试以及测试文档初步撰写。 |
| 4.22~4.23 | 计鹏玥 | 规范化文档以及添加新构件说明。 |
| 4.23~4.26 | 谢志杰 | Packer的开发和应用，使用新的PM构件，撰写Server端功能扩展。 |
| 4.23 | 谢明玥 | 完善构件选择文档。 |
| 4.24~5.2 | 徐锦程 | 在Client端使用SaveToFile构件，并测试该构件。写按照日期对文件进行归档以及相关demo。 |
| 4.27~5.2 | 王思尧 | 撰写Client端功能扩展。添加按照日期进行压缩和解压缩的功能。 |
| 5.3 | 谢志杰 | 修改Server端以及Server功能扩展文档。 |
| 5.3 | 计鹏玥 | 撰写2.0版本的程序文档。 |
| 5.3 | 谢明玥 | 根据新增构件完善测试文档。 |
| 5.4 | 王思尧 | 修复PackPerDay与PackPerDay中的问题。 |
| 5.4 | 谢志杰 | 更新PackerUtils构件及其应用。 |
| 5.5 | 计鹏玥 | 撰写File和PackerUtils的构建文档。 |
| 5.5 | 谢明玥 | 完善PackerUtils的构件文档。 |
| 5.8 | 徐锦程 | 根据新添加的FileMaxSize来修改server端的代码。 |
| 5.8 | 王思尧 | 编写FileMaxSize的demo及测试，并对client端进行重构。 |
| 5.10 | 谢志杰 | 添加FileUtils和DESEncryptor，对输出文件进行一定的控制及加密，进行了测试，并对server端进行了相应的修改，以及初步撰写server端的功能扩展文档。 |
| 5.10 | 王思尧 | 根据修改后的构件对原先的client端和其他构件的代码进行相应修改，以及初步撰写client端的功能扩展文档。 |
| 5.10~5.11 | 计鹏玥 | 更新管理文档和功能扩展文档。 |

# 5．项目总结

## 5.1交付产品

代码：源代码、测试代码、可复用构件的JAR包、demo代码

文档：可复用构件文档、程序文档、管理文档、测试文档、构件选择文档、构件改进意见文档、功能扩展文档

## 5.2总结

本项目开始与三月十六日，经过小组五名成员的共同努力，我们最终按时完成了项目，实现了需求中的所有功能。总体而言，此项目的开发比较成功，但是还是存在着一些问题，造成这些问题的原因也是多方面的，例如，前期我们对需求的理解出现了偏差，导致在开发过程中绕了少许的弯路。再比如，由于对socket的理解不够深入，导致开发遇到了一点困难。不过，这些问题最终都得以解决。

这个项目也给我们带来了很多新的认识。在项目确立后，我们就需要做好项目开发计划，需求调研用时，开发用时，测试用时，实施用时，维护用时。在我们做好了计划后，我们要随时的跟踪计划任务的完成进度，从而使我们的项目进度掌控在我们的开发周期范围之内。此外，除了对总体进度的把握，项目组成员之间的有效沟通必不可少。好的沟通能够加快项目的进度，使我们尽量地去避免一些隐藏的问题，及时地发现问题，解决问题，从而按时或者提前完成项目的开发。

随着项目的开发完成，我们积累了更多的经验，也学习到很多新的知识。相信这些经验与技巧，都将成为我们日后工作学习或者项目开发中宝贵的财富。

## 5.3改进方向

* 代码部分可以进行进一步优化，减少冗余代码，提高代码质量。
* 通过改进代码结构可以使项目的模块化程度进一步提高，符合软件复用的思想。