天津市大学软件学院

2016级卓越实验班毕业设计论文



姓 名: 王 东

学 号： 20135152

合作培养企业: 中 科 瑞 通

专 业 方 向: 大 数 据

蚂蚁热帮（AntHelp）生活服务系统的设计与实现

——基于Token认证的用户、权限管理

摘 要

通过对市场深入而充分地调研，得出准确的用户需求，结合后续的概要设计、详细设计、测试、部署和实施来完成整个系统。

热帮服务系统旨在服务大众，并且是以大众帮助大众的形式来进行服务，提高社会生产力，资源利用率，使得人们的生活更加地方便快捷。蚂蚁热帮（AntHelp）更是融合了社区模块，拉近了人与人之间的距离，让人们在轻松愉悦的交谈环境中互帮互助,让人们在互相交流的轻松氛围中获取资源。

Token认证是安全、有效的对用户登录信息进行验证以及对接口的权限验证的授权管理方式。通过Token签名，可以对用户登录信息的有效性进行验证，同时可以利用此技术实现单点登录，重复请求接口的处理。在Token签名的使用中，还可以借此对所有的接口进行权限的验证处理，给不同的用户角色进行不同的api接口开放，增加了系统的健壮性和安全性。

本系统采用的C#进行编码，同时采用最新的开发环境Microsoft Visual Studio 2017和Microsoft SQL Server 2016 开发环境进行开发。在强有力的开发环境的支持下，采用了自行开发的QX\_Frame前后端开发框架进行高效快速的开发。技术上使用了ASP.NET WebAPI构建后端服务，OWIN进行IIS解耦，Autofac作为IOC容器，EntityFrameWork6.0作为ORM框架，Angular2进行前端框架的搭建。

**关键词**：热帮服务 社交 C# QX\_Frame

The design and implementation of AntHelp life service system

——Token authentication based on user, rights management

**ABSTRACT**

The depth of the market and fully research, to obtain accurate user needs, combined with the follow-up design, detailed design, testing, deployment and implementation to complete the entire system.

The hot service system is designed to serve the public and to serve the public in the form of public help, to improve social productivity and resource utilization, making people's lives more convenient. Ants (AntHelp) is the integration of the community module, closer to the distance between people, so that people in a relaxed and pleasant conversation environment mutual help, let people in the exchange of relaxed atmosphere to obtain resources.

Token authentication is a secure and effective way to authenticate user login information and authorize authorization management of interface authority. Through the Token signature, you can verify the validity of the user login information, and can use this technology to achieve single sign-on, repeat the request interface processing. In the use of Token signature, you can also use this authority to verify the processing of all the different user roles to different api interface open, increasing the robustness and security of the system.

The system uses C # encoding, while using the latest development environment Microsoft Visual Studio 2017 and Microsoft SQL Server 2016 development environment for development. In a strong development environment with the support of the use of self-developed QX\_Frame front and back development framework for efficient and rapid development. Technically using the ASP.NET WebAPI to build back-end services, OWIN IIS decoupling, Autofac as an IOC container, EntityFrameWork6.0 as the ORM framework, Angular2 front-end frame structures.

**Key words:** Help Service Social C# QX\_Frame

目 录

[第一章 绪论 1](#_Toc481599821)

[1.1 项目背景与现状 1](#_Toc481599822)

[1.1.1 项目背景简介 1](#_Toc481599823)

[1.1.2 当前国内外现状 1](#_Toc481599824)

[1.2 选题意义与系统主要工作 1](#_Toc481599825)

[1.2.1 选题目的及意义 2](#_Toc481599826)

[1.2.2 本系统主要工作 2](#_Toc481599827)

[1.3 开发工具简介 2](#_Toc481599828)

[1.3.1 Microsoft Visual Studio 2017 2](#_Toc481599829)

[1.4 QX\_Frame开发框架介绍 5](#_Toc481599830)

[1.4.1框架简介 5](#_Toc481599831)

[1.4.2 框架技术要点 5](#_Toc481599832)

[1.4.3框架展望 5](#_Toc481599833)

[1.5 本章小结 5](#_Toc481599834)

[第二章 需求分析 6](#_Toc481599835)

[2.1 背景简介 6](#_Toc481599836)

[2.2 可行性分析 6](#_Toc481599837)

[2.2.1 社会可行性 6](#_Toc481599838)

[2.2.2 经济可行性 6](#_Toc481599839)

[2.2.3 技术可行性 7](#_Toc481599840)

[2.2.4 可行性分析结论 7](#_Toc481599841)

[2.3 功能需求分析 7](#_Toc481599842)

[2.3.1 功能划分 8](#_Toc481599843)

[2.3.2 系统使用角色分析 9](#_Toc481599844)

[2.5 本章小结 11](#_Toc481599845)

[第三章 详细设计 12](#_Toc481599846)

[3.1业务流程分析(活动图) 12](#_Toc481599847)

[3.2业务对象模型 13](#_Toc481599848)

[3.3业务类图分析 14](#_Toc481599849)

[3.4数据库设计 15](#_Toc481599850)

[3.5业务流程顺序图 17](#_Toc481599851)

[第四章 系统实现与代码编写 20](#_Toc481599852)

[4.1系统实现 20](#_Toc481599853)

[4.2功能代码 22](#_Toc481599854)

[第五章 测试 22](#_Toc481599855)

[5.1测试的目的 22](#_Toc481599856)

[5.2测试的过程 22](#_Toc481599857)

[参考文献 30](#_Toc481599858)

[附录 31](#_Toc481599859)

[致谢 39](#_Toc481599860)

# 绪论

## 1.1 项目背景与现状

### 1.1.1 项目背景简介

蚂蚁热帮（Ant Help）是以生产第三方服务平台为目标，致力于更方便，更快捷的，更有效的O2O生活服务平台。突出了“你出钱，我出力！”的互帮互助主题。更方便，更快捷，更有效的生活方式，改变人们生活节奏，提高人们办事效率。继滴滴、美团、饿了么之后的更进一步的良好的生活服务平台。

### 1.1.2 当前国内外现状

本系统采用的是社交+O2O的方式，纵观国内外社会的发展，如今的人们更加趋向于社交平台的使用，而传统的电话邮件的方式已经逐步落后于社交，但是并不能完全取缔。O2O的浪潮也随着各支付平台的兴起一浪接过一浪。但这并不能代表所有前景都是非常美好的。未来仍需要良好快速的发展，使得互联网更加接地气！

GlobalWeb Index在2014年年底发布了一份社交网络研究报告，报告显示，在过去的半年内，Twitter在过去半年用户和活跃用户增长率只有18%和26%，而Facebook这个全球最大的社交应用的用户和活跃用户增长率则只有6%和2%，马上就要探底。在Facebook网罗了全球超过30%人口的情形下，除非能够进入中国（这在短期内显然是不可能的），否则它的增长基本已达极限，在2015年Facebook的用户增长率很可能会低于0%。作为国内较早的社交网络之一，微博“活跃度下降”已不是一个月两个月的事了，今年微博的第三季度财报显示：微博月活跃用户数（MAU）为1.67亿，较上年同期增长36%。这个数据还算让人欣慰，但考虑到7月世界杯的淘汰赛为微博贡献了巨大的活跃度，这个数据并不那么让人乐观。可以看到，国内外的互联网社交发展并非一帆风顺，虽然客户仍然增多，但是增长速度确是异常缓慢，如果继续保持这样的发展现状，而未能融入新的元素，那么将不会再有当初爆炸时候的发展劲头。

## 1.2 选题意义与系统主要工作

### 1.2.1 选题目的及意义

为什么大家要用我们的平台？我们平台有什么样的优势呢？且不说点餐需要使用点餐软件进行点餐，打车需要运行打车软件进行打车，而且很多司机并未注册打车软件进行打车服务。况且点到的餐由餐馆随便找个“爆粗口”的大妈配送外卖引起的各种食欲的下降、开车师傅因为长相太“魁梧”而阻碍了社会的沟通交流…种种可能想到的各种问题，在我们这里都就能找到很好的解决办法！

我们同时旨在以社交为基础扩展开来的服务，这就意味着帮助你的对象可能是你心仪已久的同校音乐系帅哥，你帮助的对象可能是你心动多年的美术系美女，凡是你平时能接触到的社交群体，都可能作为服务被服务的对象，如果不幸匹配到了校外彪悍大叔，同时你还不想让他帮助你，那么你可以一键cut掉他。

该产品以大学生为前期对象，以各类大学为前期试点，通过试点的实际实施情况进行分析优化改进，逐步向全面市场推广，争取服务到广大人民群众中去。

### 1.2.2 本系统主要工作

如果你想要别人帮助你，不管是什么样的要求，你都可以进行下单，下单的结果就好比是在朋友圈发了一条动态“x号宿舍楼 有x美女 需要找人代取一下快递 ，报酬 xx 。恳切希望帅哥来拯救” 当然如果你有足够的魅力的话，完全可以写报酬0哦。然后就有人在圈内进行不断地刷刷刷，是的，下拉刷新，然后看到了你发的这条动态，刚好顺路（当然也可以不顺路的），进行接单。接下来我们下单接单双方可以随时进行对对方信息的查看，如果你觉的对方不太靠谱，大可以取消掉订单，然后选择重新发布。如此可以直到选中您满意的服务对象。服务完毕，双方可以在聊天中约定交付方式，然后进行交付，系统就可以按照原先的约定进行交付啦。除了快递，托熟人捎饭, 打车…不限信息种类，只要你能想到的任务，都可以进行发布。只要你有足够的勇气完成，都可以接受。通过我们的“蚂蚁热帮”社交生活服务平台，我们更加扩宽了我们日常生活的圈子，更进一步提升了交友的范围，使我们的日常交流更加丰富，同时能提高我们的生活质量和节约时间、提升生产效率。“蚂蚁热帮”，让我们的生活更精彩！

## 1.3 本章小结

本章对本课题的选题意义和国内外发展情况进行了调查，明确了项目的背景信息，然后对本系统所涉及的核心业务进行了大概的介绍。在完善的调查结果以及对项目背景信息明确的前提下继而进行项目的正式开展。

# 系统可行性分析

## 2.1 背景简介

该项目是以生产第三方服务平台为目标，致力于更方便，更快捷的，更有效的O2O生活服务平台。是在当前人们越来越快的生活节奏以及越来越方便的生活行为方式的大环境下进行启动的一个良好的生活服务平台项目。如果项目能达到预期的效果，一定是继滴滴、美团、饿了么之后的齐肩并进的良好生活服务平台。

## 2.2 可行性分析

### 2.2.1 社会可行性

项目针对当前人们日益加快的生活节奏以及日益方便的生活方式研发，在社会上的认可程度以及社会接受程度上有比较大的先天优势。除了在系统业务上的用户接受度的优势外，项目中还有人们生活中常见的社区模块，社区丰富了人们日常生活中的沟通方式，以一种更方便快捷的方式带给人们日常的交流的同时也让人们互帮互助。

法律因素方面。开发中所有商业软件都采用购买的方式使用正版软件。所有源代码为自主开发，自有知识产权。涉及到第三方控件和支付接口，也通过正规渠道进行商务购买。开发中使用的开源技术不存在法律纠纷。在用户使用可行性方面，用户对系统的熟悉和使用能再很短的周期适应，同时管理员也能迅速接受系统的管理方式，能很好的融入系统，乐在其中。

综上所述，本系统没有违背法律和道德，所以在社会上是可行的。

### 2.2.2 经济可行性

1.人员费用

本系统开发期2个月，开发人员3人，作为毕业设计，开发人员都为毕业设计相关人员，故而不需要开发人员费用。

2.硬件设备费用

系统的开发使用均由毕业设计相关人员自备，故而不需要硬件设备费用。

3.软件费用

系统所需软件费用为0元，其中包括：

1. Microsoft Visual Studio 2017 community（免费）
2. Microsoft SQL Server 2014 Express（免费）
3. Microsoft Visual Code （免费）
4. Google Chrome（免费）

4.耗材费用

由于文档等需求，各种耗材费用预计约50元。由于是社会性的社交O2O项目，故需要社会调研问卷印刷费200元。

5.其他费用

毕业设计相关文档查重费用200元。

6.不可预见费用

由于开发过程中遇到的事件不可知，拟不可预见费用约200元。

7.总开发费用

综上分析，该系统总的开发费用为650元。

### 2.2.3 技术可行性

该系统采用微软的Asp.net webApi2以及市场上较为流行的Angular2，自行研发的QX\_Frame，大大减轻了系统开发的难度，在有成熟框架作为依托的前提下，结合自行开发的代码生成器，加之团队成员对开发环境的适应以及对开发技术的悉心学习，具备了足够的专业基础知识。因此，在技术上是不存在困难的。

### 2.2.4 可行性分析结论

通过从经济、技术、社会可行性三方面的分析，得知系统的设计实现在经济上、技术上、社会上都具有很高的可行性，故而得出结论，项目可行。

# 系统开发环境及相关技术简介

## 3.1 Microsoft Visual Studio 2017 简介

微软借着Visual Studio品牌20周年之际，于美国太平洋时间2017年3月7日9点召开发布会议，宣布正式发布新一代开发利器Visual Studio 2017。作为Microsoft公司的主力集成开发工具（IDE），历经了20年的革新，今天终于走到了vs2017这个有历史意义的节点，很有幸，我能在第一时间安装体验了visual studio 2017，并充分体验并利用了他带来的各项好处。

这里我列举Visual Studio 2017的新增功能：

1. 与云集成，内置工具可让所有的 NET Core、Azure 应用程序、服务、Docker 容器等全面集成。 无缝体验让你感觉如同在 Azure 数据中心工作一样。
2. 有效协作，直接管理任意提供程序托管的团队项目，包括 Visual Studio Team Services、Team Foundation Server 或 GitHub。 或者，使用新的“打开任意文件夹”功能，无需通过正式项目或解决方案即可快速打开并处理几乎所有代码文件。
3. 交付优质移动应用，借助 Xamarin 的高级调试、分析工具以及单元测试生成功能，与以往相比你可以更快、更轻松地构建、连接和调整适用于 Android、iOS 和 Windows 的本机移动应用。 你还可以选择使用 Apache Cordova 开发移动应用，或构建 C++ 跨平台库。
4. 提高语言水平，Visual Studio 继续加强对最新编程语言功能的支持。 无论使用 C#、Visual Basic、C++、TypeScript、F# 还是使用第三方语言（例如 JavaScript），在整个开发体验中你都能获得一流的功能支持。
5. 性能得以优化，Visual Studio 包含了对日常使用的核心功能的大量性能改进。 你还会发现在整个开发生命周期中，启动速度显著加快，内存占用大大降低，并且响应能力明显提高。

## 3.2 Microsoft SQL Server 2016 简介

作为当前最新版本的Microsoft SQL Server数据库，Microsoft SQL Server 2016 在延续了历史版本的各项优势以外，更是添加了诸多的新特性。这些新的特性使我们的使用更加的方便快捷，提高了我们的开发效率。下面我展示一些Microsoft SQL Server 2016的新特性：

1. 全程加密技术(Always Encrypted)，全程加密技术(Always Encrypted)支持在SQL Server中保持数据加密，只有调用SQL Server的应用才能访问加密数据。
2. 动态数据屏蔽(Dynamic Data Masking)，利用动态数据屏蔽功能，你可以将SQL Server数据库表中待加密数据列混淆，那些未授权用户看不到这部分数据。利用动态数据屏蔽功能，你还可以定义数据的混淆方式。
3. JSON支持，在SQL Server 2016中，你现在可以在应用和SQL Server数据库引擎之间用JSON格式交互。微软公司在SQL Server中增加了对JSON的支持，可以解析JSON格式数据然后以关系格式存储。
4. SQL SERVER支持R语言，SQL Server支持R语言处理以后，数据科学家们可以直接利用现有的R代码并在SQL Server数据库引擎上运行。这样我们就不用为了执行R语言处理数据而把SQL Server数据导出来处理。该功能把R语言处理带给了数据。
5. Query Store，如果你经常使用执行计划，你就会喜欢新版的Query Store功能。在2016之前的版本中，你可以使用动态管理试图(DMV)来查看现有执行计划。
6. 行级安全(Row Level Security)，SQL数据库引擎具备了行级安全特性以后，就可以根据SQL Server登录权限限制对行数据的访问。限制行是通过内联表值函数过滤谓词定义实现的。安全策略将确保过滤器谓词获取每次“SELECT”或者 “DELETE”操作的执行。

## 3.3 Angular2前端框架和Typescript语言简介

AngularJS当初是提供给设计人员用来快速构建HTML表单的一个内部工具。随着时间的推移，各种特性 被加入进去以适应不同场景下的应用开发。然而由于最初的架构限制（比如绑定和模板机制），性能的提升已经非常困难了。由此为引，Angualr2便这样诞生了。

Angular2的开发语言采用Microsoft的Typescript语言，当年的浏览器大战，让人记忆犹新，Chrome的出品商Google和IE的出品商微软正是浏览器大战的两大主角。俗话说：没有永远的朋友，也没有永远的敌人，只有永远的精益求精。Google与微软相逢一笑泯恩仇，进行了具有历史意义的合作，这将是IT互联网领域的一次历史性的牵手。

Angular2由八个主要部分构成，正是这八大件的密切配合，相互协作，才有了Angular2的高效、快速、方便等先天优势：

* 模块 (module)
* 组件 (component)
* 模板 (template)
* 元数据 (metadata)
* 数据绑定 (data binding)
* 指令 (directive)
* 服务 (service)
* 依赖注入 (dependency injection)

Typescript是微软开发的自由的，开源的编程语言，语言中包含了后端开发人员熟悉的静态类型和基于类的面向对象编程特点。使得前后端的开发人员在语法的使用上更近了一步。TypeScript是JavaScript类型的超集，它可以编译成纯JavaScript。TypeScript可以在任何浏览器、任何计算机和任何操作系统上运行，并且是开源的。TypeScript扩展了JavaScript的语法，所以任何现有的JavaScript程序可以不加改变的在TypeScript下工作。TypeScript是为大型应用之开发而设计，而编译时它产生 JavaScript 以确保兼容性。Typescript有几大特性：

* 类型批注和编译时类型检查
* 类
* 接口
* 模块
* lambda 函数

## 3.4 QX\_Frame开发框架介绍

### 1.4.1框架简介

QX\_Frame.FrameWork4.6 (版本号：2.0.0) 为本人亲自开发的开源项目，该项目于2016年11月开始启动，至今已投入使用。框架采用最新的.NetFramework4.6和AngularJS2平台进行搭建的前后台通用开发框架，框架包含了Console控制台应用程序，WindowsForm应用程序，Web应用程序，项目间采用插拔式的方式进行配置，仅仅需要使用需要的项目进行使用即可，大大降低了应用程序间的耦合性。目前该项目已部开源于GitHub社区-> <https://github.com/dong666/QX_Frame.FrameWork4.6>。

### 1.4.2 框架技术要点

1. 封装了通用的数据访问层基类WcfService、Entity<T>，并且采用Wcf<T>方法创建自定义数据层管道，传入查询条件对象，通用管道查询方法QueryAll()、QuerySingle()进行对象查询,返回一致的查询结果WcfQueryResult；支持分页条件查询；极大提高了编码效率。
2. 框架层的数据库使用db\_qxframe，和业务数据库分离，充分解耦了后期框架和业务逻辑，使得框架重用性得到提高。
3. Ioc容器采用高效的依赖注入框架Autofac框架，提高了程序的运行效率。
4. 框架采用AOP技术进行通用的异常处理，并且加入了系统日志，提高了项目的编码效率和后期的可维护性。
5. 框架内使用缓存，提升查询效率。
6. 框架的数据库处理采用ORM+Sql的兼容方式，丰富了开发人员的开发使用方式。
7. 插拔式的项目结构使得开发更加丰富且随心所欲，极大的给开发人员带来了便捷。同时也大大增加了框架的兼容性和后续可维护性。

### 1.4.3框架展望

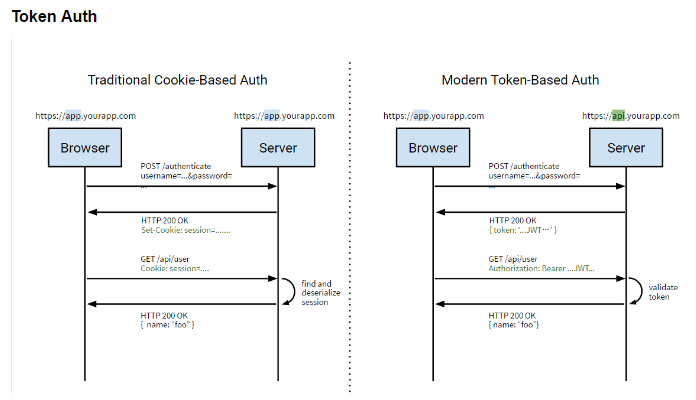
1. 增加多种数据库的支持，不依赖于.net环境下的Microsoft SQL Server，新增对MongoDB的非关系型数据库的支持。
2. 采用Memcached实现分布式集群服务器缓存。
3. 增加提示信息国际化的配置方式，使得框架更加人性化。
4. 升级采用.net Core 跨平台的开发方案，增加框架健壮性。

## 3.5 Token认证简介

Token认证是当下流行的身份验证机制，用于保证服务端接口的安全性和时限性。其原理初始时用户提交账号数据给服务端，服务端采用一定的策略生成一个字符串（token），token字符串中包含了少量的用户信息，并且有一定的期限。服务端会把token字符串传给客户端，客户端保存token字符串，并在接下来的请求中带上这个字符串。使用基于 Token 的身份验证方法，在服务端不需要存储用户的登录记录。大概的流程是这样的：

1. 客户端使用用户名跟密码请求登录
2. 服务端收到请求，去验证用户名与密码
3. 验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端
4. 客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage 里
5. 客户端每次向服务端请求资源的时候需要带着服务端签发的 Token
6. 服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据

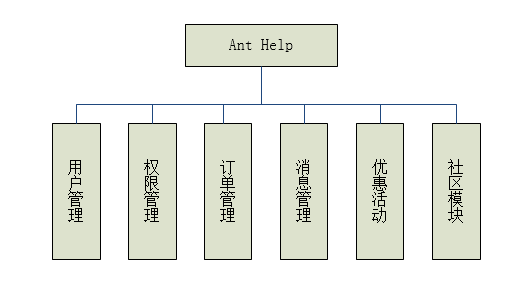
通过Token认证的机制，保障了服务器用户数据的安全性，同时也使服务器的安全性得到了提高。

下面以图形方式展示了传统的认证方式以及Token认证方式的差异，在Token认证中，客户端请求资源的时候，服务端会验证Token的有效性来确定是否有获取资源的权限，来达到提升api接口安全性的目的：

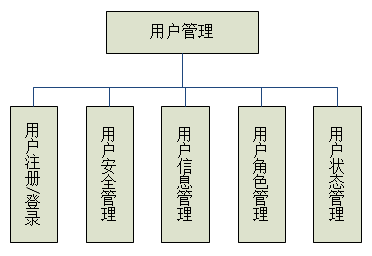
# 第四章 需求分析

## 功能划分

### 4.1.1总功能模块划分

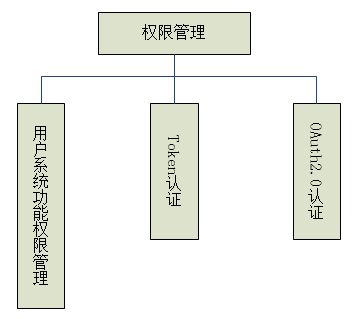
1. 用户管理：在用户管理模块中，设计系统相关用户的角色以及对各种角色的用户信息进行管理。在对用户信息进行管理的同时还涉及到了用户的注册登录等功能，该模块有涉及到系统的Token认证相关概念。
2. 权限管理：该模块主要用于系统的各项功能的访问权限的管理，对不同的用户开放不同的权限进行操作，使得系统的稳定性和安全性得到了提高，加强了管理员对人员的管理以及系统的维护。
3. 订单管理：订单管理模块用户接收用户的下单请求，并且由系统分发订单信息，其他用户抢单接单进行对订单的后续流程的操作。不同的用户角色对订单有不同的操作策略。
4. 消息管理：消息管理模块主要用于管理员对用户发送系统消息，或者是用户对系统的意见或建议，以及用户对其他用户的投诉信息，可以向系统投诉信箱发送信息，系统管理员登录后可以查看用户发送来的投诉信息，并进行对投诉信息的处理。
5. 优惠活动：优惠活动模块，由系统发起的优惠，用户可以在规定期间内进行活动的参与。
6. 社区模块：社区模块类似于WeChat Moments等社交媒体的沟通平台，用户可以在平台内进行交流分享。

### 4.1.2用户管理模块

用户管理模块主要有对用户的登录注册想关信息的管理以及对用户信息的管理，同时还有对用户的安全、用户的角色、用户的状态信息进行管理。该模块涉及到用户登录后对Token签名的生成以及对签名的维护，以及单点登录的实现。用户注册采用邮箱验证的方式进行验证注册。用户角色以及用户状态由管理员角色进行对普通用户的授权，角色和状态的改变会直接影响到后续的接口访问权限。

### 4.1.3权限管理模块

权限管理模块，主要是利用了用户管理模块中，用户登录之后生成的Token签名进行对用户信息的有效性进行验证，并对用户角色、用户状态进行验证，所有的api接口都会涉及到Token值的获取以及通过对Token签名值解密的到的用户信息进行权限认证。



## 4.2 用例分析

1.管理员

(1)用户信息管理。

A.查看用户信息，包含对用户状态，用户角色的查看。

B.修改用户信息，包含对用户状态的修改，以及对用户角色的修改。用户角色的修改直接关系到对接口访问的授权控制。

(2)管理员注册登录。

A. 管理员的身份注册，可以由其他管理员或者系统管理员直接任命。

B. 管理员登录，登录完成会自动保存角色信息，以便在访问接口时判断是否为管理员以及对接口权限的控制。

如图2.1所示，该图刻画的是管理员的用例图。

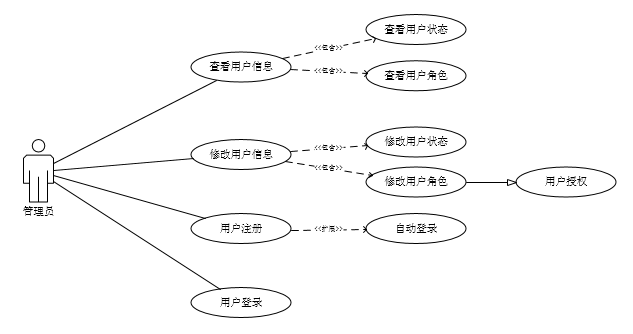


图2.1 管理员的用例图

Fig .2.1 Administrator's rights management use case diagram

2.用户

(1)用户信息管理。

A.查看用户信息，包含对用户状态，用户角色的查看。只能查看个人的全部信息。

B.修改用户信息，包含对个人用户的信息修改。

(2)用户注册登录。

A. 用户注册，以邮箱认证的方式注册用户，并且在注册完成后实现自动登录。

B. 用户登录，登录完成会自动保存角色信息，以便在访问接口时判断是否具有足够的权限。

如图2.2所示，该图刻画的是用户的用例图。

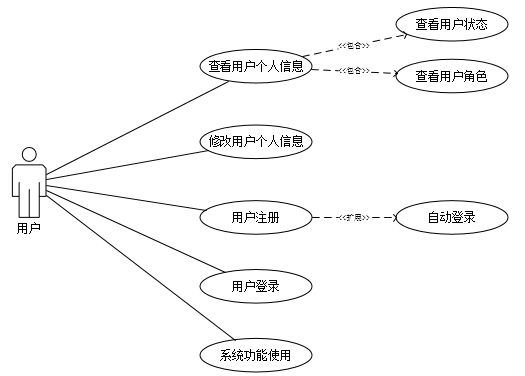


图2.2 用户的用例图

Fig .2.2 User's rights management use case diagram

### 4.2.1 用户（管理员）注册

1. 用例说明

用户（管理员）注册用例的明细说明如下表5.1所示：

表 5.1 用户（管理员）注册用例明细

Table 5.1 Search case detail

|  |  |
| --- | --- |
| 用例名 | 用户（管理员）注册 |
| 参与者 | 用户（管理员） |
| 前置条件 | 用户输入合法的邮箱、账号、密码并提交 |
| 事件流程 | 1. 用户输入邮箱、账号、密码进行注册提交； 2. 客户端对账号密码进行验证； 3. 客户端将请求发送到服务器； 4. 服务器对邮箱账号密码进行验证； 5. 服务器处理账号密码，存入临时空间，并向用户邮箱发送注册验证链接； 6. 用户接收邮件并点击其中的注册链接进行跳转注册； 7. 客户端自动跳转提示用户注册成功； 8. 注册成功自动进行登录； |

2. 用例流程

用户注册处理流程如下图5.3所示：

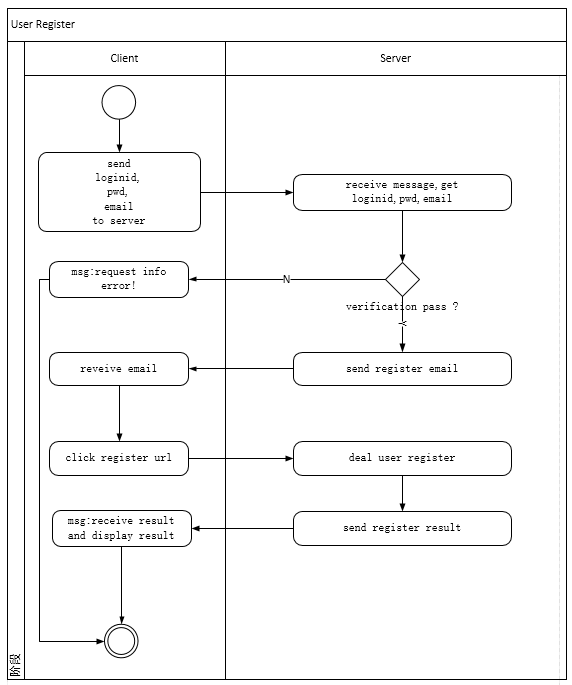


图 5.3 用户注册处理流程

Graph 5.3 process

3. 人机交互

用户注册用例人机交互图如下图5.4所示：

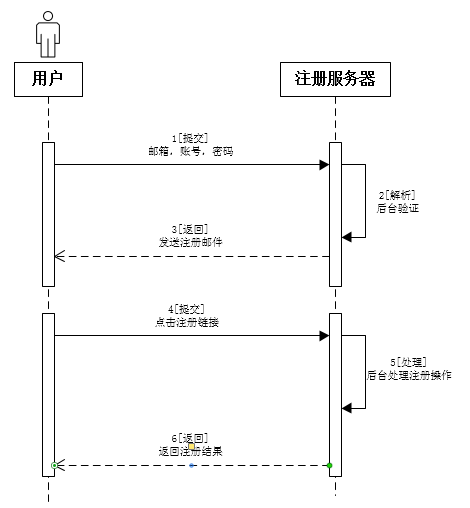


图 5.4 关键字搜索人机交互

Graph 5.4 Search case HC interaction

### 4.2.2 用户（管理员）登录

1. 用例说明

用户（管理员）登录用例的明细说明如下表5.1所示：

表 5.1 用户（管理员）登录用例明细

Table 5.1 Search case detail

|  |  |
| --- | --- |
| 用例名 | 用户（管理员）登录 |
| 参与者 | 用户（管理员） |
| 前置条件 | 用户输入合法的账号、密码并提交 |
| 事件流程 | 1. 用户输入合法账号密码进行提交； 2. 客户端生成时间戳和随机数，并进行对发送参数的加密，加密格式如下MD5[logiId+MD5[Pwd]+random+timestamp]然后将loginId+random+timestamp+MD5\_String发送到服务器； 3. 服务器对时间戳进行校验判断是否过期；对时间戳和随机数进行校验判断是否重复请求；获取服务器密码并进行同样格式的加密校验，判断是否账号密码错误； 4. 服务器生成公私钥、服务器密钥secretKey=MD5[loginid+MD5[pwd]+timestamp]、token签名tokenSign=MD5[loginid+logintimestamp+random]并将生成的结果存储到服务器端； 5. 服务器生成token令牌token=RSA\_publicKey[uid+loginid+expiretimestamp+tokensign]； 6. 服务器返回appKey，secretKey，token，用户信息； 7. 客户端保存secretKey，token，loginId；’ |

2. 用例流程

用户登录处理流程如下图5.3所示：

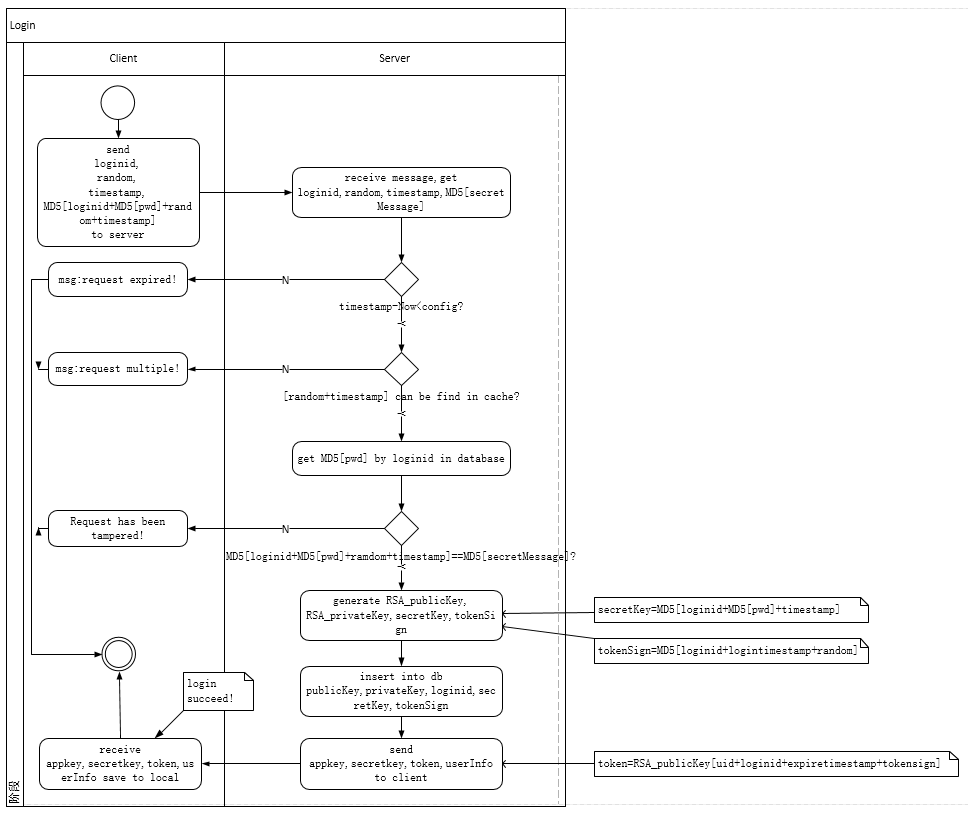


图 5.3 用户登录处理流程

Graph 5.3 process

3. 人机交互

用户登录用例人机交互图如下图5.4所示：

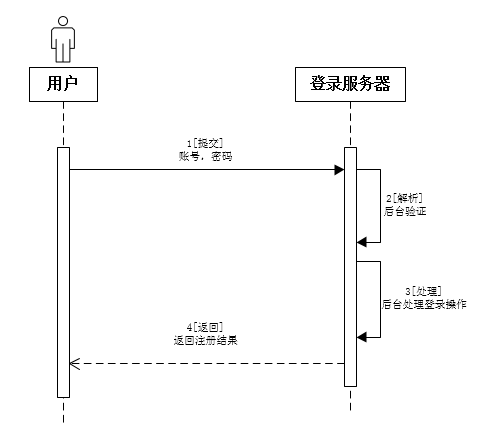


图 5.4 用户登录人机交互

Graph 5.4 Search case HC interaction

### 4.2.3 系统功能使用（Token认证、权限认证、以查询用户信息为例）

1. 用例说明

用户信息查询用例的明细说明如下表5.1所示：

表 5.1 用户（管理员）登录用例明细

Table 5.1 Search case detail

|  |  |
| --- | --- |
| 用例名 | 系统功能使用（Token认证，权限认证，用户信息查询为例） |
| 参与者 | 用户（管理员） |
| 前置条件 | 用户点击某用户的信息链接查询 |
| 事件流程 | 1. 用户点击某用户信息链接进行查询； 2. 客户端从本地存储空间获取appKey、token、sign，生成random，timestamp，请求参数（分页参数、查询条件）。Sign的生成算法为sign=MD5[random+timestamp+a+a.value+b+b.value+c+c.value+secretkey]； 3. 服务器对时间戳进行校验判断是否过期；对时间戳和随机数进行校验判断是否重复请求； 4. 服务端用相同的签名算法对接口参数进行验证sign=MD5[random+timestamp+a+a.value+b+b.value+c+c.value+secretkey]，用客户端发送来的签名和服务端生成的签名进行匹配，如果不能匹配，即认为请求参数被篡改； 5. 服务端获取用户端的appKey和Token值，用appKey得到服务端的RSA私钥，并用私钥进行对Token解密，得到uid,loginid,expiretimestamp,tokensign，通过uid获取用户的权限，并和当前接口权限进行对比验证，expireTimeStamp和当前时间进行比较校验，判断该Token是否过期（登录信息过期，需要重新登录），tokenSign和通过appKey获取的tokenSign进行匹配，校验token是否正确； 6. 验证全部通过，返回请求资源； |

2. 用例流程

用户信息查询流程如下图5.3所示：

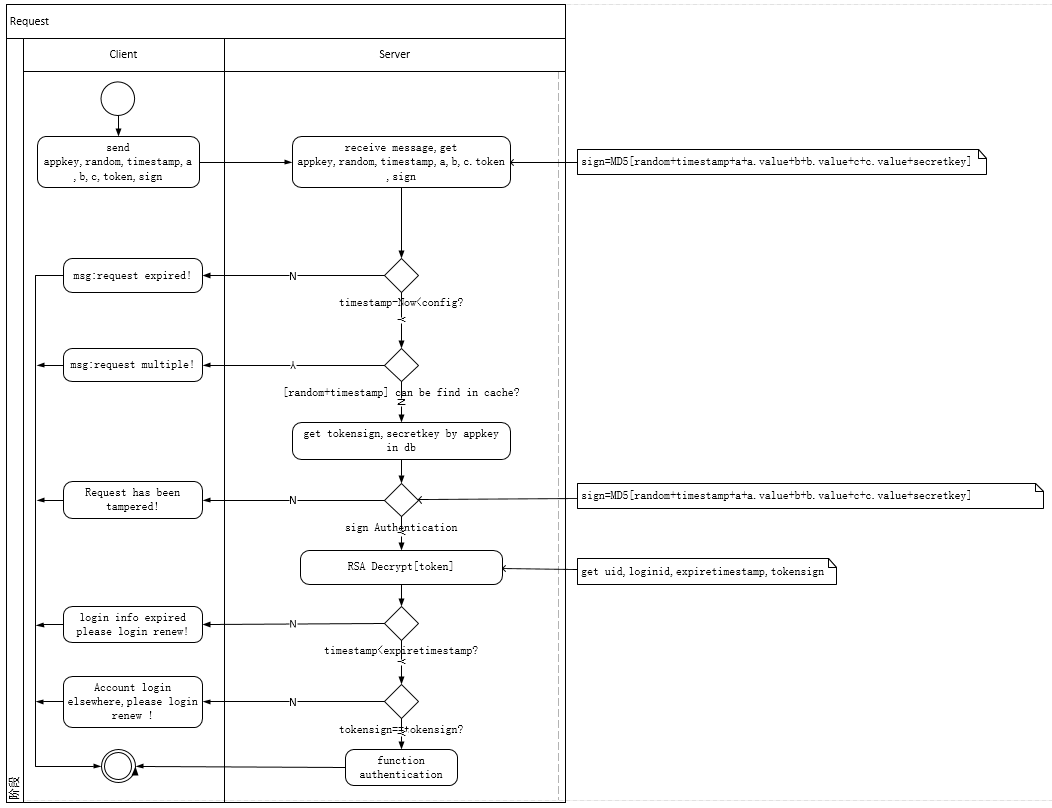


图 5.3 用户注册处理流程

Graph 5.3 process

3. 人机交互

用户信息查询用例人机交互图如下图5.4所示：

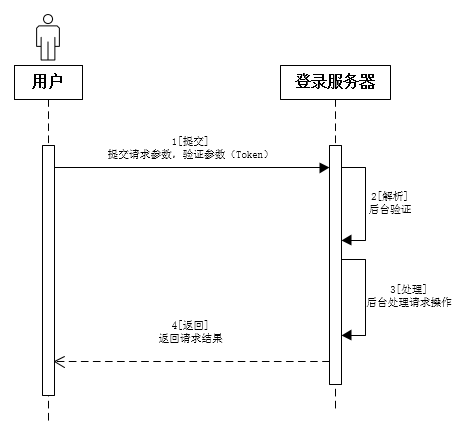


图 5.4 关键字搜索人机交互

Graph 5.4 Search case HC interaction

## 2.5 本章小结

本章对统计过程管理进行了需求分析。在功能需求分析中。首先对该项目的可行性进行了分析，包括社会可行性、经济可行性和技术可行性。在明确该项目的可行性后，该部分对系统的功能模块进行了分析，明确了本系统的功能用例，绘制了系统的功能用例图。

# 第五章 系统详细设计

## 5.1对象模型设计

### 5.1.1 业务对象

根据对本系统的业务模块进行分析以及对Token认证的流程的掌握，再通过对当前业务的内容扩展，可以逐步提取出系统的业务实体对象以及Token认证的实体对象。在本系统中，将用户角色、用户状态、用户信息、Token认证相关的对象从AntHelp中抽离出来，便于后续在其他系统上的重用以及扩展，故而当前框架级的业务对象有：用户账户（密保）实体、用户角色（角色描述）实体、用户状态（状态描述）实体、用户信息实体（用户账单、账单记录）、Token认证实体等，从内容上来说，每一个实体都会有相应的扩展实体存在，而每一个扩展实体对于主实体而言，又都是至关重要的，密不可分的。

### 5.1.2 对象属性

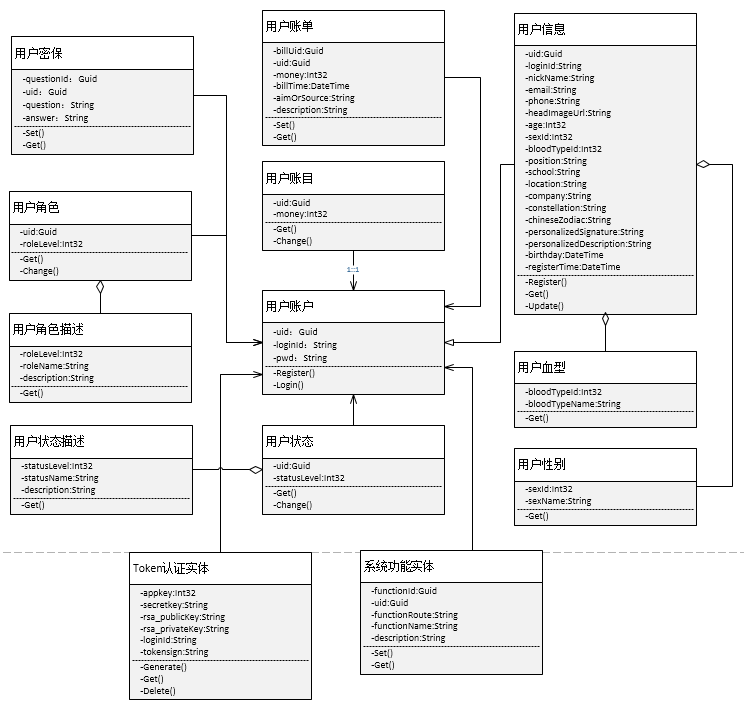
通过业务对象结合业务的分析，得到了初步的对象以及其属性。对初步的业务对象和框架对象进行进一步抽取、合并、扩展，降低了各对象之间的耦合性以及提升了对象的扩展性。最终我们得到了优化以后的业务对象模型如下图所示：



在当前的业务对象模型中，用户是主体，所有的业务对象模型都是为用户服务，因此，用户账号是所有业务对象的核心。这里采用用户账号和用户信息分离的模式，便于系统的设计使用，充分考虑到后续系统的扩展问题，将用户角色、用户状态、用户血型、用户性别、用户账目等抽取成单个的用户信息对象实体，既降低系统之间的耦合性又能充分考虑到后续的系统扩展。Token认证实体则是专门用来进行Token认证的功能实体，当然Token认证实体的使用又必然会涉及到用户账户的实体，因此说，用户账户是所有系统实体的核心，所有的实体都为用户账户实体服务。

### 5.1.3 对象关系

通过对业务对象的分析、提取、优化，我们已然得到了我们系统所需的业务实体，并且我们的业务实体充分考虑了系统适用性及系统扩展性，因此我们所有的工作都完美完成，下一步我们需要进行对上一步抽取出来的实体对象进行对其对象之间的关系进行分析。当然，在设计实体的同时我们已经充分考虑了实体对象之间的关系，这里我们进行更详细，更具体的分析，并用图示的形式展示出来，实体对象之间的关系如图：



## 5.2业务逻辑设计

### 5.2.1 提取业务逻辑类

本系统的架构严格按照QX\_Frame的标准格式进行类的创建及命名，QX\_Frame规定的项目结构为：Data、Data.Contract、Data.Service、WebApi.Controllers因此，涉及到的类分为实体类Data、服务数据处理接口Data.Contract、服务数据处理类（逻辑类）、控制器类WebApi.Controllers。下面对本系统用到的业务逻辑类进行一一的介绍。

1. 实体类

根据本系统前面进行的业务对象模型设计，可以获取本系统所需要的业务对象，也就是本系统涉及到的业务逻辑实体类Entities。包括用户账户类、用户信息类（用户血型类、用户性别类）、用户角色类（角色描述类）、用户状态类（状态描述类）、用户密保类、用户账目类、用户账单类、Token认证实体类、系统功能权限实体类。

1. 实体业务逻辑类

根据框架的标准格式，实体业务逻辑类封装了每一个业务对象实体的基础增删改查操作以及对查询操作的大力扩充。在此基础上，实体业务逻辑类还包括了一些复杂的对实体的操作流程，并提供相应的方法供控制器调用。综上所述，我们的实体业务逻辑类应包括：用户账户逻辑类、用户信息逻辑类（用户血型逻辑类、用户性别逻辑类）、用户角色逻辑类（角色描述逻辑类）、用户状态逻辑类（状态描述逻辑类）、用户密保逻辑类、用户账目逻辑类、用户账单逻辑类、Token认证实体逻辑类、系统功能权限实体逻辑类。

1. 控制器类

控制器类是提供一个对外的接口供客户端调用，除了实现对应的操作方法外，控制器类还同时肩负着对用户权限的控制，保证了系统的安全性。系统控制器类的实现采用Restful风格的Action接口，这里约定了控制类的作用是对对应资源的各种有效操作。在系统的业务逻辑分析详尽的基础上，我们提取出了必要的需要操作的资源，并且把现有资源实现到控制器中：账号控制类、用户控制类、权限控制类、登录控制类、用户金额控制类、用户角色控制类（角色描述控制类）、用户状态控制类（状态描述控制类）、文件控制类。

1. 边界类

在控制器类实现的过程中，系统的权限控制是所有的接口都需要使用的公共部分，因此将系统权限的认证类抽离出来进行封装成一个统一的过滤器类来进行使用，这里的Token认证、权限控制类属于各业务控制器类的边界类，由控制器类实现的过程中统一进行调用。

### 5.2.2 业务逻辑类分析

1. 关键字搜索

搜索模块的关键字搜索用例逻辑类图如下图6.5所示：

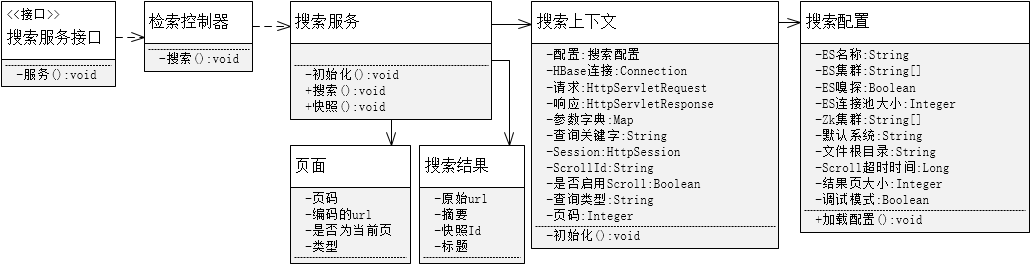


图 6.5 关键字搜索逻辑类图

Graph 6.5 Search case logic class

1. 快照查看

搜索模块的快照查看用例逻辑类图如下图6.6所示：

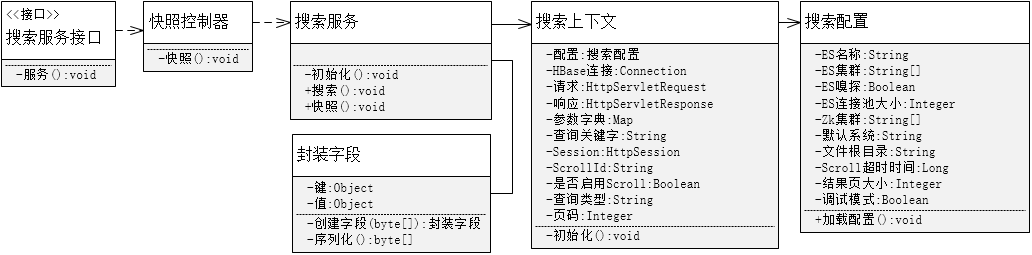


图 6.6 快照查看逻辑类图

Graph 6.6 Snapshot case logic class

### 5.2.3 业务逻辑交互分析

实现一个用例涉及到边界类、控制类和实体类三种逻辑要素，这三种类通过消息交互完成用例功能。以下为通过B.。。CE模式描述的用例功能逻辑设计，包含功能逻辑模型和说明。

1. 关键字搜索

用户的搜索请求通过搜索服务接口传递到控制器，由搜索服务进行处理，其逻辑模型如下图6.11所示：

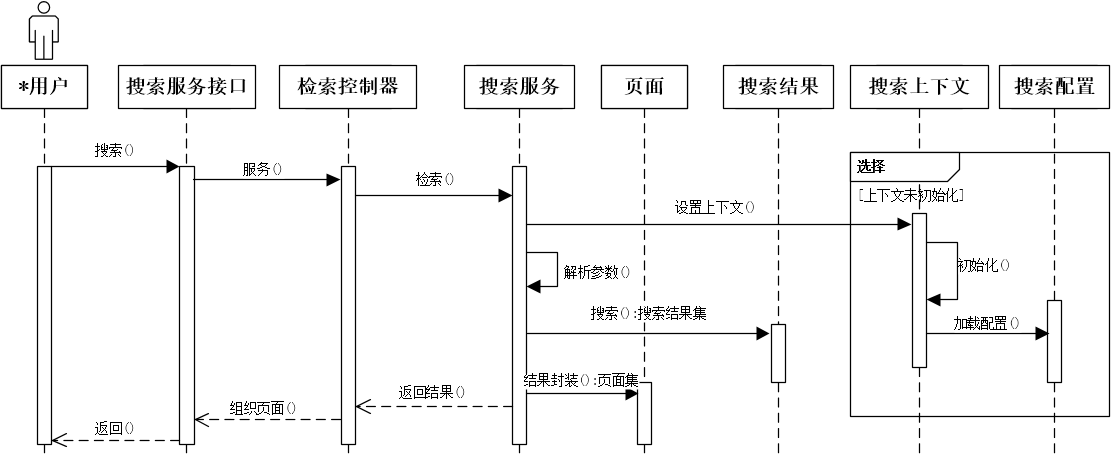


图 6.11 关键字搜索功能逻辑模型

Graph 6.11 Search case logic model

对功能逻辑的说明如下表6.1所示：

表 6.1 关键字搜索功能逻辑模型说明

Table 6.1 Search case logic model detail

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 说明 | 资源 |
| \*用户 | 参与者 | \*表示任何匿名发起者 | - |
| 搜索服务接口 | 边界类 | 搜索服务对外Servlet | WebSearchServlet.java |
| 检索控制器 | 控制类 | 搜索服务调用控制 | WebSearchHandler.java |
| 搜索服务 | 实体类 | 搜索服务业务实现 | SearchService.java |
| 页面 | 实体类 | 分页数据封装实体 | PageHolder.java |
| 搜索结果 | 实体类 | 搜索结果解析存储的实体 | SearchResult.java |
| 搜索上下文 | 实体类 | 搜索执行过程上下文 | DefaultContext.java |
| 搜索配置 | 实体类 | 搜索服务控制配置 | DefaultContext$Config.class |

**5.2数据库设计**

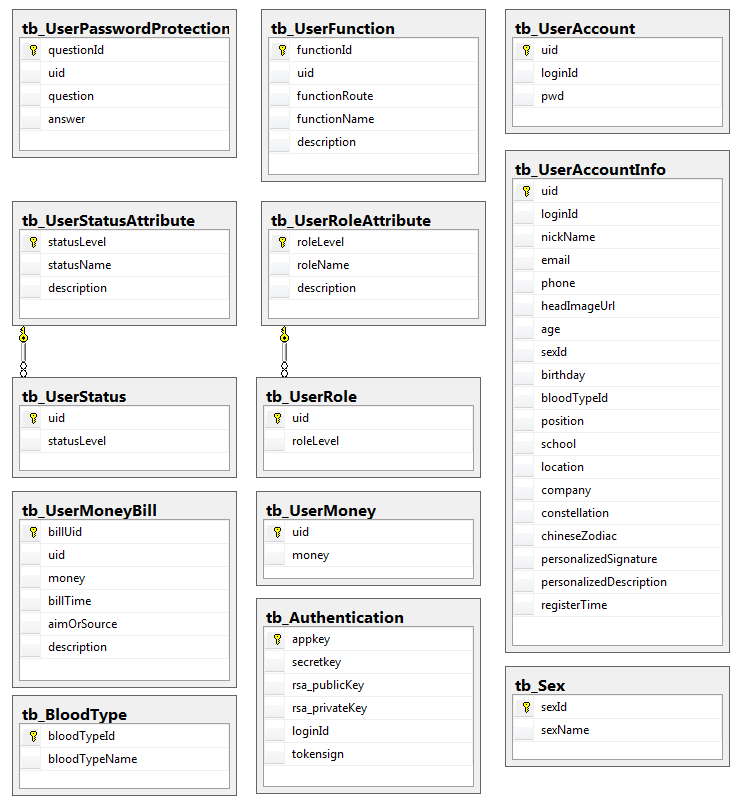
根据前面的业务对象模型设计接口，可以很轻松的分析出本系统所需要数据库表以及表之间的主外键关联关系。最终的出所需要设计的数据表为：用户账户表、用户角色表、角色描述表、用户状态表、状态描述表、用户信息表、用户血型表、用户性别表、用户账目表、账单记录表、Token认证表、密保问题表、系统功能权限表。在系统表的主外键设计过程中，由于系统的初衷是通用的用户管理模板，基于插拔式的用户管理系统，在后续的系统架构中对系统的扩展性要求较高，因此系统在主外键关联关系上并不是很紧密的设计，都是为了能在后续便于扩展。通过舍弃系统表之间的范式约束，得到了系统的松耦合、高复用性、高扩展性的优势，凡事有舍必有得，由此可以看出。

本系统设计的数据库都是以用户账号表为核心，在系统的实现中也是先实现用户账号的业务流程，再实现其他后续的业务流程，因为所有和用户账号相关联的表都是以用户账号表的UID字段为基础进行后续的实现扩展的，这里在对表的操作需要用事务进行。

下面先以图示的方式将系统用户管理、Token认证部分的数据库表展示出来，如图所示：



在上面的数据库表结构展示中，我们可以清晰地看到我们业务逻辑实现所要依赖的数据库表，在上述的表结构中缺少了对主外键的关联展示，我们用Microsoft SQL Server 自带的数据库表关联图示来进行对表之间的关联关系进行展示，如图所示：



依据设计，我们的用户账号表和用户信息表等其他表之间的关联不用主外键的强关联形式，而用统一的Uid来进行表之间的关联，在对数据库的操作中，如果涉及到对表之间关系冲击操作，我们会在代码段中用事务的方法来进行统一的管理。

1. 用户账户表tb\_UserAccount

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户信息表tb\_UserAccountInfo

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户密保表tb\_UserPasswordProtectionQuestion

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户角色表tb\_UserRole

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户角色描述tb\_UserRoleAttribute

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户状态tb\_UserStatus

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户状态描述tb\_UserStatusAttribute

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户血型tb\_BloodType

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户性别tb\_Sex

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户账目tb\_UserMoney

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 用户账单tb\_UserMoneyBill

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. Token认证实体tb\_Authentication

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

1. 系统功能实体tb\_UserFunction

|  |  |  |
| --- | --- | --- |
| 键名 | 类型 | 说明 |
|  |  |  |
|  |  |  |

# 第四章 系统实现与代码编写

## 4.1系统实现

通过为期15天的项目编码周期，本项目编码工作如期完成，在项目编码过程中出现若干bug，在逐一进行改正后项目代码按预期正常运行。

1. 项目整体结构

项目由ASP.NET MVC及三层架构开发完成，以下是项目的组织结构图。

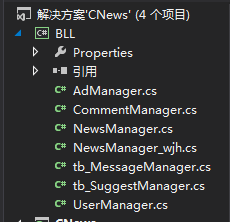
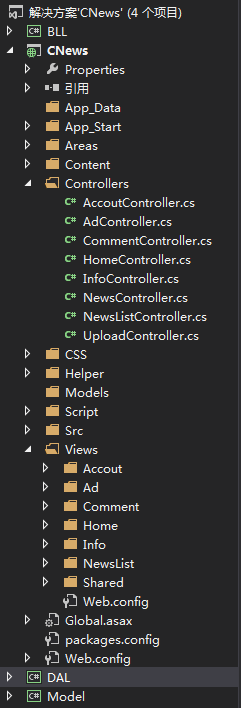


图6.1.1项目组织结构图1

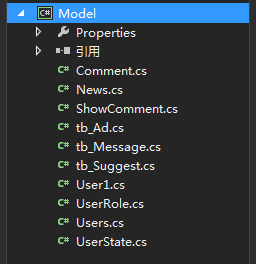
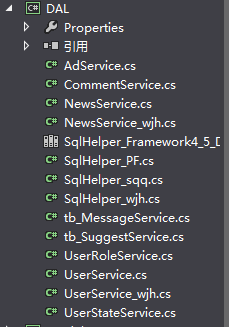


图6.1.2项目组织结构图2

项目类文件结构：

表6.1.1项目组织结构表

|  |  |  |
| --- | --- | --- |
| 命名空间 | 类名 | 说明 |
| CNews.Controllers | 控制器层 | |
| AccountController.cs | 账户控制器类 |
| AdController.cs | 广告业务控制器类 |
| HomeController.cs | 主页面视图控制器类 |
| InfoController.cs | 系统信息视图控制器类 |
| NewsController.cs | 新闻控制器类 |
| NewsListController.cs | 新闻列表控制器类 |
| BLL | 业务逻辑层 | |
| AdManager.cs | 广告操作业务逻辑类 |
| NewsManager.cs | 新闻操作业务逻辑类 |
| Tb\_MessageManager.cs | 系统消息业务逻辑类 |
| Tb\_SuggestManager.cs | 用户建议业务逻辑类 |
| UserManager.cs | 用户管理业务逻辑类 |
| DAL | 数据访问层 | |
| AdService.cs | 广告数据库操作类 |
| NewsService.cs | 新闻数据库操作类 |
| Tb\_MessageService.cs | 系统消息数据库操作类 |
| Tb\_SuggestService.cs | 用户建议数据库操作类 |
| UserService.cs | 用户管理数据库操作类 |
| SqlHelper\_Framework4\_5\_DG.dll | 数据库操作帮助类库 |
| Model | 模型层 | |
| News.cs | 新闻模型类 |
| Tb\_Ad.cs | 广告模型类 |
| Tb\_Message.cs | 系统消息模型类 |
| Tb\_Suggest.cs | 用户建议模型类 |
| User.cs | 用户模型类 |

## 4.2功能代码

本网站项目功能代码详见附录一。

# 第五章 测试

## 5.1测试的目的

本新闻网站旨在构建一个公众及时了解各界新闻的平台，在系统的界面上保持整洁，简单的风格。在系统的功能上采用简约易上手的原则，为社会公众提供便利。本网站的测试工作就是基于此要求来进行网站的测试，在对网站的界面功能进行改进的同时尽最大努力来发现软件的缺陷，以便给公众提供良好的产品，为社会大众负责。

## 5.2测试的过程

在本项目主要用到白盒测试和黑盒测试两大部分，白盒测试又主要用于单元测试和集成测试中，单元测试在系统项目编码过程中进行测试完成，集成测试在项目正式发布过程中进行测试。黑盒测试用于在项目的系统测试以及项目验收阶段进行整体的测试。本项目中，按照系统需求规格说明书和设计文档进行了相关的测试，以便测试项目是否达到了需求规格说明书中提到的项目需求以及项目的整体健壮性是否达到标准。

一、测试参考文档

《软件项目计划》

《软件测试》

《软件确认测试用例》

二、测试环境与配置

测试环境：

系统服务器：腾讯云服务器 182.254.210.156

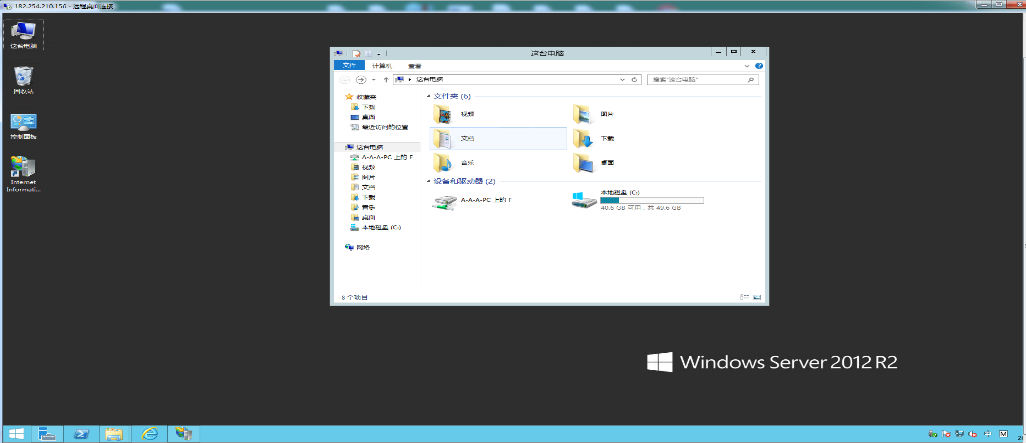


图7.2.1软件测试服务器平台截图

数据库：sqlserver2012在服务器中安装完成

客户端：windows7旗舰版IE9.0

网络：校园网

三、测试方法

本项目网站采用黑盒测试中的等价类划分方法进行测试。

四、测试

用例设计及实现：

1、广告服务测试用例及实现

表7.3.1广告服务测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例 | 测试截图 | 测试结果 |
| 1 | 输入测试用例：空值 |  | 输入为空，输入框晃动提醒 |
| 2 | 测试用例空值提交 |  | 二次提醒机制 |
| 3 | 正常数值验证 |  | 通过客户端验证 |
| 4 | 正常数值提交结果 |  | 通过验证并且存在二次确认机制 |
| 5 | 超出50个字符限制 |  | 存在前台50个字符限制控制，不可以输入超过50个字符 |
| 6 | 数据库重复验证  输入用例中新社… |  | 第三重验证：数据库重复性验证、达到要求 |
| 7 | 输入url地址/info/guanggaofuwu |  | 跳转到广告管理页面并且正常显示广告信息第一页 |
| 8 | 输入删除指令 |  | 二次提醒是否删除 |
| 9 | 广告回复页面url |  | 成功加载到删除的广告信息 |
| 10 | 恢复指令输入 |  | 恢复信息二次提醒 |
| 11 | 广告服务页面url  /info/guanggaofuwu |  | 成功加载到恢复的广告信息 |
| 12 | 广告停用指令 |  | 输入广告停用指令成功停用广告 |
| 13 | 输入广告修改指令 |  | 成功加载广告信息到修改区域 |
| 14 | 输入xinhuashe.cn |  | 成功修改了广告信息 |

2、新闻展示测试用例及实现：

表7.3.2新闻展示测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例 | 测试截图 | 测试结果 |
| 1 | 输入网站url测试用例/Home/Index |  | 网站新闻正常加载 |
| 2 | 输入网站url  /NewsList/Index  ?CategoryId=国内 |  | 参数传递正确表达 |
| 3 | 输入错误的测试用例  ?categryId=方法 |  | 参数未能找到对应结果 |

3、系统消息测试用例及实现：

表7.3.3系统消息测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例 | 测试截图 | 测试结果 |
| 1 | 进入个人用户管理部分链接  /Accout/Mail |  | 正确显示个人系统消息（有新提示） |
| 2 | 对个人信息进行操作  /Accout/Update  Message |  | 应该成功接收到对应的系统提示 |
| 3 | 链接到系统消息  /Accouot/Mail |  | 成功接收到一条新的系统消息，通知刚才作出的修改 |
| 4 | 后台管理员管理修改用户信息  输入电话号码改为44 |  | 应该收到对应的系统提示，见下一个测试用例 |
|  | 登录用户账号  Sqq：123456 |  | 成收到管理员更改用户信息的消息提示 |

4、用户建议测试用例及实现：

表7.3.4用户建议测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 测试用例 | 测试截图 | 测试结果 |
| 1 | 空值测试用例 |  | 通过提交意见（此处应为检测机制） |
| 2 | 重复提交测试用例 |  | 不能重复提交（检测机制） |
| 3 | 正常建议输入测试用例 |  | 可以通过验证并且提交 |
| 4 | 后台链接/Admin/User/Suggest |  | 正确加载用户建议信息 |
|  | 忽略用户建议指令操作 |  | 正常 |

五、测试情况

通过新闻网站的黑盒测试，发现了诸多系统问题，在下一部分缺陷统计及缺陷分析部分会详细对测试用例的测试情况作为总结。通过系统黑白盒的测试，系统基本完成了需求规格说明书中的需求说明，达到了预期的目标，系统存在一些细节上的问题，不影响系统的整体使用。

六、缺陷的统计及缺陷分析

在系统的白盒测试中对系统的编码单元进行了测试，发现了诸多bug并且在团队成员的努力下修正了bug保证了系统的正常运行。

在系统的黑盒等价类划分测试中，通过设计响应的测试用例，发现了在用户建议模块中用户建议的输入没有进行前台验证，不能保证用户输入的建议信息对于系统管理方面的有效性，函待改正。

七、测试结论及建议

结合本新闻网站项目的软件测试工程来看，测试的进行是非常必要的，也是非常重要的一个环节，通过测试我们发现了诸多在编码阶段甚至是程序设计，需求设计阶段的一些漏洞已经未能考虑周全的部分，也通过测试进行了逐一的改正，总而言之，充分的测试对于我们程序本身来说是一种质的飞跃，对于程序员来说更是一笔不可多得的收获。

在测试中发现的一系列问题，对于今后的工作学习生活都是一份难得的经验，对我们自身来说也是一种提升。当然测试只是尽可能地发现系统的漏洞，不可能百分百覆盖到系统漏洞，因此，在测试阶段我们努力吸收前人的教训的同时更应该注重创新，我们应该打破固有的思维模式，才能在测试过程中发现更多的有价值的漏洞。

# 参考文献

[1] 《质量管理与质量控制》 作者: (美)詹姆斯·R·埃文斯//威廉·M·林赛 中国人民大学出版社 译者：焦叔斌 2010-5

[2] 《C#从入门到精通》 作者: 明日科技 清华大学出版社 2012-9

[3] 《C#高级编程（第9版）》 Christian Nagel清华大学出版社

[4] 《SQL Server2008编程入门经典》作者:Robert Vieria清华大学出版社

[5] Andrew Troelsen 著 朱晔 译《C#与.NET 4高级程序设计》 2011年4月 人民邮电出版社

[6] 王家华.软件工程[M].东北大学出版社.2001

[7] 张俊 崔海波 著 《ADO.NET数据库应用开发》2008年1月机械工业出版社

[8] 张越廷 顾彦玲 著《ASP.NET从入门到精通》2008年9月 清华大学出版社

[9] 徐孝凯 贺佳英 著 《数据库基础与SQL Server应用开发》2008年4月 清华大学出版社

[10]侯志荣. WEB应用程序开发-算法分析与应用 [M].北京：人民邮电出版社,2000.1

[11]萨师煊，王小珊. 数据库系统概论（第三版）[M] 北京：高等教育出版社，2008.6

[12]Stephen Walther.ASP.NET 3.5 Unleashed.2008-01 Sams

[13]马伟.ASP.NET 4权威指南.机械工业出版社,2011.1.

[14]Adam Freeman Steven Standerson.Pro ASP.NET MVC 3 Framework . Apress,2012.5.

[15]张昌龙,辛永平.ASP.NET 4.0从入门到精通. 机械工业出版社,2011.1.

[16]李彦,高博,唐继强,许惠彬.ASP.NET 4.0 MVC敏捷开发给力起飞. 电子工业出版社,2011 .9.

# 附录

附录一、注册接口代码

// POST: api/User/loginId mail click register api

public IHttpActionResult Post(string loginId)

{

if (string.IsNullOrEmpty(loginId))

{

throw new Exception\_DG("loginId", "loginId must be provide.", 1002);

}

bool addSuccess = true;

tb\_UserAccount userAccount = tb\_UserAccount.Build();

userAccount.uid = Guid.NewGuid();

userAccount.loginId = loginId;

object pwd\_mail\_cache = Cache\_Helper\_DG.Cache\_Get(loginId);//get pwd and mail from cache

if (pwd\_mail\_cache == null)

{

throw new Exception\_DG("register info expired,please register renew.", 3003);

}

using (var fact = Wcf<UserAccountService>())

{

var channel = fact.CreateChannel();

int userAccountCountByloginId = channel.QueryCount(new tb\_UserAccountQueryObject { QueryCondition = t => t.loginId.Equals(loginId) });

if (userAccountCountByloginId > 0)

{

throw new Exception\_DG("the loginId has been exist!", 3002);

}

}

userAccount.pwd = pwd\_mail\_cache.ToString().Split(',')[0];

Transaction\_Helper\_DG.Transaction((Action)(() =>

{

//add tb\_UserAccount

using (var fact = Wcf<UserAccountService>())

{

var channel = fact.CreateChannel();

addSuccess = addSuccess && channel.Add(userAccount);

}

//add tb\_UserAccountInfo

using (var fact = Wcf<UserAccountInfoService>())

{

var channel = fact.CreateChannel();

addSuccess = addSuccess && channel.Add(new tb\_UserAccountInfo { uid = userAccount.uid, loginId = userAccount.loginId, nickName = loginId, email = pwd\_mail\_cache.ToString().Split(',')[1], registerTime = DateTime.Now, birthday = DateTime.Now });

}

//add tb\_UserStatus

using (var fact = Wcf<UserRoleService>())

{

var channel = fact.CreateChannel();

addSuccess = addSuccess && channel.Add(new tb\_UserRole { uid = userAccount.uid, roleLevel = 0 });

}

//add tb\_UserStatus

using (var fact = Wcf<UserStatusService>())

{

var channel = fact.CreateChannel();

addSuccess = addSuccess && channel.Add(new tb\_UserStatus { uid = userAccount.uid, statusLevel = 0 });

}

}));

if (!addSuccess)

{

throw new Exception\_DG("register error.", 3004);

}

long timeStamp = DateTime\_Helper\_DG.GetCurrentTimeStamp() \* 1000;

int random = new Random().Next(100, 999);

//get rsa keys

RSA\_Keys rsa\_Keys = RSA\_GetKeys();

tb\_Authentication authentication;

using (var fact = Wcf<AuthenticationService>())

{

var channel = fact.CreateChannel();

authentication = channel.QuerySingle(new tb\_AuthenticationQueryObject { QueryCondition = t => t.loginId.Equals(loginId) }).Cast<tb\_Authentication>();

if (authentication == null)

{

authentication = tb\_Authentication.Build();

//secretKey=MD5[loginid+MD5[pwd]+timestamp]

authentication.secretkey = MD5\_Encrypt($"{loginId}{userAccount.pwd}{timeStamp}");

authentication.rsa\_publicKey = rsa\_Keys.publicKey;

authentication.rsa\_privateKey = rsa\_Keys.privateKey;

authentication.loginId = loginId;

//tokenSign=MD5[loginid+logintimestamp+random]

authentication.tokensign = MD5\_Encrypt($"{loginId}{timeStamp}{random}");

if (channel.Add(authentication))

{

authentication = channel.QuerySingle(new tb\_AuthenticationQueryObject { QueryCondition = t => t.loginId.Equals(loginId) }).Cast<tb\_Authentication>();

}

}

else

{

//secretKey=MD5[loginid+MD5[pwd]+timestamp]

authentication.secretkey = MD5\_Encrypt($"{loginId}{userAccount.pwd}{timeStamp}");

authentication.rsa\_publicKey = rsa\_Keys.publicKey;

authentication.rsa\_privateKey = rsa\_Keys.privateKey;

authentication.loginId = loginId;

//tokenSign=MD5[loginid+logintimestamp+random]

authentication.tokensign = MD5\_Encrypt($"{loginId}{timeStamp}{random}");

channel.Update(authentication);

}

}

Int Auth\_Token\_ExpireTime\_days = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_days").ToInt();

int Auth\_Token\_ExpireTime\_hours = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_hours").ToInt();

int Auth\_Token\_ExpireTime\_minutes = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_minutes").ToInt();

long expireTimeStamp = DateTime\_Helper\_DG.GetTimeStampByDateTimeUtc(DateTime.UtcNow.AddDays(Auth\_Token\_ExpireTime\_days).AddHours(Auth\_Token\_ExpireTime\_hours).AddMinutes(Auth\_Token\_ExpireTime\_minutes));

//token=RSA\_publicKey[uid+loginid+expiretimestamp+tokensign]

string token = RSA\_Encrypt($"{userAccount.uid}&{userAccount.loginId}&{expireTimeStamp}&{authentication.tokensign}", authentication.rsa\_publicKey);

//send appkey,secretkey,token,userInfo to client

return Json(Return\_Helper\_DG.Success\_Msg\_Data\_DCount\_HttpCode("register user succeed , login succeed", new { loginId = userAccount.loginId, appKey = authentication.appkey, secretKey = authentication.secretkey, token = token }, 1));

}

附录二、判断是否登录接口代码

// GET: api/Login

public IHttpActionResult Get(int appKey,int random,long timeStamp,string token)

{

//timeStamp verification

bool isTimeStampValid = (DateTime\_Helper\_DG.GetCurrentTimeStamp() - timeStamp / 1000) <= Config\_Helper\_DG.AppSetting\_Get("RequestExpireTime").ToInt() \* 60;

if (!isTimeStampValid)

{

throw new Exception\_DG("request expired", 3006);

}

//[random+timestamp] can be find in cache?

if (Cache\_Helper\_DG.Cache\_Get($"{random}{timeStamp}") != null)

{

throw new Exception\_DG("request multiple", 3007);

}

Cache\_Helper\_DG.Cache\_Add($"{random}{timeStamp}", 1);//add [random+timestamp] into cache

tb\_Authentication authentication = AuthenticationController.GetAuthenticationByAppKey(appKey);

//get token array from decrypt token string

string[] tokenArray = Encrypt\_Helper\_DG.RSA\_Decrypt(token, authentication.rsa\_privateKey).Split('&');

//$"{userAccount.uid}&{userAccount.loginId}&{expireTimeStamp}&{authentication.tokensign}"

long expireTimeStamp = tokenArray[2].ToInt64();

string tokenSign = tokenArray[3];

if (expireTimeStamp < DateTime\_Helper\_DG.GetCurrentTimeStamp())

{

throw new Exception\_DG("login info expired please login renew", 3011);

}

if (!tokenSign.Equals(authentication.tokensign))

{

throw new Exception\_DG("account login elsewhere,please login renew", 3012);

}

return Json(Return\_Helper\_DG.Success\_Msg\_Data\_DCount\_HttpCode("account has been login",new { loginId=authentication.loginId},1));

}

附录三、登录接口代码

// POST: api/Login

public IHttpActionResult Post([FromBody]dynamic query)

{

if (query == null)

{

throw new Exception\_DG("arguments must be provide", 1001);

}

if (query.loginId == null)

{

throw new Exception\_DG("loginId must be provide", 1002);

}

if (query.random == null)

{

throw new Exception\_DG("random must be provide", 1006);

}

if (query.timeStamp == null)

{

throw new Exception\_DG("timeStamp must be provide", 1007);

}

if (query.secretString == null)

{

throw new Exception\_DG("secretString must be provide", 1009);

}

string loginId = query.loginId;

int random = query.random;

long timeStamp = query.timeStamp;

string secretString = query.secretString;

//timeStamp verification

bool isTimeStampValid =( DateTime\_Helper\_DG.GetCurrentTimeStamp() - timeStamp / 1000)<= Config\_Helper\_DG.AppSetting\_Get("RequestExpireTime").ToInt() \* 60;

if (!isTimeStampValid)

{

throw new Exception\_DG("request expired", 3006);

}

//[random+timestamp] can be find in cache?

if (Cache\_Helper\_DG.Cache\_Get($"{random}{timeStamp}") != null)

{

throw new Exception\_DG("request multiple", 3007);

}

Cache\_Helper\_DG.Cache\_Add($"{random}{timeStamp}", 1);//add [random+timestamp] into cache

//get MD5[pwd] from database

tb\_UserAccount userAccount;

using (var fact = Wcf<UserAccountService>())

{

var channel = fact.CreateChannel();

userAccount = channel.QuerySingle(new tb\_UserAccountQueryObject { QueryCondition = t => t.loginId.Equals(loginId.Trim()) }).Cast<tb\_UserAccount>();

}

if (userAccount == null)

{

throw new Exception\_DG("no user account found by loginId", 3001);

}

//MD5[loginid+MD5[pwd]+ramdom+timestamp]==MD5[secretMessage]?

bool secretStringMatched = MD5\_Encrypt($"{loginId}{userAccount.pwd}{random}{timeStamp}").Equals(secretString);

if (!secretStringMatched)

{

throw new Exception\_DG("the request has been tampered also mains account or pwd error", 3008);

}

//get rsa keys

RSA\_Keys rsa\_Keys = RSA\_GetKeys();

tb\_Authentication authentication;

using (var fact = Wcf<AuthenticationService>())

{

var channel = fact.CreateChannel();

authentication = channel.QuerySingle(new tb\_AuthenticationQueryObject { QueryCondition = t => t.loginId.Equals(loginId) }).Cast<tb\_Authentication>();

if (authentication==null)

{

authentication =tb\_Authentication.Build();

//secretKey=MD5[loginid+MD5[pwd]+timestamp]

authentication.secretkey = MD5\_Encrypt($"{loginId}{userAccount.pwd}{timeStamp}");

authentication.rsa\_publicKey = rsa\_Keys.publicKey;

authentication.rsa\_privateKey = rsa\_Keys.privateKey;

authentication.loginId = loginId;

//tokenSign=MD5[loginid+logintimestamp+random]

authentication.tokensign = MD5\_Encrypt($"{loginId}{timeStamp}{random}");

if (channel.Add(authentication))

{

authentication = channel.QuerySingle(new tb\_AuthenticationQueryObject { QueryCondition = t => t.loginId.Equals(loginId) }).Cast<tb\_Authentication>();

}

}

else

{

//secretKey=MD5[loginid+MD5[pwd]+timestamp]

authentication.secretkey = MD5\_Encrypt($"{loginId}{userAccount.pwd}{timeStamp}");

authentication.rsa\_publicKey = rsa\_Keys.publicKey;

authentication.rsa\_privateKey = rsa\_Keys.privateKey;

authentication.loginId = loginId;

//tokenSign=MD5[loginid+logintimestamp+random]

authentication.tokensign = MD5\_Encrypt($"{loginId}{timeStamp}{random}");

channel.Update(authentication);

}

}

int Auth\_Token\_ExpireTime\_days = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_days").ToInt();

int Auth\_Token\_ExpireTime\_hours = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_hours").ToInt();

int Auth\_Token\_ExpireTime\_minutes = Config\_Helper\_DG.AppSetting\_Get("Auth\_Token\_ExpireTime\_minutes").ToInt();

long expireTimeStamp = DateTime\_Helper\_DG.GetTimeStampByDateTimeUtc(DateTime.UtcNow.AddDays(Auth\_Token\_ExpireTime\_days).AddHours(Auth\_Token\_ExpireTime\_hours).AddMinutes(Auth\_Token\_ExpireTime\_minutes));

//token=RSA\_publicKey[uid+loginid+expiretimestamp+tokensign]

string token = RSA\_Encrypt($"{userAccount.uid}&{userAccount.loginId}&{expireTimeStamp}&{authentication.tokensign}", authentication.rsa\_publicKey);

//send appkey,secretkey,token,userInfo to client

return Json(Return\_Helper\_DG.Success\_Msg\_Data\_DCount\_HttpCode("login succeed", new { loginId = loginId, appKey = authentication.appkey, secretKey = authentication.secretkey, token = token }, 1));

}

# 致谢

在蚂蚁热帮（AntHelp）系统的研发中，我首先要感谢我的老师对我的帮助支持和鼓励，是我的老师从我的选题到整个毕业设计过程中孜孜不倦的教诲让我有充足的信心去完成我的毕业设计。并且在我遇到困难和问题时候能及时对我施以帮助，在我有做的不够完好的地方能及时纠正，保证了我毕业设计项目的完美完成。其次我要感谢我的团队成员，也是我很好的朋友们，是他们始终与我同仇敌忾，以一个共同的目标坚持不懈，同时正因为有了他们，才能在毕业设计中融入很多优秀的想法，也正因如此，我们的毕业设计才能优秀地完成。在团队开发中或许也会出现一些摩擦，但这都不是问题，每一次的小问题都促使了我们的成长，同时也是人生路上一段宝贵的经验，让我在以后遇到相同的问题时候波澜不惊，能冷静地去思考、处理。

除了项目中的参与人员外，我还要感谢我的父母和我亲爱的同学们，是他们一直在背后默默地鼓励我，支持我，才有了今天毕业设计很好的完成，是你们减轻了我背后的重担，我由衷地感谢你们。

在毕业设计完成之际，我还感谢我自己，能以一百二十分的精力去调研、分析、设计、实现整个系统，并且在整个过程中学到了很多在学校的课堂上学习不到的知识，在系统架构上的进步便是我对自己开发的框架日益完善的过程，经历了这段毕业设计，我所开发的QX\_Frame也日趋成熟，越来越能应对各种各样的项目需求，我很感激这次毕业设计，也很感激参与这次毕业设计的我。

虽然我们的项目接近尾声，但我始终相信，山外青山楼外楼，强中更有强中手。我们项目完成的很好，并不代表我们掌握了必备的知识，在漫漫人生路上，我们还有更多的知识需要我们去学习，还有更多的辉煌等着我们去创造！生命不息，奋斗不止！