****

****

**本 科 毕 业 设 计**

**院 系** 软件学院

**专 业** 软件工程

**题 目** eBay Friends项目服务端模块的设计与实现

**年 级** 09级 **学 号** 091250168

**学生姓名** 王子宁

**指导教师** 任桐炜 **职 称** 讲师

**论文提交日期**

**南京大学本科生毕业论文（设计）中文摘要**

毕业论文题目： eBay Friends项目服务端模块的设计与实现

软件学院 院系 软件工程 专业 09 级本科生姓名： 王子宁

指导教师（姓名、职称）： 任桐炜 讲师

摘要：

随着社交网络的普及，企业对社交网络在企业产品的推广中的作用愈发重视。与此同时，更多的电商平台开始研究如何使用社交网络的交流分享功能为企业带来更多利润。eBay Friends系统就是为了帮助用户在购买商品之后可以迅速分享自己的购物乐趣，同时，用户也可以从自己的社交圈子中的好友那儿获得最新的最快捷的最准确的商品信息，从而促进用户的购买。

项目开发过程中使用的主要技术是android开发和python语言。其中前台使用的android开发，后台使用的是基于python语言的django框架，使用mongoDB作为数据库，使用mongoengine作为服务器和数据库的连接。django框架的服务器具有简洁方便的特点，MongoDB在处理快速读写方面的性能优异，mongoengine能够将MongoDB与django中的models模块很好的结合。

整个项目组针对了eBay Friends用户购买商品后，实名进入系统并分享购物乐趣的整个过程，完成的发送新鲜事、获取新鲜事和评论新鲜事的主要功能，同时提供了购买商品的功能。

在整个项目开发的过程中，本人主要负责后台服务端模块的开发。主要包括django服务器的搭建、服务器请求的处理和返回、数据库的连接和MongoDB的搭建。

关键词：eBay Friends项目，python语言，django框架，MongoDB

**南京大学本科生毕业论文（设计）英文摘要**

THESIS：

DEPARTMENT：

SPECIALIZATION:

UNDERGRADUATE:

MENTOR:

ABSTRACT：

KEY WORDS:

# 摘 要

# 

**目 录**

[THESIS： III](#_Toc356501615)

[DEPARTMENT： III](#_Toc356501616)

[SPECIALIZATION: III](#_Toc356501617)

[UNDERGRADUATE: III](#_Toc356501618)

[MENTOR: III](#_Toc356501619)

[ABSTRACT： III](#_Toc356501620)

[KEY WORDS: III](#_Toc356501621)

[摘 要 I](#_Toc356501622)

[Abstract II](#_Toc356501623)

[图目录 V](#_Toc356501624)

[表目录 VI](#_Toc356501625)

[第一章 引言 7](#_Toc356501626)

[1.1 项目背景 7](#_Toc356501627)

[1.2 国内外社交网络NOSQL技术研究现状 8](#_Toc356501628)

[1.2.1 NOSQL数据库技术现状研究 8](#_Toc356501629)

[1.2.2 人人网后台NOSQL技术研究 9](#_Toc356501630)

[1.2.3 新浪微博后台NOSQL技术研究 10](#_Toc356501631)

[1.2.4 豆瓣网后台NOSQL技术研究 10](#_Toc356501632)

[1.2.4 Facebook后台NOSQL技术研究 10](#_Toc356501633)

[1.2.5 淘宝网后台NOSQL技术研究 10](#_Toc356501634)

[1.3 论文的主要工作 11](#_Toc356501635)

[1.4 论文的组织结构 11](#_Toc356501636)

[第二章 相关技术综述 12](#_Toc356501637)

[2.1 数据库技术综述 12](#_Toc356501638)

[2.1.1 NOSQL技术综述 12](#_Toc356501639)

[2.1.2 MongoDB技术详述 13](#_Toc356501640)

[2.2 服务器技术综述 14](#_Toc356501641)

[2.2.1 Python技术综述 14](#_Toc356501642)

[2.2.2 Django技术综述 14](#_Toc356501643)

[2.3 本章小结 15](#_Toc356501644)

[第三章 eBay Friends系统需求分析 16](#_Toc356501645)

[3.1 eBay Friends项目整体概述 16](#_Toc356501646)

[3.1.1 eBay Friends系统的前台概述 16](#_Toc356501647)

[3.1.2 eBay Friends系统的后台概述 16](#_Toc356501648)

[3.2 eBay Friends系统的需求分析 16](#_Toc356501649)

[3.2.1eBay Friends系统需求概述 16](#_Toc356501650)

[3.2.2 eBay Friends系统服务端需求分析。 17](#_Toc356501651)

[3.2.2 eBay Friends系统用例图 22](#_Toc356501652)

[3.2.3 eBay Friends项目服务端顺序图 23](#_Toc356501653)

[3.2.4 eBay Friends项目服务端活动图 25](#_Toc356501654)

[3.2.5 eBay Friends系统ERD 26](#_Toc356501655)

[3.3 本章小结 27](#_Toc356501656)

[第四章 eBay Friends项目服务器模块设计 28](#_Toc356501657)

[4.1 eBay Friends系统的概要设计 28](#_Toc356501658)

[4.1.1 eBay Friends系统前台概要设计 28](#_Toc356501659)

[4.1.2 eBay Friends系统后台服务器概要设计 29](#_Toc356501660)

[4.1.3 eBay Friends系统数据库概要设计 30](#_Toc356501661)

[4.1.4 eBay Friends系统接口描述 32](#_Toc356501662)

[4.2 服务端模块详细设计 33](#_Toc356501663)

[4.2.1 服务器模块概述 33](#_Toc356501664)

[4.2.2 数据库模块概述 33](#_Toc356501665)

[4.2.3 服务器模块的详细设计 34](#_Toc356501666)

[4.2.4 服务器模块复杂任务详细设计 35](#_Toc356501667)

[4.2.5 数据库模块的详细设计 37](#_Toc356501668)

[4.2.6 数据库模块复杂任务详细设计 39](#_Toc356501669)

[4.3本章小结 40](#_Toc356501670)

[第五章 eBay Friends项目服务器模块的实现 41](#_Toc356501671)

[5.1 服务端模块的实现 41](#_Toc356501672)

[5.1.1 服务器模块配置部分的实现 41](#_Toc356501673)

[5.1.2服务器模块功能需求的实现 42](#_Toc356501674)

[5.2 数据库模块实现 44](#_Toc356501675)

[5.3 服务器运行截图 45](#_Toc356501676)

[5.4 数据库运行截图 46](#_Toc356501677)

[5.5 本章小结 47](#_Toc356501678)

[第六章 总结与展望 48](#_Toc356501679)

[6.1 总结 48](#_Toc356501680)

[6.2 展望 48](#_Toc356501681)

[参考文献 50](#_Toc356501682)

# 图目录

[图3.1：eBay Friends系统用例图 23](#_Toc356501797)

[图3.2：eBay Friends系统顺序图1 24](#_Toc356501798)

[图3.3：eBay Friends系统顺序图2 25](#_Toc356501799)

[图3.4：eBay Friends系统活动图 26](#_Toc356501800)

[图3.5：eBay Friends系统ERD 27](#_Toc356501801)

[图4.1：eBay Friends系统前台设计包图 28](#_Toc356501802)

[图4.2：后台系统概要设计图 30](#_Toc356501803)

[图4.3：数据库设计图 32](#_Toc356501804)

[图4.4：服务器详细设计类图 34](#_Toc356501805)

[图4.5：登录以及获取新鲜事列表的顺序图 36](#_Toc356501806)

[图4.6：数据库collections设计 37](#_Toc356501807)

[图4.7：发送新鲜事详细设计顺序图 40](#_Toc356501808)

[图5.1：settings.py代码 41](#_Toc356501809)

[图5.2：urls.py代码 42](#_Toc356501810)

[图5.3：user模块中views.py代码 43](#_Toc356501811)

[图5.4：news模块views.py实现 44](#_Toc356501812)

[图5.5：news模块models.py的实现 45](#_Toc356501813)

[图5.6：users模块models.py的实现 45](#_Toc356501814)

[图5.7：服务器运行时截图 46](#_Toc356501815)

[图5.8：数据库运行时截图 47](#_Toc356501816)

# 

# 表目录

[表3.1：登录用例详细描述 17](#_Toc356501817)

[表3.2：浏览新鲜事用例详细描述 18](#_Toc356501818)

[表3.3：听取新鲜事声音用例详细描述 18](#_Toc356501819)

[表3.4：查看新鲜事评论用例详细描述 19](#_Toc356501820)

[表3.5：播放评论声音用例详细描述 19](#_Toc356501821)

[表3.6：评论新鲜事声音用例详细描述 20](#_Toc356501822)

[表3.7：获取购物历史用例详细描述 20](#_Toc356501823)

[表3.8：发送新鲜事用例详细描述 21](#_Toc356501824)

[表3.9：获取相关评论用例详细描述 21](#_Toc356501825)

[表4.1：django\_session表设计 37](#_Toc356501826)

[表4.2：goods表的设计 37](#_Toc356501827)

[表4.3：news表设计 37](#_Toc356501828)

[表4.4：user表设计 38](#_Toc356501829)

[表4.5：image.chunks表设计 38](#_Toc356501830)

[表4.6：image.files表设计 38](#_Toc356501831)

[表4.7：file.chunks表设计 39](#_Toc356501832)

[表4.8：image.files表设计 39](#_Toc356501833)

# 第一章 引言

## 1.1 项目背景

随着SNS(Social Network Service)社交网络的普及，企业对社交网络在企业产品的推广中的作用愈发重视。与此同时，更多的电商平台开始研究如何使用社交网络的交流分享功能为企业带来更多利润。

在电商购物平台网站中，很多都将客户的评价作为商品的重要信息布置在页面上，并以此作为买家和卖家之间交流的社交活动。这样的评价，虽然能够帮助买家判断商品是否合适，但是，也具有诸多弊端：

1. 由于评价的用户没有实行实名制，所以评价的真实性值得怀疑，存在大量不实数据。
2. 购物网站仅仅提供了文字的描述，不够直观，不具有足够的说服力。
3. 随着移动端应用的普及，越来越多的用户开始使用手机或者平板电脑等移动终端登录购物网站。这些往往用户会因为文字输入较为困难而放弃评价，或者采用默认的评价。

在常见的社交网络中，例如国内的人人网，新浪微博，国外的Facebook，twitter等，他们都采用实名注册的方式，从而有效的保证了用户看到的信息来源的可靠性和真实性。eBay Friends系统同样采用实名注册的方式，保证了信息的真实可靠。

随着大量的图片分享网站的出现，例如pinterest，花瓣网等，大量的用户愿意相信“眼见为实”，图片作为信息交流的载体在社交网络中开始普及。因此，考虑到文字描述不能够具有很强的说服力，eBay Friends系统采用图片作为信息交流的个体，提供图片的交流平台，保证信息真实性。

相比于传统的PC端，移动客户端输入文字较为不便，但是却可以方便的获取音频的输入。因此，eBay Friends系统提供了一个完整的音频录入、传输和存储的系统。

在社交网络中，很多用户会将自己刚刚购买的物品以照片的形式向朋友们展示并推荐。虽然这样的方式能够有效地推广产品，但是也有诸多弊端：

1. 用户在社交网络中发现自己的好友购买到了商品，但是却无从获得该商品的具体信息。
2. 用户从社交网络中只能获得自己的好友对该商品的评价和心得，却无法获得该商品的其他评价。
3. 如果用户对好友推荐的商品很感兴趣，但是却无法立即获得相应的购买链接，这样会使得用户失去购买的兴趣。

eBay Friends系统出了提供良好的社交网络，还能够同时连接到用户在eBay的数据系统。用户除了从好友那里获取相关信息，还可以获得该商品在eBay网站上的官方详细信息。

为了保证用户能够获取更多的信息，eBay Friends系统提供评价功能。以为用户发送一条购买商品的新鲜事后，其他用户可以直接对好友发出的新鲜事进行评价，也可以看到他人对该新鲜事的评价，用户们可以以此为平台对该商品进一步讨论。

除了提供社交功能，eBay Friends系统会提供购买链接。当用户看中好友推荐的某件商品，即可通过系统登录eBay网站进行购买，从而达到方便用户消费的目的。

目前的互联网产业中，使用的数据库主要有两种：关系型数据库（SQL）[[[1]](#endnote-2)]和非关系型数据库（NOSQL）

[[[2]](#endnote-3)]。由于社交网络的进一步发展，传统的关系数据库在应付超大规模和高并发的SNS类型的纯动态网站已显费力，对数据库高并发读写的需求，对海量数据的高效率存储和访问的需求，对数据库的高可扩展性和高可用性的需求等问题关系数据库遇到了难以克服的障碍。

eBay Friends系统由于有大量用户的并发操作，因此，对于数据库的并发性要求很高，同时，由于数据中存在大量的图片和音频等，对于海量数据的存储效率要求也很高。基于上述两个主要的原因，eBay Friends系统采用NOSQL数据存储的数据库。

## 1.2 国内外社交网络NOSQL技术研究现状

目前，NOSQL技术蓬勃发展，各种形式的开源NOSQL系统如雨后春笋般层出不穷。国内外很多社交网络为了应付越来越频繁的数据交互和越来越大量的数据存储，都采用了NOSQL数据库作为数据存储仓库。本文将介绍最新的各个NOSQL开源数据库，并以人人网、新浪微博、豆瓣网、facebook和淘宝网作为对象，进行比较研究。

### 1.2.1 NOSQL数据库技术现状研究

传统的关系型数据库读写操作都是事务的，具有ACID（原子性Atomicity、一致性Consistency、隔离性Isolation、持久性Durability）的特点，关系型数据库保证了数据的一致性和稳定性，但是带来的缺点是读写性能的下降。在有每天上亿次访问量的SNS网站面前，这样的缺点被无限放大。

于此相反，NOSQL因其数据存储不需要固定的表结构和相应的连接操作，

目前，由于NOSQL技术的开源性，各种NOSQL数据库层出不穷。比较著名的有HBase[[[3]](#endnote-4)]，Neo4j[[[4]](#endnote-5)]，MongoDB[[[5]](#endnote-6)]等：

1. HBase：

HBase是一个分布式的、面向列的开源数据库。Hstore存储是Hbase的核心。它由两部分构成：MemStore 和StoreFiles。数据存储过程中，现将所有对数据库的操作存储到MemStore中，MemStore满了之后存储成一个StoreFile。StoreFile文件数量增长到一定阈值，触发合并操作多个文件合并为一个大的StoreFile。

Hbase只有增加数据，所有的更新和删除操作都是在后续的合并过程中进行，这使得用户的写操作只要进入内存就可以立即返回，保证了Hbase I/O的高性能。

1. Neo4j：

Neo4j是一个嵌入式的、基于磁盘的、具备完全的事务特性的数据库。相对于关系型数据库来说，但是Neo4j将数据存储在图中而不是表中。图形数据库善于处理大量复杂、互联接、低结构化的数据。以图形方式存储数据可提高其数据查询效率。Neo4j可以应用更加敏捷和快速的开发模式，具有成熟和健壮的数据库的所有特性，它具有面向对象的、灵活的网络结构而不是严格、静态的表，同时可以享受到具备完全的事务特性、企业级的数据库的所有好处。

1. MongoDB：

MongoDB可以理解为是一个单一的数据库系统，同时加上了复制和分片功能。MongoDB的内部存储的数据是BSON[[[6]](#endnote-7)]文档，提供GridFS机制来存储二进制内容。MongoDB的写入方式是 in-place方式，即只要定义一个数组，然后传递给MongoDB的insert/update方法，就可以自动插入或更新，如果记录存在，那么更新，否则插入。MongoDB具有很高的写性能和写吞吐。

### 1.2.2 人人网后台NOSQL技术研究

目前，国内著名的社交网站人人网使用NOSQL技术主要是是人人开发团队自助研发的Nuclear。Nuclear系统构建于Java之上，具有高可拓展性，增加新节点的过程，系统服务无需停止，无需人工干预迁移数据；具有高可靠性，假设数据都存储在3个节点上，那么读的话必须成功读到两个节点上的数据才认为读成功，写的话必须成功写到两个节点上才认为写成功，系统永远可写入；具有高性能，单节点每秒最高可以达到2.5万次请求。

### 1.2.3 新浪微博后台NOSQL技术研究

目前，新浪微博使用的NOSQL技术主要是通过修改Redis源码满足自己的业务需求，完善它的replication机制，加入position的概念，让维护更容易，同时failover能力也大大增强。改善Hashset的存储方式，提升复杂数据类型的加载速度。此外，新浪微博中，NOSQL和MySQL是结合使用的，根据应用的特点选择合适存储方式。例如：索引使用MySQL存储；而一些K/V需求的，对并发要求比较高的放入NOSQL产品存储，或者通过关系数据复制到NOSQL来显示不同的应用需求。

### 1.2.4 豆瓣网后台NOSQL技术研究

目前，豆瓣网使用的NOSQL技术主要是BeansDB。BeansDB是由豆瓣网开发团队自主开发的主要针对大数据量、高可用性的分布式KeyValue存储系统，采用HashTree和简化的版本号来快速同步保证最终一致性。它具有高可用性、最终一致性、容易扩展、高性能和协议简单等优点。

### 1.2.4 Facebook后台NOSQL技术研究

目前，Facebook使用的NOSQL技术主要是Cassandra。除了Facebook之外，Twitter使用的也是这种NOSQL数据库技术。

Cassandra是一种面向列的NOSQL数据库，也是一种混合型的非关系数据库。同时，Cassandra又不是数据库，只是由一堆数据库节点共同构成的分布式网络服务。

Cassandra具有三个明显的特点：

1. Flexible schema：数据不需要像数据库一样使用预先设计的模式，增加或删除字段非常方便。
2. True scalability：单点故障不影响集群服务，集群的性能可线性扩展。
3. Multi-datacenter awareness：通过调整节点的布局避免某一个数据中心出现故障，一个备用的数据中心将至少有每条记录的完全复制。

Cassandra也因为这些优点，被多个SNS网站采用。

### 1.2.5 淘宝网后台NOSQL技术研究

目前，淘宝网使用的NOSQL技术主要是Oceanbase和Tair。

OceanBase把一段时间内的增删改等修改操作以增量形式记录下来保存在内存中，这样也使得了主体数据在一段时间内保持了相对稳定。OceanBase把它保存在独立的[服务器](http://baike.baidu.com/view/899.htm)UpdateServer的[内存](http://baike.baidu.com/view/1082.htm)中。以[内存](http://baike.baidu.com/view/1082.htm)保存增删改记录极大地提高了系统写[事务](http://baike.baidu.com/view/121511.htm)的性能。OceanBase既保持了关系型数据库的表结构的稳定性，同时增加了读写性能。

Tair是一个Key/Value结构数据的解决方案，它默认支持基于内存和文件的两种存储方式，分别和我们通常所说的缓存和持久化存储对应。Tair除了普通Key/Value系统提供的功能，比如get、put、delete以及批量接口外，还有一些其他专用的接口，用于提高数据库操作的速率。

## 1.3 论文的主要工作

本文主要介绍了eBay Friends系统在数据库设计和实现方面采用的技术和方法。用一个基于python[[[7]](#endnote-8)]语言的django[[[8]](#endnote-9)]服务器与Mongodb连接，从而实现服务器与数据库之间的信息传递。

## 1.4 论文的组织结构

第一章：概述和前言部分，主要介绍了eBay Friends项目的背景，研究当前国内外在社交网络数据存储领域的最新技术。

第二章：主要介绍Mongodb的具体技术，介绍django框架使用的基本知识。

第三章：从项目概述和需求分析两个方面，描述eBay Friends系统开发的整体过程。

第四章：对系统不同功能模块的详细设计和实现进行描述，主要包括服务器端的代码和数据库存取代码。

第五章：总结该项目已实现的功能，探讨项目的缺点和不足，并指出该项目未来的扩展和发展方向。

# 第二章 相关技术综述

## 2.1 数据库技术综述

### 2.1.1 NOSQL技术综述

由于现在社交网络迅速普及，人们与网络的信息交流量迅速增大，一个社交网络的数据库处理能力往往决定了这个社交网络的成败。尽管关系型数据库仍然是最广泛被使用的数据库，但是其固有的缺点也暴露无疑：无法实现高并发的write并发请求、无法处理海量数据的存储和快速查询、scale out高横向扩展性差等[[[9]](#endnote-10)]。正是因为这样的缺点，很多社交网络放弃关系型数据库，转而使用非关系型数据库，也就是NOSQL数据库。

NOSQL有时也称作Not Only SQL的简写，是对不同于传统的关系型数据库的数据库管理系统的统称[[[10]](#endnote-11)]。

NOSQL数据库大多以key-value的形式存储数据，能够打破传统关系型数据库对字段的束缚，能够更加灵活、更加自由的存储数据。此外，NOSQL支持横向的弹性扩展，当服务器的压力和负载到达某个定值，NOSQL数据库可以通过增加和删除节点，自由的实现负载均衡。而且，NOSQL可以部署在廉价的服务器集群上，大大节省了开发的成本。

NOSQL数据库在具有巨大优势的同时，也有很多缺点[[[11]](#endnote-12)]：

1. 成熟度。经过实践的证明，RDBMS满足了过去30年来的软件需求，为数据存储作出了巨大的贡献，无数的应用以RDBMS作为后端的存储。NOSQL的实际应用刚刚起步，但对企业来说，稳定压倒一切，关系型数据库的成熟度是呵以让人放心的，它运行稳定，功能也很丰富。相比而言，大部分NOSQL的产品都还处于实验阶段。
2. RDBMs有着强大的商业支持。Oracle、Microsoft和IBM等世界级的数据库厂商提供了完善的服务。相比而言，大部分的NOSQL数据库都是开源项目，虽然也有一些商业公司的支持，但支持的效果不可预计。如果出现故障，通常只能自己解决。
3. 软件开发人员已经习惯了RDBMS，大部分的开发人员的思想已经被固定在关系模型上，想要改变他们的思想是一件比较难的事情。使用NOSQL数据库的开发人员在短时间内很难快速增加。

由此可以看出，NOSQL的特点鲜明，十分适合用于作为高并发操作，并有着海量数据的社交网络数据库。

### 2.1.2 MongoDB技术详述[[[[12]](#endnote-13)]]

MonogoDB是一个基于文档的具有高性能、高可用性和易扩展性的数据库。文档数据库的优势主要体现在：MongoDB中存储的数据（对象）能够与编程语言中的数据结构完美匹配；嵌入式文档和数组的存储方式能够减少join操作；动态的存储模式使得多态更容易实现。高性能的优势主要体现在：嵌入式存储方式使得读写更加快捷；索引可以包括嵌入式文档和数组的键；可以使用流输入。高可用性的优势主要体现在：当出现故障时，采用复制服务器的方式自主修复。易扩展性的优势主要体现在：自动分片功能将数据散布在机器的不同部分；最终实现对不同复制服务器的分布数据的一致性读取。

MonogoDB的数据模型。一个MongoDB系统内有多个数据库，一个数据库内有多个聚集，一个聚集由一系列的文档组成，每个文档都是由很多对的key-value键值对组成。文档有动态的模式，动态模式意味着同一个聚集的文档不必采取相同的结构，一个聚集的文档可以存储不同类型的数据。

MongoDB的查询。MongoDB中的查询提供了一系列的操作符来定义find()函数如何基于具体查询从一个聚集中获取文档，使用结合结合精确相等匹配和条件使用查询操作符。

MongoDB的部署架构。尽管MongoDB支持一个“单独”或者单例的部署，MongoDB产品默认采用分布式部署。复制数据集可以在故障时提供高性能复制替换，而分片簇实现了将大数据分割到多个机器中，明显的呈现给客户。MongoDB的使用者结合了复制数据集和分片簇的功能，保证了大数据集的高层次的冗余。

MongoDB的设计哲学。现在的互联网技术需要满足数据层的横向扩展，易于部署和保存比过去使用的数据量呈指数型上升趋势的数据量的新型技术；非关系型的方式是处理横向扩展数据库到多个机器的最好方式；同时，如果这些新技术使得编程更加困难，那么也是不能接受的，代码会因为这项技术变得更快、更容易、更敏捷；文档数据模型（BSON）很容易应用于代码，易于管理，通过将相关的数据分组的方式使得它展现出完美的性能；通过保持深层次的函数来保持编程的快速简单是很重要的，尽管某些东西一定会被忽略；数据库技术应该能够在任何环境运行，无论是你自己的服务器或者虚拟机，还是作为一个付费的云服务。正式在这样的实际理念下，MongoDB应运而生，并因为它的灵活性、与关系型数据库的结合性、快速性和易用性在业界被广泛使用。

## 2.2 服务器技术综述

### 2.2.1 Python技术综述

Python是一种被广泛应用于应用程序的引人注目的强大的动态编程语言。Python的显著特点包括但不限于：

1. 清晰易读的语法：Python的语法十分简洁，用户可以花费更多的时间研究程序的功能或逻辑，而不是研究Python语法本身。
2. 直观的面向对象能力：Python既支持程序化编程也支持面向对象编程，与其他语言相比，Python实现面向对象的方法更为简洁。
3. 完整的模块性支持层次化的包：Python语言支持引用模块，可以很方便地利用其他类库实现多种功能。
4. 易于使用的用C，C++或者其他语言编写的扩展包或模块：Python支持第三方类库，而且不仅仅是Python语言编写的类库，而且包括很多基于C、C++或者其他语言的类库。
5. 内嵌于应用作为一个脚本接口：Python程序可以方便的作为一个脚本运行于其他的应用之中，兼容性很好。
6. 易学性：除了简单的语法，方便的使用方式，Python还是一个开源的软件，使用者可以方便的阅读源代码，学习十分方便。
7. 可移植性：Python适用于多种平台，无论是windows系统，Linux系统或者其他的操作系统，都可以搭建Python脚本运行的环境。

### 2.2.2 Django技术综述

Django是一个[开放源代码](http://zh.wikipedia.org/wiki/%E5%BC%80%E6%94%BE%E6%BA%90%E4%BB%A3%E7%A0%81)的Web应用框架，由Python写成。采用了MVC的软件设计模式，即模型M，视图V和控制器C。它最初是被开发来用于一些以新闻内容为主的网站的。

Django的主要目标是使得开发复杂的、数据库驱动的网站变得简单。Django注重组件的重用性和“可插拔性”，[敏捷开发](http://zh.wikipedia.org/wiki/%E6%95%8F%E6%8D%B7%E5%BC%80%E5%8F%91)和[DRY法则](http://zh.wikipedia.org/wiki/%E4%B8%80%E6%AC%A1%E4%B8%94%E4%BB%85%E4%B8%80%E6%AC%A1)（Don't Repeat Yourself）。在Django中[Python](http://zh.wikipedia.org/wiki/Python)被普遍使用，甚至包括配置文件和数据模型。Django 于 2008年6月17日正式成立基金会。

Django的命令行管理工具使得用户可以方便的创建管理Django的项目。Django中models模块定义的数据模型可以直接映射到后台的数据库中，数据库的API设计十分简洁。Django系统采用统一的urls.py文件管理整个网站的所有url，使用者可以直接通过一个文件实现对整个网站url结构的管理。Django系统中有一个自带的管理员，为开发人员节省了大量的时间。Django系统有一个自带的服务器，省去了调试服务器的时间，增加了开发效率。

## 2.3 本章小结

本章简单介绍了eBay Friends系统数据库模块设计的所有的技术，其中包括NOSQL技术（MongoDB数据库）和Python语言。系统结合了Django框架开发方便的特点和MongoDB对于音频图片处理的优势，保证了系统后台的高效快捷。

# 第三章 eBay Friends系统需求分析

## 3.1 eBay Friends项目整体概述

### 3.1.1 eBay Friends系统的前台概述

随着智能手机的普及，越来越多的用户通过手机或者平板电脑等设备接触互联网。在所有移动设备中，Android系统占据了很大的市场。因此，eBay Friends系统前台采用Android开发技术。

### 3.1.2 eBay Friends系统的后台概述

因为社交网络系统的特殊性，为了加强系统的反应能力和数据的读写存储效率。后台服务器采用Python作为脚本语言，采用MongoDB作为主要的数据库。为了开发的高效率，采用Django作为服务器主要框架，使用mongoengine插件连接服务器与数据库。

## 3.2 eBay Friends系统的需求分析

### 3.2.1eBay Friends系统需求概述

eBay Friends项目主要有功能性需求和非功能性需求。

功能性需求中，系统要实现以下用例：

1. 登录：用户使用用户名和密码登录系统
2. 获取新鲜事：用户获取最新的新鲜事信息
3. 播放新鲜事声音：用户播放某一条新鲜事的声音
4. 查看评论：用户查看新鲜事的评论信息
5. 播放评论：用户播放新鲜事评论的声音
6. 发表语音评论：用户对新鲜事进行评论（评论内容为语音）
7. 发表文字评论：用户对新鲜事进行评论（评论内容为文字）
8. 获取购物历史：用户获取自己购买过的商品
9. 选择物品：用户选择需要发送的新鲜事的对应商品
10. 选择图片：用户选择需要发送的图片
11. 美化图片：用户美化需要发送的图片
12. 录制语音：用户录制发送的语音内容
13. 美化语音：用户美化发送的语音内容
14. 发布新鲜事：用户发送新鲜事
15. 推送：评论用户推送评论给新鲜事发布者
16. 获取相关评论：用户获取所有的与自己相关的评论
17. 聊天：用户与其他用户实现聊天

系统需要实现的非功能性需求：

1. 在用户登录系统时，用户信息加密处理，避免出现用户隐私泄露的危险
2. 用户进行任何操作后，系统能及时进行相应。在网络顺畅时，系统响应时间应小于1秒钟；在网络不顺畅时，系统响应时间应小于3秒钟。

### 3.2.2 eBay Friends系统服务端需求分析。

在系统的所有需求中，对与服务器有交互的用例进行分析。

1、登录

用户下载eBay Friends系统后，需要登录，而后才可以使用系统，详细描述参见表3.1。

表3.1：登录用例详细描述

|  |  |
| --- | --- |
| ID | 1 |
| 名称 | 登录eBay Friends系统 |
| 描述 | 产生原因：客户端发送登录请求  大致过程： 服务端接收到发送的登录请求，进行用户名密码的匹配认证，返回结果。  输出结果：成功登录系统 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发送请求 |
| 后置条件 | 系统返回登录结果 |
| 正常流程 | 1. 服务端接收到登录请求。 2. 服务端从请求内容中获取用户名和密码 3. 服务端检查用户名和密码 4. 服务端返回登录结果 |
| 异常流程及解决方案 | 用户名密码检查失败：返回登录失败 |
| 相关用例 | 无 |

2、获取新鲜事

用户登录系统后，进入主页面，即可浏览刷新新鲜事，详细描述表3.2。

表3.2：浏览新鲜事用例详细描述

|  |  |
| --- | --- |
| ID | 2 |
| 名称 | 获取新鲜事 |
| 描述 | 产生原因：用户希望能够获得好友发布的最新消息。  大致过程：服务端接收到获取新鲜事请求，根据请求的内容参数，获取需要的新鲜事  输出结果：成功刷新新鲜事 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出获取新鲜事请求 |
| 后置条件 | 系统返回需要的新鲜事 |
| 正常流程 | 1. 服务端接收到新鲜事请求 2. 获取新鲜事的起始条目 3. 获取该用户的好友列表 4. 获取所有好友的新鲜事 5. 处理新鲜事的内容格式 6. 返回新鲜事 |
| 异常流程及解决方案 | 起始条目超过新鲜事所有条目：返回空列表 |
| 相关用例 | 1、3 |

1. 播放新鲜事声音

用户看到某条新鲜事后，希望获取新鲜事的声音，听取新鲜事声音。播放新鲜事声音用例的详细描述参见表3.3。

表3.3：听取新鲜事声音用例详细描述

|  |  |
| --- | --- |
| ID | 3 |
| 名称 | 播放新鲜事声音 |
| 描述 | 产生原因：用户发现某一条比较感兴趣的新鲜事，希望听取新鲜事的声音。  大致过程：服务端获取新鲜事声音请求，获取需要的音频信息，返回音频。  输出结果：成功听到新鲜事的声音。 |
| 优先级 | 高 |
| 前置条件 | 用户刷新出了最新的新鲜事列表。 |
| 后置条件 | 用户听到该新鲜事的声音 |
| 正常流程 | 1. 服务端接收到播放新鲜事声音的请求 2. 根据请求的参数取出需要的音频 3. 返回音频 |
| 异常流程及解决方案 | 播放声音不成功：返回空值 |
| 相关用例 | 2、4 |

4、查看新鲜事评论

用户刷新新鲜事列表，希望进入某一个新鲜事，查看其评论。详细描述参见表3.4。

表3.4：查看新鲜事评论用例详细描述

|  |  |
| --- | --- |
| ID | 4 |
| 名称 | 查看新鲜事评论 |
| 描述 | 产生原因：用户希望查看新鲜事的评论  大致过程：服务端接收到查看新鲜事评论的请求，获取请求的参数，根据参数获取需要的新鲜事的评论并返回。  输出结果：成功进入查看评论。 |
| 优先级 | 高 |
| 前置条件 | 用户刷新出了最新的新鲜事列表。 |
| 后置条件 | 用户成功进入新鲜事的详细信息页面。 |
| 正常流程 | 1. 服务端接收到查看新鲜事评论的请求。 2. 获取需要的新鲜事评论信息 3. 返回评论信息 |
| 异常流程及解决方案 | 获取评论不成功：返回空值 |
| 相关用例 | 2 |

5、播放评论声音

用户进入评论信息的页面后，希望播放某一条评论的音频。详细描述参见表3.5。

表3.5：播放评论声音用例详细描述

|  |  |
| --- | --- |
| ID | 5 |
| 名称 | 播放评论声音 |
| 描述 | 产生原因：用户希望播放某条评论的声音  大致过程：服务端接收到播放评论声音的请求，获取请求的参数，根据参数获取需要的声音并返回。  输出结果：成功播出声音。 |
| 优先级 | 高 |
| 前置条件 | 用户刷新出了最新的新鲜事列表。 |
| 后置条件 | 用户成功进入新鲜事的详细信息页面。 |
| 正常流程 | 1. 服务端接收到查看新鲜事评论的请求。 2. 获取需要的新鲜事评论信息 3. 返回评论信息 |
| 异常流程及解决方案 | 获取评论不成功：返回空值 |
| 相关用例 | 2 |

6、评论新鲜事

用户进入某条新鲜事的详细信息页面后，希望对此新鲜事进行评论。评论新鲜事用例的详细描述参见表3.6。

表3.6：评论新鲜事声音用例详细描述

|  |  |
| --- | --- |
| ID | 6 |
| 名称 | 评论新鲜事 |
| 描述 | 产生原因：用户发现某一条比较感兴趣的新鲜事，希望评论该新鲜事。  大致过程：服务端接收到评论新鲜事的请求，获取评论的对象和内容，并添加评论并返回结果。  输出结果：成功发送评论。 |
| 优先级 | 高 |
| 前置条件 | 客户端发送评论新鲜事请求 |
| 后置条件 | 用户成功评论该新鲜事 |
| 正常流程 | 1. 服务端接收到评论新鲜事的请求 2. 服务端获取评论的对象和内容 3. 对新鲜事添加评论 4. 返回发送结果。 |
| 异常流程及解决方案 | 评论不成功：返回失败。 |
| 相关用例 | 4,5 |

7、获取购物历史

用户在发送新鲜事前，需要获取自己购买过的商品的信息，选择需要的商作为品发送新鲜事对象。获取购物历史的详细描述参见表3.7。

表3.7：获取购物历史用例详细描述

|  |  |
| --- | --- |
| ID | 6 |
| 名称 | 获取购物历史 |
| 描述 | 产生原因：用户需要针对某一件曾经购买过的商品，发送新鲜事。  大致过程：服务端接收到获取购物历史的请求，服务器根据用户信息，返回购物的历史。  输出结果：成功获取购物历史。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发送获取购物历史的请求。 |
| 后置条件 | 用户成功获取到购物的历史记录。 |
| 正常流程 | 1. 服务端接收到客户端发送的获取购物历史的请求 2. 服务端获取用户信息 3. 服务端返回用户的购物历史 |
| 异常流程及解决方案 | 获取购物历史不成功：返回空值。 |
| 相关用例 | 7 |

8、发送新鲜事

用户登录系统后，希望发送一条新鲜事，详细描述参见表3.8。

表3.8：发送新鲜事用例详细描述

|  |  |
| --- | --- |
| ID | 8 |
| 名称 | 发送新鲜事 |
| 描述 | 产生原因：用户从eBay网站购买了商品，希望展示给好友。  大致过程：服务器接收到发送新鲜事的请求，获取请求中的内容信息，添加新鲜事。  输出结果：成功发送新鲜事。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出发送新鲜事的请求 |
| 后置条件 | 用户成功发送新鲜事。 |
| 正常流程 | 1. 客户端接收到发送新鲜事请求 2. 获取新鲜事对应的商品 3. 获取新鲜事的内容（图片和声音） 4. 添加新鲜事 5. 返回发送结果 |
| 异常流程及解决方案 | 发送失败：返回失败 |
| 相关用例 | 7 |

9、获取相关评论

用户获取所有的与自己相关的评论，详细描述参见表3.9。

表3.9：获取相关评论用例详细描述

|  |  |
| --- | --- |
| ID | 9 |
| 名称 | 获取相关评论 |
| 描述 | 产生原因：用户进入消息系统，希望获取所有相关的评论  大致过程：服务器接收到获取相关评论的请求，获取请求中用户信息，返回与该用户相关的评论。  输出结果：成功获取评论。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出获取评论的请求 |
| 后置条件 | 客户端成功获取所有相关评论 |
| 正常流程 | 1. 客户端接收到发送新鲜事请求 2. 获取请求中的用户信息 3. 获取该用户相关的评论内容 4. 返回评论内容 |
| 异常流程及解决方案 | 获取失败：返回空值 |
| 相关用例 | 无 |

### 3.2.2 eBay Friends系统用例图

eBay Friends系统用例图参见图3.1。

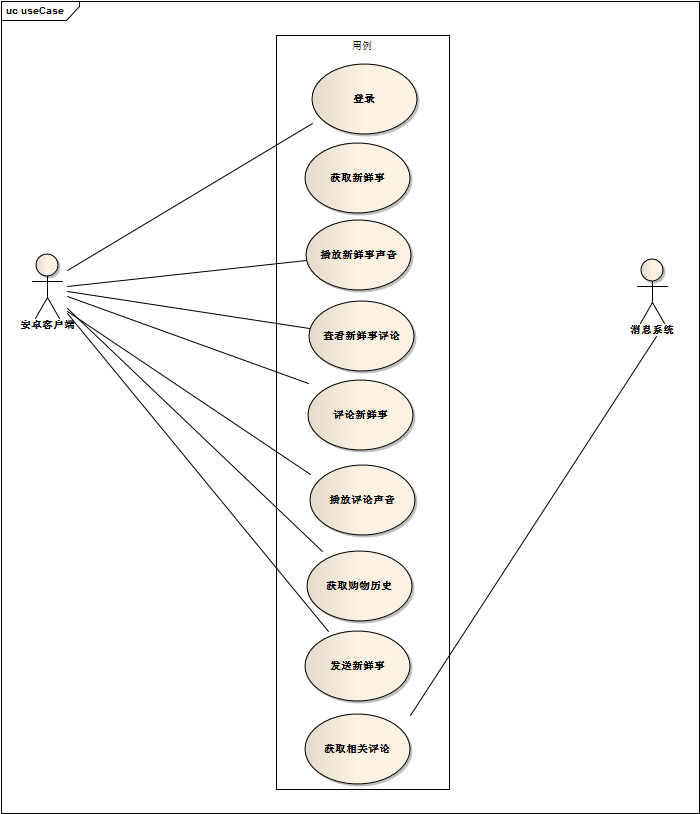


图3.1：eBay Friends系统用例图

### 3.2.3 eBay Friends项目服务端顺序图

eBay Friends系统用例1-6的详细顺序图，参见图3.2。

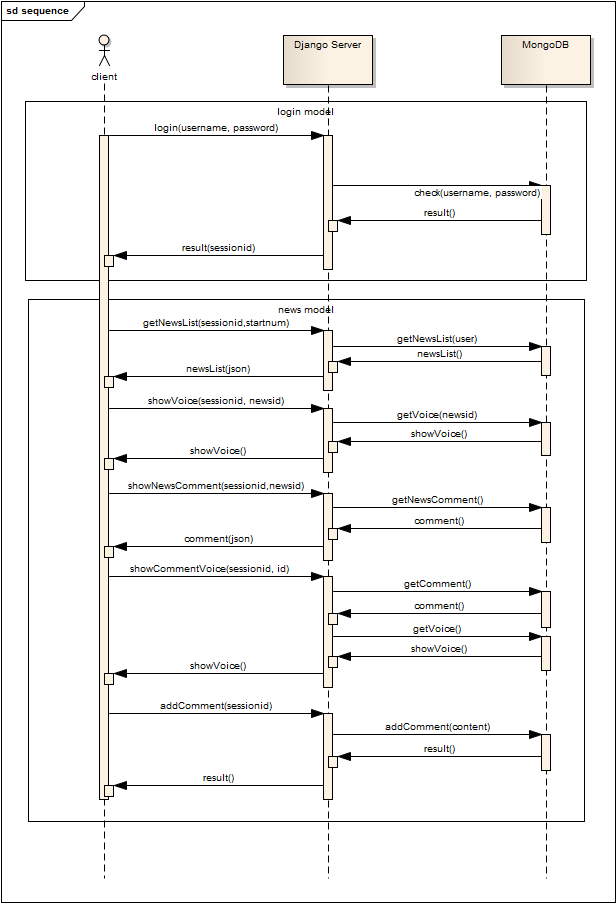


图3.2：eBay Friends系统顺序图1

eBay Friends系统用例7-9的详细顺序图，参见图3.3。

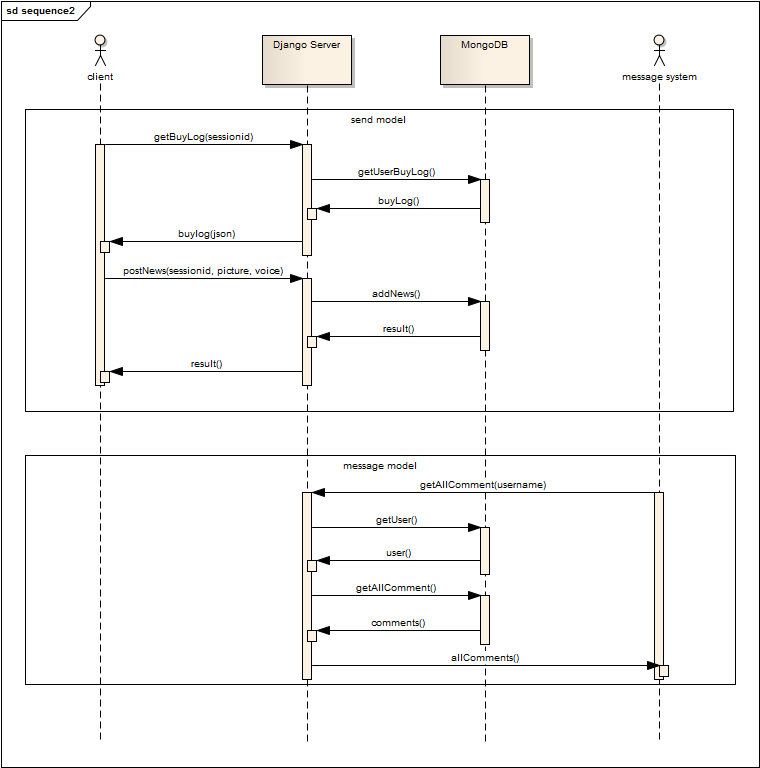


图3.3：eBay Friends系统顺序图2

### 3.2.4 eBay Friends项目服务端活动图

eBay Friends系统的详细活动图，参见图3.4。

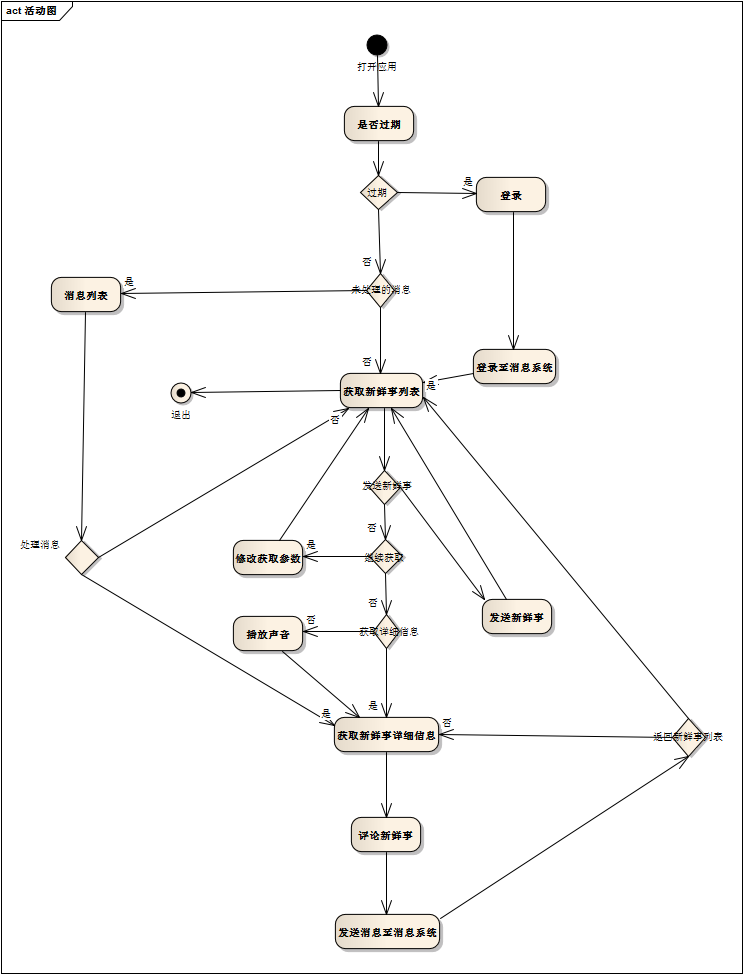


图3.4：eBay Friends系统活动图

### 3.2.5 eBay Friends系统ERD

eBay Friends系统数据的ERD如图3.5所示。

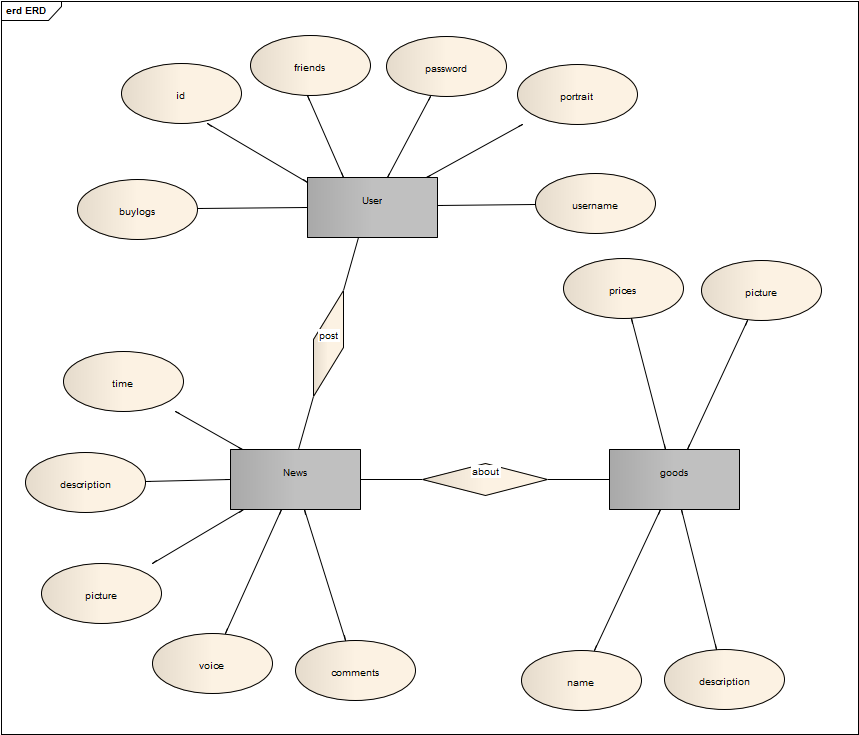


图3.5：eBay Friends系统ERD

## 3.3 本章小结

本章详细描述了系统的功能性需求和非功能性需求，针对本文完成的服务端模块进行了详细的分析。给出了详细的用例描述、用例图、顺序图和ER图。从需求方面对eBay Friends系统进行了分析。

# 第四章 eBay Friends项目服务器模块设计

## 4.1 eBay Friends系统的概要设计

### 4.1.1 eBay Friends系统前台概要设计

eBay Friends系统前台采用MVC设计。Models内容包括新鲜事模块，好友模块和消息模块；Views包括登录视图，新鲜事视图，发送新鲜事视图，功能列表视图，设置视图，推荐视图，消息视图和商品视图；Controller模块控制所有用例的流程。前台概要设计参见图4.1。

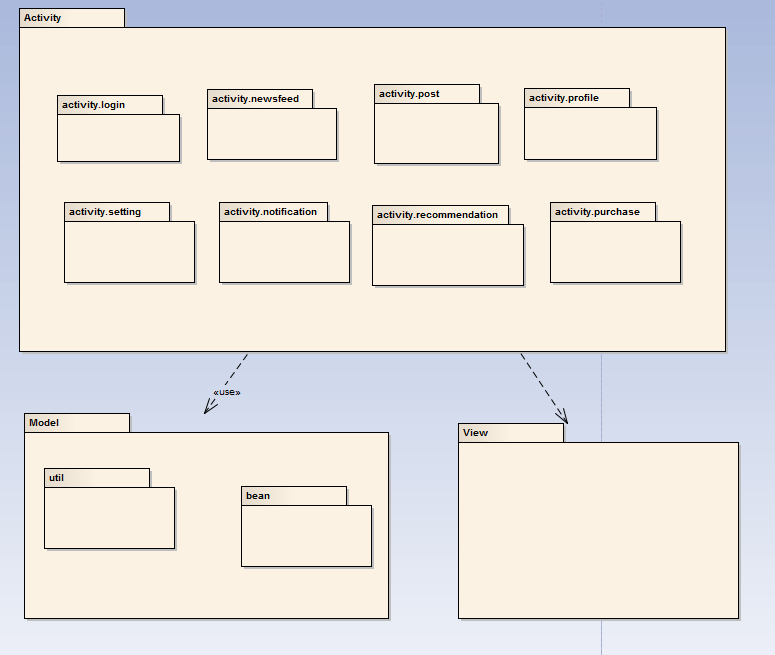


图4.1：eBay Friends系统前台设计包图

### 4.1.2 eBay Friends系统后台服务器概要设计

eBay Friends系统后台采用Django框架，Django框架与传统的MVC设计模式少有不同，采用独有的MTV模式，包括Models、Template和Views。由于前台使用的android开发，所以后台无须使用Template模板，只需要设计Models和Views。Models定义数据结构，与后台数据库的collection划分相对应，分为users、news和goods三个模块；Views与Models相对应，也分为users、news和goods三个模块，主要定义一些基本的方法。系统的概要设计如图4.2所示。

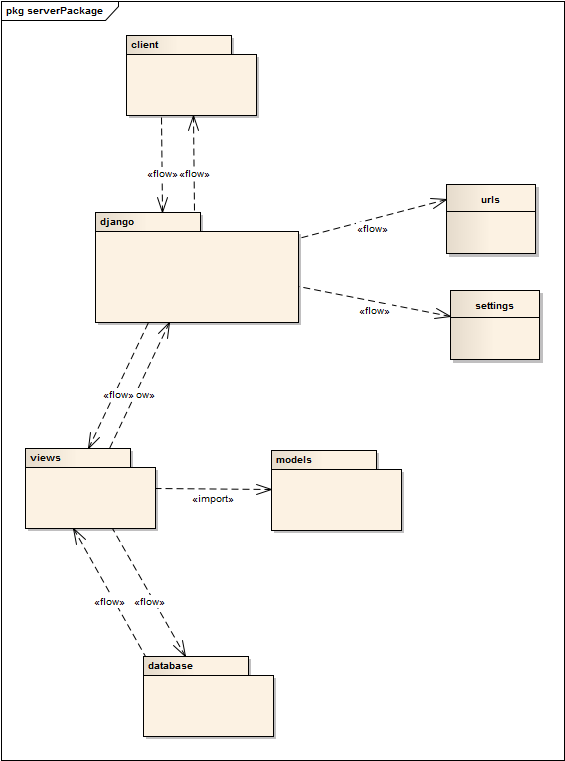


图4.2：后台系统概要设计图

### 4.1.3 eBay Friends系统数据库概要设计

eBay Friends系统后台数据库采用mongoDB，主要数据结构有user、news和goods三个。News有引用域，分别与user和goods相关联。此外，MongoDB中有django.session这个表，用于存放django服务器产生的session内容。

由于发送的内容有大量的图片和音频，MongoDB采用GridFS技术存进行文件存储。主要方式是：文件存储由表名.files和 表名.chunks 构成，前者是文件信息构成，后者是文件的内容，两者通过\_id与files\_id建立关联。上传上来的图片将会被分割成多个小的chunk，每个chunk将作被存储在chunks集合中，模块会创建files表，用于关联chunks。

数据库设计图，如图4.3所示。

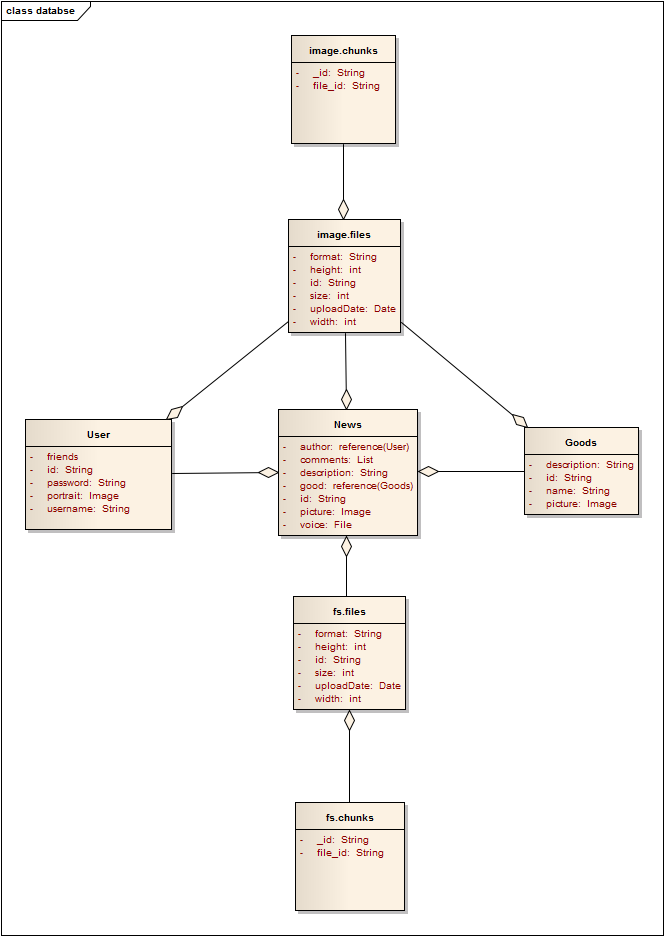


图4.3：数据库设计图

### 4.1.4 eBay Friends系统接口描述

eBay Friends系统大致分为四个模块：前端，服务器，消息系统和数据库。

前端与服务器之间的接口使用http协议，前端通过发送http请求获取需要的服务。所有http请求分为两种，get请求和post请求。服务器获取到请求后，通过url匹配，找到合适的方法。如果是接收到的是post请求，服务器解析客户端的json[[[13]](#endnote-14)]参数，将参数传入需要的方法，获取返回值；如果接收到的是get请求，服务器从url中获取parameter参数。传入需要的方法，获取返回值。返回值中，数据将以json的格式传到前端。

服务器与数据库通过mongoengine实现，在django服务器的配置文件中，将数据库引擎的参数设置成需要的MongoDB的相关参数。后台models文件中，将所有设计的数据库的类继承自mongoengine中的Document父类，用Document中的方法代替原先框架自带的数据库API。这样，数据将会以BSON的格式存储到数据库或从数据库中读出。

## 4.2 服务端模块详细设计

### 4.2.1 服务器模块概述

服务器运行在windows平台下，使用Django框架搭建。Django框架是一个基于python语言的开源的小型服务器。

Django框架具有开发简洁，开发效率高等优势。Django 的对象相关映射机制帮助开发者在 Python 类中灵活定义数据模型，并且Django 具有功能丰富的动态数据库访问 API，可以大幅度简化书写数据库语句的繁杂工作。此外，在本项目中，Django并不是与传统的SQL数据库相连，而是与NOSQL型的数据库MongoDB相连接。通过使用mongoengine作为数据库引擎，Django不仅可以自由向MongoDB数据库中存储数据，还能使用那些Django数据库原有的动态数据库访问API，节省了大量的数据库开发工作。Django 的 URL 分发设计的十分简洁美观，不会在链接中产生一大串杂乱且难以理解的字符。服务器可以根据前段发出的不同的url请求，对url字段进行匹配，找到对应的处理方法，省去了费时的url设计过程。

### 4.2.2 数据库模块概述

数据库木块主要采用MongoDB存储数据，主要存储的信息是：用户的个人信息，包括用户名、密码和头像；新鲜事信息，包括作者、描述、图片、声音、评论和对应商品；商品信息，包括名称、描述、价格和图片。除此之外，每条新鲜事的评论会以BSON的形式存储在新鲜事的collection中。

由于数据中涉及到大量的图片和音频的存储，后台采用GridFS作为存储格式，使用.files存储文件索引，通过id连接到.chunks表中。这样，大文件将以碎片的形式存储到数据库中，保证了读取时的高效率。

### 4.2.3 服务器模块的详细设计

服务器模块的详细设计类图，如图4.4所示。

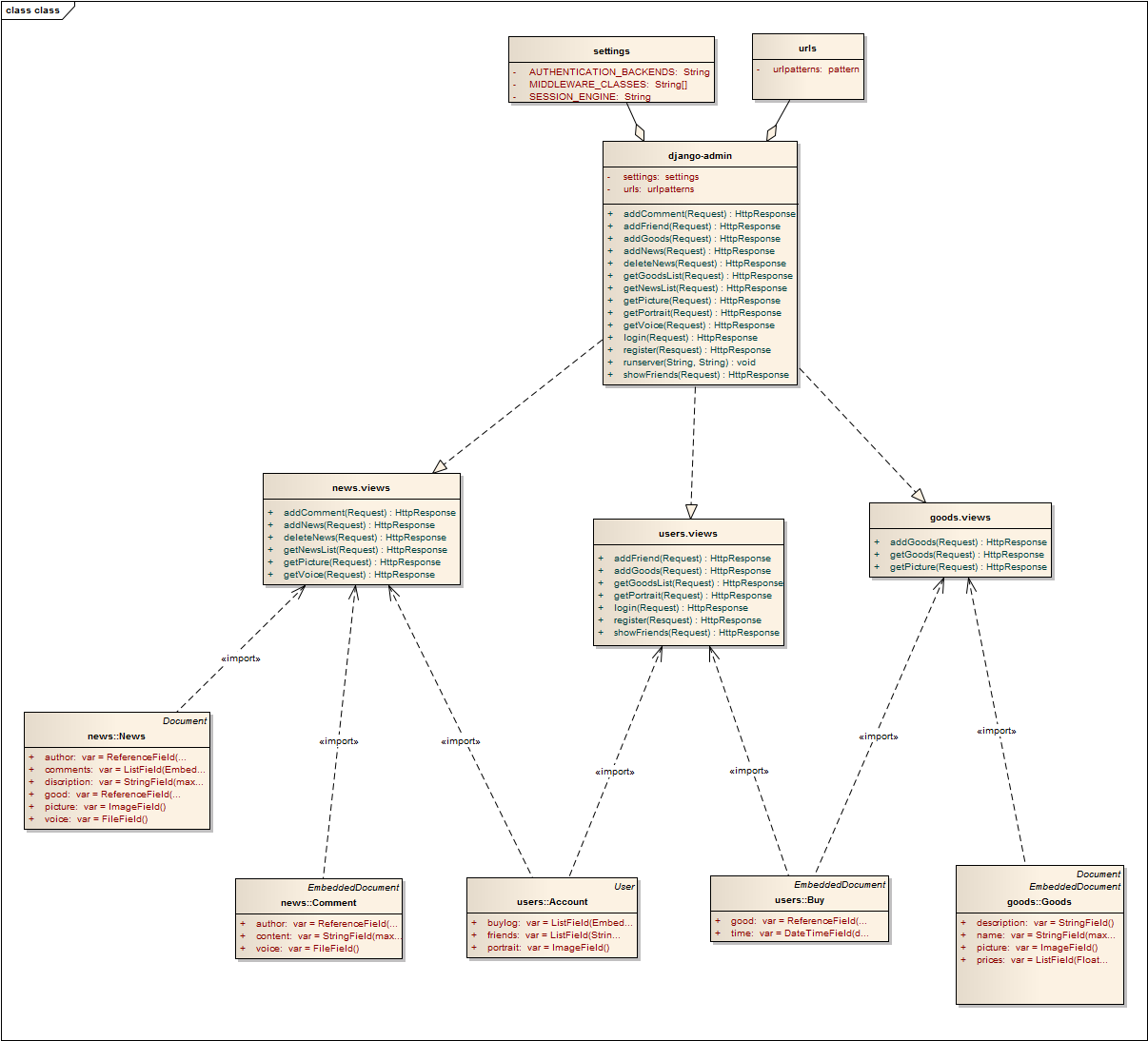


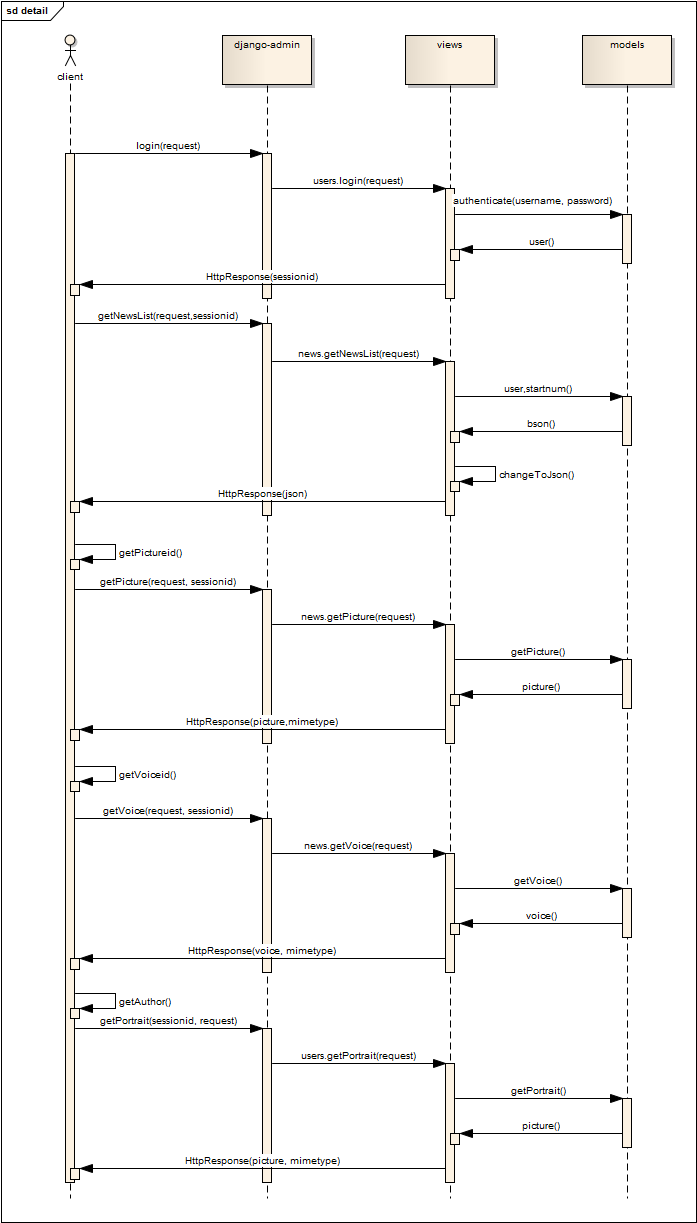
图4.4：服务器详细设计类图

如上图所示，服务器大概分为三个部分：

1. django-admin部分，这个部分主要由django框架自动生成。为了完成服务器与数据库的连接，需要对settings.py中的设置进行相应修改。为了保证前端每个请求都能有准确地处理，需要在urls.py中设置urlpatterns。
2. views部分，这个部分由自己设计完成。主要是各个模块的方法，django框架找到合适的处理请求的方法后，调用views中的具体方法，并返回HttpResponse。
3. models部分，这个部分由自己设计完成。主要是各个模块的数据结构的设计，与后台数据库中的collection相对应。Views方法中会通过import的方式调用需要的models，通过models类的方法完成对数据库的存取和修改。

### 4.2.4 服务器模块复杂任务详细设计

登录以及获取新鲜事列表的详细设计：用户发送登录请求，django-admin模块处理请求，调用users包中的login方法，login方法调用user类中的验证方法检验是否成功登录。登录成功后，用户发出getNewsList请求，django-admin模块处理请求，调用news模块中getNewsList的方法返回新鲜事列表的json文件。客户端根据收到的json，提取出每条新鲜事的图片获取请求和声音获取请求。发送图片请求，django-admin模块处理请求，调用news模块中getPicture方法，返回图片。用户点击播放声音后，发送播放声音的请求，django-admin模块处理请求，调用news模块中getVoice请求，返回声音。具体设计参见图4.5。

图4.5：登录以及获取新鲜事列表的顺序图

### 4.2.5 数据库模块的详细设计

数据库使用MongoDB存储信息，以BSON为主要的存储格式，对于文件和图片采用GridFS进行存储。数据库的所有collections如图4.6所示。

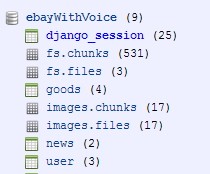


图4.6：数据库collections设计

django\_session表的设计如表4.1所示。

表4.1：django\_session表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | session的唯一标示符 |
| expire\_date | 记录过期的时间 |
| session\_data | session的内容 |

goods表的设计如表4.2所示。

表4.2：goods表的设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | goods的唯一标示符 |
| picture | goods的图片id |
| prices | goods的价格变化 |

news表的设计如下表4.3所示。

表4.3：news表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | news的唯一标示符 |
| author | 一个引用类型和objectid指向user |
| comments | 该新鲜事的所有评论。存储的格式是comment类型，comment包括content字段存储评论，包括voice字段存储评论声音的ObjectId，包括author字段存储引用类型指向一个用户。 |
| picture | 新鲜事图片的ObjectId |
| voice | 新鲜事声音的ObjectId |

users的设计如下图所示。

表4.4：user表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | user的唯一标示符 |
| buylog | 用户的购买记录，主要包括购买的商品和购买时间 |
| date\_joined | 用户生成的时间 |
| friends | 用户的好友列表 |
| portrait | 用户的头像ObjectId |
| username | 用户的用户名 |
| password | 用户的密码 |

image.chunks和image.files两张表格用于存储图片文件，具体设计如表4.5和表4.6所示。

表4.5：image.chunks表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | chunks的唯一标示符 |
| n | 标明这个chunk是file的第几个文件碎片 |
| files\_id | 指向image.files的引用 |
| data | 二进制流文件 |

表4.6：image.files表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | files的唯一标示符 |
| chunkSize | 这个file对应的chunk的大小 |
| format | 存储图片的格式 |
| thumbnail\_id | 指向对应的thunbnail |
| height | 图片高度 |
| width | 图片宽度 |
| length | 二进制流字节长度 |
| uploadDate | 标明上传日期 |
| md5 | md5值 |

fs.file和fs.chunks用于存储音频文件，与image.files和image.chunks两张表结构类似，如下图所示。

表4.7：file.chunks表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | chunks的唯一标示符 |
| n | 标明这个chunk是file的第几个文件碎片 |
| files\_id | 指向image.files的引用 |
| data | 二进制流文件 |

表4.8：image.files表设计

|  |  |
| --- | --- |
| 字段 | 描述 |
| id | files的唯一标示符 |
| chunkSize | 这个file对应的chunk的大小 |
| length | 二进制流字节长度 |
| uploadDate | 标明上传日期 |
| md5 | md5值 |

### 4.2.6 数据库模块复杂任务详细设计

用户发送新鲜事后，后台存储图片和音频的过程：用户发送购买记录的请求，django-admin处理请求，调用users中getGoodsList方法返回购买记录。客户端获取需要的商品链接，获取用户需要发送的照片和音频，发出请求。django-admin处理请求，调用news中addNews方法，保存新鲜事到数据库。具体的设计顺序图参见图4.7。

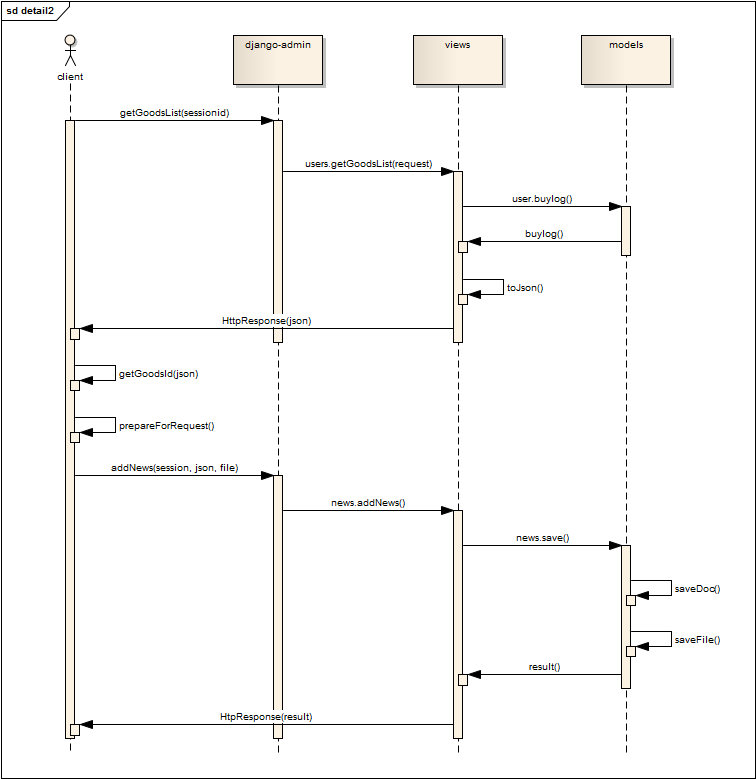


图4.7：发送新鲜事详细设计顺序图

## 4.3本章小结

本章介绍了eBay Friends系统服务端概要设计和详细设计。给出了整个系统的整体框架，对服务器和数据库连接模块进行了详细的描述。介绍了eBay Friends系统服务端模块的类图和复杂用例的顺序图。

# 第五章 eBay Friends项目服务器模块的实现

## 5.1 服务端模块的实现

### 5.1.1 服务器模块配置部分的实现

服务端使用的是Django的模块。在安装[[14]](#footnote-2)好django框架的开源模块代码基础上，需要对urls.py和setting.py两个文件进行修改。

setting.py文件中确定了整个服务端的配置，包括数据库连接、文件上传处理、登录验证、会话处理和url指向等，具体代码如图5.1所示。

|  |
| --- |
| from mongoengine import connect  #import mongengine  connect("eBayWithVoice")  #connect to db  FILE\_UPLOAD\_HANDLERS=("django.core.files.uploadhandler.TemporaryFileUploadHandler",)  #add handlers to handle the uploaded files  AUTHENTICATION\_BACKENDS = ('mongoengine.django.auth.MongoEngineBackend',)  #add authentication for the users to login  SESSION\_ENGINE = 'mongoengine.django.sessions'  #add session engine for the session of users  ROOT\_URLCONF = 'eBayWithVoice.urls'  #decide the urls.py for the project |

图5.1：settings.py代码

urls.py文件主要负责将各个模块的处理函数与各个请求对应，找到解决的方法，urls.py的具体实现如图5.2所示。

|  |
| --- |
| from django.conf.urls import patterns, url  urlpatterns = patterns('',  url(r'^users/login$','users.views.login'),  url(r'^users/logout$','users.views.logout'),  url(r'^users/register$','users.views.register'),  url(r'^users/newPage$','users.views.newPage'),  url(r'^users/addFriend$','users.views.addFriend'),  url(r'^users/showFriends$','users.views.showFriends'),  url(r'^users/getPortrait$','users.views.getPortrait'),  url(r'^users/addGoods$','users.views.addGoods'),  url(r'^users/getGoodsList$','users.views.getGoodsList'),  #urls used in users model  url(r'^news/addNews$','news.views.addNews'),  url(r'^news/getNewsList$','news.views.getNewsList'),  url(r'^news/getPicture$','news.views.getPicture'),  url(r'^news/getVoice$','news.views.getVoice'),  url(r'^news/addComment$','news.views.addComment'),  url(r'^news/deleteNews$','news.views.deleteNews'),  #urls used in news model  url(r'^goods/addGoods$','goods.views.addGoods'),  url(r'^goods/getGoods$','goods.views.getGoods'),  url(r'^goods/getPicture$','goods.views.getPicture'),  #urls used in goods model  ) |

图5.2：urls.py代码

### 5.1.2服务器模块功能需求的实现

除了Django自带的框架文件，各个功能点的实现代码主要有views实现。下面根据各个功能描述，给出实现代码。

1. user模块中views代码参见图5.3。该模块主要包含三个函数，分别是登录、获取用户头像和获取购买列表。

|  |
| --- |
| def login(request):  data=request.raw\_post\_data  jsonObject=json.loads(data)  userName=jsonObject['username']  passWord=jsonObject['password']  #从json中获取数据username和password  user=auth.authenticate(username=userName, password=passWord)  #使用django框架中的user类自带的验证函数，对用户名和密码进行验证  if user is not None:  if user.is\_active:  auth.login(request, user)  #使用django中的user类中的login，登录成功  return HttpResponse("success")  ……  def getPortrait(request):  nid=request.GET['id']  user=Account.objects(pk=nid).first()  #根据请求中的id找到用户  return HttpResponse(user.portrait.read()  ,mimetype="image/jpeg")  #返回用户头像的图片  @login\_required  def getGoodsList(request):  buylogs=request.user.to\_mongo()['buylog']  #根据请求的session，从数据库中找到对应用户的buylogs  for buylog in buylogs:  buylog["time"]=str(buylog["time"])  name=Goods.objects(pk=buylog["good"].id).first().name  #根据id从商品collection中获取商品名称  buylog['name']=name  buylog['id']="http://192.168.47.19:8080/goods/getGoods?id="+str(buylog["good"].  id)  #使用for循环对数据进行处理，返回一个json  return HttpResponse(ju.dumps(buylogs)) |

图5.3：user模块中views.py代码

1. news模块中views.py代码参见图5.4。news模块的views.py代码中包含四个函数，主要用于完成发送新鲜事和获取新鲜事列表的功能。

|  |
| --- |
| @login\_required  def addNews(request):  url=request.POST['url']  news = News()  ……  news.picture = open(request.FILES['picture'].temporary\_file\_path(), "rb")  news.voice = open(request.FILES['voice'].temporary\_file\_path(), "rb")  news.save()  #获取请求中的参数，并生成一个news保存至数据库  return HttpResponse("success")  @login\_required  def getNewsList(request):  num = request.GET['num']#获取请求参数  allFriends = Account.objects(pk\_\_in=request.user.friends).all()#获取好友列表  result = News.objects(author\_\_in=allFriends)[int(num\*15):int(num)\*15 + 15].order\_by("-time").as\_pymongo()  #获取新鲜事  result = list(result)  for news in result:  ……#处理新鲜事信息  result = dumps(result)#转化成json格式  return HttpResponse(result)  def getPicture(request):  ……#根据请求获取新鲜事图片  return HttpResponse(image, mimetype="image/jpeg")  def getVoice(request):  ……#根据请求获取新鲜事图片  return HttpResponse(voice, mimetype="audio/mpeg") |

图5.4：news模块views.py实现

## 5.2 数据库模块实现

数据库模块的实现主要是基于mongoengine中提供的各个字段，将数据库中的collection在django中以类的形式重新展现出来。

1、news模块中的models.py的实现如图5.5所示。

|  |
| --- |
| class Comment(EmbeddedDocument):  author = ReferenceField(Account)  content = StringField(max\_length=140)  voice = FileField()  #comment类，用于存储news的评论，将以列表的形式出现在news表中comments字段中  class News(Document):  author = ReferenceField(Account, reverse\_delete\_rule=CASCADE)  discription = StringField(max\_length=140)  picture = ImageField()  voice = FileField()  comments= ListField(EmbeddedDocumentField(Comment))  good= ReferenceField(Goods)  time=DateTimeField(default=datetime.datetime.now)  #news字段，用于存储新鲜事 |

图5.5：news模块models.py的实现

1. users模块中models.py的实现如图5.6所示。

|  |
| --- |
| class Buy(EmbeddedDocument):  good=ReferenceField(Goods)  time=DateTimeField(default=datetime.datetime.now)  #buy用于记录用户购买的商品，将以列表的形式出现在Account表的buylog中  class Account(User):  friends=ListField(StringField())  portrait=ImageField()  buylog=ListField(EmbeddedDocumentField(Buy))  #Account继承自mongoengine中的User类，类中自带有username和password字段。 |

图5.6：users模块models.py的实现

## 5.3 服务器运行截图

服务器运行步骤：

1. 进入服务器代码存放的文件夹。例如，E:/study\workspace\python\ebayWithVoice
2. 运行命令：python manage.py runserver 0.0.0.0:8080。此时，服务器成功启动。
3. 服务端服务器运行时截图如图5.7所示。

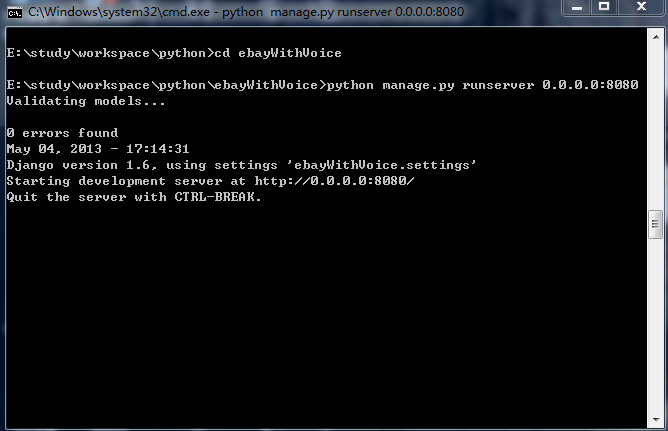


图5.7：服务器运行时截图

## 5.4 数据库运行截图

数据库运行步骤：

1. 进入数据库文件夹。例如，E:\database\mongo
2. 运行命令：mongod --dbpath E:\database\mongo\ebayWithVoice
3. 服务端数据库运行时截图如图5.8所示。

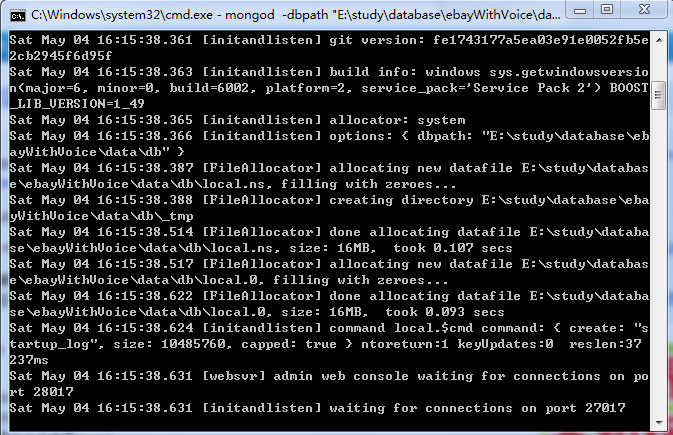


图5.8：数据库运行时截图

## 5.5 本章小结

本章介绍了eBay Friends系统后台的实现代码。对服务器和数据库的具体的实现函数和方法进行了详细的描述。此外，还给出了服务器和数据库的运行方法，以及运行时的截图。

# 第六章 总结与展望

## 6.1 总结

为了更有效的提高社会关系在购物平台中的巨大推动作用，尤其是利用好友关系实现商品资源的分享和评价，本文从需求分析、系统设计和系统实现角度描述了ebay Friends系统中的服务端模块的开发过程，为前台实现提供了所有需要的接口。本文的主要研究成果如下：

* 论述了eBay Friends系统对于购物平台的巨大作用，以及相比于其他现有的社交购物系统的优势。论述了服务端模块在整个系统中的重要地位。突出展现了eBay Friends服务端模块的重要意义。
* 描述了eBay Friends系统的整体需求，并对服务端模块的需求进行了详细的分析和整理。
* 阐述了eBay Friends系统的整体架构和接口连接。对服务端模块的设计和框架进行了详细的描述，使得整个模块划分明确，数据与方法分割独立，有效地降低了系统的耦合度。
* 给出了eBay Friends系统的服务端实现，具体阐述了关键功能的实现步骤和算法。

## 6.2 展望

在后续的工作中，项目组会对现有的功能性需求和非功能性需求做出修改。

在功能性需求方面，系统将会有下列改善：

* 更加完善的实名制系统。具体的实现方式是，将eBay Friends账号和FaceBook等多个国内外已经实现实名制的社交网站的账号进行绑定，同时导入原有社交网络中已经存在的好友关系，使得系统的实名制更加完善，保证最终的使用效果。
* 商品信息的导入。具体的实现方式是，实现注册功能，提供接口使用户可以自由注册。之后，将eBay Friends账号和eBay的购买账号进行绑定，用户登录后可以自由的增加、删除、修改或查看自己在eBay网站上留下的个人信息，包括自己的购买记录。
* 实现在线购买。具体的实现方式是，将eBay Friends系统中的购买功能完善，使其能够自由的连接到eBay网站的账号，并使用eBay的账号进行购买和支付。

在非功能需求方面，系统将会有下列改善：

* 使用eBay现有的云平台系统搭建服务主机。调用eBay的云平台，搭建一个支持django框架的服务器，以代替目前使用的开发版服务器。使用eBay云平台中的LoadBalance系统实现服务器的负载均衡，保证服务器的响应速度和处理速度。
* 使用分布式数据库系统代替目前使用的单机版MongoDB。考虑到系统投入使用后会有大量的数据进行读写操作，现有的数据库将不能保证读写效率。因此使用分布式数据库实现数据的实时更新。

# 参考文献

1. [] 关系型数据库-维基百科，自由的百科全书，http://zh.wikipedia.org/wiki/%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93 [↑](#endnote-ref-2)
2. [] 非关系型数据库-维基百科，http://zh.wikipedia.org/wiki/%E9%9D%9E%E5%85%B3%E7%B3%BB%E5%9E%8B%E6%95%B0%E6%8D%AE%E5%BA%93 [↑](#endnote-ref-3)
3. [] HBase-Apache HBase Home，http://hbase.apache.org/ [↑](#endnote-ref-4)
4. [] Neo4j，http://www.neo4j.org/ [↑](#endnote-ref-5)
5. [] MongoDB，http://www.mongodb.org/ [↑](#endnote-ref-6)
6. [] BSON-维基百科，http://en.wikipedia.org/wiki/BSON [↑](#endnote-ref-7)
7. [] Python Programming Language，http://www.python.org/ [↑](#endnote-ref-8)
8. [] The Web framework for perfectionists with deadlines | Django，https://www.djangoproject.com/ [↑](#endnote-ref-9)
9. []蔡金花，浅析NOSQL及使用，《电脑知识与技术》,2011年07期：第一页 [↑](#endnote-ref-10)
10. []NOSQL-维基百科，自由的百科全书，http://zh.wikipedia.org/wiki/Nosql [↑](#endnote-ref-11)
11. []卢冬海，何先波，浅析NOSQL数据库，《中国西部科技》，2011，10（2）：P16 [↑](#endnote-ref-12)
12. [] Introduction to MongoDB，http://www.mongodb.org/about/introduction/ [↑](#endnote-ref-13)
13. [] JSON-维基百科，http://zh.wikipedia.org/wiki/JSON [↑](#endnote-ref-14)
14. 关于安装django的更多信息，请参见：https://docs.djangoproject.com/en/1.4/intro/install/ [↑](#footnote-ref-2)