****

****

**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** 助理研究员

**论文提交日期**

# 摘 要

随着电子购物的普及，企业对商品的评价系统在企业产品的推广中的作用愈发重视，顾客在购买商品之前会仔细推敲商品的评价。与此同时，更多的电子购物网站开始研究如何使用完善评价系统为企业带来更多利润。

eBay Friends项目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，将自己对于该商品的使用心得传递给好友，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。对比其他评价系统，eBay Friends系统具有实名制、评价形式多样和信息来源广泛等优势。本文主要对eBay Friends项目服务端开发过程中的各个阶段进行详细的描述和介绍。

服务端模块用于处理客户端发送的请求，对请求进行处理和分析，完成数据库操作并返回相应。首先，根据eBay Friends项目的整体需求，整理出与服务端相关的需求用例，并对其进行分析。然后，设计服务端整体架构，使用的主要技术是Python语言Django框架，并使用MongoDB作为数据库。最后，根据需求，搭建完成Django服务器并连接MongoDB数据库，分别实现服务端模块各个部分的代码，并完成最终的调试。

本文就整个eBay Friends项目进行了全面的需求分析，设计和实现，详细的描述了项目的背景和最终结果，并从技术角度表现了项目的整个开发过程，为下一步的改进打下了基础。

**关键词**：eBay Friends项目，服务端，Python，MongoDB，Django

# Abstract

# 目 录

[摘 要 II](#_Toc357460599)

[Abstract III](#_Toc357460600)

[目 录 IV](#_Toc357460601)

[图目录 VI](#_Toc357460602)

[表目录 VII](#_Toc357460603)

[第一章 引言 1](#_Toc357460604)

[1.1 项目背景 1](#_Toc357460605)

[1.2 国内外商品评价系统的现状 2](#_Toc357460606)

[1.2.1亚马逊网站商品评价系统现状 2](#_Toc357460607)

[1.2.2天猫网站商品评价系统 3](#_Toc357460608)

[1.2.3 Fancy应用的商品评价系统 4](#_Toc357460609)

[1.3 论文的主要工作和组织结构 5](#_Toc357460610)

[第二章 相关技术综述 7](#_Toc357460611)

[2.1 NoSQL数据库技术综述 7](#_Toc357460612)

[2.2 MongoDB技术详述 8](#_Toc357460613)

[2.3 Python技术综述 9](#_Toc357460614)

[2.4 Django技术综述 10](#_Toc357460615)

[2.5本章小结 12](#_Toc357460616)

[第三章eBay Friends项目服务端的需求分析 13](#_Toc357460617)

[3.1 eBay Friends项目整体概述 13](#_Toc357460618)

[3.2 eBay Friends项目服务端需求分析 13](#_Toc357460619)

[3.2.1 eBay Friends项目需求概述 13](#_Toc357460620)

[3.2.2 eBay Friends项目服务端需求用例图 14](#_Toc357460621)

[3.2.3 eBay Friends项目服务端用例分析 15](#_Toc357460622)

[3.2.4 eBay Friends项目服务端ERD 25](#_Toc357460623)

[3.3 本章小结 26](#_Toc357460624)

[第四章 eBay Friends项目服务端模块的设计 27](#_Toc357460625)

[4.1 eBay Friends项目的概要设计 27](#_Toc357460626)

[4.2 eBay Friends项目服务端的详细设计 29](#_Toc357460627)

[4.2.1 eBay Friends项目服务器模块概述 29](#_Toc357460628)

[4.2.2 eBay Friends项目服务器模块的详细设计 29](#_Toc357460629)

[4.2.3 eBay Friends项目服务端模块复杂任务详细设计 30](#_Toc357460630)

[4.3 eBay Friends项目数据库设计 32](#_Toc357460631)

[4.3.1 eBay Friends项目数据库概述 32](#_Toc357460632)

[4.3.2 eBay Friends项目数据库的详细设计 32](#_Toc357460633)

[4.4 eBay Friends项目接口描述 35](#_Toc357460634)

[4.5 本章小结 36](#_Toc357460635)

[第五章 eBay Friends项目服务端模块的实现 37](#_Toc357460636)

[5.1 eBay Friends项目服务端模块的实现环境 37](#_Toc357460637)

[5.2 eBay Friends项目服务端模块的实现代码 37](#_Toc357460638)

[5.2.1 eBay Friends项目服务器模块配置部分的实现 37](#_Toc357460639)

[5.2.2 eBay Friends项目服务端模块视图的实现 39](#_Toc357460640)

[5.2.3 eBay Friends项目服务端模块数据库部分实现 41](#_Toc357460641)

[5.3 eBay Friends项目服务端运行截图 42](#_Toc357460642)

[5.4本章小结 44](#_Toc357460643)

[第六章 总结与展望 45](#_Toc357460644)

[6.1总结 45](#_Toc357460645)

[6.2展望 45](#_Toc357460646)

[参考文献 47](#_Toc357460647)

[致谢 48](#_Toc357460648)

# 图目录

[图1.1：亚马逊网站的评价系统 3](#_Toc357460899)

[图1.2：天猫网的评价系统 4](#_Toc357460900)

[图1.3：Fancy应用截图 5](#_Toc357460901)

[图2.1：Django框架整体架构 11](#_Toc357460902)

[图2.2：Django框架处理request流程图 12](#_Toc357460903)

[图3.1：eBay Friends项目用例图 15](#_Toc357460904)

[图3.2：eBay Friends项目ERD 25](#_Toc357460905)

[图4.1：eBay Friends项目整体概要设计图 27](#_Toc357460906)

[图4.2：服务端概要设计图 28](#_Toc357460907)

[图4.3：服务端详细设计类图 30](#_Toc357460908)

[图4.4：登录以及获取新鲜事列表的顺序图 31](#_Toc357460909)

[图4.5：数据库设计图 33](#_Toc357460910)

[图5.1：settings.py代码 38](#_Toc357460911)

[图5.2：urls.py代码 39](#_Toc357460912)

[图5.3：users模块views.py的实现 40](#_Toc357460913)

[图5.4：news模块views.py的实现 41](#_Toc357460914)

[图5.5：users模块models.py的实现 41](#_Toc357460915)

[图5.6：news模块models.py的实现 42](#_Toc357460916)

[图5.7：数据库运行时截图 43](#_Toc357460917)

[图5.8：服务器运行时截图 43](#_Toc357460918)

# 表目录

[表3.1：eBay Friends项目需求用例 13](#_Toc357462742)

[表3.2：登录用例详细描述 16](#_Toc357462743)

[表3.3：浏览新鲜事用例详细描述 16](#_Toc357462744)

[表3.4：听取新鲜事声音用例详细描述 17](#_Toc357462745)

[表3.5：查看新鲜事评论用例详细描述 17](#_Toc357462746)

[表3.6：播放评论声音用例详细描述 18](#_Toc357462747)

[表3.7：评论新鲜事声音用例详细描述 19](#_Toc357462748)

[表3.8：获取购物历史用例详细描述 19](#_Toc357462749)

[表3.9：发送新鲜事用例详细描述 20](#_Toc357462750)

[表3.10：添加删除好友用例的详细描述 21](#_Toc357462751)

[表3.11：获取商品信息的详细描述 21](#_Toc357462752)

[表3.12：获取购买商品用例的详细描述 22](#_Toc357462753)

[表3.13：获取好友列表用例的详细描述 22](#_Toc357462754)

[表3.14：搜索用户用例的详细描述 23](#_Toc357462755)

[表3.15：获取个人新鲜事用例的详细描述 24](#_Toc357462756)

[表3.16：获取新鲜事详细信息的详细描述 24](#_Toc357462757)

[表4.1：django\_session表设计 33](#_Toc357462758)

[表4.2：goods表的设计 33](#_Toc357462759)

[表4.3：news表设计 34](#_Toc357462760)

[表4.4：user表设计 34](#_Toc357462761)

[表4.5：image.chunks表设计 34](#_Toc357462762)

[表4.6：image.files表设计 34](#_Toc357462763)

[表4.7：file.chunks表设计 35](#_Toc357462764)

[表4.8：image.files表设计 35](#_Toc357462765)

[表5.1：eBay Friends项目的实现环境 37](#_Toc357462766)

# 第一章 引言

## 1.1 项目背景

随着互联网技术的飞速发展，很多现实生活中的行为习惯在虚拟社会中得以实现。在众多技术中，电子购物技术凭借其与现实生活的密切联系脱颖而出，得到了越来越多的重视。人们平时只能在商场进行的购物活动，现在可以轻而易举的通过鼠标和键盘的几下操作实现。

电子购物能够迅速普及，除了因为方便快捷之外，还因为互联网可以提供更大量更快捷的商品信息，帮助顾客判断商品的性价比。然而，虽然网上已有很多信息描述商品，但是，相比而言，顾客还是更加愿意查看其他已经购买过该商品的顾客对于商品的评价。出于这样的目的，越来越多的电商平台开始将客户的评价作为商品的重要信息布置在页面上，并以此作为顾客和商家之间交流活动。也正是由于这样的评价系统的出现，越来越多的顾客在决定购买商品之前，会首先查看该商品的销售记录和评价留言。同时，在收到购买的商品后，顾客也会更加愿意分享自己对于该商品的使用心得和购物体验。这样的评价有效地帮助顾客做出正确的购物选择，也帮助信誉好服务好的商家赢得了更多的商机。

但是，这样的评价系统也具有诸多弊端。

首先，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，存在大量虚假数据。目前的诸多购物网站中，都会有很多恶意刷好评的情况出现。这样的虚假数据使得评价对于顾客购物的决策的效果大打折扣，也严重影响了其他商家的正常营销。为了保证信息的真实性，实名注册成为很多网站使用的手段，尤其是很多以社交为主要目的的网站，例如国内的人人网和国外的Facebook。评价系统也可以采用这样的设计，将用户的账号与其个人的某个社交网站的账户绑定在一起，这样可以使得用户看到的信息都有准确地来源。

其次，目前的购物网站大多数仅仅提供了文字的描述，不够直观，不具有足够的说服力。众多的研究和实践表明，图片和音频等多媒体能够更加直观更加准确地向顾客展现商品的使用效果。因此，图片或者音频作为商品的评价比文字更能打动客户。为了弥补文字描述表达力不足的缺点，评价的系统可以采用图片作为载体，辅之以音频，这样其他的用户可以更加直观的感受到商品的使用感觉和真实评价。

此外，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。这样就会使得商品的评价来源大量减少，对于商家的损失会很大。相比于文字输入的困难，手机和其他移动端的智能设备却可以方便快捷的利用手机的拍照和录音功能获取图片和音频，以此提高用户使用的方便程度。

对于网上购物的用户而言，商品的评价系统应该能够提供给用户该商品一个真实可靠的商品评价，用户不仅仅可以查看商品的官方信息，而且可以从评价系统中获取自己社交圈子中其他好友对该商品的评价；用户能够从评价系统中获得直观而准确地评价，而不仅仅是简单的文字描述；除了准确地商品评价信息，用户还应该可以直接从好友的分享中准确找到商品的购买链接，以此方便购买，降低了搜索需要的时间。

对于商家而言，评价系统应该能够客观的获取所有购买过该商品的用户的反馈，而不是随意的接受任意的评价，这样可以有效提高商家的信誉；商家可以直接从该评价系统中看出自己的产品的受欢迎程度。

综上所述，一个能够有效帮助用户的商品评价系统是很有必要的。它帮助用户从社交圈子中的其他好友处获取最新的商品评价，同时也可以帮助用户分享自己的购物心得与乐趣，还能够帮助商家树立良好的信誉，有效促进网上交易量的提高。

## 1.2 国内外商品评价系统的现状

### 1.2.1亚马逊网站商品评价系统现状

亚马逊公司是美国最大的一家网络电子商务公司，位于华盛顿州的西雅图。是网络上最早开始经营电子商务的公司之一

亚马逊电子购物网站的评价系统如图1.1所示。如图所示，亚马逊网站的评价系统的主要优势有：对所有评价进行分类打分，将不同等级的打分单独列出，如图中左上角所示；提供了良好的搜索功能，帮助用户搜索自己需要的评价如图中右上角所示；将最好的评价和最差的评价进行重点突出，方便用户选择商品，而后将其他用户的评论以分页的形式列举出来。在这样的系统中，用户可以获取到该商品的用户反馈，但是也有诸多弊端：所有商品的评价超过2000条，用户无法准确判断哪些评价值得相信；此外，评价页面中所有的图片均来自商家，没有任何图片或者其他多媒体资料来自于用户的上传，这样的评价形式，不利于用户对该商品形成一个直观的印象。

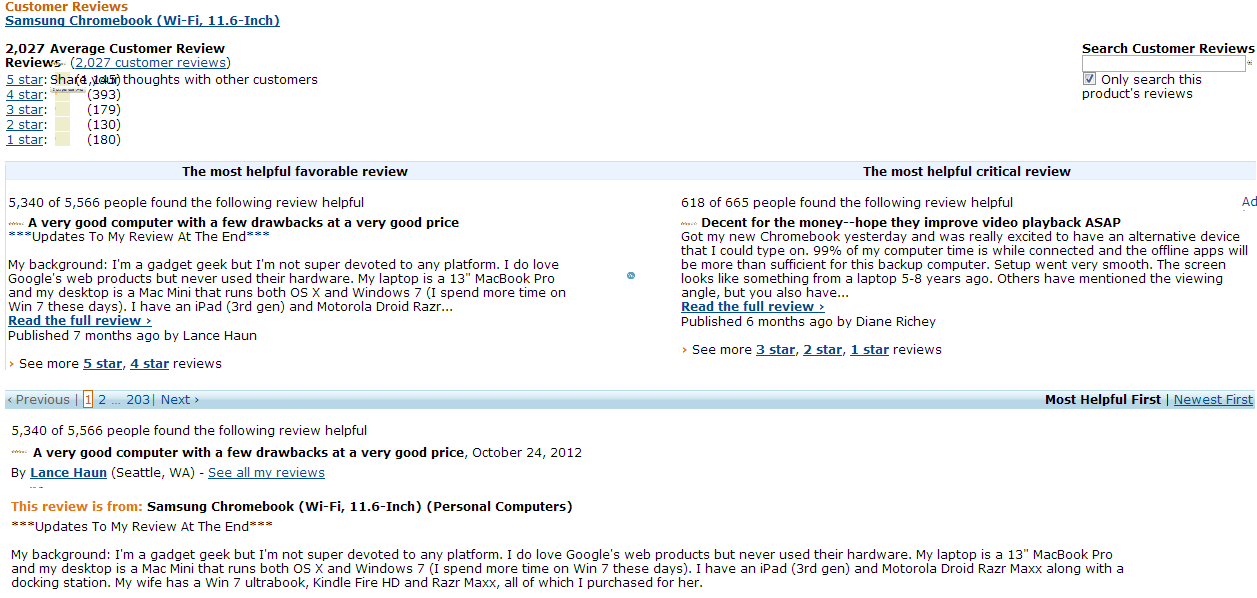


图1.1：亚马逊网站的评价系统

### 1.2.2天猫网站商品评价系统

天猫网站原名“淘宝商城”，是一个综合性购物网站，是有阿里巴巴公司全新打造的B2C（Business-to-Consumer，商业零售）商城。

天猫网的评价系统如图1.2所示。如图所示，天猫网的评价系统的主要优势有：天猫网根据用户对该商品的打分，将商品的总体评价以得分的形式体现出来，同时提取出评价中的关键字，有效地节省了用户判断该商品质量话费的时间；天猫网提供了追加评价的功能，可以获取用户使用过一段时间后对于该商品的评价，更加具有真实性。在这样的评价系统中，用户可以清晰地看到该商品的评价，但是，也会有很多的弊端：尽管天猫网提供了评价排序的功能，但是用户仍然无法准确获取自己希望看到的评论；天猫网将账户与评价的内容绑定，一定程度上保证了评价来源的真实性，但是由于账号不具备实名制，因此这样绑定的效果十分有限。



图1.2：天猫网的评价系统

### 1.2.3 Fancy应用的商品评价系统

Fancy是一款基于手机等移动设备的一款购物分享平台，它的主要功能是帮助购买过商品的用户在该平台上分享自己拍摄的商品图片，并给出自己的配音。

Fancy应用的具体截图如图1.3所示。左图中给出了一系列的图片，每个图片展示了一种商品；点击进去之后进入中图所示的页面，详细描述了这个商品的情况；点击评论的情况，可以看到对于这个分享，其他用户的一些看法。Fancy应用的优势在于：采用了图片等多媒体手段作为分享的载体，避免了单调的文字描述；Fancy是一款手机的应用，用户的使用十分方便。但是，Fancy也有不可忽视的缺点：Fancy应用并未实行实名制，分享的内容真实性不够，难以成为用户决策的因素之一；而且，Fancy并没有给出详细的商品的来源和具体的购买链接，所以，尽管用户可能对该分享十分感兴趣，但是并不能立即找到购买的方式，这样使得用户购买的积极性大大下降。

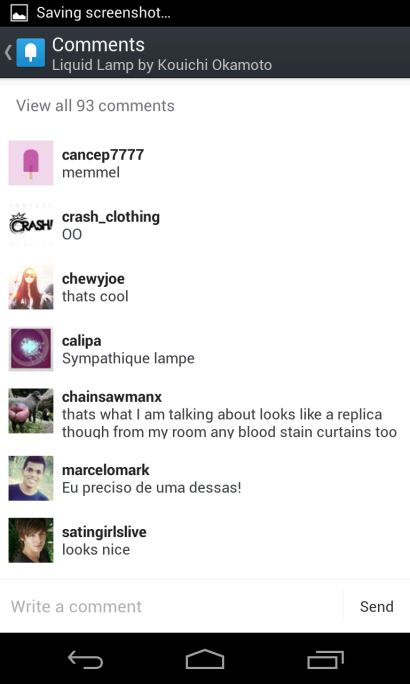
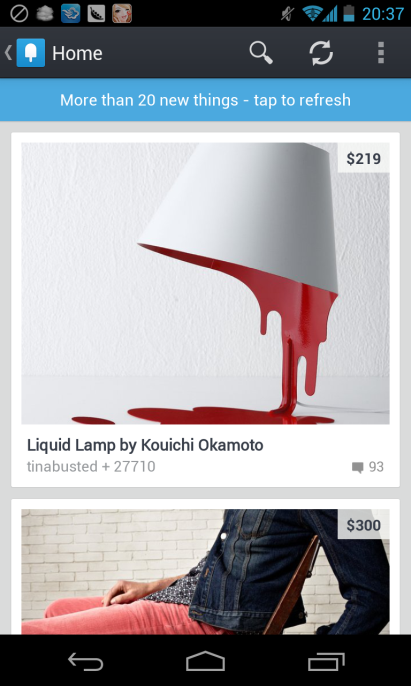


图1.3：Fancy应用截图

## 1.3 论文的主要工作和组织结构

根据上述背景情况，eBay公司为了将社交元素融入自己的购物平台，打算开发一款全新的购物分享应用——eBay Friends。

eBay Friends是一款基于Android系统的应用软件。用户可以免费注册并使用，每当用户在eBay购物平台购买了某项商品后，可以使用该平台分享自己与商品的合影，并为照片配上一段自己的录音。生成的分享可以在社交圈子中被其他好友看见，并被评论。同时，eBay Friends可以获得eBay网站的相关信息，包括商品的官方描述和用户在eBay的账号信息等，这样，用户可以直接使用eBay Friends进行商品的选购和下单，极大的方便了用户购买。

eBay Friends的特点在于实名制，用户可以将eBay Friends账号与Facebook等社交网络的账号绑定，从而可以在自己的社交圈子中分享购物心得，同时，用户也可以从好友那里获得更准确更真实可信的的商品评价，购买到更多物美价廉值得相信的商品。

此外，eBay Friends采用多媒体的方式展示商品的评价，而不是采用单一的文字作为载体。因此，用户可以更加直观更加方便的查看商品的详细情况，从而做出更加合理的购物决策。由于eBay Friends绑定了用户在eBay的账号，因此，用户可以直接下单购买，省去了很多的时间。

本文主要介绍eBay Friends项目在服务端的设计和实现。从需求分析，到软件设计，然后到最终的实现三个方面详细描述了eBay Friends项目服务端的开发过程。

本文首先介绍项目需求，对整个项目的需求进行了一个概述。对与服务端模块相关的用例进行了详细的分析。然后，本文描述了系统整体设计的架构，对服务端模块进行了详细的分析。最后，本文详细描述了系统的所有模块的实现，给出了各个模块的核心代码和主要算法，并给出了系统运行过程中的截图。

论文的整体结构分为六章，每章内容如下：

第一章：概述和前言部分，主要介绍了eBay Friends项目的背景，研究当前国内外电子购物网站的商品评价系统的优缺点。

第二章：主要介绍eBay Friends项目服务端使用到的具体技术，介绍服务端模块实现的理论基础。

第三章：从项需求分析方面，描述eBay Friends项目开发的功能性需求和非功能性需求，并针对服务端模块的需求进行详细分析。

第四章：从项目设计方面，描述eBay Friends项目的整体设计并针对服务端模块的详细设计进行分析。

第五章：从项目的实现方面，描述整个项目的实现代码，主要包括服务器端的代码和数据库存取代码。

第六章：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 NoSQL数据库技术综述

传统的关系型数据库[[1]](#endnote-1)读写操作都是事务的，事物是构成单一逻辑工作单元的操作集合，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，由于现在因特网迅速普及，人们与网络的信息交流量迅速增大，一个网站的数据库处理能力往往决定了这个网站的成败。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[[2]](#endnote-2)：不擅长处理大量数据的写入，不擅长为有数据更新的表做索引或表结构变更，不擅长处理字段不固定时应用，不擅长处理需要快速返回结果的简单查询。正是因为这样的缺点，很多网站放弃关系型数据库，转而使用非关系型数据库，也就是NoSQL数据库。

NoSQL有时也称作NotOnlySQL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。NoSQL[[3]](#endnote-3) 易于处理数据的分散，NoSQL数据库不支持JOIN操作，各个数据都是独立设计的，很容易把数据分散到多个服务器上；NoSQL数据可易于提升性能和增大规模；NoSQL数据库使大量数据的写入处理更加容易。

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。NoSQL数据库大致分为[[4]](#endnote-4)，键值存储数据库，面向文档的数据库和面向列的数据库。

NoSQL数据库在具有巨大优势的同时，也有很多缺点。首先，NoSQL数据库的成熟度十分欠缺。关系型数据库出现在软件工程界已经30年了，在数据存储方面是公认的优秀工具，相比而言，NoSQL的实际应用刚刚起步，对企业来说，稳定压倒一切，更多的软件还是更愿意选择关系型数据库。然后，关系型数据库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，在商业化程度上明显落后于关系型数据库。最后，大多数软件开发人员已经习惯了RDBMS，大多数开发人员的思想也已经被固定在关系模型上，因此使用NoSQL数据库的开发人员在短时间内很难快速增加。

## 2.2 MongoDB技术详述[[5]](#endnote-5)

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是一个基于文档的数据库系统，将所有的记录、数据都以文档的形式存储在数据库中。文档是大多数用户可接触的数据结构的默认展现形式。MongoDB中的文档结构是BSON对象，支持所有的BSON类型，BSON是一种类似于JSON对象的文档格式。因为MongoDB支持所有的BSON类型，因此MongoDB的文档中可能包含一个值指向另一个文档的键值对，也可能包含由double、string或者日期类组成的数组。每一个MongoDB中的文档，都包含一个\_id字段。\_id字段是一个不可变的唯一的值，作为数据的主键，内容是一个ObjectId的类。ObjectId是一个12位的BSON类型，其中4位根据UNIX时间生成，3位为机器标识，两位为进程id，还有3位为随机生成的。因为ObjectId数据很小而且能够快速生成唯一值，所以，当\_id值没有指定时，MongoDB用它作为默认的\_id字段的值。

MongoDB的图片存储。MongoDB使用GridFS作为图片的存储机制。GridFS是一种能够将大型文件存储在MongoDB的文件规范，目前，所有的MongoDB的官方驱动都实现了GridFS规范。GridFS能够突破BSON文档对于文件大小不超过16M的限制，将文件用分割的形式存进数据库的一种规则。GridFS没有将大的文件存储到一个单独的文档中，而是将文件切分成一个个小的部分，将每个文件碎片作为独立的文档进行存储。默认情况下，GridFS将碎片的大小限制在256k之内。GridFS使用两个聚集来存储文件，一个用来存储文件碎片，另一个用来存储文件的元数据类型。当用户查询一个用GridFS存储的文件时，数据库驱动或者数据库客户端会重新聚集需要的文件碎片。用户可以用GridFS进行范围的查询，也可以使用GridFS进行“跳跃式”的查询获取任意视频或者音频碎片。GridFS不仅仅在存储超过16M限制的大文件时很有效，而且能够帮助用户在不将整个文件载入内存就使用文件。

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。

## 2.3 Python技术综述

Python是一种能够让用户更加快速工作并且更加有效地集成系统的编程语言，同时，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的显著优点[[6]](#endnote-6)：Python是一种面向对象的语言，它的类模块支持多态、操作符重载和多重继承等高级概念，并且以Python特有的简洁的语法和类型，面向对象编程十分易于使用；Python的使用和分发是完全免费的，用户可以从互联网上免费获得Python系统的源代码；Python拥有良好的可移植性，它是由美国国家标准化协会根据C语言编写的，可以在目前所有主流的平台上编译和运行；Python功能十分强大，它吩咐的工具集使它介于传统的脚本语言和系统语言之间，提供了所有脚本语言的简单和易用性，并且具有在编译语言中才能找到的高级软件工程工具；Python具有可混合型，可以以多种方式轻易地与其他语言编写的组件“粘结”在一起；Python简单易用，只需要简单地键入Python程序并运行就可以了；Python易于学习，Python语言的核心是非常简单易学的，它的语法清晰易读，尽量使用无异义的英语单词，便于上手入门。当然，Python语言也有一定的缺点，与C和C++这类变易语言相比，Python的执行速度还不够快。

Python的设计原则[[7]](#endnote-7)是“优雅”、“明确”、“简单”。“总是有多种方法来做同一件事”的理念在Python开发者中通常是难以忍受的。Python开发者的原则是“用一种方法，最好是只有一种方法来做一件事”。在设计Python语言时，如果面临多种选择，Python开发者一般会拒绝花俏的语法，而选择明确的没有或者很少有歧义的语法。由于这种设计观念的差异，Python源代码通常被认为具备更好的可读性，并且能够支撑大规模的软件开发。这些准则被称为[Python格言](http://zh.wikipedia.org/w/index.php?title=Zen_of_Python&action=edit&redlink=1)。Python的应用范围十分广泛，无论是web开发、操作系统的开发甚至是GUI的美化等方面，都有Python的影子。Python经常被用于Web开发。Python定义了WSGI标准应用接口来协调HTTP服务器与基于Python的Web程序之间的沟通。一些Web框架，如[Django](http://zh.wikipedia.org/wiki/Django)等，可以让程序员轻松地开发和管理复杂的Web程序。

## 2.4 Django技术综述

Django[[8]](#endnote-8)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。

Django的主要目标是[[9]](#endnote-9)：Django框架希望尽量Pythonic（即Django框架展现的属性应该符合语言的各个方面）一点，例如精悍的语法等；Django框架不要重复自己（Don’t Repeat yourself，DRY），将其作为值得努力接近的目标；松耦合与灵活性，Django提供了所有动态web应用所需的一切必要组件，同时也努力保持开放，可以按需要选择或多或少的Django组件；Django框架支持快速开发和敏捷开发，在不同层面上提供多种快捷方式，结合Python的灵活、简洁和强大。

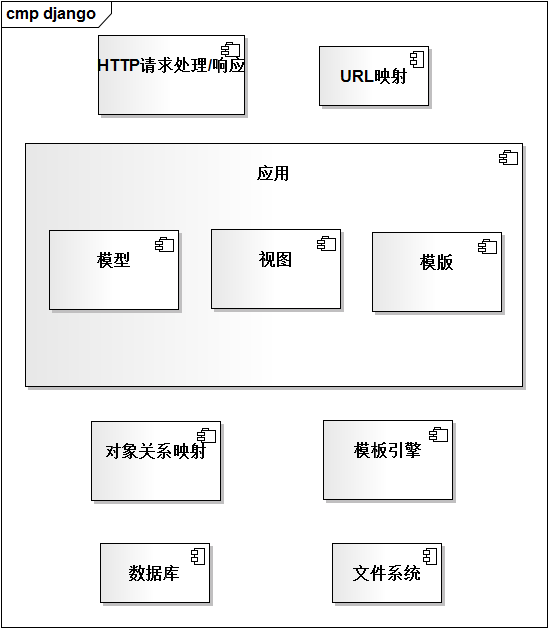


图2.1：Django框架整体架构

Django的具体框架结构图如图2.1所示。Http请求处理/响应模块用于处理接收到的request请求和返回response；URL映射用于将整个网站的URL进行映射，选择正确的处理函数；中间的应用层包含模块、视图和模板三个部分，模块部分对应数据的存储，视图部分是处理request的具体函数，模板用于生成返回的内容；对象关系引擎和数据库用于处理数据的存储；模板引擎和文件系统用于处理返回的response。

系统整个的处理request的请求流程，参见图2.2。系统接收到HTTP请求后，将HTTP请求打包成需要的格式，有request中间件进行处理。如果request请求需要使用view中间件，则由view中间件对request进行处理，而后调用视图函数。中途出现任何exception，则有exception中间件进行处理。最后由response中间件生成返回的HTTP Response。

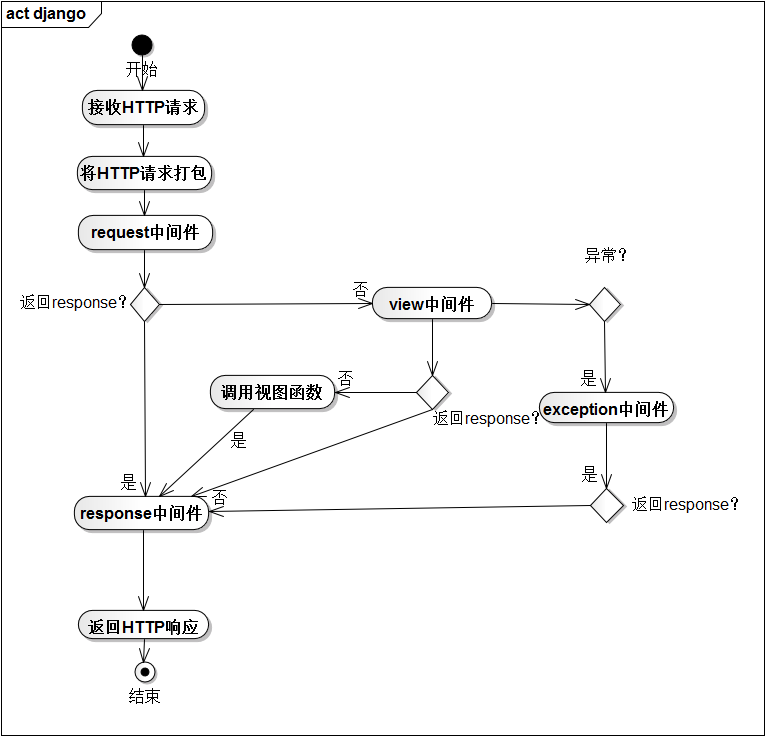


图2.2：Django框架处理request流程图

## 2.5本章小结

本章简单介绍了eBay Friends项目服务端模块设计的所有的技术，其中包括NoSQL技术，主要是MongoDB数据库和Python语言，主要是Django开发框架。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项目服务端的需求分析

## 3.1 eBay Friends项目整体概述

eBay Friends是一款基于Android系统的应用软件。用户可以免费注册并使用，每当用户在eBay购物平台购买了某项商品后，可以使用该平台分享自己与商品的合影，并为照片配上一段自己的录音。生成的分享可以在社交圈子中被其他好友看见，并被评论。

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。服务端采用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。

整个项目的运行过程是：客户端发送request请求至服务端，服务端中Django框架接收request并处理，其中通过Mongoengine对数据库MongoDB进行操作，而后由Django框架返回response相应。

## 3.2 eBay Friends项目服务端需求分析

### 3.2.1 eBay Friends项目需求概述

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，系统要实现用例参见表3.1。

表3.1：eBay Friends项目需求用例

|  |  |  |
| --- | --- | --- |
| 需求名称 | 需求概述 | 是否与服务端相关 |
| 登录 | 用户使用用户名和密码登录系统 | 是 |
| 获取新鲜事 | 用户获取最新的新鲜事信息 | 是 |
| 播放新鲜事声音 | 用户播放某一条新鲜事的声音 | 是 |
| 查看评论 | 用户查看新鲜事的评论信息 | 是 |
| 播放评论 | 用户播放新鲜事评论的声音 | 是 |
| 发表语音评论 | 用户对新鲜事进行评论（评论内容为语音） | 是 |
| 发表文字评论 | 用户对新鲜事进行评论（评论内容为文字） | 是 |
| 获取购物历史 | 用户获取自己购买过的商品 | 是 |
| 选择物品 | 用户选择需要发送的新鲜事的对应商品 | 是 |
| 选择图片 | 用户选择需要发送的图片 | 否 |
| 美化图片 | 用户美化需要发送的图片 | 否 |
| 录制语音 | 用户录制发送的语音内容 | 否 |
| 美化语音 | 用户美化发送的语音内容 | 否 |
| 发布新鲜事 | 用户发送新鲜事 | 是 |
| 推送 | 评论用户推送评论给新鲜事发布者 | 否 |
| 获取好友列表 | 用户获取自己所有的好友的列表 | 是 |
| 聊天 | 用户与其他用户实现聊天 | 否 |
| 获取个人信息 | 用户获得自己的个人信息 | 是 |
| 添加删除好友关系 | 用户可以自由添加或者删除与某人的好友关系 | 是 |
| 获取商品信息 | 用户可以根据新鲜事后附上的链接获取商品详细信息 | 是 |
| 购买商品 | 用户可以购买商品 | 是 |
| 搜索用户 | 用户可以通过用户名查找需要的用户 | 是 |
| 获取新鲜事详细信息 | 用户获取某一新鲜事的详细信息 | 是 |

### 3.2.2 eBay Friends项目服务端需求用例图

eBay Friends项目服务端需求用例图参见图3.1。在eBay Friends项目的所有用例中，与服务端相关的用例有：登陆用例，获取新鲜事用例，播放新鲜事声音用例，查看新鲜事评论用例，评论新鲜事用例，播放评论声音用例，获取购物历史用例，发送新鲜事用例，获取商品信息用例，获取新鲜事详细信息用例，搜索用户用例，购买商品用例，获取个人新鲜事列表用例，添加删除好友关系用例，获取好友列表用例。

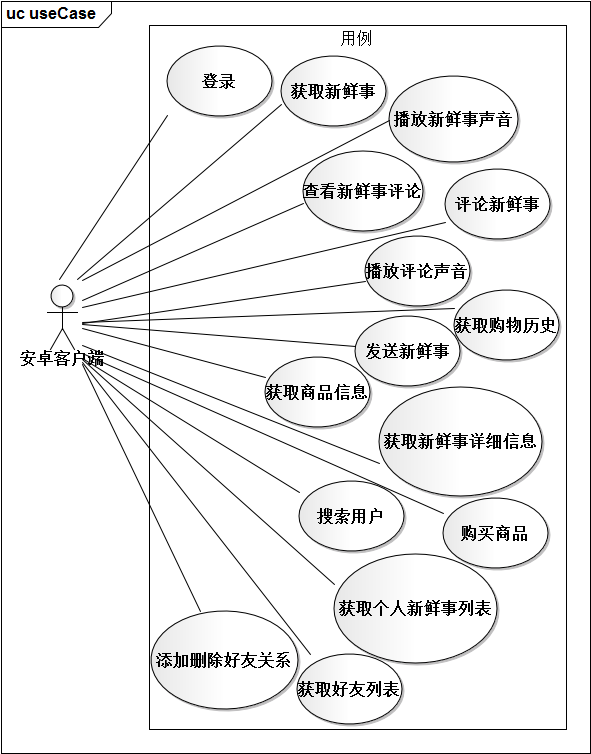


图3.1：eBay Friends项目用例图

### 3.2.3 eBay Friends项目服务端用例分析

在系统的所有需求中，对与服务端有交互的用例进行分析。

1.登录

用户下载eBay Friends项目后，需要登录，而后才可以使用系统，详细描述参见表3.2。

表3.2：登录用例详细描述

|  |  |
| --- | --- |
| ID | 1 |
| 名称 | 登录eBay Friends |
| 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3.3。

表3.3：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 | 服务端返回需要的新鲜事 |
| 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获取该用户的好友列表 3. 获取所有好友的新鲜事 4. 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 |

3.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3.4。

表3.4：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 | 客户端成功发出获取新鲜事声音的请求 |
| 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. 获取请求的音频参数 3. 从数据库获取需要的音频 4. 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.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3.5。

表3.5：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 | 客户端成功发出获取新鲜事评论的请求 |
| 后置条件 | 服务端返回新鲜事的评论 |
| 正常流程 | 1. 服务端接收到查看新鲜事评论的请求。 2. 获取请求的参数 3. 获取需要的新鲜事评论信息 4. 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3.6。

表3.6：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 | 客户端成功发出播放评论声音的请求 |
| 后置条件 | 用户成功听到评论的声音 |
| 正常流程 | 1. 服务端接收到播放评论声音的请求。 2. 获取声音的参数 3. 获取需要的声音 4. 返回音频 |
| 异常流程及解决方案 | 获取声音不成功：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3.7。

表3.7：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3.8。

表3.8：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. 服务端获取该用户的购物历史 4. 服务端返回购物历史 |
| 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 | 8 |

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3.9。

表3.9：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和新鲜事的内容（图片和声音） 3. 添加新鲜事 4. 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.添加删除好友

用户在系统中发现另一个用户，希望关注他。或者，用户希望对某个用户解除关注。详细描述参见表3.10。

表3.10：添加删除好友用例的详细描述

|  |  |
| --- | --- |
| ID | 9 |
| 名称 | 添加删除好友 |
| 描述 | 产生原因：用户进入消息系统，希望关注某个用户或者解除关注。  大致过程：服务器接收到添加删除好友的请求，获取请求中用户信息，将关注或者解除关注的信息添加到数据库。  输出结果：成功关注或者解除关注。 |
| 优先级 | 中 |
| 前置条件 | 客户端成功发出关注或者解除关注的请求 |
| 后置条件 | 成功添加关注或者解除关注 |
| 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. 添加关注或者解除关注 4. 返回操作结果 |
| 异常流程及解决方案 | 操作失败：返回操作失败 |
| 相关用例 | 12 |

10.获取商品信息

用户根据其他用户发送的新鲜事，希望查看某个商品的详细信息。详细描述参见表3.11。

表3.11：获取商品信息的详细描述

|  |  |
| --- | --- |
| ID | 10 |
| 名称 | 获取商品信息 |
| 描述 | 产生原因：用户进入消息系统，希望获取某个商品的信息。  大致过程：服务器接收到获取商品信息的请求，获取请求中用户信息，返回用户需要的商品信息。  输出结果：成功返回商品信息。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出获取商品信息的请求 |
| 后置条件 | 客户端成功返回商品信息 |
| 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. 获取需要的商品信息 4. 返回商品信息 |
| 异常流程及解决方案 | 获取失败：返回操作失败 |
| 相关用例 | 11 |

11.购买商品

用户查看商品信息后，希望购买这个商品。详细描述参见表3.12。

表3.12：获取购买商品用例的详细描述

|  |  |
| --- | --- |
| ID | 11 |
| 名称 | 购买商品 |
| 描述 | 产生原因：用户进入消息系统，获取商品信息后，希望购买商品的信息。  大致过程：服务器接收到购买商品的请求，获取请求中用户信息和商品信息，购买商品。  输出结果：成功购买结果。 |
| 优先级 | 中 |
| 前置条件 | 客户端成功发出购买商品信息的请求 |
| 后置条件 | 成功返回购买结果 |
| 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和商品信息 3. 将购买信息添加到数据库 4. 返回商品信息 |
| 异常流程及解决方案 | 购买失败：返回操作失败 |
| 相关用例 | 10 |

12.获取好友列表

用户进入系统后，可以查看自己的好友列表。详细描述参见表3.13

表3.13：获取好友列表用例的详细描述

|  |  |
| --- | --- |
| ID | 12 |
| 名称 | 获取好友列表 |
| 描述 | 产生原因：用户进入消息系统，希望获取自己的好友列表。  大致过程：服务器接收到获取好友列表的请求，获取请求中用户信息，查看该用户的好友列表。  输出结果：成功返回好友列表。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出查看好友列表的请求 |
| 后置条件 | 成功返回好友列表 |
| 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. 从数据库中获取好友信息 4. 返回好友信息 |
| 异常流程及解决方案 | 购买失败：返回操作失败 |
| 相关用例 | 9 |

13.搜索用户

用户进入系统后，可以根据用户名搜索用户。详细描述参见表3.14

表3.14：搜索用户用例的详细描述

|  |  |
| --- | --- |
| ID | 13 |
| 名称 | 搜索用户 |
| 描述 | 产生原因：用户进入消息系统，希望搜索某个用户。  大致过程：服务器接收到搜索用户的请求，获取请求中搜索的用户名，返回搜索结果。  输出结果：成功返回好友列表。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出搜索用户的请求 |
| 后置条件 | 服务端成功返回搜索结果 |
| 正常流程 | 1. 客户端接收到发送新鲜事请求 2. 获取请求中的搜索用户名 3. 在数据库中匹配用户 4. 返回搜索结果 |
| 异常流程及解决方案 | 搜索无结果：返回空列表 |
| 相关用例 | 无 |

14.获取个人新鲜事

用户进入系统后，可以查看某个用户的发送的新鲜事。详细描述参见表3.15

表3.15：获取个人新鲜事用例的详细描述

|  |  |
| --- | --- |
| ID | 14 |
| 名称 | 获取个人新鲜事 |
| 描述 | 产生原因：用户进入消息系统，希望获取某个用户发送的所有新鲜事。  大致过程：服务器接收到获取个人新鲜事的请求，获取请求中的用户名，查看该用户发送的所有新鲜事。  输出结果：成功返回用户的所有新鲜事。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出获取个人新鲜事的请求 |
| 后置条件 | 成功返回用户的所有新鲜事 |
| 正常流程 | 1. 客户端接收到发送新鲜事请求 2. 获取请求中的用户名 3. 从数据库获取该用户的所有新鲜事 4. 返回新鲜事列表信息 |
| 异常流程及解决方案 | 获取新鲜事失败：返回空值 |
| 相关用例 | 无 |

15.获取新鲜事详细信息

用户进入消息系统后，可以根据消息系统的推送值获取某个新鲜事的所有信息。详细描述参见表3.16

表3.16：获取新鲜事详细信息的详细描述

|  |  |
| --- | --- |
| ID | 15 |
| 名称 | 获取新鲜事详细信息 |
| 描述 | 产生原因：用户进入消息系统，根据消息系统推送的新鲜事信息，获取该新鲜事的详细信息。  大致过程：服务器接收到获取新鲜事详细信息请求，获取请求中新鲜事信息，查看该新鲜事的详细信息。  输出结果：成功返回新鲜事的详细信息。 |
| 优先级 | 高 |
| 前置条件 | 客户端成功发出获取新新鲜事详细信息的请求 |
| 后置条件 | 成功返回新鲜事的详细信息 |
| 正常流程 | 1. 客户端接收到发送新鲜事请求 2. 获取请求中的新鲜事信息 3. 从数据库中获取该新鲜事的详细信息 4. 返回新鲜事信息 |
| 异常流程及解决方案 | 获取新鲜事失败：返回空值 |
| 相关用例 | 无 |

### 3.2.4 eBay Friends项目服务端ERD

eBay Friends项目数据的ERD如图3.2所示。实体关系包含三个主要的实体，user实体表示用户，包括6个属性：个人头像、密码、好友列表、id、购买记录和用户名；新鲜事实体包括5个属性：上传时间、描述、图片、声音和评论；商品实体包括4个属性：价格、图片列表、名称和描述。用户实体可以发出新鲜事，每个新鲜事都必须对应一个商品。

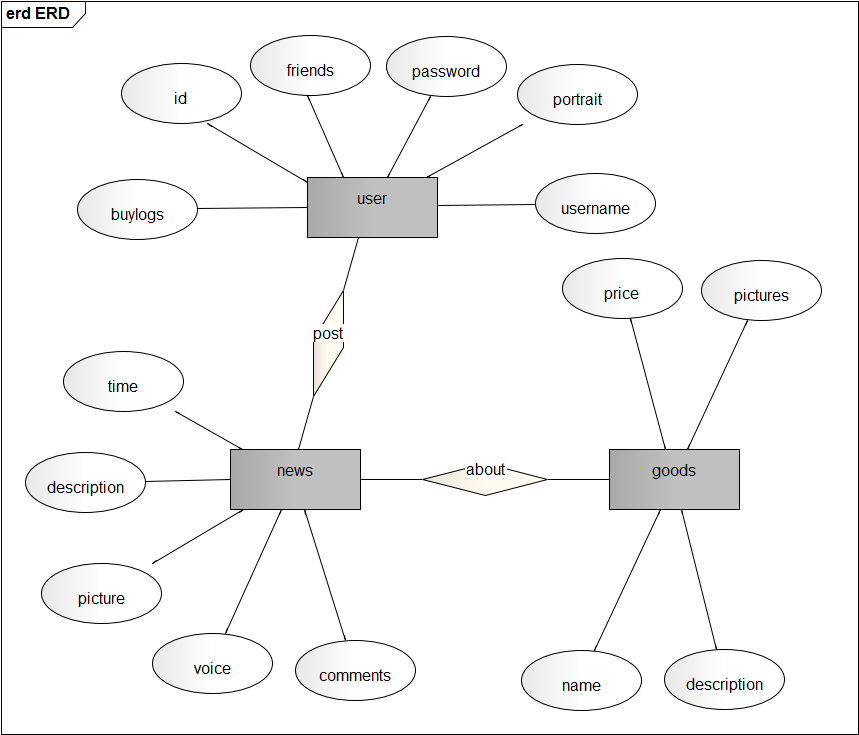


图3.2：eBay Friends项目ERD

## 3.3 本章小结

本章详细描述了eBay Friends项目的需求，针对本文完成的服务端模块进行了详细的分析。给出了详细的用例图、用例描述和ER图。从需求的角度，详细说明了eBay Friends项目的开发过程。

# 第四章 eBay Friends项目服务端模块的设计

## 4.1 eBay Friends项目的概要设计

eBay Friends项目的整体概要设计如图4.1所示。总体的架构分为客户端和服务端，客户端使用android技术实现，服务端使用Django框架、MongoDB服务器和Mongoengine中间件，客户端与服务端的通信使用http请求实现。

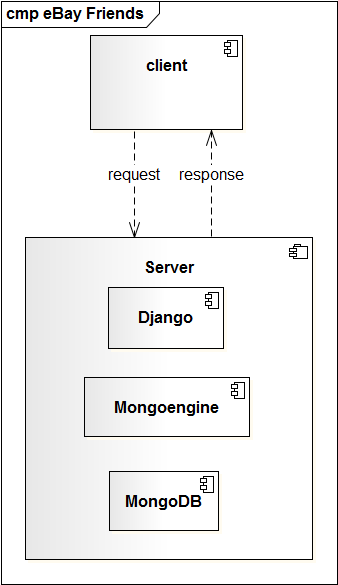


图4.1：eBay Friends项目整体概要设计图

eBay Friends项目采用Django框架搭建服务器，Django框架有两个文件urls.py和settings.py作为服务器的配置文件，分别用于设置服务器的相关属性和网站url的架构。Django框架与传统的MVC设计模式稍有不同，采用独有的MTV模式，包括Models、Templates和Views。由于客户端使用的android开发，所以服务端无须使用Template模板，只需要设计Models和Views。Models定义数据结构，与数据库的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用于处理客户端发送的请求，并返回需要的响应。系统服务端的概要设计如图4.2所示。

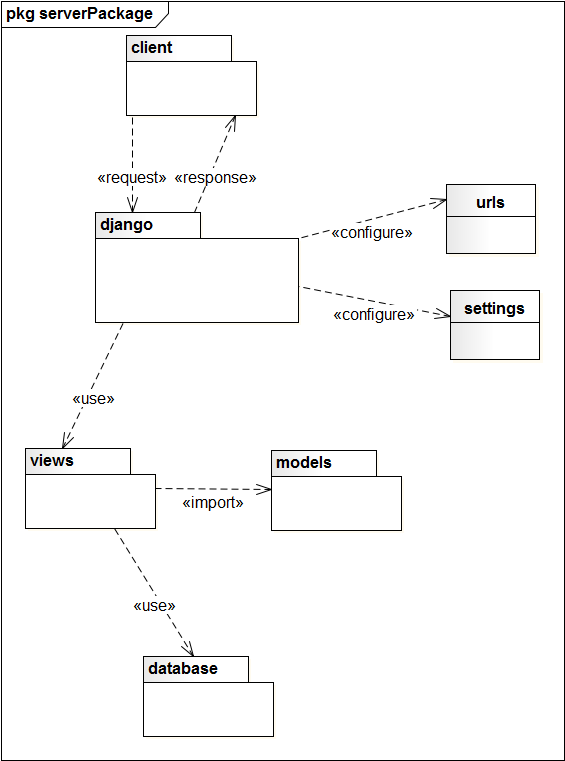


图4.2：服务端概要设计图

## 4.2 eBay Friends项目服务端的详细设计

### 4.2.1 eBay Friends项目服务器模块概述

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 eBay Friends项目服务器模块的详细设计

服务器模块的详细设计类图，如图4.3所示。如下图所示，服务器大概分为三个部分：Django-admin部分，views部分和models部分。

Django-admin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，同时，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。

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服务器找到匹配的url后，调用views中的具体方法，对数据库进行操作获取数据，并按照需要对数据进行处理，生成需要的HttpResponse并返回。

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。

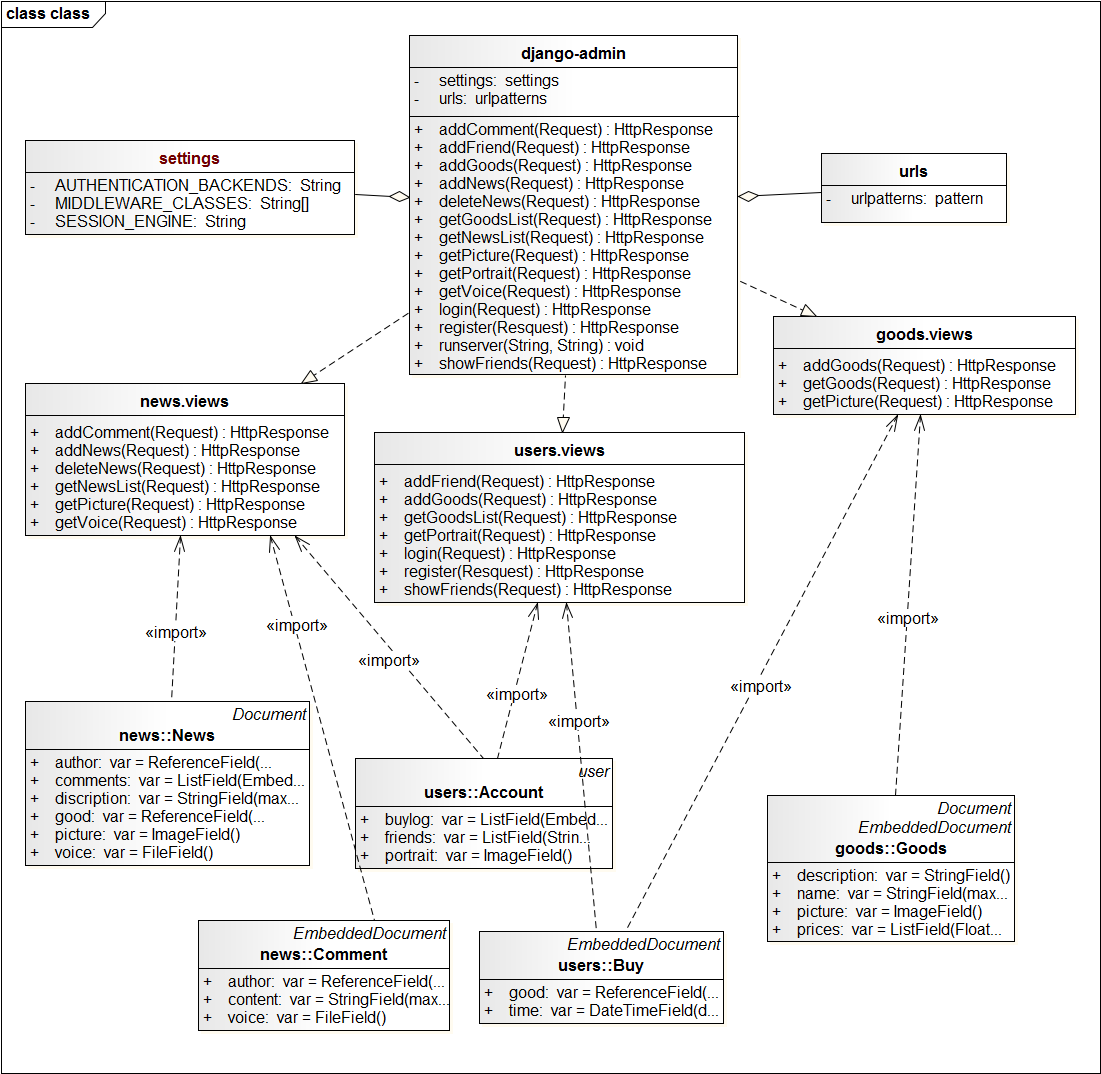


图4.3：服务端详细设计类图

### 4.2.3 eBay Friends项目服务端模块复杂任务详细设计

对于每个用例，服务端的设计思想是：首先服务器端接收客户端发送的请求，分析请求的类型、路径和参数，根据请求路径对应的views方法，对请求进行处理和分析，对数据库进行操作。本文将对登录以及获取新鲜事列表的用例进行详细描述，具体设计顺序图参见图4.4。。

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，如果登录成功则返回用户名和头像的图片路径；否则，返回登录失败的信息。

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，根据请求参数从数据库中获取新鲜事列表，对新鲜事列表进行处理，返回一个包含json文的响应。客户端根据收到的响应中的json，提取出每条新鲜事的图片获取路径和声音获取路径。

发送获取图片请求，Django-admin模块处理请求，调用news模块中getPicture方法，根据请求参数从数据库中获取图片，返回图片；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，根据请求参数从数据库中获取声音，返回声音。

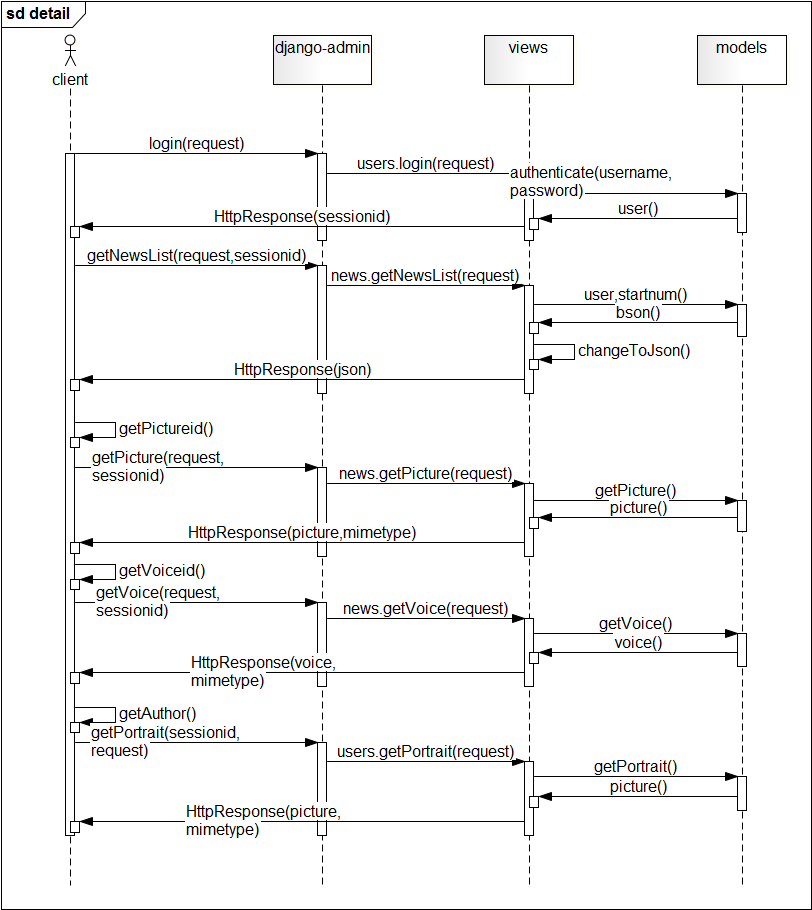


图4.4：登录以及获取新鲜事列表的顺序图

## 4.3 eBay Friends项目数据库设计

### 4.3.1 eBay Friends项目数据库概述

数据库模块主要采用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，数据库采用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。

### 4.3.2 eBay Friends项目数据库的详细设计

数据库设计图，如图4.5所示。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。数据库包括七张表格，其中User、News和Goods三张表格对应服务端模块的models中定义的三个数据类型，image.chunks、images.files、fs.chunks和fs.files四张表格有GridFS存储系统生成，对图片和音频的存储。除此之外，数据库中还包含一张Django-session表，这张表的主要用途是记录用户登录后留在服务端的session[[10]](#endnote-10)（在计算领域中，session是用于记录用户在一个多用户的系统中登录或登出的日志），由Django框架中的中间件生成。

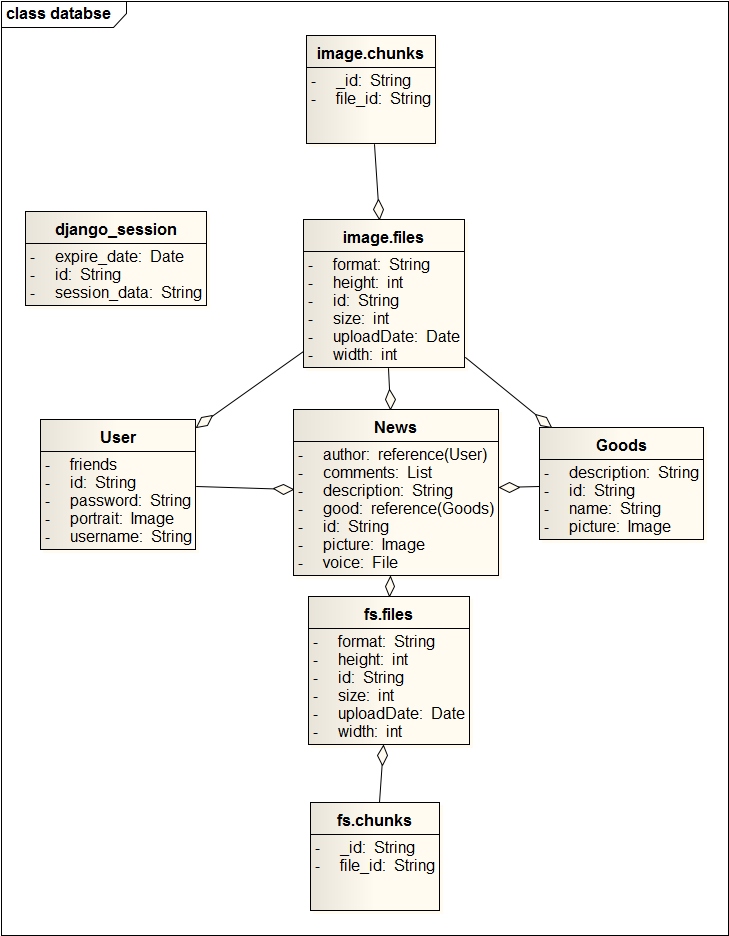


图4.5：数据库设计图

数据库中的每张表的数据描述如表4.1-表4.8所示，表格中详细列举了每个collection中的字段，并对字段的意义进行了详细的描述。

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 |

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 |

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 |

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 |

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 |

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 |

## 4.4 eBay Friends项目接口描述

eBay Friends项目大致分为两个模块：客户端和服务端。客户端与服务端之间的接口使用http协议，客户端发送http请求，服务端接收http请求，根据请求的参数和路径进行相应的操作，返回响应。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[[[11]](#endnote-11)]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。服务端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的格式保存到数据库。

## 4.5 本章小结

本章介绍了eBay Friends项目整体概要设计，对服务端模块的详细设计进行了分析，列举了服务端的复杂用例的具体实现过程，同时给出了数据库的详细设计，并描述了整个项目不同模块直接的接口连接方式。

# 第五章 eBay Friends项目服务端模块的实现

## 5.1 eBay Friends项目服务端模块的实现环境

eBay Friends项目实现在windows7系统下，使用基于Python语言的Django框架作为服务器，使用MongoDB作为数据库，使用Mongoengine作为服务器与数据库的连接。eBay Friends开发过程中使用git[[12]](#endnote-12)（git是一款分布式版本控制系统，可以让人更加高效的协同工作，从而提高生产率。）作为版本控制工具，使用eclipse作为开发工具（使用pydev[[13]](#endnote-13)兼容Python，pydev是一款基于eclipse的Python IDE，用于Python的开发），使用RockMongo[[14]](#endnote-14)（RockMongo是PHP5写的一个MongoDB管理工具）作为数据库可视化工具，使用EA（Enterprise Architect）作为作图的工具。具体的开发和实现环境参见表5.1。

表5.1：eBay Friends项目的实现环境

|  |  |  |
| --- | --- | --- |
| 名称 | 工具 | 版本 |
| 操作系统 | Windows | Windows7 32位 专业版 |
| 语言 | Python | 2.7.3 |
| 框架 | Django | 1.4.1 |
| 数据库 | MongoDB | 2.4.3 |
| 数据库连接 | Mongoengine | 0.8 |
| 版本控制 | Git | 1.8.1.2 |
| 开发IDE | Eclipse | 4.2 |
| 开发插件 | Pydev | 2.6 |
| 数据库可视化 | RockMongo | 1.1.2 |
| 作图 | Enterprise Architect | 8.0.858 |

## 5.2 eBay Friends项目服务端模块的实现代码

### 5.2.1 eBay Friends项目服务器模块配置部分的实现

服务端使用的是Django的模块。安装好Django框架的开源模块代码之后，框架自动生成Django-admin.py服务器，在此基础上，需要对urls.py和setting.py两个文件进行修改，使之能够正确处理客户端发送的请求。

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设置等，具体代码如图5.1所示。

|  |
| --- |
| from Mongoengine import connect  connect("eBayWithVoice")  #配置数据库连接  FILE\_UPLOAD\_HANDLERS=("Django.core.files.uploadhandler.TemporaryFileUploadHandler",)  #文件上传处理设置  AUTHENTICATION\_BACKENDS = ('Mongoengine.Django.auth.MongoEngineBackend',)  #登录验证设置  SESSION\_ENGINE = 'Mongoengine.Django.sessions'  #session处理设置  ROOT\_URLCONF = 'eBayWithVoice.urls'  #url设置 |

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，找到需要调用的方法，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$','users.views.getGoodsList'),  #user包中涉及的urls  url(r'^news/addNews$','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$','news.views.deleteNews'),  #news包中的涉及的urls  url(r'^goods/addGoods$','goods.views.addGoods'),  url(r'^goods/getGoods$','goods.views.getGoods'),  url(r'^goods/getPicture$','goods.views.getPicture'),  #goods包中涉及的urls  ) |

图5.2：urls.py代码

### 5.2.2 eBay Friends项目服务端模块视图的实现

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

1.user模块中views代码主要用于处理关于用户个人信息的请求，该模块主要包含六个函数，分别对应登录用例、获取购物列表用例、添加删除好友关系用例、购买商品用例、获取好友列表用例和搜索用户用例，其中登录用例，添加删除好友关系用例和获取用户购买记录用例的关键代码如图5.3所示。

|  |
| --- |
| endpoint = "http://192.168.47.19:8080/"#定义目标url  #登录用例  def login(request):  ……#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  ……  auth.login(request, user)  #使用Django中的user类中的login，登录成功  return HttpResponse("success")  #获取用户头像  def getPortrait(request):  ……#从json中获取id并根据请求中的id找到用户  return HttpResponse(user.portrait.read(),mimetype="image/jpeg")  #返回用户头像的图片  #添加删除好友关系用例  @login\_required#获取用户session，判断用户身份  def changeFollow(request):  ……#获取参数  if str(friend.id) in user.friends:  user.friends.remove(str(friend.id))  else:  user.friends=user.friends+[str(friend.id)]  user.save()  #如果已经关注，则取消关注；否则，添加关注。  return HttpResponse("add success")  #获取购买记录用例  def getGoodsList(request):  buylogs=request.user.to\_Mongo()['buylog']  for buylog in buylogs:  ……#处理数据，返回json格式  return HttpResponse(ju.dumps(buylogs)) |

图5.3：users模块views.py的实现

2.news模块中views代码主要用于处理新鲜事相关信息的处理，这个模块主要包含八个函数，分别对应发送新鲜事用例、获取新鲜事列表用例、播放新鲜事声音用例、添加评论用例、获取评论列表用例、播放评论声音用例、获取个人新鲜事用例、获取新鲜事详细信息用例，其中发送新鲜事和获取新鲜事列表用例参见图5.4

|  |
| --- |
| #发送新鲜事用例  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")  #获取新鲜事列表  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) |

图5.4：news模块views.py的实现

### 5.2.3 eBay Friends项目服务端模块数据库部分实现

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. users模块中models.py的实现如图5.5所示，包括friends字段、portrait字段、buylog字段。

|  |
| --- |
| 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5.5：users模块models.py的实现

2.news模块中的models.py的实现如图5.6所示，包括author字段、discription字段、voice字段、comments字段、picture字段、good字段和time字段。

|  |
| --- |
| 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  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5.6：news模块models.py的实现

## 5.3 eBay Friends项目服务端运行截图

数据库运行步骤[[15]](#endnote-15)：首先，进入数据库文件夹；运行命令Mongod，加上参数—dbpath，加上数据需要保存到的文件夹路径；最后，服务端数据库运行时截图如图5.7所示。

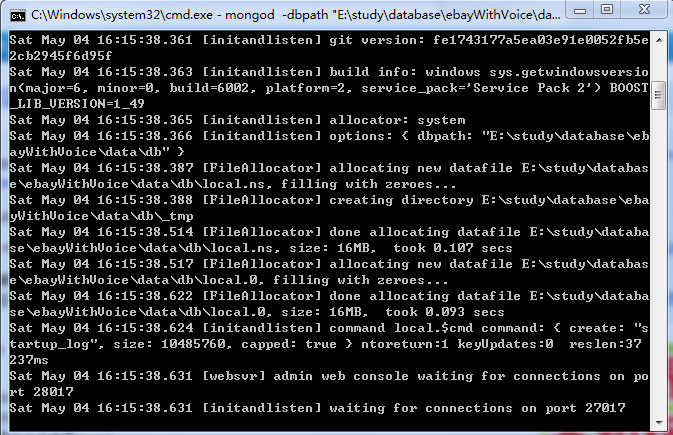


图5.7：数据库运行时截图

服务器运行步骤[[16]](#endnote-16)：首先，进入服务器代码存放的文件夹；然后，运行命令python manage.py runserver 0.0.0.0:8080启动服务器；最后，服务器运行时截图如图5.8所示。

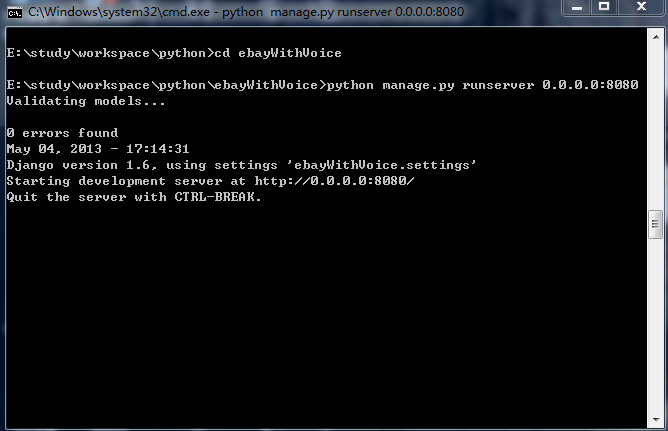


图5.8：服务器运行时截图

## 5.4本章小结

本章介绍了eBay Friends项目服务端的实现代码，对服务端模块各个部分中对应的用例给出了核心算法，对数据库各个字段的具体描述做了详细的描述。此外，本章还介绍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

本文概述了目前互联网上电子购物网站的商品评价系统的背景，详细介绍了商品评价系统对于购物网站、商家以及顾客的重要意义，研究综述了当前各大网站的评价系统的情况。

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一个新型的基于eBay购物平台的商品分享和评价系统——eBay Friends项目，论述了eBay Friends项目对于购物平台的巨大作用，以及相比于其他现有的商品评价系统的优势，并且详细地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。

首先，本文描述了eBay Friends项目的整体需求，并对服务端模块的需求用例进行了整理，对每一个用例的实现步骤以及出现的异常情况做了详细的描述。本文对服务端涉及到的各种数据进行了划分，确定了使用的数据结构和类型，对数据库中各个实体之间的关系和属性作了明确的定义。

然后，本文阐述了eBay Friends项目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。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。服务端采用Django服务器与MongoDB数据库结合，使用Mongoengine作为服务器与数据库的连接。前台与后台使用最为普遍的HTTP请求作为交互的载体，前台发送HTTP请求，服务器接收请求，对不同的请求采用不同的处理方式，对数据库进行操作，返回需要的响应。

最后，本文给出了eBay Friends项目的服务端实现。根据设计时定义的服务端模块，本文分别对用户部分、新鲜事部分和商品部分的方法进行了具体阐述，给出了各个方法与需求部分分析的用例之间的对应关系，并对复杂的用例过程进行了详细的代码分析，标明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。

在非功能需求方面，系统将会使用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。

# 参考文献

# 致谢

花费了大学最后的几个月，我终于完成了这篇论文。尽管这段时光略带苦涩，当我仍然觉得它将永远雕刻在我的青春记忆中。在项目实现和论文写作的过程中，我遇到过诸多障碍，在老师和同学们的热情帮助下，我成功克服了这些困难，对此，我充满感激。

首先，我要感谢我的导师任桐炜老师。任老师在整个毕业设计完成的过程中给了我很多帮助和指导，一次次地帮助我修改文章，提出改进意见。从论文的选题开始直至最终成文，任老师始终认真负责的鼓励我们，给我们细心地指导。正是在任老师这样的认真辅导下，我最终顺利完成了这篇毕业设计。

同时，我要感谢给我提供项目平台的eBay公司，正是公司的项目活动给了我们这样一个开发并展示的机会。

我还要感谢项目组的其他三位同学：张雨寒、叶韵致和张航，正式在他们的细心配合下，项目能够顺利完成。此外，他们在论文的思路和内容上都给了我很大的帮助。

我还要感谢我的室友，感谢他们大学四年给了我诸多的关心和照顾，提出了诸多的宝贵意见。

我还要感谢的我的父母，正是在他们的默默关心和支持下，我才能够顺利的完成大学四年的学业。

最后，我要感谢所有关心我、指导我和给过我帮助的朋友们，谢谢你们！

1. Abraham Silberschatz等，杨冬青 唐世伟，数据库系统概念（原书第4版），北京市西城区百万庄大街22号：机械工业出版社，2003年3月，398页 [↑](#endnote-ref-1)
2. 佐佐木达也，罗勇，NoSQL数据库入门，北京市崇文区夕照寺街14号a座：人民邮电出版社，2011年5月，6-7页 [↑](#endnote-ref-2)
3. 佐佐木达也，罗勇，NoSQL数据库入门，北京市崇文区夕照寺街14号a座：人民邮电出版社，2011年5月，9-10页 [↑](#endnote-ref-3)
4. 佐佐木达也，罗勇，NoSQL数据库入门，北京市崇文区夕照寺街14号a座：人民邮电出版社，2011年5月，13-15页 [↑](#endnote-ref-4)
5. Introduction to MongoDB，http://www.Mongodb.org/about/introduction/ [↑](#endnote-ref-5)
6. Mark Lutz，李军 刘红伟，Python学习手册（第4版），北京市西城区百万庄大街22号：机械工业出版社，2011年4月，23-32页 [↑](#endnote-ref-6)
7. Python-维基百科，http://zh.wikipedia.org/wiki/Python [↑](#endnote-ref-7)
8. Django-维基百科，http://zh.wikipedia.org/wiki/Django [↑](#endnote-ref-8)
9. Jeff Forcier等，徐旭铭等，Django Web开发指南，北京市西城区百万庄大街22号：机械工业出版社，2009年5月，66-67页 [↑](#endnote-ref-9)
10. Login\_session-维基百科，http://en.wikipedia.org/wiki/Login\_session [↑](#endnote-ref-10)
11. JSON-维基百科，http://zh.wikipedia.org/wiki/JSON [↑](#endnote-ref-11)
12. 蒋鑫，Git权威指南，北京市西城区百万庄大街22号：机械工业出版社，2011年4月，1页 [↑](#endnote-ref-12)
13. PyDev，http://pydev.org/ [↑](#endnote-ref-13)
14. RockMongo-介绍，http://www.rockMongo.com/wiki/introduction?lang=zh\_cn [↑](#endnote-ref-14)
15. Install MongoDB On Windows，http://docs.Mongodb.org/manual/tutorial/install-Mongodb-on-windows/ [↑](#endnote-ref-15)
16. Jeff Forcier等，徐旭铭等，Django Web开发指南，北京市西城区百万庄大街22号：机械工业出版社，2009年5月，240-254页 [↑](#endnote-ref-16)