浙 江 大 學

本 科 生 毕 业 设 计 报 告

项目名称 基于 iOS 的投资信息系统的设计与实现

姓　　名 谭歆

学　　号 3080101204

指导教师 蔡亮

专　　业 软件工程

学　　院 计算机学院

A Dissertation Submitted to Zhejiang University for the Degree of Bachelor of Engineering

TITLE: The Design and Implementation of an Investment Information System based on iOS

Author: Xin Tan

StudentID 3080101204

Mentor Liang Cai

Major: Software Engineering

College: Computer Science

Submitted Date: 2012-05-22

浙江大学本科生毕业论文（设计）诚信承诺书

1.本人郑重地承诺所呈交的毕业论文（设计），是在指导教师的指导下严格按照学校和学院有关规定完成的。

2.本人在毕业论文（设计）中引用他人的观点和参考资料均加以注释和说明。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

毕业论文（设计）作者签名：

_____年_____月_____日

摘要

自从苹果公司发布第一代 iPad 以来，平板设备就在越来越多的领域上展现了其卓越的生命力与竞争力。云计算（Cloud Computing）是一个新兴的互联网应用领域，云存储是云计算中着重于提供高性能存储空间的一种互联网服务，二者可以弥补平板设备的不足的存储及性能问题。

本项目以平板电脑为平台，云存储为辅助存储开发了一套将 iPad 应用到投资信息管理上，并借助云存储扩展 iPad 对资源的访问能力的信息系统。本文介绍了系统的大体架构，并分别从客户端和服务端的设计与实现细节上讲述了项目的开发过程。文章重点阐述了客户端实现上的技术难点及解决方法，通过对设计模式的理解讲解了一些常用设计模式在实际项目中的应用。

此项目中本人负责全部客户端代码的设计与实现，同时参与了部分服务端的设计。客户端应用程序基于 iOS 系统开发，主要功能包括文档的下载及查看，创建与同步 Fund、Notes，RSS 订阅浏览等，其中支持多个云存储的访问及文档下载和文档的预览图生成是本项目的难点。

关键词 iPad；平板电脑；iOS；云存储；同步；信息系统；RSS

Abstract

Since Apple released the 1st generation of iPad, tablet devices are showing its superior vitality and competitiveness in a growing number of fields. Cloud Computing is an emerging Internet application domain, cloud storage is a kind of cloud computing which focused on providing high-performance storage space, these two can make up the performance problem of tablet devices.

This project which depends on Tablet PC platform, and uses cloud storage as secondary storage, develops a system that applied the iPad application into investment management, and improved the accessibility to resources by cloud storage. This article introduced the system in general architecture, and explained the project development process from both the customer side and service side design, to the detailed implementation. This article emphasized on the technique difficulties of the implementation of iPad client and the solution, and also explained the usage of some common design patterns in practical projects.

In this project, I am responsible for all client code design and implementation, and also participated in some of the design of the services side. Client application development is based on iOS, main features include document download and view, create and sync Fund, Notes, RSS feed View, and supporting access to multiple cloud storage, file downloading and document preview generation are the difficult parts of the project.

Keywords iPad, Tablet, iOS, Cloud Storage, Synchronize, Information System, RSS

目录

第1章 绪论

1.1 项目背景

在讯息千变万化的 21 世纪，如何及时、准确地把握投资信息将成为成功的关键。随着生活节奏的加快，信息来源的增加，传统的 PC 上处理事务的方式已经捉襟见肘，略显不足。人们越来越迫切地需要能够随时随地、自由地办公，虽然笔记电脑可以部分满足要求，然而由于其自身的如电池续航能力不强、体积大，携带笨重等缺点，大部分时间人们还是会在桌面上使用笔记本电脑，它的移动性也就成为了摆设。

平板电脑的出现会让这一切有所改观。

1.2 平板设备的发展

2010 年 1 月 28 号，苹果公司发布了第一代 iPad，全球便掀起了一股平板电脑的浪潮。

平板电脑这一概念并不是苹果首创，而是来自微软。当时微软构想了一款不需要键盘、不用翻盖、方便携带却又功能完整的 PC 产品[1] ，却又没能成功地进行推广，但由此衍生出了可用触笔写且屏幕可旋转的平板电脑变体——tablet PC。IBM 的 ThinkPad X 系列中的几款产品便是很好的原型。

苹果公司的 iPad 重新定义了平板这一名词，使其更加切合产品的本意。与以往的 tablet PC 不同，平板电脑（Pad）自身具有诸如多点触控、重力感应、携带方便等特点，很快便占领了消费者的市场。对于商务人士而言，出差或旅行途中携带大量沉重的行李不是件轻松的事，传统的笔记本电脑自然无法胜任。平板电脑的体积则介于笔记本电脑与掌上电脑（PDA）之间，结合了掌上电脑的轻便与笔记本电脑的强大功能，对日常的上网、即时通讯、文档查看、收发邮件等都可以轻松应对。可以说，平板电脑的流行是对上网本的一个致命打击。

基于平板独道的交互体验，各种新颖的应用层出不穷；基于 iOS 的 iPad 和基于开源的 Android 系统及 Windows 8 的各种平板电脑新产品的出现也更是为消费者们提供了极大的选择余地。

1.2.1 平板电脑的优势

当今主流的平板电脑都配备了 10 寸左右的多点触控屏、重力及加速度感应计、磁场感应计、三轴陀螺仪、蓝牙、WIFI、光线感应器等等，多元化的传感器也使得它拥有足以颠覆传统的键盘、鼠标等的交互体验。

总体上讲，平台电脑较笔记本电脑和普通桌面电脑有以下方面的优势：

- 便携性。小且方便移动使用。
- 交互性。多点触控，方向感应等。
- 持久性。更长的电池续航时间。
- 活跃性。应用开发周期短，更新及维护方便。
- 安全性。目前病毒、恶意代码较少。

如图 1-1 所示，除了 1 月份增长数较低外，2 月、3 月都达到了八万多个应用，即每天都有 2800 多新应用上架。而这还仅仅是 App Store 的中国区的统计结果[13] 。平板电脑的活跃性可想而知。

图 1-1 2012 年第一季度 App Store 中国区新增应用个数

但同时，另一方面由于体积小，散热差，它又存在存储空间小、性能低下、单个硬件不能更新升级、外设接入不方便等不足。

1.2.2 平板电脑的定位

平板电脑从一开始流行就注重在娱乐上，不难发现，苹果公司 App Store 中游戏应用总是更新最快的（如图 1-2）。其他一些行业应用，如电子商务、医学、教育、物联网等也都在发展，但相对缓慢，这也是本项目诞生的背景和机遇。平板电脑定位于商务是十分合理的，首先商务注重移动性，其次注重信息的实时性和可展示性。移动性平板天生具备，配合 WIFI 和 3G 平板也能轻松做到实时在线，外加比掌上电脑好得多的显示效果，向客户介绍业务和成绩等均不在话下。

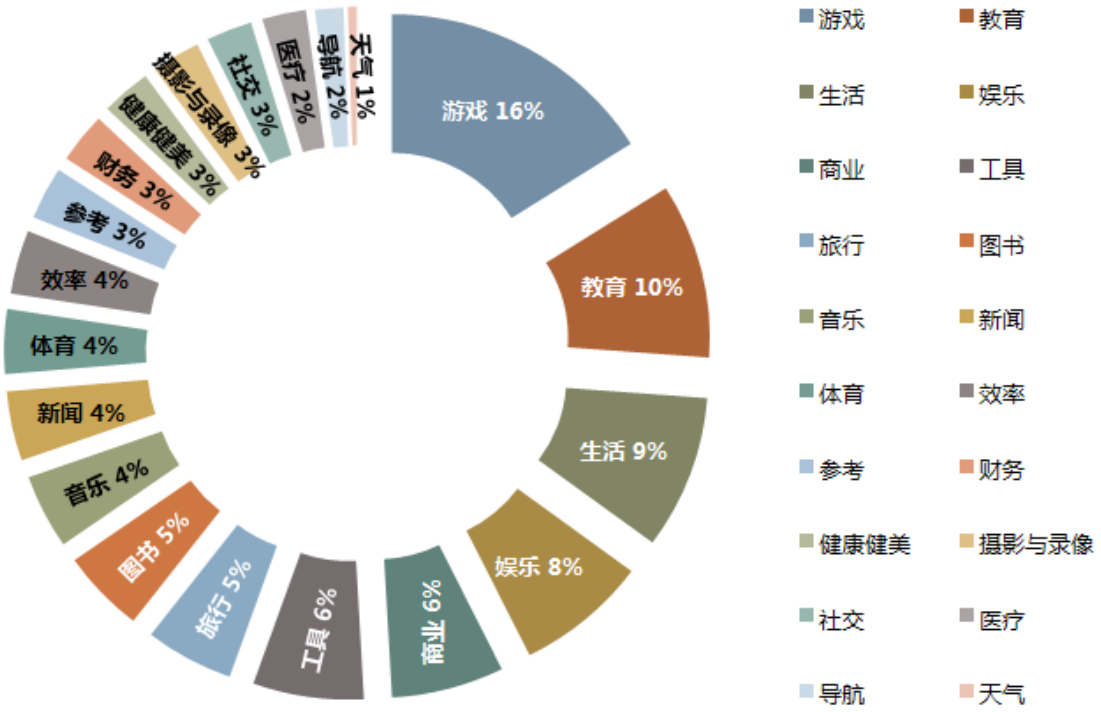

图 1-2 2012 年 3 月新增 iPad 应用类别比例

1.3 云计算

云计算是一种基于互联网的新型的计算方式，这种方式允许将集成化的、拥有强大性能的软硬件资源按需共享给其它设备使用，它是 80 年代大型计算机到客户端、服务器的大转变之后的又一种巨变[2] 。云计算拥有复杂的后台控制，却可以留给用户简单、明了的应用接口，用户不需要了解云计算背后的技术细节，不需要知道云计算服务的地理位置，无需计算机相关的知识，更不用直接控制云计算资源。对于用户来说，云资源是趋于无限大的，当然这得益于云计算服务采用的虚拟化技术。

一般来说云计算包括以下三个层次的服务：

[1] 基础设施即服务（IaaS）
[2] 平台即服务（PaaS）
[3] 软件即服务（SaaS）

1.3.1 云计算的优势

正如上一节所提到的，云计算具有常规计算方式所不具备的优势。总体来

讲，云计算具有地理位置无关性、设备无关性、时效性、可靠性等特点，这使得它迅速建立来了一个全新的市场领域，新的服务模式。

位置无关性主要表现在：只要有网络的地方就可以计算。云计算本身是一种基于互联网的服务，在有网络访问的地放就可以访问相应的服务。设备无关性在于所有的设备上的应用都是基于同一种标准开发，无论客户端设备是什么，均能得到同等的服务。时效性一方面在于所有的服务都已经现有，而且随时可用；另一方面也在于云计算本身强大的计算能力，其性能瓶颈基本上在网络传输上而不是计算本身。云计算一般都会在一定范围内集群大量计算机，任何一部分的计算机故障都不会导致整个系统的崩溃，可靠性比小型计算机系统好得多。

1.3.2 云存储

云存储可以理解为云计算的一种特殊模式。当云计算系统运算和处理的核心是大量数据的存储和管理时，云计算系统中就需要配置大量的存储设备，那么云计算系统就转变成为一个云存储系统[3] ，所以云存储是一个以数据存储和管理为核心的云计算系统。现在有许多成熟的商业云存储服务，如 Box，Dropbox，LiveOffice 等，当然也有不少优秀的开源云存储平台，如 GlusterFS。

1.4 本文结构

绪论部分到此结束，本文后面将分章节具体介绍项目实施和实现的技术细节，其中：

第二章介绍项目实施方案；

第三章讲项目实现的一些细节，其中重点为 3.1.2 节及 3.1.3 节；

第四章为项目成果展示部分；

第五章是项目应用前景展望；

最后为参考文献与致谢。

第2章 项目实施方案

2.1 项目目标

本项目为外包项目，需求方为美国一家公司（以后简称公司）。该项目力求构建一个能将各类资讯（主要是文档）整合在一起的、方便销售人员查看与分享的信息平台。

2.1.1 基本目标

项目的基本需求是：用户能够访问云存储服务 Box 以提升存储 iPad 的存储能力；用户可以在本地建立文件夹以放置相关度较高的文档，并搜索文件名；用户可以将最常用的一个文档放到一个特别的地方以供快速查看；用户可以对这些文档做一些笔记、心得等，并提供搜索；用户可能从 RSS 中获取最新资讯，并可以查看详情；用户可以预览云存储文档，并下载云存储中的文档到本地；用户可以对下载好的本地文档做删除和通过 Email 分享操作；等。

基于以上需求，项目中产生如下名词：

- Fund。一个本地的文件夹，用于集中放置一些相关的文档。
- Notes。与 Fund 相关的笔记。
- News。主要是一些投资、销售方面的 RSS 订阅。

2.1.2 高级目标

在基本目标的基础上，需要实现如下高级版本的功能：

1. 应用可以多用户使用。使用前用户需要到公司官方网站上注册一个帐号，在应用启动时提供一个登录页面，使用注册好的信息登录。
2. 用户登录后，只能看到在这个 iPad 上自己的 Fund 及 Notes。
3. 用户可以同步自己的 Notes 到服务器，在另一台设备上登录后依然可以看到。
4. 用户可以绑定自己的社交网络帐号以分享 Notes。
5. 用户可以访问多个云存储服务，并对其进行上传、下载、重命名、删除等多种操作。
6. 用户可以向公司邮箱以特定的格式绑定到某个 Fund 发送邮件，然后登

录 iPad 应用后可以在对应的 Fund 下看到邮件内容及附件，并可以像云存储一样下载附件。

7. 用户可以定制自己的 RSS 链接。

2.2 使用的技术框架

在项目的实现过程中，考虑性能及实用性，大概会用到以下技术框架：

2.2.1 iOS

iOS 是苹果公司开发的一款移动设备的操作系统，原名为 iPhone OS，且只用于 iPhone 上，不过现在随着苹果产品的革新，iOS 已经用于苹果的多种设备上，甚至包括 Apple TV。iOS 基于 Darwin 开发，其系统核心架构分为四层[8] ：

- 核心操作系统层（the Core OS layer）
- 核心服务层（the Core Services layer）
- 媒体层（the Media layer）
- 可轻触层（the Cocoa Touch layer）

得益 iOS 精巧的设计，使得它可以高效率地运行在硬件配置并不高的设备上。它对触屏拥有原生的良好支持，让用户可以简单地通过手指来进行各种操作，且精度很高。它流畅的动画效果更是为 iPhone 等产品增色不少。

和 Mac OS 一样，iOS 的软件开发也必须在 Mac 系统中。苹果的 Xcode 是进行 iOS 开发的得力工具，用户只要安装完无需任何配置便可以马上进行开发与调试。目前 iOS 的最新版本为 5.1，Xcode 最新版本为 4.3。

2.2.2 JSON

JSON 是一种数据交换格式，它较 XML 具有数据量小得多的优点。简单地来说，JSON 用于对象——文本——对象的转换。JSON 的对象是一组无序的名称/值（key/value）对，这是许多编程语言中都有的概念，相当于字典的概念。数组在 JSON 是一种特殊的的对象，它包含一系列的有序的对象。JSON 用于数据交换主要优势有：数据量小，解析生成操作方便，便于人和机器的理解。

JSON 也是一种语言及客户端无关的技术，且其本身传输为文本，可以压缩以提高传输效率。所以在数据的传输上考虑使用 JSON。

2.2.3 RSS

RSS（Really Simple Syndication）英文原意是"聚合真的很简单"，它是一种消息格式，主要用在需要经常更新内容的网站上，它提供一种标准让网站可以将标题、时间、摘要、消息来源链接等信息自动发布，并使用户可以根据自己的喜好轻松地订阅这些消息。

RSS 消息格式基于 XML 标准，任何能够解析 XML 的程序可以对它进行解析，只不过它使用定制化后标签。

2.2.4 RESTFul

REST（Representational State Transfer）是一种软件架构风格，它从资源的角度来观察整个网络，分布在各处的资源由 URI 确定，而客户端的应用通过 URI 来获取资源的表征。RESTFul 则是遵循 REST 并基于 HTTP 的一种 WEB 服务。较于如 C#之类的 Web Service，RESTFul 具有代码无关性、客户端无关性等特点。因为它是基于 HTTP 的，任何能够发送 HTTP 请求的设备均可调用 RESTFul 服务。另外，采用 REST 的服务器端可以利用自身的缓存机制来加快响应速度、提高性能，也能提高可扩展性及兼容性等。

在本项目服务器的 API 设计中，可以考虑采用 REST 风格。

2.2.5 OAuth

随着互联网的发展，信息之间的相互依赖越来越强，各类互联网服务的整合已经成为一种趋势。与此同时，服务商也越来越注重用户信息的安全，如何既不让 A 服务知道用户在 B 服务上的帐号密码，又能使 A 安全地访问 B 服务的资源成为一个难题。OAuth（开放授权）的发展解决了服务整合过程中的验证与授权问题。

使用 OAuth 进行认证和授权的过程如下所示[9] ：

1. 客户端（应用程序或网站）想操作用户存放在服务提供方的资源。
2. 客户端请求一个临时令牌 (Request Token)。
3. 服务端验证身份后，返回临时令牌。
4. 客户端获得临时令牌后，将用户引导到授权页面请求用户授权。
5. 用户在服务方的网页上输入用户名和密码，然后授权该客户端访问所请求的资源。
6. 授权成功后，客户端以临时令牌为参数从服务端请求访问令牌 (Access Token)。

7. 服务端验证临时令牌的授权情况后返回访问令牌。
8. 客户端使用访问令牌访问用户受保护的资源。

如图 2-1 描述了整个流程。

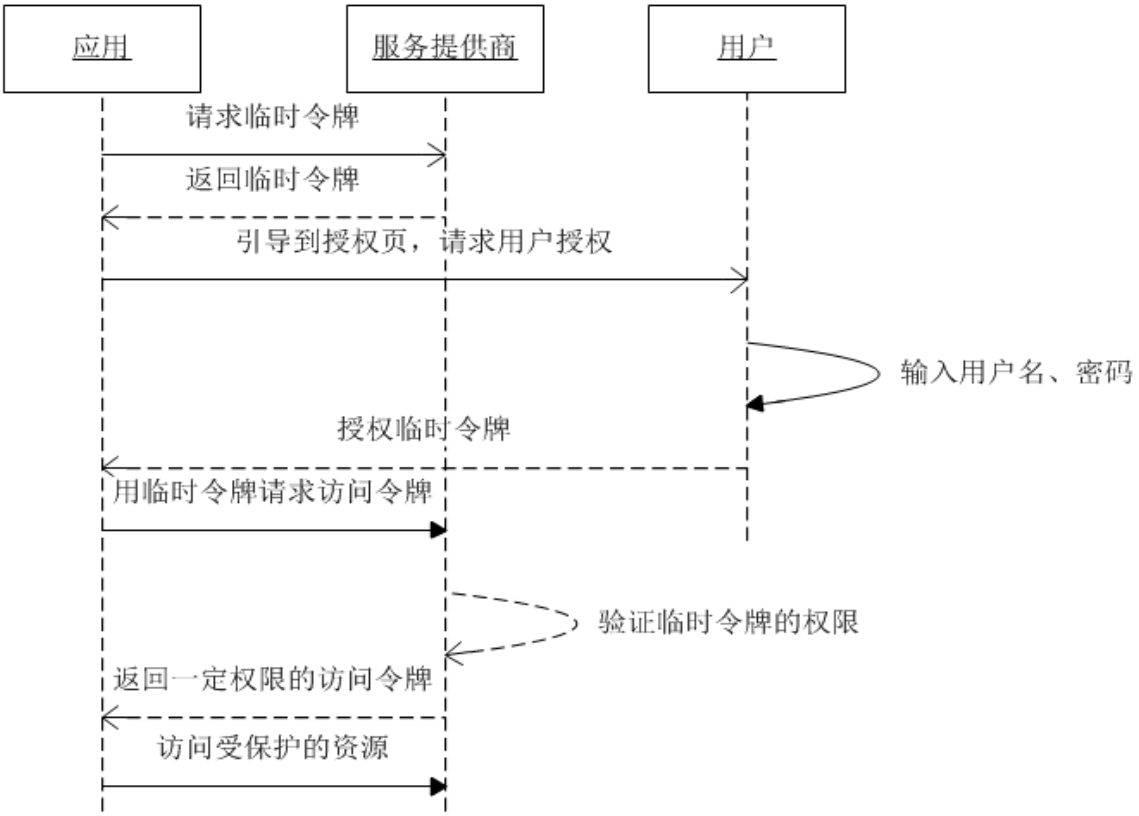

图 2-1 OAuth 认证流程

2.3 iPad 客户端设计

iPad 应用程序在用户交互界面上主要分为五块内容（如表 2-1），其中 Home，Favorite，News，Library 集成为一个主界面的四个子标签页，方便用户点击切换，Management 页面在用户点击按钮时才会出现。

表 2-1 主要交互界面说明

界面名称	界面说明
Home	这里是用户自己创建的 Fund 及与 Fund 相关的 Notes，Fund 中的文件分可读文档和多媒体文件分别放置，并显示文件的预览图，方便用户查找。用户在这个界面中查看下载到本地的文件，并可以在这个界面中删除文件，以及拖拽一个常用文件到 Favorite 中。
Favorite	这里是用户收藏的一个常用文件，可以是可读文档，也可以是多媒体文件。
News	这里是 RSS 订阅的内容，用户可以选择不同的 RSS 源，并可以查看详细内容。
Library	这里是云存储访问入口，可以浏览云服务中的文件，并可以点击预览。用户可以登录以授权应用访问云服务，并可以登出。
Management	这是用户创建和删除 Fund 的地方，并可以通过拖拽方式下载云存储中的文件到 Fund，

功能上，iPad 客户端应用主要分为以下模块。

2.3.1 预览图生成模块

在 Home 界面中需要显示所有文档预览图列表。应用需要支持的可阅读文档格式包括：PPT，DOC，XLS，PDF，多媒体文档包括：MP4，MOV，MP3。iOS 对 PDF 文档格式有良好的底层支持，可以方便简单地获取到预览图，而对 MS Office 系统文档支持有限，需要自己实现。另外，由于文件量可能很大，生成的过程应该是异步的，生成好后回调显示预览的方法，并可以并发操作。

2.3.2 RSS 读取模块

除了文档之外，另一个重要的信息来源就是 RSS 了。一方面 RSS 来源广，另一方面更新及时。

因为 RSS 消息格式是基于 XML 的，对它的解析本身不是问题，但于 RSS 格式其内部又分为好几种，每种都使用了不同的标签，所以处理起来还是稍微有点难度的。考虑到 RSS 已经是一种成熟的标准，必定有不少开源解析器，所以这里使用基于 Objective-C 的 MWFeedParser[14] 来读取解析 RSS 链接。

2.3.3 云存储访问与多线程下载模块

在本项目中，iPad 客户端需要对云存储服务平台 Box 进行一些操作。由于 Box 平台的文件访问是通过文件 ID（文件夹也是文件）来进行的，而文件 ID 并不能直观地反映出文件的层级关系，并且高级目标中还要支持其他云平台，而

它们的 ID 等又可能与 Box 不同。所以在设计上需要有所讲究，既要能方便访问，又要能后台无关。

2.3.4 文档操作模块

文档操作主要是文档的显示、查看和删除。文档显示常规上都是按网格（Grid）对齐显示，这里采用开源的 Grid 控件 GMGridView[15] 加快开发过程。同时，如果是正在下载的文档，这里会显示下载进度条。用户通过点击以全屏查看文档，并可以通过发送邮件将文档共享出去。正在下载的文档不可查看。在文档显示视图上，用户拖拽文档时，会出现一个垃圾箱，当用户继续拖拽文档到垃圾箱中去时，文档被删除。正在下载的文档不可删除。

这里操作对象是统一的，不同的文档类型及网格大小不同。所以对于统一的操作可以定义一个基类，不同的文档类型在子类中区分开来。

2.3.5 技术难点

客户端实现的技术难点主要是多种格式的文件预览图的生成、文件的多线程下载及进度条显示、后台（云存储）无关的代码设计。

2.3.5.1 预览图生成

如 2.3.1 中所提到的，iOS 对于 MS Office 系列文档支持并不太好，不能在底层代码上绘制出预览图。然而 MS Office 文档可以在快速查看控制器（QLViewController）及网页视图（UIWebView）中打开，这是唯一可行的两个突破口。对于 PDF 文档可以用 iOS 的 CG 函数直接绘制，且速度很快。

iOS 对 MP4 和 MOV 等 MPEG-4 编码的视频也有较好的支持，可以获取视频中的一帧作为预览图，这里不多阐述。

2.3.5.2 多线程下载及进度条显示

进度条显示本身不是难题，问题就在 iOS 系统中内存有限，经常会有视图的重用，即滑出可见区域的视图将被移除并作为正要滑入可见区域视图，以节约内存。如果简单地将一个指针指向一个正在下载的视图的进度条，那么当这个视图被重用时就会出现进度条错误，因为指针是不可能知道被指的对象是不是被重用了。本项目中可以使用消息机制解决这个问题。

2.3.5.3 后台无关的设计

这个的难点在于，对多种不同的云存储能实现统一形式的访问，且都能支持进度条的更新。项目中采用了多种设计模式以适应这种需求。

2.4 服务端设计

2.4.1 数据库设计

在高级目标下，考虑到用户可以登录、同步 Notes、下载 Email 等，服务器上应有以下数据表。如图 2-2。

- Users。此表是用户的一些基本信息。但实际上，公司的网站已经有注册、登录功能，所以可以复用那张表。
- Funds。这是用户创建的 Fund，它包含一个指明所属关系的外键 OwnerID。
- Notes。用户在某个 Fund 下创建的 Notes，它绑定在这个 FundID 下。
- Email。
- Attachment。

项目的 Email 功能要求用户可以用自己的注册邮箱向公司一个特定的邮箱以特定的规则发邮件，然后服务器上的守护进程会监视并处理这些邮件，将其对应到用户指定的 Fund 下，并保存其附件。所以这里需要两张表，一张保存邮件本身，一张保存附件信息。

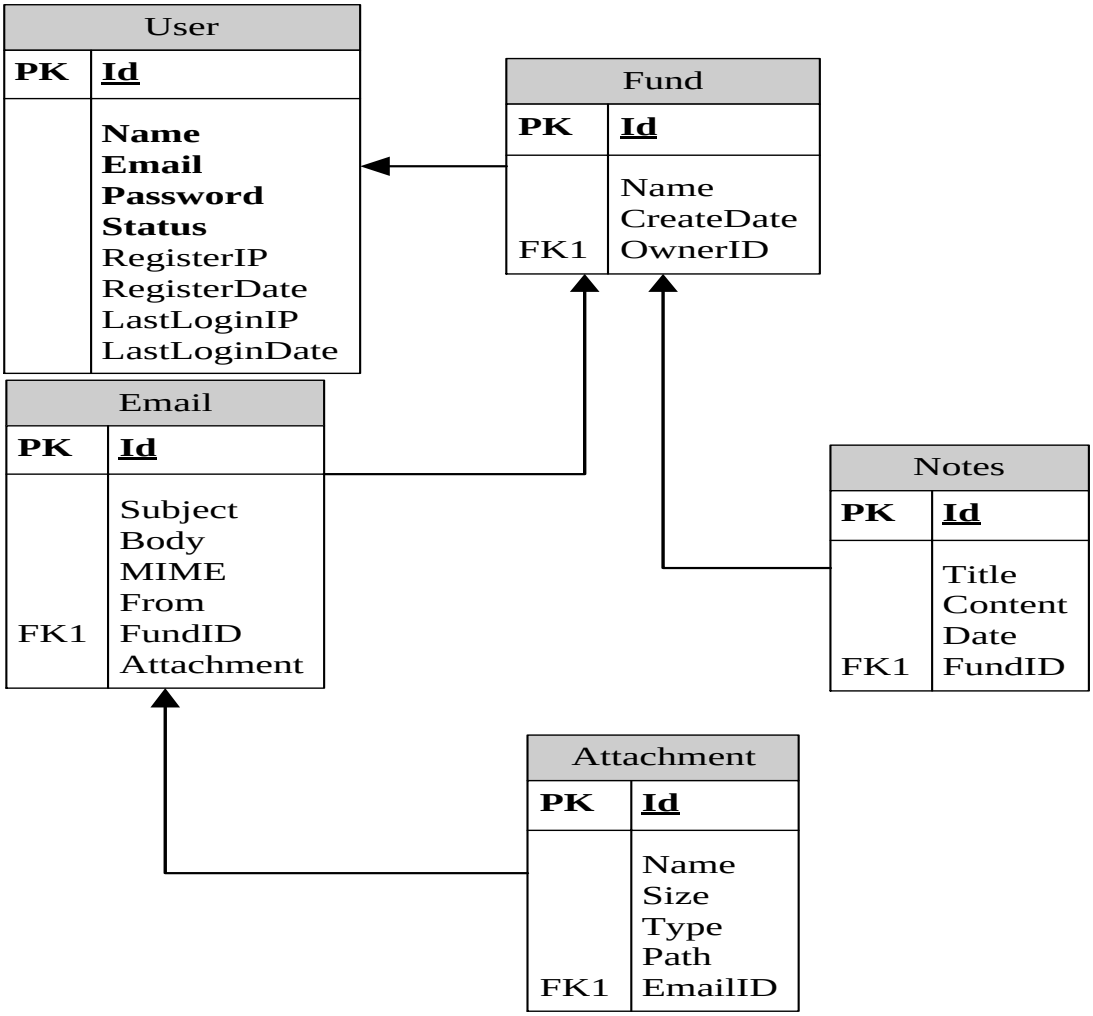

图 2-2 数据库 E-R 图

2.4.2 接口设计

服务器需要完成以下工作：邮件的抓取及存储，附件的抓取及存储，用户登录验证，Notes 的同步与更新等。其中用户验证是比较难的一部分，现有的基于 cookie 的登录机制仅适用于浏览器，而 OAuth 对于这个小系统又太复杂。为了简化操作，基于 OAuth 的原理及 RESTFul 的思想，设计自己的一套验证机制。

术语定义：

- Token。服务端返回的一段随机字符串，用于客户端验证。
- Expire。Token 的过期时间。
- Sign。客户端对服务端的操作及操作参数，和 Token 值一起通过一定的运算得到的一个 HASH 值，作为服务端的权限验证。

2.4.3 Email 处理模块

在高级目标中，用户可以在邮件主题前用“[]”将邮件要放置的文件夹名指定好，然后像发送普通邮件一样发送到公司的特定邮箱。服务端需要运行一个守护进程，它监视公司邮箱的变化，一旦收到新的邮件，它通过发件人找到用户 ID，通过邮件主题找到文件夹名，结合用户 ID 就可以找到唯一的一个文件夹 ID，然后在 Email 及 Attachment 表中插入数据。

2.5 本章小结

本章介绍了项目的目标及项目实现前期的一些技术框架和方案，并分别描述了客户端及服务端的一些设计细节，为增强了项目成功的信心，为后面项目的具体实现打下了基础。

第3章 项目实现

3.1 客户端实现

客户端功能上的细节较多，有些逻辑判断也比较复杂，下面就几个主要功能简单说明实现过程。

3.1.1 预览图生成与缓存

生成 PDF 文件的预览图可以用 CGPDFDocumentCreateWithURL()函数来生成，操作简单。而对于 MS Office 系列文件，可以将文件在一个 UIWebView 中打开，成功打开后，让 UIWebView 绘制在自己创建的 Context 并然后生成图像。这里需要引入 iOS 的类库 QuarzCore.framework，主要用到视图图层 CALayer 的 - (void)renderInContext:方法。具体实现见代码片断 1。

预览图的生成是一种批量操作，如当用户切换到新的 Fund 时，需要一次产生此 Fund 下所有文件的预览图。同时它必须是异步操作，否则在预览图生成好之前程序将被阻塞，用户无法进行其它操作，会造成不好的用户体验。基于上面的考虑，于是将生成预览图的操作封装为一个 NSOperation 的子类 HTThumbnailGenerator，并覆盖其- (void)main 方法。

main 方法中主要做两件事情（图 3-1）：

[1] 判断要生成预览图的文件是否已经有缓存，有则直接返回缓存；

[2] 调用对应的生成方法生成不同文件格式的预览图。

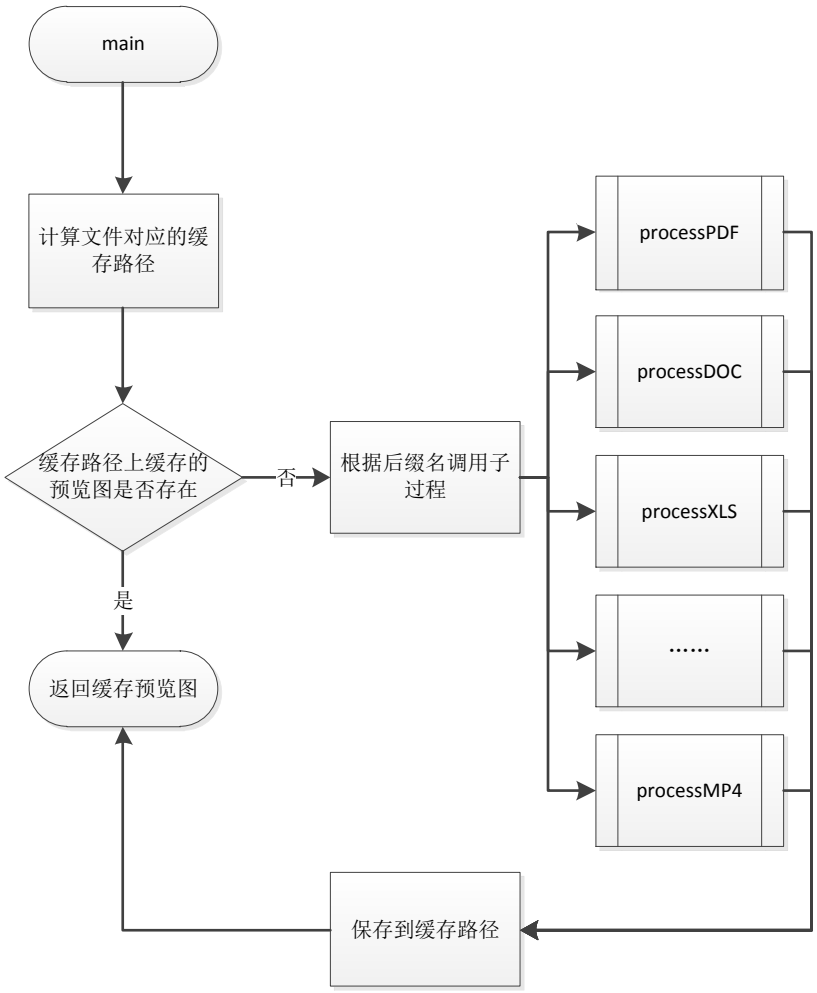

图 3-1 预览图生成流程

上图中计算文件对应的缓存路径过程如下：

$$缓存路径 = 临时目录 + \mathrm{MD5}\left(文件完整路径 + \mathrm{ToString}(预览图大小)\right) + ".\mathrm{png}"$$

由于同一个文件可能生成不同大小的预览图（Favorite 中就是个大的预览图），所以上式中加入了预览图大小作为参数之一。

项目实际开发过程中，加入缓存机制后可以发现，预览图生成效率高了很多，第一次生成后，后面再次打开同一个 Fund 预览图瞬间就加载完毕。

生成预览图的操作都加入操作队列以实现并行生成，生成好的预览图将用 iOS 的 block 机制进行回调，如图 3-2。

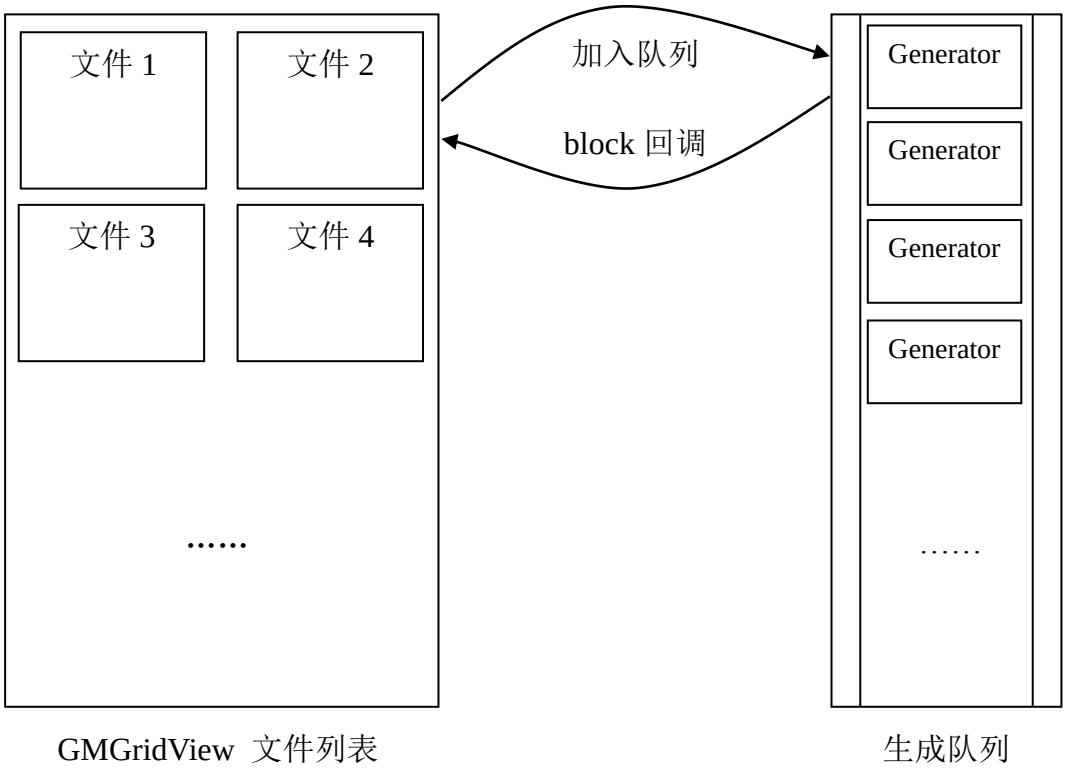

图 3-2 生成与回调

3.1.2 桥接、类工厂与单例模式

云存储访问的实现是项目的关键，因为透过云存储以扩展 iPad 无限的存储能力是本项目的要点。考虑到需要对多个云存储服务提供支持，在代码设计上就需要一些巧妙的模式。

由于在高级目标中要对多个云存储作底层支持，那么就要求代码实现对上提供统一接口，对下提供不同的实现。即抽象的接口与它的具体实现是分离的。当然这里可以想到用继承的方式来实现接口。但是有时候继承方式不太灵活，原因在于，它将抽象部分与实现部分绑定在一起，难以对两部分进行独立的修改及扩充。

我们不希望下载的接口操作与它的实现在有固定的依赖关系，因为在运行过程中，用户可以切换不同的云存储服务。同时，当具体实现修改时不应当影响接口调用中已经写好的代码。另一方面，需要对上彻底隐藏云存储服务访问的具体实现。此需求下，桥接模式是一种很好的选择。

桥接模式要求有一个 Abstraction 提供操作接口及多种不同的 Implementor，

所有的 Implementor 都是继承自同一基类，然后在 Abstraction 类是维持着一个基类的指针，并将所有的接口操作请求转发给 Implementor 的具体实现。

本项目的特例中，扮演 Abstraction 类的角色的是 HTFileDownloader 类，它提供了一套文件操作的接口，并且它将这些操作请求转发给它的一个成员变量 id<HTFileDownloaderDelegate> downloaderDelegate，这里 HTFileDownloaderDelegate 就是 Implementor 的基类，它是 Objective-C 语言中的一个协议，类似于 C++及 Java 中的抽象类 Abstract Class。如代码片断 2，HTFileDownloaderDelegate 中定义了项目可能用到的所有文件操作接口，当然实际上用不到全部，具体的云存储只要实现部分接口即可工作。每个具体的云存储访问的实现类，如 Box，Dropbox 等都服从这一协议。HTFileDownloader 类在初始化时可以选择一个默认的实现，并且在程序运行过程中修改实现！

所有的云存储的实现在项目中都被看作是一个 Service，如果将 Service 看成一种产品，那么可以定义一个工厂类 HTFileServiceFactory，可以调用工厂的类方法创建一个具体的产品 Service，它返回 id<HTFileDownloaderDelegate>，即 Implementor。本项目中的工厂类是一种退化了的形式，因为只有实际上一个工厂，没有工厂接口及子类，这里仅仅是为了创建的方便。

程序运行过程中的所有下载及其云服务的操作需要统一进行管理，即在全局中 HTFileDownloader 只有一个实例。于是可以将 HTFileDownloader 单独设计成单例（Singleton），以提供全局访问点，而这个就体现了桥接模式的优势，它的接口与实现可以单独变化，这是继承方式无法做到的。HTFileDownloader 的初始化代码如代码片断 8。

云存储访问模块设计的结构图如下所示：

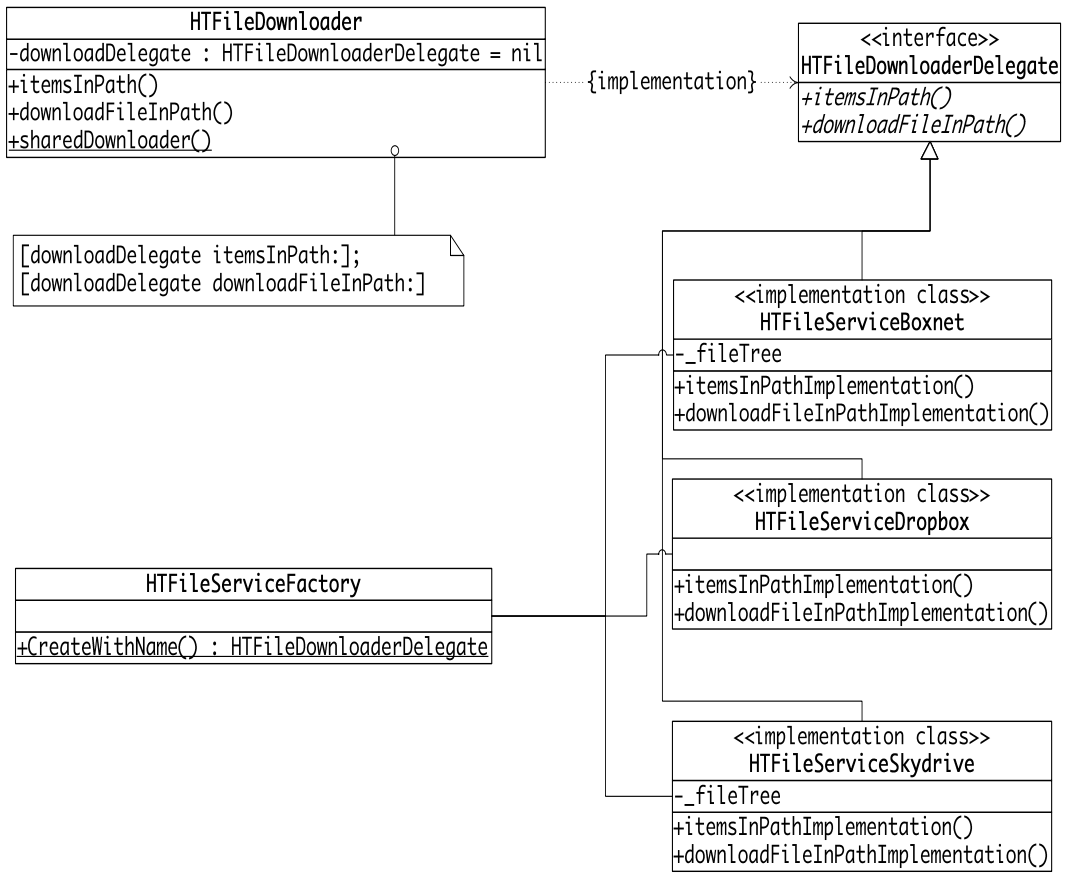

图 3-3 云存储模式类结构图

在这种设计下，既可以提供一个全局文件操作点，同时又可以切换不同的云存储，让程序的实现变得清晰、简单。

下面就 Box，Dropbox，SkyDrive 三种云服务来看看具体实现过程。

3.1.2.1 Box

Box 的 SDK 中文件类 BoxFile 与文件夹类 BoxFolder 都继承自 BoxObject，用一个整型 objectId 来区分不同的文件与文件夹，根文件夹的 objectId 为 0。Box 对文件的所用操作都基于文件 ID。

定义 HTFileServiceBoxnet 类，服从 HTFileDownloaderDelegate 协议，它内部维护一个变长的字典 NSMutableDictinory *_fileTree，用以支持统一的路径形式的访问，即，把路径形式的访问转换为 ID。这个字典主要结构如下：

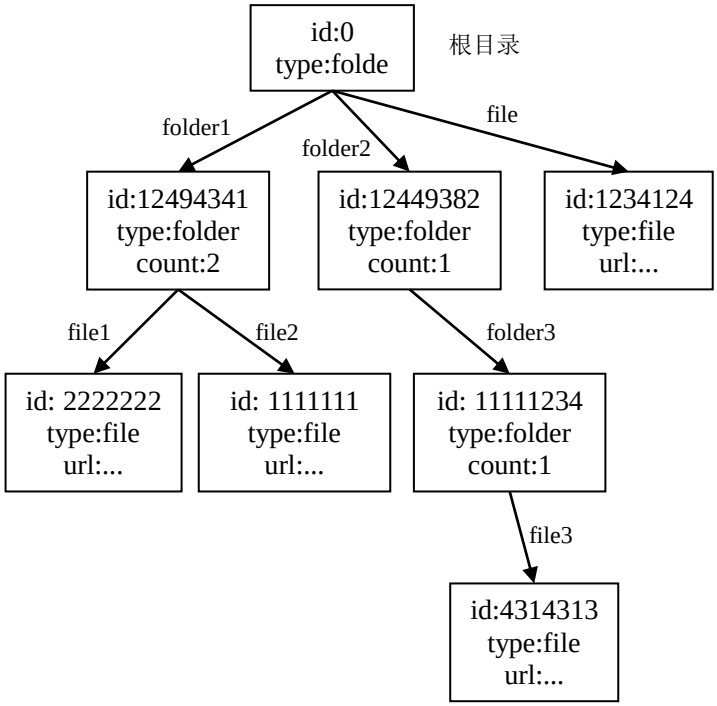

图 3-4 字典结构

在每个节点中 id 是个 NSNumber，它是文件的唯一标识符，type 是 NSString，它用以区分文件与文件夹。当节点是文件夹时，它会有一个 items 键，它的值又是变长字典 NSMutableDictionary（如代码片断 3）。当要访问 /folder2/folder3/file3 时，首先将访问的路径拆分为四个节点：“/”，“folder2”，“folder3”，“file3”。“/”节点就是_fileTree 本身，从它的 items 中找到“folder2”键对应的值，这样就进入“folder2”节点，然后再从当前节点的 items 中找到“folder3”键对应的值，于是就进入 folder3 节点，最后找到文件 file3 对应的 ID，就可以对其进行其他操作了。

HTFileServiceBoxnet 对象初始化时，只有根目录是已知的，其它节点将在用户的不断访问过程中构造并保存。初始化在代码片断 4。

文档树的访问主要方法有（代码片断 5）：

- - (NSDictionary*)getNodeinPath:(NSString*)path
- - (NSNumber*)getFileIDinPath:(NSString*)path
- - (NSMutableDictionary*)buildFileMapTree:(NSNumber*)folderID

在上面 getNodeinPath:方法中，如果 items 为 nil，则需要构造目录的子节点，

将文件夹的 ID 作参数传入 buildFileMapTree:方法即返回文件夹中的所有子节点。

在 Box 的 SDK 中，比较关键的类为 BoxNetworkOperation，它是所有操作的基类，也是 NSOperation 的子类，所以可以加到 NSOperationQueue 中以实现异步操作，但它本身是同步操作。BoxDownloadOperation 是 BoxNetworkOperation 的子类，所以 Box 实现的-(void) downloadFile:toPath:withProgressHandler:方法为同步的，这会影响到后续云存储访问代码的实现。下载实现的代码片断如代码片断 6。

3.1.2.2 Dropbox

Dropbox 文件访问的后台实现是跟本地文件系统一样，使用的是路径。所以它的代码实现要简单得多。

跟 Box 类似，定义 HTFileServiceDropbox 类作为 Implementro，服从 HTFileDownloaderDelegate，然后实现接口方法。Dropbox 的 SDK 中关键类是 DBSession 和 DBRestClient，前者用于授权及保存授权信息，后者用于实际 API 调用。

需要注意的是，Box 中的下载操作是同步，而 Dropbox 是异步的，为尽量减少原有代码改动，将 Dropbox 也实现为同步方式。实现方法为将 HTFileServiceDropbox 继承自 NSOperation，并在异步操作后进入消息循环，直到异步操作完成才退出。现实见代码片断 7，其中关键为 CFRunLoopRun()和 CFRundLoopStop()函数的使用。

3.1.2.3 SkyDrive

同 Box 一样，SkyDrive 的文件操作也是基于 ID 的，不同的是 SkyDrive 的文件 ID 是字符串，而且更为复杂。它的根目录为/me/skydrive，代码实现上有着与 Box 差不多的思路，同样需要维护一个_fileTree，也需要在第一次时构造节点并保存。同时 SkyDrive 的下载也是异步的，需要像 Dropbox 一样将其转换为同步操作。所以可以认为 SkyDrive 的实现是 Box 与 Dropbox 实现的结合。具体实现不再过多阐述。

3.1.3 消息机制实现的进度条

在 iPad 这种小型移动设备上，内存是种珍贵的资源，iOS 在底层上对内存管理进行了优化，极在限度上减少了内存占用率，同时它要求开发者们养成良

好习惯，不用的视图和缓存应当在内存不足时马上释放掉。在 2.3.4 中提到的优秀的开源控件 GMGridView 即采用了内存节约型的设计。当文档过多时，视图会横向形成滚动，而当滚动页较多时（超过 3 页），那么已经滚出可见区域较远的那部分 Grid 可以重用到即将滚入的 Grid 上。所以基本上只要 3 页的内存就可以显示任意多页。

为了显示进度条，Grid 中添加了 UIProgressView 子视图，当然将这个视图的指针传给下载的 delegate，当进度发生改变时更新进度条位置是可以的，问题就在于分页过多出现 Grid 重用就麻烦了。前面是在下载，但后面可能是已经下好的文件，当用户滚动到后面时，前面的 Grid 被重用，那么前面文件的进度条就显示到后面已经下好的文件上了。

这里 iOS 内部的消息机制就派上用场了。消息机制的优点是：消息发送者不需要知道谁在接收消息；消息接收者也不需要知道谁在发送消息。从而达到一定程度上的解耦合。无论是什么云存储，在下载文件过程中进度改变时发送以文件名命名的消息，同时每个 Grid 接收自己显示的文件名对应的消息，这样即使页数多，用户滚动时，被重用的 Grid 会因为监听了新的消息而不会被影响到。

定义 HTDownloadProgressDelegate 协议，它提供三个方法（见表 3-1），在不同的云存储访问实现中需要添加自己的类服从这个协议。主要目的是为了提供一个标准化的进度通知接口，因为不同的云存储可能有自己不同的实现方式。这样才能在下载时传入统一的参数，即 HTFileDownloadDelegate 中

```
- (void)downloadFile:(NSString *)path
              toPath:(NSString *)localPath
withProgressHandler:(id<HTDownloadProgressDelegate>)handler
```

方法的 handler 参数。

表 3-1 HTDownloadProgressDelegate 协议

方法名	说明
-(void)downloadFileStart:(NSString*)path;	文件开始下载通知，参数为文件本地路径
-(void)downloadFile:(NSString*)path progress:(float)ratio;	文件下载进度更新

方法名	说明
-(void)downloadFile:(NSString*)path complete:(BOOL)success;	文件下载中止，参数 success 说明是否完成下载

定义 HTDownloadOperation 继承自 NSOperation 类，并服从 HTDownloadProgressDelegate 协议，使得它既可以添加为多线程任务，又可以管理自己任务的下载进度（见代码片断 9）。

HTDownloadOperation 覆写 NSOperation 的 main 等方法，调用 HTFileDownloader 的方法下载文件。

```
- (void)main
{
    if (self.isCancelled)
       return;

   HTFileDownloader *fd = [HTFileDownloader sharedDownloader];
   [fd downloadFile:self.remotePath       // 同步方法
            toFolder:self.localPath
 withProgressHandler:self];
}
```

并在协议方法中发送消息（代码片断 11）。

注意，代码片断 11 发送消息的代码必须在主线程中执行，因为消息接收者（Grid）需要更新自己的视图，而屏幕画面的更新要在主线程中，不在主线程上发送这些消息会产生不可预料的后果！

定义 HTFileBrowserViewCell 类作为显示文件预览图及文件名的 Grid，以及 HTFileBrowserController 作为控制器。HTFileBrowserController 一次只显示一个本地文件夹，它接收以自己显示的文件夹路径为名称的所有消息。在控制器中维护一个变长字典 docCells，它保存文件名与实际显示的 Grid 的对应关系，控制器接收消息，并通过这个字典找到相应的 Grid 更新进度条（代码片断 10）。

整个流程可以用图 3-5 来表示：

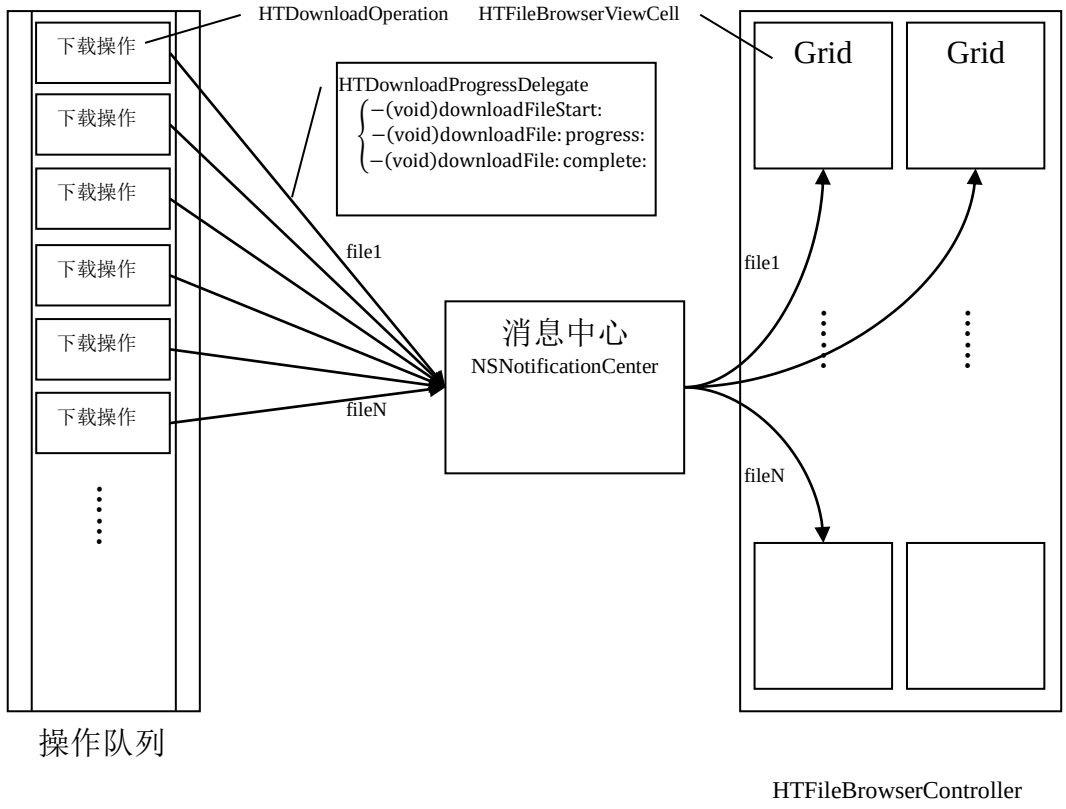

图 3-5 多线程下载及进度条显示实现机理文件显示控制器

3.1.4 消息机制的其他应用

消息机制在项目中还有其他一些应用，这得益于消息机制本身解耦合的特点。

3.1.4.1 文件删除

在 Favorite 界面中显示了用户常用的、感兴趣的文档，而事实上，用户可能在 Home 界面中将文档删除，或者在 Management 界面中将包含这个文档的 Fund 删除，这时候如果还显示着这个文档那就出错了，重则导致程序崩溃。然而由于是相对独立的界面，它们相互之间都不知道对方的存在，那如何通知 Favorite 界面文件已经删除了呢？消息机制是个很好的选择。

用户在 Home 下通过拖拽方式删除一个文件时，Home 就发送文件删除通知：

文件 1.doc 已经删除了。Favorite 则监听此消息，并匹配已经删除的文件与自己正在显示的文件的文件路径。当用户删除一个 Fund 时，Management 发送 Fund 删除通知：Fund1 已经删除。Favorite 同时也监听此消息，并匹配正在显示的文件所在的文件夹。匹配成功就将显示的预览图去掉。

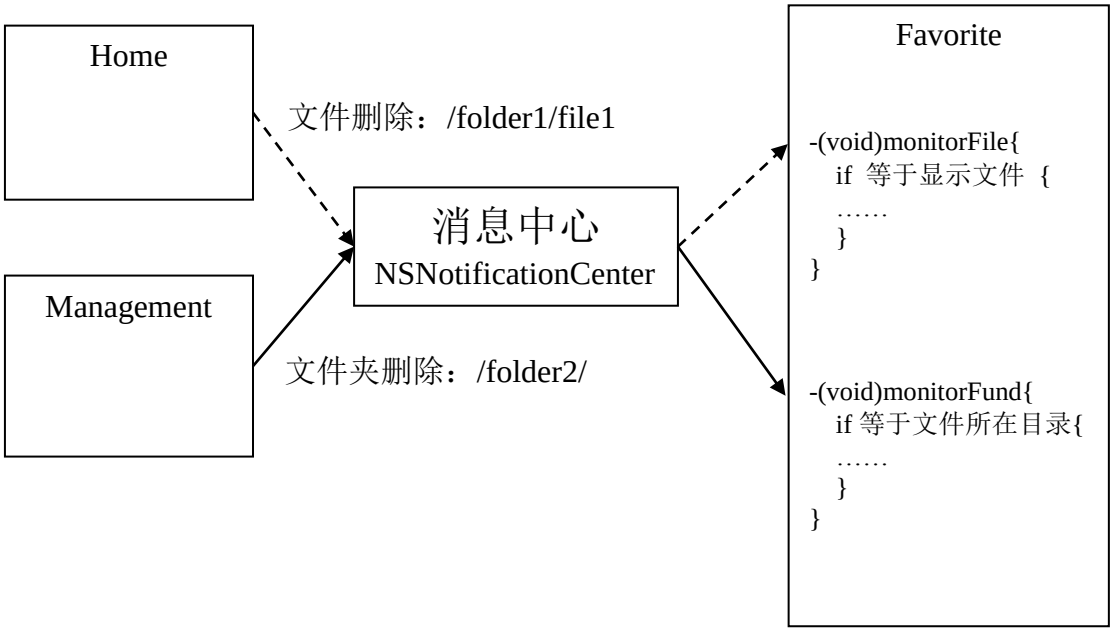

图 3-6 文件删除消息机制

3.1.4.2 文件拖拽

用户可以在 Home 中拖拽一文件到 Favorite 中作为常用文件方便查找，但 Favorite 是不知道 Home 界面的存在的，它只是监听着一个拖拽消息，消息体中包含文件的路径信息。同样，在 Home 界面中拖动一个文件，它就不断发送拖拽消息：某文件已经拖动到某位置了。Favorite 接收消息后将判断，文件是否已经在自己显示的区域，如果是，则将它的大预览图显示在界面中。

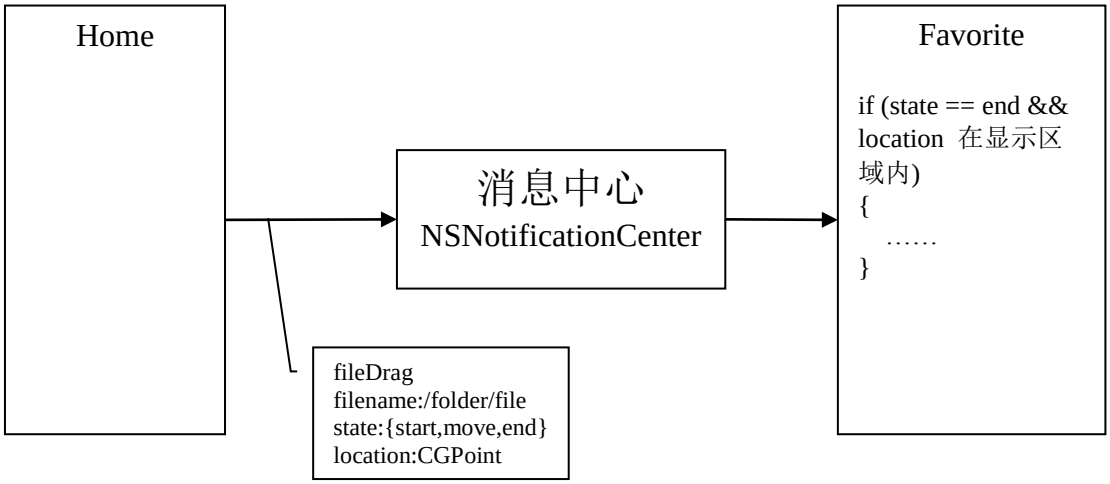

图 3-7 文件拖拽消息机制

3.2 服务端实现

3.2.1 登录接口

服务器对用户的登录包括用户名密码的校验其 Token 的生成过程。主要实现如下：

```
function login($username,$password) {
    $password = md5($password);
    $id = 查找用户 ID;
    if (匹对成功) {
        $token = 随机生成 Token;
        $expire = 当前时间加一个小时;
        保存$token，$expire 及对应的用户$id 以供后面验证;
        返回信息到客户端
    } else {
        返回错误信息到客户端;
    }
}
```

如表 3-2 客户端将用户填写的用词名、密码以 POST 方式发送到服务器，服务器验证通过后会返回一个 Token，并设置好它的 Expire。Token 会保存在服务器端的数据库中，并与当前用户关联，直到过期。客户端接收 Token 后可以将它保存到磁盘上，或内存中，以供后面的操作使用。

为了保证用户名、密码，及 Token 不会被人恶意抓取，整个过程以 HTTPS 方式连接。

表 3-2 登录接口设计

登入接口	https://hostname/login
方式	POST
参数说明	username 用户名；password 密码
返回值	输入正确： {"status":"OK","token":"…","expire":1337001234,"user_id":1234} 输入错误： {"status":"Error","description":"invalid username or password"}

3.2.2 同步接口

用户登录到系统后，首先客户端会同步本地 Fund 信息，然后当用户点开一个 Fund 时，开始同步当前 Fund 的 Notes 及邮件列表。同步操作采用 RESTFul 规范，同时会带上一个 Sign，作用服务端对此操作的权限认证。登录之后，所有的操作中都不会出现 Token，它是作为一种保密信息，只有客户端和服务端知道。Sign 其实隐含到 Token 的信息，但却不能还原出来，达到加强安全的目的。

3.2.2.1 同步 Fund

同步过程不仅要验证签名的正确性，还要验证操作的合法性，即 Fund 的 ID 是不是当前请求的用户的 Fund。基本过程如下：

```
// 同步 Fund
function sync_fund($userid,$sign) {
    $token = 查找用户的 Token($uesrid);
    $thesign = md5($token+参数);
    if ($thesign === $sign) {  // 验证通过
        $funds = 查找用户所有的 Fund($userid);
        返回数据到客户端;
    } else {
        返回未授权错误信息;
    }
}

// 删除 Fund
function del_fund($fundid,$userid,$sign) {
    验证$sign;
    $theuser = 查找 Fund 所属的用户($fundid);
```

```
    if ($theuser === $userid) {
        删除 Fund 记录;
        返回成功信息;
    } else {
        返回非法操作信息;
    }
}

function add_fund($fundname,$userid,$sign) {
......
}
```

表 3-3 同步 Fund 接口设计

Fund 接口	https://hostname/funds/<user_id>/<sign>
方式	GET
参数说明	user_id 用户 ID；sign 链接签名

返回值	输入正确：
	{
	"status":"OK",
	"funds":[
	{"id":1,"name":"fund1","date":1337001234},
	{"id":2,"name":"fund2","date":1337003333}
	]
	}
	输入错误：
	{"status":"Error","description":"invalid sign"}

当然另外还有一些添加、删除的接口，这里不进行详细列表。

3.2.2.2 同步 Notes

同步 Notes 与同步 Fund 的操作基本一样，同样要验证签名及所属关系，比 Fund 多了一层判断，避免误操作他人的数据。

```
// 同步 Notes
function sync_notes($fundid,$userid,$sign) {
    验证$sign;
    if ($fundid 属于 $userid) {
        返回此 Fund 下的 Notes;
    } else {
        返回非法操作信息;
    }
}

// 删除 Notes
function del_notes($noteid,$fundid,$userid,$sign) {
    验证$sign;
```

```
    if ($fundid 属于 $userid) {
        if ($noteid 属于 $fundid) {
            返回成功信息;
        }
    }
    返回非法操作;
}

......
```

表 3-4 同步 Notes 接口设计

Notes 接口	https://hostname/notes/<fund_id>/<user_id>/<sign>
方式	GET
参数说明	fund_id 文件夹 ID；user_id 用户 ID；sign 链接签名

返回值	输入正确：
	{
	"status":"OK",
	"notes":[
	{"id":1,"title":"note1","content":"test","date":1337001234},
	{"id":2,"title":"note2","content":"test","date":1337003333}
	]
	}
	输入错误：
	{"status":"Error","description":"invalid sign"}

3.2.3 Email 抓取

服务器端运行着的守护进程可以通过 POP3 查询邮件服务器的邮件变化，当收到新的邮件时便触发一个操作。如 2.4.3 节中提到的，用户需要按规定的格式设置邮件主题，即将邮件要放置的 Fund 名用“[]”括起来，如表 3-5 所示。

表 3-5 邮件发送格式

主题：	[Fund1] This is Email Subject
附件：	1.pdf; 2.doc;
正文：	This is Email Body, Test content.

大致处理过程见代码片断 12。

3.3 系统架构

项目完成后应当有以下架构：

项目基本目标不需要服务器支持，结构较为简单，大致可以描述如图 3-8：

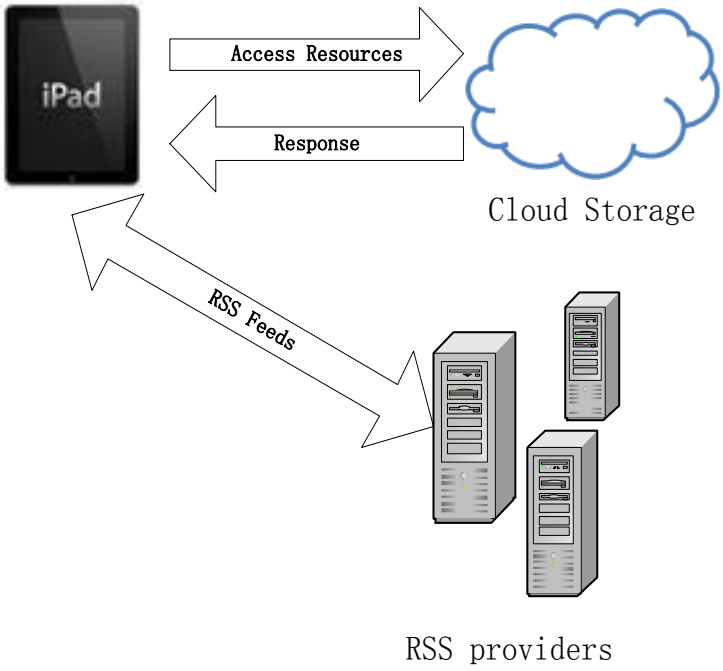

图 3-8 基本目标下的系统架构

高级目标加入了服务端的需求，所以在整体结构上多了一层，如图 3-9：

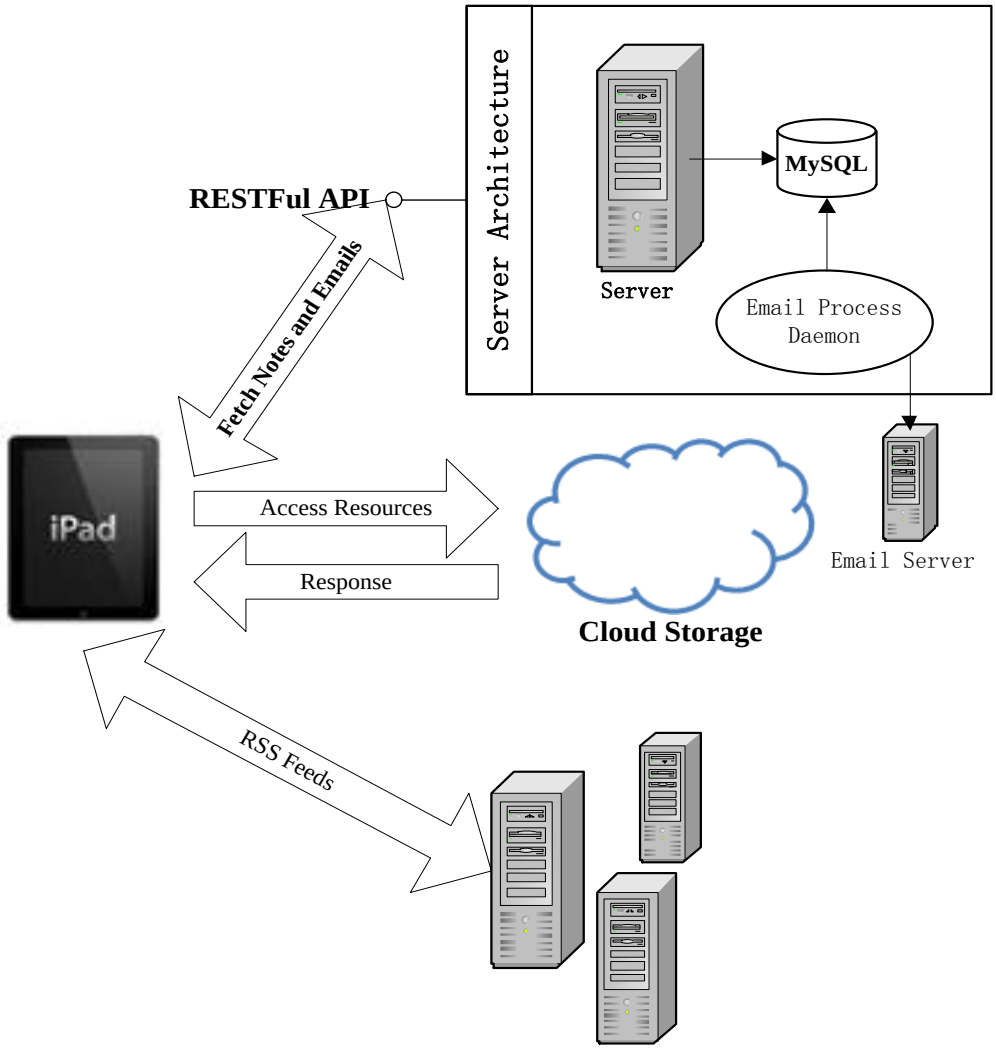

图 3-9 高级目标下的系统架构图

3.4 本章小结

本章以代码或伪代码的方式分别讲述了客户端与服务端一些主要功能的实现细节，如何攻克难点并最终成功实现的方法。

在客户端的设计上用到了委托、单例、桥接、观察者等多种设计模式，降低了代码的维护成本，并且更加可读、易于理解。其中预览图的生成中使用了磁盘缓存，加快了响应速度，比不加缓存拥有更好的使用体验。

服务端的代码实现较为常规，但考虑到其安全性，验证判断逻辑也有一定难度。

第4章 项目成果

项目成果分为两块内容，一个 iPad 应用程序，另一个是服务器代码。其中基本目标下的 iPad 应用已经上线 Apple Store，所有用户均可免费下载；服务器端还未完全实现。

4.1 iPad 应用

目前拥有基本目标下的功能的 iPad 应用已经开始在 Apple Store 上销售。这里分别看看它的五个主要交互界面并加以说明。

4.1.1 Home

如图 4-1，在 Home 标签下用户可以看到本地所有的 Fund。点击进入一个 Fund，左边以列表形式列出与当前 Fund 相关的 Notes，右边是已经下载的文件。

图 4-1 Home 界面的所有 Fund 视图

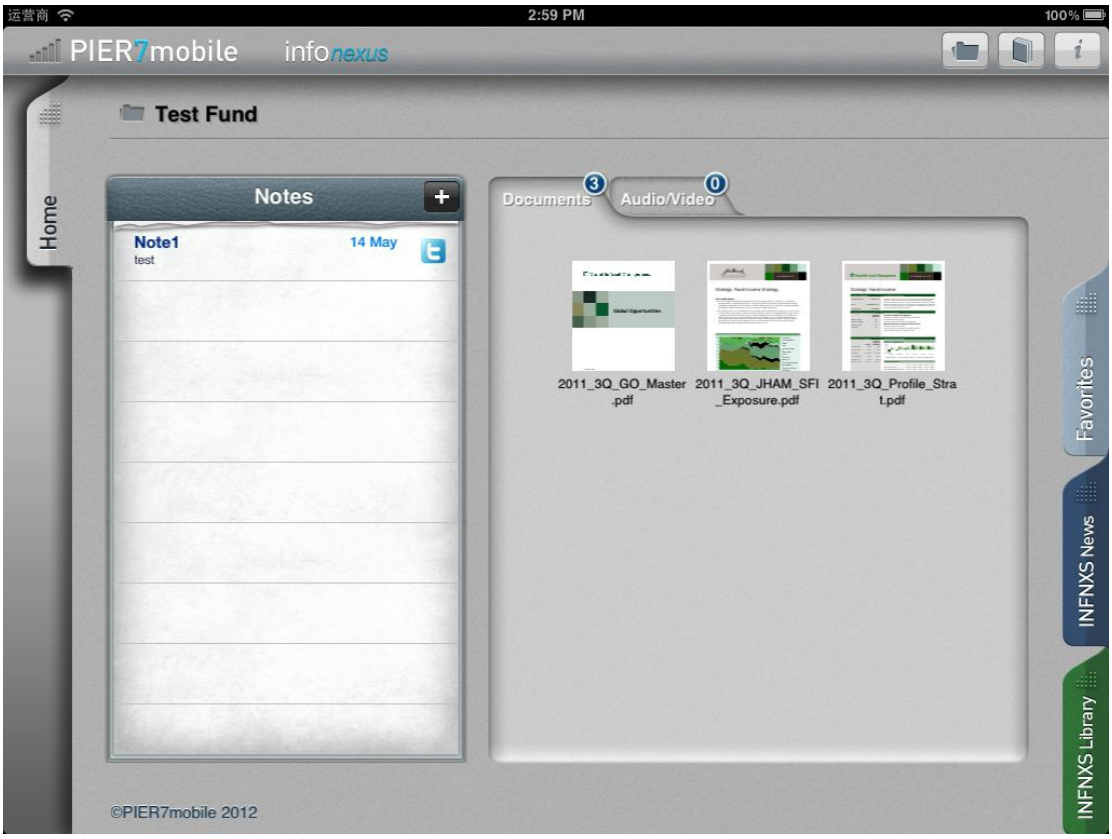

图 4-2 Home 界面的单个 Fund 视图

在左边用户可以点击“+”新建 Notes，点击某个 Notes 可以查看详情，并可以编辑。右边点击文档可以全屏查看，长按拖拽可以进行删除和放入 Favorite。

4.1.2 Favorite

此界面功能较为单一，只显示一个用户拖拽到这里的常用文件，双击可以打开查看。如图 4-3。

4.1.3 News

News 内容全部来自制定化的 RSS Feeds，用户可以选择不同的 RSS 源以查看感兴趣的信息。

图 4-4 中界面分为两列，右边为 RSS 内容摘要，点击一个后，左边显示内容的来源网页。点击按钮可以在新的界面中全屏查看该网页。

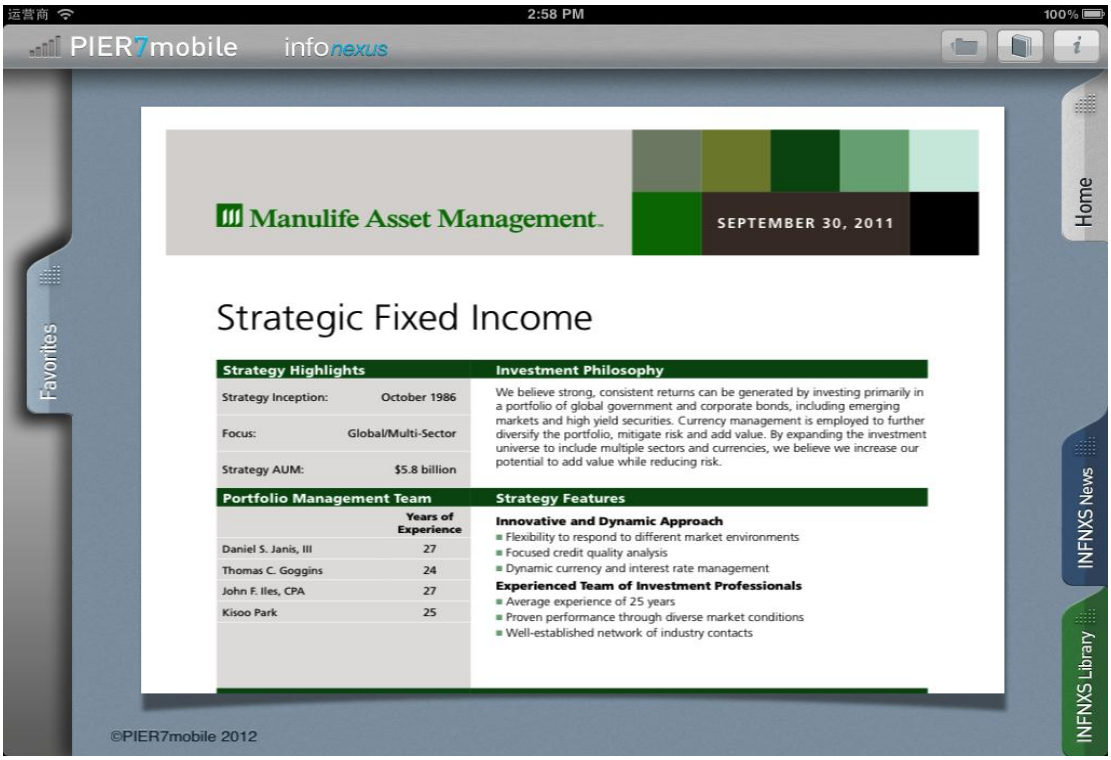

图 4-3 Favorite 界面

图 4-4 News 界面

4.1.4 Library

这里是云存储中的文件列表，用户首先需要登录才能看到。在这个版本中只支持 Box 云服务。

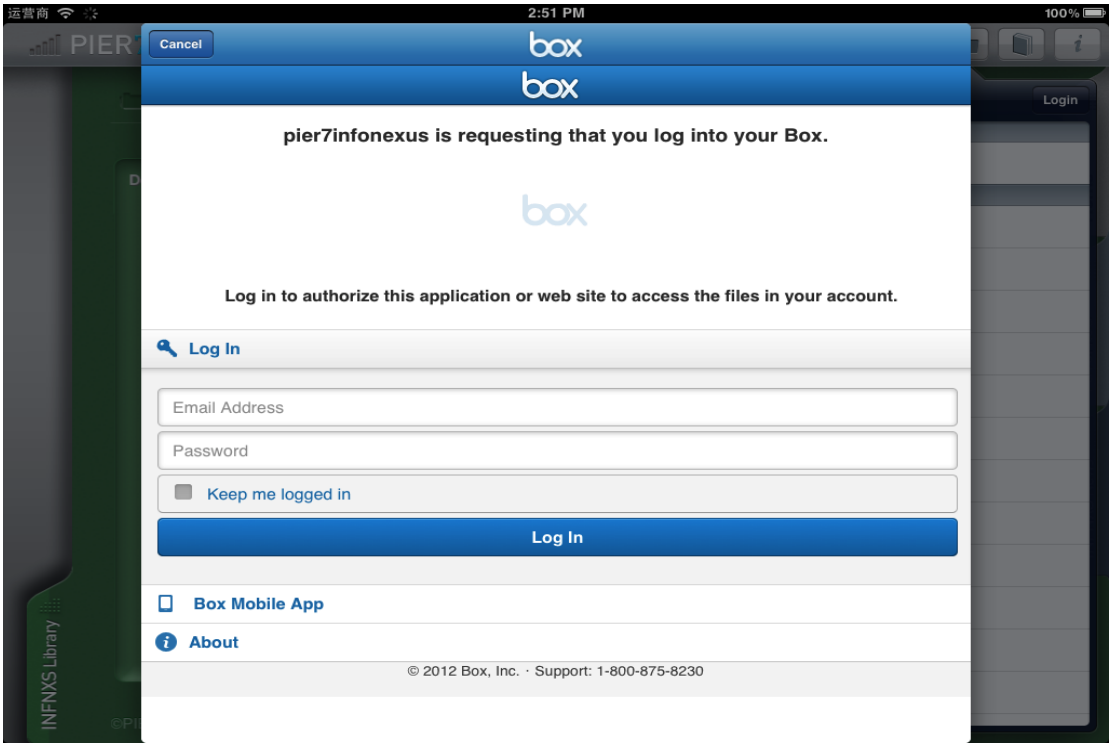

图 4-5 Box 的登录界面

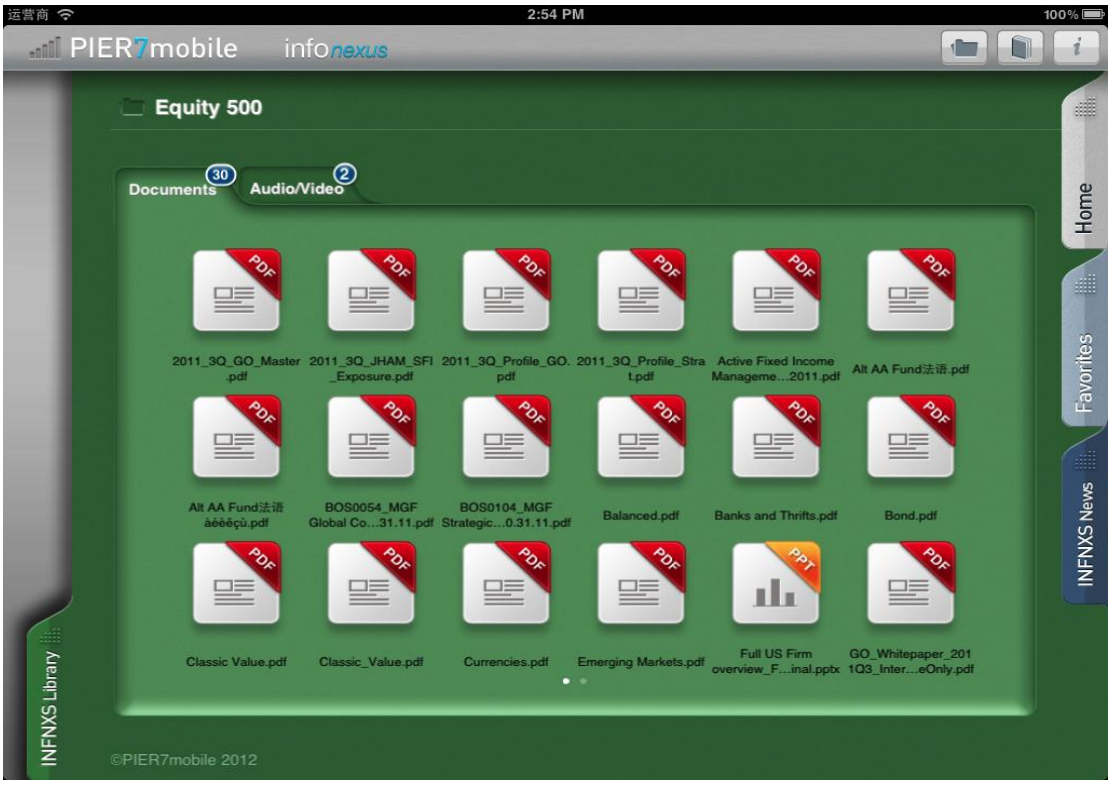

图 4-6 登录后选择 Equity 500 文件夹的视图

4.1.5 Management

在这个界面中，用户可以以简单的拖拽方式下载云存储的文件到本地（图 4-7），前提是在 Library 中登录。

拖拽完成后返回到 Home 下可以看到正在下载的文件及进度（图 4-8）。

图 4-7 文件下载

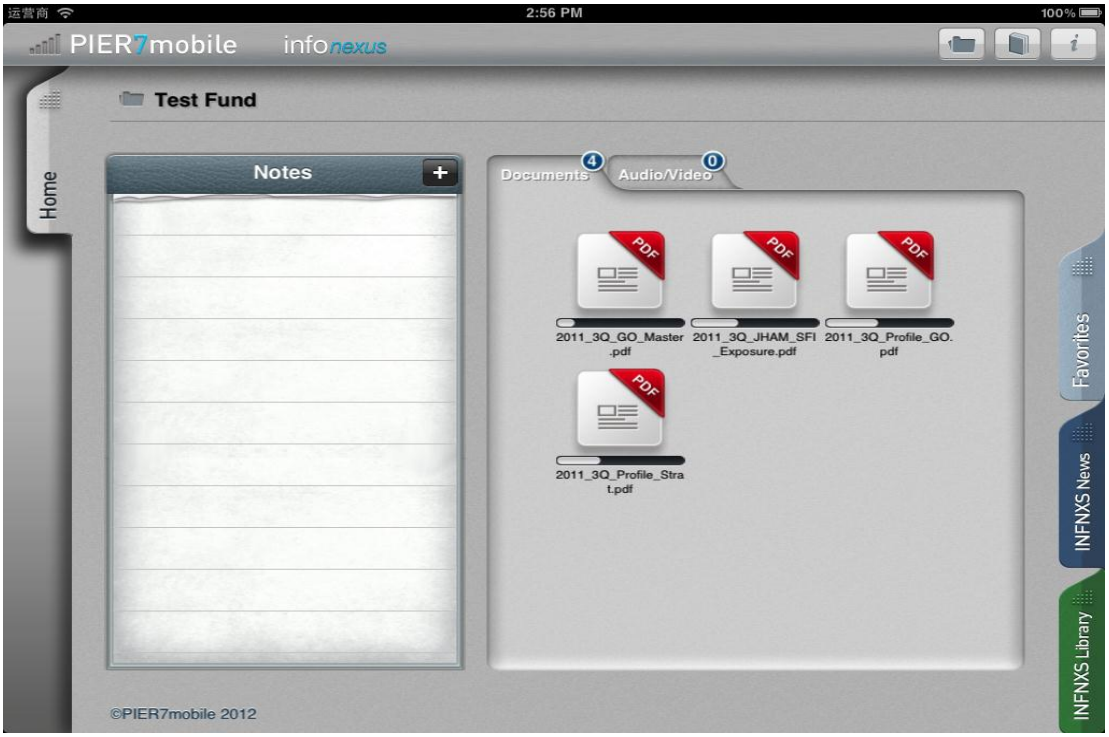

图 4-8 文件下载进度

4.2 服务端代码

由于项目进度原因，在没有支持服务端的情况下发布了 iPad 应用的基本目标版。目前服务端只实现了 Email 的抓取，其他接口还在进一步的优化设计和调试中。

4.3 本章小结

本章以应用截图的方式讲解了 iPad 应用的几个主要功能界面，限于篇幅没有作详尽的功能说明，更多内容请参考应用的使用说明书。由于服务端实现没有完全实现，目前没有可展示的成果。

第5章 总结与展望

5.1 项目总结

在商务越来越智能化、信息越来越多元化的今天，传统 PC 上商务活动转向平板电脑是一种趋势，本项目正是抓住了这样一种机遇，为销售、投资人员提供一个信息集成化的系统，以简单、高效地处理业务。

项目前期需求并没有定下来，并且在中期做了较大幅度的调整、修改，这在一度程度了减缓了项目的进度。同时，由于项目过程中出现了人员的调动，在很长时间内基本上由我一个人进行着实际的开发工作。不过令人欣慰的是，最后还是发布出来一个可用的版本，这也是对自己努力的一种回报和能力的证明。

项目中应用了多种设计模式，提高了代码编写的效率及可读性和可扩展性，并多处使用了消息机制使得类与类之间解耦合。为了加快响应速度，在预览图生成模块中加入了缓存，比项目最初没加缓存时要流畅得多。同时，客户端也在几处使用了 iOS 自带的线程池技术 NSOperationQueue，方便了线程的控制与使用，也可以有效防止死锁，提升运行效率。

5.2 项目展望

本项目拥有良好的应用前景，移动终端集成化的资讯，加上去存储近乎无限的存储能力，将极大方便使用者的信息查找。用简单的移动内容管理来连接整个世界，并与全球的销售与市场人员沟通、交流。它集文档管理、信息共享、云存储访问等于一身，功能较为完善。

项目后期将继续添加新的功能，配合服务端的支持，将越来越突出其信息整合的优势。相信随着平板设备进一步的普及和项目后期的顺利进展，本系统将发挥巨大的商业潜力。

参考文献

[1] 纵观平板电脑发展史. 中国电脑教育报. 2010-11-15, 第 A03 版.

[2] 云计算. 维基百科. http://zh.wikipedia.org/wiki/%E9%9B%B2%E7%AB%AF%E9%81%8B%E7%AE%97.

[3] 云存储. 百度百科. http://baike.baidu.com/view/2044736.htm.

[4] Irfan Gul, Atiq ur Rehman, M Hasan Islam. Cloud Computing Security Auditing. Next Generation Information Technology (ICNIT), 2011 the 2nd International Conference, 2011.

[5] Chen Danwei, Huang Xiuli, and Ren Xunyi. Access Control of Cloud Service Based on UCON. Proceedings of the 1st International Conference on Cloud Computing, Nov. 2009.

[6] Ramgovind S, Eloff MM, Smith E. The Management of Security in Cloud Computing. Information Security for South Africa (ISSA), 2010.

[7] Shahryar Shafique Qureshi, Toufeeq Ahmad, Khalid Rafique, Shuja-ul-islam. MOBILE CLOUD COMPUTING AS FUTURE FOR MOBILE APPLICATIONS – IMPLEMENTATION METHODS AND CHALLENGING ISSUES. Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference.

[8] iOS. 维基百科. http://zh.wikipedia.org/wiki/IOS.

[9] OAuth. 维基百科. http://zh.wikipedia.org/wiki/OAuth.

[10] Parekh Tanvi, Gawshinde Sonal, Sharma Mayank Kumar. Token Based Authentication using Mobile Phone. 2011 International Conference on Communication Systems and Network Technologies.

[11] Carlo Chung. Objective-C 编程之道：iOS 设计模式解析. 北京：人民邮电出版社, 2011.

[12] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. 设计模式——可复用面向对象软件的基础. 北京：机械工业出版社. 2000.

[13] 2012 年 3 月中国区 APP Store 数据监测报告. http://www.ruanlie.com.

[14] RSS 解析库 MWFeedParser. https://github.com/mwaterfall/MWFeedParser

[15] 开源网格视图 GMGridView. https://github.com/gmoledina/GMGridView

[16] Dave Mark, Jeff LaMarche. iPhone3 开发基础教程. 北京：人民邮电出版社, 2009.11.

致谢

本文的完成离不开浙江大学提供优质的教学资源，项目的成功离不开网新恒天有限公司提供的良好工作环境及各种软硬件资源。

在即将毕业之际，首先感谢父母二十多年来的养育之恩，让我衣食无忧，快乐成长，您们的付出不久定会有回报；

感谢浙江大学四年的培养，这四年会是我人生中重要的四年；

感谢导师蔡亮的悉心指导，您在学术上的严格要求让我印象深刻，在以后的学术活动中也将以此为要求约束自己；

感谢网新恒天 AM 高波、PM 胡坚，您们的信任与支持让我能够发掘出自己的潜力并有平台去展示。

最后感谢所有关心与帮助过我的同事与同学们，你们让我的生活充满趣味且多姿多彩。

附录

代码片断 1 用 UIWebView 生成预览图

```
- (void)loadWebView
{
    UIWebView *webView = ...;
    webView.delegate = self;
    [webView loadRequest:...]; // 加载文件
}

- (void)webViewDidFinishLoad:(UIWebView *)webView  // 成功加载
{
    ......

   UIGraphicsBeginImageContext(预览图大小);      // 创建自己的 Context
   CGContextRef context = UIGraphicsGetCurrentContext();
   [webView.layer renderInContext:context];    // 绘制到这个 Context
   预览图 = UIGraphicsGetImageFromCurrentImageContext();    // 生成预览图
   UIGraphicsEndImageContext();

    ......
}
```

代码片断 2 HTFileDownladerDelegate 接口

```
- (NSString*)home;
- (NSString*)home;
- (BOOL)cdTo:(NSString*)path;
- (void)cdUp;
- (void)logout;
- (void)loginWithCompletionHandle:(void(^)())handle;
- (NSArray*)itemsInPath:(NSString*)path;
- (void)itemsInPath:(NSString*)path
asyncWithBlock:(void(^)(NSArray *items))block;
- (NSArray*)itemsInPathForceReload:(NSString*)path;
- (BOOL)isLogin;
- (void)login;
- (void)downloadFile:(NSString*)path
              toPath:(NSString*)localPath
 withProgressHandler:(id<HTDownloadProgressDelegate>)handler;
```

代码片断 3 _fileTree 字典结构

```
{
    "id":0;
    "type":"folder";    // 根目录
```

```
    "items":
    {
        "folder1":
        {
            "id":12494341;
            "type":"folder";
            "items":
            {
                "file1":
                {
                    "id":2222222;
                    "type":"file";
                    "url":"https://www.box.com/download/....doc";
                }
                "file2":
                {
                    "id":1111111;
                    "type":"file";
                    "url":"https://...";
                }
            }
        }
        "file3":
        {
            "id":2232323;
            "type":"file";
            "url":"https://....";
        }
    }
}
```

代码片断 4 HTFileServiceBoxnet 初始化代码

```
- (id)init
{
   self = [super init];
   if (self){
      _fileTree = [[NSMutableDictionary alloc] initWithObjectsAndKeys:
                  [NSNumber numberWithInt:0], @"id",
                  @"folder", @"type", nil];
   }
   return self;
}
```

代码片断 5 HTFileServiceBoxnet 的主要方法片断

```
// 给定完整路径，返回路径对应的节点
- (NSDictionary*)getNodeinPath:(NSString *)path
{
```

```
    if (!path || [path isEqualToString:@""])
        return nil;

    NSArray *paths = [path pathComponents]; // 拆分路径为部件
    NSDictionary *current = _fileTree;          // 从根目录开始
    for (NSString *folder in paths) {           // 遍历路径部件
        if ([folder isEqualToString:@"/"]) {
            continue;                           // 跳过根目录
        }
        NSMutableDictionary *items = [current valueForKey:@"items"];
        if (!items) {        // 如果当前还有没缓存子目录，构造子目录
            items = [self buildFileMapTree:[current valueForKey:@"id"]];
            if (items)
                  /*
                   * 构造好后，保存到_fileTree，方便下次访问
                   */
                [current setValue:items forKey:@"items"];
            else {
                current = nil;
                break;
            }
        }
        current = [items valueForKey:folder];   //  进入下一级节点
        if (!current) {
            break;        // 没有这个目录，退出
        }
    }
    return current;
}

// 给定完整路径，返回文件的 ID
- (NSNumber*)getFileIDinPath:(NSString *)path
{
    NSDictionary *current = [self getNodeinPath:path]; // 获取节点
    return [current valueForKey:@"id"];
}
- (NSMutableDictionary*)buildFileMapTree:(NSNumber *)folderID
{
......

    BoxFolder * folderModel = 同步获取文件夹信息(folderID);
    if (成功执行) {
        int count = [folderModel.objectsInFolder count];
        NSMutableDictionary *items = [NSMutableDictionary
dictionaryWithCapacity:count]; // 生成空节点

        for (BoxObject *boxojb in folderModel.objectsInFolder) {
            BOOL isFile = [boxojb isKindOfClass:[BoxFile class]];
            NSDictionary *dic = nil;
            if (isFile) {    // 如果是文件
                NSString *fileURL = 生成文件完整链接;
```

```
            dic = [NSMutableDictionary dictionaryWithObjectsAndKeys:
                  boxojb.objectId,@"id",
                  @"file",@"type",
                  [NSURL URLWithString:fileURL],@"url", nil];
        }
        else {                // 如果是文件夹
            dic = [NSMutableDictionary dictionaryWithObjectsAndKeys:
                  boxojb.objectId,@"id",
                  @"folder",@"type",
                  ((BoxFolder*)boxojb).fileCount,@"count", nil];
        }
        [items setValue:dic forKey:boxojb.objectName];// 保存子节点
      }
      return items;        // 返回构造好的节点
   }
   else
       return nil;
}
```

代码片断 6 Box 的文件下载

```
- (void)downloadFile:(NSString *)path
               toPath:(NSString *)localPath
 withProgressHandler:(id<HTDownloadProgressDelegate>)handler
{
   NSFileManager *fm = [NSFileManager defaultManager];
   NSString *to = [localPath stringByAppendingPathComponent:fileName];
   NSString *or = to;
    int count = 1;
    // 这里主要用来判断重名文件是否存在，如果存在就给它加上编号，即
    // file.doc 重命名为 file(1).doc，若还存在，则为 file(2).doc，依此类推
   while ([fm fileExistsAtPath:or]) {          // 如果文件存在
      NSString *ext = [to pathExtension];      // 保存后缀名
       // 去掉后缀名
      NSString *tmp = [to stringByDeletingPathExtension];
       // 加上编号及后缀名
      or = [tmp stringByAppendingFormat:@"(%d).%@",count++,ext];
   }
   to = or; // 一个不会重名的完整路径

    BoxDownloadOperation *download =  ...
   [download start];       // 开始同步下载，其中 delegate 用于通知外面下载进度
}
```

代码片断 7 Dropbox 的文件下载

```
- (void)downloadFile:(NSString*)path
              toPath:(NSString*)localPath
 withProgressHandler:(id<HTDownloadProgressDelegate>)handler
{
```

```
    NSString *fileName = path.lastPathComponent;

    // 检查文件重名，并加上编号，同 Box

    [downloadDelegateQueue setObject:handler
                              forKey:to];   // 保存 handler，以便更新下载进度
    [handler downloadFileStart:to];     // 通知下载开始
    [self.restClient loadFile:path
                   intoPath:to];     // 开始异步下载
    CFRunLoopRun();                 // 当前线程进入消息循环，阻塞以实现同步
}

// 文件下载完成后
- (void)restClient:(DBRestClient*)client
        loadedFile:(NSString*)destPath
{
    id<HTDownloadProgressDelegate> handler = [downloadDelegateQueue
objectForKey:destPath];
    [handler downloadFile:destPath
                  complete:YES];    // 通知下载完成
    [downloadDelegateQueue removeObjectForKey:destPath];// 移除 handler

    CFRunLoopStop(CFRunLoopGetCurrent());   // 退出消息循环
}
```

代码片断 8 HTFileDownloader 单例实现

```
// 全局静态变量，保存唯一的一个实例
static HTFileDownloader * _sharedDownloader = nil;

+ (HTFileDownloader*)sharedDownloader
{
    @synchronized(self){    // 同步以防止多线程访问时出错
        if (!_sharedDownloader) {
            _sharedDownloader = [[HTFileDownloader alloc] init];
        }
    }
    return _sharedDownloader;
}

+ (id)alloc
{
    @synchronized(self){
        NSAssert(_sharedDownloader == nil, @"Singleton can't be allocated
twice!");    // 确保只有一个实例
        return [super alloc];
    }
}
```

代码片断 9 添加下载任务

```
...
// 下载云存储文件 /folder/remotefile1 到本地文件 /localfolder/localfile1
NSString *remoteFile = @"/folder/remote_file1";
NSString *localFile = @"/localfolder/localfile1";

HTDownloadOperation *download = [HTDownloadOperation
                            operationWithRemotePath:remoteFile
                                        andLocalPath:localFile];

[[HTFileDownloader sharedDownloader] addDownloadOperation:download];
```

代码片断 10 文件下载消息的监听

```
......
// 当控制器的显示的路径改变时，监听新的消息
- (void)setLocalPath:(NSString *)localPath
{
   ......
   NSNotificationCenter *center = [NSNotificationCenter defaultCenter];
   [center removeObserver:self];   // 不再监听之前的消息
   [center addObserver:self            // 监听新路径的消息
               selector:@selector(monitorFileDownload:)
                   name:localPath
                 object:nil];
}

- (void)monitorFileDownload:(NSNotification*)notification
{
    // 获取消息信息
NSDictionary *info = (NSDictionary*)notification.object;
    // 找出进度更新的文件名
   NSString *file = [info objectForKey:@"file"];

    if (找到 Grid) {
        根据以下情况更新视图：
        进度为 0~1 时，更新进度条；
        进度小于 0 时，此文件下载失败；
        进度大于 1 时，下载已经完成；
    }
    else {
        创建一个新的 Grid，并保存到 docCells 字典中；
    }
}
```

代码片断 11 发送文件下载消息

```
#pragma mark - HTDownload Delegate
```

```
- (void)downloadFileStart:(NSString*)path
{
    发送消息：文件 path 开始下载
}

- (void)downloadFile:(NSString*)path progress:(float)ratio
{
    发送消息：文件 path 已经下载了 ratio*100%
}

- (void)downloadFile:(NSString*)path complete:(BOOL)success
{
     发送消息：文件 path 下载完成
}
```

代码片断 12 服务端 Email 处理过程

```
Email email = 收到的新邮件;
String send = email.getFrom();
String subject = email.getSubject();
String fundname = 正则匹配找出 Fund(subject);

int userid = 通过注册邮箱查询用户 ID(send);
if (找到) {
    ArrayList<Fund> funds = 通过用户 ID 找到所有的 Fund(userid);
    Fund tofund = null;
    for (Fund fund:funds) {
        if (fund.name && fund.name.equals(fundname)) {
            tofund = fund;
            break;
        }
    }
    if (fund) {     // 找到对应的 Fund
        保存 email 及所属的 fund.id;
    }
}
else
    不做处理，退出;
```

本科生毕业论文（设计）任务书

一、题目：基于 iOS 的投资信息系统的设计与实现

二、指导教师对毕业论文（设计）的进度安排及任务要求：

要求查阅云存储及 iOS 开发相关的论文 10 篇以上，定期汇报毕业设计进度，独立完成毕业论文。4 月 2 日前完成开题报告。，4 月 30 日前完成中期报告，5 月 20 日前完成毕业论文。

具体要求有：

1. 对 iOS 系统进行深入的学习
2. 了解云服务授权过程，并测试实际效果
3. 对系统架构的特点进行分析，如果可能，提出改进方案并实现

起讫日期 **20** 年 月 日 至 **20** 年 月 日

指导教师（签名）＿＿＿＿职称＿＿＿＿

三、系或研究所审核意见：

负责人（签名）＿＿＿＿

年 月 日

毕 业 论 文（设计） 考 核

一、指导教师对毕业论文（设计）的评语：

指导教师(签名）

年 月 日

二、答辩小组对毕业论文（设计）的答辩评语及总评成绩：

成绩比例	文献综述占（10%）	开题报告占（20%）	外文翻译占（10%）	毕业论文（设计）质量及答辩占（60%）	总 评成绩
分值					

答辩小组负责人（签名）

年 月 日