基于IOS的社区金融App的关键技术研究

秦 业,高建华
(上海师范大学 计算机科学与工程系,上海 200234)

摘 要:随着移动互联网的兴起和智能终端平台的普及,人们对移动理财的需求越来越大,然而市场上并没有一款简单易用、功能广泛的在线理财App产品。对此,文中基于"产学研"的需要提出社区金融App项目,完成基本UI的设计与实现,功能模块API的开发、分析前后端数据交互、JSON解析等。实验重点研究和解决了多线程和缓存框架的选取和代码实现的问题。首先分析了并发操作可能存在的问题,以及多线程技术是如何予以解决的,具体讨论了GCD框架的优势和工作原理;然后分析了缓存对App的重要性以及不同缓存策略的适用场景和工作流程;最后在实际项目中组合使用不同的缓存策略完成所需要的功能。阐述了未来开发的重点在于统一两大移动平台,并指出了未来App开发的重点在于混合App的开发。

关键词:社区金融;IOS;JSON;多线程;缓存
中图分类号:TP39 **文献标识码**:A **文章编号**:1673-629X(2018)12-0157-05
doi:10.3969/j.issn.1673-629X.2018.12.033

Research on Key Technology of Community Finance App Based on IOS

QIN Ye,GAO Jian-hua
(Department of Computer Science and Technology,Shanghai Normal University,Shanghai 200234,China)

Abstract:With the rise of mobile Internet and the popularity of intelligent terminal platform,people's demand for mobile finance is growing. However,there is not a simple and wide range of online financial App products on the market. Therefore,we put forward the community financial App project based on the requirement of "production,learning and research",which completes the design and implementation of basic UI,development of functional module API,analysis of front and back data interaction,JSON analysis and so on. The experiment focuses on solving the selection and code realization of multi-thread and the cache framework. First we analyze the possible problems of concurrent operation and how to solve it,and then discuss the advantages and working principles of GCD framework. After that,the significance of caching to App,the application scenarios and workflow of different caching strategies are also analyzed. Finally,different caching strategies are combined to fulfill the functions needed in practical projects. In addition,we expound that the focus of future development is to unify the two big mobile platforms,and point out that the focus of future App development lies in the development of hybrid App.

Key words:community finance;IOS;JSON;multithreading;cache

0 引 言

随着智能终端普及率的不断提高,生活中的很多问题都能用App来解决。对于生活社区,一款能解决用户金融理财、生活服务、缴费的App必不可少,因此文中推出了社区金融App产品。该App功能包括手机充值、代缴水煤电费;在线问诊、预约、挂号、咨询;查阅购买基金、股票、黄金等金融理财产品。

1 IOS平台

IOS系统主要用在iPhone、iPad等产品[1],是移动端最受欢迎的操作系统。IOS App开发采用Xcode,它集成了各个版本的模拟器,适合开发iPhone、iPad、等苹果公司产品的应用。IOS系统基于FreeBSD系统,从本质上说IOS是Unix的一个分支,特点主要体现在后台运行机制上,包括三个方面:

收稿日期:2018-01-03 **修回日期**:2018-05-09 **网络出版时间**:2018-07-04
基金项目:国家自然科学基金(61672355);上海市引进技术的吸收与创新年度计划项目(JJ-YJCX-01-15-5250)
作者简介:秦 业(1993-),男,硕士研究生,CCF会员(72459M),研究方向为软件工程、软件重构;高建华,博士,教授,博导,研究方向为软件可靠性理论与设计、软件开发环境与开发技术、数据安全性与计算机安全、网络测试、LSIVLSI测试等。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20180703.1510.016.html

(1)IOS 系统有独特的任务管理机制。当应用程序不在前台运行时,除了部分服务,其他应用在 10 分钟后都被系统挂起。被挂起等同于不执行,只是数据驻留在内存而已。

(2)内存管理机制。在执行任意应用时,应用向系统申请内存空间,如果应用在使用的过程中不断申请内存,超过了系统限定的内存区间,系统会发出内存警报,严重时会直接将应用杀死。同样,如果应用向系统申请内存时,系统内存空间不足,系统会结束后台应用的运行,以释放空间资源[2]。

(3)伪多任务。例如微信,退出后不在后台运行。用户收到消息是因为系统推送服务。无论用户的应用程序是否运行,IOS 都会在后台维护这个服务实现伪多任务,所有应用程序共用这一服务。

2 社区金融 App 架构

IOS App 开发主要有 Native App、Hybrid App、Web App[3]等三种开发架构,各自特点如表 1 所示。

表 1 三种 IOS 开发架构比较

特点	Web App (网页应用)	Hybrid App (混合应用)	Native App (原生应用)
开发成本	低	中	高
维护更新	简单	简单	复杂
体验	差	中	优
Store、Market 认可	不认可	认可	认可
安装	不需要	需要	需要
跨平台	优	优	差

该项目根据用户体验选择 Native App 作为开发架构。

2.1 基本视图设计原则

在手机平台上,手指触摸不是一个精确点击,而是一个"块"状的点击范围,在范围内的控件会被点击操作激发,这点和鼠标完全不一样。因此在功能设计中,"块"的设计相当重要。基于目前业界主流的页面设计,结合 UITabbarController[4]和 UITableViewController 实现需求。前者可以将屏幕底部等分或自定义分成项目需要的模块,用来布置项目中最常用的几个功能模块;后者则能够以类似表格的形式展现每个功能模块对应的内容。

2.2 App 开发

IOS App 开发分为客户端和服务器端开发,是一个典型的 C/S 程序,计算工作由服务器端完成,客户端实现 GUI 界面的展示、数据的获取和解析、用户操作的捕捉等。这种前后端分离的开发模式有利于各司其职和实现松耦合,开发出更好的应用。文中专注客户端开发过程中的部分关键技术,主要包括 MVVM 设计模式、JSON 数据解析、多线程编程技术和缓存机制技术。

3 关键开发技术

3.1 MVVM 设计模式

MVVM 模式是 Model-View-ViewMode 模式的简称。由视图(View)、视图模型(ViewModel)、模型(Model)三部分组成。通过这三部分实现 UI 逻辑、呈现逻辑和状态控制、数据与业务逻辑的分离[5]。

Model 层代表了描述业务逻辑和数据的一系列类的集合。它也定义了数据修改和操作的业务规则。

View 代表了 UI 组件,像 CSS、Jquery、html 等,只负责展示从 Presenter 接收到的数据,也就是把模型转化成 UI。

View Model 负责暴露方法,命令,其他属性来操作 View 的状态,组装 Model 作为 View 动作的结果,并且触发 View 自己的事件。

MVVM 的技术关键点是:

(1)用户只通过 View 和服务端交互信息。

(2)View 和 ViewModel 是多对一关系。意味着一个 ViewModel 只映射多个 View(在 MVC 中一个 View 对应一个 ViewController)。

(3)View 持有 ViewModel 的引用,但是 ViewModel 没有任何 View 的信息。

(4)View 和 ViewModel 之间有着双向数据绑定关系。

三个组件之间的关系如图 1 所示。

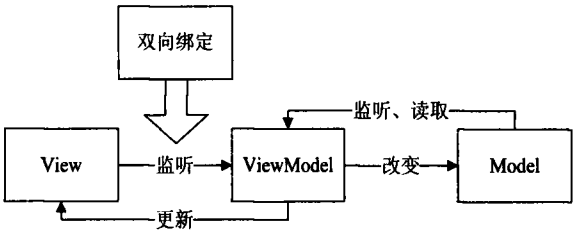

图 1 MVVM 模块图

MVVM 这种新的设计模式真正做到了将页面与数据逻辑分离,解决了传统 MVC 模式中 Controller 过于臃肿的问题,并且可以将使用次数多的视图逻辑放到 ViewModel 中,通过一对多的映射关系供更多的 View 重用,提高了复用性。

3.2 JSON 数据解析

客户端向服务器端发送数据请求,服务器端响应后返回 XML 或者 JSON 格式的数据。由于 JSON 格式是一种轻量级数据交互格式[6],目前已经取代 XML,该项目也采用 JSON 格式。

JSON 解析指的是将服务器端传来的 JSON 格式

数据转化为 IOS 端显示的模型数据,并将 JSON 中各个属性数据赋值给模型对应属性。从 IOS5 开始,Xcode 原生 JSON 解析类库 NSJSONSerialization,但在实际开发中,NSJSONSerialization 使用难度大、易出错。Github 有许多类似的第三方类库更加优秀,有 YYModel、MJExtension 等,其中 YYModel 性能更好,使用起来更加简单方便。因此该项目使用 YYModel 解析 JSON 数据。

YYModel 是通过对 NSObject 的部分内容进行封装来实现功能,具体是:

YYClassInfo 是对 Class 进行封装描述:

· YYClassIvarInfo 对 Class 的 Ivar 进行封装描述;

· YYClassMethodInfo 对 Class 的 Method 进行封装描述;

· YYClassPropertyInfo 对 Class 的 Property 进行封装描述。

YYModel 是对 YYClassInfo 进行封装,并暴露调用接口给用户,具体是:

· YYModelMeta 对 YYClassInfo 进行封装描述;

· YYModelPropertyMeta 对 YYClassProperty 进行封装描述。

其中主要提供了三种解析类别:

(1) NSObject(YYModel):提供一些字典模型互转的方法,将对 key/value 进行匹配,赋值给 Model 对应的 property。

(2) NSArray(YYModel):为 NSArray 提供字典转模型的方法。

(3) NSDictionary(YYModel):为 NSDictionary 提供字典转模型方法。

下面以用户点击加载"股票信息"为例,具体的解析流程如图 2 所示。

对应加载股票信息模块的部分代码如下:

```
//result 表示服务器端返回的 JSON 数据,先保存到字典当中
NSDictionary * dic=result[@"singleData"][0];
//StockDetail 调用 YYModel 里的 yy_modelWithJSON 方法,返回一个 StockDetail 模型对象
StockDetail * detail=[StockDetail yy_modelWithJSON:dic];
self.stockDetail=detail;
//YYModel 中的 yy_modelWithJSON 方法将 JSON 数据转化成预定义好的 StockDetail 对象
+(instancetype)yy_modelWithJSON:(id)json {
NSDictionary * dic=[self _yy_dictionaryWithJSON:json];
//将 StockDetail 对象赋值给模型中的 self.stockDetail,显示到界面上
return [self yy_modelWithDictionary:dic];}
```

上述代码是通过 AFNetworking 实现 HTTP 或 HTTPS 链接请求,获取后端 API 返回的 JSON 字符串[7],再通过 YYModel 实现将 JSON 字符串转化为 NSDictionary 格式的模型对象,最后将模型的数据显示到界面上。

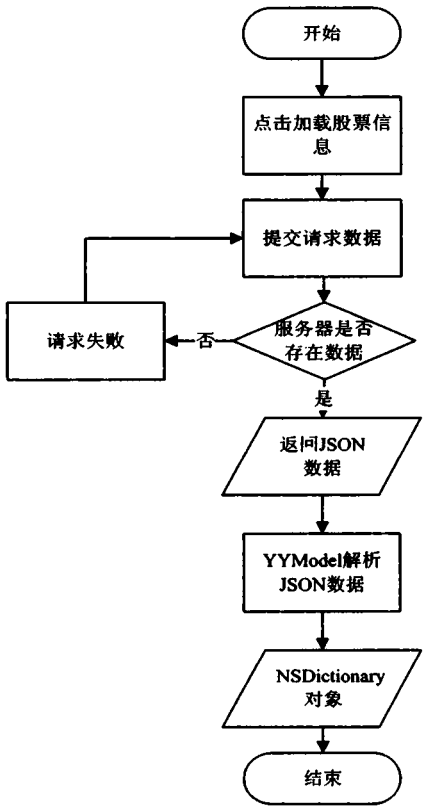

图 2　获取股票 JSON 流程

3.3　多线程编程技术

实际开发中,主线程主要负责完成主控制器的调用、控制和控制器之间的通信任务,而控制器所控制的视图上的数据获取则交给其他线程完成。因为线程中任务的执行是顺序执行的,也就是说在一个时间段内一个线程只能执行一个任务,对于需要同时执行的任务就需要多线程并发执行[8]。

IOS 中实现多线程的方式有很多种,包括 NSThread、NSOperation 等。但最常用的是 GCD。GCD 是一个在后端管理线程池的工具[9],让开发人员无需和线程直接打交道。

GCD 是基于 C 语言的底层 API,其中最重要的概念就是 dispatch_queue,它是一个对象,可以接受任务并以先到先执行的方式处理任务。根据 GCD 队列的种类处理方式有所不同,最主要的有以下 2 种:

Serial:串行队列以先进先出(FIFO)的顺序执行任务,所以串行队列经常用来做访问某些特定资源的同步处理。可根据需要创建多个队列,而这些队列相对其他队列都是并发执行的。即若创建了 4 个串行队列,每一个队列在同一时间都只执行一个任务,对这四个任务来说,它们是相互独立且并发执行的。如果需

要创建串行队列，一般用 dispatch_queue_create 这个方法来实现。

Concueerent：并发队列虽然是能同时执行多个任务，但这些任务仍然是按照先到先执行（FIFO）的顺序来执行的。并发队列会基于系统负载来合适地选择并发执行这些任务。而在 iOS5 之后，也可以用 dispatch_queue_create，并指定队列类型为 DISPATCH_QUEUE_CONCURRENT 来自己创建一个并发队列。

创建需要的队列后，需要将其添加到任务当中，这又分为同步和异步两种。一般情况下，使用 dispatch_async 和 dispatch_async_f 来执行异步操作。比如，添加一个 block 对象或 C 函数到一个队列后就会立即返回，任务会由 GCD 决定执行顺序，以及任务执行完毕时间。好处是，若需要在后台执行一个基于网络或 CPU 密集型任务，使用异步方法不会阻塞当前线程[10]。

尽管一般情况下，优先选择异步操作，但是在某些情况下，还是需要任务同步来执行。比如需要用同步操作来防止资源竞争或其他同步问题。这时可以用 dispatch_sync 和 dispatch_sync_f 方法把任务添加到队列中，这样被添加的任务会阻塞当前线程，直到这些任务执行完，确保同一资源在同一时间只能被一个线程访问到。

以用户点击“详情”获取对应股票的详情信息为例，主线程在获取视图控制器和文字内容时，子线程负责加载图片，这样可以做到在加载图片的过程中不会出现卡顿现象，步骤如图 3 所示。

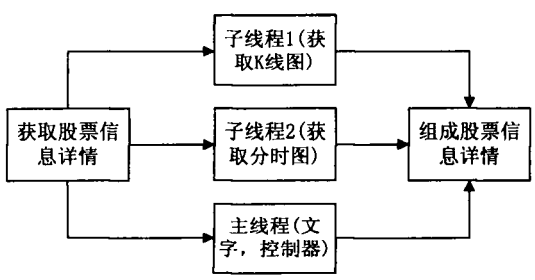

图 3 YYCache 结构关系图

当然，GCD 的使用场景很多，该项目在清理缓存信息时使用 GCD 异步删除本机的缓存，部分实现代码如下：

```
//判断缓存文件路径是否存在
if (! [path isEqualToString:_cachePath]) {
//调用 GCD,开启异步线程,传递线程队列和回调函数
dispatch_async(_queue, ^{
//将字符串路径转化为 IOS 识别的文件路径
NSURL *fileURL=[NSURL URLWithString:path];
NSDictionary *dictionary=[fileURL
resourceValuesForKeys: @ [NSURLContentModificationDateKey] error:nil];
//得到内容的最后修改时间
NSDate *modificationDate = [dictionary objectForKey:NSURLContentModificationDateKey];
if (modificationDate.timeIntervalSince1970 - date.timeIntervalSince1970 <0) {
[_fileManager removeItemAtPath:fileURL.absoluteString error:nil];
//从文件路径和缓存中删除缓存文件
[_memoryCache removeObjectForKey:fileURL.lastPathComponent];}});}
```

3.4 缓存机制技术

在 App 的使用过程中缓存机制是必不可少的，根据存储的不同缓存也可以分为内存缓存和文件缓存[11]，分别具有高速度低容量和低速度高容量的特点。YYCache 同时具有这两种缓存机制，项目选择 YYCache 作为缓存框架。

YYCache 中内存缓存和文件缓存的关系如图 4 所示。

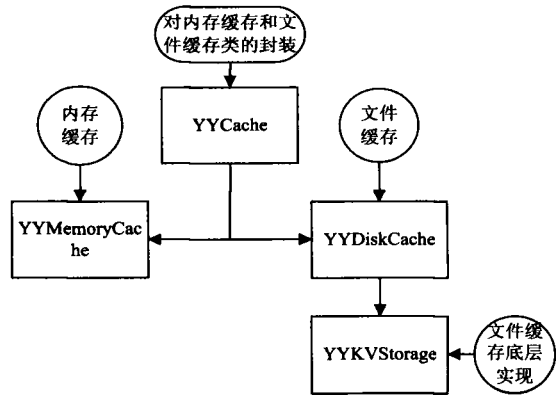

图 4 YYCache 结构关系图

内存缓存是初次请求数据时将获取到的数据存放到内存当中，当用户再次访问到该数据时无需再向服务器请求数据，直接从内存中获取缓存数据，具有高速度的特点，但由于内存较为昂贵，所以容量较低。

文件缓存是将获得的数据存放在客户端数据库文件中，社区金融 App 使用了 SQLite 数据库做文件缓存。在用户初次请求服务器获得数据时，将音乐、图片等大容量文件按照键值对的形式存放到 SQLite.db 文件中，当客户端再次访问相同页面时，控制器会先检测是否有对应的 db 文件，若无则发送请求给服务器，并将返回的数据缓存到 SQLite 中[12]，若有则直接从 db 文件中取得数据并显示在界面上。

在请求社区金融新闻信息时，可能同时存在内存缓存和文件缓存，使用 YYCache 得到的缓存流程如图 5 所示。

在获取金融类新闻信息时使用 YYCache 的部分代码如下所示：

```
//初始化 YYCache
```

```
YYCache * cache = [YYCache cacheWithName:@"mydb"];
//缓存普通字符
[cache setObject:@"中国中车" forKey:@"name"];
NSString * name = (NSString *)[cache objectForKey:@"name"];
NSLog(@"name: %@",name);
//缓存模型
[cache setObject:(id)model forKey:@"user"];
//缓存数组
NSMutableArray * array=@[].mutableCopy;
For (NSInteger i=0;i<10;i++) {
[array addObject:model];
}
//异步缓存
[cache setObject:array forKey:@"user" withBlock:^{
// 异步回调
NSLog(@"%@", [NSThread currentThread]);
NSLog(@"array 缓存完成....");
}];
//延时读取
dispatch_after(dispatch_time(DISPATCH_TIME_NOW,(int64_t)(0.3 * NSEC_PER_SEC)),dispatch_get_main_queue(),^{
//异步读取
[cache objectForKey:@"user" withBlock:^(NSString * _Nonnull key,id_Nonnull object){
//异步回调
NSLog(@"%@",[NSThread currentThread]);
NSLog(@"%@",object);}];});
```

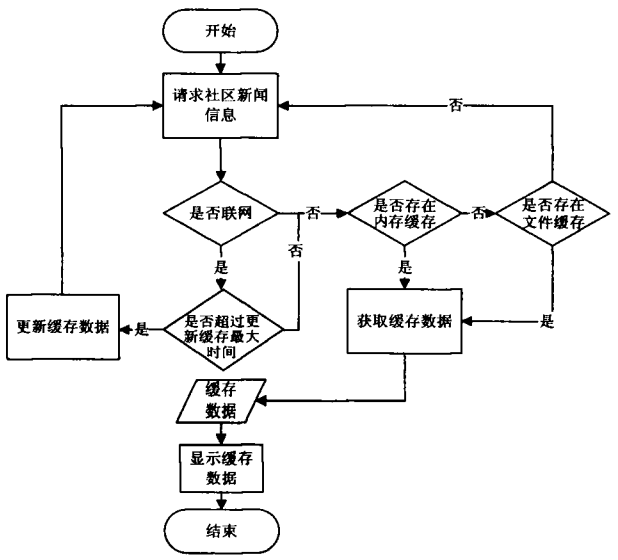

图 5　获取缓存流程

内存缓存和文件缓存的区别在于,当应用还在进程中时,应用有内存缓存和文件缓存两种缓存方式,当应用进程被杀死时,应用只存在文件缓存,内存缓存将被清空。因此对于图片、音频、视频等大容量文件,App 使用了文件缓存技术将数据缓存到 SQLite 数据库中,节省资源;对于文字等小容量内容则缓存到内存中,方便获取[13]。

4　结束语

为了更好地适应移动互联网时代,社区金融 IOS 端 App 将不断更新维护,未来开发的重点在于功能模块的扩展,让 App 满足更多理财需求。考虑到移动开发成本问题,统一 IOS 和安卓 App 的跨平台开发[14]是大势所趋,因此从 Native App 到 Hybrid App 的迁移是以后工作的重点。

参考文献:

[1] BUCANEK J. Learn IOS App development[M]. [s. l.]: Apress,2013:18-21.

[2] 刘乐廷,李敬兆. IOS 内存开发管理机制的研究[J]. 计算机与现代化,2013(3):196-199.

[3] KIPAR D. Test automation for mobile hybrid applications:using the example of the BILD App for Android and iOS [M]. [s. l.]:[s. n.],2014.

[4] SADUN E. The iOS 5 developer's cookbook:core concepts and essential recipes for iOS programmers[M]. [s. l.]:Addison Wesley,2012:161-167.

[5] MISHRA A. The MVVM architectural pattern[M]//iOS code testing. [s. l.]:[s. n.],2017:43-60.

[6] STARK J. Support Everything:building apps that run everywhere with HTML5, REST, and JSON - O'Reilly media free,live events[M]. [s. l.]:Jonathan Stark,2017.

[7] 胡扬帆,杨　刚,胡颢石. 结合 LBS 和信息推送的博物馆 APP 的设计实现[J]. 计算机应用与软件,2013,30(12):108-112.

[8] SAKAMOTO K, FURUMOTO T. Pro multithreading and memory management for iOS and OS X[M]. [s. l.]:[s. n.],2013.

[9] 潘小龙. IOS 系统中不同多线程技术的研究和比较[J]. 中国新通信,2014(24):21-22.

[10] OBJECTS B, SYNTAX T, DISPATCH G C, et al. Concurrent programming in Mac OS X and iOS[M]. [s. l.]:[s. n.],2011.

[11] 黄天柱,涂时亮. iOS 开发 UITableView 加载图片的内存管理[J]. 计算机系统应用,2012,21(9):113-118.

[12] 梁效宁,黄　旭,朱星海. SQLite 数据库文件恢复提取技术研究[J]. 计算机科学,2016,43(12A):16-19.

[13] 仲　媛,王　菁,韩燕波,等. HTML5 混合式移动社交应用中缓存管理机制的研究[J]. 计算机科学,2017,44(2):82-87.

[14] 李张永,陈和平,顾进广. 跨平台移动 Web 开发框架与数据交互方法[J]. 计算机工程与设计,2014,35(5):1827-1832.