

读懂「唱吧KTVHTTPCache」设计思想

Lefe (/u/7c787af04cd1) [+ 关注](#)

2017.12.10 22:22* 字数 2505 阅读 1556 评论 7 喜欢 26

(/u/7c787af04cd1)

读懂「唱吧 KTVHTTPCache」设计思想

经过作者的指点，本文更新了日志原理说明，HttpServer 作用，图片缓存

最近看到各大V转发关于 唱吧音视频框架 KTVHTTPCache 的开源消息，首先我非常感谢唱吧 iOS 团队能够无私地把自己的成果开源。我本人对于缓存的设计也比较感兴趣，也喜欢写一些东西，希望能把自己一些小技巧分享给需要的同学，这也是我们 #iOS 知识小集# 一直做的事情。抱着好奇的心，想了解一下唱吧是如何设计 KTVHTTPCache (<https://link.jianshu.com?t=https://github.com/ChangbaDevs/KTVHTTPCache>) 的，没想到越看越难，最后竟然花了将近2天的时间看完了。

安装时解读

在进行安装的时候，发现 KTVHTTPCache (<https://link.jianshu.com?t=https://github.com/ChangbaDevs/KTVHTTPCache>) 主要依赖了 CocoaHTTPServer (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaHTTPServer>) 这个库，而 CocoaHTTPServer (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaHTTPServer>) 又依赖了 CocoaAsyncSocket (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaAsyncSocket>) 和 CocoaLumberjack (<https://link.jianshu.com?t=https://github.com/CocoaLumberjack/CocoaLumberjack>)。可以肯定一点 KTVHTTPCache 使用 CocoaHTTPServer (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaHTTPServer>) 作为 HttpServer。

CocoaHTTPServer is a small, lightweight, embeddable HTTP server for Mac OS X or iOS applications.

Sometimes developers need an embedded HTTP server in their app. Perhaps it's a server application with remote monitoring. Or perhaps it's a desktop application using HTTP for the communication backend. Or perhaps it's an iOS app providing over-the-air access to documents. Whatever your reason, CocoaHTTPServer can get the job done

```
→ KTVHCHTTPCacheDemo git:(master) x pod install
Analyzing dependencies
Downloading dependencies
Installing CocoaAsyncSocket (7.6.2)
Installing CocoaHTTPServer (2.3)
Installing CocoaLumberjack (3.3.0)
Installing KTVHTTPCache (1.0.0)
Generating Pods project
Integrating client project
```

podinstall.png



Readme 中提到:

KTVHTTPCache 由 HTTP Server 和 Data Storage 两大模块组成。

另外一个主要模块就是 Data Storage，它主要负责资源加载及缓存处理。从这里可以看出，KTVHTTPCache (<https://link.jianshu.com?t=https://github.com/ChangbaDevs/KTVHTTPCache>) 主要的工作量是设计 Data Storage 这个模块，也就是它的核心所在。

使用

其本质是对 HTTP 请求进行缓存，对传输内容并没有限制，因此应用场景不限于音视频在线播放，也可以用于文件下载、图片加载、普通网络请求等场景。 --- KTVHTTPCache

既然这么好使，我们可以试试各种情况，demo 中虽然没有给出其它方式的缓存示例，我们可以探索一下。不过我测试了下载图片的，并没有成功，其它中情况也就没有试验。我猜测，如果想支持这几种情况，应该需要修改源码（如果作者能看到，忘解答一下，不知道我的猜测是否正确）。

视频缓存（Demo 中提供，亲测可以）

- 全局启动一次即可，主要用来启动 HttpServer，不理解的话，你可以把它想成手机端的 HTTP 服务器，当你向 HTTP 服务器发出 Request 后，服务器会给你一个 Response，后面我们会特意分析一个 HttpServer。

```
[KTVHTTPCache proxyStart:&error];
```

- 根据原 url 生成一个 proxy url（代理 Url），并使用代理 url 获取数据，这样 HttpServer 就会截获这次请求。比如原 url 为 <http://lzauiw.changba.com/userdata/video/940071102.mp4> 它对应的 proxy url 为

```
http://localhost:53112/request-940071102.mp4?requestType=content&originalURL=http%3A%2F%2Flzauiw.changba.com%2Fuserdata%2Fvideo%2F940071102.mp4
```

看图会更好理解:

proxyUrl.png

```
NSString * proxyURLString = [KTVHTTPCache proxyURLStringWithOriginalURLString:URLString];
```

- 播放，注意这里使用的是代理url，进行播放，而不是原url。

```
[AVPlayer playerWithURL:[NSURL URLWithString: proxyURLString]];
```

图片缓存

这次试验结果没能成功，在缓存中找不到缓存图片，或许是我少些什么。

```
- (void)testImageCache
{
    NSString *imageUrl = @"http://g.hiphotos.baidu.com/image/pic/item/e824b899a9014c08d8614343007b02087af4f4fa";
    NSString *proxyStr = [KTVHTTPCache proxyURLStringWithOriginalURLString:imageUrl];
    NSURLSessionTask *task2 = [[NSURLSession sharedSession] downloadTaskWithURL:[NSURL URLWithString:proxyStr] completionHandler:^(NSURLResponse *response, NSData *data, NSError *error) {
        [task2 resume];
    }];
}
```

控制台打印出的日志可以发现，图片没能缓存是因为 content type 错误导致的，因为 KTVHTTPCache (<https://link.jianshu.com?t=https://github.com/ChangbaDevs/KTVHTTPCache>) 目前只支持 video, audio 和 application/octet-stream 三种类型的，如果想缓存图片可以添加一种 content type。

```
KTVHTTPCache[818:16793] KTVHCDDataNetworkSource : response error
http://g.hiphotos.baidu.com/image/pic/item/e824b899a9014c08d8614343007b02087af4f4fa.
content type error
```

在 KTVHCDDataRequest 类中的初始化方法中修改

```
self.acceptContentTypes = @[KTVHCDDataContentTypeVideo,
                             KTVHCDDataContentTypeAudio,
                             KTVHCDDataContentTypeOctetStream];
```

为

```
self.acceptContentTypes = @[KTVHCDDataContentTypeVideo,
                             KTVHCDDataContentTypeAudio,
                             KTVHCDDataContentTypeOctetStream,
                             KTVHCDDataContentTypeImage];
```

这样既可以支持缓存图片的需求。这里作者 @程序员Single 特别提示，如果做图片缓存，不建议使用 HttpServer 做中转，也就不需要 Proxy URL，这样会节省不必要的开销。HttpServer 的主要目的是为了 Hook 播放器的网络请求，从而可以做到缓存数据。可以直接使用 KTVHTTPCache 中的方法来生成 Reader 读取数据。可以参考 KTVHCDHTTPResponse 的实现。

```
+ (KTVHCDDataReader *)cacheConcurrentReaderWithRequest:(KTVHCDDataRequest *)request;

+ (KTVHCDDataReader *)cacheSerialReaderWithRequest:(KTVHCDDataRequest *)request;
```

框架设计

KTVHTTPCache 由 HTTP Server 和 Data Storage 两大模块组成。前者负责与 Client 交互，后者负责资源加载及缓存处理。

这句话如果没有看源码，其实很难理解，涉及到如何交互的问题（我是这样认为的，也许你比我聪明，能理解作者的含意）。通俗地讲，HTTP Server 和 Data Storage 是 KTVHTTPCache 两大重要组成部分，HTTP Server 主要负责与用户交互，也就是最顶层，最直接与用户交互（比如下载数据），而 Data Storage 则在后面为 HTTP Server 提供数据，数据主要从 DataSourceer 中获取，如果本地有数据，它会从 KTVHCDDataFileSource 中获取，反之会从 KTVHCDDataNetworkSource 中读取数据，这里会走下载逻辑（KTVHCDDownload）。





HttpServer

这层设计比较简单，主要是用了 CocoaHTTPServer (<https://link.jianshu.com?t=https://github.com/robbiehanson/CocoaHTTPServer>) 来作为本地的 HttpServer。HttpServer 说白了就是一个手机端的服务器，用来与用户（作者说的 client）交互，用户提出数据加载需求后，它会从不同的地方来获取数据源，如果本地没有会从网络中下载数据。它主要的作用是 hook 播放器的网络请求，进行数据的加载。它主要的类如图：



- KTVHCHTTPServer：是一个单例，用来管理 HttpServer 服务，负责开启或关闭服务；
- KTVHCHTTPConnection：它继承于 HTTPConnection，表示一个连接，它主要为 HttpServer 提供 Response。
- KTVHCHTTPRequest：一个请求，也就是一个数据模型；
- KTVHCHTTPResponse：一个 Response；
- KTVHCHTTPResponsePing：主要用来 ping 时的 Response；
- KTVHCHTTPURL：主要用来处理 URL，比如把原 Url 生成 proxy url；

其实 HttpServer 的关键点是在 KTVHCHTTPConnection 中下面这个方法，它是连接缓存模块的一个桥梁。使用 KTVHCHTTPRequest 和 KTVHCHTTPConnection 来生成 KTVHCHTTPResponse，**关键点在于生成这个 Response**。这段代码仅仅为了说明问题，有删减：



```

- (NSObject<HTTPResponse> *)httpResponseForMethod:(NSString *)method URI:(NSString *)uri {
    {
        KTVHCHTTPURL * URL = [KTVHCHTTPURL URLWithString:path];

        switch (URL.type)
        {
            case KTVHCHTTPURLTypePing:
            {
                return [KTVHCHTTPResponsePing responseWithConnection:self];
            }
            case KTVHCHTTPURLTypeContent:
            {
                KTVHCHTTPRequest * currentRequest = [KTVHCHTTPRequest requestWithOrigin:method URI:uri];

                KTVHCDataRequest * dataRequest = [currentRequest dataRequest];
                KTVHCHTTPResponse * currentResponse = [KTVHCHTTPResponse responseWithConnection:self dataRequest:dataRequest];

                return currentResponse;
            }
        }
        return nil;
    }
}

```

connetction.png

DataStroage

主要用来缓存数据，加载数据，也就是提供数据给 HttpServer。上面代码中关键的一句代码 [KTVHCHTTPResponse responseWithConnection:self dataRequest:dataRequest]，它会在在这个方法的内部使用 KTVHCDataStorage 生成一个 KTVHCDataReader，负责读取数据。生成 KTVHCDataReader 后通过 [self.reader prepare] 来准备数据源 KTVHCDataSourcer，这里主要有两个数据源，KTVHCDataFileSource 和 KTVHCDataNetworkSource，它实现了协议 KTVHCDataSourceProtocol。KTVHCDataNetworkSource 会通过 KTVHCDownload 下载数据。

需要说明一点，缓存是分片处理的



aonaotu-download.png

- KTVHCDataStorage: 是一个单例，它负责管理整个缓存，比如读取、保存和合并缓存；
- KTVHCDataReader: 主要用来读取数据；
- KTVHCDataRequest: 用来请求数据，表示一个请求；
- KTVHCDataResponse: 一个数据响应；
- KTVHCDataReader: 读取数据；
- KTVHCDataCacheItem: 缓存数据模型，表一个缓存项；
- KTVHCDataCacheItemZone: 缓存区，一个缓存项中会有多个缓存区，比如0-99, 100-299 等；
- KTVHCDataSourceer: 数据源中心，负责处理不同数据源，它包含有一个数据队列 KTVHCDataSourceQueue；
- KTVHCDataSourceQueue: 数据队列；
- KTVHCDataSourceProtocol: 一个协议，作为数据源时需要实现这个协议；
- KTVHCDataFileSource: 本地数据源，实现了 KTVHCDataSourceProtocol 协议；
- KTVHCDataNetworkSource: 网络数据源，实现了 KTVHCDataSourceProtocol 协议；
- KTVHCDataUnit: 数据单元，相当于一个缓存目录，比如一个视频的缓存；
- KTVHCDataUnitItem: 数据单元项，缓存目录下不同片段的缓存；
- KTVHCDataUnitPool: 数据单元池，它是一个单例，含有一个 KTVHCDataUnitQueue；
- KTVHCDataUnitQueue: 数据单元队列，保存了多个 KTVHCDataUnit，它会以 archive 的方式缓存到本地；

dataUnit.png

缓存目录构成

结构

05f68836443a1535b73bfcf3c2e86d99 这个是由请求的原 url，md5 后生成的字符串，其中它的子目录下会有多个文件，命名规则为：urlmd5_offset_数字。文件名最后一位数字因为

Data Storeage 同时支持并行和串行。在并行场景中极端情况可能遇到恰好同时存在两个相同 Offset 的 Network Source，用来保证并行加载的安全性（实际场景中也没遇到过，但在结构设计时把这部分考虑进去了） -- @程序员Single



- 05f68836443a1535b73bfcf3c2e86d99
- 05f68836443a1535b73bfcf3c2e86d99_0_0
- 05f68836443a1535b73bfcf3c2e86d99_196608_0
- 05f68836443a1535b73bfcf3c2e86d99_738108_0

沙盒目录：

cache.png

缓存策略

例如一次请求的 Range 为 0-999，本地缓存中已有 200-499 和 700-799 两段数据。那么会对应生成 5 个 Source，分别是：

- 网络: 0-199
- 本地: 200-499
- 网络: 500-699
- 本地: 700-799
- 网络: 800-999

日志系统

做音视频项目时，一个好的 Log 管理可以提高调试效率，而 KTVHTTPCache (https://link.jianshu.com?t=https://github.com/ChangbaDevs/KTVHTTPCache) 可以追踪到每一次异常的请求。而且回记录到一个 KTVHTTPCache.log 文件中。

```
KTVHTTPCache[818:16603] Proxy Start Success
KTVHTTPCache[818:16603] <KTVHCHTTPURL: 0x6040002349e0> : alloc
KTVHTTPCache[818:16603] KTVHCHTTPURL           : Ping, original url, KTVHCHTTPURL
KTVHTTPCache[818:16603] KTVHCHTTPURL           : proxy url, http://localhost:4983
KTVHTTPCache[818:16603] <KTVHCHTTPURL: 0x6040002349e0> : dealloc
KTVHTTPCache[818:16795] <KTVHCHTTPConnection: 0x6000001331a0> : alloc
KTVHTTPCache[818:16793] KTVHCHTTPConnection    : receive request, GET, /request-1
KTVHTTPCache[818:16793] <KTVHCHTTPURL: 0x604000238b00> : alloc
KTVHTTPCache[818:16793] KTVHCHTTPURL           : Server URI, /request-KTVHCHTTPUR
KTVHTTPCache[818:16793] <KTVHCHTTPResponsePing: 0x604000238b40> : alloc
KTVHTTPCache[818:16793] <KTVHCHTTPURL: 0x604000238b00> : dealloc
KTVHTTPCache[818:16793] KTVHCHTTPResponsePing  : conetnt length, 4
KTVHTTPCache[818:16793] KTVHCHTTPResponsePing  : read data length, 4, offset, 4 p
KTVHTTPCache[818:16793] KTVHCHTTPResponsePing  : check done, 1
KTVHTTPCache[818:16793] KTVHCHTTPResponsePing  : connection did close, 4, 4
KTVHTTPCache[818:16793] <KTVHCHTTPResponsePing: 0x604000238b40> : dealloc
KTVHTTPCache[818:16603] KTVHCHTTPServer        : ping result, 1
```

日志的定义主要在类 KTVHLog 中定义了一些宏。可以通过下面的方法开启日志：

```
// 开启控制台打印
[KTVHTTPCache logSetConsoleLogEnable:YES];
// 开启本地日志记录
[KTVHTTPCache logSetRecordLogEnable:YES];
```

特别说明一点可以控制到每个类是否打印：

```
KTVHLogEnable(HTTPServer, YES, YES)
```



主要一个日志的方法：

```
#define KTVHCLogging(target, console_log_enable, record_log_enable, ...)
if ((console_log_enable) || (record_log_enable)) \
{
    NSString * va_args = [NSString stringWithFormat:__VA_ARGS__]; \
    NSString * log = [NSString stringWithFormat:@"%@@ : %@", target, va_args]; \
    if (record_log_enable) { \
        NSLog(@"%@", log); \
    } \
    if (console_log_enable) { \
        NSLog(@"%@", log); \
    } \
}
```

总结

学习这个库总体来说比较耗时，但是能学到作者的思想，这里总结一下：

- 职责明确，每个类的作用定义明确；
- KTVHCDataFileSource 和 KTVHCDataNetworkSource，使用协议 KTVHCDataSourceProtocol 的方式实现不同的 Source，而不用继承，耦合性更低；
- 使用简单，内部定义复杂，缓缓相扣；
- 使用 NSLock 保证线程安全；
- 日志定义周全，调试更容易；

===== 我是有底线的 =====

喜欢我的文章，欢迎关注我的新浪微博 Lefe_x，我会不定期的分享一些开发技巧
(https://link.jianshu.com?t=http://www.weibo.com/5953150140/profile?rightmod=1&wvr=6&mod=personnumber&is_all=1)



Lefe (/u/7c787af04cd1) ♂
写了 54787 字，被 280 人关注，获得了 311 个喜欢
(/u/7c787af04cd1)

+ 关注


iOS开发，熟悉Node开发，正在修行，深爱React Native和微信小程序。正在踏入音视频这条不归路，希望...

♡ 喜欢 (/sign_in?utm_source=desktop&utm_medium=not-signed-in-like-button) | 26



更多分享

(<http://cwb.assets.jianshu.io/notes/images/2087803>)



下载简书 App ▶
随时随地发现和创作内容



(/apps/download?utm_source=nbc)

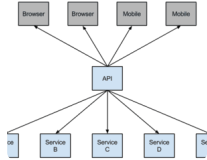


被以下专题收入，发现更多相似内容

-  iOS开发 (/c/7da05825783d?utm_source=desktop&utm_medium=notes-included-collection)
-  待读清单 (/c/b0b744c450e5?utm_source=desktop&utm_medium=notes-included-collection)
-  移动前沿 (/c/5aac963ca52d?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS 大神之路 (/c/9bafef2216bd?utm_source=desktop&utm_medium=notes-included-collection)
-  iOS技术交流收藏 (/c/cd81e3ea00e2?utm_source=desktop&utm_medium=notes-included-collection)
-  小知识点 (/c/24491a80859c?utm_source=desktop&utm_medium=notes-included-collection)
-  好东西 (/c/ea7dca305051?utm_source=desktop&utm_medium=notes-included-collection)


展开更多

(/p/46fd0faecac1?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
Spring Cloud (/p/46fd0faecac1?utm_campaign=maleskine&utm_conte...

Spring Cloud为开发人员提供了快速构建分布式系统中一些常见模式的工具（例如配置管理，服务发现，断路器，智能路由，微代理，控制总线）。分布式系统的协调导致了样板模式，使用Spring Cloud开发人员可...

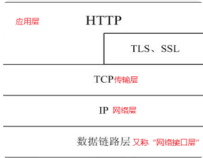
 卡卡罗2017 (/u/d90908cb0d85?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

HTTP协议 (/p/078fc3c75239?utm_campaign=maleskine&utm_content=...

深入浅出HTTP协议(WEB开发和面试必备) 1.基础概念篇 a.简介 HTTP是Hyper Text Transfer Protocol（超文本传输协议）的缩写。它的发展是万维网协会（World Wide Web Consortium）和Internet工作小组IETF（...


 半世韶华忆阑珊 (/u/5e34ebe06e5c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/93b9ef80dd1e?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)
HTTP协议基础 (/p/93b9ef80dd1e?utm_campaign=maleskine&utm_cont...


本文整理自MIN飞翔博客 [1] 1. 概念 协议是指计算机通信网络中两台计算机之间进行通信所必须共同遵守的规定或规则，超文本传输协议(HTTP)是一种通信协议，它允许将超文本标记语言(HTML)文档从Web服务...

 HoyaWhite (/u/995c0e221de2?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

iOS 第三方库、插件、知名博客总结 (/p/fa0b6f594c36?utm_campaign=m...


用到的组件1、通过CocoaPods安装项目名称项目信息 AFNetworking网络请求组件 FMDB本地数据库组件 SDWebImage多个缩略图缓存组件 UICKeyChainStore存放用户账号密码组件 Reachability监测网络状态...



 大灰狼的小绵羊哥哥 (/u/524ab6e6e423?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

API设计思考与实践 (/p/2fd232980f9b?utm_campaign=maleskine&utm...

API定义规范 本规范设计基于如下使用场景： 请求频率不是非常高：如果产品的使用周期内请求频率非常高，建议使用双通道的长连接，例如 socket/websocket，避免不必要的请求头浪费资源以及重建连接带来...

 有涯逐无涯 (/u/1547a852edf2?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/815ea0d2f6b2?



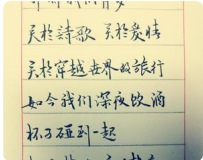
utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

【014】第三人称-Hush! (/p/815ea0d2f6b2?utm_campaign=maleskine&...

点这里，音乐和文章更配哦~ 快乐多少有一点，不过寂寞更强烈。难过时候不流泪，流泪也不算伤悲。男人必须保持风度，所以他只能在心里开怀高兴，终于又成为了普通朋友了，再也不会失去她了。刚听这首歌...


 Rose邢 (/u/202d821039fe?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

(/p/639106c8eb8a?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)

无题 (/p/639106c8eb8a?utm_campaign=maleskine&utm_content=note...

 蜗牛漫漫 (/u/e5c7e56aaaab?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

知道这些你也可以做聪明人（二）高手都是吃出来的 (/p/11e9e8e6b92c?ut...

饿了当然要吃东西。比如我们去饭店吃饭，从感到饥饿，到选择哪家饭店，再经过点菜，等菜，上菜，满意地开吃，最后结账离开。这个看似简单的过程却蕴含了很深的学问。聪明人在学习某一项技能的过程中会...

 心伊始晨未央 (/u/a03b3ab2b25d?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)


(/p/4df35caf0948?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation)


如何帮助孩子有效学习 (/p/4df35caf0948?utm_campaign=maleskine&ut...

自打孩子上中学以来，我家可以用兵荒马乱来形容，中学时期对于孩子来说是非常关键的，所以我们做父母的放弃了所有娱乐时间，时时刻刻陪在孩子身边，像看管小学生一样监督孩子学习，我们整日诚惶诚恐，...

 文心若雕 (/u/3fac20774350?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

蛀虫 (/p/ad6fac69eba1?utm_campaign=maleskine&utm_content=note...

土地上结满了痂 脓液流进海洋 鱼儿迁徙到天空 云朵掉落凡间 动物搬进有门的屋 人们蜷缩在树干里 偷偷满足自己的胃 顺便笨拙地躲避着啄木鸟的喙

 洲十三 (/u/90b07c14b233?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommendation)

