文件下载详细设计

|  |  |  |  |
| --- | --- | --- | --- |
| 作者 |  | | |
| 修订记录 | V1.2 | 2016/1/22 | 设计再详细点 |
| V1.1 | 2016/1/18 | 添加逻辑和辅助功能定义 |
| V1.0 | 2016/1/14 | 创建 |

# 封装要求

该模块封装为framework , 名称为VGNetworkFramework。要求该模块具有功能划分清晰、易扩展、易于维护的特点。同时，该模块的注释要详尽，要注明逻辑过程。

该模块将使用日志记录系统，用Cocoapods集成，便于性能分析及bug复现。

# 需求

使用者给出队列名和资源的url就能过获取到网络资源，在获取的过程中能获取实时的状态，并能进行暂停、启动和删除任务操作，还能够清理缓存的文件。

# 应用场景详细分析

## 下载

用户点击下载，根据几种情况，在回调中返回不同的结果：

* 已经下载过，且没有过期，回调出本地文件地址
* 已经下载过，但是文件过期，重新下载，并在完成后返回结果
* 没有下载过，开始下载，并在完成后回调出本地文件地址
* 下载过程出错，返回错误结果

## 获取状态

下载过程中，用户获取下载状态。

* 当正在进行下载时，返回的下载对象中包含下载状态
* 当已经下载完成后，返回的下载对象中包含文件信息

## 操作

* 重启、暂停、删除任务
* 清除文件

# 物理结构



# 详细逻辑



# 详细实现

### 单任务下载

#### 类名

VGTaskDownload

#### 接口

##### 下载代理

/\*\*

\* 任务代理

\*/

@protocol protocol\_downloadTask <NSObject>

/\*\*

\* 任务下载中抛出信息

\* 可能是任务完成百分比...

\* @param error 错误信息

\*/

- (void) taskExecutingWithInfo:(VGTaskExecutingInfo \*)executingInfo;

/\*\*

\* 任务报错

\*

\* @param error 错误信息

\*/

- (void) taskDidFaildWithError:(NSError \*)error;

/\*\*

\* 任务正常结束

\*

\* @param strFilePath 抛出下载后的文件路径

\*/

- (void) taskDidFinishedSuccessed:(NSString\*)strFilePath;

@end

##### 下载类

###### 属性

@interface VGTaskDownload : NSObject<Protocol\_VGNetworkStatus>

/\*\*

\* 下载任务类型：上传 或 下载，暂时只有下载

\*/

@property(nonatomic , assign , readonly) VGTASK\_TYPE m\_taskType;

/\*\*

\* 下载文件网络路径

\*/

@property(nonatomic , strong , readonly) NSString \*m\_strSourceUrl;

/\*\*

\* 下载文件信息

\*/

@property(nonatomic , strong) VGBaseFileInfo \*m\_fileInfo;

/\*\*

\* 下载过程中的状态信息

\*/

@property(nonatomic , strong) VGTaskExecutingInfo \*m\_fileExecutingInfo;

/\*\*

\* 下载文件保存方式

\*/

@property(nonatomic , assign) VGFILEPOSITIONTYPE m\_saveType;

/\*\*

\* 下载状态

\*/

@property(nonatomic , assign) TASK\_STATUS m\_taskStatus;

/\*\*

\* 起始时间

\*/

@property(nonatomic , strong) NSDate \*m\_dateStart;

/\*\*

\* 结束时间

\*/

@property(nonatomic , strong) NSDate \*m\_dateFinish;

/\*\*

\* 任务代理

\*/

@property(nonatomic , strong) id<protocol\_downloadTask> m\_protocol\_baseTask;

###### 接口

创建任务

/\*\*

\* 创建任务并启动

\*

\* @param strUrl 网络资源路径

\*/

- (instancetype) initTaskAndStartwithUrl:(NSString\*)strUrl;

根据url，使用[文件存储管理类](#_文件存储管理)，创建临时文件；然后创建下载任务，下载过程中随时抛出状态，如果[网络变化](#_网络状态)时，根据情况进行暂停操作。

* 下载时随时检测网络，并根据网络更改任务状态
* 下载过程中不停地写文件和更改下载大小到数据库
* 下载完成后更改文件名和位置

暂停任务

/\*\*

\* 暂停任务

\*/

-(void)taskPause;

* 停止网络下载
* 数据同步到数据库
* 更改任务状态

重新开始任务

/\*\*

\* 重新开始任务

\*/

-(void)taskRestart;

* 开始下载
* 更改任务状态

记录进度

/\*\*

\* 根据“队列名”和“url” 记录下载进度

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\* @param length 本次下载长度

\*

\*/

- (void) progressDownloadedLength:(double) length;

每次写完文件，要在[下载列表管理类](#_下载列表管理)中记录下载进度，以便断点续传。把参数中的长度累加到记录中。

### 任务下载管理

本类是现有任务的管理类。

实际上，当前所有任务列表，是存在数据库中的，所以要与数据库高度统一。那么就另外需要一个加载数据库任务的管理类。

任务都是先写到数据库，然后再同步到该 任务下载管理 类中。

#### 类名

VGTaskDownloadManager

#### 属性

/\*\*

\* 所有任务

\*/

@property (nonatomic, strong) NSDictionary \*listTaskQueue;

通过

#### 接口

/\*\*

\* 下载管理中心--单例

\*

\* @return 实例对象

\*/

+ (instancetype) sharedManagerCenter;

##### 是否需要创建下载任务

通过 [文件是否过期管理类](#_文件是否过期管理) VGFileOverduleCheck的静态接口实现，只需要传入任务参数。

+ (BOOL) fileOverDuleCheckWithFileInfo:(VGBaseFileInfo\*)fileInfo

##### 要创建的任务是否已经存在列表

/\*\*

\* 根据“队列名”和“url” 检查任务是否在任务列表中

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\* @return 下载任务对象

\*/

- (bool \*) checkTaskInQueueWithUrl:(NSString \*)strUrl queue:(NSString \*)strQueue;

遍历所有任务，包括其它队列，如果队列中已有该任务，则返回其它队列的任务，否则，创建该任务。

如果队列中已有任务，但是没有执行，则启动任务。启动任务时，更改任务的状态。

##### 创建一个下载任务

/\*\*

\* 根据“队列名”和“url” 创建/获取/重启一个下载任务

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\* @return 下载任务对象

\*/

- (VGTaskDownload \*) createDownloadTaskWithUrl:(NSString \*)strUrl queue:(NSString \*)strQueue;

* 创建任务要先放到数据库中
* 要先创建任务列表，和任务信息，任务信息有name\length\downloadedLength\url
* 数据库中如果已经有该任务信息，说明该任务以前处理过，也就是未完成任务，那就断点下载
* 如果任务完成，数据库中的记录应该清除，一方面不影响新的下载；另一方面，减少数据量，减少查询时间。如果非要保存下载记录，可以把下载记录保存到另外的一个表中
* 根据数据记录创建任务的起始进度
* 下载过程中，同步任务进度
* 下载完成后，先把文件进行更名拷贝，然后在文件表中写入name和url
* 返回完整路径



##### 添加一个任务到下载列表

/\*\*

\* 根据“队列名”和“url” 添加一个下载任务到下载列表

\*

\* @param task下载任务

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\*/

- (void) addDownloadTaskWithUrl:( VGTaskDownload \*)task queue:(NSString \*)strQueue ;

* 任务队列是2级目录，有个专门的下载列表来管理VGDownloadList,如果没有该队列则创建，删除没用的队列的操作只在程序退出之后。要不一会建一会删，没有必要。先把任务信息放到数据库中。任务信息有，name,url,length,downloadedLength
* 然后再创建任务，需要同步下载信息到数据库中，同步不要太频繁，同步downloadedSize

##### 暂停任务

/\*\*

\* 根据“队列名”和“url” 暂停一个下载任务

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\* @return 下载任务对象

\*/

- (void ) pauseDownloadTaskWithUrl:(NSString \*)strUrl queue:(NSString \*)strQueue;

* 查询列表，找到任务；
* 暂停任务；
* 刷新数据库
* 刷新任务列表（请求数据库）

##### 删除任务

/\*\*

\* 根据“队列名”和“url” 删除一个下载任务

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\*/

- (void) deleteDownloadTaskWithUrl:(NSString \*)strUrl queue:(NSString \*)strQueue;

执行步骤：

* 查询列表，如果找到任务，则继续下步操作
* 删除列表中的任务
* 删除数据库中的任务，如果某个队列中没有任务，则删除队列
* 刷新数据库列表

##### 查找任务

/\*\*

\* 根据“队列名”和“url” 查询一个下载任务

\*

\* @param strUrl 下载链接

\* @param strQueue 下载队列名称--为自定义队列名称

\*

\*/

- (VGTaskDownload \*) findDownloadTaskWithUrl:(NSString \*)strUrl queue:(NSString \*)strQueue;

查询本类的属性就可以获取该任务。

* 队列名和url 都可以在任务中查询到。
* 先根据队列名找到任务队列，再根据url在队列找到对应任务
* 找到则返回，找不到返回nil
* 如果返回的任务不为空，但是返回之后，立即执行完毕，被回收了。这时候，再操作该任务，就会出现崩溃？不过，根据ARC的原则，如果还有引用，则不会释放该资源，所有不要担心这个没问题。

##### 返回所有任务

/\*\*

\* 查询所有下载任务

\*

\* @param task下载任务

\* @param strQueue 下载队列名称--为自定义队列名称

\* @return 字典，key是列表名，值是任务数组

\*/

- (NSDictionary \*) listDownloadTask；

字典里包含队列，队列里是所有的任务。在确定任务执行完毕后，要执行回收工作，因为，如果一直不回收，就会导致内存不停地增长。当然，如果考虑内存碎片问题，创建线程池或叫任务池，可以考虑在下个版本优化。

### 文件存储管理

该模块是单纯的文件操作，不再做详细设计。

#### 类名

VGFileDownloadManager

#### 属性

/\*\*

\* 文件总大小

\*/

@property (nonatomic, assign) double m\_dlFileSize;

/\*\*

\* 完成的大小

\*/

@property (nonatomic, assign) double m\_dlFinishedSize;

#### 接口

/\*\*

\* 根据“url” 创建临时下载文件

\*

\* @param strUrl下载任务链接

\* @

\*/

- (NSString \*) createFileWithUrl:( NSString \*)strUrl size:(double)size;

/\*\*

\* 根据“url”的MD5值生成文件名称

\*

\* @param strUrl下载任务链接

\* @return 字典，key是列表名，值是任务数组

\*/

- (NSString \*) makeNameWithUrl:( NSString \*)strUrl;

/\*\*

\* 写入数据

\*

\* @param data 数据

\* @param length数据长度

\*

\*/

- (void ) writeFileWithData:( NSData\*)data length(long)length;

/\*\*

\* 文件下载完成后重命名文件，并且移动文件，然后把文件名称放到文件列表中

\*

\*/

- (void ) renameAndMoveFile;

/\*\*

\* 删除文件

\*

\*/

- (void ) deleteFileWithUrl(NSString\*)strUrl;

### 本地文件列表管理

#### 类名

VGFileDownloadListManager

#### 属性

#### 接口

##### 载入列表

/\*\*

\* 写入数据

\*

\* @param data 数据

\* @param length数据长度

\*

\*/

- (void ) writeFileWithData:( NSData\*)data length(long)length;

### 文件是否过期管理

#### 类名

VGFileOverduleManager

#### 接口

给出VGBaseFileInfo文件信息，判断下载的文件是否已经过期、不是服务器上最新的文件。这时候，我还应该传入服务器文件的信息。

通过文件名+大小的方式，此方式并不能保证完全正确，但是把该功能单独封装，统一接口，这样也方便以后升级。

/\*\*

\* 没有该文件，也视为过期；有文件，但是大小不同时，为过期文件，需要重新下载

\*

\* @param fileInfo 包含文件的名称和大小等信息

\*

\* @return

\*/

+ (BOOL) fileOverDuleCheckWithFileInfo:(VGBaseFileInfo\*)fileInfo{

return NO;

}

### 网络状态

#### 类名

VGNetworkStatus

#### 接口

分析网络是wifi、慢速、还是其它。并要检测变化。



网络变化时，根据下载文件的类型的不同，可能有不同的操作。这部分的功能要单独的模块来处理。

网络分为有网络和无网络两类，有网络有分为几类。