# 文档前言

## 基本描述

俗话说的好：“好记性，不如烂笔头”。说明的一个道理就是纪录的重要性，哪怕是一个记忆天才，纪录在其他地方，可以随时看到也是非常重要的。何况我是如此一个平凡的人啊！谦虚学习，踏踏实实，不能好高骛远，天道酬勤才是王道。

本文档旨在纪录开发过程中遇到的一些重要点、难点，也能帮助我纪录我的成长过程。

少波，好好纪录，这是很重要的一个事情，不要觉得我写了一些知道的东西，没什么价值，磨刀不误砍柴工。

## 内容概括

本文包括如下内容：iOS开发、设计模式、数据结构、java服务器开发、前段开发等，若有新技术的添加会及时的添加到文档中。

IOS开发部分

# 网络部分

## 实现离线断点下载功能（2016年4月19日星期二）

### 基本描述：

实现大文件的离线重新下载，在程序运行过程可以暂停、继续以及在程序app下一次启动后，可以继续下载，上次下载的文件进行保存，下次从上次下载完成的部分继续下载。

基本的实际案例：百度音乐下载、喜马拉雅音乐下载、百度云文件的下载。

其实本人认为，网络部分断点续传算是难度和复杂度都比较高的部分。

### 解决方案

#### 解决方案理论

用户发送下载文件请求（如下载音乐、大文件等），当用户首次发送下载文件，将请求头部设置为请求的文件二进制位置为0，在下载过程中不断将下载的二进制文件存储在本地，并且记录下载量，一旦用户暂停下载，app记录下最后下载完成的数量以及下载请求的位置，从多少字节开始下载。服务器端则根据客户端的请求，从head区域的文件请求位置，开始传输二进制数据，发送给客户端，在完成总的文件下载后，客户端取消任何关于本文件的下载信息。

当前，一般的网络处理，都会用到AFNetworking 网络请求库，但是此框架对实现离线形式（重启app，从上次的下载点继续下载）的断点续传支持不好。因此我此处使用到的是apple系统api原生的网络请求方式。

#### 解决实践

1. 用到NSURLSession ->session，调用含有代理设置的初始化方法:

+ (NSURLSession \*)sessionWithConfiguration:(NSURLSessionConfiguration \*)configuration delegate:(nullable id <NSURLSessionDelegate>)delegate delegateQueue:(nullable NSOperationQueue \*)queue;

2. 使用NSMutableRequest->request，设置头部文件为：

//设置表头信息

NSString \*rang = [NSString stringWithFormat:”@bytes=%zd-”,currrentSize];

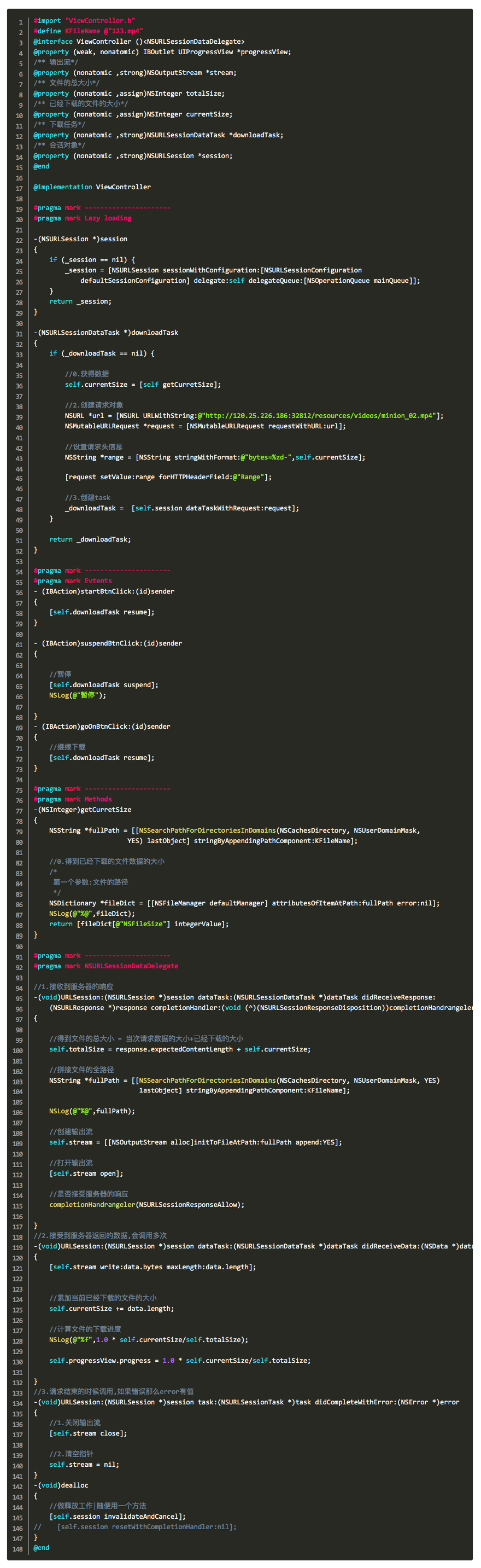
[request setValue:rang forHTTPHeanderField:”@Range”];

3.调用创建task方法

//创建task

NSURLSessionDataTask \*task = [session dataTaskWithRequest:request];

下图是参考别人写的，我觉的很好，直接贴上来，如果觉的图片不清楚，可以通过放大文档的形式查看，如下图：



# 多线程

## NSThread

是三种方法里面相对轻量级的，但需要管理线程的生命周期、同步、加锁问题，这会导致一定的性能开销

### 优缺点

1.优点：NSThread比其他两种多线程方案较轻量级，更直观地控制线程对象

2.缺点：需要自己管理线程的生命周期，线程同步。线程同步对数据的加锁会有一定的系统开销

## Cocoa Operations

是基于OC实现的，NSOperation以面向对象的方式封装了需要执行的操作，不必关心线程管理、同步等问题。NSOperation是一个抽象基类，iOS提供了两种默认实现：NSInvocationOperation和NSBlockOperation，当然也可以自定义NSOperation

## GCD（grand gentral dispatch）

简称GCD，iOS4才开始支持)：提供了一些新特性、运行库来支持多核并行编程，它的关注点更高：如何在多个cpu上提升效率

## 同步锁（NSLock）

## @ synchronized

使用方法；

在防止多个线程访问的对象和代码处使用如下：

@synchronized (\_elements) {

[\_elements addObject:element];

}

”synchronized block” 与 [\_lock lock] 和 [\_lock unlock] 效果相同。你可以把它当成是锁住 self，仿佛 self 就是个 NSLock。锁在左括号 { 后面的任何代码运行之前被获取到，在右括号 } 后面的任何代码运行之前被释放掉。这爽就爽在妈妈再也不用担心我忘记调用 unlock 了！

其实上面的\_elements可以设置为当前的类对象，self，关键是通过@synchronized关键字会将过好内的代码进行加锁，等代码执行完成后才允许其他线程访问这些代码。

# 数据结构

# 设计模式

## 设计原则

### 单一职责原则

一个类只负责一个功能领域中相应的职责

### 开闭原则

一个类或者软件实体尽量实现对扩展开发，对修改关闭

### 里氏代换

所有引用基类（父类）的地方，都能够使用子类的对象

### 依赖倒转

具体依赖抽象，而不是抽象依赖具体。

### 接口隔离

### 合成复用

### 迪米特法则

## 创建型