[SDWebImage缓存图片的机制(转)](http://blog.csdn.net/zhun36/article/details/8900327)

分类： [iOS Dev](http://blog.csdn.net/zhun36/article/category/1371866)2013-05-08 14:54 1738人阅读 [评论](http://blog.csdn.net/zhun36/article/details/8900327#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/zhun36/article/details/8900327#report)

[**SDWebImage**](https://github.com/rs/SDWebImage)**是一个很厉害的图片缓存的框架。既ASIHttp+AsyncImage之后，我一直使用AFNetworking集成的**[**UIImageView+AFNetworking.h**](https://github.com/AFNetworking/AFNetworking/blob/master/AFNetworking/UIImageView%2BAFNetworking.h)**，但后者对于图片的缓存实际应用的是NSURLCache自带的cache机制。而NSURLCache每次都要把缓存的raw  data 再转化为UIImage，就带来了数据处理和内存方面的更多操作。具体的比较在**[**这里**](https://github.com/rs/SDWebImage/wiki/How-is-SDWebImage-better-than-X%3F)**。**

SDWebImage提供了如下三个category来进行缓存。

* [MKAnnotationView(WebCache)](http://hackemist.com/SDWebImage/doc/Categories/MKAnnotationView(WebCache).html)
* [UIButton(WebCache)](http://hackemist.com/SDWebImage/doc/Categories/UIButton(WebCache).html)
* [UIImageView(WebCache)](http://hackemist.com/SDWebImage/doc/Categories/UIImageView(WebCache).html)

以最为常用的UIImageView为例:

1. UIImageView+WebCache:  setImageWithURL:placeholderImage:options: 先显示 placeholderImage ，同时由SDWebImageManager 根据 URL 来在本地查找图片。
2. SDWebImageManager: downloadWithURL:delegate:options:userInfo: SDWebImageManager是将UIImageView+WebCache同SDImageCache链接起来的类， SDImageCache： queryDiskCacheForKey:delegate:userInfo:用来从缓存根据CacheKey查找图片是否已经在缓存中
3. 如果内存中已经有图片缓存， SDWebImageManager会回调SDImageCacheDelegate : imageCache:didFindImage:forKey:userInfo:
4. 而 UIImageView+WebCache 则回调SDWebImageManagerDelegate:  webImageManager:didFinishWithImage:来显示图片。
5. 如果内存中没有图片缓存，那么生成 NSInvocationOperation 添加到队列，从硬盘查找图片是否已被下载缓存。
6. 根据 URLKey 在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。
7. 如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小，会先清空内存缓存）。SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo:。进而回调展示图片。
8. 如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片，回调 imageCache:didNotFindImageForKey:userInfo:。
9. 共享或重新生成一个下载器 SDWebImageDownloader 开始下载图片。
10. 图片下载由 NSURLConnection 来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。
11. connection:didReceiveData: 中利用 ImageIO 做了按图片下载进度加载效果。
12. connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。
13. 图片解码处理在一个 NSOperationQueue 完成，不会拖慢主线程 UI。如果有需要对下载的图片进行二次处理，最好也在这里完成，效率会好很多。
14. 在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。
15. imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。
16. 通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。
17. 将图片保存到 SDImageCache 中，内存缓存和硬盘缓存同时保存。
18. 写文件到硬盘在单独 NSInvocationOperation 中完成，避免拖慢主线程。
19. 如果是在iOS上运行，SDImageCache 在初始化的时候会注册notification 到 UIApplicationDidReceiveMemoryWarningNotification 以及  UIApplicationWillTerminateNotification,在内存警告的时候清理内存图片缓存，应用结束的时候清理过期图片。
20. SDWebImagePrefetcher 可以预先下载图片，方便后续使用。