SDWebImage原理

1. 主要功能
   * 1. 提供UIImageView的一个分类，以支持网络图片的加载与缓存管理
     2. 一个异步的图片加载器
     3. 一个异步内存+磁盘图片缓存器
     4. 支持GIF图片
     5. 支持webp图片
     6. 后台图片解压缩处理
     7. 确保同一个URL的图片不被下载多次
     8. 确保虚假的URL不会被反复加载
     9. 确保下载及缓存时，主线程不被阻塞
     10. SDWebimage下载的核心其实就是利用NSURLSession对象来加载数据。每个图片的下载都由一个operation操作来完成，并将这些操作放到一个操作队列中。这样可以实现并发下载图片
2. 缓存

SDWebimage提供了对图片缓存的支持，该功能是由SDImageCache类来完成的，该类负责处理内存缓存及一个可选的磁盘缓存。其中磁盘缓存的写操作是异步，这样就不会对UI操作造成影响。

内存缓存的处理是使用类似NSCache对象来实现的。NSCache是一个类型有集合的容器。它存储key-value对，这点类型NSDictionary类。通常使用缓存来临时存储短时间使用但创建昂贵的对象。重用这些对象可以优化性能，因为他们的值不需要重新计算。在内存吃紧时，会丢弃一些不重要的对象

磁盘缓存的处理则是是使用NSFileManager对象来实现的。图片存储的位置是位于Cache文件夹

SDWebImage定义了一个串行队列，来异步存储图片，SDImageCache提供了大量方法来缓存、获取、移除与清空图片。而对于每个图片，为了方便的在内存或磁盘中对它进行这些操作，我们需要一个key值来索引它。在内存中，我们将其作为NSCache的key值，而在磁盘中，我们使用这个key作为图片的文件名。

SDWebImage的主要任务就是图片的下载和缓存。为了支持这些操作，主要使用了一下知识点

Dispatch\_barrier\_sync函数：该方法用于对操作设置屏幕，确保在执行完任务后才会执行后续操作，常用于确保类的线程安全性操作

NSMutableURLRequest:用于创建网络请求对象，配置请求头等信息

NSOperation及NSOperationQueue：操作队列是ObjC中的一种高级的并发处理方法，基于GCD来实现的，相对于GCD来说，操作队列的优点是可以取消在任务处理队列中的任务，在管理操作间的依赖关系相对容易些。在SDWebImage中，我们就看到了它是如何使用依赖将下载顺序设置成先进先出的顺序

NSURLSession类，用于网络请求及响应处理，开启一个后台任务

清理缓存图片的策略：特别是最大缓存空间大小的设置。如果所有缓存文件的总大小超过该设置，则会按照文件最后修改时间的逆序，以每次一半的递归来移除那些过早的文件，知道缓存的实际大小小于我们设置的最大使用空间

对图片的解压缩操作：这一操作可以查看SDWebImageDecoder的+decodedImageWithImage方法的实现

对GIF图片的处理

对WebPage图片的处理

图片下载—>显示：

异步下载：NSOperation+操作队列

业务逻辑：图片缓存（防止图片错位，提升效率）+ 操作缓存（防止重复创建操作）+ 沙盒缓存（磁盘缓存）

SDWebImage的UIImageView的一个分类提供了一句话代码接口实现异步图片的下载，缓存，显示

1. 框架SDWebImage的缓存机制
2. UIImageView+WebCache：setImageWithURL:placeholderImage:options:先显示placeholderImage，同时由SDWebImageManager根据URL来查询本地图片
3. SDWebImageManager:downloadWithURL:delegate:options:userInfo: SDWebImageManager是将UIImageView+WebCache同SDImageCache连接起来的类，SDImageCache:queryDiskCacheForKey:delegate:userInfo:用来从缓存中根据CacheKey查找图片是否已经在缓存中
4. 如果内存中已经有图片缓存，SDWebImageManager会回调SDImageCacheDelegate:imageCache:didFindImage:forKey:userInfo:
5. 而UIImageView+WebCache则回调SDWebImageManagerDelegate:webImageManager:didFinishWithImage:来显示图片
6. 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
7. 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:
8. 如果上一步操作从硬盘读取到了图片，将图片添加到内存缓存中（如果缓存过小，先清空缓存）。SDImageCacheDelegate回调imageCache:didFindImage:forKey:userInfo:进而回调展示图片
9. 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:
10. 共享或重新生成一个下载器SDWebImageDownloader开始下载图片
11. 图片下载由NSURLSession实现，通过相关的Delegate方法来判断图片下载中、下载完成和下载失败
12. session:task:didReceiveData:中利用ImageIO做了按图片下载进度加载效果
13. session:task: didCompleteWithError:数据下载完成后交给SDWebImageDecoder做图片解码处理
14. 图片解码处理在一个NSOperationQueue完成，不会拖慢主线程UI。
15. 接着在主线程notifyDelegateOnMainThreadWithInfo:宣告解码完成，通过SDWebImageDownloader的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，回调给需要的地方显示图片
17. 讲图片保存到SDImageCache中，内存缓存和磁盘缓存同时保存
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