返回目录:全网各大厂 iOS 面试题-题集大全

分析下 SDWebImage

1.SDWebImage 加载图片的流程

1.入口 setImageWithURL:placeholderImage:options: 会先把 placeholderImage 显示，然后 SDWebImageManager 根据 URL 开始处理图片。

2.进入 SDWebImageManager-downloadWithURL:delegate:options:userInfo:，交给 SDImageCache 从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:.

3.先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo: 到 SDWebImageManager。

4.SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache 等前端展示图片。

5.如果内存缓存中没有，生成 NSInvocationOperation 添加到队列开始从硬盘查找图片是否已经缓存。

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。写文件到硬盘也在以单独 NSInvocationOperation 完成，避免拖慢主线程。

18.SDImageCache 在初始化的时候会注册一些消息通知，在内存警告或退到后台的时候清理内存图片缓存，应用结束的时候清理过期图片。

19.SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。

20.SDWebImagePrefetcher 可以预先下载图片，方便后续使用。

SDWebImage 原理图

SDWebImage 原理图

SDWebImage 原理图

SDWebImage 原理图

2. SDImageCache 是怎么做数据管理的？

SDImageCache 分两个部分，一个是内存层面的，一个是硬盘层面的。内存层面的相当是个缓存器，以 Key-Value 的形式存储图片。当内存不够的时候会清除所有缓存图片。用搜索文件系统的方式做管理，文件替换方式是以时间为单位，剔除时间大于一周的图片文件。当 SDWebImageManager 向 SDImageCache 要资源时，先搜索内存层面的数据，如果有直接返回，没有的话去访问磁盘，将图片从磁盘读取出来，然后做 Decoder，将图片对象放到内存层面做备份，再返回调用层。

3.内部做 Decoder 的原因 (典型的空间换时间)

由于 UIImage 的 imageWithData 函数是每次画图的时候才将 Data 解压成 ARGB 的图像，所以在每次画图的时候，会有一个解压操作，这样效率很低，但是只有瞬时的内存需求。为了提高效率通过 SDWebImageDecoder 将包装在 Data 下的资源解压，然后画在另外一张图片上，这样这张新图片就不再需要重复解压了

返回目录:全网各大厂 iOS 面试题-题集大全

更多精选大厂 · iOS 面试题答案 PDF 文集

*

获取加小编的 iOS 技术交流圈：**937 194 184**，直接获取