返回目录:全网各大厂 iOS 面试题-题集大全

SDWebImage 原理

SDWebImage

一个为 UIImageView 提供一个分类来支持远程服务器图片加载的库。

功能简介：

1、一个添加了 web 图片加载和缓存管理的 UIImageView 分类
2、一个异步图片下载器
3、一个异步的内存加磁盘综合存储图片并且自动处理过期图片
4、支持动态 gif 图
5、支持 webP 格式的图片
6、后台图片解压处理
7、确保同样的图片 url 不会下载多次
8、确保伪造的图片 url 不会重复尝试下载
9、确保主线程不会阻塞

工作流程

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。connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。

12、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。

13、在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。

14、通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。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。

15、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。

16、SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。

17、SDWebImagePrefetcher 可以预先下载图片，方便后续使用。

源码分析

主要用到的对象

一、图片下载

1、 SDWebImageDownloader * 1.单例，图片下载器，负责图片异步下载，并对图片加载做了优化处理

- 2.图片的下载操作放在一个 NSOperationQueue 并发操作队列中，队列默认最大并发数是 6
- 3.每个图片对应一些回调（下载进度，完成回调等），回调信息会存在 downloader 的 URLCallbacks（一个字典，key 是 url 地址，value 是图片下载

回调数组）中，URLCallbacks 可能被多个线程访问，所以 downloader 把下载任务放在一个 barrierQueue 中，并设置屏障保证同一时间只有一个线程访问 URLCallbacks。，在创建回调 URLCallbacks 的 block 中创建了一个 NSOperation 并添加到 NSOperationQueue 中。

- 4.每个图片下载都是一个 operation 类，创建后添加到一个队列中，SDWebimage 定义了一个协议 SDWebImageOperation 作为图片下载操作的基础协议，声明了一个 cancel 方法，用于取消操作。

```
@protocol SDWebImageOperation <NSObject>
-(void)cancel;
@end
```

- 5.对于图片的下载，SDWebImageDownloaderOperation 完全依赖于 NSURLConnection 类，继承和实现了 NSURLConnectionDataDelegate 协议的方法

```
connection:didReceiveResponse:
connection:didReceiveData:
connectionDidFinishLoading:
connection:didFailWithError:
connection:willCacheResponse:
connectionShouldUseCredentialStorage:
-connection:willSendRequestForAuthenticationChalleng
```

`-connection:didReceiveData:`方法，接受数据，创建一个 `CGImageSourceRef` 对象，在首次获取数据时（图片 `width`，`height`），图片下载完成之前，使用 `CGImageSourceRef` 对象创建一个图片对象，经过缩放、解压操作生成一个 `UIImage` 对象供回调使用，同时还有下载进度处理。

注：缩放：`SDWebImageCompat` 中 `SDScaledImageForKey` 函数

解压：`SDWebImageDecoder` 文件中 `decodedImageWithImage`

2、SDWebImageDownloaderOption

- 1.继承自 NSOperation 类，没有简单实现 main 方法，而是采用更加灵活的 start 方法，以便自己管理下载的状态

- 2.start 方法中创建了下载使用的 NSURLConnections 对象，开启了图片的下载，并抛出一个下载开始的通知，

- 3.小结：下载的核心是利用 NSURLSession 加载数据，每个图片的下载都有一个 operation 操作来完成，并将这些操作放到一个操作队列中，这样可以实现图片的并发下载。

3、SDWebImageDecoder（异步对图片进行解码）

二、缓存

减少网络流量，下载完图片后存储到本地，下载再获取同一张图片时，直接从本地获取，提升用户体验，能快速从本地获取呈现给用户。 SDWebImage 提供了对图片进行了缓存，主要由 SDImageCache 完成。该类负责处理内存缓存以及一个可选的磁盘缓存，其中磁盘缓存的写操作是异步的，不会对 UI 造成影响。

1、内存缓存及磁盘缓存 * 1.内存缓存的处理由 NSCache 对象实现，NSCache 类似一个集合的容器，它存储 key-value 对，类似于 nsdictionary 类，我们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新能，同时这些对象对于程序来说不是紧要的，如果内存紧张就会自动释放。

- 2.磁盘缓存的处理使用 NSFileManager 对象实现，图片存储的位置位于 cache 文件夹，另外 SDImageCache 还定义了一个串行队列来异步存储图片。
- 3.SDImageCache 提供了大量方法来缓存、获取、移除及清空图片。对于图片的索引，我们通过一个 key 来索引，在内存中，我们将其作为 NSCache 的 key 值，而在磁盘中，我们用这个 key 值作为图片的文件名，对于一个远程下载的图片其 url 实作为这个 key 的最佳选择。

2、存储图片 先在内存中放置一份缓存，如果需要缓存到磁盘，将磁盘缓存操作作为一个 task 放到串行队列中处理，会先检查图片格式是 jpeg 还是 png，将其转换为响应的图片数据，最后吧数据写入磁盘中（文件名是对 key 值做 MD5 后的串）

3、查询图片 内存和磁盘查询图片 API：

```
- (UIImage *)imageFromMemoryCacheForKey:(NSString *)key;
- (UIImage *)imageFromDiskCacheForKey:(NSString *)key;
```

查看本地是否存在 key 指定的图片，使用一下 API：

```
- (NSOperation *)queryDiskCacheForKey:(NSString *)key done:(SDWebImageQ
ueryCompletedBlock)doneBlock;
```

4、移除图片 移除图片 API：

```
- (void)removeImageForKey:(NSString *)key;
- (void)removeImageForKey:(NSString *)key withCompletion:(SDWebImageNoP
aramsBlock)completion;
- (void)removeImageForKey:(NSString *)key fromDisk:(BOOL)fromDisk;
- (void)removeImageForKey:(NSString *)key fromDisk:(BOOL)fromDisk withC
ompletion:(SDWebImageNoParamsBlock)completion;
```

5、清理图片（磁盘）

清空磁盘图片可以选择完全清空和部分清空，完全清空就是吧缓存文件夹删除。

```
- (void)clearDisk;
- (void)clearDiskOnCompletion:(SDWebImageNoParamsBlock)completion;
```

部分清理 会根据设置的一些参数移除部分文件，主要有两个指标：文件的缓存有效期（maxCacheAge：默认是 1 周）和最大缓存空间大小（maxCacheSize：如果所有文件大小大于最大值，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哪些过早的文件，知道缓存文件总大小小于最大值），具体代码参考-(void)cleanDiskWithCompletionBlock；

6、小结 SDImageCache 处理提供以上 API，还提供了获取缓存大小，缓存中图片数量等 API， 常用的接口和属性：

（1）-getSize ：获得硬盘缓存的大小

（2）-getDiskCount ： 获得硬盘缓存的图片数量

（3）-clearMemory ： 清理所有内存图片

（4）- removeImageForKey:(NSString *)key 系列的方法 ： 从内存、硬盘按要求指定清除图片

（5）maxMemoryCost ： 保存在存储器中像素的总和

（6）maxCacheSize ： 最大缓存大小 以字节为单位。默认没有设置，也就是为 0，而清理磁盘缓存的先决条件为 self.maxCacheSize > 0，所以 0 表示无限制。

（7）maxCacheAge ： 在内存缓存保留的最长时间以秒为单位计算，默认是一周

三、SDWebImageManager

实际使用中并不直接使用 SDWebImageDownloader 和 SDImageCache 类对图片进行下载和存储，而是使用 SDWebImageManager 来管理。包括平常使用 UIImageView+WebCache 等控件的分类，都是使用 SDWebImageManager 来处理，该对象内部定义了一个图片下载器（SDWebImageDownloader）和图片缓存（SDImageCache）

```
@interface SDWebImageManager : NSObject

@property (weak, nonatomic) id <SDWebImageManagerDelegate> delegate;

@property (strong, nonatomic, readonly) SDImageCache *imageCache;
@property (strong, nonatomic, readonly) SDWebImageDownloader *imageDown
loader;

...

@end
```

SDWebImageManager 声明了一个 delegate 属性，其实是一个 id 对象，代理声明了两个方法

```
// 控制当图片在缓存中没有找到时，应该下载哪个图片
- (BOOL)imageManager:(SDWebImageManager *)imageManager shouldDownloadImageForURL:(NSURL *)imageURL;

// 允许在图片已经被下载完成且被缓存到磁盘或内存前立即转换
- (UIImage *)imageManager:(SDWebImageManager *)imageManager transformDownloadedImage:(UIImage *)image withURL:(NSURL *)imageURL;
```

这两个方法会在 SDWebImageManager 的-downloadImageWithURL:options:progress:completed:方法中调用，而这个方法是 SDWebImageManager 类的核心所在（具体看源码）

SDWebImageManager 的几个 API：

（1）- (void)cancelAll ： 取消 runningOperations 中所有的操作，并全部删除

（2）- (BOOL)isRunning ：检查是否有操作在运行，这里的操作指的是下载和缓存组成的组合操作

（3） - downloadImageWithURL:options:progress:completed: 核心方法

（4）- (BOOL)diskImageExistsForURL:(NSURL *)url ：指定 url 的图片是否进行了磁盘缓存

四、视图扩展

在使用 SDWebImage 的时候，使用最多的是 UIImageView+WebCache 中的针对 UIImageView 的扩展，核心方法是 sd_setImageWithURL:placeholderImage:options:progress:completed:， 其使用 SDWebImageManager 单例对象下载并缓存图片。

除了扩展 UIImageView 外，SDWebImage 还扩展了 UIView，UIButton，MKAnnotationView 等视图类，具体可以参考源码，除了可以使用扩展的方法下载图片，同时也可以使用 SDWebImageManager 下载图片。

UIView+WebCacheOperation 分类： 把当前 view 对应的图片操作对象存储起来（通过运行时设置属性），在基类中完成 存储的结构：一个 loadOperationKey 属性，value 是一个字典（字典结构： key：UIImageViewAnimationImages 或者 UIImageViewImageLoad，value 是 operation 数组（动态图片）或者对象）

UIButton+WebCache 分类 会根据不同的按钮状态，下载的图片根据不同的状态进行设置 imageURLStorageKey:{state:url}

五、技术点

- 1.dispatch_barrier_sync 函数，用于对操作设置顺序，确保在执行完任务后再确保后续操作。常用于确保线程安全性操作
- 2.NSMutableURLRequest：用于创建一个网络请求对象，可以根据需要来配置请求报头等信息
- 3.NSOperation 及 NSOperationQueue：操作队列是 OC 中一种告诫的并发处理方法，基于 GCD 实现，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，另外在管理操作间的依赖关系方面容易一些，对 SDWebImage 中我们看到如何使用依赖将下载顺序设置成后进先出的顺序
- 4.NSURLSession：用于网络请求及相应处理
- 5.开启后台任务
- 6.NSCache 类：一个类似于集合的容器，存储 key-value 对，这一点类似于 nsdictionary 类，我们通常用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，因为它们的值不需要重新计算。另外一方面，这些对象对于程序来说不是紧要的，在内存紧张时会被丢弃
- 7.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这一大小，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，直到缓存的实际大小小于我们设置的最大使用空间。
- 8.图片解压操作：这一操作可以查看 SDWebImageDecoder.m 中 +decodedImageWithImage 方法的实现。
- 9.对 GIF 图片的处理
- 10.对 WebP 图片的处理。

返回目录:全网各大厂 iOS 面试题-题集大全
