真的会使用**SDWebImage?!!!**

使用场景：自定义的UITableViewCell上有图片需要显示，要求网络状态为WiFi时，显示图片高清图；网络状态为蜂窝移动网络时，显示图片缩略图。

如下图样例：

图中显示的图片符合根据网络状态下载要求，由于要监听网络状态，在这里推荐使用AFNetWorking。

在AppDelegate.m文件中的application:didFinishLaunchingWithOptions:方法中监听网络状态。

```
// AppDelegate.m 文件中
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions
{
    // 监控网络状态
    [[AFNetworkReachabilityManager sharedManager] startMonitoring];

}
```

```
// 以下代码在需要监听网络状态的方法中使用
AFNetworkReachabilityManager *mgr = [AFNetworkReachabilityManager sharedManager];
    if (mgr.isReachableViaWiFi)    {
          // 在使用Wifi, 下载原图
    } else    {
          // 其他，下载小图
    }
 }
```

这不是很简单吗？于是三下五除二写完了以下代码。

```
// 利用MVC，在设置cell的模型属性时候，下载图片

– setItem:(CustomItem *)item
{
    _item = item;
    UIImage *placeholder = [UIImage imageNamed:@"placeholderImage"];
    AFNetworkReachabilityManager *mgr = [AFNetworkReachabilityManager sharedManager];
    if (mgr.isReachableViaWiFi) { // 在使用Wifi, 下载原图
        [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
    } else { // 其他，下载小图
        [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.thumbnailImage]
placeholderImage:placeholder];
    }
}
```

此时，确实能完成基本的按照当前网络状态下载对应的图片，但是真实开发中，这样其实是不合理的。以下是需要注意的细节：
1.SDWebImage会自动帮助开发者缓存图片（包括内存缓存，沙盒缓存），所以我们需要设置用户在WiFi环境下下载的高清图，下次在蜂窝网络状态下打开应用也应显示高清图，而不是去下载缩略图。
2.许多应用设置模块带有一个功能：移动网络环境下仍然显示高清图。这个功能其实是将设置记录在沙盒中，关于数据保存到本地，可以查看本人另一篇简书首页文章
3.iOS本地数据存取，看这里就够了
4.当用户处于离线状态时候，无法合理处理业务。

于是，开始加以改进。为了让更容易理解，我先贴出伪代码：

```
– setItem:(CustomItem *)item
{
    _item = item;
    if (缓存中有原图)
    {
        self.imageView.image = 原图;
```

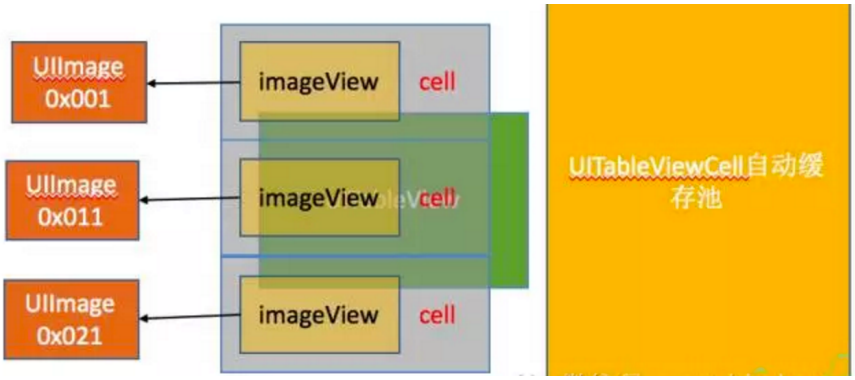
```
    } else
    {
        if (Wifi环境)
        {
            下载显示原图
        } else if (手机自带网络)
        {
            if (3G\4G环境下仍然下载原图)
            {
                下载显示原图
            } else
            {
                下载显示小图
            }
        } else
        {
            if (缓存中有小图)
            {
                self.imageView.image = 小图;
            } else  // 处理离线状态
            {
                self.imageView.image = 占位图片;
            }
        }
    }
}
```

实现上面的伪代码：读者可以一一对应上面的伪代码。练习的时候推荐先写伪代码，再写真实代码注意注释解释。

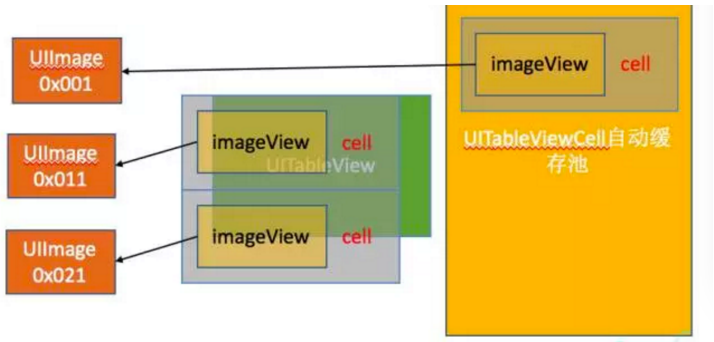
```
- setItem:(CustomItem *)item
{
    _item = item;
     // 占位图片
    UIImage *placeholder = [UIImage imageNamed:@"placeholderImage"];
    // 从内存\沙盒缓存中获得原图,
    UIImage *originalImage = [[SDImageCache sharedImageCache]
imageFromDiskCacheForKey:item.originalImage];
    if (originalImage) { // 如果内存\沙盒缓存有原图，那么就直接显示原图（不管现在是什么网络状态)
        self.imageView.image = originalImage;
    } else { // 内存\沙盒缓存没有原图
        AFNetworkReachabilityManager *mgr = [AFNetworkReachabilityManager
sharedManager];
        if (mgr.isReachableViaWiFi) { // 在使用Wifi, 下载原图
```

```
            [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
        } else if (mgr.isReachableViaWWAN) { // 在使用手机自带网络
            //     用户的配置项假设利用NSUserDefaults存储到了沙盒中
            //    [[NSUserDefaults standardUserDefaults] setBool:NO
forKey:@"alwaysDownloadOriginalImage"];
            //    [[NSUserDefaults standardUserDefaults] synchronize];
#warning 从沙盒中读取用户的配置项：在3G\4G环境是否仍然下载原图
            BOOL alwaysDownloadOriginalImage = [[NSUserDefaults standardUserDefaults]
boolForKey:@"alwaysDownloadOriginalImage"];
            if (alwaysDownloadOriginalImage) { // 下载原图
                [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
            } else { // 下载小图
                [self.imageView sd_setImageWithURL:[NSURL
URLWithString:item.thumbnailImage] placeholderImage:placeholder];
            }
        } else { // 没有网络
            UIImage *thumbnailImage = [[SDImageCache sharedImageCache]
imageFromDiskCacheForKey:item.thumbnailImage];
            if (thumbnailImage) { // 内存\沙盒缓存中有小图
                self.imageView.image = thumbnailImage;
            } else { // 处理离线状态，而且有没有缓存时的情况
                self.imageView.image = placeholder;
            }
        }
    }
}
```

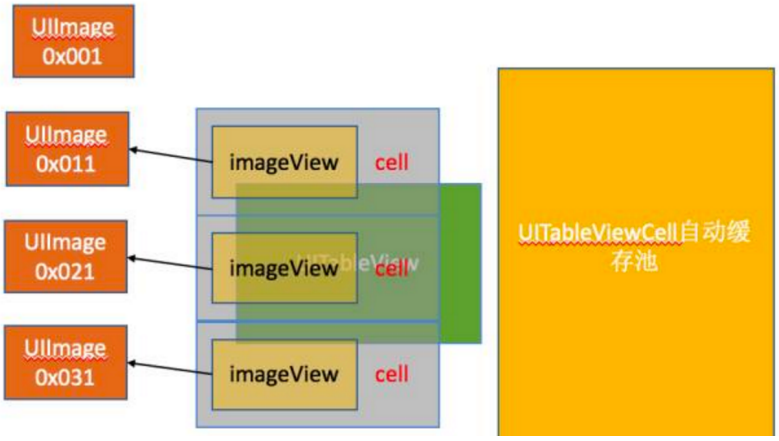
解决了吗？真正的坑才刚刚开始。
在表述上述代码的坑之前，先来分析一下UITableViewCell的缓存机制。
请看下图：有一个tableView正在同时显示三个UITableViewCell，每个tableViewCell包含一个imageView的子控件，而且每个cell都有一个对应的模型属性用来设置imageView的图片内容。
注意：由于没有cell被推出屏幕，此时缓存池为空。



当有一个cell被推到屏幕之外时，系统会自动将这个cell放入自动缓存池。注意:cell对应的UIImage图片数据模型并没有清空！还是指向上一个使用的cell。



当下一个cell进入屏幕，系统会根据tableView注册的标识找到对应的cell，拿来应用。上述进入缓存池的cell被重新添加进tableView，在tableView的Data Source方法tableView:cellForRowAtIndexPath:中设置改cell对应的模型数据，此时cell对应的如图：



以上就是tableView重用机制的简单介绍

重新回来，那么上面所说的真正的坑在哪呢?

用一个场景来描述一下吧：当用户所处环境WiFi网速不够快(不能立即将图片下载完毕)，而在上述代码，在WiFi环境下又是下载高清大图。所以需要一定的时间来完成下载。而就在此时，用户不愿等，想看看上次打开App时显示的图片，此时用户会滑到处于下面的cell来查看。注意：此时，上面的cell下载图片操作并没有暂停，还在处于下载图片状态中。当用户在查看上次打开App的显示图片时（上次打开App下载完成的图片，SDWebImage会帮我们缓存，不用下载），而正好需要显示上次打开App时的图片的cell是利用tableView重用机制而从缓存池中拿出来的cell，等到处于上面的cell的高

清大图已经下载好了的时候，SDWebImage默认做法是，立马把下载好的图片设置给ImageView，所以我们这时候会在底下的显示的cell显示上面的图片，造成数据错乱，这是非常严重的BUG。

那么该如何解决这个棘手的问题呢?

如果我们能在cell被从缓存池中拿出来使用的时候，将这个cell放入缓存池之前的下载操作移除，那么就不会出现数据错乱了。
这时候你可能会问我：怎么移除下载操作？下载操作不是SDWebImage帮我们做的吗?
说的没错，确实是SDWebImage帮我们下载图片的，我们来扒一扒SDWebImage源码，看看他是怎么完成的。

```
- (void)sd_setImageWithURL:(NSURL *)url placeholderImage:(UIImage *)placeholder options:
(SDWebImageOptions)options progress:
(SDWebImageDownloaderProgressBlock)progressBlock completed:
(SDWebImageCompletionBlock)completedBlock {
    // 关闭当前图片的下载操作
    [self sd_cancelCurrentImageLoad];
```

发现，原来SDWebImage在下载图片时，第一件事就是关闭imageView当前的下载操作!
是不是开始感叹SDWebImage多么神奇了？没错，我们只需要把我们写的那段代码所有的直接访问本地缓存代码利用SDWebImage进行设置就OK了!
下面就是完成版代码。

```
- setItem:(CustomItem *)item
{
    _item = item;
  // 占位图片
    UIImage *placeholder = [UIImage imageNamed:@"placeholderImage"];
    // 从内存\沙盒缓存中获得原图
    UIImage *originalImage = [[SDImageCache sharedImageCache]
imageFromDiskCacheForKey:item.originalImage];
    if (originalImage) { // 如果内存\沙盒缓存有原图，那么就直接显示原图（不管现在是什么网络状态)
        [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
    } else { // 内存\沙盒缓存没有原图
        AFNetworkReachabilityManager *mgr = [AFNetworkReachabilityManager
sharedManager];
        if (mgr.isReachableViaWiFi) { // 在使用Wifi, 下载原图
            [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
        } else if (mgr.isReachableViaWWAN) { // 在使用手机自带网络
            //     用户的配置项假设利用NSUserDefaults存储到了沙盒中
            //    [[NSUserDefaults standardUserDefaults] setBool:NO
forKey:@"alwaysDownloadOriginalImage"];
```

```
        //    [[NSUserDefaults standardUserDefaults] synchronize];
#warning 从沙盒中读取用户的配置项：在3G\4G环境是否仍然下载原图
        BOOL alwaysDownloadOriginalImage = [[NSUserDefaults standardUserDefaults]
boolForKey:@"alwaysDownloadOriginalImage"];
        if (alwaysDownloadOriginalImage) { // 下载原图
            [self.imageView sd_setImageWithURL:[NSURL URLWithString:item.originalImage]
placeholderImage:placeholder];
        } else { // 下载小图
            [self.imageView sd_setImageWithURL:[NSURL
URLWithString:item.thumbnailImage] placeholderImage:placeholder];
        }
    } else { // 没有网络
        UIImage *thumbnailImage = [[SDImageCache sharedImageCache]
imageFromDiskCacheForKey:item.thumbnailImage];
        if (thumbnailImage) { // 内存\沙盒缓存中有小图
            [self.imageView sd_setImageWithURL:[NSURL
URLWithString:item.thumbnailImage] placeholderImage:placeholder];
        } else {
            [self.imageView sd_setImageWithURL:nil placeholderImage:placeholder];
        }
    }
}
```