iOS网络缓存扫盲

张应龙

日期：2016年 7月27日

58同城

- 什么是缓存
- URLCache机制
- 如何设置缓存
- 缓存有效性控制
- 项目实战

网络缓存

- 什么是cache？
- cache就是高速缓存，存在于主存RAM和CPU之间，主要用于解决CPU处理数据的速度远远大于读取主存数据的速度。
- 手机上也有cache，主要作用是保存一些软件生成的临时文件，避免每次都要重复地向服务器请求相同的数据，既浪费用户流量，也影响APP响应速度。
- 类型：（memory）内存缓存和（disk）硬盘缓存。

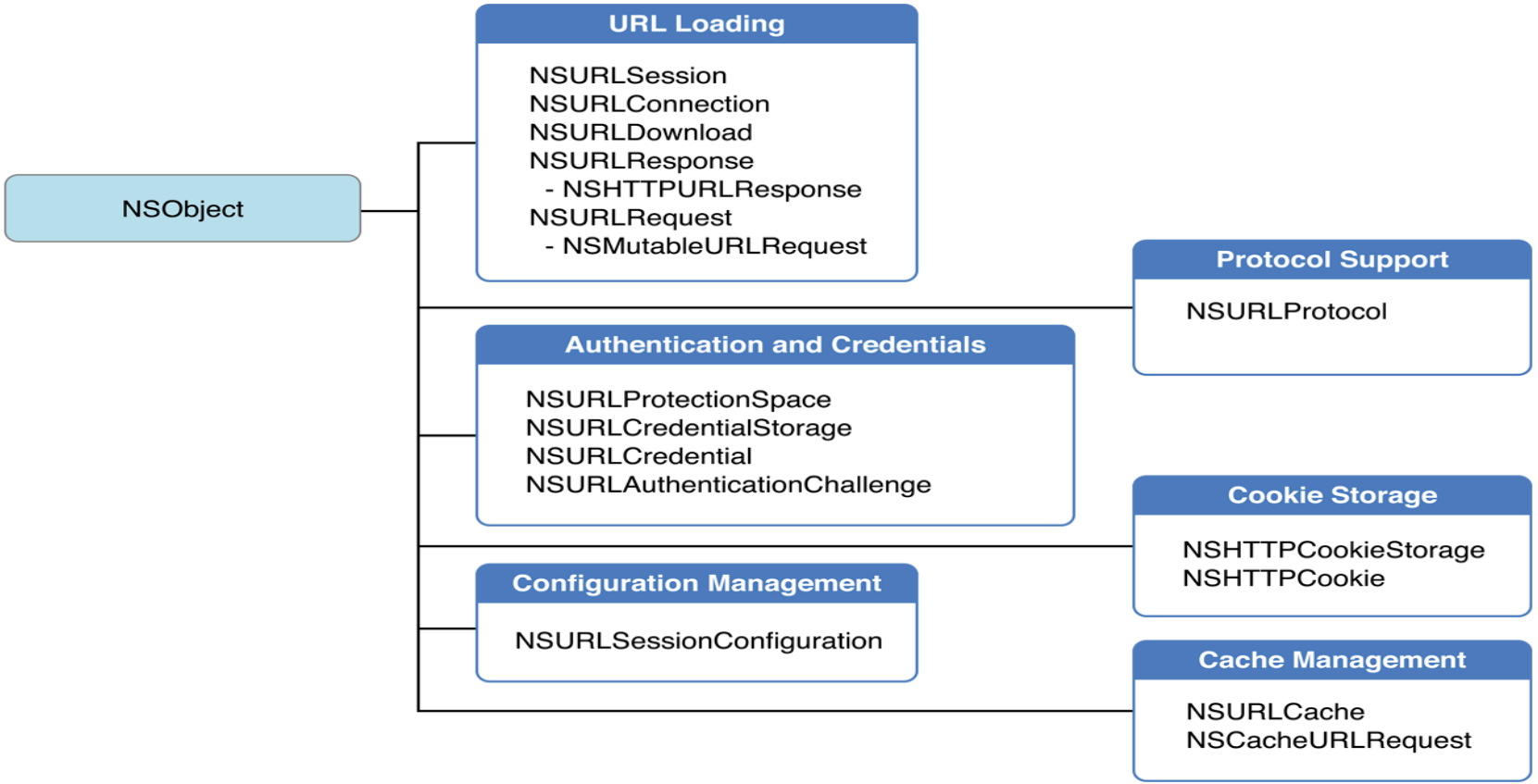
网络缓存

- 为什么需要网络缓存？

  只有你真正感受到痛的时候，才会考虑使用缓存。

-	第一种	第二种
目的	优化型缓存	功能型缓存
具体描述	出于优化考虑：服务器压力、用户体验、为用户剩流量等等。 同时优化型缓存也有内存缓存和磁盘缓存之分。	App离线也能查看， 出于功能考虑，属于存储范畴
常见概念	GET网络请求缓存、WEB缓存	离线存储
典型应用	微信首页的会话列表、微信头像、朋友圈、网易新闻新闻列表、	微信聊天记录、

About the URL Loading System



iOS基础类库NSURLCache

- NSURLCache 在你进行URL请求时为你的应用提供了一套内存和磁盘双重缓存机制。

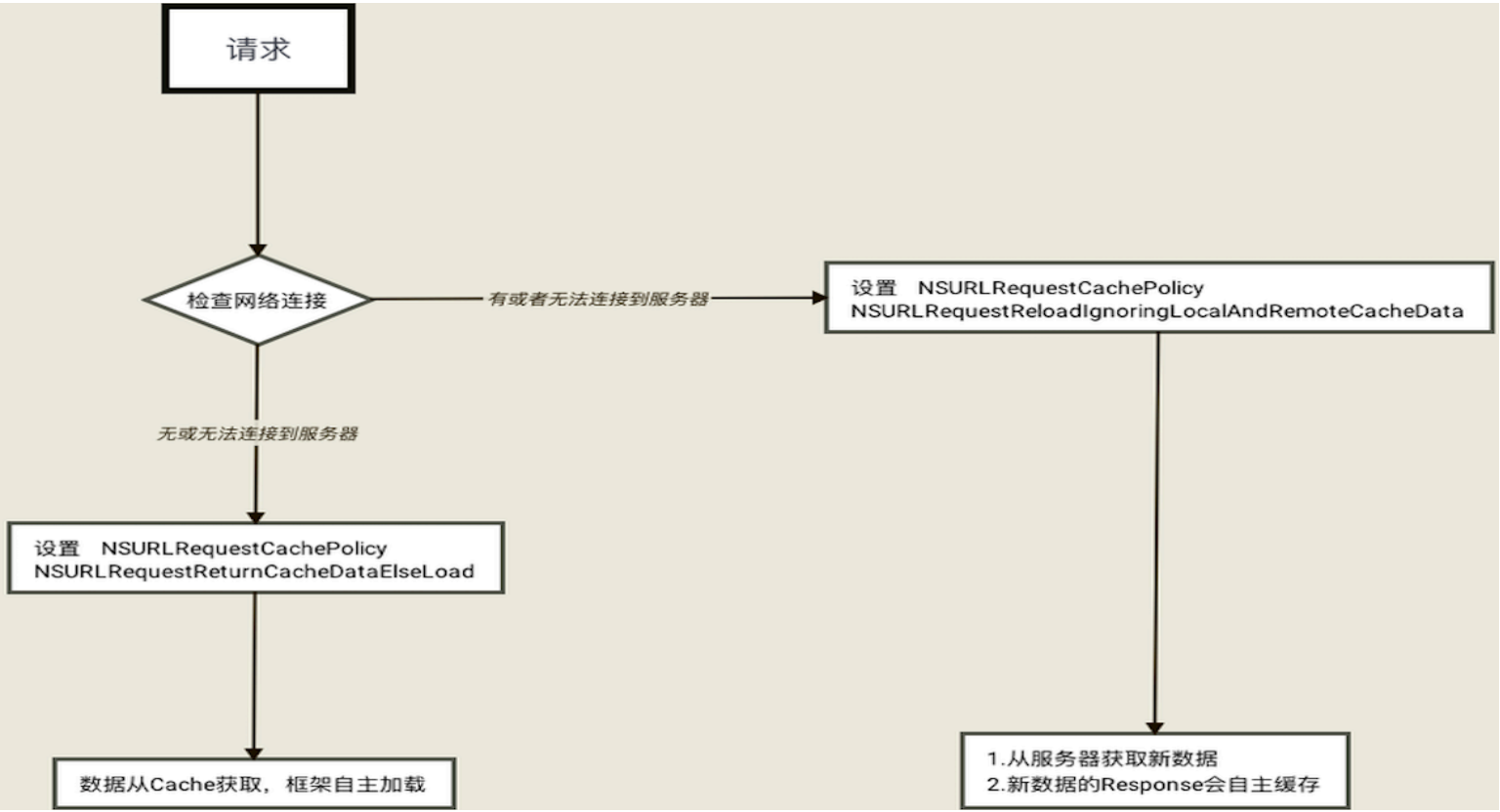
- 作为Foundation框架的URL Loading System的一部分，NSURLCache会处理所有通过NSURLConncetion(NSURLSession)载入的请求。

- 缓存策略由请求（客户端）和回应（服务端）分别指定。理解这些策略以及它们如何相互影响，是为您的应用程序找到最佳行为的关键。

NSURLCache缓存策略

策略	意义
UseProtocolCachePolicy	默认行为
ReloadIgnoringLocalCacheData	不使用缓存
ReloadIgnoringLocalAndRemoteCacheData*	我是认真地，不使用任何缓存
ReturnCacheDataElseLoad	使用缓存（不管它是否过期），如果缓存中没有，那从网络加载吧
ReturnCacheDataDontLoad	离线模式：使用缓存（不管它是否过期），但是不从网络加载
ReloadRevalidatingCacheData*	在使用前去服务器验证

流程示例


请求
检查网络连接
有或者无法连接到服务器
设置 NSURLRequestCachePolicy
NSURLRequestReloadIgnoringLocalAndRemoteCacheData
无或无法连接到服务器
设置 NSURLRequestCachePolicy
NSURLRequestReturnCacheDataElseLoad
数据从Cache获取，框架自主加载
1.从服务器获取新数据
2.新数据的Response会自主缓存

iOS设置网络缓存

- 设置缓存只需两个步骤：

  1、请使用GET请求(HTTP协议)；2、iOS只需要两行代码

```
NSURLCache *cache = [[NSURLCache alloc] initWithMemoryCapacity:4*1024*1024
                                                  diskCapacity:20*1024*1024
                                                      diskPath:nil];
[NSURLCache setSharedURLCache:cache];
```

- 你只要设置了这两行代码，基本就可满足80%的缓存需求。
- Mattt曾经说过：无数开发者尝试自己做一个简陋而脆弱的系统来实现网络缓存的功能，殊不知 NSURLCache 只要两行代码就能搞定且性能好上100倍。

使用80%的代码来完成剩下的20%的缓存需求

- 系统提供的NSURLCache已解决了80%的需求，剩下20%的需求怎么办？
  1、不够灵活，不能自定义
  2、只能是GET请求的资源

- 自定义有效性控制策略，完善剩下20%的需求。

Demo例子（简单超时机制）

```objc
#pragma mark - NSURLCache

- (NSCachedURLResponse *)cachedResponseForRequest:(NSURLRequest *)request {
    NSCachedURLResponse *cachedResponse = [super cachedResponseForRequest:request];
        if (cachedResponse) {
            NSDate* cacheDate = cachedResponse.userInfo[@"CustomURLCacheExpirationKey"];
            NSDate* cacheExpirationDate = [cacheDate dateByAddingTimeInterval:60];
            if ([cacheExpirationDate compare:[NSDate date]] == NSOrderedAscending) {
                [self removeCachedResponseForRequest:request];
                return nil;
            }
        }
    }
    return cachedResponse;
}

- (void)storeCachedResponse:(NSCachedURLResponse *)cachedResponse
                 forRequest:(NSURLRequest *)request {
    NSMutableDictionary *userInfo = [NSMutableDictionary dictionaryWithDictionary:cachedResponse.userInfo];
    userInfo[@"CustomURLCacheExpirationKey"] = [NSDate date];
    NSCachedURLResponse *modifiedCachedResponse = [[NSCachedURLResponse alloc] initWithResponse:cachedResponse.response
                                                                                           data:cachedResponse.data
                                                                                       userInfo:userInfo
                                                                                  storagePolicy:cachedResponse.storagePolicy];

    [super storeCachedResponse:modifiedCachedResponse forRequest:request];
}
```

HTTP缓存机制

- 只要是缓存，总会过期。（文件版本管理，超时时间管理等等）

- 如果我们对数据实时性要求很高，即使1秒钟后数据变更了，客户端也必须展示最新数据。这种情况如何处理？
  1、采用资源变动后就重新生成一个链接
  2、采用标签Etag或者Last-Modified判断是否有效
  3、MD5校验

HTTP请求缓存机制

在默认情况下，NSURLRequest 会用当前时间决定是否返回缓存的数据。为了更精确地控制，可以使用以下请求头：

- If-Modified-Since 这个请求头与 Last-Modified 响应头相对应。把这个值设为最后一次请求时返回的 Last-Modified 字段的值。

- If-None-Match - 这个请求头与与 Etag 回应头相对应。使用同一终端最后一次请求的 Etag 值。

HTTP响应缓存机制

NSHTTPURLResponse 包含多个 HTTP 头，当然也包括以下指令来说明回应应当如何缓存：

- Cache-Control - 这个头必须由服务器端指定以开启客户端的 HTTP 缓存功能。这个头的值可能包含 max-age（缓存多久），是公共 public 还是私有 private，或者不缓存 no-cache 等信息。
- Last-Modified - 这个头的值表明所请求的资源上次修改的时间。例如，一个客户端请求最近照片的时间线，/photos/timeline，Last-Modified 的值可以是最近一张照片的拍摄时间。
- Etag - 这是 “entity tag” 的缩写，它是一个表示所请求资源的内容的标识符。在实践中，Etag 的值可以是类似于资源的 MD5 之类的东西。

58同城

THANK YOU

张应龙