**1、SDWebImage的简单使用**

SDWebImage实现缓存的步骤

SDWebImage.png

1、当我们需要获取网络图片的时候，我们首先需要的便是URL，获得URL后我们SDWebImage实现的并不是直接去请求网路，而是去检查图片缓存中有没有和URl相关的图片，如果有则直接返回image，如果没有则进行下一步。  
2、当图片缓存中没有图片时，SDWebImage依旧不会直从网络上获取，而是检查沙盒中是否存在图片，如果存在，则把沙盒中对应的图片存进image缓存中，然后按着第一步的判断进行。  
3、如果沙盒中也不存在，则显示占位图，然后根据图片的下载队列缓存判断是否正在下载，如果下载则等待，避免二次下载。如果不存则创建下载队列，下载完毕后将下载操作从队列中清除，并且将image存入图片缓存中。!  
4、刷新UI（当然根据实际情况操作）将image存入沙盒缓存。

1、下载图片并显示（内存缓存&磁盘缓存）

/\*

第一个参数：图片的url地址

第二个参数：设置的占位图片

\*/

[self.headIcon sd\_setImageWithURL:[NSURL URLWithString:topicItem.profile\_image]

placeholderImage:[[UIImage imageNamed:@"defaultUserIcon"] circleImage]

completed:^(UIImage \* \_Nullable image, NSError \* \_Nullable error, SDImageCacheType cacheType, NSURL \* \_Nullable imageURL) {

self.headIcon.image = [image circleImage] ;

}] ;

2、下载图片显示并计算下载进度（内存缓存&磁盘缓存&下载进度）

-(void)download

{ /\*

第一个参数：图片的url地址

第二个参数：设置的占位图片

第三个参数：下载图片选项（策略）

第四个参数：进度回调block

receivedSize:已经下载的数据大小

expectedSize：图片的总大小

第五个参数：completed图片下载结束回调（成功|失败）

image：下载后得到的图片，如果下载失败，那么image的值为nil

error：错误信息，如果失败，则error有值

cacheType：图片来源（枚举：内存缓存|磁盘缓存|直接下载）

imageURL：下载图片的url

\*/

[self.imageView sd\_setImageWithURL:[NSURL URLWithString:topicItem.gifFistFrame] placeholderImage:[UIImage imageNamed:@"imageBackground"] options:SDWebImageProgressiveLoad progress:^(NSInteger receivedSize, NSInteger expectedSize, NSURL \* \_Nullable targetURL) {

if (!expectedSize) return ;

topicItem.pictureProgress = 1.0 \* receivedSize / expectedSize ;

//进度是在子线程中完成的 所以要回到主线程

dispatch\_async(dispatch\_get\_main\_queue(), ^{

[self.progressView setProgress:topicItem.pictureProgress animated:YES] ;

}) ;

} completed:^(UIImage \* \_Nullable image, NSError \* \_Nullable error, SDImageCacheType cacheType, NSURL \* \_Nullable imageURL) {

// [self.progressView performAction:M13ProgressViewActionSuccess animated:YES];

self.progressView.hidden = YES ;

switch (cacheType) {

case SDImageCacheTypeNone:

NSLog(@"直接下载");

break;

case SDImageCacheTypeDisk:

NSLog(@"磁盘缓存");

break;

case SDImageCacheTypeMemory:

NSLog(@"内存缓存");

break;

default:

break;

}

// NSLog(@"当前是非WIFI状态下，动图不会自动播放") ;

}] ;

**2、Http 和 Https 的区别？为什么更加安全？**

**区别**

* 1.HTTPS 需要向机构申请 CA 证书，极少免费。
* 2.HTTP 属于明文传输，HTTPS基于 SSL 进行加密传输。
* 3.HTTP 端口号为 80，HTTPS 端口号为 443 。
* 4.HTTPS 是加密传输，有身份验证的环节，更加安全。

**安全**

**SSL(安全套接层)**  
**TLS(传输层安全)**  
以上两者在传输层之上，对网络连接进行加密处理，保障数据的完整性，更加的安全。

**3.Http的请求方式有哪些？Http 有什么特性？**

**Http的请求方式有一下 6 种：**

* 1.GET
* 2.POST
* 3.HEAD
* 4.PUT
* 5.DELETE
* 6.OPTIONS
* 7.TRACE
* 8.CONNECT  
  **Http 的特点有：**
* 1.无连接：可以理解为持久链接。
* 2.无状态：利用 session && cookie 区分用户。

**4.解释一下 三次握手 和 四次挥手？**

**三次握手**

* 1.由客户端向服务端发送 SYN 同步报文。
* 2.当服务端收到 SYN 同步报文之后，会返回给客户端 SYN 同步报文和 ACK 确认报文。
* 3.客户端会向服务端发送 ACK 确认报文，此时客户端和服务端的连接正式建立。

**建立连接**

* 1.这个时候客户端就可以通过 Http 请求报文，向服务端发送请求
* 2.服务端接收到客户端的请求之后，向客户端回复 Http 响应报文。

**四次挥手**

**当客户端和服务端的连接想要断开的时候，要经历四次挥手的过程，步骤如下：**

* 1.先由客户端向服务端发送 FIN 结束报文。
* 2.服务端会返回给客户端 ACK 确认报文 。此时，由客户端发起的断开连接已经完成。
* 3.服务端会发送给客户端 FIN 结束报文 和 ACK 确认报文。
* 4.客户端会返回 ACK 确认报文到服务端，至此，由服务端方向的断开连接已经完成。

**解释一下为什么是"三次握手"**

客户端在与服务端建立连接的过程中，客户端向服务端发送连接请求，如果因为网络的原因，导致它超时触发了重传机制，那么客户端会重新向服务端发送一次连接请求，那有可能第二次发送的连接请求，服务端收到之后，直接返回给客户端确认连接的报文。

如果只需要两次就可以建立连接，假如说第一次建立连接的请求终于被服务端接收到了，那服务端收到之后，会以为客户端想新建另外一个链接，这样的话就会建立两个连接。

而采取三次握手，在客户端收到服务端的确认报文之后，客户端再向服务端发送一个确认报文，这时才正式建立连接。那么第一次因为超时而迟迟没有收到的建立连接的请求，这时在收到之后，服务端依然会向客户端返回一个确认报文，但是服务端现在已经与客户端建立的连接，所以客户端就不会再向服务端发送确认报文，服务端迟迟没有收到确认报文，那么也就不会真正的建立再一次连接。

**又为什么是"四次挥手"？**

建立的时候是双方面的建立，而释放的时候，也必然是客户端和服务端双方面的断开连接。

客户端与服务端建立的连接是双通的，这也就意味着客户端可以向服务端发送请求，服务端向客户端返回响应，同样的也可以从服务端向客户端发送，然后由客户端向服务端返回，如果你只关闭了客户端，向服务端通道，这只叫半关闭状态，并没有完全的切断客户端与服务端的关联。

**6.GET 和 POST 请求的区别？**

**1.参数位置**

* 1.GET 方式的请求参数以 ? 拼接到 Url 后面。
* 2.Post 请求的参数在请求体内。

**2.参数长度**

* 1.GET 方式的请求参数长度限制为 2048字符。
* 2.Post 请求的参数长度没有该限制。

**3.安全性**

* 1.GET 方式不会引起服务端的任何变化

**4.幂等性（同一方法执行多次和执行一次效果相同）**

* 1.GET 幂等。
* 2.POST 不幂等。

**5.缓存性**

* 1.GET 可缓存。
* 2.Post 不可缓存。

**7.HTTP 请求报文 和 响应报文的结构？**

**1.请求报文的结构。**

* ①：请求行 （请求方式、URL）
* ②：首部行（键值对形式，CRLF 代表换行结束）
* ③：实体主体（post请求中，存放请求参数）

**1.响应报文的结构。**

* ①：响应行 （版本、状态码）
* ②：首部行 （键值对形式，CRLF 代表换行结束）
* ③：实体主体（响应返回的数据）

**9.数据传输的加密过程？**

**SSL(安全套接层)**  
**TLA(传输层安全)**

以上两者在传输层之上，对网络连接进行加密处理，保障数据的完整性，更加的安全。

* 1.客户端：发送数据 -> HTTP(应用层) -> SSL&TLS(加密处理) -> TCP(传输层) -> IP(网络层)
* 2.服务端： IP(网络层) -> TCP(传输层) -> SSL&TLS(解密处理) -> HTTP(应用层) -> 接受数据

**10.说一下 TCP/IP 五层模型的协议?**

**应用层**

**传输层**

**网路层**

**数据链路层**

**物理层**

**11.说一下 OSI 七层模型的协议?**

* 应用层 文件传输，电子邮件，文件服务，虚拟终端 TFTP，HTTP，SNMP，FTP，SMTP，DNS，Telnet
* 表示层 数据格式化，代码转换，数据加密 没有协议
* 会话层 解除或建立与别的接点的联系 没有协议
* 传输层 提供端对端的接口 TCP，UDP
* 网络层 为数据包选择路由 IP，ICMP，RIP，OSPF，BGP，IGMP
* 数据链路层 传输有地址的帧以及错误检测功能 SLIP，CSLIP，PPP，ARP，RARP，MTU
* 物理层 以二进制数据形式在物理媒体上传输数据 ISO2110，IEEE802，IEEE802.2

**13.断点续传 功能该怎么实现？**

* 1.利用 http 的头部字段 ，设置 http 的 Range 属性
* 2.检测下载进度，并且判断是否需要重新创建下载文件

**14、 分类和类拓展的区别?**

* 1.分类 的加载在 运行时，类拓展 的加载在 编译时。
* 2.分类 不能给系统的类添加方法。
* 3.类拓展 只以声明的形式存在，一般存在 .m 文件中。

**分类和类拓展的区别?**

* 1.分类 的加载在 运行时，类拓展 的加载在 编译时。
* 2.分类 不能给系统的类添加方法。
* 3.类拓展 只以声明的形式存在，一般存在 .m 文件中。

**16.NSUrlProtocol用过吗？用在什么地方了？**

可以拦截全局的网络请求。

**17.如何在测试过程中 MOCK 各种网络环境？**

[https://blog.csdn.net/u010828718/article/details/46986795](https://links.jianshu.com/go?to=https%3A%2F%2Fblog.csdn.net%2Fu010828718%2Farticle%2Fdetails%2F46986795)

**18.DNS 的解析过程？网络的 DNS 优化。**

DNS 是域名到 IP 地址的映射，DNS 解析使用 UDP 数据报，端口号53，并且采用明文传输的方式

客户端在向服务端发送请求时，会先将 域名 到 DNS 服务器映射出 IP 地址，然后再访问。

**DNS 解析的两种方式**

**递归查询**

不断地自下而上遍历解析，“我去给你问一下”的方式

**迭代查询**

迭代查询 是 “我告诉你谁可能知道”的方式

**DNS 解析存在的问题**

**DNS 劫持**

被钓鱼网站劫持，有可能返回错误的 IP，浏览的不是目标浏览器

**DNS 解析转发**

晓得运营商可能将 DNS 解析请求转发，解析的比较慢，效率低

**DNS 劫持解决办法**

**httpDNS**

使用 http 协议向 DNS 服务器 80 端口进行请求

**长连接**

找一个中间的长连 server ,在内网专线进行 Http 请求。客户端和这个长连 server通信即可。

**19.Post请求体有哪些格式？**

**application/x-www-form-urlencoded**

这应该是最常见的 POST 提交数据的方式了。浏览器的原生 form 表单，如果不设置 enctype 属性，那么最终就会以 application/x-www-form-urlencoded 方式提交数据

POST HTTP/1.1

Host: www.demo.com

Cache-Control: no-cache

Postman-Token: 81d7b315-d4be-8ee8-1237-04f3976de032

Content-Type: application/x-www-form-urlencoded

key=value&testKey=testValue

**multipart/form-data**

POST HTTP/1.1

Host: www.demo.com

Cache-Control: no-cache

Postman-Token: 679d816d-8757-14fd-57f2-fbc2518dddd9

Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="key"

value

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="testKey"

testValue

------WebKitFormBoundary7MA4YWxkTrZu0gW

Content-Disposition: form-data; name="imgFile"; filename="no-file"

Content-Type: application/octet-stream

<data in here>

------WebKitFormBoundary7MA4YWxkTrZu0gW--

**application/json**

**text/xml**

**20.网络请求的状态码都大致代表什么意思？**

**1.1xx**

1xx 代表临时响应，需要请求者继续执行操作的状态代码。

**2.2xx**

2xx 代表的多是操作成功。

**3.3xx**

3xx 代表重定向，表示要完成请求，需要进一步操作

**4.4xx**

4xx 代表请求错误，表示请求可能出错，妨碍了服务器的处理。

**5.5xx**

5xx 代表服务器错误，表示服务器在尝试处理请求时发生内部错误。 这些错误可能是服务器本身的错误，而不是请求出错

**21.抓包软件 Charles 的原理是什么？说一下中间人攻击的过程。**

Charles 的原理实际就是网络的 中间人攻击。

* 客户端发给服务端的请求，被中间人劫持篡改之后，再发送给服务端。
* 服务端返回给客户端的数据，被中间人劫持篡改之后，再返回给客户端。

**22.如何判断一个请求是否结束？**

* 1.查看 Content-Length 是否达到 1024 字节。
* 2.通过 Post 发送请求时，消息一般都是一段一段返回的，看最后的一个 chunked 是否为空?

**23. SSL 传输协议？说一下 SSL 验证过程？**

* 1.配备了 身份证书，防止被冒充。
* 2.有 校验机制，一旦被篡改，双方都会知道。
* 3.通过 加密 进行传播。

**连接过程如下：**

* ①：客户端向服务端发消息，包括客户端支持的 加密算法 以及 随机数1 、TLS 版本号。
* ②：服务端向客户端返回信息，包括双方已经匹配好的 加密算法 以及 随机数2、Server证书 。
* ③：服务端向客户端返回 验证证书。
* ④：客户端进行证书验证，首先要评估信任证书
  + 1.服务端向客户端返回证书的 数字摘要 与服务端证书返回客户端，解密后的是否一致，防止被篡改。
  + 2.验证证书链：逐级验证服务端证书是否在可信任证书列表。
  + 3.验证通过后：
    - ①：将 随机数1、随机数2、服务器端返回 证书公钥 加密过后的 预主秘钥 三者组合成 会话密钥。
    - ②：服务端拿到 会话秘钥 后，会用服务端证书 私钥进行解密，得到 预主秘钥。
    - ③：服务端将 随机数1、随机数2、服务器端证书私钥解密得到的 预主秘钥 三者组合成 会话密钥。此时客户端的 会话秘钥 和 服务端的 会话秘钥 应该是一致的。
    - ④：接下来会经行验证：  
      - 1.客户端通过 会话秘钥，加密信息发给服务端，看服务端能否成功接收。  
      - 2.服务端通过 会话秘钥，加密信息返给客户端，看客户端能否成功解析。
  + 4.以上步骤都通过了，此时 SSL 验证连接。

**24.解释一下 Http 的持久连接？**

每次建立连接的时候都要经过三次握手等必须的程序，如果我们拥有一条可以一直使用的连接的话，也就意味着我们只需要进行一次连接的建立，这就省去了每次建立连接的时间。  
使用过的连接会比新建立的连接速度会快一些，这是由于TCP连接慢启动的特性，每次建立新的连接，当然不如已经被调教的很好的连接速度快。  
每个连接对于服务器和客户端来说都是负担，能少开尽量少开，当然是在不影响功能和体验的前

**keep-alive**

**connection：keep-alive**，在头部字段添加该字段，如果服务器返回的也是这个字段，就可以保持持续连接。

**timeout**

**timeout:20** 表示持续时间

**max**

**max:30** 表示最多建立多少个 Http 连接