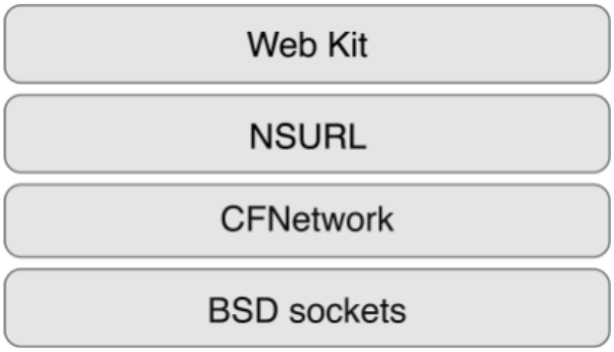
Zalo iOS Fresher: Network

Trần Đình Tôn Hiếu

1. Kiến trúc:

Theo CFNetwork Programming Guide, iOS có cung cấp một số Network framework như sau:



Hình ảnh: CFNetwork and other software layers on OS X

Trong đó, mức độ trừu tượng của các framework được sắp xếp theo thứ tự từ cao xuống thấp. CFNetwork là framework low-level, high-performance. Các framework như Web Kit hay NSURL được xây dựng dựa trên CFNetwork với độ trừu tượng cao hơn, người sử dụng sẽ ít phải quan tâm đến các giao thức mạng hơn.

Nếu tìm kiếm tài liệu về System của iOS, thì ngoài CFNetwork, còn có một framework khác cũng là low-level hỗ trợ việc thực hiện các kết nối trao đổi dữ liệu qua mạng là Network framework.

1.1. Sơ lược kiến trúc của CFNetwork:

Nếu so sánh CFNetwork với các API Network ở mức độ Foundation (điển hình là URL Loading System), thì CFNetwork tập trung nhiều vào các giao thức mạng hơn, trong khi các API ở mức độ Foundation tập trung nhiều vào truy cập dữ liệu, chẳng hạn như truyền dữ liệu qua HTTP hoặc FTP.

Kiến trúc của CFNetwork dựa trên 2 API chính là: CFSocket API và CFStream API. Cụ thể hơn:

- CFSocket API: Socket là cấp độ cơ bản nhất của Network communications. CFSocket là một phiên bản trừu tượng cao hơn của BSD sockets. CFSocket gần như tích hợp toàn bộ chức năng của BSD sockets.
- CFStream API: Các read stream và write stream cung cấp một phương thức trao đổi dữ liệu một cách độc lập với thiết bị. CFStream là một API cung cấp sự trừu tượng hoá cho các luồng đọc/ghi này thông qua 2 đối tượng mới là CFReadStream và CFWriteStream. CFStream được xây dựng dựa trên CFSocket và là nền tảng cho CFHTTP và CFFTP.

1.2. Sơ lược kiến trúc của Network framework:

Network framework sẽ giúp chúng ta tạo ra kết nối mạng để truyền và nhận dữ liệu thông qua transport và security protocols.

Sử dụng Network frame khi chúng ta muốn truy cập trực tiếp đến các giao thức tầng Transport (mô hình OSI) như TCP, UDP, TLS.

Về kiến trúc, Network framework gồm những thành phần chính như:

- nw_endpoint_t: Đối tượng đại diện cho một điểm cuối trong kết nối mạng.
- nw_parameters_t: Đối tượng lưu giữ các giao thức cho connection, các tuỳ chỉnh để gửi dữ liệu, các ràng buộc về đường truyền mạng. Ví dụ như để tạo một kết nối UDP, ta sẽ sử dụng nw_parameters_create_secure_udp() hay để tạp kết nối TCP, ta sử dụng nw_parameters_create_secure_tcp() để tạo ra parameter làm tham số cho nw_connection_create(endpoint, parameters).
- nw_connection_t: Kết nối 2 chiều giữa 2 nw_endpoint_t. Có thể tạo ra một nw_connection_t thông qua hàm nw_connection_create(endpoint, parameters) với 2 tham số là nw_endpoint_t và nw_parameters_t cho biết kết nối đi đâu và kết nối như thế nào.
- nw_listener_t: Đối tượng sử dụng để lắng nghe các kết nối mạng đi đến.

1.3. Kiến trúc của URL Loading System:

So với CFNetwork và Network framework, URL Loading System là framework ở mức độ cao hơn (Foundation). Điều này đồng nghĩa với tính trừu tượng cao hơn, sử dụng dễ dàng hơn nhưng cũng sẽ bị hạn chế can thiệp vào các giao thức mạng hơn so với 2 giao thức kia.

URL Loading System cung cấp kết nối đến tài nguyên được định nghĩa bởi URL, sử dụng các giao thức phổ biến như https hoặc tuỳ chỉnh do bạn tạo. Quá trình tải được thực hiện bất đồng bộ, nên ứng dụng vẫn có thể phản hồi tương tác của người dùng và xử lý dữ liệu/lỗi khi chúng đến.

Thành phần cơ bản nhất của URL Loading System framework là NSURLSession. NSURLSession là một đối tượng quản lý một nhóm các công việc (task) truyền dữ liệu mạng có liên quan.

NSURLSession và các class có liên quan cung cấp API để truyền tải dữ liệu đến các endpoints được chỉ định bởi URL.

Ta có thể dùng NSURLSession để tạo ra một hoặc nhiều đối tượng NSURLSessionTask – đối tượng dùng để thực thi các công việc như lấy dữ liệu về app, tải files, upload dữ liệu hoặc files. Để tuỳ chỉnh session, ta có thể sử dụng đối tượng NSURLSessionConfiguration.

Mỗi session đều được liên kết với delegate để cập nhật dữ liệu hoặc báo lỗi. Delegate mặc định sẽ gọi completion handler block mà ta cung cấp. Hoặc nếu ta muốn tự custom delegate thì khối completion handler sẽ không được gọi.

Ta cũng có thể tuỳ chỉnh để session chạy ở background, để khi ứng dụng bị treo thì quá trình download vẫn có thể tiếp tục. Hệ thống sẽ đánh thức ứng dụng lên khi quá trình download kết thúc.

*** NSURLSessionTask:**

Như đã đề cập ở trên NSURLSessionTask là một đối tượng dùng để thực thi các công việc liên quan đến network. Vậy thì cụ thể thì NSURLSessionTask sẽ thực thi các công việc đó bằng cách nào?

Thực chất, NSURLSessionTask là một lớp ảo, và bạn không thể tạo trực tiếp một NSURLSessionTask được. Bạn có thể tạo ra một đối tượng NSURLSessionTask thông qua phương thức từ NSURLSession. Và phương thức bạn gọi sẽ quyết định xem, bạn tạo ra

một đối tượng thuộc kiểu nào trong các lớp con của NSURLSessionTask, và kiểu đối tượng bạn tạo ra cũng sẽ quyết định công việc mà task có thể thực thi là gì.

Bao gồm:

- **NSURLSessionDataTask:** được tạo ra nếu bạn sử dụng NSURLSession's dataTaskWithURL:. Đối tượng NSURLSessionDataTask có thể request tài nguyên từ phía server, và nhận dữ liệu trả về thông qua một hoặc nhiều đối tượng NSData được lưu trong memory. DataTask không hỗ trợ đối với Background session. NSURLSessionData là cực kì phù hợp đối với các tương tác nhỏ đến server (như gọi các web services) vì dữ liệu trả về sẽ được ghi thẳng vào memory.

NSURLSessionDataTask có thể nhận kết quả trả về thông qua completion handler block hoặc delegate như đã trình bày chung cho các NSURLSessionDataTask ở trên.

- **NSURLSessionDownloadTask:** được tạo bởi NSURLSession's downloadTaskWithURL: hoặc downloadTaskFromResumeData:. DownloadTask khác DataTask là thay vì tải dữ liệu rồi ghi vào memory, thì DownloadTask sẽ ghi trực tiếp lên disk. Chính vì vậy, DownloadTask phù hợp cho các tác vụ cần tải dữ liệu có kích thước lớn. DownloadTask hỗ trợ mọi loại session, bao gồm cả background session.

NSURLSessionDownloadTask có cung cấp delegate để nhận thông tin về tiến trình tải.

Tập tin được tải xuống sẽ được lưu ở Temporary document. Vì vậy nên sau khi quá trình tải xuống hoàn tất, ta nên di chuyển tập tin đó sang Dictionary document.

- **NSURLSessionUploadTask:** được tạo bởi NSURLSession's uploadTaskWithRequest:fromData:. UploadTask được dùng để gửi dữ liệu đến server và nó có thể chạy trên background.

- **NSURLSessionStreamTask**: được tạo bởi NSURLSession's streamTaskWithHostName:port: hoặc streamTaskWithNetService:. StreamTask cung cấp giao diện kết nối TCP/IP. StreamTask có thể thiết lập kết nối TCP/IP từ host name và port hoặc một net service.

Ngoài ra, tất cả các thuộc tính của task đều hỗ trợ KVO (Key-Value Observing).

1.4. Legacy URL Loading Systems:

Đây là một framework cũ, tiền thân của URL Loading System, cung cấp các đối tượng thực thi các công việc tương tự như NSURLSessionTask, bao gồm: NSURLConnection, NSURLDownload, NSURLHandle. Vì đây là một framework đã cũ, và Apple cũng khuyến cáo là không nên sử dụng trên chính Documents nên em sẽ không đi sâu vào tìm hiểu framework này.

1.5. Tổng quan về App Transport Security:

Kể từ iOS 9, Apple đã giới thiệu một tính năng bảo mật mới là App Transport Security (ATS). ATS được bật mặc định trong các ứng dụng mới và thực thi các kết nối an toàn. ATS nhằm cải thiện tính riêng tư và toàn vẹn dữ liệu cho tất cả các ứng dụng và tiện ích mở rộng của ứng dụng.

ATS yêu cầu tất cả các kết nối HTTP được thực hiện bởi URL Loading System (thường sử dụng lớp NSURLSession) sử dụng HTTPS. Nó cũng áp đặt các kiểm tra bảo mật bổ sung cho việc đánh giá độ tin cậy của server được quy định bởi giao thức TLS (Transport Layer Security). ATS sẽ chặn những kết nối không đáp ứng các thông số kĩ thuật tối thiểu.

Tuy nhiên, bạn cũng có thể không áp dụng ATS cho ứng dụng của mình bằng cách thêm key NSAppTransportSecurity vào trong info.plist và set NSAllowsArbitraryLoads thành true.

2. Đánh giá ưu/khuyết điểm:

2.1. CFNetwork:

Ưu điểm	Khuyết điểm
- Performance cao. - Vì là low-level nên có thể can thiệp sâu đến các giao thức, tính tuỳ biến cao hơn. - Có thể sử dụng socket thông qua CFSocket. - Mặc dù là low-level nhưng các API được thiết kế thành các đối tượng rất dễ đọc. Tên lớp và các phương thức sử dụng camelCase.	- Sẽ khá phức tạp nếu người dùng không cần phải can thiệp sâu mà chỉ quan tâm đến việc tạo kết rối và truyền nhận dữ liệu.

2.2. Network framework:

Ưu điểm	Khuyết điểm
- Performance cao. - Có thể can thiệp trực tiếp đến các giao thức như TCP/UDP, tính tuỳ biến cao.	- Khó sử dụng. - Các đối tượng và phương thức được viết bằng snake_case.

2.3. URL Loading System:

Ưu điểm	Khuyết điểm
- Tính trừu tượng cao nên rất dễ hiểu. - Các đối tượng thiết kế rõ ràng với chức năng cụ thể, dễ sử dụng. Code sẽ trở nên đơn giản hơn. - Đáp ứng khá đầy đủ các nhu cầu cơ bản về networking.	- Không thể can thiệp trực tiếp đến các giao thức ở tầng Transport. Không hỗ trợ giao tiếp trực tiếp thông qua socket.