Concurrency & Networking

5/5’ 16

Asynchronous Tasks

Asynchronous Tasks

In iOS and OS X, each process has a main thread which is used for responding UI events.

The main thread is used to manipulate UI elements and is also the only thread which you can do so.

Heavy tasks like performing disk I/O and fetching data from the network should be moved to background threads.

Asynchronous Tasks

The `pthread`, thread model of Unix systems, is the fundamental of OS X and iOS's concurrency APIs.

The Foundation provides `NSThread` as a wrapper of low-level pthread APIs.

The `NSLocking` protocol declares the elementary methods adopted by classes that define lock objects.

By using a lock object, an application can protect critical sections of code from being executed simultaneously by separate threads.

Grand Central Dispatch

The `libdispatch`, GCD, is Apple's technology to provide support of concurrency tasks and multi-processor programming.

The fundamental idea is to move the management of the thread pool out of the hands of the developer, and closer to the operating system.

The implementation of GCD is based on thread pool pattern. Tasks are defined by closures/blocks and put into queues for execution scheduling.

Grand Central Dispatch still uses threads at the low level but abstracts them away from the programmer.

Dispatch Queues

Dispatch queues are an easy way to perform tasks asynchronously and concurrently in your application. Each queues has its own priority when competing with other queues.

Queues are very similar to threads, but they are different at all.
i.e. dispatching tasks to queues is not equal to spawning threads for tasks. The GCD manages threads for you and only creates threads when necessary.

You should treat dispatch queues like a series of task schedulers with different priorities.

Dispatch Queues Types

Serial Queue	It executes one task at a time in the order in which they are added to the queue. Serial queues are often used to synchronize access to a specific resource.
Main Queue	A globally available serial queue that executes tasks on the application's main thread. You should manipulate UI events and elements in this queue.
Concurrent Queue	It executes multiple tasks concurrently. The exact number of tasks executing at any given time is variable and depends on *system conditions*.

Dispatch Queues Priorities

QOS Level	Queue Priority	Description
User-interactive	*(Higher than High)*	Work that is interacting with the user.
User-initiated	High	Work that the user has initiated and requires immediate results.
Default	Default	This level falls between user-initiated and utility, and is not intended to be used by developers.

Dispatch Queues Priorities

QOS Level	Queue Priority	Description
Utility	Low	Work that may take some time to complete and doesn't require an immediate result. Utility tasks typically have a progress bar that is visible to the user.
Background	Background	Work that operates in the background and isn't visible to the user, such as indexing, synchronizing, and backups.

Demo: Using libdispatch

NSOperation

It's a high-level wrapper of libdispatch API.

The NSOperation class is an abstract class you use to encapsulate the code and data associated with a single task.

An operation object is a single-shot object—that is, it executes its task once and cannot be used to execute it again. You typically execute operations by adding them to an `NSOperationQueue`.

Operations could have *priorities* and dependencies between each other.

Networking

NSURL

An NSURL object represents a URL that can potentially contain the location of a resource on a remote server, the path of a local file on disk, or even an arbitrary piece of encoded data.

Data types have methods to fetch resources pointed by urls.
Like `NSString(contentsOfURL:)`

NSURLRequest and NSURLSession are the fundamental elements of networking in Foundation framework.

Use Alamofire, a third-party open-source package for networking.

JSON

{} is object/map.

[] is list.

```
{
    "firstName": "John",
    "lastName": "Doe",
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021"
    },
    "phoneNumber": [
        "212 555-1234",
        "646 555-4567"
    ]
}
```

Key: Value pairs

RESTful API

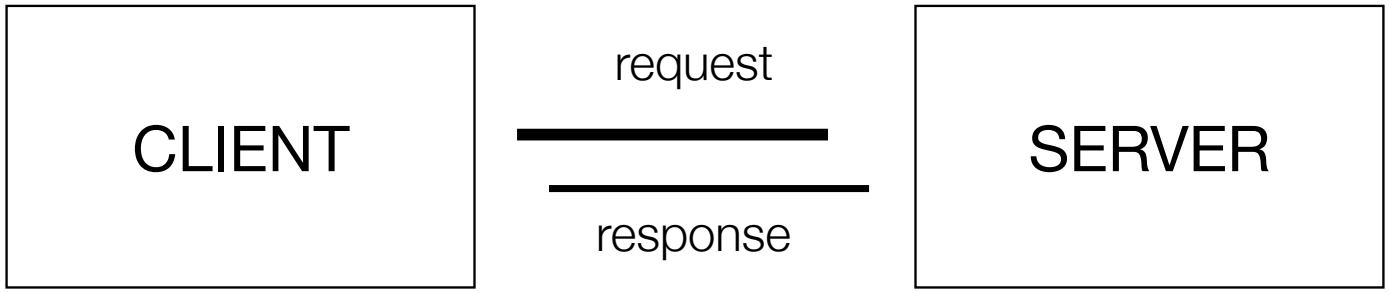
```
https://api.example.com/posts/

https://api.example.com/post/12/
```

URI is based on resources.

Using HTTP method to operate resources - CRUD.
GET, POST, PUT, DELETE, HEAD, OPTIONS

HTTP Request

GET /index.html HTTP/1.1
Host: www.example.com



HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Server: Apache/2.2.15 (Unix) (Red-Hat/Linux)
...

HTTP Messages

```
HTTP/1.1 200 OK
Date: Thr, 5 May 2016 14:31:06 GMT
Server: nginx/1.9.5
Last-Modified: Wed, 4 May 2016 08:32:59 GMT
Content-Length: 51
Connection: close
Content-Type: text/html

<!html><html><body><h1>It Works!</h1></body></html>
```

A message is composed by headers and body. They are separated by an empty new line.

HTTP QueryString

`http://api.example.com/posts/?author=rolling&series=harry%20potter`

The query string is the part after question mark, *“?”*, in the URL.

The representation form is called url-encoded.

HTTP Request Methods

HEAD

Retrieve meta-information written in response headers only.

OPTIONS

Return available HTTP methods of specific URL/Resources.

GET

Request a representation of the specified resource. Default method of HTTP Request.

POST

Submit data to be processed to the identified resource.

PUT

Uploads a new representation of the specified resource.

DELETE

Deletes the specified resource

HTTP Response Statuses

100+ Informational

Request received, continuing process.

200+ Success

The action requested by the client was received, understood, accepted and processed successfully.

300+ Redirection

The client must take additional action to complete the request.

400+ Client Error

Client seems to have erred. These are typically the most common error codes for users.

500+ Server Error

The server failed to fulfill an apparently valid request.

Thread Programming Guide
https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/Multithreading/Introduction/Introduction.html#//apple_ref/doc/uid/10000057i-CH1-SW1

Concurrency Programming Guide
https://developer.apple.com/library/ios/documentation/General/Conceptual/ConcurrencyProgrammingGuide/Introduction/Introduction.html

Concurrent Programming: APIs and Challenges
https://www.objc.io/issues/2-concurrency/concurrency-apis-and-pitfalls/