LITS

Асинхронні та мережеві операції

LITS

Що ми розглянемо

- Фонові операції
- Grand Central Dispatch
- NSOperation
- NSURLRequest, NSURLConnection, NSURLSession

LITS

“Фонова” операція

- Дві різні речі:
  - Аппа запущена системою в background-режимі
  - Операція виконується не в main thread

LITS

Потоки

- Main (UI) Thread
  - iOS UI - не thread-safe
  - Всі операції з UI мають виконуватися в main thread

LITS

Потоки

- Main (UI) Thread
  - iOS UI - не thread-safe
  - Всі операції з UI мають виконуватися в main thread
- Більш-менш низькорівневе API
  - NSThread
  - performSelectorOnMainThread…
  - performSelector:onThread…

LITS

Потоки

- Main (UI) Thread
  - iOS UI - не thread-safe
  - Всі операції з UI мають виконуватися в main thread
- Більш-менш низькорівневе API
  - NSThread
  - performSelectorOnMainThread…
  - performSelector:onThread…

LITS

GCD (grand central dispatch)

- Керує “чергами” операцій
- Розподіляє операції виконання по потокам

LITS

GCD (grand central dispatch)

```
dispatch_async(dispatch_get_main_queue(),
                                        { () -> Void in
    // ...
})
```

LITS

Swift Closure

- Objective-C - “блок”, функціональні мови програмування - “лямбда-функції”
- Можна записати у змінну, передавати як параметри, і т п

```
{ (parameters) -> returnType in

   statements...

}
```

LITS

Closures: Capturing Values

```
func myFunction() {

    let value = 2
    let obj = ...

    var closure = { in

        let y = value * 2
        obj.doSomething()


    }
}
```

LITS

Closures: Capturing Values

```
func myFunction() {

    let value = 2
    let obj = ...

    var closure = { in

        let y = value * 2
        obj.doSomething()

        self.anotherFunction()
    }
}
```

LITS

Closures: Retain cycles

```
class MyClass {

    var myClosure: ()->Void?


    func myFunction() {

        self.myClosure = { in

            self.anotherFunction()
        }
    }
}
```

LITS

Closures: Retain cycles

```
class MyClass {

    var myClosure: ()->Void?


    func myFunction() {

        self.myClosure = { [unowned self] in

            self.anotherFunction()
        }
    }
}
```

LITS

GCD (grand central dispatch)

```
dispatch_async(queue, { () -> Void in

      // do something
})

queue = dispatch_get_global_queue(identifier, 0)
```

LITS

GCD (grand central dispatch)

```
dispatch_async(queue, { () -> Void in

      // do something
})
```

```
queue = dispatch_get_global_queue(identifier, 0)
```

flags = 0
завжди

LITS

Черги виконання

Головна серійна черга

```
dispatch_get_main_queue()
```

За “quality of service”

```
dispatch_get_global_queue(QOS_CLASS_USER_INITIATED, 0)
dispatch_get_global_queue(QOS_CLASS_DEFAULT, 0)
dispatch_get_global_queue(QOS_CLASS_UTILITY, 0)
dispatch_get_global_queue(QOS_CLASS_BACKGROUND, 0)
```

За пріоритетом

```
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0)
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_HIGH, 0)
dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_LOW, 0)
```

LITS

1. Виконати операцію в іншому потоці

```
dispatch_async(dispatch_get_global_queue(
        DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), { () -> Void in

      // do something
})
```

LITS

2. Виконати операцію в головному потоці

```
dispatch_async(dispatch_get_main_queue(), { () -> Void in

    // do something
})
```

LITS

3. Виконати операцію через деякий час

(в заданій черзі)

```
dispatch_after(
        dispatch_time(DISPATCH_TIME_NOW,
                      Int64(delaySeconds * Double(NSEC_PER_SEC))),
        queue, { () -> Void in


    // do something
})
```

LITS

NSTimer

(Виконати операцію через деякий час)

```
timer = NSTimer.scheduledTimerWithTimeInterval(1.0,
      target: self, selector: "doSomething", userInfo: nil,
      repeats: false)
...


func doSomething() {
    ...
}
```

LITS

NSTimer

(Виконати операцію через проміжки часу)

```
self.timer = NSTimer.scheduledTimerWithTimeInterval(1.0,
      target: self, selector: "doSomething", userInfo: nil,
      repeats: true)
...


func doSomething() {
    ...
    if enough {
        self.timer.invalidate()
        self.timer = nil
    }
}
```

Асинхронні операції

Demo

LITS

NSOperation

```
operationQueue = NSOperationQueue()
operationQueue.queuePriority = .Low


for ... {
    let operation = NSBlockOperation( {
        // ...
    })

    operationQueue.addOperation(operation)
}
```

LITS

Мережеві операції

```
let request = NSMutableURLRequest(url,
                                  cachePolicy: .UseProtocolCachePolicy,
                                  timeoutInterval: 30.0)
request.HTTPMethod = “GET”
```

```
NSURLConnection.sendAsynchronousRequest(request, queue: self._queue,
            completionHandler: { (response: NSURLResponse?,
               resultData: NSData?, err: NSError?) -> Void in
}
```

LITS

Мережеві операції

```
let request = NSMutableURLRequest(url,
                                  cachePolicy: .UseProtocolCachePolicy,
                                  timeoutInterval: 30.0)
request.HTTPMethod = “PUT”

let paramsData = try? NSJSONSerialization.dataWithJSONObject(params,
                                              options: NSJSONWritingOptions())

request.setValue("application/json", forHTTPHeaderField: "Content-Type")
request.setValue(String(paramsData.length), forHTTPHeaderField: "Content-Length")
request.HTTPBody = paramsData

NSURLConnection.sendAsynchronousRequest(request, queue: self._queue,
            completionHandler: { (response: NSURLResponse?,
               resultData: NSData?, err: NSError?) -> Void in
}
```

LITS

Мережеві операції

NSURLSession - рекомендоване API

```
let task = NSURLSession.dataTaskWithRequest(urlRequest,
             completionHandler: { (resultData: NSData?,
             response: NSURLResponse?, err: NSError?) -> Void in
   // ...

})

task.resume()
```

LITS

Мережеві операції

iOS 9 - App Transport Security

Для non-secure urls, треба його вимкнути в Info.plist наприклад:

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

Див. “App Transport Security Technote” на developer.apple.com

LITS

Мережеві операції

Demo

LITS

Reachability

- Задача: дізнатися, чи є зараз зв’язок з інтернетом
  - Або: чи є саме wi-fi зв`язок, чи мобільні дані
- Див. на developer.apple.com:
  - SCNetworkReachability Reference
  - Reachability sample code

LITS

Отже

- Щоб писати код з асинхронними операціями, треба відійти від “процедурної” парадигми мислення
- Робота з користувацьким інтерфейсом - завжди головний потік
- Мережеві операції мають бути асинхронними (не блокувати UI thread)
    - Обробка результату мережевої операції (оновлення UI) має виконуватися на головному потоці

LITS

Самостійне завдання

1. Почитати документацію

LITS

Самостійне завдання

2 Дипломна робота