Objective-C

Курс лекций и семинаров для студентов,
желающих научиться программировать под iPhone

Осень-Зима 2013

Лекция №8

Networking: Synchrony / Asynchrony. REST. HTTp

Автор: Дмитрий Волков, iPhone Developer, Sibers

Sibers®

Что вы узнаете сегодня?

- Basic guidelines
- APIs, HTTP, Sync / Async
- NSURLConnection , NSURLRequest and NSURLConnectionDelegate
- Synchronous and Asynchronous requests
- Data formats
- JSON
- UIWebView
- Debugging
- 3rd party

Networking

На данный момент, почти каждое мобильное приложение использует какой-либо back-end сервис для сохранения информации пользователя:

- Ваш собственный сервер на PHP, .NET, Ruby, Python и др.
- Cloud-based сервис: iCloud, Azure, Google clouds и прочие
- BaaS (Backend as a Service) — Parse, Fat fractal, Stackmob и прочие

К тому же, соединение с сетью на мобильных устройствах довольно непредсказуемо, поэтому код, работающий с сетью становится довольно сложным.

Basic guidelines

При создании приложений, которые работают с внешними ресурсами посредством сетевых соединений, желательно:

- Передавать / получать минимальное необходимое количество данных.
- Позволять пользователям отменять операции, которые занимают долгое время, например, останавливать передачу или получение большого количества данных.
- Создавать возможности работы в офлайн режиме — позволять пользователю хоть чем-нибудь заниматься в офлайне.
- Используйте API, наиболее подходящий для задачи.
- Используйте известные методы защиты передачи данных: HTTPS / SSL / TSL

APIs

Cocoa предоставляет большой набор API для работы с сетью на самых разных уровнях:

- Отправка HTTP запросов (POST, GET и прочее)
- Работа с сетевыми сервисами (передача данных между приложениями в одной сети)
- Открытие сокетов и передача данных между ними
- Передача / получение потоковых данных
- Проигрывание потокового аудио / видео
- Bonjour сервисы

HTTP

- В большинстве случаев, используются high-level API для работы с сетью:

  NSURConnection

  initWithContentsOfURL:

  NSURLSession (iOS 7+)

- Так как, обычно, вы работаете с REST сервисами, большая часть общения с сервером будет происходить посредством HTTP / HTTPS протоколов.
- Для передачи данных используется протокол TCP / IP.
- Фреймворки абстрагируют какую именно технологию для передачи данных вы используете — Wi-Fi / Bluetooth / 3g / 4g / GPRS

Sync / Async

Запросы к веб-ресурсам из вашего кода могут быть выполнены в двух режимах:

- ***Синхронный*** — текущий поток останавливается, пока вы не получите данные от ресурса.
- ***Асинхронный*** — запустить операцию получения данных в другом потоке, который вам сообщит о завершении операции.

Cocoa предоставляет оба подхода:

- Для асинхронных операций используется класс ***NSURLConnection***, использующий делегирование для сообщения о прогрессе запроса.
- Для синхронных операций так же используется NSURLConnection, но и многие классы из Foundation / UIKit предоставляют методы для получения данных с какого-либо URL.

NSURLConnection

NSURLConnection — основной класс, используемый для выполнения сетевых операций, тесно связан с

NSURLRequest — класс, описывающий запрос к конкретному ресурсу по URL:

- Позволяет указывать REST-метод, использующийся для выполнения запроса
- Позволяет выставлять заголовки запроса
- Позволяет передавать данные на ресурс (все разом или потоком данных)

NSURL — класс, описывающий URL и облегчающий работу с ними.

NSURLConnection

```
NSURL* url = [NSURL URLWithString:@"http://www.google.com"];

NSURLRequest* request = [NSURLRequest requestWithURL:url];

NSURLConnection* connection = [NSURLConnection
connectionWithRequest:request delegate: self];
```

NSURLRequest

```
NSString *bodyData = @"name=Jane+Doe&address=123+Main+St";

NSMutableURLRequest *postRequest = [NSMutableURLRequest
requestWithURL:[NSURL URLWithString:@"https://www.apple.com"]];

[postRequest setValue:@"application/x-www-form-urlencoded"
forHTTPHeaderField:@"Content-Type"];

[postRequest setHTTPMethod:@"POST"];

[postRequest setHTTPBody:[NSData dataWithBytes:[bodyData UTF8String]
length:strlen([bodyData UTF8String])]];
```

NSURLConnectionDelegate

Для сообщения о текущем прогрессе соединения, ***NSURLConnection*** использует делегирование.

> **NSURLConnection** один из тех редких случаев, где объект обладает сильной ссылкой на своего делегата. **NSURLConnection** освободит своего делегата по окончанию загрузки запроса.

Основные методы, которые реализует делегат:

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data

- (void)connectionDidFinishLoading:(NSURLConnection *)connection

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error

```
- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse
*)response
{
    _mutableData = [NSMutableData new];
}
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data
{
    [_mutableData appendData: data];
}
- (void)connectionDidFinishLoading:(NSURLConnection *)connection
{
    //do something with received data
    _image = [UIImage imageWithData:_mutableData];
}
(void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error
{
    [_mutableData setLength:0];
    //report error somewhere
}
```

Synchronous requests

//NSURLConnection synchronous API

```
NSURL* url = [NSURL URLWithString:@” http://placekitten.com/g/200/300”];
NSURL* request = [NSURLRequest  requestWithURL:url];
NSData* kittenImageData = [NSURLConnection sendSynchronousRequest:request
returningResponse:nil error:nil];
UIImage* kittenImage = [UIImage imageWithData: kittenImageData];
```

//NSData synchronous API

```
NSData* data = [NSData dataWithContentsOfURL:url];
kittenImage = [UIImage imageWithData:data];
```

Asynchronous requests

```
NSOperationQueue *queue = [[NSOperationQueue alloc] init];
NSURL* url = [NSURL URLWithString:@" http://placekitten.com/g/200/300"];
NSURL* request = [NSURLRequest  requestWithURL:url];

[NSURLConnection sendAsynchronousRequest:urlRequest queue:queue
completionHandler:^(NSURLResponse *response, NSData *data, NSError
*error)
{
}];
```

Data formats

При работе с веб-сервисами приходится передавать данные.

Для работы с веб-сервисами используются форматы:

- **XML**
- **JSON**
- **SOAP**
- Ваш личный бинарный формат данных, согласованный на стороне клиента и сервера

Сосоа предоставляет парсеры форматов XML и JSON:

- XML парсер для iOS использует SAX-модель (NSXMLParser). Так же можно использовать libxml для получения DOM модели.
- JSON парсер позволяет преобразовывать json-данные в Objective-C объекты, так и наоборот (NSJSONSerialization).

JSON

JSON (JavaScript Object Notation) — формат передачи объектов, описывающий данные в стиле JavaScript «ключ-значение».

Поддерживает следующие типы данных:

- Словари (пары ключ-значение)
- Массивы (содержат любые другие типы)
- Строки
- Числа

Для работы с JSON фреймворк Foundation предоставляет класс ***NSJSONSerialization***. Так же существует несколько open-source решений: **JSONKit**, **SBJSON** и прочие.

JSON example

```
{
  "firstName": "Иван",
  "lastName": "Иванов",
  "address": {
      "streetAddress": "Московское ш., 101, кв.101",
      "city": "Ленинград",
      "postalCode": 101101
  },
  "phoneNumbers": [
      "812 123-1234",
      "916 123-4567"
  ]
}
```

JSON parsing

```
NSData* jsonData = …;
NSDictionary* personDict = [NSJSONSerialization JSONObjectWithData:jsonData
options:nil error:nil];

NSString* firstName = personDict[@”firstName”];
NSDictionary* address = personDict[@”address”];
NSNumber* postalCode = address[@”postalCode”];
NSArray* phones = personDict[@”phoneNumbers”];
NSString* phone = phones[0];

NSData* dictJSONData = [NSJSONSerialization dataWithJSONObject:personDict
options:0 error:nil];
```

UIWebView

Для отображения веб-контента внутри приложения используется наследник UIView — ***UIWebView***.

- UIWebView так же работает с **NSURLRequest**, загружая данные с указанного ресурса.
- Так же может загружать и отображать локальный HTML-код.
- Позволяет исполнять JavaScript-код
- К UIWebView так же можно добавить делегата, реализующего протокол ***UIWebViewDelegate***
- Использует WebKit

Using UIWebView

```
//url loading
UIWebView* webView = [[UIWebView alloc] initWithFrame:CGRectMake(0,0,200,200)];
[self.view addSubview: webView];
NSURL* url = [NSURL URLWithString:@"http://www.google.com"];
NSURLRequest* request = [NSURLRequest requestWithURL:url];
[webView loadRequest:request];
//html loading
NSString* htmlString = …;
[webView loadHTMLString:htmlString baseURL:nil];
//javascript execution
NSString* jsString = …;
[webView stringByEvaluatingJavaScriptFromString:jsString];
```

Debugging

Для отладки и проверки работоспособности вашего сетевого кода очень важно проверить его при различных состояниях сети.

Mac OS X и iOS предоставляют специальную утилиту — Network Conditioner для симуляции различных состояний сети.

- Позволяет как использовать стандартные предустановки для различных типов сетей (3G, Wi-Fi с потерями и прочее).
- Позволяет самостоятельно задать параметры симулируемой сети — количество потерь, задержки и т.д.

Для получения информации о типе текущего подключения используется структура ***SCNetworkReachabilityRef*** из фреймворка System Configuration.

3rd party

ASIHTTPRequest — стар и страшен, deprecated создателем. Не используем.

AFNetworking — самое используемое на данный момент стороннее решение для работы с сетью. На данный момент вышла версия 2.0.

MKNetworkingKit

Ваша самодельная абстракция.

Заключение

Сосоа предоставлет большой набор API для работы с сетевыми и веб-ресурсами:

NSURLConnection

NSURLSession

Объекты из **Core Foundation**

Системные BSD сокеты.

NSURLConnection — основной класс для работы с веб-ресурсами. Реализует неплохую поддержку HTTP-протокола.

Сосоа предоставляет парсеры для основных форматов данных, передаваемых в клиент-серверной архитектуре: XML и JSON.

Спасибо за внимание!

Жду ваших вопросов.