Лекция №6

Сетевое взаимодействие

Артур Сардарян

образование

Организационная часть

- Не нарушая традиций - отметиться

- Немного про webView
- Выясним как получать данные из сети
- Научимся парсить данные
- Демо

- Оставить отзыв (после занятия)

Сетевое взаимодействие

- показ веб-страниц
  - просто показ
  - дополнительное взаимодействие
- быстрые загрузки
  - парсинг данных
- долгие загрузки (фон)

Как показать веб-страницу

?

Показ веб-страниц

Просто показывать веб-страницы умеет SFSafariViewController

- внутри обычный safari
- общие с safari пароли и прочее
- никакого контроля со стороны приложения

Открыть в Safari

```
guard let url = URL(string: “https://google.com") else {
    return
}

UIApplication.shared.open(url)
```

Показ веб-страниц

Иногда этого недостаточно, и надо использовать WKWebView

- контроль над навигацией и вводом
- обработка actions из webView
- ускоренный js

WebSocket

Постоянно живущее двустороннее соединение поверх http

- много запросов

- получение обновлений от сервера

- сторонние библиотеки

-URLSessionWebSocketTask (iOS 13+)

Низкий уровень (TCP, UDP etc.)

Это если совсем что-то странное нужно

- CFNetworking
- Библиотеки по ссылкам

HTTP

Чаще всего нужно получать данные:

- json

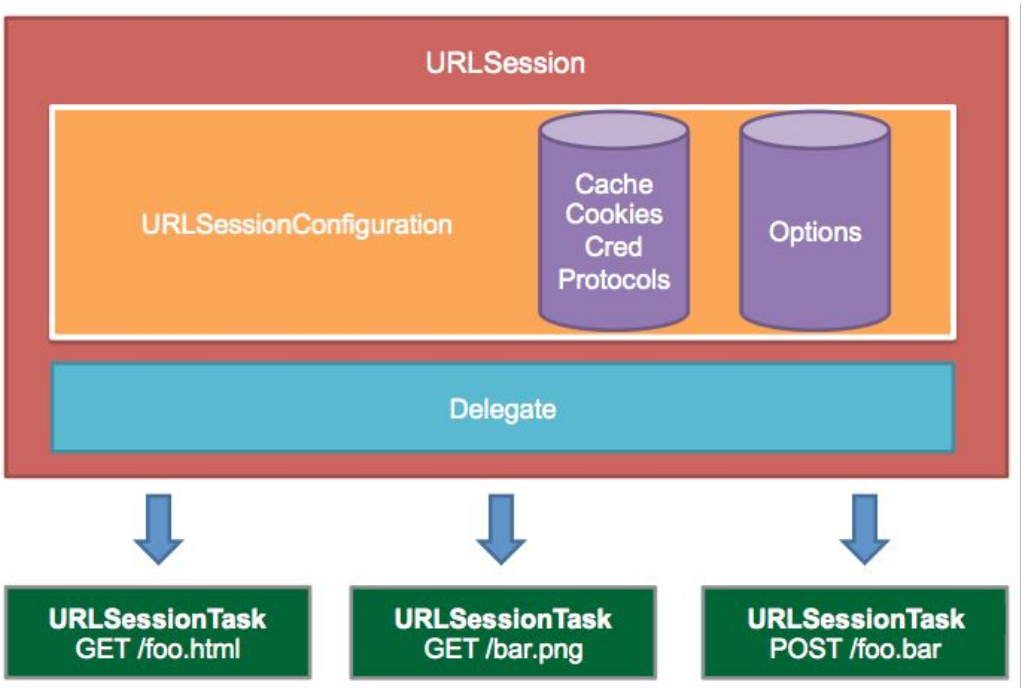
- картинки

HTTP

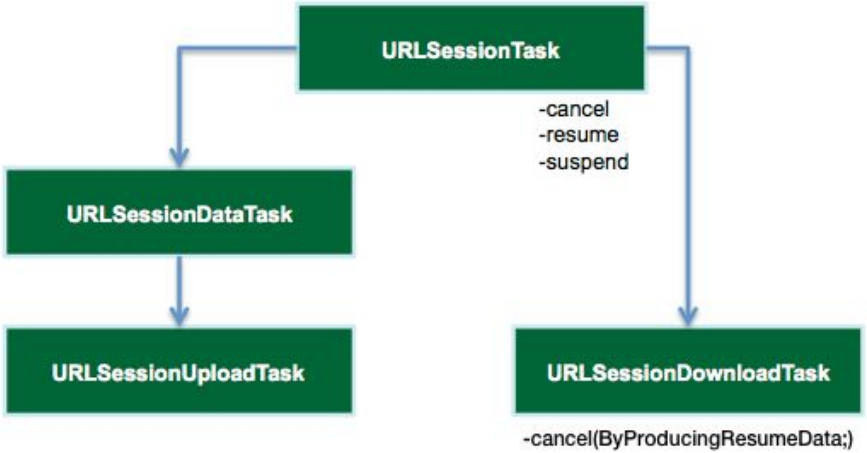
Загрузка данных как с сервера, так и на сервер делается посредством URLSession

- http, ftp протоколы
- https
- работа в фоне

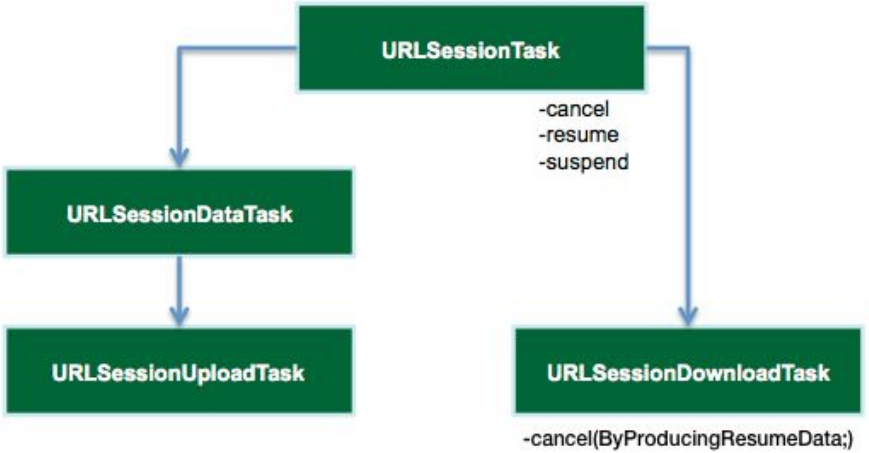
URLSession


URLSession
URLSessionConfiguration
Cache
Cookies
Cred
Protocols
Options
Delegate
URLSessionTask
GET /foo.html
URLSessionTask
GET /bar.png
URLSessionTask
POST /foo.bar

URLSessionTask



URLSessionTask



iOS13: URLSessionWebSocketTask

GET

```
63      func get() {
64          let url = URL(string: "https://test.org/get")!
65
66          let task = URLSession.shared.dataTask(with: url) { data, response, error in
67              if let error = error {
68                  print(error.localizedDescription)
69                  return
70              }
71
72              guard let data = data else {
73                  print("empty data")
74                  return
75              }
76
77              let responseJSON = try? JSONSerialization.jsonObject(with: data, options: [])
78              if let responseJSON = responseJSON as? [String: Any] {
79                  print(responseJSON)
80              }
81          }
82
83          task.resume()
84      }
```

POST

```swift
func post() {
    let json: [String: Any] = ["ket": "123",
                               "post": ["title": "Hello",
                                        "subtitle": "World"]]

    let jsonData = try? JSONSerialization.data(withJSONObject: json)

    let url = URL(string: "https://test.org/post")!
    var request = URLRequest(url: url)
    request.httpMethod = "POST"
    request.httpBody = jsonData

    let task = URLSession.shared.dataTask(with: request) { data, response, error in
        if let error = error {
            print(error.localizedDescription)
            return
        }

        guard let data = data else {
            print("empty data")
            return
        }

        let responseJSON = try? JSONSerialization.jsonObject(with: data, options: [])
        if let responseJSON = responseJSON as? [String: Any] {
            print(responseJSON)
        }
    }

    task.resume()
}
```

HTTP

При загрузке json его надо распарсить

- JSONSerialization превращает в Dictionary/Array

```
let json = try? JSONSerialization.jsonObject(with: data, options: [])
```

HTTP

JSON объект


```
{
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas tota
m\nnostrum rerum est autem sunt rem eveniet architecto"
  }
```


```
struct Post {
    let userId: Int
    let id: Int
    let title: String
    let body: String
}
```

Достаем все руками из словаря

```
extension Post {

    init?(dict: NSDictionary) {
        guard
            let userId = dict["userId"] as? Int,
            let id = dict["id"] as? Int,
            let title = dict["title"] as? String,
            let body = dict["body"] as? String
            else { return nil }

        self.userId = userId
        self.id = id
        self.title = title
        self.body = body
    }


}
```

Codable

```
typealias Codable = Encodable & Decodable
```

- JSONDecoder превращает JSON в нашу модель (класс)
- JSONEncoder превращает инстанс класса в JSON

Codable

```
struct Post: Codable {
    let userId: Int
    let id: Int
    let title: String
    let body: String

    // генерится сама
    private enum CodingKeys: String, CodingKey {
        case userId
        case id
        case title
        case body
    }
}
```

Codable

```
struct Post: Codable {
    let userId: Int
    let id: Int
    let title: String
    let body: String
}
```

JSONEncoder

```
let post = Post(userId: 1, id: 1, title: "hey", body: "you")

let encoder = JSONEncoder()
encoder.outputFormatting = .prettyPrinted

let data = try encoder.encode(post)
print(String(data: data, encoding: .utf8))
```

JSONDecoder

```
let decoder = JSONDecoder()

do {
    let posts = try decoder.decode([Post].self, from: data)
    print(posts)
} catch let error  {
    print("Parsing Failed \(error.localizedDescription)")
}
```

Сторонние штуки

- ALAMOFIRE
- MOYA
- REST KIT

Демо

- Запрос
- Парсинг
- WebView

Ссылки

- https://github.com/tidwall/SwiftWebSocket
- https://github.com/facebook/SocketRocket
- https://github.com/robbiehanson/CocoaAsyncSocket
- https://github.com/IBM-Swift/BlueSocket
- https://swiftbook.ru/post/tutorials/everything-about-codable-in-swift4/ - codable
- https://habr.com/en/post/414221/

Вопросы

Спасибо за внимание!

образование