Networking

with

O czym porozmawiamy?

- HTTP vs RESTful API
- networking a wielowątkowość
- podstawy: URL, URLRequest i URLSession
- trochę więcej o RESTful API (coming soon)

HTTP vs RESTful API

HTTP - podstawowy protokół WWW, służący do przesyłania plików w sieci

RESTful API - (w dużym skrócie) usługa sieciowa oparta na komunikacji HTTP, która umożliwia wymianę klient-serwer, najczęściej przy pomocy JSON’a

Podstawowe metody HTTP

CRUD = Create-Read-Update-Delete

GET	używane do pobierania danych
POST	używane do tworzenia contentu na serwerze
PUT	update danych na serwerze
DELETE	usuwanie danych z serwera

UWAGA

Gdy tworzymy kod networkingowy, należy pisać go w osobnym wątku. W przeciwnym wypadku może on blokować UI.

```
DispatchQueue.global(qos: DispatchQoS.QoSClass.background).async(execute: {
    doNetworkingOperation { (results) in
        DispatchQueue.main.async(execute: { showResultsInMainQueue(results) })
    }
})
```

Filary networkingu

Można powiedzieć, że to załatwia dla nas 80% operacji związanych z networkingiem:

URL

URLRequest

URLSession

URL

Reprezentacja ścieżki zasobów - od tych na dysku, aż do tych w sieci

```
var httpURL = URL(string: "https://apple.com")!
```

URLRequest

Reprezentacja requesta na bazie zadanego URL

Przykładowy ‚POST'

```
var postURL = URL(string: "https://myserver.com/path")!
var postRequest = URLRequest(url: postURL)

postRequest.httpMethod = "POST"
postRequest.setValue("application/json", forHTTPHeaderField: "Content-Type")
postRequest.setValue("application/json", forHTTPHeaderField: "Accept")
```

URLSession

Za pośrednictwem tej klasy można wykonywać URLRequest.

Gdy już korzystamy z URLSession, to raczej nie tworzymy osobnego obiektu, a korzystamy z tzw. singletonu ‚shared’:

```
URLSession.shared.dataTask(with: "whatever.com")
```

URLSession c.d.

Czasami jednak potrzebujemy ustawić bardziej szczegółowe parametry dla naszego requesta.

Wtedy przychodzi czas na skorzystanie z *URLSessionConfiguration*

```
var customConfiguration = URLSessionConfiguration.default
customConfiguration.urlCache = URLCache.init(memoryCapacity: 1000, diskCapacity: 10000, diskPath: "/")
customConfiguration.allowsCellularAccess = false
customConfiguration.httpAdditionalHeaders = ["Authorization": "Bearer 23642837642837ab89d8cbd9f8a"]
```

```
var mySession = URLSession(configuration: backgroundConfiguration, delegate: nil, delegateQueue: nil)
```

Przykład pobierania strony

```
var httpURL = URL(string: "https://apple.com")!
var httpTask = URLSession.shared.dataTask(with: httpURL) {
    (data, response, error) in
    guard let validData = data, error == nil else {
        DispatchQueue.main.async(execute: {
            print("Error getting apple website\n") })
        return
    }
    var results = String(data: data!, encoding: String.Encoding.utf8) ?? "Error decoding Apple's
        website HTML\n"
    DispatchQueue.main.async(execute: {
        print("Correctly read \(results.characters.count) characters from Apple's website
            HTML\n")
    })
}
DispatchQueue.global(qos: DispatchQoS.QoSClass.background).async(execute: {
    httpTask.resume()
})
```

Troche bardziej zaawansowane rzeczy

.

.

Coming Soon :)

JSON and stuff

Serwer z reguły zapewnia odpowiedź dla requestów w formie JSON’a.

Stąd czas oswoić tą bestie :)

Jak obsługiwać JSON'a?

Klasyczne podejście
(wcale nie trudne)

Z wykorzystaniem
Decodable
(feature w Swifcie 4)

Demo

Kod z Playgrounds będzie na githubie!