iOS の通信処理

2023/09/02 黒田 光

動機

- アプリ開発初学者の最初に作るアプリは、通信しない
- だた、世の中のアプリのほとんどは通信している

→ iOSやってる人: どんなコードで動いているか気になるよね！！！

→ iOSやってない人: iPhoneユーザー多いし気になるよね！！！

(エンジニア職限定で就活周りの話もあるよ)

自己紹介

- 名前: 黒田 光 (@1rsrx)
- 年齢: 24歳 (22卒 白石ゼミ)
- 在住: 梅林 (年内に佐賀に引っ越す予定)
- 会社: 株式会社Newbees (2023/09/01〜)
- 職種: iOSエンジニア
- 目標: リードエンジニア

職歴

- カラビナテクノロジー株式会社 (2022/04 ~ 2023/08)
    - アパレル系のECアプリ保守(iOS)
- init株式会社 (2022/10 ~ 現在)
    - プログラミングスクールのメンター
- 株式会社Newbees (2023/09 ~ 現在)
    - 30・40代向けの婚活アプリ(iOS, Android)

目次

- URLSession
- Swift Concurrency
- デモアプリの紹介
    - いらすとやの画像をダウンロードする
- デモアプリ > 画像ダウンロード機能の説明
- 就活についてのお話
    - (エンジニア職, 会社の選び方, やっておいたほうが良いこと)
- 会社の紹介

URLSession

- URLSession
  - URLのエンドポイントからデータをダウンロードしたり、アップロードするためのAPIを提供する
  - 今回はdata関数に焦点を当てる

- `data(for: URLRequest, delegate: (URLSessionTaskDelegate)?) -> (Data, URLResponse)`
  - URL先の内容をData型でダウンロードする関数
  - URLは第1引数に格納されている
  - 返り値・・・(取得したデータ, HTTP通信のヘッダー・ステータスコード)

Swift Concurrency

- Swift Concurrency (Swift5.5〜)
    - 非同期処理を簡潔に書くための言語の機能
    - async/awaitを用いて処理完了を待つことができる
    - 以前はクロージャによるコールバックが使われており、直列実行しようとすると、どうしてもネストが深くなっていた。(詳細は次のスライド)
- 非同期処理
    - 時間がかかるためメインスレッドから一時的に抜ける処理
- コールバック処理
    - ある処理が終わったあとに実行する処理
- 直列実行
    - 通信1 → 通信2 → 通信3

```
// コールバック版の直列実行
fetchData1() {
  fetchData2() {
    fetchData3() { print("処理終了" }
  }
}

// 処理1のコールバック内で処理2を実行→
// 処理2のコールバック内で処理3を実行...

// 通称コールバック地獄
// Swiftに限った話ではない
```

```
// NGパターン（完了前に次の処理がスタートする）
fetchData1() { } // 10秒かかる
fetchData2() { } // 1秒かかる
fetchData3() { } // 3秒かかる

// fetchData1() -> fetchData2() -> fetchData()3 ->
// fetchData2().CallBack -> fetchData3().CallBack ->
// fetchData1().CallBack

// 処理の重さによって完了タイミングが変わる
```

- Swift Concurrencyの使い方
  - awaitというキーワードを使って、通信が終わるまで待つことができる
  - awaitできるのはasyncな関数

```
func fetchData() async -> Data? {
    // 時間がかかる処理
    let data = ...

    return data
}

Task {
    let data1 = await fetchData()
    let data2 = await fetchData()
    let data3 = await fetchData()
}
```

```
// Before
fetchData1() {
  fetchData2() {
    fetchData3() {}
  }
}
```

アプリの紹介

画像のダウンロード方法

処理の流れ

- URL, URLRequestを生成
- URLSession.data関数でData型でダウンロードする
- データを画像に変換
- 画面に表示(省略)

URL, URLRequestの生成

```
// 文字列に日本語、無効な記号等が入っていると失敗
let url = URL(string: "https://...")!
let request = URLRequest(url: url)
```

画像をダウンロードする

```
let url = URL(string: "https://...")!
let request = URLRequest(url: url)

// 追加
Task {
   do {
        let (data, response) = try await URLSession.shared.data(for: request)
     } catch { }
}

// sharedを使うとCookie，キャッシュ...etcを共通化できる
// shared・・・シングルトンパターンの慣例的な命名。
// シングルトン・・・アプリ内でインスタンスを１つに限定するクラス設計手法
```

画像に変換する

```
let url = URL(string: "https://...")!
let request = URLRequest(url: url)

Task {
   do {
        let (data, response) = try await URLSession.shared.data(for: request)
        let image = UIImage(data: data) // オプショナル型
     } catch { }
}
```

```
__人人人人人__
> 簡単！！！<
￣Y^Y^Y^Y^Y^￣
```

まとめ

- 通信はURLSessionクラスで行う
- 通信で取得した値はData型
- Data型を期待する値・型に変換してアプリ上で表示を行う

エンジニア職について

- 良い点
  - 給与は平均より上(ぶっ飛んで高いわけでは無い)
  - 自由度が高い職場が多い(リモートワーク・フレックス)
  - 副業・独立できたりと個人の力で割とどうにでもなる
- 悪い点
  - 継続した学習が必要
    - プロジェクト参画時, 新しい技術が登場されたとき...

就活のアドバイス的な話

(エンジニア職限定)

- 会社の選び方(新卒編)
  - 開発経験が積める環境
    - エンジニア転職では開発経験が最も重要視されるから
    - 未経験〜経験2,3年の転職だと、年収を100~200万上げることが可能
    - 副業を探すときにも活かせる
  - 広く使われている技術スタック
    - 転職時、会社選びで役に立つ
  - 上記に加えて、変数を追加する
    - 給与, リモート, フレックス...

やっておいたほうが良いこと

(エンジニア職限定)

コードを書いて、公開する

開発業務ができるインターンに行く

会社の紹介

- カラビナテクノロジー株式会社
  - フルリモート・フルフレックス
  - 開発経験は職種に関わらず積める
  - インターン・新卒募集中
    - https://www.wantedly.com/companies/karabiner-tech
- 株式会社Newbees
  - 中途のみ
  - https://newbees.jp/

リソース

- ソースコード
  - https://github.com/1rsrx/ZemiCamp
  - アブストラクト、スライドも入ってます
- ご質問等あれば↓↓↓
  - h1kqaa@gmail.com
  - Twitter(@1rsrx)