Server-Side Swift a Vapor

Filip Klembara
Michal Tomlein

4. 5. 2022, FIT VUT Brno

IN2CORE

Prečo

Požiadavky

Hľadá sa jazyk pre implementáciu backendu

Požiadavky

- Typová bezpečnosť
- Rýchlosť
- Udržateľnosť kódu v malom tíme
- Framework
- Potenciál do budúcnosti
- Minimalizácia počtu šedivých vlasov

Aké sú možnosti?

Krátke subjektívne porovnanie

Skriptovacie jazyky

PHP, Python, Perl, Ruby

- Overené
- Bohatý ekosystém
- Pomalé
- Pravdepodobne správna voľba

- ~~Typová bezpečnosť~~
- ~~Rýchlosť~~
- ~~Udržateľnosť kódu v malom tíme~~
- Framework
- Potenciál do budúcnosti
- Minimalizácia počtu šedivých vlasov

Kompilované VM-based jazyky

Java, JVM-based jazyky, C#

- Statické typy (väčšinou)
- Rýchlejšie
- Tiež bohatý ekosystém
- Enterprise-friendly

- Typová bezpečnosť
- Rýchlosť
- ~~Udržateľnosť kódu v malom tíme~~
- Framework
- Potenciál do budúcnosti
- Minimalizácia počtu šedivých vlasov

Nové low-level kompilované jazyky

Go, Rust

- Statické typy
- Dôraz na bezpečnosť
- Paralelizmus
- Rýchle
- Menej rozvinutý ekosystém

- Typová bezpečnosť
- Rýchlosť
- ~~Udržateľnosť kódu v malom tíme~~
- Framework
- Potenciál do budúcnosti
- Minimalizácia počtu šedivých vlasov

JavaScript

- Zdieľanie kódu s front-endom

- ~~Typová bezpečnosť~~
- Rýchlosť
- Udržateľnosť kódu v malom tíme
- Framework
- Potenciál do budúcnosti
- ~~Minimalizácia počtu šedivých vlasov~~

Swift

Server-Side

- Zdieľanie kódu s iOS/macOS
- Statické typy
- Dôraz na bezpečnosť
- Paralelizmus
- Rýchly
- ...
- Takmer neexistujúci ekosystém

- Typová bezpečnosť
- Rýchlosť
- Udržateľnosť kódu v malom tíme
- Framework
- Potenciál do budúcnosti
- Minimalizácia počtu šedivých vlasov

V čom Swift vyniká na serveri

Codable

Enums

Codable Enums

Concurrency

Completion Handlers

```
func fetchThumbnail(for id: String, completion: @escaping (UIImage?, Error?) -> Void) {
    let request = thumbnailURLRequest(for: id)
    let task = URLSession.shared.dataTask(with: request) { data, response, error in
        if let error = error {
            completion(nil, error)
        } else if (response as? HTTPURLResponse)?.statusCode != 200 {
            completion(nil, FetchError.badID)
        } else {
            guard let image = UIImage(data: data!) else {
                completion(nil, FetchError.badImage)
                return
            }
            image.prepareThumbnail(of: CGSize(width: 40, height: 40)) { thumbnail in
                guard let thumbnail = thumbnail else
                    completion(nil, FetchError.badImage)
                    return
                }
                completion(thumbnail, nil)
            }
        }
    }
    task.resume()
}
```

Zdroj: WWDC 2021, 10132

Futures/Promises

```
func fetchThumbnail(for id: String) -> Future<UIImage> {
    let request = thumbnailURLRequest(for: id)
    let task = URLSession.shared.dataTask(with: request)
    task.resume()
    let image = task.mapThrowing { response, data -> UIImage in
        guard (response as? HTTPURLResponse)?.statusCode != 200 else {
            throw FetchError.badID
        }
        guard let image = UIImage(data: data!) else {
            throw FetchError.badImage
        }
        return image
    }
    return image.flatMap { $0.prepareThumbnail(of: CGSize(width: 40, height: 40)) }
}
```

Async/Await

```
func fetchThumbnail(for id: String) async throws -> UIImage {
    let request = thumbnailURLRequest(for: id)
    let (data, response) = try await URLSession.shared.data(for: request)
    guard (response as? HTTPURLResponse)?.statusCode == 200 else {
        throw FetchError.badID
    }
    guard let thumbnail = await UIImage(data: data)?.thumbnail else {
        throw FetchError.badImage
    }
    return thumbnail
}
```

Zdroj: WWDC 2021, 10132

Actors

```
actor MessageNotificationCenter {
    private var state: [UUID: NotificationHandler] = [:]

    nonisolated func startListening(on database: any Database) { … }

    func notifyAll(_ notification: Notification) {
        state.forEach { $0.value(notification) }
    }
}

let notificationCenter: MessageNotificationCenter = …

// Bez prístupu k `state`
notificationCenter.startListening(on: db)

Task {
    // S prístupom k `state`
    await notificationCenter.notifyAll(.init("Hello"))
}
```

Distributed Actors

Ako

Zhrnutie

- Typy sú silný nástroj
- Concurrency cez async/await
- Actors chránia zdieľaný stav
- Multi-server chat sa dá napísať na pár riadkov
- Databázové notifikácie sú fajn
- Pridajte sa k nám: we’re hiring!
- Diplomové a bakalárske práce v spolupráci s IN2CORE