EVOLUCIONANDO CON SWIFT

Como escribir código más expresivo y seguro utilizando las nuevas funcionalidades del lenguaje.

MARCOS GRISELLI

iOS Developer @ Toptal

EVOLUCIÓN

SWIFT 1.2

» flatMap

» @noescape

» Set

» Multiple if-let

SWIFT 2

» try/catch

» guard

» defer

» API availability

SWIFT 2.1

» String interpolation

```
"String \(value)"
```

SWIFT 2.2

Primer release desde que es open source

» C style deprecations (++, --, loops)

» var parameters deprecation

SWIFT 3

» Better Translation of Objective-C APIs Into Swift

» Apply API Guidelines to the Standard Library

» ~100 proposals

SWIFT 3.1

» Concrete Constrained Extensions

» Nested Generics

SWIFT 4

» Codable

» String revision (SE-0163)

» Multi-line string literals

» Improve keypaths and Dictionary

SWIFT 4.1

» Synthesized Equatable and Hashable (SE-0185)

» Conditional conformances (SE-0143)

» Key decoding strategy for Codable

» Recursive constraints on associated types

» Introduce Sequence.compactMap(_:)

HASHABLE & EQUATABLE

SWIFT 4.0

```
struct User: Decodable {
    let id: String
    let name: String
    let email: String
    let registerDate: Date
    let talks: [Talk]
}
```

```swift
// MARK: - Hashable
extension User: Hashable {
    var hashValue: Int {
        return combineHashes([id.hashValue,
                              name.hashValue,
                              email.hashValue,
                              registerDate.hashValue,
                              talks.hashValue])
    }
}
```

```swift
// MARK: - Equatable
extension User: Equatable {
    static func == (lhs: User, rhs: User) -> Bool {
        return lhs.id == rhs.id &&
            lhs.name == rhs.name &&
            lhs.email == rhs.email &&
            lhs.registerDate == rhs.registerDate &&
            lhs.talks == rhs.talks
    }
}
```

HASHABLE & EQUATABLE EN SWIFT 4.0

» Implementación duplicada

» Propenso a errores al modificar nuestras estructuras

» Complejo de agregar en ciertos casos (enum con associated values)

SE-0185
SYNTHESIZED EQUATABLE AND HASHABLE

SWIFT 4.1

```
struct User: Decodable, Hashable, Equatable {
    let id: String
    let name: String
    let registerDate: Date
    let talks: [Talk]
}
```

» Eliminar boilerplate y errores del programador.

» Conformar a Hashable o Equatable de manera simple.

SE-0143
CONDITIONAL CONFORMANCES

“THE ABILITY FOR TYPES THAT STORE OTHER TYPES TO CONFORM TO A PROTOCOL”

https://swift.org/blog/conditional-conformance/

Esto puede verse reflejado en Arrays u Opcionales

```swift
extension Array: Equatable where Element: Equatable {
    // implementation of == for Array
}

extension Optional: Equatable where Wrapped: Equatable {
    // implementation of == for Optional
}
```

```swift
/// [Int?]
let a = [Int("1"), Int("2")]
let b = [Int("1"), Int("2")]

a == b // Swift 4.0 ❌
a == b // Swift 4.1 ✅
```

COMO PODEMOS UTILIZARLO?

```swift
struct SomeWrapper<Wrapped> {
  let wrapped: Wrapped
}

protocol HasIdentity {
  static func ===(lhs: Self, rhs: Self) -> Bool
}

extension SomeWrapper: Equatable where Wrapped: Equatable {
  static func ==(lhs: SomeWrapper<Wrapped>, rhs: SomeWrapper<Wrapped>) -> Bool {
    return lhs.wrapped == rhs.wrapped
  }
}

// error: SomeWrapper already stated conformance to Equatable
extension SomeWrapper: Equatable where Wrapped: HasIdentity {
  static func ==(lhs: SomeWrapper<Wrapped>, rhs: SomeWrapper<Wrapped>) -> Bool {
    return lhs.wrapped === rhs.wrapped
  }
}
```

» Podemos crear wrappers alrededor de cualquier tipo y aún conformar a los protocolos que necesitemos.

» Con ayuda de Phantom Types podemos generar tipos que sean únicos aunque el tipo que contengan sea el mismo.

TAGGED
BY POINTFREE

```
struct Tagged<Tag, RawValue> {
    var rawValue: RawValue
}
```

CONDITIONAL CONFORMANCES

```swift
// MARK: - Equatable
extension Tagged where RawValue: Equatable {
    static func == (lhs: Tagged, rhs: Tagged) -> Bool {
        return lhs.rawValue == rhs.rawValue
    }
}

// MARK: - Decodable
extension Tagged: Decodable where RawValue: Decodable {
    init(from decoder: Decoder) throws {
        self.init(rawValue: try decoder.singleValueContainer()
        .decode(RawValue.self))
    }
}
```

XCODE!

SWIFT 4.2

» Enhanced Hashable

» Dynamic Member Lookup (SE-0195)

» Case Iterable (SE-0194)

» Random API

» Warnings and Errors (SE-0196)