Балалаев Сергей

IOS TECH LEAD

Swift 4

1. Почему Swift крут для бизнеса?

- Улучшение работы приложения: безотказность, безопасность, производительность
- iOS (с 7.0), OS X (с 10.9), watchOS, tvOS, и другие (например Linux)
- Упрощена читаемость и понимание кода
- Легок в обучении, можно легко переучиться с Obj-C
- OpenSource
- Использования не только на железе Apple
- Легко обновлять приложения, написанные на нём
- Упрощенная кросплатформенная разработка
- Интерактивное программирование: playgrounds
- Есть задел для будующего роста

2. Принципиальные отличия Swift от Objective-C

- Всё в одном файле .swift
- Есть REPL (Read-Eval-Print Loop): Playground
- Generic и строгая типизация, которая позволяет опускать в ряде случаев тип объекта для компилятора
- Появился Struct
- Расширен Enum
- Option & Non-option: защита доступа объектов
- Появились виртуальные таблицы классов помимо динамической адресации методов из Objective-C
- модификаторы доступа
- Упрощенная и кастомизируемая сериализация

3. Чистое использование Swift без примеси Objective-C

- Протокол ориентированный подход
- Максимальная польза Enum
- В качестве модели использование Struct
- Реализация протокола Codable
- Варианты Option

4. Enum

```
enum Gender : String { // enum with type

    case male = "I am a man!"
    case female = "I am the best"
    case uni = "I don't know"

}

func printAll() {

    print(Gender.uni) // prints "uni"

    print(Gender.male.rawValue) // prints "I am a man!"

    print(Gender.female.hashValue) // prints 1

}
```

5. Enum для статических таблиц

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
{
    return Cells.count.hashValue
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell
{
    switch indexPath.row {
    case Cells.allCountry.index:
        return tableView.dequeueReusableCell(
            withIdentifier: Cells.allCountry.identifier, for: indexPath)
    case Cells.nameCountry.index:
        return tableView.dequeueReusableCell(
            withIdentifier: Cells.nameCountry.identifier, for: indexPath)
    case Cells.codeCountry.index:
        return tableView.dequeueReusableCell(
            withIdentifier: Cells.codeCountry.identifier, for: indexPath)
    default:
        return UITableViewCell()
    }
}
```

6. Удобства Enum

```
fileprivate enum Cells: String
{
    case allCountry = "All information"
    case nameCountry = "Only name of countries"
    case codeCountry = "Only code of countries"

    // It's finished. That should be end.
    case count

    var index : Int { return hashValue }
    var identifier : String { return "\(self)" }
    var title: String { return rawValue }

    static func getFrom(index: Int) -> Cells? {
        return allValues()[index]
    }
}
```

7. Enum как величина

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
{
    return Cells.count.hashValue
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) ->
    UITableViewCell
{
    guard let cellType = Cells.getFrom(index: indexPath.row) else {
        return UITableViewCell()
    }
    return tableView.dequeueReusableCell(
        withIdentifier: cellType.identifier, for: indexPath)
}
```

8. Protocols

```
protocol CountryProtocol {
    var name: String {get}
    var code: String {get}
}

struct CountryEntity: CountryProtocol {
    var name: String
    var code: String

    init(name: String, code: String) {
        self.name = name
        self.code = code
    }
}
```

9. Протокол ориентированный подход

```
16 protocol CountriesListCellProtocol {
17
18     func update(from entity: CountryProtocol)
19
20 }
21
22 class CountriesListCell: UITableViewCell, CountriesListCellProtocol {
23
◉      @IBOutlet weak var codeLabel: UILabel!
◉      @IBOutlet weak var nameLabel: UILabel!
26
27 }
28
29 extension CountriesListCellProtocol where Self: CountriesListCell {
30
31     func update(from entity: CountryProtocol) {
32         codeLabel.text = entity.code
33         nameLabel.text = entity.name
34     }
35
36 }
```

10. Реализации ячеек таблицы

```
func tableView(_ tableView: UITableView,
               cellForRowAt indexPath: IndexPath) -> UITableViewCell
{
    let cellType = ...

    let entity = countries[indexPath.row]

    let cell = tableView.dequeueReusableCell(withIdentifier: cellType.cellID,
                                             for: indexPath)

    if let cell = cell as? CountriesListCellProtocol {
        cell.update(from: entity)
    }

    return cell
}
```

11. Профит ПОПа

```
extension CountriesListCellProtocol where Self: CountriesListCell {
    func update(from entity: CountryProtocol) {
        codeLabel.text = entity.code
        nameLabel.text = entity.name
    }
}

extension CountriesListCellProtocol where Self: CountriesCodeListCell {
    func update(from entity: CountryProtocol) {
        textLabel?.text = entity.code
    }
}

extension CountriesListCellProtocol where Self: CountriesNameListCell {
    func update(from entity: CountryProtocol) {
        textLabel?.text = entity.name
    }
}
```

Практика

https://github.com/Altarix/MeetupSwiftCountry.git

2-й блок

1. Struct VS Class

1. struct всегда копирует поля
2. struct не наследуемый и потому быстрее

```
struct CountryEntity
{
    var name: String
    var code: String

    init(name: String, code: String) {
        self.name = name
        self.code = code
    }
}
```

2. Protocols

```
protocol CountryProtocol {
    var name: String {get}
    var code: String {get}
}

struct CountryEntity: CountryProtocol {
    var name: String
    var code: String

    init(name: String, code: String) {
        self.name = name
        self.code = code
    }
}
```

3. Использование протокола Codable

```
struct CountryEntity: CountryProtocol, Codable {
    var name: String
    var code: String
}
```

```
let decoder = JSONDecoder()
do {
    result = try decoder.decode([CountryEntity].self,
                                from: responseData as Data)
} catch {
    print("""
        error trying to convert data to JSON
        from '\(mockupFile)'
        """)
    print(error)
}
```

4. Переменные Option / Non option

```
16  class CountryDetailsController: UIViewController {
17
        @IBOutlet weak var editButton: UIBarButtonItem!
        @IBOutlet weak var codeTextField: UITextField!
        @IBOutlet weak var nameTextField: UITextField!
21
22      var country: CountryEntity?
23
24      var editHandler: ((_:CountryEntity) -> Void)? = nil
25
26      var firstIndex: Int = 0
27
28
```

5. Значения Option / Non option

```
    @IBAction func editClick(_ sender: Any) {
        if let country = country {
            editHandler?(country)
        }
        navigationController!.popViewController(animated: true)
    }

    @IBAction func codeEdit(_ sender: Any) {
        guard var country = country else {
            return
        }
        country.code = codeTextField.text ?? ""
        country.name = ""
    }

    @IBAction func nameEdit(_ sender: Any) {
        country?.name = nameTextField.text ?? ""
    }
```

6. Плохая практика

```
@IBAction func editClick(_ sender: Any) {
    navigationController!.popViewController(animated: true)
}

@IBAction func nameEdit(_ sender: Any) {
    country?.name = nameTextField.text!
}

@IBAction func codeEdit(_ sender: Any) {
    if var country = country {
        country.code = codeTextField.text ?? ""
    } else {
        print("country is not initialized")
    }
}
```

7. if let & if var

1. Работа с полем без прерывания работы метода
2. Частое использование поля
3. Подразумевается реакция на пустое значение

```
@IBAction func codeEdit(_ sender: Any) {
    if var country = country {
        if let code = codeTextField.text {
            country.code = code
        } else {
            print("country textField is null!")
        }
    } else {
        print("country is not initialized")
    }
}
```

8. guard

1. Требуется прерывание работы метода
2. Еще более частое использование поля чем if let
3. Необратимая реакция на пустое значение

```
func update() {
    guard let country = country else {
        print("country is not initialized")
        return
    }
    nameTextField.text = country.name
    codeTextField.text = country.code
    navigationItem.title = country.code
}
```

9. Другие формы сокращения

```
@IBAction func editClick(_ sender: Any) {
    update()
    if let country = country
    {
        editHandler?(country)
    } else {
        let code = codeTextField.text ?? ""
        let name = nameTextField.text ?? ""
        let country = CountryEntity(name: name,
                                    code: code.uppercased())
        addHandler!(country)
    }
    navigationController?.popViewController(animated: true)
}
```

Практика

https://github.com/Altarix/MeetupSwiftCountry.git

Источники

1. https://www.indianappdevelopers.com/blog/swift-next-gen-mobile-app-programming-language/
2. https://medium.com/@abhimuralidharan/enums-in-swift-9d792b728835
3. https://www.maxmind.com/en/free-world-cities-database
4. https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/OptionalChaining.html
5. https://medium.com/@johnsundell/handling-non-optional-optionals-in-swift-e5706390f56f
6. https://developer.apple.com/videos/play/wwdc2015/408/

СПАСИБО

Сергей Балалаев

sof.bix@mail.ru