Как построить модульность

на SPM с мультирепой
и не поседеть

Ренат Гафаров, ВТБ, 2021

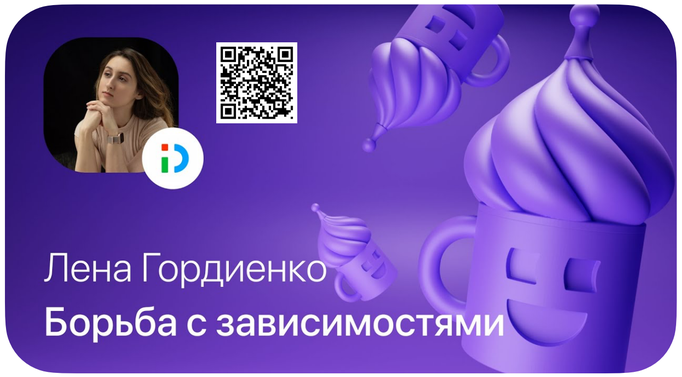
Сломай и изучи!

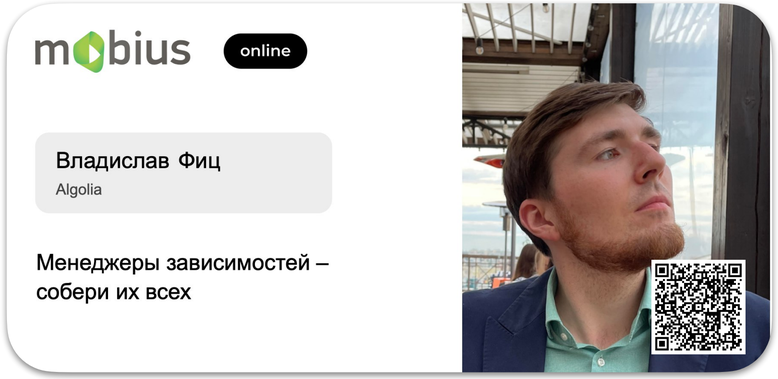
О чем мы будем говорить?

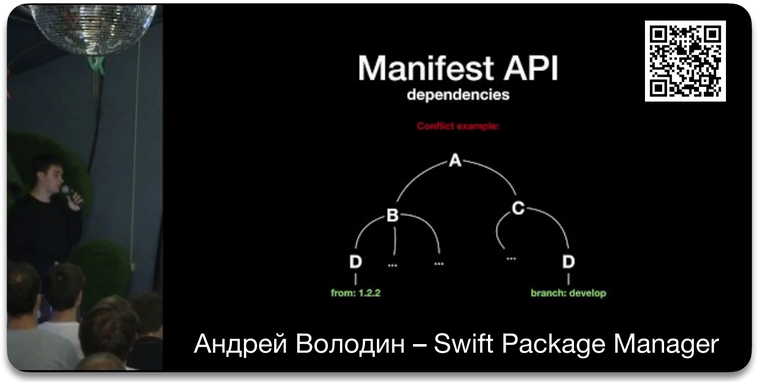
Введение

1. Модульность
2. Swift Package Manager
3. Проблемы SPM

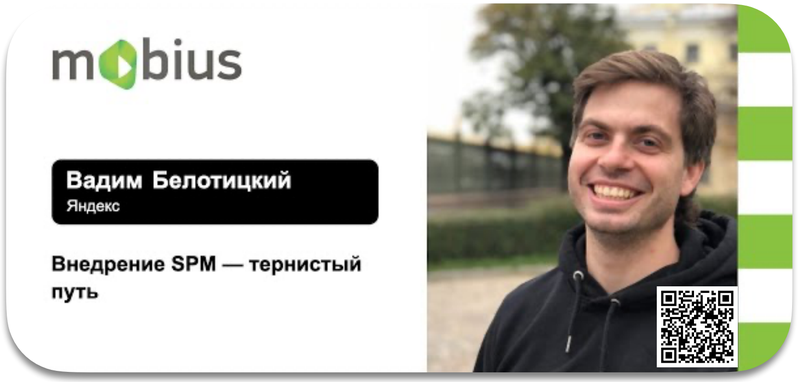
Доклады

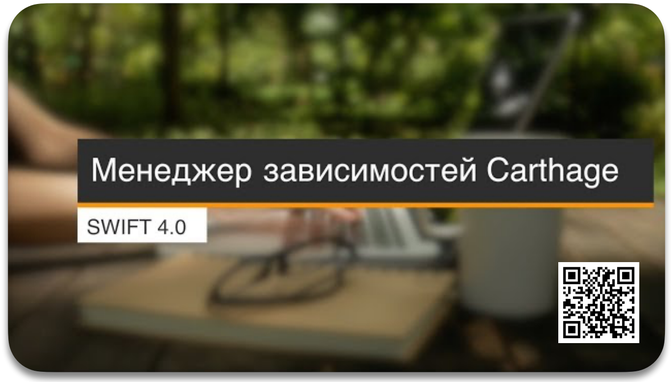

Лена Гордиенко
Борьба с зависимостями


mobius
online
Владислав Фиц
Algolia
Менеджеры зависимостей –
собери их всех

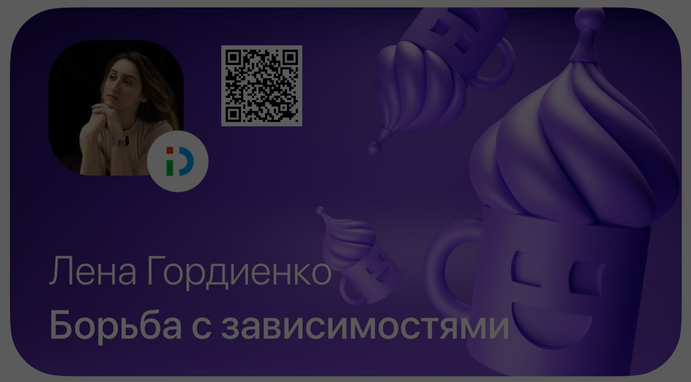

Manifest API
dependencies
Conflict example:
A
B
C
D
D
from: 1.2.2
branch: develop
Андрей Володин – Swift Package Manager

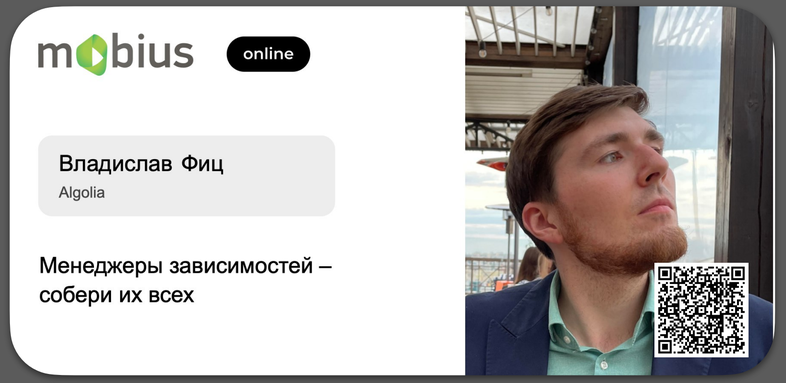

13.09
Модуляризируем свое приложение с помощью собственных Pod'ов
Султан Сейдалин
iOS/Android разработчик, Arbuz.kz
CocoaHeads #2
Kazakhstan

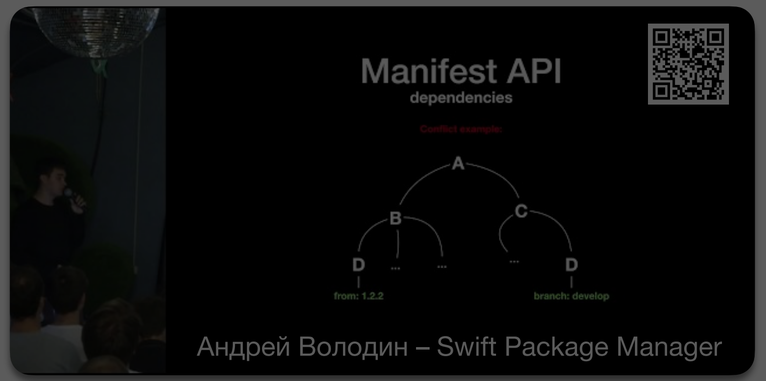

mobius
Вадим Белотицкий
Яндекс
Внедрение SPM — тернистый путь

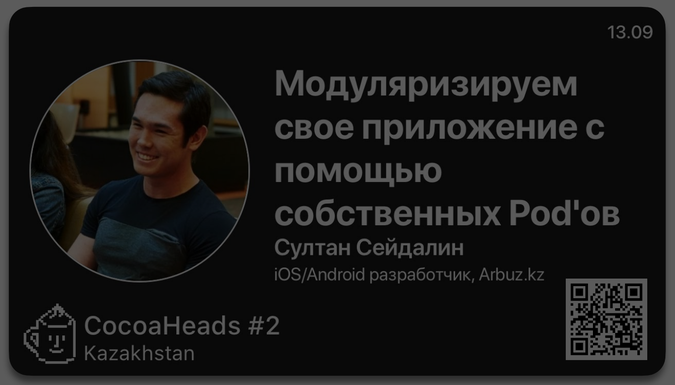

Менеджер зависимостей Carthage
SWIFT 4.0

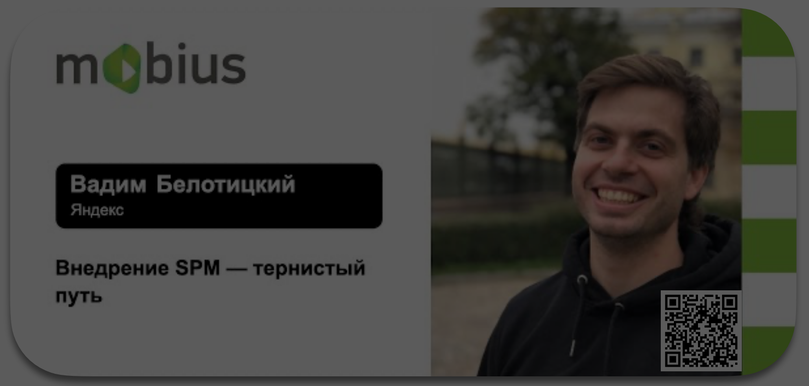
Доклады

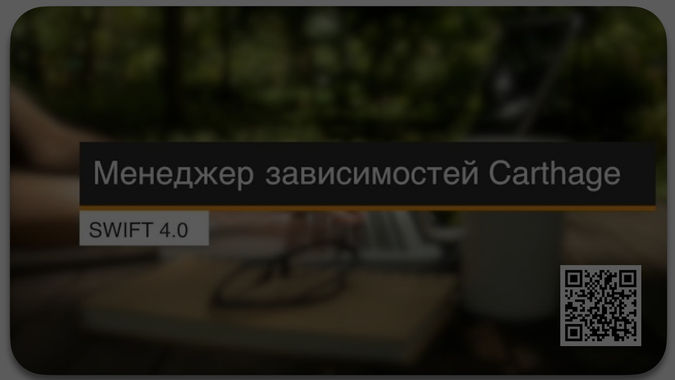

Лена Гордиенко
Борьба с зависимостями


mobius
online
Владислав Фиц
Algolia
Менеджеры зависимостей –
собери их всех


Manifest API
dependencies
Conflict example:
A
B
C
D
D
from: 1.2.2
branch: develop
Андрей Володин – Swift Package Manager


13.09
Модуляризируем свое приложение с помощью собственных Pod'ов
Султан Сейдалин
iOS/Android разработчик, Arbuz.kz
CocoaHeads #2
Kazakhstan


mobius
Вадим Белотицкий
Яндекс
Внедрение SPM — тернистый путь


Менеджер зависимостей Carthage
SWIFT 4.0

Что было?

Минутка рекламы

ВТБ Онлайн 2019

40 iOS разработчиков

CocoaPods (LocalPods(3) + RemotePods(29))

Монолитное приложение

Холодная компиляция < 10 мин

Инкрементальная - плавающая, зависит от графа зависимостей файла

А что сейчас?

Минутка рекламы **ВТБ Онлайн 2021**

140 iOS разработчиков

SPM(38) + CocoaPods(1)

Многомодульное приложение

Холодная компиляция ~ 30-40 мин

Инкрементальная - пересобирается только модуль в который внесены правки. < 5 мин(как правило секунд 40)

Когда пора резать приложение на модули?

Введение

- На этапе проектирования(но на самом деле никогда не поздно)
- Количество кода растет, а время компиляции падает
- Довольно часто случается сборка на “холодную”
- Доработка собственных SDK проблемна в релизе
- Есть потенциальная возможность переиспользовать модули в рамках компании

Микро-модульный подход

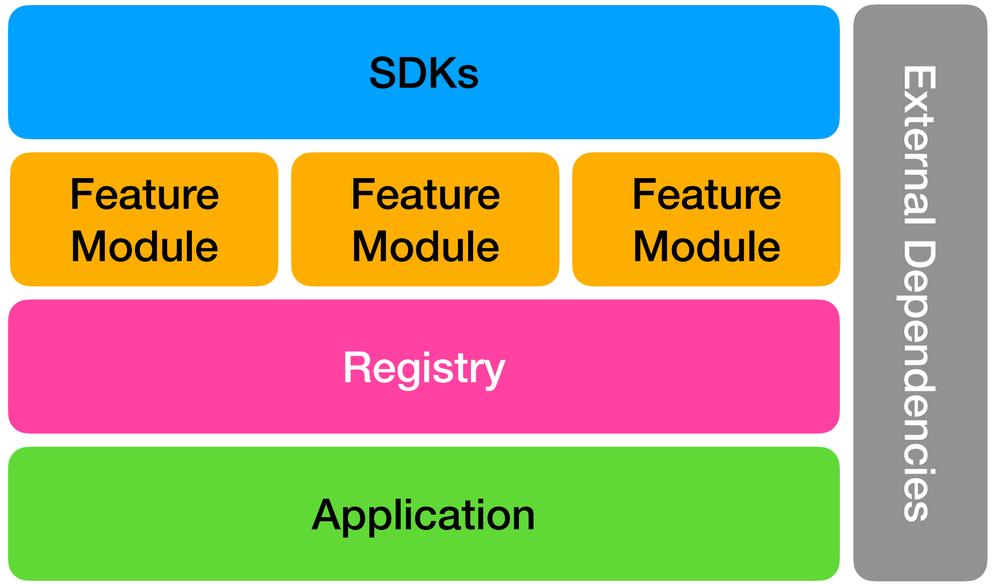
Модульность

- Фичи выделяются в таргеты
- Таргеты имеют явный интерфейс и зависимости

Макро-модульный подход

Модульность

- Явное разделение слоев
- Отсутствие горизонтальных зависимостей
- Независимые функциональные модули
- Единый интерфейс функциональных модулей



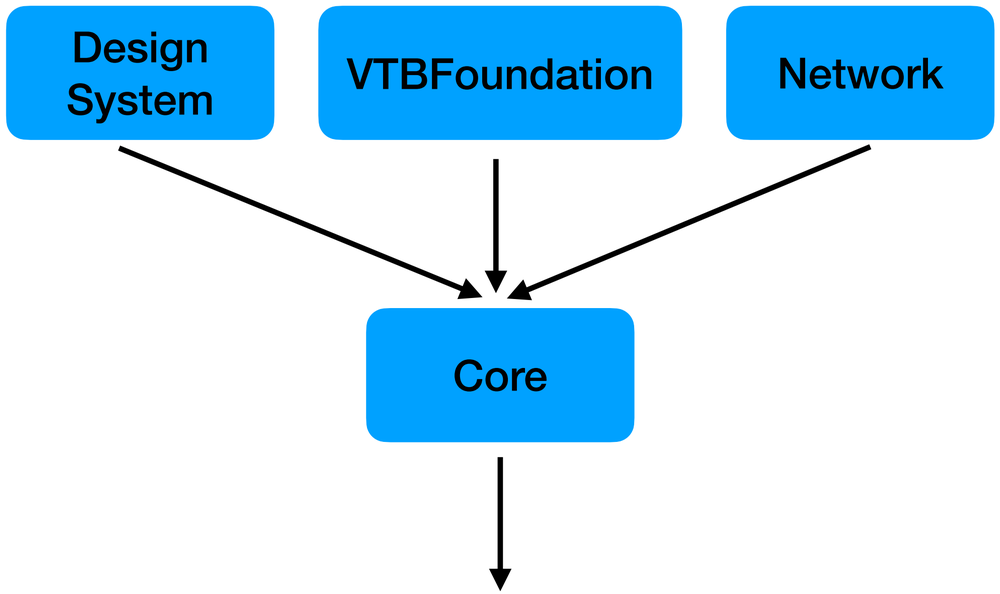
Registry
Модульность

- Регистр всех зависимостей
- Включает в себя логику по поддержанию инфраструктуры модульности
- Единая точка контроля версий всех зависимостей

SDK

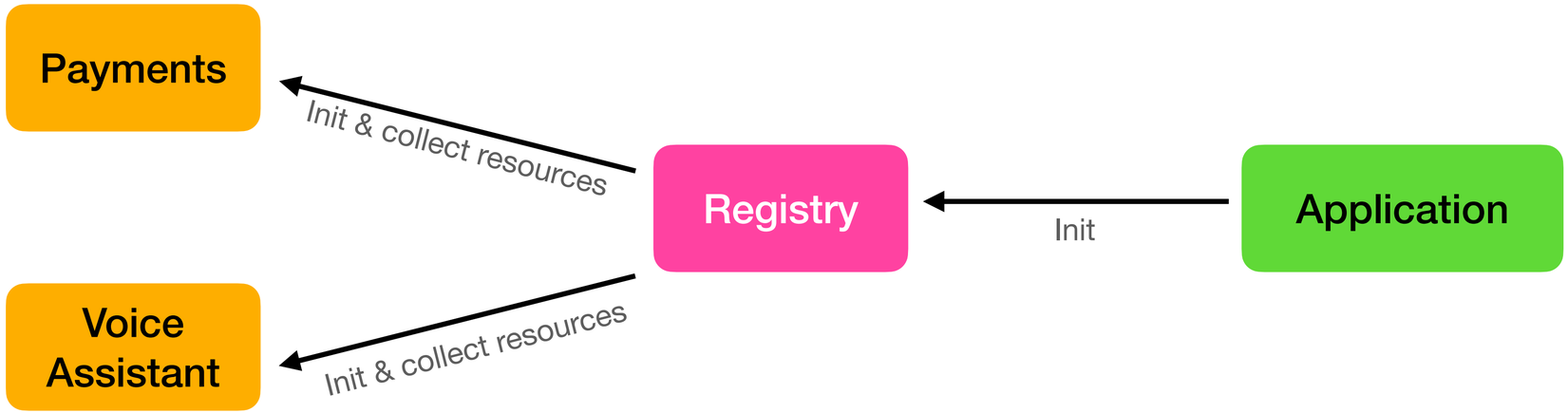
Модульность

- Набор инструментов
- Разделение по слоям
- Реализация шаблона модуля



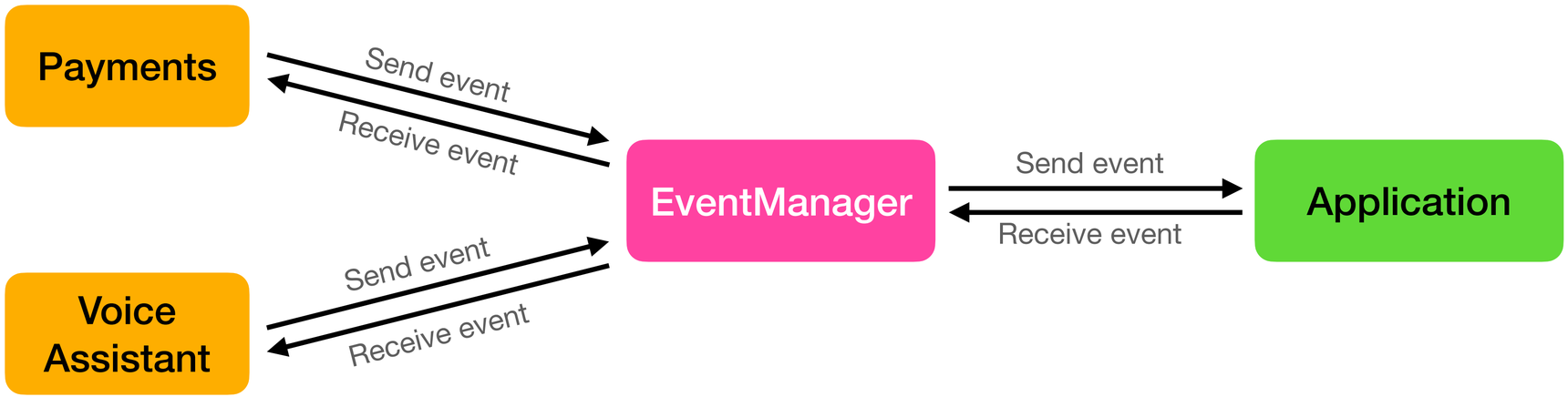
Инициализация

Модульность



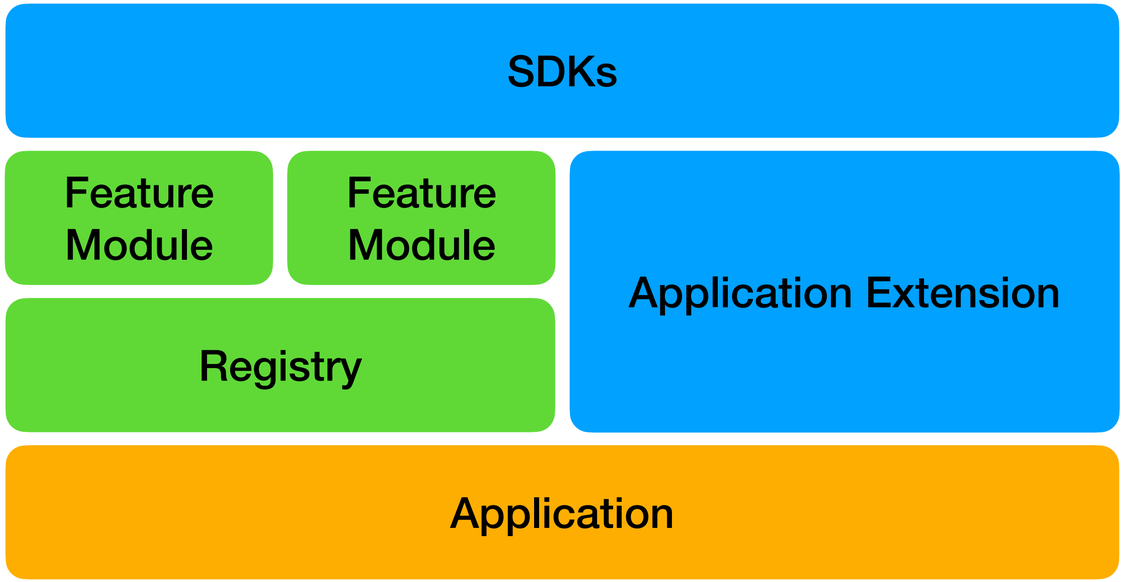
Общение с модулями

Модульность



Как будем линковать?

Модульность



Static

Dynamic

Подытожим

Модульность

- 4 слоя: SDK, Функциональные модули, Регистр, Приложение
- Функциональные модули имеют единый интерфейс
- Функциональные модули независимы друг от друга
- Межмодульное взаимодействие строится на обмене событиями

Менеджеры зависимостей

CocoaPods

Carthage

Swift Package Manager

Swift Package Manager

Модульность

Swift PM

Quick intro

Swift Package Manager

- Нативный и интегрированный в Xcode
- Манифест файл на Swift
- Простое версионирование
- Умеет работать с бинарными файлами
- Локальные и Remote пакеты
- Сделан с прицелом на мультиплатформу
- Относительно большое коммьюнити на forums.swift.org

Звучит многообещающе

```
$  ~ mkdir EmptyPackage
$  ~ cd EmptyPackage
$  swift package init
```

Package Manifest

Swift Package Manager

- Package.swift вместо .xcodeproj
- Только 2 конфигурации: [Debug, Release]
- Можно использовать переменные окружения для формирования манифеста пакета

Package.swift вместо .xcodeproj

Swift Package Manager

```
// swift-tools-version:5.5

import PackageDescription

let package = Package(
    name: "EmptyPackage",
    products: [
        .library(name: "EmptyPackage", targets: ["EmptyPackage"]),
    ],
    targets: [
        .target(name: "EmptyPackage")
    ]
)
```

- Package.swift описывает всю сборку проекта
- Понятная и простая структура
- Отсутствуют XCConfig
- Можно сгенерировать .xcodeproj (will be deprecated soon)
- Каждый пакет можно собирать собственной версией Swift PM

Конфигурации
Swift Package Manager

- Доступные конфигурации - Debug и Release
- Для каждой конфигурации прописан свой набор флагов
- [-wmo, -whole-module-optimization] вырезаются по умолчанию

Подключение зависимостей

Swift Package Manager

- В рамках проекта можно подключить только 1 версию каждой зависимости
- Для разных платформ можно подключать разные зависимости
- Remote пакеты можно подключать по ветке, тэгу и хэшу коммита

```
dependencies: [
        .package(
            url: "https://github.com/apple/swift-algorithms.git",
            .upToNextMajor(from: .init(stringLiteral: "1.0.0"))
        ),
        .package(path: "./local-swift-algorithms")
    ],
    targets: [
        .target(
            name: "EmptyPackage",
            dependencies: [
                .product(
                    name: "Algorithms",
                    package: "swift-algorithms",
                    condition: .when(platforms: [.iOS, .watchOS, .tvOS])
                ),
                .byName(name: "LocalAlgorithms")
            ]
        )
    ]
```

Версионирование. SemVer

Swift Package Manager

- 1.1.2(Major.Minor.Patch)
- Major - “ломающие” изменения публичного интерфейса
- Minor - обратно совместимое добавление функциональности
- Patch - обратно совместимые багфиксы

Способы подключения

Swift Package Manager

```
.package(
    url: "https://github.com/apple/swift-algorithms.git",
    .exact(.init(stringLiteral: "1.0.0"))
)
```

Tag

```
.package(
    url: "https://github.com/apple/swift-algorithms.git",
    .branch(“main")
)
```

Branch

```
.package(
    url: "https://github.com/apple/swift-algorithms.git",
    .revision("e2fa131d62604027889fcf87bb4d024e6576ac66")
)
```

Commit

Ресурсы и Локализация

Swift Package Manager

- Ресурсы появились в 5.3
- Имеют 2 вида копирования: [.copy(path: String), .process(path: String)]
- Локализация закидывается в ресурсы
- Можно указать дефолтный язык пакета
- Можно отдельно конфигурировать ресурсы под каждую локализацию

Plugins 5.5(5.6)
Swift Package Manager

- Добавляет исполнение скриптов в качестве билд фазы SPM
- Экспериментальная фича swift-tools-version:5.5
- Включается флагом SWIFTPM_ENABLE_PLUGINS в переменных окружения
- Функциональность полностью доступна в 5.6

Plugins 5.5(5.6)
Swift Package Manager

```
let package = Package(
    name: "PluginExample",
    products: [
        .library(
            name: "TestableLib",
            targets: ["TestableLib"]
        ),
        .plugin(
            name: "BuildPlugin",
            targets: ["BuildPlugin"]
        )
    ],
    targets: [
        .plugin(
            name: "BuildPlugin",
            capability: .buildTool()
        ),
        .target(name: "TestableLib", plugins: ["BuildPlugin"]),
        .testTarget(
            name: "TestableLibTests",
            dependencies: ["TestableLib"]
        )
    ]
)
```

Мультиязычные пакеты
Swift Package Manager

- Можно делать пакеты Swift, C, C++, ObjC
- Можно смешивать языки в рамках пакетов
- 1 таргет = 1 язык
- modulemap должен храниться в папке include

Линковка

Swift Package Manager

```
.library(
    name: "TensorFlowLite",
    targets: [
        "TensorFlowLiteSwift",
        "TensorFlowLiteC"
    ]
)
```

Automatic

```
.library(
    name: "TensorFlowLite",
    type: .static,
    targets: [
        "TensorFlowLiteSwift",
        "TensorFlowLiteC"
    ]
)
```

Static

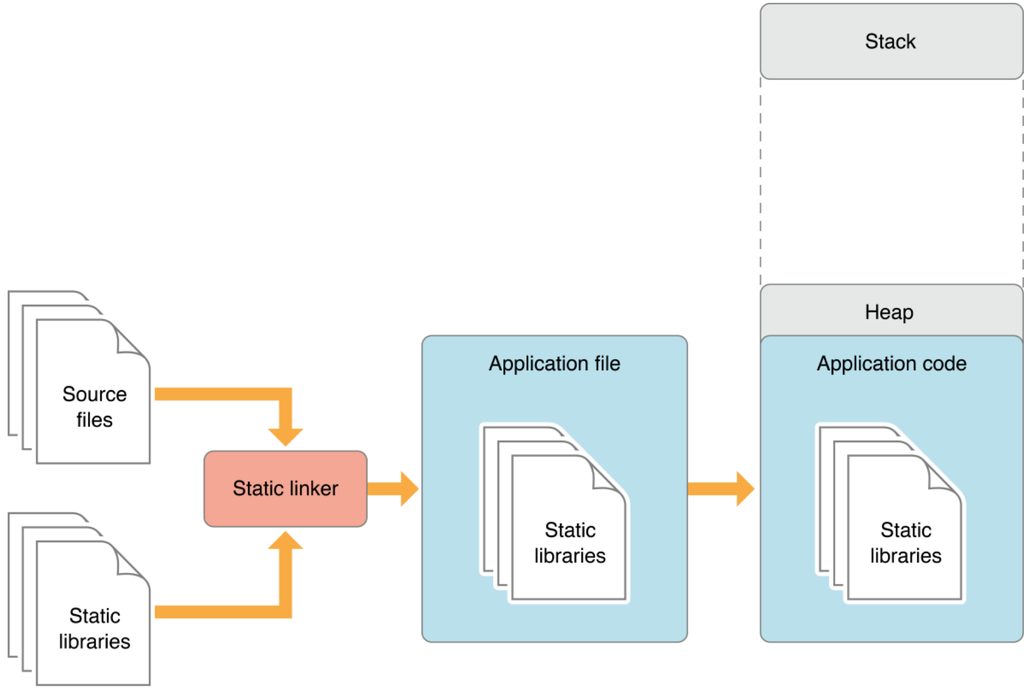
```
.library(
    name: "TensorFlowLite",
    type: .dynamic,
    targets: [
        "TensorFlowLiteSwift",
        "TensorFlowLiteC"
    ]
)
```

Dynamic

Static Library

Swift Package Manager

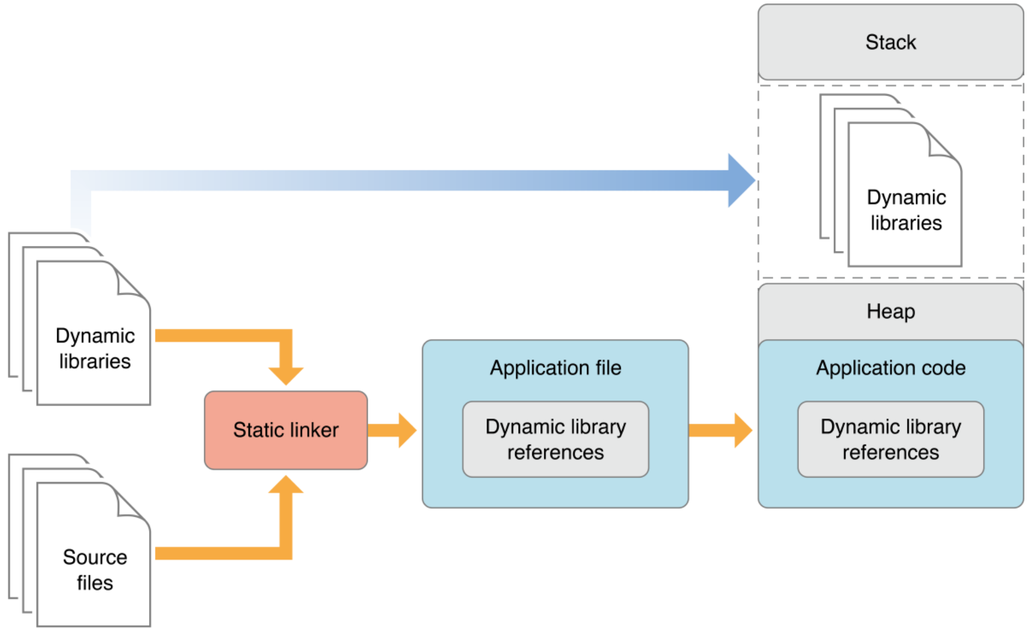
- Является частью конечного бинаря
- Чем больше статических либ, тем больше бинарь
- Быстрей запускается приложение



Dynamic Library

Swift Package Manager

- Не включается в конечный бинарь
- Копируется финальных фазах компиляции
- Загружаются на старте приложения, следовательно приложение загружается дольше



Манифест SDK

Swift Package Manager & Модульность

```
// swift-tools-version:5.5

import PackageDescription

let package = Package(
    name: "VTBFoundation",
    products: [
        .library(
            name: "VTBFoundation",
            type: .dynamic,
            targets: ["VTBFoundation"]
        ),
    ],
    targets: [
        .target(
            name: "VTBFoundation"
        ),
        .testTarget(
            name: "VTBFoundationTests",
            dependencies: ["VTBFoundation"]
        )
    ]
)
```

```
// swift-tools-version:5.5

import PackageDescription

let package = Package(
    name: "Network",
    products: [
        .library(
            name: "Network",
            type: .dynamic,
            targets: ["Network"]
        ),
    ],
    targets: [
        .target(name: "Network"),
        .testTarget(
            name: "NetworkTests",
            dependencies: ["Network"]
        ),
    ]
)
```

Манифест SDK

Swift Package Manager & Модульность

```
let package = Package(
    name: "Core",
    platforms: [.iOS(.v9)],
    products: [
        .library(
            name: "Core",
            type: .dynamic,
            targets: ["Core"]
        ),
    ],
    dependencies: [
        .package(
            name: "VTBFoundation",
            url: "git@github.com:RenatGafarov/VTBFoundation.git",
            .upToNextMajor(from: .init(stringLiteral: "0.1.0"))
        ),
        .package(
            name: "Network",
            url: "git@github.com:RenatGafarov/Network.git",
            .upToNextMinor(from: .init(stringLiteral: "1.0.0"))
        )
    ],
    targets: [
            .target(
                name: "Core",
                dependencies: [
                    .byName(name: "VTBFoundation"),
                    .byName(name: "Network")
                ]
            ),
            .testTarget(
                name: "CoreTests",
                dependencies: ["Core"]
            ),
        ]
)
```

Манифест FeatureModule

Swift Package Manager & Модульность

```
// swift-tools-version:5.5

import PackageDescription

let package = Package(
    name: "Transfers",
    products: [
        .library(
            name: "Transfers",
            targets: ["Transfers"]),
    ],
    dependencies: [
        .package(
            name: "Core",
            url: "git@github.com:RenatGafarov/core.git",
            .upToNextMajor(from: .init(stringLiteral: "2.0.0"))
        )
    ],
    targets: [
        .target(
            name: "Transfers",
            dependencies: [
                .byName(name: "Core")
            ]
        ),
        .testTarget(
            name: "TransfersTests",
            dependencies: ["Transfers"]
        ),
    ]
)
```

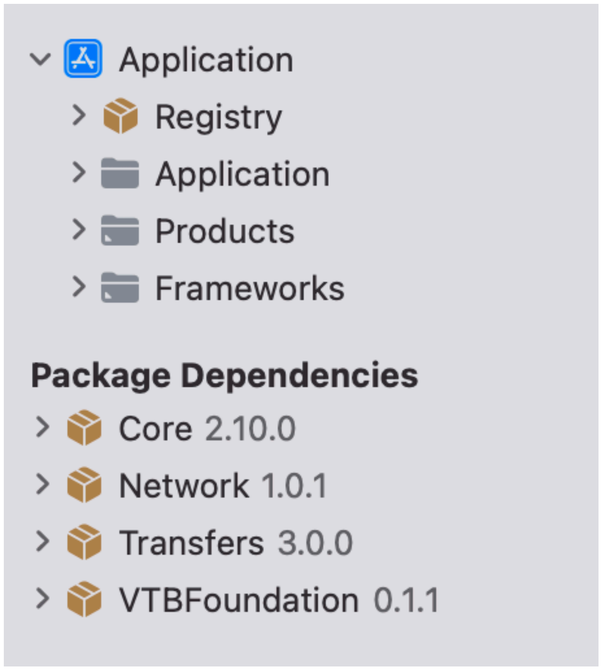
Манифест Registry

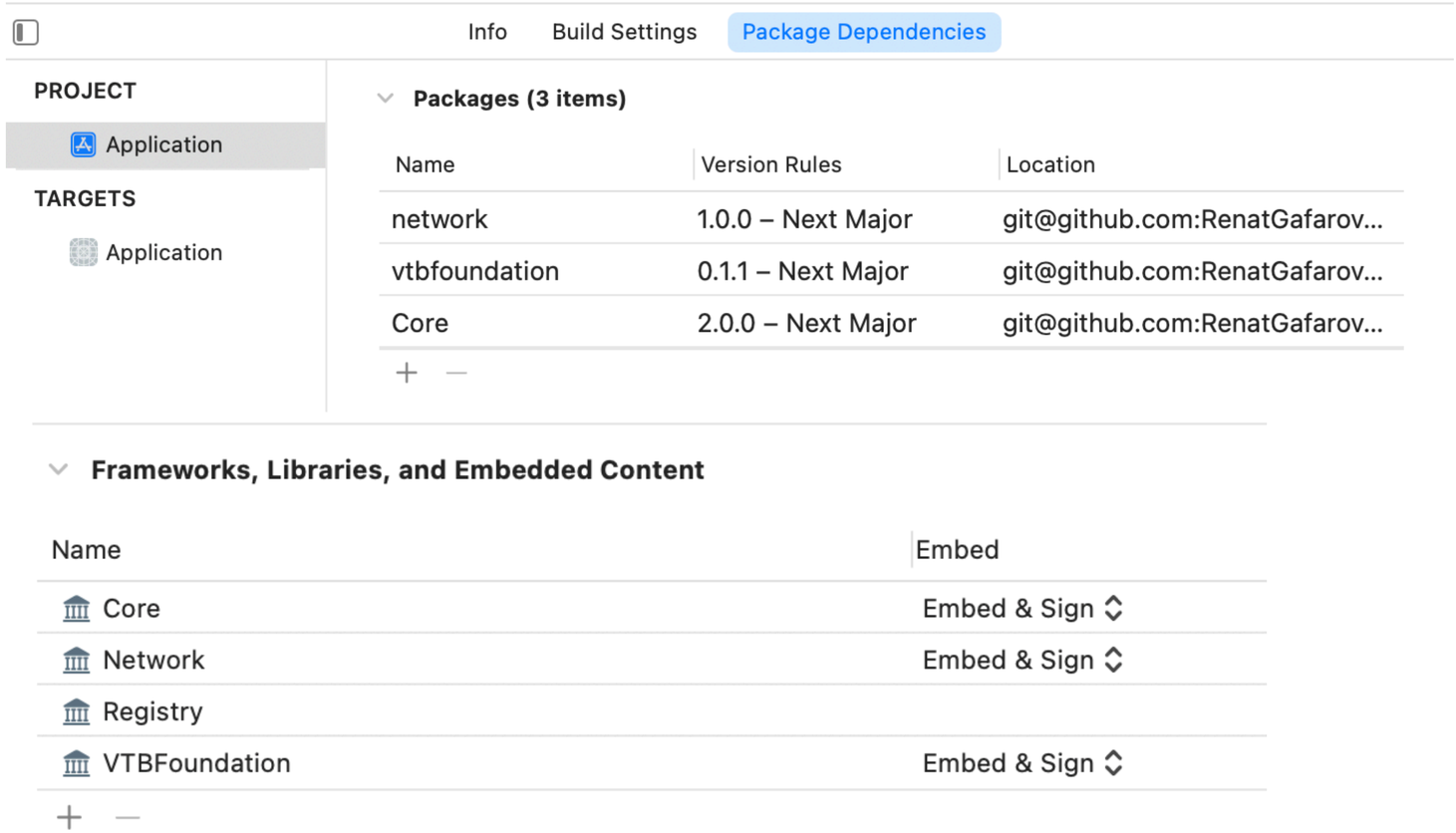
Swift Package Manager & Модульность

```
let package = Package(
    name: "Registry",
    products: [
        .library(
            name: "Registry",
            targets: ["Registry"]
        ),
    ],
    dependencies: [
        .package(
            name: "Transfers",
            url: "git@github.com:RenatGafarov/transfers.git",
            .exact(.init(stringLiteral: "3.0.0"))
        ),
        .package(
            name: "Core",
            url: "git@github.com:RenatGafarov/core.git",
            .exact(.init(stringLiteral: "2.10.0"))
        ),
        .package(
            name: "Network",
            url: "git@github.com:RenatGafarov/network.git",
            .exact(.init(stringLiteral: "1.0.1"))
        ),
        ///...
    ]
    targets: [
        .target(
            name: "Registry",
            dependencies: [
                .byName(name: "Transfers"),
                .byName(name: "Core"),
                .byName(name: "Network"),
                .byName(name: "VTBFoundation")
            ]
        ),
        .testTarget(
            name: "RegistryTests",
            dependencies: ["Registry"]
        ),
    ]
)
```

Манифест Registry

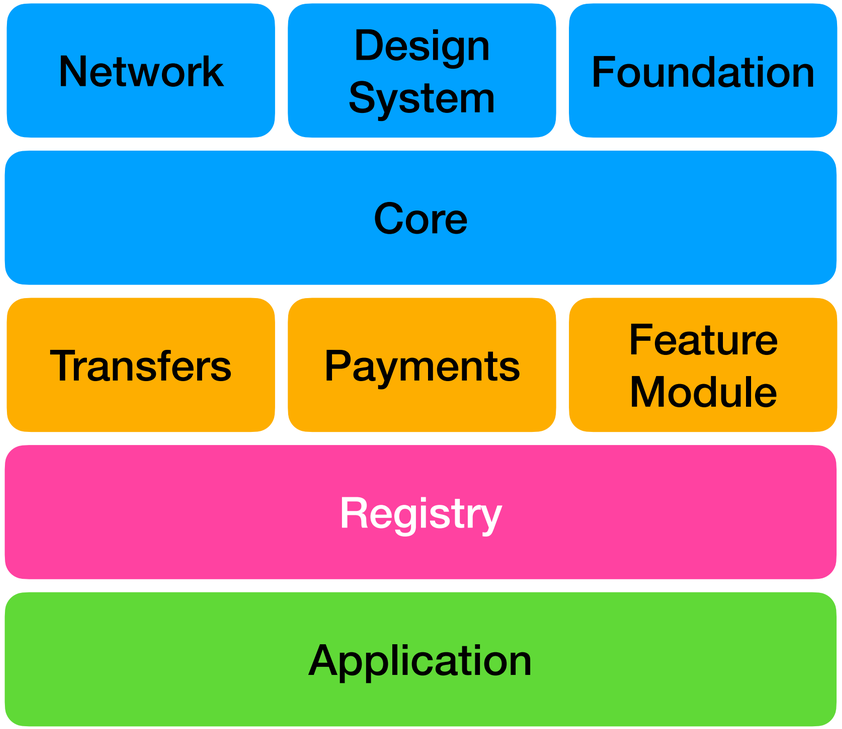
Swift Package Manager & Модульность





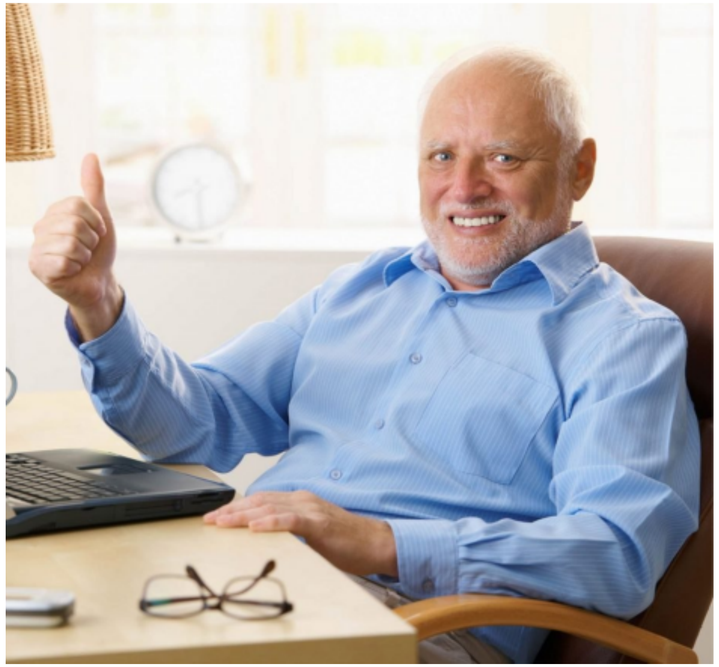
Подытожим

Swift Package Manager & Модульность



Проблемы
Swift Package Manager

- Авторизация для выкачивания зависимостей
- Fat frameworks
- Копирование фреймворков в архив
- Проблема Xcode 13



Виды авторизации
Swift Package Manager

- Для выкачивания из remote scm(github, bitbucket, gitlab, etc)
- Basic Auth авторизация

Авторизация в удаленном репозитории

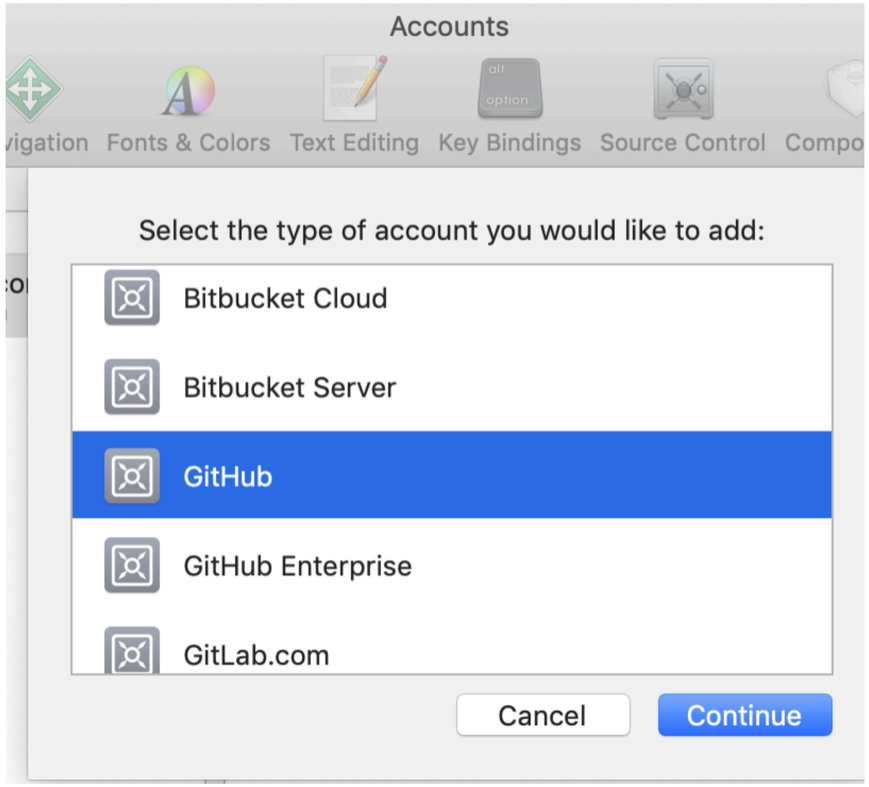
Swift Package Manager

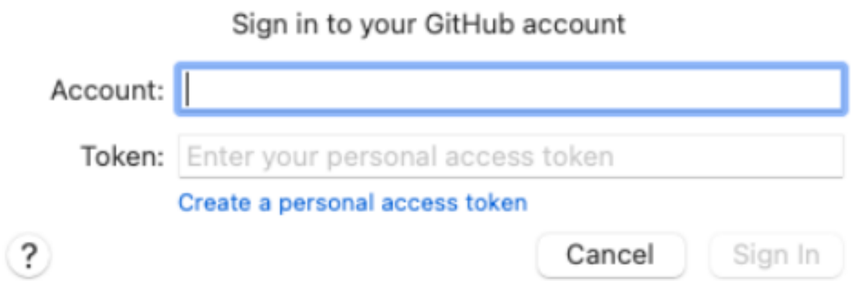
- Простой - SSH
- Еще проще - Добавить API токен в XCode
- И еще один

API Token в Xcode

Swift Package Manager

- Создаем токен
- Добавляем в Xcode





И еще один
Swift Package Manager

- Авторизация по API токену вшитому в URL

https://github-token@github.com/username/repo.git

Basic Auth

Swift Package Manager

- Используется для выкачивания binary targets
- Данные берутся из .netrc файла
- # - ваш враг

```
machine bitbucket.org
login RenatGafarov
password Qwerty-123456
```

Binary Targets. Fat frameworks

Swift Package Manager

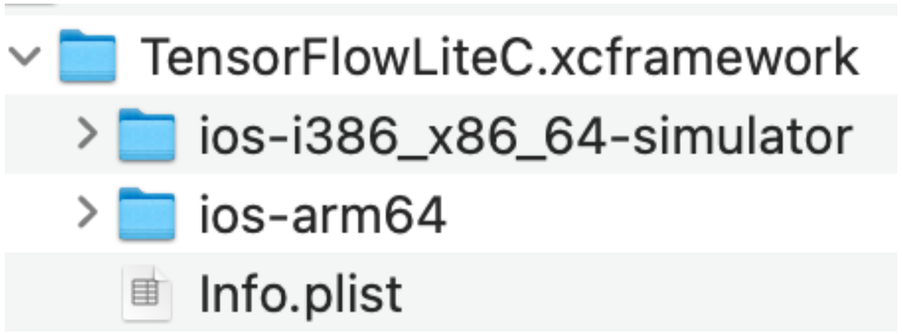
- Содержит все архитектуры
- Поставляется в виде .framework
- Абсолютно не подходит для использования с SPM

Binary Targets. Fat frameworks
Swift Package Manager

```
mkdir -p iphoneos & mkdir -p iphonesimulator

cp -R TensorFlowLiteC.framework/iphoneos/TensorFlowLiteC.framework
cp -R TensorFlowLiteC.framework/iphonesimulator/TensorFlowLiteC.framework

lipo -remove i386 -remove x86_64 -remove armv7
    ./iphoneos/TensorFlowLiteC.framework/TensorFlowLiteC -o ./iphoneos/TensorFlowLiteC.framework/TensorFlowLiteC

lipo -remove i386 -remove arm64 -remove armv7
    ./iphonesimulator/TensorFlowLiteC.framework/TensorFlowLiteC -o ./iphonesimulator/TensorFlowLiteC.framework/TensorFlowLiteC

xcodebuild -create-xcframework \
    -framework iphoneos/TensorFlowLiteC.framework/ \
    -framework iphonesimulator/TensorFlowLiteC.framework/ \
    -output "TensorFlowLiteC.xcframework"
```

Binary Targets. Fat frameworks

Swift Package Manager



Копирование фреймворков

Swift Package Manager

- Статические Фреймворки
- Динамические Фреймворки
- Appex файлы

Особенности Xcode 13.X

Swift Package Manager

- За каждый SPM пакет добавляется путь в SearchPath
- Много пакетов, много дублирующихся SearchPath
- Отваливаемся по длине строки с аргументами CompileSwift

Выводы

Swift Package Manager

- Пока ещё сырой
- Почти для любой проблемы можно найти велосипед или написать его самому
- Достаточно простой
- Перспективный инструмент
- Not ready for enterprise

Контакты

@reclosure

gafarovrenat