Cocoapods

Pawel Dudek

How can we manage dependencies in Cocoa?

Copy & Paste

Submodules

Copy & Paste

1. Copy & Paste files
2. Add other linker flags
3. Add ARC flags
4. Add frameworks
5. Add any other missing build settings
6. Add resources
7. Finally, use the component

Copy & Paste Issues

- Issues with duplicate symbols
- **Really** hard to manage versions
- Missing other linker flags and build settings
- Missing resources

```
duplicate symbol _OBJC_IVAR_$_AFQ
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _OBJC_IVAR_$_AFQ
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _AFQueryStringFr
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _AFQueryStringPa
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _AFQueryStringPa
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _OBJC_IVAR_$_AFH
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _OBJC_IVAR_$_AFH
    /Users/eldudi/Library/Develop
    /Users/eldudi/Library/Develop
duplicate symbol _OBJC_IVAR_$_AFH
```

Submodules

- Issues with duplicate symbols
- Somewhat easier to manage versions (if they're properly tagged)
- Other linker flags
- And other build settings
- Resources

Submodules

1. Add submodule (and check it out)
2. Add source to project
3. Add ARC flags
4. Add frameworks
5. Add other linker flags
6. Add any other missing build settings
7. Fix duplicate symbols
8. Add resources
9. Finally, use the component

This is all wrong.

We are to create things.

This is all just wasting our time.

Enter Cocoapods

Cocoapods goals

- Make working with dependencies simple.
- Improve library discoverability and engagement by providing an ecosystem that facilitates this.

Cocoapods advantages

- Automatically handle source code and static libraries
- Automatically handle ARC
- Automatically handle frameworks
- Automatically handle builds settings
- Automatically handle resources

Cocoapods advantages

The responsibility for configuration requirements lie with the creator of component, not you.

How can I install them?

```
gem install cocoapods
```

Basics

How do Cocoapods work?

- Pod - single definition of a component
- Podfile - list of dependencies
- Dependencies use semantic versioning
- Resolving dependencies lists all your dependencies and their dependencies
- Dependencies definitions are a Github repository

What happens when I install pods?

- Resolve dependencies from Podfile
- Take an .xcodeproj as a start
- Generate .xcconfing files and attaches them to your project
- Generate another .xcoproject with static library from defined dependencies

What happens when I install pods?

- Generate an .xcworkspace with your project and generated .xcodeproject
- Add a dependency on the generated project results to your targets
- Lock used versions in Podfile.lock

Basic commands

Installing pods

```
pod install
```

Updating pods

```
pod update
```

pod install

- When there is no Podfile.lock - will use latest version or version defined in Podfile
- When there is a Podfile.lock - will use version from Podfile.lock or version defined in Podfile

pod update

- Ignored Podfile.lock
- Will work as ‘pod install’ without a Podfile.lock

Podfile

- Defines platform
- Defines project (optional)
- Defines dependencies
  - Defines specific version
- Multiple targets

Podfile example

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

Project

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

← Dependencies

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

Exclusive target name

Podfile

```
platform :ios, '5.0'
xcodeproj 'TwitterUserTimeline'

pod 'STTwitter'
pod 'Mantle', '1.2'

target :cedar do
  link_with 'TwitterUserTimelineSpecs'
  pod 'Cedar'
end
```

← Exclusive pod

Cleaning up

By wiping whole Cocoapods caches

```
rm -rf Pods/
rm -rf ~/Library/Caches/CocoaPods/Git/
rm -rf ~/Library/Caches/CocoaPods/GitHub/
rm -rf ~/.cocoapods/
```

Moving patch version

Will automatically update to new available patch version

```
pod 'Mantle', '~> 1.2.0'
```

Fun stuff

Encourages reusable code

Over 28 components at Taptera

Really encourage to use it!

Using Cocoapods for in-house components

What you'll need

- Cocoapods installed
- Git repo for specs definitions
- And you're all set!

Your own Cocoapods

Pod spec

Example

Things you'll have to do first

- Add your own specs repo to local cocoapods repo list
- Push the podspec to your repository

Adding custom specs repo

```
pod repo add <repo_name> <repo_address>
```

Pushing to Cocoapods specs repo

```
pod push <repo_name>
```

Demo

Resources & Contact

Code Examples
github.com/paweldudek

Contact
@eldudi
pawel@dudek.mobi