
cocoapods-rome



Rome makes it easy to build a list of frameworks for consumption outside of Xcode, e.g. for a Swift script.

Installation

```
1 $ gem install cocoapods-rome
```

Important

In the examples below the target 'caesar' could either be an existing target of a project managed by cocapods for which you'd like to run a swift script **or** it could be fictitious, for example if you wish to run this on a standalone Podfile and get the frameworks you need for adding to your xcode project manually.

Usage

Write a simple Podfile, like this:

MacOS

```
1 platform :osx, '10.10'
2
3 plugin 'cocoapods-rome'
4
5 target 'caesar' do
6   pod 'Alamofire'
7 end
```

ios

```
1 platform :ios, '8.0'
2
3 plugin 'cocoapods-rome', { :pre_compile => Proc.new { |installer|
4   installer.pods_project.targets.each do |target|
5     target.build_configurations.each do |config|
6       config.build_settings['SWIFT_VERSION'] = '4.0'
7     end
8   end
9
10  installer.pods_project.save
11 },
12
13   dsym: false,
14   configuration: 'Release'
15 }
16
17 target 'caesar' do
18   pod 'Alamofire'
19 end
```

then run this:

```
1 pod install
```

and you will end up with dynamic frameworks:

```
1 $ tree Rome/
2 Rome/└──
3   Alamofire.framework
```

Advanced Usage

For your production builds, when you want dSYMs created and stored:

```
1 platform :osx, '10.10'
2
```

```

3 plugin 'cocoapods-rome', {
4   dsym: true,
5   configuration: 'Release'
6 }
7
8 target 'caesar' do
9   pod 'Alamofire'
10 end

```

Resulting in:

```

1 $ tree dSYM/
2 dSYM/|
3   iphoneos|
4     └─ Alamofire.framework.dSYM|
5       └─ Contents|
6         └─ Info.plist|
7           └─ Resources|
8             └─ DWARF|
9               └─ Alamofire ──
10  iphonesimulator ──
11    Alamofire.framework.dSYM ──
12      Contents ──
13        Info.plist ──
14          Resources ──
15            DWARF ──
16              Alamofire

```

Hooks

The plugin allows you to provides hooks that will be called during the installation process.

pre_compile

This hook allows you to make any last changes to the generated Xcode project before the compilation of frameworks begins.

It receives the `Pod::Installer` as its only argument.

post_compile

This hook allows you to run code after the compilation of the frameworks finished and they have been moved to the `Rome` folder.

It receives the `Pod::Installer` as its only argument.

Example Customising the Swift version of all pods

```
1 platform :osx, '10.10'
2
3 plugin 'cocoapods-rome',
4   :pre_compile => Proc.new { |installer|
5     installer.pods_project.targets.each do |target|
6       target.build_configurations.each do |config|
7         config.build_settings['SWIFT_VERSION'] = '4.0'
8       end
9     end
10
11     installer.pods_project.save
12   },
13   :post_compile => Proc.new { |installer|
14     puts "Rome finished building all the frameworks"
15   }
16
17 target 'caesar' do
18   pod 'Alamofire'
19 end
```