

Modular App Architecture

Slicing up large applications into reasonably-sized chunks

What this talk is about? ✓

- App Architecture
- A practical example of modular app architecture
 - Creating your first package
 - Leveraging .xcworkspaces
- How to structure and work with growing codebases, complexity and teams
- Sharing partly anecdotal experiences



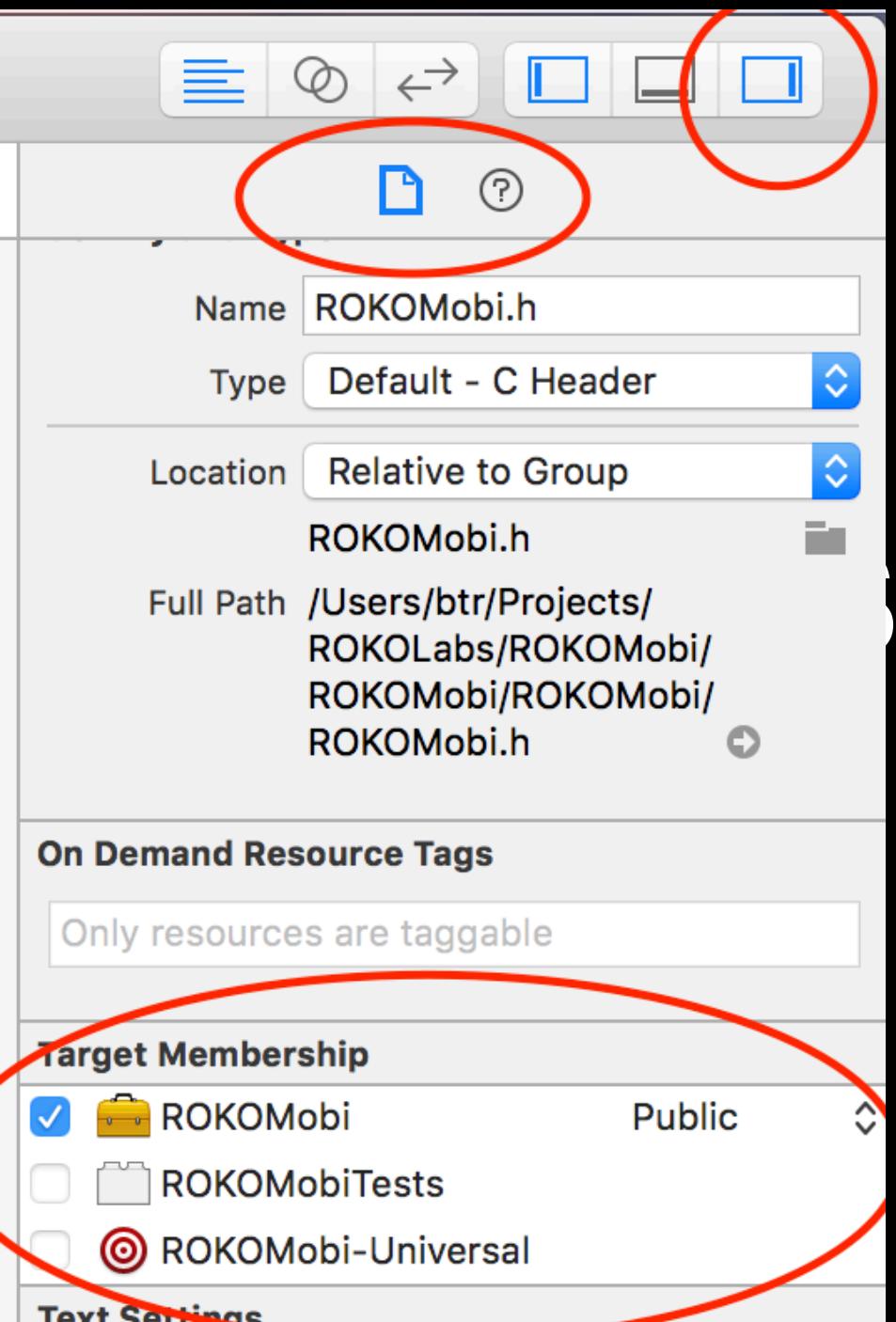
What this talk is not about? X

- Screen Architecture
 - MVVM, VIPER, MVC, MOAA
- How to migrate your app
 - But feel free to ask questions!
- Integrating non-Swift code via SPM
- Bananas 🍌



 Build succeeded 15.08.21, 18:58 6614 seconds
No issues

"We'll release our application"



Some developer, somewhere

How to merge conflicts (file project.pbxproj) in Xcode use svn?

Asked 11 years, 7 months ago Active 8 months ago Viewed 88k times



Reasons to modularise your app

- Growing project with a lot of reused code
- Improving developer experience
 - Easier onboarding
 - Shorter build times
 - Clear ownership over modules



What you need

- A form of dependency manager that allows local packages / pods / carts
 - Swift Package Manager (Our weapon of choice)
 - CocoaPods (Local pod specs + test specs)
 - Whatever floats your boat 
- Code to modularise

What is Swift Package Manager?

Managing dependencies with Style

- Apple's dependency manager
- Tightly integrated into Xcode
- All Swift! No more Ruby / YAML / JSON config files
- Supports multiple platforms
- Open source (<https://github.com/apple/swift-package-manager>)
- Reference (<https://swift.org/package-manager/>)



What is a Module?

Name-spaced Swift Code



- Swift organises Code into modules
 - `import Foundation` <- imports the Foundation module
- Modules import other modules as dependencies
- Allows code reusability in multiple projects
- Eliminate the need to reimplement existing things

What is a Package?

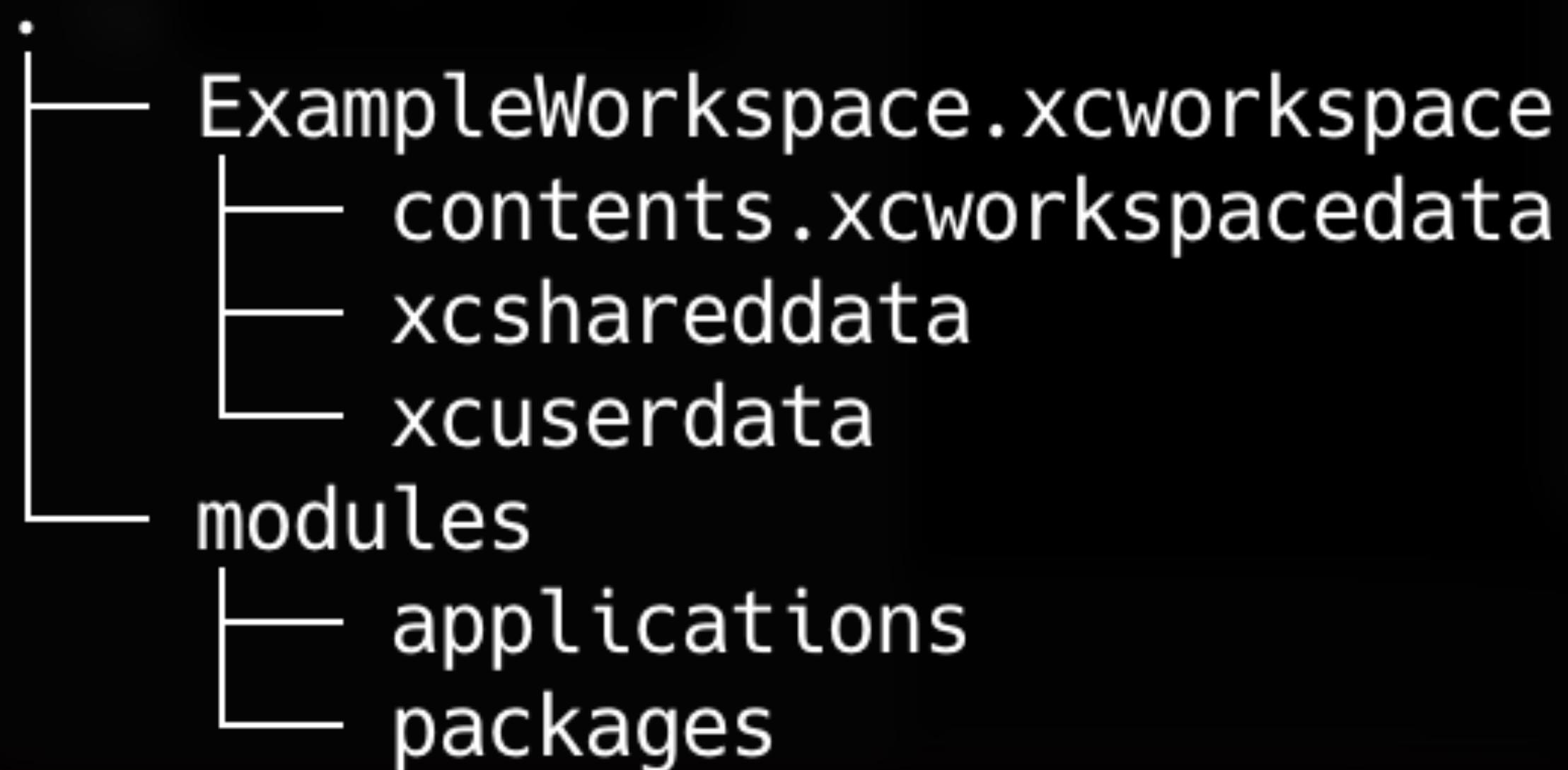
A module container



- A Swift Package is described by the `Package.swift` manifest file
- Packages define targets
 - Each Target is a Module
 - Targets build into Products
 - Products can be libraries or executables
 - Targets can define (versioned) dependencies
- Packages can be imported in Xcode or into other Swift Packages

Modular App Architecture with SPM

By folder structure

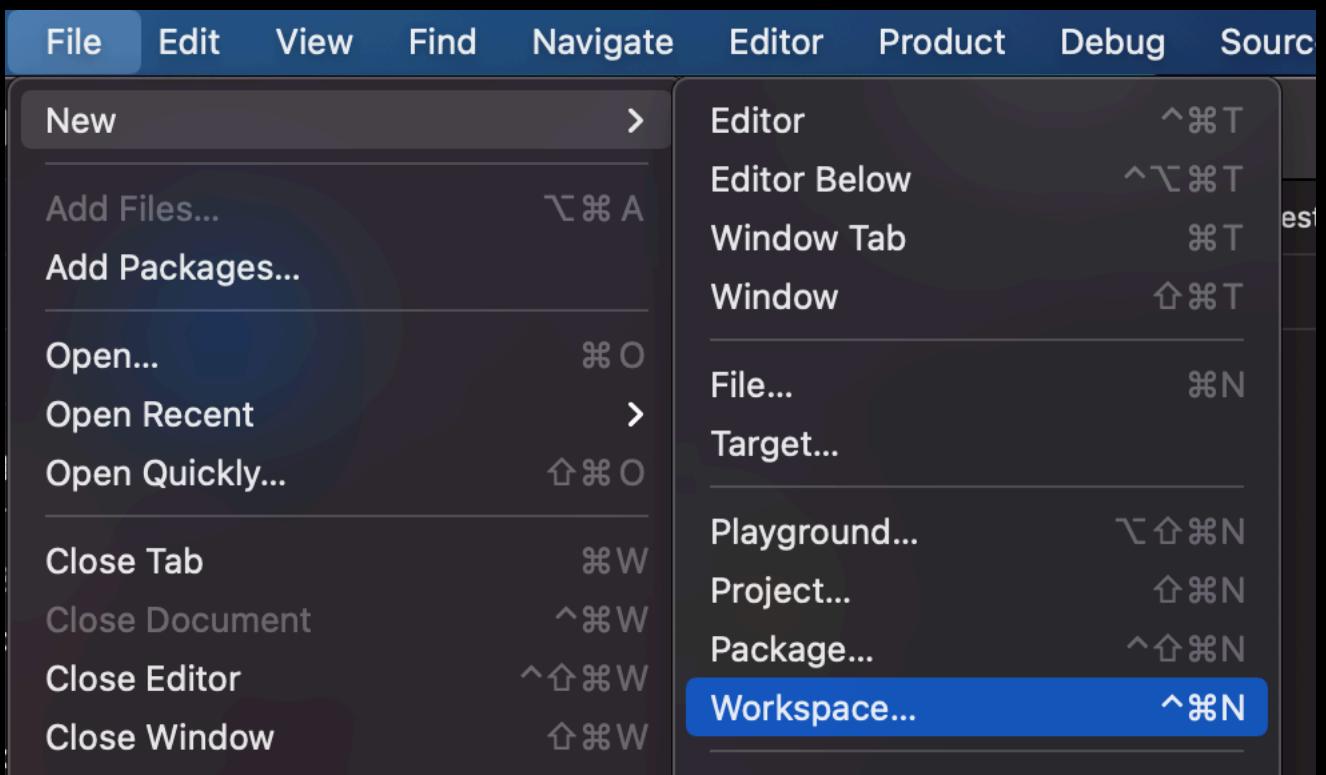


Creating a workspace

A place where everything comes together



1. Create a new folder. This will be the root folder of your application.
2. Open Xcode.
3. Create a new workspace



4. Will remain empty for now, we'll get back to it later.

Creating your first package

Step by step



1. Create a new folder in modules/packages. The folder name defines the package name.
2. Open your terminal and `cd` into the folder
3. Run swift package init

```
~/Code/Misc/talks/Examples/ModularAppArchitecture/modules/packages ➤ main ➤ swift package init
Creating library package: packages
Creating Package.swift
Creating README.md
Creating .gitignore
Creating Sources/
Creating Sources/packages/packages.swift
Creating Tests/
Creating Tests/packagesTests/
Creating Tests/packagesTests/packagesTests.swift
```

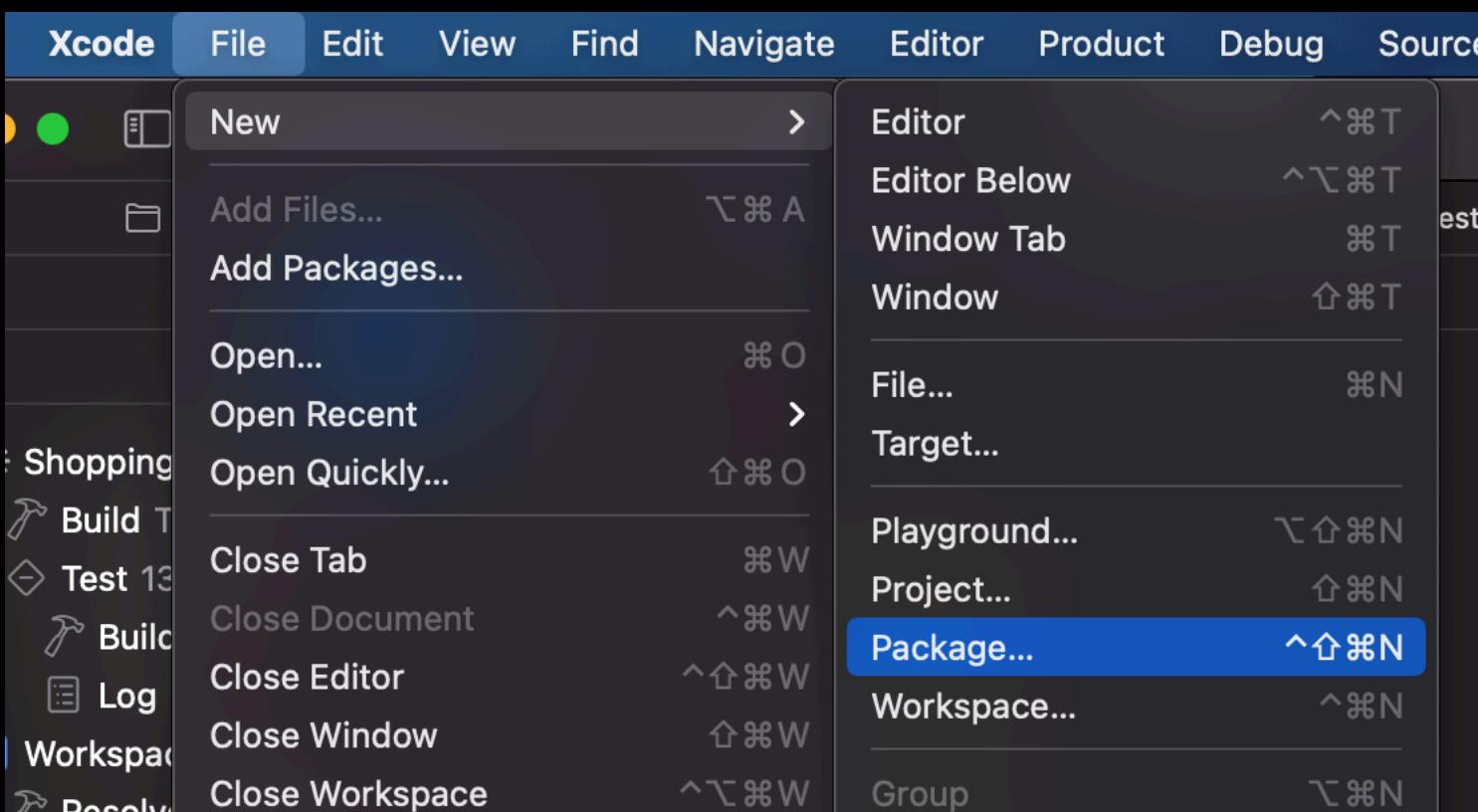
4. Double click the created Package.swift file

The Xcode Way

GUI-based package creation



1. Open Xcode
2. Create a new package via File -> New -> Package... and follow the steps



3. Xcode automatically opens the newly created package

Fun fact

swift package init --type executable

```
~/Code/Misc/talks/Examples/ModularAppArchitecture/modules/packages ➔ main ➔ swift package init -help
```

OVERVIEW: Initialize a new package

USAGE: swift package init <options>

OPTIONS:

- type <type>
- name <name>
- version
- help, -h, --help

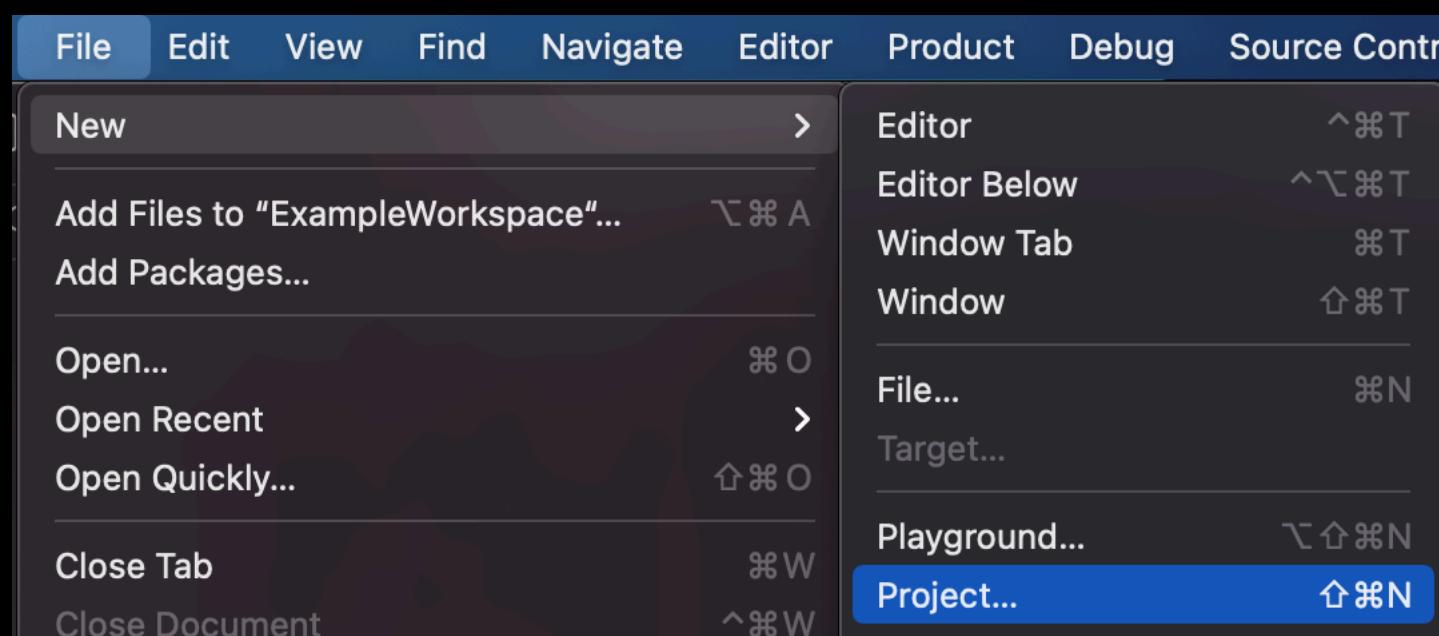
- Package type: empty | library | executable | system-module | manifest (default: library)
- Provide custom package name
- Show the version.
- Show help information.

Creating your first application

Probably not your first application



1. Back to Xcode.
2. Create your first application in modules/applications via File -> New -> Project.



3. Add the app to your workspace by drag and dropping the .xcproject file into the project navigator (left Xcode pane)

Putting together the pieces

Package + App + Workspace = ❤



1. Drag and drop the Package folder into the Project Navigator pane to add it to the workspace.

The screenshot shows the Xcode interface with a dark theme. On the left, the Project Navigator pane displays a file tree for a project named "MAAD". In the center, the Editor pane shows the main.swift file of an app named "MAADApp". The code defines a struct MAADApp that contains a @main attribute and a body property of type some Scene, which is a WindowGroup with a ContentView. On the right, the Utilities pane is open, showing the "packages" tab of the Finder. A folder named "Core" is selected, and its details are shown in the preview area. The status bar at the bottom indicates "Today at 19:35".

```
// MAADApp.swift
// MAAD
//
// Created by Daniel Peter on 15.08.21.

import SwiftUI

@main
struct MAADApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

Putting together the pieces

Package + App + Workspace = ❤



1. Import the module in your app

```
import Core
```

2. Use public functionalities of your module

```
Button(  
    action: { HelloWorldPrinter().sayHello() },  
    label: { Text("Hello, world!") })
```

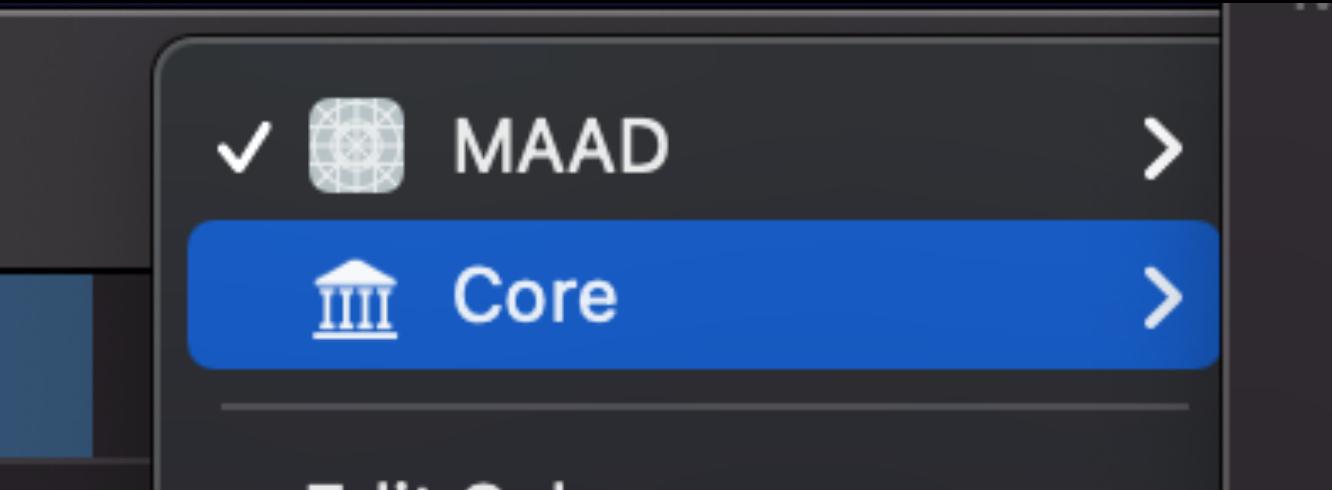
3. Celebrate your modularised app! 🎉

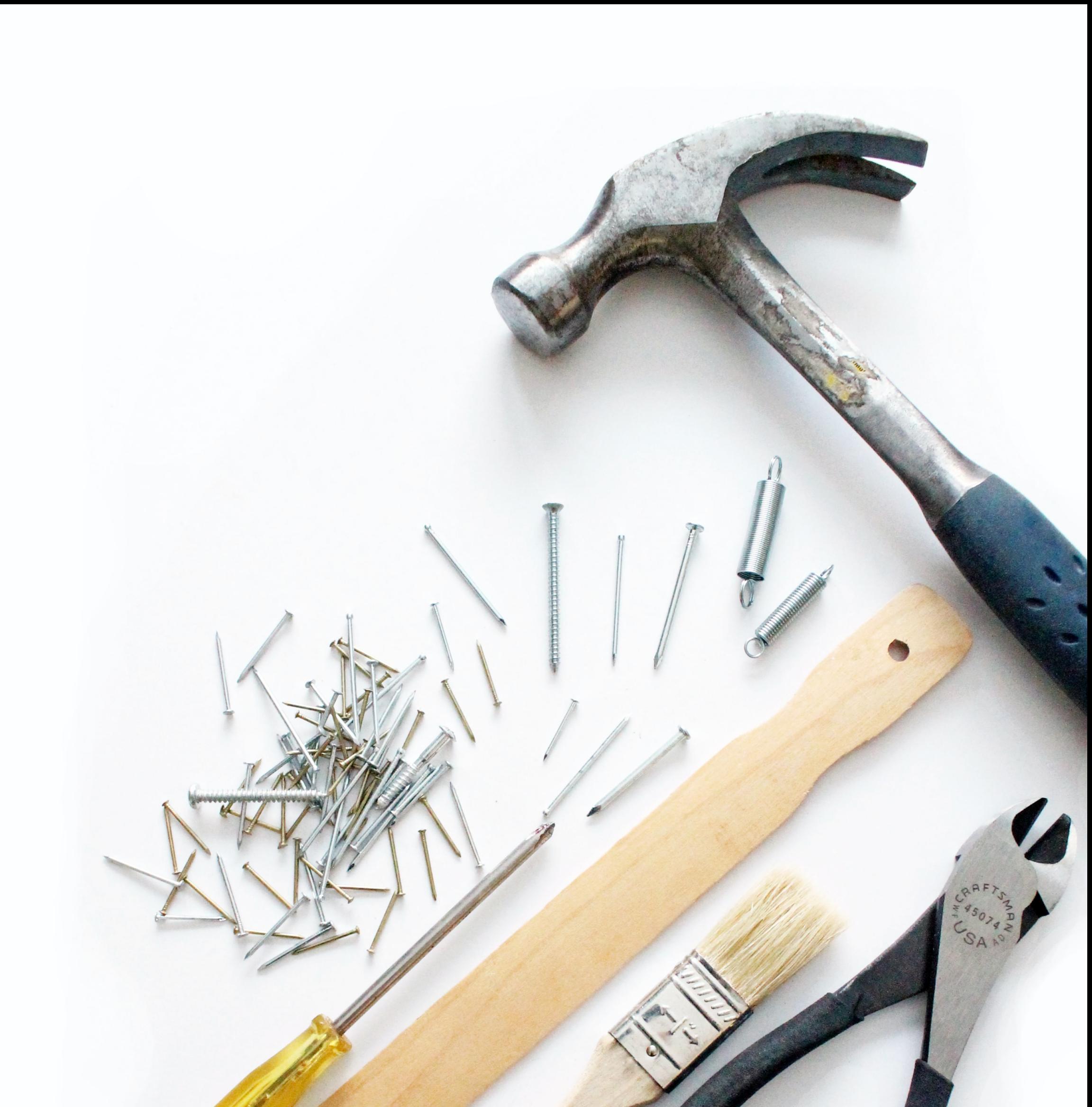
Important!

- Always open your workspace file, .xcproject files are only used for applications.
- You can use multiple workspaces, if your workspace gets to larger
- Most files are now managed in modules / packages
 - ➡ Fewer .xcproject merge conflicts



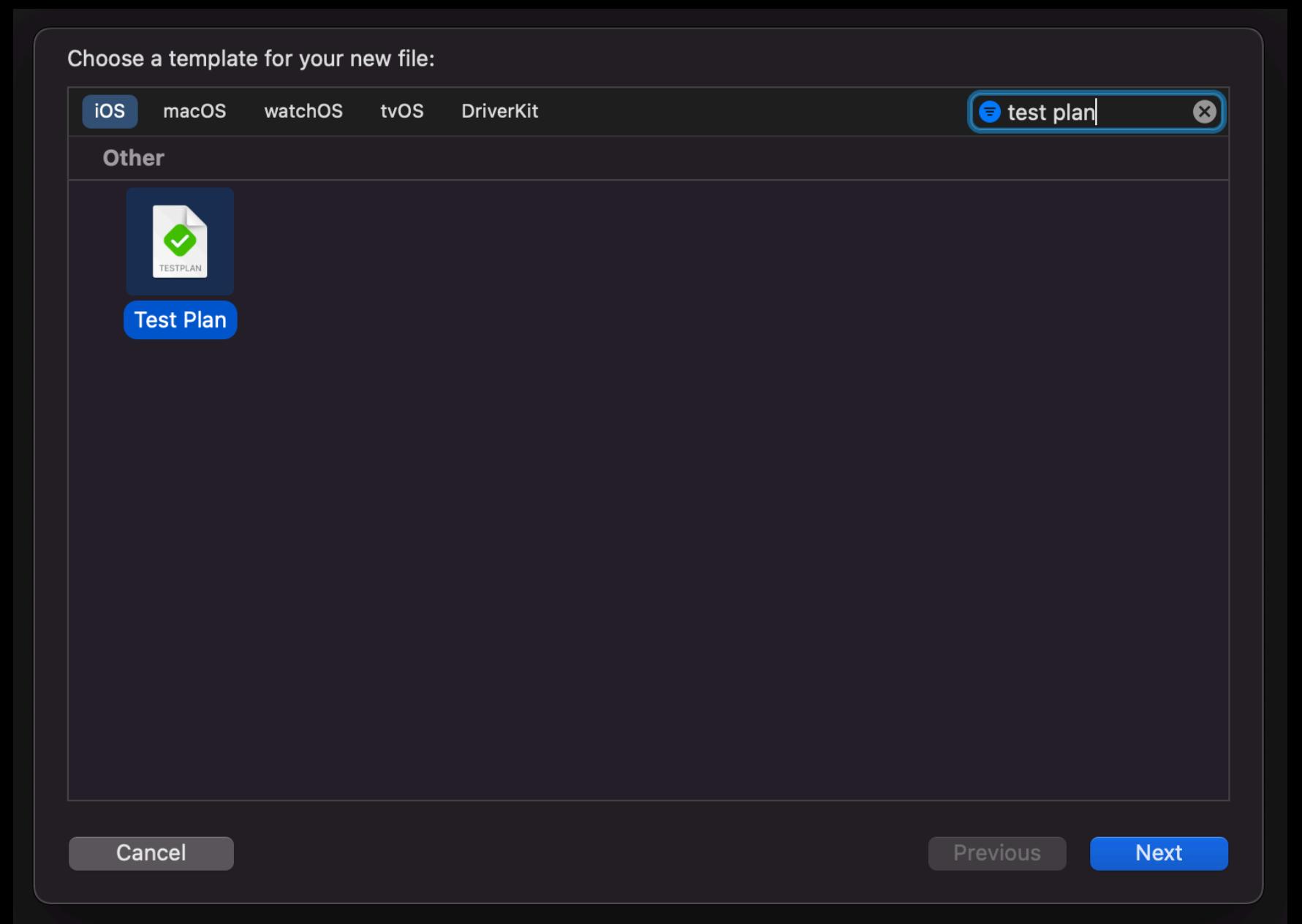
Benefits

- Each module can be built independently -> Fast build times!
- 
- Each module can have an example applications
 - Each module can have a test target
 - All example applications can be accessed / run via the main workspace



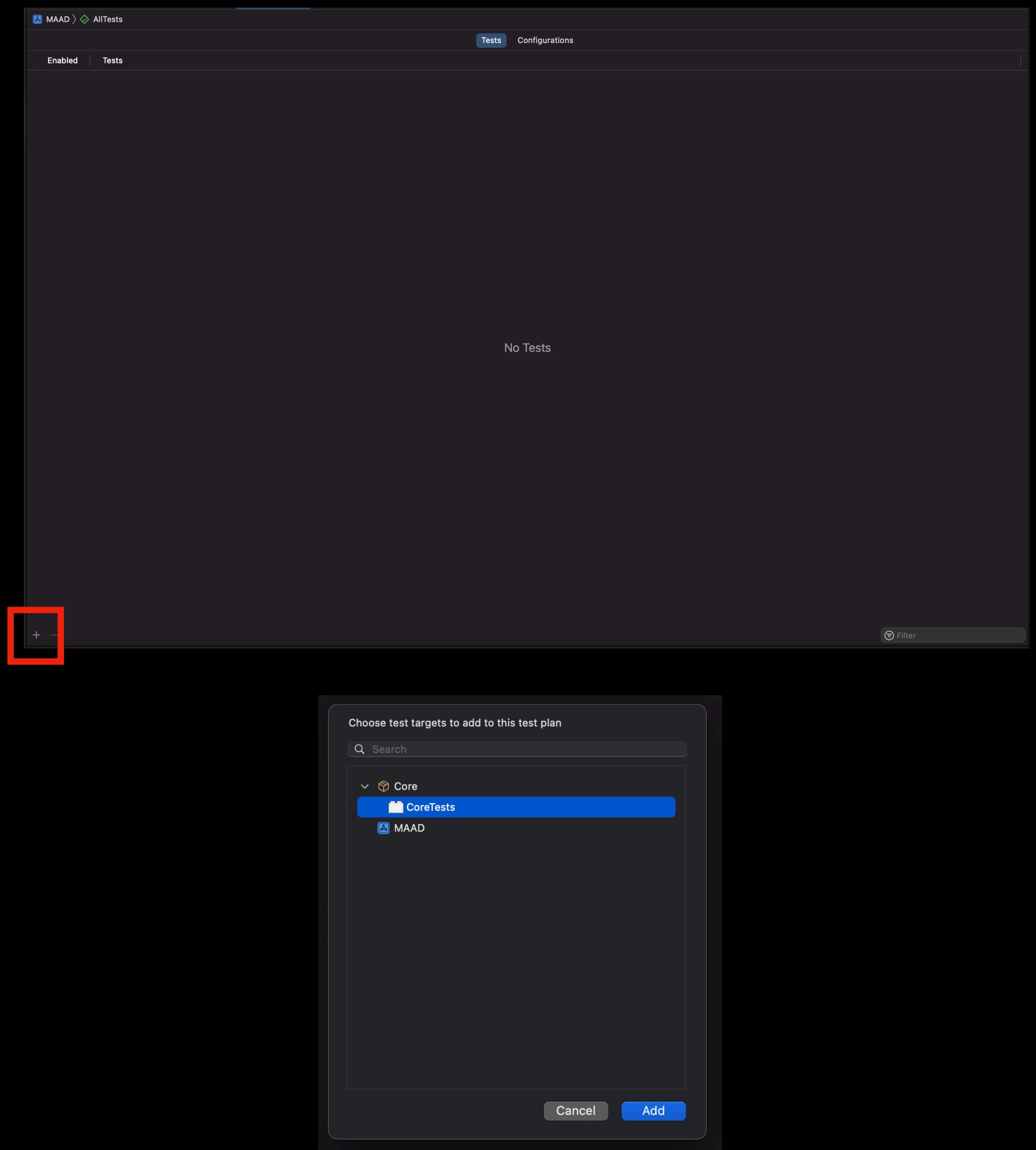
Using test plans

- A test plan is a container for multiple test targets
- Each module has its own test target
- You can add a test plan that runs all module tests
- Test plans can be run on a pre-built app (huge advantage on CI!)
- You can create a new testplan in the "New file" window



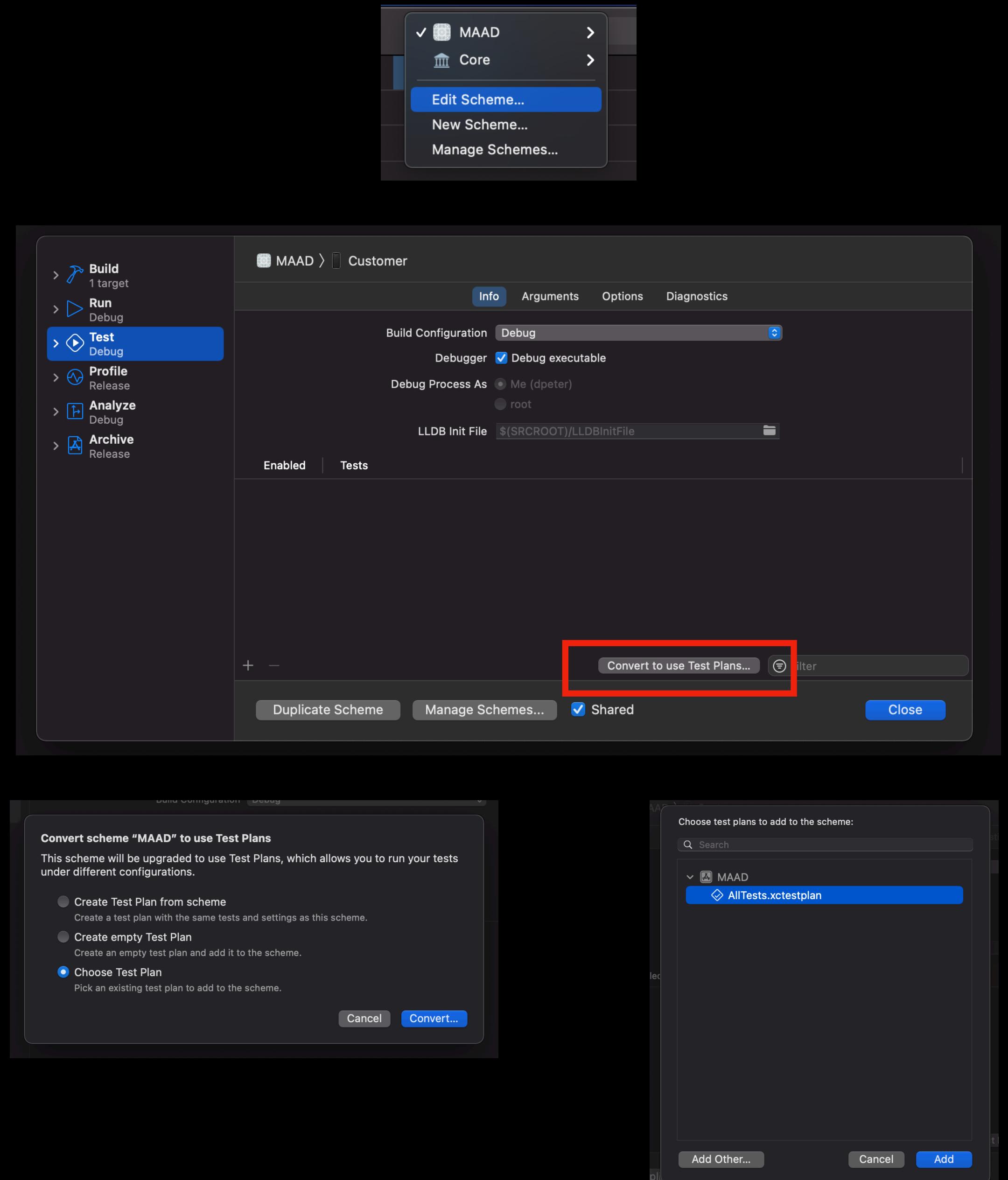
Adding test targets

- Double click your test plan
- Click the small plus on the bottom left
- Add a module test target



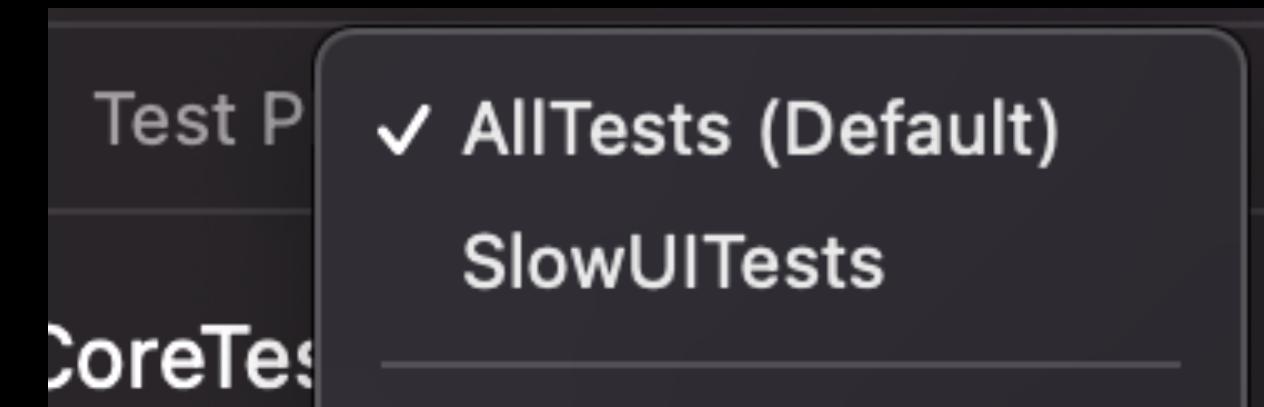
Running test plans

- Configure your app scheme to run the test plan
- Select the app scheme in the scheme selector and click on edit scheme
- Convert your test stage to use test plans
- Choose your created test plan
- Run test via CMD + U (or via the Test navigator pane)



Fun fact

You can define multiple test plans



But how do I slice up my code?

- Typical candidates for packages:
 - Reusable UI components
 - Design Systems
 - Useful helper functions
 - 1 package per feature
 - Clear overview over cross-dependencies
 - Define public entry points, the rest is a black box!



Learning over the years

- Try to keep things mono-repo
 - Easier to update dependencies
 - Better overview over pull requests and fewer PRs overall
- Modularise early to structure your code into small, understandable chunks
- Add a small Readme to each module, explaining what it does and who maintains it
 - Easier onboarding for new joiners
 - Clear definition of scope
- Use OWNERSHIP file in git to assign pull requests to owners
- Keep it maintainable and low cost.
 - Add helper scripts to generate modules fast
 - Automate the process where possible



Further reading

How the Zalando iOS App Abandoned CocoaPods and Reduced Build Time

<https://engineering.zalando.com/posts/2017/02/how-the-zalando-ios-app-abandoned-cocoapods-and-reduced-build-time.html>

Architecture and Automation: How Development Processes Work at N26

<https://medium.com/insiden26/architecture-and-automation-how-development-processes-work-at-n26-e204500fbc10>

A Tour of isowords (hyper-modularised code base)

<https://www.pointfree.co/episodes/ep142-a-tour-of-isowords-part-1>

<https://github.com/pointfreeco/isowords>





THIS IS IT!

Questions?

Connect with me:

Twitter @oh_its_daniel



LinkedIn ->

