Mastering SwiftPM 

Swift Package manager

Ratnesh Jain

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Access Modifiers

Access Modifiers

Allow what should be visible/readable/sub-classable/overridable

- Public
- Private
- FilePrivate
- Open
- Internal

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Package Manifest

Package.swift

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    products: [
        // Products define the executables and libraries a package produces, making them visible to other packages.
        .library(
            name: "MasteringSwiftPM",
            targets: ["MasteringSwiftPM"]),
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module or a test suite.
        // Targets can depend on other targets in this package and products from dependencies.
        .target(
            name: "MasteringSwiftPM"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["MasteringSwiftPM"]),
    ]
)
```

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Package Manager Vocabulary

Terminology used by Swift Package Manager

```swift
import SwiftUI

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading")
        }
    }
}
```

Package Manager Vocabulary

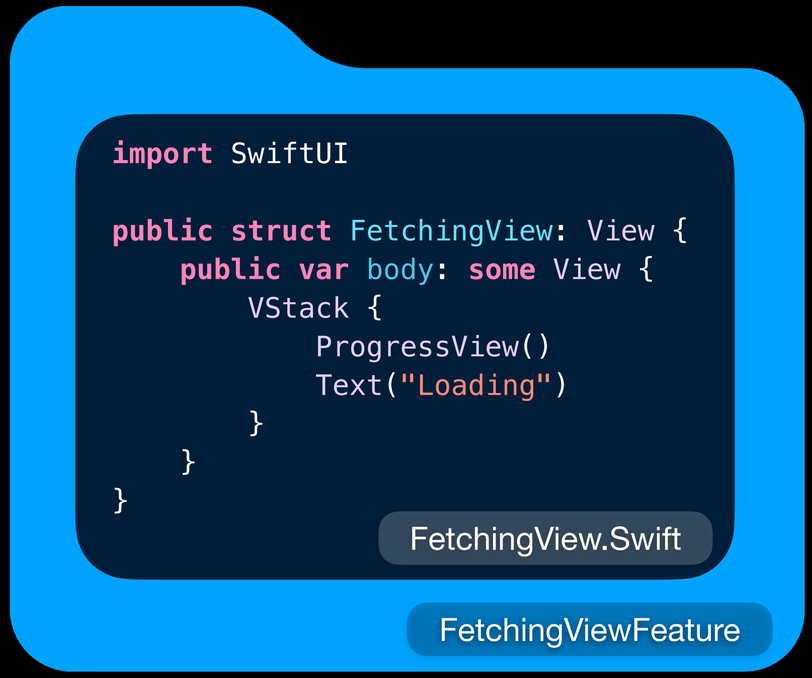
Directory Structure

```swift
import SwiftUI

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading")
        }
    }
}
```

FetchingView.Swift

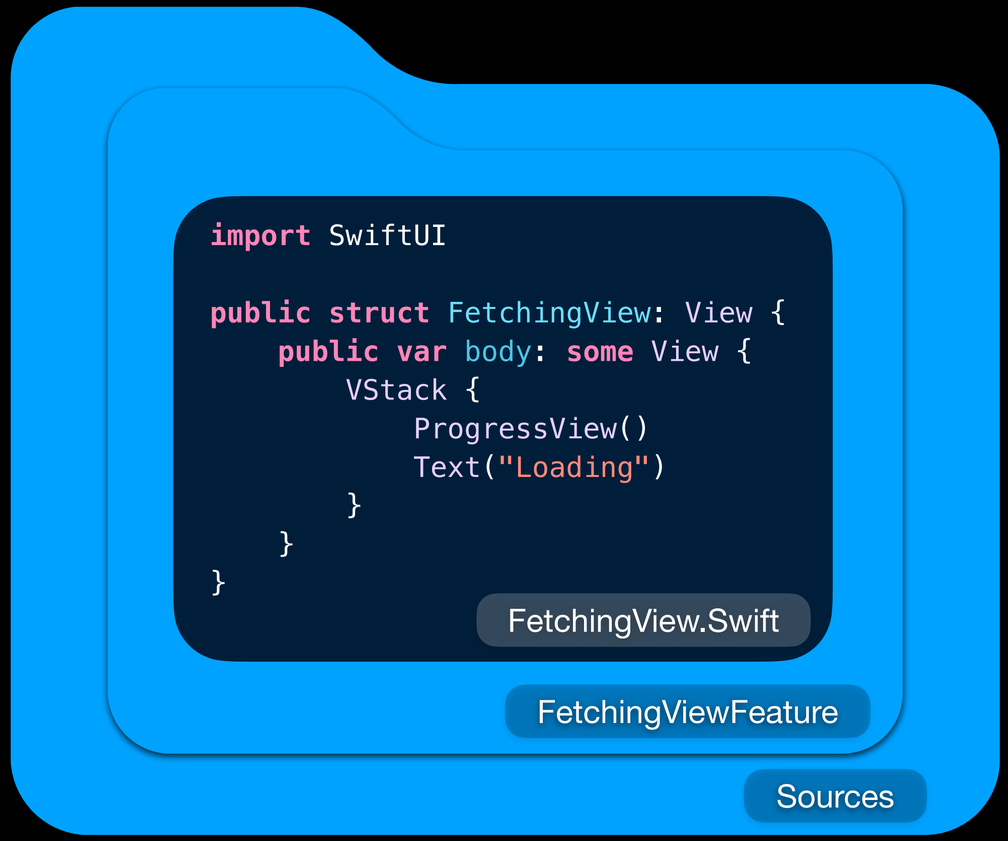
Package Manager Vocabulary

Directory Structure



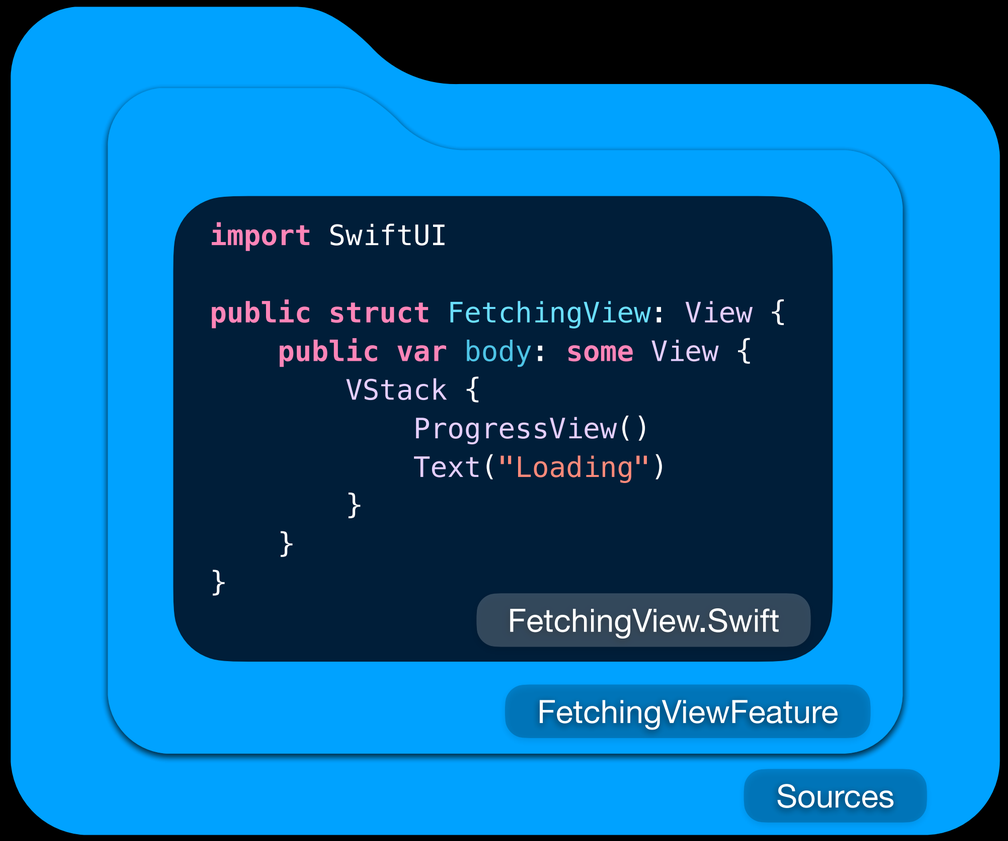
Package Manager Vocabulary

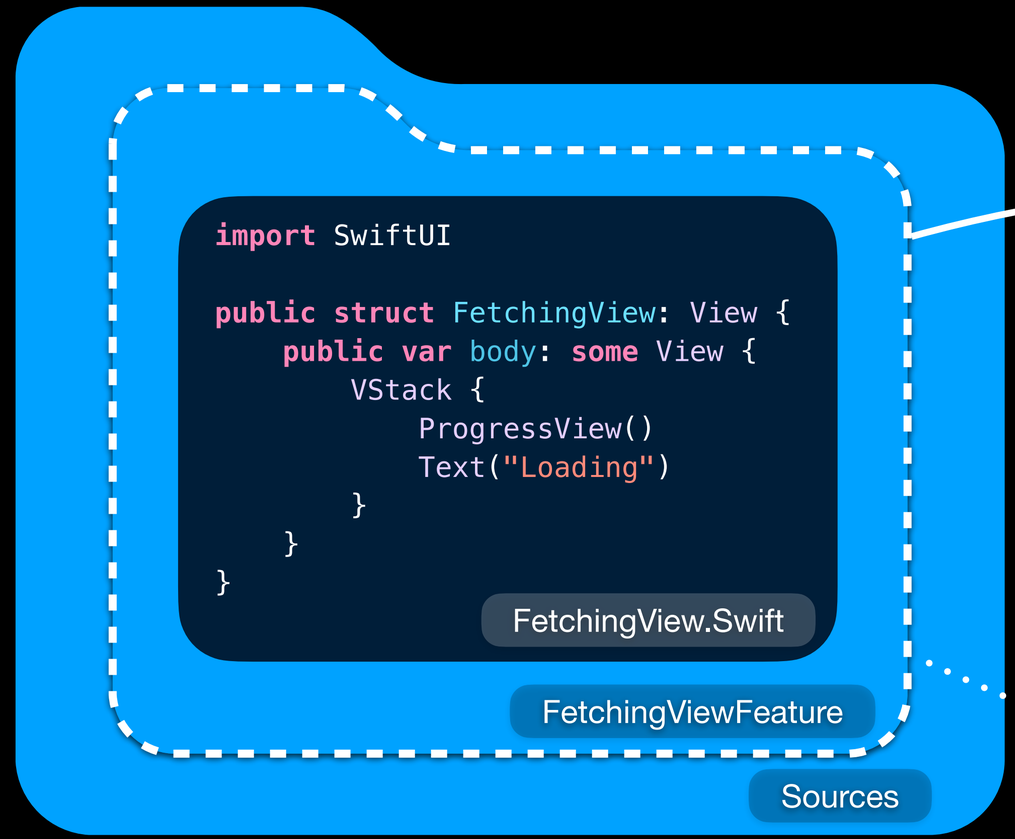
Directory Structure



Mastering SwiftPM

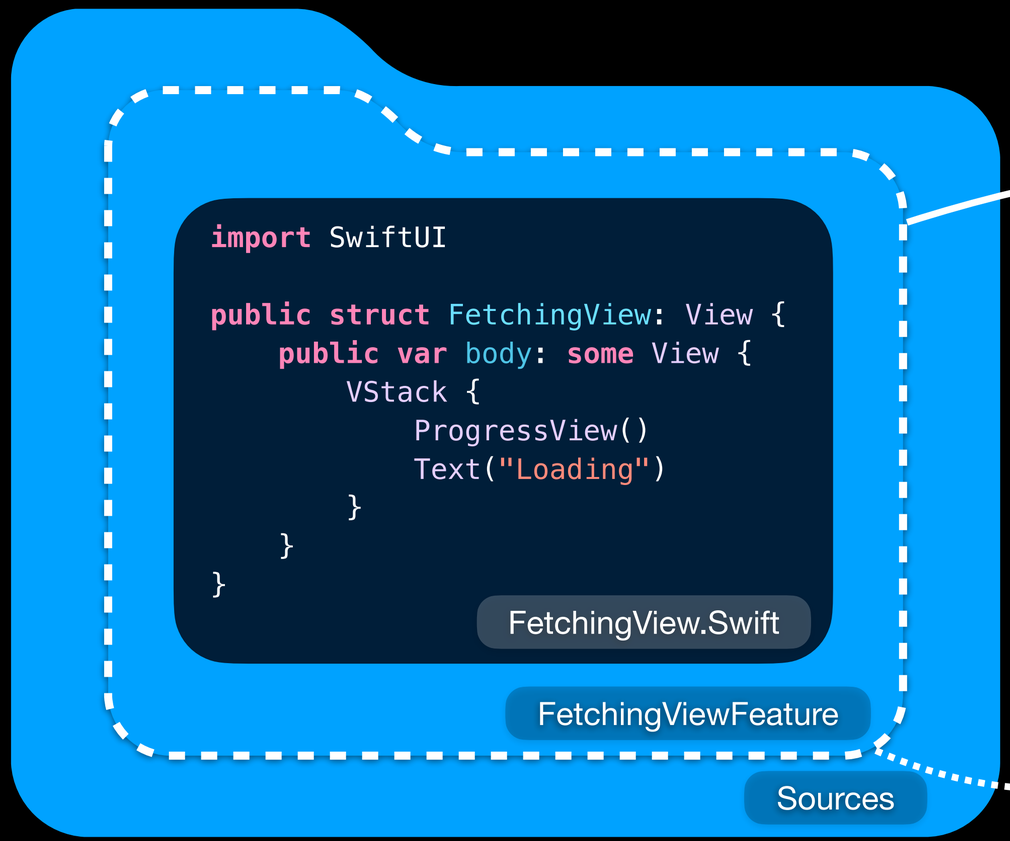
Ratnesh Jain


import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources



Target

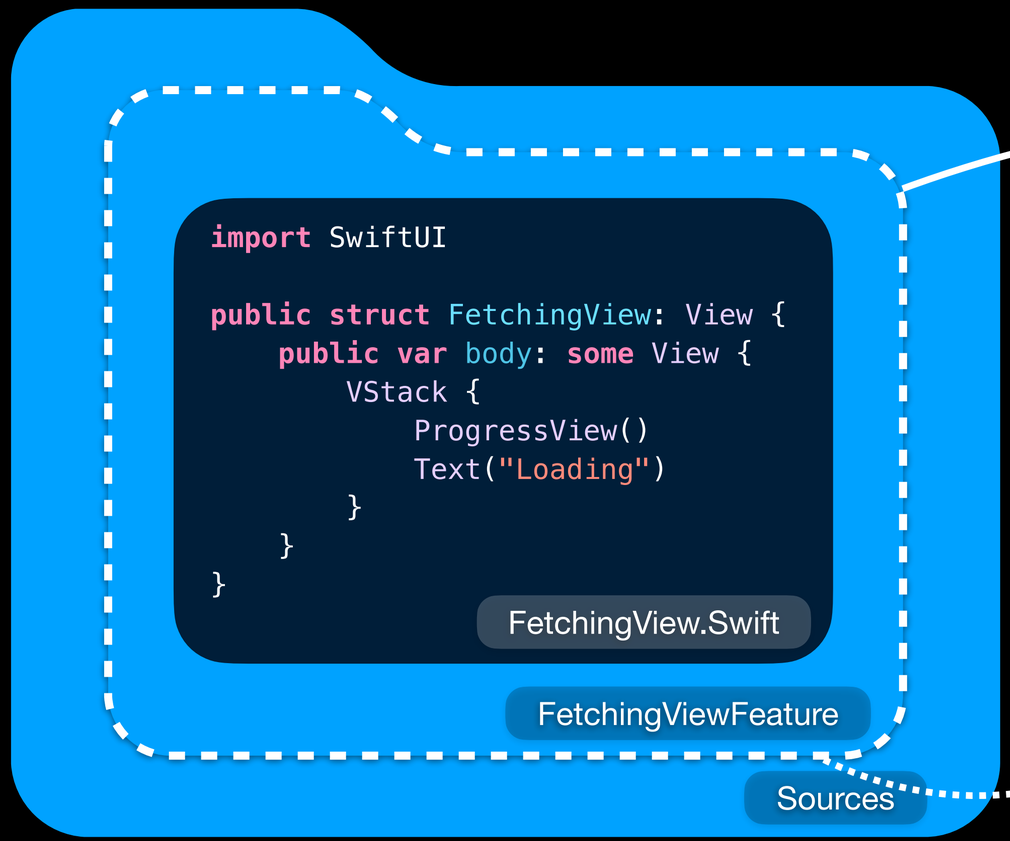
Module



Target

Module

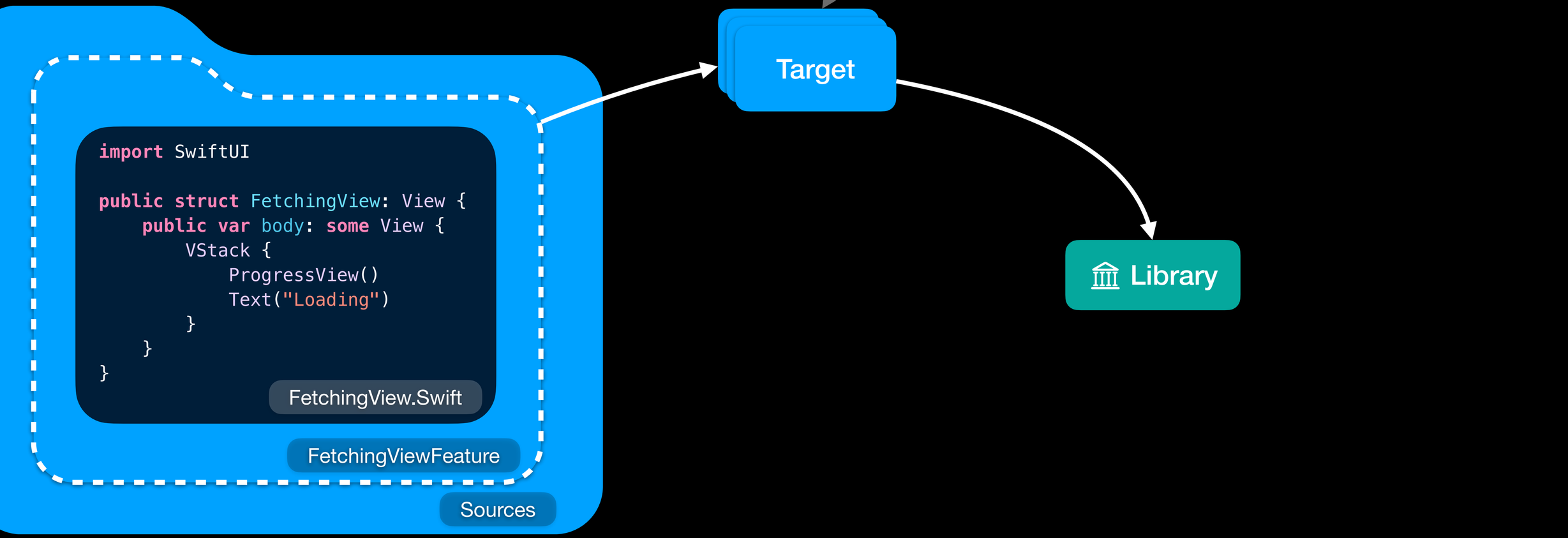
```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum vers

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

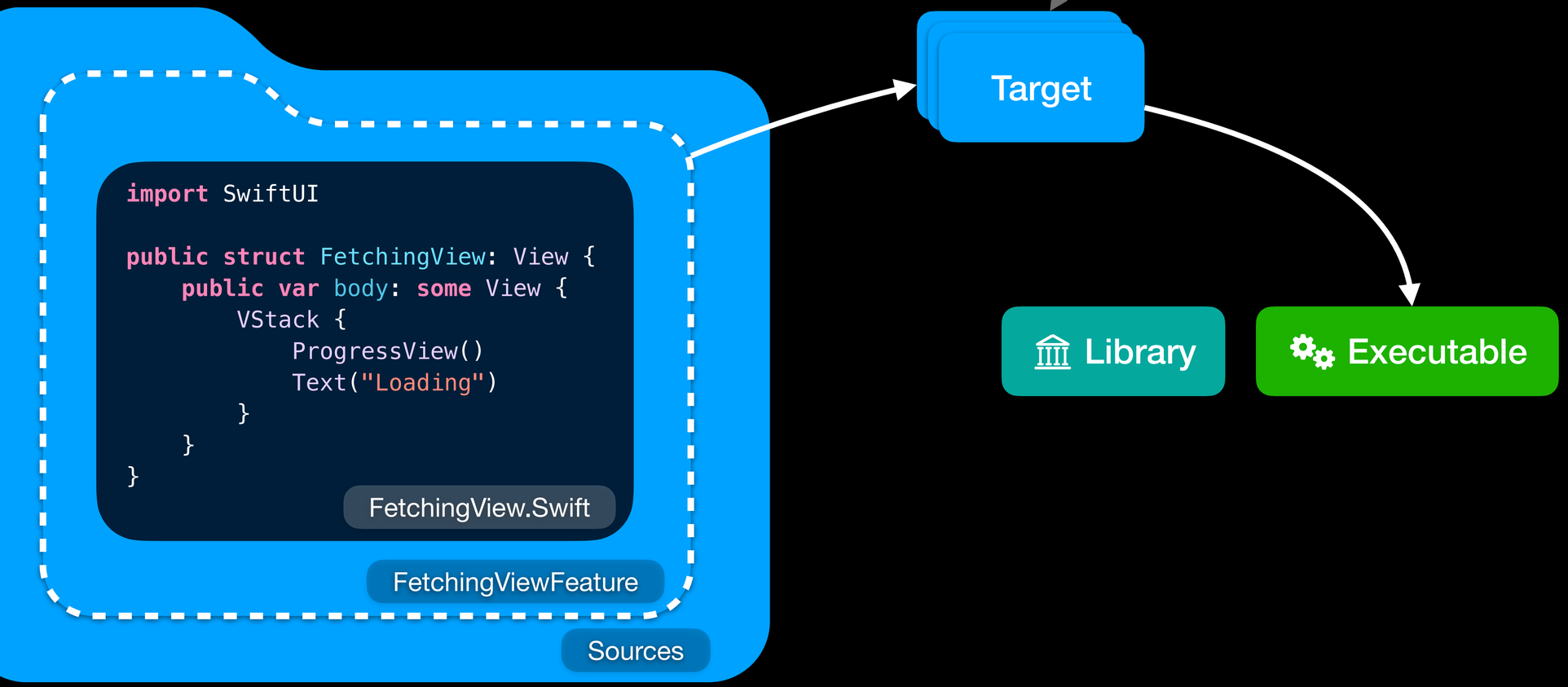


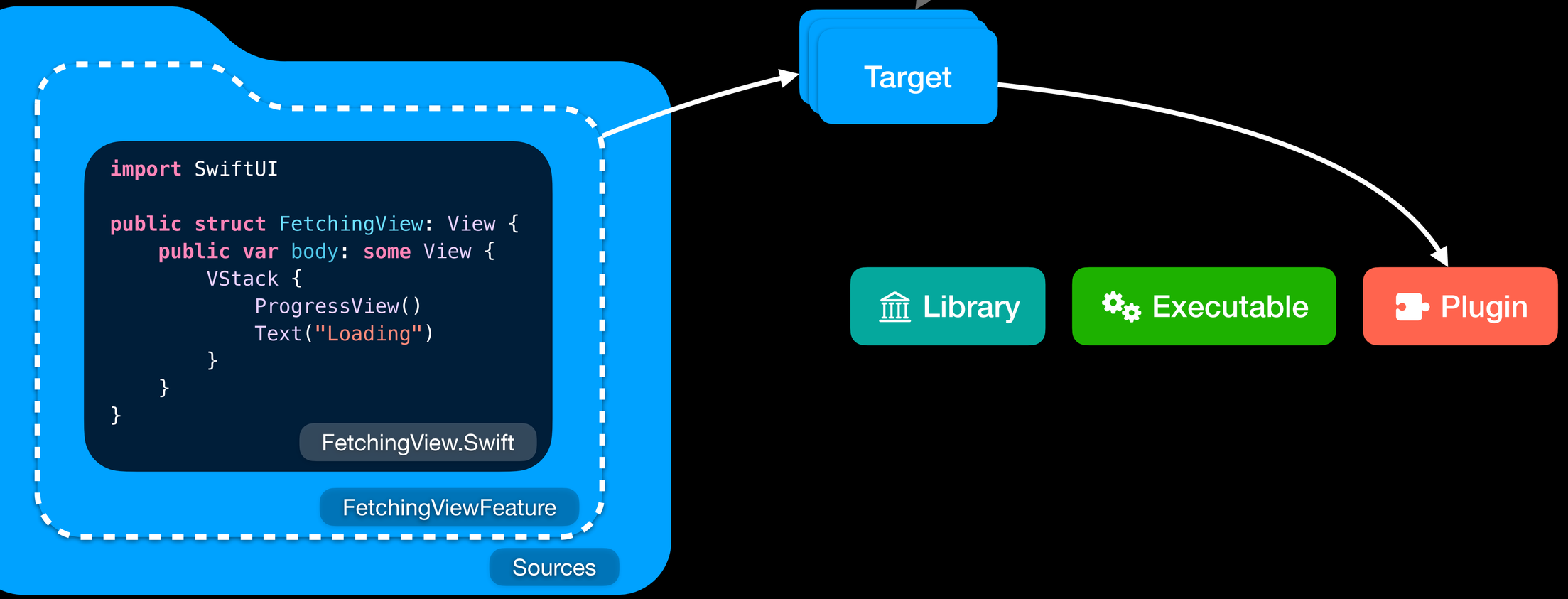
Target

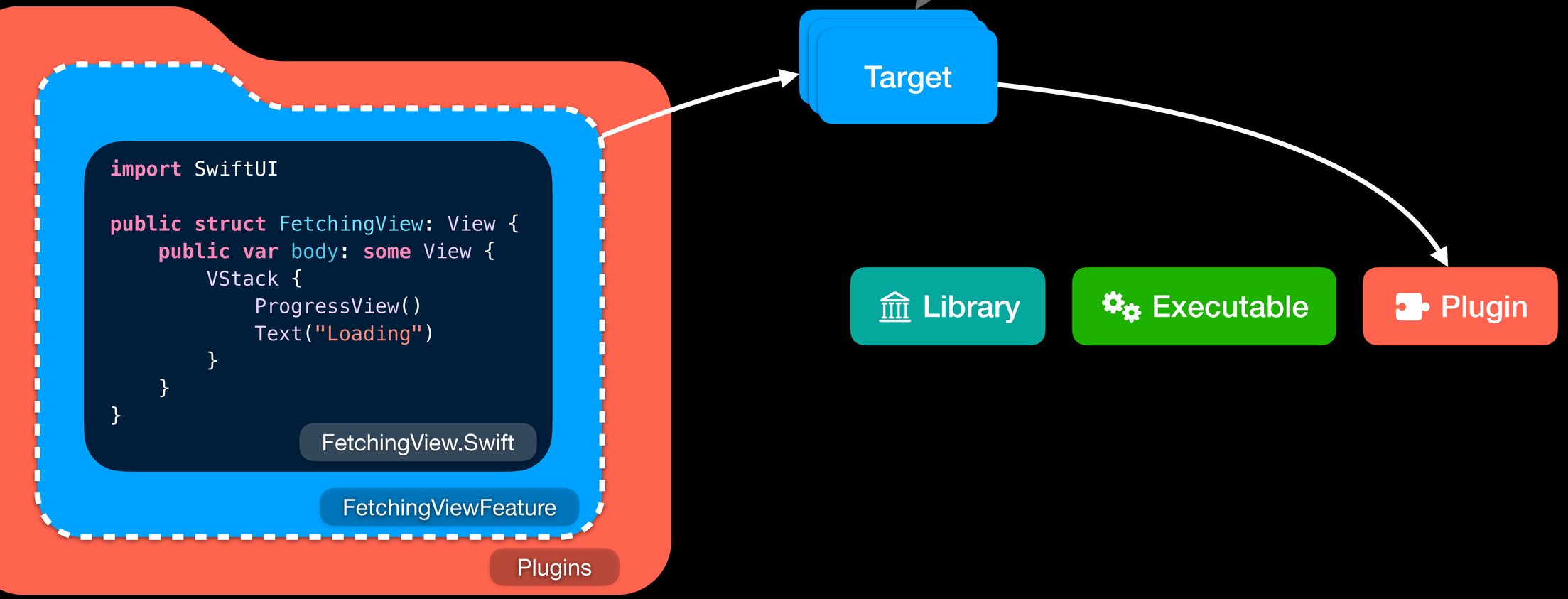
Module

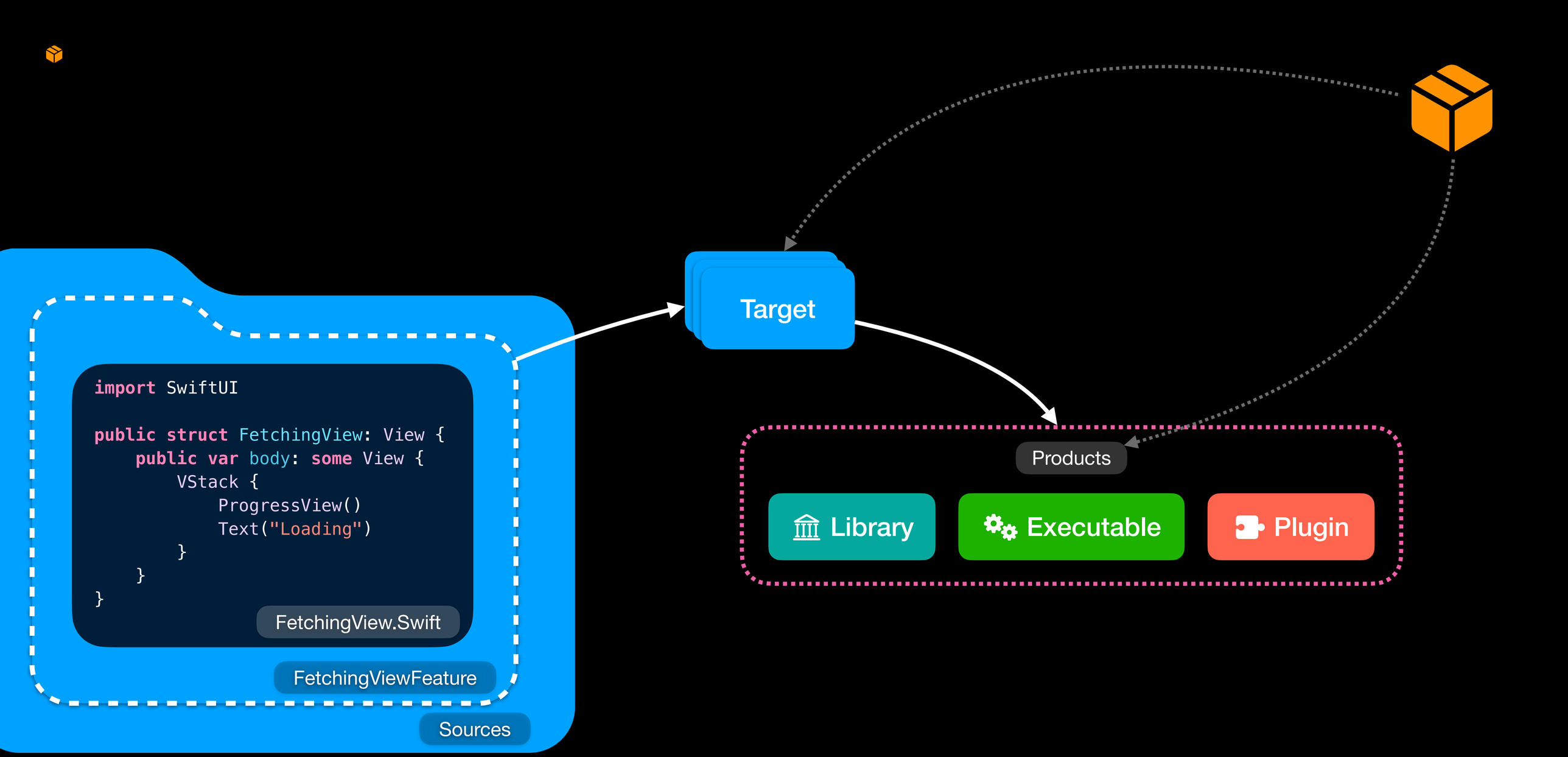
```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum vers

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

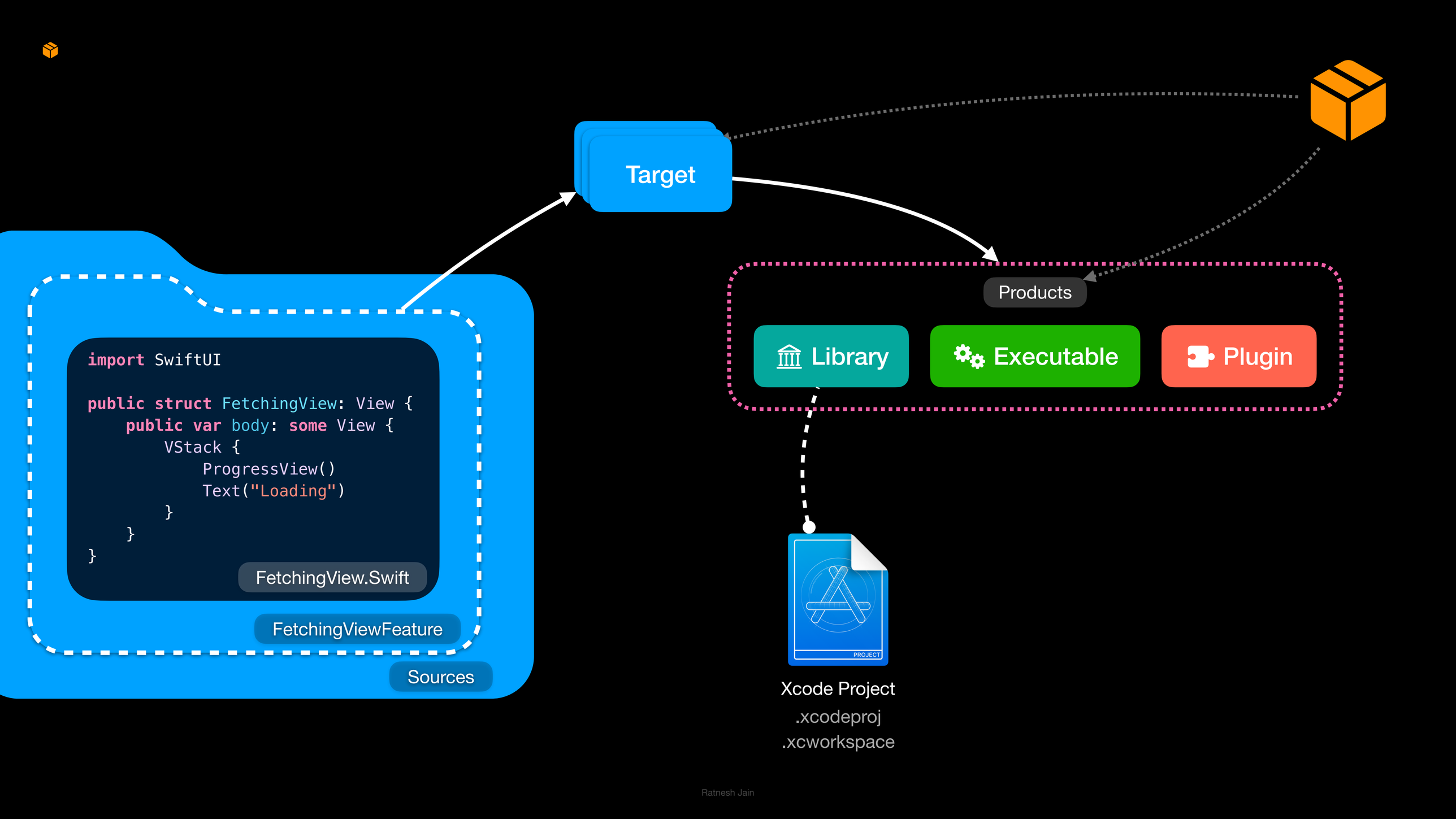

import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Target
Library

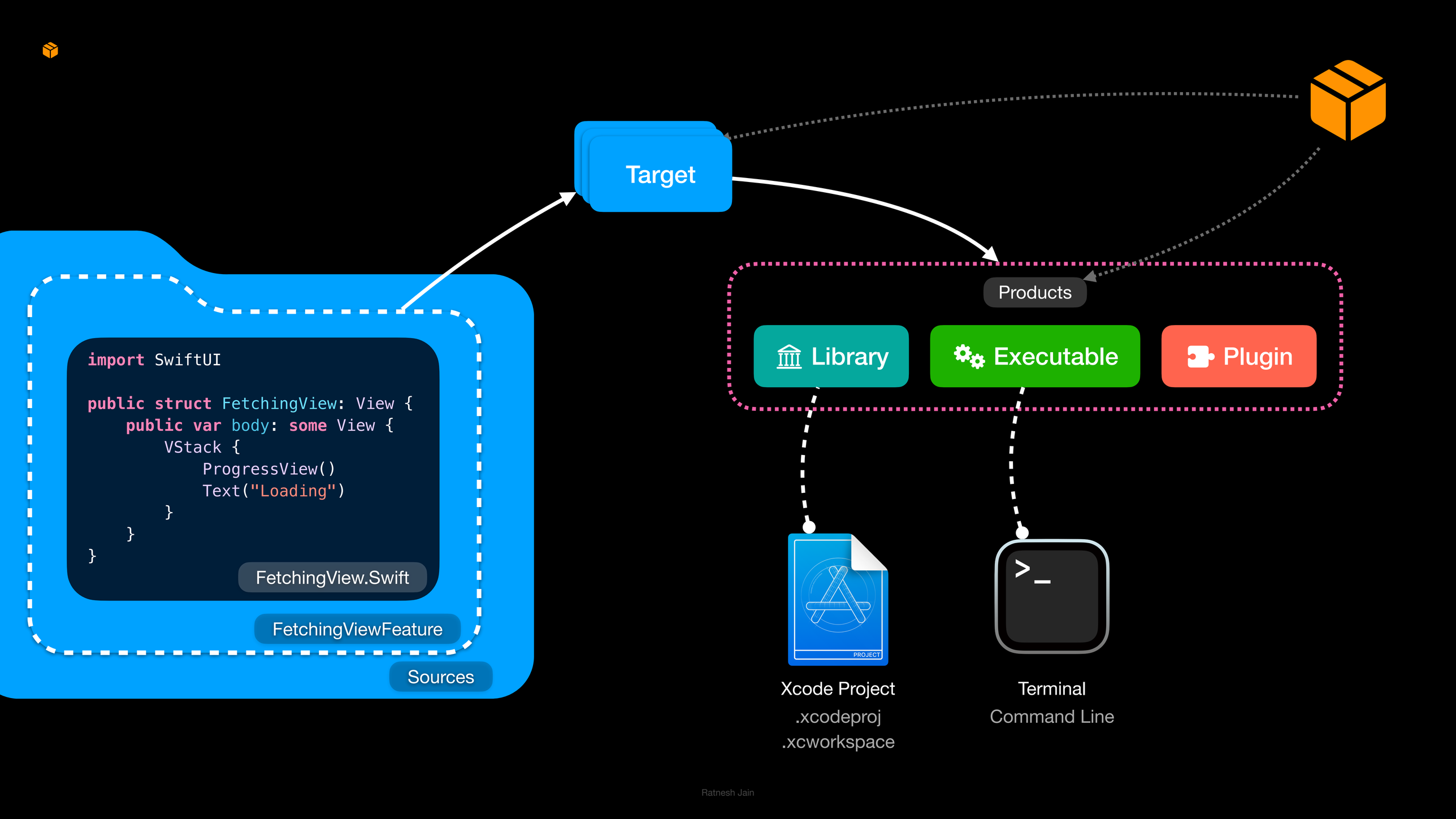

import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Target
Library
Executable

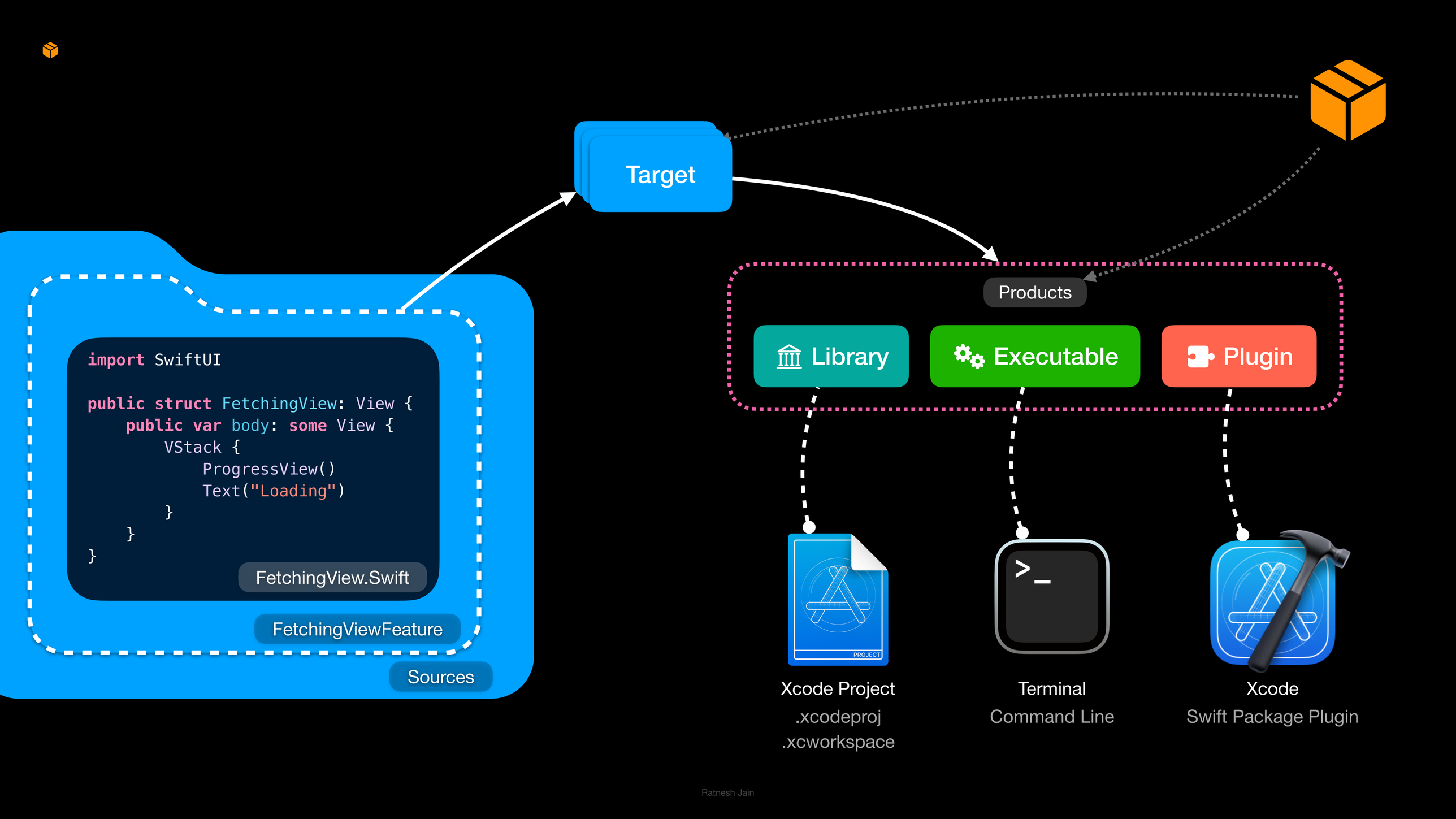

import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Target
Library
Executable
Plugin

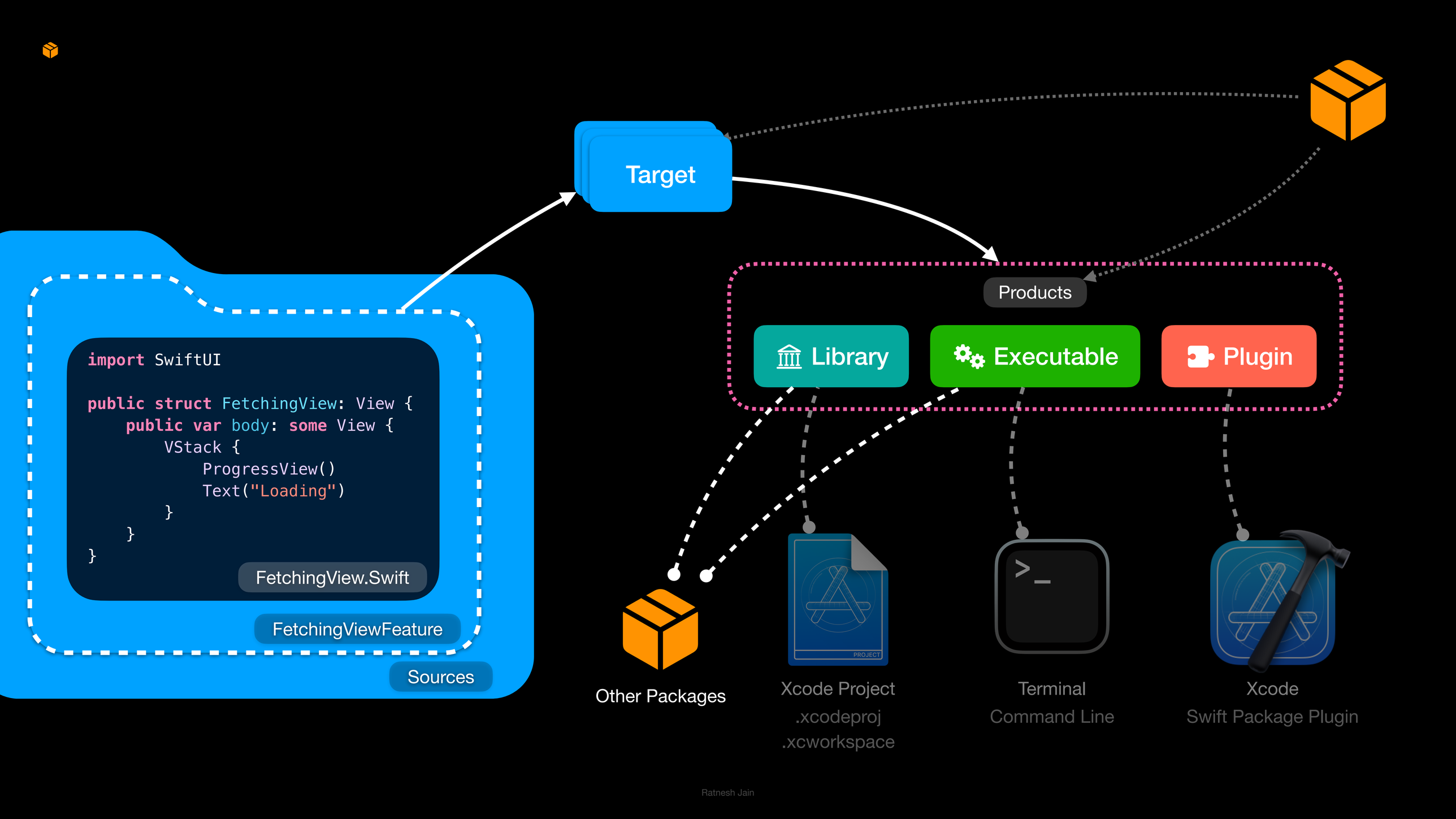

import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Plugins
Target
Library
Executable
Plugin


import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Target
Products
Library
Executable
Plugin


Target
Products
Library
Executable
Plugin
import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
PROJECT
Xcode Project
.xcodeproj
.xcworkspace


Target
Products
Library
Executable
Plugin
import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
PROJECT
Xcode Project
.xcodeproj
.xcworkspace
Terminal
Command Line


Target
import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Products
Library
Executable
Plugin
PROJECT
Xcode Project
.xcodeproj
.xcworkspace
Terminal
Command Line
Xcode
Swift Package Plugin


Target
import SwiftUI
public struct FetchingView: View {
public var body: some View {
VStack {
ProgressView()
Text("Loading")
}
}
}
FetchingView.Swift
FetchingViewFeature
Sources
Products
Library
Executable
Plugin
Other Packages
Xcode Project
.xcodeproj
.xcworkspace
Terminal
Command Line
Xcode
Swift Package Plugin

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Dependencies

```swift
import SwiftUI

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading")
        }
    }
}
```

Dependencies

```
import SwiftUI
import Alamofire

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading")
        }
    }
}
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

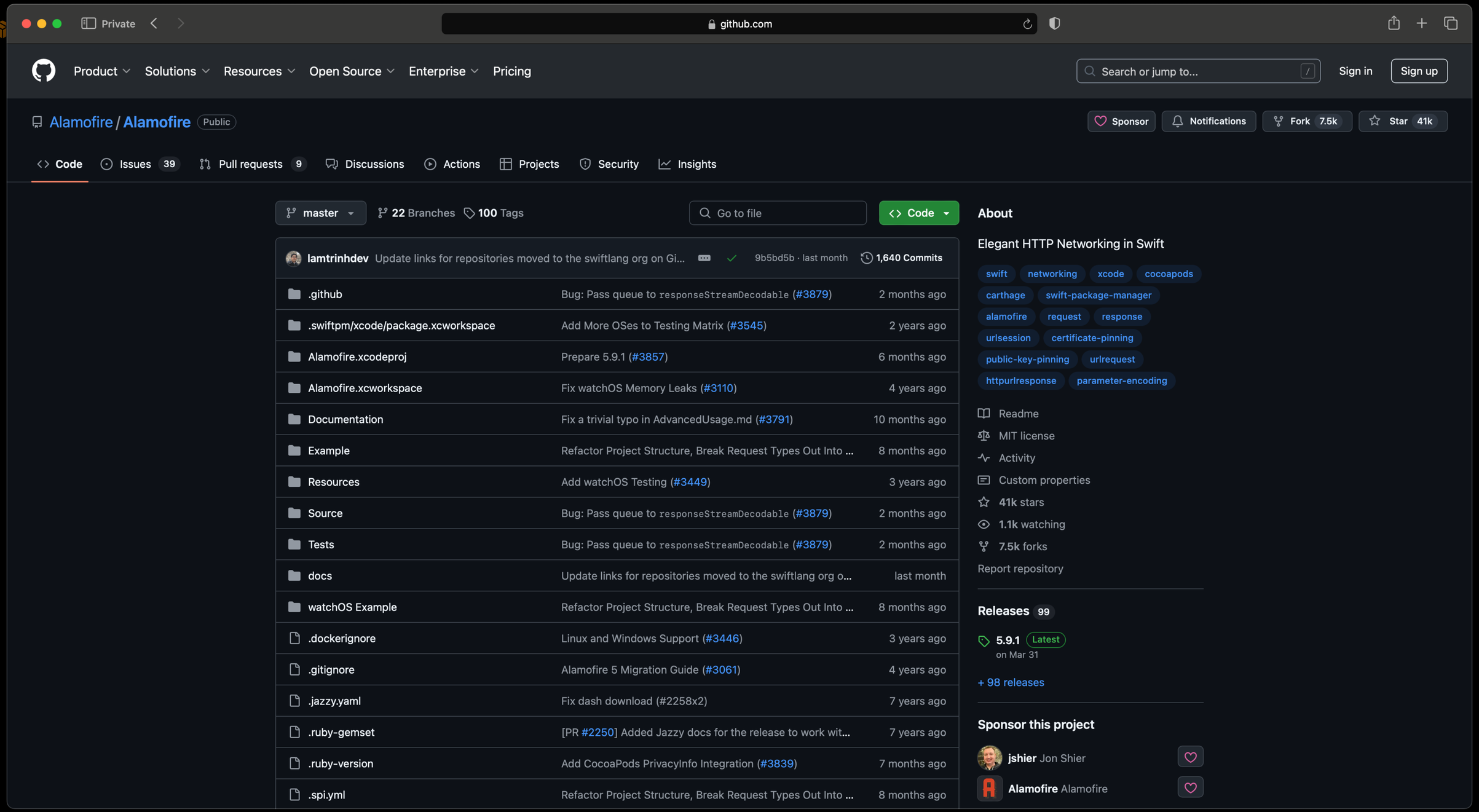
Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Where to get the VALID Dependency URL and Version number?



README MIT license

- Alamofire 4.0 Migration Guide
- Alamofire 3.0 Migration Guide
- Alamofire 2.0 Migration Guide

Communication

- If you **need help with making network requests** using Alamofire, use Stack Overflow and tag `alamofire`.
- If you need to **find or understand an API**, check our documentation or Apple's documentation for `URLSession`, on top of which Alamofire is built.
- If you need **help with an Alamofire feature**, use our forum on swift.org.
- If you'd like to **discuss Alamofire best practices**, use our forum on swift.org.
- If you'd like to **discuss a feature request**, use our forum on swift.org.
- If you **found a bug**, open an issue here on GitHub and follow the guide. The more detail the better!

Installation

Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

Once you have your Swift package set up, adding Alamofire as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift` or the Package list in Xcode.

```
dependencies: [
    .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.9.1"
]
```

Normally you'll want to depend on the `Alamofire` target:

```
.product(name: "Alamofire", package: "Alamofire")
```

But if you want to force Alamofire to be dynamically linked (do not do this unless you're sure you need it), you can depend on the `AlamofireDynamic` target:

```
.product(name: "AlamofireDynamic", package: "Alamofire")
```

Installation

Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

Once you have your Swift package set up, adding Alamofire as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift` or the Package list in Xcode.

```
dependencies: [
    .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.9.1"
]
```

Normally you'll want to depend on the `Alamofire` target:

```
.product(name: "Alamofire", package: "Alamofire")
```

But if you want to force Alamofire to be dynamically linked (do not do this unless you're sure you need it), you can depend on the `AlamofireDynamic` target:

```
.product(name: "AlamofireDynamic", package: "Alamofire")
```

Installation

Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

Once you have your Swift package set up, adding Alamofire as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift` or the Package list in Xcode.

```
dependencies: [
    .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.9.1"
]
```

Normally you'll want to depend on the `Alamofire` target:

```
.product(name: "Alamofire", package: "Alamofire")
```

But if you want to force Alamofire to be dynamically linked (do not do this unless you're sure you need it), you can depend on the `AlamofireDynamic` target:

```
.product(name: "AlamofireDynamic", package: "Alamofire")
```

README MIT license

- Alamofire 4.0 Migration Guide
- Alamofire 3.0 Migration Guide
- Alamofire 2.0 Migration Guide

Communication

- If you **need help with making network requests** using Alamofire, use Stack Overflow and tag `alamofire`.
- If you need to **find or understand an API**, check our documentation or Apple's documentation for `URLSession`, on top of which Alamofire is built.
- If you need **help with an Alamofire feature**, use our forum on swift.org.
- If you'd like to **discuss Alamofire best practices**, use our forum on swift.org.
- If you'd like to **discuss a feature request**, use our forum on swift.org.
- If you **found a bug**, open an issue here on GitHub and follow the guide. The more detail the better!

Installation

Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

Once you have your Swift package set up, adding Alamofire as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift` or the Package list in Xcode.

```
dependencies: [
    .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.9.1")
]
```

Normally you'll want to depend on the `Alamofire` target:

```
.product(name: "Alamofire", package: "Alamofire")
```

But if you want to force Alamofire to be dynamically linked (do not do this unless you're sure you need it), you can depend on the `AlamofireDynamic` target:

```
.product(name: "AlamofireDynamic", package: "Alamofire")
```

README MIT license

- Alamofire 4.0 Migration Guide
- Alamofire 3.0 Migration Guide
- Alamofire 2.0 Migration Guide

Communication

- If you **need help with making network requests** using Alamofire, use Stack Overflow and tag `alamofire`.
- If you need to **find or understand an API**, check our documentation or Apple's documentation for `URLSession`, on top of which Alamofire is built.
- If you need **help with an Alamofire feature**, use our forum on swift.org.
- If you'd like to **discuss Alamofire best practices**, use our forum on swift.org.
- If you'd like to **discuss a feature request**, use our forum on swift.org.
- If you **found a bug**, open an issue here on GitHub and follow the guide. The more detail the better!

Installation

Swift Package Manager

The Swift Package Manager is a tool for automating the distribution of Swift code and is integrated into the `swift` compiler.

Once you have your Swift package set up, adding Alamofire as a dependency is as easy as adding it to the `dependencies` value of your `Package.swift` or the Package list in Xcode.

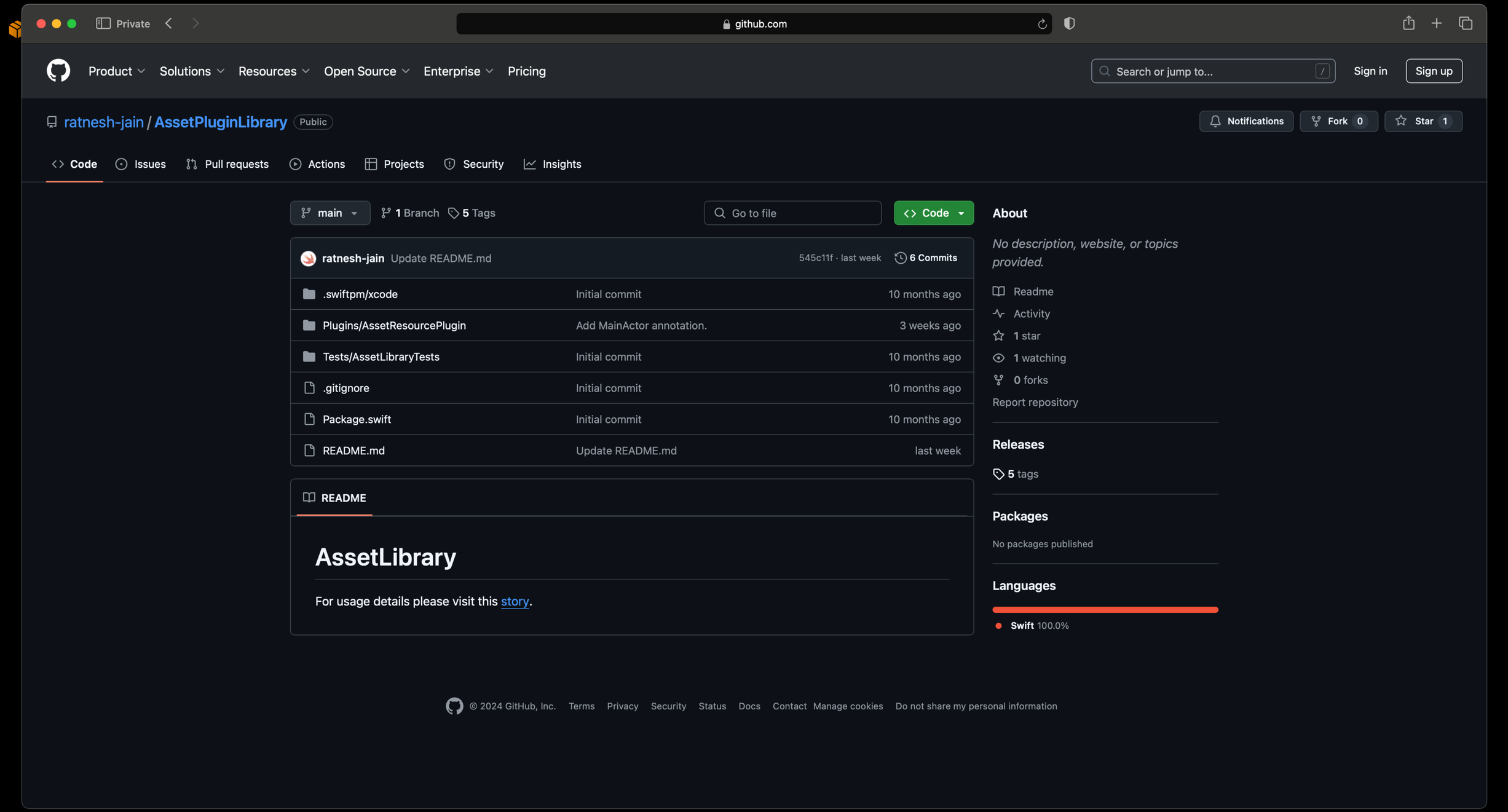
```
dependencies: [
    .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.9.1"
]
```

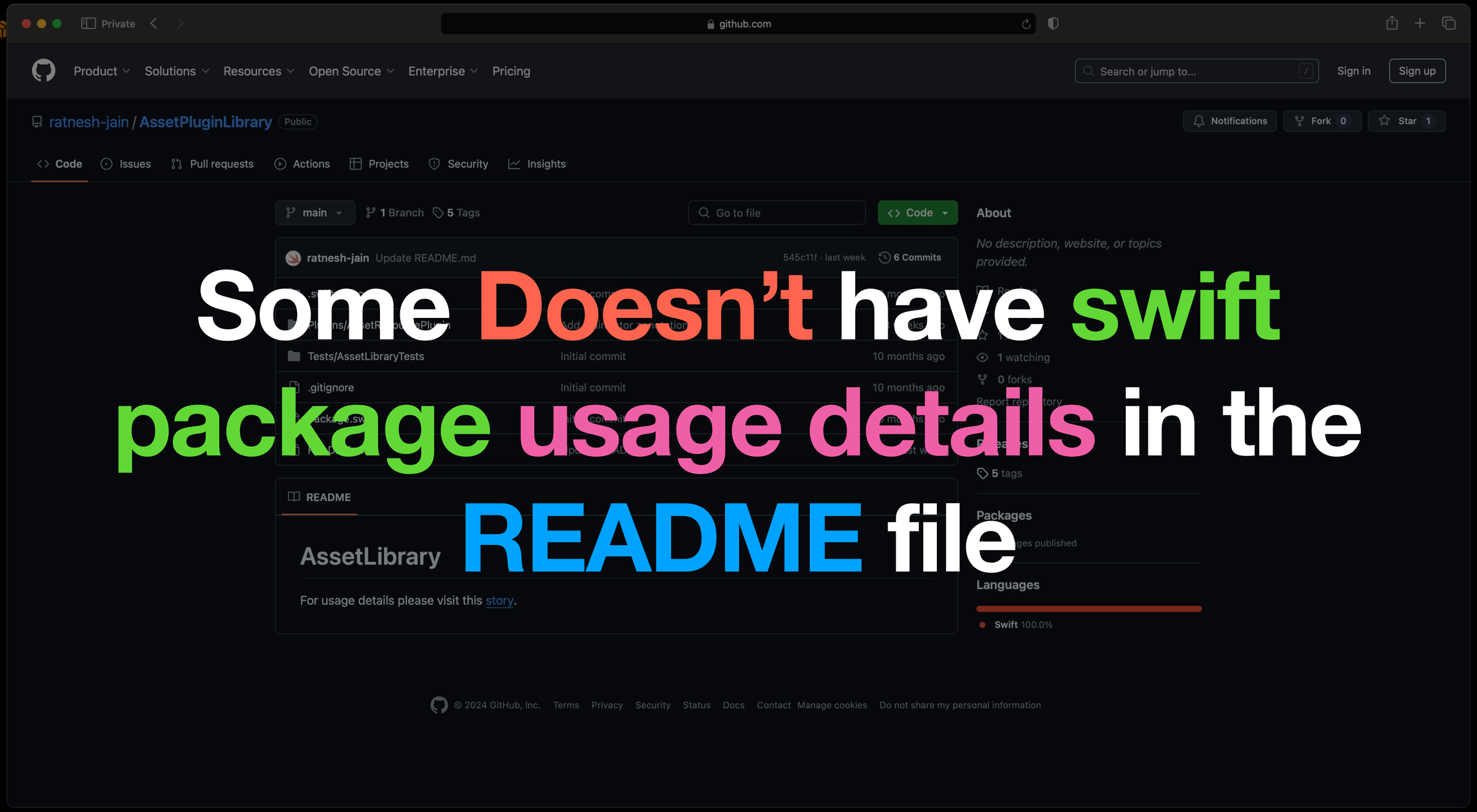
Normally you'll want to depend on the `Alamofire` target:

```
.product(name: "Alamofire", package: "Alamofire")
```

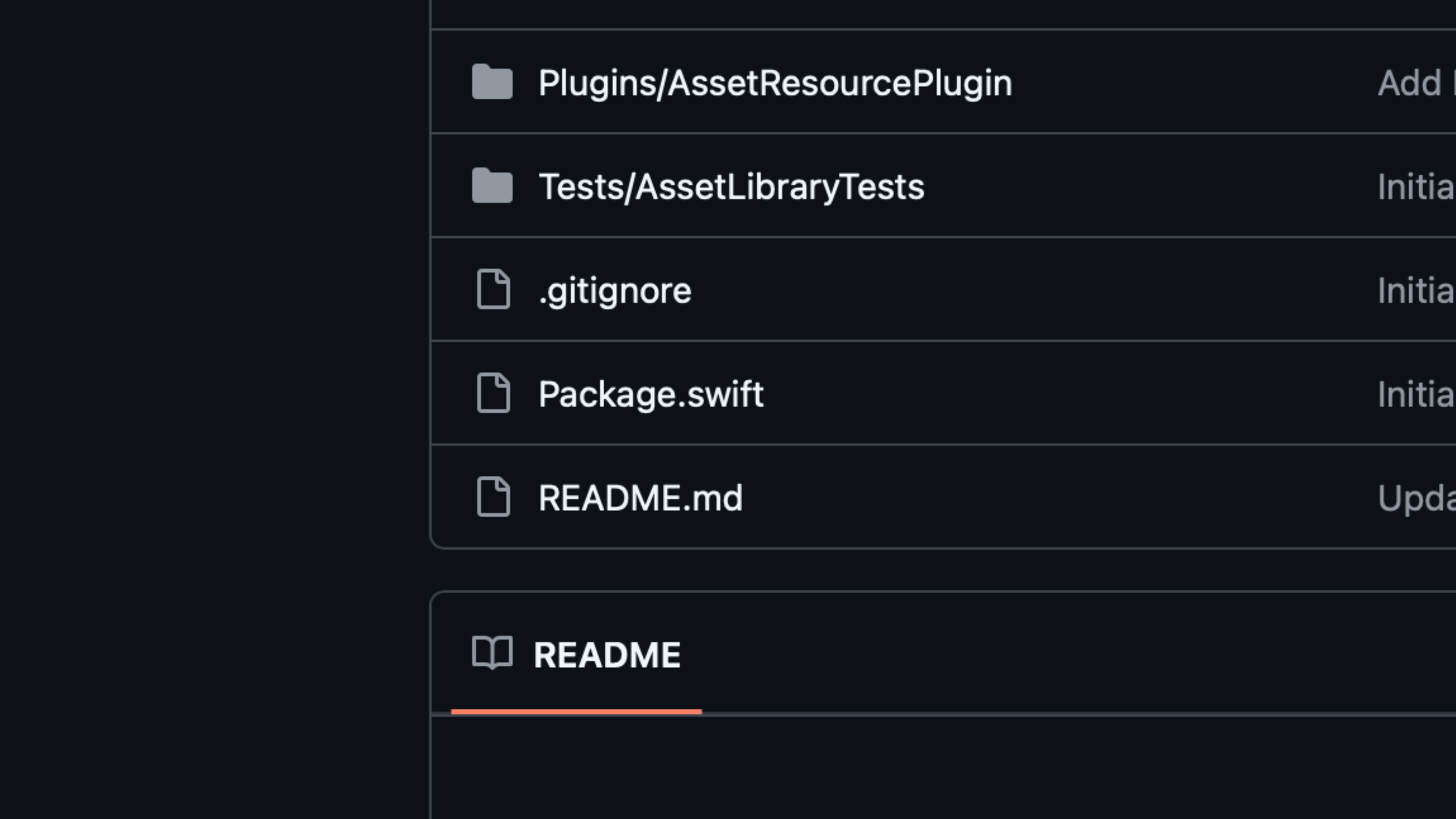
But if you want to force Alamofire to be dynamically linked (do not do this unless you're sure you need it), you can depend on the `AlamofireDynamic` target:

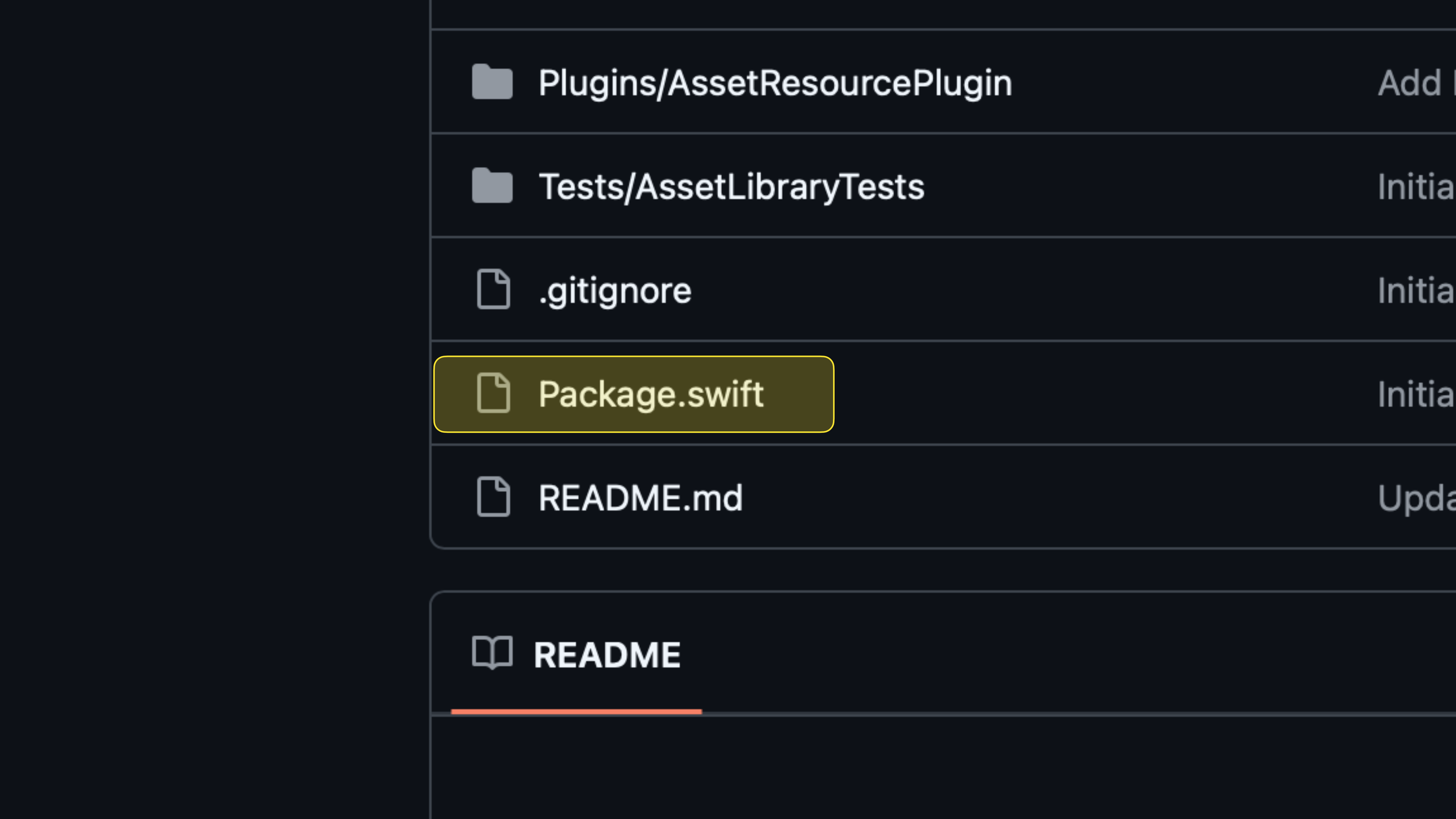
```
.product(name: "AlamofireDynamic", package: "Alamofire")
```


Private
github.com
Product
Solutions
Resources
Open Source
Enterprise
Pricing
Search or jump to...
Sign in
Sign up
ratnesh-jain / AssetPluginLibrary
Public
Notifications
Fork 0
Star 1
Code
Issues
Pull requests
Actions
Projects
Security
Insights
main
1 Branch
5 Tags
Go to file
Code
ratnesh-jain Update README.md
545c11f · last week
6 Commits
.swiftpm/xcode
Initial commit
10 months ago
Plugins/AssetResourcePlugin
Add MainActor annotation.
3 weeks ago
Tests/AssetLibraryTests
Initial commit
10 months ago
.gitignore
Initial commit
10 months ago
Package.swift
Initial commit
10 months ago
README.md
Update README.md
last week
README
AssetLibrary
For usage details please visit this story.
About
No description, website, or topics provided.
Readme
Activity
1 star
1 watching
0 forks
Report repository
Releases
5 tags
Packages
No packages published
Languages
Swift 100.0%
© 2024 GitHub, Inc.
Terms
Privacy
Security
Status
Docs
Contact
Manage cookies
Do not share my personal information



Some Doesn't have swift package usage details in the README file


Plugins/AssetResourcePlugin
Add
Tests/AssetLibraryTests
Initia
.gitignore
Initia
Package.swift
Initia
README.md
Upda
README


Plugins/AssetResourcePlugin
Add
Tests/AssetLibraryTests
Initia
.gitignore
Initia
Package.swift
Initia
README.md
Upda
README

ratnesh-jain / AssetPluginLibrary Public

Notifications Fork 0 Star 1

Code Issues Pull requests Actions Projects Security Insights

main 1 Branch 5 Tags Go to file Code

ratnesh-jain Update README.md 545c11f · last week 6 Commits

.swiftpm/xcode	Initial commit	10 months ago
Plugins/AssetResourcePlugin	Add MainActor annotation.	3 weeks ago
Tests/AssetLibraryTests	Initial commit	10 months ago
.gitignore	Initial commit	10 months ago
Package.swift	Initial commit	10 months ago
README.md	Update README.md	last week

README

AssetLibrary

For usage details please visit this story.

About

No description, website, or topics provided.

Readme
Activity
1 star
1 watching
0 forks
Report repository

Releases

5 tags

Packages

No packages published

Languages

Swift 100.0%

3 weeks ago

10 months ago

10 months ago

10 months ago

last week

Activity

1 star

1 watching

0 forks

Report repository

Releases

5 tags

Packages

No packages published

Languages

ratnesh-jain / AssetPluginLibrary Public

Notifications Fork 0 Star 1

Code Issues Pull requests Actions Projects Security Insights

main 1 Branch 5 Tags Go to file Code

ratnesh-jain Update README.md 545c11f · last week 6 Commits

.swiftpm/xcode	Initial commit	10 months ago
Plugins/AssetResourcePlugin	Add MainActor annotation.	3 weeks ago
Tests/AssetLibraryTests	Initial commit	10 months ago
.gitignore	Initial commit	10 months ago
Package.swift	Initial commit	10 months ago
README.md	Update README.md	last week

README

AssetLibrary

For usage details please visit this story.

About

No description, website, or topics provided.

Readme
Activity
1 star
1 watching
0 forks
Report repository

Releases

5 tags

Packages

No packages published

Languages

Swift 100.0%

ratnesh-jain / AssetPluginLibrary Public

Releases Tags

Tags

0.0.5
3 weeks ago 9351d10 zip tar.gz

0.0.4
on Jun 8 d724beb zip tar.gz

0.0.3
on Jun 7 d43ccbd zip tar.gz

0.0.2
on Jan 15 2299c13 zip tar.gz

0.0.1
on Nov 16, 2023 1d779c1 zip tar.gz

Releases | Tags

Tags

0.0.5

3 weeks ago 9351d10 zip tar.gz

0.0.4

on Jun 8 d724beb zip tar.gz

0.0.3

on Jun 7 d43ccbd zip tar.gz

0.0.2

on Jan 15 2299c13 zip tar.gz

0.0.1

on Nov 16, 2023 1d779c1 zip tar.gz

Private

github.com

Product Solutions Resources Open Source Enterprise Pricing

Search or jump to... / Sign in Sign up

ratnesh-jain / AssetPluginLibrary Public

Notifications Fork 0 Star 1

Code Issues Pull requests Actions Projects Security Insights

Releases Tags

Tags

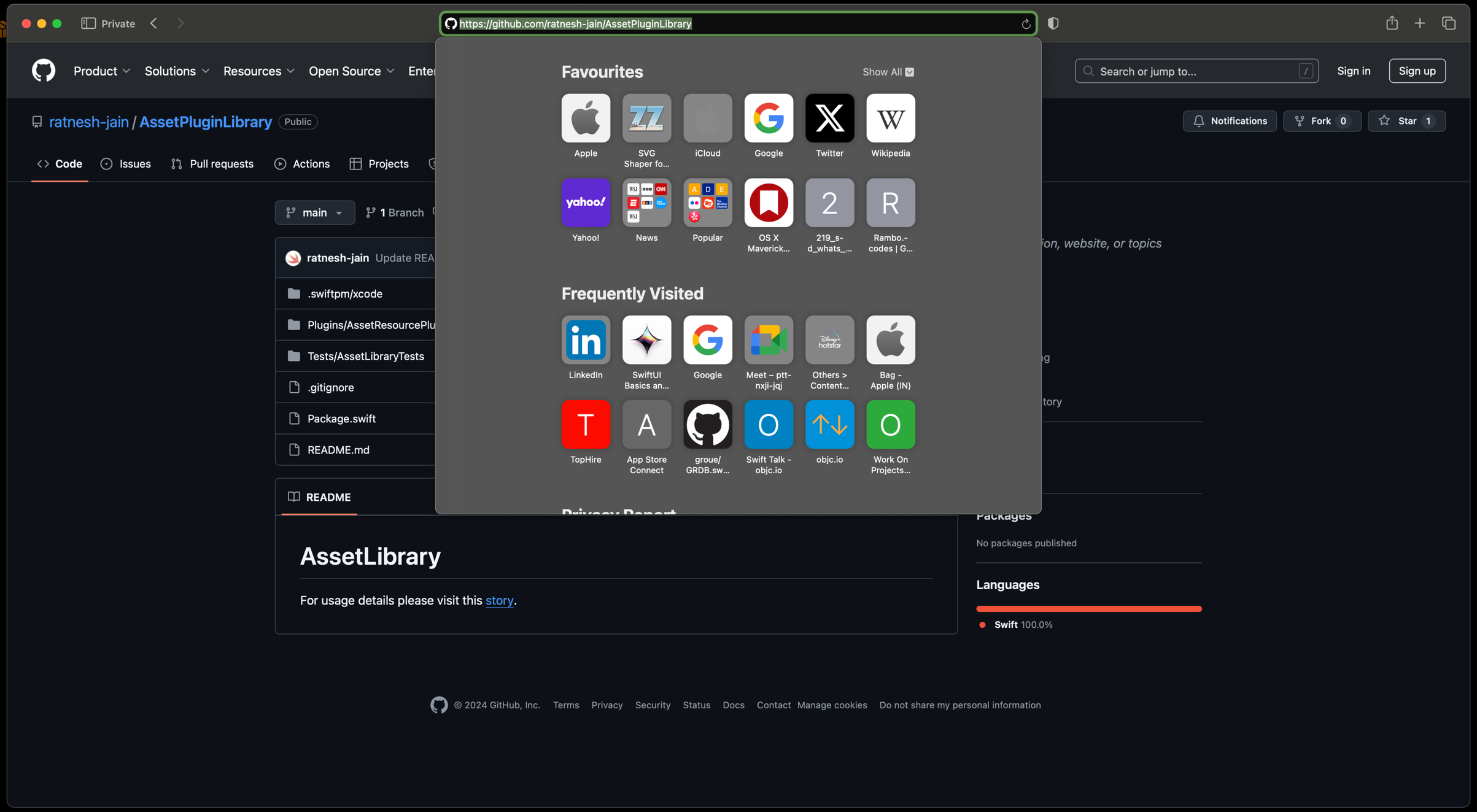
0.0.5
3 weeks ago 9351d10 zip tar.gz

0.0.4
on Jun 8 d724beb zip tar.gz

0.0.3
on Jun 7 d43ccbd zip tar.gz

0.0.2
on Jan 15 2299c13 zip tar.gz

0.0.1
on Nov 16, 2023 1d779c1 zip tar.gz

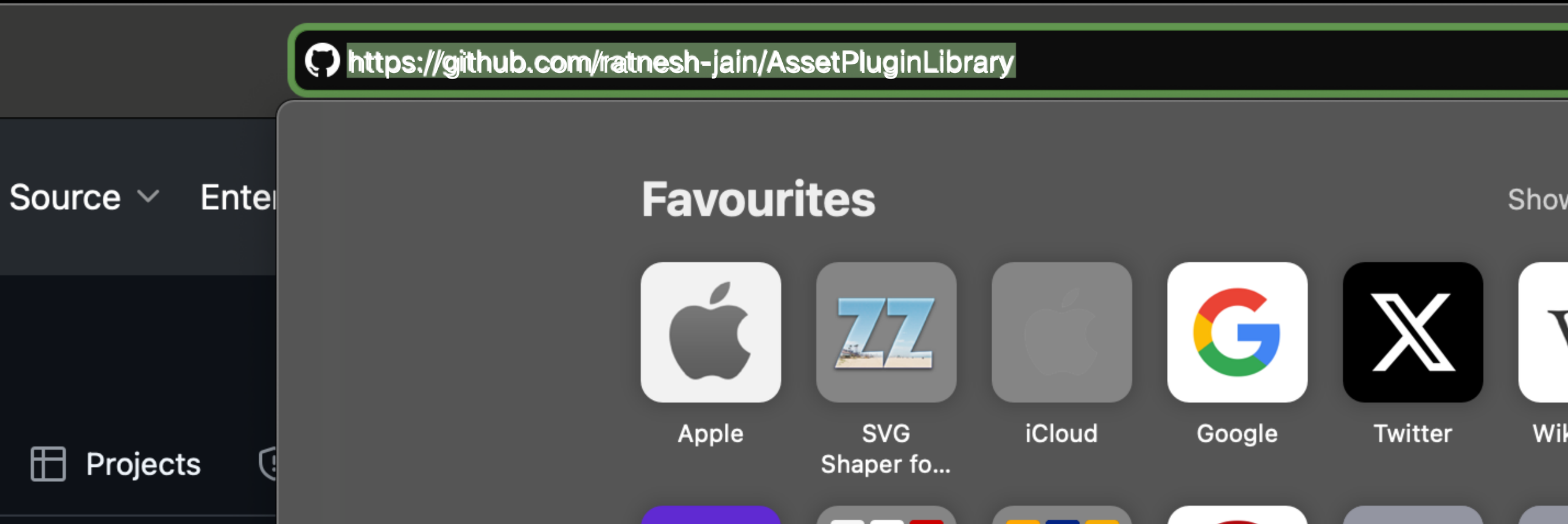


Ratnesh Jain

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
```

https://github.com/ratnesh-jain/AssetPluginLibrary



Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

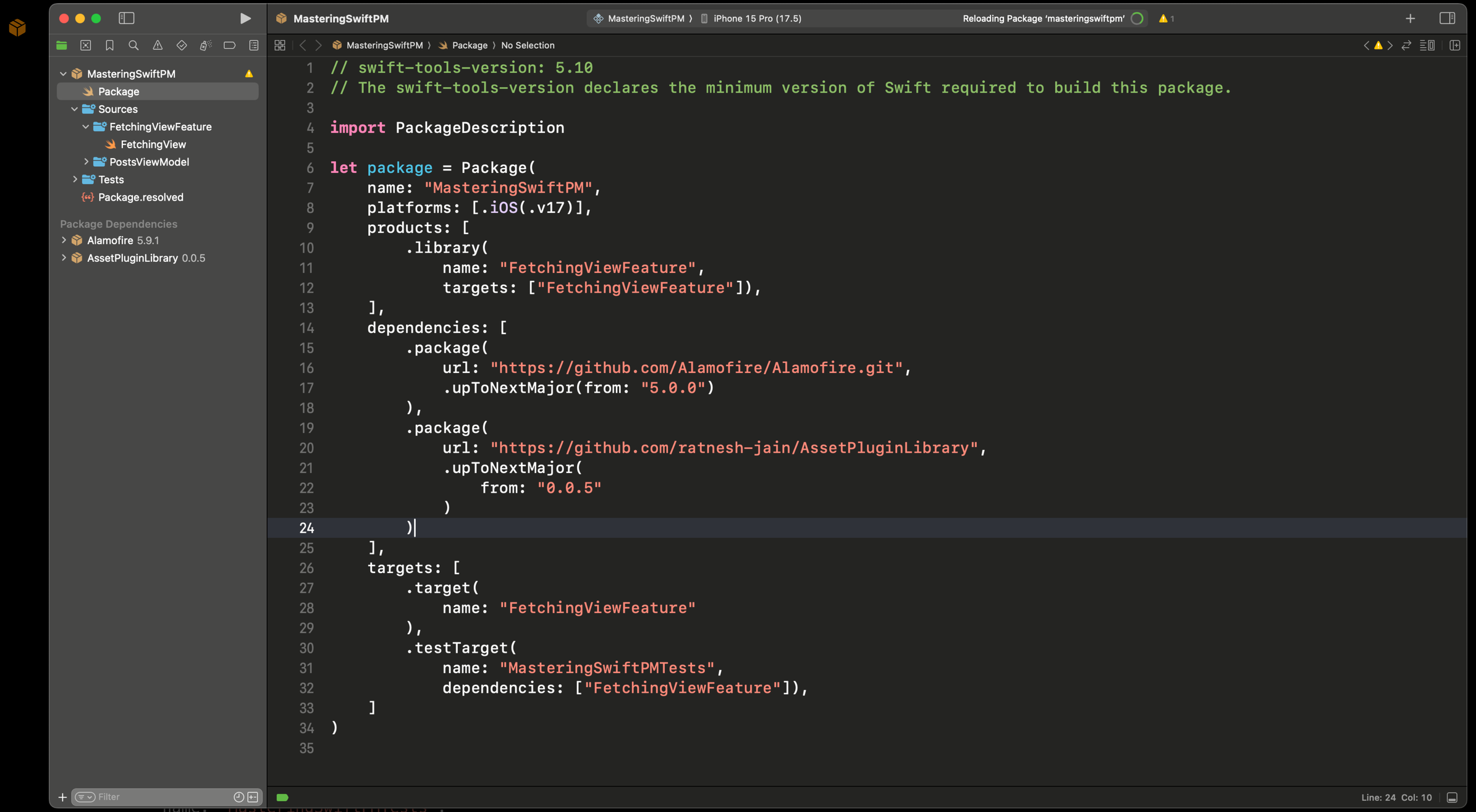
https://github.com/ratnesh-jain/AssetPluginLibrary

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/A
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
```



dependencies: ["FetchingViewFeature"]),

]

Ratnesh Jain

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
```

MasteringSwiftPM 〉 iPhone 15 Pro (17.5) Reloading Package ‘masteringswiftpm’ ⚠ 1

```
on
 5.10
ion declares the minimum version of Swift required to build this pac

on


ftPM",
17)],
```

Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
```

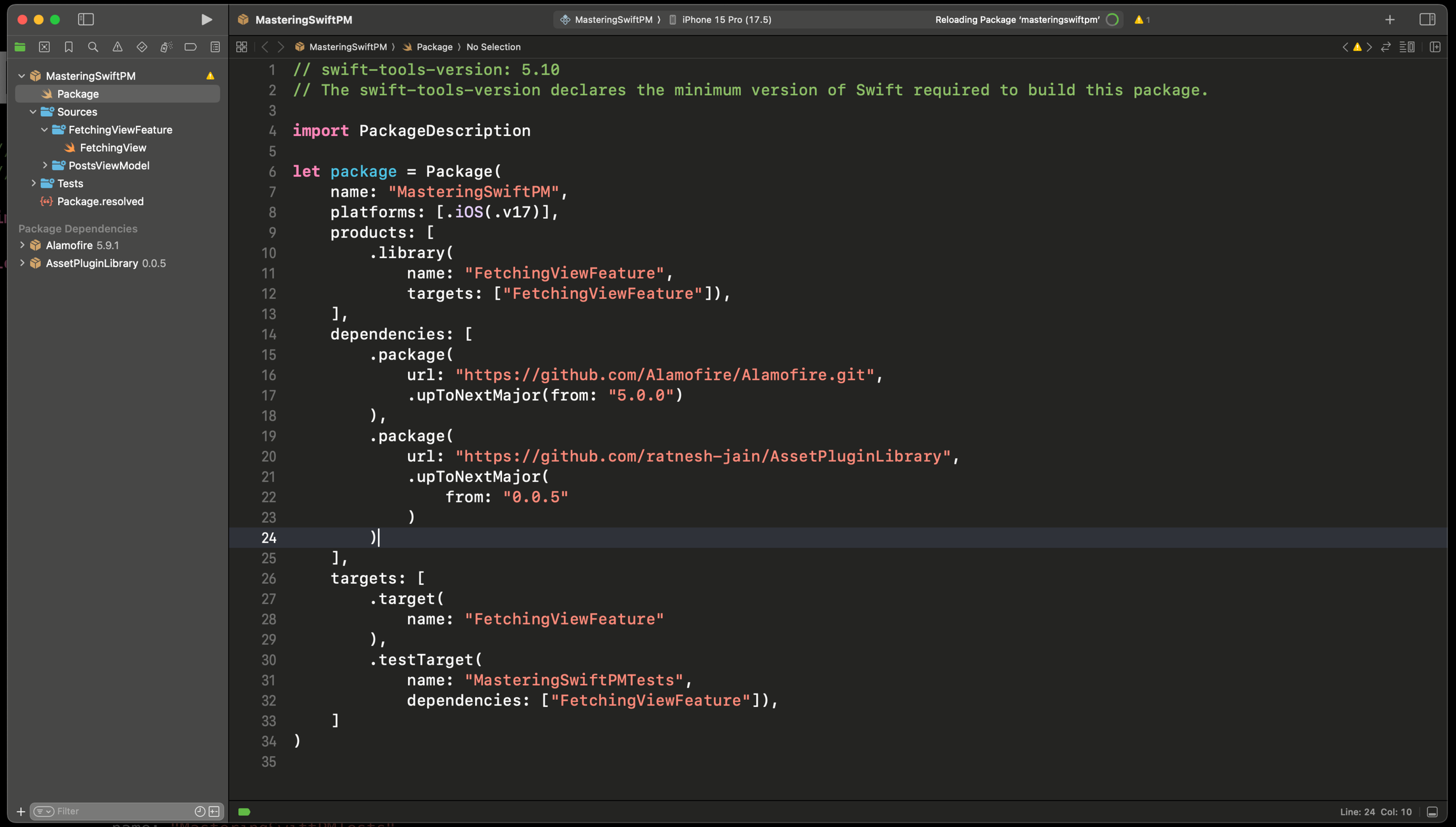
MasteringSwiftPM 〉 iPhone 15 Pro (17.5)

Reloading Package 'masteringswiftpm'

⚠ 1

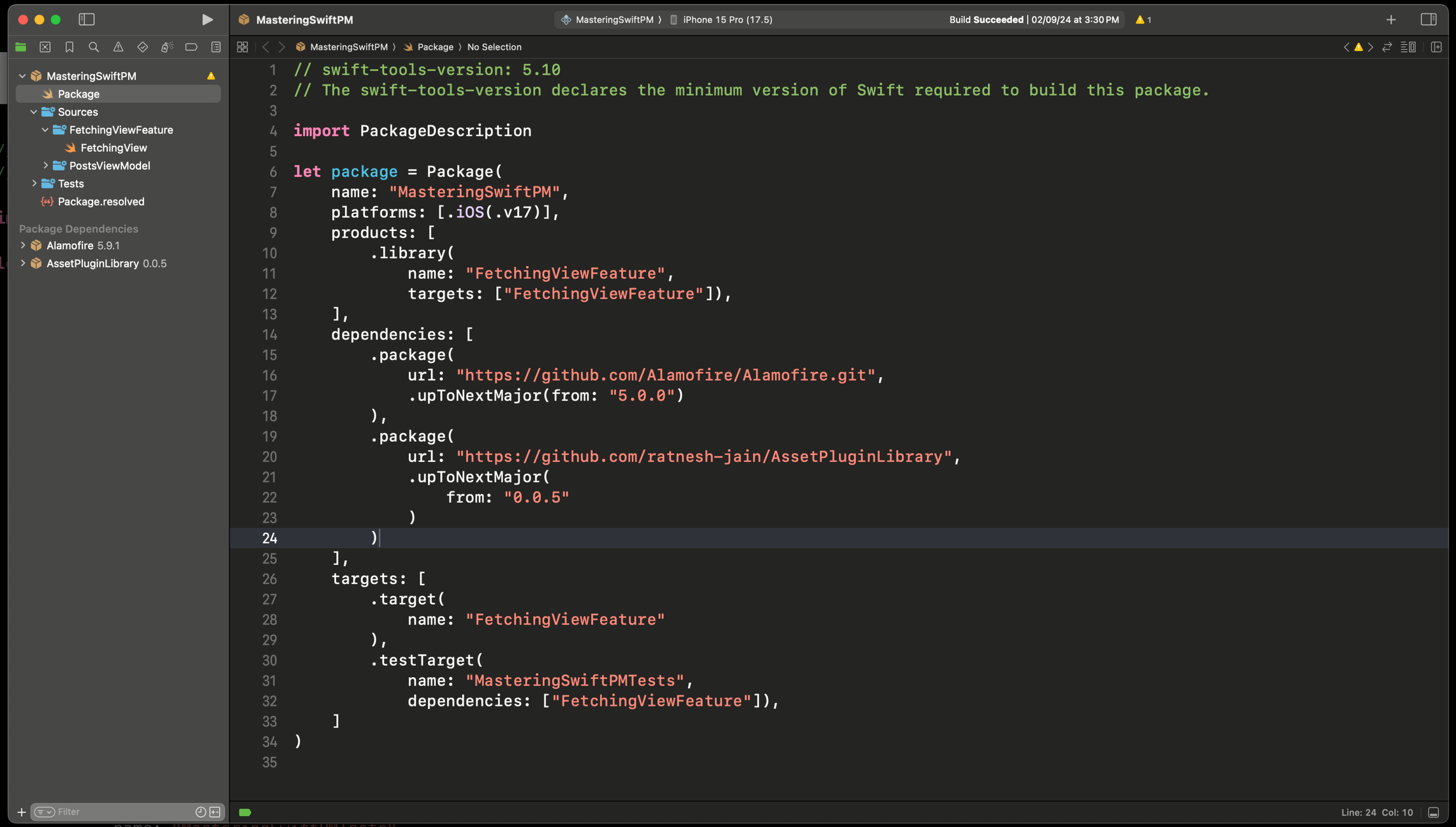
on

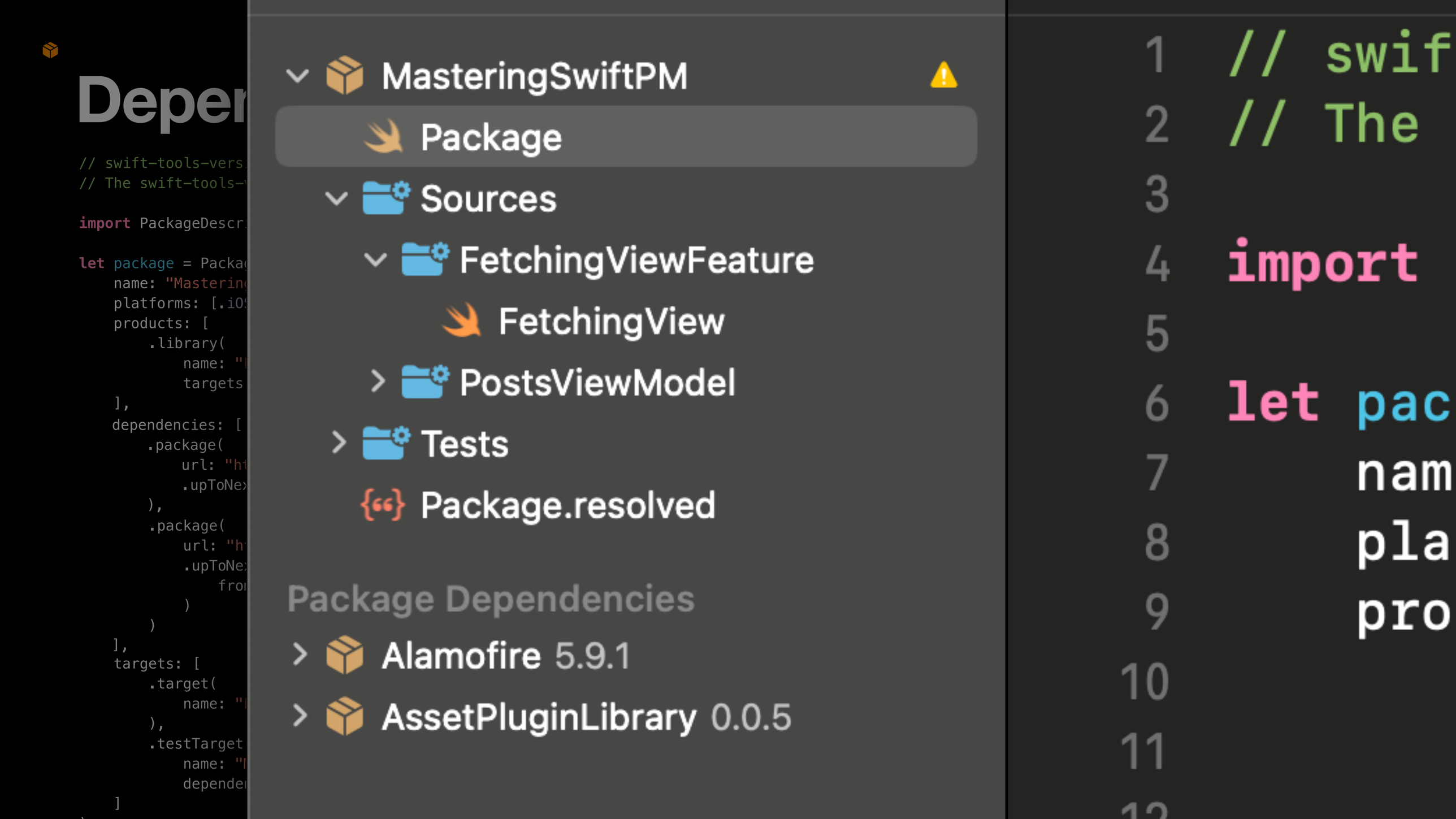
```
 5.10
ion declares the minimum version of Swift required to build this pac

on


ftPM",
17)],
```

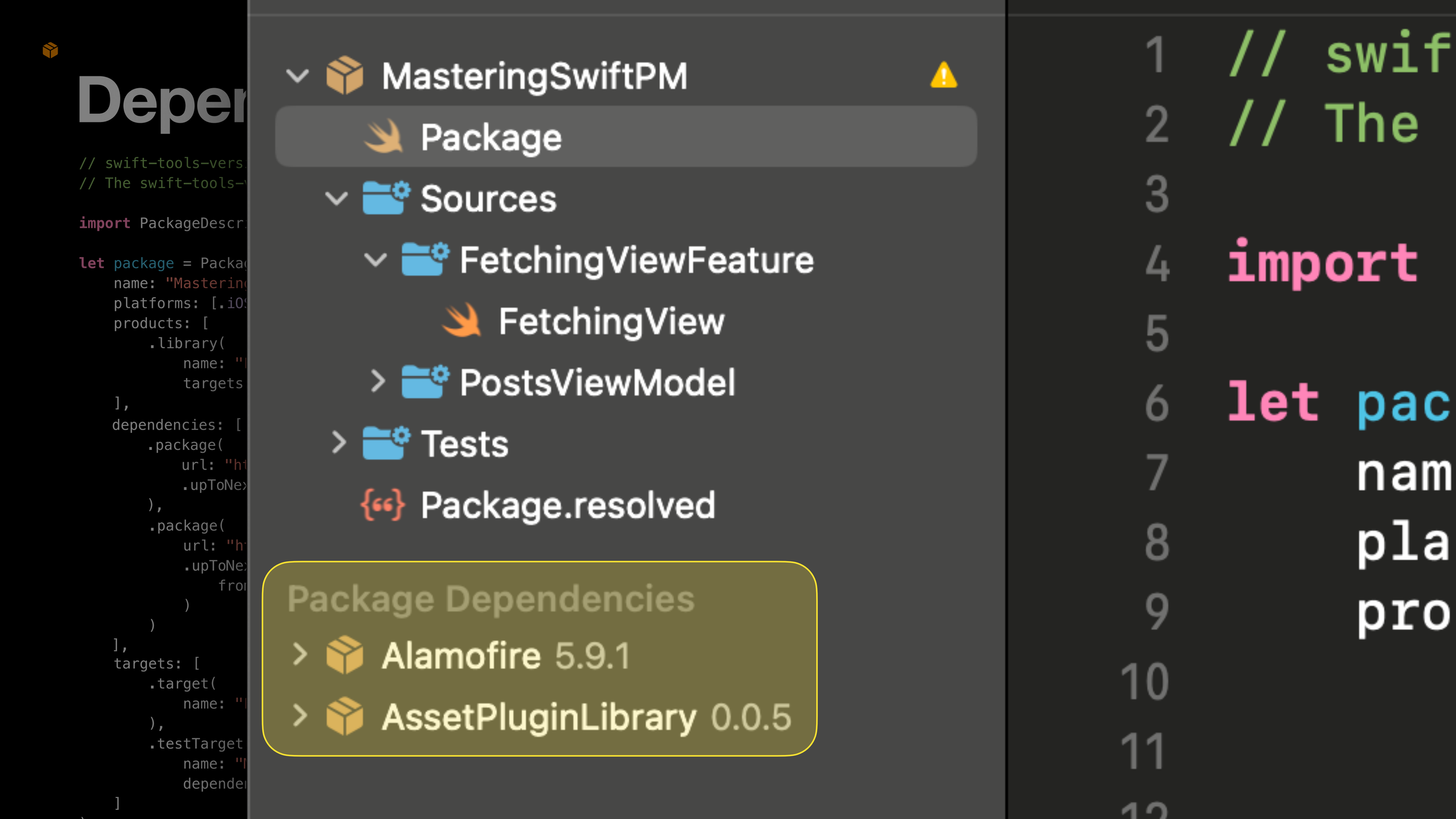


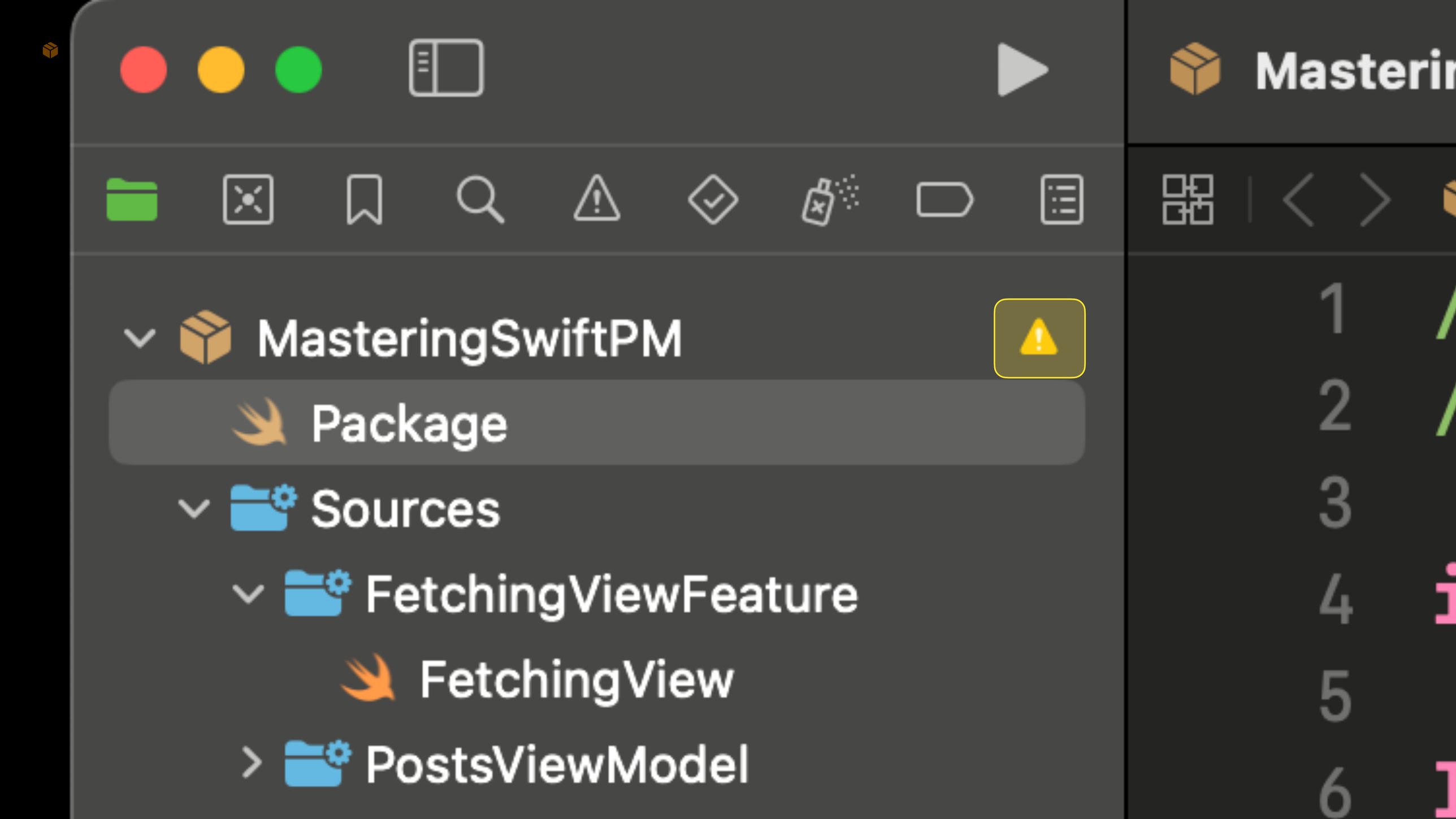
name: "MasteringSwiftPMTests",
dependencies: ["FetchingViewFeature"]),
]

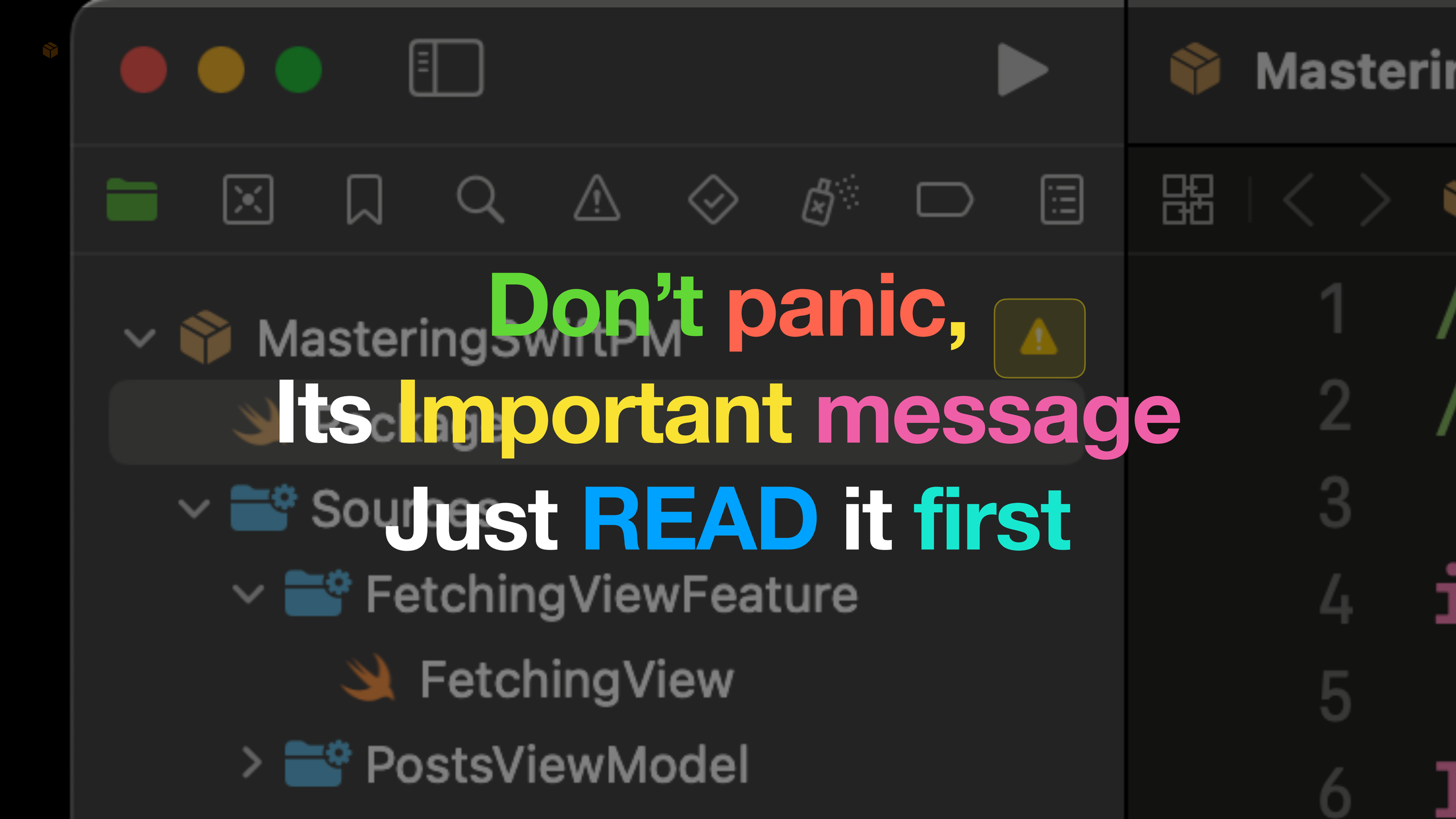
Ratnesh Jain

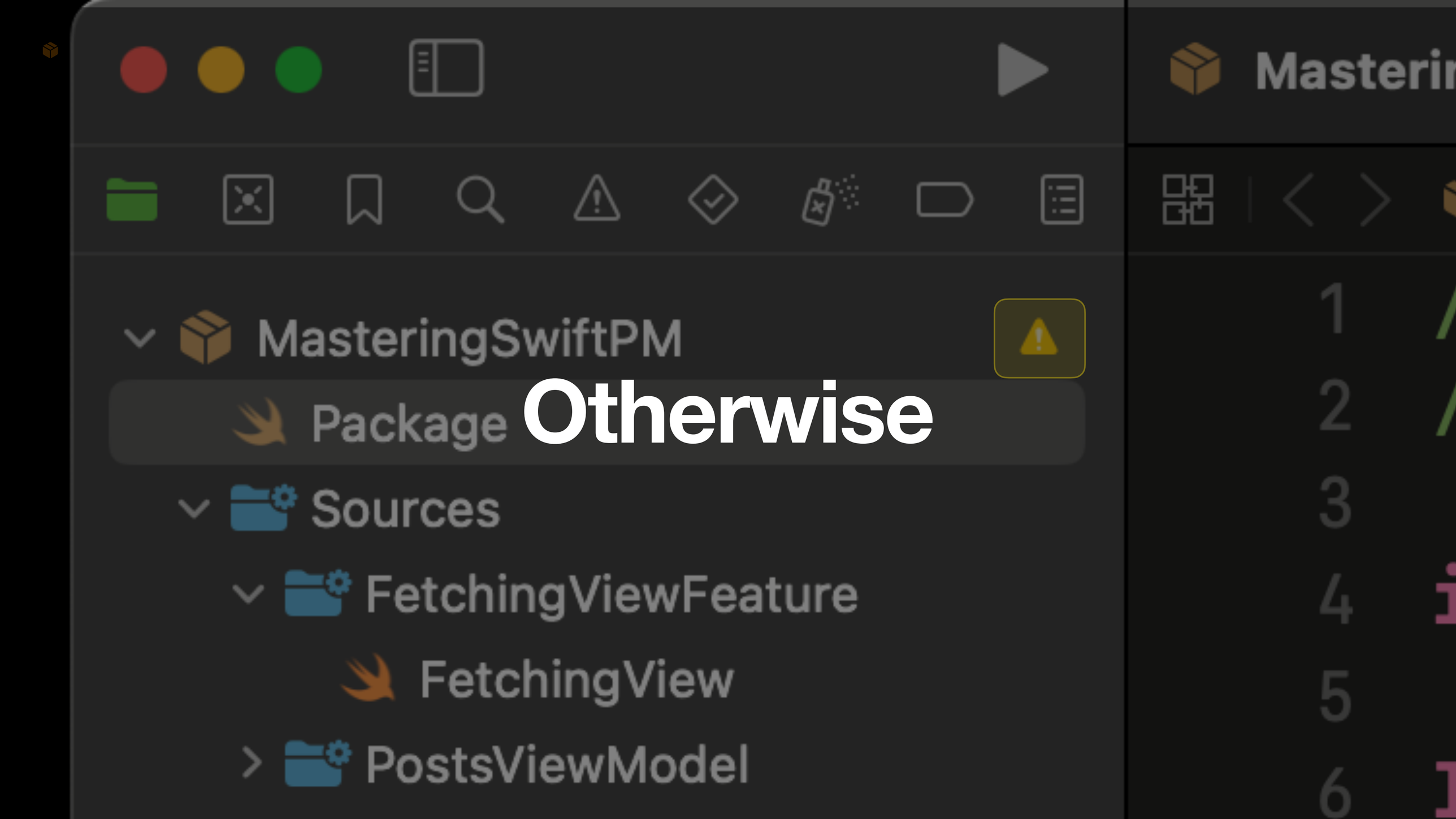



Deper
// swift-tools-vers
// The swift-tools-
import PackageDescr
let package = Packa
name: "Masterin
platforms: [.iO
products: [
.library(
name: "
targets
],
dependencies: [
.package(
url: "h
.upToNe
),
.package(
url: "h
.upToNe
fro
)
)
],
targets: [
.target(
name: "
),
.testTarget
name: "
depende
]
MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Tests
Package.resolved
Package Dependencies
Alamofire 5.9.1
AssetPluginLibrary 0.0.5
1 // swif
2 // The
3
4 import
5
6 let pac
7 nam
8 pla
9 pro
10
11


Deper
// swift-tools-vers
// The swift-tools-
import PackageDescr
let package = Packa
name: "Masterin
platforms: [.iO
products: [
.library(
name: "
targets
],
dependencies: [
.package(
url: "h
.upToNe
),
.package(
url: "h
.upToNe
fro
)
)
],
targets: [
.target(
name: "
),
.testTarget
name: "
depende
]
MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Tests
Package.resolved
Package Dependencies
Alamofire 5.9.1
AssetPluginLibrary 0.0.5
1 // swif
2 // The
3
4 import
5
6 let pac
7 nam
8 pla
9 pro
10
11

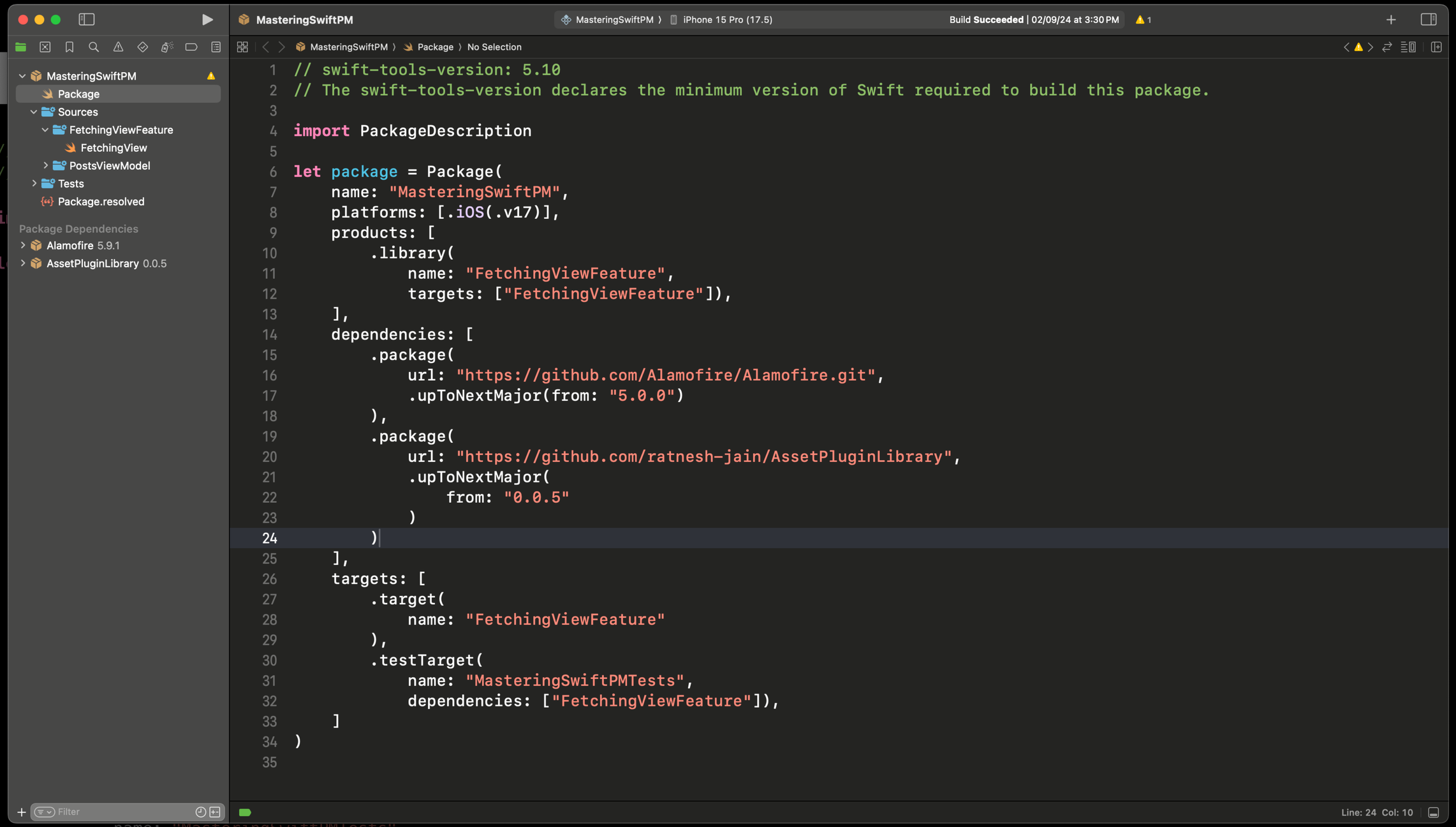

Masterin
MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
1
2
3
4
5
6

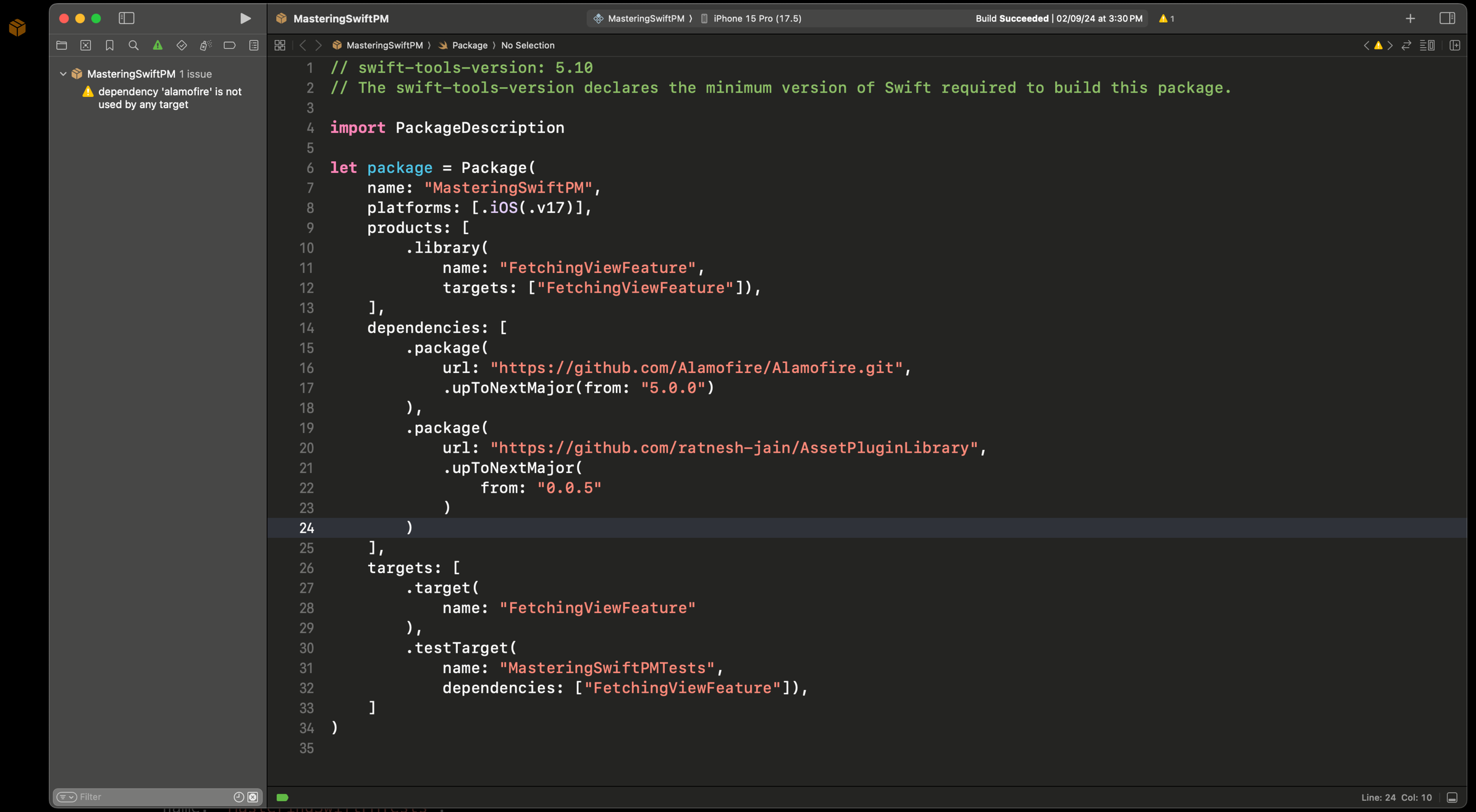

Masterin
MasteringSwiftPM
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Don’t panic,
Its Important message
Just READ it first


MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Otherwise

```swift
// The Swift Programming Language
// https://docs.swift.org/swift-book

import SwiftUI
import Alamofire

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading")
        }
    }
}

```

No such module 'Alamofire'



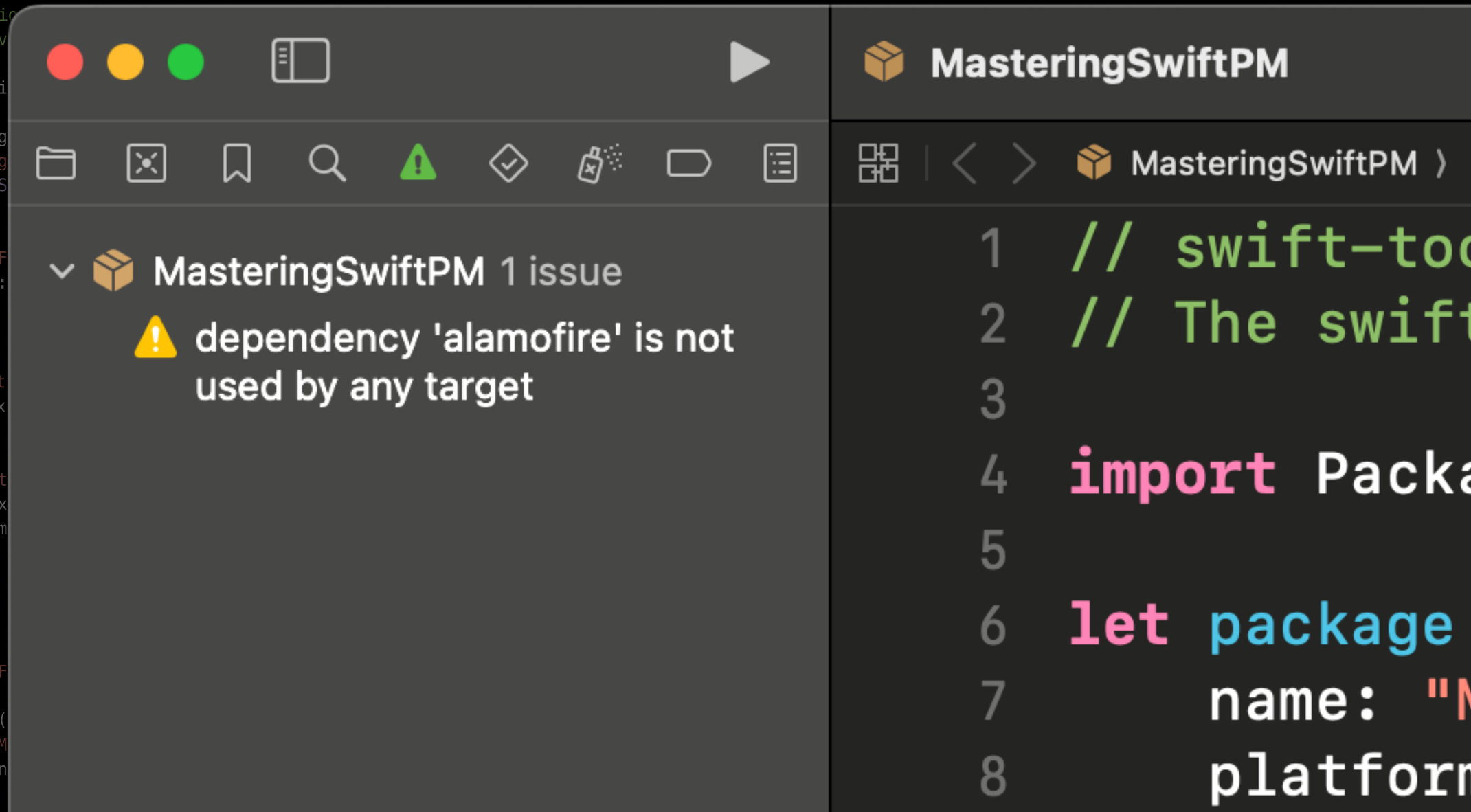


dependencies: ["FetchingViewFeature"]),

]

Ratnesh Jain

Dependencies



Dependencies

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

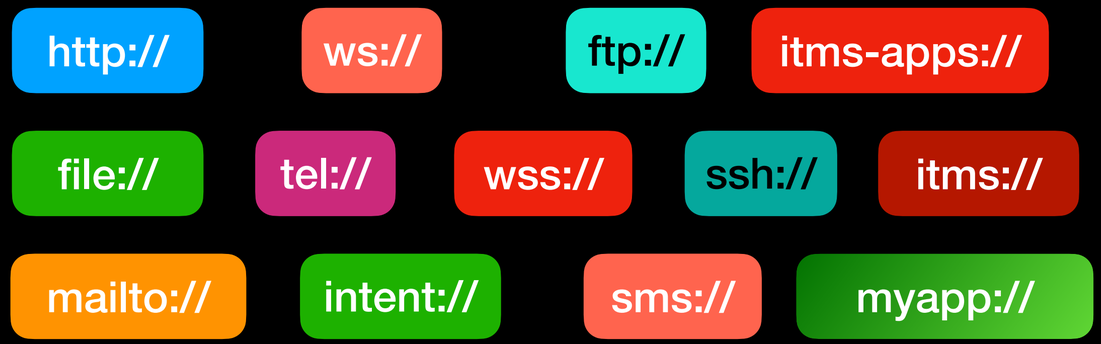
```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

URL Component

`https://github.com/Alamofire/Alamofire.git`

URL Scheme



URL Component

`https://github.com/Alamofire/Alamofire.git`

Host

- github.com
- gitlab.com
- bitbucket.com
- Self hosted

URL Component

```
https://github.com/Alamofire/Alamofire.git
```

Author/Organisation

```
https://github.com/ratnesh-jain/AssetPluginLibrary
https://github.com/hackiftekhar/IQKeyboardManager
https://github.com/SDWebImage/SDWebImageSwiftUI
https://github.com/pointfreeco/swift-composable-architecture
```

URL Component

https://github.com/Alamofire/Alamofire.git

Repository Name

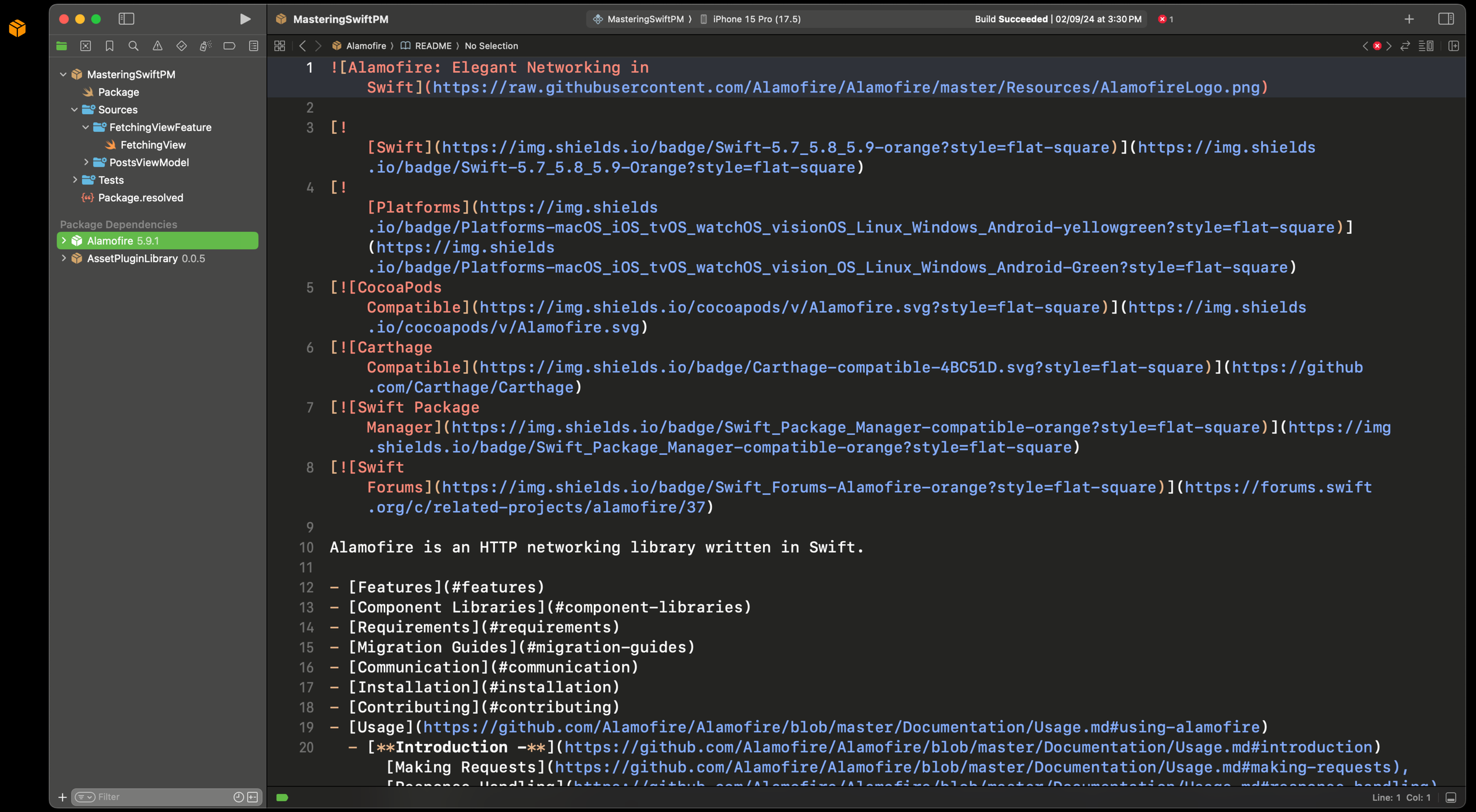
https://github.com/ratnesh-jain/AssetPluginLibrary

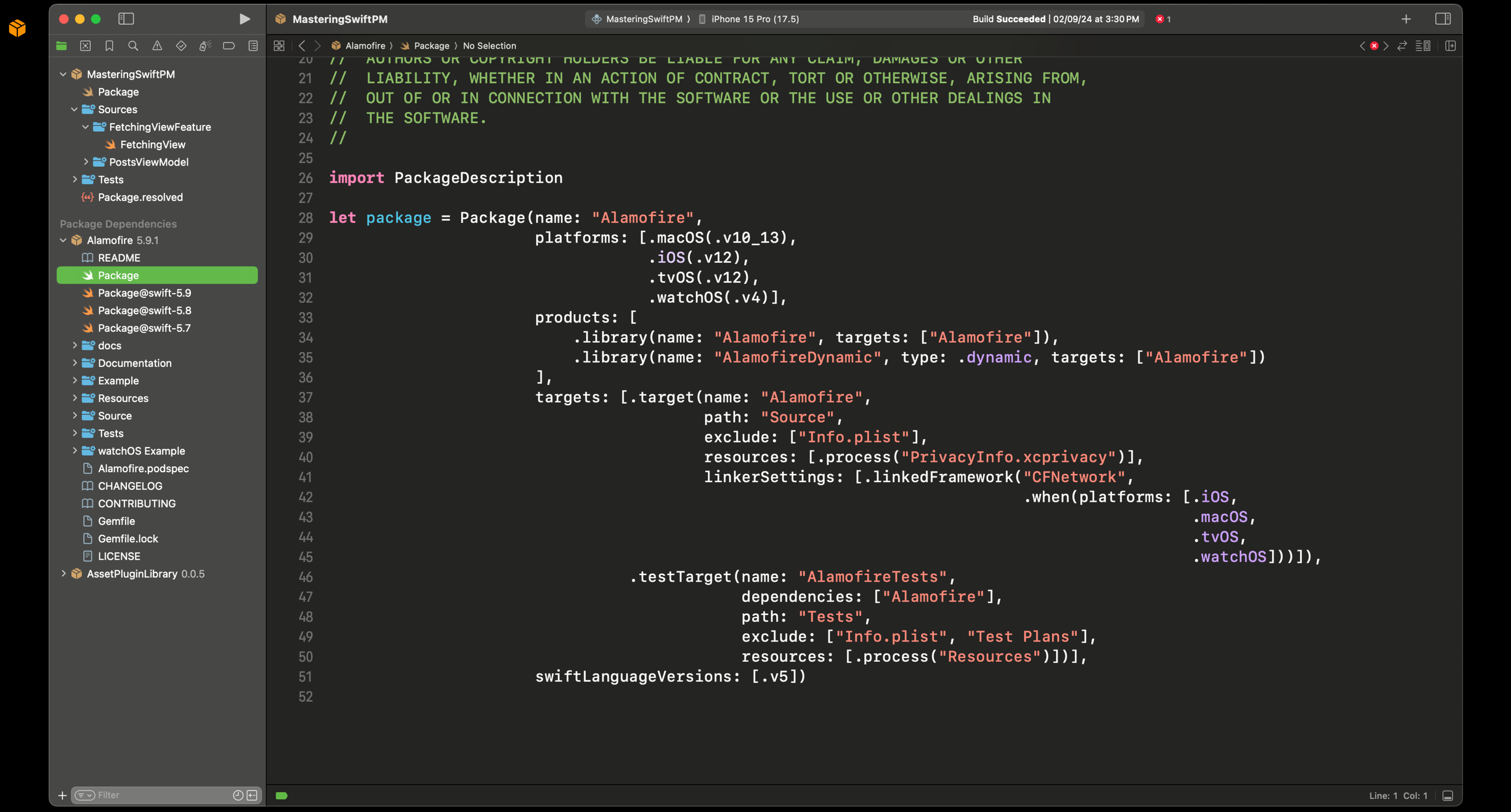
https://github.com/hackiftekhar/IQKeyboardManager

https://github.com/SDWebImage/SDWebImageSwiftUI

https://github.com/pointfreeco/swift-composable-architecture

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```


MasteringSwiftPM
MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Tests
Package.resolved
Package Dependencies
Alamofire 5.9.1
AssetPluginLibrary 0.0.5
MasteringSwiftPM 〉 iPhone 15 Pro (17.5)
Build Succeeded | 02/09/24 at 3:30 PM
Alamofire 〉 README 〉 No Selection
1 ![Alamofire: Elegant Networking in Swift](https://raw.githubusercontent.com/Alamofire/Alamofire/master/Resources/AlamofireLogo.png)
2
3 [! [Swift](https://img.shields.io/badge/Swift-5.7_5.8_5.9-orange?style=flat-square)](https://img.shields.io/badge/Swift-5.7_5.8_5.9-Orange?style=flat-square)
4 [! [Platforms](https://img.shields.io/badge/Platforms-macOS_iOS_tvOS_watchOS_visionOS_Linux_Windows_Android-yellowgreen?style=flat-square)](https://img.shields.io/badge/Platforms-macOS_iOS_tvOS_watchOS_vision_OS_Linux_Windows_Android-Green?style=flat-square)
5 [![CocoaPods Compatible](https://img.shields.io/cocoapods/v/Alamofire.svg?style=flat-square)](https://img.shields.io/cocoapods/v/Alamofire.svg)
6 [![Carthage Compatible](https://img.shields.io/badge/Carthage-compatible-4BC51D.svg?style=flat-square)](https://github.com/Carthage/Carthage)
7 [![Swift Package Manager](https://img.shields.io/badge/Swift_Package_Manager-compatible-orange?style=flat-square)](https://img.shields.io/badge/Swift_Package_Manager-compatible-orange?style=flat-square)
8 [![Swift Forums](https://img.shields.io/badge/Swift_Forums-Alamofire-orange?style=flat-square)](https://forums.swift.org/c/related-projects/alamofire/37)
9
10 Alamofire is an HTTP networking library written in Swift.
11
12 - [Features](#features)
13 - [Component Libraries](#component-libraries)
14 - [Requirements](#requirements)
15 - [Migration Guides](#migration-guides)
16 - [Communication](#communication)
17 - [Installation](#installation)
18 - [Contributing](#contributing)
19 - [Usage](https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#using-alamofire)
20 - [**Introduction -**](https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#introduction) [Making Requests](https://github.com/Alamofire/Alamofire/blob/master/Documentation/Usage.md#making-requests),
Filter
Line: 1 Col: 1


MasteringSwiftPM
MasteringSwiftPM 〉 iPhone 15 Pro (17.5)
Build Succeeded | 02/09/24 at 3:30 PM
Alamofire 〉 Package 〉 No Selection
MasteringSwiftPM
Package
Sources
FetchingViewFeature
FetchingView
PostsViewModel
Tests
Package.resolved
Package Dependencies
Alamofire 5.9.1
README
Package
Package@swift-5.9
Package@swift-5.8
Package@swift-5.7
docs
Documentation
Example
Resources
Source
Tests
watchOS Example
Alamofire.podspec
CHANGELOG
CONTRIBUTING
Gemfile
Gemfile.lock
LICENSE
AssetPluginLibrary 0.0.5
Filter
// AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
// LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
// OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
// THE SOFTWARE.
//
import PackageDescription
let package = Package(name: "Alamofire",
platforms: [.macOS(.v10_13),
.iOS(.v12),
.tvOS(.v12),
.watchOS(.v4)],
products: [
.library(name: "Alamofire", targets: ["Alamofire"]),
.library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamofire"])
],
targets: [.target(name: "Alamofire",
path: "Source",
exclude: ["Info.plist"],
resources: [.process("PrivacyInfo.xcprivacy")],
linkerSettings: [.linkedFramework("CFNetwork",
.when(platforms: [.iOS,
.macOS,
.tvOS,
.watchOS]))]),
.testTarget(name: "AlamofireTests",
dependencies: ["Alamofire"],
path: "Tests",
exclude: ["Info.plist", "Test Plans"],
resources: [.process("Resources")])],
swiftLanguageVersions: [.v5])
Line: 1 Col: 1

```
20  //  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21  //  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
22  //  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23  //  THE SOFTWARE.
24  //
25
26  import PackageDescription
27
28  let package = Package(name: "Alamofire",
29                        platforms: [.macOS(.v10_13),
30                                    .iOS(.v12),
31                                    .tvOS(.v12),
32                                    .watchOS(.v4)],
33                        products: [
34                            .library(name: "Alamofire", targets: ["Alamofire"]),
35                            .library(name: "AlamofireDynamic", type: .dynamic, tar
36                        ],
37                        targets: [.target(name: "Alamofire",
38                                          path: "Source",
39                                          exclude: ["Info.plist"],
40                                          resources: [.process("PrivacyInfo.xcpriv
```

```
20  //  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21  //  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
22  //  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23  //  THE SOFTWARE.
24  //
25
26  import PackageDescription
27
28  let package = Package(name: "Alamofire",
29                        platforms: [.macOS(.v10_13),
30                                    .iOS(.v12),
31                                    .tvOS(.v12),
32                                    .watchOS(.v4)],
33                        products: [
34                            .library(name: "Alamofire", targets: ["Alamofire"]),
35                            .library(name: "AlamofireDynamic", type: .dynamic, tar
36                        ],
37                        targets: [.target(name: "Alamofire",
38                                          path: "Source",
39                                          exclude: ["Info.plist"],
40                                          resources: [.process("PrivacyInfo.xcpriv
```

```
20  //  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21  //  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM
22  //  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
23  //  THE SOFTWARE.
24  //
25
26  import PackageDescription
27
28  let package = Package(name: "Alamofire",
29                        platforms: [.macOS(.v10_13),
30                                    .iOS(.v12),
31                                    .tvOS(.v12),
32                                    .watchOS(.v4)],
33                        products: [
34                            .library(name: "Alamofire", targets: ["Alamofire"]),
35                            .library(name: "AlamofireDynamic", type: .dynamic, tar
36                        ],
37                        targets: [.target(name: "Alamofire",
38                                          path: "Source",
39                                          exclude: ["Info.plist"],
40                                          resources: [.process("PrivacyInfo.xcpriv
```

```
T OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
E SOFTWARE.


 PackageDescription

ckage = Package(name: "Alamofire",
                platforms: [.macOS(.v10_13),
                            .iOS(.v12),
                            .tvOS(.v12),
                            .watchOS(.v4)],
                products: [
                    .library(name: "Alamofire", targets: ["Alamofire"]),
                    .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamo
                ],
                targets: [.target(name: "Alamofire",
                                  path: "Source",
                                  exclude: ["Info.plist"],
                                  resources: [.process("PrivacyInfo.xcprivacy")],
                                  linkerSettings: [.linkedFramework("CFNetwork",
                                                                    .when(platforms: [.
                                                                          .
                                                                          .
```

```
T OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
E SOFTWARE.


 PackageDescription

ckage = Package(name: "Alamofire",
                platforms: [.macOS(.v10_13),
                            .iOS(.v12),
                            .tvOS(.v12),
                            .watchOS(.v4)],
                products: [
                    .library(name: "Alamofire", targets: ["Alamofire"]),
                    .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamo
                ],
                targets: [.target(name: "Alamofire",
                                  path: "Source",
                                  exclude: ["Info.plist"],
                                  resources: [.process("PrivacyInfo.xcprivacy")],
                                  linkerSettings: [.linkedFramework("CFNetwork",
                                                                    .when(platforms: [.
                                                                          .
                                                                          .
```

```
cription

kage(name: "Alamofire",
     platforms: [.macOS(.v10_13),
                 .iOS(.v12),
                 .tvOS(.v12),
                 .watchOS(.v4)],
     products: [
         .library(name: "Alamofire", targets: ["Alamofire"]),
         .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamofire"])
     ],
     targets: [.target(name: "Alamofire",
                       path: "Source",
                       exclude: ["Info.plist"],
                       resources: [.process("PrivacyInfo.xcprivacy")],
                       linkerSettings: [.linkedFramework("CFNetwork",
                                                         .when(platforms: [.iOS,
                                                                           .macOS,
                                                                           .tvOS,
                                                                           .watchOS]))]
```

```
cription

kage(name: "Alamofire",
     platforms: [.macOS(.v10_13),
                 .iOS(.v12),
                 .tvOS(.v12),
                 .watchOS(.v4)],
     products: [
         .library(name: "Alamofire", targets: ["Alamofire"]),
         .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamofire"])
     ],
     targets: [.target(name: "Alamofire",
                       path: "Source",
                       exclude: ["Info.plist"],
                       resources: [.process("PrivacyInfo.xcprivacy")],
                       linkerSettings: [.linkedFramework("CFNetwork",
                                                         .when(platforms: [.iOS,
                                                                           .macOS,
                                                                           .tvOS,
                                                                           .watchOS]))]
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

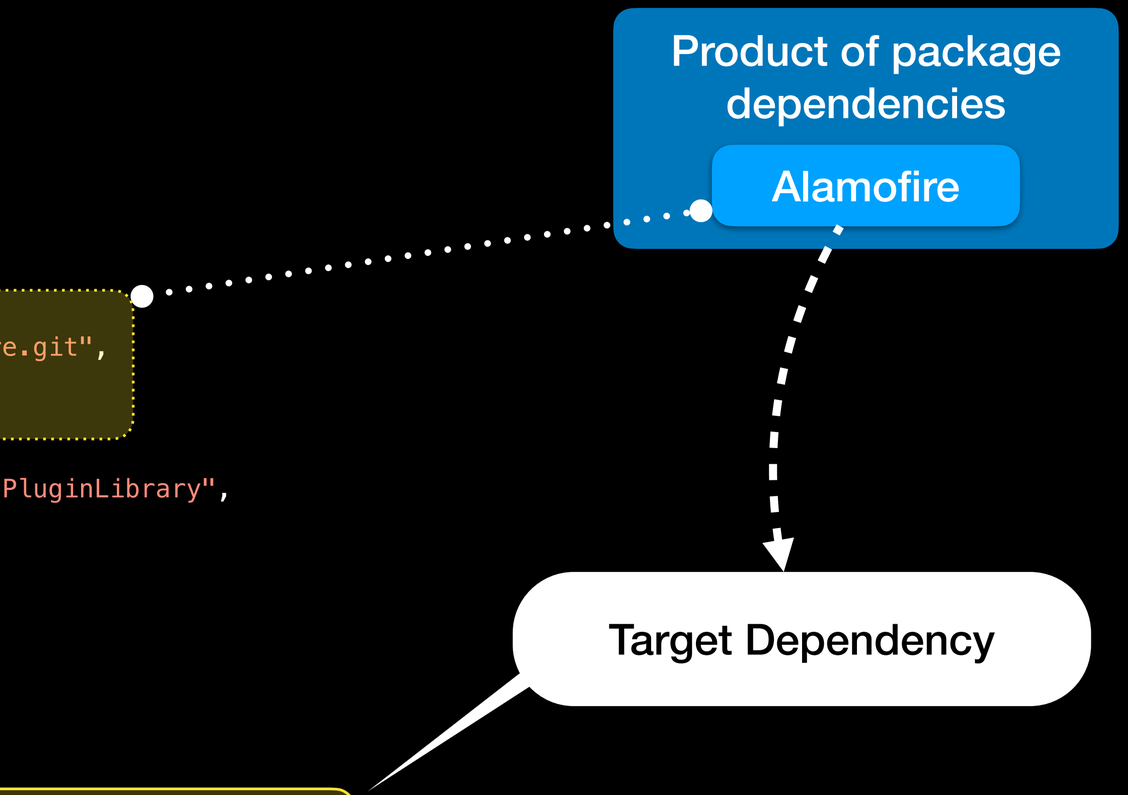
```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Target Dependency

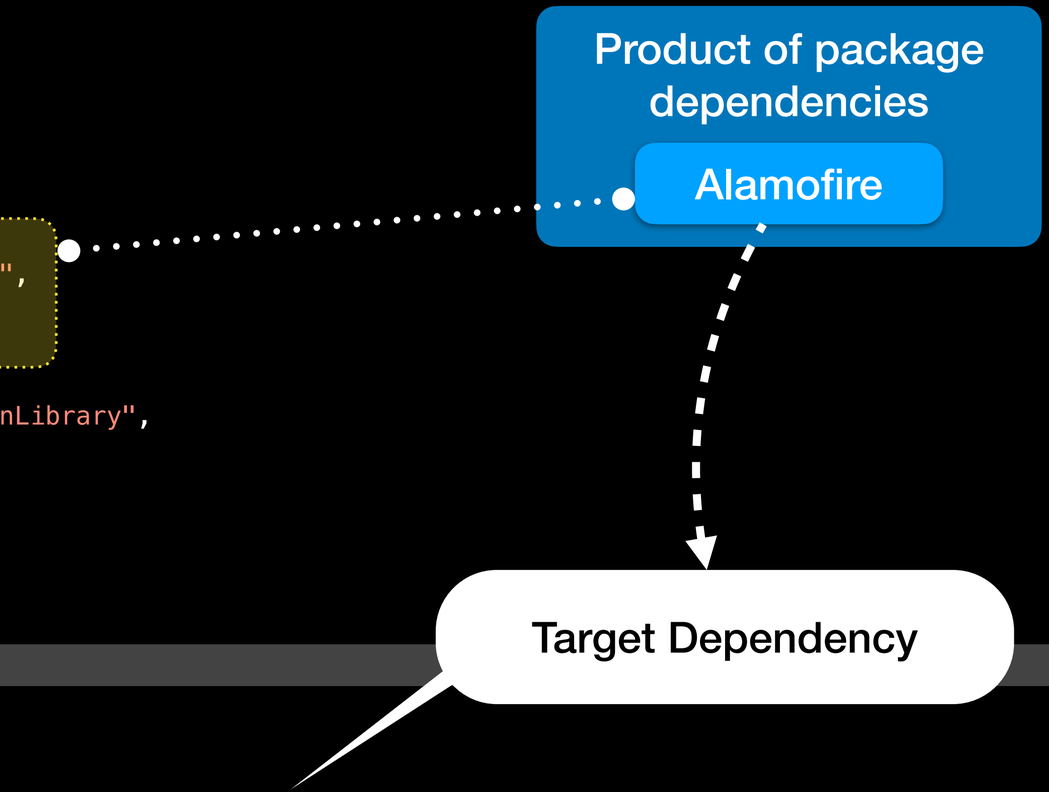
```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Package Dependencies

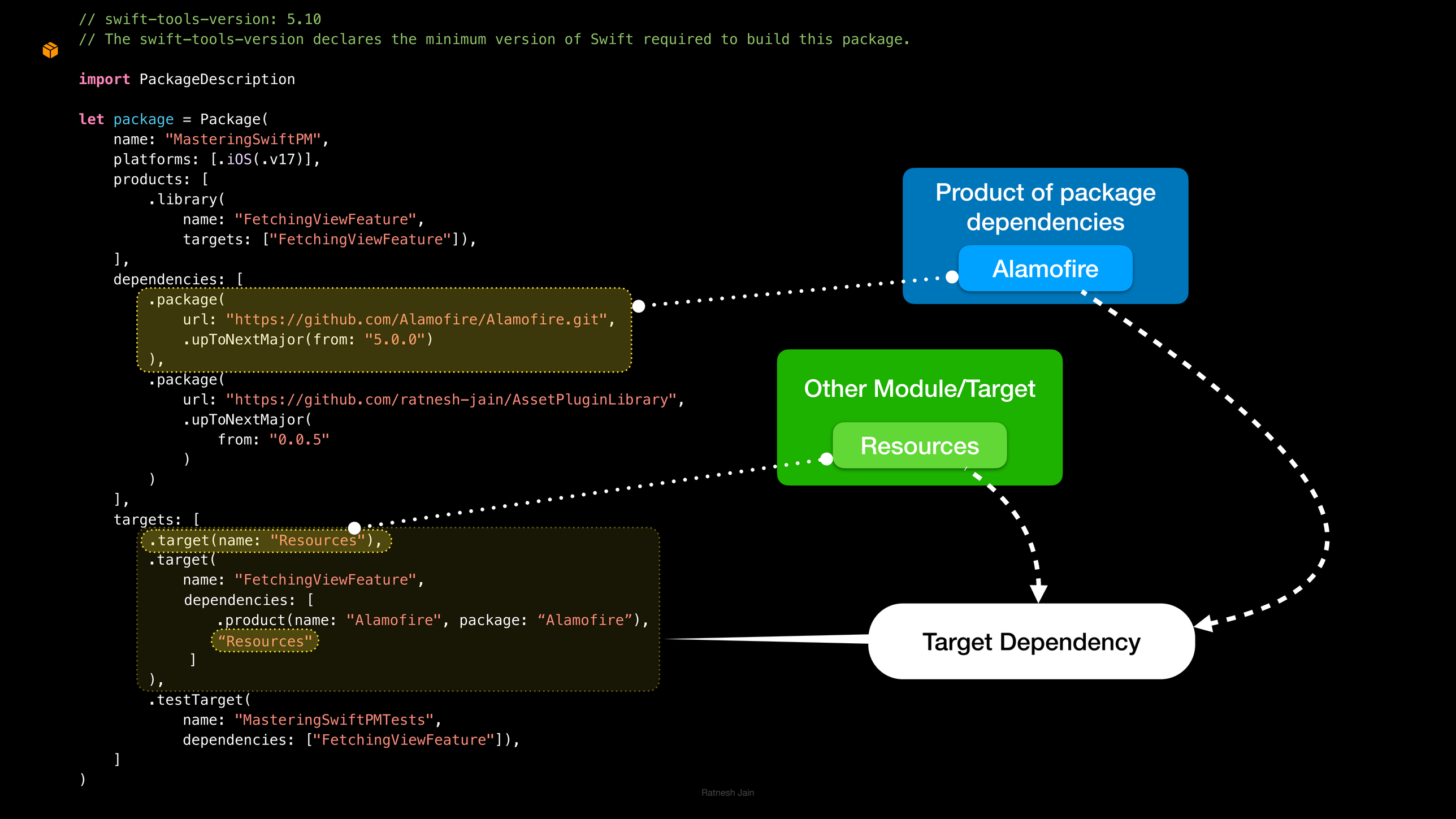
```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```



```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(name: "Resources"),
        .target(
            name: "FetchingViewFeature",
            dependencies: [
                .product(name: "Alamofire", package: “Alamofire”),
                “Resources"
             ]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```



```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(name: "Resources"),
        .target(
            name: "FetchingViewFeature",
            dependencies: [
                .product(name: "Alamofire", package: “Alamofire”),
                “Resources"
             ]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```



```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

```swift
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

Dependencies

Source Code + Dependencies

Source Code + Dependencies

Source Code + Dependencies

Source Code + Dependencies

Target

Module

Source Code + Dependencies

Target

Module

Library

Source Code + Dependencies

↓

Target

↓

Module

↓

Library

↓

Product

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

New Feature using SwiftPM

Mastering SwiftPM

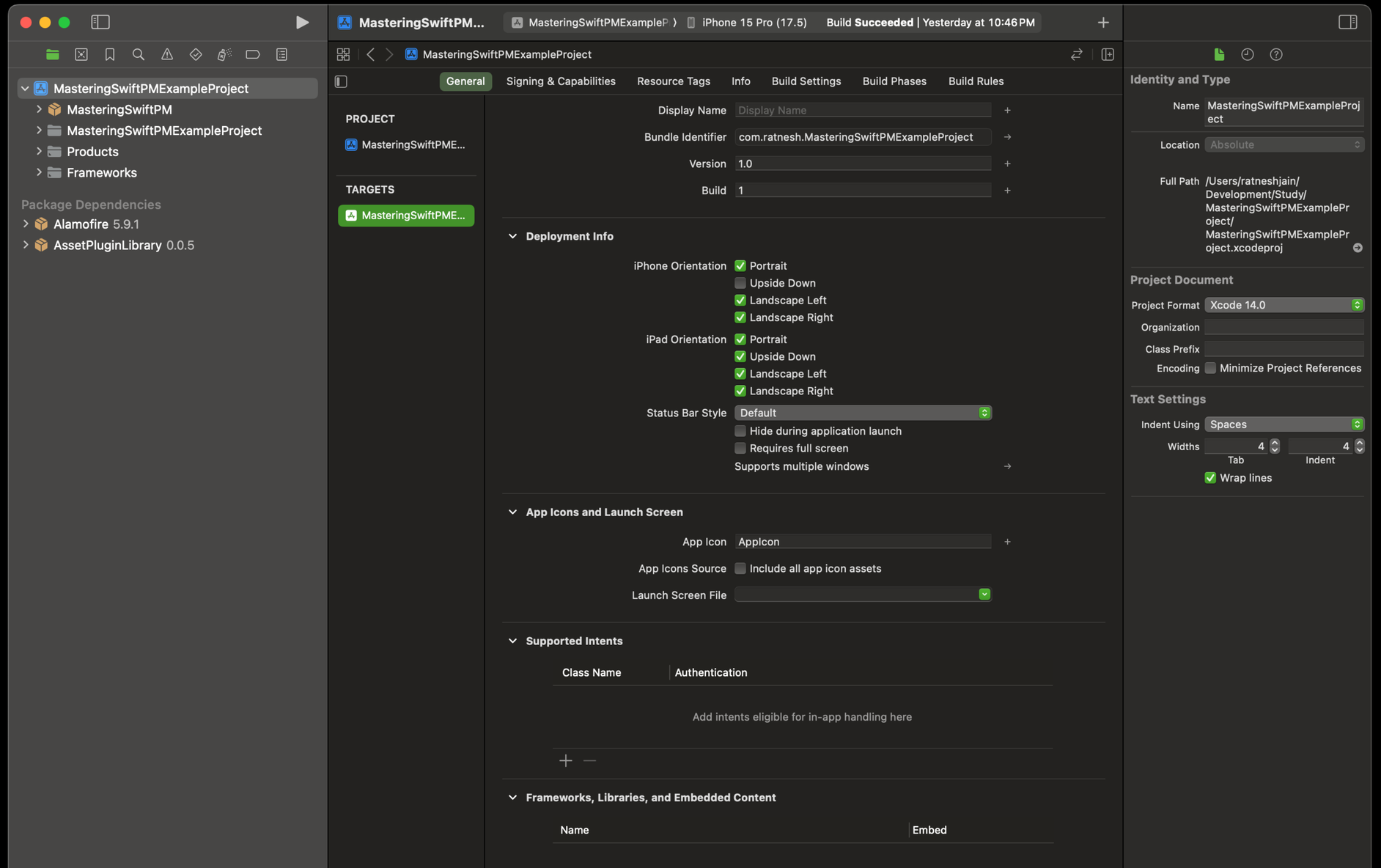
Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

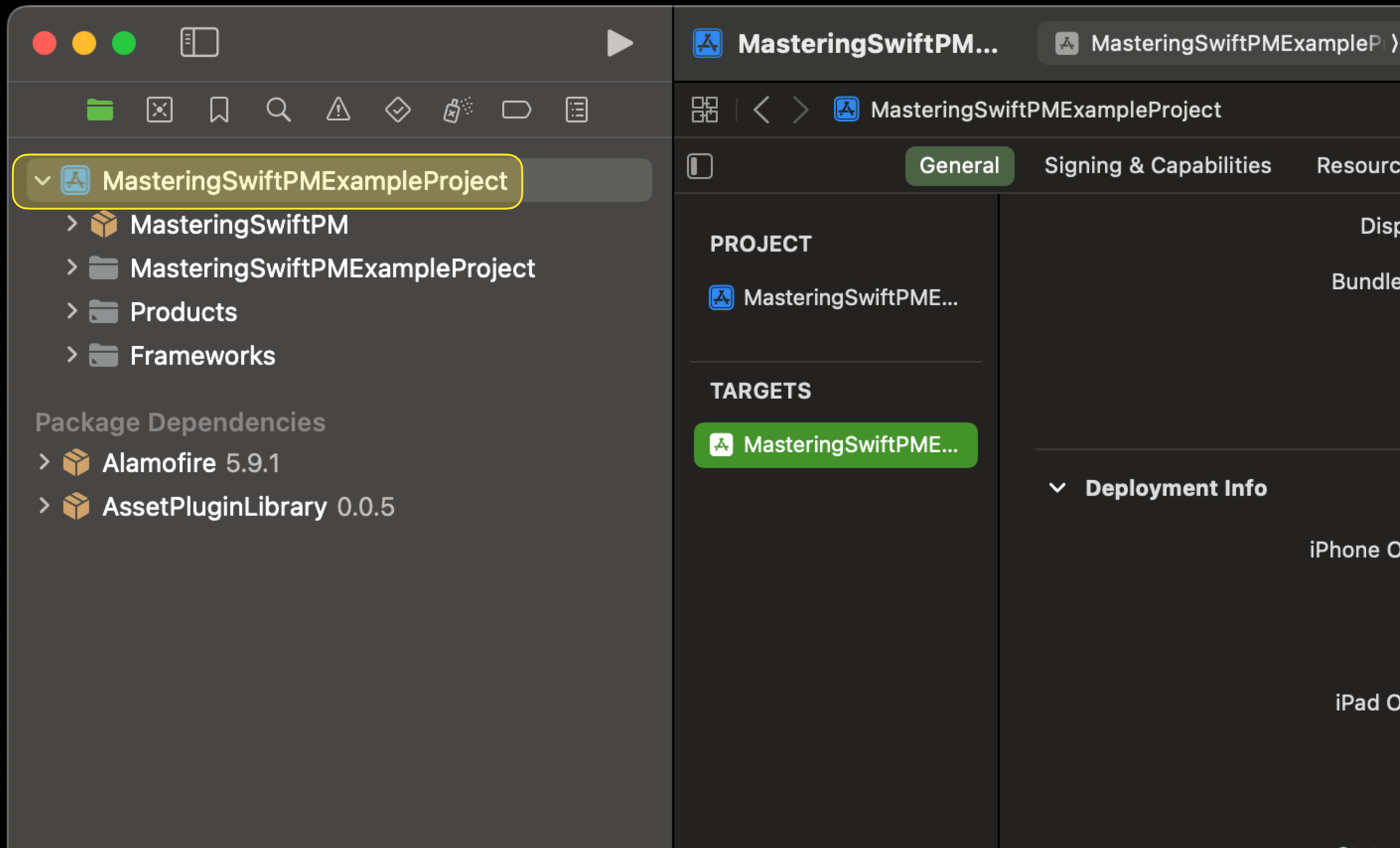
Integration with Xcode Project

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

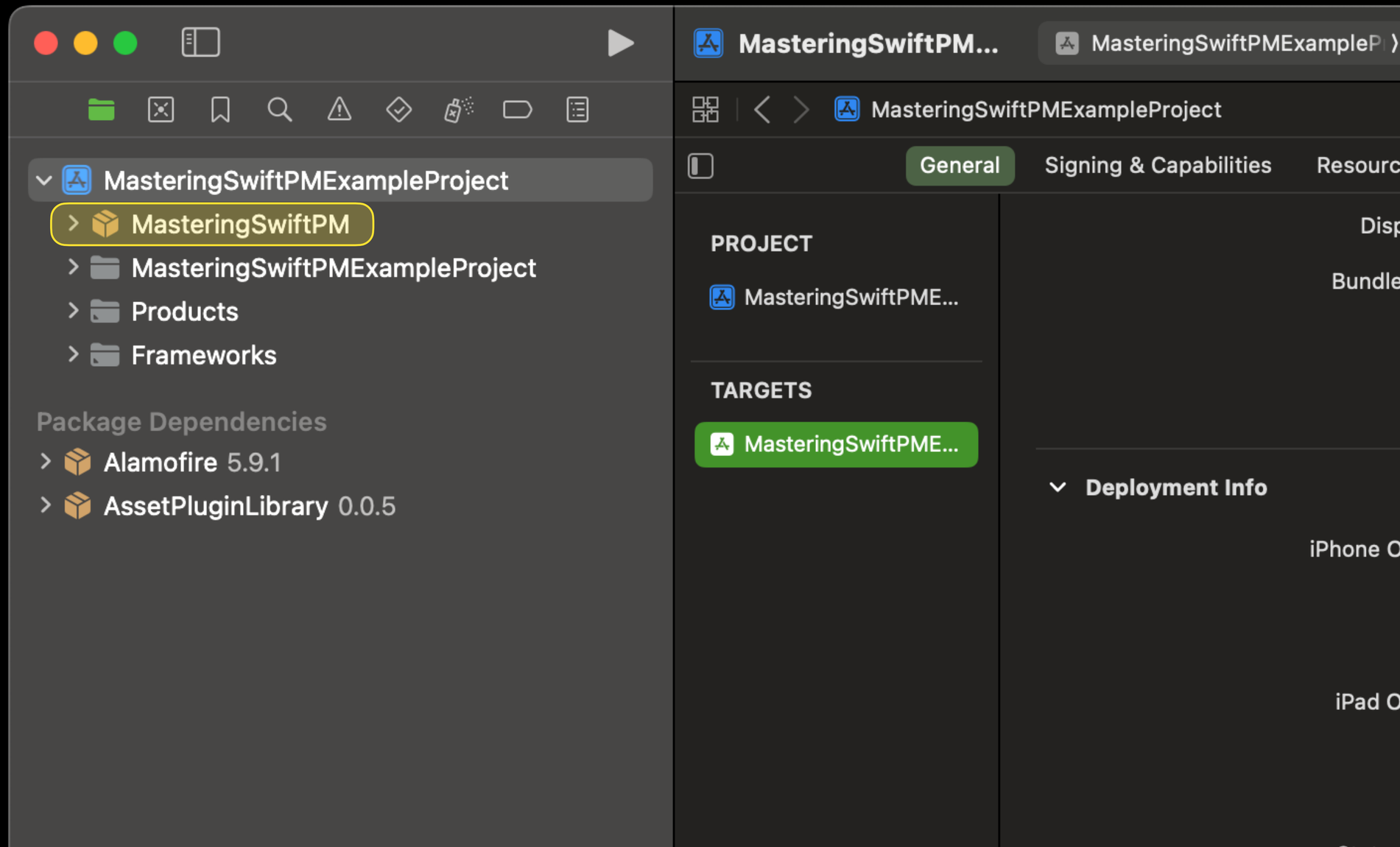
Integration with Xcode Project



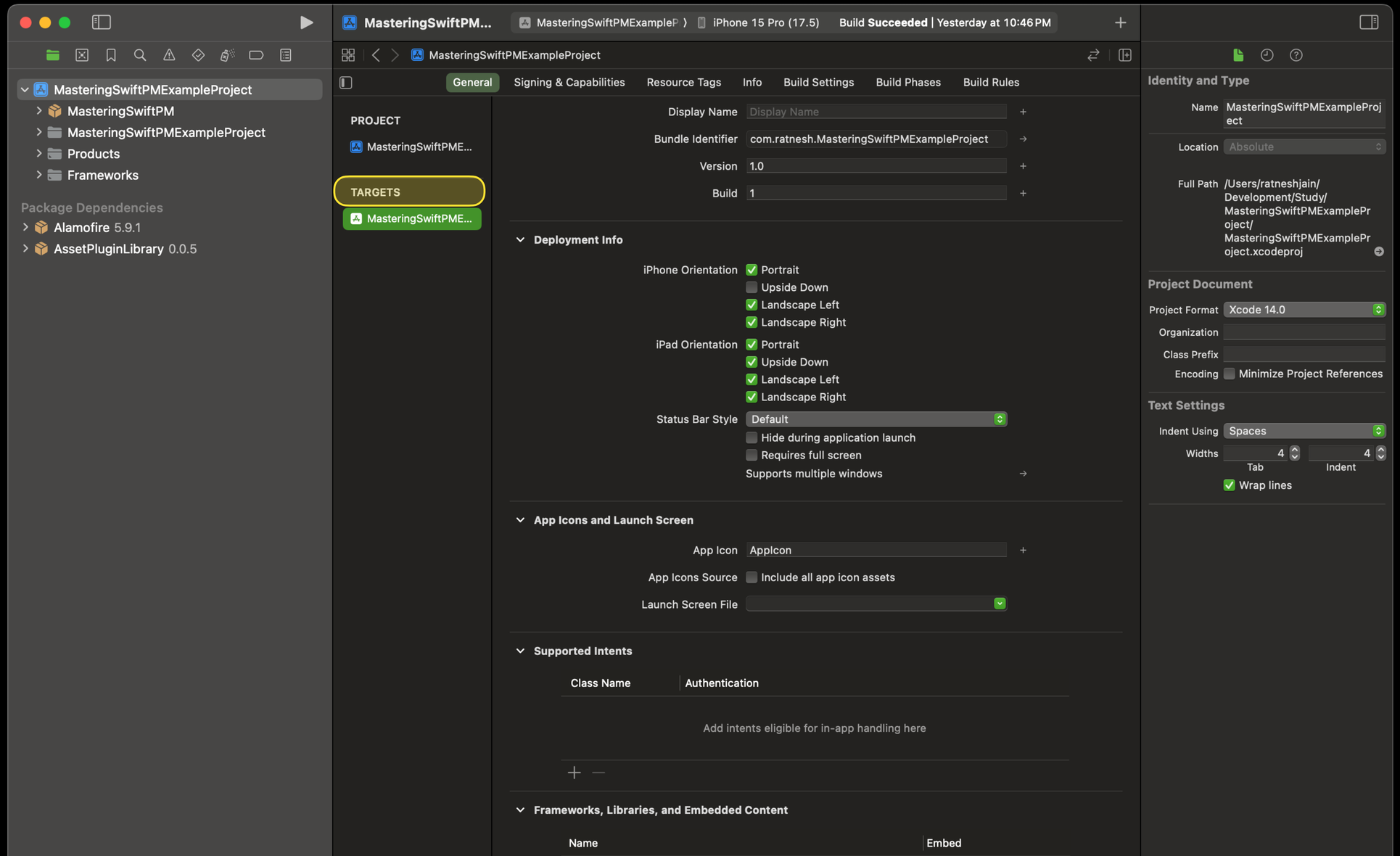
Integration with Xcode Project



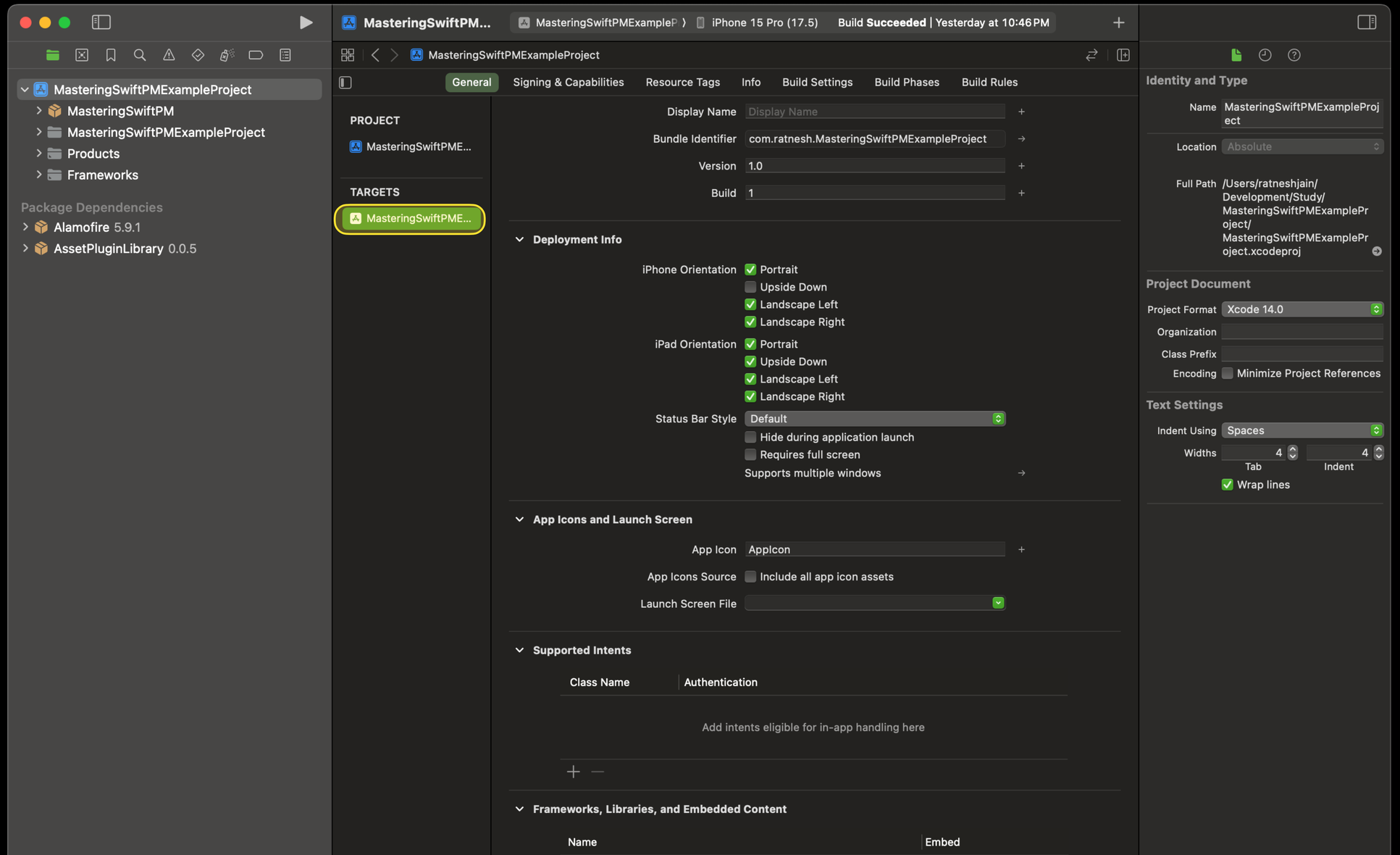
Integration with Xcode Project



Integration with Xcode Project



Integration with Xcode Project



App Icons and Launch Screen

App Icon: AppIcon

App Icons Source: Include all app icon assets

Launch Screen File:

Supported Intents

Class Name	Authentication

Add intents eligible for in-app handling here

Frameworks, Libraries, and Embedded Content

Name	Embed

Add frameworks, libraries, and embedded content here

Development Assets

MasteringSwiftPMExampleProject/Preview Content

Filter

App Icons and Launch Screen

App Icon: AppIcon

App Icons Source: Include all app icon assets

Launch Screen File:

Supported Intents

Class Name	Authentication

Add intents eligible for in-app handling here

Frameworks, Libraries, and Embedded Content

Name	Embed

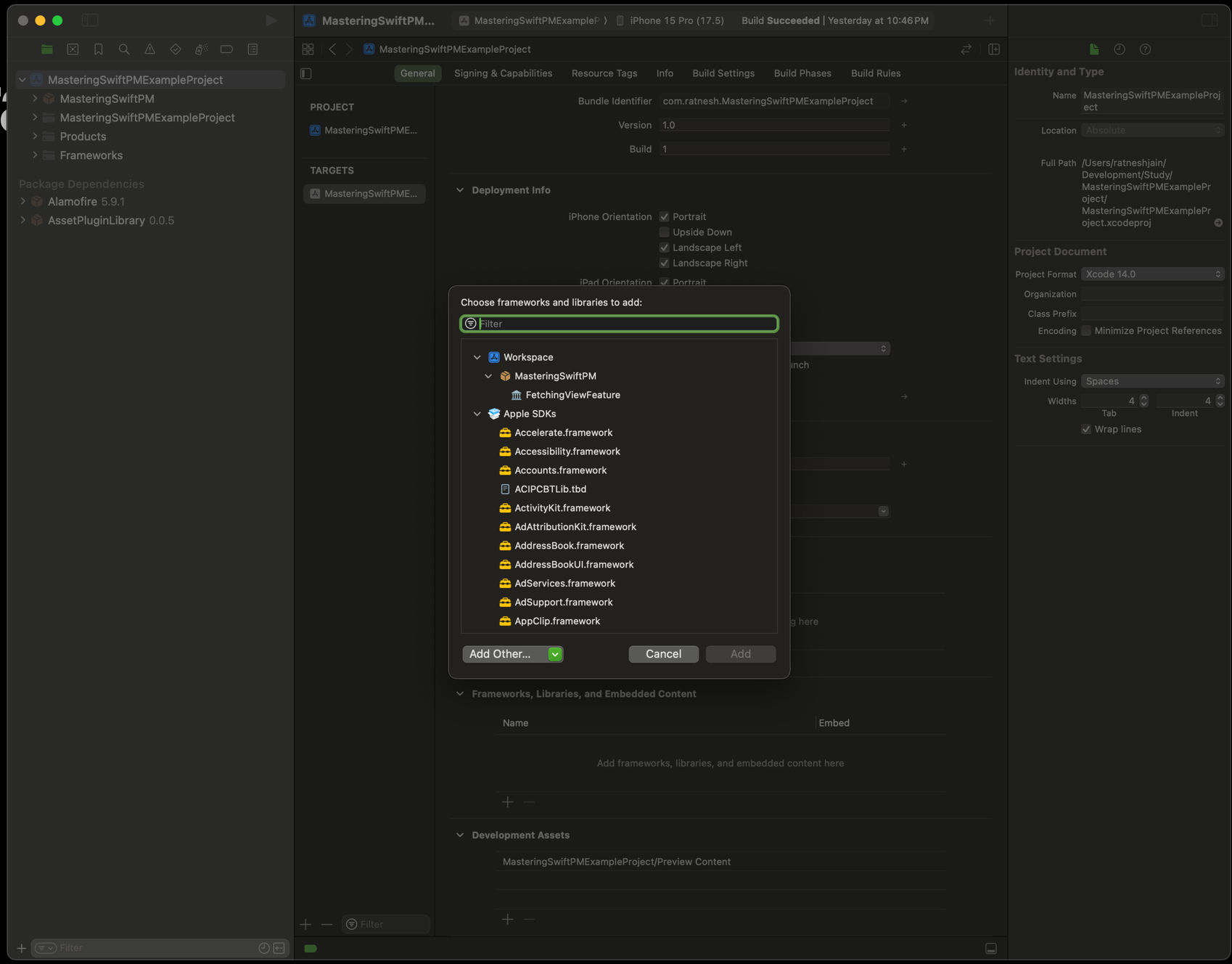
Add frameworks, libraries, and embedded content here

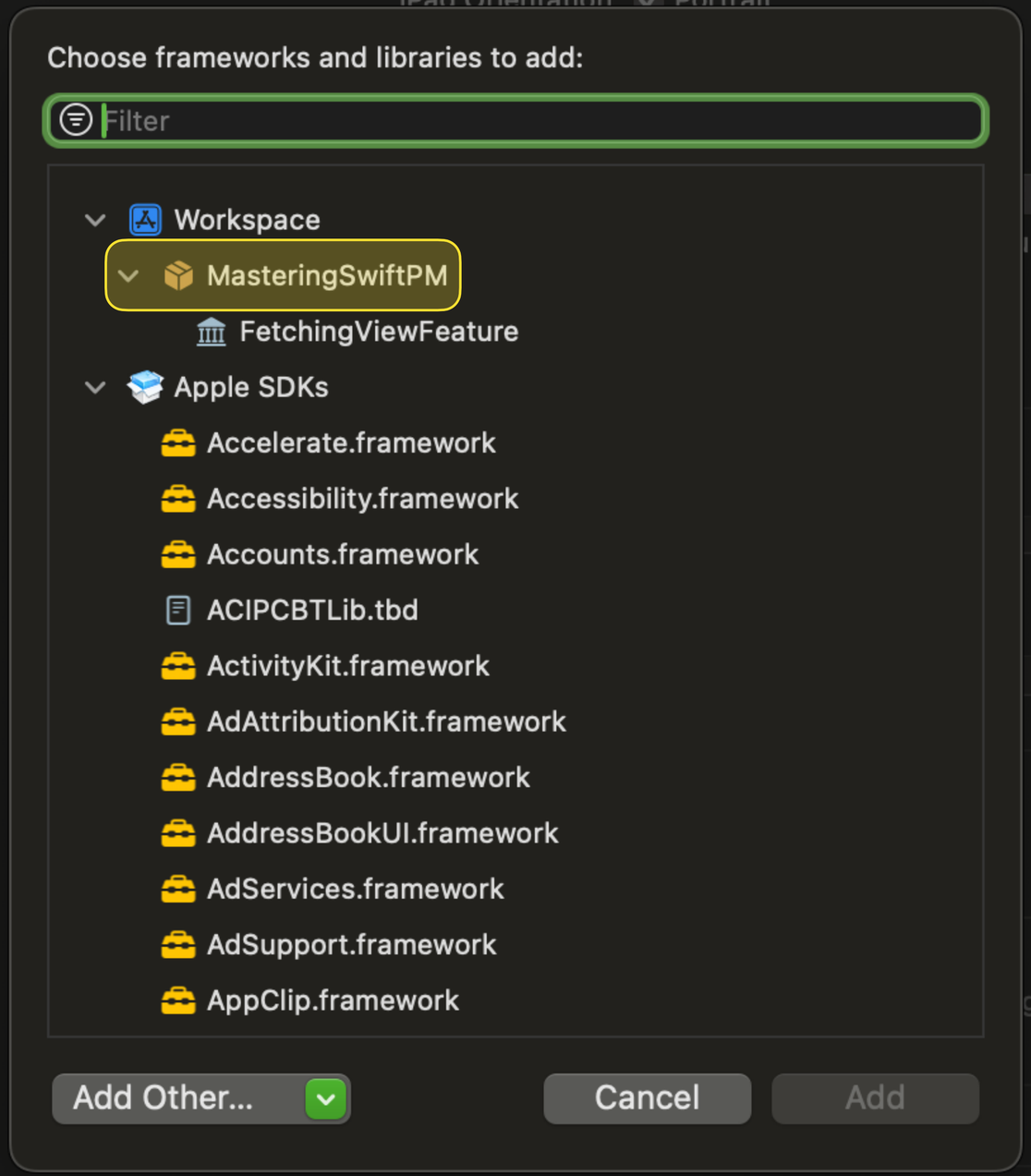
Development Assets

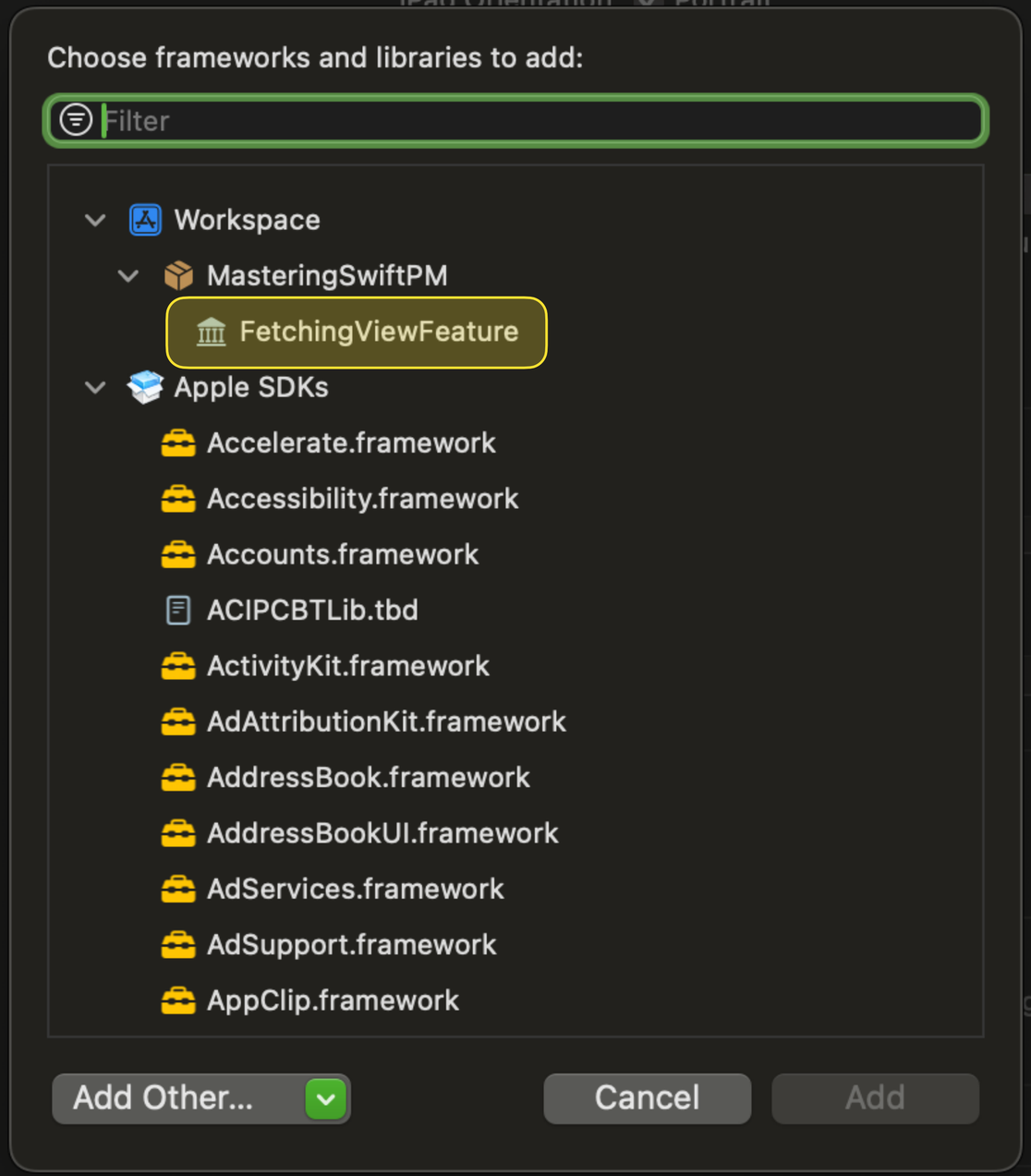
MasteringSwiftPMExampleProject/Preview Content

Filter

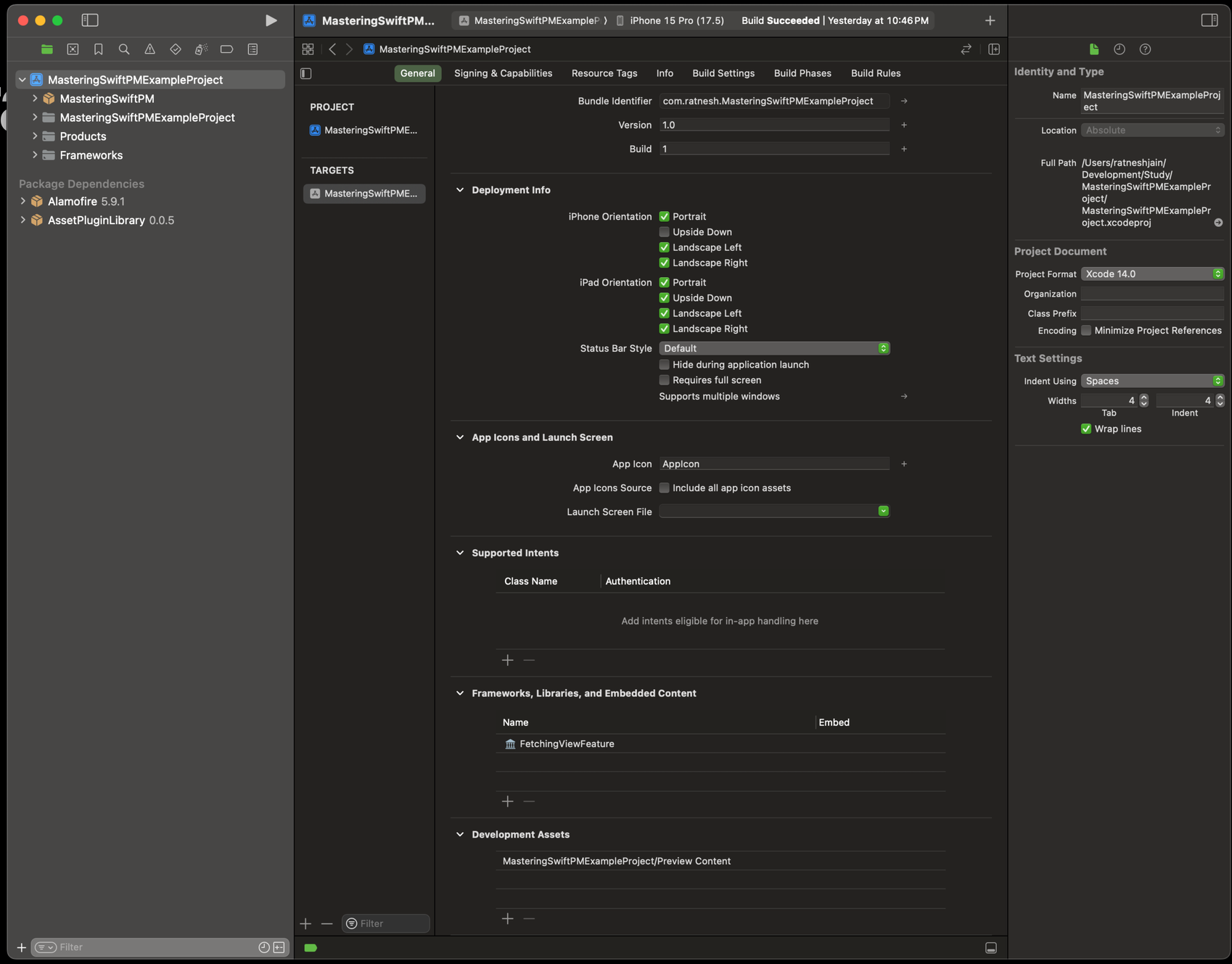
Integra


MasteringSwiftPM...
MasteringSwiftPMExampleP
iPhone 15 Pro (17.5)
Build Succeeded | Yesterday at 10:46 PM
MasteringSwiftPMExampleProject
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
MasteringSwiftPME...
TARGETS
MasteringSwiftPME...
Bundle Identifier com.ratnesh.MasteringSwiftPMExampleProject
Version 1.0
Build 1
Deployment Info
iPhone Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
iPad Orientation
Portrait
Choose frameworks and libraries to add:
Filter
Workspace
MasteringSwiftPM
FetchingViewFeature
Apple SDKs
Accelerate.framework
Accessibility.framework
Accounts.framework
ACIPCBTLib.tbd
ActivityKit.framework
AdAttributionKit.framework
AddressBook.framework
AddressBookUI.framework
AdServices.framework
AdSupport.framework
AppClip.framework
Add Other...
Cancel
Add
Frameworks, Libraries, and Embedded Content
Name
Embed
Add frameworks, libraries, and embedded content here
Development Assets
MasteringSwiftPMExampleProject/Preview Content
Identity and Type
Name MasteringSwiftPMExampleProject
Location Absolute
Full Path /Users/ratneshjain/Development/Study/MasteringSwiftPMExampleProject/MasteringSwiftPMExampleProject.xcodeproj
Project Document
Project Format Xcode 14.0
Organization
Class Prefix
Encoding Minimize Project References
Text Settings
Indent Using Spaces
Widths 4 4
Tab Indent
Wrap lines
MasteringSwiftPMExampleProject
MasteringSwiftPM
MasteringSwiftPMExampleProject
Products
Frameworks
Package Dependencies
Alamofire 5.9.1
AssetPluginLibrary 0.0.5
Filter


Choose frameworks and libraries to add:
Filter
Workspace
MasteringSwiftPM
FetchingViewFeature
Apple SDKs
Accelerate.framework
Accessibility.framework
Accounts.framework
ACIPCBTLib.tbd
ActivityKit.framework
AdAttributionKit.framework
AddressBook.framework
AddressBookUI.framework
AdServices.framework
AdSupport.framework
AppClip.framework
Add Other...
Cancel
Add


Choose frameworks and libraries to add:
Filter
Workspace
MasteringSwiftPM
FetchingViewFeature
Apple SDKs
Accelerate.framework
Accessibility.framework
Accounts.framework
ACIPCBTLib.tbd
ActivityKit.framework
AdAttributionKit.framework
AddressBook.framework
AddressBookUI.framework
AdServices.framework
AdSupport.framework
AppClip.framework
Add Other...
Cancel
Add

Integra


MasteringSwiftPM...
MasteringSwiftPMExampleP
iPhone 15 Pro (17.5)
Build Succeeded | Yesterday at 10:46 PM
MasteringSwiftPMExampleProject
MasteringSwiftPMExampleProject
MasteringSwiftPM
MasteringSwiftPMExampleProject
Products
Frameworks
Package Dependencies
Alamofire 5.9.1
AssetPluginLibrary 0.0.5
General
Signing & Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules
PROJECT
MasteringSwiftPME...
TARGETS
MasteringSwiftPME...
Bundle Identifier
com.ratnesh.MasteringSwiftPMExampleProject
Version
1.0
Build
1
Deployment Info
iPhone Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
iPad Orientation
Portrait
Upside Down
Landscape Left
Landscape Right
Status Bar Style
Default
Hide during application launch
Requires full screen
Supports multiple windows
App Icons and Launch Screen
App Icon
AppIcon
App Icons Source
Include all app icon assets
Launch Screen File
Supported Intents
Class Name
Authentication
Add intents eligible for in-app handling here
Frameworks, Libraries, and Embedded Content
Name
Embed
FetchingViewFeature
Development Assets
MasteringSwiftPMExampleProject/Preview Content
Identity and Type
Name
MasteringSwiftPMExampleProject
Location
Absolute
Full Path
/Users/ratneshjain/Development/Study/MasteringSwiftPMExampleProject/MasteringSwiftPMExampleProject.xcodeproj
Project Document
Project Format
Xcode 14.0
Organization
Class Prefix
Encoding
Minimize Project References
Text Settings
Indent Using
Spaces
Widths
4
4
Tab
Indent
Wrap lines
Filter
Filter

Supported Intents

Class Name	Authentication

Add intents eligible for in-app handling here

Frameworks, Libraries, and Embedded Content

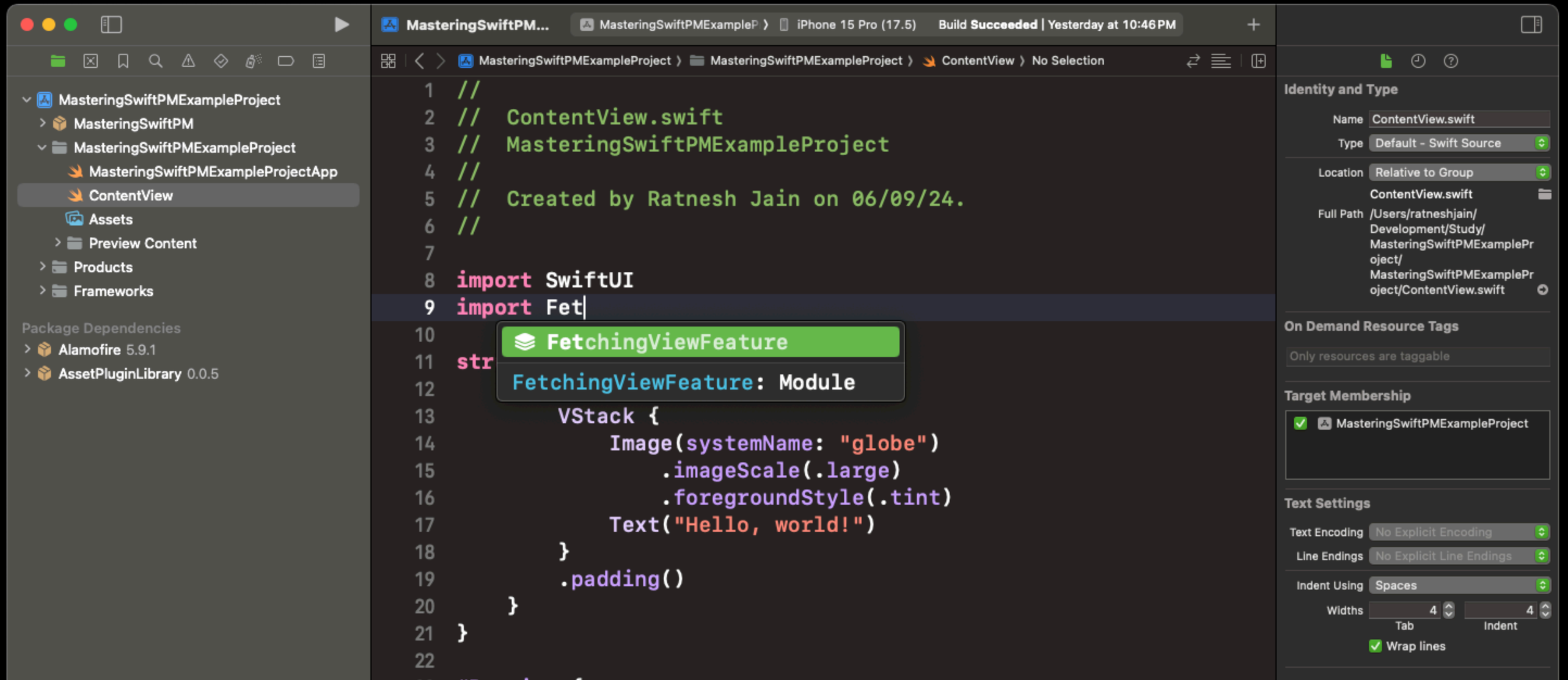
Name	Embed
FetchingViewFeature	

Development Assets

MasteringSwiftPMExampleProject/Preview Content

Filter

Integration with Xcode Project



Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Sharing Resources
With Other Modules

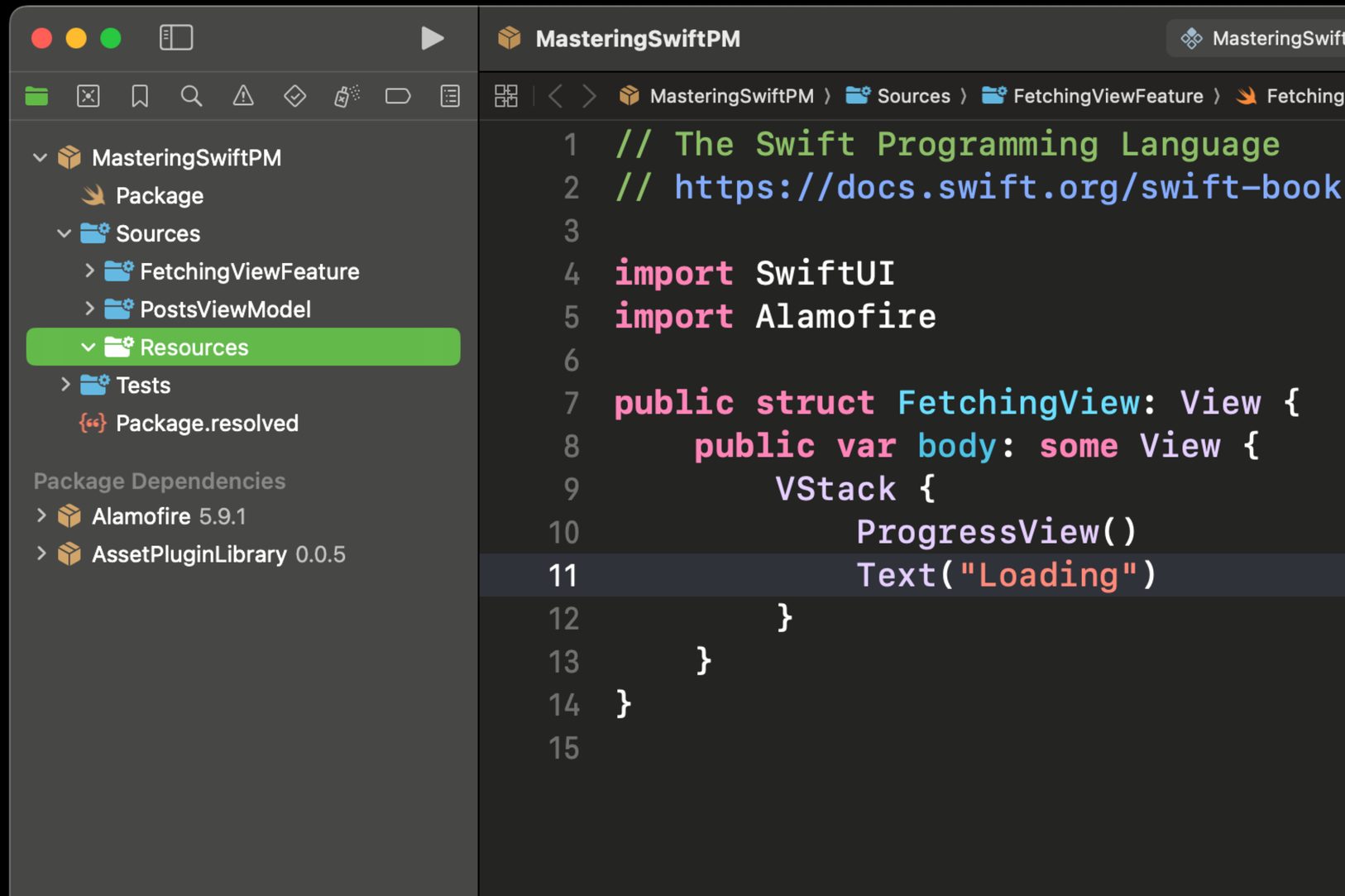
- Assets
  - Images
  - Colors
- Fonts
- Core Data
- Other Resources

Assets.xcassets

Sharing Resources

Assets

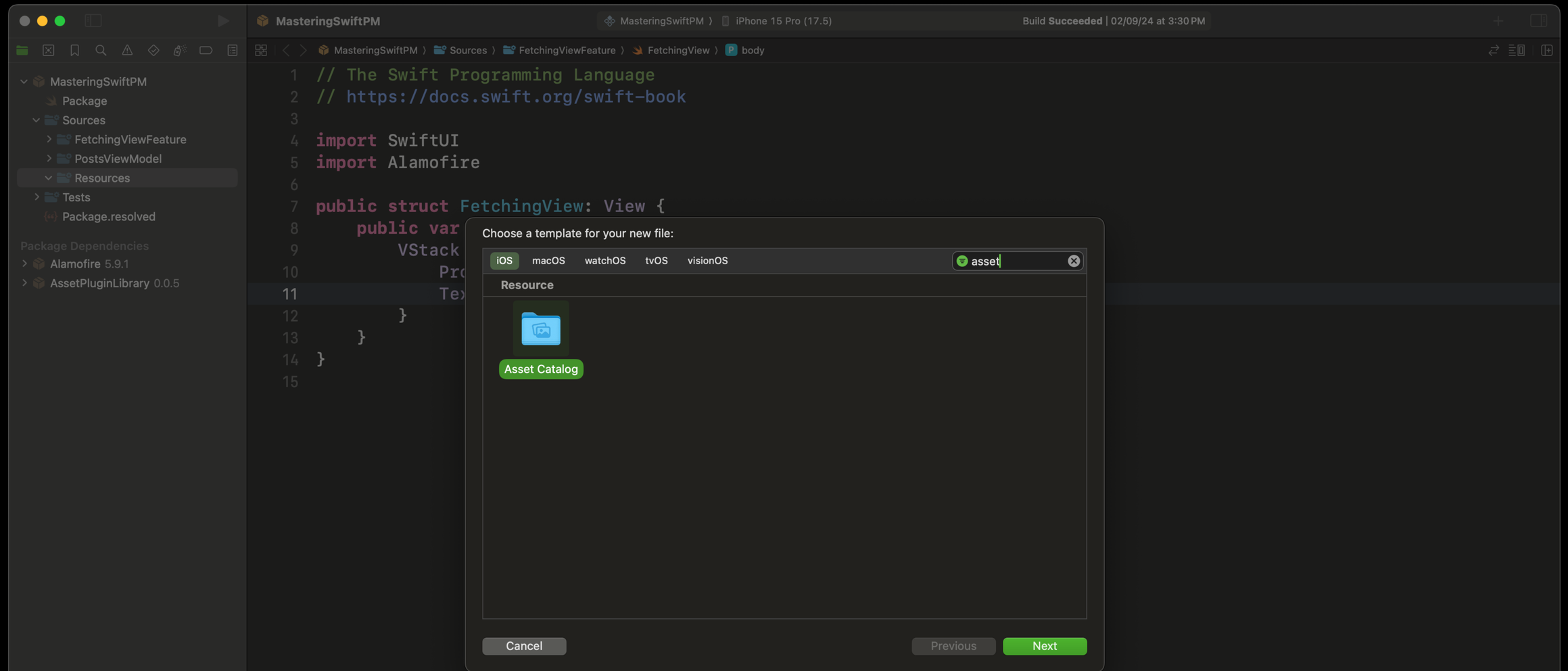
- Create New Folder
- Name it “Resources”



Sharing Resources

Assets

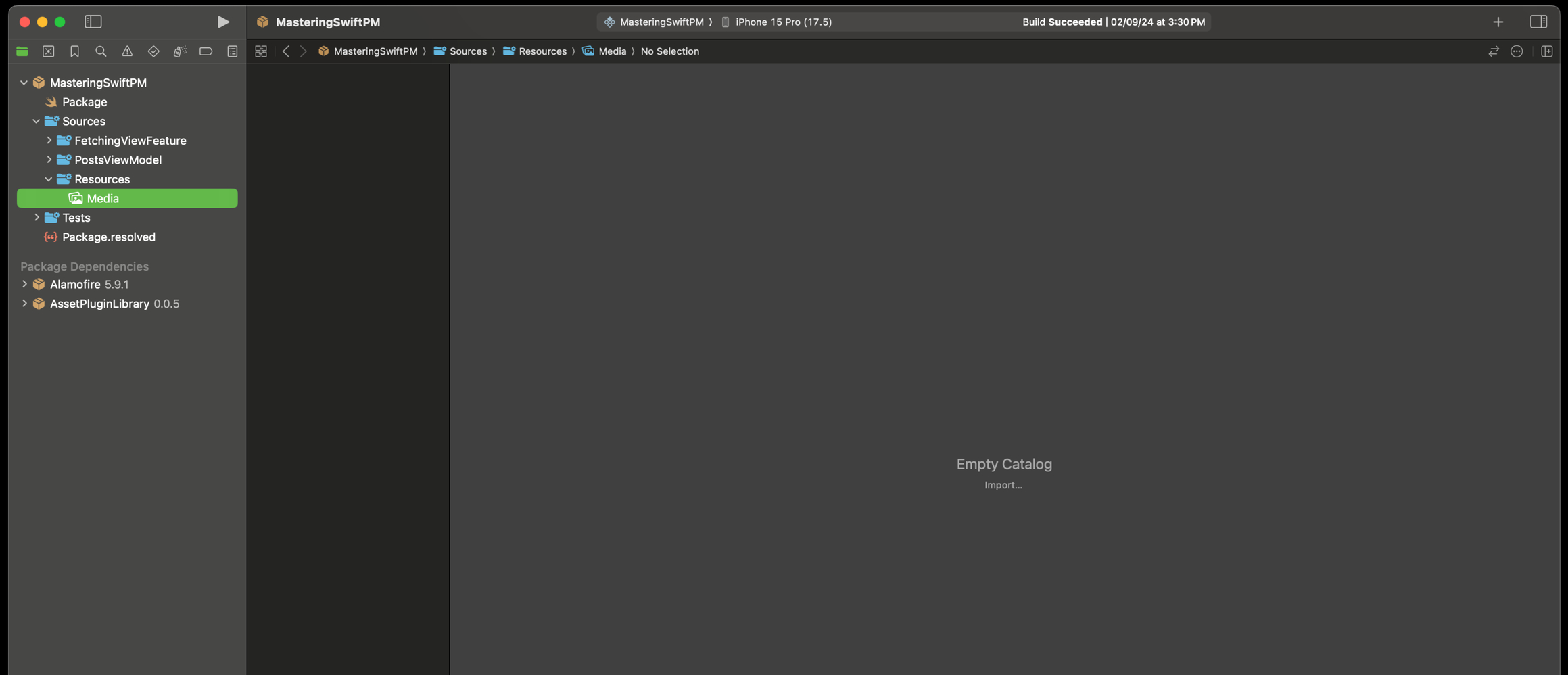
- File → New File
- Search for asset



Sharing Resources

Assets

- Will place images here



Sharing Resources

Assets 

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [ ... ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
       .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

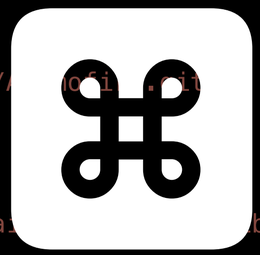
Sharing Resources

Assets 

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [ ... ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
       .target(name: "Resources"),
       .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

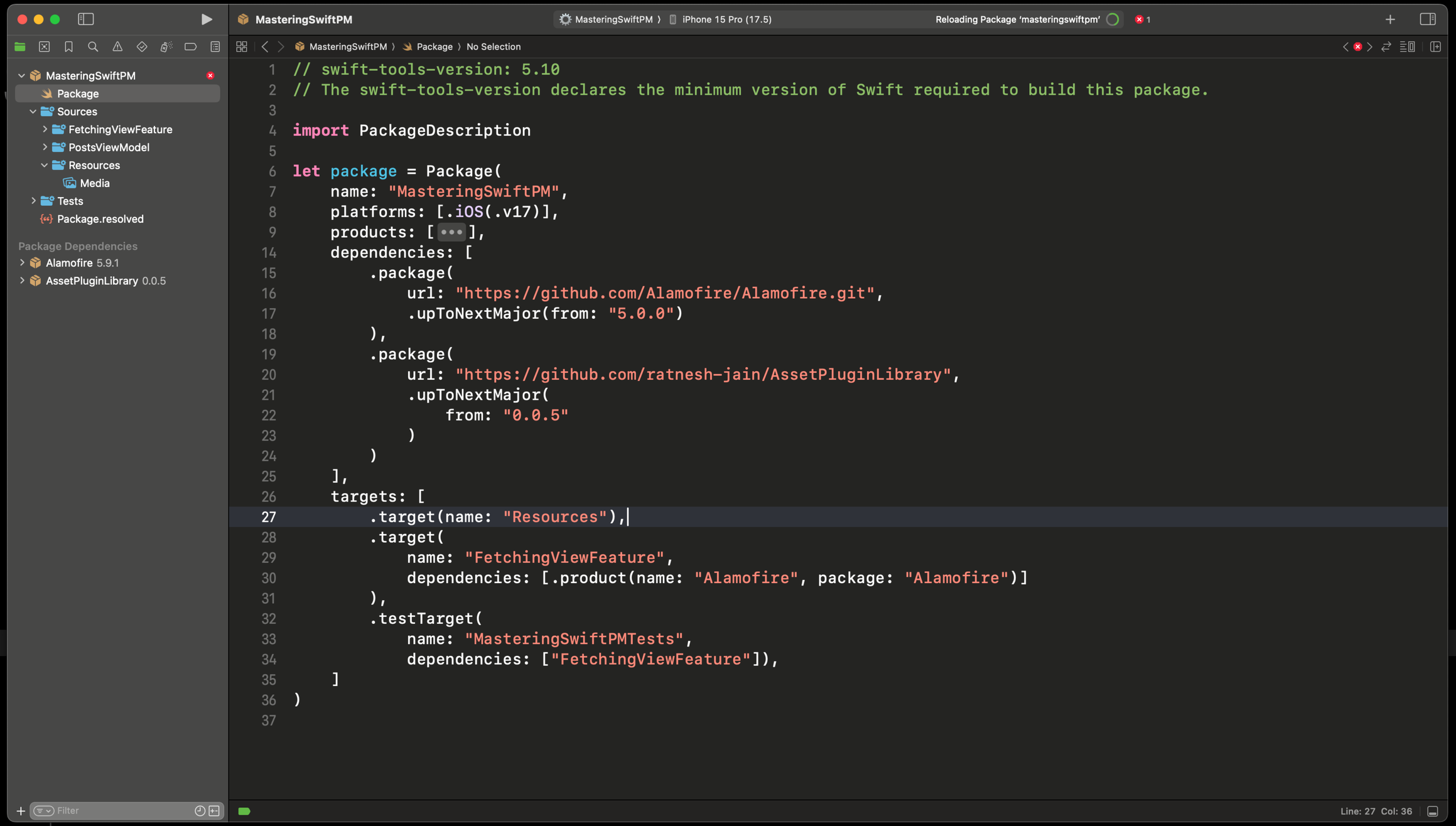
Sharing Resources

Assets

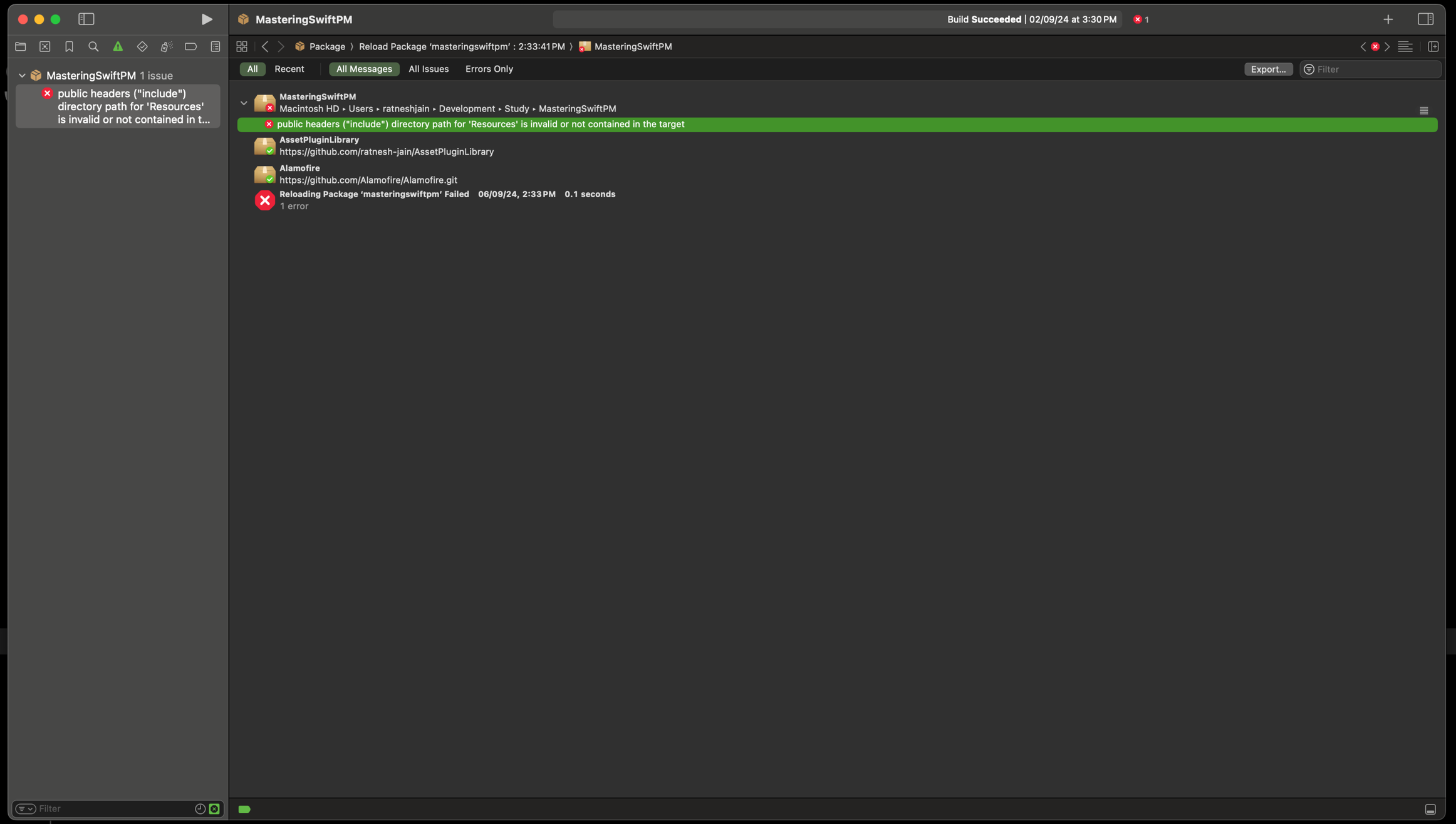


Save

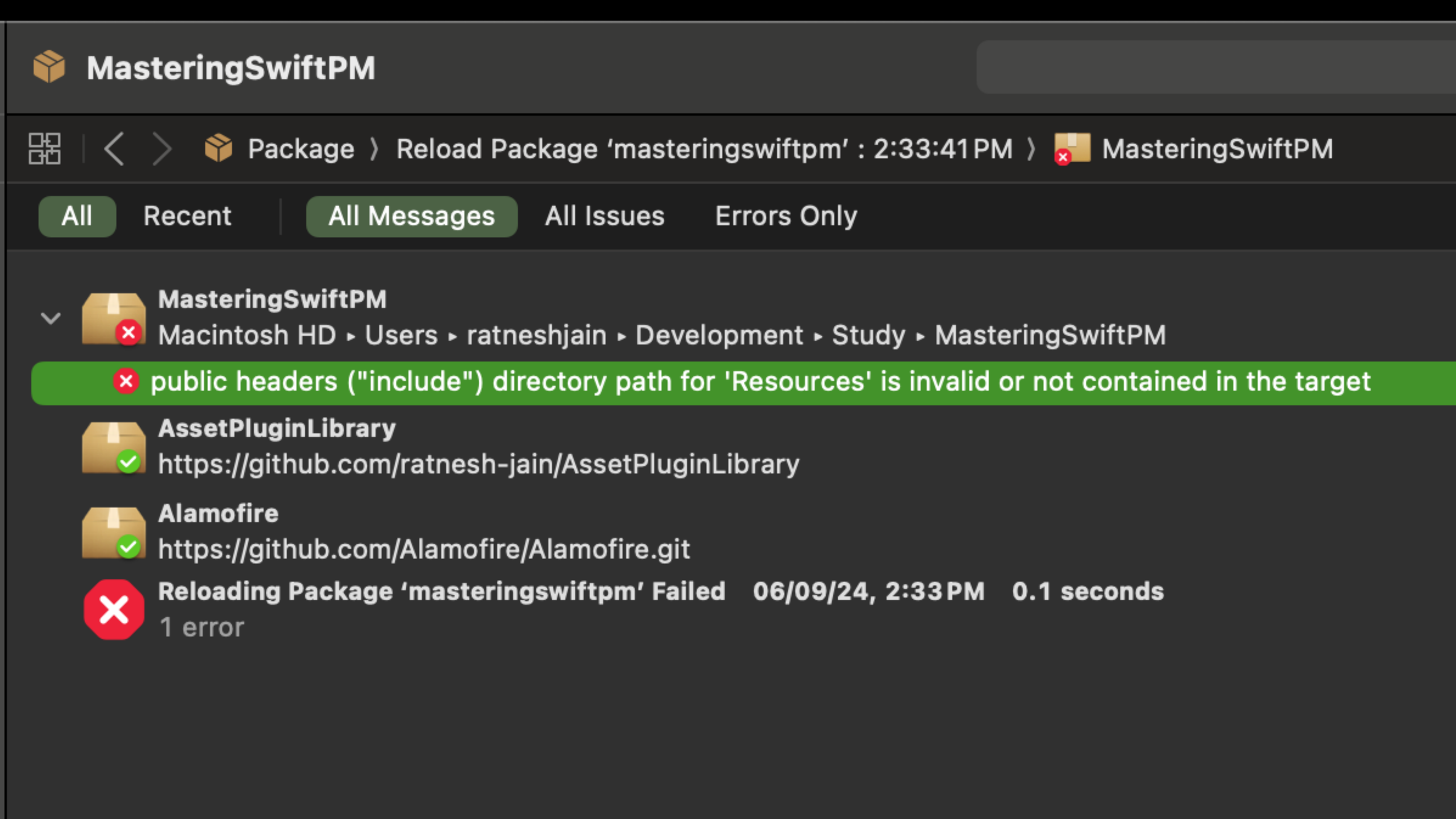
```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [ ... ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/A      ofi      .git
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jai                brary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
       .target(name: "Resources"),
       .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```

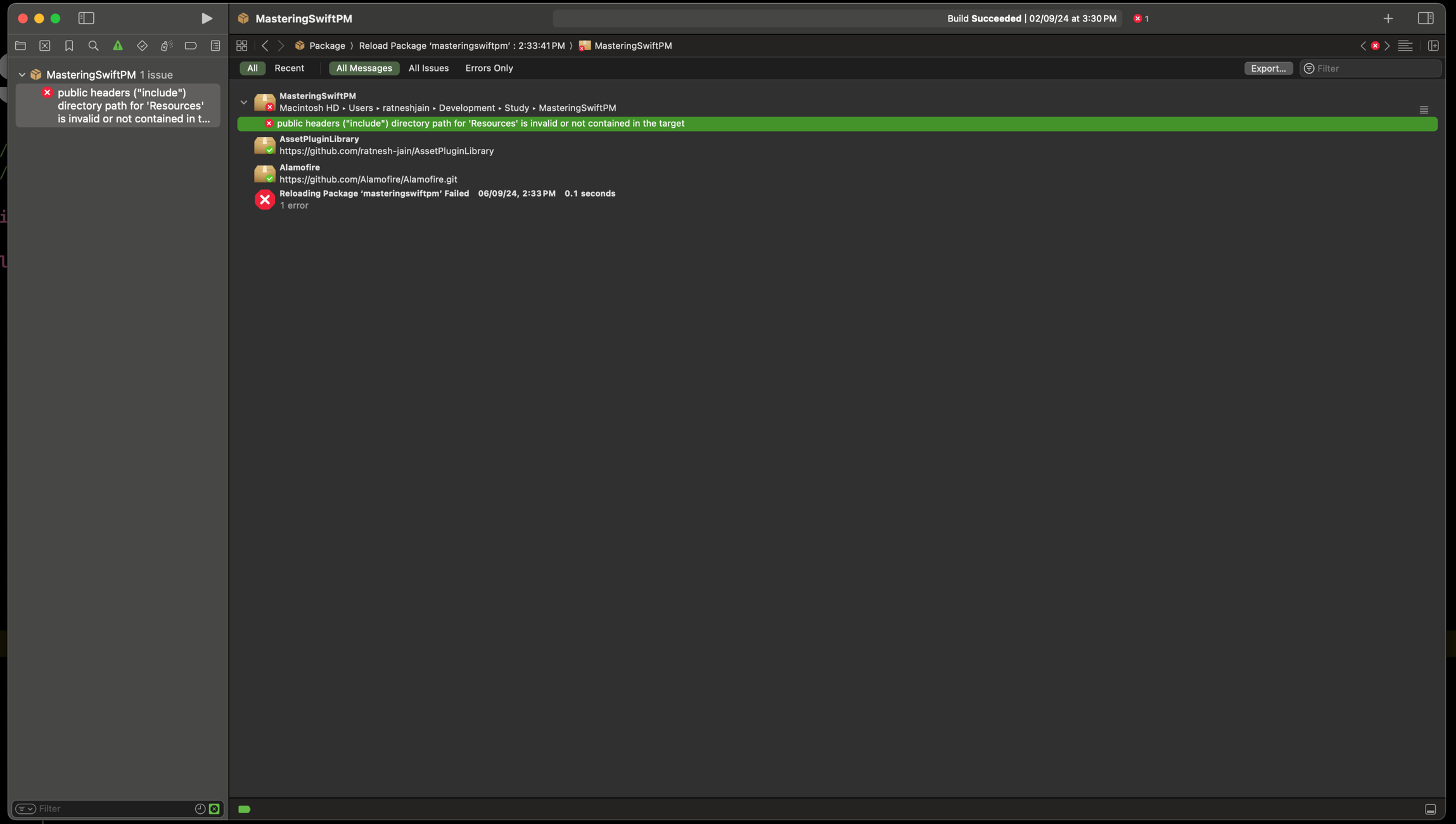


Ratnesh Jain

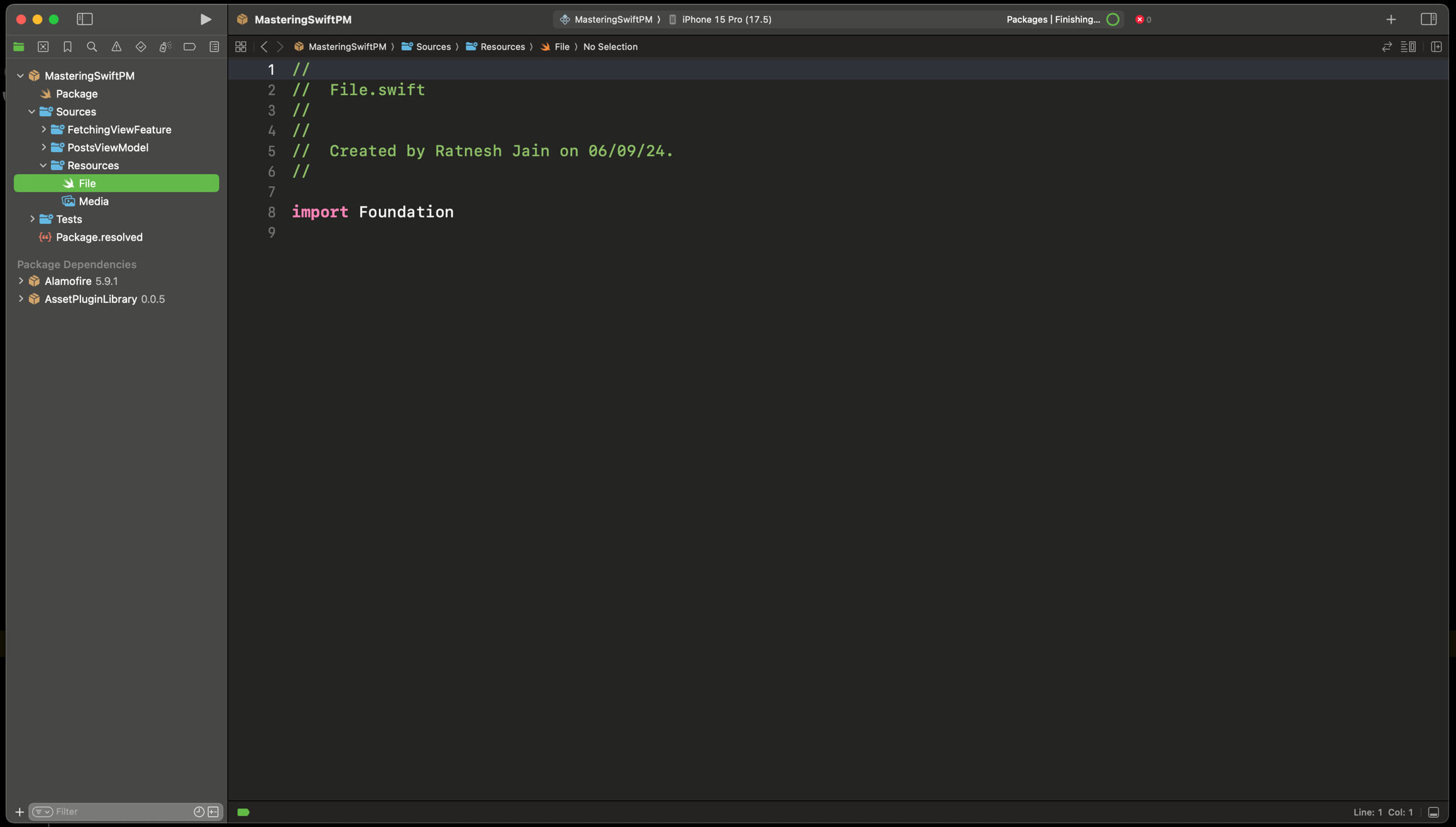


Ratnesh Jain


MasteringSwiftPM
Package 〉 Reload Package ‘masteringswiftpm’ : 2:33:41 PM 〉 MasteringSwiftPM
All
Recent
All Messages
All Issues
Errors Only
MasteringSwiftPM
Macintosh HD ▸ Users ▸ ratneshjain ▸ Development ▸ Study ▸ MasteringSwiftPM
public headers ("include") directory path for 'Resources' is invalid or not contained in the target
AssetPluginLibrary
https://github.com/ratnesh-jain/AssetPluginLibrary
Alamofire
https://github.com/Alamofire/Alamofire.git
Reloading Package ‘masteringswiftpm’ Failed 06/09/24, 2:33 PM 0.1 seconds
1 error



Ratnesh Jain



Ratnesh Jain

MasteringSwiftPM

Resources ⟩ iPhone 15 Pro (17.5) Build Succeeded | 02/09/24 at 3:30 PM

MasteringSwiftPM ⟩ Sources ⟩ Resources ⟩ Media ⟩ AdamSavage

- MasteringSwiftPM
  - Package
  - Sources
    - FetchingViewFeature
    - PostsViewModel
    - Resources
      - File
      - Media
  - Tests
  - Package.resolved

Package Dependencies

- Alamofire 5.9.1
- AssetPluginLibrary 0.0.5

Filter

AdamSavage
MikeAsh
OleBegemann

Filter

AdamSavage Image

1x 2x 3x

Universal

MikeAsh Image

1x 2x 3x

Universal

OleBegemann Image

1x 2x 3x

Universal

Ratnesh Jain

Sharing Resources

Assets

```
import Foundation
import SwiftUI

struct InfluencersView: View {
    var body: some View {
        List {
            Image(.adamSavage)
            Image(.mikeAsh)
            Image(.oleBegemann)
        }
    }
}

#Preview {
    InfluencersView()
}
```

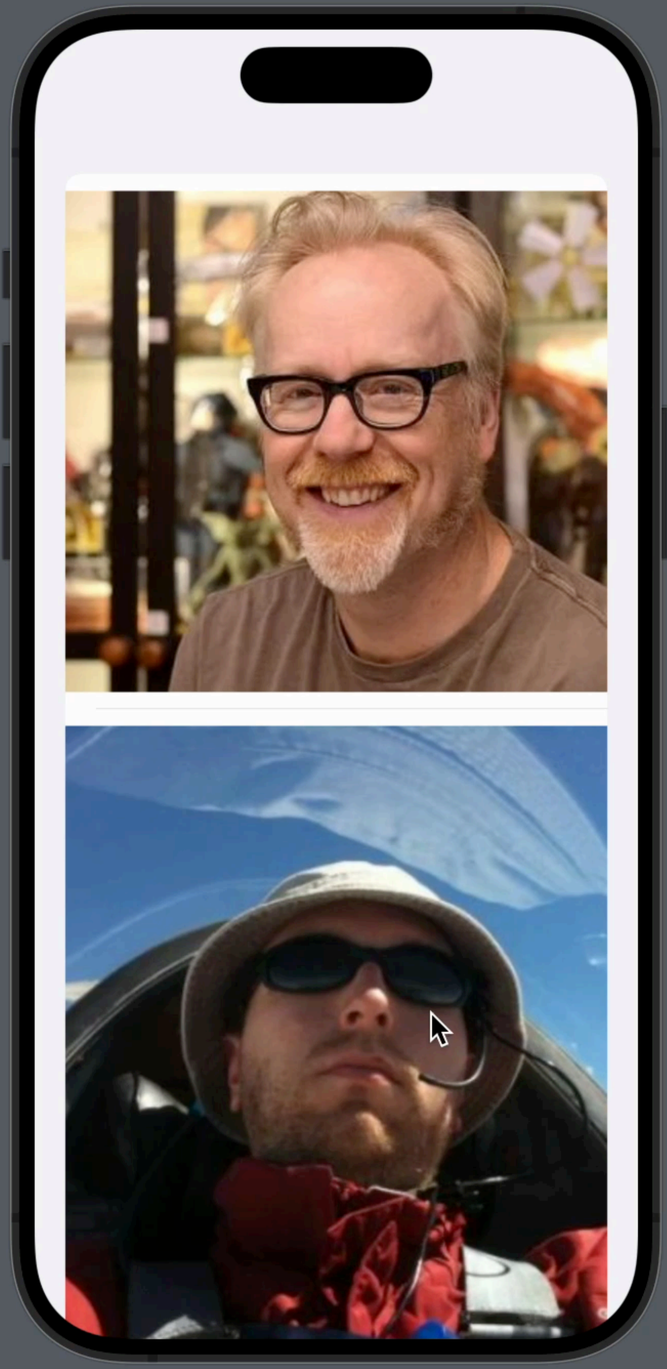
Sharing Resources

```
import Foundation
import SwiftUI

struct InfluencersView: View {
    var body: some View {
        List {
            Image(.adamSavage)
            Image(.mikeAsh)
            Image(.)
        }
    }
}

#Preview {
    InfluencersVie
}
```

- M adamSavage
- M mikeAsh
- M oleBegemann
- M actions
- M add
- M checkmark
- M remove
- M strokedCheckmark

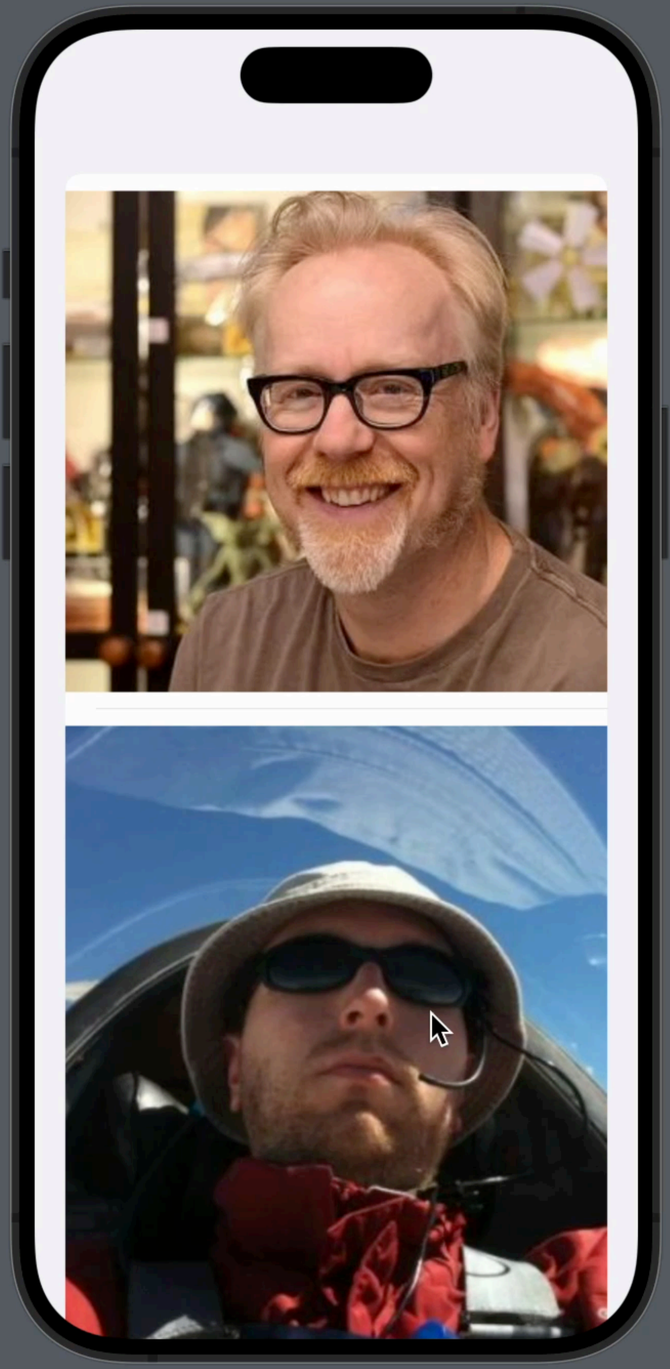
adamSavage: ImageResource

The “AdamSavage” asset catalog image resource.



Sharing Resources

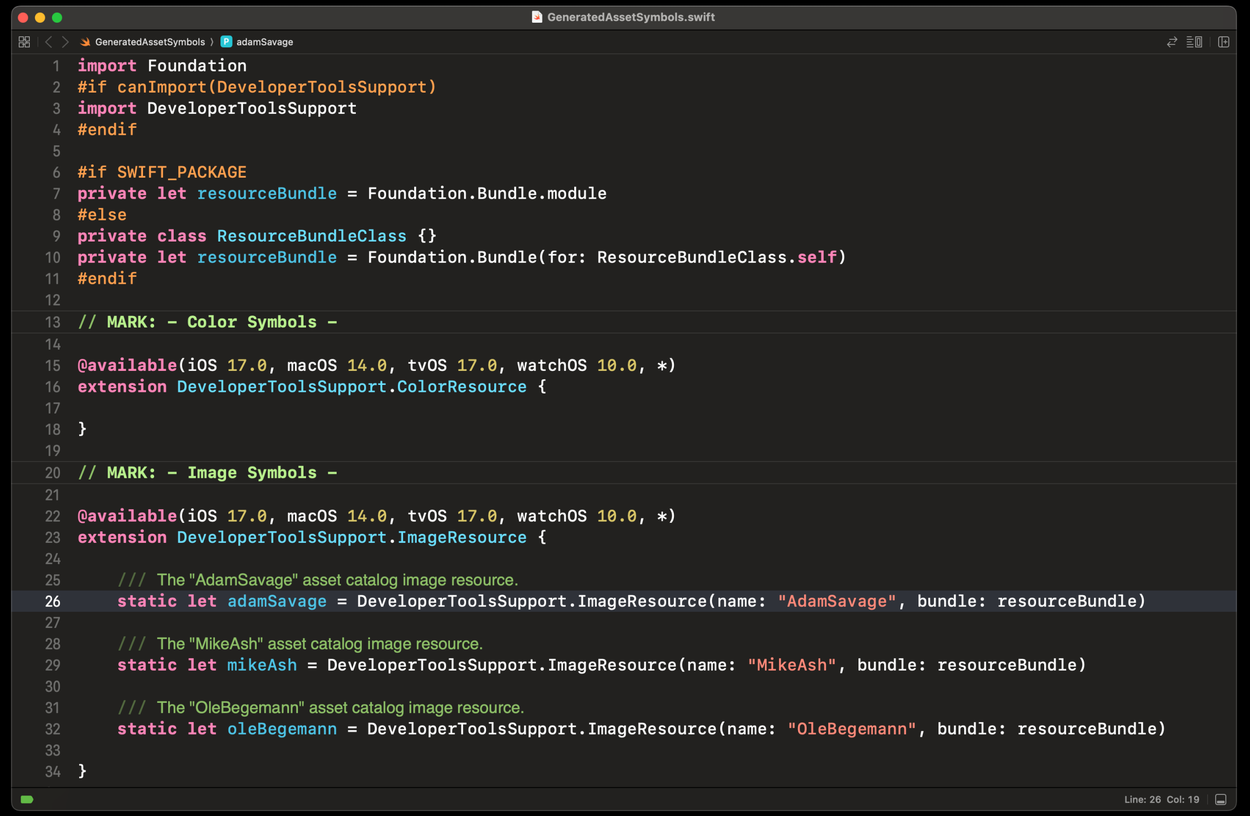
```
import Foundation
import SwiftUI

struct InfluencersView: View {
    var body: some View {
        List {
            Image(.adamSavage)
            Image(.mikeAsh)
            Image(.oleBegemann)
        }
    }
}

#Preview {
    InfluencersView()
}
```



Sharing Resources

Assets

DerivedData/MasteringSwiftPM-bgofwlsbcmyvvlaocnlrajdewtnq/Build/Intermediates.noindex/Previews/iphonesimulator/Resources/Intermediates.noindex/MasteringSwiftPM.build/Debug-iphonesimulator/Resources.build/DerivedSources/GeneratedAssetSymbols.swift




```swift
import Foundation
#if canImport(DeveloperToolsSupport)
import DeveloperToolsSupport
#endif

#if SWIFT_PACKAGE
private let resourceBundle = Foundation.Bundle.module
#else
private class ResourceBundleClass {}
private let resourceBundle = Foundation.Bundle(for: ResourceBundleClass.self)
#endif

// MARK: - Color Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ColorResource {

}

// MARK: - Image Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ImageResource {

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = DeveloperToolsSupport.ImageResource(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = DeveloperToolsSupport.ImageResource(name: "MikeAsh", bundle: resourceBundle)

    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = DeveloperToolsSupport.ImageResource(name: "OleBegemann", bundle: resourceBundle)

}
```

```
#endif

#if SWIFT_PACKAGE
private let resourceBundle = Foundation.Bundle.module
#else
private class ResourceBundleClass {}
private let resourceBundle = Foundation.Bundle(for: ResourceBundleClass.self)
#endif

// MARK: - Color Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ColorResource {

}

// MARK: - Image Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ImageResource {

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = DeveloperToolsSupport.ImageResource(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = DeveloperToolsSupport.ImageResource(name: "MikeAsh", bundle: resourceBundle)

    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = DeveloperToolsSupport.ImageResource(name: "OleBegemann", bundle: resourceBundle)

}
```

```
#endif

#if SWIFT_PACKAGE
private let resourceBundle = Foundation.Bundle.module
#else
private class ResourceBundleClass {}
private let resourceBundle = Foundation.Bundle(for: ResourceBundleClass.self)
#endif

// MARK: - Color Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ColorResource {

}

// MARK: - Image Symbols -

@available(iOS 17.0, macOS 14.0, tvOS 17.0, watchOS 10.0, *)
extension DeveloperToolsSupport.ImageResource {

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = DeveloperToolsSupport.ImageResource(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = DeveloperToolsSupport.ImageResource(name: "MikeAsh", bundle: resourceBundle)

    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = DeveloperToolsSupport.ImageResource(name: "OleBegemann", bundle: resourceBundle)

}
```

Auto generated Image resource does not mark “Public” access control

Sharing Resources

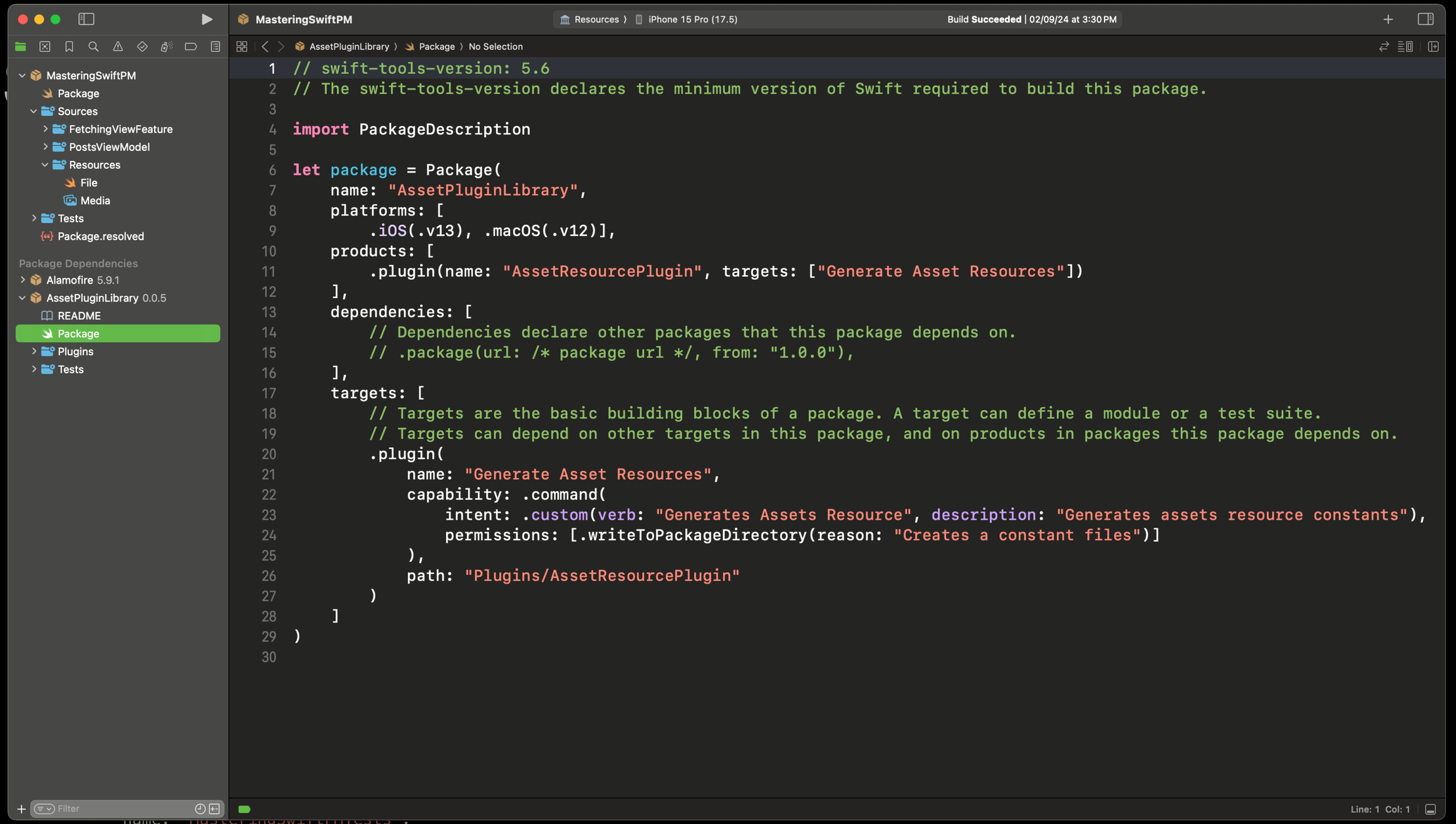
Assets

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
```

Sharing Resources

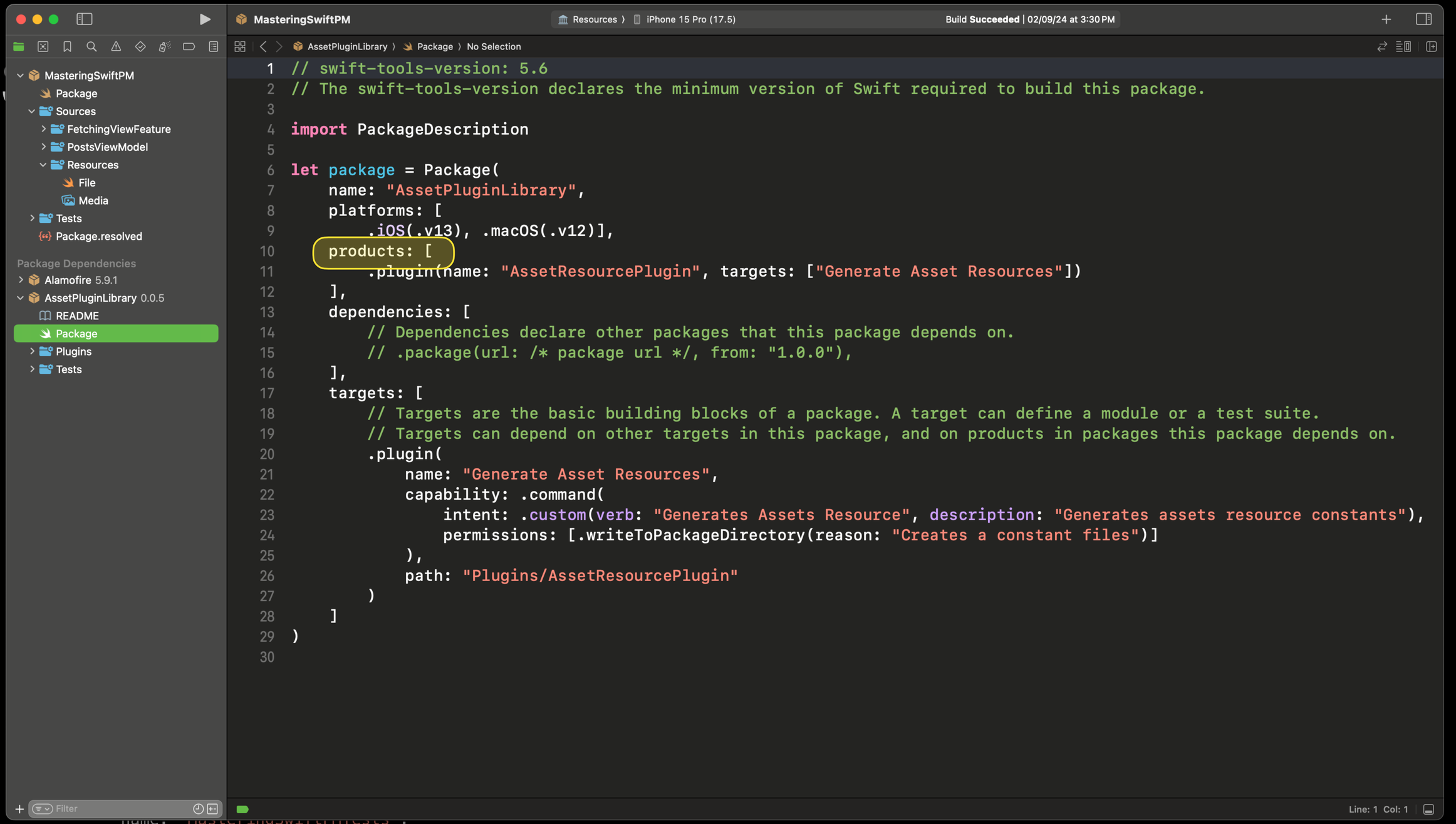
Assets

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
)
```




```
        dependencies: ["FetchingViewFeature"]),
]
```

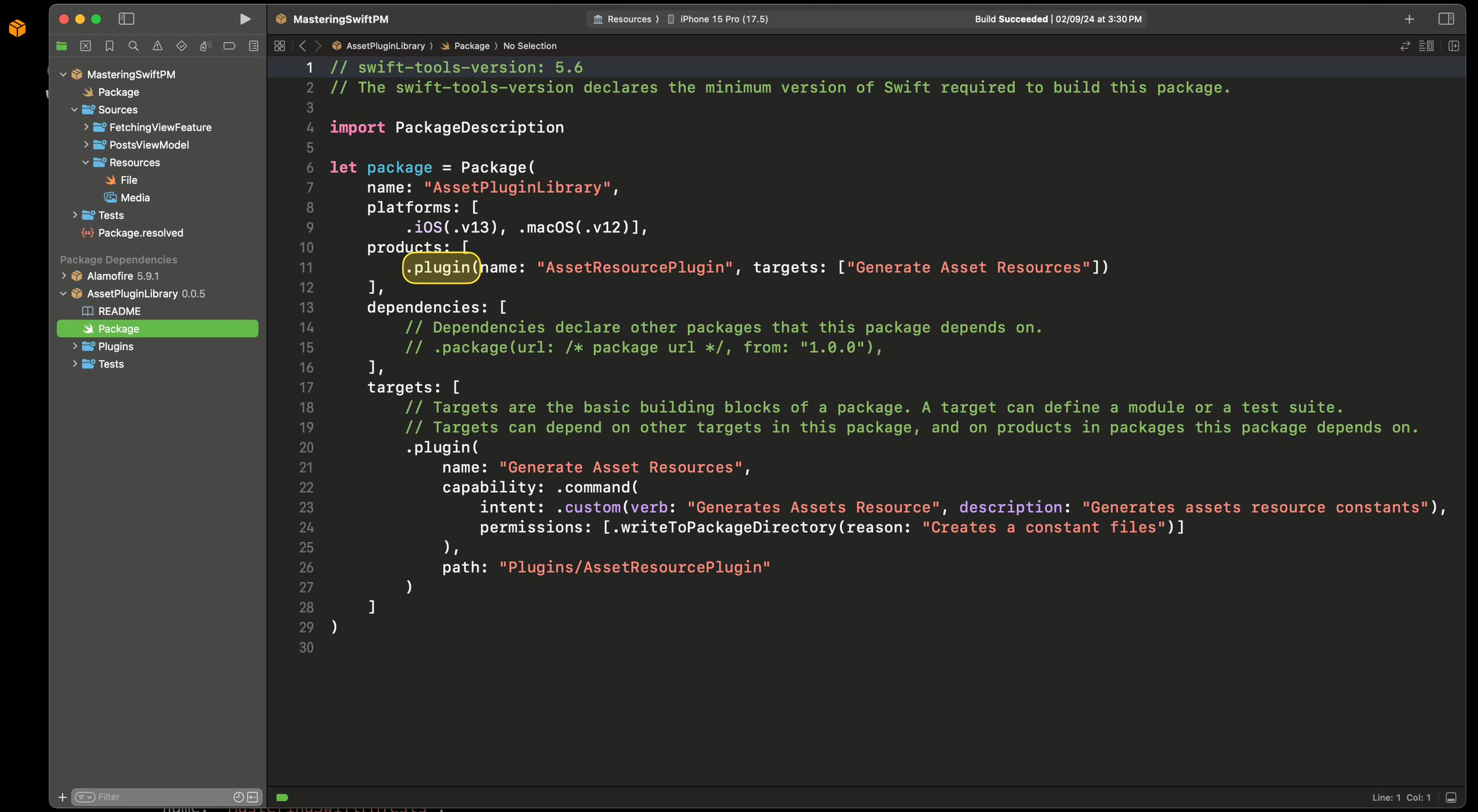
Ratnesh Jain



dependencies: ["FetchingViewFeature"]),

]

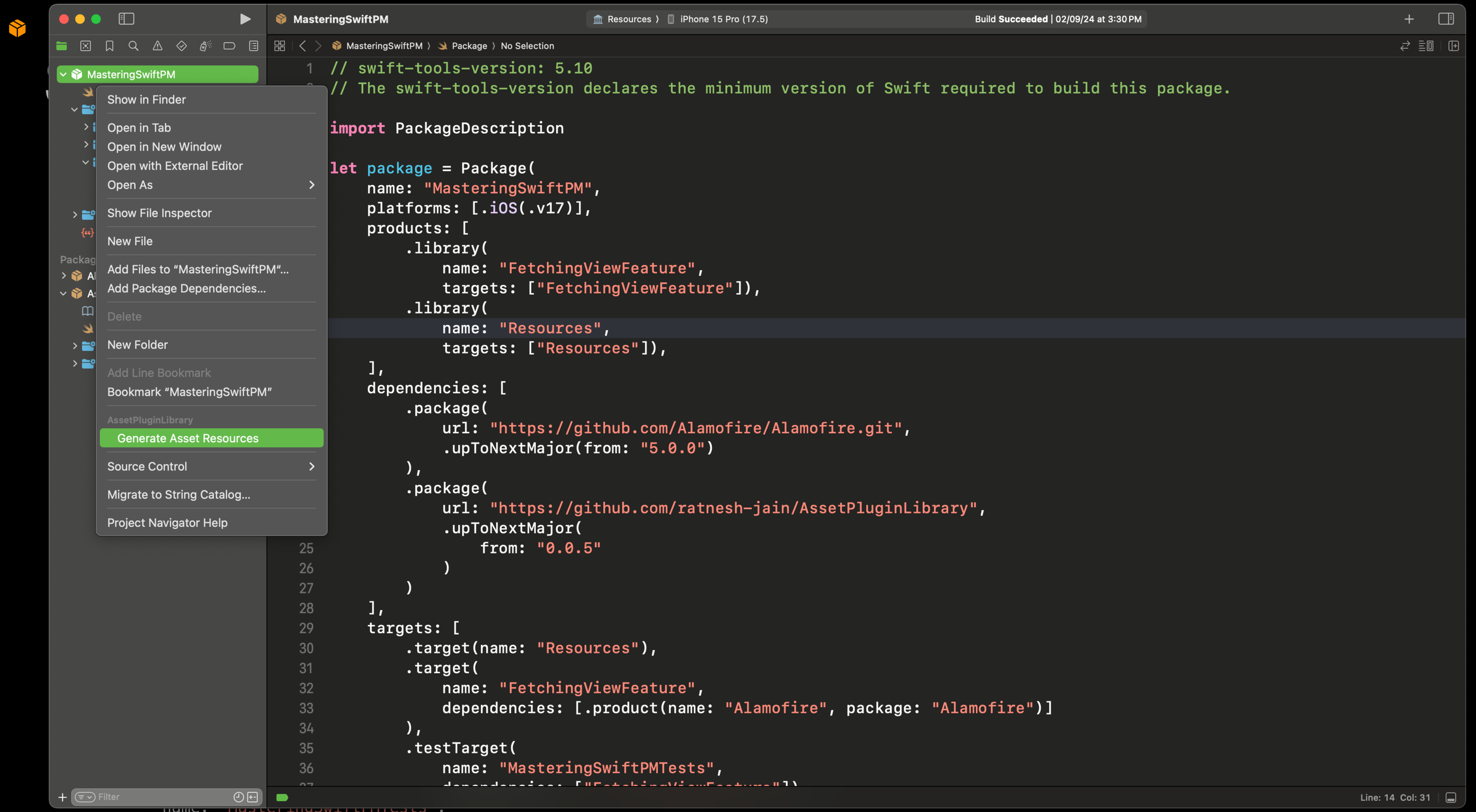
Ratnesh Jain



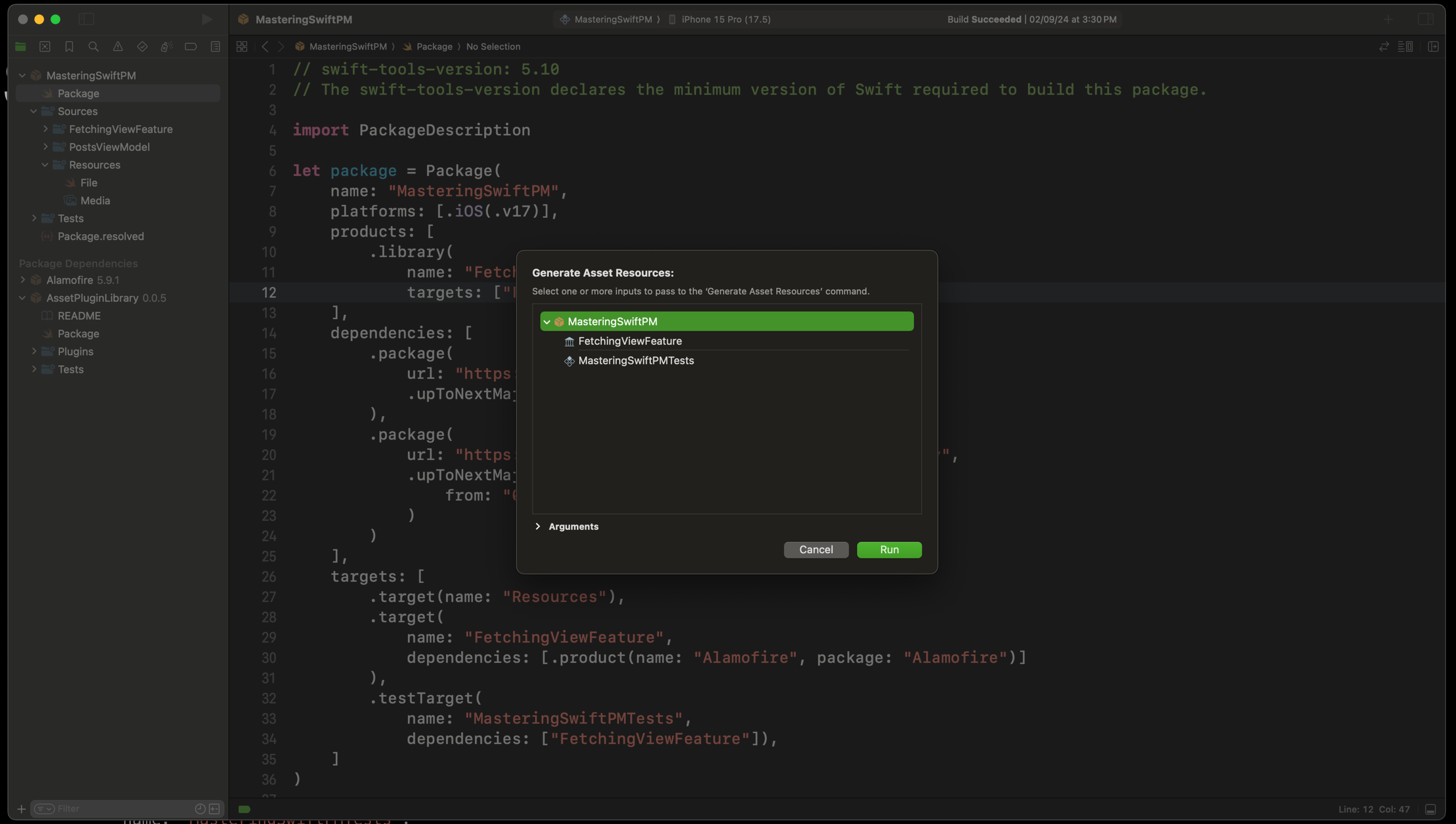
dependencies: ["FetchingViewFeature"]),

]

Ratnesh Jain



Ratnesh Jain



dependencies: ["FetchingViewFeature"]),

]

Ratnesh Jain

Sharing Resources

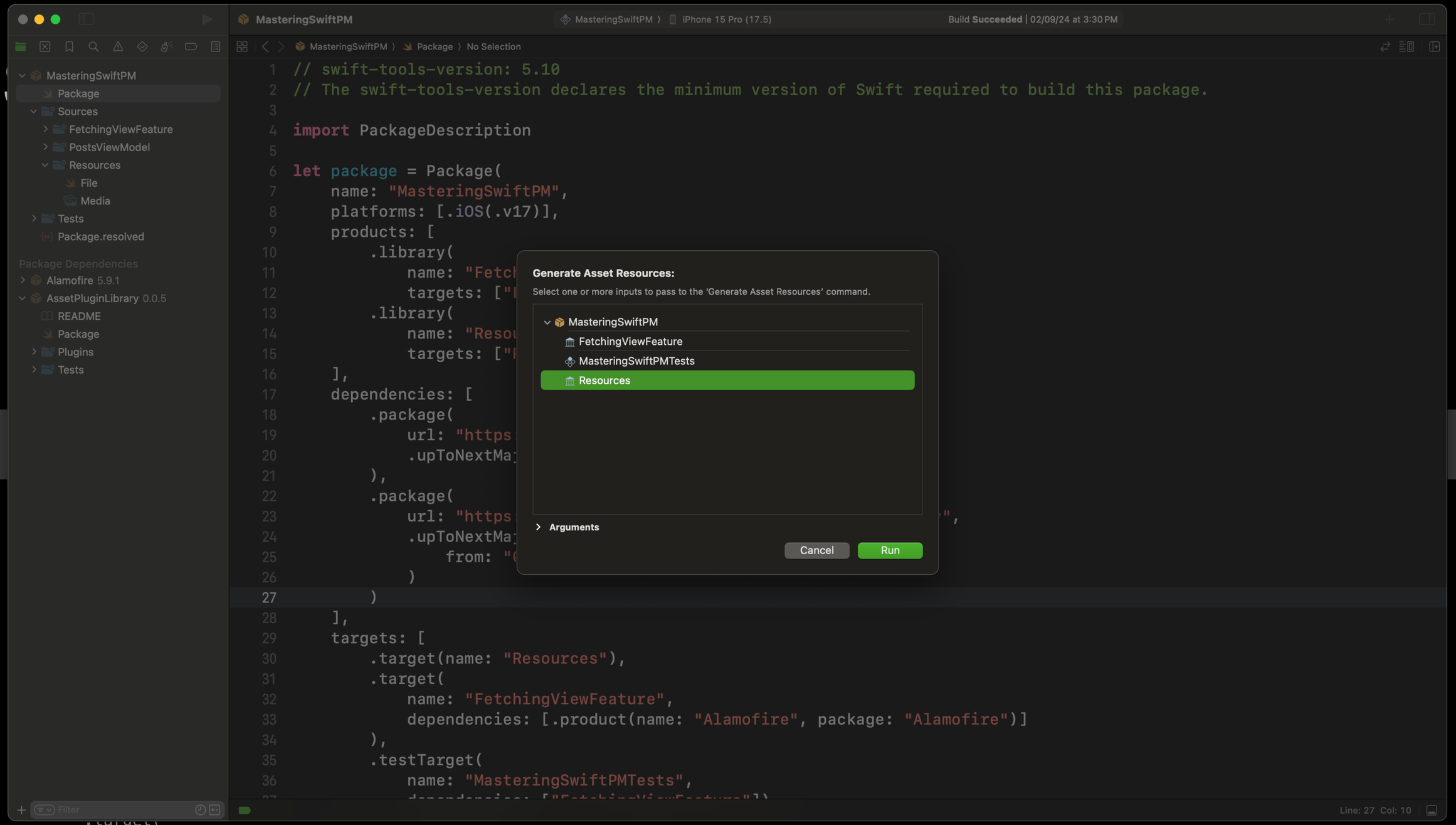
Assets

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(
            name: "FetchingViewFeature"
        ),
        .testTarget(
            name: "MasteringSwiftPMTests",
            dependencies: ["FetchingViewFeature"]),
    ]
```

Sharing Resources

Assets

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
        .library(
            name: "Resources",
            targets: ["Resources"])
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(name: "Resources"),
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
```



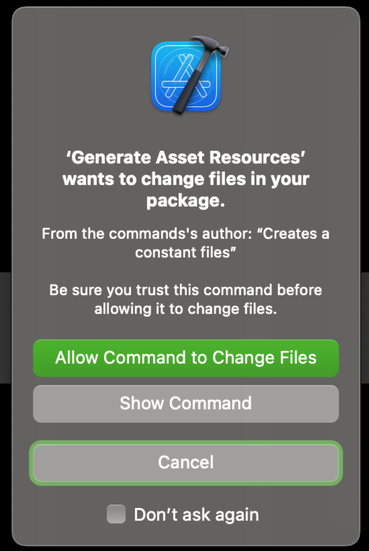
name: "FetchingViewFeature",
dependencies: [.product(name: "Alamofire", package: "Alamofire")]

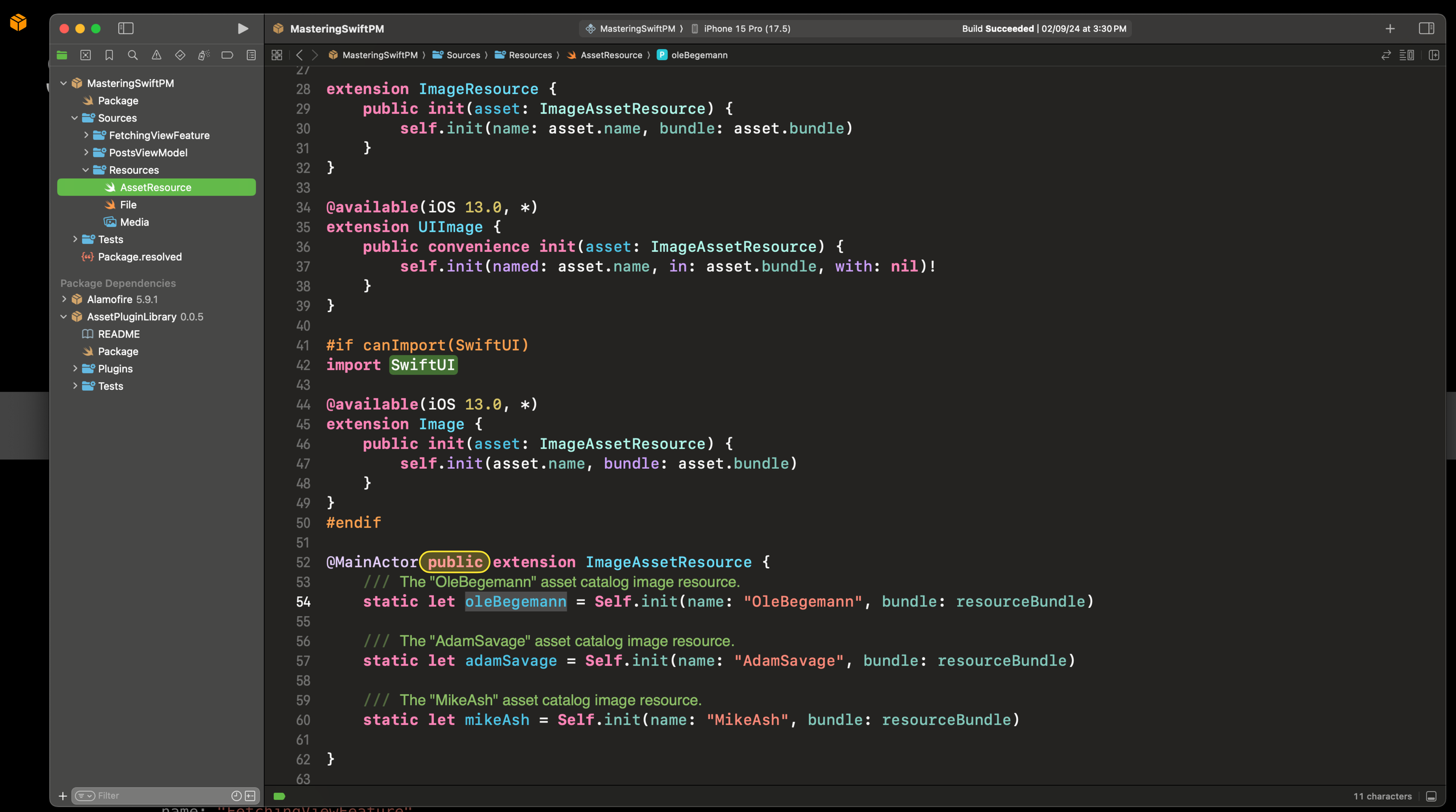
Ratnesh Jain

Sharing Resources

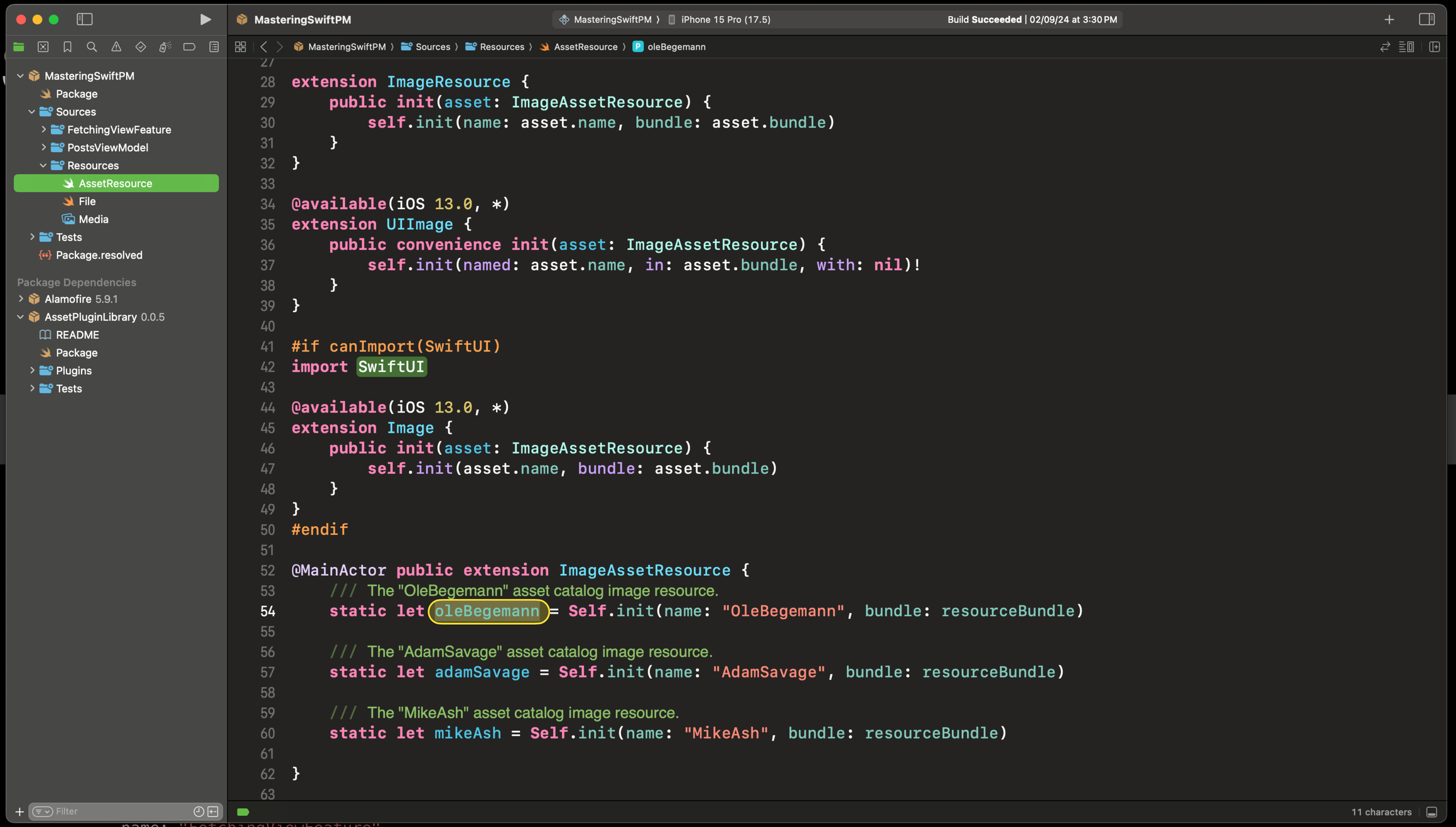
Assets

```
// swift-tools-version: 5.10
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "MasteringSwiftPM",
    platforms: [.iOS(.v17)],
    products: [
        .library(
            name: "FetchingViewFeature",
            targets: ["FetchingViewFeature"]),
        .library(
            name: "Resources",
            targets: ["Resources"])
    ],
    dependencies: [
        .package(
            url: "https://github.com/Alamofire/Alamofire.git",
            .upToNextMajor(from: "5.0.0")
        ),
        .package(
            url: "https://github.com/ratnesh-jain/AssetPluginLibrary",
            .upToNextMajor(
                from: "0.0.5"
            )
        )
    ],
    targets: [
        .target(name: "Resources"),
        .target(
            name: "FetchingViewFeature",
            dependencies: [.product(name: "Alamofire", package: "Alamofire")]
```

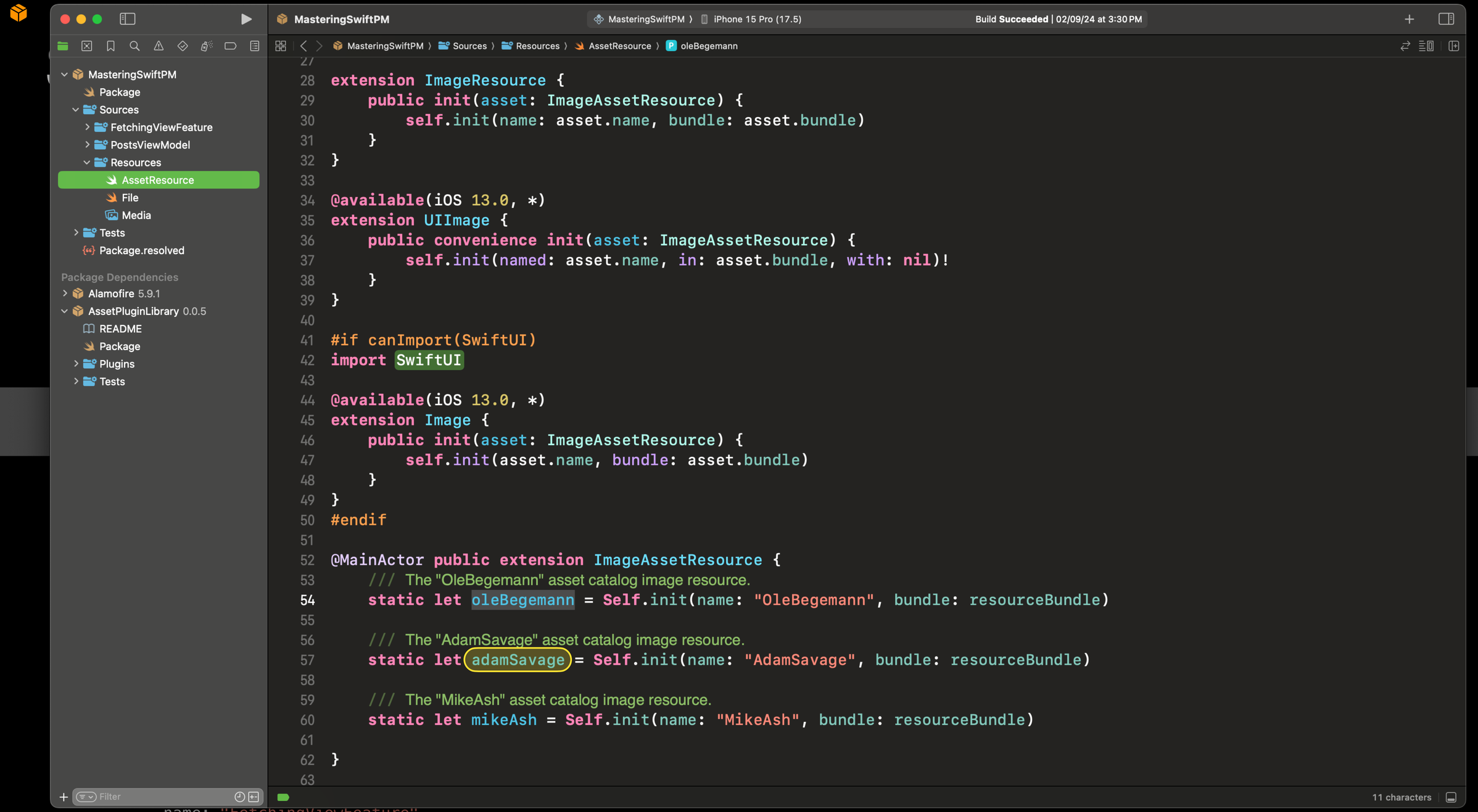




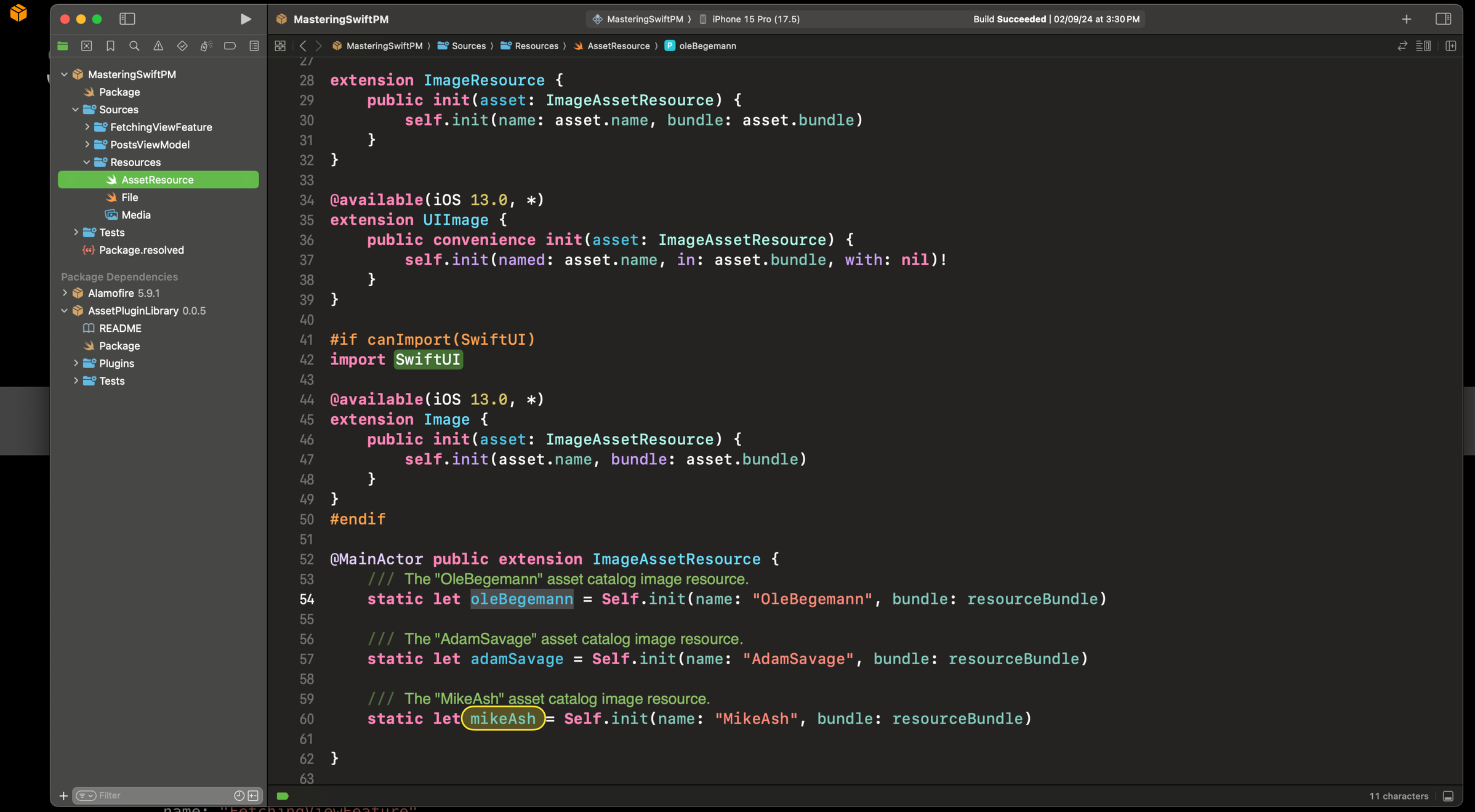

```
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```



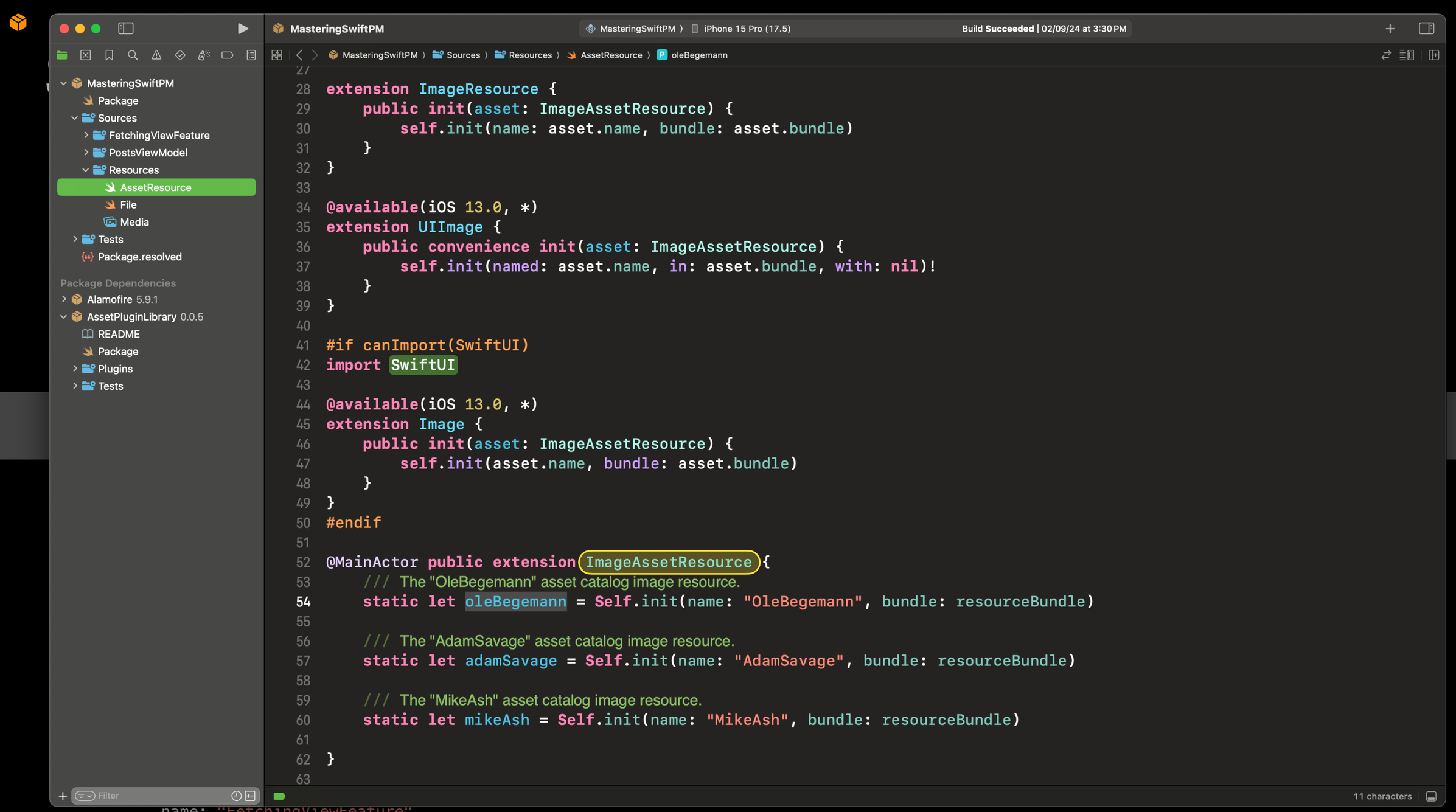

```swift
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```



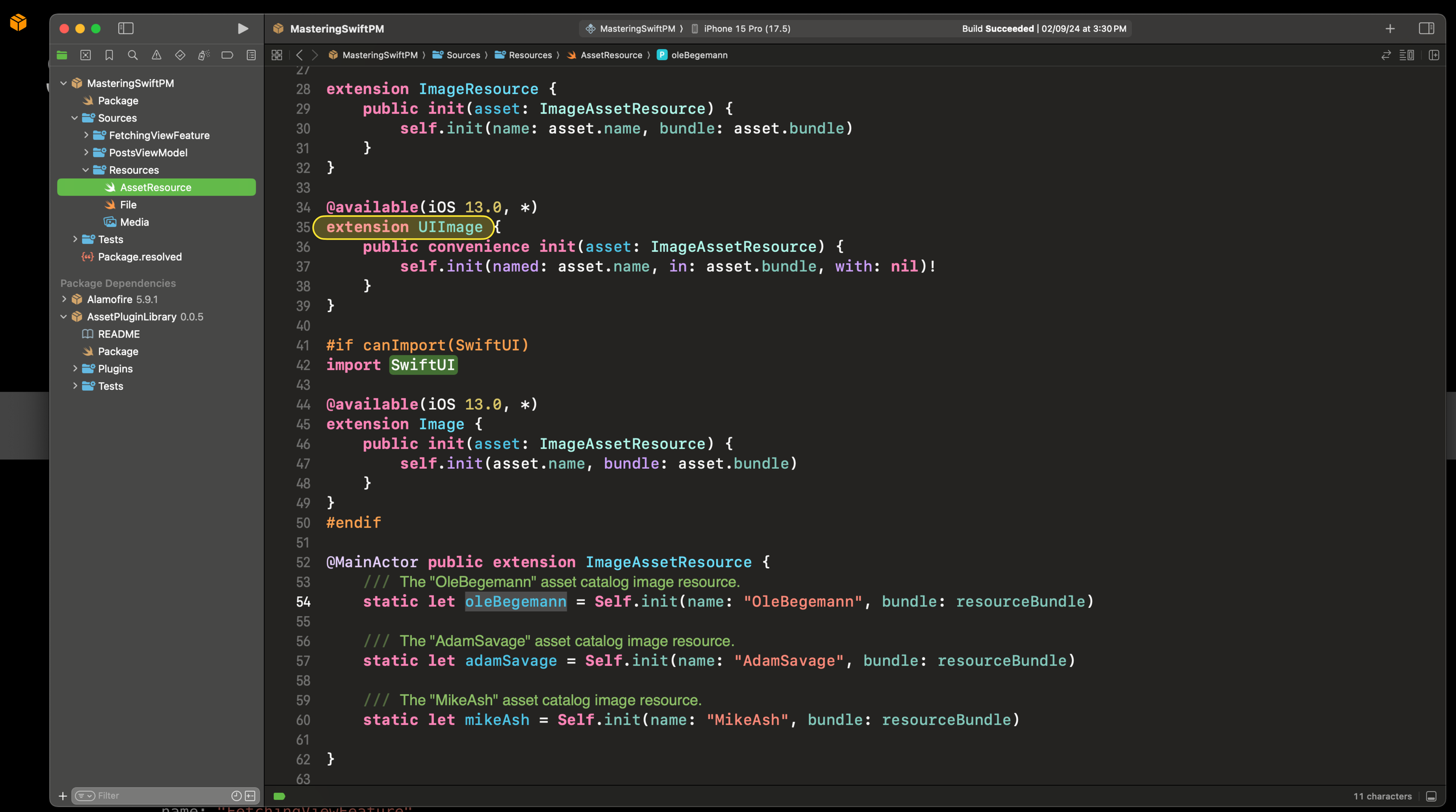

```
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```



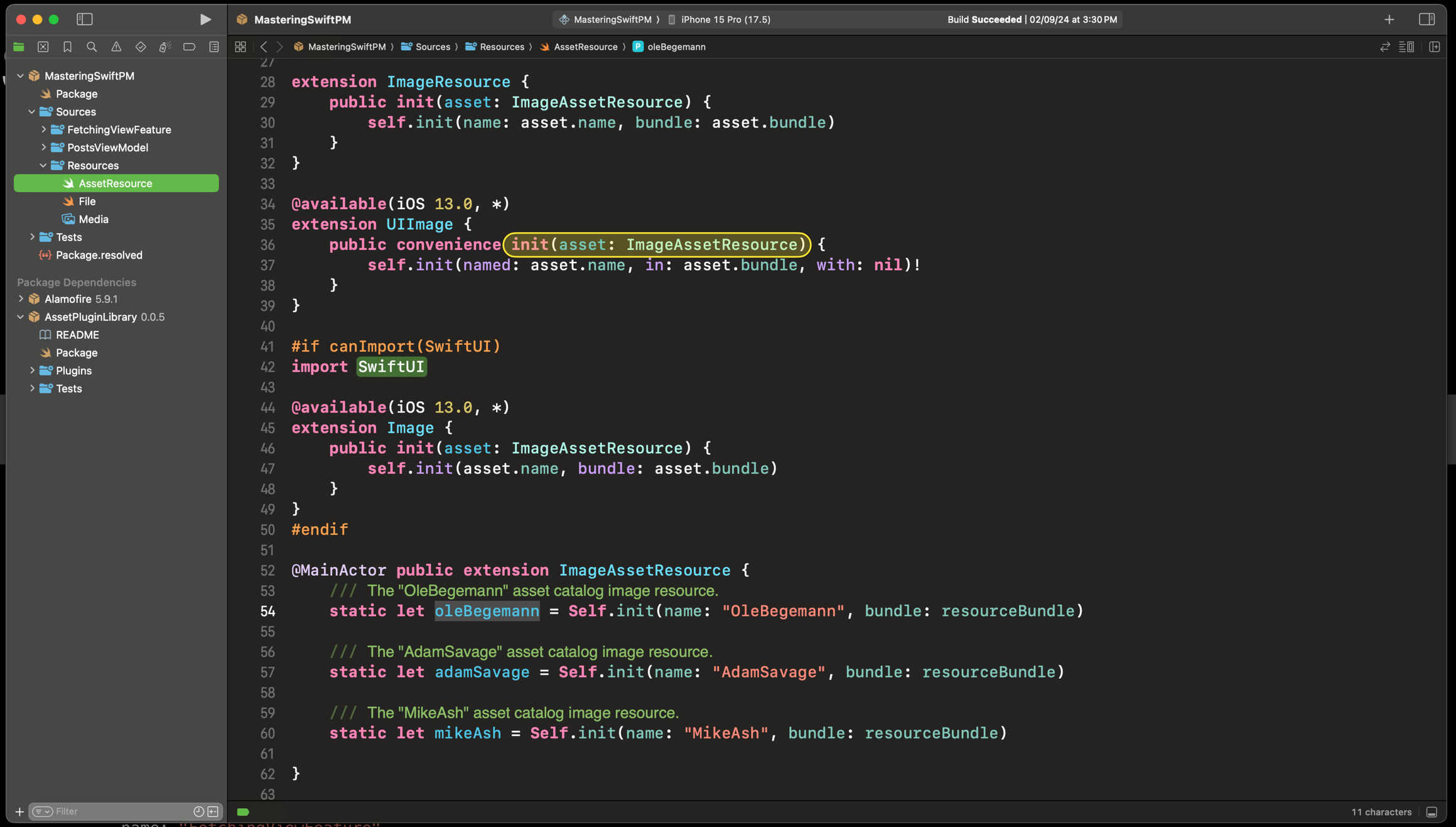

```swift
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```



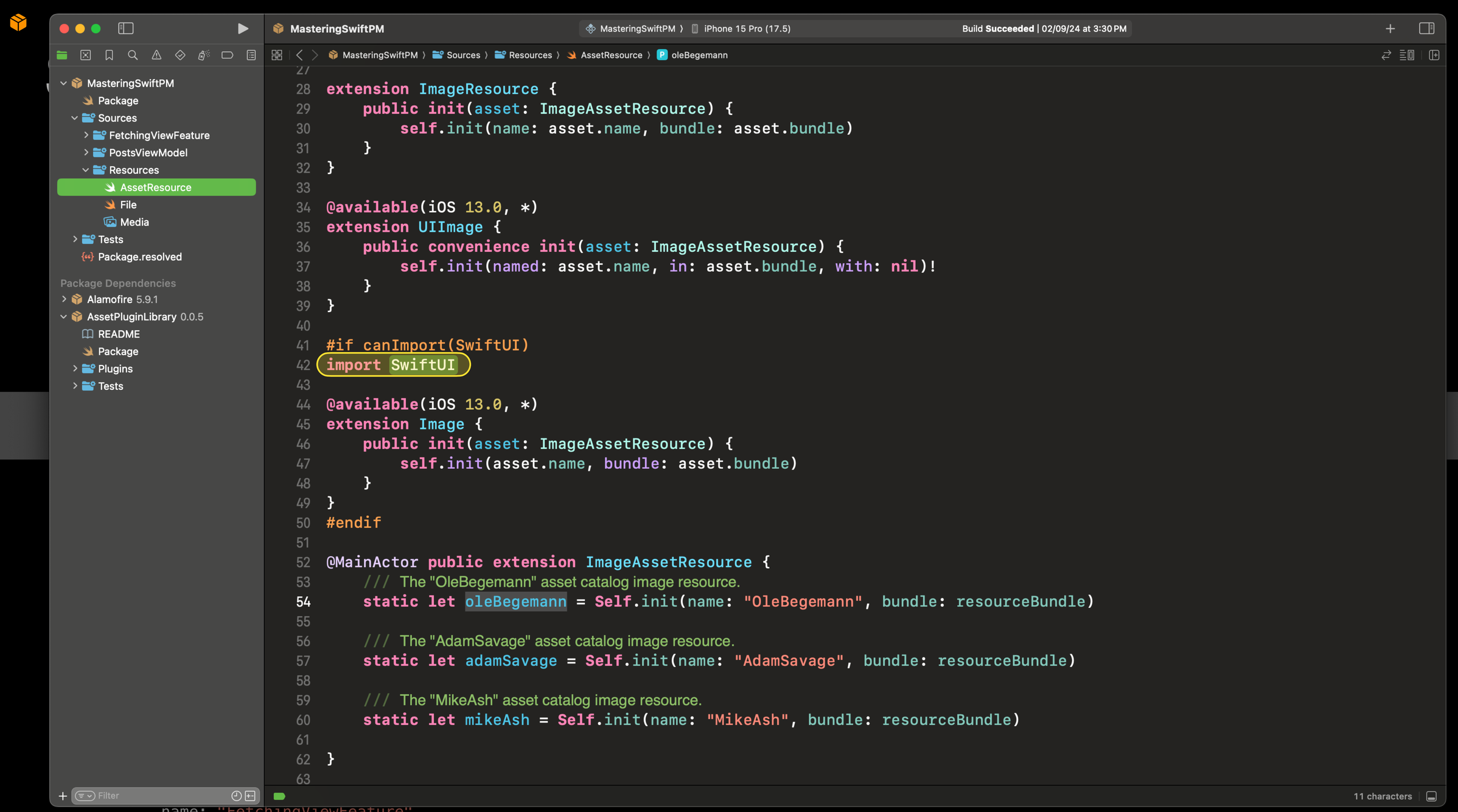
name: "FetchingViewFeature",
dependencies: [.product(name: "Alamofire", package: "Alamofire")]



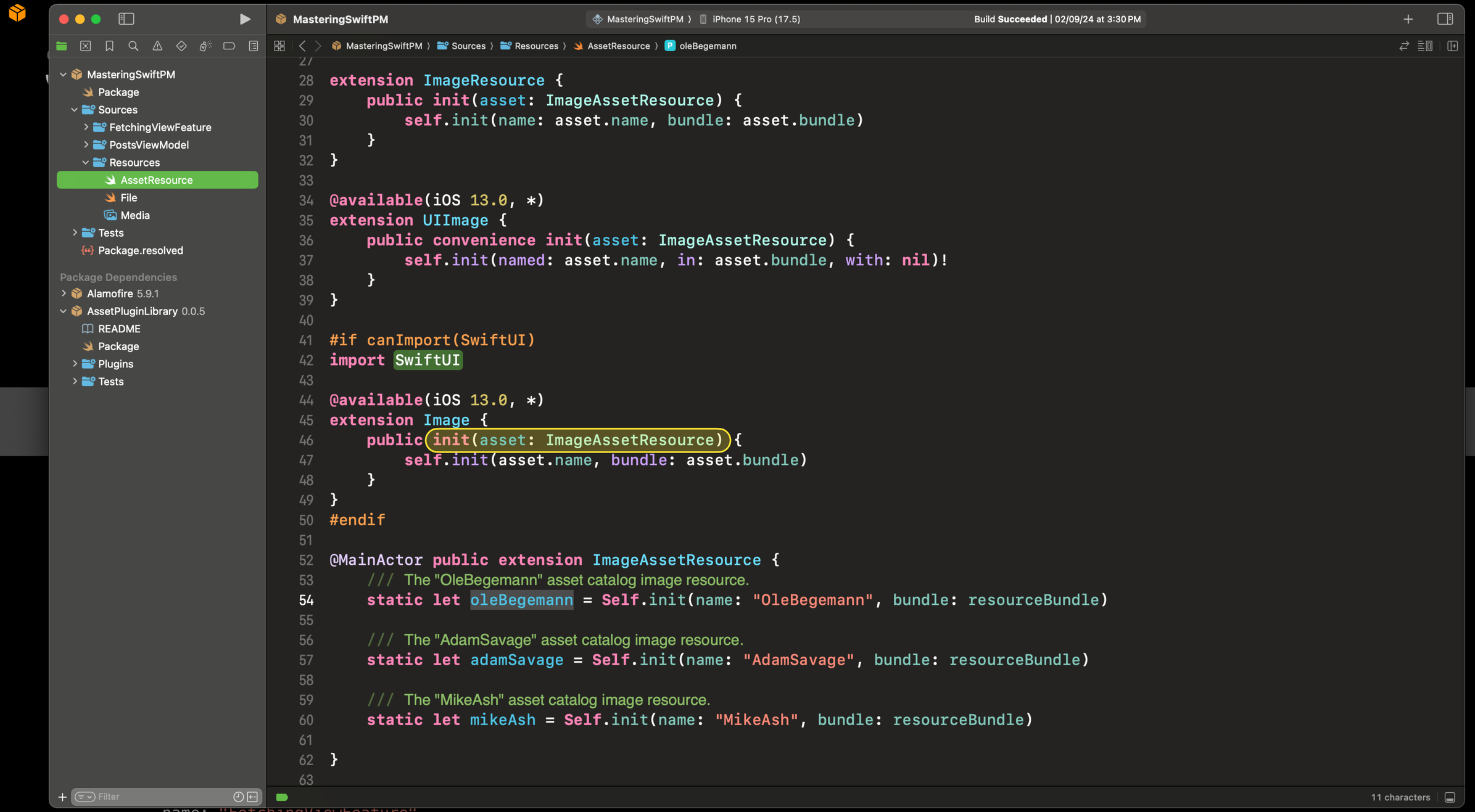

```swift
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```




```swift
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```




```
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```




```swift
extension ImageResource {
    public init(asset: ImageAssetResource) {
        self.init(name: asset.name, bundle: asset.bundle)
    }
}

@available(iOS 13.0, *)
extension UIImage {
    public convenience init(asset: ImageAssetResource) {
        self.init(named: asset.name, in: asset.bundle, with: nil)!
    }
}

#if canImport(SwiftUI)
import SwiftUI

@available(iOS 13.0, *)
extension Image {
    public init(asset: ImageAssetResource) {
        self.init(asset.name, bundle: asset.bundle)
    }
}
#endif

@MainActor public extension ImageAssetResource {
    /// The "OleBegemann" asset catalog image resource.
    static let oleBegemann = Self.init(name: "OleBegemann", bundle: resourceBundle)

    /// The "AdamSavage" asset catalog image resource.
    static let adamSavage = Self.init(name: "AdamSavage", bundle: resourceBundle)

    /// The "MikeAsh" asset catalog image resource.
    static let mikeAsh = Self.init(name: "MikeAsh", bundle: resourceBundle)

}
```

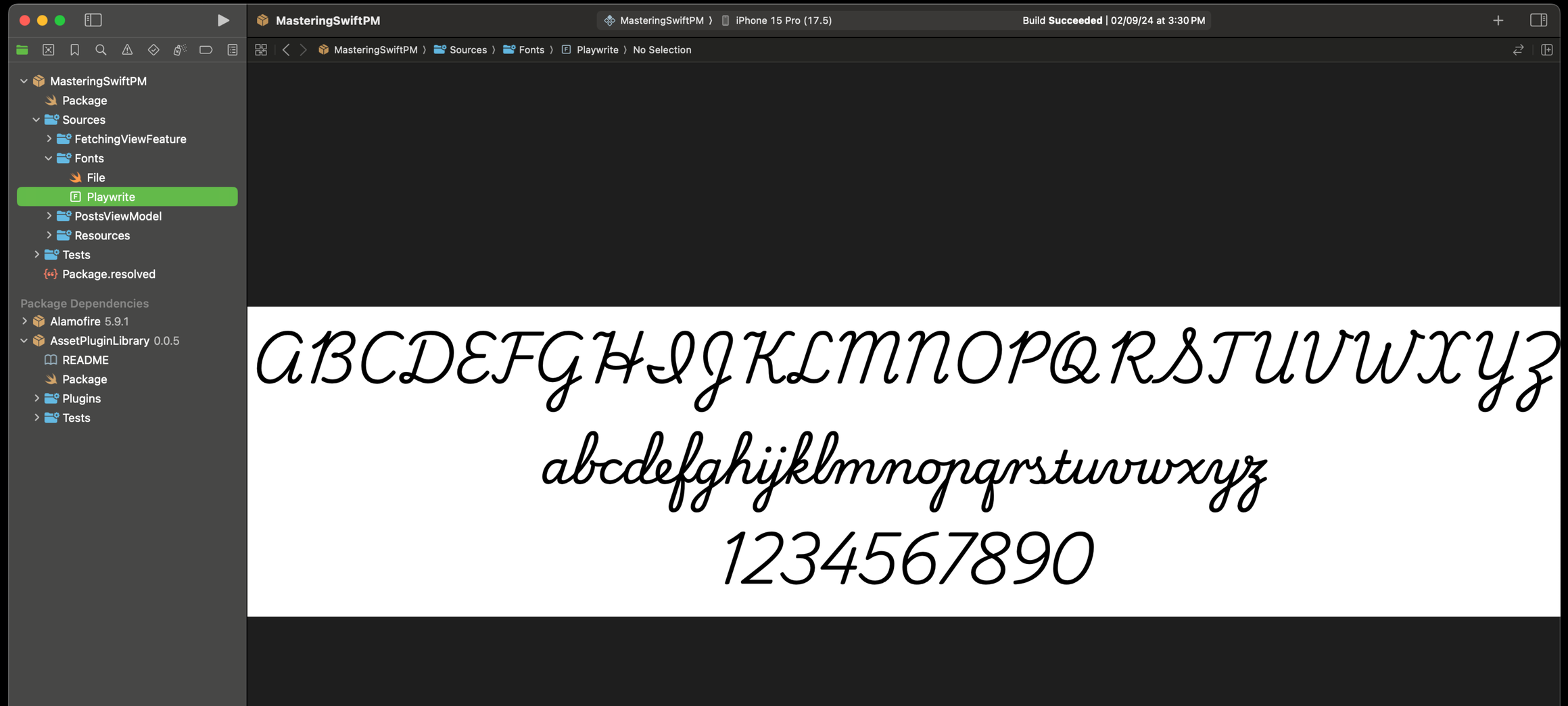
Sharing Resources
With Other Modules

- Assets
  - Images
  - Colors
- Fonts
- Core Data
- Other Resources

Font.ttf

Sharing Resources

Fonts



Sharing Resources

Fonts

- Add fonts file to the target directory
- Declare explicit intent for the .ttf font files
  - Creates a Module bundle instance
- Register the font
  - Using `CTFontManagerRegisterGraphicsFont`
  - Using `CTFontManagerRegisterFontsForURL`
- Get the font family name
- Create (typed*) font weights to use with

Sharing Resources

```swift
extension UIFont {
    static func registerFont(bundle: Bundle, fontName: String, fontExtension: String) -> Bool {
        guard let fontURL = bundle.url(forResource: fontName, withExtension: fontExtension) else {
            print("Couldn't find font \(fontName)")
            return false
        }

        // NOTE: -
        // Previously we were using `CTFontManagerRegisterGraphicsFont` which takes CTFont.
        // but this was not working in the WidgetKit extension.
        // So as per the this thread,
        // https://developer.apple.com/forums/thread/659332
        // Apple engineer suggested to use `CTFontManagerRegisterFontsForURL`.
        // Here scope like `CTFontManagerScope.persistent` is not supported in this method.
        // `CTFontManagerScope.none` should used for un-registering method only.
        // `CTFontManagerScope.process` works for both App and WidgetKit extension Target.

        var error: Unmanaged<CFError>?
        let success = CTFontManagerRegisterFontsForURL(fontURL as CFURL, .process, &error)
        guard success else {
            print(
                """
                Error registering font: \(fontName). Maybe it was already registered.\
                \(error.map { " \($0.takeUnretainedValue().localizedDescription)" } ?? "")
                """
            )
            return false
        }

        return true
    }
}
```

Sharing Resources

```
extension UIFont {
    // Should be called from test env.
    static func unregisterFont(bundle: Bundle, fontName: String, fontExtension: String) -> Bool {
        guard let fontURL = bundle.url(forResource: fontName, withExtension: fontExtension) else {
            print("Couldn't find font \(fontName)")
            return false
        }
        guard let fontDataProvider = CGDataProvider(url: fontURL as CFURL) else {
            print("Couldn't load data from the font \(fontName)")
            return false
        }
        guard let font = CGFont(fontDataProvider) else {
            print(“Couldn't create font from data")
            return false
        }

        var error: Unmanaged<CFError>?
        let success = CTFontManagerUnregisterGraphicsFont(font, &error)
        guard success else {
            print(
                """
                Error unregistering font: \(fontName). Maybe it was already registered.\
                \(error.map { " \($0.takeUnretainedValue().localizedDescription)" } ?? "")
                """
            )
            return false
        }
       return true
    }
}
```

Sharing Resources

```
extension UIFont {
    static func allFontNames() -> [String] {
        let familyNames = UIFont.familyNames.sorted()
        return familyNames.flatMap { familyName in
            let fontName = UIFont.fontNames(forFamilyName: familyName)
            return fontName
        }
    }
}
```

Sharing Resources

```swift
extension UIFont {
    static func allFontNames() -> [String] {
        let familyNames = UIFont.familyNames.sorted()
        return familyNames.flatMap { familyName in
            let fontName = UIFont.fontNames(forFamilyName: familyName)
            return fontName
        }
    }
}
```

```
["AcademyEngravedLetPlain", "AlNile", "AlNile-Bold", "AmericanTypewriter", "AmericanTypewriter-Light", "AmericanTypewriter-Semibold",
"AmericanTypewriter-Bold", "AmericanTypewriter-Condensed", "AmericanTypewriter-CondensedLight", "AmericanTypewriter-CondensedBold", "AppleColorEmoji",
"AppleSDGothicNeo-Regular", "AppleSDGothicNeo-Thin", "AppleSDGothicNeo-UltraLight", "AppleSDGothicNeo-Light", "AppleSDGothicNeo-Medium",
"AppleSDGothicNeo-SemiBold", "AppleSDGothicNeo-Bold", "AppleSymbols", "ArialMT", "Arial-ItalicMT", "Arial-BoldMT", "Arial-BoldItalicMT", "ArialHebrew",
"ArialHebrew-Light", "ArialHebrew-Bold", "ArialRoundedMTBold", "Avenir-Book", "Avenir-Roman", "Avenir-BookOblique", "Avenir-Oblique", "Avenir-Light",
"Avenir-LightOblique", "Avenir-Medium", "Avenir-MediumOblique", "Avenir-Heavy", "Avenir-HeavyOblique", "Avenir-Black", "Avenir-BlackOblique",
"AvenirNext-Regular", "AvenirNext-Italic", "AvenirNext-UltraLight", "AvenirNext-UltraLightItalic", "AvenirNext-Medium", "AvenirNext-MediumItalic",
"AvenirNext-DemiBold", "AvenirNext-DemiBoldItalic", "AvenirNext-Bold", "AvenirNext-BoldItalic", "AvenirNext-Heavy", "AvenirNext-HeavyItalic",
"AvenirNextCondensed-Regular", "AvenirNextCondensed-Italic", "AvenirNextCondensed-UltraLight", "AvenirNextCondensed-UltraLightItalic",
"AvenirNextCondensed-Medium", "AvenirNextCondensed-MediumItalic", "AvenirNextCondensed-DemiBold", "AvenirNextCondensed-DemiBoldItalic",
"AvenirNextCondensed-Bold", "AvenirNextCondensed-BoldItalic", "AvenirNextCondensed-Heavy", "AvenirNextCondensed-HeavyItalic", "Baskerville",
"Baskerville-Italic", "Baskerville-SemiBold", "Baskerville-SemiBoldItalic", "Baskerville-Bold", "Baskerville-BoldItalic", "BodoniSvtyTwoITCTT-Book",
"BodoniSvtyTwoITCTT-BookIta", "BodoniSvtyTwoITCTT-Bold", "BodoniSvtyTwoOSITCTT-Book", "BodoniSvtyTwoOSITCTT-BookIt", "BodoniSvtyTwoOSITCTT-Bold",
"BodoniSvtyTwoSCITCTT-Book", "BodoniOrnamentsITCTT", "BradleyHandITCTT-Bold", "ChalkboardSE-Regular", "ChalkboardSE-Light", "ChalkboardSE-Bold",
"Chalkduster", "Charter-Roman", "Charter-Italic", "Charter-Bold", "Charter-BoldItalic", "Charter-Black", "Charter-BlackItalic", "Cochin", "Cochin-
Italic", "Cochin-Bold", "Cochin-BoldItalic", "Copperplate", "Copperplate-Light", "Copperplate-Bold", "CourierNewPSMT", "CourierNewPS-ItalicMT",
"CourierNewPS-BoldMT", "CourierNewPS-BoldItalicMT", "DINAlternate-Bold", "DINCondensed-Bold", "Damascus", "DamascusLight", "DamascusMedium",
"DamascusSemiBold", "DamascusBold", "DevanagariSangamMN", "DevanagariSangamMN-Bold", "Didot", "Didot-Italic", "Didot-Bold", "EuphemiaUCAS",
"EuphemiaUCAS-Italic", "EuphemiaUCAS-Bold", "Farah", "Futura-Medium", "Futura-MediumItalic", "Futura-Bold", "Futura-CondensedMedium", "Futura-
CondensedExtraBold", "Galvji", "Galvji-Bold", "GeezaPro", "GeezaPro-Bold", "Georgia", "Georgia-Italic", "Georgia-Bold", "Georgia-BoldItalic",
"GillSans", "GillSans-Italic", "GillSans-Light", "GillSans-LightItalic", "GillSans-SemiBold", "GillSans-SemiBoldItalic", "GillSans-Bold", "GillSans-
BoldItalic", "GillSans-UltraBold", "GranthaSangamMN-Regular", "GranthaSangamMN-Bold", "Helvetica", "Helvetica-Oblique", "Helvetica-Light", "Helvetica-
LightOblique", "Helvetica-Bold", "Helvetica-BoldOblique", "HelveticaNeue", "HelveticaNeue-Italic", "HelveticaNeue-UltraLight", "HelveticaNeue-
UltraLightItalic", "HelveticaNeue-Thin", "HelveticaNeue-ThinItalic", "HelveticaNeue-Light", "HelveticaNeue-LightItalic", "HelveticaNeue-Medium",
"HelveticaNeue-MediumItalic", "HelveticaNeue-Bold", "HelveticaNeue-BoldItalic", "HelveticaNeue-CondensedBold", "HelveticaNeue-CondensedBlack",
"HiraMaruProN-W4", "HiraMinProN-W3", "HiraMinProN-W6", "HiraginoSans-W3", "HiraginoSans-W4", "HiraginoSans-W5", "HiraginoSans-W6", "HiraginoSans-W7",
```

["AcademyEngravedLetPlain", "AlNile", "AlNile-Bold", "AmericanTypewriter", "AmericanTypewriter-Light", "AmericanTypewriter-Semibold", "AmericanTypewriter-Bold", "AmericanTypewriter-Condensed", "AmericanTypewriter-CondensedLight", "AmericanTypewriter-CondensedBold", "AppleColorEmoji", "AppleSDGothicNeo-Regular", "AppleSDGothicNeo-Thin", "AppleSDGothicNeo-UltraLight", "AppleSDGothicNeo-Light", "AppleSDGothicNeo-Medium", "AppleSDGothicNeo-SemiBold", "AppleSDGothicNeo-Bold", "AppleSymbols", "ArialMT", "Arial-ItalicMT", "Arial-BoldMT", "Arial-BoldItalicMT", "ArialHebrew", "ArialHebrew-Light", "ArialHebrew-Bold", "ArialRoundedMTBold", "Avenir-Book", "Avenir-Roman", "Avenir-BookOblique", "Avenir-Oblique", "Avenir-Light", "Avenir-LightOblique", "Avenir-Medium", "Avenir-MediumOblique", "Avenir-Heavy", "Avenir-HeavyOblique", "Avenir-Black", "Avenir-BlackOblique", "AvenirNext-Regular", "AvenirNext-Italic", "AvenirNext-UltraLight", "AvenirNext-UltraLightItalic", "AvenirNext-Medium", "AvenirNext-MediumItalic", "AvenirNext-DemiBold", "AvenirNext-DemiBoldItalic", "AvenirNext-Bold", "AvenirNext-BoldItalic", "AvenirNext-Heavy", "AvenirNext-HeavyItalic", "AvenirNextCondensed-Regular", "AvenirNextCondensed-Italic", "AvenirNextCondensed-UltraLight", "AvenirNextCondensed-UltraLightItalic", "AvenirNextCondensed-Medium", "AvenirNextCondensed-MediumItalic", "AvenirNextCondensed-DemiBold", "AvenirNextCondensed-DemiBoldItalic", "AvenirNextCondensed-Bold", "AvenirNextCondensed-BoldItalic", "AvenirNextCondensed-Heavy", "AvenirNextCondensed-HeavyItalic", "Baskerville", "Baskerville-Italic", "Baskerville-SemiBold", "Baskerville-SemiBoldItalic", "Baskerville-Bold", "Baskerville-BoldItalic", "BodoniSvtyTwoITCTT-Book", "BodoniSvtyTwoITCTT-BookIta", "BodoniSvtyTwoITCTT-Bold", "BodoniSvtyTwoOSITCTT-Book", "BodoniSvtyTwoOSITCTT-BookIt", "BodoniSvtyTwoOSITCTT-Bold", "BodoniSvtyTwoSCITCTT-Book", "BodoniOrnamentsITCTT", "BradleyHandITCTT-Bold", "ChalkboardSE-Regular", "ChalkboardSE-Light", "ChalkboardSE-Bold", "Chalkduster", "Charter-Roman", "Charter-Italic", "Charter-Bold", "Charter-BoldItalic", "Charter-Black", "Charter-BlackItalic", "Cochin", "Cochin-Italic", "Cochin-Bold", "Cochin-BoldItalic", "Copperplate", "Copperplate-Light", "Copperplate-Bold", "CourierNewPSMT", "CourierNewPS-ItalicMT", "CourierNewPS-BoldMT", "CourierNewPS-BoldItalicMT", "DINAlternate-Bold", "DINCondensed-Bold", "Damascus", "DamascusLight", "DamascusMedium", "DamascusSemiBold", "DamascusBold", "DevanagariSangamMN", "DevanagariSangamMN-Bold", "Didot", "Didot-Italic", "Didot-Bold", "EuphemiaUCAS", "EuphemiaUCAS-Italic", "EuphemiaUCAS-Bold", "Farah", "Futura-Medium", "Futura-MediumItalic", "Futura-Bold", "Futura-CondensedMedium", "Futura-CondensedExtraBold", "Galvji", "Galvji-Bold", "GeezaPro", "GeezaPro-Bold", "Georgia", "Georgia-Italic", "Georgia-Bold", "Georgia-BoldItalic", "GillSans", "GillSans-Italic", "GillSans-Light", "GillSans-LightItalic", "GillSans-SemiBold", "GillSans-SemiBoldItalic", "GillSans-Bold", "GillSans-BoldItalic", "GillSans-UltraBold", "GranthaSangamMN-Regular", "GranthaSangamMN-Bold", "Helvetica", "Helvetica-Oblique", "Helvetica-Light", "Helvetica-LightOblique", "Helvetica-Bold", "Helvetica-BoldOblique", "HelveticaNeue", "HelveticaNeue-Italic", "HelveticaNeue-UltraLight", "HelveticaNeue-UltraLightItalic", "HelveticaNeue-Thin", "HelveticaNeue-ThinItalic", "HelveticaNeue-Light", "HelveticaNeue-LightItalic", "HelveticaNeue-Medium", "HelveticaNeue-MediumItalic", "HelveticaNeue-Bold", "HelveticaNeue-BoldItalic", "HelveticaNeue-CondensedBold", "HelveticaNeue-CondensedBlack", "HiraMaruProN-W4", "HiraMinProN-W3", "HiraMinProN-W6", "HiraginoSans-W3", "HiraginoSans-W4", "HiraginoSans-W5", "HiraginoSans-W6", "HiraginoSans-W7", "HiraginoSans-W8", "HoeflerText-Regular", "HoeflerText-Italic", "HoeflerText-Black", "HoeflerText-BlackItalic", "Impact", "Kailasa", "Kailasa-Bold", "Kefa-Regular", "KhmerSangamMN", "KohinoorBangla-Regular", "KohinoorBangla-Light", "KohinoorBangla-Semibold", "KohinoorDevanagari-Regular", "KohinoorDevanagari-Light", "KohinoorDevanagari-Semibold", "KohinoorGujarati-Regular", "KohinoorGujarati-Light", "KohinoorGujarati-Bold", "KohinoorTelugu-Regular", "KohinoorTelugu-Light", "KohinoorTelugu-Medium", "LaoSangamMN", "MalayalamSangamMN", "MalayalamSangamMN-Bold", "MarkerFelt-Thin", "MarkerFelt-Wide", "Menlo-Regular", "Menlo-Italic", "Menlo-Bold", "Menlo-BoldItalic", "DiwanMishafi", "MuktaMahee-Regular", "MuktaMahee-Light", "MuktaMahee-Bold", "MyanmarSangamMN", "MyanmarSangamMN-Bold", "Noteworthy-Light", "Noteworthy-Bold", "NotoNastaliqUrdu", "NotoNastaliqUrdu-Bold", "NotoSansKannada-Regular", "NotoSansKannada-Light", "NotoSansKannada-Bold", "NotoSansMyanmar-Regular", "NotoSansMyanmar-Light", "NotoSansMyanmar-Bold", "NotoSansOriya", "NotoSansOriya-Bold", "NotoSansSyriac-Regular", "NotoSansSyriac-Regular_Thin", "NotoSansSyriac-Regular_ExtraLight", "NotoSansSyriac-Regular_Light", "NotoSansSyriac-Regular_Medium", "NotoSansSyriac-Regular_SemiBold", "NotoSansSyriac-Regular_Bold", "NotoSansSyriac-Regular_ExtraBold", "NotoSansSyriac-Regular_Black", "Optima-Regular", "Optima-Italic", "Optima-Bold", "Optima-BoldItalic", "Optima-ExtraBlack", "Palatino-Roman", "Palatino-Italic", "Palatino-Bold", "Palatino-BoldItalic", "Papyrus", "Papyrus-Condensed", "PartyLetPlain", "PingFangHK-Regular", "PingFangHK-Ultralight", "PingFangHK-Thin", "PingFangHK-Light", "PingFangHK-Medium", "PingFangHK-Semibold", "PingFangMO-Regular", "PingFangMO-Ultralight", "PingFangMO-Thin", "PingFangMO-Light", "PingFangMO-Medium", "PingFangMO-Semibold", "PingFangSC-Regular", "PingFangSC-Ultralight", "PingFangSC-Thin", "PingFangSC-Light", "PingFangSC-Medium", "PingFangSC-Semibold", "PingFangTC-Regular", "PingFangTC-Ultralight", "PingFangTC-Thin", "PingFangTC-Light", "PingFangTC-Medium", "PingFangTC-Semibold", "PlaywriteCU-Regular", "PlaywriteCU-Regular_Thin", "PlaywriteCU-Regular_ExtraLight", "PlaywriteCU-Regular_Light", "Rockwell-Regular", "Rockwell-Italic", "Rockwell-Bold", "Rockwell-BoldItalic", "STIXTwoMath-Regular", "STIXTwoText", "STIXTwoText-Italic", "STIXTwoText_Medium", "STIXTwoText-Italic_Medium-Italic", "STIXTwoText_SemiBold", "STIXTwoText-Italic_SemiBold-Italic", "STIXTwoText_Bold", "STIXTwoText-Italic_Bold-Italic", "SavoyeLetPlain", "SinhalaSangamMN", "SinhalaSangamMN-Bold", "SnellRoundhand", "SnellRoundhand-Bold", "SnellRoundhand-Black", "Symbol", "TamilSangamMN", "TamilSangamMN-Bold", "Thonburi", "Thonburi-Light", "Thonburi-Bold", "TimesNewRomanPSMT", "TimesNewRomanPS-ItalicMT", "TimesNewRomanPS-BoldMT", "TimesNewRomanPS-BoldItalicMT", "TrebuchetMS", "TrebuchetMS-Italic", "TrebuchetMS-Bold", "Trebuchet-BoldItalic", "Verdana", "Verdana-Italic", "Verdana-Bold", "Verdana-BoldItalic", "ZapfDingbatsITC", "Zapfino"]

CondensedExtraBold", "Galvji", "Galvji-Bold", "GeezaPro", "GeezaPro-Bold", "Georgia", "Georgia-Italic", "Georgia-Bold", "Georgia-BoldItalic", "GillSans", "GillSans-Italic", "GillSans-Light", "GillSans-LightItalic", "GillSans-SemiBold", "GillSans-SemiBoldItalic", "GillSans-Bold", "GillSans-BoldItalic", "GillSans-UltraBold", "GranthaSangamMN-Regular", "GranthaSangamMN-Bold", "Helvetica", "Helvetica-Oblique", "Helvetica-Light", "Helvetica-LightOblique", "Helvetica-Bold", "Helvetica-BoldOblique", "HelveticaNeue", "HelveticaNeue-Italic", "HelveticaNeue-UltraLight", "HelveticaNeue-UltraLightItalic", "HelveticaNeue-Thin", "HelveticaNeue-ThinItalic", "HelveticaNeue-Light", "HelveticaNeue-LightItalic", "HelveticaNeue-Medium", "HelveticaNeue-MediumItalic", "HelveticaNeue-Bold", "HelveticaNeue-BoldItalic", "HelveticaNeue-CondensedBold", "HelveticaNeue-CondensedBlack", "HiraMaruProN-W4", "HiraMinProN-W3", "HiraMinProN-W6", "HiraginoSans-W3", "HiraginoSans-W4", "HiraginoSans-W5", "HiraginoSans-W6", "HiraginoSans-W7", "HiraginoSans-W8", "HoeflerText-Regular", "HoeflerText-Italic", "HoeflerText-Black", "HoeflerText-BlackItalic", "Impact", "Kailasa", "Kailasa-Bold", "Kefa-Regular", "KhmerSangamMN", "KohinoorBangla-Regular", "KohinoorBangla-Light", "KohinoorBangla-Semibold", "KohinoorDevanagari-Regular", "KohinoorDevanagari-Light", "KohinoorDevanagari-Semibold", "KohinoorGujarati-Regular", "KohinoorGujarati-Light", "KohinoorGujarati-Bold", "KohinoorTelugu-Regular", "KohinoorTelugu-Light", "KohinoorTelugu-Medium", "LaoSangamMN", "MalayalamSangamMN", "MalayalamSangamMN-Bold", "MarkerFelt-Thin", "MarkerFelt-Wide", "Menlo-Regular", "Menlo-Italic", "Menlo-Bold", "Menlo-BoldItalic", "DiwanMishafi", "MuktaMahee-Regular", "MuktaMahee-Light", "MuktaMahee-Bold", "MyanmarSangamMN", "MyanmarSangamMN-Bold", "Noteworthy-Light", "Noteworthy-Bold", "NotoNastaliqUrdu", "NotoNastaliqUrdu-Bold", "NotoSansKannada-Regular", "NotoSansKannada-Light", "NotoSansKannada-Bold", "NotoSansMyanmar-Regular", "NotoSansMyanmar-Light", "NotoSansMyanmar-Bold", "NotoSansOriya", "NotoSansOriya-Bold", "NotoSansSyriac-Regular", "NotoSansSyriac-Regular_Thin", "NotoSansSyriac-Regular_ExtraLight", "NotoSansSyriac-Regular_Light", "NotoSansSyriac-Regular_Medium", "NotoSansSyriac-Regular_SemiBold", "NotoSansSyriac-Regular_Bold", "NotoSansSyriac-Regular_ExtraBold", "NotoSansSyriac-Regular_Black", "Optima-Regular", "Optima-Italic", "Optima-Bold", "Optima-BoldItalic", "Optima-ExtraBlack", "Palatino-Roman", "Palatino-Italic", "Palatino-Bold", "Palatino-BoldItalic", "Papyrus", "Papyrus-Condensed", "PartyLetPlain", "PingFangHK-Regular", "PingFangHK-Ultralight", "PingFangHK-Thin", "PingFangHK-Light", "PingFangHK-Medium", "PingFangHK-Semibold", "PingFangMO-Regular", "PingFangMO-Ultralight", "PingFangMO-Thin", "PingFangMO-Light", "PingFangMO-Medium", "PingFangMO-Semibold", "PingFangSC-Regular", "PingFangSC-Ultralight", "PingFangSC-Thin", "PingFangSC-Light", "PingFangSC-Medium", "PingFangSC-Semibold", "PingFangTC-Regular", "PingFangTC-Ultralight", "PingFangTC-Thin", "PingFangTC-Light", "PingFangTC-Medium", "PingFangTC-Semibold", "PlaywriteCU-Regular", "PlaywriteCU-Regular_Thin", "PlaywriteCU-Regular_ExtraLight", "PlaywriteCU-Regular_Light", "Rockwell-Regular", "Rockwell-Italic", "Rockwell-Bold", "Rockwell-BoldItalic", "STIXTwoMath-Regular", "STIXTwoText", "STIXTwoText-Italic", "STIXTwoText_Medium", "STIXTwoText-Italic_Medium-Italic", "STIXTwoText_SemiBold", "STIXTwoText-Italic_SemiBold-Italic", "STIXTwoText_Bold", "STIXTwoText-Italic_Bold-Italic", "SavoyeLetPlain", "SinhalaSangamMN", "SinhalaSangamMN-Bold", "SnellRoundhand", "SnellRoundhand-Bold", "SnellRoundhand-Black", "Symbol", "TamilSangamMN", "TamilSangamMN-Bold", "Thonburi", "Thonburi-Light", "Thonburi-Bold", "TimesNewRomanPSMT", "TimesNewRomanPS-ItalicMT", "TimesNewRomanPS-BoldMT", "TimesNewRomanPS-BoldItalicMT", "TrebuchetMS", "TrebuchetMS-Italic", "TrebuchetMS-Bold", "Trebuchet-BoldItalic", "Verdana", "Verdana-Italic", "Verdana-Bold", "Verdana-BoldItalic", "ZapfDingbatsITC", "Zapfino"]

Sharing Resources

```
extension UIFont {
    static func allFontNames() -> [String] {
        let familyNames = UIFont.familyNames.sorted()
        return familyNames.flatMap { familyName in
            let fontName = UIFont.fontNames(forFamilyName: familyName)
            return fontName
        }
    }
}
```

```
PlaywriteCU-Regular
PlaywriteCU-Regular_Thin
PlaywriteCU-Regular_ExtraLight
PlaywriteCU-Regular_Light
```

Sharing Resources

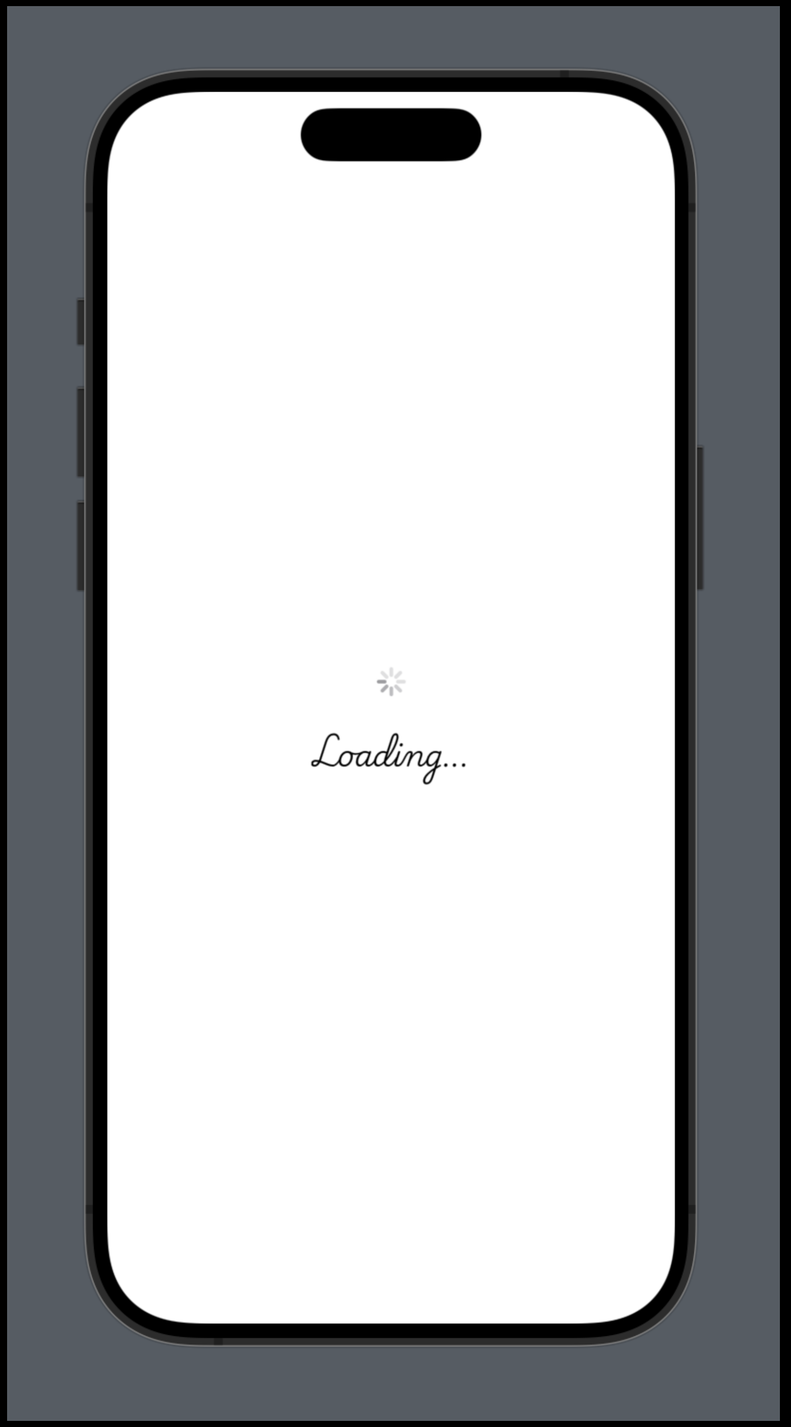
```
public enum AppFont: CaseIterable {
    case playWrite

    var fontName: String {
        switch self { case .playWrite: “Playwrite" }
    }

    var fontExtension: String {
        switch self { case .playWrite: “.ttf" }
    }

    public enum PlayWriteWeight: String {
        case regular
        case thin
        case extraLight
        case light

        var name: String {
            switch self {
            case .regular:
                "PlaywriteCU-Regular"
            case .thin:
                "PlaywriteCU-Regular_Thin"
            case .extraLight:
                "PlaywriteCU-Regular_ExtraLight"
            case .light:
                "PlaywriteCU-Regular_Light"
            }
        }
    }
}
```

Sharing Resources

```
extension UIFont {
    public static func register(_ font: AppFont) -> Bool {
        Self.registerFont(bundle: .module, fontName: font.fontName, fontExtension: font.fontExtension)
    }

    public static func registerAll() {
        AppFont.allCases.forEach {
            _ = Self.register($0)
        }
    }
}
```

Sharing Resources

```swift
import SwiftUI
import Alamofire
import Fonts

public struct FetchingView: View {
    public var body: some View {
        VStack {
            ProgressView()
            Text("Loading...")
                .font(.playwrite(weight: .light, size: 20))
        }
    }
}

#Preview {
    _ = UIFont.register(.playWrite)
    return FetchingView()
}
```



Sharing Resources
With Other Modules

- Assets
  - Images
  - Colors
- Fonts
- Core Data
- Other Resources

CoreData

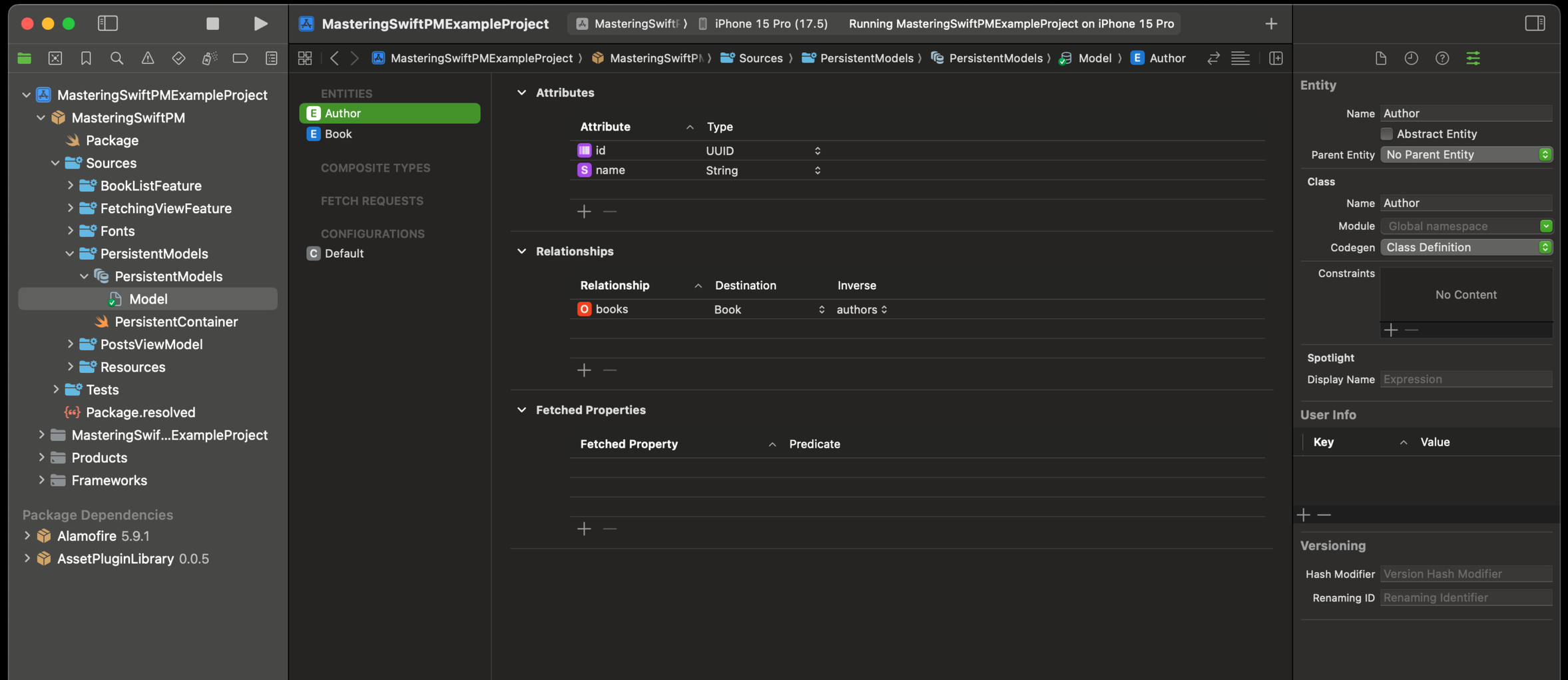
Sharing Resources

Core Data

- New directory (folder) in Sources directory
- Empty .swift file to make SwiftPM happy
- New core Data Model (.xcdatamodeld) file
- Restart Xcode to add new entities
- Add few Entities
- Set codegen to “Class Definition”
- Set module to “Global namespace”
- Use swift file to Create a Persistent Container instance

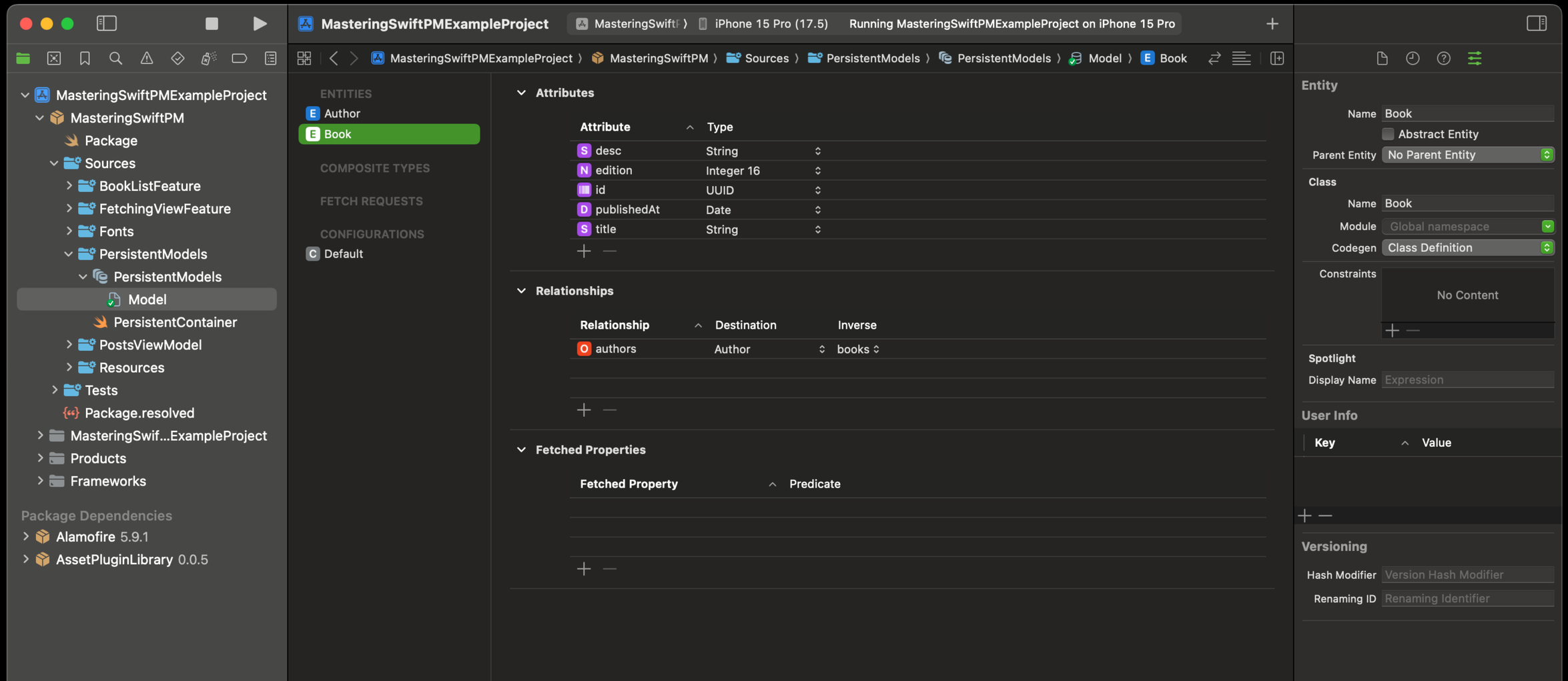
Sharing Resources

Core Data



Sharing Resources

Core Data



Sharing Resources

PersistentModelStore.swift

```swift
open class PersistentContainer: NSPersistentContainer {

}

public enum PersistentModelStore {
    public static let container: PersistentContainer = {
        guard let modelURL = Bundle.module.url(forResource: "PersistentModels", withExtension: "momd") else { fatalError() }
        guard let model = NSManagedObjectModel(contentsOf: modelURL) else { fatalError() }
        let container = PersistentContainer(name:"PersistentModels", managedObjectModel: model)
        container.loadPersistentStores(completionHandler: { (storeDescription, error) in
            if let error = error as NSError? {
                print("Unresolved error \(error), \(error.userInfo)")
            }
        })
        return container
    }()
}
```

Sharing Resources

BookListViewModel.swift

```swift
import Foundation
import PersistentModels
import CoreData
import Combine

@MainActor @Observable public class BookListViewModel {
    var bookLists: [Book] = []
    private var cancellables: Set<AnyCancellable> = []

    public init(bookLists: [Book] = []) {
        self.bookLists = bookLists
        performFetchRequest()
        observeChanges()
    }

    func observeChanges() {
        NotificationCenter.default
            .publisher(for: NSManagedObjectContext.didChangeObjectsNotification)
            .receive(on: DispatchQueue.main)
            .sink { [weak self] notification in
                if (notification.userInfo?[NSInsertedObjectsKey] as? Set<NSManagedObject>)?.contains(where: {$0.entity.name == "Book"}) ?? false {
                    self?.performFetchRequest()
                }
            }
            .store(in: &cancellables)
    }

    private func performFetchRequest() {
```

```
                if (notification.userInfo?[NSInsertedObjectsKey] as? Set<NSManagedObject>)?.contains(where: {$0.entity.name == "Book"}) ?? false {
                    self?.performFetchRequest()
                }
            }
            .store(in: &cancellables)
    }

    private func performFetchRequest() {
        let request = Book.fetchRequest()
        let context = PersistentModelStore.container.viewContext
        do {
            let items = try context.fetch(request)
            self.bookLists = items
        } catch {
            print(error)
        }
    }

    func createNew() throws {
        let context = PersistentModelStore.container.viewContext

        let author = Author(context: context)
        author.id = UUID()
        author.name = "The Author"

        let book = Book(context: context)
        book.id = UUID()
        book.title = "The Book"
        book.desc = "Authorization Book"
        book.edition = 1
        book.publishedAt = Date()
        book.authors = author

        try context.save()
    }
}
```

Sharing Resources
With Other Modules

- Assets
  - Images
  - Colors
- Fonts
- Core Data
- Other Resources

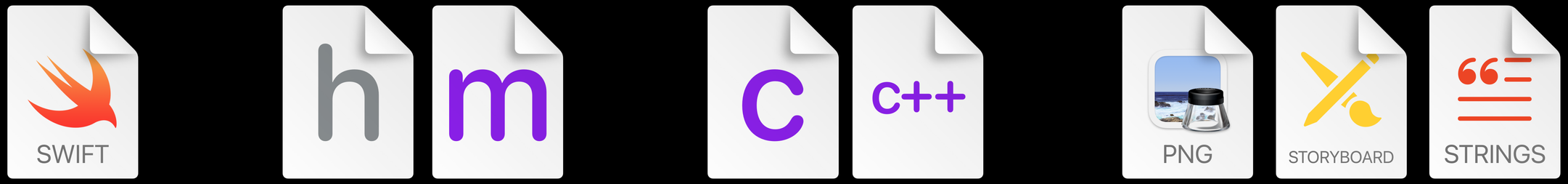
CoreData

Sharing Resources

Other Resources

Sharing Resources

Supported in SPM



Sharing Resources

Mastering SwiftPM

Today

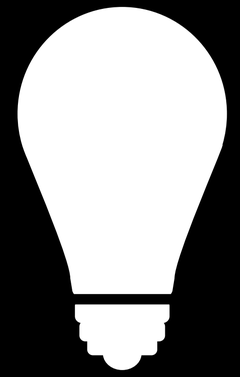
- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Dealing with Non-SPM external Dependency

- Cocoapods
- Carthage
- Frameworks/SDKs

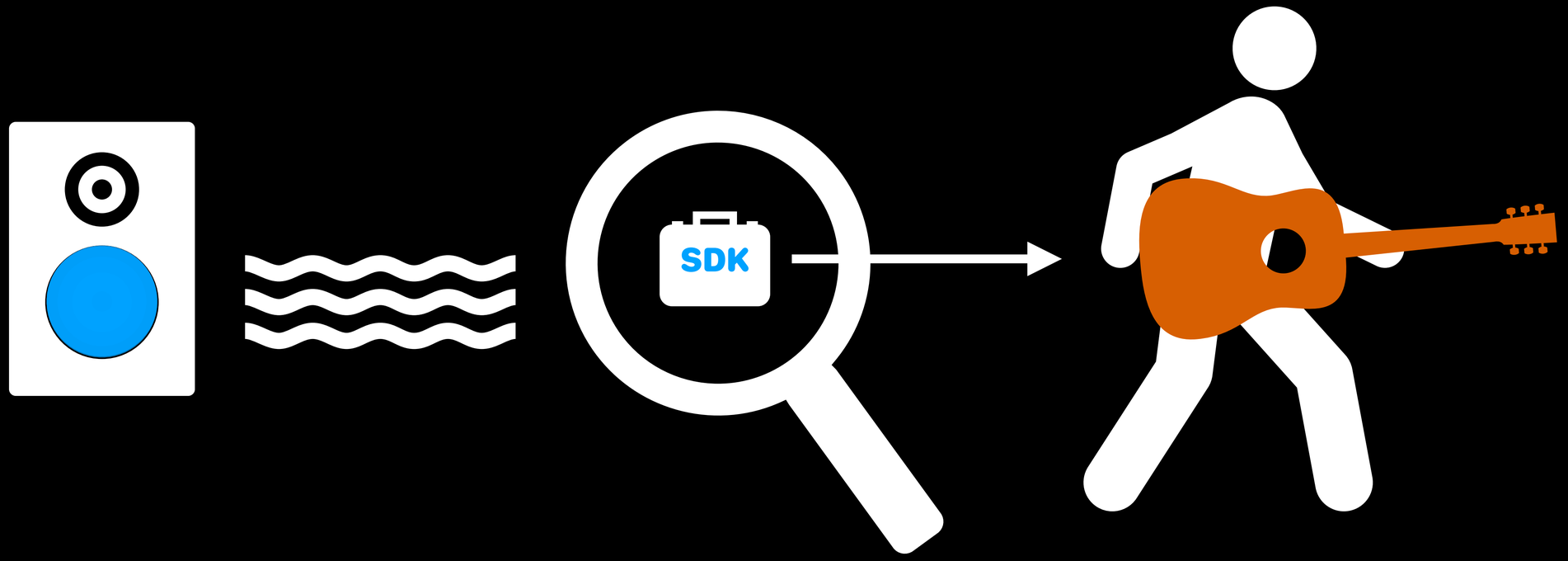
Dealing with Non-SPM external Dependency

- Understand the feature
- Study the APIs of the dependency
- Study the API with respect to feature
- Use abstraction for APIs
  - Protocol or Protocol Witness



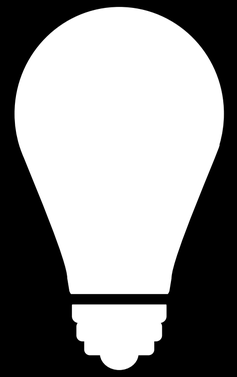
Dealing with Non-SPM external Dependency

Understand the Feature



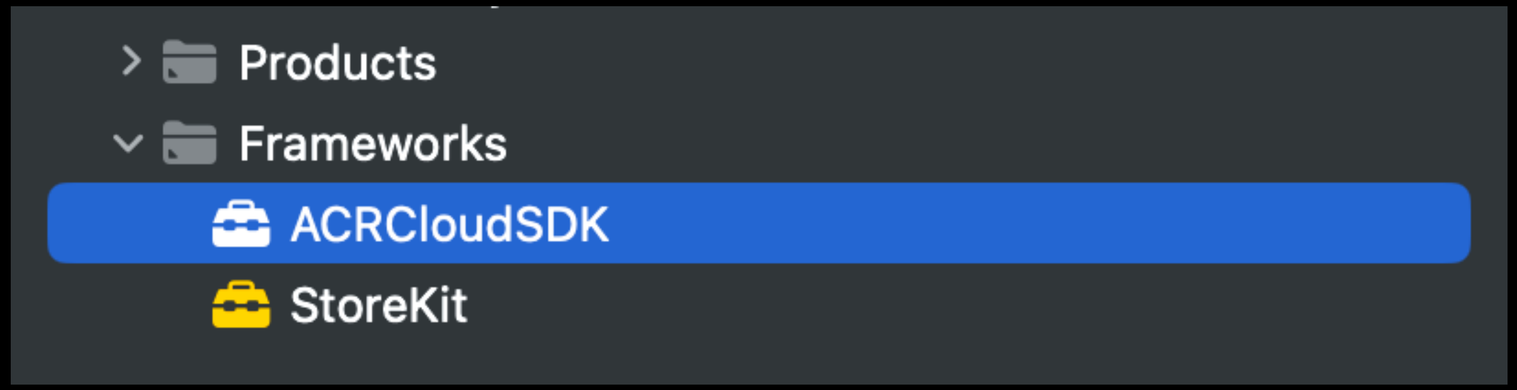
Dealing with Non-SPM external Dependency

- Understand the feature
- Study the APIs of the dependency
- Study the API with respect to feature
- Use abstraction for APIs
  - Protocol or Protocol Witness
- Use Dependency injection or environment



Dealing with Non-SPM external Dependency

Understand API of Dependency



App-Bridging-Header.h

```
1 //
2 //  Use this file to import your target's public headers that you would like to expose to Swift.
3 //
4
5 #import "ACRCloudConfig.h"
6 #import "ACRCloudRecognition.h"
7
```

Dealing with Non-SPM external Dependency

Understand API of Dependency

```
#import <Foundation/Foundation.h>
#import "ACRCloudConfig.h"

@interface ACRCloudRecognition : NSObject

-(id)initWithConfig:(ACRCloudConfig*)config;

-(void)startPreRecord:(NSInteger)recordTime;
-(void)stopPreRecord;

-(void)startRecordRec;

-(void)stopRecordRec;

-(void)stopRecordAndRec;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(char*)buffer len:(int)len;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(NSData*)pcm_data;

-(void)recognize_fp:(NSData*)fingerprint
        resultBlock:(ACRCloudResultBlock)resultBlock;

-(NSString*)recognize_fp:(NSData*)fingerprint;
-(NSString*)recognize_hum_fp:(NSData*)fingerprint;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
+(NSData*)get_fingerprint:(char*)pcm len:(int)len;
+(NSData*)get_fingerprint:(NSData*)pcm;
+(NSData*)get_hum_fingerprint:(NSData*)pcm;

+(NSData*)get_fingerprint:(char*)pcm
```

```
//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(char*)buffer len:(int)len;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(NSData*)pcm_data;

-(void)recognize_fp:(NSData*)fingerprint
        resultBlock:(ACRCloudResultBlock)resultBlock;

-(NSString*)recognize_fp:(NSData*)fingerprint;
-(NSString*)recognize_hum_fp:(NSData*)fingerprint;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
+(NSData*)get_fingerprint:(char*)pcm len:(int)len;
+(NSData*)get_fingerprint:(NSData*)pcm;
+(NSData*)get_hum_fingerprint:(NSData*)pcm;

+(NSData*) get_fingerprint:(char*)pcm
                       len:(unsigned)len
                sampleRate:(unsigned)sampleRate
                  nChannel:(short)nChannel;
+(NSData*)get_fingerprint:(NSData*)pcm
               sampleRate:(unsigned)sampleRate
                 nChannel:(short)nChannel;

//convert pcm/wav to mono 8000HZ
+(NSData*) resample:(NSData*)pcm
         sampleRate:(unsigned)sampleRate
           nChannel:(short)nChannel;

+(NSData*) resample:(char*)pcm
                len:(unsigned)len
         sampleRate:(unsigned)sampleRate
           nChannel:(short)nChannel;

+(NSData*) resample_bit32:(char*)pcm
                      len:(unsigned)bytes
               sampleRate:(unsigned)sampleRate
                 nChannel:(short)nChannel
                  isFloat:(bool)isFloat;

+(NSString*) version;
@end
```

Dealing with Non-SPM external Dependency

Understand API of Dependency

```
#import <Foundation/Foundation.h>
#import "ACRCloudConfig.h"

@interface ACRCloudRecognition : NSObject

-(id)initWithConfig:(ACRCloudConfig*)config;

-(void)startPreRecord:(NSInteger)recordTime;
-(void)stopPreRecord;

-(void)startRecordRec;

-(void)stopRecordRec;

-(void)stopRecordAndRec;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(char*)buffer len:(int)len;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(NSData*)pcm_data;

-(void)recognize_fp:(NSData*)fingerprint
        resultBlock:(ACRCloudResultBlock)resultBlock;

-(NSString*)recognize_fp:(NSData*)fingerprint;
-(NSString*)recognize_hum_fp:(NSData*)fingerprint;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
+(NSData*)get_fingerprint:(char*)pcm len:(int)len;
+(NSData*)get_fingerprint:(NSData*)pcm;
+(NSData*)get_hum_fingerprint:(NSData*)pcm;

+(NSData*)get_fingerprint:(char*)pcm
```

Dealing with Non-SPM external Dependency

Study the API with respect to feature

```
#import <Foundation/Foundation.h>
#import "ACRCloudConfig.h"

@interface ACRCloudRecognition : NSObject

-(id)initWithConfig:(ACRCloudConfig*)config;

-(void)startPreRecord:(NSInteger)recordTime;
-(void)stopPreRecord;

-(void)startRecordRec;

-(void)stopRecordRec;

-(void)stopRecordAndRec;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(char*)buffer len:(int)len;

//only support RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 8000 Hz
-(NSString*)recognize:(NSData*)pcm_data;

-(void)recognize_fp:(NSData*)fingerprint
        resultBlock:(ACRCloudResultBlock)resultBlock;

-(NSString*)recognize_fp:(NSData*)fingerprint;
-(NSString*)recognize_hum_fp:(NSData*)fingerprint;
```

Dealing with Non-SPM external Dependency

Use Abstraction (Protocol)

Inside the feature (SPM) module

```
import Foundation
import Models

public protocol SongRecogniser {
    init(configuration: Configuration)
    func start()
    func stop()
    func statusStream() -> AsyncThrowingStream<String, Error>
    func songRecognitionResultStream() -> AsyncThrowingStream<Models.MediaRecognisionResult, Error>
    func isRecognisingStream() -> AsyncStream<Bool>
}
```

Dealing with Non-SPM external Dependency

Use Abstraction (Protocol)

Inside the feature (SPM) module

```
import Foundation
import Models

public protocol SongRecogniser {
    init(configuration: Configuration)
    func start()
    func stop()
    func statusStream() -> AsyncThrowingStream<String, Error>
    func songRecognitionResultStream() -> AsyncThrowingStream<Models.MediaRecognisionResult, Error>
    func isRecognisingStream() -> AsyncStream<Bool>
}

public struct SongRecogniserDefault: SongRecogniser {
    public init(configuration: Configuration) {}
    public func start() {}
    public func stop() {}
    public func statusStream() -> AsyncThrowingStream<String, Error> {
        AsyncThrowingStream { _ in }
    }
    public func songRecognitionResultStream() -> AsyncThrowingStream<MediaRecognisionResult, Error> {
        AsyncThrowingStream { continuation in
            continuation.finish()
        }
    }
    public func isRecognisingStream() -> AsyncStream<Bool> {
        AsyncStream {
            $0.finish()
        }
    }
}
```

Dealing with Non-SPM external Dependency

Conforming to Protocol

Inside the App Target

```
public class SongRecognitionManager: SongRecogniser {
    var client: ACRCloudRecognition
    var configuration: Configuration
    var states: ((String?) -> Void)?
    let config = ACRCloudConfig()
    private var isRecognising: CurrentValueSubject<Bool, Never> = .init(false)

    @Dependency(\.logger["SongRecognitionManager"]) private var logger
    static let jsonDecoder = JSONDecoder()

    public enum SongRecognitionError: Swift.Error, LocalizedError {
        case message(String)

        public var errorDescription: String? {
            switch self {
            case .message(let message):
                return message
            }
        }
    }

    required public init(configuration: Configuration) {
        self.configuration = configuration
        config.accessKey = configuration.accessKey
        config.accessSecret = configuration.accessSecret
        config.host = configuration.host
        config.protocol = configuration.protocol
        config.recMode = configuration.recognitionMode.recMode

        if (config.recMode == rec_mode_local || config.recMode == rec_mode_both) {
            config.homedir = Bundle.main.resourcePath!.appending("/acrcloud_local_db");
        }
```

```
        AsyncThrowingStream { (continuation: AsyncThrowingStream<Models.MediaRecognisionResult, Error>.Continuation) in
            @Dependency(\.logger["SongRecognitionResultStream"]) var logger
            self.config.resultBlock = { result, type in
                logger.debug("Music recognition Result: \(result ?? "NO-RESULT")")
                let type = RecognitionResultType(type: type)
                logger.debug("Music Recognition Type: \("\(type)")")

                guard let data = result?.data(using: .utf8) else {
                    continuation.finish(throwing: SongRecognitionError.message("Can not get the result."))
                    return
                }
                do {
                    let message = try Self.jsonDecoder.decode(Models.MediaRecognisionResult.self, from: data)
                    logger.debug("Decoded recognition result: \("\(message)")")
                    continuation.yield(message)
                    continuation.finish()
                } catch {
                    print("Error decoding MediaRecognitionResult: \(error)")
                    continuation.finish(throwing: SongRecognitionError.message("Could not decode result"))
                }
            }

            continuation.onTermination = { _ in
                self.stop()
            }
        }
    }

    public func isRecognisingStream() -> AsyncStream<Bool> {
        AsyncStream { [weak self] continuation in
            guard let self else {
                continuation.finish()
                return
            }
            let token = self.isRecognising.sink { state in
                continuation.yield(state)
            }
            continuation.onTermination = { @Sendable _ in
                token.cancel()
            }
        }
    }
}
```

Dealing with Non-SPM external Dependency

Use Dependency Injection or environment override

Inside the App Target

```
@main
struct RecognApp: App {

    let store: StoreOf<AppFeature> = .init(initialState: .init()) {
        AppFeature()
    }

    var body: some Scene {
        WindowGroup {
            AppView(store: store)
        }
    }
}
```

Dealing with Non-SPM external Dependency

Use Dependency Injection

Inside the App Target

```
@main
struct RecognApp: App {

    let store: StoreOf<AppFeature> = .init(initialState: .init()) {
        @Dependency(\.config) var config

        let manager = SongRecognitionManager(configuration: .init(
            accessKey: config.acrClould.hummingAccessKey,
            accessSecret: config.acrClould.hummingSecretKey,
            host: config.acrClould.host,
            recognitionMode: .remote
        ))

        AppFeature()
    }

    var body: some Scene {
        WindowGroup {
            AppView(store: store)
        }
    }
}
```

Dealing with Non-SPM external Dependency

Use Dependency Injection

Inside the App Target

```
@main
struct RecognApp: App {

    let store: StoreOf<AppFeature> = .init(initialState: .init()) {
        @Dependency(\.config) var config

        let manager = SongRecognitionManager(configuration: .init(
            accessKey: config.acrClould.hummingAccessKey,
            accessSecret: config.acrClould.hummingSecretKey,
            host: config.acrClould.host,
            recognitionMode: .remote
        ))

        AppFeature()
            .transformDependency(\.songRecogniserClient) { dependency in
                dependency.set(manager)
            }
    }

    var body: some Scene {
        WindowGroup {
            AppView(store: store)
        }
    }
}
```

Dealing with Non-SPM external Dependency

Use Environment override

Inside the App Target

```
@main
struct RecognApp: App {

    @Dependency(\.config) var
    let manager = SongRecognitionManager(configuration: .init(
       accessKey: config.acrClould.hummingAccessKey,
       accessSecret: config.acrClould.hummingSecretKey,
       host: config.acrClould.host,
       recognitionMode: .remote
    ))

    var body: some Scene {
        WindowGroup {
            AppView()
        }
    }
}
```

Dealing with Non-SPM external Dependency

Use Environment override

Inside the App Target

```
@main
struct RecognApp: App {
    @Dependency(\.config) var
    let manager = SongRecognitionManager(configuration: .init(
       accessKey: config.acrClould.hummingAccessKey,
       accessSecret: config.acrClould.hummingSecretKey,
       host: config.acrClould.host,
       recognitionMode: .remote
    ))

    var body: some Scene {
        WindowGroup {
            AppView()
                .environmentObject(manager)
       }
    }
}
```

Mastering SwiftPM

Today

- Access Modifiers
- Package Manifest
- Package Manager Vocabulary
- Dependencies
- New Feature using SwiftPM
- Integration with Xcode Project
- Sharing Resources
- Dealing with Non-SPM external Dependency

Getting to know Swift Package Manager WWDC18

https://developer.apple.com/wwdc18/411

Creating Swift Package WWDC19

https://developer.apple.com/wwdc19/410

Adopting Swift Packages in Xcode WWDC19

https://developer.apple.com/wwdc19/408

Swift Packages: Resources & Localization WWDC20

https://developer.apple.com/wwdc20/10169

Creating Swift Package Plugin WWDC22

https://developer.apple.com/wwdc22/110401

Swift Package Manager Dependency Owners

https://alejandromp.com/blog/swift-package-manager-dependency-owners/

Adding Core Data to a Swift Package

https://ishabazz.dev/blog/2020/7/5/using-core-data-with-swift-package-manager

---

Localizating Swift Package

https://developer.apple.com/documentation/xcode/localizing-package-resources

---

SwiftSettings

https://developer.apple.com/documentation/packagedescription/swiftsetting

Mastering SwiftPM

Swift Package manager

Ratnesh Jain

Thank You