Tools

Why Go ▼

Go Wiki: Mobile **Table of Contents**

Building and deploying to Android Building and deploying to iOS App icon SDK applications and generating bindings Building and deploying to Android Building and deploying to iOS

This article will contain step-by-step guides to explain how to achieve these strategies. Tools

 Building and deploying to Android Building and deploying to iOS

App icon

 SDK applications Building and deploying to Android

Tools

gomobile also supports Go Modules, e.g. using

\$ gomobile bind -v -o android.aar -target=android ./package

under a project directory.

To install gomobile tools:

\$ go install golang.org/x/mobile/cmd/gomobile@latest

On macOS, you will need to have Xcode Command Line Tools installed.

\$ gomobile init

The following sections will help you how to use the gomobile tool. **Native applications**

The native category includes applications entirely written in Go. Currently, the golang.org/x/mobile contains only a small set of packages that focus on:

App control and configuration

• OpenGL ES 2 and ES 3 bindings Asset management

• Event management • Experimental packages include OpenAL bindings, audio, font, sprite and motion sensors

Run gomobile build to build an Android APK.

There are various example native applications under golang.org/x/mobile/example. We will build and deploy the basic example both to an Android and iOS device. Grab the application.

\$ go get -d golang.org/x/mobile/example/basic **Building and deploying to Android**

\$ gomobile build -target=android -androidapi 19 golang.org/x/mobile/example/basic Build command will build an APK named basic.apk.

If an AndroidManifest.xml is defined in the package directory, it is added to the APK output. Otherwise, a default manifest is generated. If you have the adb command installed on your machine, you can use gomobile install to build and push the APK to your mobile device.

Building and deploying to iOS Run gomobile build to build the package as an iOS application.

Note: target=ios requires the host machine running macOS. You need to obtain a signing identity and download provisioning profiles in order to continue. \$ gomobile build -target=ios golang.org/x/mobile/example/basic

You can deploy .app files by dragging and dropping them to the device. In Xcode, open Window > Devices.

Drag and drop the .app file to "Installed Apps" section.

Select the physical device from the left pane.

Check the "Copy items if needed" option

The build command will build an application bundle, named basic app.

\$ gomobile install golang.org/x/mobile/example/basic

DEVICES

My Mac

8.3 (12F69)

iPhone 4s 8.3 (12F69)

iPhone 5

8.3 (12F69)

iPhone 5s

8.3 (12F69)

8.3 (12F69)

8.3 (12F69)

iPhone 6 Plus

SIMULATORS

 $Q \land \bigcirc \equiv \Box$

10.10.4 (14E46) iPhone 6 Model s the fundamental anagement model for Burcu Dogan's iPhone Capacity 55.91 GB (45.2 GB available) 8.3 (12F70) apps. You rarely 67% Battery

Burcu Dogan's iPhone

8.3 (12F70)

Version

1.0

Basic

• There are a few limitations on how the exported APIs should look due to the limitations of the target language.

\$ gomobile bind -o app/hello.aar -target=android golang.org/x/mobile/example/bind/hello

golang.org/x/mobile v0.0.0-20210716004757-34ab1303b554 // indirect

🔄 🗦 📄 🖹 m Vie...r.m 🖟 📵 @implementation ViewController 🔍 🕕 🗦

Identifier

golang.gotest

iPad Air 8.3 (12F69) Take Screenshot View Device Logs iPad Retina 8.3 (12F69)

Device Information

Name

iOS

Identifier

Installed Apps

Name

gotest

- | 🕸

Resizable iPad Remote Disc 8.3 (12F69) Resizable iPhone Android... ≜ 8.3 (12F69) Tags Red Alternatively, you can deploy application bundles to your iOS device by using the ios-deploy utility command line tool. Use ios-deploy to push the application to your device. \$ ios-deploy -b basic.app App icon It is possible to set an app icon by creating assets/icon.png. SDK applications and generating bindings In this category, we will show you how you can use a Go package in your existing Android or iOS application. The advantages to following this strategy: • You can reuse a Go package from a mobile app without making significant changes to your existing application. • In cases where you want to share a common code base between your Android and iOS application, you can write the common functionality once in Go and glue them to the platform-specific code by invoking the Go package through bindings. Current limitations are listed below. Only a subset of Go types are currently supported.

devices is not yet supported. • Run the following command to generate the aar file that is suitable for importing into Android projects:

Tips: From 1.16, it is recommended to execute go get -d golang.org/x/mobile/cmd/gomobile before each execution of gomobile bind go get will

automatically add indirect references to go.mod. These indirect references maybe automatically deleted by ide or go mod tidy, but they are required!

Note: Go Mobile runs on the same architectures as Go, which currently means ARM, ARM64, 386 and amd64 devices and emulators. Notably, Android on MIPS

We will use the example package under golang.org/x/mobile/example/bind/hello to generate bindings and invoke Greetings function from Java and Objective-C.

golang.org/x/mod v0.4.2 // indirect golang.org/x/sys v0.0.0-20210510120138-977fb7262007 // indirect golang.org/x/tools v0.1.2 // indirect golang.org/x/xerrors v0.0.0-20200804184101-5ec99f83aff1 // indirect

nello

build.gradle

hello.iml

android.iml

build.gradle

gradlew.bat

local.properties

gradlew

README

+ repositories {

dependencies {

+ }

flatDir {

dirs '.'

Building and deploying to iOS

\$ open ios/bind.xcodeproj

Cancel

🛅 図 昂 Q 🛕 🖯 🎹 🗅 🗐 器 🕻 > 🗟 bind

М

PROJECT

TARGETS

bind

📥 bind

Settings.

▼ <u>bind</u>

▶ 🚔 Hello.framework

h AppDelegate.h

m AppDelegate.m

h ViewController.h

m ViewController.m

Main.storyboard kaunchScreen.xib

Supporting Files

Products

Filter

bind

bind

Hello.framework

h AppDelegate.h

m AppDelegate.m

h ViewController.h

m ViewController.m

bind.app

@import Hello

iOS Simulator

require (

android

build

gradle

▼ 🗀 hello

Project

 Launch Android Studio. • File > Import Project... to import the reference project from \$GOPATH/src/golang.org/x/mobile/example/bind/android.

build.gradle

• Language bindings have a performance overhead.

\$ go get -d golang.org/x/mobile/example/bind/...

Grab the example by running the command below.

Building and deploying to Android

Select Gradle Project Import android (~/src/golang.org/x/mobile/example/bine pen Event Log Show Log in Finder Select build.gradle or settings.gradle 🗀 .idea <u></u> арр • 🖫 🖼 × 💋 📖 Hide path

sers/jbd/src/golang.org/x/mobile/example/bind/android

bind

cmd 🗀

doc doc

event

example

audio

▶ 🗀 basic

▼ 🗀 bind

Drag and drop a file into the space above to quickly locate it in the tree.

Besides, if you try to add yourmodule.aar into your own project, after copy the yourmodule.aar file and yourmodule.jar file to "android\app" folder, below editing

android

android.iml

build.gradle

gradlew.bat

Cancel

OK

gradlew

DPATH elements or

le binary if the

pin directory. */

Finish

< △ > | ⊞

non-recursive

ello) */

README X (huild gradle X

ings.gradle 🗀 арр **External Libraries** build 🗀 🗀 gradle hello

?

Gradle 'android' project refresh failed: // Cause: org/golang/mobile/GobindPlugin : Unsupported major.minor version 5... (yesterday 4:29 PM) 25:1 LF 🛊 UTF-8 🛊 🚡 🔠 Build and deploy the application to the device. The app module contains the main application that invokes the hello. Greetings. When the application is launched the text view is updated with the string returned value. If you are not using Android Studio, in order to work with bindings for Android, you need to have Android SDK installed and ANDROID_HOME environment variable set to the SDK path. You also need the NDK installed; the easiest way is to run the SDK command sdkmanager ndk-bundle. Alternatively, if you are not familiar with android development, and you do not wish to set up all the required environment (Android SDK, Gradle, etc), you can use this docker image to build the application in docker instead.

implementation (name:'yourmodulename', ext:'aar')

\$ gomobile bind -target=ios golang.org/x/mobile/example/bind/hello

Note: target=ios requires the host machine to be running macOS.

\$ cd \$GOPATH/src/golang.org/x/mobile/example/bind

Choose options for adding these files:

in "android\app\build.gradle" file should be done in order to make your module imported correctly.

Added folders: Create groups Create folder references

If you decide to keep Hello. framework in the main directory you have to add the main directory to the Framework Search Paths in the the targets Build

hind 🖈

No ≎

hind 🖈

Yes 🗘

/Applications

Universal (

hind 🖈

🚘 bind

bind

📥 bind

bind/Info.plist

\$(inherited)

\$(PROJECT_DIR)

\$(PROJECT_DIR)/..

bind | Build bind: Succeeded | Today at 5:22 PM

#import "ViewController.h"

in hello.framework

@interface ViewController ()

@implementation ViewController

@synthesize textLabel;

[super loadView];

Build and run it on the simulator or an actual device (Cmd+R). When the application launches, the label on the main view will be modified with the string returned

- (void)loadView {

#import "Hello/Hello.h" // Gomobile bind generated header file

textLabel.text = GoHelloGreetings(@"iOS and Gopher");

// Copyright 2015 The Go Authors. All rights reserved.

// Use of this source code is governed by a BSD-style

// license that can be found in the LICENSE file.

@executable_path/Frameworks

Resource Tags

Info

Build Settings

Build Phases

Build Rules

Signing & Capabilities

General

Customized

Setting

Info.plist File

Product Name

Enable Bitcode

▼ Build Options

▼ Deployment

▼ Packaging

▼ Search Paths

М

器

@end

}

@end

Drag and drop the Hello. framework bundle to the Xcode project. Check "Copy items if needed" if you need a different copy of the framework bundle within

Gomobile bind will generate a framework bundle called Hello. framework. Open the sample Xcode project by running the command below.

the Xcode otherwise. Otherwise, modifying the Go package source code and rerunning gomobile bind will update the hello.framework.

Destination: <a>Copy items if needed

Add to targets: 🗸 🚕 bind

Installation Directory Hello.framework Strip Linked Product **Targeted Device Family ▼** Linking Setting **Runpath Search Paths**

Setting ▶ Framework Search Paths **▼** User-Defined ASSETCATALOG_COMPILER_APPICON_NAME **▼** MTL_ENABLE_DEBUG_INFO Debug Release

Your project layout should look like what's shown below.

Main.storyboard X LaunchScreen.xib **Supporting Files** Info.plist m main.m Products

from GoHelloGreetings which invokes the hello.Greetings function.

let msg = Hello.GoHelloGreetings("gopher")

Note that you can also invoke GoHelloGreetings from Swift by importing Hello.

64-bit ARM binaries, and the simulator will run the amd64 binary.

Brand Guidelines Code of Conduct

 Building and deploying to iOS iOS Simulator

Native applications

You need to have Go 1.16 or above to install mobile tools. Go Mobile introduces a tool, gomobile, to help you with the build and the binding process.

Native applications

Learn

Docs ▼

Community ~

obile.

?

ntroller:

Controller

Name

Favorites

All My Files

iCloud Drive

Applications

Desktop

Downloads

Devices

Documents

liewController class

ate UIViewController

directly. Instead, you

ate subclasses of the

.DS_Store

Basic

main.go

Packages

The Go mobile subrepository adds support for mobile platforms (Android and iOS) and provides tools to build mobile applications. There are two strategies you can follow to include Go into your mobile stack: • Writing all-Go native mobile applications. • Writing SDK applications by generating bindings from a Go package and invoke them from Java (on Android) and Objective-C (on iOS).

Why Go **Get Started** Packages About Connect

As of Go 1.5, only darwin/amd64 works on the iOS simulator. To use the simulator, you need to configure Xcode to only try to run 64-bit binaries.

Xcode matches the bit width of the ARM binaries when running on the X86 simulator. That is, if you configure Xcode to build both 32-bit and 64-bit ARM

binaries (the default), it will attempt to run 32-bit X86 binaries on the simulator, which will not work with Go today. Modify the Xcode build settings to only build

Golang Weekly go.dev uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic. Learn more. Okay

This content is part of the Go Wiki. Playground Use Cases Standard Library **Twitter** Download About Go Packages Case Studies Tour Blog GitHub Stack Overflow Issue Tracker Slack Help Release Notes r/golang Meetup