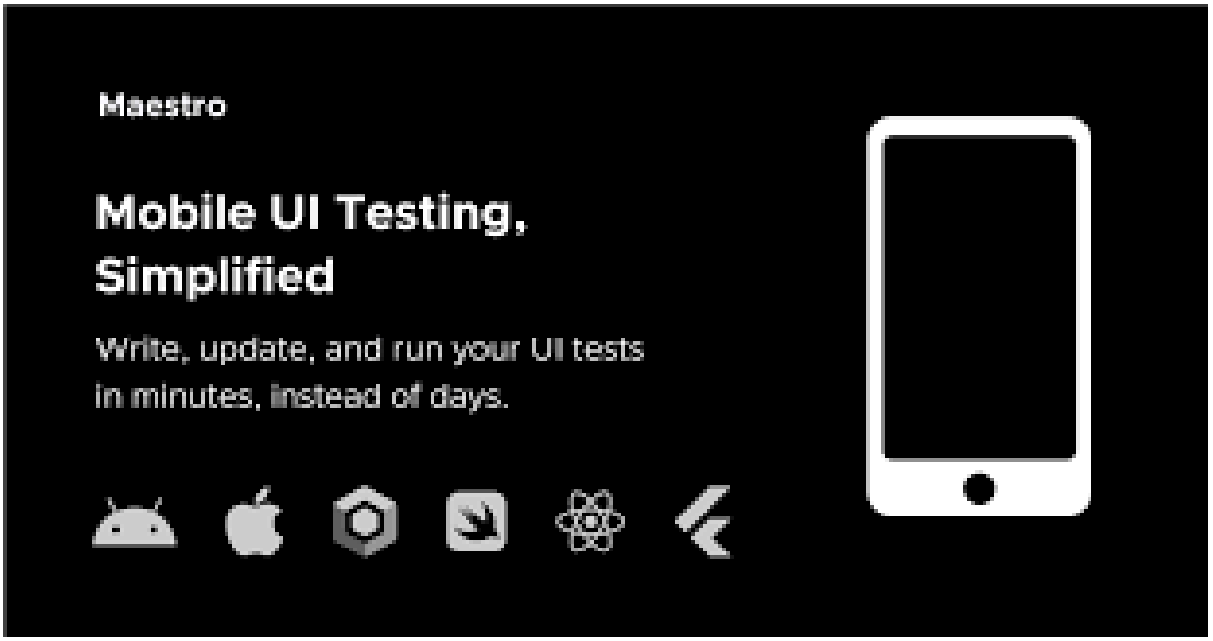


Maestro by mobile.dev

R&D

17.01.2024

—

What is Maestro?

Maestro is the simplest and most effective mobile UI testing framework.

Overview

The existing mobile UI automation tooling available today is not living up to its promise. Teams are implementing UI tests in hopes of iterating faster without breaking things — but instead, test flakiness, infrastructure issues, brittle tests, and framework complexity end up slowing developers down.

Why Maestro?

Pros:

- ★ Easy to Define Test Flows:
  - Maestro is built on learnings from its predecessors (Appium, Espresso, UIAutomator, XCTest) and allows you to easily define and test your Flows.
    - What are Flows? Think of Flows as parts of the user journey in your app. Login, Checkout and Add to Cart are three examples of possible Flows that can be defined and tested using Maestro.
- ★ Flows are Helpful:
  - Flows are helpful to test the scenarios like Signup, Login, Add to Favorites, Add to cart etc.
- ★ Built-in Tolerance to Flakiness:
  - Built-in tolerance to flakiness. UI elements will not always be where you expect them, screen tap will not always go through, etc. Maestro embraces the instability of mobile applications and devices and tries to counter it.
- ★ Built-in Tolerance to Delays:
  - No need to pepper your tests with sleep() calls. Maestro knows that it might take time to load the content (i.e. over the network) and automatically waits for it (but no longer than required).
- ★ Blazingly Fast iteration:
  - Tests are interpreted, no need to compile anything. Maestro is able to continuously monitor your test files and rerun them as they change.
- ★ Easy Syntax:
  - Declarative yet powerful syntax. Define your tests in a yaml file.
- ★ Simple Setup.
  - Maestro is a single binary that works anywhere.

Cons:

- ★ Maintenance Overhead:
  - Maintaining and updating test scripts, especially in dynamic applications, can require ongoing effort to keep tests in sync with application changes.
- ★ Community and Support:
  - This tool has a less community or support from the tool provider can be essential for problem-solving and sharing best practices.
- ★ Restricted Selector Support:
  - Maestro provides support for only a limited set of selectors(like id, coordinates etc) , it may be challenging to target specific elements(Xpath isn't supported) in more complex user interfaces, impacting the precision of test automation.
- ★ Potential Learning Curve for Complex Scenarios:
  - it lacks advanced features needed for complex test scenarios, potentially requiring users to explore alternative solutions for complex projects.
- ★ Comparative Analysis with Other Tools:
  - The functionality provided by Maestro is similar to other record-and-play tools like Katalon Studio and that seems more suitable.

Installing Maestro

Installing the CLI

Dependencies:

1. Xcode (recommended version is 14 or higher)

   Please make sure that Command Line Tools are installed (Xcode -> Preferences -> Locations -> Command Line Tools)

2. Run the following command to install Maestro on Mac OS, Linux or Windows (WSL):
   a. curl -Ls "https://get.maestro.mobile.dev" | bash

Upgrading the CLI

1. Simply run the installation script again:
   a. curl -Ls "https://get.maestro.mobile.dev" | bash

Installing a specific version of Maestro

1. To install a specific version, declare a MAESTRO_VERSION property and run the same installation command as before:
   a. export MAESTRO_VERSION={version}; curl -Ls "https://get.maestro.mobile.dev" | bash

Connecting to Your Device

1. iOS
   a. Before running Flows on iOS Simulator, install Facebook IDB tool
      i. brew tap facebook/fb

         brew install facebook/fb/idb-companion

ii. *Note: At the moment, Maestro does not support real iOS devices*

2. Android
    a. maestro test will automatically detect and use any local emulator or USB-connected physical device.

Build and Install your App

Currently maestro supports emulator and device builds for Android (.apk) and iOS Simulator (.app).

1. **Android**

   Android app binary requirements:
   - APK (AAB not supported)
   - Compatible with x86_64 architecture
   - Requires Android API level 26 or newer
   - Release and Debug builds both supported

Building with Gradle

Build your app using one of the commands below. Then find the appropriate APK file in the build/outputs/apk/ output directory.

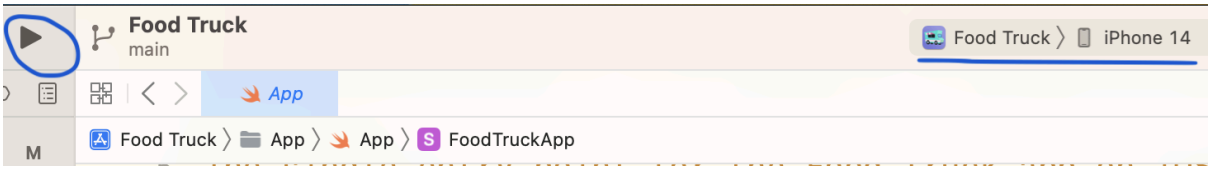
```
# Release build
./gradlew assembleRelease

# Debug build
./gradlew assembleDebug
```

2. **iOS**

Building with Xcode

Maestro currently supports iOS Simulator builds (**.app**), AppStore distribution device builds (**.ipa**) are not currently supported. The easiest way to get an app installed on the simulator is by building and running it from Xcode (make sure that simulator target is selected):



Default build location

If you want to install the .app manually, the file can be found under the Products folder at Xcode Menu: **Product -> Show Build Folder in Finder -> Then go to path Products/Debug-iphonesimulator**
The .app file can then be installed on the simulator by simple dragging and dropping the file or via Xcode command line tools by running:
xcrun simctl install $DEVICE_UDID /path/to/your/app

Building with Xcode command line tools

To build an app with Xcode command line tools **xcrun xcodebuild** should be used. Here is an example on how to build Food Truck app for iOS simulator target:

```
xcrun xcodebuild -scheme 'Food Truck' \
-project 'Food Truck.xcodeproj' \
```

-configuration Debug \
-sdk 'iphonesimulator' \
-destination 'generic/platform=iOS Simulator' \
-derivedDataPath \
build
The .app file can be then found under **/build** folder

Writing Your First Flow

Write a test in a YAML file

```
# flow.yaml

appId: your.app.id
---
- launchApp
- tapOn: "Text on the screen"
```

Make sure an Android device/emulator or iOS simulator is running (instructions) and app is correctly installed on a selected device (instructions).
Run it!

```
maestro test flow.yaml
```

Run a Sample Flow

Use the download-samples command to download the samples:

maestro download-samples

This will download the build-in sample Flows and app into a samples/ folder in your current directory.

Run the Sample Flow

Install the sample app and then run the associated Flow using the maestro test command.

iOS

cd ./samples

unzip sample.zip

xcrun simctl install Booted Wikipedia.app

maestro test ios-flow.yaml

Comparison and Details Chart:

Feature	Maestro	Appium	Espresso	Detox
Ease of use	High (YAML syntax)	Moderate (code-based)	Moderate (code-based)	Moderate (code-based)
Platform support	Android & most platforms (emulators)	Android, iOS, Web	Android	iOS
Test language	YAML	JavaScript, Python, Java, etc.	Java	JavaScript
Flakiness tolerance	High	Moderate	Low	Moderate
Test speed	Moderate	High	High	High
Community size	Small	Large	Large	Medium
Feature richness	Moderate	High	High	High
Complexity handling	Limited	High	High	High
Cross-platform testing	Limited (mostly Android)	High	Limited	Moderate
Reporting	Basic	Detailed	Detailed	Detailed

Learning curve	Low	Moderate	Moderate	Moderate