おり、fvmコマンドを使わなければそのまま利用可能です。ちなみに、筆者はshellのエイリアスを利用してfvmの入力を省略しています。

```
~/.zshrc
alias flutter="fvm flutter"
```

なお、第2章以降ではfvmコマンドを省略してflutterコマンド、dartコマンドを扱います。ご自身の環境、コマンドを実行するディレクトリにあわせて読み替えてください。

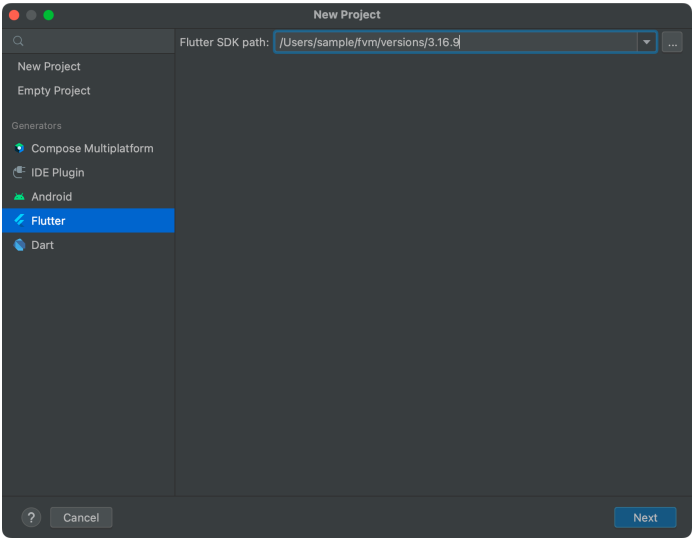
1.4 プロジェクトの作成

Flutterプロジェクトは Android Studio から GUI（*Graphical User Interface*）で作成します。

Android Studioでの作成手順

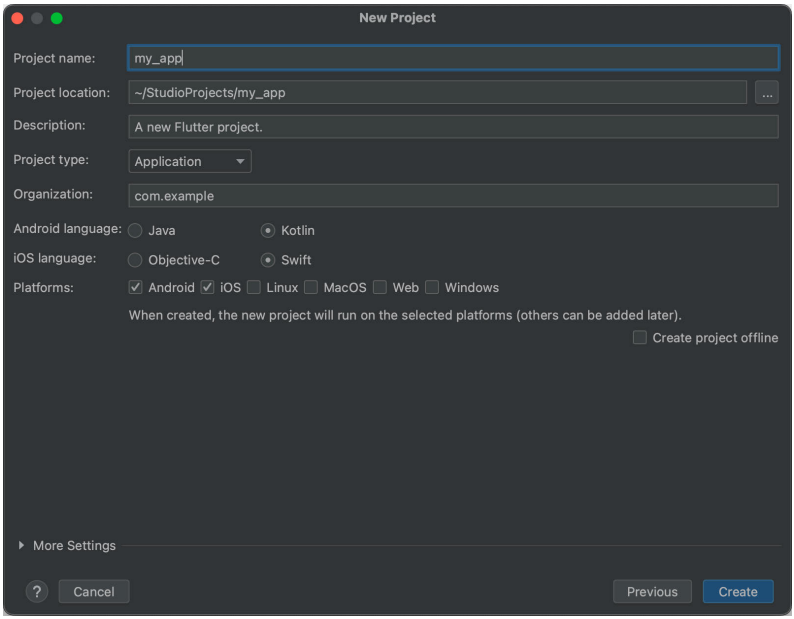
Android Studioのアプリケーションメニューから「File」➡「New」➡「New Flutter Project」の順で選択すると、ウィンドウが表示されます（**図1.5**）。

図1.5　**新規プロジェクトウィンドウ**



ウィンドウ左のリストで「Flutter」が選択されていることを確認します。「Flutter SDK path」は図1.5のようにfvmでインストールしたFlutter SDKを選択します。「Next」をクリックすると、プロジェクトの詳細を選択する画面に切り替わります(**図1.6**)。

図1.6　**新規プロジェクトの詳細ウィンドウ**



「Project name」は任意の名前でけっこうです。プロジェクトを作成するディレクトリを変更したい場合は「Project location」を任意に編集してください。今回はiOSとAndroidで動作するアプリのプロジェクトを作成するので「Project type」が`Application`になっていること、「Platforms」ではAndroidとiOSにチェックが入っていることを確認してください。

「Create」をクリックすると、Android Studioで作成したFlutterのテンプレートプロジェクトが開きます。

テンプレートプロジェクトをのぞいてみよう

作成されたプロジェクトのファイル、ディレクトリは**表1.1**のような構成になっています。.(ドット)から始まるファイル、ディレクトリやAndroid Studioのモジュールファイルなどを直接操作することは少ないかと思います。

表1.1 **プロジェクトのファイル、ディレクトリ構成**

ファイル名、ディレクトリ名	説明
.dart_tool	Dart言語のツールが配置されるディレクトリ
.idea	Android Studioのプロジェクト設定ファイルが配置されるディレクトリ
.metadata	Flutterツールが利用するファイル
analysis_options.yaml	コード静的解析のオプションファイル。lintルールを変更する場合に編集する
android	Android Studioのプロジェクト。Androidネイティブのコード、しくみを利用する場合に閲覧、編集する
ios	Xcodeのプロジェクト。iOSネイティブのコード、しくみを利用する場合に閲覧、編集する
lib	Flutterの実装ファイルを配置するディレクトリ。Dartの実装ファイルはここに配置する
my_app.iml	Android Studioのモジュールファイル
pubspec.lock	パッケージ(ライブラリなど)のバージョンを解決するファイル
pubspec.yaml	Flutterプロジェクトの設定、依存関係を記述するファイル。パッケージ(ライブラリなど)やアセット類はこのファイルに記述する
test	Flutterのテストコードを配置するディレクトリ

fvmの設定

作成したプロジェクトに対してfvmの設定を行います。今回はFlutterのバージョンを3.16.9に設定します。プロジェクトのルートディレクトリで以下のコマンドを実行します。

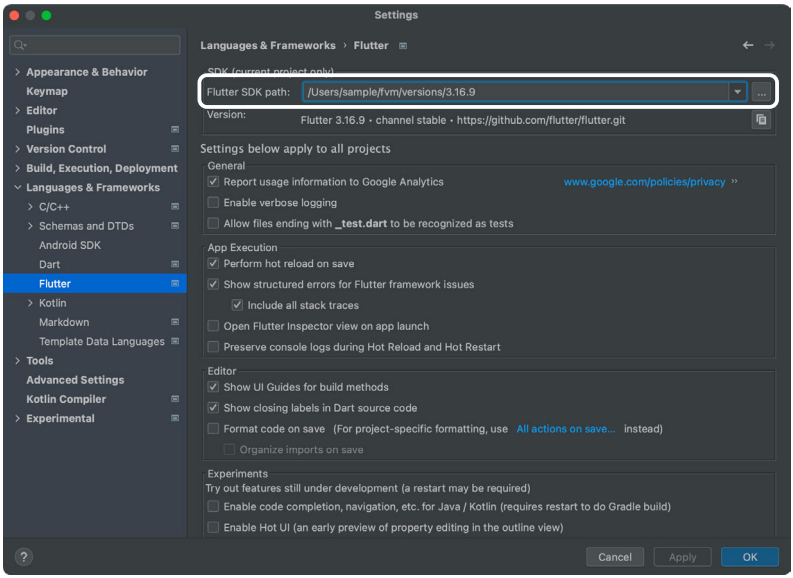
```
# プロジェクトで利用するfvmのバージョンを設定
$ fvm use 3.16.9
```

すると、プロジェクトのルートディレクトリに.fvmというディレクトリが作成され、その中にFlutter SDKへのシンボリックリンクが配置されます。もし、gitでバージョン管理する場合は除外するように.gitignoreを編集しておきましょう。

.gitignore
```
# 省略
.fvm/flutter_sdk # この行を追加
```

続いて、Android Studioが参照するFlutter SDKのパスを変更します。Android Studioを起動し、アプリケーションメニューから「Settings」を選択します。Settingsウィンドウの検索窓に「Flutter」と入力し、ツリーの中から「Flutter」を選択します(**図1.7**)。「Flutter SDK Path」へ.fvm/flutter_sdkのシンボリックリンクが示している先のパスを入力し、OKボタンを押します。

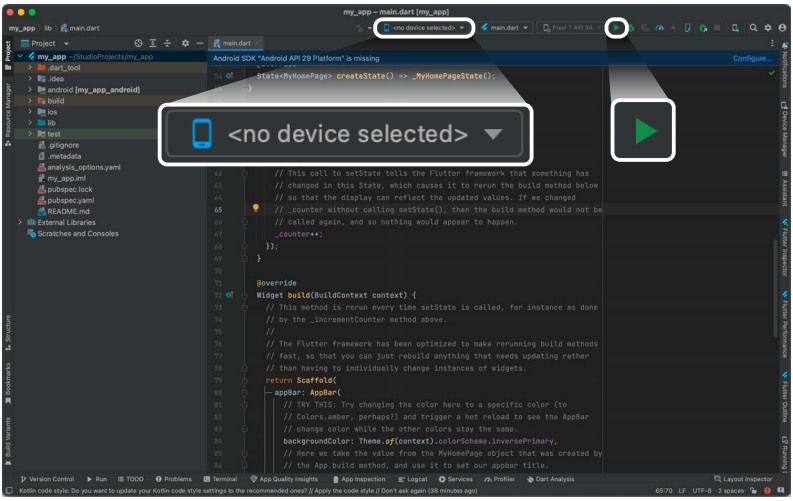
図1.7　Settingsダイアログで Flutter SDK Path を設定



1.5 Flutterアプリの実行

それではさっそくアプリを実行してみましょう。まずは**図1.8**に示す「Flutter Device Selection」ボタンから実行デバイスを選択します。

図1.8 **Android StudioのFlutter Device Selectionボタン（拡大左）と実行ボタン（拡大右）**



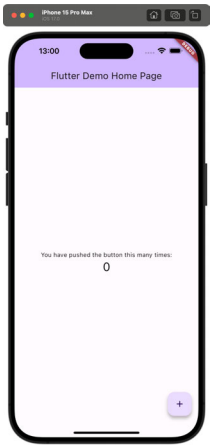
iOS Simulatorでの実行

「Flutter Device Selection」のボタンをクリックすると「Open iOS Simulator」という項目があります。これを選択するとiOS Simulatorが起動します。

iOS Simulatorが起動したら、Android StudioのNavigation Barにある実行ボタンをクリックします（図1.8の右側）。

するとiOS Simulator上でテンプレートのFlutterアプリが起動します（**図1.9**）。

図1.9 **iOS Simulatorで動作するFlutterアプリ**



今まで一度もiOS Simulatorを起動したことがない環境では、エラーが発生することがあります。筆者の環境では、Xcodeから一度だけiOS Simulatorを起動すると解決しました。

XcodeからiOS Simulatorを起動するには、Xcodeのアプリケーションメニューから「Xcode」➡「Open Developer Tool」➡「Simulator」の順で選択します。

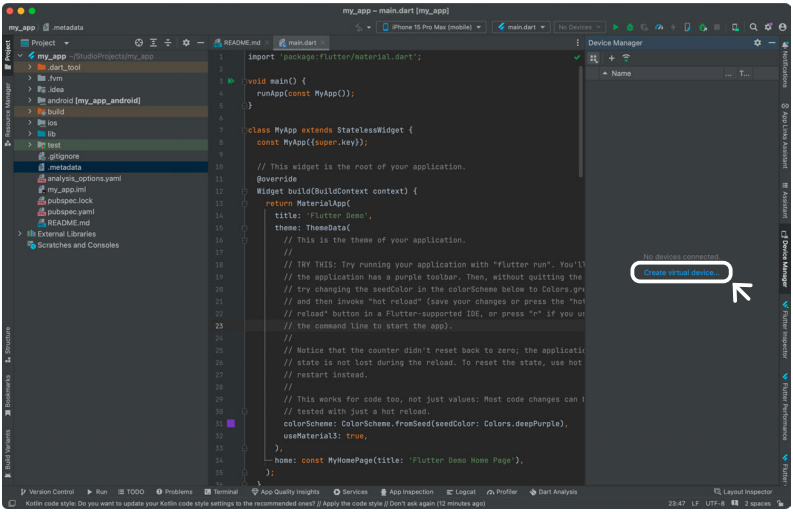
Android Emulatorでの実行

作成済みのAndroid Emulatorを探します。Android Studioのバージョンによって、初期状態でAndroid Emulatorが作成されている場合と作成されていない場合があります。Android Studioのアプリケーションメニューから「Tools」➡「Device Manager」を選択します。Device Managerの画面にEmulatorが表示されなければ、以下の手順で作成します。

Android Emulatorを作成する

Device Managerの画面で「Create virtual device...」ボタンをクリックします(**図1.10**)。

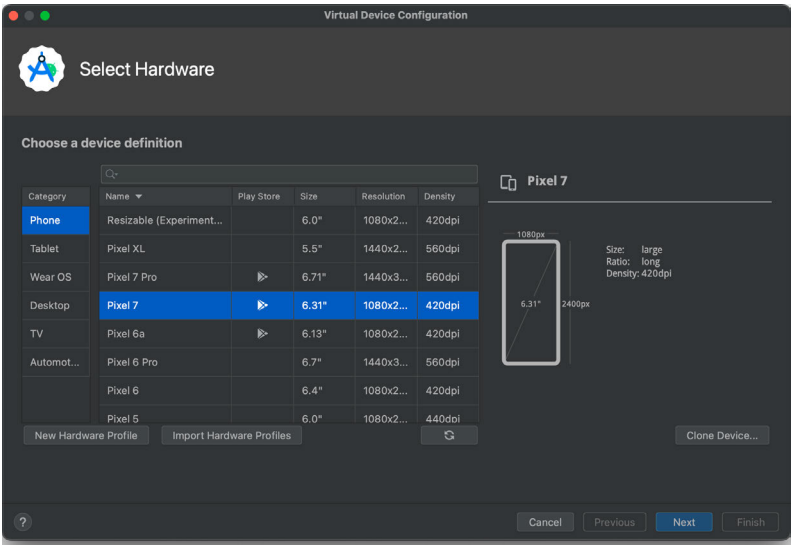
図1.10　**Device Manager画面のCreate deviceボタン**



すると、Android Emulatorを作成するウィンドウが表示されます(**図1.11**)。まず、Emulatorの画面解像度や仮想ディスプレイサイズを選択し「Next」ボタ

ンをクリックします。

図1.11 **Android Emulatorを作成するウィンドウ**



続いてEmulatorのAPIレベルを選択します。本書ではAPI Level 34を選択しました。なお、Flutterがサポートしている APIレベルは公式ドキュメントの「Supported deployment platforms」[注6]で確認できます。

選択したAPIレベルのシステムイメージをダウンロード、選択し「Next」ボタンをクリックします。

最後にEmulatorの名前やその他の設定を行う画面に遷移します。特に必要なければ初期値のまま「Finish」ボタンをクリックします。これでAndroid Emulatorの作成が完了しました。

Android Emulatorを起動し、アプリを実行する

iOSのときと同様に「Flutter Device Selection」のボタンをクリックして、作成したAndroid Emulatorを選択します。リストに現れない場合は「Refresh」を選択してみましょう。

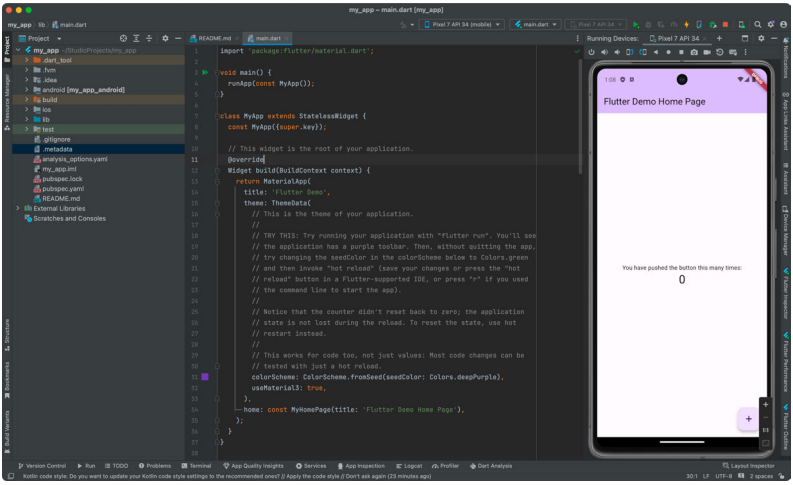
Android Emulatorのウィンドウが表示されない場合は、Android Studioのメニューから「View」➡「Tool Windows」➡「Running Devices」を選択すると起動中のAndroid Emulatorが表示されます。

注6 https://docs.flutter.dev/development/tools/sdk/release-notes/supported-platforms

Android Emulatorが起動したら実行ボタン(図1.8)をクリックします。

するとAndroid Emulator上でテンプレートのFlutterアプリが起動します(**図1.12**)。

図1.12 **Android Emulatorで動作するFlutterアプリ**



1.6 まとめ

Flutterの開発環境をインストールし、iOS SimulatorとAndroid Emulatorでアプリが実行できるところまでを体験しました。

マルチプラットフォームであるため環境を整える作業は多くの手順が必要となりますが、`flutter doctor`コマンドのようなサポートツールが用意されていました。アプリの実行もエディタのプラグインによってスムーズだったかと思います。これらの優れた開発者体験もFlutterの魅力の一つですね。