FLUTTER TUTORIALS

BY: **TECMAN**

TECMAN Lesson 10

Adding Google Map to Flutter

Introduction

Flutter is Google's mobile app SDK for crafting high-quality native experiences on iOS and Android in record time.

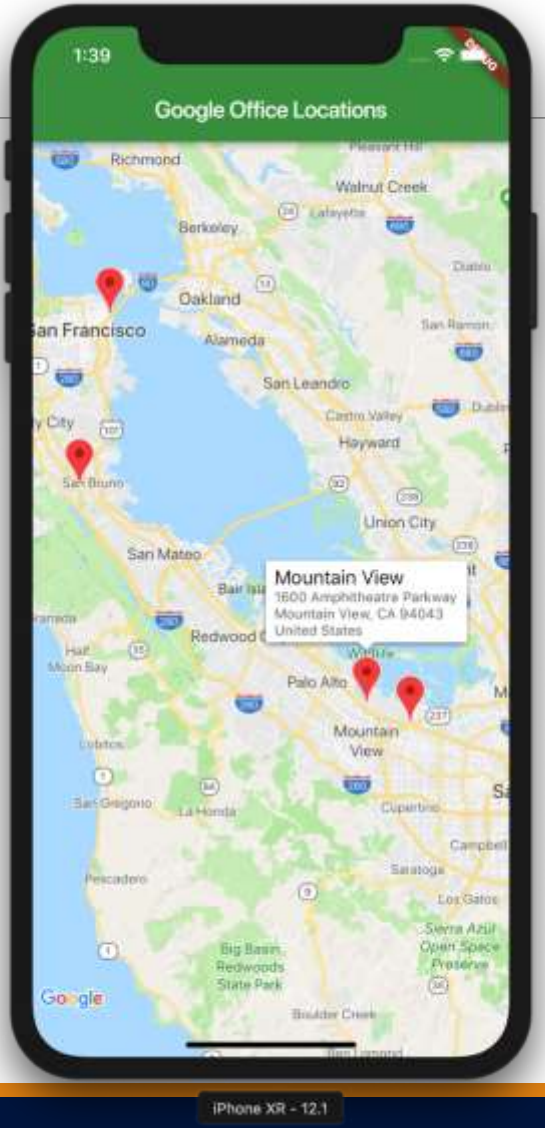
With the Google Maps Flutter plugin, you can add maps based on Google maps data to your application. The plugin automatically handles access to the Google Maps servers, map display, and response to user gestures such as clicks and drags. You can also add markers to your map. These objects provide additional information for map locations, and allow the user to interact with the map.

What you'll build

In this lesson, you'll build a mobile app featuring a Google Map using the Flutter SDK. Your app will:

- Display a Google Map
- Retrieve map data from a web service
- Display this data as markers on the Map

What you'll build



What you'll learn

- How to create a new Flutter application.
- How to configure a Google Maps Flutter plugin.
- How to add Markers to a map, using location data from a web service.

This lesson focuses on adding a Google map to a Flutter app. Non-relevant concepts and code blocks are glossed over and are provided for you to simply copy and paste.

Adding Google Maps to the app

It's all about the API keys

To use Google Maps in your Flutter app, you need to configure an API project with the Google Maps Platform, following both the Maps SDK for Android's Get API key, and Maps SDK for iOS' Get API key processes. With API keys in hand, carry out the following steps to configure both Android and iOS applications.

Adding Google Maps to the app

Adding an API key for an Android app

To add an API key to the Android app, edit the AndroidManifest.xml file in android/app/src/main. Add a single meta-data entry containing the API key created in the previous step.

Check the git link below:

android/app/src/main/AndroidManifest.xml

Adding Google Maps to the app

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.google_maps_in_flutter">

    <application
        android:name="io.flutter.app.FlutterApplication"
        android:label="google_maps_in_flutter"
        android:icon="@mipmap/ic_launcher">

        <!-- Add the following entry, with your API key -->
        <meta-data android:name="com.google.android.geo.API_KEY"
               android:value="YOUR-KEY-HERE"/>

        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:theme="@style/LaunchTheme"
            android:configChanges="orientation|keyboardHidden|keyboard|screenSize|locale
            android:hardwareAccelerated="true"
            android:windowSoftInputMode="adjustResize">
            <meta-data
                android:name="io.flutter.app.android.SplashScreenUntilFirstFrame"
                android:value="true" />
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Adding Google Maps to the app

Adding an API key for an iOS app

To add an API key to the iOS app, edit the AppDelegate.m file in ios/Runner. Unlike Android, adding an API key on iOS requires changes to the source code of the Runner app. The AppDelegate is the core singleton that is part of the app initialization process.

Make two changes to this file. First, add an #import statement to pull in the Google Maps headers, and then call the provideAPIKey() method of the GMSServices singleton. This API key enables Google Maps to correctly serve map tiles.

Adding Google Maps to the app

Check the git link below:

ios/Runner/AppDelegate.m

TECMAN LIMITED 2019

Adding Google Maps to the app

```
#import "AppDelegate.h"
#import "GeneratedPluginRegistrant.h"
// Add the following import.
#import "GoogleMaps/GoogleMaps.h"

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Add the following line, with your API key
  [GMSServices provideAPIKey: @"YOUR-API-KEY"];
  return [super application:application didFinishLaunchingWithOptions:launchOptions];
}

@end
```

Adding Google Maps to the app

You also need to add a setting to ios/Runner/Info.plist. This entry forces Flutter on iOS into a single threaded mode, which is required for the platform view embedding to work. This technical restriction is being worked on and will be lifted before Google Maps moves out of Developer Preview.

Check git link below:

ios/Runner/Info.plist

Adding Google Maps to the app

Putting a map on the screen

Now it's time to get a map on the screen. Update lib/main.dart as follows:.

Check git link below:

lib/main.dart

TECMAN LIMITED 2019

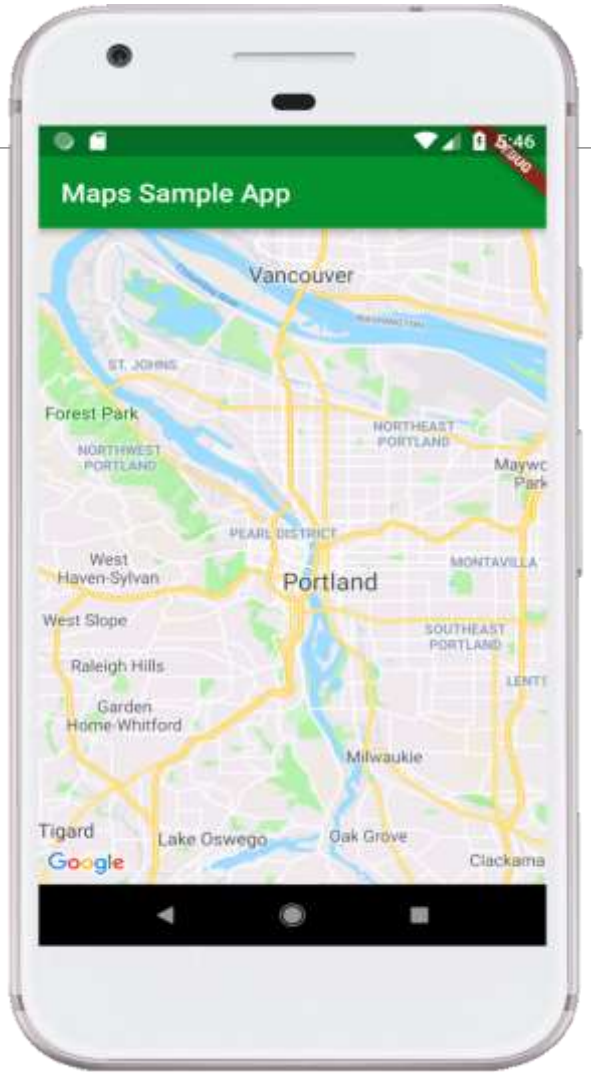
```dart
import 'package:flutter/material.dart';
import 'package:google_maps_flutter/google_maps_flutter.dart';

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {
  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  GoogleMapController mapController;

  final LatLng _center = const LatLng(45.521563, -122.677433);

  void _onMapCreated(GoogleMapController controller) {
    mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Maps Sample App'),
          backgroundColor: Colors.green[700],
        ),
        body: GoogleMap(
          onMapCreated: _onMapCreated,
          initialCameraPosition: CameraPosition(
            target: _center,
            zoom: 11.0,
          ),
        ),
      ),
    );
  }
}
```

Adding Google Maps to the app

Running the app

Run the Flutter app in either iOS or Android to see a single map view, centered on Portland. Feel free to modify the map center to represent your hometown, or somewhere that is important to you.

Adding Google Maps to the app



Put Google on the Map

Google has many offices around the world, from North America, Latin America, Europe, Asia Pacific, to Africa & Middle East. The nice thing about these maps, if you investigate them, is that they have an easily usable API endpoint for supplying the office location information in JSON format. In this step, you put these office locations on the map.

Put Google on the Map

As you grow your codebase, it's time to start using tooling that Dart provides to make the code more readable and maintainable. In this step, you use code generation to parse JSON, and code linting to surface potential code smells.

TECMAN LIMITED 2019

Put Google on the Map

To use these capabilities, add some new dependencies to the pubspec.yaml file. These dependencies provide access to http requests, the ability to mechanize JSON parsing, a configuration of useful lint rules used widely at Google, and a build runner that ties all of it together. Edit the dependencies stanza of your pubspec.yaml file as follows:

Adding Google Maps to the app

Check git link below:

pubspec.yaml

TECMAN LIMITED 2019

Adding Google Maps to the app

```
name: google_maps_in_flutter
description: A new Flutter project.
version: 1.0.0+1

environment:
  sdk: ">=2.2.0 <3.0.0"

dependencies:
  flutter:
    sdk: flutter
  google_maps_flutter: ^0.5.11
  # Add the following two lines
  http: ^0.12.0+1
  json_serializable: ^2.0.2

# Add the following three lines
dev_dependencies:
  pedantic: ^1.4.0
  build_runner: ^1.2.7

flutter:
  uses-material-design: true
```

Adding Google Maps to the app

Run flutter packages get on the command line to retrieve these new dependencies, and to prepare the app for the next stages.

```
$ flutter packages get
Running "flutter packages get" in google_maps_in_flutter...                 0.5s
$
```

Adding Google Maps to the app

Using the providing tooling

Two of the nice additions to programming languages in recent years are pragmatic defaults for code formatting, and linting for known problematic code patterns. For code formatting, you can use flutter format, although you can probably configure your code editor to run this on a certain key combination, or on file save.

Adding Google Maps to the app

```
$ flutter format .
Formatting directory .:
Unchanged test/widget_test.dart
Skipping link ios/Pods/Headers/Public/google_maps_flutter/GoogleMapMarkerController.h
Skipping link ios/Pods/Headers/Public/google_maps_flutter/GoogleMapController.h
Skipping link ios/Pods/Headers/Public/google_maps_flutter/GoogleMapsPlugin.h
Skipping link ios/Pods/Headers/Private/google_maps_flutter/GoogleMapMarkerController.h
Skipping link ios/Pods/Headers/Private/google_maps_flutter/GoogleMapController.h
Skipping link ios/Pods/Headers/Private/google_maps_flutter/GoogleMapsPlugin.h
Skipping link ios/.symlinks/plugins/google_maps_flutter
Skipping link ios/.symlinks/flutter
Unchanged lib/main.dart
Unchanged lib/src/locations.g.dart
Unchanged lib/src/locations.dart
Skipping hidden path .dart_tool
$
```

Adding Google Maps to the app

For linting, Dart provides the ability to configure a customized code linter. In this step, you add a handful of linters to the app, but the full list of available lints is specified in the Linter for Dart documentation.

Add a file to the root of the project called analysis_options.yaml and fill it with the following content.

Check git link below:

analysis_options.yaml

Adding Google Maps to the app

```
include: package:pedantic/analysis_options.yaml

analyzer:
  exclude:
    - lib/src/locations.g.dart

linter:
  rules:
    - always_declare_return_types
    - camel_case_types
    - empty_constructor_bodies
    - annotate_overrides
    - avoid_init_to_null
    - constant_identifier_names
    - one_member_abstracts
    - slash_for_doc_comments
    - sort_constructors_first
    - unnecessary_brace_in_string_interps
```

Adding Google Maps to the app

The first line includes a default set of rules used widely at Google, and the linter rules section gives a taste of what is possible. The exclude line references a file that hasn't been generated yet. To run the lint rules, analyze the code as follows:

Adding Google Maps to the app

```
$ flutter analyze
Analyzing google_maps_in_flutter...
No issues found! (ran in 1.8s)
$
```

If the analyzer issues warnings, don't worry, you will fix them shortly.

Adding Google Maps to the app

Parsing JSON with code generation

You might notice that the JSON data returned from the API endpoint has a regular structure. It would be handy to generate the code to marshal that data into objects that you can use in code. While Dart provides a variety of options for de-serializing JSON data (from build-it-yourself, to signing the data and using built_value), this step uses JSON annotations.

Adding Google Maps to the app

In the lib/src directory, create a locations.dart file and describe the structure of the returned JSON data as follows:

Check git link below for full code:

lib/src/locations.dart

TECMAN LIMITED 2019

Adding Google Maps to the app

Once you've added this code, your IDE (if you are using one) should display some red squiggles, as it references a nonexistent sibling file, locations.g.dart. This generated file converts between untyped JSON structures and named objects. Create it by running the build_runner:

Adding Google Maps to the app

```
$ flutter packages pub run build_runner build
[INFO] Generating build script...
[INFO] Generating build script completed, took 291ms

[INFO] Initializing inputs
[INFO] Reading cached asset graph...
[INFO] Reading cached asset graph completed, took 65ms

[INFO] Checking for updates since last build...
[INFO] Checking for updates since last build completed, took 595ms

[INFO] Running build...
[INFO] 1.2s elapsed, 0/1 actions completed.
[INFO] Running build completed, took 1.2s

[INFO] Caching finalized dependency graph...
[INFO] Caching finalized dependency graph completed, took 27ms

[INFO] Succeeded after 1.2s with 1 outputs (1 actions)

$
```

Adding Google Maps to the app

Your code should now analyze cleanly again.

Modify the main.dart file to request the map data, and then use the returned info to add offices to the map:

Check git link below for full code:

lib/main.dart

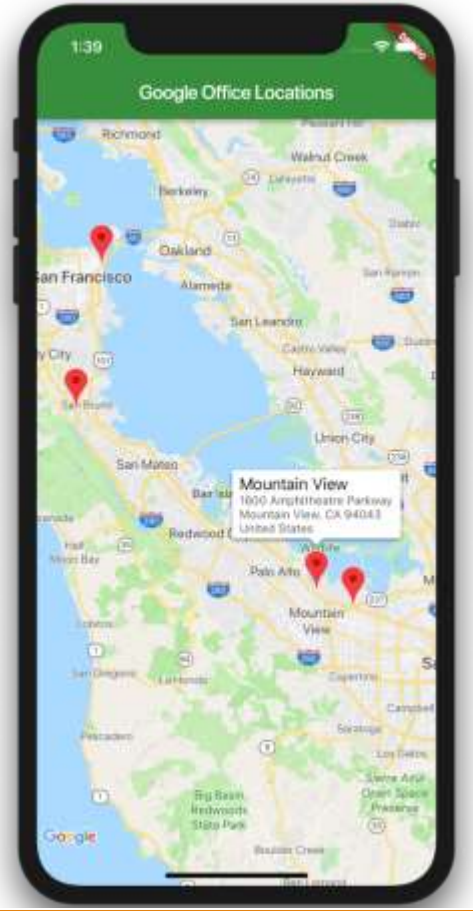
Adding Google Maps to the app

This code performs several operations:

In _onMapCreated, it uses the JSON parsing code from the previous step, awaiting until it's loaded. It then uses the returned data to create Markers inside a setState() callback. Once the app receives new markers, setState flags Flutter to repaint the screen, causing the office locations to display.

The markers are stored in a Map that is associated with the GoogleMap widget. This links the markers to the correct map. You could, of course, have multiple maps and display different markers in each.

Adding Google Maps to the app



TECMAN LIMITED 2019

THANK YOU

TECMAN LIMITED 2019