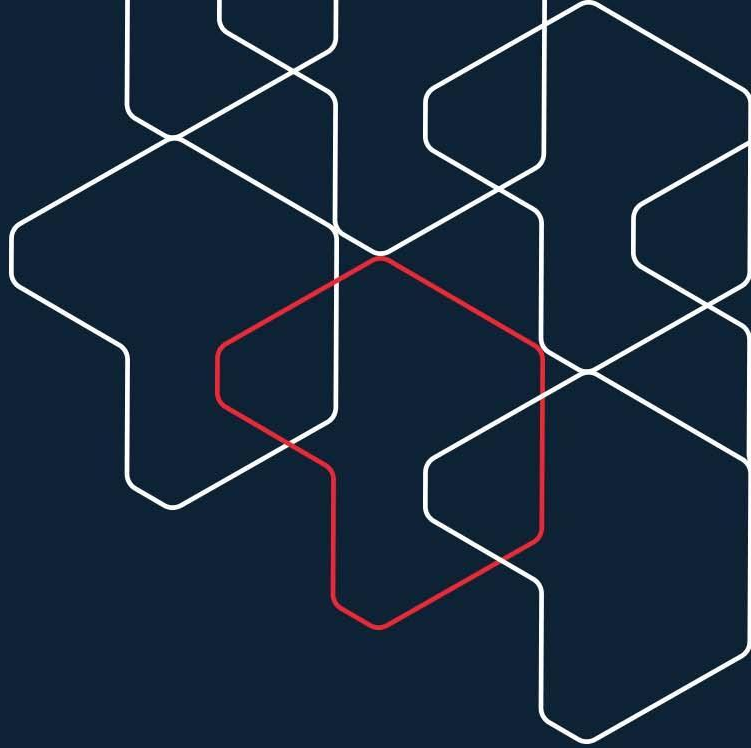


VentureDive

Flutter Training

PLATFORM CHANNELS

Agenda

- Why do we need Platform channels
- Different ways to implement it
- Message passing
- Practical demonstration for message passing
- Pigeon
- Practical demonstration for Pigeon

Why do we need Platform Channels

- Dart is good with on-screen animations plus designing detailed UI
- Non Trivial tasks such as
  - Notifications
  - App Life-Cycle
  - Manipulating Sensors, Camera, Geo-Location and sound
  - Broadcasting information to other apps or opening other native applications
  - Persisting user data in special folders or retrieving specific device information

Ways of Implementation

- Message Passing
  - Flutter’s builtin platform-specific API support does not rely on code generation, but rather on a flexible message passing style
  - This message passing style is not type safe.
- Pigeon
  - The package Pigeon can be used for sending structured typesafe messages via code generation:

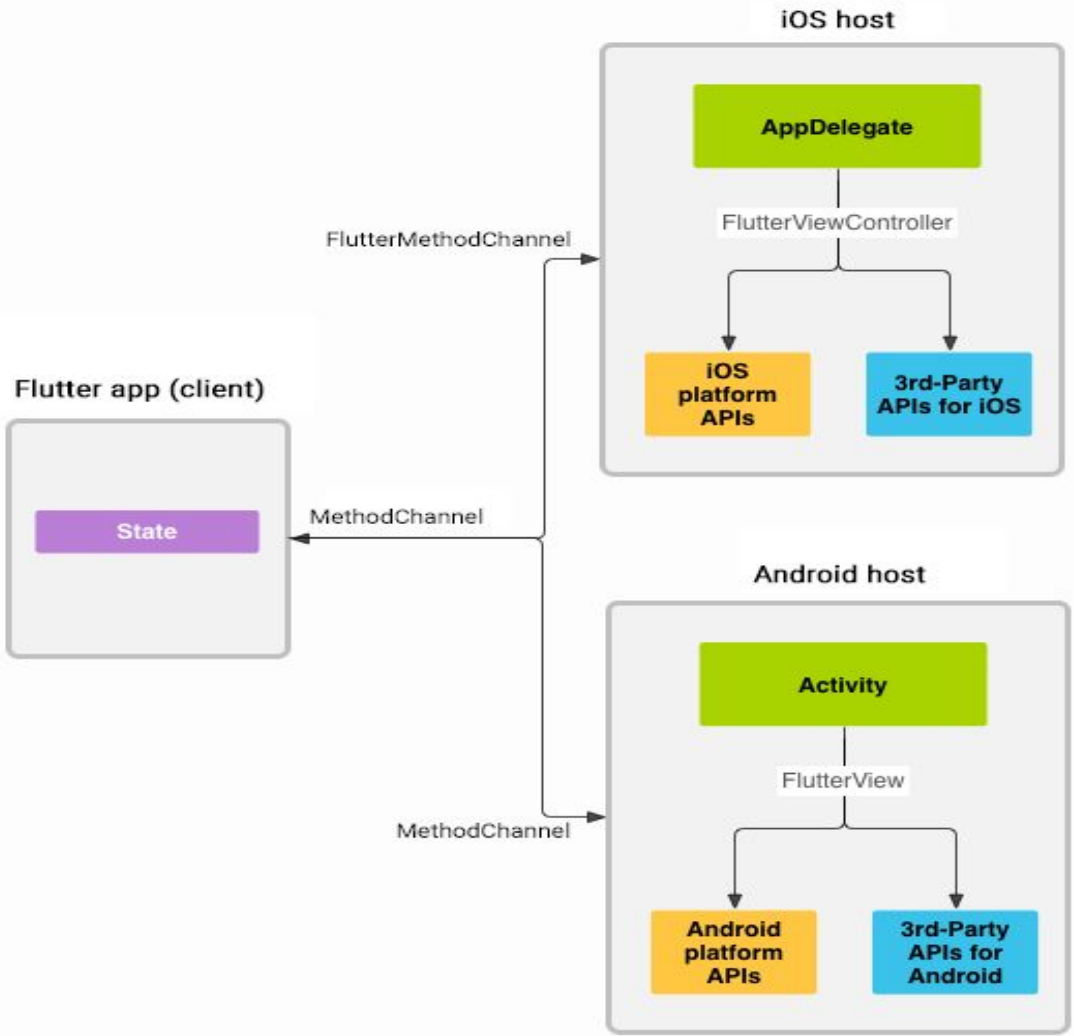
Message Passing

- The Flutter portion of the app sends messages to its host, the iOS or Android portion of the app, over a platform channel.
- The host listens on the platform channel, and receives the message. It then calls into any number of platform-specific APIs—using the native programming language—and sends a response back to the client, the Flutter portion of the app.
- Messages and responses are passed asynchronously, to ensure the user interface remains responsive.

Architectural overview:

platform channels

Messages are passed between the client (UI) and host (platform) using platform channels as illustrated in this diagram:

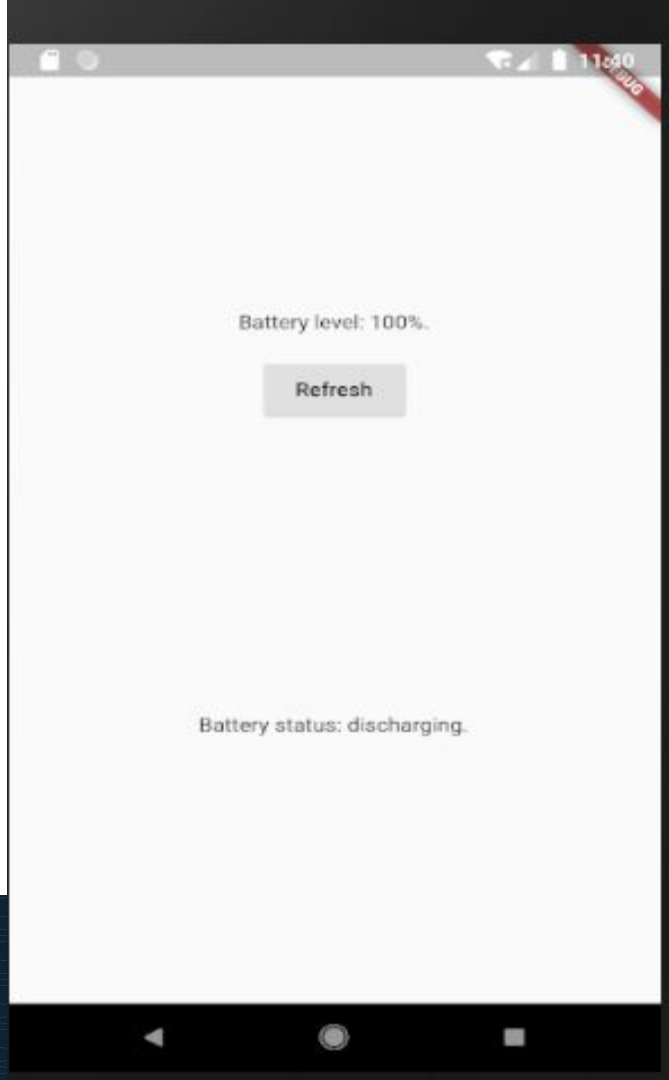


Platform Channels Data-Type Serialization

Dart	Java	Kotlin	Obj-C	Swift
null	null	null	nil (NSNull when nested)	nil
bool	java.lang.Boolean	Boolean	NSNumber numberWithBool:	NSNumber(value: Bool)
int	java.lang.Integer	Int	NSNumber numberWithInt:	NSNumber(value: Int32)
int, if 32 bits not enough	java.lang.Long	Long	NSNumber numberWithLong:	NSNumber(value: Int)
double	java.lang.Double	Double	NSNumber numberWithDouble:	NSNumber(value: Double)
String	java.lang.String	String	NSString	String
Uint8List	byte[]	ByteArray	FlutterStandardTypedData typedDataWithBytes:	FlutterStandardTypedData(bytes: Data)
Int32List	int[]	IntArray	FlutterStandardTypedData typedDataWithInt32:	FlutterStandardTypedData(int32: Data)
Int64List	long[]	LongArray	FlutterStandardTypedData typedDataWithInt64:	FlutterStandardTypedData(int64: Data)
Float64List	double[]	DoubleArray	FlutterStandardTypedData typedDataWithFloat64:	FlutterStandardTypedData(float64: Data)
List	java.util.ArrayList	List	NSArray	Array
Map	java.util.HashMap	HashMap	NSDictionary	Dictionary

Example of Message Passing

- Now we will implement an example to demonstrate fundamentals of message passing.



Type Safe Platform Specific Code Pigeon

- The previous example uses MethodChannel to communicate between the host and client which isn't typesafe.
- Calling and receiving messages depends on the host and client declaring the same arguments and datatypes in order for messages to work.
- The Pigeon package can be used as an alternative to MethodChannel to generate code that sends messages in a structured typesafe manner
- With Pigeon the messaging protocol is defined in a subset of Dart which then generates messaging code for Android or iOS.

Advantages of using Pigeon

- Using Pigeon eliminates the need to match strings between host and client for the names and datatypes of messages.
- It supports: nested classes, grouping messages into APIs, generation of asynchronous wrapper code and sending messages in either direction.
- The generated code is readable and guarantees there will be no conflicts between multiple clients of different versions.
- Supported languages are Objective-C, Java, Kotlin and Swift (via Objective-C interop).

Demo For Pigeon

- Practical Demonstration in flutter
    - Reference: https://pub.dev/packages/pigeon#-readme-tab-

Flutter Plugins

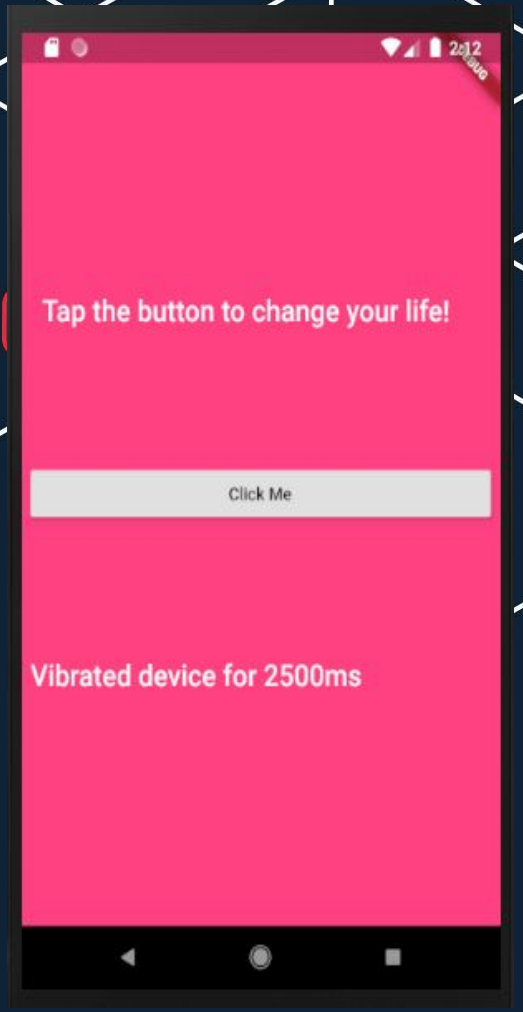
- A specialized Dart package that contains an API written in Dart code combined with one or more platform-specific implementations.
- Plugin packages can be written for Android (using Kotlin or Java), iOS (using Swift or Objective-C), web, macOS, Windows, or Linux, or any combination thereof.
- To publish your implemented plugin run command
    - flutter pub publish
- Reference: https://flutter.dev/docs/development/packages-and-plugins/developing-packages

Demo Flutter Toast Plugin

VentureDive

- Take home assignment
  Complete all Todos marked in the project Flutter-method-channel-tutorial.

- Write Android specific code to make it vibrate.
  - In case of any confusion you can refer to the blog.
    - https://medium.com/flutter-community/flutter-using-platform-channels-e8c80b869e3d



Assignment submission

Upload your code on github and submit it’s link on the Google chat group.

References

- https://flutter.dev/docs/development/platform-integration/platform-channels#example
- https://pub.dev/packages/pigeon#-example-tab-
- https://medium.com/flutter/flutter-platform-channels-ce7f540a104e

VentureDive

End