Flutter Packages

A package is a namespace that contains a group of similar types of classes, interfaces, and sub-packages. We can think of packages as similar to different folders on our computers where we might keep movies in one folder, images in another folder, software in another folder, etc. In Flutter, Dart organizes and shares a set of functionality through a package. Flutter always supports shared packages, which is contributed by other developers to the Flutter and Dart ecosystem. The packages allow us to build the app without having to develop everything from scratch.

The general structure of the package is given below (Assume a demo package as mycustom\_package):

**lib/src/\*:** It contains private Dart code files.

[**next →**](https://www.javatpoint.com/flutter-splash-screen)[**← prev**](https://www.javatpoint.com/flutter-creating-android-platform-specific-code)

# Flutter Packages

A package is a namespace that contains a group of similar types of classes, interfaces, and sub-packages. We can think of packages as similar to different folders on our computers where we might keep movies in one folder, images in another folder, software in another folder, etc. In Flutter, Dart organizes and shares a set of functionality through a package. Flutter always supports shared packages, which is contributed by other developers to the Flutter and Dart ecosystem. The packages allow us to build the app without having to develop everything from scratch.

The general structure of the package is given below (Assume a demo package as mycustom\_package):

**lib/src/\*:** It contains private Dart code files.

**lib/mydemo\_package.dart:** It is a main Dart code file. We can import it into an application as below:

1. **import** 'package:mycustom\_package/mycustom\_package.dart'

We can also export any other code file into the main code file as below syntax:

1. Export src/my\_code.dart

**lib/\*:** It is a directory, which contains the public code in the package. We can access this code as below:

1. **import** 'package:mydemo\_package/sub\_folder/custom\_file.dart'

**pubspec.yaml:** It is the project's configuration file that will use a lot during working with the Flutter project. This file contains:

* Project general settings such as name, description, and version of the project.
* Project dependencies.
* Project assets (e.g., images).

### **Types of Packages**

According to the functionality, we can categorize the package into two types:

1. Dart Package
2. Plugin Package

**Dart Package:** It is a general package, which is written in the dart language, such as a path package. This package can be used in both the environment, either it is a web or mobile platform. It also contains some Flutter specific functionality and thus has a dependency on the Flutter framework, such as **fluro package**.

**Plugin Package:** It is a specialized Dart package that includes an API written in Dart code and depends on the Flutter framework. It can be combined with a platform-specific implementation for an underlying platform such as Android (using Java or Kotlin), and iOS (using Objective C or Swift). The example of this package is the battery and image picker plugin package.

1. library flutter\_custom\_package;
3. **import** 'package:flutter/material.dart';
5. **class** CustomPackageAlertBox {
6. **static** Future showCustomAlertBox({
7. @required BuildContext context,
8. @required Widget willDisplayWidget,
9. }) {
10. **assert**(context != **null**, "If context is null!!");
11. **assert**(willDisplayWidget != **null**, "If willDisplayWidget is null!!");
12. **return** showDialog(
13. context: context,
14. builder: (context) {
15. **return** AlertDialog(
16. shape: RoundedRectangleBorder(
17. borderRadius: BorderRadius.all(Radius.circular(20)),
18. ),
19. content: Column(
20. mainAxisSize: MainAxisSize.min,
21. children: <Widget>[
22. willDisplayWidget,
23. MaterialButton(
24. color: Colors.white70,
25. child: Text('Close Alert'),
26. onPressed: () {
27. Navigator.of(context).pop();
28. },
29. )
30. ],
31. ),
32. elevation: 12,
33. );
34. });
35. }
36. }

Now, you need to test your newly created package. To test the package, create a new project. In this project, first, open the **pubspec.yaml** file and the following code in the dependency section.

1. dependencies:
2. flutter:
3. sdk: flutter
4. flutter\_custom\_package:
5. path: ../

When you add the custom package in pubspec.yaml file, Android Studio will alert you to update this file. To update the file, click on the Get dependencies and make sure that you have an internet connection during updating of file. The Android Studio automatically get the package from the internet and configure it for your application. Now, you are able to use this package. You can import the package in the dart file as below line:

1. **import** 'package: flutter\_custom\_package/flutter\_custom\_package.dart';

### **How to Publish a Package**

When you have successfully implemented a package, you can publish it on the [pub.dev](https://pub.dev/packages/http" \t "_blank), so that anyone can use it easily in the project.

Before publishing the package, make sure that the content of pubspec.yaml, README.md, and CHANGELOG.md file is complete and correct.

Next, run the following command in the terminal window to analyze every phase of the package.

1. $ flutter pub publish --dry-run

Finally, you need to run the following command to **publish** the package.

1. $ flutter pub publish