VentureDive

Flutter Training

Writing plugins/packages and product flavors

Agenda

- Package introduction
- Using packages
- Write dart packages
- Write native android/ios plugins
- Implementing product flavors in Flutter

Package Introduction

- Packages enable creation of modular code that can be shared easily
- Pubspec.yaml - A metadata file that declares the package name, version, author, and so on.
- Lib - The lib directory contains the public code in the package.

Package Types

- Dart packages
  - General packages are written in Dart.
  - Some packages can depend on Flutter framework, restricting their use to Flutter only.
- Plugin package
  - A specialized Dart package that contains an API written in Dart code combined with one or more platform-specific implementations
  - Can be written for Android, iOS, web, macos, or any combination

Using Packages

1. Searching package
2. Adding package dependency
3. Conflict resolution

Searching Packages

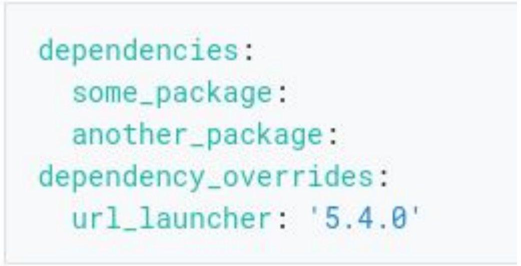
- Flutter supports using shared packages contributed by other developers to the Flutter and Dart ecosystems.
- Packages are published to pub.dev.
- The Flutter Favorites list packages that you should first consider when writing Flutter app.
- Flutter Favorite packages have passed high quality standards using a defined metrics.

Adding Package Dependency

1. Depend on it
2. Install it
3. Import it
4. Stop and restart the app, if necessary
    - Hot reload only refresh Dart code

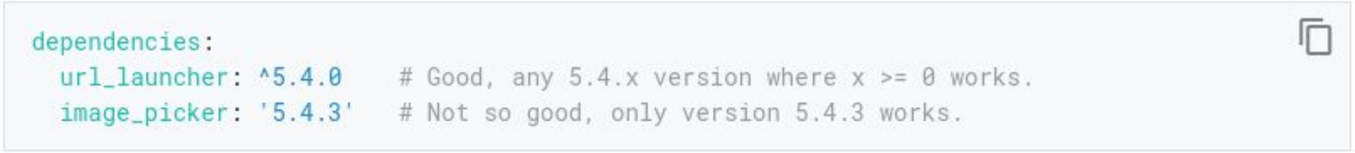
Conflict Resolution

- Dependency override



```yaml
dependencies:
  some_package:
  another_package:
dependency_overrides:
  url_launcher: '5.4.0'
```

- Version ranges



```yaml
dependencies:
  url_launcher: ^5.4.0    # Good, any 5.4.x version where x >= 0 works.
  image_picker: '5.4.3'   # Not so good, only version 5.4.3 works.
```

Semantic Versioning

- Dart community follows semantic versioning
- Format X.Y.Z
- Major.Minor.Patch
- 0.y.z is for initial development
- For more information go to: Semantic Versioning

VentureDive

Developing Dart Packages

1. Create the package
    - **flutter create --template=package hello**
    - Understand created files
2. Implement the package
    - Organize package structure
    - Implement functionality
3. Use package

Developing Plugin Packages

Supports federated plugin

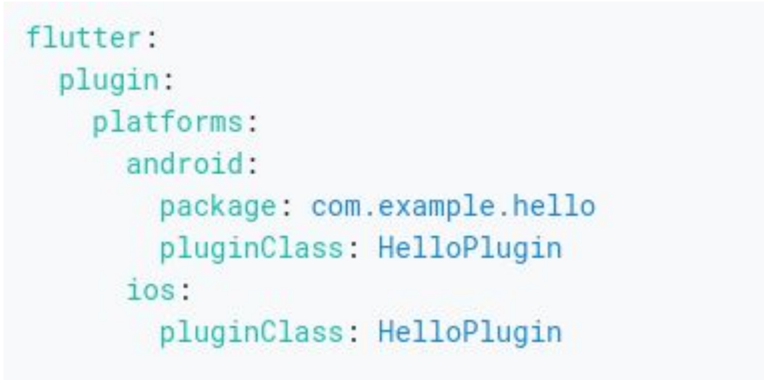
1. Create the package
    - **flutter create -a java --org com.example --template=plugin hello**
    - --org option to specify your organization, used as an identifier in generate plugin code.
    - -a option to specify language for Android
    - -i option to specify language for iOS
    - Plugin files

Developing Plugin Packages

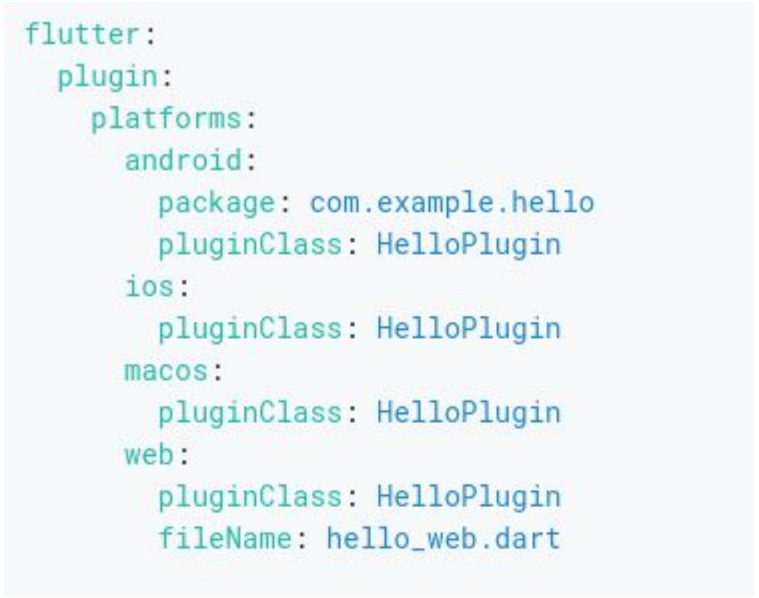
2. Implement the package
    a. Define the package API (.dart)
    b. Add Android platform code (.kt/java)
    c. Add iOS platform code (.swift/.h+.m
    d. Connect API and platform code
3. Using your plugin

VentureDive

Specifying a Plugin’s Supported Platform



```yaml
flutter:
  plugin:
    platforms:
      android:
        package: com.example.hello
        pluginClass: HelloPlugin
      ios:
        pluginClass: HelloPlugin
```



```yaml
flutter:
  plugin:
    platforms:
      android:
        package: com.example.hello
        pluginClass: HelloPlugin
      ios:
        pluginClass: HelloPlugin
      macos:
        pluginClass: HelloPlugin
      web:
        pluginClass: HelloPlugin
        fileName: hello_web.dart
```

Handling Package Interdependencies

- Package dependency

```
dependencies:
  url_launcher: ^5.0.0
```

- Android

```
android {
    // lines skipped
    dependencies {
        compileOnly rootProject.findProject(":url_launcher")
    }
}
```

- iOS

```
Pod::Spec.new do |s|
  # lines skipped
  s.dependency 'url_launcher'
```

VentureDive

Publishing Your Package

1. Publishing is forever
2. Preparing to publish
3. Run **flutter pub publish --dry-run** to see if everything passes analysis
4. Run **flutter pub publish** to publish on pub.dev

Flutter Build Modes

- Debug
  - flutter run
  - Used during development with hot reload option
- Profile
  - flutter run --profile
  - Used to analyze performance
- Release
  - flutter run --release or flutter build
  - Used when you are ready to release app

Product Flavors Flutter

- Define product flavor in Android
- Crete flavor configuration in Flutter
- Create main target file for each flavor in Flutter
- Create configuration in Android Studio
- Run app

Assignment

- Create a package with a creative content of your choice
- Publish the package on pub.dev
- Submit your package link on Google Chat group

VentureDive

End