Templates are only used for Engine code samples.

They should not be used in the Framework, since the code samples should reside in the `examples/api` directory.

If you are creating engine code samples, the following document may be of interest.

Eventually, Engine code samples will also be converted to point to separate files as the framework does.

Creating Code Snippets

In general, creating application snippets can be accomplished with the following syntax inside the dartdoc comment for a Flutter class/variable/enum/etc.:

```
/// {@tool snippet}
/// Any text outside of the code blocks will be accumulated and placed at the
/// top of the snippet box as a description. Don't try and say "see the code
/// above" or "see the code below", since the location of the description may
/// change in the future. You can use dartdoc [Linking] in the description, and
/// __Markdown__ too.
///
/// ```dart preamble
/// class Foo extends StatelessWidget {
///   const Foo({this.value = ''});
///
///   String value;
///
///   @override
///   Widget build(BuildContext context) {
///     return Text(value);
///   }
/// }
/// ```
/// This will get tacked on to the end of the description above, and shown above
/// the snippet.  These two code blocks will be separated by `///...` in the
/// short version of the snippet code sample.
/// ```dart
/// String myValue = 'Foo';
///
/// @override
/// Widget build(BuildContext) {
///   return const Foo(myValue);
/// }
/// ```
/// {@end-tool}
```

This will result in the template having the section that's inside "```dart" interpolated into the template's stateful widget's state object body.

For other sections of the template, the interpolation occurs by appending the string that comes after `code-` into the code block. For example, the `stateful_widget` template contains `{{code-imports}}`. To interpolate code into `{{code-imports}}`, you would have to do add the following:

```
/// ```dart imports
/// import 'package:flutter/rendering.dart';
/// ```
```

All code within a code block in a snippet needs to be able to be run through dartfmt without errors, so it needs to be valid code (This shouldn't be an additional burden, since all code will also be compiled to be sure it compiles).

Available Templates

The templates available for use as an argument to the snippets tool are as follows:

- `freeform` : This is a simple template for which you provide everything. It has no code of its own, just the sections for `imports` , `main` , and `preamble` . You must provide the `main` section to have a `main()` .

WidgetsApp Templates

These templates create a `WidgetsApp` that encloses the snippet widget. These templates import the widgets library.

- `stateful_widget` : The default code block will be placed as the body of the `State` object of a `StatefulWidget` subclass. Because the default code block is placed as the body of a stateful widget, you will need to implement the `build()` method and any state variables. It also has a `preamble` in addition to the default code block, which will be placed at the top level of the Dart file, so bare method calls are not allowed in the preamble. The default code block is placed as the body of a stateful widget, so you will need to implement the `build()` method, and any state variables. It also has an `imports` section to import additional packages. Please only import things that are part of Flutter or part of default dependencies for a `flutter create` project.
- `stateful_widget_ticker` : Identical to the `stateful_widget` template, with the addition of the `TickerProviderStateMixin` class, enabling easy generation of animated samples.
- `stateful_widget_restoration` : Similar to the `stateful_widget` template, but the widget also imports `RestorationMixin` and has a `restorationId` field which it uses to implement the `restorationId` getter on the `State` .
- `stateless_widget` : Identical to the `stateful_widget` template, except that the default code block is inserted as a method (which should be the `build` method) in a `StatelessWidget` . The `@override` before the build method is added by the template so must be omitted from the sample code.

MaterialApp Templates

These templates follow the same conventions as the `WidgetsApp` templates above but use a `MaterialApp` instead. These templates import the material library.

- `stateful_widget_material`
- `stateful_widget_material_ticker`
- `stateless_widget_material`

- `stateful_widget_restoration_material` : Similar to the `stateful_widget_restoration` template, but for `MaterialApp` .
- `stateless_widget_restoration_material` : Similar to the `stateless_widget` template, but the `MaterialApp` has `restorationScopeId` defined.
- `stateful_widget_scaffold` : Adds a `Scaffold` widget as the home of the enclosing `MaterialApp` to wrap the stateful widget snippet. The `Scaffold` widget contains an `AppBar` .
- `stateful_widget_scaffold_center` : Similar to `stateful_widget_scaffold` , except that it wraps the stateful widget with a `Center` .
- `stateful_widget_scaffold_center_freeform_state` : Similar to `stateful_widget_scaffold_center` except that the code block has to contain the entire state class defined as: `class _MyStatefulWidgetState extends State<MyStatefulWidget>`
- `stateless_widget_scaffold` : Similar to `stateless_widget_material` , except that it wraps the stateless widget with a `Scaffold` .
- `stateless_widget_scaffold_center` : Similar to `stateless_widget_scaffold` , except that it wraps the stateless widget with a `Center` .

CupertinoApp Templates

These templates follow the same conventions as the `WidgetsApp` templates above but use a `CupertinoApp` instead. These templates import the Cupertino library.

- `stateful_widget_cupertino`
- `stateful_widget_cupertino_ticker`
- `stateless_widget_cupertino`
- `stateful_widget_cupertinoPageScaffold` : Similar to `stateful_widget_cupertino` , except that it wraps the stateful widget with a `CupertinoPageScaffold` .
- `stateless_widget_cupertinoPageScaffold` : Similar to `stateless_widget_cupertino` , except that it wraps the stateless widget with a `CupertinoPageScaffold` .
- `stateless_widget_restoration_cupertino` : Similar to the `stateless_widget` template, but the `CupertinoApp` has `restorationScopeId` defined.
- `stateful_widget_restoration_cupertino` : Similar to the `stateful_widget_restoration` template, but for `CupertinoApp` .