Flutter Medium-Sized Sample Code

SUMMARY

Enable adding medium-sized API documentation samples as separate files so that they aren't part of the API documentation comments.

Author: Greg Spencer (gspencergoog)
Go Link: flutter.dev/go/medium-sized-code-samples
Created: 1/2021 / **Last updated:** 2/2021

OBJECTIVE

To enable medium-sized code samples to help document Flutter APIs without just shoehorning them into the dartdoc comments.

BACKGROUND

Flutter currently has several kinds of code samples. There are very short code samples, meant to show the usage of a widget or class, that are embedded in the dartdoc API documentation comments of the class. These are only tested to see that they pass the analyzer, they have no functional tests.

There are some medium-sized samples linked on the website, from the samples repo and the cookbook, etc, but those aren't well bound to the API documentation or a particular Flutter release, and they often focus on more than one concept or widget.

The last kind is a full-application sample that can be thousands of lines long, such as the Gallery app.

The objective of this document is to describe a way of producing samples that are somewhere between the two extremes, are focused in their scope, and are connected to the API documentation and a particular version of Flutter.

Glossary

- **Snippet** - A code block embedded in the Flutter API documentation comments.
- **DartPad** - An embeddable web component for running Flutter/Dart code.
- **Dartdoc** - A documentation system for Dart code that pulls the API documentation out of specially formatted comments in the code.
- **Medium-Sized** - An example of between 100 and 1000 lines of code, roughly.

OVERVIEW

The motivation driving this objective is that there are some concepts and widgets that could use more context in explaining them, but that causes them to be unwieldy when packed into the Dartdoc comments: the reader ends up paging through pages of code that isn't relevant, and it gets hard to maintain, test, and format.

Secondary goals include making these examples more testable and more easily edited, as well as enabling other uses of the examples outside of the API docs (e.g. linked from the website). Most samples should be displayable in Dartpad so that they can be interactive on the API docs website (however, some samples don't make sense in a Dartpad context).

Benefits we hope to achieve:

- Make more in-depth examples for APIs available
- Easier authoring of samples
- Testability of samples
- Increased reusability of samples in other contexts
- Create more, smaller samples, which is in line with conclusions from the User Research team.

DETAILED DESIGN/DISCUSSION

Example sizes

The current API documentation samples are an average of 51 lines long, and the longest is 229 lines, the shortest is 3 lines long.

What types of examples would we like to enable?

Larger. Certainly we'd like to have samples larger than the current API sample sizes:

they often are just showing the syntax for how to use them, and are not a realistic use case.

But also small enough: we don't want to get carried away with the examples, making them as large as the Gallery or many of our other sample apps: they're too large to absorb in a few minutes, don't focus on a single concept, and are hard to maintain and test.

The proposed guideline for "Medium-Sized" is between 100 and 1000 lines of code. Smaller than 100, and it can just be in an API doc comment. Larger than 1000, and it's just too large to wrap your head around. This is not a hard and fast rule, but a guideline: authors can use their judgement.

The samples should be focused examples of a single widget or concept.

Two Types of Examples

We really would like to take advantage of Dartpad in the API doc web pages: the ones we have are really popular and helpful in explaining concepts. At the same time, there are limitations that Dartpad must impose (mainly for UI complexity or security reasons), such as not allowing un-whitelisted packages, or multiple files.

This keeps us from being able to demonstrate the use of assets or images, and can make things more complicated to understand, since the concepts can't be broken into multiple files.

To address this, the proposal is to continue to have two kinds of examples: one which uses Dartpad, and one that does not. The non-Dartpad examples would be called "Standalone" samples. They would be built for the web, published as part of the API documentation website, and placed in iframes, without the ability to modify the source code. At the moment, we have a few of these standalone samples as the `{@tool sample}` type in the DartDocs, but they don't deal with multiple files or assets.

Source code location

Since these examples are linked to API documentation, they need to be matched with the APIs in the Flutter version that they will work with. Because of this, the proposal is to include the source code in the main Flutter repo, under the top-level `examples/` directory, in separate directories under `examples/api/`. The directory would contain both Dartpad and standalone examples. The contents of the subdirectories would determine which kind they were, and allow easy conversion from one to the other.

Placing the samples in the main Flutter repo has the drawback that landing a sample in the stable branch quickly isn't feasible, since we'd have to wait for the next release. There are multiple other mechanisms for publishing samples, however, so this is probably not too large of a loss.

Dartpad Samples

Each Dartpad sample will have its own subdirectory in the `examples/api` directory, containing a `test` directory and a `lib` directory. They will *not* contain an entire flutter application, with supporting platform shims and metadata, since that would increase the size of the repo unnecessarily, and have to be kept up-to-date with the newest `flutter create` templates. Users will be able to run "`flutter create .`" in the directory and run the sample from that location.

There must only be a `main.dart` file in the lib directory, and no other files.

There must not be a `pubspec.yaml` file, these will be created automatically by `flutter create` during testing so that we won't need to keep them all up to date with the framework revision, and to avoid needing to have `flutter update-packages` run on them. It also prevents the use of external packages, which is desirable for running in Dartpad.

It is tempting to include a `README.md` containing detailed explanatory text, but to avoid duplicating the explanatory text the explanatory text must reside in the API documentation comments, to make it more readable when reading the source code, and so that source code readers know what the example contains so that they know if they want to explore it or not.

Dartpad samples must not contain assets like fonts and images, since those are hard to safely include in a Dartpad context.

Standalone Samples

These are non-Dartpad samples which would contain a `lib` and `test` directory, as well as a `pubspec.yaml` and any additional assets in an `assets` directory. They would be required to have a `lib/main.dart` file, but also be allowed more than one file in the `lib` directory. The supporting platform shims would *not* be checked in.

They are still recommended to be less than 1000 lines of code, when summing up all source files.

Since they include a `pubspec.yaml`, this pubspec will need to contain all of the autogenerated boilerplate that is built with the `flutter update-packages` command. Exempting these directories from `update-packages` is undesirable because it means that the SDK constraints in the pubspec would get stale. From an educational point of view, it's too bad that they are so cluttered, and people may wonder why their own projects don't have those lines. Maybe we can modify the update-packages command to treat these files differently, and exempt them from version pinning and the other auto-generated dependencies, but still update SDK constraints.

When displayed in the API documentation webpage, standalone samples would present a tabbed frame that defaults to the web version of the sample running in an iframe, and tabs for each of the source files in the sample, as well as links for downloading the entire sample as a zip file.

API Documentation linking

In the API documentation, the snippets tool will support a new syntax for describing the source code:

In addition to:

```
/// {@tool dartpad --template=freeform}
/// This example shows FooWidget in action...
///
/// ```dart
/// // The code...
/// ```
/// {@end-tool}
```

It will also be possible to do this:

```
/// {@tool dartpad --template=freeform}
/// This example shows FooWidget in action...
///
/// [Visit sample code in the examples/api/foo_widget directory]
/// {@end-tool}
```

Which would bind it to the code in examples/api/foo_widget.

The idea is that the new syntax would be required to be exactly the string [Visit sample code in the examples/api/foo_widget directory] (with the foo_widget part changing for each sample, and allowing any kind of whitespace between words) so that the syntax is both explicit and self-documenting for the uninitiated user. It could also be turned into links by the IDEs.

In a {@tool dartpad} or {@tool sample} block, there must be no instance of triple-back quotes (```) in the block. Sample names must be unique across all samples, and be valid pub package names.

The snippets tool that currently processes the sample code would be enhanced to support the new syntax, load the sample source code from the appropriate directory, and supply it to dartdoc for publishing in the same way that it does for existing samples.

Standalone samples would use identical syntax as Dartpad samples, but instead of `{@tool dartpad}`, they would use `{@tool sample}`.

Testing plan

Each sample will include a `test` subdirectory that would include at least one test file that tests the sample against the version of flutter that it is associated with. Continuous integration presubmit tests will run all of the tests, after running `flutter create .` and `flutter pub get` in each of the sample directories.

OPEN QUESTIONS

- How should we integrate with the IDEs so that we don't lose the benefit of having the sample code embedded in the source code? Maybe they can linkify that `Visit sample` line in the comment?
- Should the sample `README.md` file contain explanatory text about the example or should that remain only in the API documentation?
- ~~Should we create two different kinds of samples, one designed for Dartpad, and one designed to be published as an iframe and not be modified on the web page?~~ Answer: **Yes**.
    - ~~This would allow some samples to show customizing of startup options, include packages and assets, and be formed of more than one source file.~~