RIVERPOD

FLUTTER STATE-MANAGEMENT

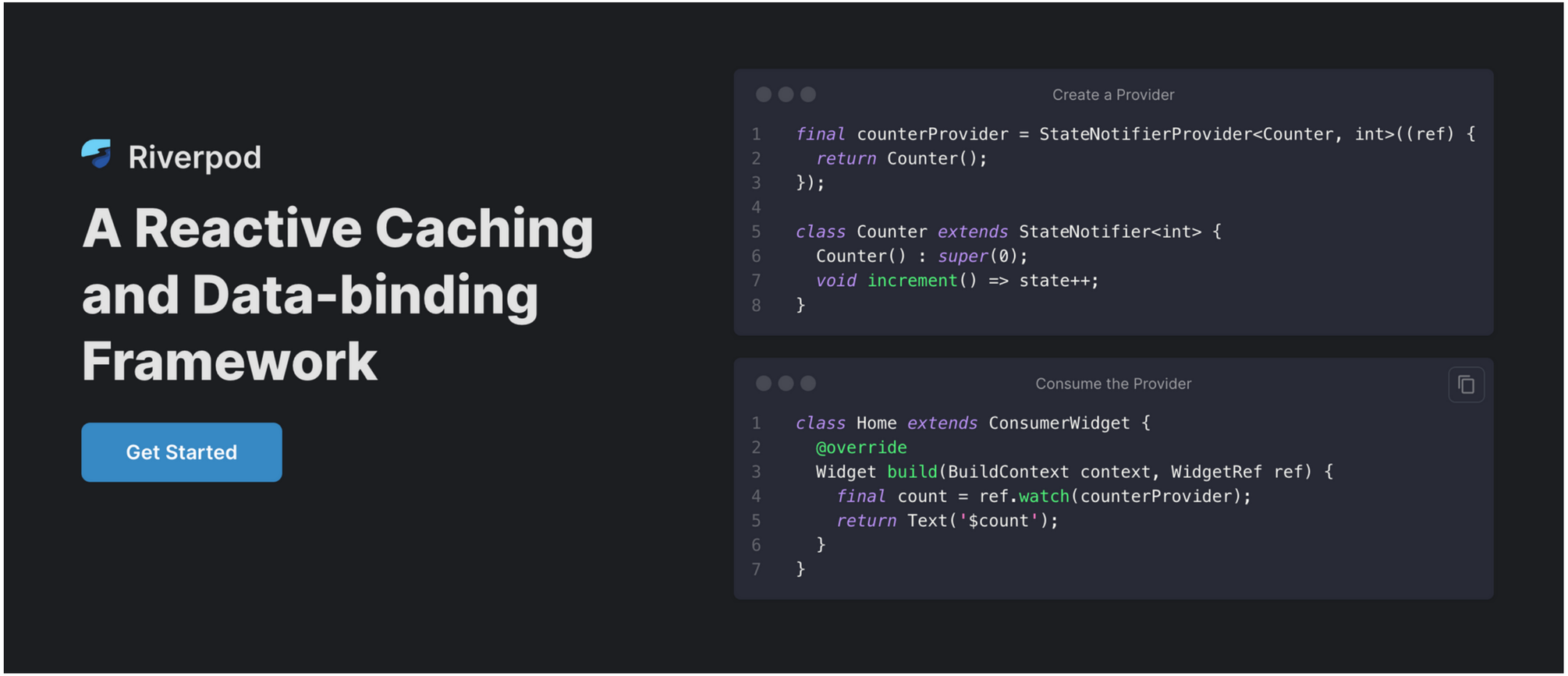
Tushar Asodariya

@tushar_asodaria

State Management

- WHERE THE USER IS IN AN APP,
- WHAT THEY ARE DOING,
- WHAT DATA THEY ARE INPUTTING, ETC.

riverpod.dev



PACKAGES

riverpod	
flutter_riverpod	
hooks_riverpod	

Dart

Flutter

Riverpod with

flutter_hooks

Getting Started

Step: 1

Add *flutter_riverpod* to *pubspec.yaml*

pubspec.yaml

```yaml
name: my_app_name
environment:
  sdk: ">=2.17.0 <3.0.0"
  flutter: ">=3.0.0"

dependencies:
  flutter:
    sdk: flutter
  flutter_riverpod: ^2.0.2
```

Step: 2

Wrap entire application with *ProviderScope* widget

```dart
void main() {
  runApp(
    // For widgets to be able to read providers, we need to wrap the entire
    // application in a "ProviderScope" widget.
    // This is where the state of our providers will be stored.
    ProviderScope(
      child: MyApp(),
    ),
  );
}
```

Basics

of

Riverpod

1. StatelessWidget

ConsumerWidget

```dart
class Home extends StatelessWidget {
  const Home({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

```dart
class Home extends ConsumerWidget {
  const Home({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context, WidgetRef ref) {
    return Container();
  }
}
```

StatefulWidget

```dart
class Home extends StatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  State<Home> createState() => _HomeState();
}

class _HomeState extends State<Home> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

ConsumerStatefulWidget

```dart
class Home extends ConsumerStatefulWidget {
  const Home({Key? key}) : super(key: key);

  @override
  ConsumerState<Home> createState() => _HomeState();
}

class _HomeState extends ConsumerState<Home> {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```

2. Providers

CODE

Types Of Providers

StateNotifierProvider

It listens to and expose StateNotifier.

CODE

FutureProvider

- performing and caching asynchronous operations (such as network requests)
- nicely handling error/loading states of asynchronous operations
- combining multiple asynchronous values into another value

CODE

StreamProvider

- listening to Firebase or web-sockets

- rebuilding another provider every few seconds

StateProvider

expose a way to modify state without StateNotifier.

CODE

Modifiers

.family

For example, we can combine `family` with FutureProvider to fetch a `Message` from its ID:

```
final messagesFamily = FutureProvider.family<Message, String>((ref, id) async {
  return dio.get('http://my_api.dev/messages/$id');
});
```

.autoDispose

To tell Riverpod to destroy the state of a provider when it is no longer used, simply append `.autoDispose` to your provider:

```
final userProvider = StreamProvider.autoDispose<User>((ref) {

});
```

That's All

FOR RIVERPOD

References

- https://riverpod.dev/docs/getting_started
- https://www.desuvit.com/state-management-in-flutter-a-comprehensive-guide/
- https://codewithandrea.com/articles/flutter-state-management-riverpod/