Flutter HooksとRiverpodの解説

Press Space for next page →

自己紹介

- 📝 飯野陽平（wheatandcat）
- 🏢 会社: 合同会社UNICORN 代表社員
- 📚 Blog: https://www.wheatandcat.me/
- 🛠️ 今までに作ったもの
  - memoir
  - OOMAKA
  - MarkyLinky

Flutterとは

- Google開発のオープンソースのマルチプラットフォームの開発フレームワーク
- 一つのコードベースから、**iOS、Android、Web、Windows、Mac、Linuxアプリ**など複数のプラットフォームの作成が可能
- 言語は**Dart**を使用
- Google独自のUI Widgetを使用（Material Designベース）

Flutter Hooksとは

- React HooksのFlutter版
  - flutter_hooks
- コンポーネントの状態を管理するためのライブラリ
- Flutter標準の**Stateful Widget**のコードを簡潔に書くことができる

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```dart
class Count extends StatefulWidget {
  @override
  State<Count> createState() ⇒ _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () ⇒ {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```dart
class Count extends StatefulWidget {
  @override
  State<Count> createState() ⇒ _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () ⇒ {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```dart
class Count extends StatefulWidget {
  @override
  State<Count> createState() => _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () => {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```dart
class Count extends StatefulWidget {
  @override
  State<Count> createState() ⇒ _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () ⇒ {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```dart
class Count extends StatefulWidget {
  @override
  State<Count> createState() ⇒ _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () ⇒ {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

StatefulWidgetの例

- ①. StatefulWidget継承したクラスを宣言
- ②. Stateクラスを宣言
- ③. Stateの初期化
- ④. Stateの更新
- ⑤. Stateの参照

```
class Count extends StatefulWidget {
  @override
  State<Count> createState() ⇒ _CountState();
}

class _CountState extends State<Count> {
  int _counter = 0;

  @override
  Widget build(BuildContext context) {
    return TextButton(
      onPressed: () ⇒ {
        setState(() {
          _counter++;
        }),
      },
      child: Text('Increment: $_counter'),
    );
  }
}
```

Flutter Hooksの例

- ①. HookWidgetを継承したクラスを宣言
- ②. Stateの初期化
- ③. Stateの更新
- ④. Stateの参照

```
class Count extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final count = useState(0);
    return TextButton(
        onPressed: () => count.value++,
        child: Text('Increment: ${count.value}'),
      );
  }
}
```

Flutter Hooksの例

- ①. HookWidgetを継承したクラスを宣言
- ②. Stateの初期化
- ③. Stateの更新
- ④. Stateの参照

```
class Count extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final count = useState(0);
    return TextButton(
        onPressed: () ⇒ count.value++,
        child: Text('Increment: ${count.value}'),
      );
  }
}
```

Flutter Hooksの例

- ①. HookWidgetを継承したクラスを宣言
- ②. Stateの初期化
- ③. Stateの更新
- ④. Stateの参照

```
class Count extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final count = useState(0);
    return TextButton(
        onPressed: () ⇒ count.value++,
        child: Text('Increment: ${count.value}'),
      );
  }
}
```

Flutter Hooksの例

- ①. HookWidgetを継承したクラスを宣言
- ②. Stateの初期化
- ③. Stateの更新
- ④. Stateの参照

```
class Count extends HookWidget {
  @override
  Widget build(BuildContext context) {
    final count = useState(0);
    return TextButton(
        onPressed: () => count.value++,
        child: Text('Increment: ${count.value}'),
      );
  }
}
```

useEffectの解説

- Widgetのライフサイクルで処理をさせたい時にuseEffectを使ってみる
  - 生成時、破棄時、更新時
- 概念的にはReactのuseEffectと同じ

useEffectの例

```
final id = useState<int>(0);

useEffect(() {
  // ①. idが変更されたら実行
  getItems(id.value);
}, [id.value]);

useEffect(() {
  // ②. 初回のみ実行
    debugPrint('初回のみ実行');
}, const []);

useEffect(() {
  // ③. 破棄時に実行
  return () ⇒ {
    debugPrint('破棄時に実行');
  };
}, [id.value]);
```

useEffectの例

```
final id = useState<int>(0);

useEffect(() {
  // ①. idが変更されたら実行
  getItems(id.value);
}, [id.value]);

useEffect(() {
  // ②. 初回のみ実行
    debugPrint('初回のみ実行');
}, const []);

useEffect(() {
  // ③. 破棄時に実行
  return () => {
    debugPrint('破棄時に実行');
  };
}, [id.value]);
```

useEffectの例

```
final id = useState<int>(0);

useEffect(() {
  // ①. idが変更されたら実行
  getItems(id.value);
}, [id.value]);

useEffect(() {
  // ②. 初回のみ実行
    debugPrint('初回のみ実行');
}, const []);

useEffect(() {
  // ③. 破棄時に実行
  return () => {
    debugPrint('破棄時に実行');
  };
}, [id.value]);
```

useEffectの例

```
final id = useState<int>(0);

useEffect(() {
  // ①. idが変更されたら実行
  getItems(id.value);
}, [id.value]);

useEffect(() {
  // ②. 初回のみ実行
    debugPrint('初回のみ実行');
}, const []);

useEffect(() {
  // ③. 破棄時に実行
  return () ⇒ {
    debugPrint('破棄時に実行');
  };
}, [id.value]);
```

Custom Hookの例

- コード量が多くなってくると状態管理とViewのコードを分割したくなる
- その時にCustom Hookを使うと便利
- コードの例は以下を参照
  - コードのURL
- 差分は以下の通り
  - 差分のURL

Riverpodとは

- Riverpod
- グローバルな状態管理をしたい時に使うライブラリ
- ReactでいうところのContext APIのようなもの

Riverpodの例

- ①. Providerを宣言
- ②. Providerを取得
- ③. Providerの値を参照
- ④. Providerの値を更新
- Demo用のURL

■ lib/providers/counter.dart

```dart
final countProvider = StateProvider((ref) => 0);
```

■ lib/main.dart

```dart
class HomePage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(countProvider);

    return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('$count'),
            FloatingActionButton(
              onPressed: () => ref.read(countProvider.notif
              child: Icon(Icons.add),
            ),
          ],
        ),
    );
  }
}
```

Riverpodの例

- ①. Providerを宣言
- ②. Providerを取得
- ③. Providerの値を参照
- ④. Providerの値を更新
- Demo用のURL

■ lib/providers/counter.dart

```dart
final countProvider = StateProvider((ref) => 0);
```

■ lib/main.dart

```dart
class HomePage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(countProvider);

    return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('$count'),
            FloatingActionButton(
              onPressed: () => ref.read(countProvider.notif
              child: Icon(Icons.add),
            ),
          ],
        ),
    );
  }
}
```

Riverpodの例

- ①. Providerを宣言
- ②. Providerを取得
- ③. Providerの値を参照
- ④. Providerの値を更新
- Demo用のURL

■ lib/providers/counter.dart

```dart
final countProvider = StateProvider((ref) => 0);
```

■ lib/main.dart

```dart
class HomePage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(countProvider);

    return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('$count'),
            FloatingActionButton(
              onPressed: () => ref.read(countProvider.notif
              child: Icon(Icons.add),
            ),
          ],
        ),
    );
  }
}
```

Riverpodの例

- ①. Providerを宣言
- ②. Providerを取得
- ③. Providerの値を参照
- ④. Providerの値を更新
- Demo用のURL

■ lib/providers/counter.dart

```dart
final countProvider = StateProvider((ref) => 0);
```

■ lib/main.dart

```dart
class HomePage extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final count = ref.watch(countProvider);

    return Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text('$count'),
            FloatingActionButton(
              onPressed: () => ref.read(countProvider.notif
              child: Icon(Icons.add),
            ),
          ],
        ),
    );
  }
}
```

まとめ

- Flutterの状態管理で**Flutter Hooks**と**Riverpod**を触ってみたが、思っていた以上にReactと同じ方式だったので、Reactの知識があればすぐに使える
- Flutterがクラスベースで作成されている言語なのに、Hooksが関数ベースで作成されているのが、若干ちぐはぐな感じはしている
- ただ、riverpod_generatorのようなRiverpodの宣言自体を自動生成できる部分はReactよりも進んでいるかも
  - Providerの宣言の種類が多すぎるのでコード生成すると楽
  - 各Providerの役割と使い分け | Flutter x Riverpod でアプリ開発！実践入門

ご清聴ありがとうございました🎉