Flutter 状态管理之 Provider


mclzone LV.3
2019年05月18日 01:41 · 阅读 13299

关注

前言：

Provider 是 Google I/O 2019 大会上宣布的现在官方推荐的状态管理方式，我在学习和使用之后感觉很不错，因为它真的很容易上手,所以就写篇博客记录一下吧！！

Provider GitHub 地址：github.com/rrousselGit...

再贴两个油管上的视频（第一个只讲了 ChangeNotifierProvider，没有讲 MultiProvider 的情况）：

www.youtube.com/watch?v=xcS...

www.youtube.com/watch?v=d_m...

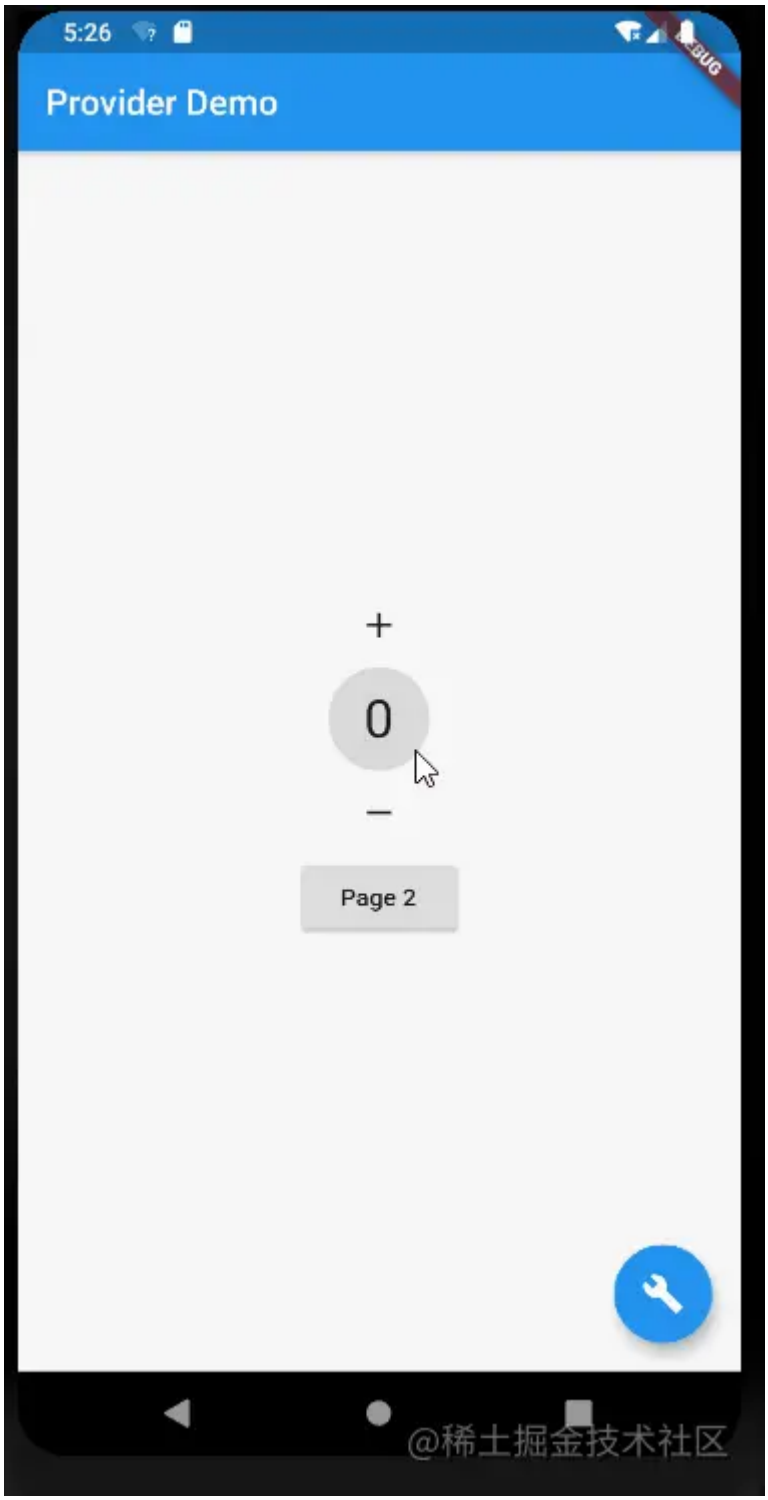
OK，那废话不多说。下面具体看看在 Flutter 中如何使用 Provider 做状态管理。

我将通过一个小 demo 来展示如何使用 Provider 做状态管理。分别使用单个 Provider 和 MultiProvider 实现（第一种将两个数据放在同一个类中，第二种是将两个数据拆分到两个不同的类中）。

该 demo 的功能很简单，计数 + 切换主题。仅有两个页面，两个页面共享相同的数据。

我已经将这个小 demo 的代码上传至 GitHub：github.com/MzoneCL/Flu... (MultiProvider) github.com/MzoneCL/Flu... （Single Provider）

看一下下效果图：



下面就来分别讲讲单个和多个 Provider 分别是如何实现上面功能的吧。

一. 单个 Provider 的情况

ChangeNotifierProvider 可以说是 Provider 的一种。使用它来管理只有一个共享数据类的情况比较方便。

第一步： 添加依赖

在 pubspec.yaml 文件中添加依赖。

provider 包 pub 地址：pub.dev/packages/pr...

第二步：创建共享数据类

ini 复制代码

```
class DataInfo with ChangeNotifier {
  int _count = 0;
  ThemeData _themeData = ThemeData.light();

  get count => _count;
  get themeData => _themeData;

  addCount() {
    _count++;
    notifyListeners();
  }
  subCount() {
    _count--;
    notifyListeners();
  }
  changeTheme() {
    if (_themeData == ThemeData.light()) {
      _themeData = ThemeData.dark();
    } else {
      _themeData = ThemeData.light();
    }
    notifyListeners();
  }
}
```

数据类需要 with ChangeNotifier 以使用 notifyListeners() 函数通知监听者以更新界面。

第三步：访问数据

Provider 获取数据状态有两种方式：

1. 使用 Provider.of<T>(context)
2. 使用 Consumer

不过这两种方式都需要在顶层套上 ChangeNotifierProvider():

1. 使用 Provider.of<T>(context)

例如，指定主题部分的代码如下：

scala 复制代码

```scala
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    var dataInfo = Provider.of<DataInfo>(context);
    return MaterialApp(
      home: MyHomePage(),
      theme: dataInfo.themeData,
    );
  }
}
```

通过Provider.of<DataInfo>(context) 获取 DataInfo 实例，需要在 of 函数后指明具体的数据类。然后就可以直接通过 getter 访问到 themeData 了。

2.使用 Consumer

同样，以指定主题部分代码为例：

scala 复制代码

```scala
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Consumer<DataInfo>(
      builder: (context, dataInfo, _) => MaterialApp(
            home: MyHomePage(),
            theme: dataInfo.themeData,
          ),
    );
```

```
    }
  }
```

直接用 Consumer 包住需要使用共享数据的 Widget，同样的，Consumer 也要指明类型。

二. 使用 MultiProvider 管理多个共享数据类

这个可以搭配 Stream 使用。有关 Stream 相关的讲解，这位大佬(@Vadaski)的文章很棒：juejin.cn/post/684490...

第一步：添加依赖

同上，依然是添加 provider 包。

第二步：创建共享数据类

我这里创建了两个类：CounterBloc 和 ThemeDataBloc 分别用于管理计数值和ThemeData。（从类的名字就能看出，其实是借鉴了BLoC的思想的）

ini 复制代码

```
class CounterBloc {
  StreamController<int> _streamController;
  Stream<int> _stream;
  int _count;

  CounterBloc() {
    _count = 0;
    _streamController = StreamController.broadcast();
    _stream = _streamController.stream;
  }

  Stream<int> get stream => _stream;
  int get count => _count;

  addCounter() {
    _streamController.sink.add(++_count);
  }
  subCounter() {
    _streamController.sink.add(--_count);
  }
  dispose() {
    _streamController.close();
```

```
  }
}
```

由于 CounterBloc 在两个页面都有监听，所以这里的 _streamController 需要是广播类型的，需要支持多订阅，否则会报错。

ini 复制代码

```
class ThemeDataBloc {
  StreamController<ThemeData> _streamController;
  Stream<ThemeData> _stream;
  ThemeData _themeData;

  ThemeDataBloc() {
    _themeData = ThemeData.light();
    _streamController = StreamController();
    _stream = _streamController.stream;
  }

  Stream<ThemeData> get stream => _stream;

  changeTheme() {
    _themeData = _themeData == ThemeData.light()?ThemeData.dark():ThemeData.light();
    _streamController.sink.add(_themeData);
  }
  dispose() {
    _streamController.close();
  }
}
```

由于这里使用的 Stream，就不用像上面那样 with ChangeNotifier 了。

第三步：在应用顶层放置 MultiProvider

MultiProvider 有一个必填的参数：providers，我们需要给它传入一个 Provider 列表。 这样，我们就可以在所有子 Widget 访问到相关共享数据了。

less 复制代码

```
main() {
  var counterBloc = CounterBloc();
  var themeDataBloc = ThemeDataBloc();
  runApp(MultiProvider(providers: [
    Provider<CounterBloc>.value(value: counterBloc),
    Provider<ThemeDataBloc>.value(value: themeDataBloc),
  ], child: MyApp()));
}
```

第四步：访问数据

使用 Provider.of<T>(context) 获取指定类型的数据。

还是以指定主题的代码为例：

scala 复制代码

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return StreamBuilder(
      builder: (context, snapshot) {
        return MaterialApp(
          home: MyHomePage(),
          theme: snapshot.data,
        );
      },
      initialData: ThemeData.light(),
      stream: Provider.of<ThemeDataBloc>(context).stream,
    );
  }
}
```

同样需要在 of 函数后面指明类型才能找到具体的 Provider。

好了，以上就是 Flutter 中 Provider 做状态管理的基本使用啦！

分类： 前端 标签： Flutter