

使用 InheritedWidget 对豆瓣电影 App 进行重构

InheritedWidget 是 Flutter 中非常重要的一个功能型 Widget，它可以高效的将数据在 Widget 树中向下传递、共享，因此 InheritedWidget 可以对全局状态进行管理。

本节就使用 InheritedWidget 对豆瓣电影 App 进行重构。InheritedWidget 管理全局状态，本地状态还是使用 StatefulWidget 和 setState()。

重构后的工程路径

StateManager/flutter_doubanmovie_inheritedwidget

使用 InheritedWidget 重构

InheritedWidget 的使用方法：

1. 定义 InheritedWidget，InheritedWidget 里存储全局的数据。
2. InheritedWidget 必须是父 Widget。
3. InheritedWidget 的子 Widget 通过特定的方法取到 InheritedWidget 的实例，然后就可以访问 InheritedWidget 的数据。
4. 当全局数据发生变化时，需要重新创建 InheritedWidget，重建的同时会根据条件通知 InheritedWidget 的子 Widget。

在豆瓣电影 App 里的 curCity 是全局状态，所以首先定义一个 ShareDataInheritedWidget：

```

class ShareDataInheritedWidget extends
InheritedWidget{
    String curCity ;

    ShareDataInheritedWidget(this.curCity,{Widget
child}):super(child:child);

    @override
    bool updateShouldNotify(InheritedWidget
oldWidget) {
        // TODO: implement updateShouldNotify
        return (oldWidget as
ShareDataInheritedWidget).curCity != curCity;
    }
}

```

curCity 就是要存储的全局数据，而且要通过构造函数传入。

updateShouldNotify() 方法是，当全局数据发生变化，InheritedWidget 发生重建，判断需不需要通知依赖 InheritedWidget 数据的子 Widget，返回 true 是通知，返回 false 是不通知，这里写的是：

```

return (oldWidget as
ShareDataInheritedWidget).curCity != curCity;

```

意思是当旧的数据和新的数据不一致时，就返回 true，否则返回 false。

因为 HotWidget、HotMoviesListWidget、CitysWidget 都需要 curCity，所以要让 ShareDataInheritedWidget 是这几个 Widget 的父 Widget。如下：

```

class _MyHomePageState extends State<MyHomePage>
{
    ...

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            body: ShareDataInheritedWidget(
                '深圳',
                child: _widgetItems[_selectedIndex],
            ), //选中不同的选项显示不同的界面,
            ...
        );
    }

    ...
}

```

写完之后，HotWidget、HotMoviesListWidget、CitysWidget 就可以通过特定的方法访问到 ShareDataInheritedWidget 里的 curCity，这个特定的方法是：

```

context.inheritFromWidgetOfExactType(Type
targetType)

```

context 就是子 Widget 的 context，Type 是类型，就是要访问的 ShareDataInheritedWidget，所以要得到 ShareDataInheritedWidget 的实例就是：

```

context.inheritFromWidgetOfExactType(ShareDataInh
heritedWidget);

```

因为子 Widget 中要经常获取 ShareDataInheritedWidget 的实例，所以可以把这个方法写在 ShareDataInheritedWidget 里：

```
class ShareDataInheritedWidget extends
InheritedWidget {
    ...
    //定义一个便捷方法，方便子树中的 Widget 获取
ShareDataInheritedWidget 实例
    static ShareDataInheritedWidget of(BuildContext
context) {
        return
context.inheritFromWidgetOfExactType(ShareDataInh
heritedWidget);
    }
    ...
}
```

这样在子 Widget 中获取 ShareDataInheritedWidget 的实例就是：

```
ShareDataInheritedWidget.of(context);
```

获取 curCity 就是：

```
ShareDataInheritedWidget.of(context).curCity;
```

现在对 HotWidget 进行重构，因为 curCity 现在是存储在 ShareDataInheritedWidget 里，所以把 HotWidget 里的 curCity 变量删掉，用到 curCity 的地方，用 ShareDataInheritedWidget.of(context).curCity 代替，同时把 curCity 读取的逻辑从 HotWidget 放到 _MyHomePageState 里。

_MyHomePageState 的代码就变为：

```
class _MyHomePageState extends State<MyHomePage>
{
```

```

...
String _curCity;

@override
void initState() {
    // TODO: implement initState
    super.initState();
    initData();
}

void initData() async {
    final prefs = await
SharedPreferences.getInstance(); //获取 prefs

    String city = prefs.getString('curCity'); //获
    取 key 为 curCity 的值

    if (city != null && city.isNotEmpty) {
        //如果有值
        setState(() {
            _curCity = city;
        });
    } else {
        //如果没有值，则使用默认值
        setState(() {
            _curCity = '深圳';
        });
    }
}

@override
Widget build(BuildContext context) {
    return Scaffold(

```

```
body: ShareDataInheritedWidget(  
  _curCity, //默认值  
  child: _widgetItems[_selectedIndex],  
) , //选中不同的选项显示不同的界面,  
  ...  
);  
}  
...  
}
```

接下来对 HotMoviesListWidget 进行重构，之前 HotMoviesListWidget 的构造函数里需要传入 curCity，现在就不需要了，HotMoviesListWidget 里的 curCity 变量全都用 ShareDataInheritedWidget.of(context).curCity 代替。

这里还有一个问题要解决，HotMoviesListWidget 的代码：

```
@override
void initState() {
  // TODO: implement initState
  super.initState();
  _getData();
}

void _getData() async {
  List<HotMovieData> serverDataList = new
List();
  var response = await http.get(
    'https://api.douban.com/v2/movie/in_theaters?
apikey=0b2bdeda43b5688921839c8ecb20399b&city=' +
ShareDataInheritedWidget.of(context).curCity +
    '&start=0&count=10');
  //成功获取数据
  if (response.statusCode == 200) {
    var responseJson =
json.decode(response.body);
    for (dynamic data in
responseJson['subjects']) {
      HotMovieData hotMovieData =
HotMovieData.fromJson(data);
      serverDataList.add(hotMovieData);
    }
    setState(() {
      hotMovies = serverDataList;
    });
  }
}
```

因为 `_getData()` 里的 `ShareDataInheritedWidget.of(context).curCity` 用到了 `context`，所以 `_getData()` 不能放在 `initState()` 里调用，因为在 `initState()` 里 `context` 还不能使用，所以要重构成：

```
@override
void initState() {
  // TODO: implement initState
  super.initState();
}

@override
void didChangeDependencies() {
  // TODO: implement didChangeDependencies
  super.didChangeDependencies();
  _getData();
}

void _getData() async {
  ...
}
```

`didChangeDependencies()` 方法会在它依赖的数据发生变化的时候调用，而这里 `HotMoviesListWidget` 依赖的数据就是其父 `Widget` `ShareDataInheritedWidget` 的数据，`didChangeDependencies()` 调用的条件就是 `ShareDataInheritedWidget` 的 `updateShouldNotify()` 方法返回 `true`。

最后对 `CitysWidget` 进行重构，涉及到 `CitysWidget` 的代码是：从 `HotWidget` 里点击，携带当前城市的数据跳转到 `CitysWidget`，在 `CitysWidget` 里选完数据后返回 `HotWidget`，并刷新 `HotWidget`：


```

void _jumpToCitysWidget() async {
    var selectCity =
        await Navigator.pushNamed(context,
'/Citys', arguments:
ShareDataInheritedWidget.of(context).curCity);
    if (selectCity == null) return;

    final prefs = await
SharedPreferences.getInstance();
    prefs.setString('curCity', selectCity); //存取
数据

    setState(() {

ShareDataInheritedWidget.of(context).curCity =
selectCity;
    });
}

```

这里去实际操作，你会发现，HotWidget 里所选城市的数据没有变化，数据也没有重新刷新，因为
ShareDataInheritedWidget.of(context).curCity =
selectCity; 用法是错误的，虽然你改了
ShareDataInheritedWidget 里的数据，但是
ShareDataInheritedWidget 没有重建，所以子 Widget 也不会收到
didChangeDependencies(), 那如何让
ShareDataInheritedWidget 重建呢？

使用 InheritedWidget 是无法做到的，因为 InheritedWidget 只能
把数据在 Widget 树中向下传递，下面的数据是无法向上传递，所以
下面 curCity 的变化，无法通知到 InheritedWidget。

总结

通过 InheritedWidget 对豆瓣电影 App 的重构，我们可以发现 InheritedWidget 的优点：

- 可以对全局状态进行管理

但是，也有很多的缺点：

- UI 逻辑和业务逻辑没有分开
- 无法管理本地状态
- 数据只能从上到上传递，无法从下到上传递
- 随着 App 变大，代码维护也会变得越来越难。

所以，不要使用 InheritedWidget 对状态进行管理。