Flutter之Stream、StreamController的使用(一)

卢叁

在 Flutter 中，Stream 用于处理一个异步事件序列。Stream 可以生成一系列的异步事件，这些事件可以包含数据、错误信息、或者标识流的结束的信号。你可以把它理解为时间上的序列或者流，可以监听它并对其上的事件作出响应。

创建 `Stream`

1. 单次事件的 `Stream`

   如果你有一个单次事件的 `Future`，你可以使用它来产生一个 `Stream`。

```
final myStream = Stream.fromFuture(Future.value('Hello, Streams!'));
```

2. 周期性生成事件的 `Stream`

   你也可以创建一个周期性生成事件的 `Stream`。

```
final periodicStream = Stream.periodic(Duration(seconds: 1), (computationCount) {
  return "This is number $computationCount";
});
```

监听 `Stream`

要从 `Stream` 中获取数据，你需要订阅（或者监听）它。

```
final subscription = myStream.listen((data) {
  print('Data from stream: $data');
});
```

你还可以监听 Stream 上的错误和结束信号。

```
final subscription = myStream.listen(
  (data) {
    print('Data from stream: $data');
  },
  onError: (error) {
    print('Error: $error');
  },
  onDone: () {
    print('Stream is done!');
  },
);
```

使用 `StreamBuilder`

在 `Flutter` 中，你通常会用 `StreamBuilder` 来构建基于 `Stream` 的 UI。

```
StreamBuilder<String>(
  stream: myStream,
  builder: (BuildContext context, AsyncSnapshot<String>
snapshot) {
    if (snapshot.hasError) {
      return Text('Error: ${snapshot.error}');
    }
    switch (snapshot.connectionState) {
```

```
      case ConnectionState.none:
        return Text('Select lot');
      case ConnectionState.waiting:
        return Text('Awaiting data...');
      case ConnectionState.active:
        return Text('Active data: ${snapshot.data}');
      case ConnectionState.done:
        return Text('Stream was done');
    }
  },
)
```

`StreamController`

你可以使用 `StreamController` 来手动控制 `Stream`。

```
final controller = StreamController<String>();

final subscription = controller.stream.listen(
  (data) {
    print('Data from stream: $data');
  },
  onError: (error) {
    print('Error: $error');
  },
  onDone: () {
    print('Stream is done!');
  },
);

controller.sink.add('This is a data event');
controller.sink.addError('This is an error event');
controller.sink.close();
```

注意要关闭创建的 `StreamController`，以防止内存泄漏。

总结

`Stream` 是一个事件序列，可以发送数据、错误或完成事件。
使用 `.listen()` 可以监听 `Stream` 上的事件。
在 `Flutter` 中，你可以使用 `StreamBuilder` 来构建基于 `Stream` 的 UI。
使用 `StreamController` 可以手动控制 `Stream` 的事件。
记得在不需要监听 `Stream` 的时候取消订阅以防止内存泄露。