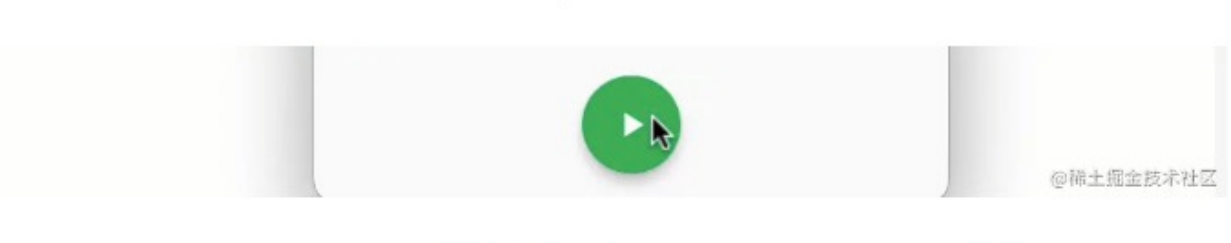
一、秒表运行状态的维护

现在先实现底部工具按钮点击时的状态变化，效果如下：

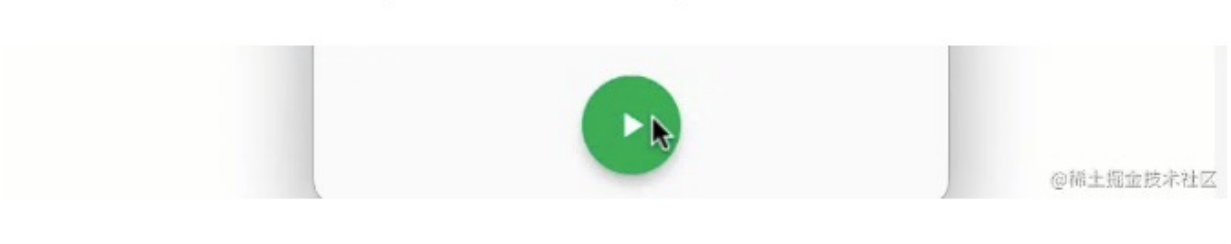

@稀土掘金技术社区

在上一章时，底部按钮被封装为 `ButtonTools` 组件，按钮的展现效果由 `state` 状态决定。现在点击按钮时，需要对运行状态进行变化，并更新界面显示。这和计数器项目中，界面上数值变化是异曲同工的，只不过这里需要变化的 `状态量` 是 `StopWatchType` 对象。

如下，在 `_HomePageState` 中定义 `StopWatchType` 类型的状态量 `_type` ，在构建 `ButtonTools` 时将 `_type` 变量作为入参，决定界面显示：

```dart
---->[timer/04/home_page/home_page.dart#_HomePageState]----
StopWatchType _type = StopWatchType.none;

Widget buildTools() {
  return ButtonTools(
    state: _type,
    onRecoder: onRecoder,
    onReset: onReset,
    toggle: toggle,
  );
}
```

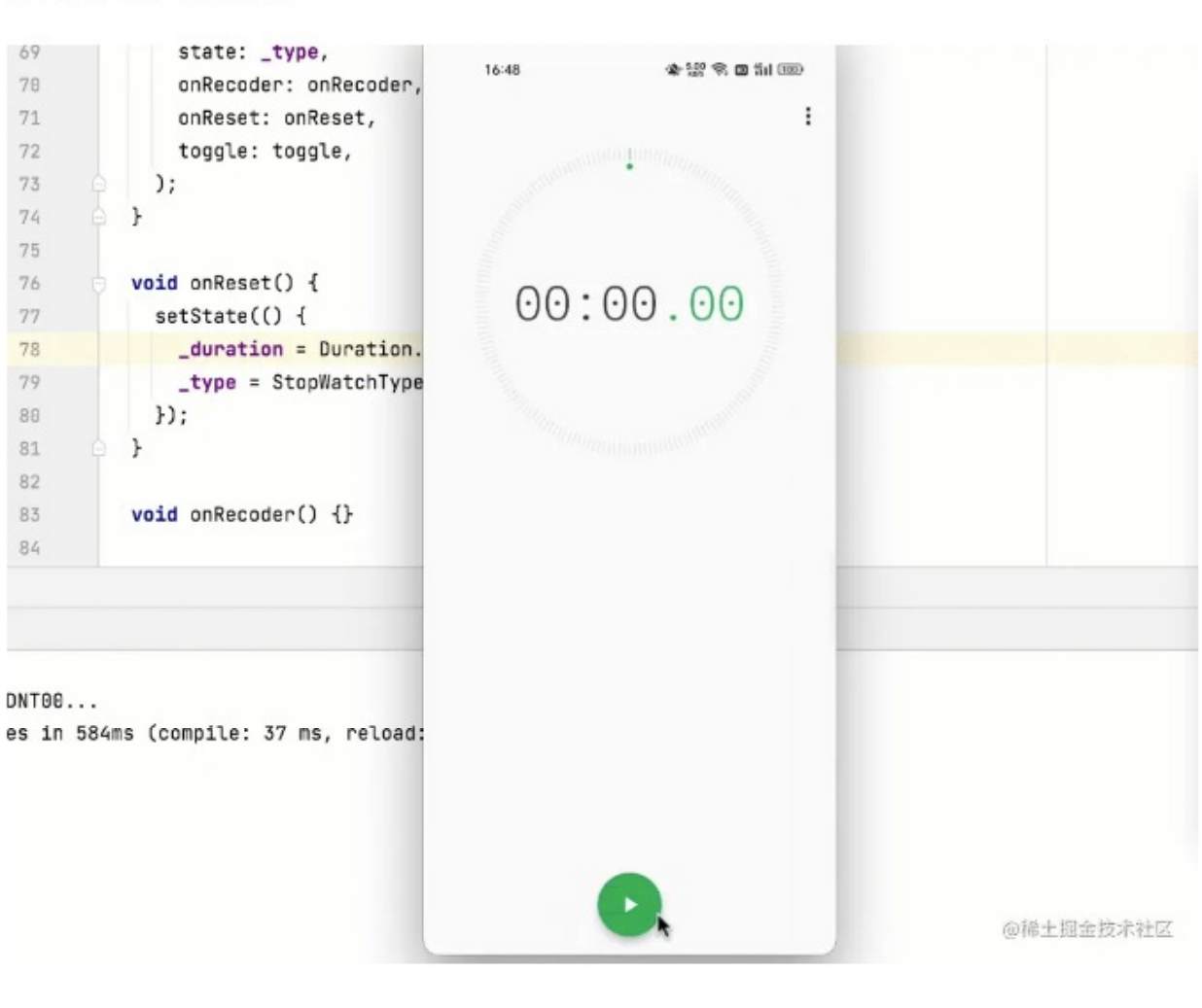
然后在按钮点击时，维护 `_type` 数据并通过 `setState` 触发组件构建，从而更新界面显示。这样就可以实现通过点击按钮，更改秒表的运行状态，从而修改界面呈现。


@稀土掘金技术社区

```dart
---->[_HomePageState]----
void onReset() {
  setState(() {
    _type = StopWatchType.none;
  });
}

void onRecoder() {}

void toggle() {
  bool running = _type == StopWatchType.running;
  setState(() {
    _type = running ? StopWatchType.stopped : StopWatchType.running;
  });
}
```

二、秒表的运动

接下来我们需要让秒表运动，效果如下：表盘上的时长会不断变化，这和之前也是类似，也是更新数据，触发界面更新。



```
      state: _type,
      onRecoder: onRecoder,
      onReset: onReset,
      toggle: toggle,
    );
  }

  void onReset() {
    setState(() {
      _duration = Duration.
      _type = StopWatchType
    });
  }

  void onRecoder() {}
```

```
DNT00...
es in 584ms (compile: 37 ms, reload:
```

@稀土掘金技术社区

---

1. 时长数据的维护与 Ticker 的使用

在 `_HomePageState` 中维护 `_duration` 对象，表示时长状态量。在构建秒表面板时，为 `StopwatchWidget` 传入 `_duration` 来决定显示的时长数据。这样当 `_duration` 发生变化时，重构界面时就能更新显示。

```dart
---->[timer/05/home_page/home_page.dart#_HomePageState]----
Duration _duration = Duration.zero;

Widget buildStopwatchPanel() {
  double radius = MediaQuery.of(context).size.shortestSide / 2 * 0.75;
  return StopwatchWidget(
    radius: radius,
    duration: _duration,
  );
}
```

接下来的重点就是如何开启秒表，让 `_duration` 时长数据不断变化。在 `Flutter` 框架中，提供了 `Ticker` 类来根据手机刷新率不断触发事件。如下所示，在 `initState` 中创建 `Ticker` 对象，构造入参中传入回调函数 `_onTick` ，该函数在每次事件时都会被调用。

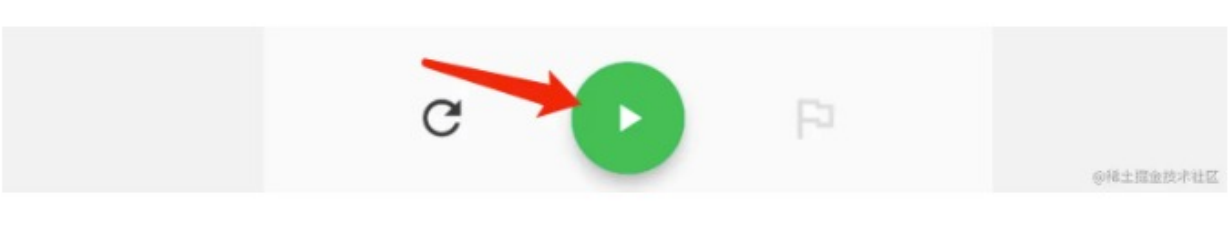
在回调中可以得到 `Ticker` 运行的时长，只要在 `_onTick` 中维护 `_duration` 变量，并通过 `setState` 触发重构，更新界面即可完成秒表运行的时长变化。

```dart
late Ticker _ticker;

@override
void initState() {
  super.initState();
  _ticker = Ticker(_onTick);
}

Duration dt = Duration.zero;
Duration lastDuration = Duration.zero;

void _onTick(Duration elapsed) {
  setState(() {
    dt = elapsed - lastDuration;
    _duration += dt;
    lastDuration = elapsed;
  });
}

@override
void dispose() {
  _ticker.dispose();
  super.dispose();
}
```

---

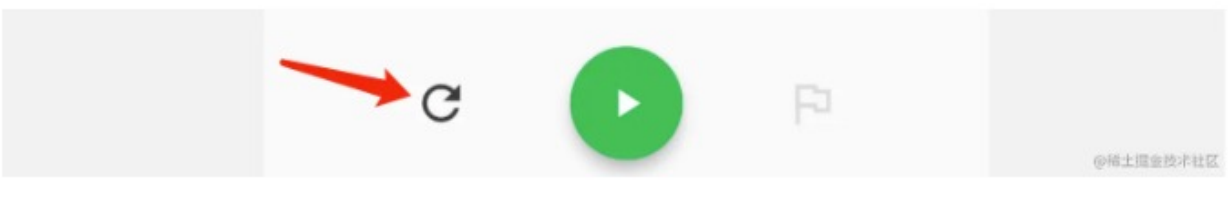
2. 按钮对秒表运行的控制

点击中间按钮可用让秒表 `暂停` 或 `启动` ，如下是 `toggle` 方法的实现逻辑：当秒表状态是运行中，执行 `_ticker.stop()` 暂停；否则执行 `_ticker.start()` 开启。


@稀土掘金技术社区

```dart
void toggle() {
  bool running = _type == StopWatchType.running;
  if(running){
    _ticker.stop();
    lastDuration = Duration.zero;
  }else{
    _ticker.start();
  }
  setState(() {
    _type = running ? StopWatchType.stopped : StopWatchType.running;
  });
}
```

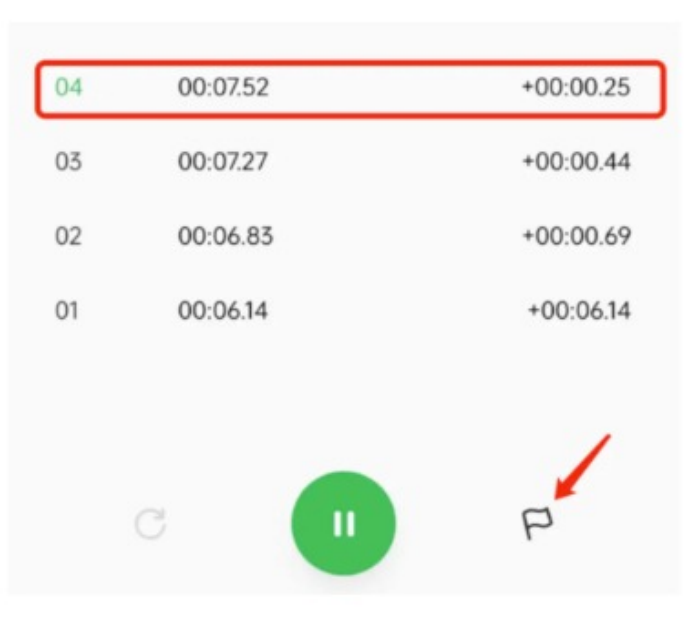
点击重置按钮时，触发 `onReset` 方法，将 `_duration` 数值置为 `Duration.zero` 。到这里就实现了秒表最基础的计时功能。


@稀土掘金技术社区

```dart
void onReset() {
  setState(() {
    _duration = Duration.zero;
    _type = StopWatchType.none;
  });
}
```

三、秒表的数据记录

接下来实现秒表记录功能，如下所示，在秒表运行中点击右侧的旗子按钮，记录当前的时刻信息。并且将记录在中间的面板显示。



```
04  00:07.52  +00:00.25
03  00:07.27  +00:00.44
02  00:06.83  +00:00.69
01  00:06.14  +00:06.14
```

@稀土掘金技术社区

---

1. 记录数据的定义

现在先来分析一些需要维护的数据，从上图可以看出，记录数据包括 `索引值` 、 `当前时刻` 、 `当前时刻和记录值的上一时刻差值` 。如下，定义 `TimeRecord` 类作为数据的载体：

```dart
---->[timer/06/home_page/model/time_record.dart]----
class TimeRecord {
  final Duration record; // 当前时刻
  final Duration addition; // 与上一时刻差值

  const TimeRecord({
    required this.record,
    required this.addition,
  });
}
```

接下来就是在交互时维护记录数据列表，如下所示，在 `_HomePageState` 中定义 `_record` 列表。点击旗子的按钮事件回调是 `onRecoder` ，在其中添加一条记录，并触发重构，更新界面。

```dart
---->[home_page.dart#_HomePageState]---
List<TimeRecord> _record = [];

void onRecoder() {
  Duration current = _duration;
  Duration addition = _duration;
  if(record.isNotEmpty){
    addition = _duration - record.last.record;
  }
  setState(() {
    _record.add(TimeRecord(record: current, addition: addition));
  });
}
```

在重置按钮点击时，清空记录数据。这样对于记录数据的维护就完成了，接下来看一下如何根据 `记录数据` 来构建列表面板组件。

```dart
void onReset() {
  setState(() {
    _duration = Duration.zero;
    _type = StopWatchType.none;
    _record.clear(); // 清空记录
  });
}
```

---

2. 构建记录列表面板

同样，为了让构建逻辑分离，这里单独封装 `RecordPanel` 组件来完成记录面板构建的逻辑。其中持有 `TimeRecord` 数据， 也就是说现在的要点是如何根据 `record` 列表来显示一条条的记录。

```dart
---->[home_page/record_panel.dart]----
class RecordPanel extends StatelessWidget {
  final List<TimeRecord> record;

  const RecordPanel({super.key,required this.record});

  @override
  Widget build(BuildContext context) {
    // TODO 根据数据 构建列表组件
  }
}
```

在界面中，构建记录面板的任务交由 `buildRecordPanel` 方法。如下在其中返回 `RecordPanel` ，并提供四条测试数据，来辅助构建界面：

```dart
---->[_HomePageState]----
Widget buildRecordPanel() {
  return Expanded(
    child: RecordPanel(
      record: [
        TimeRecord(record: Duration(seconds: 3), addition: Duration(seconds: 3)),
        TimeRecord(record: Duration(seconds: 4), addition: Duration(seconds: 1)),
        TimeRecord(record: Duration(seconds: 5), addition: Duration(seconds: 1)),
        TimeRecord(record: Duration(seconds: 6), addition: Duration(seconds: 1)),
      ],
    ),
  );
}
```

界面构建如下，每个条目由 `3` 个文字通过 `Row` 进行横向排列；其中三个文字的样式一致，可以通过 `DefaultTextStyle` 组件为子树中的文字提供默认样式，这样就无需每个 `Text` 组件都指定样式。

```
00  00:03.00  +00:03.00
01  00:04.00  +00:01.00
02  00:05.00  +00:01.00
03  00:06.00  +00:01.00
```

@稀土掘金技术社区

使用 `ListView` 组件的. `builder` 构造展示可滑动列表， `_buildItemByIndex` 方法用于根据索引构建条目组件。

```dart
@override
Widget build(BuildContext context) {
  return Padding(
    padding: const EdgeInsets.only( top: 20),
    child: DefaultTextStyle(
      style: const TextStyle(fontFamily: 'IBMPlexMono', color: Colors.black),
      child: ListView.builder(
        itemBuilder: _buildItemByIndex,
        itemCount: record.length,
      ),
    ),
  );
}
```

如下是 `_buildItemByIndex` 构造条目的逻辑，左右两个文字通过 `Padding` 添加边距； `Spacer` 组件用于 `Row` 组件中可以占据剩余空间，将其放在第二个和第三个文字之间，就可以 `"撑开"` 两个文字。

```dart
final EdgeInsets itemPadding = const EdgeInsets.symmetric(horizontal: 20, vertical: 4);
Widget _buildItemByIndex(BuildContext context, int index) {
  return Row(
    children: [
      Padding(
        padding: itemPadding,
        child: Text(index.toString().padLeft(2, '0')),
      ),
      Text(durationToString(record[index].record)),
      const Spacer(),
      Padding(
        padding: itemPadding,
        child: Text("+" + durationToString(record[index].addition)),
      ),
    ],
  );
}
```

`durationToString` 方法用于把 `Duration` 对象转化为显示的 `String` ，给个小练习：使用如下逻辑，定义一个 `Duration` 类的拓展方法 `beautifyString` 。

```dart
String durationToString(Duration duration) {
  int minus = duration.inMinutes % 60;
  int second = duration.inSeconds % 60;
  int milliseconds = duration.inMilliseconds % 1000;
  String commonStr = '${minus.toString().padLeft(2, "0")}:${second.toString().padLeft(2, "0'
  String highlightStr = ".${(milliseconds ~/ 10).toString().padLeft(2, "0")}";
  return commonStr + highlightStr;
}
```

---

3. 布局优化

刚才是正序排列的，但我们需要 `倒序排列` ，而且最后一条数据是的所有需要高亮，如下所示：

```
03  00:06.00  +00:01.00
02  00:05.00  +00:01.00
01  00:04.00  +00:01.00
00  00:03.00  +00:03.00
```

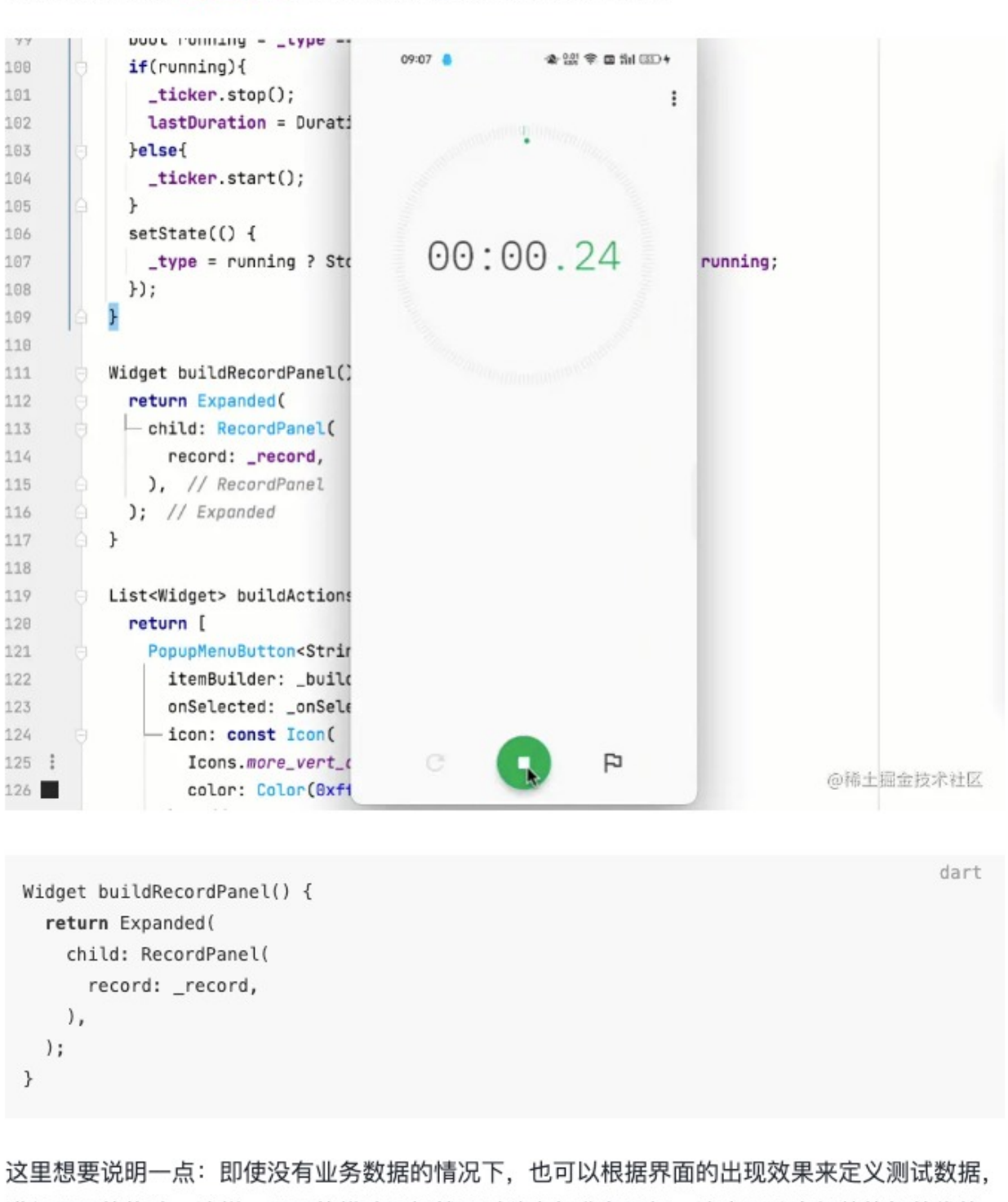
@稀土掘金技术社区

那怎么才能将列表倒序显示呢？其实很简单，界面的显示内容是由数据决定的，也就是说 `倒序显示` 本质上就是把数据倒序。方案一：将列表数据倒序。不过每次更新时的都要反转一下列表，并没有太大的必要。方案二：在记录数据时将最新数据记录到首位，众所周知，对于 `有序列表` ，频繁的首位插入是不明智的，如果通过链表结构维护数据，会把简单的问题搞复杂。

仔细想想，展示什么是由数据决定的，而在 `_buildItemByIndex` 方法中，数据是什么由 `index` 决定。所以只要在 `index` 上做文章即可，比如当 `index = 0` 时，显示最后一条数据， `index = 2` 时，显示倒数第二条数据。这样只需要将索引反转即可，代码如下：

```dart
Widget _buildItemByIndex(BuildContext context, int index) {
  int reverseIndex = (record.length - 1) - index; // 反转索引
  bool lightIndex = reverseIndex == record.length - 1;
  Color themeColor = Theme.of(context).primaryColor;
  Color? indexColor = lightIndex ? themeColor : null;
  // 略同
}
```

将测试数据换成 `_record` 即可完成记录时刻的功能，如下所示：



```
    bool running = _type ==
    if(running){
      _ticker.stop();
      lastDuration = Durati
    }else{
      _ticker.start();
    }
    setState(() {
      _type = running ? St                running;
    });
  }

  Widget buildRecordPanel(
    return Expanded(
      child: RecordPanel(
        record: _record,
      ), // RecordPanel
    ); // Expanded
  }

  List<Widget> buildAction
    return [
      PopupMenuButton<Strin
        itemBuilder: _buil
        onSelected: _onSele
        icon: const Icon(
          Icons.more_vert_
          color: Color(0xff
```

@稀土掘金技术社区

```dart
Widget buildRecordPanel() {
  return Expanded(
    child: RecordPanel(
      record: _record,
    ),
  );
}
```

这里想要说明一点：即使没有业务数据的情况下，也可以根据界面的出现效果来定义测试数据，进行界面的构建。这样，界面的搭建逻辑就可以独立与业务逻辑，这也是分离组件的额外优势：组件只和数据建立关联关系

> 我不管你数据怎么获取、怎么变化，传给我数据就能进行显示。

当有记录值时，需要显示和上一次记录的时间差值，如下红框所示。这个小功能就作为一个练习，大家自己尝试在 `timer/06` 的代码中完善一下。小提示： `StopwatchWidget` 中有个 `secondDuration` 的入参，用于显示小字时长。自己练习完成后，结果可以参考 `timer/07` 中的代码。

```
06  00:06.82  +00:00.19
05  00:06.62  +00:00.24
04  00:06.37  +00:00.60
03  00:05.76  +00:00.44
```

@稀土掘金技术社区

到这里，通过两章的介绍，秒表的核心功能就实现完毕，基本上可以作为秒表进行实际应用了。

@稀土掘金技术社区

不过别高兴的太早，虽然功能实现了，但是代码还有一些优化的空间，接下来的几章进行完善。下一章，将通过设置界面，对应用的周边功能进行拓展。了解一下界面跳转、主题切换和国际化实现的方式。

留言

输入评论（Enter换行，⌘ + Enter发送）

发表评论

全部评论（5）

苏灿烤鱼 LV.4 JY.5 | 安卓 | 3月前

如下所示，在 initState 中初创建 Ticker 对象,学生认为，修改为"如下所示，在 initState 中创建 Ticker 对象"更通顺。

点赞 1

> **张风捷特烈** （作者） | 3月前
>
> 恩，很多小笔误，感谢指正
>
> 点赞 回复

訾博ZiBo LV.3 JY.4 | 学生 @ 大学 | 4月前

"从上图可可以看出" => 去掉一个"可"

点赞 1

> **张风捷特烈** （作者） | 4月前
>
> 已更正
>
> 点赞 回复

未来大佬 LV.1 JY.3 | 4月前

点赞 回复