Flutter 滚动组件之 PageView

走在路上的日子

介绍

PageView是一页一页滚动的列表组件，类似于Android中的ViewPager

属性

```
// PageView构造函数
PageView({
    Key? key,
    this.scrollDirection = Axis.horizontal,
    this.reverse = false,
    PageController? controller,
    this.physics,
    this.pageSnapping = true,
    this.onPageChanged,
    List<Widget> children = const <Widget>[],
    this.dragStartBehavior = DragStartBehavior.start,
    this.allowImplicitScrolling = false,
    this.restorationId,
    this.clipBehavior = Clip.hardEdge,
    this.scrollBehavior,
    this.padEnds = true,
  })
```

1. scrollDirection = Axis.horizontal

滚动方向，水平或者垂直，默认垂直

2. reverse

   这个规定了 children(子节点) 的排序是否是倒序

3. controller

   滚动控制器，可以定位页面，获取当前页面等信息

4. physics

   滑动效果，不设置，会根据不同平台有不同的滚动效果

- NeverScrollableScrollPhysics 设置后，页面就不可滚动
- BouncingScrollPhysics 表示滚动到底了会有弹回的效果，就是 iOS 的默认交互
- ClampingScrollPhysics 表示滚动到底了就给一个效果，就是 Android 的默认交互
- FixedExtentScrollPhysics 就是 iOS 经典选择时间组件 UIDatePicker 那种交互

5. pageSnapping

   设置是否整页滚动

6. onPageChanged

   页面发生改变时的回调

7. children

   子页面组件列表

8. dragStartBehavior

   这个属性是设置认定开始拖动行为的方式，可以选择的

是down和start两个，默认是start. down是第一个手指按下认定拖动开始，start是手指拖动才算开始。

9. allowImplicitScrolling
   这个属性一般提供给视障人士使用的，默认是fasle

示例

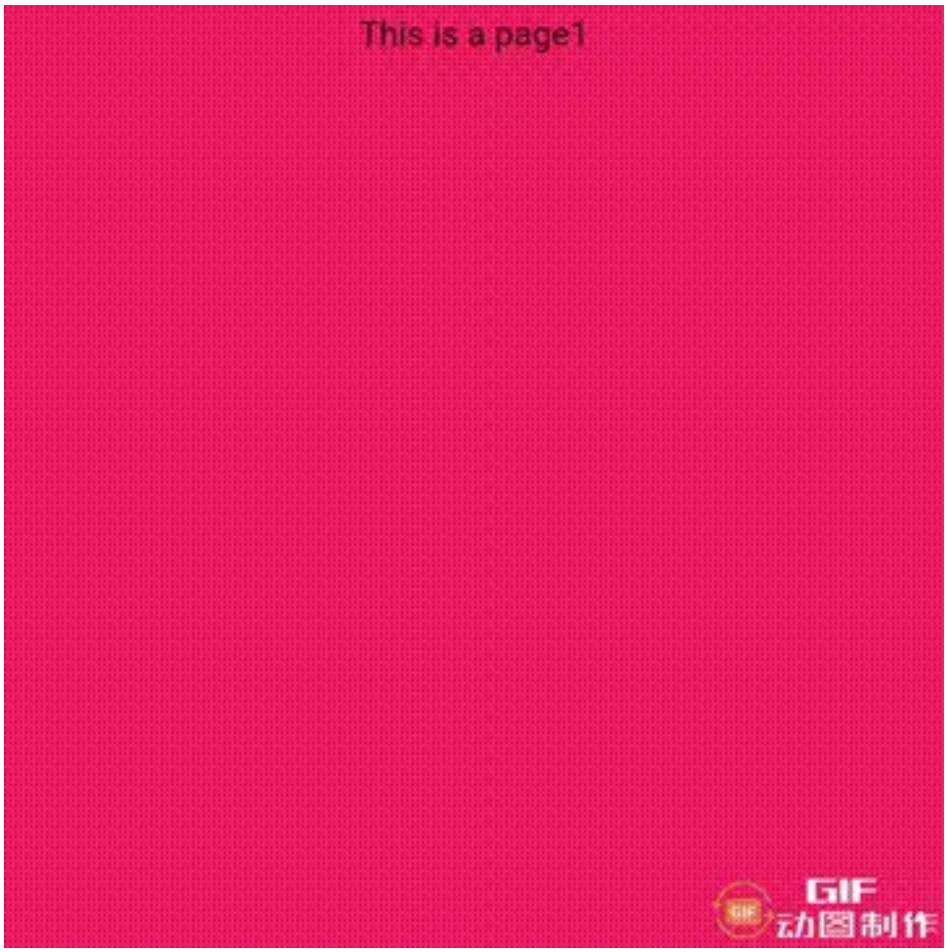
PageView一般使用方式如下：

1. PageView()
   这种方式没有实现Sliver模型，会把所有的子页面一次性全部初始化出来。

```
PageView(
    scrollDirection: Axis.horizontal,
    children: [
      Container(
        height: 300,
        color: Colors.pink,
        child: const Center(
          child: Text("This is a page1"),
        ),
      ),
      Container(
        color: Colors.teal,
        child: const Center(
          child: Text("This is a page2"),
        ),
      ),
      Container(
```

```
        color: Colors.amber,
        child: const Center(
          child: Text("This is a page3"),
        ),
      ),
    ],
  );
```

效果如下：



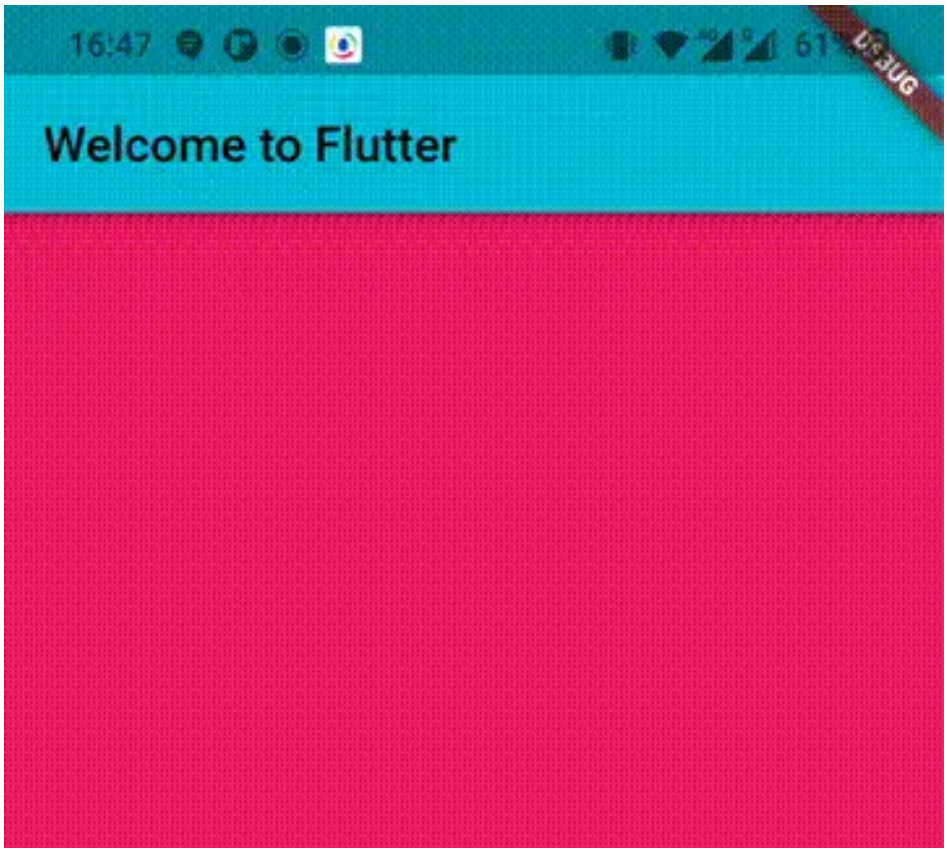
2 PageView.builder()　　Sliver 偷窥

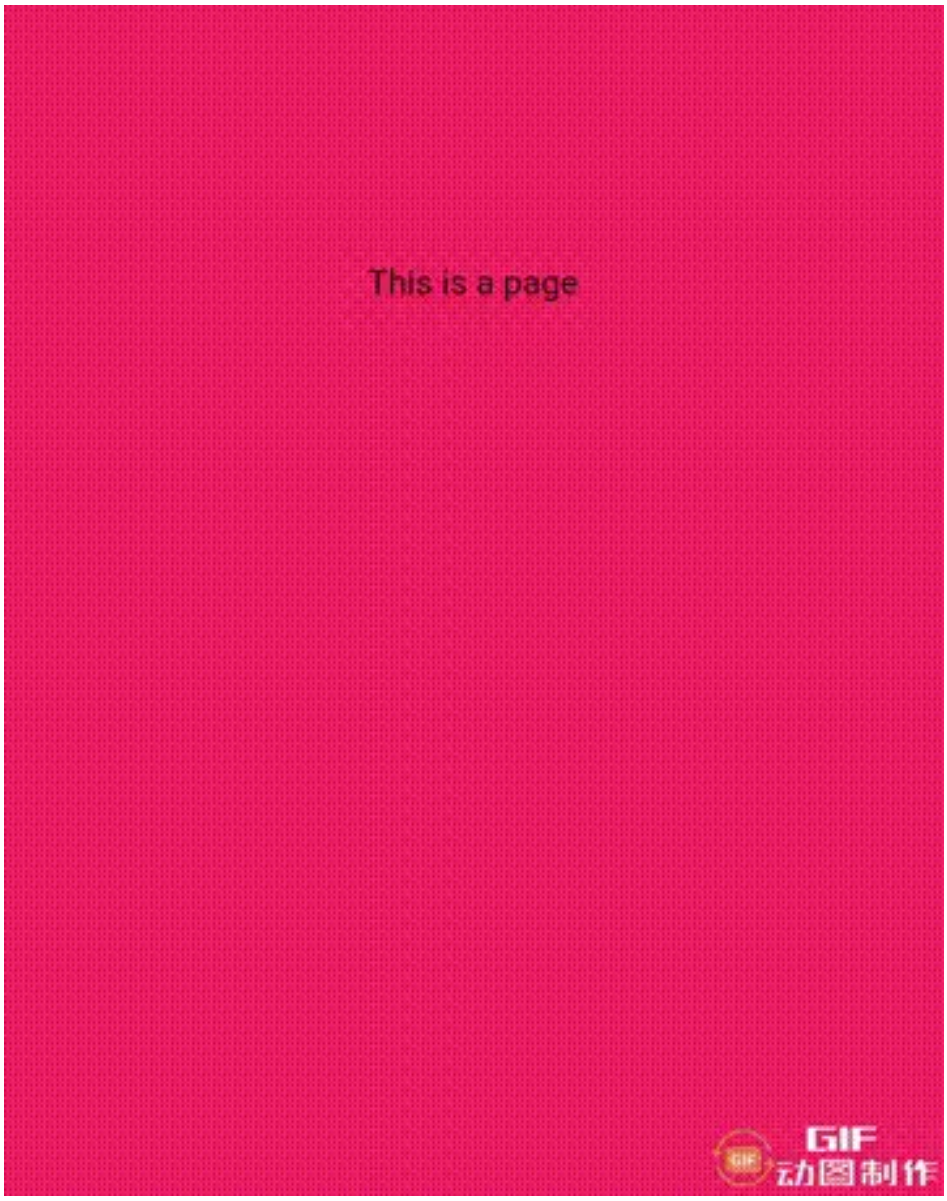
这个方式不会把所有的子组件一次性全部创建，是 Sliver 懒加载，推荐使用，会传一个 itemCount 和 itemBuilder

```
PageView.builder(
    scrollDirection: Axis.vertical,
    itemCount: 3,
    itemBuilder: (context, index) {
      Color? color;
      if (index == 0) {
        color = Colors.pink;
```

```
      } else if (index == 1) {
        color = Colors.teal;
      } else {
        color = Colors.amber;
      }
      return Container(
        color: color,
        child: const Center(
          child: Text("This is a page"),
        ),
      );
    },
  );
```

效果如下：





3. PageView.custom()

   这种方式是需要一个SliverChildDelegate，来自定义子组件行为，属于比较进阶的用法，这里先不讲了。