19、Flutter Widgets 之 粘合剂CustomScrollView, NestedScrollView滚动控件

风雨_83 LV.4
2022年10月16日 11:29 · 阅读 6572

概述：

Flutter中常用的滑动布局 `ScrollView` 有 `SingleChildScrollView`、 `NestedScrollView`、 `CustomScrollView`。

`SingleChildScrollView` 用来处理简单可滑动的页面布局视图，如一般的数据详情页面，当内容足够多时，一屏显示不下时，就需要滑动处理。

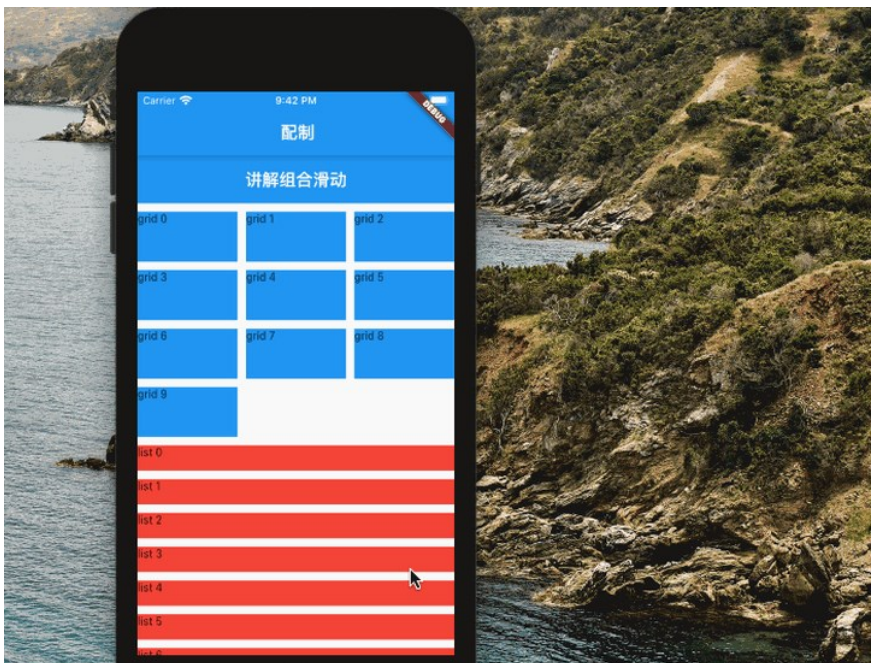
`NestedScrollView` 滑动组件是用来处理复杂情况下的滑动应用场景，如向上滑动视图时，要折叠隐藏一部分内容，这时候就需要使用到 NestedScrollView 与 SliverAppBar 的结合使用。

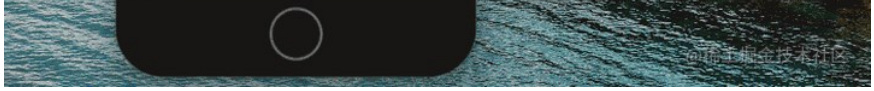
`CustomScrollView` 用来处理更为复杂的布局结合 SliverAppBar，SliverList和SliverGrid SliverPadding SliverToBoxAdapter SliverPersistentHeader, SliverFillRemaining，SliverFillViewport等来使用。

如一个详情页面中 即需要 GridView 来实现二维宫格效果，也需要 ListView 列表效果，如下图所示的图片效果，当使用 CustomScrollView 结合 SliverList 和SliverGrid 就可轻松实现，当然结合一下 SliverAppBar 也能实现折叠效果的头部布局，所以说 CustomScrollView 很强大。

在实际应用开发中，如果只是一个简单的适配页面滑动，建议码农使用 SingleChildScrollView 就可以。

如图：





CustomScrollView

CustomScrollView是使用Sliver组件创建自定义滚动效果的滚动组件，使用场景：

复制代码

```
ListView和GridView相互嵌套场景，ListView嵌套GridView时，需要给GridView指定高度，但我们希望高度随内容而变化(不指定)，List

一个页面顶部是AppBar，然后是GridView，最后是ListView，这3个区域以整体来滚动，AppBar具有吸顶效果。
```

CustomScrollView就像一个粘合剂，将多个组件粘合在一起，具统一的滚动效果。

Sliver系列组件有很多，比如SliverList、SliverGrid、SliverFixedExtentList、SliverPadding、SliverAppBar等。

相互嵌套场景

在实际业务场景中经常见到这样的布局，顶部是悬浮导航，接下来网格布局(GridView)，然后是列表布局(ListView)，滚动的时候做为一个整体，此场景是无法使用GridView+ListView来实现的，而是需要使用CustomScrollView+SliverAppBar+SliverGrid+SliverList来实现，

dart 复制代码

```
CustomScrollView buildCustomScrollView() {
    return CustomScrollView(
      ///反弹效果
      physics: BouncingScrollPhysics(),
      ///Sliver 家族的 Widget
      slivers: <Widget>[
        ///复杂的标题
        buildSliverAppBar(),
        ///间距
        SliverPadding(
          padding: EdgeInsets.all(5),
        ),

        ///九宫格
        buildSliverGrid(),
        ///间距
        SliverPadding(
          padding: EdgeInsets.all(5),
        ),
        ///列表
        buildSliverFixedExtentList()
      ],
    );
  }
```

SliverAppBar 常用来实现复杂的可折叠效果的头布局，代码如下：

dart 复制代码

```
SliverAppBar buildSliverAppBar() {
    return SliverAppBar(
      title: Text("讲解组合滑动"),
    );
  }
```

CustomScrollView 中使用的九宫格你不能再去使用 GridView了，在Sliver家族中，有SliverGridView，当然它与 GridView 的用法是一致的，代码如下：

dart 复制代码

```
SliverGrid buildSliverGrid() {
    return SliverGrid(
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        ///九宫格的列数
        crossAxisCount: 3,
        ///子Widget 宽与高的比值
        childAspectRatio: 2.0,
        ///主方向的 两个 子Widget 之间的间距
        mainAxisSpacing: 10,
        /// 次方向 子Widget 之间的间距
        crossAxisSpacing: 10,
      ),
      ///子Item构建器
      delegate: new SliverChildBuilderDelegate(
            (BuildContext context, num index) {
          ///每一个子Item的样式
          return Container(
```

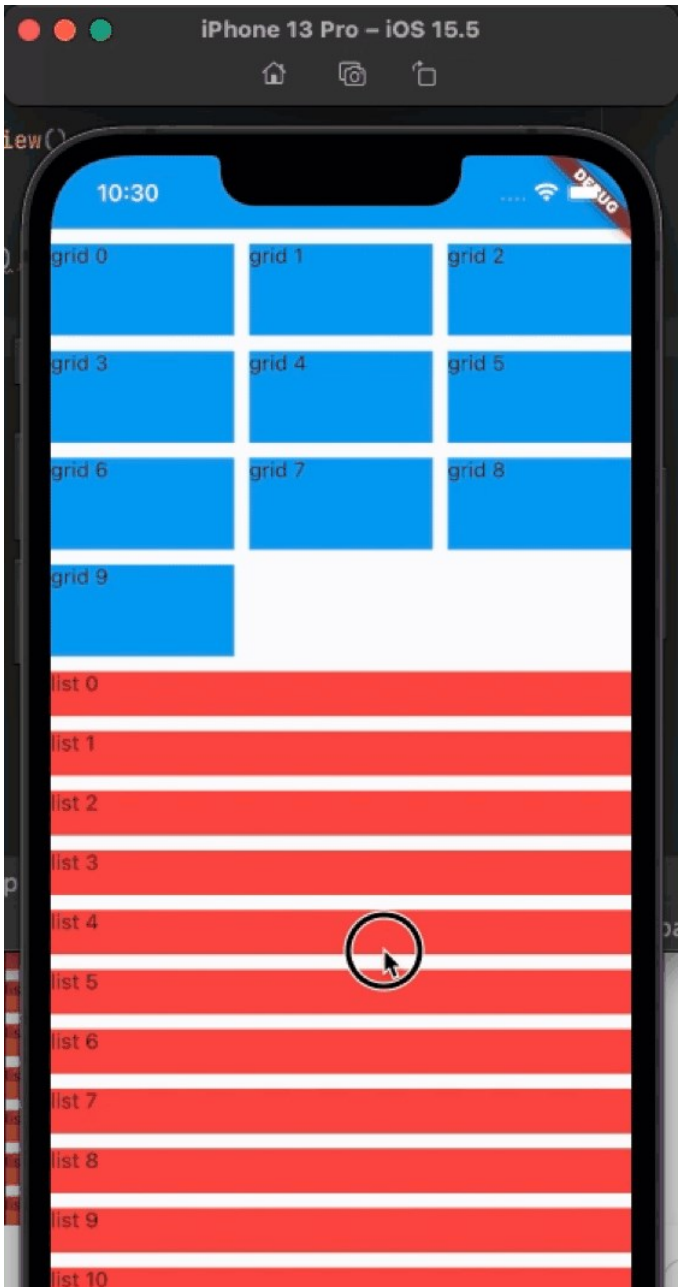
```
          color: Colors.blue,
          child: Text("grid $index"),
        );
      },
      ///子Item的个数
      childCount: 10,
    ),
  );
}
```

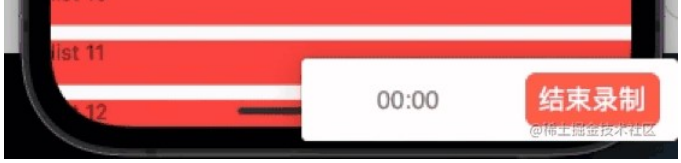
CustomScrollView 使用的列表你也不能使用 ListView了，需要使用SliverListView 或者是 SliverFixedExtentList，当然在这里使用了 SliverFixedExtentList 代码如下：

dart 复制代码

```
SliverFixedExtentList buildSliverFixedExtentList() {
  return SliverFixedExtentList(
    ///子条目的高度
    itemExtent: 40,
    ///子条目布局构建代理
    delegate: new SliverChildBuilderDelegate(
          (BuildContext context, num index) {
        ///子条目的布局样式
        return Container(
          color: Colors.red,
          child: Text("list $index"),
          margin: EdgeInsets.only(bottom: 10),
        );
      },
      ///子条目的个数
      childCount: 40,
    ),
  );
}
```

最终效果(上滑导航隐藏，下滑导航悬浮)：



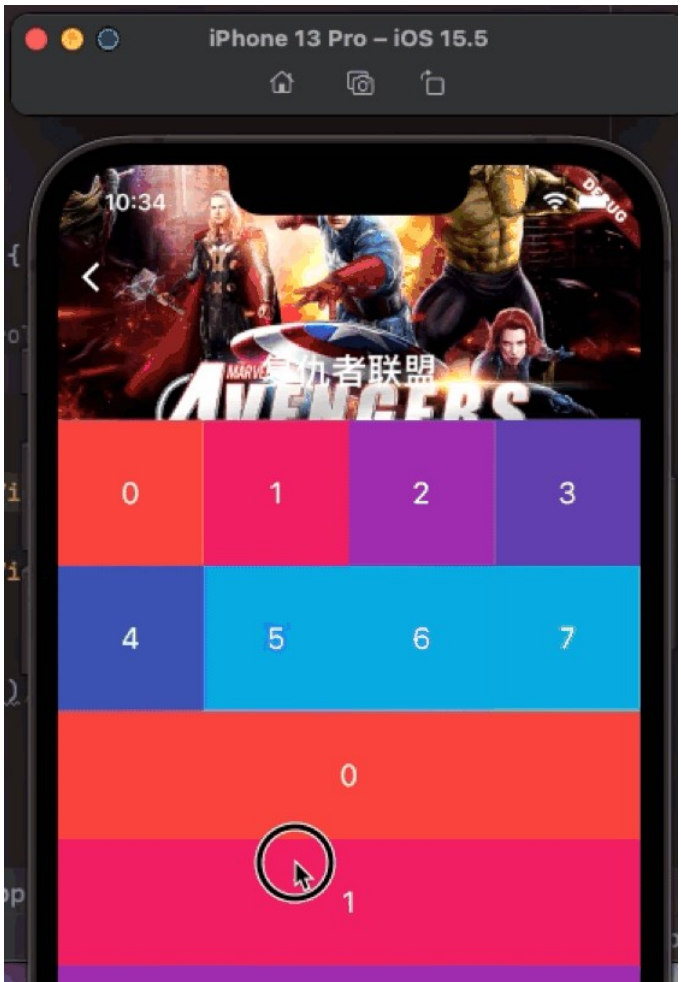


实际项目中页面顶部是AppBar，然后是GridView，最后是ListView，这3个区域以整体来滚动，AppBar具有吸顶效果，此效果也是我们经常遇到的，用法如下：

dart 复制代码

```
CustomScrollView buildCustomScrollView1() {
   return CustomScrollView(
     slivers: <Widget>[
       SliverAppBar(
         pinned: true,
         expandedHeight: 230.0,
         flexibleSpace: FlexibleSpaceBar(
           title: Text('复仇者联盟'),
           background: Image.network(
             'http://img.haote.com/upload/20180918/2018091815372344164.jpg',
             fit: BoxFit.fitHeight,
           ),
         ),
       ),
       SliverGrid.count(crossAxisCount: 4,children: List.generate(8, (index){
         return Container(
           color: Colors.primaries[index%Colors.primaries.length],
           alignment: Alignment.center,
           child: Text('$index',style: TextStyle(color: Colors.white,fontSize: 20),),
         );
       }).toList(),),
       SliverList(
         delegate: SliverChildBuilderDelegate((content, index) {
           return Container(
             height: 85,
             alignment: Alignment.center,
             color: Colors.primaries[index % Colors.primaries.length],
             child: Text('$index',style: TextStyle(color: Colors.white,fontSize: 20),),
           );
         }, childCount: 25),
       )
     ],
   );
 }
```

运行效果：



通过 `scrollDirection` 和 `reverse` 参数控制其滚动方向，用法如下：

```dart 复制代码
CustomScrollView(
  scrollDirection: Axis.horizontal,
  reverse: true,
  ...
)
```

`scrollDirection` 滚动方向，分为垂直和水平方向。

`reverse` 参数表示反转滚动方向，并不是垂直转为水平，而是垂直方向滚动时，默认向下滚动，`reverse` 设置 false，滚动方向改为向上，同理水平滚动改为水平向左。

设置 `scrollDirection:Axis.horizontal` 效果：

primary设置为true时，不能设置controller，因为primarytrue时，controller使用PrimaryScrollController，这种机制带来的好处是父组件可以控制子树中可滚动组件的滚动行为，例如，Scaffold正是使用这种机制在iOS中实现了点击导航栏回到顶部的功能。

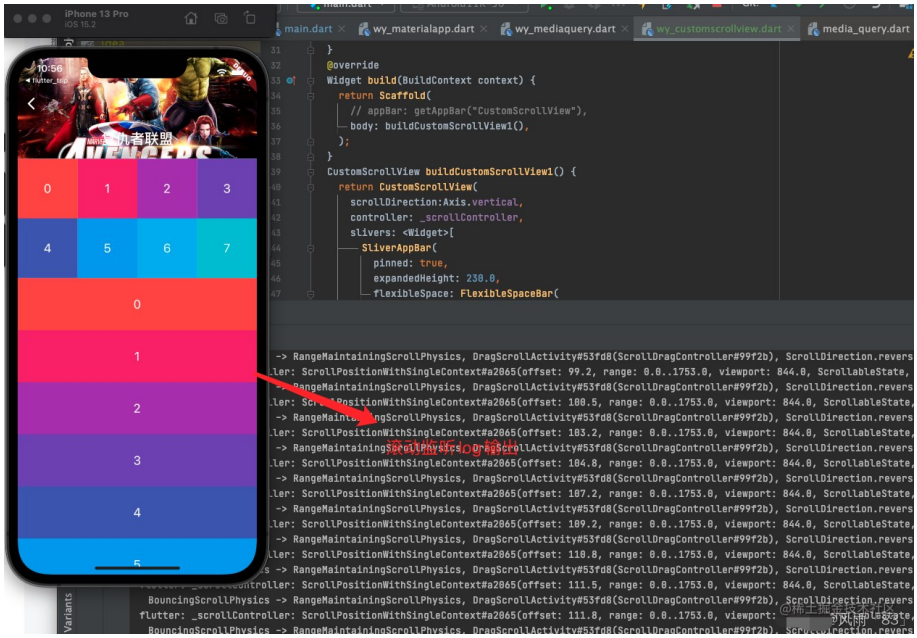
`controller` 为滚动控制器，可以监听滚到的位置，设置滚动的位置等，用法如下：

```dart 复制代码
_scrollController = ScrollController();

//监听滚动位置
    _scrollController.addListener((){
      print('${_scrollController.position}');
    });
    //滚动到指定位置
```

```
    _scrollController.animateTo(20.0);

CustomScrollView(
  controller: _scrollController,
  ...
)
```



physics表示可滚动组件的物理滚动特性，系统提供的ScrollPhysics有：

复制代码

```
AlwaysScrollableScrollPhysics:总是可以滑动

NeverScrollableScrollPhysics:禁止滚动

BouncingScrollPhysics :内容超过一屏 上拉有回弹效果

ClampingScrollPhysics :包裹内容 不会有回弹
```

NestedScrollView

可以在其内部嵌套其他滚动视图的组件，其滚动位置是固有链接的。

在普通的ScrollView中，如果有一个Sliver组件容纳了一个TabBarView，它沿相反的方向滚动（例如，允许用户在标签所代表的页面之间水平滑动，而列表则垂直滚动），则该TabBarView内部的任何列表都不会相互作用 与外部ScrollView。例如，浏览内部列表以滚动到顶部不会导致外部ScrollView中的SliverAppBar折叠以展开。

滚动隐藏AppBar

代码如下：

dart 复制代码

```dart
_buildNestedScrollView(){
    return NestedScrollView(
        headerSliverBuilder: (BuildContext context,bool innerBoxIsScrolled){
          return [
            SliverAppBar(title: Text('导航测试'),)
          ];
        },
        body: MediaQuery.removePadding(
          removeTop: true,
            context: context,
            child: ListView.builder(itemBuilder: (BuildContext context,int index){
          return Container(
            height: 120,
            color: Colors.primaries[index%Colors.primaries.length],
            alignment: Alignment.center,
            child: Text(
              '组合ListView $index',
              style: TextStyle(color: Colors.white,fontSize: 30),
            ),
          );
        }))
    );
  }
```

运行效果：

注意，如不不加 `MediaQuery.removePadding` 移除顶部间距，有个小bug。ListView和Appbar之间存在一个很大的间距，这个时候就需要MediaQuery去移除ListView的padding。

SliverAppBar展开折叠

```php
_buildNestedScrollView1(){
    return NestedScrollView(
        headerSliverBuilder: (BuildContext context,bool innerBoxIsScrolled){
          return [
            SliverAppBar(
              expandedHeight: 230,
              pinned: true,
              flexibleSpace: FlexibleSpaceBar(
                title: Text('复仇者联盟'),
                background: Image.network(
                  'http://img.haote.com/upload/20180918/2018091815372344164.jpg',
                  fit: BoxFit.fitHeight,
                ),
              ),
            )
          ];
        },
        body: ListView.builder(itemBuilder: (BuildContext context,int index){
          print('create $index');
          return Container(
            height: 100,
            color: Colors.primaries[index%Colors.primaries.length],
            alignment: Alignment.center,
            child: Text('$index 测试ListView',style: TextStyle(fontSize: 30),
            ),
          );
        }));
  }
```

运行效果（白色间距可以增加removepadding去除）：

与TabBar配合使用

用法如下：

dart 复制代码

```dart
_buildNestedScrollViewTabBar() {
    return NestedScrollView(
        headerSliverBuilder: (BuildContext context, bool innerBoxIsScrolled) {
          return [
            const SliverAppBar(
              expandedHeight: 230,
              pinned: true,
              flexibleSpace: Padding(
                padding: EdgeInsets.symmetric(vertical: 8),
                child: WyPageView(),
              ),

            ),
            SliverPersistentHeader(
              pinned: true,
              delegate: StickyTabBarDelegate(
                TabBar(
                  labelColor: Colors.black,
                  controller: this._tabController,
                  tabs: [
                    const Tab(text: '资讯',),
                    const Tab(text: '技术',),
                  ],
                ),
              ),
            )

          ];
        },
        body: TabBarView(
          controller: this._tabController,
            children: [
              RefreshIndicator(
                  child: _buildTabNewsList('----资讯类----'),
                  onRefresh: _handelRefresh,
              ),
              _buildTabNewsList('----技术类----'),
            ])
    );
  }
  //Refresh异步刷新方法
   Future<Null >_handelRefresh()async{
    print('加载数据');
    return null;
   }
   //构建newstlist列表
  _buildTabNewsList(String name) {
    return ListView.separated(itemBuilder: (context, int index) {
      return Column(
        children: [
          Text('$name $index 通过scrollDirection和reverse参数控制其滚动方向，用法如下：',
            style: TextStyle(fontSize: 18),),
          Text(
            '作者 csdn账号 ', style: TextStyle(fontSize: 12, color: Colors.grey),),
        ],
      );
    },
        separatorBuilder: (context, index) => Divider(),
        itemCount: 50);
  }
}
//StickyTabBarDelegate 代码如下：
class StickyTabBarDelegate extends SliverPersistentHeaderDelegate {
  final TabBar child;

  StickyTabBarDelegate(this.child);

  @override
  Widget build(BuildContext context, double shrinkOffset,
      bool overlapsContent) {
    // TODO: implement build
    return Container(
```

```
      color: Theme
          .of(context)
          .backgroundColor,
      child: this.child,
    );
  }

  @override
  // TODO: implement maxExtent
  double get maxExtent => this.child.preferredSize.height;

  @override
  // TODO: implement minExtent
  double get minExtent => this.child.preferredSize.height;

  @override
  bool shouldRebuild(covariant SliverPersistentHeaderDelegate oldDelegate) {
    // TODO: implement shouldRebuild
    throw true;
  }

}
```

运行效果：

总结：

本篇主要介绍了 `CustomScrollView` 和 `NestedScrollView`，`CustomScrollView` 的自定义灵活性更大，所有子组件都用Sliver家族组件，一般需求用NestedScrollView即可。