Flutter 扩展NestedScrollView（二）列表滚动同步解决

法的空间 LV.5
2018年11月14日 22:27 · 阅读 10404

Extended NestedscrollView 相关文章

- Flutter 扩展NestedScrollView（一）Pinned头引起的bug解决
- Flutter 扩展NestedScrollView（二）列表滚动同步解决
- Flutter 扩展NestedScrollView（三）下拉刷新的解决

接着上篇，没看上篇的小伙伴建议先看下上篇，免得断片中。。

FlutterCandies QQ群:181398081 pub v6.0.0

我继续讲下第2个问题的解决方案。

当在里面放上tabview，并且tab是缓存状态的时候，会出现滚动会互相影响的问题

上篇我们说到 在我们的主角NestedScrollView当中，有2个ScrollController.

dart 复制代码
```dart
class _NestedScrollController extends ScrollController {
  _NestedScrollController(
      this.coordinator, {
        double initialScrollOffset = 0.0,
        String debugLabel,
```

一个是inner，一个outer。outer是负责headerSliverBuilder里面的滚动widgets inner是负责body里面的滚动widgets 当outer滚动到底了之后，就会看看inner里面是否有能滚动的东东，开始滚动

Tabview是在body里面，这里我们肯定需要对inner进行处理。首先我们要明白，NestedScrollView是怎么处理outer和inner的关系的。

找到这个_NestedScrollCoordinator 的applyUserOffset方法中处理了整个NestedScrollView的滑动处理

dart 复制代码
```dart
@override
 void applyUserOffset(double delta) {
   updateUserScrollDirection(delta > 0.0 ? ScrollDirection.forward : ScrollDirection.reverse);
   assert(delta != 0.0);
   if (_innerPositions.isEmpty) {
     _outerPosition.applyFullDragUpdate(delta);
   } else if (delta < 0.0) {
     // dragging "up"
     // TODO(ianh): prioritize first getting rid of overscroll, and then the
     // outer view, so that the app bar will scroll out of the way asap.
     // Right now we ignore overscroll. This works fine on Android but looks
     // weird on iOS if you fling down then up. The problem is it's not at all
     // clear what this should do when you have multiple inner positions at
     // different levels of overscroll.
     final double innerDelta = _outerPosition.applyClampedDragUpdate(delta);
     if (innerDelta != 0.0) {
       for (_NestedScrollPosition position in _innerPositions)
         position.applyFullDragUpdate(innerDelta);
     }
   } else {
     // dragging "down" - delta is positive
     // prioritize the inner views, so that the inner content will move before the app bar grows
     double outerDelta = 0.0; // it will go positive if it changes
     final List<double> overscrolls = <double>[];
     final List<_NestedScrollPosition> innerPositions = _innerPositions.toList();
     for (_NestedScrollPosition position in innerPositions) {
       final double overscroll = position.applyClampedDragUpdate(delta);
       outerDelta = math.max(outerDelta, overscroll);
       overscrolls.add(overscroll);
     }
     if (outerDelta != 0.0)
       outerDelta -= _outerPosition.applyClampedDragUpdate(outerDelta);
     // now deal with any overscroll
     for (int i = 0; i < innerPositions.length; ++i) {
       final double remainingDelta = overscrolls[i] - outerDelta;
       if (remainingDelta > 0.0)
         innerPositions[i].applyFullDragUpdate(remainingDelta);
     }
   }
 }
```

```kotlin
Iterable<_NestedScrollPosition> get _innerPositions {
    return _innerController.nestedPositions;
  }
```

看到_innerPositions是我们要关注的东西，通过debug，我发现，如果tabview的每个tab做了缓存，那么每个tab里面列表的ScrollPosition将一直缓存在这个ScrollController里面。当tab到tabview的某个tab的时候，ScrollController将会将这ScrollPosition attach 上,如果没有缓存，将会在离开的时候detach掉。

```scss
  @override
  void attach(ScrollPosition position) {
    assert(position is _NestedScrollPosition);
    super.attach(position);
    coordinator.updateParent();
    coordinator.updateCanDrag();
    position.addListener(_scheduleUpdateShadow);
    _scheduleUpdateShadow();
  }

  @override
  void detach(ScrollPosition position) {
    assert(position is _NestedScrollPosition);
    position.removeListener(_scheduleUpdateShadow);
    super.detach(position);
    _scheduleUpdateShadow();
  }
```

真相只有一个。。是的。。可以说。。造成缓存tabview的各个tab里面的列表互相影响的原因，是因为官方说：as design(我就是这样设计的，不服吗).

按照我的思想啊，我滚动的时候。当然只想影响当前显示的这个列表啊。这不科学啊。。

找到原因找到原理，一切就都好解决了。现在的关键点在于，我怎么能知道显示对应的是哪个列表的？！

这个问题问了很多人。。也查找了好久都没找到好的方式去获取当前 激活的 列表对应的 ScrollPosition。。终于我只能想到一个 workaround。暂时解决这个问题。

提供一个容器，把inner里面的滚动列表包裹起来，并且设置它的tab 的唯一key

```dart
//pack your inner scrollables which are in  NestedScrollView body
//so that it can find the active scrollable
//compare with NestedScrollViewInnerScrollPositionKeyBuilder
class NestedScrollViewInnerScrollPositionKeyWidget extends StatefulWidget {
  final Key scrollPositionKey;
  final Widget child;
  NestedScrollViewInnerScrollPositionKeyWidget(
      this.scrollPositionKey, this.child);
  @override
  _NestedScrollViewInnerScrollPositionKeyWidgetState createState() =>
      _NestedScrollViewInnerScrollPositionKeyWidgetState();
}

class _NestedScrollViewInnerScrollPositionKeyWidgetState
    extends State<NestedScrollViewInnerScrollPositionKeyWidget> {
  @override
  Widget build(BuildContext context) {
    return widget.child;
```

```
  }

//  @override
//  void didChangeDependencies() {
//    // TODO: implement didChangeDependencies
//    //print("didChangeDependencies"+widget.scrollPositionKey.toString());
//    super.didChangeDependencies();
//  }
//
//  @override
//  void didUpdateWidget(NestedScrollViewInnerScrollPositionKeyWidget oldWidget) {
//    // TODO: implement didUpdateWidget
//    //print("didUpdateWidget"+widget.scrollPositionKey.toString()+oldWidget.scrollPositionKey.toString()
//    super.didUpdateWidget(oldWidget);
//  }
}
```

然后在刚才attach方法中通过先祖NestedScrollViewInnerScrollPositionKeyWidget

dart 复制代码

```
@override
  void attach(ScrollPosition position) {
    assert(position is _NestedScrollPosition);

    super.attach(position);
    attachScrollPositionKey(position as _NestedScrollPosition);
    coordinator.updateParent();
    coordinator.updateCanDrag();
    position.addListener(_scheduleUpdateShadow);
    _scheduleUpdateShadow();
  }

  @override
  void detach(ScrollPosition position) {
    assert(position is _NestedScrollPosition);
    position.removeListener(_scheduleUpdateShadow);
    super.detach(position);
    detachScrollPositionKey(position as _NestedScrollPosition);
    _scheduleUpdateShadow();
  }

  void attachScrollPositionKey(_NestedScrollPosition position) {
    if (position != null && scrollPositionKeyMap != null) {
      var key = position.setScrollPositionKey();
      if (key != null) {
        if (!scrollPositionKeyMap.containsKey(key)) {
          scrollPositionKeyMap[key] = position;
        } else if (scrollPositionKeyMap[key] != position) {
          //in demo ,when tab to tab03, the tab02 key will be tab00 at first
          //then it become tab02.
          //this is not a good solution

          position.clearScrollPositionKey();
          Future.delayed(Duration(milliseconds: 500), () {
            attachScrollPositionKey(position);
          });
        }
      }
    }
  }

  void detachScrollPositionKey(_NestedScrollPosition position) {
    if (position != null &&
        scrollPositionKeyMap != null &&
        position.key != null &&
        scrollPositionKeyMap.containsKey(position.key)) {
      scrollPositionKeyMap.remove(position.key);
      position.clearScrollPositionKey();
    }
  }
```

获取先祖NestedScrollViewInnerScrollPositionKeyWidget方法

ini 复制代码

```
Key setScrollPositionKey() {
    //if (haveDimensions) {
    final type = _typeOf<NestedScrollViewInnerScrollPositionKeyWidget>();

    NestedScrollViewInnerScrollPositionKeyWidget keyWidget =
    (this.context as ScrollableState)
        ?.context
        ?.ancestorWidgetOfExactType(type);
    _key = keyWidget?.scrollPositionKey;
    return _key;
  }
```

找到这个_NestedScrollCoordinator 的applyUserOffset方法中我们现在要替换掉 _innerPositions为 _currentInnerPositions

```dart
Iterable<_NestedScrollPosition> get _innerPositions {
    //return _currentPositions;
    return _innerController.nestedPositions;
  }

  Iterable<_NestedScrollPosition> get _currentInnerPositions {
    return _innerController
        .getCurrentNestedPositions(innerScrollPositionKeyBuilder);
  }
```

getCurrentNestedPositions里面的代码

```kotlin
  Iterable<_NestedScrollPosition> getCurrentNestedPositions(
      NestedScrollViewInnerScrollPositionKeyBuilder
      innerScrollPositionKeyBuilder) {
    if (innerScrollPositionKeyBuilder != null &&
        scrollPositionKeyMap.length > 1) {
      var key = innerScrollPositionKeyBuilder();
      if (scrollPositionKeyMap.containsKey(key)) {
        return <_NestedScrollPosition>[scrollPositionKeyMap[key]];
      } else {
        return nestedPositions;
      }
    }

    return nestedPositions;
  }
```

SampeCode

```dart
extended.NestedScrollView(
        headerSliverBuilder: (c, f) {
          return _buildSliverHeader(primaryTabBar);
        },
        //
        pinnedHeaderSliverHeightBuilder: () {
          return pinnedHeaderHeight;
        },
        innerScrollPositionKeyBuilder: () {
          var index = "Tab";
          if (primaryTC.index == 0) {
            index +=
                (primaryTC.index.toString() + secondaryTC.index.toString());
          } else {
            index += primaryTC.index.toString();
          }
          return Key(index);
        },
```

这里由你自己协定tab key。。我这里是一级tab+二级tab的index。。比如 Tab00代表一级tab第一个下面的二级tab的第一个。

定义tab里面的列表的时候如下，比如第一个tab下面的二级tab的第一个列表，那么它的key 为Tab00.

```dart
return extended.NestedScrollViewInnerScrollPositionKeyWidget(
        Key("Tab00"),
        // myRefresh.RefreshIndicator(
        // child:
        ListView.builder(
            itemBuilder: (c, i) {
              return Container(
                //decoration: BoxDecoration(border: Border.all(color: Colors.orange,width: 1.0)),
                alignment: Alignment.center,
                height: 60.0,
                child: Text(widget.tabKey.toString() + ": List$i"),
              );
            },
            itemCount: 100)
        //,
        //onRefresh: onRefresh,
        // )
        );
```

最后放上 **Github extended_nested_scroll_view**，如果你有更好的方式解决这个问题或者有什么不明白的地方，都请告诉我，由衷感谢。

pub v6.0.0