

Flutter 可滚动Widget —— ListView

ListView 是可以线性排列 子Widget 的可滚动Widget。ListView 可以和数据绑定用来实现瀑布流。

ListView 的快速上手

有四种使用 ListView 的方法：

1.使用默认的构造函数，给 children 属性赋值

代码所在位置

flutter_widget_demo/lib/listview/ListViewDefaultWidget.dart

使用方法

使用默认构造函数写 ListView，需要给 children 属性赋值，但只适用于那些只有少量 子Widget 的 ListView,ListView 创建的时候，其子Widget 也会一起创建。

Demo 如下：

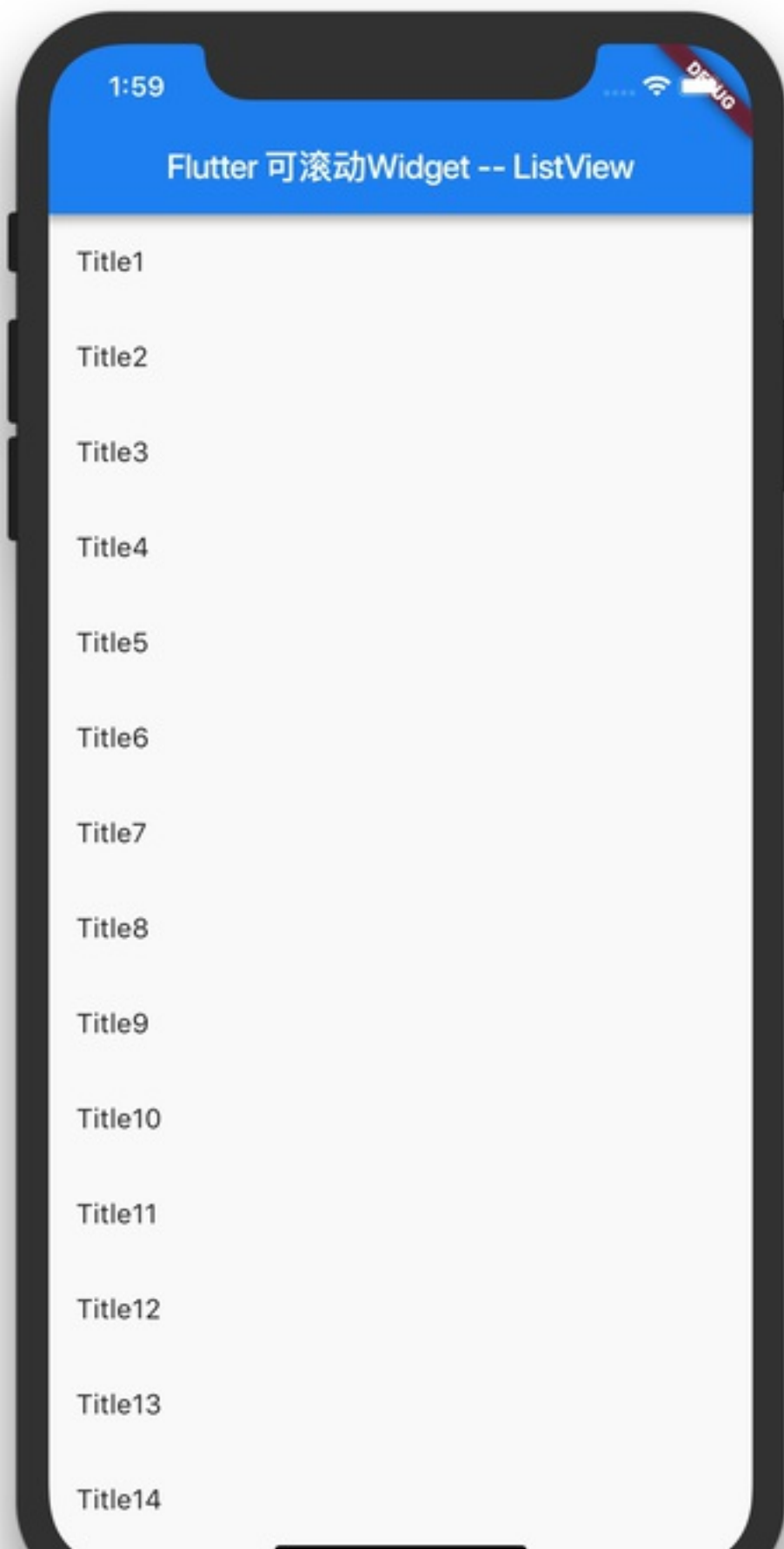
```
import 'package:flutter/material.dart';

main() => runApp(new ListViewDefaultWidget());

class ListViewDefaultWidget extends
StatelessWidget {
```

```
@override
Widget build(BuildContext context) {
  return new MaterialApp(
    title: 'Test',
    home: new Scaffold(
      appBar:
        new AppBar(title: new
Text('Flutter 可滚动Widget -- ListView')),
      body: ListView(
        children: <Widget>[
          ListTile(title: Text('Title1')),
          ListTile(title: Text('Title2')),
          ListTile(title: Text('Title3')),
          ListTile(title: Text('Title4')),
          ListTile(title: Text('Title5')),
          ListTile(title: Text('Title6')),
          ListTile(title: Text('Title7')),
          ListTile(title: Text('Title8')),
          ListTile(title: Text('Title9')),
          ListTile(title: Text('Title10')),
          ListTile(title: Text('Title11')),
          ListTile(title: Text('Title12')),
          ListTile(title: Text('Title13')),
          ListTile(title: Text('Title14')),
          ListTile(title: Text('Title15')),
          ListTile(title: Text('Title16')),
          ListTile(title: Text('Title17')),
          ListTile(title: Text('Title18')),
          ListTile(title: Text('Title19')),
        ],
      )),
  );
}
```

运行效果为：



2.使用 ListView.builder,可用于和数据绑定实现大量或无限的列表

代码所在位置

flutter_widget_demo/lib/listview/ListViewBuilderWidget.dart

使用方法

ListView.builder 可以用于构建大量或无限的列表，是因为 ListView.builder 只会构建那些实际可见的 子Widget。

ListView.builder 的定义为：

```
ListView.builder({
  Key key,
  Axis scrollDirection = Axis.vertical,
  bool reverse = false,
  ScrollController controller,
  bool primary,
  ScrollPhysics physics,
  bool shrinkWrap = false,
  EdgeInsetsGeometry padding,
  this.itemExtent,
  @required IndexedWidgetBuilder itemBuilder,
  int itemCount,
  bool addAutomaticKeepAlives = true,
  bool addRepaintBoundaries = true,
  bool addSemanticIndexes = true,
  double cacheExtent,
  int semanticChildCount,
  DragStartBehavior dragStartBehavior =
DragStartBehavior.down,
})
...
```

大部分属性都和 ListView 的默认构造函数一样，除了这两个：

- int itemCount

代表 子Widget 的数量，虽然是可选的，但是还是建议赋值，可以让 ListView 预估最大滑动距离，从而提升性能。如果为 null，则子节点数由[itemBuilder]返回null的最小索引确定。

- @required IndexedWidgetBuilder itemBuilder

itemBuilder 用于创建实际可见的 子Widget，只有索引大于或等于零且小于 itemCount 才会调用 itemBuilder。

下面写一个数据和 ListView.builder 绑定使用的例子:

```
import 'package:flutter/material.dart';

void main() => runApp(ListViewBuilderWidget(
  items: List<String>.generate(10000, (i) =>
    "Item $i"),
  ));

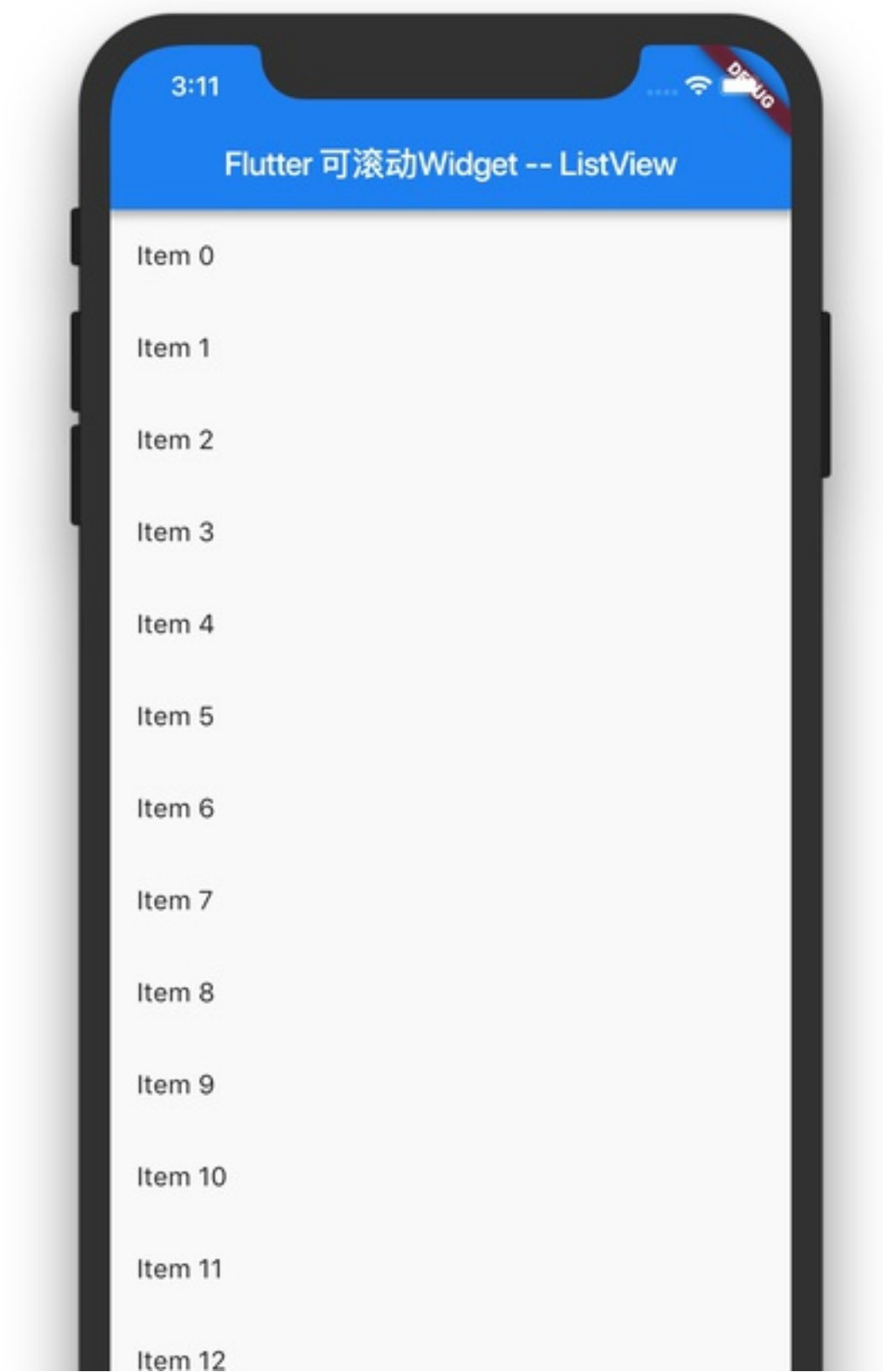
class ListViewBuilderWidget extends
StatelessWidget {
  final List<String> items;


  ListViewBuilderWidget({Key key, @required
this.items}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Test',
      home: Scaffold(
        appBar: AppBar(title: new Text('Flutter 可
滚动Widget -- ListView')),
        body: ListView.builder(
          itemCount: items.length,
          itemBuilder: (context, index) {
            return ListTile(
              title: Text('${items[index]}'),
            );
          },
        ),
      ),
    );
  }
}
```

```
}  
}
```

运行后的效果为：





Item 13

要实现一个无限循环列表，只要不给 itemCount 赋值就行，如下：

```
ListView.builder(  
  padding: EdgeInsets.all(8.0),  
  itemBuilder: (BuildContext context, int  
index) {  
    return ListTile(title: Text('Title  
$index'),);  
  },  
)
```

3.使用 ListView.separated，具有分割项的 ListView.builder

代码所在位置

flutter_widget_demo/lib/listview/ListViewSeparatedWidget.dart

使用方法

看 ListView.separated 的定义：


```

ListView.separated({
  Key key,
  Axis scrollDirection = Axis.vertical,
  bool reverse = false,
  ScrollController controller,
  bool primary,
  ScrollPhysics physics,
  bool shrinkWrap = false,
  EdgeInsetsGeometry padding,
  @required IndexedWidgetBuilder itemBuilder,
  @required IndexedWidgetBuilder
separatorBuilder,
  @required int itemCount,
  bool addAutomaticKeepAlives = true,
  bool addRepaintBoundaries = true,
  bool addSemanticIndexes = true,
  double cacheExtent,
})
...

```

相比 `ListView.builder` 多了一个 `separatorBuilder`, `separatorBuilder` 就是用于构建分割项的, 而且 `itemBuilder`、`separatorBuilder`、`itemCount` 都是必选的。

使用的 demo 如下:

```

import 'package:flutter/material.dart';

void main() => runApp(ListViewSeparatedWidget(
  items: List<String>.generate(10000, (i) =>
    "Item $i"),
));

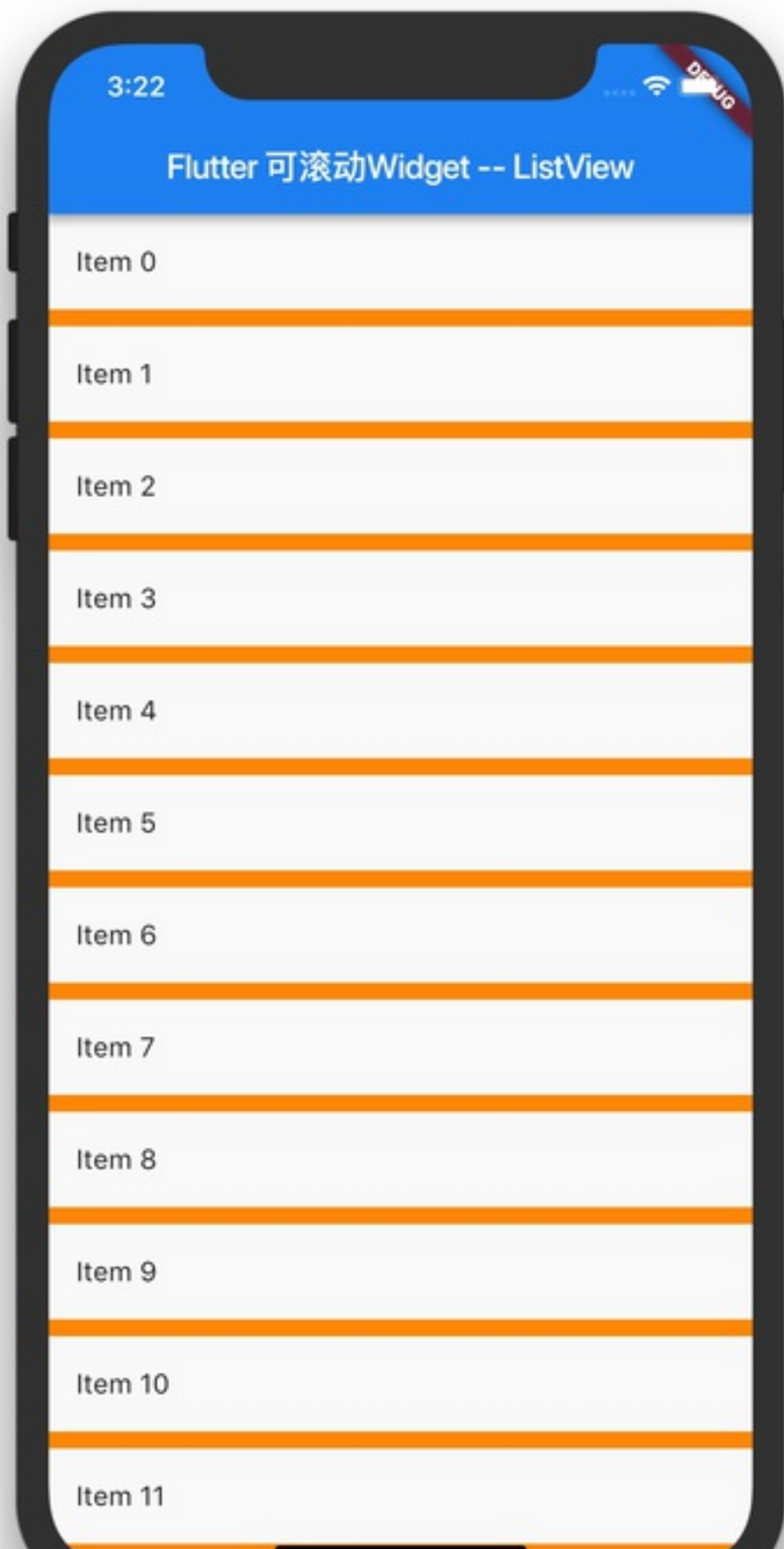
```

```
class ListViewSeparatedWidget extends
StatelessWidget {
  final List<String> items;

  ListViewSeparatedWidget({Key key, @required
this.items}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Test',
      home: Scaffold(
        appBar: AppBar(title: new Text('Flutter 可
滚动Widget -- ListView')),
        body: ListView.separated(
          itemCount: items.length,
          itemBuilder: (context, index) {
            return ListTile(
              title: Text('${items[index]}'),
            );
          },
          separatorBuilder: (context, index) {
            return Container(
              constraints:
BoxConstraints.tightFor(height: 10),
              color: Colors.orange,
            );
          },
        ),
      ),
    );
  }
}
```

运行效果为：



4.使用 ListView.custom, 需要使用 SliverChildDelegate

代码所在位置

flutter_widget_demo/lib/listview/ListViewCustomWidget.dart

使用方法

SliverChildDelegate 提供了定制 子Widget 的能力。

首先看 ListView.custom 的定义：

```
const ListView.custom({  
  Key key,  
  Axis scrollDirection = Axis.vertical,  
  bool reverse = false,  
  ScrollController controller,  
  bool primary,  
  ScrollPhysics physics,  
  bool shrinkWrap = false,  
  EdgeInsetsGeometry padding,  
  this.itemExtent,  
  @required this.childrenDelegate,  
  double cacheExtent,  
  int semanticChildCount,  
})
```

childrenDelegate 为必选参数，在看如何实现 SliverChildDelegate，发现 SliverChildDelegate 是一个抽象类，SliverChildDelegate 的 build 方法可以对单个子Widget进行自定义处理，而且 SliverChildDelegate 有个默认实现 SliverChildListDelegate，所以我们用 SliverChildListDelegate 来实现 ListView.custom，代码如下：

```
import 'package:flutter/material.dart';

void main() => runApp(ListViewCustomWidget(
  items: List<String>.generate(10000, (i) =>
    "Item $i"),
  ));

class ListViewCustomWidget extends
StatelessWidget {
  final List<String> items;

  ListViewCustomWidget({Key key, @required
this.items}) : super(key: key);

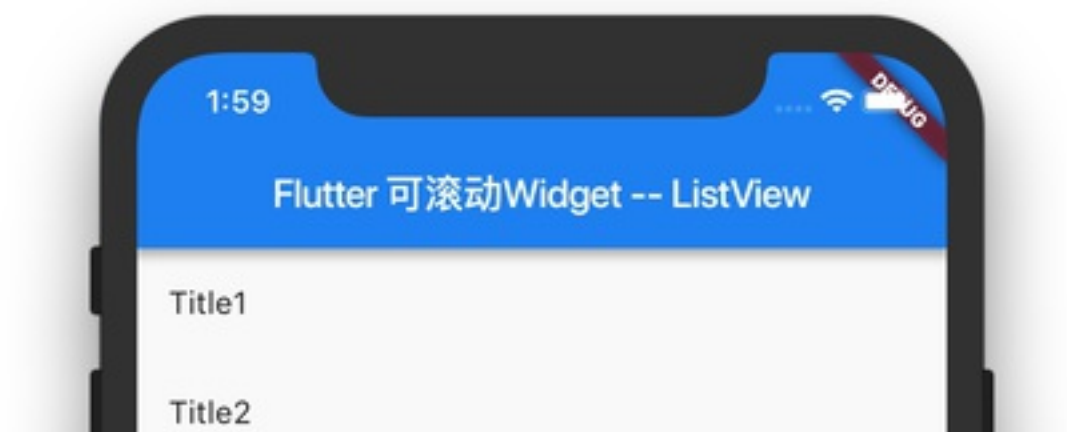
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Test',
      home: Scaffold(
        appBar: AppBar(title: new Text('Flutter 可
滚动Widget -- ListView')),
        body: ListView.custom(
          childrenDelegate:
SliverChildListDelegate(<Widget>[
          ListTile(title: Text('Title1')),
          ListTile(title: Text('Title2')),
```

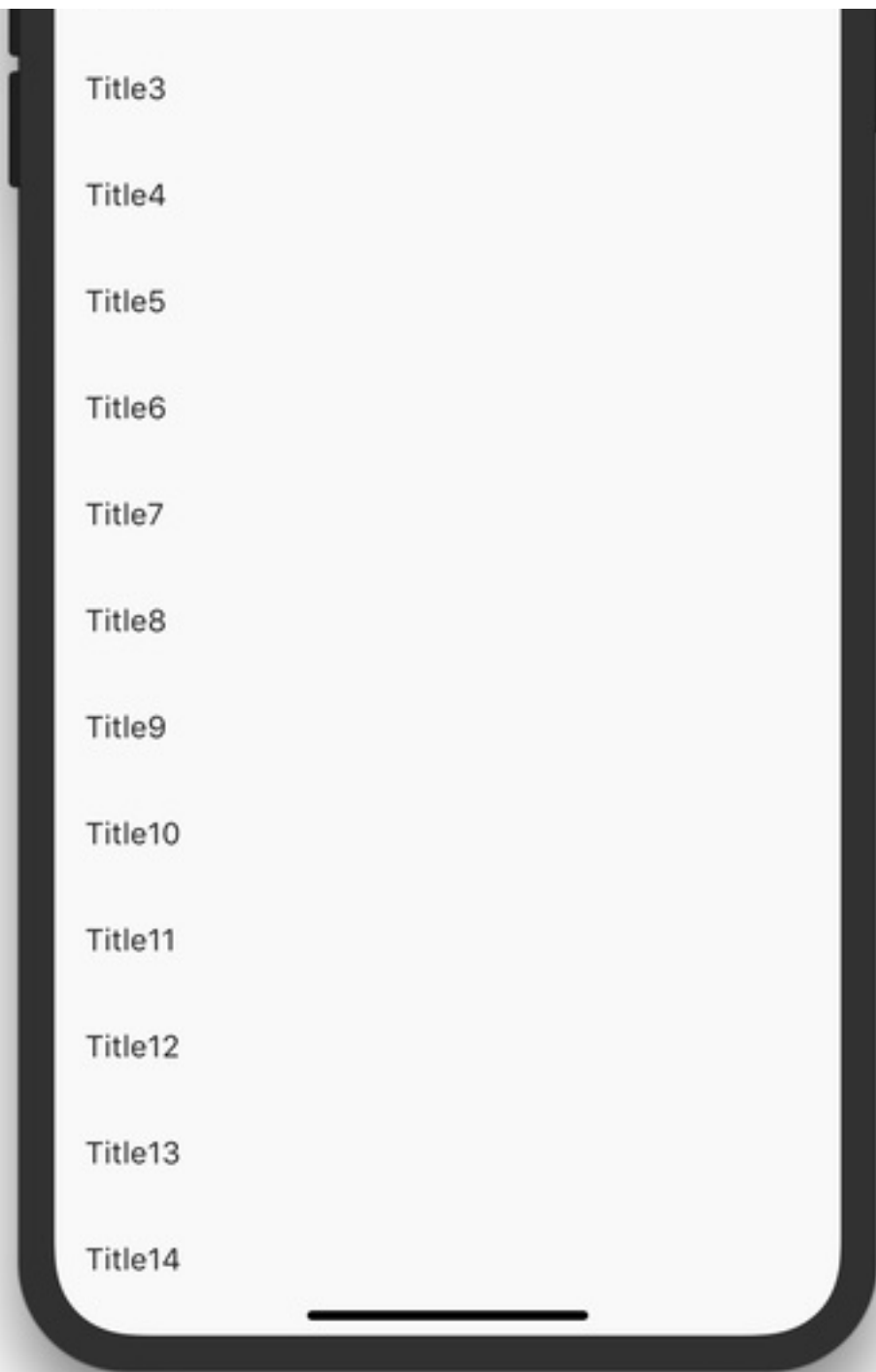
```

ListTile(title: Text('Title3')),
ListTile(title: Text('Title4')),
ListTile(title: Text('Title5')),
ListTile(title: Text('Title6')),
ListTile(title: Text('Title7')),
ListTile(title: Text('Title8')),
ListTile(title: Text('Title9')),
ListTile(title: Text('Title10')),
ListTile(title: Text('Title11')),
ListTile(title: Text('Title12')),
ListTile(title: Text('Title13')),
ListTile(title: Text('Title14')),
ListTile(title: Text('Title15')),
ListTile(title: Text('Title16')),
ListTile(title: Text('Title17')),
ListTile(title: Text('Title18')),
ListTile(title: Text('Title19')),
]),
),
),
);
}
}

```

运行效果为：





ListView 的构造函数及参数使用

首先看 ListView 的构造函数:

```

class ListView extends BoxScrollView {
  ListView({
    Key key,
    Axis scrollDirection = Axis.vertical,
    bool reverse = false,
    ScrollController controller,
    bool primary,
    ScrollPhysics physics,
    bool shrinkWrap = false,
    EdgeInsetsGeometry padding,
    this.itemExtent,
    bool addAutomaticKeepAlives = true,
    bool addRepaintBoundaries = true,
    bool addSemanticIndexes = true,
    double cacheExtent,
    List<Widget> children = const <Widget>[],
    int semanticChildCount,
    DragStartBehavior dragStartBehavior =
DragStartBehavior.down,
  })
  ...
}

```

参数名字	参数类型	意
key	Key	Widget 的标识
scrollDirection	Axis	滑动的方向 默认为 Axis.verti 可滑动

reverse	bool	<p>控制 ListView 里的顺序，是按照插入顺序插入顺序相反的方式默认为 false，就排序，第一个插入，当 reverse 为 true 插入的会在底部</p> <p>可以控制 ListView 的 ScrollController 功能：</p> <ol style="list-style-type: none"> 1.设置 ListView 的 ScrollController 2.可以控制 ListView 恢复滑动的位置 3.可以读取、设置 ScrollController 可以继承 ScrollController 定义的功能 <p>当 primary 为 true 时， ScrollController 必须为 true</p>
controller	ScrollController	<p>是否是与父级关联</p> <p>当为 true 时，即使没有足够的内容也可以设置 ListView 的 ScrollController 值必须为 ScrollPhysics</p> <p>比如有如下的值：</p>
primary	bool	<p>AlwaysScrollable 可以让 ListView 的内容也能滑动</p>
physics	ScrollPhysics	<p>ScrollPhysics():List 足够的内容的时候是否根据列表项的 ListView 的长度， false。</p>

shrinkWrap	bool	<p>当 shrinkWrap 为 true 时，ListView 会在滚动时占用最大的空间</p> <p>当 shrinkWrap 为 false 时，ListView 在滚动时占用的空间就是其列表项的总高度，这会很耗性能，因为每次滚动时，ListView 都要重新计算列表项的总高度</p>
padding	EdgeInsetsGeometry	ListView 的内边距
itemExtent	double	<p>itemExtent 指定的列表项的高度，如果滚动方向是垂直的，itemExtent 代表列表项的高度，如果滚动方向为水平的，itemExtent 代表列表项的长度</p> <p>如果 itemExtent 不为 null，则强制所有子Widget 大小都为 itemExtent 指定的大小，指定 itemExtent 为子Widget 的高度，则 ListView 会根据列表的长度来计算，ListView 会根据列表的长度来计算</p> <p>是否用 AutomaticKeepAlive 包列表项,默认为 true，在一个 lazy list 中，子Widget 为了保证在界面时不被回收，需要调用 addAutomaticKeepAlives 方法，true</p>
addAutomaticKeepAlives	bool	

		<p>当子Widget不需活时，为了提升性能，addAutomaticKeepAlive默认为false。</p> <p>如果子Widget自身处于KeepAlive状态，addAutomaticKeepAlive置为false。</p>
addRepaintBoundaries	bool	<p>是否用 RepaintBoundary 包裹列表项，默认为 true。</p> <p>当 addRepaintBoundaries 为 true 时，可以避免不必要的重绘，从而提高性能。</p> <p>但是当列表项重绘（如一个颜色块，一段文本）时，不添加 RepaintBoundary 包裹没有效果。</p>
addSemanticIndexes	bool	<p>是否用 IndexedSemantics 包裹列表项，默认为 true。</p> <p>使用 IndexedSemantics 包裹列表项，可以提供正确的用于辅助模式浏览的语义信息。</p>
cacheExtent	double	<p>ListView 可见部分的区域可以用来缓存这部分区域的 item。当这部分区域也会加载出来，所缓存的 item 可见的时候，缓存的 item 可见，cacheExtent 就表示可见部分的前面和后面多少 item 可见。</p>
children	List<Widget>	<p>ListView 的列表项。</p> <p>提供语义信息的列表项。</p>

semanticChildCount	int	默认为 ListView
		确定处理拖动开始
		如果设置为
		[DragStartBehav
dragStartBehavior	DragStartBehavior	检测到拖动手势时
		行为
		如果设置为
		[DragStartBehav
		将在首次检测到向