

Flutter 可滚动 Widget —— GridView

GridView 是一个可以构建二维网格列表的可滚动Widget。

GridView 的快速上手

GridView 和 ListView 一样，有五种用法：

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/gridview/GridViewDefaultWidget.dart

使用方法

使用默认构造函数写 GridView，只适用于那些只有少量子Widget的 GridView。

demo 如下：

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

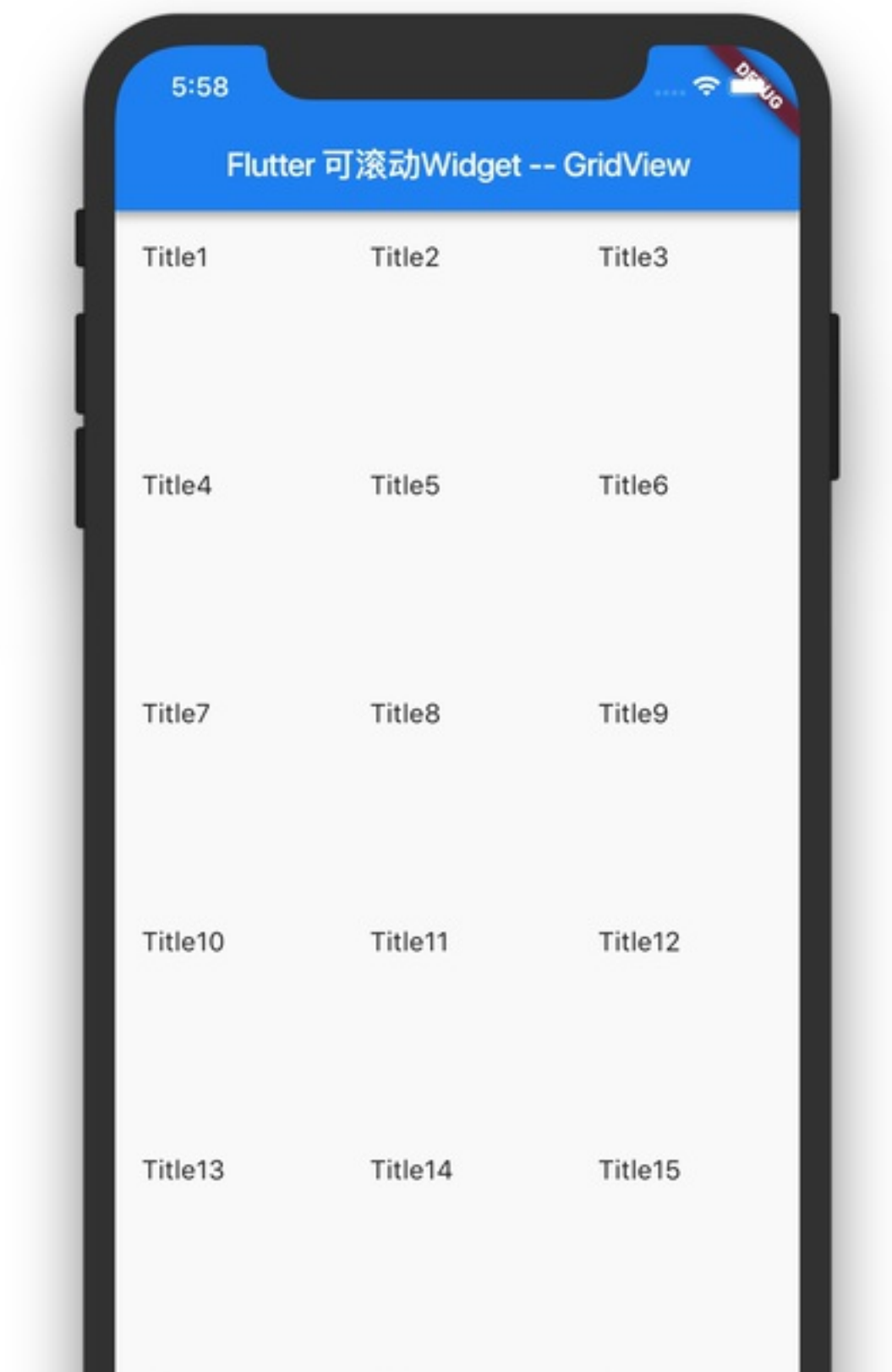
void main() => runApp(GridViewDefaultWidget());

class GridViewDefaultWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
```

```
return MaterialApp(  
  title: 'Test',  
  home: Scaffold(  
    appBar: AppBar(title: new Text('Flutter  
可滚动Widget -- GridView')),  
    body: GridView(  
      gridDelegate:  
  
SliverGridDelegateWithFixedCrossAxisCount(crossAx  
isCount: 3),  
      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')),  
      ],  
    ),  
  );
```

```
}  
}
```

运行效果如下：





2.使用 GridView.count

代码所在位置

`flutter_widget_demo/lib/gridview/GridViewCountWidget.dart`

使用方法

GridView.count 的定义如下：

```
GridView.count({
  Key key,
  Axis scrollDirection = Axis.vertical,
  bool reverse = false,
  ScrollController controller,
  bool primary,
  ScrollPhysics physics,
  bool shrinkWrap = false,
  EdgeInsetsGeometry padding,
  @required int crossAxisCount,
  double mainAxisSpacing = 0.0,
  double crossAxisSpacing = 0.0,
  double childAspectRatio = 1.0,
  bool addAutomaticKeepAlives = true,
  bool addRepaintBoundaries = true,
  bool addSemanticIndexes = true,
  double cacheExtent,
  List<Widget> children = const <Widget>[],
  int semanticChildCount,
  DragStartBehavior dragStartBehavior =
DragStartBehavior.down,
})
```

相比于默认构造函数，其实是将默认构造函数里的 `gridDelegate` 属性，拆分成了 `crossAxisCount`、`mainAxisSpacing`、`crossAxisSpacing` 和 `childAspectRatio`。

使用 `GridView.count` 的 demo 如下：

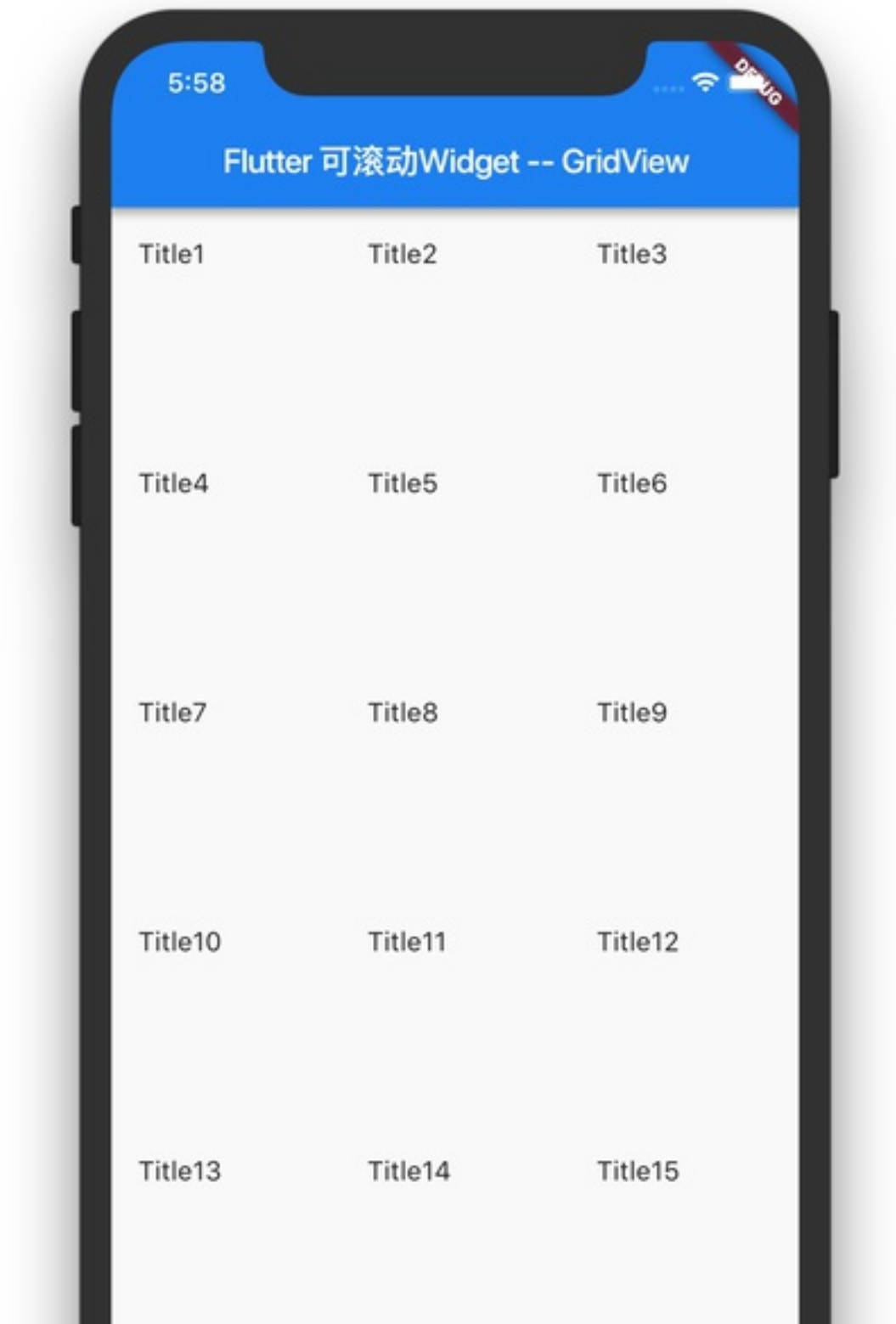
```
import 'package:flutter/material.dart';

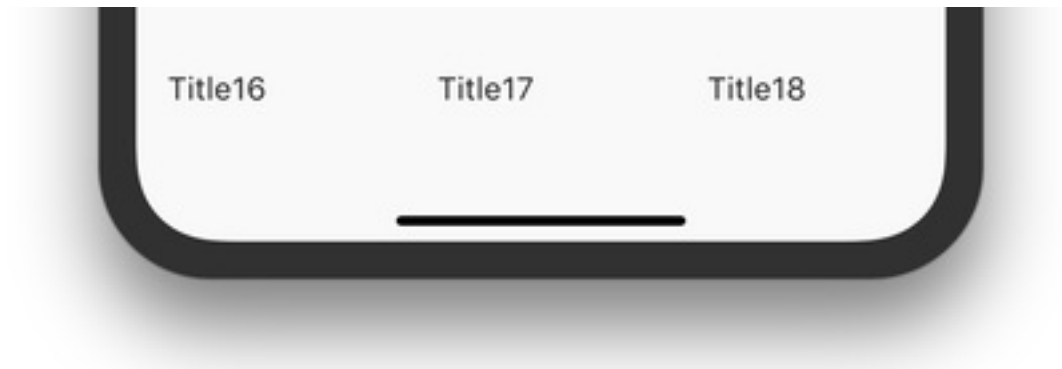
void main() => runApp(GridViewCountWidget());
```

```
class GridViewCountWidget extends StatelessWidget
{
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Test',
      home: Scaffold(
        appBar: AppBar(title: new Text('Flutter
可滚动Widget -- GridView')),
        body: GridView.count(
          crossAxisCount: 3,
          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')),
          ],
        ),
      ),
    );
  }
}
```

```
}  
}  
)  
};
```

运行效果如下：





3.使用 GridView.extent

代码所在位置

flutter_widget_demo/lib/gridview/GridViewExtentWidget.dart

使用方法

GridView.extent 的定义如下：


```
GridView.extent({
  Key key,
  Axis scrollDirection = Axis.vertical,
  bool reverse = false,
  ScrollController controller,
  bool primary,
  ScrollPhysics physics,
  bool shrinkWrap = false,
  EdgeInsetsGeometry padding,
  @required double maxCrossAxisExtent,
  double mainAxisSpacing = 0.0,
  double crossAxisSpacing = 0.0,
  double childAspectRatio = 1.0,
  bool addAutomaticKeepAlives = true,
  bool addRepaintBoundaries = true,
  bool addSemanticIndexes = true,
  List<Widget> children = const <Widget>[],
  int semanticChildCount,
  DragStartBehavior dragStartBehavior =
DragStartBehavior.down,
})
```

这里类似于 GridView.count,因为 GridView.count 相当于 GridView+SliverGridDelegateWithFixedCrossAxisCount, 而 GridView.extent 相当于 GridView+SliverGridDelegateWithFixedCrossAxisCount。

和 GridView.count 的布局算法不同。

使用 GridView.extent 的 demo 如下:

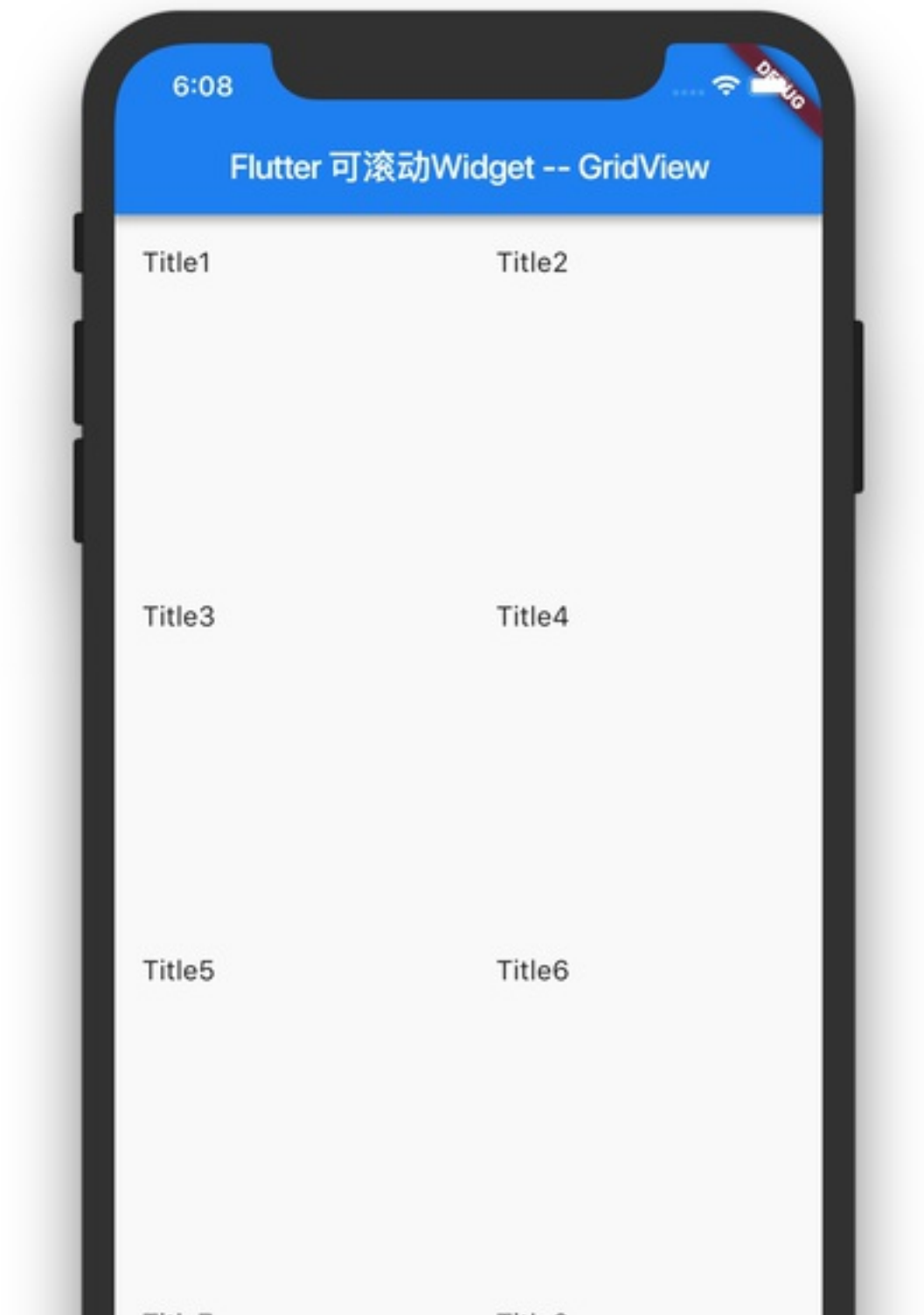
```
import 'package:flutter/material.dart';

void main() => runApp(GridViewExtentWidget());
```

```
class GridViewExtentWidget extends
StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Test',
      home: Scaffold(
        appBar: AppBar(title: new Text('Flutter
可滚动Widget -- GridView')),
        body: GridView.extent(
          maxCrossAxisExtent: 300,
          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')),
          ],
        ),
      ),
    );
  }
}
```

```
}  
}  
);  
)
```

运行效果如下：





4.使用 **GridView.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/gridview/GridViewBuilderWidget.dart`

使用方法

GridView.builder 可以和数据绑定，用于构建大量或无限的列表。而且只会构建那些实际可见的 子Widget。

GridView.builder 的定义如下：

```

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

```

多了和 `ListView.builder` 类似的 `itemCount` 和 `itemBuilder` 属性，用法也是一样的：

```

import 'package:flutter/material.dart';

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

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

```

```

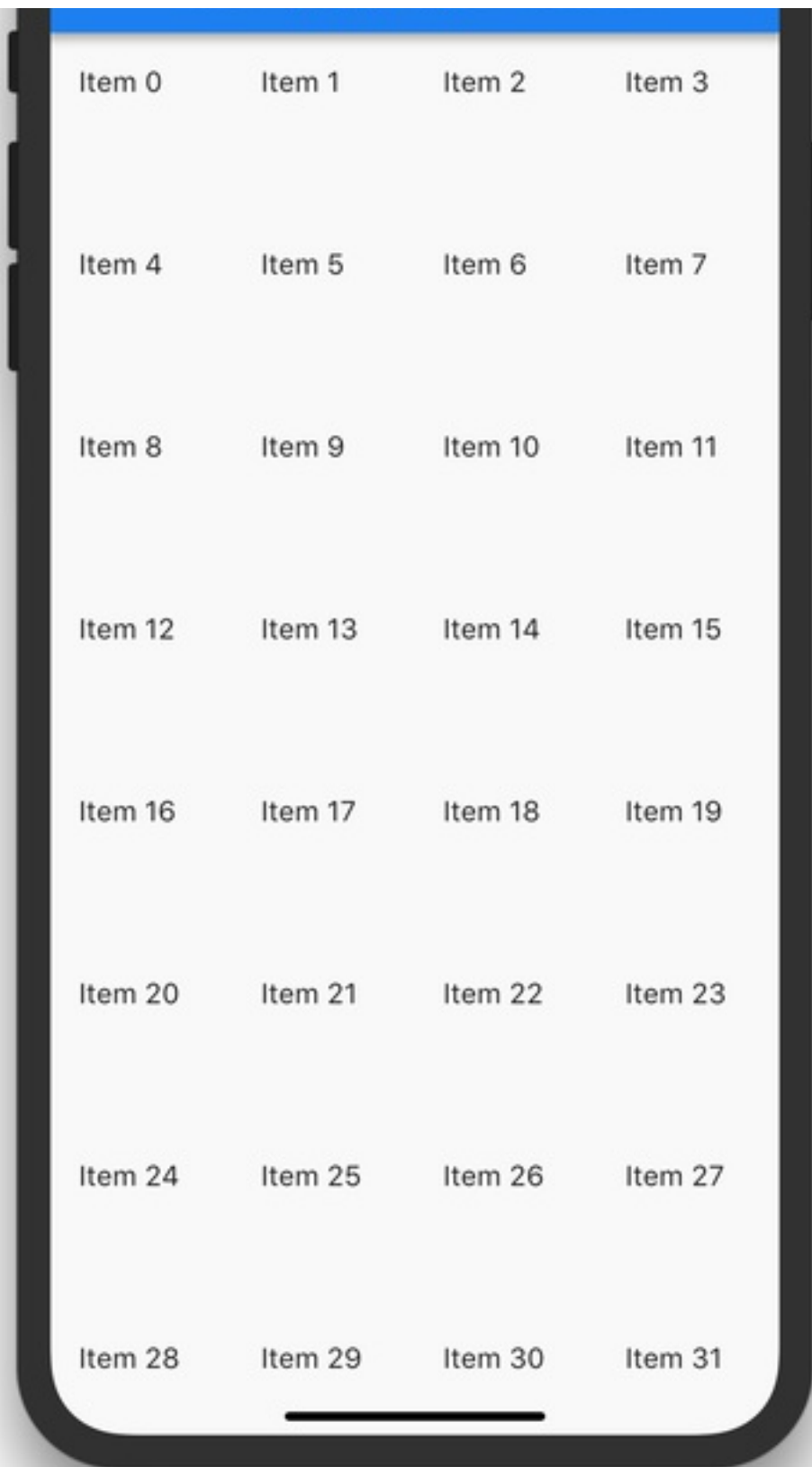
GridViewBuilderWidget({Key key, @required
this.items}) : super(key: key);

@override
Widget build(BuildContext context) {
  return MaterialApp(
    title: 'Test',
    home: Scaffold(
      appBar: AppBar(title: new Text('Flutter 可
滚动Widget -- GridView')),
      body: GridView.builder(
        gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(crossAx
isCount: 4),
        itemCount: items.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text('${items[index]}'),
          );
        },
      ),
    ),
  );
}
}

```

运行效果如下：





5.使用 GridView.custom

代码所在位置

flutter_widget_demo/lib/gridview/GridViewController.dart

使用方法

GridView.custom 的定义如下:

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

增加了 childrenDelegate 的属性, 类型为 SliverChildDelegate, 具有定制 子Widget 的能力, 和 ListView.custom 里的一样, 所以用法也一样:

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

void main() => runApp(GridViewCustomWidget(
  items: List<String>.generate(10000, (i) =>
```



```
"Item $i"),
    ));
```

```
class GridViewCustomWidget extends
```

StatelessWidget {

```
final List<String> items;
```

```
GridViewCustomWidget({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 -- GridView')),
```

```
body: GridView.custom(
```

gridDelegate:

```
SliverGridDelegateWithFixedCrossAxisCount(crossAx  
isCount: 3),
```

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')),
```

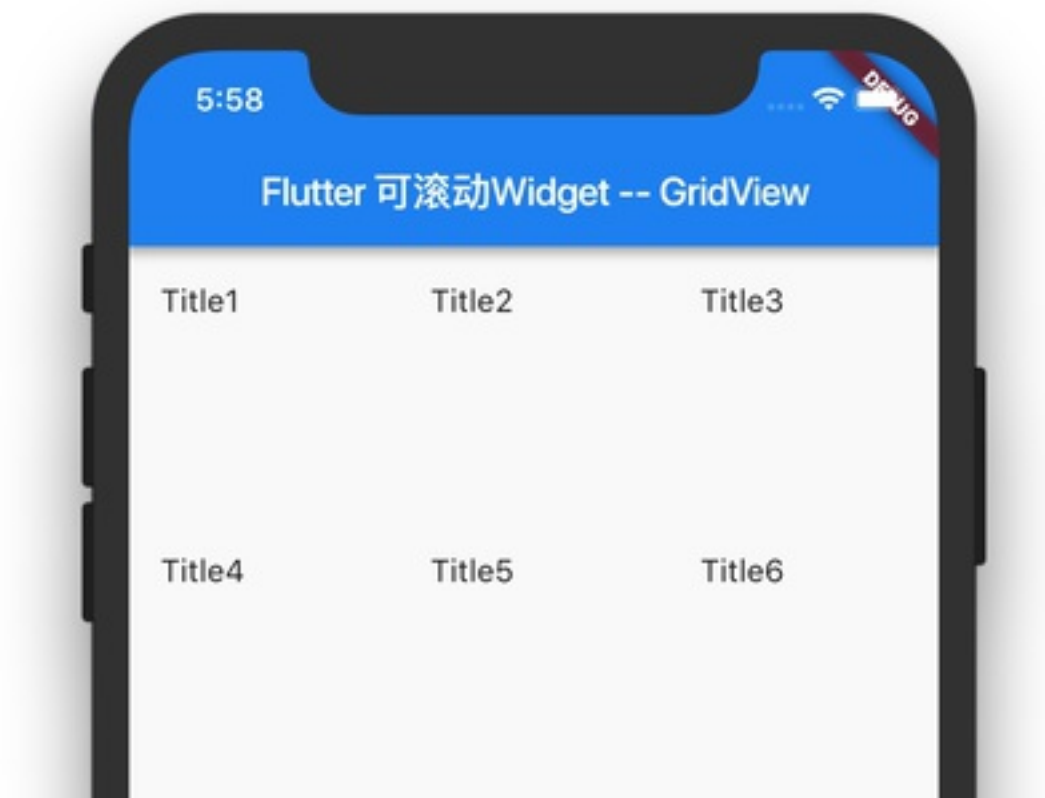
```
ListTile(title: Text('Title9')),
```

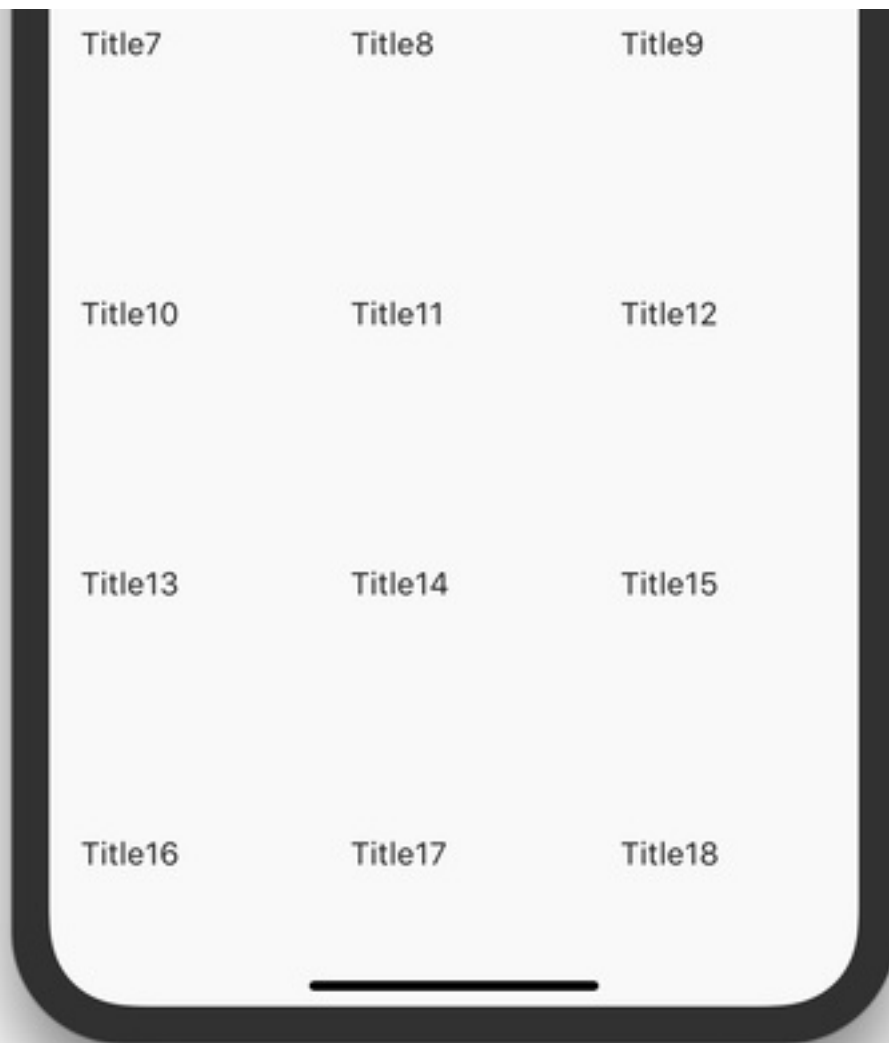
```

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')),
]),
),
),
);
}
}

```

运行效果如下：





GridView 的构造函数及参数说明

GridView 的构造函数，会发现 GridView 的大部分属性都和 ListView 一样：

```

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

```

参数名字

参数类型

key	Key	Widget 的标识
scrollDirection	Axis	滑动的方向 默认为 Axis.verti 控制 GridView 里 顺序排，还是按照

reverse	bool	默认为 false，就入的在头部，当 reverse 为 true 可以控制 GridView ScrollController
controller	ScrollController	1.设置 GridView 2.可以控制 GridView 3.可以读取、设置 可以继承 ScrollController 当 primary 为 true 是否是与父级关联 当为 true 时，即也能滑动
primary	bool	设置 GridView 的值必须为 ScrollPhysics 值：
physics	ScrollPhysics	AlwaysScrollable GridView 里没有 ScrollPhysics():GridView 时候不能滑动 是否根据列表项的度，默认值为 false 当 shrinkWrap 为 true 方向扩展到可占用 当 shrinkWrap 为 false 占用的空间就是其耗性能，因为当其的大小会重新计算
shrinkWrap	bool	
padding	EdgeInsetsGeometry	GridView 的内边距 控制 GridView 中 SliverGridDelegate

gridDelegate	SliverGridDelegate	SliverGridDelegate 横轴 子Widget 为 SliverGridDelegate 横轴 子Widget 为 是否用 Automatic true 在一个 lazy list 里 在滑出可视界面时 addAutomaticKeepAlives 当 子Widget 不需 性能，请把 addA 如果 子Widget 自 此参数必须置为fa 是否用 RepaintBo true 当 addRepaintBo 列表项重绘，提高 但是当列表项重绘 或者一个较短的文 RepaintBoundar 是否用 IndexedS true 使用 IndexedSen 式 GridView 可见部 缓存列表项， 这部分区域的 iter 所以当滑动到这个 的可见， cacheExtent 就表 后面有多少像素
addAutomaticKeepAlives	bool	
addRepaintBoundaries	bool	
addSemanticIndexes	bool	
cacheExtent	double	
children	List<Widget>	GridView 的列表:

`semanticChildCount` `int`

提供语义信息的列
默认为 `GridView`