Al-Mustaqbal University
College of Science
Intelligent Medical System Department

College of Sciences
Intelligent Medical System Department

Lecture 4

GridView Widget in Flutter

Subject: Mobile Applications

Level: Third

Semester: second

Lecturer: Asst. Lect. Ali Saleem Haleem

Google Class Room

Study Year: 2025-2026

Introduction

Flutter's **GridView** is a widget similar to a 2-D array found in many programming languages. As its name indicates, the GridView widget is used to display content in a grid format. This allows us to showcase images, text, icons, and more within the GridView. There are several ways to implement GridView in Flutter:

- *GridView()*
- *GridView.count()*
- *GridView.builder()*
- *GridView.custom()*
- *GridView.extent()*

1. GridView

This constructor enables the creation of a grid by manually providing a list of child elements. It requires a layout to determine how items are arranged.

Property	Description
controller	This property holds the **the** ScrollController class as the object to control the position of the scroll view.
clipBehavior	This property takes **the** Clip enum as the object to decide whether the content in the GridView will be clipped or not.
dragStartBehavior	This property takes DragStartBehavior enum as the object. It controls the way the drag behaviour works.
gridDelegate	SliverGridDelegate class is the object to this property. It is responsible for the delegate that handles the layout of the children widget in the GridView.

```
GridView(
   gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
     crossAxisCount: 2, // 2 columns
     crossAxisSpacing: 10,
     mainAxisSpacing: 10,
   ),
   children: List.generate(20, (index) {
     return Padding(
       padding: const EdgeInsets.all(8.0),
       child: Container(
         color: Colors.green.shade500,
         child: Center(
             child: Text('Item $index',
                 style: TextStyle(fontSize: 30, color: Colors.white))),
       ),
     );
   }),
 ),
```

Output:

2. GridView.count()

GridView.count() is one which is used frequently, the it is used when we already know the size of Grids. Whenever we have to implement GridView dynamically, we use GridView.builder(). Both are just like a normal array and dynamic array. In Flutter, the two GridView is mostly used.

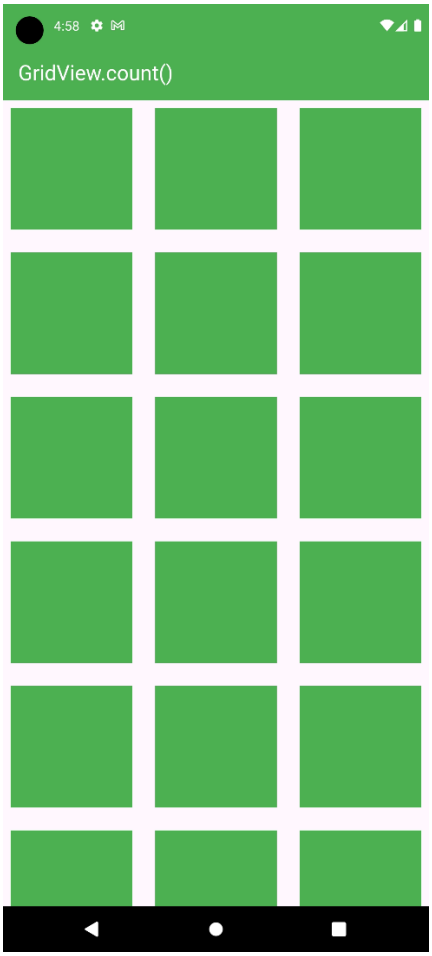
GridView.count() is used with some named parameters. The properties that we can use with GridView.count() are:

Properties of GridView:

Property	Description
crossAxisCount	It defines the number of columns in its **its**.
crossAxisSpacing	It defines the number of pixels between each child listed along the cross axis.
mainAxisSpacing	It defines the number of pixels between each child listed along the main axis.
padding(EdgeInsetsGeometry)	It defines the amount of space to surround the whole list of widgets.
primary	If true, its; 'Scroll Controller' is obtained implicitly by the framework.
scrollDirection	It defines the direction in which the items on GridView will move; the idefault is vertical.
reverse	If it is set to true, it simply reverses the list of widgets in the **use on**irection along the main axis.
physics	It determines how the list of widgets behaves when the user reaches the end or the start of the widget while scrolling.
shrinkWrap	By default, the `shrinkWrap` setting is false. This causes the scrollable list to take up more space than necessary, wasting memory and potentially slowing down the app. To prevent memory issues, set `shrinkWrap` to true. This allows the scrollable list to use only the space needed for its child widgets.

```
GridView.count(
   crossAxisCount: 3, // 3 columns
   crossAxisSpacing: 8,
   mainAxisSpacing: 8,
   children: List.generate(40, (index) {
     return Padding(
       padding: const EdgeInsets.all(8.0),
       child: Container(color: Colors.green),
     );
   }),
 ),
```

Output :



3. GridView.builder()

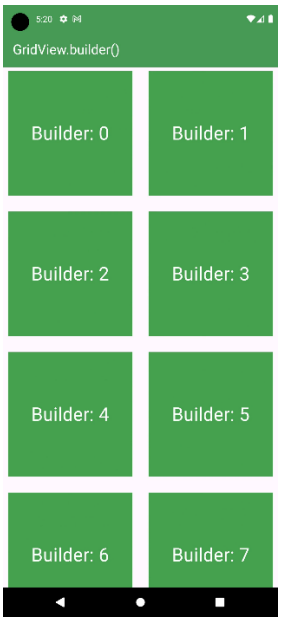
GridView.builder() creates dynamic grids and loads items as needed, improving efficiency for large datasets. It is suitable for displaying data from APIs. **;** presenting data.

Properties of GridView.builder():

Property	Description
gridDelegate	The grid structure is defined.
itemCount	Indicates the quantity of items.
itemBuilder	Creates items dynamically.
physics	Regulates the scrolling behavior.
scrollDirection	Indicates the direction of scrolling.
shrinkWrap	Enhances the efficiency of memory usage.

```
GridView.builder(
   gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,
    crossAxisSpacing: 10,
    mainAxisSpacing: 10,
   ),
   itemCount: 20,
   itemBuilder: (context, index) {
    return Padding(
     padding: const EdgeInsets.all(8.0),
     child: Container(
      color: Colors.green,
      child: Center(
       child: Text("Builder: $index",
       style: TextStyle(
        color: Colors.white,
        fontSize: 30
       ),),),),);}, ),
```

Output:



4. GridView.custom()

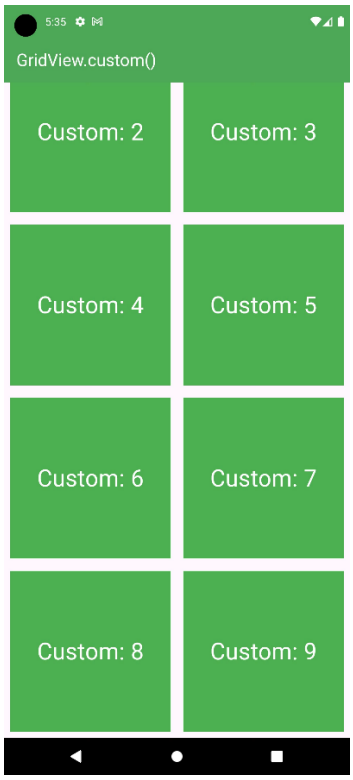
GridView.custom() provides advanced control over the grid structure, allowing for the creation of complex grid layouts tailored to specific needs.

Properties of GridView.custom():

Property	Description
gridDelegate	The grid structure is defined.
childrenDelegate	Efficiently manages child widgets.
physics	Regulates the scrolling behavior.
shrinkWrap	Enhances the efficiency of memory usage.

```
GridView.custom(
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
    crossAxisCount: 2,
  ),
  childrenDelegate: SliverChildBuilderDelegate(
    (context, index) => Padding(
      padding: const EdgeInsets.all(8.0),
      child: Container(
        color: Colors.green,
        child: Center(
          child: Text(
            "Custom: $index",
            style: TextStyle(color: Colors.white, fontSize: 30),
          ),
        ),
      ),
    ),
    childCount: 10,
  ),
 ),
```

Output :



5. GridView.extent()

GridView.extent() is similar to GridView.;**,**ount, it creates a grid where the width of each item is defined, enabling a more flexible and responsive grid layout**.**

Properties of GridView.extent()

Property	Description
maxCrossAxisExtent	Sets the maximum width for each item.
crossAxisSpacing	Adjusts the spacing between columns.
mainAxisSpacing	Adjusts the spacing between rows.
scrollDirection	Sets the scrolling direction.
shrinkWrap	Enhances the efficiency of memory usage.

```
GridView.extent(
   maxCrossAxisExtent: 100,
   crossAxisSpacing: 10,
   mainAxisSpacing: 10,
   children: List.generate(50, (index) {
    return Padding(
      padding: const EdgeInsets.all(8.0),
      child: Container(
        color: Colors.green,
        child: Center(
          child: Text(
            "Extent: $index",
            style: TextStyle(color: Colors.white, fontSize: 10),
          ),
        ),
      ),
    );
   }),
 ),
```

Output:


5:45
GridView.extent()
Extent: 5
Extent: 6
Extent: 7
Extent: 8
Extent: 9
Extent: 10
Extent: 11
Extent: 12
Extent: 13
Extent: 14
Extent: 15
Extent: 16
Extent: 17
Extent: 18
Extent: 19
Extent: 20
Extent: 21
Extent: 22
Extent: 23
Extent: 24
Extent: 25
Extent: 26
Extent: 27
Extent: 28
Extent: 29
Extent: 30
Extent: 31
Extent: 32
Extent: 33
Extent: 34
Extent: 35
Extent: 36
Extent: 37
Extent: 38
Extent: 39
Extent: 40
Extent: 41
Extent: 42
Extent: 43
Extent: 44
Extent: 45
Extent: 46
Extent: 47
Extent: 48
Extent: 49