06. Views

Displaying information
in different ways

- Flutter Views
- GridViews
  - gridView.count (gridview1.dart)
  - gridView.builder (gridview2_builder.dart)
- ListViews
  - listview1.dart
  - listview2_builder.dart
  - Comparison of GridView and ListView
- PageView
  - pageview1.dart
- SingleChildScrollView
  - singlechildscrollview1.dart
- Scrollability

Flutter Views

- Flutter supports specialized Views.
- These views exist because mobile apps have unique challenges, such as limited memory, touch gestures, animations, and scrolling through large datasets

GridViews

- **Grid format**: Ideal for galleries and catalogs
- **Space-efficient**: Shows more items in less space
- **Visually organized**: Great for images, cards, tiles
- **Smooth scrolling** built-in

gridView.count (gridview1.dart)

```
body: GridView.count(
  children: <Widget>[
    Container(...),
    Container(...),
    Container(...),
  ]
```

- The `Grideview.count` factory is used.

```
body: GridView.count(
  crossAxisCount: 2,
  children: <Widget>[
    Container(
      color: Colors.red,
      padding: const EdgeInsets.all(8.0),
      margin: const EdgeInsets.all(8.0),
    ),
```

- `crossAxisCount: 2` makes a two-column layout.
- Multiple widgets are stored in the children property.

gridView.builder (gridview2_builder.dart)

```
GridView.builder(
  itemCount: students.length,
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(...),
  itemBuilder: (context, index) { ... }
```

- We can use the `GridView.builder` factory to make an `itemCount` number of widgets.

gridDelegate property

```
gridDelegate: const liverGridDelegateWithFixedCrossAxisCont(
  // Number of items in each row
  crossAxisCount: 2,
  // Spacing between columns
  crossAxisSpacing: 8.0,
  // Spacing between rows
  mainAxisSpacing: 8.0,
  // Aspect ratio of each grid item
  childAspectRatio: 0.8,
),
```

- The gridDelegate property controls the layout of the grid.

itemBuilder

```
final List<String> students = const ['Alice Johnson', ...];

itemCount: students.length,
itemBuilder: (BuildContext context, int index) {
  ...
}
```

- The students' names are displayed using the itemBuilder.
- The itembuilder iterates `itemCount` times.

```
return Card(
  child: Container(
    child: Column(
      children: <Widget>[
        CircleAvatar(child: Text('${index + 1}',),),
        Text(students[index],),
        Text('ID: ${1000 + index}',),
      ],
```

- For each student, a Card widget is generated with the following:
  - CircleAvatar with the index
  - Student name
  - Student ID

ListViews

- Displaying many items efficiently without performance issues.
  - **Memory efficiency**: Only builds visible items
  - **Infinite scrolling**: Can handle thousands of items
  - **Built-in scroll behavior**: Smooth scrolling behavior

listview1.dart

```
ListView(
  children: const <Widget>[
      Text(...),
      Text(...),
      Text(...)
      ...
  ],
```

- GridView displays the widgets in the `children` property.

listview2_builder.dart

```
ListView.builder(
  itemCount: students.length,
  itemBuilder: (context, index) {
    return ...;
  },
)
```

- We can use `ListView.builder` factory to make `itemCount` number of widgets.

```
body: ListView.builder(
  itemCount: students.length,
  itemBuilder: (context, index) {
    return ListTile(
      leading: CircleAvatar(child: Text('${index + 1}')),
      title: Text(students[index]),
      subtitle: Text('Student ID: ${1000 + index}'),
    );
  },
),
```

- We use ListTile to display the student information.

Comparison of GridView and ListView

- **ListView:** Items arranged in a single line (vertical or horizontal).
- **GridView:** Items arranged in a grid with rows and columns.

- Use GridView for arranging items in a grid (images, icons, products)
- Use GridView for arranging items in a grid (images, icons, products)

PageView

- Swipeable screens let users navigate content (like onboarding, carousels, or tabs) by swiping.
- Each screen is usually full-screen.
- Navigation is smooth, using familiar swipe gestures.

pageview1.dart

```
PageView(
  children: <Widget>[
    Container(color: Colors.red,),
    Container(color: Colors.green,),
    Container(color: Colors.blue,),
  ],
),
```

- The Pageview widget enables swipeable pages.

SingleChildScrollView

- Creating complex scrollable layouts that don't fit ListView or GridView patterns.
  - **Mixed content**: Combines different widget types
  - **Complex layouts**: When ListView/GridView are too restrictive
  - **Full control**: Custom spacing, alignment, and organization

singlechildscrollview1.dart

```
return SingleChildScrollView(
  // ListBody displays Widgets in a List
  child: ListBody(
    children: items.map((i) => Text('Hello $i')).toList(),
  ),
);
```

- With this view and ListBody, we can generate a scrollable screen to display all the content.

Scrollability

> Notice that GridView, ListView, and ScrollView support the scroll feature.

- GridView and ListView use a factory with an itemBuilder.
- SingleChildScrollView supports more complicated controls