CustomScrollView()

Şeritleri (Slider) kullanarak özel kaydırma efektleri oluşturan bir ScrollView .

Bir CustomScrollView , listeler, ızgaralar ve genişleyen başlıklar gibi çeşitli kaydırma efektleri oluşturmak için doğrudan şeritler sağlamanıza olanak tanır . Örneğin SliverAppBar , SliverList ve SliverGrid .

slivers: parametresi bir liste [ ] içerisine sliver türünden nesneler bekler. Container, Text vb. widgetlar olmaz.

```
CustomScrollView(   // Bu widget ile kullanabileceklerimiz…
  slivers: [
    SliverAppBar(),
    SliverList(delegate: delegate),
    SliverFixedExtentList(delegate: delegate, itemExtent: itemExtent),
    SliverGrid(delegate: delegate, gridDelegate: gridDelegate),
    SliverGrid.count(crossAxisCount: crossAxisCount),
    SliverGrid.extent(maxCrossAxisExtent: maxCrossAxisExtent)
    ])
```

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

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scrool(),
    );
  }
}

class Scrool extends StatelessWidget {
  const Scrool({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(),
        body: CustomScrollView(
            slivers: [                          // Buraya yukarıda bahsedilenlerin
                                                // hepsini yerleştireceğiz.
            ])

    );
  }
}
```

SliverAppBar():

```
SliverAppBar(
  title: Text("Sliver App Bar"),       // Üst Bölüme Başlık
  centerTitle: true,                   // Başlığı ortalar
  backgroundColor: Colors.greenAccent,          // Arka plan rengi
  expandedHeight: 250,                 // Yükseklik boyutu
  //flexibleSpace: FlexibleSpaceBar(
    //title: Text('Custom Sliver'),
    //centerTitle: true,
    //background:
Image.network("https://www.alaturkadijital.com/images/services/998546131886-9-
mobil%20app.jpg",
      //fit: BoxFit.cover,          // resim tüm alanı kaplar, genişletilir.
    //),
  //),
  //floating: true,        // Listeyi aşağı kaydırırken listenin elemanları
                           // açılmadan önce AppBarın açılmasını sağlar
  // pinned: true,         // Liste kaydırılırken AppBarın ekrandan tamamen
                           // kaybolmasını engeller. Tek satır şeklinde kalır
  //snap: false,           // Floating true iken true olarak kullanılabilir.
                           // Liste aşağı kaydırılmak istendiği anda AppBar
                           // tüm olarak bir anda ekrana gelir.
),
```

SliverList():

delegate: Bu parametre SliverChildListDelegate sınıfını kullanarak listeyi belirlememizi ister. Sabit Liste oluşturmak için bu sınıftan faydalanırız.

```
SliverList(
  delegate: SliverChildListDelegate([
    Container(
      height: 200,
      color: Colors.blue[100],
      child: Text('Sabit Bölüm 1'),
    ),
    Container(
      height: 200,
      color: Colors.blue[200],
      child: Text('Sabit Bölüm 2'),
    ),
    Container(
      height: 200,
      color: Colors.blue[300],
      child: Text('Sabit Bölüm 3'),
    ),
    Container(
      height: 200,
      color: Colors.blue[400],
      child: Text('Sabit Bölüm 4'),
    ),
  ]),
),
```

Dinamik Liste oluşturmak için *SliverChildBuilderDelegate* kullanırız;

```
SliverList(
  delegate: SliverChildBuilderDelegate((context, index) {
    return Container(
```

```
      height: 200,
      color: Colors.blue[100 * ++index],
      child: Text("Dinamik Bölüm $index"),
    );
  },
      childCount: 3,     // Kaç adet Liste oluşturacağını belirler.
  ),
),
```

SliverFixedExtentList(): SliverList yerine kullanılabilir. Aynı işi yapar yalnız Listelerin boyutu sabit olarak belirlenir. Listeleri oluşturan Contanier içindeki boyut dikkate alınmaz.

Sabit Liste oluşturmak için;

```
SliverFixedExtentList(
    itemExtent: 100, // Listelerin boyutunu piksel olarak belirler.
    delegate: SliverChildListDelegate([
      Container(
        height: 200,
        color: Colors.blue[100],
        child: Text('Sabit Bölüm 1'),
      ),
      Container(
        height: 200,
        color: Colors.blue[200],
        child: Text('Sabit Bölüm 2'),
      ),
      Container(
        height: 200,
        color: Colors.blue[300],
        child: Text('Sabit Bölüm 3'),
      ),
      Container(
        height: 200,
        color: Colors.blue[400],
        child: Text('Sabit Bölüm 4'),
      ),
    ]
    ),
),
```

Dinamik Liste oluşturmak için;

```
SliverFixedExtentList(
  itemExtent: 100,
  delegate: SliverChildBuilderDelegate((context, index) {
    return Container(
      height: 200,
      color: Colors.blue[100 * ++index],
      child: Text("Dinamik Bölüm $index"),
    );
  },
    childCount: 4,     // Kaç adet Liste oluşturacağını belirler.
  ),
),
```

SliverGrid(): Sabit Liste kullanmak için;

```
SliverGrid(
    delegate: SliverChildListDelegate([      // Liste bölümü buraya yazılır
      Container(
        height: 200,
        color: Colors.blue[100],
        child: Text('Sabit Bölüm 1'),
      ),
      Container(
        height: 200,
        color: Colors.blue[200],
        child: Text('Sabit Bölüm 2'),
      ),
      Container(
        height: 200,
        color: Colors.blue[300],
        child: Text('Sabit Bölüm 3'),
      ),
      Container(
        height: 200,
        color: Colors.blue[400],
        child: Text('Sabit Bölüm 4'),
      ),
    ]),

  //gridDelegate:  // Grid bölümlerini belirler. İki tane seçim
                   // kullanabiliriz. FixedCross  ve MaxCross

  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,  // Grid elemanlarını adet olarak belirler
  ),

  //gridDelegate: SliverGridDelegateWithMaxCrossAxisExtent(
  //    maxCrossAxisExtent: 100,  // 100 piksel, Grid elemanlarını boyut olarak
belirler
  //),
),
```

Dinamik Liste kullanarak SliverGrid;

```
SliverGrid(delegate: SliverChildBuilderDelegate((context, index) {
  return Container(
    height: 200,
    color: Colors.blue[100 * ++index],
    child: Text("Dinamik Bölüm $index"),
  );
},
  childCount: 6,  // Nesne adedi
),
  gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 3,
  ),
),
```

Grid Listelerini isimlendirilmiş yapıcılarla da oluşturabiliriz. SliverGrid.count ve SliverGrid.extent;

```
SliverGrid.count(
  crossAxisCount: 3,
  children: [
    Container(
      height: 200,
      color: Colors.blue[100],
      child: Text('Sabit Bölüm 1'),
    ),
    Container(
      height: 200,
      color: Colors.blue[200],
      child: Text('Sabit Bölüm 2'),
    ),
    Container(
      height: 200,
      color: Colors.blue[300],
      child: Text('Sabit Bölüm 3'),
    ),
    Container(
      height: 200,
      color: Colors.blue[400],
      child: Text('Sabit Bölüm 4'),
    ),
  ],
),
```

```
SliverGrid.extent(
  maxCrossAxisExtent: 100,
  children: [
    Container(
      height: 200,
      color: Colors.blue[100],
      child: Text('Sabit Bölüm 1'),
    ),
    Container(
      height: 200,
      color: Colors.blue[200],
      child: Text('Sabit Bölüm 2'),
    ),
    Container(
      height: 200,
      color: Colors.blue[300],
      child: Text('Sabit Bölüm 3'),
    ),
    Container(
      height: 200,
      color: Colors.blue[400],
      child: Text('Sabit Bölüm 4'),
    )
  ],
),
```

Navigator(): Sayfalar arası geçiş sağlar.

Bir sayfadan başka bir sayfaya geçmek için, önceki sayfa yığında kalır;

```
Navigator.push(context, route)
Navigator.of(context).push(route)
```

Bir sayfadan başka bir sayfaya geçmek için, önceki sayfa yığından silinir;

```
Navigator.pushReplacement(context, newRoute)
Navigator.of(context).pushReplacement(newRoute)
```

Bir sayfadan başka bir sayfaya geçmek için, önceki açılmış tüm sayfalar yığından silinir;

```
Navigator.pushAndRemoveUntil(context, newRoute, (route) => false)
Navigator.of(context).pushAndRemoveUntil(newRoute, (route) => false)
```

route: Rota tanımlaması için **MaterialPageRoute** kullanılması gerekir.

```
MaterialPageRoute(builder: builder) // builder parametresi widget döndüren bir
fonksiyon bekler. Fonksiyon ile ekrana gelecek diğer sayfanın bilgisi verilir.

MaterialPageRoute(builder: (context)=>A())  // A() Gidilecek olan sayfa(widget)
```

Gidilen sayfadan bir önceki sayfaya geri dönmek için;

```
Navigator.pop(context);
Navigator.of(context).pop();
```

Bir sayfadan geri dönerken belli bir şarta bağlı olarak istenilen sayfaya geri dönmek için;

```
Navigator.popUntil(context, (route) => false)
Navigator.of(context).popUntil((route) => false)
```

Bir sayfadan geriye dönerken dönülecek sayfa varsa çalışan yoksa çalışmayan;

```
Navigator.maybePop(context);
```

Geriye dönülecek bir sayfa olup olmadığını kontrol etmek için; geriye dönmez.

```
Navigator.canPop(context) // boolean değer döndürür. If kontrolünde kullanılabilir.
```

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

import 'A.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Anasayfa(),
    );
  }
}

class Anasayfa extends StatelessWidget {
  const Anasayfa({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ElevatedButton(
        onPressed: (){

          //Navigator.push(context, MaterialPageRoute(builder: (context)=>A()));

          //Navigator.of(context).push(MaterialPageRoute(builder:
(context)=>A()));

          //Navigator.pushReplacement(context, MaterialPageRoute(builder:
(context)=>A()));
          //Navigator.of(context).pushReplacement(MaterialPageRoute(builder:
(context)=>A()))

          //Navigator.pushAndRemoveUntil(context, MaterialPageRoute(builder:
(context)=>A()), (route) => false);
          //Navigator.of(context).pushAndRemoveUntil(MaterialPageRoute(builder:
(context)=>A()), (route) => false);

        },
        child: Text('A Sayfası'),
      ),
    );
  }
}
```

İkinci bir sayfa oluşturmak için A.dart adında yeni bir dart file oluşturup içerisine aşağıdaki kodları yazarız.

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

class A extends StatelessWidget {
  A({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
          //automaticallyImplyLeading: false, // Gidilen sayfada başlık kısmındaki
                                              // geriye dön işareti gösterilmez.
```

```
      ),
      body: Column(children: [
        Text('A Sayfası'),
        ElevatedButton(
            onPressed: (){

              Navigator.pop(context);
              //Navigator.of(context).pop();

              //Navigator.popUntil(context, (route) => route.isFirst);
              //Navigator.of(context).popUntil((route) => route.isFirst);

              //Navigator.maybePop(context);

              //if (Navigator.canPop(context)) print("Sayfa var");

            },
            child: Text('Geri Dön'))
      ],
      ),
    );
  }
}
```