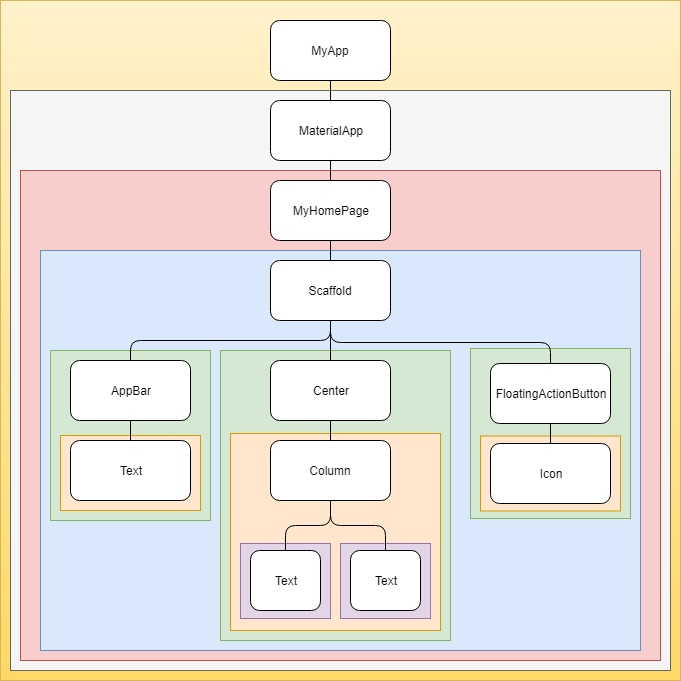
**Widget**

Flutter’da neredeyse her şey widget’tır ve widget dediğimiz kavram bize hazır olarak verilmiş bileşenlerdir. Her bir arayüz elemanı gibi birçok yapı birer widget olarak geçer. Flutter’ın widget özelliklerine ve örneklerine bakmak için widget kataloğunu (flutter.dev) inceleyebilirsiniz.

Widgetler iç-içe geçmiş sınıflardır. İç içe widgetlar demişken, örnek verebiliriz. Bazı widgetlar, içine sadece bir widget kabul ederken; bazı widgetlar içine birden çok widget kabul edebiliyor. Bunu da şöyle düşünebiliriz. Flutter’da, Column diye bir widget vardır ki ilerleyen derslerde sayfa yapıları olarak bunu da ayrıntılı işleyeceğiz, birden çok widget alabilir. Çünkü Column widgetını, alt alta hizalanan bir liste gibi düşünebiliriz. Böylece aslında birden fazla eleman alabileceğini anlayabiliriz kolaylıkla. Fakat Container widgetını ise kutu gibi düşünüyoruz ve buna ise sadece bir eleman ekleyebiliyoruz.



En çok kullanılan widgetlardan MaterialApp ve Scaffold ve AppBar widgetlarına da değinelim.

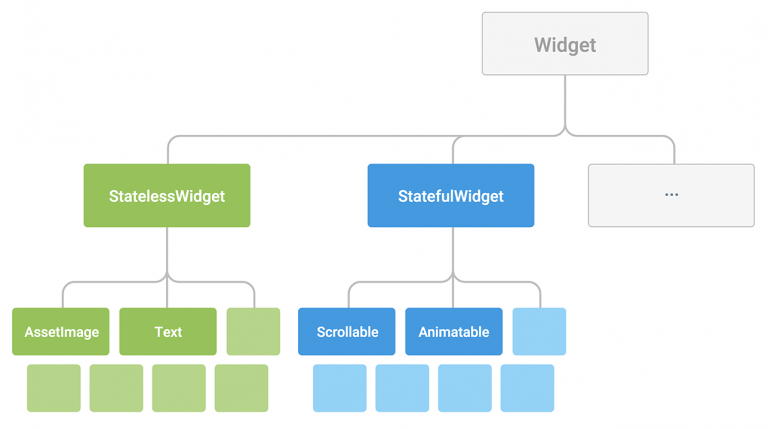
MaterialApp widgetı, en zengin widgetlardan biridir. Main sınıfımızda projemizi oluşturduktan sonra run app metodu içerisine genellikle bir material app veririz. Diğer widgetları kullanabilmek için özellikle yönlendirme işlemi için MaterialApp widgetına ihtiyaç duyarız. Uygulamamızı oluştururken, en dış kısmı olarak bu widgetı düşünebiliriz.

Ardından Scaffold widgetı, kelime manası inşaat iskelesi olan aslında bize ekran iskeletini oluşturan yapıdır. Scaffold ile ekranımızı oluştururuz. Eğer scaffold oluşturmazsak, içerisine ne eklersek ekleyelim siyah bir ekran görürüz. Scaffold ile beyaz bir ekran oluşturduktan sonra bunu şekillendirmek tamamen bize kalıyor.

AppBar widgetı da, Scaffold ile oluşturduğumuz ekranın içinde, üst kısımda oluşturduğumuz bir yapıdır. Burada uygulamanın adını yazabilir, bu üstte çıkan çizgi gibi alanın rengini değiştirebiliriz.

State nedir?

State, oluşturduğumuz uygulamanın durumu olarak tanımlanabilir. Ekranın anlık görüntüsüdür. Bir ekranın görüntüsünü yani State’i etkileyen birçok widget vardır. Örnek vermek gerekirse: yazılar, resimler yine en basit örnekleridir. State değiştiğinde, ekrandaki görüntü de değişir. O yüzden, kullandığımız widgetlara göre state seçimini doğru kulanmak önemlidir.



Stateful ve Stateless Widget farkı nedir?

Uygulamamızı Stateful veya Stateless widget sınıflarıyla oluşturuyoruz. Bu sınıflar da birer widgettır. Fakat neye göre Stateless veya Stateful widget sınıfı oluşturuyoruz, bu biraz kafa karıştırabiliyor. Şimdi sırayla incelediğimizde aslında çok kolay olduğunu beraber görebiliriz.

Öncelikle anlamlarına bakarak ip ucu yakalayabiliriz.

Stateless = Durumsuz,   Stateful = Durumsal

Eğer, kullanacağımız ekranda değişen herhangi bir yapımız yoksa bunu Stateless widget kullanarak oluştururuz. Sabit yapılarla, durumsuz bir haldir. Yani değişen bir şey yoktur. Örnek olarak, koyacağımız bir başlık yazısı gibi değişmeyen widgetlarla, stateless widget kullanırız.

Eğer, kullanacağımız ekranda widgetlarda değişiklik olacaksa bunu Stateful widget kullanarak oluştururuz. Değişken yapılarla, durumsal bir haldir yani belirli durumlara sahiptir. Örnek olarak, ekranda bir saat göstermek istersek veya sayaçlı bir sistem gibi sürekli değişen değerlerde, stateful widget kullanırız.

Sayfa Yapısı

Center

Bunlardan ilk olarak Center widget’ı ile başlayabiliriz. Center widget’ı, merkez anlamına gelir. Yani, içine aldığı elemanı ekranın merkezine yerleştirir. Ve Center widget’ı sadece bir tane widget alabildiği için, child kullanır.

Örnek olarak kod yazımı ve nasıl göründüğünü aşağıdaki gibi siz de deneyebilirsiniz:

Center(

child: Text( "İbrahim AYAZ",

style: TextStyle(

fontSize: 20

),)

)

Container

Container, kelime anlamıyla da benzerlik gösterir. Bu widget’ımızı, bir kutu gibi düşünebiliriz. Yani sayfamızı bir kutunun içine inşa ediyormuşuz gibi hayal ederiz. Container, bize bir kutu çizer ve bu kutunun özelliklerini kendimiz değiştirebiliyoruz. Container widget’ı da sadece bir widget alabildiği için child kullanır.

Şimdi yukarıda öğrendiğimiz Center’ın içine Container widget’ını oluşturarak, Container’ın özelliklerine bakacağız. Center’ın içine yazmadan, direk tüm sayfayı kutu gibi de kullanabilirdik yani direk Container yazabiliriz body kısmına. Bu örneklerimizde sadece, sayfanın merkezinde olursa daha iyi görünüm açısından kolaylık sağlamasını amaçlıyoruz.

Center(

child: Container(

child: Text(

"İbrahim AYAZ",

style: TextStyle(

fontSize: 20 ),

),

)

)

Container’a biraz özellik vererek aşağıdaki görüntüyü elde ederiz. width ile genişlik değeri, height ile yükseklik değerini kendimiz belirleriz. color ile de Colors. dedikten sonra istediğimiz rengi seçebiliriz, arkaplanı için.

Center(

child: Container(

width: 150,

height: 100,

color: Colors.redAccent,

child: Text(

"İbrahim AYAZ",

style: TextStyle(

fontSize: 20

),

),

)

)

Row

Row widget’ı ile sütun kontrol ediyoruz. Yani Row widget’ı kullanırsak, içine ekleyeceğimiz widget’lar yan yana hizalanır. Bu yüzden sütun sütun düşünebiliriz. Haliyle birden fazla widget aldığı için children kullanırız.

Kullanımına örnek olarak hemen aşağıdaki kodu deneyebiliriz:

Row(

children: <Widget>[

Container(

width: 50,

height: 50,

color: Colors.purple,

),

Container(

width: 50,

height: 50,

color: Colors.yellowAccent,

),

Container(

width: 50,

height: 50,

color: Colors.deepOrange,

),

Container(

width: 50,

height: 50,

color: Colors.lightGreenAccent,

)

],

)

Aşağıda sırası ile row özelliğine …

mainAxisAligment: MainAxisAligment.start  (başlangıç noktasına göre hizalamaya başlar) ,

mainAxisAligment: MainAxisAligment.center  (ortalayarak hizalar) ,

mainAxisAligment: MainAxisAligment.end  (sonda hizalar) özelliklerinin sonuçlarını görüyoruz.

Aşağıda ise sırasıyla,

mainAxisAligment: MainAxisAligment.spaceBetween (başta ve sonda boşluk olmayacak şekilde aralarına eşit mesafe koyarak hizalama) ,

mainAxisAligment: MainAxisAligment.spaceEvenly  (satıra yayarak aralarına eşit mesafe koyarak hizalama) özelliklerinin sonucunu görüyoruz.

Column

Column widget’ı ile de satırları yönetiyoruz. Yani içine eklediğimiz widgetlar, alt alta hizalanır. Böylece bunu satır satır düşünebiliriz. Column widget’ı da birden fazla widget almaya elverişli olduğundan children kullanırız.

Kullanımına örnek olarak hemen aşağıdaki kodu deneyebiliriz:

Column(

crossAxisAlignment: CrossAxisAlignment.start,

children: <Widget>[

Row(

children: <Widget>[

Container(

width: 50,

height: 50,

decoration: BoxDecoration(

color: Colors.blueAccent,

borderRadius: BorderRadius.all(Radius.circular(50))

),

),

Container(

width: 50,

height: 50,

decoration: BoxDecoration(

color: Colors.blueAccent,

borderRadius: BorderRadius.all(Radius.circular(50))

),

),

Container(

width: 50,

height: 50,

decoration: BoxDecoration(

color: Colors.blueAccent,

borderRadius: BorderRadius.all(Radius.circular(50))

),

),

],

),

Container(

width: 50,

height: 50,

color: Colors.purple,

),

Container(

width: 50,

height: 50,

color: Colors.yellowAccent,

),

Container(

width: 50,

height: 50,

color: Colors.deepOrange,

),

Container(

width: 50,

height: 50,

color: Colors.lightGreenAccent,

)

],

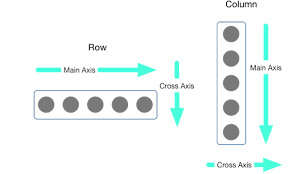
)

Oluşturduğumuz Column widget’ının içinde: ilk satıra bir Row, diğer satırlarına da birer Container widget’ı ekleyerek 5 satıra tamamladık. Flutter’da bildiğimiz gibi, widget içine widget ekleyerebiliriz. Böylece sayfayı çok kolay bir şekilde istediğimiz gibi tasarlayabiliriz.

Ayrıca burada dikkat ettiyseniz, Column widget’ında hizalamak için crossAxisAligment kullandık.

Column ve Row widget’ları benzer özelliklerdedir. Sadece içine aldıkları widgetlarda hizalama yönleri değişiklik kazanır. mainAxisAlignment, Row’da x ekseni olabilirken; Column’da y eksenidir. crossAxisAlignment da, Row’da y ekseni iken; Column’da x eksenidir.

Bunu da aşağıdaki görseldeki gibi ayırt edebiliriz.



Kaynak:

https://www.mobilhanem.com/flutter-genel-sayfa-yapilari/