YouTube BM DERSLERİ

www.youtube.com/BMdersleri

MAKÜ
BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ

Flutter ile Mobil Programlamaya Giriş

Flutter

9.HAFTA

Stateless ve Statefull Widget Kavramları
Hot Reload ve Full Restart Kavramları
Temel Widgetlar (Text, container, Column ve row)

Hazırlayan : **Zeynep İrem KESLER 1911404048**

Tarih : 11/05/2022

Sürüm : v1

Ders Yürütücüsü : Doç. Dr. İsmail KIRBAŞ

İÇİNDEKİLER

Stateless ve Statefull

Stateless Widget (Durumsuz Widget):

Durumu değişmeyecek ve ekranda bir kere çizildikten sonra değişime uğramayacak widget'ların sınıfıdır. Yani oluşturulan sayfa üzerinde bir etkileşim olmayacaksa, herhangi bir değişiklik yapılmayacaksa stateless widgetlar kullanılır.

Stateless widget içinde bulunan **build(BuildContext context)** yöntemi, bu pencere aracının temsil ettiği kullanıcı ara yüzünün bölümünü açıklar ve widget ağacındaki konumunu belirler.

Stateless ve Statefull

StatelessWidget (Durumsuz Widget):

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

void main() {
  runApp(
    MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        appBar: AppBar(
          title: Text("Stateless Widget"),
        ),
        body: Deneme(),
      ),
    ),
  );
}
class Deneme extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        width: MediaQuery.of(context).size.width / 2,
        height: 100,
        child: Center(
          child: Text(
            "Stateless Widget",
            style: TextStyle(fontSize: 25),
          ),
        ),
        decoration: BoxDecoration(
            color: Colors.blue,
            borderRadius: BorderRadius.all(Radius.circular(20.0))),
      ),); } }
```

Stateless ve Statefull

Statefull Widget (Durumlu Widget):

Kullanıcıyla etkileşim halinde olan widgetlardır. Sayfa üzerinde yeniden işlem yapılabilmesini sağlarlar.

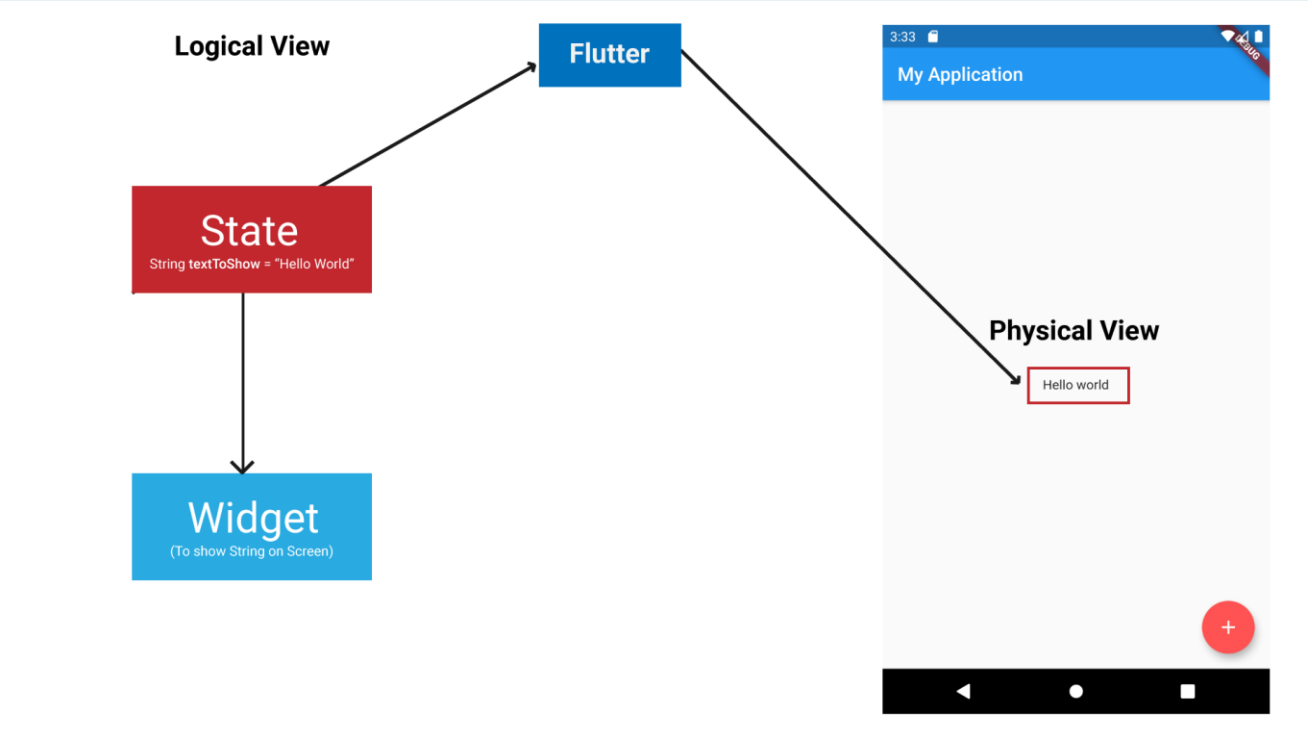
```
class HorizontalSeperator extends StatelessWidget {

  final Color color;
  final double thickness;

  HorizontalSeperator({this.color,this.thickness});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: MediaQuery.of(context).size.width,
      height: thickness,
      decoration: BoxDecoration(
          color: color,
          borderRadius: BorderRadius.all(Radius.circular(20.0))),
    );
  }
}
```

- İki farklı class yapısından oluşur. İlk class'ta class kelimesinden sonra isim verilir. Kullanıcı isteğini belirtir, o da alt class'a iletir. Şöyle düşünülebilir; bakkalı arayıp sipariş veriliyor, o da çırağına iletiyor ve siparişlerin teslimi sağlanıyor.
- Üst class'ta tanımlanan değişkenler final türündedir ve değiştirilemez.
- Alt class'taki değişkenler ise dinamiktir yani başına final tanımlaması yapılmaz.

Stateless ve Statefull

State Nedir ?

State, yapısı sadece **StatefullWidget** içerisinde kullanılmaktadır. Uygulamanın durumunu temsil etmektedir. Bu durum arka planda birçok parametreye bağlıdır ancak kullanıcı tarafından sadece arayüz gözüktüğünden ekranın anlık görüntüsüdür diyebiliriz. Bir ekranın görüntüsünü yani State'i etkileyen bir çok widget vardır. State değiştiğinde ekrandaki görüntü de değişir. Bu sebeple kullandığımız widgetlara göre state seçimini doğru yapmak oldukça önemlidir.



Stateless ve Statefull

***initState()* Kullanımı:**

StatefullWidget çalıştığı anda onunla birlikte çalışmasını istediğimiz kodların yazıldığı yerdir. Sayfa ilk açıldığında kendiliğinden çalışması istenen kod yapıları varsa **initState()** içinde kullanılır. Çalışması istenilen kodlar ise **süper.initState()** yapısının altına yazılır.

```
@override
Void initState() {
Super.initState() ;
……. Kod Blokları
}
```

***Dispose()* Kullanımı:**

initState() yapısının altında çalışan kod blokları program kapatılırken program hafızasından da kaldırılmalıdır. **Dispose** metodu bu kodların kaldırılmasını sağlar. Kaldırılmasını istediğimiz kodlar süper.initState() yapısının üstüne yazılır.

```
@override
Void dispose() {
……. Kod Blokları
Super.dispose() ;
}
```

Stateless ve Statefull

SetState Yapısı:

SetState yapısı yine çok önemli yapılardan biridir. StatefullWidget içinde kullanılır. Değişken ya da değişkenlere atadığımız değer ya da değerlerin değişmesi durumunda değişikliğin ekranda yeniden işlenmesi için "SetState" fonksiyonu içinde yapılması gerekir. SetState içinde yapılan değişiklik "build" metoduna bildirilir. Eğer "**setState**" yapısı içinde yapılmazsa değişkenin değeri değişir fakat ekranda işlenme gerçekleşmeyeceği için ekranda herhangi bir değişiklik görülmez.

Örnek:

```
49
50  class _MyHomePageState extends State<MyHomePage> {
51    int _counter = 0;
52
53    void  incrementCounter() {
54      setState(() {
55        // This call to setState tells the Flutter framework that something has
56        // changed in this State, which causes it to rerun the build method below
57        // so that the display can reflect the updated values. If we changed
58        // _counter without calling setState(), then the build method would not be
59        // called again, and so nothing would appear to happen.
60        _counter++;
61      });
62    }
```

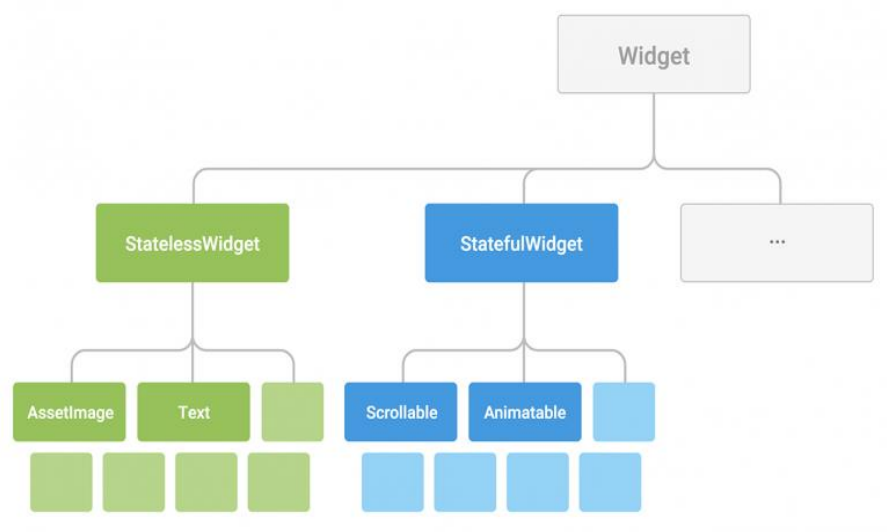
Bir butonun tıklama olayında ekranda bir şeylerin değişmesi isteniyorsa değişikliği yapacak değişken değeri **SetState** yapısı içinde yapılmalıdır.

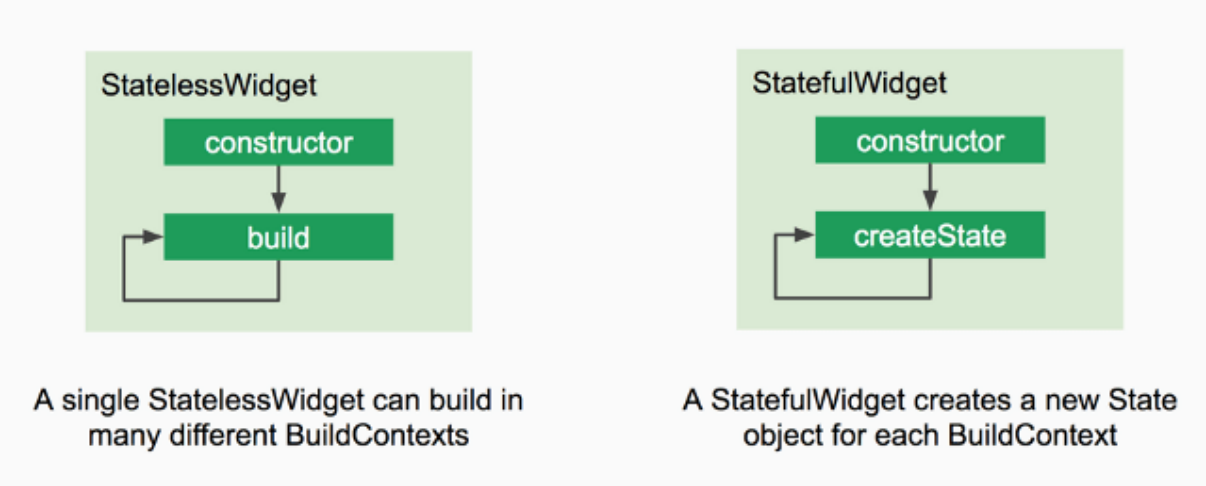
Stateless ve Statefull

Stateless ve Statefull Farkı:

Eğer, kullanacağımız ekranda değişen herhangi bir yapımız yoksa bunu ***Stateless widget*** kullanarak oluştururuz. 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.





Hot Reload ve Full Restart

Flutter'ın çalışırken yeniden yükleme özelliği, hızlı ve kolay bir şekilde denemeler yapmanıza, kullanıcı arayüzleri oluşturmanıza, özellikler eklemenize ve hataları düzeltmenize yardımcı olur. Çalışırken yeniden yükleme, güncellenmiş kaynak kodu dosyalarını çalışan Dart Sanal Makinesi'ne (VM) enjekte ederek çalışır . VM, alanları ve işlevlerin yeni sürümleriyle sınıfları güncelledikten sonra, Flutter çerçevesi, pencere öğesi ağacını otomatik olarak yeniden oluşturarak, değişikliklerinizin etkilerini hızlı bir şekilde görüntülemenize olanak tanır.

Bir Flutter uygulaması çalışırken hot reload yapmak için:

1. Uygulamayı, desteklenen bir Flutter düzenleyicisinden veya bir terminal penceresinden çalıştırın. Fiziksel veya sanal bir cihaz hedef olabilir. **Yalnızca hata ayıklama modundaki Flutter uygulamaları çalışırken hot reload yapılabilir.**
2. Projenizdeki Dart dosyalarından birini değiştirin. Çoğu kod değişikliği türü çalışırken hot reload yapılabilir. Hot restart gerektiren değişikliklerin listesine HotRestart adresinden ulaşabilirsiniz.
3. Flutter'ın IDE araçlarını destekleyen bir IDE/editörde çalışıyorsanız, Tümünü Kaydet'i (cmd-s/ ctrl-s) seçin veya araç çubuğundaki çalışırken yeniden yükle düğmesini tıklayın.

Hot Reload ve Full Restart

Çalışırken Hot Reload, çalışırken Hot Restart ve Full Restart arasındaki fark nedir?

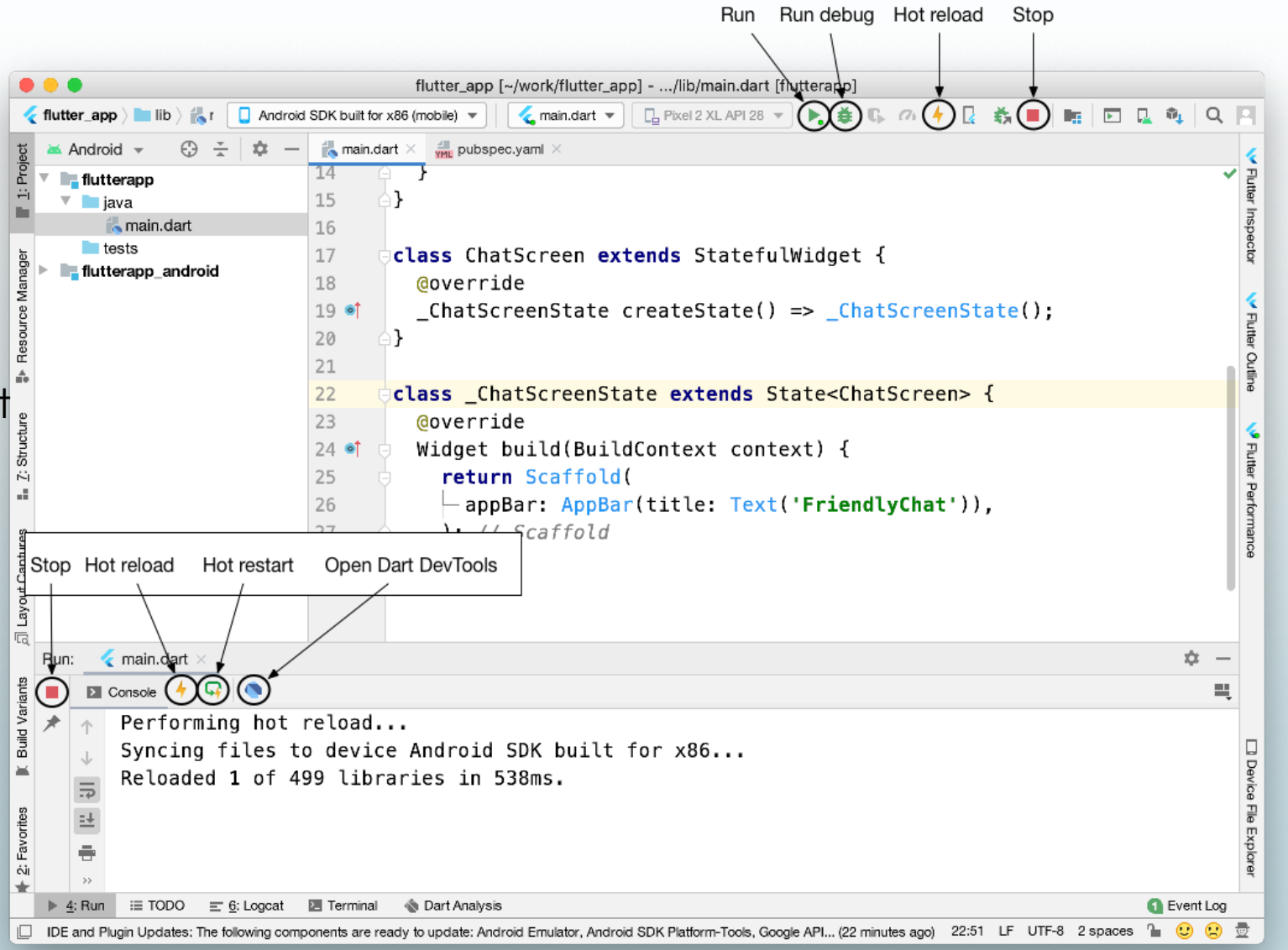
- Çalışırken **Hot Reload**, kod değişikliklerini sanal makineye yükler ve uygulama durumunu koruyarak widget ağacını yeniden oluşturur; tekrar çalışmaz main() veya initState().
- Çalışırken **Hot Restart**, kod değişikliklerini sanal makineye yükler ve Flutter uygulamasını yeniden başlatarak uygulama durumunu kaybeder.

- **Full Restart**, iOS, Android veya web uygulamasını yeniden başlatır. Java / Kotlin / ObjC / Swift kodunu da yeniden derlediği için bu daha uzun sürer. Web'de Dart Geliştirme Derleyicisini de yeniden başlatır. Bunun için belirli bir klavye kısayolu yoktur; çalıştırma yapılandırmasını durdurmanız ve başlatmanız gerekir.

NOT: Flutter web şu anda çalışırken yeniden başlatmayı destekliyor ancak çalışırken yeniden yüklemeyi desteklemiyor.

Hot Reload ve Full Restart

Bir kod değişikliği, yalnızca değiştirilmiş Dart kodu değişiklikten sonra yeniden çalıştırılırsa görünür bir etkiye sahiptir. Özellikle, çalışırken hot reload, mevcut tüm pencere öğelerinin yeniden oluşturulmasına neden olur. Yalnızca pencere öğelerinin yeniden oluşturulmasına dahil olan kod otomatik olarak yeniden yürütülür. Örneğin ve işlevleri yeniden çalıştırılmaz main() initState().



Hot Reload ve Full Restart

Enumerated types

Hot reload doesn't work when enumerated types are changed to regular classes or regular classes are changed to enumerated types.

For example:

Before the change:

```
enum Color {
  red,
  green,
  blue,
}
```

After the change:

```
class Color {
  Color(this.i, this.j);
  final int i;
  final int j;
}
```

Temel Widgetlar

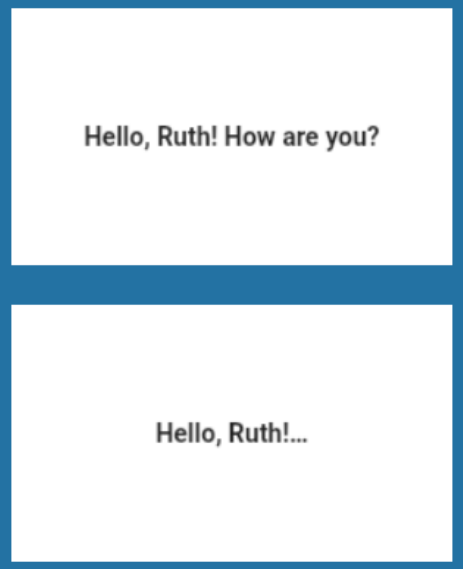
- **Text:** Text Widget, uygulamanız içinde bir dizi stil metni oluşturmanıza olanak tanır.
- **Row,Column:** Bu esnek widget'lar, hem yatay (Row) hem de dikey (Column) yönlerde esnek düzenler oluşturmanıza olanak tanır. Bu nesnelerin tasarımı, web'in esnek kutu yerleşim modeline dayanmaktadır.
- **Stack:** Bir pencere öğesi , doğrusal olarak (yatay veya dikey olarak) yönlendirilmek yerine, Stack pencere öğelerini boya sırasına göre birbirinin üzerine yerleştirmenize olanak tanır. Ardından, a öğesinin alt öğelerinde, onları yığının üst, sağ, alt veya sol kenarına göre konumlandırmak için Positionedwidget'ı kullanabilirsiniz. StackYığınlar, web'in mutlak konumlandırma düzeni modelini temel alır.
- **Container:** Container Widget, dikdörtgen bir görsel öğe oluşturmanıza olanak tanır. BoxDecoration Bir kap, arka plan, kenarlık veya gölge gibi bir ile dekore edilebilir . A'nın Container boyutuna uygulanan kenar boşlukları, dolgu ve kısıtlamalar da olabilir. Ek olarak a Container, bir matris kullanılarak üç boyutlu uzaya dönüştürülebilir.

➢ Widget Örneklerinizi Deneyin

Temel Widgetlar

Text:

Bu örnek, taşma TextOverflow.ellipsis olarak ayarlanmış şekilde Metin pencere aracını kullanarak metnin nasıl görüntüleneceğini gösterir .




```
Text(
  'Hello, $_name! How are you?',
  textAlign: TextAlign.center,
  overflow: TextOverflow.ellipsis,
  style: const TextStyle(fontWeight: FontWeight.bold),
)
```

Temel Widgetlar

Row,Column:

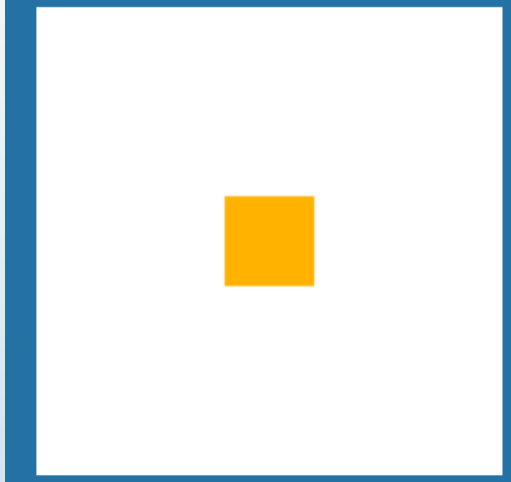
Bu örnek, üç widget'ı dikey olarak düzenlemek için bir Sütun kullanır , sonuncusu kalan tüm alanı dolduracak şekilde yapılmıştır.


```
Column(
  children: const <Widget>[
    Text('Deliver features faster'),
    Text('Craft beautiful UIs'),
    Expanded(
      child: FittedBox(
        fit: BoxFit.contain, // otherwise the logo will be tiny
        child: FlutterLogo(),
      ),
    ),
  ],
)
```

Temel Widgetlar

Stack:

Bu örnek, komşu gereçlerden uzak durması için bir kenar boşluğu ile 48x48 sarı bir kareyi (üst pencere aracının Kapsayıcının alması gereken boyutla ilgili kendi fikirleri olması durumunda bir Merkez pencere aracının içine yerleştirilmiş) gösterir:

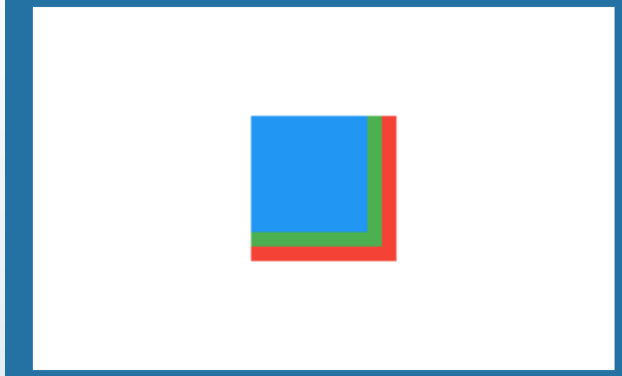


```
Center(
  child: Container(
    margin: const EdgeInsets.all(10.0),
    color: Colors.amber[600],
    width: 48.0,
    height: 48.0,
  ),
)
```

Temel Widgetlar

Container:

Bir Yığın kullanarak widget'ları birbiri üzerine konumlandırabilirsiniz.



```
Stack(
  children: <Widget>[
    Container(
      width: 100,
      height: 100,
      color: Colors.red,
    ),
    Container(
      width: 90,
      height: 90,
      color: Colors.green,
    ),
    Container(
      width: 80,
      height: 80,
      color: Colors.blue,
    ),
  ],
)
```

MAKÜ
BURDUR MEHMET AKİF ERSOY ÜNİVERSİTESİ

Yardımcı Kaynaklar

- Adım Adım Flutter İle Mobil Uygulamalar (Rakıcı Oğuz , 2021)

İlginiz için teşekkürler…

Hazırlayan : **Zeynep İrem KESLER 1911404048**
E-posta : zeynepiremkesler@gmail.com
Tarih : 11/05/2022
Ders Yürütücüsü : Doç. Dr. İsmail KIRBAŞ
E-posta : ismkir@gmail.com