

# Flutter ile Uygulama Geliştirme Kursu | Android & IOS

## Flutter Çalışma Yapısı

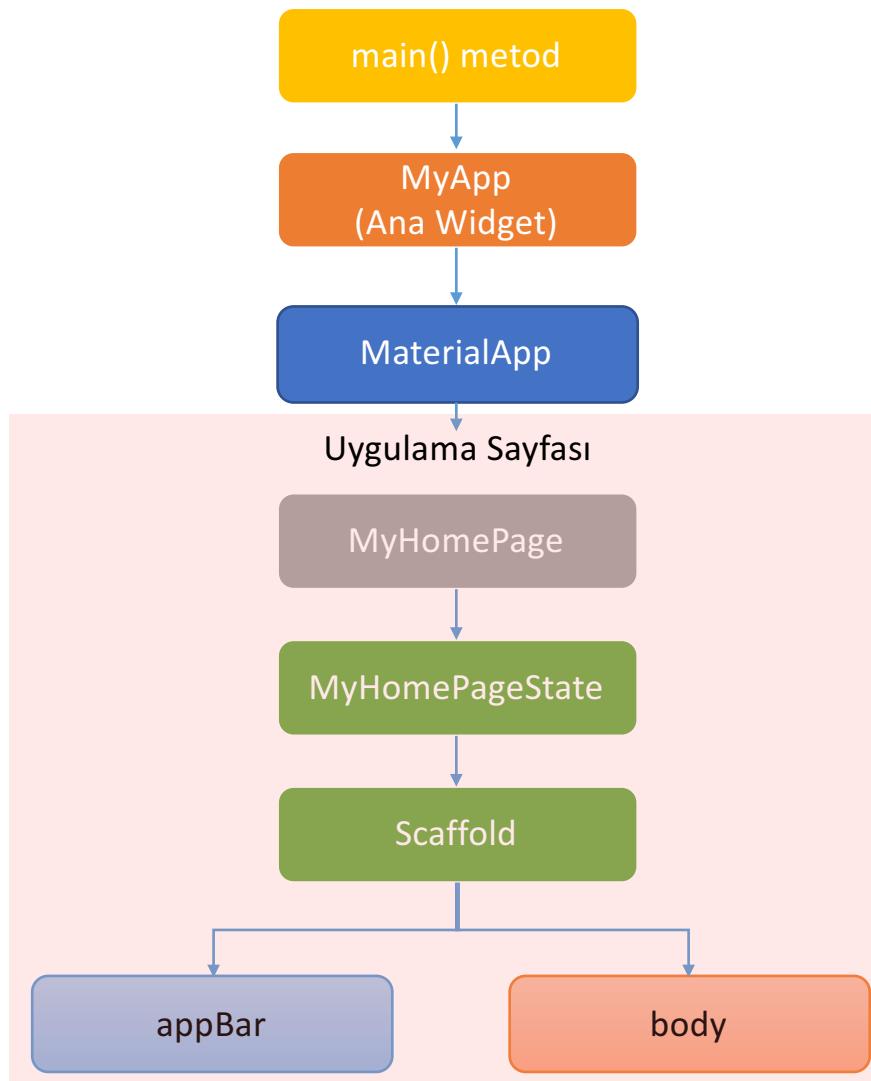
Kasım ADALAN

Elektronik ve Haberleşme Mühendisi  
Android - IOS Developer and Trainer

# Eğitim İçeriği

- Uygulama Yapısı
- State Yapısı
- Uygulama Sayfası Oluşturma
  - StatefulWidget
  - StatelessWidget
- Sayfalar Arası Geçiş
- Geri Dönüş Tuşları Kullanımı
- Sayfalar Arası Veri Transferi
- Back Stack
- Yaşam Döngüsü
- Widget İçinde Kodlama Teknikleri
- FutureBuilder
- Uygulama APK'sı oluşturma
- Google Play Üzerinde Yayınlama

# Uygulama Yapısı



```
void main() {  
  runApp(MyApp());  
}  
main() metod
```

```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Flutter Demo',  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
        visualDensity: VisualDensity.adaptivePlatformDensity,  
      ), // ThemeData  
      home: MyHomePage(title: 'Uygulama Yapısı'),  
    ); // MaterialApp  
  }  
}
```

MyApp  
(Ana Widget)

MaterialApp

```
class MyHomePage extends StatefulWidget {  
  MyHomePage({Key key, this.title}) : super(key: key);  
  
  final String title;  
  
  @override  
  _MyHomePageState createState() => _MyHomePageState();  
}
```

```
class _MyHomePageState extends State<MyHomePage> {  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ), // AppBar  
      body: Row(  
        children: <Widget>[  
          Text("Merhaba")  
        ], // <Widget>[]  
      ), // Row  
    ); // Scaffold  
  }  
}
```

Uygulama Sayfası

# main() metod

main() metod

```
|void main() {  
|    runApp(MyApp());  
|    //Main metod yani ilk çalışan metoddur.  
|    //Uygulamanın ana widgettini çalıştırır.  
|}
```

# Ana Widget

MyApp  
(Ana Widget)

- Uygulamanın genel özelliklerinin verildiği ve uygulamanın anasayfasının çalıştırılmak için tanımlandığı widgettir.

```
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Flutter Demo',  
      debugShowCheckedModeBanner: false,  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
        visualDensity: VisualDensity.adaptivePlatformDensity,  
      ), // ThemeData  
      home: MyHomePage(title: "Uygulama Yapısı"),  
    ); // MaterialApp  
  }  
}
```

Uygulamaya Material özelliği verdığımız yerdir.  
Android için görev yöneticisinde çalışan uygulamalarda bu isim görünür.  
IOS için arka plandaki uygulamalar listesi çıktığında bu isim kullanılır.

appBar üzerindeki banner'ı kaldırır.

Uygulamanın temasını belirler appBar rengi gibi.

appBar rengi

Arayüz birleşenlerinin örn: button vb.bütün platformlarda güzel bir şekilde görünmesini sağlar.

Anasayfa Tanımlaması : Uygulamanın ilk sayfası burada tanımlanır.  
Bu sayfa StatefulWidget veya StatelessWidget özelliğinde olabilir.

# Uygulama Sayfa Yapısı

# Uygulama Sayfası

- Uygulamanın kullanıcı tarafından görünen kısmıdır.
- Tasarım ve sayfa çalışması ile ilgili kodlamalar burada yer alır.
- Uygulama sayfası 2 çeşit olabilir.
  1. StatelessWidget
  2. StatefulWidget
- Sağ taraftaki örnek StatefulWidget'tır.

```
class MyHomePage extends StatefulWidget {  
  MyHomePage({Key key, this.title}) : super(key: key);  
  
  final String title;  
  
  @override  
  _MyHomePageState createState() => _MyHomePageState();  
}
```

```
class _MyHomePageState extends State<MyHomePage> {  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ), // AppBar  
      body: Row(  
        children: <Widget>[  
          Text("Merhaba")  
        ], // <Widget>[]  
      ), // Row  
    ); // Scaffold  
  }  
}
```

Tasarım ve sayfa çalışması ile ilgili kodlamalar için gerekli alan.

# Uygulama Sayfa Yapısı

```
class _MyHomePageState extends State<MyHomePage> {  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text(widget.title),  
      ), // AppBar  
      body: Row(  
        children: <Widget>[  
          Text("Merhaba")  
        ], // <Widget>[]  
      ), // Row  
    ); // Scaffold  
  }  
}
```



# State Yapısı

# State Yapısı

- Eğer ara yüzde değişiklik yapacak değişken oluşturacaksanız kullanılır.
- `setState()` metodu değişkene state özelliği katmaktadır.
- `setState()` metodu ile değişken değeri kodlama içinde değiştiği anda ilişkili olduğu bütün kodlamalarda anlık olarak değişir.
- `setState()` metodu StatefulWidget özelliği olan sınıflar içinde olabilir.
- Örnek kodlamada sayac 1'er 1'er artacaktır ve Text içinde anlık değeri görülecektir.

```
class _MyHomePageState extends State<MyHomePage> {  
  
    int sayac = 0;  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text("Yeni Başlık"),  
            ), // AppBar  
            body: Center(  
                child: Column(  
                    mainAxisAlignment: MainAxisAlignment.center,  
                    children: [  
                        ElevatedButton(  
                            child: Text("Tıkla"),  
                            onPressed: (){  
                                setState(() {  
                                    sayac = sayac + 1;  
                                });  
                            },  
                        ), // ElevatedButton  
                        Text("Sonuç : $sayac"),  
                    ],  
                ), // Column  
            ), // Center  
        ); // Scaffold  
    }  
}
```

# Klasik Yöntem ve State Yapısı Karşılaştırması.

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var etiket: UILabel!

    var sayac = 0

    override func viewDidLoad() {
        super.viewDidLoad()

        etiket.text = "Sonuç : \(self.sayac)"
    }

    @IBAction func tikla(_ sender: Any) {
        self.sayac = self.sayac + 1
        etiket.text = "Sonuç : \(self.sayac)"
    }
}
```

Uygulama ilk açıldığı anda label içinde sayıç değeri 0 görünür. Butona tıklanıldığı anda sayıç artar ve sayıç değeri button metodu içinde anlık olarak arayüzde görünmesi için label içine aktarılır. Bu sayıç artımı ve label içine aktarımı dikkat edin button metodu içinde gerçekleşti.

```
class _MyHomePageState extends State<MyHomePage> {

    int sayac = 0;

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text("Yeni Başlık"),
            ), // AppBar
            body: Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                        ElevatedButton(
                            child: Text("Tıkla"),
                            onPressed: () {
                                setState(() {
                                    sayac = sayac + 1;
                                });
                            },
                        ), // ElevatedButton
                        Text("Sonuç : $sayac"),
                    ],
                ), // Column
            ), // Center
        ); // Scaffold
    }
}
```

Uygulama ilk açıldığı anda Text içinde sayıç değeri 0 görünür. Butona tıklanıldığı anda sayıç artar ve sayıç değeri kodlama içinde var olan her yerdeki değerini değiştirir ve kodları günceller. Bu sayıç artımı dikkat edin button metodu içinde olduğu halde sayıç değeri text içine aktarım farklı bir yerde yer almaktadır. Bu durum state özelliği ile oluşmaktadır.

# Uygulama Sayfası Oluşturma

#StatelessWidget  
#StatefulWidget

# Stateless Widget

- Bir kere çalışan widget'lardır.
- State özelliği ile arayüzde tekrar tekrar değişiklik yapılamaz.
- Build metodu bir kere çalışır.
- **StatelessWidget'lar state özelliğini kullanamazlar.**
- StatelessWidget'lara örnek verebiliriz : Text, Column, Row, Container vb.
- Bu widgetlar bir durum değiştiremezler.
- Örnek olarak Row sadece içeriğini gösterebilir bir durum değiştiremez.

StatelessWidget

build

# Stateless Widget Sayfa Yapısı

StatelessWidget

build

```
class Sayfa1 extends StatelessWidget{  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Stateless Sayfa"),  
      ), // AppBar  
      body: Row(  
        children: <Widget>[  
          Text("Merhaba"),  
        ], // <Widget>[]  
      ), // Row  
    ); // Scaffold  
  }  
}
```

build içindeki return  
icerisine gerekli widget  
kodlaması yapılabilir.

Not : key her widget'ın tanımlayıcısıdır.  
Widget'a özel işlemlerde key kullanılabilir.  
Key kullanılmamasada olur.

# Stateless Widget Sayfa Yapısı

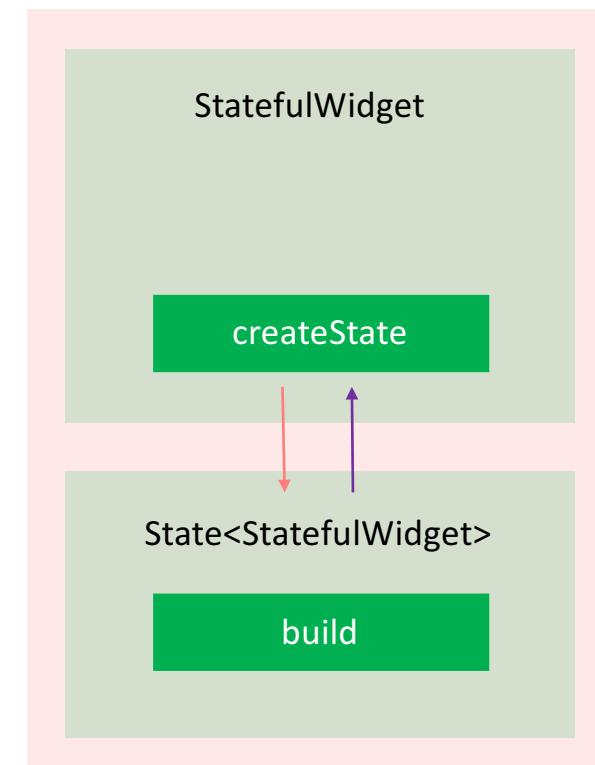
```
void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ), // ThemeData
      home: Sayfa1(),
    ); // MaterialApp
  }
}

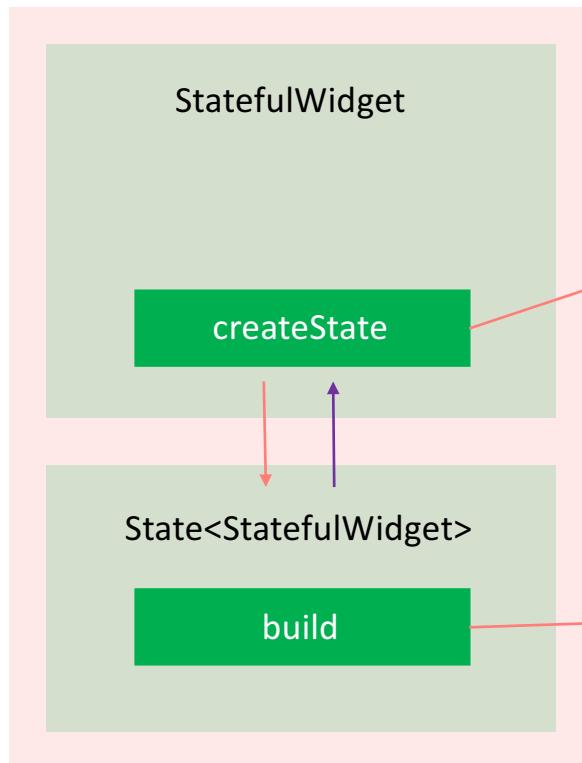
class Sayfa1 extends StatelessWidget{
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Stateless Sayfa"),
      ), // AppBar
      body: Row(
        children: <Widget>[
          Text("Merhaba"),
        ], // <Widget>[]
      ), // Row
    ); // Scaffold
  }
}
```

# Stateful Widget

- Arayüzde tekrar tekrar değişiklik yapabileceğimiz widgetlardır.
- Build metodu arayüzdeki her değişimde çalışır. Build metodunu setState() metodu tetiklemektedir.
- State özelliği bu yapı ile kullanılabilir.
- State özelliğini kullanmak için StatefulWidget özelliği olan bir sınıf olmalı ve bu sınıfı implement etmeliyiz.
- Bundan dolayı iki sınıf ile çalışmaktadır.
- Biri  `StatefulWidget`  özelliği alan sınıf ve diğerinin  `State< StatefulWidget >` özelliği olan sınıf.
- StatefulWidget'lara örnek verebiliriz : Button, Checkbox, TextField vb.
- Bu widgetlar bir durum değiştirebilirler.
- Örnek olarak buttona basıldığında bir işlem olabilir.



# Stateful Widget Sayfa Yapısı



```
class Sayfa2 extends StatefulWidget {  
    @override  
    _Sayfa2State createState() => _Sayfa2State();  
}  
  
class _Sayfa2State extends State<Sayfa2> {  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text("Stateful Sayfa"),  
            ), // AppBar  
            body: Row(  
                children: <Widget>[  
                    Text("Merhaba"),  
                ], // <Widget>[]  
            ), // Row  
        ); // Scaffold  
    }  
}
```

build içindeki return  
icerisine gerekli widget  
kodlaması yapılabilir.

Not : key her widget'ın tanımlayıcısıdır.  
Widget'a özel işlemlerde key kullanılabilir.  
Key kullanılmamasada olur.

# Stateful Widget Sayfa Yapısı

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ), // ThemeData
      home: Sayfa2(),
    ); // MaterialApp
  }
}
```

```
class Sayfa2 extends StatefulWidget {
  @override
  _Sayfa2State createState() => _Sayfa2State();
}

class _Sayfa2State extends State<Sayfa2> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Stateful Sayfa"),
      ), // AppBar
      body: Row(
        children: <Widget>[
          Text("Merhaba"),
        ], // <Widget>[]
      ), // Row
    ); // Scaffold
  }
}
```

# Sayfalar Arası Geçiş

- Sayfalar arası geçiş için *Navigator* yapısı kullanılır.

```
Navigator.push(context, MaterialPageRoute(builder: ( context) => SayfaA() ));
```



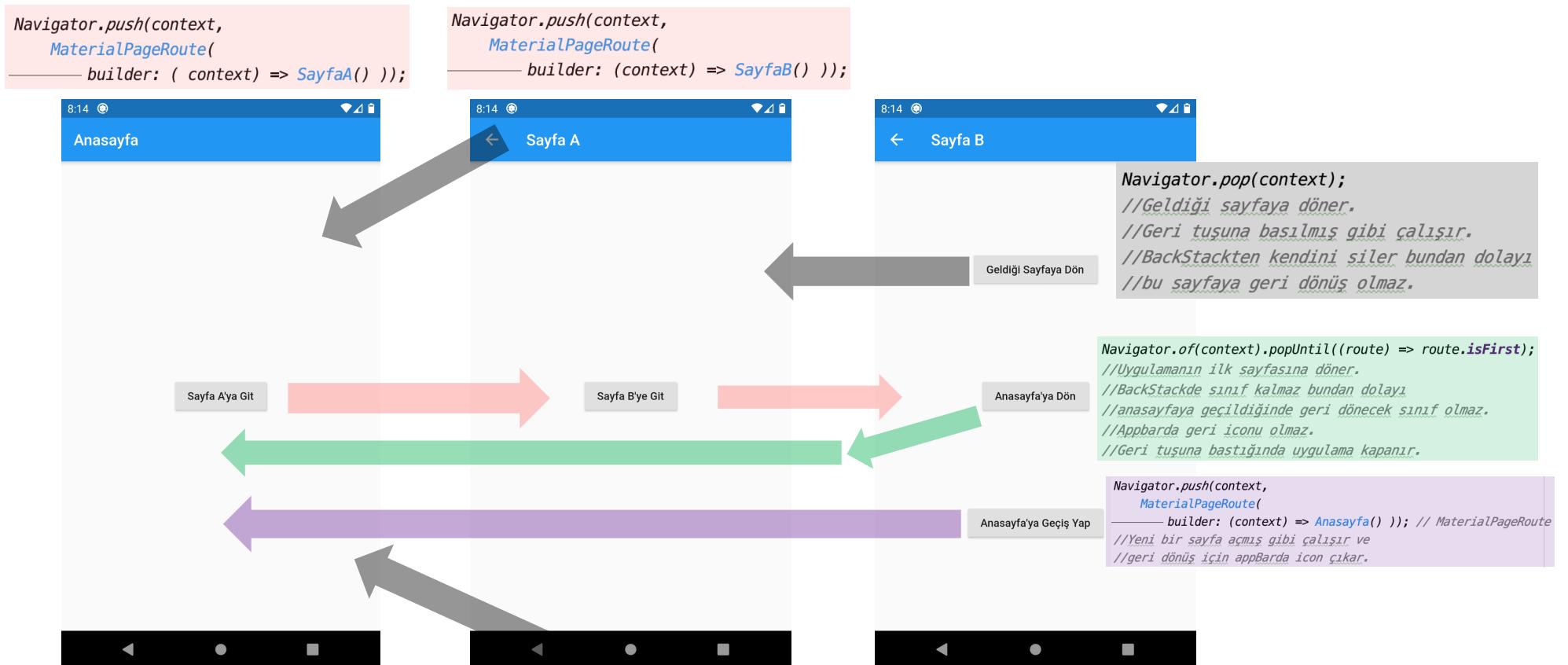
# Otomatik Geri Dönüş

```
Navigator.push(context, MaterialPageRoute(builder: ( context) => SayfaA() ));
```

- Bu kodlama ile geçiş yaparsak otomatik olarak geri dönüş sağlayabiliriz.
- Geçiş yapılan sayfadan geri dönmek için ;
- *Android* : Hem geri tuşu hem appBar üzerindeki geri iconu kullanılır.
- *IOS* : appBar üzerindeki geri iconu kullanılır çünkü IOS işletim sisteminde geri tuşu yoktur.



# Sayfalar Arası Geçiş



# Geri Dönüş Tuşlarını Kullanma

- AppBar üzerindeki tuşu kontrol edebiliriz.
- Bu tuş hem ios hem android için aktif kullanılır.
- AppBar üzerinde kendimiz geri tuşu oluştururuz.
- Daha sonra içine yazılan kodlamayı çalıştırabiliriz.

```
class _SayfaAState extends State<SayfaA> {  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text("Sayfa A"),  
                leading: IconButton(  
                    icon: Icon(Icons.arrow_back),  
                    onPressed: () {  
                        print("AppBar üzerindeki geri tuşu tıklandı");  
                    },  
                ), // IconButton  
            ), // AppBar  
            body: Center(  
                child: Column(  
                    children:  
                ),  
            ),  
        );  
    }  
}
```

## Örnek

```
leading: IconButton(  
    icon: Icon(Icons.arrow_back),  
    onPressed: () {  
        print("AppBar üzerindeki geri tuşu tıklandı");  
        Navigator.push(context,  
            MaterialPageRoute(  
                builder: (context) => SayfaB() )); // MaterialPageRoute  
    },  
), // IconButton
```



# Geri Dönüş Tuşlarını Kullanma

- Aşağıdaki geri tuşunu kontrol edebiliriz.
- Bu tuş sadece android için aktif kullanılır.
- body içinde willPopScope oluşturulur.
- Geri tuşuna basıldığında çalışacak bir metod oluşturulur.

```
class SayfaB extends StatelessWidget {  
  
  Future<bool> geriDonusTusu(BuildContext context) async {  
    print("Geri tuşu tıklandı.");  
    return false;  
    //true olursa geri dönüş yapar.  
    //false veya hiçbir return ifade olmazsa geri dönüş olmaz.  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Sayfa B"),  
      ), // AppBar  
      body: WillPopScope(  
        onWillPop: () => geriDonusTusu(context),  
        child: Center(  
          child: Column(  
            mainAxisSize: MainAxisSize.spaceEvenly,
```

context sayfa  
geçişinde veya  
widget işlemlerinde  
gerekli olabilir.

## Örnek

```
Future<bool> geriDonusTusu(BuildContext context) async {  
  print("Geri tuşu tıklandı.");  
  Navigator.of(context).popUntil((route) => route.isFirst);  
  return true;  
}
```



# Sayfalar Arası Veri Transferi

- Sayfalar arası veri transferi için geçiş yapılacak sayfada değişken oluşturmalıyız.

```
class SayfaA extends StatelessWidget {

    String isim;
    int yas;
    double boy;
    bool bekarMi;

    SayfaA({required this.isim,required this.yas,required this.boy,required this.bekarMi,})

    @override
    Widget build(BuildContext context) {
        return Scaffold(
            appBar: AppBar(
                title: Text("Sayfa A"),
            ), // AppBar
            body: Center(
                child: Column(
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: <Widget>[
                        Text("İsim : ${isim}"), Text("Yaş : ${yas}"),Text("Boy : ${boy}"),Text("Bekar Mı : ${bekarMi}"),
                    ], // <Widget>[]
            ), // Column
        ), // Center
    ); // Scaffold
}
}
```

# Sayfalar Arası Veri Transferi

- Veriyi transfer ederken geçiş yapacağımız sınıfın parametresine gerekli veriler yerleştirilir.

```
body: Center(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: <Widget>[
      ElevatedButton(
        child : Text("Veri Gönder"),
        onPressed: (){

          Navigator.push(context,
            MaterialPageRoute(builder: ( context) =>
              SayfaA(isim:"Ahmet",yas:18,boy:1.78,bekarMi:true,) )); // MaterialPageRoute

        },
      ), // ElevatedButton
    ], // <Widget>[]
  ), // Column
), // Center
```

# Sayfalar Arası Veri Transferi

- Nesne tabanlı çalışmak için geçiş yapılacak sınıfıta veri transfer etmek için nesne oluşturulur.

```
class Kisiler{  
    String isim;  
    int yas;  
    double boy;  
    bool bekarMi;  
  
    Kisiler(this.isim, this.yas, this.boy, this.bekarMi);  
}
```

```
var kisi = Kisiler("Ahmet", 18, 1.78, true);  
  
Navigator.push(context,  
    MaterialPageRoute(builder: ( context) =>  
        SayfaA(kisi: kisi) )); // MaterialPageRoute
```

```
class SayfaA extends StatelessWidget {  
  
    Kisiler kisi;  
    SayfaA({required this.kisi});  
  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text("Sayfa A"),  
            ), // AppBar  
            body: Center(  
                child: Column(  
                    mainAxisAlignment: MainAxisAlignment.center,  
                    children: <Widget>[  
                        Text("İsim : ${kisi.isim}"), Text("Yaş : ${kisi.yas}"),  
                        Text("Boy : ${kisi.boy}"), Text("Bekar Mı : ${kisi.bekarMi}"),  
                    ], // <Widget>[]  
                ), // Column  
            ), // Center  
        ); // Scaffold  
    }  
}
```

# Sayfalar Arası Veri Transferi

- StatefulWidget özelliği olan sayfa

```
class Kisiler{  
    String isim;  
    int yas;  
    double boy;  
    bool bekarMi;  
  
    Kisiler(this.isim, this.yas, this.boy, this.bekarMi);  
}
```

```
var kisi = Kisiler("Ahmet", 18, 1.78, true);  
  
Navigator.push(context,  
    MaterialPageRoute(builder: ( context) =>  
        SayfaB(kisi: kisi) )); // MaterialPageRoute
```

```
class SayfaB extends StatefulWidget {  
  
    Kisiler kisi;  
    SayfaB({required this.kisi});  
  
    @override  
    _SayfaBState createState() => _SayfaBState();  
}  
  
class _SayfaBState extends State<SayfaB> {  
    @override  
    Widget build(BuildContext context) {  
        return Scaffold(  
            appBar: AppBar(  
                title: Text("Sayfa B"),  
            ), // AppBar  
            body: Center(  
                child: Column(  
                    mainAxisAlignment: MainAxisAlignment.center,  
                    children: <Widget>[  
                        Text("İsim : ${widget.kisi.isim}"), Text("Yaş : ${widget.kisi.yas}"),  
                        Text("Boy : ${widget.kisi.boy}"), Text("Bekar Mı : ${widget.kisi.bekarMi}"),  
                    ], // <Widget>[]  
            ), // Column  
            center: Center(  
            ), // Center  
        ); // Scaffold  
    }  
}
```

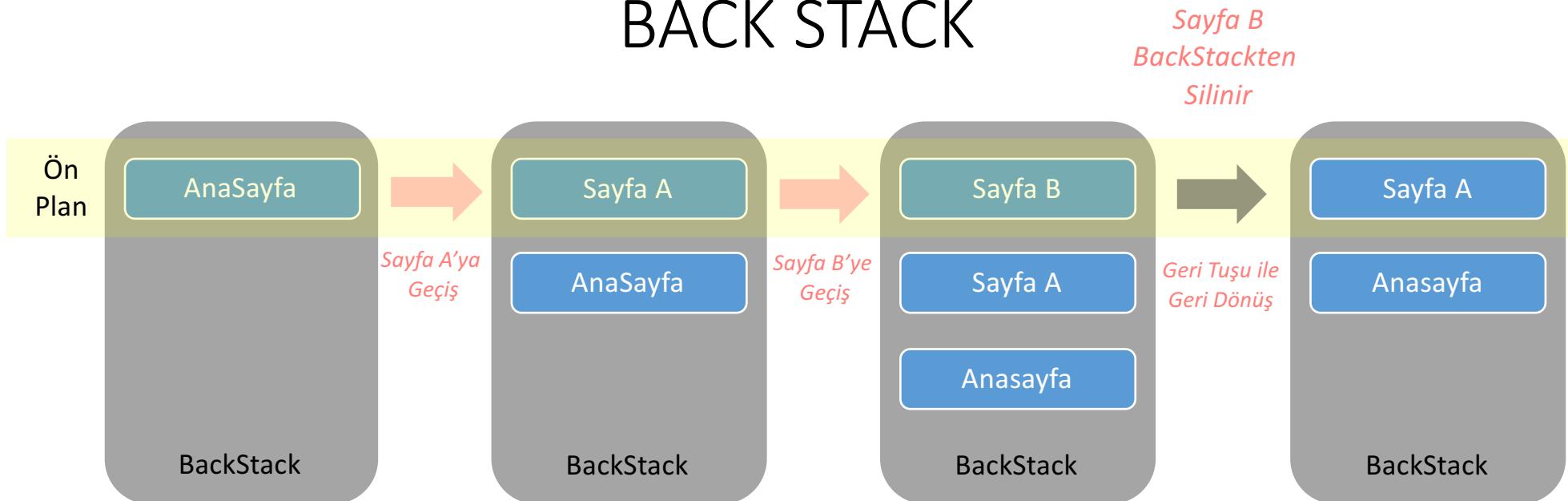
widget ile üst sınıfın  
değişkenine erişim  
sağlanır.

# BACK STACK

# BACK STACK

- **Task** nedir daha iyi anlamak için web tarayıcının çalışmasını örnek olarak alalım.
- **Task** sekme, etkileşim pencere (**Activity**) ve link **Intent** varsayıyalım. İstediğiniz zaman **Back**(Geri) butona basıp geri dönebilirsiniz ama ileriye dönemezsınız çünkü ilerdeki sayfa siliniyor.
- İleriye dönmek için yine linke (**Intent**) basmamız gerekecek.
- Sonuçta web tarayıcının ve Android arasındaki benzerliği şöyle gösterebiliriz:
  - Tarayıcı – Android
  - Sekme – **Task** (Görev)
  - Pencere – Etkileşim (**Activity**)
  - Link – **Intent**

# BACK STACK



*Bu işlemler otomatik olarak gerçekleşir.  
İstenirse sayfa geçişinde istediğimiz sayfayı back stackten silebiliriz.*

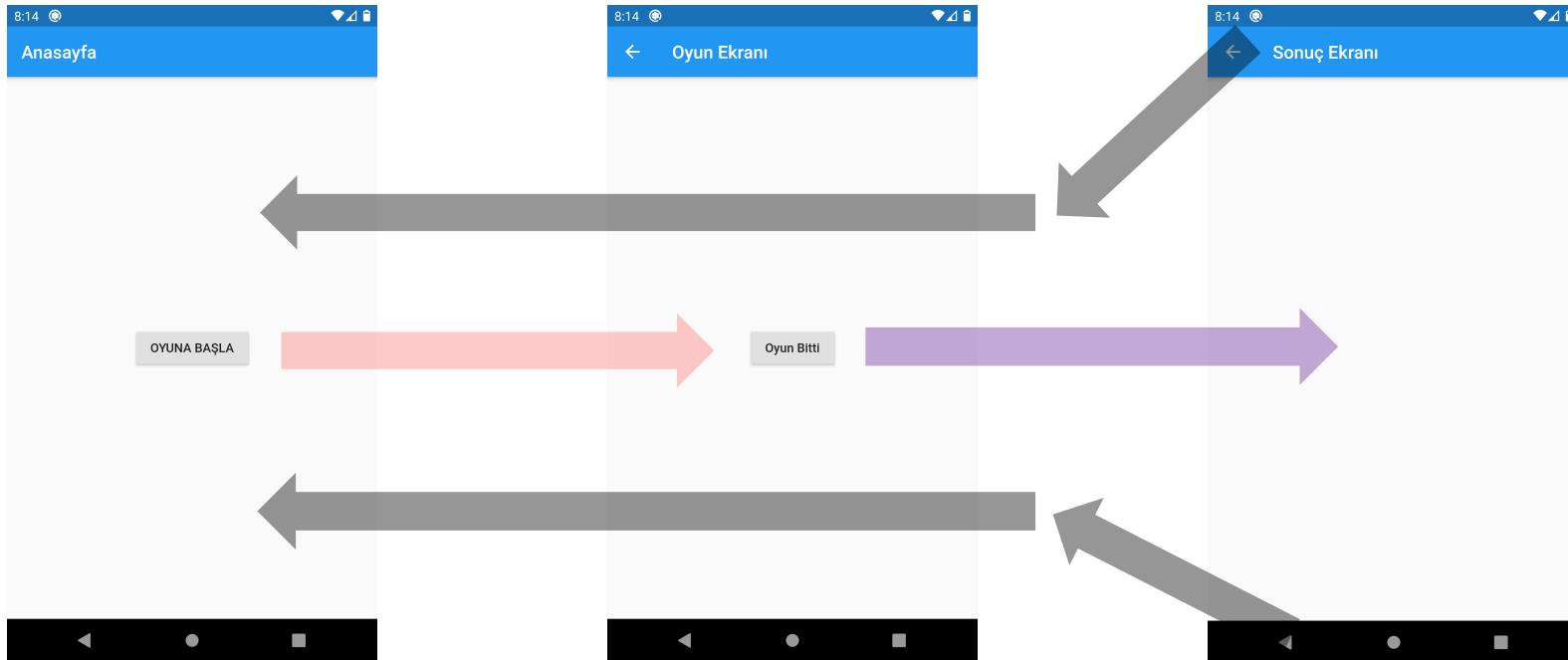
```
Navigator.pushReplacement(context, MaterialPageRoute(builder: (context) => SayfaA() ));  
//Android içindeki Finish metodunu gibi çalışır  
//sayfaya geçiş yapıldığında back stackten kendisini siler
```

# Sayfayı Back Stackten Silme

- Android içindeki Finish metodu gibi çalışır
- Bu yapı ile sayfaya geçiş yapıldığında back stackten kendisini siler.
- Kendisini sildiği için geçiş yapıldığı sayfadan geri tuşu ile bu sayfaya gelinemez çünkü back stackten silinmiştir.

```
ElevatedButton(  
    child : Text("Oyun Bitti"),  
    onPressed: (){  
  
        Navigator.pushReplacement(context,  
            MaterialPageRoute(builder: (context) => SonucEkranı() ));  
  
    },  
, // ElevatedButton
```

# BACK STACK



```
ElevatedButton(  
  child : Text("OYUNA BAŞLA"),  
  onPressed: (){  
  
    Navigator.push(context,  
      MaterialPageRoute(builder: (context) => OyunEkranı() ));  
  
  },  
, // ElevatedButton
```

```
ElevatedButton(  
  child : Text("Oyun Bitti"),  
  onPressed: (){  
  
    Navigator.pushReplacement(context,  
      MaterialPageRoute(builder: (context) => SonucEkranı() ));  
  
  },  
, // ElevatedButton
```

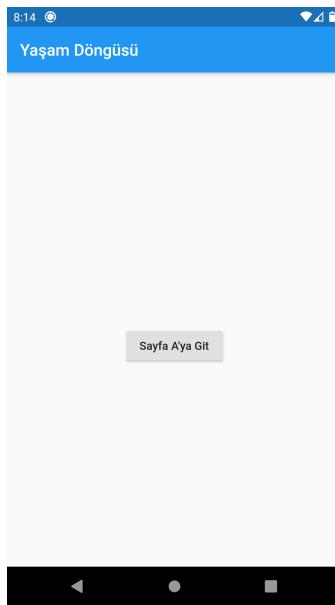
# Yaşam Döngüsü

# Yaşam Döngüsü

- *Önemli :* Yaşam döngüsü StatefulWidget üzerinde gerçekleşir.
- *initState()* : Uygulama sayfası ilk açıldığında çalışır.  
Android için `onCreate()` IOS için `viewDidLoad()` metoduna eş değerdir.
- *build()* : `setState()` metodu ile arayüzde bir değişiklik olduğunda build metodunu tetikler ve her tetiklenmede çalışır.
- *didChangeAppLifecycleState()* : Uygulama çalışmasını takip ederiz.
  - `AppLifecycleState.inactive` : Arkaplana geçiş yapıldığında çalışır.Sadece ios için
  - `AppLifecycleState.paused` : Arkaplana geçiş yapıldığında çalışır.
  - `AppLifecycleState.resume` : Arkaplandan çağrıldığında çalışır.
  - `AppLifecycleState.detached` : Geri tuşuyla arkaplandan kaldırıldığında.Sadece android için
- *deactivate() ve dispose()* : Sayfa Back Stackden silindiğinde çalışır.
  - Örn : Bulunduğu sayfaya geçiş yapıldıktan sonra geldiği sayfaya geri döndüğünde bulunduğu sayfa Back Stackten silinir.Geri tuşuna duyarlıdır.

# Yaşam Döngüsü Davranışları

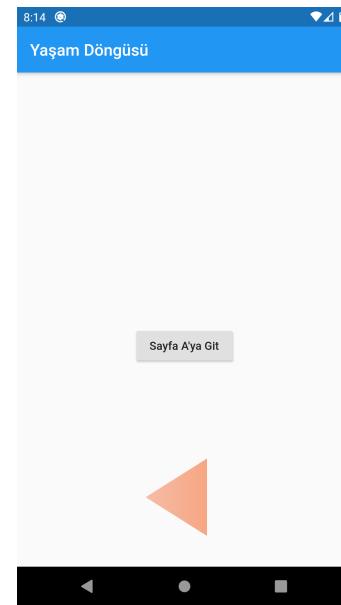
İlk Çalıştığında



1. *initState()* metodu
2. *build()* metodu

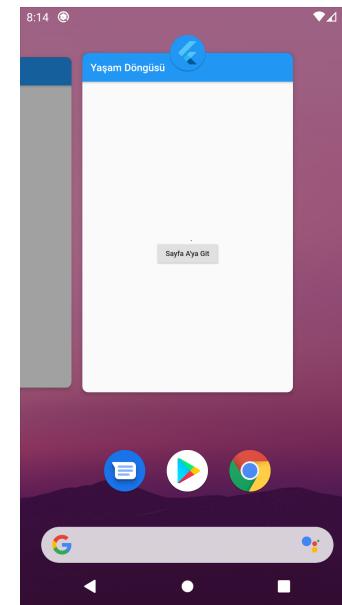
Not : *build()* metodu *setState()* metodu çalıştırıldığında çalışır.

Back Tuşuna Basıldığından



1. *inactive()* metodu
2. *paused()* metodu
3. *detached()* metodu

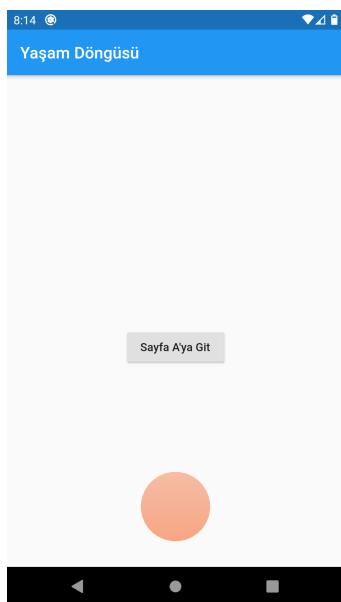
Arka Planдан Tekrar Çağrıldığında



1. *initState()* metodu
2. *build()* metodu

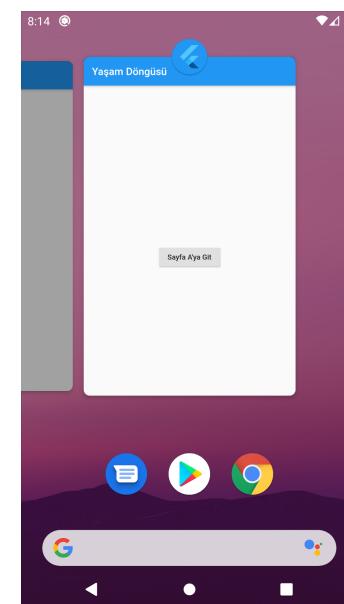
# Yaşam Döngüsü Davranışları

*Menü Tuşuna Basıldığından*



1. *inactive()* metodu
2. *paused()* metodu

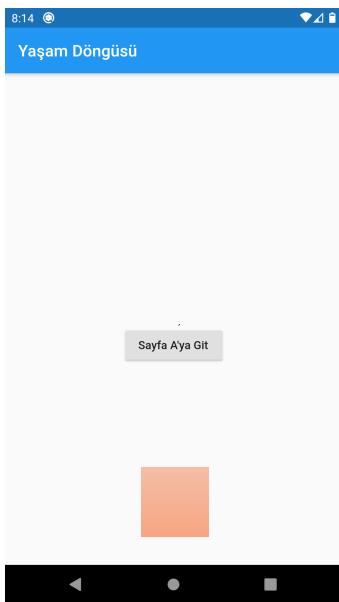
*Arka Planдан Tekrar Çağrıldığında*



1. *resumed()* metodu

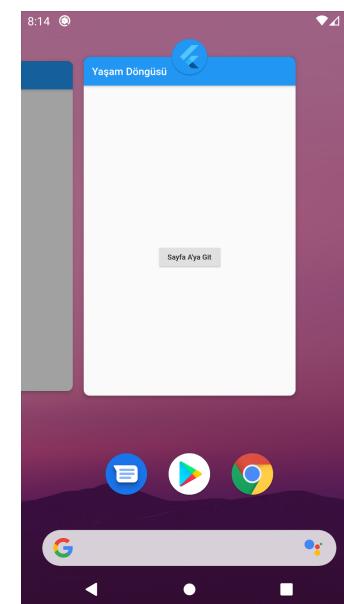
# Yaşam Döngüsü Davranışları

*Arkaplan Tuşuna Basıldığından*



1. *inactive() metodu*
2. *paused() metodu*

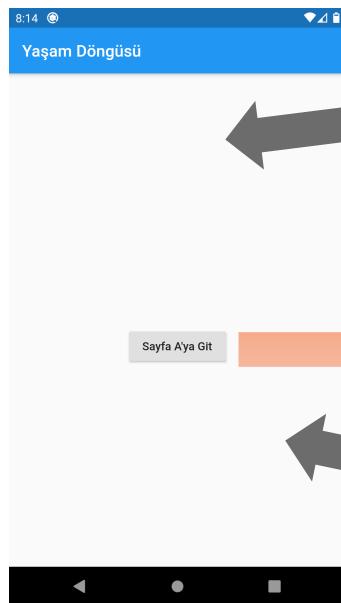
*Arka Plandan Tekrar Çağrıldığında*



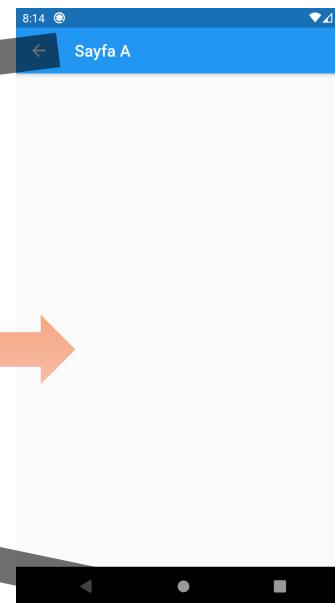
1. *resumed() metodu*

# Yaşam Döngüsü Davranışları

*Başka Sayfaya Geçiş olduğunda*



*Geri Tuşu ile Geldiği Sayfaya  
Döndüğünde*



1. *Sayfa A : deactivate()*
2. *Sayfa A : dispose()*

# Final Kod :

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ), // ThemeData
      home: Anasayfa(),
    ); // MaterialApp
  }
}
```

```
class _AnasayfaState extends State<Anasayfa> with WidgetsBindingObserver {

  @override
  void initState() {
    super.initState();
    print("initState() çalıştı");
    WidgetsBinding.instance!.addObserver(this);
  }

  @override
  void dispose() {
    super.dispose();
    WidgetsBinding.instance!.removeObserver(this);
  }

  @override
  void didChangeAppLifecycleState(AppLifecycleState state) {
    if(state == AppLifecycleState.inactive){print("inactive() çalıştı");}
    if(state == AppLifecycleState.paused){print("paused() çalıştı");}
    if(state == AppLifecycleState.resumed){print("resumed() çalıştı");}
    if(state == AppLifecycleState.detached){print("detached() çalıştı");}
  }

  @override
  Widget build(BuildContext context) {
    print("build() çalıştı");
  }
}
```

```
@override
Widget build(BuildContext context) {
    print("build() çalıştı");
    return Scaffold(
        appBar: AppBar(
            title: Text("Yaşam Döngüsü"),
        ), // AppBar
        body: Center(
            child: Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                    RaisedButton(
                        child: Text("Tıkla"),
                        onPressed: (){
                            Navigator.push(context, MaterialPageRoute(builder: (context) => SayfaA()));
                        },
                    ), // RaisedButton
                ],
            ), // Column
        ), // Center
    ); // Scaffold
}
```

```
class SayfaA extends StatefulWidget {
  @override
  _SayfaAState createState() => _SayfaAState();
}

class _SayfaAState extends State<SayfaA> {

  @override
  void deactivate() {
    super.deactivate();
    print("SayfaA : deactivate çalıştı");
  }

  @override
  void dispose() {
    super.dispose();
    print("SayfaA : dispose çalıştı");
  }

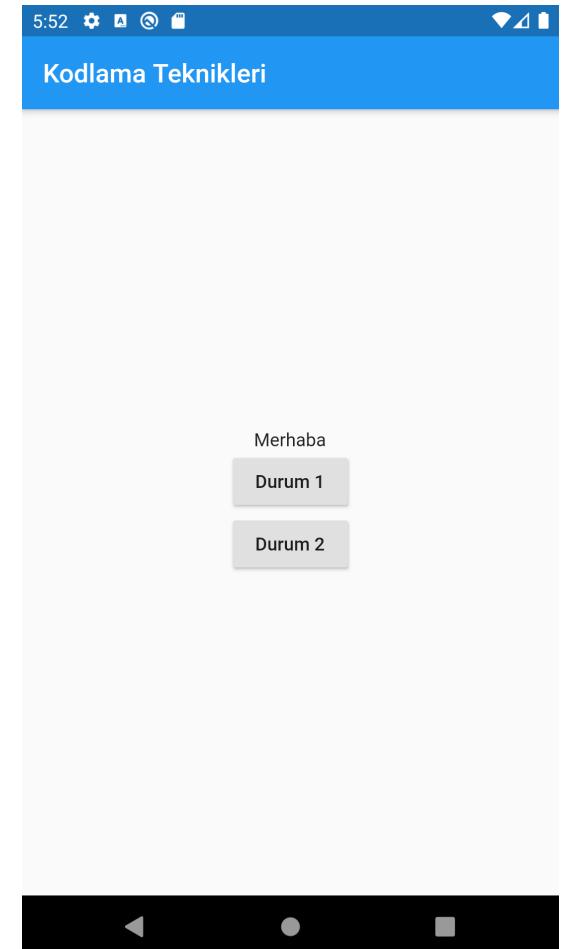
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Sayfa A"),
      ), // AppBar
      body: Center(),
    ); // Scaffold
  }
}
```

# Widget İçinde Kodlama Teknikleri

# Visibility : Görünür – Görünmez Yapma

- Widgetların görünür veya görünmez olmasını sağlar.
- Bu şekilde istenilen widgetların görünür durumunu kontrol edebiliriz.

```
class _AnasayfaState extends State<Anasayfa> {  
  bool kontrol = false;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Kodlama Teknikleri"),  
      ), // AppBar  
      body: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            Visibility(  
              visible: kontrol,  
              child: Text("Merhaba")  
            ), // Visibility  
            ElevatedButton(  
              child: Text("Durum 1"),  
              onPressed: (){  
                setState(() {kontrol = true});  
              },  
            ), // ElevatedButton  
            ElevatedButton(  
              child: Text("Durum 2"),  
              onPressed: (){  
                setState(() {kontrol = false});  
              },  
            ), // ElevatedButton  
          ],  
        ), // Column
```



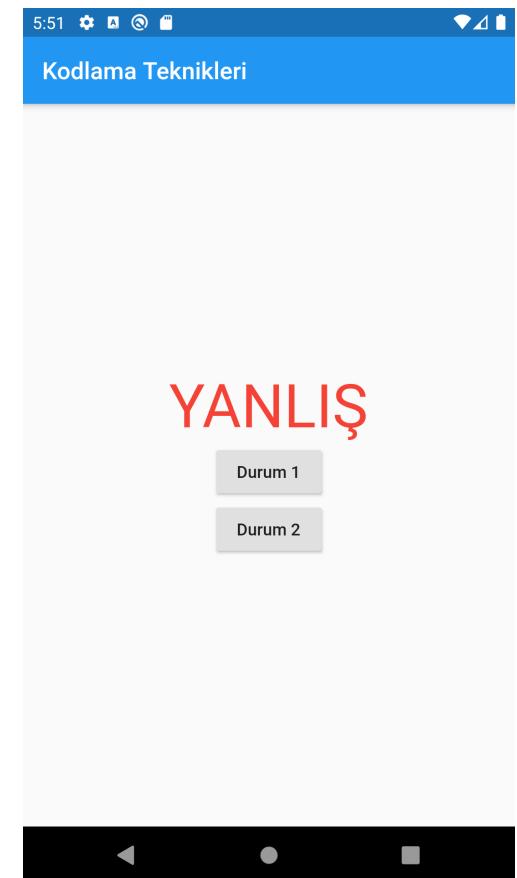
# Ternary Koşul Kullanma

- koşul ? true olma durumu : false olma durumu

- if koşulu yerine widget özelliklerini ternary koşul ile kontrol edebiliriz.

- Örn : Text'in içindeki yazıyı , boyutu vb özellikleri değiştirebiliriz.

```
class _AnasayfaState extends State<Anasayfa> {  
  
  bool kontrol = false;  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar: AppBar(  
        title: Text("Kodlama Teknikleri"),  
      ), // AppBar  
      body: Center(  
        child: Column(  
          mainAxisAlignment: MainAxisAlignment.center,  
          children: [  
            Text( kontrol ? "DOĞRU" : "YANLIŞ",  
                  style: TextStyle(  
                    color: kontrol ? Colors.blue : Colors.red,  
                    fontSize: kontrol ? 20 : 50,  
                    fontWeight: kontrol ? FontWeight.normal : FontWeight.normal,  
                  ), // TextStyle  
            ), // Text  
            ElevatedButton(  
              child: Text("Durum 1"),  
              onPressed: (){  
                setState(() {kontrol = true;});  
              },  
            ), // ElevatedButton  
            ElevatedButton(  
              child: Text("Durum 2"),  
              onPressed: (){  
                setState(() {kontrol = false;});  
              },  
            ), // ElevatedButton  
          ],  
        ),  
      ), // Center  
    );  
  }  
}
```



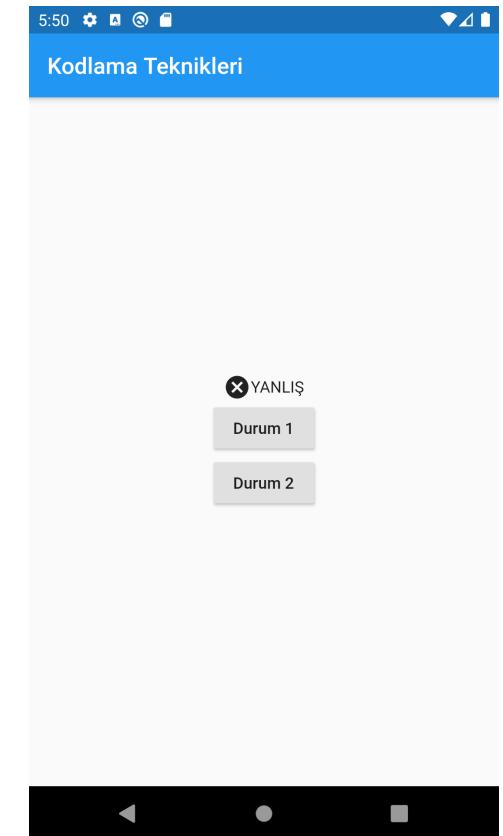
# Kodlama Alanı Açabiliriz

- Detaylı bir kodlama yapmak istiyorsak kullanılır.

```
(() {  
    //Kodlama Yapılacak Alan  
}(),
```

```
ElevatedButton(  
    child: Text("Durum 1"),  
    onPressed: (){  
        setState(() {kontrol = true;});  
    },  
, // ElevatedButton  
ElevatedButton(  
    child: Text("Durum 2"),  
    onPressed: (){  
        setState(() {kontrol = false;});  
    },  
, // ElevatedButton  
), // <Widget>[]  
, // Column  
, // Center  
}
```

```
Widget build(BuildContext context) {  
    return Scaffold(  
        appBar: AppBar(  
            title: Text("Kodlama Teknikleri"),  
        ), // AppBar  
        body: Center(  
            child: Column(  
                mainAxisAlignment: MainAxisAlignment.center,  
                children: [  
                    (){  
                        if(kontrol){  
                            return Row(  
                                mainAxisAlignment: MainAxisAlignment.center,  
                                children: [  
                                    Icon(Icons.done),  
                                    Text("DOĞRU"),  
                                ],  
                            ); // Row  
                        }else{  
                            return Row(  
                                mainAxisAlignment: MainAxisAlignment.center,  
                                children: [  
                                    Icon(Icons.cancel),  
                                    Text("YANLIŞ"),  
                                ],  
                            ); // Row  
                        }  
                    }(),  
                ]  
            )  
        )  
    );  
}
```



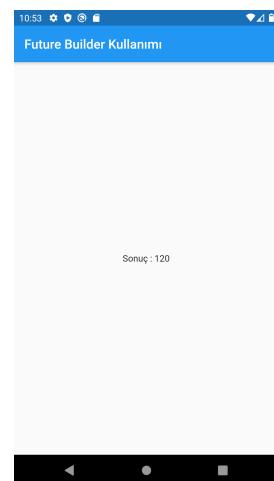
# FutureBuilder

# FutureBuilder

- Asenkron işlemler için kullanılan bir yapıdır.
- async özelliği olan fonksiyonu kullanırken await özelliği ile sadece yapması gereken işlemi bitirene kadar çalışmasını sağlarız.
- Fakat await kullanmak için async özelliği olan fonksiyon içinde olmamız gereklidir.
- async özelliği olan fonksiyonu widget içinde kullanmak istediğimizde async özelliği olması gerekmektedir.Bu özellik widgetlarda yoktur.
- **Widget içinde async özelliğini kullanmak için FutureBuilder yapısı gereklidir.**

# FutureBuilder Kullanımı

```
Future<int> faktoriyelHesapla(int sayi) async{  
  
    var sonuc = 1;  
  
    for(var i=1;i<=sayi;i++){  
        sonuc = sonuc * i ;  
    }  
  
    return sonuc;  
}
```



```
@override  
Widget build(BuildContext context) {  
    return Scaffold(  
        appBar: AppBar(  
            title: Text(widget.title),  
        ), // AppBar  
        body: Center(  
            child: Column(  
                mainAxisAlignment: MainAxisAlignment.center,  
                children: <Widget>[  
                    FutureBuilder<int>(  
                        future: faktoriyelHesapla(5),  
                        builder: (context,snapshot){  
                            if(snapshot.hasError){  
                                print("Hata sonucu : ${snapshot.error}");  
                            }  
  
                            if(snapshot.hasData){  
                                return Text("Sonuç : ${snapshot.data}");  
                            }  
                            else{  
                                return Center(child: Text("Gösterilecek Veri Yok"),);  
                            }  
                        },  
                    ), // FutureBuilder  
                ], // <Widget>[]  
            ), // Column  
        ), // Center  
    ); // Scaffold  
}
```

Annotations on the right side of the code:

- A red arrow points from the text "Çalıştıracağı fonksiyonun geri dönüş türü." to the type annotation `<int>` in the `FutureBuilder` declaration.
- A red arrow points from the text "Çalıştırılacak fonksiyon" to the `future` parameter of the `FutureBuilder`.
- A red arrow points from the text "Fonksiyonun çalışma sonucunu temsil eden değişken." to the `snapshot` parameter of the `builder` function.
- A red arrow points from the text "Fonksiyonun çalışması sonucunda hata oluşuyor mu kontrolü" to the `hasError` check in the `builder` function.
- A red arrow points from the text "Fonksiyonun çalışması sonucunda veri var mı yok mu kontrolü" to the `hasData` check in the `builder` function.
- A red arrow points from the text "Fonksiyonun sonucuna erişim." to the `snapshot.data` reference in the `Text` widget.
- A red arrow points from the text "Eğer gelen veri boş ise yani null ise burası çalışır ve tasarımda istediğimizi gösterebiliriz." to the `Text` widget containing "Gösterilecek Veri Yok".

# Uygulama Çıktısı Oluşturma

# Uygulama Çıktısı Oluşturma

## Key Dosyası Oluşturma

- Mac için :

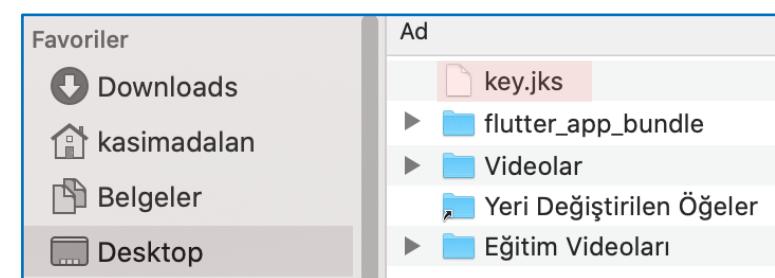
```
keytool -genkey -v -keystore ~/Desktop/key.jks -keyalg RSA -keysize 2048 -  
validity 10000 -alias key
```

- Windows için :

```
keytool -genkey -v -keystore c:/Users/USER_NAME/key.jks -storetype JKS -  
keyalg RSA -keysize 2048 -validity 10000 -alias key
```

# Key Dosyası Oluşturma

```
Terminal: Local +  
  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
Berk-MacBook-Pro: flutter_app_bundle kasimadalans$ keytool -genkey -v -keystore ~/Desktop/key.jks -keyalg RSA -keysize 2048 -validity 10000 -alias key  
Enter keystore password:  
Re-enter new password: Şifre yazarken görünmez  
What is your first and last name?  
[Unknown]: KASIM ADALAN  
What is the name of your organizational unit?  
[Unknown]: Egitim  
What is the name of your organization?  
[Unknown]: Egitim  
What is the name of your City or Locality?  
[Unknown]: Istanbul  
What is the name of your State or Province?  
[Unknown]: TURKEY  
What is the two-letter country code for this unit?  
[Unknown]: TR  
Is CN=KASIM ADALAN, OU=Egitim, O=Egitim, L=Istanbul, ST=TURKEY, C=TR correct?  
[no]: yes  
  
Generating 2.048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10.000 days  
for: CN=KASIM ADALAN, OU=Egitim, O=Egitim, L=Istanbul, ST=TURKEY, C=TR  
[Storing /Users/kasimadalan/Desktop/key.jks] Dosyanın olduğu dosya yolu
```



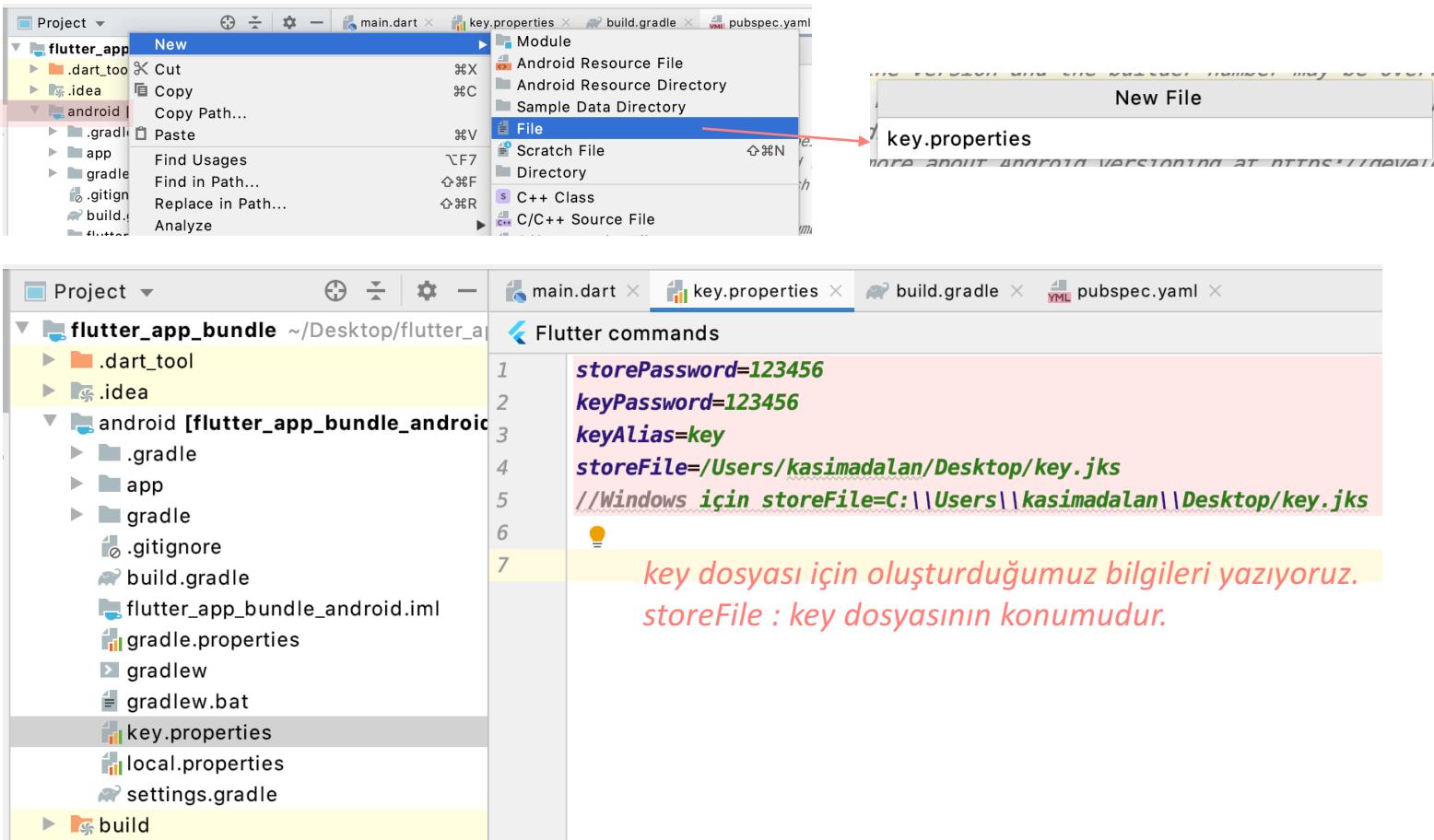
The screenshot shows the Xcode Organizer window with the 'Downloads' tab selected. A file named 'key.jks' is highlighted with a red box. Other files listed include 'flutter\_app\_bundle', 'Videolar', 'Yeri Değiştirilen Öğeler', and 'Eğitim Videoları'. The 'Favoriler' sidebar on the left includes 'Downloads', 'kasimadalan', 'Belgeler', and 'Desktop'.

Logcat Terminal Dart Analysis TODO

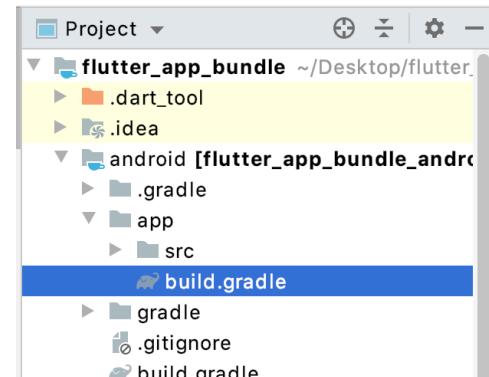
Layout Inspector Events

# key.properties dosyası oluşturma

android  
dosyası seçili  
olmalı



# Gradle dosyasını düzenleme



```
Flutter commands
1 def localProperties = new Properties()
2 def localPropertiesFile = rootProject.file('local.properties')
3 if (localPropertiesFile.exists()) {
4     localPropertiesFile.withReader('UTF-8') { reader ->
5         localProperties.load(reader)
6     }
7 }

//İlk Kod
8
9 def keystoreProperties = new Properties()
10 def keystorePropertiesFile = rootProject.file('key.properties')
11 if (keystorePropertiesFile.exists()) {
12     keystoreProperties.load(new FileInputStream(keystorePropertiesFile))
13 }
14
```

```
Flutter commands
52     versionName flutterVersionName
53
54
55     //İkinci Kod
56     signingConfigs {
57         release {
58             keyAlias keystoreProperties['keyAlias']
59             keyPassword keystoreProperties['keyPassword']
60             storeFile keystoreProperties['storeFile'] ? file(keystoreProperties['storeFile']) : null
61             storePassword keystoreProperties['storePassword']
62         }
63     }
64
65     buildTypes {
66         release {
67             // TODO: Add your own signing config for the release build.
68             // Signing with the debug keys for now, so 'flutter run --release' works.
69             //Üçüncü Kod
70             signingConfig signingConfigs.release
71         }
72     }
73 }
```

# Gradle dosyasını düzenleme

## *Birinci Kodlama*

```
def keystoreProperties = new Properties()
def keystorePropertiesFile =
rootProject.file('key.properties')
if (keystorePropertiesFile.exists()) {
keystoreProperties.load(new
FileInputStream(keystorePropertiesFile))
}
```

## *İkinci Kodlama*

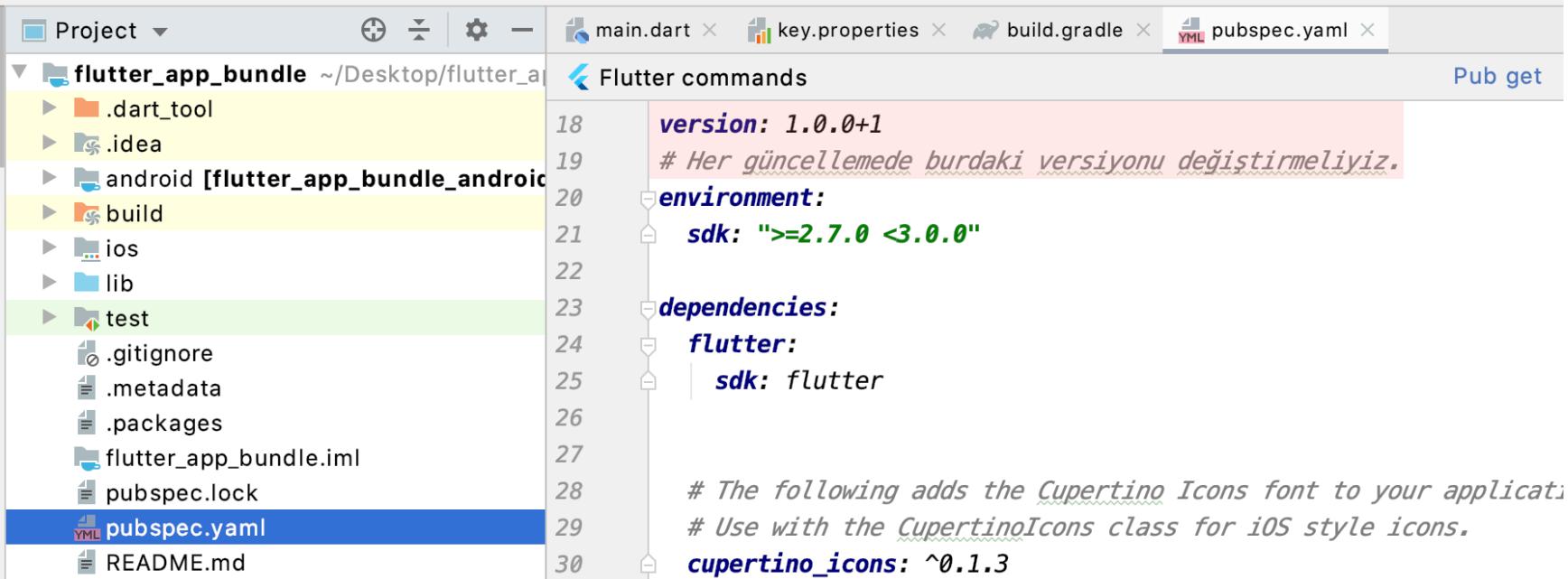
```
signingConfigs {
release {
keyAlias keystoreProperties['keyAlias']
keyPassword keystoreProperties['keyPassword']
storeFile keystoreProperties['storeFile'] ?
file(keystoreProperties['storeFile']) : null
storePassword keystoreProperties['storePassword']
}
}
```

## *Üçüncü Kodlama*

```
signingConfig signingConfigs.release
```

# Version Değiştirme

- Eğer daha önce aab dosyası oluşturduysanız versiyonu değiştirmelisiniz.



```
version: 1.0.0+1
# Her güncellemede burdaki versiyonu değiştirmeliyiz.
environment:
  sdk: ">=2.7.0 <3.0.0"
dependencies:
  flutter:
    sdk: flutter
  # The following adds the Cupertino Icons font to your application.
  # Use with the CupertinoIcons class for iOS style icons.
  cupertino_icons: ^0.1.3
```

# App Bundle Dosyasını Oluşturma

- Terminal ekranına `flutter build appbundle` yazmamız gereklidir.

The screenshot shows a Mac OS X desktop environment. On the left, a terminal window titled "Terminal: Local" displays the command "flutter build appbundle" being run. The output shows the process of building an app bundle, including resource optimization and Gradle task execution. A green checkmark indicates the successful creation of the "app-release.aab" file. On the right, a file browser window titled "Project" shows the directory structure of the Flutter project. The "build" folder is expanded, revealing subfolders like "app", "generated", "intermediates", "kotlin", "outputs", "logs", "mapping", "tmp", and "kotlin". The "outputs" folder contains the "bundle" and "release" subfolders, with the "app-release.aab" file highlighted in red, indicating it is the result of the build command. A red arrow points from the terminal's success message to this highlighted file in the file browser.

```
Terminal: Local +  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
Berk-MacBook-Pro:flutter_app_bundle kasimadalan$ flutter build appbundle  
Running "flutter pub get" in flutter_app_bundle... 0,6s  
Removed unused resources: Binary resource data reduced from 36KB to 21KB: Removed 39%  
Running Gradle task 'bundleRelease'...  
Running Gradle task 'bundleRelease'... Done 107,1s  
✓ Built build/app/outputs/bundle/release/app-release.aab (15.2MB).  
Berk-MacBook-Pro:flutter_app_bundle kasimadalan$
```

*NOT : flutter no command hatası  
aliyorsanız.Kurulum derslerimizde gösterdiğimiz  
flutter kurulumunu tekrar yapmalısınız.*

# Güncelleme için aab dosyası oluşturma

- Uygulamayı güncellemek için yeni aab dosyası oluşturabiliriz.
  - Burada önemli olan key dosyamızın olmasıdır.
  - Bu dosyayı kullanarak işlem yapmalıyız.

## 1. key.properties dosyasını kontrol et

```
storePassword=123456
keyPassword=123456
keyAlias=key
storeFile=/Users/kasimadalan/Desktop/key.jks
//Windows için storeFile=C:\\Users\\kasimadalan\\Desktop\\key.jks
```

key dosyası oluşturduğumuz bilgileri yazıyoruz.  
storeFile : key dosyasının konumudur.  
keyAlias : key dosyasının adı

## 2. Versiyon değiştir

```
# https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/availableKeys.html#//apple_ref/doc/uid/TP40009251-CH21-SW1
version: 2.0.0
# Her güncellemede burdaki versiyonu değiştirmeliyiz.
environment:
  sdk: ">=2.7.0 <3.0.0"
```

## 3. Yenide aab dosyası oluştur.

```
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
Berk-MacBook-Pro:flutter_app_bundle kasimadalan$ flutter build appbundle
Running "flutter pub get" in flutter_app_bundle...                                0,6s
Removed unused resources: Binary resource data reduced from 36KB to 21KB: Removed 39%
Running Gradle task 'bundleRelease'...
Running Gradle task 'bundleRelease'... Done                                    107,1s
✓ Built build/app/outputs/bundle/release/app-release.aab (15.2MB).
Berk-MacBook-Pro:flutter_app_bundle kasimadalan$
```

Uygulamayı Google Play  
Üzerinde Yayınlama

# Uygulamanın Google Play Yüklenmesi



# Uygulamanın Google Play Yüklenmesi

The screenshot shows the Google Play Console interface. On the left, there's a sidebar with various navigation options like 'Tüm uygulamalar', 'Hesap özet'i, and 'Mağaza giriş'i. The main area is titled 'Mağaza giriş'i and shows 'Deneme Taslak' as the product. It has tabs for 'TÜRKÇE - TR-TR' and 'Çevirileri yönet'. A text input field for 'Tam açıklama' is present, with 'Türkçe - tr-TR' selected. Below it, there's a note in red: 'Google Play Console sayfasından uygulama paylaşmak için her yıl 25 dolar verilmelidir.' At the bottom right, there's a button labeled 'TASLAĞI KAYDET'.

Google Play Console sayfasından uygulama paylaşmak için her yıl 25 dolar verilmelidir.

# Mağaza için Görsellerin Hazırlanması

Ürün bilgileri TÜRKÇE – TR-TR Çeviri

2/8 ekran görüntüsü DOSYALARA

<b>Yüksek çözünürlüklü simge *</b> Varsayılan – Türkçe – tr-TR <b>512 x 512</b> <b>32 bit PNG (alfa kanallı)</b>	<b>Öne Çıkan Grafik *</b> Varsayılan – Türkçe – tr-TR <b>1024 g x 500 y</b> JPG veya 24 bit PNG (alfa kanalsız)	<b>Tanıtım Grafiği</b> Varsayılan – Türkçe – tr-TR <b>180 g x 120 y</b> JPG veya 24 bit PNG (alfa kanalsız)
		
<b>TV Banner'i</b> Varsayılan – Türkçe – tr-TR <b>1280 g x 720 y</b> JPG veya 24 bit PNG (alfa kanalsız)	<b>Daydream 360 derecelik stereoskopik resim</b> Varsayılan – Türkçe – tr-TR <b>4096 g x 4096 y</b> JPG veya 24 bit PNG (alfa kanalsız)	

# İmzalı APK'nın yüklenmesi

The screenshot shows the Google Play Developer Console interface. On the left, there's a sidebar with various options: Hesap özet (Account summary), Uygulama sürümleri (Application versions) which is selected and highlighted with an orange border, Android Hazır Uygulamalar, Yapı kitaplığı, Cihaz kataloğu, Uygulama imzalama, Mağaza giriş, İçerik derecelendirme, Fiyatlandırma ve dağıtım, and Uygulama içi ürünler. The main area is titled '← Yeni üretim sürümü' (New production version). It shows two steps: '1 Sürümü hazırlayın' (Prepare the version) and '2 İnceleyin ve kullanıma sunun' (Review and publish). Below step 1, it says 'Google Play'den uygulama imzalama özelliği' (Application signing feature from Google Play) is active ('Etkin'). Under step 2, there's a section for 'Eklenecek Android Uygulama Paketleri ve APK'lar' (Android application packages and APKs to be added) with a 'KİTAPLIKTAN EKLE' (Add from library) button. A note says 'Bu uygulama paketleri ve APK'lar bu sürümün kullanıma sunulmasından sonra Google Play Store'da sunulacak.' (These application packages and APKs will be published on the Google Play Store after this version is made available.) At the bottom, there's a large input field for file uploads with the placeholder 'Uygulama paketlerinizi ve APK'larınızı buraya bırakın veya bir dosya seçin.' (Drop your application packages or select a file) and a 'DOSYALARA GÖZ AT' (View files) button.

Teşekkürler...



kasim-adalan



kasimadalan@gmail.com



kasimadalan