**FLUTTER’DA WIDGET YAPISI**

**2.1. Widget Nedir?**

* **Tanım:** Flutter’da her şey bir widget’dır. UI elemanları (görsel bileşenler, düzenler, etkileşimli öğeler) widget’lar olarak tanımlanır.
* **Örnek Widget’lar:** Text, Container, Row, Column, Icon vs.
* **Özellikler:**
  + **Yeniden Kullanılabilirlik:** Tekrarlanan UI bileşenleri kolayca yeniden kullanılabilir.
  + **Kompozisyon:** Küçük widget’lar, daha karmaşık widget’lar oluşturmak için bir araya getirilir.

**2.2. Stateless vs. Stateful Widget’lar**

* **Stateless Widget:**
  + Durum (state) yönetimi olmayan, sadece gelen verilere göre render yapan widget’lar.
  + Örnek: Sabit metin gösteren bir Text widget’ı.
* **Stateful Widget:**
  + Duruma (state) bağlı olarak değişebilen widget’lar.
  + Örnek: Kullanıcının butona tıklamasıyla sayının artması gibi dinamik işlemler.

**Örnek Kod: Stateless Widget**

import 'package:flutter/material.dart';

class MyStatelessWidget extends StatelessWidget {

final String message;

MyStatelessWidget({required this.message});

@override

Widget build(BuildContext context) {

return Text(message);

}

}

**Örnek Kod: Stateful Widget**

import 'package:flutter/material.dart';

class MyStatefulWidget extends StatefulWidget {

@override

\_MyStatefulWidgetState createState() => \_MyStatefulWidgetState();

}

class \_MyStatefulWidgetState extends State<MyStatefulWidget> {

int counter = 0;

void increment() {

setState(() {

counter++;

});

}

@override

Widget build(BuildContext context) {

return Column(

children: [

Text('Counter: $counter'),

ElevatedButton(

onPressed: increment,

child: Text('Artır'),

),

],

);

}

}

**2.3. Widget Ağacı (Tree) Yapısı**

* **Ağacın Yapısı:** Tüm widget’lar, hiyerarşik (ağaç yapısında) düzenlenir. Ana widget (root widget) altındaki tüm widget’lar bu ağacın dallarıdır.
* **Önem:** UI’nin yönetilebilir ve modüler olmasını sağlar. Öğrenciler, widget ağacını çizerken uygulamanın mimarisini daha iyi kavrayacaklar.

**3. Scaffold Widget’ına Giriş**

**3.1. Scaffold Nedir?**

* **Tanım:** Scaffold widget, Material Design uygulamalarında temel yapıyı oluşturur. Uygulamanın genel düzenini (appBar, body, floatingActionButton, drawer, bottomNavigationBar vb.) yönetir.
* **Kullanım Alanları:**
  + Uygulama başlığı (AppBar)
  + Ana içerik alanı (Body)
  + Yüzen eylem butonu (FloatingActionButton)
  + Yan menü (Drawer)
  + Alt gezinme çubuğu (BottomNavigationBar)

**3.2. Scaffold’un Özellikleri**

* **AppBar:** Uygulama başlığını ve üst araç çubuğunu oluşturur.
* **Body:** Uygulamanın ana içerik alanı.
* **FloatingActionButton:** Genellikle kullanıcı etkileşimlerini tetiklemek için kullanılan yüzen buton.
* **Diğer Bileşenler:** Drawer, bottomNavigationBar, snackBar gibi ek özellikler.

**4. Scaffold Widget Kullanımı: Detaylı Örnekler**

**4.1. Örnek 1: Basit Scaffold Uygulaması**

Aşağıdaki örnek, temel bir Scaffold kullanarak basit bir uygulama iskeletini göstermektedir.

import 'package:flutter/material.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Flutter Scaffold Örneği',

theme: ThemeData(

primarySwatch: Colors.blue,

),

home: MyHomePage(),

);

}

}

class MyHomePage extends StatefulWidget {

@override

\_MyHomePageState createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

int \_counter = 0;

void \_incrementCounter() {

setState(() {

\_counter++;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Flutter Scaffold Örneği'),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: <Widget>[

Text('Butona tıklama sayısı:'),

Text(

'$\_counter',

style: Theme.of(context).textTheme.headline4,

),

],

),

),

floatingActionButton: FloatingActionButton(

onPressed: \_incrementCounter,

tooltip: 'Artır',

child: Icon(Icons.add),

),

);

}

}

**Kod Analizi:**

* **MaterialApp:** Uygulamanın temel konfigürasyonunu sağlar.
* **Scaffold:** Uygulamanın temel iskeletini oluşturur.
  + **appBar:** Uygulamanın üst kısmında bulunan başlık çubuğu.
  + **body:** Merkezi yerleştirilmiş bir Column widget’ı ile sayacın gösterimi.
  + **floatingActionButton:** Butona her basıldığında sayaç artar.

**4.2. Örnek 2: Gelişmiş Scaffold Kullanımı**

Bu örnekte, Scaffold’un ek özelliklerini (Drawer, BottomNavigationBar) içeren daha zengin bir uygulama yapısını inceleyeceğiz.

import 'package:flutter/material.dart';

void main() {

runApp(MyAdvancedApp());

}

class MyAdvancedApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Gelişmiş Scaffold Örneği',

home: AdvancedHomePage(),

);

}

}

class AdvancedHomePage extends StatefulWidget {

@override

\_AdvancedHomePageState createState() => \_AdvancedHomePageState();

}

class \_AdvancedHomePageState extends State<AdvancedHomePage> {

int \_selectedIndex = 0;

// Drawer içerisindeki menü seçenekleri için örnek liste

final List<String> \_drawerOptions = ['Ana Sayfa', 'Ayarlar', 'Hakkında'];

void \_onItemTapped(int index) {

setState(() {

\_selectedIndex = index;

});

}

Widget \_buildBody() {

return Center(

child: Text(

'Seçili Sayfa: $\_selectedIndex',

style: TextStyle(fontSize: 24),

),

);

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text('Gelişmiş Scaffold Örneği'),

),

drawer: Drawer(

child: ListView(

padding: EdgeInsets.zero,

children: [

DrawerHeader(

decoration: BoxDecoration(color: Colors.blue),

child: Text(

'Menü',

style: TextStyle(color: Colors.white, fontSize: 24),

),

),

...\_drawerOptions.map((option) => ListTile(

title: Text(option),

onTap: () {

Navigator.pop(context); // Drawer’ı kapatır

},

)),

],

),

),

body: \_buildBody(),

bottomNavigationBar: BottomNavigationBar(

currentIndex: \_selectedIndex,

onTap: \_onItemTapped,

items: const <BottomNavigationBarItem>[

BottomNavigationBarItem(

icon: Icon(Icons.home),

label: 'Ana Sayfa',

),

BottomNavigationBarItem(

icon: Icon(Icons.business),

label: 'İş',

),

BottomNavigationBarItem(

icon: Icon(Icons.school),

label: 'Okul',

),

],

),

);

}

}

**Kod Analizi:**

* **Drawer:** Uygulama menüsünü yan tarafta açılan panel şeklinde sunar.
* **BottomNavigationBar:** Alt kısımda gezinme çubuğu oluşturarak, kullanıcıya farklı ekranlara geçiş imkanı verir.
* **Dinamik İçerik:** Seçilen index’e göre body içerisindeki içerik güncellenir.

**5. Derste Ele Alınacak Ek Konular**

**5.1. Widget Ağacını İnceleme**

* **Ağaç Yapısının Çizilmesi:** Öğrenciler, uygulamanın widget ağacını çizerek her bir widget’ın yerini anlamalı.
* **Debugging Araçları:** Flutter’ın sunduğu “Flutter Inspector” ile widget ağacının incelenmesi.

**5.2. Durum Yönetimi Temelleri**

* **Stateful Widget’larda setState():** Nasıl çalıştığı, performans ipuçları.
* **Küçük Ölçekli Durum Yönetimi:** Basit uygulamalarda state yönetiminin doğru kullanımı.

**5.3. Material Design İlkeleri**

* **Tasarım Standartları:** Scaffold’un sunduğu tasarım standartları ve Material Design kılavuzları.
* **Tema ve Stil Özelleştirmeleri:** Uygulamanın görünümünün Material Theme kullanılarak nasıl değiştirileceği.

**6. Uygulama Çalışması ve Ödevler**

* **Ödev 1:** Kendi Scaffold tabanlı uygulamanızı oluşturun. Uygulamada AppBar, Drawer, FloatingActionButton ve BottomNavigationBar kullanarak basit bir içerik yönetimi yapısı tasarlayın.
* **Ödev 2:** Stateless ve Stateful widget’ları içeren, etkileşimli bir örnek hazırlayın. Örneğin; bir sayaç uygulaması veya liste elemanlarına tıklanınca değişen içerikler.
* **Grup Çalışması:** Öğrencilerden, widget ağacını çizip açıklamalar yapmaları istenebilir. Her bir widget’ın işlevi ve hiyerarşi üzerindeki yeri tartışılabilir.

**7. Dersin Sonuçlandırılması**

* **Özet:**
  + Flutter’da her şeyin widget olduğu ve widget’ların birbirleriyle nasıl ilişkilendiği.
  + Stateless ile stateful widget farklarının ve kullanım senaryolarının anlaşılması.
  + Scaffold widget’ın Material Design uygulamalarında temel yapı taşı olarak işlev görmesi.
* **Öneriler:**
  + Uygulama geliştirirken widget hiyerarşisine dikkat etmek, okunabilir ve sürdürülebilir kod yazmak.
  + Flutter Inspector gibi araçlarla widget ağacını analiz etmek.

**8. Ek Kaynaklar**

* [Flutter Resmi Dokümantasyonu](https://flutter.dev/docs)
* [Flutter YouTube Kanalı](https://www.youtube.com/c/flutterdev)
* Çeşitli blog yazıları ve örnek projeler (örneğin Medium’daki Flutter rehberleri)