

FLUTTER

Future ve Asenkron Programlama

25. hafta • İnternet ve Uzak Veri

36 hafta, 72 bağımsız çalışan Flutter projesi, gerçek uygulama ekranları, uygulamalı dersler ve öğrenci geri bildiriminden sonra incelenebilen çözüm kaynakları.

Bu kitap güncel ders kaynaklarından otomatik üretilmiştir; özet ya da jenerik şablon değildir.

Öğrenme Haritası

Hafta	Ders	Projeler
25	Future ve Asenkron Programlama	Gecikmeli Mesaj • Ders Hatırlatıcısı

25. Hafta — Future ve Asenkron Programlama

Bu hafta ne yapacağız?

Dosya, veritabanı ve internet işlemlerinin sonucu hemen gelmez. Uygulama sonucu beklerken ekranın donmaması gerekir. Bu hafta Future, async, await, try/catch, yükleniyor durumu ve iptal edilen eski isteğin sonucunu yok sayma yöntemini iki çalışan uygulamayla öğreneceğiz.

Kazanımlar

- Eşzamanlı ve asenkron çalışma arasındaki farkı açıklama
- Değer döndüren Future<T> metodu yazma
- async metotta await kullanma
- Hazır, yükleniyor, başarılı ve hata durumlarını modelleme
- try/catch ile asenkron hatayı kullanıcı mesajına dönüştürme
- Widget kaldırıldıktan sonra setState çağrısını mounted ile engelleme
- İptal edilen eski Future sonucunun yeni ekran durumunu bozmasını önleme

Çözümlü uygulama — Gecikmeli Mesaj

Future döndüren servis

```
class MesajServisi {
  Future<String> getir({required bool hata}) async {
    await Future<void>.delayed(const Duration(seconds: 2));

    if (hata) {
      throw Exception('Sunucu yanıt vermedi');
    }

    return 'Merhaba 11/B! Flutter laboratuvarı saat 13.30'da başlayacak.';
  }
}
```

Future.delayed, gerçek ağ gecikmesini kontrollü biçimde temsil eder. Bu hafta zaman ve durum yönetimine odaklanıyoruz; sonraki hafta bu servis gerçek HTTP isteğiyle değiştirilecektir.

Durumları açıkça tanımlama

```
enum IstekDurumu {
  hazir,
  yukleniyor,
  basarili,
  hata,
}
```

Tek bir bool hem hata hem başarı hem de ilk açılış durumunu açıklayamaz. Enum, arayüzün hangi bileşeni göstereceğini açık hâle getirir.

Sonucu bekleme ve hatayı yakalama

```
Future<void> getir() async {
  setState(() => durum = IstekDurumu.yukleniyor);

  try {
    final sonuc = await servis.getir(hata: hataUret);
    if (!mounted) return;
    setState(() {
      mesaj = sonuc;
      durum = IstekDurumu.basariili;
    });
  } catch (e) {
    if (!mounted) return;
    setState(() {
      mesaj = 'Mesaj alınamadı. Bağlantıyı kontrol edip yeniden deneyin.';
      durum = IstekDurumu.hata;
    });
  }
}
```

await ana arayüz iş parçacığını kilitlemez. Bu metot beklerken Flutter yükleniyor göstergesini çizmeye ve kullanıcı etkileşimlerini işlemeye devam eder.

Eski sonucu geçersiz kılma

```
final benimIstek = ++istekNo;
final sonuc = await servis.getir(hata: hataUret);

if (!mounted || benimIstek != istekNo) return;
```

Kullanıcı isteği iptal ederse istekNo artırılır. Önceki Future daha sonra tamamlansa bile numarası eşleşmediği için ekranı değiştiremez.

Gerçek uygulama ekranı

Test önce hata senaryosunu çalıştırıp kullanıcı mesajını, ardından başarılı senaryoyu çalıştırıp gelen duyuruyu doğrulamıştır.

Future ile Mesaj Alma



Merhaba 11/B! Flutter laboratuvarı saat 13.30'da başlayacak.

Hata senaryosunu dene

Future tamamlandığında exception üretir

☒

 Mesajı getir

Future hemen sonucu vermez. await çalışmayı dondurmaz; yalnızca bu async metodun devamını sonuç gelene kadar bekletir.

Gecikmeli Mesaj gerçek Flutter ekranı

Özgün uygulama — Asenkron Ders Hatırlatıcısı

Kullanıcı konu ve bekleme süresi seçer. Sayaç Future ile tamamlanınca hatırlatma görünür.

```

Future<void> planla() async {
  final metin = konu.text.trim();
  if (metin.isEmpty) return;

  final benimGorevim = ++gorevNo;
  setState(() {
    bekliyor = true;
    durum = '$metin için $saniye saniyelik sayaç başladı.';
  });

  await Future<void>.delayed(Duration(seconds: saniye));

  if (!mounted || benimGorevim != gorevNo) return;
  setState(() {
    bekliyor = false;
    durum = 'Çalışma zamanı: $metin';
  });
}

```

İptal davranışı

```

void iptal() {
  gorevNo++;
  setState(() {
    bekliyor = false;
    durum = 'Hatırlatıcı iptal edildi.';
  });
}

```

Dart `Future.delayed` işlemini fiziksel olarak durdurmaz; fakat görev numarası değiştiği için tamamlandığında sonucu uygulanmaz.

Asenkron Ders Hatırlatıcısı

Çalışılacak konu



Future ve async/await

Bekleme süresi: 3 saniye



Çalışma zamanı: Future ve async/await



Hatırlatıcıyı başlat

Ders Hatırlatıcısı gerçek Flutter ekranı

Sık yapılan hatalar

- Future döndüren metodu await etmeden sonucu kullanmaya çalışmak
- Yükleme sırasında aynı düğmeye tekrar tekrar basılmasına izin vermek
- catch olmadan hata durumunda arayüzü sonsuza kadar yükleniyor bırakmak
- Asenkron işlemten sonra mounted kontrolü yapmamak
- Eski ve yeni isteklerin hangi sırada tamamlandığını önemsememek
- Teknik exception metnini doğrudan öğrenciye göstermek

Uygulama görevleri

Kolay — Gecikmeli selamlama

Adı aldıktan bir saniye sonra kişisel mesaj gösterin. Beklerken düğmeyi kapatın.

Orta — Deneme sınavı sayacı

Konu ve süre seçtin. Başlatma, iptal ve yeniden başlatma davranışlarını görev numarasıyla yönetin.

İleri — Üç aşamalı proje hazırlığı

“Dosya hazırlanıyor”, “kontrol ediliyor” ve “yükleniyor” adımlarını sıralı Future'larla çalıştırın. Her aşamanın ilerlemesini gösterin; ikinci aşamada oluşabilecek hatayı yakalayıp yeniden deneme ekleyin.

Test ve kontrol

- Future tamamlanmadan yükleniyor görünmeli.
- Başarıda veri, hatada anlaşılır mesaj gösterilmeli.
- İptal edilen Future daha sonra ekranı değiştirmemeli.
- İşlem sırasında tekrar başlatma düğmesi devre dışı olmalı.
- Widget kapandıktan sonra istisna oluşmamalı.
- Testte sanal zaman ilerletilerek gecikme gerçekten doğrulanmalı.

Geri bildirim ve çözüm erişimi

Öğrenci asenkron akışını gönderdikten sonra İpucunu göster Future zinciri ve durum geçişleri hakkında yön verir. Örnek çözümü göster gerçek servis ve widget kodunu; indirme bağlantısı ise çalışan projeyi çevrim dışı kullanım için sunar.

Kaynak projeler

- Konu uygulaması: 07 - ■nternet ve Uzak Veri/Uygulamalar/Uygulama-25-Gecikmeli Mesaj
- Özgün uygulama: 07 - ■nternet ve Uzak Veri/Uygulamalar/Ozgun-25-Ders Hat■rlat■c■s■