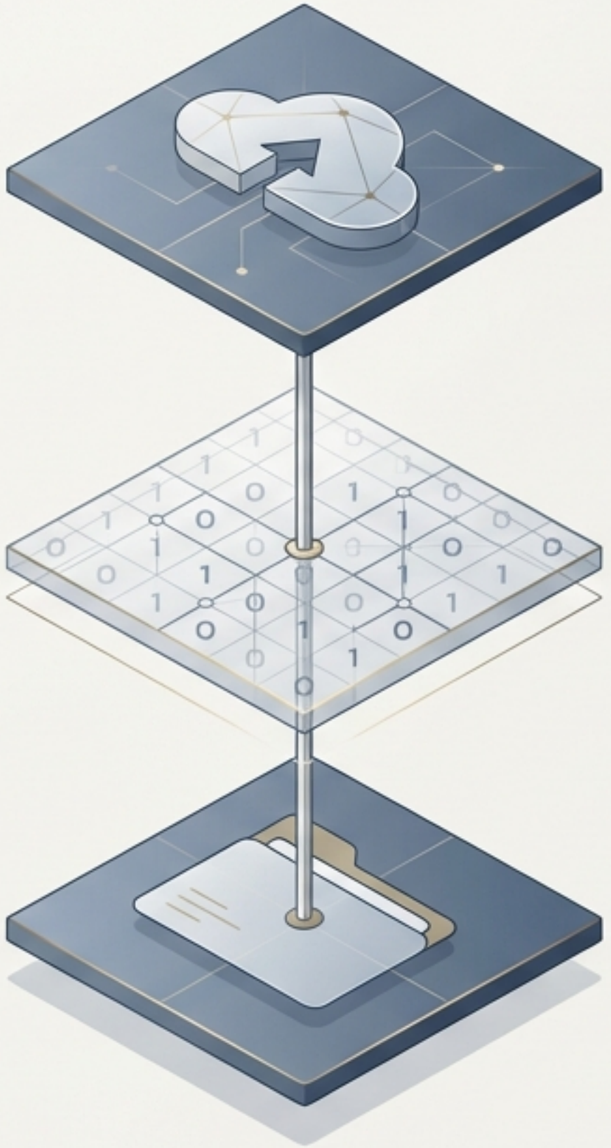
Flutter'da Görsel Yönetimi ve UX

Varlıklardan Önbelleklenmiş Ağ Görsellerine Uzmanlık Yolculuğu

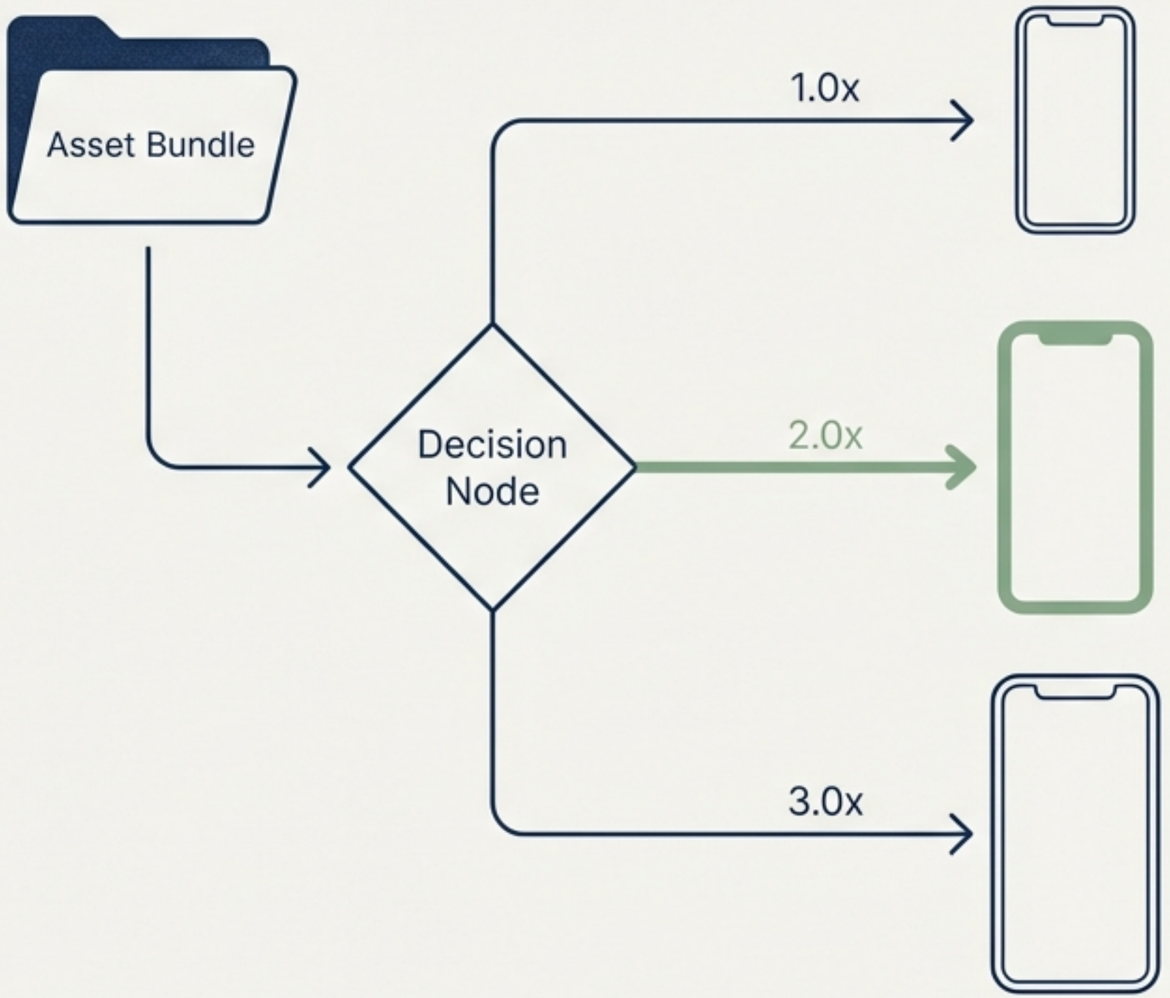


Kapsamlı Teknik Kılavuz

Yerel Varlıklarla Başlangıç: Akıllı Seçim

pubspec.yaml üzerinden tanımlanan görseller kod içinde kullanıma hazırdır. Flutter'ın Image() widget'ı, cihazın piksel oranına (pixel ratio) göre en uygun varyasyonu otomatik olarak seçer.

```
// Temel kullanım
Image(
  image: AssetImage('myassets/something.png'),
)
```



Yapıcı (Constructor) Stratejisi: const Gücü

Her iki yöntem de aynı görseli yükler, ancak const kullanımı performans optimizasyonu için kritiktir.

ÖNERİLEN

```
// Const kullanılabilir
const Center(
  child: Image(
    image: AssetImage('myassets/something.png'),
  )
);
```

ALTERNATİF

```
// Const kullanılamaz
return Center(
  child: Image(
   child: Image.asset('myassets/something.png'),
);
```

Bayt Verileriyle Çalışmak: Image.memory

Inter

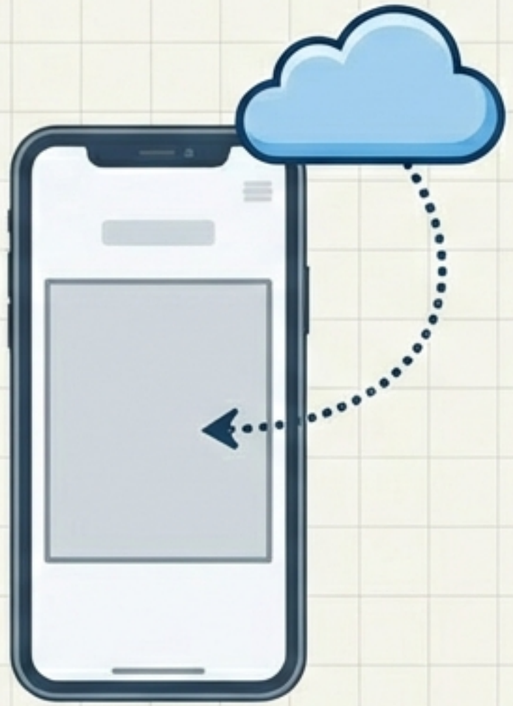
Dosya yolu yerine bellekteki bir bayt dizisi (sequence of bytes) olarak saklanan görseller için Image.memory() kullanılır.

KRİTİK UYARI: Bu yöntem yalnızca sıkıştırılmış formatları (örneğin PNG) destekler. rawRgba gibi sıkıştırılmamış formatlar çalışma zamanı hatalarına (runtime exceptions) yol açar.

Dinamik Dünyaya Geçiş: Image.network

Inter

Varlıklar yerine sunucuda depolanan görselleri yüklemek için Image.network() kurucusu kullanılır. Bu, statik içerikten dinamik içeriğe geçişin ilk adımıdır.



```
Image.network(
  'https://example.com/image.png',
)
```

Bekleme Süresini Yönetmek: loadingBuilder

Inter

Kullanıcı görsel indirilirken boş bir ekrana bakmamalıdır. loadingBuilder, indirme sürecini takip etmemizi sağlar.

```
loadingBuilder: (context, child, progress) {
  if (progress == null) return child; // Yüklendi

  // Yüzdelik hesaplama
  final total = progress.expectedTotalBytes;
  final current = progress.cumulativeBytesLoaded;

  return CircularProgressIndicator(
    value: total != null ? current / total : null
  );
}
```

Hafif Yer Tutucular: frameBuilder

Inter

İlerleme yüzdesi yerine sadece basit bir mesaj veya yer tutucu göstermek istediğinizde frameBuilder idealdir. Görselin ilk karesi hazır olana kadar geçici bir widget gösterir.

```
frameBuilder: (context, child, frame, loaded) {
  if (loaded) return child;
  return const Text('Loading...');
}
```

Stratejik Karar: Hangi Builder Ne Zaman?

loadingBuilder

- Büyük dosyalar için ideal.
- Kesin ilerleme bilgisi (yüzde) gösterir.
- UYARI: Widget çok sık yeniden oluşturulur (rebuild).

frameBuilder

- Küçük görseller ve ikonlar.
- Özel animasyonlar (fade-in).
- İlerleme verisi gerekmediğinde performans dostudur.

Barlow Condensed:Standart Yöntemin Sınırları

Inter

Image.network kullanışlıdır ancak profesyonel uygulamalar için kritik eksikleri vardır.

Pain Points:

- **Hata Yönetimi Yok**: 404 veya internet kopmasını yakalayamaz.
- **Önbellek (Cache) Yok**: Her seferinde tekrar indirir, veri israfı yapar.
- **Boilerplate Zorunluluğu:** Hataları çözmek için manuel HTTP istekleri yazmak gerekir.

Çözüm Ortağı: cached_network_image

Hata yönetimi, önbellekleme ve görsel getirme süreçlerini otomatize eden endüstri standardı paket.

Otomatik Önbellekleme

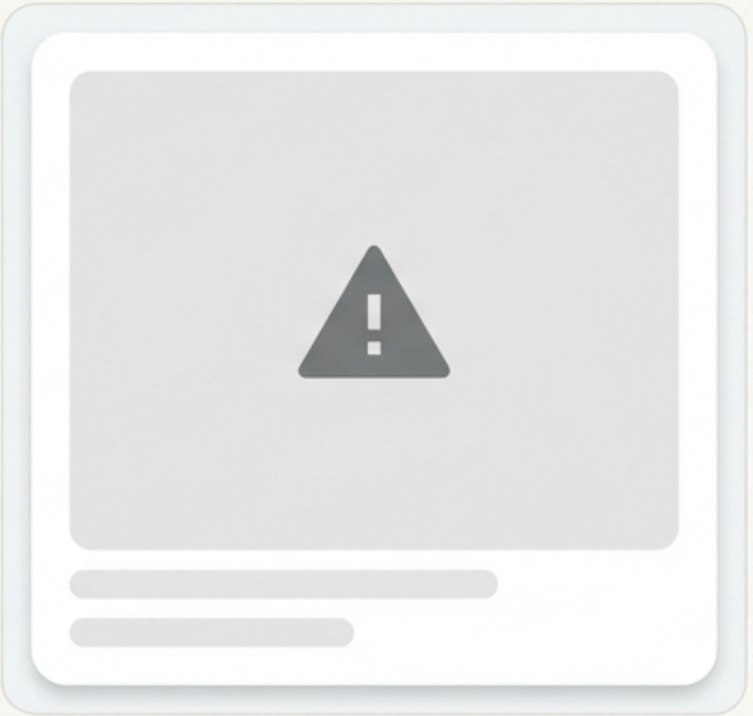
Kolay Hata Yakalama

Dahili Yükleme Animasyonları

Zarif Hata Yönetimi

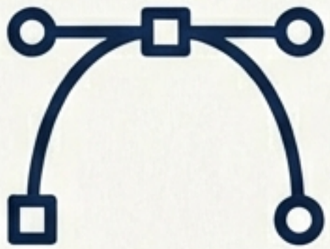
Bağlantı hataları veya geçersiz formatlar errorWidget ile kolayca yönetilir. Kullanıcı asla kırık bir deneyim yaşamaz.

```
CachedNetworkImage(
  imageUrl: 'https://example.com/img.png',
  placeholder: (context, url) =>
    CircularProgressIndicator(),
  errorWidget: (context, url, error) => const
      Icon(Icons.error),
);
```



İnce Ayarlar ve Özelleştirme

CachedNetworkImage sadece yükleme yapmaz, sunum şeklini de kontrol etmenizi sağlar.

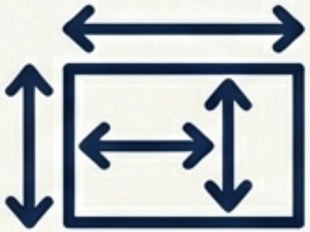


Barlow Condensed Animasyon Kontrolü

Varsayılan Curves.easeIn animasyonu, süresi ve türü değiştirilebilir.

HTTP Başlıkları

Token veya özel header gerektiren istekler desteklenir.



Boyut ve BoxFit

Görsel boyutları ve yerleşim modları standart widget gibi yönetilir.

Önbellek ve İlerleme Takibi Bir Arada

progressIndicatorBuilder hem önbellek avantajını korur hem de kullanıcıya indirme yüzdesini gösterir.

```
CachedNetworkImage(
  imageUrl: imgUrl,
  progressIndicatorBuilder: (context, url, status) {
    return CircularProgressIndicator(
      value: status.progress, ←———— Otomatik hesaplanan veri
    );
  },
  errorWidget: (context, url, error) => const Icon(Icons.error),
);
```

Önbellek Mekanizması Nasıl Çalışır?

Sistem, gereksiz ağ trafiğini önlemek için cihaz depolamasını kullanır.

Gelişmiş: BaseCacheManager ile özel cache yönetimi yapılabilir.

Özet: Neden cached_network_image?

Özellik	Image.network	CachedNetworkImage
Animasyon	Manuel Kodlama	Otomatik / Dahili
Önbellek (Cache)	Yok (Sıfırdan yazılmalı)	Otomatik (Temp dizini)
Hata Yönetimi	Yok (Karmaşık)	`errorWidget` ile Basit

Kullanıcılarınızın verisini koruyun,
hataları zarifçe yönetin.