**DART ASENKRON PROGRAM ÖRNEĞİ**

Dart dilinde **asenkron programlama**, özellikle **Future** ve **async/await** anahtar kelimeleri kullanılarak gerçekleştirilir. Asenkron programlama, kodun engellenmeden çalışmasını sağlar, böylece uzun süren işlemler (örneğin, ağ istekleri veya dosya işlemleri) uygulamanın diğer kısımlarını etkilemez.

Aşağıda, **Future**, **async/await** ve **Stream** kullanımını gösteren detaylı bir örnek yer almaktadır.

**Örnek Senaryo:**

Bir e-ticaret uygulamasında, **ürünleri sunucudan çekme, sepete ekleme ve siparişi tamamlama** işlemlerini simüle eden asenkron bir sistem oluşturacağız.

**Kod Örneği:**

import 'dart:async';

// Sunucudan ürünleri getiren asenkron fonksiyon

Future<List<String>> fetchProducts() async {

print("Ürünler getiriliyor...");

await Future.delayed(Duration(seconds: 2)); // Sunucudan veri çekme simülasyonu

return ["Laptop", "Telefon", "Tablet"];

}

// Sepete ürün ekleyen asenkron fonksiyon

Future<void> addToCart(String product) async {

print("$product sepete ekleniyor...");

await Future.delayed(Duration(seconds: 1)); // Sepete ekleme süresi

print("$product sepete eklendi!");

}

// Sipariş tamamlama fonksiyonu (Stream kullanımı ile)

Stream<String> completeOrder(List<String> cart) async\* {

print("Sipariş tamamlanıyor...");

for (var product in cart) {

await Future.delayed(Duration(seconds: 1)); // İşlem süresi

yield "$product siparişi tamamlandı!";

}

yield "Tüm siparişler tamamlandı!";

}

// Ana fonksiyon

void main() async {

// 1. Ürünleri getir

List<String> products = await fetchProducts();

print("Gelen Ürünler: $products");

// 2. İlk ürünü sepete ekle

if (products.isNotEmpty) {

await addToCart(products[0]);

}

// 3. Siparişi tamamla

List<String> cart = [products[0]];

await for (String status in completeOrder(cart)) {

print(status);

}

print("İşlem tamamlandı!");

}

**Kod Açıklaması:**

1. **fetchProducts()**: Sunucudan ürünleri getirir. **await Future.delayed()** kullanarak bir süre beklemesini sağlıyoruz.
2. **addToCart()**: Belirtilen ürünü sepete ekler. **await** kullanarak işlemin tamamlanmasını bekliyoruz.
3. **completeOrder()**: Stream kullanarak sipariş sürecini takip ederiz. **yield** anahtar kelimesiyle adım adım veri döndürülmesini sağlıyoruz.
4. **main()**: Bütün fonksiyonları **await** kullanarak sırayla çalıştırır.

**Çıktı:**

Ürünler getiriliyor...

Gelen Ürünler: [Laptop, Telefon, Tablet]

Laptop sepete ekleniyor...

Laptop sepete eklendi!

Sipariş tamamlanıyor...

Laptop siparişi tamamlandı!

Tüm siparişler tamamlandı!

İşlem tamamlandı!

**Önemli Noktalar:**

* **async/await**: Future'ları beklemek için kullanılır.
* **Future.delayed**: Asenkron işlemleri simüle etmek için.
* **Stream ve yield**: Uzun süren işlemleri adım adım yönetmek için.

Bu şekilde, **UI'yi kilitlemeden** asenkron işlemleri yönetebiliriz. Özellikle **Flutter** gibi mobil geliştirme ortamlarında performans açısından büyük avantaj sağlar. 🎯