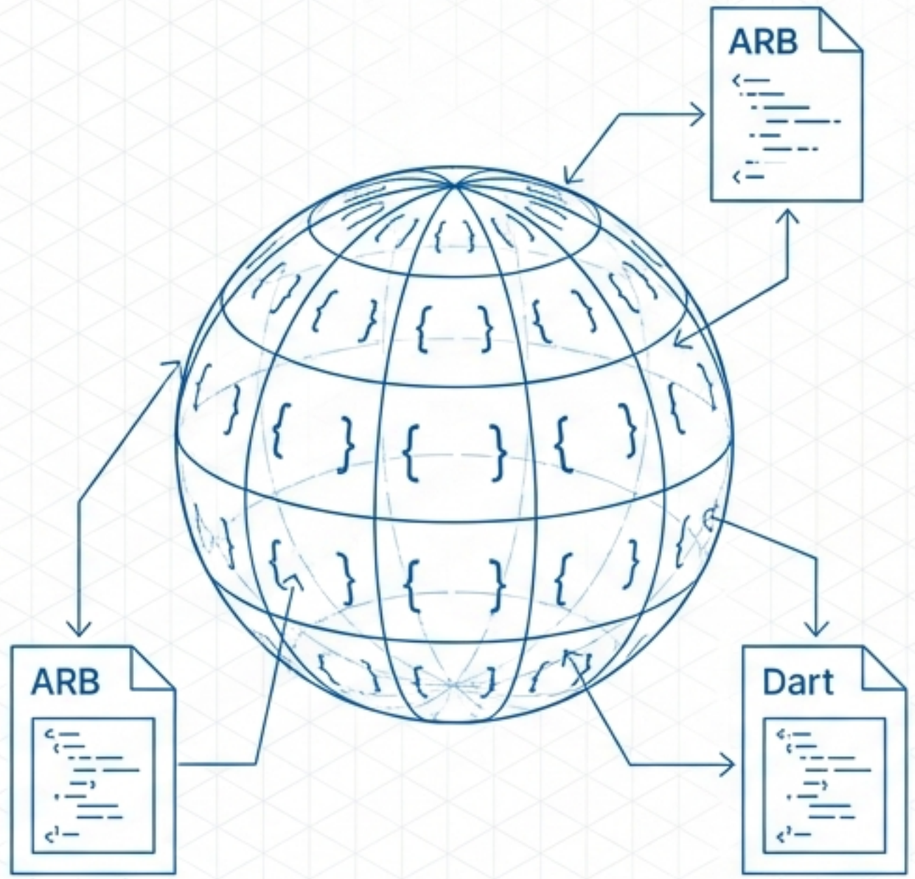
TEKNİK REHBER SERİSİ: BÖLÜM 13.3

Flutter’da Profesyonel Yerelleştirme: intl Paketi

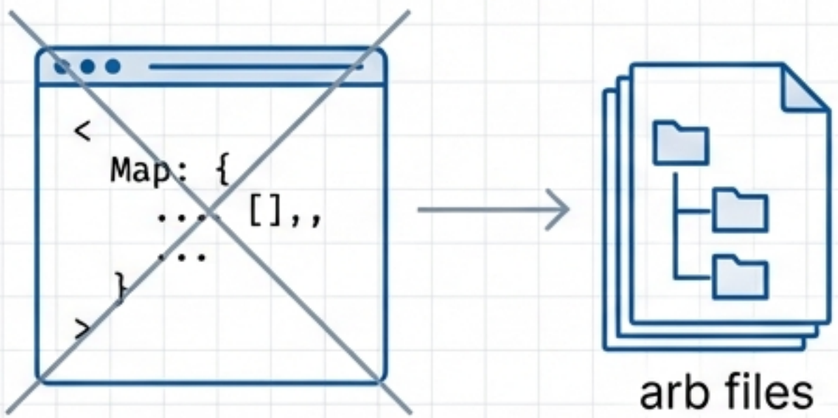
Ölçeklenebilir, Otomatize ve Dosya Tabanlı Bir Dil Altyapısı Mimarisi



Manuel Haritalardan (Map) Sistemik Yönetime Geçiş

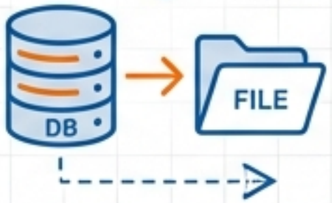
Temel Fark

`intl` paketi ile yerelleştirme süreci, basit bir `Map` kullanımından farklıdır. Kurulum aynı görünse de, çeviri veritabanı kodun içinde değil, yönetilebilir özel dosyalarda tutulur.

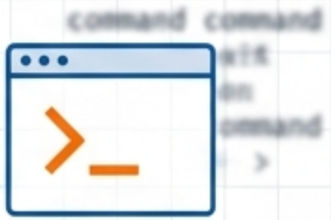


The Benefits

Avantajlar



Ayrıştırılmış Veritabanı
Çeviriler koddan bağımsız dosyalarda yaşar.



Komut Satırı Araçları
Manuel yönetim yerine otomasyon.

Standartlaştırma
Endüstri standardı formatlar (ARB).

Temel Bileşenler ve Bağımlılıklar

pubspec.yaml

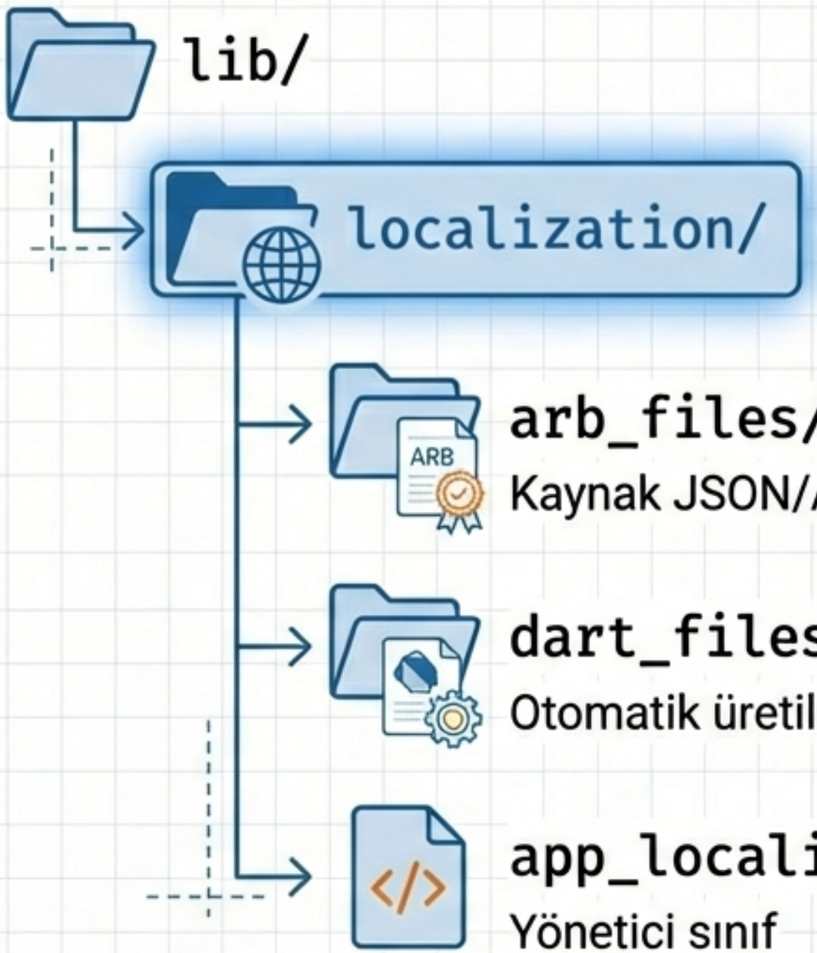
```
dependencies:
  flutter_localizations:
    sdk: flutter
  intl: ^0.16.1
  intl_translation: ^0.17.10
```

Flutter'ın yerel widget çevirileri için SDK desteği.

Tarih, sayı formatlama ve mesaj işleme motoru.

ARB dosyalarını işleyen ve Dart koduna dönüştüren araç seti.

Proje Mimarisi ve Dosya Hiyerarşisi



Yapısal Organizasyon

Basit projelerden farklı olarak, `localization/` klasörü potansiyel olarak yüzlerce dosya içerebileceği için alt klasörlere (`arb_files`, `dart_files`) bölünerek organize edilmelidir.

Motoru İnşa Etmek: AppLocalization Sınıfı

AppLocalization.dart

```
static Future load(Locale locale) async {
  final String localeName = Intl.canonicalizedLocale(locale.languageCode);

  // HATA UYARISI: Kod üretimi yapılana kadar geçici hata
  await initializeMessages(localeName);

  Intl.defaultLocale = localeName;
  return AppLocalization(locale);
}
```

Mühendis Notu

`initializeMessages` metodu henüz mevcut değildir. Bu, iş akışının doğal bir parçasıdır; ilerleyen adımlarda otomatik kod üretimi ile bu boşluk doldurulacaktır.

İletileri Tanımlamak ve İsimlendirme Kuralı

Kritik Kural: Getter ismi ile **name** parametresinin eşleşme zorunluluğu.

✅ DOĞRU KULLANIM

Tam Eşleşme

```
String get helloWorld => Intl.message("Hello world!", name: "helloWorld");
```

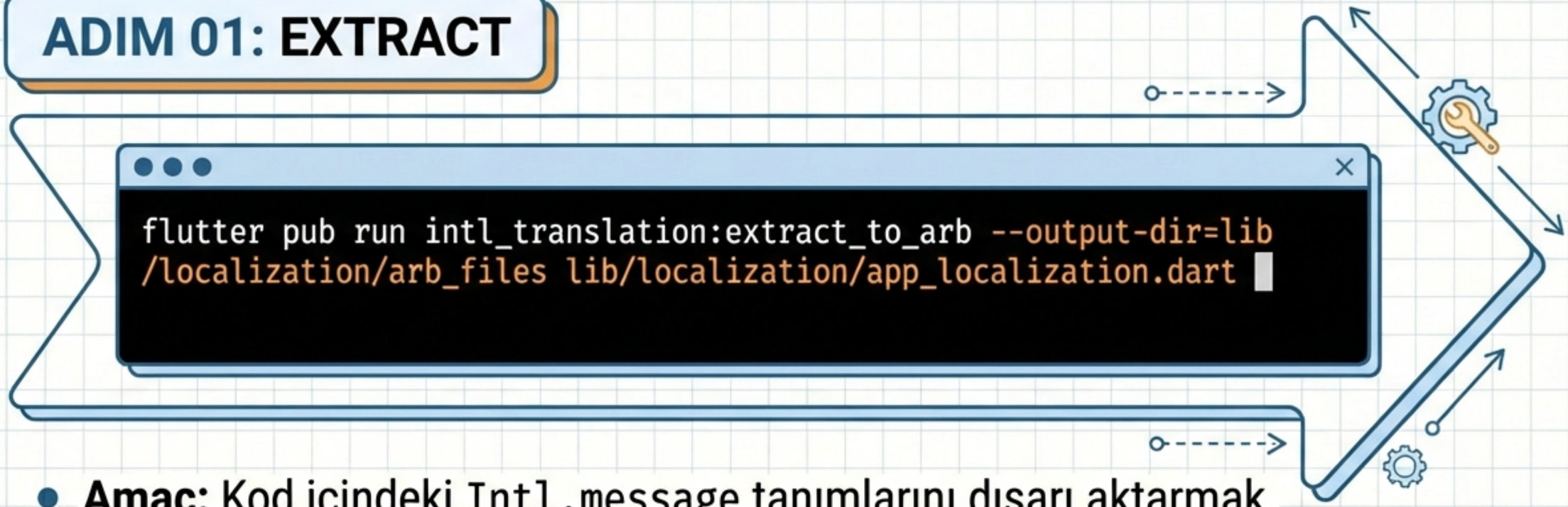
❌ YANLIŞ KULLANIM

EŞLEŞME HATASI

```
String get helloWorld => Intl.message("Hello world!", name: "hello_world"); // HATA
```

İş Akışı Adım 1: ARB Dosyalarına Çıkarma

ADIM 01: EXTRACT



```
flutter pub run intl_translation:extract_to_arb --output-dir=lib
/localization/arb_files lib/localization/app_localization.dart
```

- **Amaç:** Kod içindeki `Intl.message` tanımlarını dışarı aktarmak.
- **Sonuç:** `localization/arb_files` klasöründe `intl_messages.arb` isimli bir şablon dosya oluşur.

ARB Dosyalarını Anlamak ve Yönetmek

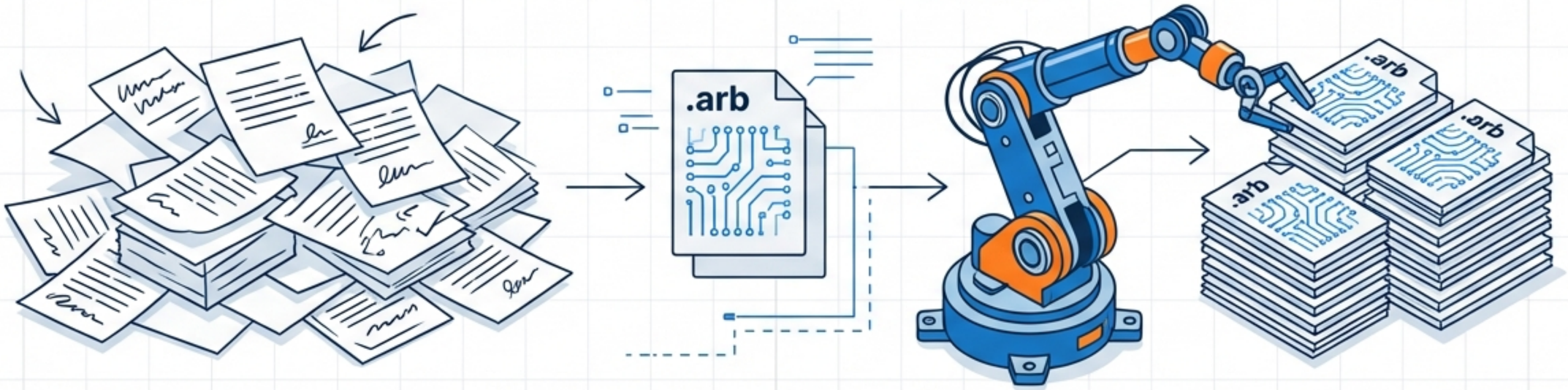
ARB Nedir?

ARB (Application Resource Bundle), '@' ile başlayan metadata anahtarları içeren özel bir JSON formatıdır.

Araç Önerileri

Dil sayısı arttıkça manuel düzenleme zorlaşır. Profesyonel yönetim için önerilen araçlar:

- **Localizely**
- **BabelEdit**
- **Crowdin**



MANUEL DÜZENLEME

PROFESYONEL YÖNETİM

İş Akışı Adım 2: Çeviri Veritabanını Oluşturmak

ADIM 02: TRANSLATE

⚠ **UYARI:** JSON haritasındaki anahtarlara (keys) dokunmayın, sadece değerleri (values) değiştirin. Anahtarların değişmesi otomasyonu bozar.

İş Akışı Adım 3: Kod Üretimi (Code Generation)

ADIM 03: GENERATE

```
flutter pub run intl_translation:generate_from_arb \
  --output-dir=lib/localization/dart_files \
  --no-use-deferred-loading lib/localization/app_localization.dart \
  lib/localization/arb_files/intl_*.arb
```

****Sonuç:**

1. `dart_files/ klasöründe gerekli dosyalar üretilir.
2. `AppLocalization` içindeki `initializeMessages` hatası ortadan kalkar ve proje derlenebilir hale gelir.

Köprüyü Kurmak: AppLocalizationDelegate

Uygulama ile çeviri mantığını bağlayan sınıf.


```dart
class AppLocalizationDelegate extends LocalizationsDelegate {
  // Desteklenen dillerin kontrolü
  bool isSupported(Locale locale) => ["en", "it", "es"].contains(locale.languageCode);

  // Yükleme işlemi
  Future load(Locale locale) => AppLocalization.load(locale);

  bool shouldReload(LocalizationsDelegate d) => false;
}
```

Ana Konfigürasyon (MaterialApp)

```
return MaterialApp(
  localizationsDelegates: [
    const AppLocalizationDelegate(), // Bizim özel sınıfımız
    GlobalMaterialLocalizations.delegate,
    GlobalCupertinoLocalizations.delegate,
    GlobalWidgetsLocalizations.delegate,
  ],
  supportedLocales: const [
    Locale.fromSubtags(languageCode: "en"),
    Locale.fromSubtags(languageCode: "it"),
    Locale.fromSubtags(languageCode: "es"),
  ],
);
```

Flutter'a kendi çeviri motorumuzu tanıtıyoruz.

Arayüz (UI) İçinde Kullanım

Standart Kullanım

```
Text(AppLocalization.of(context)?.helloWorld ?? "-");
```

Sorun: Null-check gerektirir, uzundur ve okunması zordur.

PRO İPUCU: Extension Metot

```
Text(context.localize.helloWorld);
```

Extension metot tanımlayarak `context.localize` yapısı kurun.
Kodunuz daha temiz ve nullable tür karmaşasından uzak olsun.

Akıllı Çoğul Kullanımı (Plurals)

Manuel `if/else` bloklarından kurtulun.

❌ Manuel Kontrol

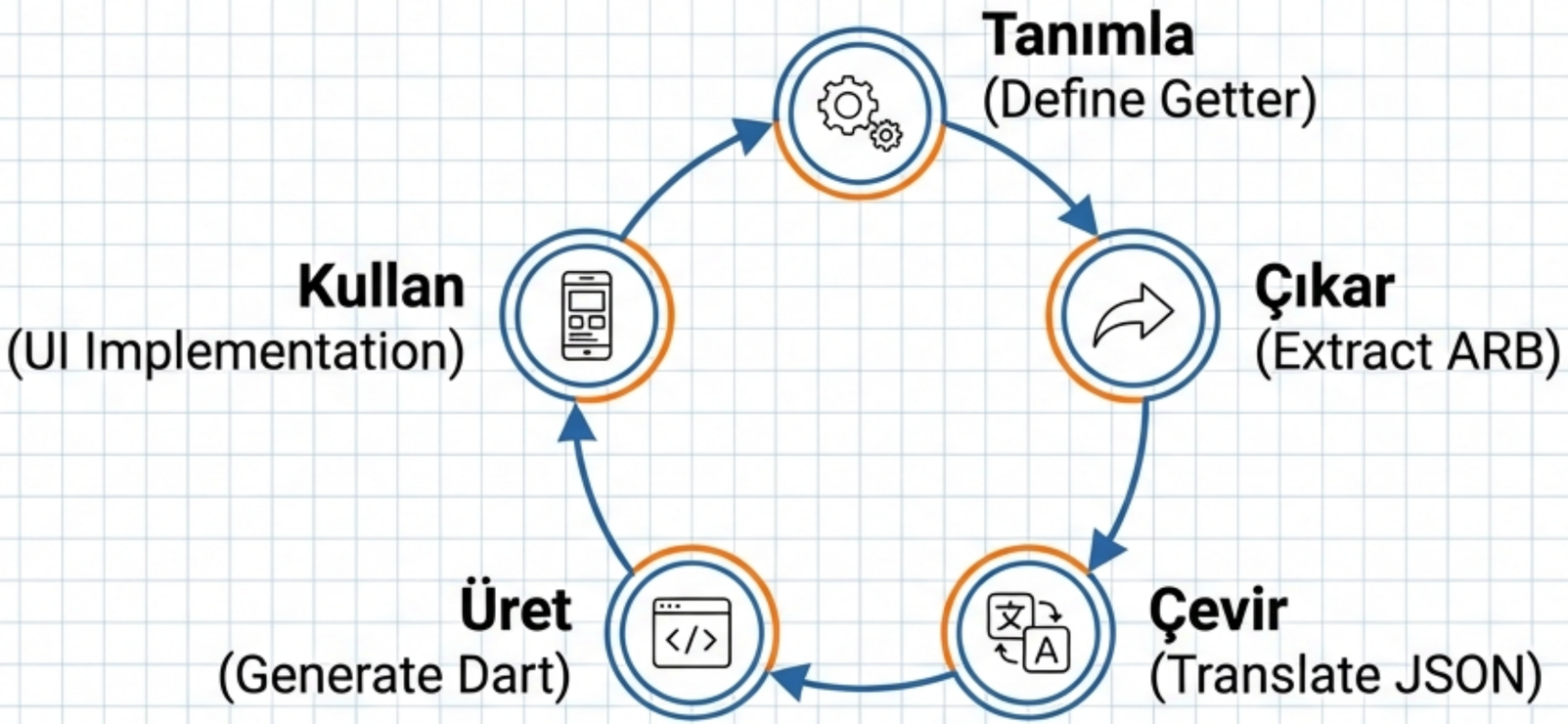
```
if (days == 0)
  "No days";

else if (days == 1)
  "1 day";
```

✅ Intl.plural

```
String remainingDays(int days) ⇒
  Intl.plural(days,
    zero: "No days remaining",
    one: "One day left",
    other: "$days days remaining",
    name: "remainingDays",
    args: [days],
);
```

⚙ Enterpolasyon için gereklidir.

Özet ve Kazanımlar: Tam Döngü



Sonuç: `intl` paketi ile sadece metinleri çevirmiyoruz; **çoğul ekleri** ve **veri formatlamayı** içeren, **koddan bağımsız yaşayan canlı bir dil altyapısı** kuruyoruz.

Technical Blueprint Editorial