Domain-Driven Design (DDD)

Ali Emre Pamuk

April 2025

Giriş

Yazılım karmaşıklığı arttıkça, bu karmaşıklığı yönetmek ve daha sağlam, esnek ve sürdürülebilir sistemler oluşturmak için yeni yaklaşımlara ihtiyaç duyulmaktadır. **Domain-Driven Design (DDD)**, bu ihtiyaçlara cevap veren güçlü bir yaklaşımdır. DDD, yazılımın karmaşıklığını domain kavramlarıyla yönetmeyi amaçlar.

Domain Nedir?

Domain, bir yazılım sisteminin ele aldığı problem alanını temsil eder. Yazılımın ne yapacağına ve hangi problemleri çözeceğine dair her şey bu kavramda tanımlanır.

Örnek Domain'ler:

- E-ticaret: Ürünler, Siparişler, Müşteriler, Ödemeler
- Sosyal Medya: Kullanıcılar, Profiller, Gönderiler
- Bankacılık: Hesaplar, İşlemler, Krediler

Domain Unsurları

- **Varlıklar (Entities)**: Kimliği olan, değiştirilebilir nesneler.
- **Kurallar**: Varlıklar arası ilişkileri tanımlar.
- **İşlemler**: Domain içindeki operasyonlar.

DDD'nin Temel Kavramları

Entity:

- Kimliği olan nesnelerdir.
- **Örnek**: Ürün, Sipariş, Müşteri

Value Object:

- Kimliksiz, immutable değer nesneleridir.
- **Örnek**: Fiyat, Sipariş Kalemi

Aggregate:

- Bir grup Entity ve ValueObject'i kapsayan tutarlılık sınırıdır.
- **Örnek**: Sipariş (Sipariş Kalemleri, Fiyat, Müşteri ile birlikte)

Bounded Context:

- Domain'in mantıksal olarak ayrılmış alt bölümleridir.
- **Örnek**: Müşteri Yönetimi, Ürün Yönetimi

4 Katmanlı Mimari

DDD genellikle aşağıdaki mimari yapı üzerine kuruludur:

- **Domain Katmanı**: İş kuralları, domain modelleri
- **Application Katmanı**: Kullanım senaryoları, servisler
- **Infrastructure Katmanı**: Veritabanı, dış servisler
- **Presentation Katmanı**: Arayüz, API, kullanıcıyla etkileşim

Bounded Context Haritası

E-ticaret sistemi aşağıdaki bounded context'lere ayrılabilir:

- Müşteri Yönetimi
- Ürün Kataloğu
- Sipariş Yönetimi
- Ödeme Sistemi
- Teslimat Takibi

Aggregate Yapısı

Bir aggregate, domain'in bir parçasını yönetir ve dış dünya ile sadece **aggregate root** üzerinden etkileşim kurar.

Sipariş Aggregatesi Örneği:

- Entity: Sipariş (Aggregate Root)
- Entity: Müşteri
- Value Object: Sipariş Kalemi, Fiyat

Ubiquitous Language (Ortak Dil)

DDD'de, tüm ekip aynı dili kullanmalıdır. Ortak dil kodlara, dökümana ve sözlü iletişime yansır.

Örnek:

- `createOrder()` metodunun adı, domain terimleriyle birebir uyuşur.
- "Sipariş Kalemi" kodda da sınıf adı olarak geçer: `OrderItem`

Domain Events

Domain Event'ler sistemdeki önemli olayları ifade eder.

Örnek:

- `OrderPlacedEvent` tetiklendiğinde, ödeme servisi bilgilendirilir.

Repositories

Repository'ler domain nesnelerinin veritabanıyla etkileşimini soyutlar.

Örnek:

- `OrderRepository.getById(orderId)`
- `CustomerRepository.save(customer)`

CQRS (Command Query Responsibility Segregation)

Komutlar (Commands) ve sorgular (Queries) ayrı yapılarla ele alınır.

Örnek:

- **Command:** `PlaceOrderCommand`
- **Query:** `GetOrderDetailsQuery`

Avantajları:

- Performans optimizasyonu
- Farklı veri modelleriyle çalışabilme

Sonuç

Domain-Driven Design, karmaşık sistemlerin tasarımında domain'e odaklanarak yazılımı daha anlamlı ve sürdürülebilir kılar. Doğru uygulandığında ekip içi iletişimi güçlendirir, kodun anlaşılırlığını ve değiştirilebilirliğini artırır.