

## Django'nun Temel Özellikleri

- Django, yüksek seviyeli, erişilebilir ve açık kaynaklı bir Python web framework'üdür. Hızlı geliştirme ve kod yeniden kullanılabilirliği sağlamak için tasarlanmıştır.
- Model-View-Controller (MVC) desenini takip ederek, geliştiricilerin web uygulamalarını hızlı ve verimli bir şekilde inşa etmelerine olanak tanır.

## Django ile Yapılabilcek Uygulamalar

- Django ile içerik yönetim sistemleri (CMS), sosyal medya platformları, iş uygulamaları ve web siteleri gibi çeşitli web uygulamaları oluşturulabilir.
- "Batteries included" ifadesi, Django'nun kapsamlı özelliklerini vurgular; birçok işlevselligi kutudan çıkar çıkmaz sunar.

## Güvenlik ve Yönetim Özellikleri

- Django, XSS, CSRF ve SQL enjeksiyonu gibi yaygın web güvenlik açıklarına karşı koruma sağlar.
- Kullanıcı hesaplarını yönetmek için yerleşik kimlik doğrulama ve yetkilendirme mekanizmaları sunar; kullanıcı izinleri ve erişim kısıtlamaları tanımlanabilir.

Django, web uygulamaları geliştirmek için harika bir çerçevedir ve birçok avantaj sunar. Öncelikle, Django yüksek seviyeli, erişilebilir ve açık kaynaklı bir Python web çerçevesidir. Bu, geliştiricilerin hızlı ve verimli bir şekilde web uygulamaları oluşturmaya olanak tanır. Django, "pil ile birlikte gelir" ifadesiyle tanımlanır; bu, çerçevenin kapsamlı özellikler ve işlevsellik sunduğu anlamına gelir. Örneğin, veri modellerini Python sınıfları kullanarak tanımlamanıza olanak tanıyan bir nesne-ilişkisel eşleme (ORM) katmanı vardır. Bu, veritabanlarıyla çalışmayı ve kayıtları sorgulama, ekleme, güncelleme ve silme gibi işlemleri kolaylaştırır.

Django ayrıca güvenlik özellikleriyle doludur; yaygın web güvenlik açıklarına karşı koruma sağlar. Kullanıcı hesaplarını yönetmek için yerleşik kimlik doğrulama ve yetkilendirme mekanizmaları sunar. Ayrıca, Django'nun genişletilebilirliği sayesinde, projelerinize özel işlevsellik eklemek için modüller (Django uygulamaları veya paketleri) kullanabilirsiniz. Örneğin, dil yerelleştirmesi veya form doğrulama gibi özellikler eklemek mümkündür. Tüm bu özellikler, Django'yu hem yeni başlayanlar hem de deneyimli geliştiriciler için cazip bir seçenek haline getirir.

## Nesne ve Sınıflar

- Nesneler, verileri ve bu verilerle ilişkili davranışları içeren yapılardır. Örneğin, bir hasta nesnesi, belirli bir hastayı temsil eder.
- Sınıflar, nesnelerin genel versiyonlarıdır ve nesne oluşturulmadan önce tanımlanır. Örneğin, "hasta" sınıfı, belirli bir hasta nesnesinin (örneğin, Nia Patel) oluşturulmasında bir şablon görevi görür.

## Sınıf Diyagramları

- Sınıf diyagramları, nesne yönelimli tasarımında sınıfların birbirleriyle olan ilişkilerini gösterir. Her kutu bir sınıfı temsil eder ve sınıfın özelliklerini içerir.
- Alt sınıflar, üst sınıflarından özellikleri miras alır. Örneğin, "doktor" sınıfı, "tıbbi personel" sınıfının alt sınıfıdır ve bu nedenle onun tüm özelliklerine sahiptir.

### Nesne Yönelimli Tasarım

- OOAD, yazılım sistemlerini nesnelerin etkileşimleri temelinde planlama sürecidir. Bu yaklaşım, birden fazla geliştiricinin uygulamanın farklı yönleri üzerinde aynı anda çalışmasına olanak tanır.
- UML diyagramları, sistemin statik yapısını ve dinamik davranışını görselleştirmek için kullanılır.

### Nesne-İlişkisel Eşleme (ORM) Nedir?

- ORM, nesne yönelimli programlama dilleri ile ilişkisel veritabanları arasında bir köprü kurar.
- Geliştiricilerin SQL yazmadan veritabanları ile etkileşimde bulunmalarını sağlar.

### SQL ve OOP Arasındaki Farklar

- SQL, verileri tablolar, satırlar ve sütunlar ile modelleyerek çalışırken, OOP sınıflar ve nesneler ile modelleme yapar.
- OOP, ilişkileri kalıtım, ilişki ve toplama gibi sınıf desenleri ile tanımlar; SQL ise JOIN ve YABANCI ANAHTAR kullanır.

### ORM'nin Avantajları ve Dezavantajları

- Avantajlar: SQL yazmadan veritabanı işlemleri yapma, farklı veritabanı sistemlerini yönetme kolaylığı.
- Dezavantajlar: ORM, nesneleri veritabanı tablolarına eşleme konusunda zorluklar çıkarabilir ve uygulama performansını etkileyebilir.

### Django ORM Temel Kavramları

- Django modelinin her biri bir veritabanı tablosuna karşılık gelir; model sınıfı, tablo satırını temsil ederken, her alan tablo sütununu temsil eder.
- Alanlar, Django'nun sağladığı alan sınıfları ile tanımlanır ve her alanın veri türü ve kısıtlamaları gibi meta verileri belirlenir.

### İlişkilerin Modellenmesi

- Django ORM, birden fazla ilişki türünü destekler: bire bir, bire çok ve çoktan çoga.
- Örneğin, bir Eğitmen ve Kullanıcı arasında bire bir ilişki tanımlanabilir; bir Eğitmen yalnızca bir Kullanıcıya karşılık gelir.

### Model Mirası

- Django'da model mirası, Python'daki mirasa benzer; üç senaryo vardır: çoklu tablo modu, soyut temel sınıflar ve proxy modeller.
- Çoklu tablo mirası, her modelin kendi veritabanı tablosuna sahip olmasını sağlar ve bu, bir bire bir ilişki gibidir.

## CRUD İşlemleri

- Django Model API'leri, SQL sorguları yazmadan nesneleri manipüle etmenizi sağlar.
- Kullanıcı, Eğitmen ve Kurs modelleri gibi örneklerle CRUD işlemleri gösterilmektedir.

## Nesne Oluşturma ve İlişkilendirme

- Django modelinde bir nesne oluşturup, modelin save metodunu çağırarak veritabanına kaydedebilirsiniz.
- İlişkili modeller arasında bağlantılar kurmak için Yabancı Anahtar veya Çoktan Çoğa alanları kullanılır.

## Veri Okuma ve Sorgulama

- Tüm nesneleri okumak için model yönetici kullanarak bir QuerySet oluşturulur.
- filter ve exclude metodları ile belirli koşullara göre alt küme oluşturulabilir.

## Veri Güncelleme ve Silme

- Nesne alanlarını güncelliyerek veritabanındaki kayıtları güncelleyebilirsiniz.
- Kayıtları silmek için model nesnesi veya QuerySet üzerinde delete методu çağrılır.

## İlişkili Modeller

- Kullanıcı, eğitmen ve öğrenci arasında birer birebir ilişki vardır.
- Kurs, eğitmen ve öğrenci ile çoktan çoha, proje ile ise çoktan bire bir ilişkiye sahiptir.

## İlişki Türleri

- Django, model ilişkisini yalnızca bir tarafta tanımlamanızı gerektirir; bu, ileri erişim olarak adlandırılır.
- Geriye erişim, Django'nun otomatik olarak oluşturduğu bir ilişkidir ve kullanıcidan öğrencisiye erişim sağlar.

## Silme Davranışları

- Django, ilişkili nesneleri silerken farklı silme davranışları destekler; örneğin, "cascade" seçeneği ile ilişkili tüm projeler silinir.
- "Protect" seçeneği, ilişkili projeleri olan bir kursun silinmesini engeller.

## İlişkili Nesnelerle Çalışma

- İlişkili nesneleri yönetmek için Django'da ekleme, oluşturma, kaldırma, temizleme ve değiştirme gibi yöntemler bulunmaktadır.

- Bu yöntemler, ilişkili nesnelerle etkileşimde bulunmayı kolaylaştırır.