Django'nun Temel Bileşenleri

Saygın Python web framework'ü Django, güvenli ve sürdürülebilir web uygulamalarının geliştirilmesini verimli bir şekilde kolaylaştırmak için tasarlanmıştır. Özünde Django, yapısını ve işlevini tanımlayan birkaç temel bileşen üzerine inşa edilmiştir ve geliştiricilerin karmaşık, veri odaklı web sitelerini hızlı bir şekilde oluşturmasını sağlar. Proje mimarisi, modeller, görünümler, şablonlar, formlar ve yönetim arayüzünü içeren Django'nun birincil yapı taşlarını inceleyelim.

Proje Mimarisi

Bir Django projesi, veritabanı ile etkileşimi ve uygulamaya özel ayarlar da dahil olmak üzere uygulamanın nasıl çalıştığını tanımlayan bir dizi ayar ve yapılandırma olarak yapılandırılmıştır. Her proje, her biri uygulama içinde farklı özellikler veya hizmetler sağlamak için uyarlanmış birden fazla uygulamayı kapsayabilir. Bu uygulamalar modüler ve yeniden kullanılabilirdir, yeniden kullanılabilirlik göz önünde bulundurularak tasarlanırsa potansiyel olarak farklı projeler arasında paylaşılabilir.

Tipik Django proje düzeni aşağıdaki gibi görünür:

```
myproject/
    manage.py
    myproject/
        __init__.py
        settings.py
        urls.py
        wsgi.py
    app_one/
        migrations/
        models.py
        views.py
        ...
    app_two/
        migrations/
        models.py
        views.py
        ...
```

Bu yapıda, manage.py Django projesi ile etkileşim için bir komut satırı yardımcı programıdır, settings.py tüm yapılandırmaları saklar ve urls.py proje için URL bildirimlerini yönetir.

Modeller

Django modelleri, veri yapısını tanımlayan, esasen veritabanı şemasını kodda temsil eden Python sınıflarıdır. DRY (Don't Repeat Yourself - Kendini Tekrar Etme) prensibine uygun olarak tasarlanmış, yönettiğiniz verilerin alanlarını ve davranışlarını içerirler.

Basit bir blog yazısı modeli şu şekilde özetlenebilir:

```python
from django.db import models

class Post(models.Model):
    title = models.CharField(max_length=200)
    text = models.TextField()
    author = models.ForeignKey('auth.User', on_delete=models.CASCADE)
    created_date = models.DateTimeField(auto_now_add=True)
    published_date = models.DateTimeField(blank=True, null=True)
```

Görünümler

Django'daki görünümler, iş mantığını yürütmekten ve modeller ile şablonlar arasında arayüz oluşturmaktan sorumludur. Modellerden veri alır ve işlerler, ardından bu verileri sunum için şablonlara aktarırlar. Görünümler fonksiyonlar veya sınıflar olarak tanımlanabilir ve urls.py'de belirtilen URL'lere bağlanır.

İşte blog gönderilerini görüntüleyen bir görünüm örneği:

```
from django.shortcuts to render
from .models import Post

def post_list(request):
    posts = Post.objects.filter(published_date__isnull=False).order_by('published_date')
    return render(request, 'blog/post_list.html', {'posts': posts})
```

Şablonlar

Şablonlar bir Django uygulamasının sunum katmanını yönetir. Bunlar, dinamik içerik oluşturma için Python benzeri ifadeler içeren Django Şablon Dili ile gömülü HTML dosyalarıdır.

Yukarıdaki görünüm için örnek bir şablon şöyle olabilir:

```
<html>
<head>
    <title>Django Blog</title>
</head>
<body>
    <h1>Posts</h1>
    {% for post in posts %}
    <div>
        <h2>{{ post.title }}</h2>
        <p>{{ post.text }}</p>
    </div>
    {% endfor %}
</body>
</html>
```

Formlar