On Bölüm

Üçüncü Taraf Uygulamalarını Entegre Etme

Django'nun üçüncü taraf paketlerini kullanma

Django, ünlü için onun aerodinamik, yüksek seviyeli çerçeve O Gelişimi hızlandırır, aynı zamanda çeşitli üçüncü taraf paketlerden önemli ölçüde yararlanır. Bu paketler, Django'nun temel özelliklerini zenginleştirerek geliştiricilerin sosyal kimlik doğrulama ve gelişmiş veri manipülasyonu gibi karmaşık işlevleri verimli bir şekilde entegre etmelerini sağlar. Bu makale, Django'nun üçüncü taraf paketlerini kullanmanın, temel paketleri spot ışığı kullanmanın esasını araştırıyor ve Django projelerine entegrasyonları için stratejiler sunuyor. Django'nun üçüncü taraf paketlerini kullanmanın faydaları

Django projelerinde üçüncü taraf paketleri benimsemek, canlı topluluklar tarafından titizlikle test edilen çözümlere önemli ölçüde zaman tasarrufu ve erişim sağlar. Bu paketler, en son Django sürümleriyle ve güvenlik protokolleriyle uyumlu olacak şekilde rutin olarak güncellenir. Django için önemli üçüncü taraf paketleri

1. Django Rest çerçevesi

Sağlam API'ler oluşturması gereken geliştiriciler için gerekli olan Django Rest Framework, serileştirme, izin yönetimi ve kimlik doğrulama için kapsamlı araçlar sunar.

Örnek Uygulama:

```
from rest_framework import serializers
from myapp.models import Account

class AccountSerializer(serializers.ModelSerializer):
    class Meta:
        model = Account
        fields = ['id', 'name', 'created', 'owner']

from rest_framework import viewsets

class AccountViewSet(viewsets.ModelViewSet):
    queryset = Account.objects.all()
    serializer_class = AccountSerializer
```

2. Kereviz asenkron görevleri ve gerçek zamanlı operasyonları işlemek için kereviz, birden fazla işçi arasında arka plan görevlerini etkili bir şekilde yönetmek için dağıtılmış mesaj geçirir.

Örnek Uygulama:

```
from celery import Celery

app = Celery('tasks', broker='pyamqp://guest@localhost//')

@app.task
def add(x, y):
    return x + y
```

3. Django Allauth Bu kapsamlı araç seti, üçüncü taraf (sosyal) hesap kimlik doğrulaması da dahil olmak üzere kullanıcı kimlik doğrulamasını, kayıtlarını ve hesap yönetimini kolaylaştırır. Kurulum Kılavuzu: 'Allauth', 'Allauth.Account' ve 'Allauth.socialAccount' ı yüklemeli_apps'ınıza ekleyin.

4. Django Hata Ayıklama Araç Çubuğu vazgeçilmez bir geliştirme aracı olan Django hata ayıklama araç çubuğu, uygulamaları optimize etmeye yardımcı olmak için istek/yanıt döngüleri ve performans metrikleri hakkında ayrıntılı bilgiler sunar.

Kurulum Kılavuzu: 'DEBUG_TOOLBAR' ı yüklemeli_apps'ınıza dahil edin ve projenizin URLConf'a ekleyin.

5. Django Uzantıları Bu paket, ek komut satırı özellikleri ve model alanları ekleyerek, geliştirme ortamını geliştirerek Django'nun yerleşik işlevlerini genişletir.

Kullanım Örneği:

```
python manage.py shell_plus
```

Üçüncü taraf paketlerini Django'ya entegre etmek için
adımlar 1: Paket Seçimi

- Aktif bakımlı paketleri seçin ve düzgün entegrasyon için Django sürümünüzle uyumlu olduklarından emin olun.

2. Adım: Paket kurulumu

- Seçtiğiniz paketi PIP kullanarak yükleyin, örneğin:

```
pip install django-allauth
```

Adım 3: Yüklü uygulamaları güncelleyin

- Paketi Django settings.py'nize Installed_Apps listesine ekleyin.

4. Adım: Yapılandırma ayarlamaları

- Uygun işlevsellik için gerekli olan ayarları veya yapılandırmaları değiştirmek için paketin belgelerini izleyin.

Adım 5: Kapsamlı test yapın

- Paketin entegrasyonunun uygulamanızın işlevselliğini bozmadığını doğrulamak için kapsamlı bir test sağlayın.

Sonuç Üçüncü taraf paketleri, Django projelerinin yeteneklerini genişletmede etkilidir ve geliştiricilerin ortak işlevleri yeniden icat etmek yerine uygulamalarının benzersiz yönlerine odaklanmalarına izin verir. Bu araçların uygun şekilde entegre edilmesi, Django uygulamalarının özellik setini, performansını ve güvenliğini büyük ölçüde artırabilir. Uygulamanın performansını ve sürdürülebilirliğini engellemek yerine bu entegrasyonların geliştirilmesini sağlamak için dikkatli kurulum prosedürlerini ve kapsamlı testleri takip etmek çok önemlidir.

Sosyal kimlik doğrulamasını entegre etmek

Uygulama sosyal kimlik doğrulama içincağ başvuru önemli ölçüde Oturum açma işlemini basitleştirerek kullanıcı deneyimini geliştirir. Kullanıcılar, hizmetlere erişmek için gereken adımları azaltan Facebook, Google veya Twitter gibi mevcut sosyal medya hesaplarını kullanarak giriş yapabilir. Bu makale, sosyal kimlik doğrulamasını Django Allauth paketini kullanmaya odaklanarak bir Django web uygulamasına entegre etmek için teknik hususları incelemektedir.

Sosyal kimlik doğrulamasının faydaları Sosyal kimlik doğrulaması, geleneksel giriş sistemlerine göre çeşitli avantajlar sunar:

1. Kolaylık: Kullanıcılar, kayıt formlarını doldurma ihtiyacından kaçınarak mevcut sosyal ağ kimlik bilgilerini kullanarak uygulamalarda oturum açabilirler.

2. Daha düşük giriş engelleri: Birden çok şifreyi hatırlama ihtiyacının ortadan kaldırılması, kullanıcı sürtünmesini azaltır ve kullanıcı kayıt dönüşüm oranlarını artırabilir.

3. Gelişmiş güvenlik ve güven: Yerleşik sosyal medya platformlarını kimlik doğrulama sağlayıcıları olarak kullanmak, başvurunuzun güvenliğini ve güvenilirliğini artırabilir.

Bir Django Sosyal Kimlik Doğrulama Paketi Seçme Django için bir sosyal kimlik doğrulama paketi seçerken, Django uyumluluğu, bakım durumu, topluluk desteği ve desteklenen sosyal platformların çeşitliliği gibi faktörleri göz önünde bulundurun. Django Allauth, kapsamlı özellik seti, birden fazla sosyal platform desteği, aktif bakım ve kapsamlı belgeler nedeniyle yaygın olarak önerilmektedir.

Django Allauth 1. Kurulumu
entegre etmek

Django Allauth'u PIP aracılığıyla kurarak başlayın:

```
pip install django-allauth
```

2. Yapılandırma

```
INSTALLED_APPS = [
    # Other apps
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    # Providers such as Google, Facebook, Twitter
    'allauth.socialaccount.providers.google',
    'allauth.socialaccount.providers.facebook',
    'allauth.socialaccount.providers.twitter',
]
```

Siteyi ayarlayın_id:

```
SITE_ID = 1
```

Allauth'un URL'lerini projenizin URL yapılandırmasına ekleyin:

```
urlpatterns = [
    path('accounts/', include('allauth.urls')),
    # Other paths
]
```

3. Veritabanı Geçişi Allauth
için geçiş uygular:

```
python manage.py migrate
```

Sosyal Uygulamalar Ayarlama Sosyal kimlik doğrulamasını etkinleştirmek için, istenen sosyal platformlarda uygulamalar ayarlayın:

1. Uygulama kaydı: Sosyal platformun geliştirici bölümünü ziyaret edin ve ad, web sitesi URL'si ve yönlendirme URL'si gibi gerekli ayrıntıları sağlayarak başvurunuzu kaydedin.

2. API Keys Alın: Kayıt sonrası, istemci kimliği ve istemci gizli anahtarları alacaksınız.

3. Django'da yapılandırın: Sosyal uygulamalar altında Django yöneticisine bu anahtarları girin. Sitenize karşılık gelen doğru siteye atayın.

Uygulama örneği, örneğin, Facebook kimlik doğrulamasını ayarlamak şunları içerir: a. Sosyal Uygulama Kurulumu: Facebook Geliştirici Portalına gidin, uygulamanızı oluşturun ve API anahtarını ve API Secret'i alın. B. Django Yapılandırması: Django yöneticisinde yeni bir sosyal uygulama ekleyin, sağlayıcı olarak Facebook'u seçin ve müşteri kimliğinizi ve istemci sırrınızı girin. Kimlik Doğrulama Verilerini Yönetme

Bir kullanıcı bir sosyal hesap üzerinden oturum açtığında, Django Allauth kimlik doğrulamasını işler ve kullanıcı verilerini sağlar. Bu verileri Django Allauth'un sinyallerine bağlayarak yönetebilirsiniz:

```python
from allauth.account.signals import user_logged_in
from django.dispatch import receiver

@receiver(user_logged_in)
def retrieve_social_data(request, user, **kwargs):
    # Example: Output user information
    print('User logged in:', user.username)
```

Sonuç Django Allauth kullanarak sosyal kimlik doğrulamayı dahil etmek, kullanıcı katılımını geliştirir ve iyi kurulmuş sosyal medya kimliklerinden yararlanarak giriş sürecini düzene aktarır. Yukarıda belirtilen adımları izleyerek ve Django Allauth'un kapsamlı yeteneklerini kullanarak, geliştiriciler güvenli ve kullanışlı bir giriş deneyimi sağlayabilir, kullanıcı güvenini ve uygulama güvenilirliğini teşvik edebilir. Bu entegrasyonun başarılı olmasını sağlamak ve performans veya güvenlikten ödün vermeden uygulamanın işlevselliğini artırmasını sağlamak için uygun kurulum, yapılandırma ve test esastır.

Pratik Örnekler: Google Giriş Ekleme

Google girişini bir web uygulamasına dahil etmek, kullanıcıların mevcut Google hesaplarını kullanarak hizmetlere erişmelerine izin vererek kimlik doğrulama işlemini büyük ölçüde basitleştirir. Bu kolaylık, kullanıcı etkileşimlerini kolaylaştırarak birden fazla hesap kimlik bilgilerine olan ihtiyacı ortadan kaldırır. Bu kılavuz, Django Allauth paketinden yararlanarak Google Giriş'in Django Web uygulamasına nasıl uygulanacağını detaylandırır. Google Giriş Entegre'nin Avantajları

Google Giriş Uygulamak çeşitli avantajlar sunar:

1. Erişim Kolaylığı: Kullanıcıların Google hesaplarını giriş için kullanmalarını, geleneksel kayıt süreçlerini atlayabilmelerini sağlar.

2. Güvenlik: Almak avantaj ile ilgili Google's sağlam güvenlik Kullanıcı verilerini korumak için çerçeve.

3. Kullanıcı Tutma: Kullanıcı deneyimini basitleştirir, potansiyel olarak artan tutma ve dönüşüm oranları.

Google Kimlik Doğrulaması için Django Allauth'u Ayarlama 1. Kurulum

İlk olarak, Django Allauth'un Django projenize kurulduğundan emin olun:

```
pip install django-allauth
```

2. Ayarları Ayarları Yapılandırma
Settings.py dosyası, katmak Django Alauth içindsenin kurulu Uygulamalar:

```
INSTALLED_APPS = [
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'allauth.socialaccount.providers.google',  # Activate Google integration
]
```

Site_id'i tanımlayın ve URL'inize Allauth URL yapılandırmalarını ekleyin.

```
urlpatterns = [
    path('accounts/', include('allauth.urls')),
    # Other URLs
]
```

3. Veritabanı Geçişi, Allauth'un modellerini içerecek şekilde veritabanı şemalarını güncellemek için geçişleri yürütür:

```
python manage.py migrate
```

Kimlik Doğrulama İçin Google API'sini Yapılandırma Google ile uygulamanızı kimlik doğrulama için kullanmak için ayarlayın: 1. Google Geliştirici Konsolu

- Google geliştirici konsoluna erişin.

- Mevcut bir proje seçin veya yeni bir proje oluşturun. "Kimlik bilgileri" altında, "OAuth İstemci Kimliği" oluşturmayı ve OAuth
- onam ekranını buna göre yapılandırmayı seçin.

2. API Keys'i Alın

- Uygulama türünüzü "web uygulaması" olarak tanımlayın. Geliştirme veya üretim sitenizi yetkili JavaScript kökenlerine
- ekleyin ve URI'leri yönlendirin (örn., Http: // localhost: 8000/hesaplar/google/login/geri arama/).

3. Django'yu API Keys ile güncelleyin Django ayarlarınızda Google API anahtarlarını ekleyin:

```
SOCIALACCOUNT_PROVIDERS = {
    'google': {
        'APP': {
            'client_id': 'your-client-id',
            'secret': 'your-client-secret',
            'key': ''
        }
    }
}
```

Google Girişini Django'da Yapılandırma Seti ile Uygulama, Google Oturum Açma İşlevselliği Entegre: 1. Şablona Giriş Bağlantısı Ekle

Google kimlik doğrulaması için şablonlarınıza bir bağlantı yerleştirin:

```
<ul>
    <li><a href="{% url 'google_login' %}?process=login">Login with Google</a></li>
</ul>
```

2. Kullanıcı-lojin sonrası eylemleri özelleştirme, Django Allauth sinyallerine bağlanarak kullanıcı girişinden sonra eylemleri özelleştirin:

```python
from allauth.account.signals import user_logged_in
from django.dispatch import receiver

@receiver(user_logged_in)
def post_login_actions(request, user, **kwargs):
    # Logic to handle user data after login
    pass
```

Sonuç Django Allauth kullanarak Django uygulamanıza Google Giriş eklemek, Google'ın güvenli ve yaygın altyapısını kullanarak kimlik doğrulama deneyimini geliştirir. Google API ve Django ayarlarını yapılandırmak için ana hatlarıyla belirtilen adımları izlemek, geliştiricilerin bu özelliği verimli bir şekilde etkinleştirmesini sağlar ve kullanıcılara uygulamalarına erişmek için sorunsuz ve güvenli bir yöntem sunar.

YAPANLAR:
UMUT TÜRKER
AYŞE KOLUTEK
EBUBEKİR POLAT
HİKMET CAN UMUTLU