Opsiyonel: Bağımsız Bir Django ORM Proje Şablonu Oluşturma

Bu laboratuvar çalışmasında, daha fazla Django ORM uygulaması geliştirmek için bir şablon olarak kullanabileceğiniz bağımsız bir Django ORM projesi oluşturacaksınız.

Gerekli tahmini süre: 20 dakika

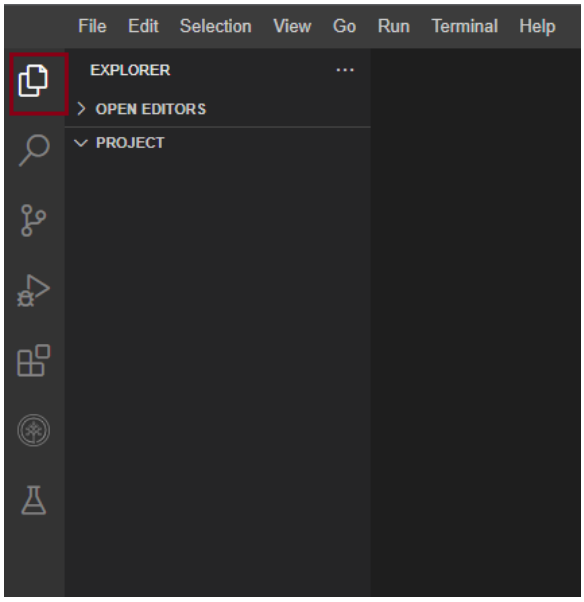
Öğrenme Hedefleri

- Bağımsız bir Django ORM projesi ve uygulaması oluşturun
- Bu projeyi, daha karmaşık Django ORM uygulamaları öğrenmek ve geliştirmek için bir şablon olarak kaydedin

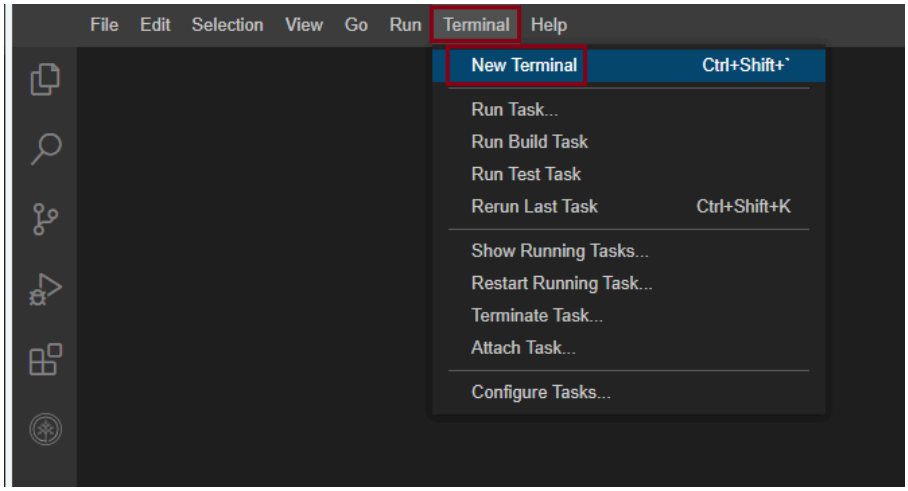
Bulut IDE'de dosyalarla çalışma

Bulut IDE'ye yeniyseniz, bu bölüm size projenizin bir parçası olan dosyaları nasıl oluşturup düzenleyeceğinizi gösterecektir.

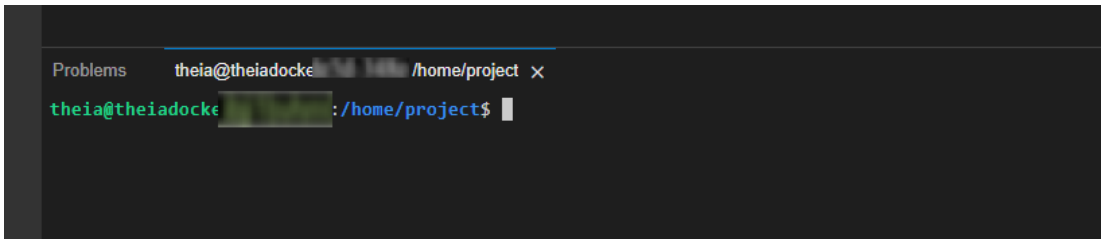
Bulut IDE içindeki dosyalarınızı ve dizinlerinizi görüntülemek için bu dosya simgesine tıklayarak açın.



`Yeni`ye tıklayın, ardından `Yeni Terminal` seçeneğine tıklayın.



Bu, komutlarınızı çalıştırabileceğiniz yeni bir terminal açacaktır.



Laboratuvarda Kapsanan Kavramlar

1. `Django ORM`: Her Django modelinin bir veritabanı tablosuna karşılık geldiği Django web uygulama çerçevesinin bir Python ORM bileşeni.
2. `PostgreSQL` veya `Postgres`: Django tarafından kullanılan açık kaynaklı bir ilişkisel veritabanı yönetim sistemi.
3. `Psycopg`: Django'nun PostgreSQL ile çalışmak için kullandığı bir arayüz.
4. `Veritabanı göçleri`: İlişkisel veritabanı şemalarına yönelik sürüm kontrollü, artımlı ve geri alınabilir değişikliklerin yönetimi, şemanın daha yeni veya daha eski bir sürüme güncellenmesi veya geri alınması için.
5. `manage.py`: Bir Django projesini yönetmek için kullanılan bir komut satırı arayüzü.

Theia'da PostgreSQL'i Başlat

Eğer Skills Network Labs tarafından barındırılan Theia ortamını kullanıyorsanız, aşağıdaki buton ile başlatabileceğiniz önceden kurulmuş bir `PostgreSQL` örneği sağlanmaktadır:

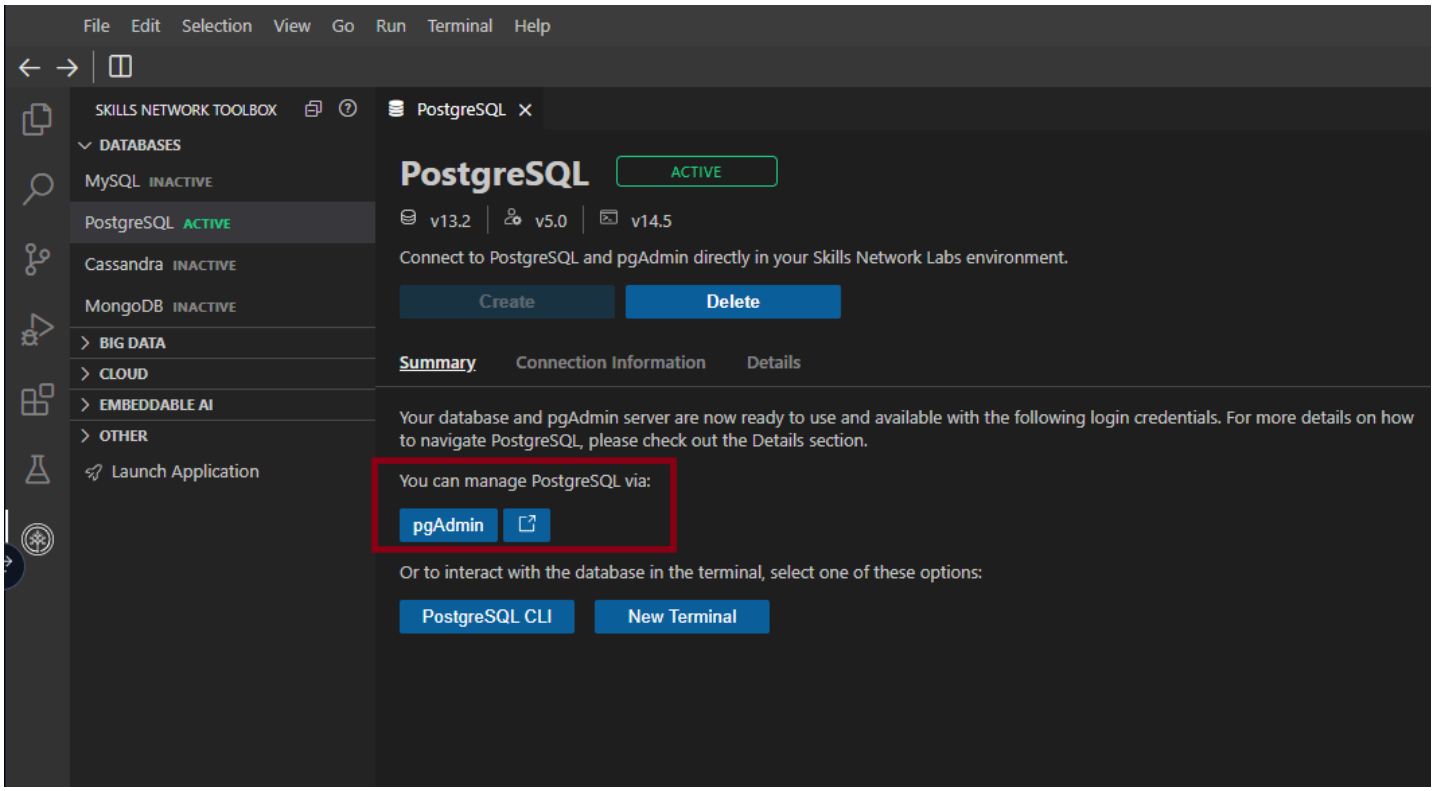
Open and Start PostgreSQL in IDE

Eğer bunu önceki laboratuvarlarda zaten başlattıysanız bu adımı atlayabilirsiniz.

- PostgreSQL başlatıldıktan sonra, sunucu bağlantı bilgilerini *Bağlantı Bilgileri* sekmesinde kontrol edebilirsiniz. Django uygulamasının bu veritabanına bağlanabilmesi için gereken `kullanıcı adı`, `şifre` ve `host` gibi bağlantı bilgilerini kaydetmeniz gerekmektedir.
- Postgres'e erişim için ortamı kurmadan önce bu gerekli paketleri yükleyin.

```
pip install --upgrade distro-info
pip3 install --upgrade pip==23.2.1
```

- Bir `pgAdmin` örneği de sizin için kuruldu ve başlatıldı.



Basit Bir Django ORM Projesi Oluşturma

Eğer terminal açık değilse, `Terminal > Yeni Terminal` seçeneğine gidin ve mevcut Theia dizininizin `/home/project` olduğundan emin olun.

Bu laboratuvar çalışmasında, komut satırı araçları kullanarak tam bir Django web projesi ve uygulaması oluşturmak yerine, yalnızca ORM bileşenine sahip basit bir uygulama oluşturacaksınız.

ORM-only uygulaması içinde, modellerinizi tanımlayabilir ve model nesneleriniz üzerinde CRUD işlemlerini kolayca gerçekleştirebilirsiniz. Daha da önemlisi, bunu yapmak için Python kodunu satır satır shell'e yazmak yerine Python script dosyaları oluşturup çalıştırabilirsiniz.

Boş bir proje klasörü oluşturarak başlayalım.

- Theia menüsünde, `Dosya > Yeni Klasör` seçeneğine tıklayın ve boş bir Django projesi için kapsayıcı klasör olarak `ormtemplate` yazın.
- `ormtemplate` klasörüne sağ tıklayın ve boş bir Django uygulaması olan `standalone` için kapsayıcı klasör olarak bir alt klasör oluşturun.
- `ormtemplate` klasörüne tekrar sağ tıklayın ve boş bir `settings.py` dosyası ile bir `manage.py` dosyası oluşturun.

Sonraki adımda, Django projemizi yöneten bir komut satırı arayüzü işlevi görecek `manage.py` dosyasına içerik ekleyeceğiz.

- Boş `manage.py` dosyasını açın ve aşağıdaki kod parçasını kopyalayıp yapıştırın.

```
import os
import sys
if __name__ == "__main__":
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError:
        try:
            import django
        except ImportError:
            raise ImportError(
                "Couldn't import Django. Are you sure it's installed and "
                "available on your PYTHONPATH environment variable? Did you "
                "forget to activate a virtual environment?"
            )
        raise
    execute_from_command_line(sys.argv)
```

Kod parçası, `ormtemplate` projemizin ayar modülü olarak `manage.py` dosyasına eklenmeli ve Django'nun yerleşik komutları olan göçler gibi komutları çalıştırabilmelidir.

Sonraki adımda, `settings.py` dosyasına basit bir veritabanı ayarı ekleyelim.

- Boş `settings.py` dosyasını açın, aşağıdaki kod parçasını kopyalayıp yapıştırın.

```
# PostgreSQL
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'postgres',
        'USER': 'postgres',
        'PASSWORD': 'Place with your password generated in Step 1',
        'HOST': 'postgres',
        'PORT': '5432',
    }
}
INSTALLED_APPS = (
    'standalone',
)
SECRET_KEY = 'SECRET KEY for this Django Project'
```

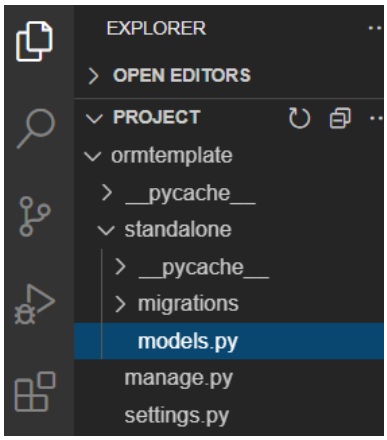
Yukarıdaki kod parçası `standalone` uygulamasını yüklü bir uygulama olarak ekler ve varsayılan veritabanı olarak 1. Adımda oluşturduğumuz önceden yüklenmiş PostgreSQL'i ayarlar.

Tek yapmanız gereken `PASSWORD` alanını 1. Adımda oluşturulan şifre ile güncellemektir.

Şu ana kadar çok basit bir Django projesi oluşturduk. Şimdi `standalone` uygulamamıza içerik ekleyelim.

- `ormtemplate/standalone` klasörüne tıklayın ve model tanımlamalarını içeren boş bir `models.py` dosyası ile `standalone` uygulamasında göç scriptlerini içeren bir `migrations` adlı klasör oluşturun.

Bundan sonra, uygulama yapınız aşağıdaki gibi görünmelidir:



- Artık Django `standalone` uygulamanız hazır ve `standalone` uygulamasının çalışıp çalışmadığını test etmeye başlayabilirsiniz.

Bağımsız Uygulamayı Test Et

- `models.py` dosyasını açın ve basit bir `Test` modeli tanımlamak için aşağıdaki kod parçacığını kopyalayıp yapıştırın.

```
from django.db import models
# Test model
class Test(models.Model):
    name = models.CharField(max_length=30)
```

Sonra, `ormtemplate` uygulamasına `Test` tablosunu oluşturması için göç komut satırlarını çalıştırabiliriz.

- `cd` komutuyla `ormtemplate` klasörüne geçin.

```
cd ormtemplate
```

ve şimdi mevcut Theia klasörünüz terminalde `/home/project/ormtemplate` olarak görünmelidir.

Gerekli tüm paketleri içerecek bir sanal ortam oluşturalım.

```
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
```

Gerekli paketleri yükleyin.

```
pip install django==4.2.4 psycopg2-binary==2.9.7
```

- Ardından, `standalone` uygulaması için göç betikleri oluşturmanız gerekiyor.

  ```
  python3 manage.py makemigrations standalone
  ```

ve `Test` modeliniz için oluşturulan göç betiklerini görmelisiniz.

```
Migrations for 'standalone':
  standalone/migrations/0001_initial.py
    - Create model Test
```

Not: Eğer aşağıdaki gibi hatalarla karşılaşırsanız

```
django.db.utils.OperationalError: FATAL: password authentication failed for user "postgres"
```
lütfen `start_postgres` komutunu tekrar çalıştırarak `PostgreSQL` sunucusunu sıfırlayın ve `settings.py` dosyasında yeni şifreyi kullanın.

- ve göç işlemini çalıştırın

  ```
  python3 manage.py migrate
  ```

Sonra, modelinizi test etmek için bir Python script dosyasında (`*.py`) bazı Python test kodları yazabilirsiniz.

- `ormtemplate` klasörüne tıklayın ve `test.py` adında yeni bir dosya oluşturun.
- Boş `test.py` dosyasını açın ve `test` modelinizi test etmek için aşağıdaki kod parçacığını ekleyin:

  ```
  # Django specific settings
  import inspect
  ```

```
import os
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "settings")
from django.db import connection
# Ensure settings are read
from django.core.wsgi import get_wsgi_application
application = get_wsgi_application()
# Your application specific imports
from standalone.models import Test
# Delete all data
def clean_data():
    Test.objects.all().delete()
# Test Django Model Setup
def test_setup():
    try:
        clean_data()
        test = Test(name="name")
        test.save()
        # Check test table is not empty
        assert Test.objects.count() > 0
        print("Django Model setup completed.")
    except AssertionError as exception:
        print("Django Model setup failed with error: ")
        raise(exception)
    except:
        print("Unexpected error")
test_setup()
```

Yukarıdaki kod parçası önce veritabanını temizler ve ardından bir test nesnesi ekler. Daha sonra test nesnesinin doğru bir şekilde eklenip eklenmediğini kontrol eder.

- Son olarak, terminalde `test.py` dosyasını çalıştırın:

  ```
  python3 test.py
  ```

ve görmelisiniz

Django Model kurulumu tamamlandı.

Artık bağımsız bir Django ORM uygulaması başarıyla oluşturduk ve basit bir `Test` modeli ile test ettik. Ayrıca, sıfırdan oluşturarak Django uygulama yapısına aşina oldunuz.

Sonraki adımda, bu projeyi yerel olarak indirip gelecekteki öğrenme ve Django ORM geliştirme faaliyetleriniz için bir şablon olarak kaydedebilirsiniz.

- Kök klasör `ormtemplate` üzerine sağ tıklayın ve çalışma alanınızı kaydetmek için `İndir` seçeneğine tıklayın.

Uygulamanız için, bunu yeni bir Theia ortamına veya daha karmaşık Django ORM modelleri geliştirmek için başlangıç noktası olarak yerel Python ortamınıza aktarabilirsiniz.

Özet

Bu laboratuvar çalışmasında, tam bir Django web projesi oluşturmadan bağımsız bir Django ORM uygulaması yarattınız. Bu basit ORM uygulamasını, Django ORM'yi öğrenmek ve gelecekte daha karmaşık Django ORM uygulamaları geliştirmek için bir şablon olarak kullanabilirsiniz.

Yazar(lar)

Yan Luo