Sorgu Kapsayan İlişkiler

Gerekli tahmini süre: 15 dakika

Bu laboratuvar çalışmasında, ilişkileri kapsayan ilgili nesneleri sorgulamayı ve erişmeyi öğreneceksiniz.

Öğrenme Hedefleri

- İlişkileri kapsayan ilgili nesneleri sorgulama

Cloud IDE'de dosyalarla çalışma

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

Cloud IDE içindeki dosyalarınızı ve dizinlerinizi görüntülemek için bu dosya simgesine tıklayın.

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

Bu, komutlarınızı çalıştırabileceğiniz yeni bir terminal açacaktır.

Laboratuvar Kapsamındaki Kavramlar

1. `QuerySet`: Bir veritabanındaki kayıtların bir koleksiyonunu temsil eder ve Django Model API'sini kullanarak nesneleri okumak için gereklidir.
2. `get()` metodu: Arama kriterine uyan tek bir nesneyi döner.
3. `all()` metodu: Veritabanındaki tüm nesnelerin bir QuerySet'ini döner.
4. `filter()` metodu: Sadece arama terimiyle eşleşen satırları döner. **büyüktür**, **küçüktür**, **içerir** veya **nulldır** gibi arama parametrelerine sahip olabilir.
5. `İleri ilişki`: Bir modelin bir Yabancı anahtarı varsa, o modelin örnekleri, modelin bir niteliği aracılığıyla ilişkili (yabancı) nesneye erişim sağlar.
6. `Geri ilişki`: Bir modelin bir Yabancı anahtarı varsa, yabancı anahtar modelinin örnekleri, ilk modelin tüm örneklerini dönen bir Yöneticiyi erişim sağlar.

Theia'da PostgreSQL Başlatma

- Eğer Skills Network Labs tarafından barındırılan Theia ortamını kullanıyorsanız, sol menü çubuğundaki SkillsNetwork simgesini bulup DATABASES menüsünden PostgreSQL'i seçerek önceden yüklenmiş `PostgreSQL` örneğini başlatabilirsiniz:

- PostgreSQL başlatıldıktan sonra, sunucu bağlantı bilgilerini UI'dan kontrol edebilirsiniz. Django uygulamanızın bu veritabanına bağlanması için kullanılacak `username`, `password`, `host` gibi bağlantı bilgilerini markdown formatında not edin.

- PostgreSQL'e erişim için ortamı kurmadan önce bu gerekli paketleri yükleyin.

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

- Ayrıca bir `pgAdmin` örneğinin yüklendiğini ve başlatıldığını göreceksiniz.

Önceden Tanımlı Modellerle Django ORM Projesi İçe Aktarma

Laboratuvara başlamadan önce, mevcut Theia dizininizin `/home/project` olduğundan emin olun.

Öncelikle Django ile ilgili paketleri yüklememiz gerekiyor.

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

- Laboratuvar 3 için bir kod şablonu indirmek üzere aşağıdaki komutları çalıştırın.

```
wget "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-CD0251EN-SkillsNetwork/labs/m3_django_orm/lab3_template.zip"
unzip lab3_template.zip
```

```
rm lab3_template.zip
```

Django projeniz aşağıdaki gibi görünmelidir:

- `settings.py` dosyasını açın ve `DATABASES` bölümünü bulun. `PASSWORD` değerini, oluşturulan `PostgreSQL` parolası ile değiştirin.

Artık `settings.py` dosyanız aşağıdaki gibi görünmelidir:

Bir `onlinecourse` Uygulaması için Modelleri Aktif Hale Getir

- `related_objects/models.py` dosyasını açın, laboratuvar `CRUD on Django Model Objects`'ta oluşturduğunuz online kurs uygulaması için aynı modelleri bulabilirsiniz.

Bu modellerin ilişkilerini özetlemek gerekirse:

- `Learner` ve `Instructor` modelleri `User` modelinden One-To-One ilişkisi ile türetilmiştir.
- `Lesson`, `Course` ile One-To-Many ilişkisine sahiptir.
- `Instructor`, `Course` ile Many-To-Many ilişkisi vardır.
- `Learner`, `Enrollment` aracılığıyla `Course` modeli ile Many-To-Many ilişkisine sahiptir.

Şimdi, bu modelleri aktifleştirmek için göçleri çalıştıralım.

- Eğer mevcut çalışma dizininiz `/home/project/lab3_template` değilse, proje klasörüne `cd` komutuyla geçin.

```
cd lab3_template
```

Gerekli tüm paketlerin bulunduğu bir sanal ortam oluşturalım.

```
pip install virtualenv
virtualenv djangoenv
source djangoenv/bin/activate
pip install django==4.2.4 psycopg2-binary==2.9.7
```

- Ardından `related_objects` uygulaması için göç betikleri oluşturun.

```
python3 manage.py makemigrations related_objects
```

ve Django'nun aşağıdaki tabloları oluşturmak üzere olduğunu görmelisiniz.

```
Migrations for 'related_objects':
  crud/migrations/0001_initial.py
    - Create model Course
    - Create model User
```

```
    - Create model Instructor
    - Create model Learner
    - Create model Lesson
    - Create model Enrollment
    - Add field instructors to course
    - Add field learners to course
```

- sonra göçü çalıştırın

```
python3 manage.py migrate
```

ve göreceksiniz ki `related_objects.0001_initial` göç betiği çalıştırıldı.

```
Operations to perform:
  Apply all migrations: related_objects
Running migrations:
  Applying related_objects.0001_initial... OK
```

Veri İlişkileri ile Doldurma

Bu aşamada, tüm modelleri taşıdınız. Şimdi nesneleri ilişkilerle doldurmayı ve bunları veritabanına kaydetmeyi deneyelim.

`lab3_template/write.py` betiği öncelikle `CRUD on Django Model Objects` laboratuvarında aynı model nesnelerini oluşturur.

Buna ek olarak, referans alanlarını nesnelerle güncelleyerek ilişkiler oluşturmak için iki `populate_course_instructor_relationships()` ve `populate_course_enrollment_relationships()` yöntemi ekler.

- Şimdi çalıştırın

```
python3 write.py
```

Terminalde kaydedilmiş mesajlar, nesneleri ve ilişkileri görmelisiniz.

```
Course objects saved...
Instructors objects saved...
Learners objects saved...
Lessons objects saved...
Course-instructor relationships saved...
Course-learner relationships saved...
```

Kodlama pratiği: Daha Fazla Nesne ve İlişki Oluşturun

Lütfen `write.py` dosyasındaki örnekleri gözden geçirin ve ilişkileri olan daha fazla nesne oluşturun.

Sorgulama span ilişkilerini oluşturma

Veriler doldurulduktan sonra, bunları sorgulamaya başlayabiliriz.

Bu laboratuvar, span ilişkilerini sorgulamaya odaklanacak.
Örneğin, bir kurs için tüm eğitmenleri almak veya belirli bir kursa kayıtlı öğrenicileri almak.

- Aşağıdaki senaryolar için ilişkiler arasında nesneleri sorgulamaya başlayalım:

1. Eğitmen `Yan` tarafından verilen kursları, hem ileri (açık) hem de geri (örtük) erişim ile al
2. Cloud uygulama geliştirme kursunun eğitmenlerini al
3. Eğitmen `Yan` tarafından verilen kursların mesleklerini kontrol et

- `read_course_instructor.py` dosyasını açın, aşağıdaki sorguları ekleyin:

```
# Course has instructors reference field so can be used directly via forward access
courses = Course.objects.filter(instructors__first_name='Yan')
print("1. Get courses taught by Instructor `Yan`, forward")
print(courses)
print("\n")
# For each instructor, Django creates a implicit course_set. This is caleld backward access
instructor_yan = Instructor.objects.get(first_name='Yan')
print("1. Get courses taught by Instructor `Yan`, backward")
print(instructor_yan.course_set.all())
print("\n")
instructors = Instructor.objects.filter(course__name__contains='Cloud')
print("2. Get the instructors of Cloud app dev course")
print(instructors)
print("\n")
courses = Course.objects.filter(instructors__first_name='Yan')
occupation_list = set()
for course in courses:
    for learner in course.learners.all():
        occupation_list.add(learner.occupation)
print("3. Check the occupations of the courses taught by instructor Yan'")
print(occupation_list)
```

Yukarıdaki kod parçası, ilişkili nesnelere hem ileri hem de geri erişim yoluyla erişir.
Ayrıca, `Course`'dan `Instructor`'a gibi ilişkiler boyunca arama yapmak için sorgu parametreleri ile ilişkili nesneleri sorgular.

- Sorguları çalıştırın ve sonuçları kontrol edin.

```
python3 read_course_instructors.py
```

- Sorgu sonuçları:

```
1. Get courses taught by Instructor `Yan`, forward
<QuerySet [<Course: Name: Cloud Application Development with Database,Description: Develop and deploy application on cloud>]>
1. Get courses taught by Instructor `Yan`, backward
<QuerySet [<Course: Name: Cloud Application Development with Database,Description: Develop and deploy application on cloud>]>
2. Get the instructors of Cloud app dev course
<QuerySet [<Instructor: First name: Yan, Last name: Luo, Is full time: True, Total Learners: 30050>, <Instructor: First name: Joy, Last name: Li
3. Check the occupations of the courses taught by instructor Yan'
{'dba', 'data_scientist', 'developer'}
```

Kodlama pratiği: `Course`, `Learner` ve `User` Nesneleri ile İlgili Sorgular

`read_enrollments.py` dosyasını açın ve `Course`, `Learner` ve `User` için aşağıdaki ilişkisel sorguları tamamlayın.

1. Öğrenci `David` hakkında kullanıcı bilgilerini alın.
2. Kullanıcıdan öğrenci `David` bilgilerini alın.
3. `Introduction to Python` kursu için tüm öğrencileri alın.
4. Eğitmen `Yan` tarafından öğretilen kurslar için meslek listesini kontrol edin.
5. Geliştirici öğrencilerin Ağustos 2020'de hangi kurslara kayıtlı olduğunu kontrol edin.

```
print("1. Get the user information about learner `David`")
learner_david = Learner.objects.get(first_name="David")
print( #<HINT> use the usr_ptr field created by Django for the Inheritance relationship# )
print("2. Get learner `David` information from user")
user_david = User.objects.get(first_name="David")
print( #<HINT> use the learner field created by Django# )
print("3. Get all learners for `Introduction to Python` course")
course = Course.objects.get(name='Introduction to Python')
learners = #<HINT> use the learners field in Course model#
print(learners)
print("4. Check the occupation list for the courses taught by instructor `Yan`")
courses = Course.objects.filter( #<HINT> query the first name of instructor# )
occupation_list = set()
for course in courses:
    for learner in #<HINT>use the learners field in Course Model#  :
        occupation_list.add(learner.occupation)
print(occupation_list)
print("5. Check which courses developers are enrolled in Aug, 2020")
enrollments = Enrollment.objects.filter(date_enrolled__month=8,
                                        date_enrolled__year=2020,
                                        #<HINT>use the occupation field from learner #)
courses_for_developers = set()
for enrollment in enrollments:
    course = enrollment.course
    courses_for_developers.add(course.name)
print(courses_for_developers)
```

▼ Çözümü görmek için buraya tıklayın

```
learner_david = Learner.objects.get(first_name="David")
print("1. Get the user information about learner `David`")
print(learner_david.user_ptr)
user_david = User.objects.get(first_name="David")
print("2. Get learner `David` information from user")
print(user_david.learner)
course = Course.objects.get(name='Introduction to Python')
learners = course.learners.all()
print("3. Get all learners for `Introduction to Python` course")
print(learners)
courses = Course.objects.filter(instructors__first_name='Yan')
occupation_list = set()
for course in courses:
    for learner in course.learners.all():
        occupation_list.add(learner.occupation)
print("4. Check the occupation list for the courses taught by instructor `Yan`")
print(occupation_list)
enrollments = Enrollment.objects.filter(date_enrolled__month=8,
                                        date_enrolled__year=2020,
                                        learner__occupation='developer')
courses_for_developers = set()
for enrollment in enrollments:
    course = enrollment.course
    courses_for_developers.add(course.name)
print("5. Check which courses developers are enrolled in Aug, 2020")
print(courses_for_developers)
```

Sorgu sonuçları:

```
1. Get the user information about learner `David`
David Smith
2. Get learner `David` information from user
First name: David, Last name: Smith, Date of Birth: 1983-07-16, Occupation: developer, Social Link: https://www.linkedin.com/david/
3. Get all learners for `Introduction to Python` course
<QuerySet [<Learner: First name: Robert, Last name: Lee, Date of Birth: 1999-01-02, Occupation: student, Social Link: https://www.facebook.com/r
4. Check the occupation list for the courses taught by instructor `Yan`
{'data_scientist', 'developer', 'dba'}
5. Check which courses developers are enrolled in Aug, 2020
{'Cloud Application Development with Database'}
```

▶ Detayları Göster

::page{title=”Özet”}

Bu laboratuvarda, nesneleri sorgulama ve ilişkili nesnelere, açık ileri erişim ve örtük geri erişim yoluyla erişim sağlamayı öğrendiniz.

Author(s)

Yan Luo