Django ORM

أشهر أنواع قواعد البيانات

- قواعد البيانات المبنية على العلاقات (Relational Databases)
هي تلك التي تحفظ البيانات على هيئة جداول (أعمدة و صفوف) و تعتمد في تصميمها على العلاقات ما بين الجداول المختلفة .
و تعد أشهر انواع قواعد البيانات و تستخدم بشكل واسع جدا .

- قواعد البيانات المبنية على الوثائق (Document based Databases)
هي قواعد البيانات التي تعتمد في عملها على تخزين البيانات على شكل وثائق مبنية بالعادة بـالـ JSON . اكتسبت شهرة كبيرة مؤخرا بسبب مرونتها و سهولة الإستخدام .

- قواعد البيانات المعتمدة على المفاتيح و القيم (Key-Value Databases)
يتم تخزين البيانات في هذا النوع من قواعد البيانات على هيئة مفتاح و قيمة مرتبطة بذلك المفتاح . و تعد احدى ابسط انواع قواعد البيانات .

ما هو الـ ORM ؟

هي اختصار لـ Object Relational Mapping و هي أداة تعمل على تمثيل البيانات من قواعد البيانات على هيئة كائنات (objects) . نشأت هذه الادوات من اجل تسهيل عمل المطورين في التعامل مع قواعد البيانات ، و ايضا من اجل تسهيل اجراء العمليات بإستخدام لغات البرمجة بدلا من استخدام الـ SQL مباشرة . بحيث ، يصبح بإمكان اي مبرمج التعامل مع قواعد البيانات من دون الحاجة لتعلم الـ SQL و لو أنه يُفضل وجود معرفة بها.

كمثال ، لدينا هنا جدول من قاعدة بيانات Relational تحت اسم Movie

rating	description	name	id
.....	.....	.....	1
.....	.....	.....	2

بإستخدام الـ ORM يتم تمثيل هذا الجدول السابق على هيئة Class يحمل اسم الجدول Movie و يحتوي properties (مزايا Attributes) تمثل الأعمدة في قاعدة البيانات تلك :

لاحظ: الكود التالي فقط للتوضيح و ليس حقيقي (sudo code)

```
Class Movie:
  id = 1
  name = "...."
  description = "...."
  rating = "...."
```

بحيث يتم لاحقا تنفيذ جميع العمليات المطلوبة على كائنات من هذا الـ Class . من انشاء بيانات جديدة ، تحديث ، حذف ، جلب و عرض ، و كلها تتم من خلال الـ ORM عوضا عن استخدام لغة الـ SQL .

Django ORM -

يوفر لنا اطار عمل الـ Django أداة ORM قوية جدا تساعدنا في التعامل مع قواعد البيانات . و لنتعلم استخدامها ، سنمر بعدة مراحل نتعرف فيها على كل مرحلة و ما هي .

اعدادات قاعدة البيانات

يتم حفظ قيم اعدادات قاعدة البيانات المراد استخدامها في ملف الـ settings.py داخل مجلد المشروع و ذلك تحت المتغير DATABASES. حيث هناك يمكنك تعريف اعدادات قاعدة البيانات من نوع ، البورت ، اسم المستخدم ، كلمة المرور و غيرها . لاحظ انك يمكنك استخدام قاعدة بينات sqlLite الخفيفة و المحلية لغرض التطوير و الإختبار .

لمعلومات اكثر و تفصيل يرجى زيارة :
https://docs.djangoproject.com/en/4.0/ref/settings/#databases

إنشاء Model لتمثيل البيانات

مع انشاء كل تطبيق جديد في مشروع الـ Django ستلاحظ وجود ملف اسمه models.py سنستخدم هذا الملف من اجل انشاء Model يمثل البيانات التي نريدها في قاعدة البيانات .

و لهذا الغرض سنحتاج ان نرث من Model الخاص بالـ Django DB . في ملف الـ models.py ، نفذ التالي :

- انشاء كلاس Movies يرث من الـ models.Model
- انشئنا Attributes تمثل الأعمدة من name , description و rating

- لكل Attribute قمنا بتحديد نوع ذلك العمود من خلال الأنواع الموفرة لنا من الـ Django ROM Model

```
from django.db import models
# Create your models here.

class Movie(models.Model):
    name = models.CharField(max_length=512)
    description = models.TextField()
    rating = models.FloatField()
```

لاحظ هنا أننا لم نزود المودل بـ id معين ، فهذا حقل يتم انشاءه تلقائيا لنا من قبل Django ORM .

أنواع Fields المستخدمة في Model

في Django توجد أنواع Field متعددة يمكن استخدامها لتعريف البيانات داخل في Models ومن أشهرها التالي:

النوع	استخدامه
CharField	يستخدم لتخزين نصوص قصيرة
TextField	يستخدم لتخزين نصوص طويلة
IntegerField	يستخدم لتخزين الأرقام
FloatField	يستخدم لتخزين الأرقام من نوع float
DateField	يستخدم لتخزين بيانات التاريخ
BooleanField	يستخدم لتخزين القيم True و False
EmailField	يستخدم للتحقق مما إذا كان النص تعبر عن بريد إلكتروني أم لا و يتم تخزينه
URLField	يستخدم للتحقق مما إذا كان النص يعبر عن URL صحيح أم لا و من م تخزينه

وللاطلاع على المزيد من أنواع Field عن طريق الرابط:

https://docs.djangoproject.com/en/3.2/ref/models/fields/

عمل Migration

في كل مرة تنشئ فيها Model جديد أو تعدل على مزايا احدها ، نحتاج طريقة من اجل تطبيق هذه التعديلات على قاعدة البيانات ، و هنا يأتي دور الـ migration . بحيث سنستخدم عدة خطوات من اجل هذا الغرض . نبدأها بالتالي :

1- عمل بيانات الـ Migration

سنستخدم الأمر makemigrations من اجل انشاء ملف يحتوي على التعديلات المطلوب تطبيقها في قاعدة البيانات بناء على الـ models الموجودة في مشروعك . قم بفتح الـ command line / Terminal و تأكد انك في مسار المشروع الخاص بك ، من ثم نفذ التالي :

لاحظ: اختياريا يمكنك تحديد اسم (او اسماء) الـ app الذي تريد عمل migration عليه أو ترك مكانه فارغ و سيتم عمل migration لكل التطبيقات الموجودة في مشروعك .

```
python manage.py makemigrations app_name
```

بعد تنفيذ هذا الأمر ستلاحظ انشاء ملف جديد داخل مجلد الـ migrations داخل التطبيقات في مشروعك ، هذا الملف يحمل الإسم initial.py_0001 , هذا الملف يحتوي على التعديلات الجديد التي طرأت على المودل . و في كل مرة تجري تعديلات او تغيرات على المودل في تطبيقك ، سيتم انشاء ملفات جديدة تحمل رقم نسخة أعلى ، بحيث يساعد الـ Django ORM على تتبع التعديلات و تطبيقها على قاعدة البيانات .

2- تطبيق التعديلات على قاعدة البيانات

سنستخدم الأمر migrate من اجل تطبيق التعديلات على قاعدة البيانات . نفذ التالي :
لاحظ: بالإمكان ايضا تزويد الامر بإسم التطبيق من اجل تطبيق التعديلات من تطبيق معين ، او تركه فارغاً لتطبيق جميع التعديلات على جميع التطبيقات في مشروعك.

```
python manage.py migrate
```

هذا الامر سيقوم بتنفيذ جميع التعديلات التي طرأت على المودل في تطبيقاتك على قاعدة البيانات . فسيتم مزامنة حالة البيانات الممثلة في التطبيق على قاعدة البيانات .

أوامر مفيدة من الـ Django في التعامل مع الـ ORM

أمر showmigrations

- يساعد الأمر [python manage.py showmigrations] يعرض جميع التعديلات التي حدثت على database التابعة لكل تطبيق

أمر makemigrations

- يقوم الأمر [python manage.py makemigrations App_Name] باستعراض جميع التعديلات التي سوف تحدث على قواعد البيانات عن طريق بإنشاء migration scripts للتطبيقات الموجودة في المشروع.

أمر migrate

- يساعد الأمر [python manage.py migrate] بتطبيق التعديلات على قاعدة البيانات وتحديثها بحيث تكون متوافقة مع أكواد model.

أمر sqlmigrate

- يساعد الأمر [python manage.py sqlmigrate App_Name migration_name] بطباعة sql الخاص بملف migration_name .

عمليات CRUD على الـ Model

لتنفيذ عمليات على Model الذي انشئناه ، يجب علينا في البداية استيراده . في حالتنا هنا سقوم بإستيراد الموديل Movie .

```
from .models import Movie
```

- الإنشاء

إنشاء كائن جديد (مدخل جديد في قاعدة البيانات)

```
    movie = Movie(name="Spider Man", descrption="A great movi
e", rating=3)
    movie.save()
```

- العرض

لجلب بينات جميع الأفلام المخزنة في قاعدة البيانات

```
movies = Movie.objects.all()
```

- التحديث

للتحديث على فيلم في قاعدة البيانات (نحتاج بالبداية أن نجلب الفيلم بإستخدام الـ id مع دالة get، من ثم ننفذ عملية التعديل)

```
    movie = Movie.objects.get(id=movie_id)
```

```
    movie.name = updated_movie["name"]
    movie.rating = updated_movie["rating"]
    movie.save()
```

- الحذف

لحذف فيلم من قاعدة البيانات (نحتاج بالبداية أن نجلب الفيلم بإستخدام الـ id مع دالة get، من ثم ننفذ عملية الحذف)

```
    movie = Movie.objects.get(id=index)
    movie.delete()
```