Software_Engineering Lab5

Heyam Hameed

الاستعلامات (Queries) في Django ORM

تعريف Query في Django

الـ **Query** هو طلب (Instruction) يوجّه إلى قاعدة البيانات لاسترجاع بيانات أو تعديلها أو حذفها. في Django ، هذه الطلبات تُكتب بأسلوب برمجي من خلال الـ **QuerySet**، وهي بنية بيانات تمثل مجموعة من الكائنات المسترجعة من قاعدة البيانات.

أنواع الاستعلامات الأساسية

1 - جلب جميع البيانات

يُستخدم all() لاسترجاع جميع السجلات من جدول معين:

```
students = Student.objects.all()
```

هذا الاستعلام يعيد جميع الطلاب من جدول **Student**.

2- التصفية بشروط

باستخدام filter() يمكن جلب بيانات وفق شروط محددة:

```
active_docs = Document.objects.filter(status="active")
```

يعيد المستندات التي حالتها "نشط."

3-استبعاد بيانات

يُستخدم exclude() لاستثناء سجلات لا تنطبق عليها الشروط:

```
non_crypto = Document.objects.exclude(title__icontains="crypto")
```

يستبعد المستندات التي تحتوي كلمة "crypto" في العنوان.

.4جلب سجل واحد

باستخدام ()get، بشرط أن يكون هناك سجل وحيد يطابق الشرط:

```
student = Student.objects.get(student_id="2025001")
```

إذا كان هناك أكثر من نتيجة أو لا توجد نتيجة، سيتم رفع استثناء.

الفرز والترتيب

يمكن ترتيب النتائج باستخدام :()order_by

```
latest_docs = Document.objects.order_by("-uploaded_at")
```

ترتيب تنازلي حسب تاريخ الرفع.

Django CleanApp

هو أسلوب لبناء تطبيق Django بحيث يكون:

- **منفصل الطبقات** :كل طبقة (نموذج، عرض، قالب، خدمة، منطق) مستقلة.
- **قابل للاختبار** :يسهل كتابة اختبارات لكل جزء.
- **قابل للتوسعة** :يمكن إضافة ميزات بدون كسر الكود القديم.
- **نظيف ومنظم** :يسهل فهمه حتى بعد شهور من العمل عليه.

مثال عملي: نموذج مستخدم + عرض الملف الشخصي

1- نموذج المستخدمmodels/user.py

```
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    bio = models.TextField(blank=True)

    def __str__(self):
        return self.name
```

2- **خدمة المستخدمservices/user_service.py**

```
from core.models.user import User

def get_user_by_email(email):
    try:
        return User.objects.get(email=email)
    except User.DoesNotExist:
        return None
```

3- عرض المستخدمviews/user_view.py

```
from django.shortcuts import render
from core.services.user_service import get_user_by_email

def profile_view(request):
    email = request.GET.get("email")
    user = get_user_by_email(email)
    return render(request, "user/profile.html", {"user": user})
```

4-قالبHTML – templates/user/profile.html

```
<h2>{{ user.name }}</h2>
<p>{{ user.email }}</p>
<p>{{ user.bio }}</p>
```

5-ربط المسارcore/urls.py –

```
from django.urls import path
from core.views.user_view import profile_view

urlpatterns = [
    path("profile/", profile_view, name="profile"),
]
```

ثم تربطيه في cleanapp/urls.py:

```
from django.urls import path, include

urlpatterns = [
    path("", include("core.urls")),
]
```

مميزات هذا الأسلوب

الفائدة	الميزة
يسهل اختبار الخدمات بدون الحاجة لواجهة المستخدم	فصل المنطق عن العرض
يسهل التنقل بين الملفات وفهم المشروع	تنظيم الملفات
يمكن إضافة ميزات جديدة بدون تعديل الملفات القديمة	قابلية التوسعة
يمكن كتابة اختبارات لكل طبقة على حدة	سهولة الاختبار