# Videofile: 73

# Code file : ch63 Model Manager in Django 5

See below for self notes

**Model Manager in Django 5**

* A **Model Manager** is the interface through which database query operations are provided to Django models.
* By default, every model has at least one manager called **objects**.
* Managers are responsible for:
  + **Database queries** (all(), filter(), get(), etc.)
  + **Customizing querysets** (by overriding or adding methods).
  + **Encapsulating business logic** (so you don’t repeat queries everywhere).

**✅ Why use Custom Managers?**

* To add **custom query methods** for frequently used filters.
* To change the **default queryset behavior** (e.g., only return active users).

**🔑 Key Points:**

1. **Default Manager** → objects is automatically added if no manager is defined.
2. **Custom Manager** → You can create your own by subclassing models.Manager.
3. **Custom QuerySet** → For reusable filters, you can also chain custom QuerySet methods.

👉 Example for notes:

from django.db import models

class StudentManager(models.Manager):

    # Custom method: return students with marks > 60

    def top\_students(self):

        return self.filter(marks\_\_gt=60)

class Student(models.Model):

    name = models.CharField(max\_length=100)

    marks = models.IntegerField()

    # Attach custom manager

    objects = StudentManager()

# Usage:

# Student.objects.top\_students()

⚡ **Summary for Notes:**  
Model Managers in Django control how database queries are made. The default manager is objects, but you can create custom managers to add extra methods or modify default queries. This helps keep query logic organized and reusable.

Self notes:

If we create model in our project so their manager is created automatically if we don’t make like

Models.py:

from django.db import models

# from school.managers import CustomStudentManager

class Student(models.Model):

 name = models.CharField(max\_length=70)

 roll = models.IntegerField(unique=True, null=False)

 city = models.CharField(max\_length=70)

 marks = models.IntegerField()

 pass\_date = models.DateField()

 admission\_date=models.DateTimeField()

 objects = models.Manager()# this can create automatically if we dont create it

 #this is use to change the manager name object into students

 students = models.Manager()

and than run makemigraitons and migrate:

objects is the interface between the model and the view:

in models.py:

if we write

students = models.Manager()

so in views.py:

Student.students.all()

Custom manager:

For custom mamager we can code inside the models and also create new folder names as managers.py:

managers.py:

from django.db import models

class CustomStudentManager(models.Manager):

    def get\_queryset(self):

        return super().get\_queryset().order\_by('name')

in models.py:

from django.db import models

from school.managers import CustomStudentManager

# here are our old models fields

# custommanager

#  objects = CustomStudentManager()

 students = CustomStudentManager()

in viewspy:

def home(request):

 student\_data = Student.students.all()

#  student\_data = Student.students.all()

**We also make self method using that**

Managers.py:

class CustomStudentManager(models.Manager):

    def get\_student\_roll\_range(self,r1,r2):

        return super().get\_queryset().filter(roll\_\_range=(r1,r2))

models.py:

 students = CustomStudentManager()

views.py:

from django.shortcuts import render

from school.models import Student

def home(request):

 student\_data = Student.students.get\_student\_roll\_range(101,105)

**we also work with proxy model:**

class ProxyStudent(Student):

 students = CustomStudentManager()

 class Meta:

  proxy = True

  ordering = ['name']

**complete files with codes :**

**models.py:**

from django.db import models

from school.managers import CustomStudentManager

class Student(models.Model):

 name = models.CharField(max\_length=70)

 roll = models.IntegerField(unique=True, null=False)

 city = models.CharField(max\_length=70)

 marks = models.IntegerField()

 pass\_date = models.DateField()

 admission\_date=models.DateTimeField()

 #manager name changing

#  objects = models.Manager()

#  students = models.Manager() # this can create automatically if we dont create it

# custommanager

#  objects = CustomStudentManager()

 students = CustomStudentManager()

class ProxyStudent(Student):

 students = CustomStudentManager()

 class Meta:

  proxy = True

  ordering = ['name']

**manager..py:**

from django.db import models

# class CustomStudentManager(models.Manager):

#     def get\_queryset(self):

#         return super().get\_queryset().order\_by('name')

class CustomStudentManager(models.Manager):

    def get\_student\_roll\_range(self,r1,r2):

        return super().get\_queryset().filter(roll\_\_range=(r1,r2))

**views.py:**

from django.shortcuts import render

from school.models import Student

def home(request):

#  student\_data = Student.students.all()

 student\_data = Student.students.get\_student\_roll\_range(101,105)

#  student\_data = Student.students.all()

#  student\_data = Student.students.get\_stu\_roll\_range(101, 105)

 context = {'students':student\_data}

 return render(request, 'school/home.html', context)