TSU. en Tecnologías de la Información

en Entornos Virtuales y Negocios Digitales

Made By:

“Humaran Beltran Saul Alejandro“

Teacher:

“Ing. Ray Brunett Parra Galaviz”

Assigment:

“Web Aplications“

Actividad:

“Django Manager “

Cuatrimestre: Third ***Grupo:*** 3-F

Date

“25/10/2023“

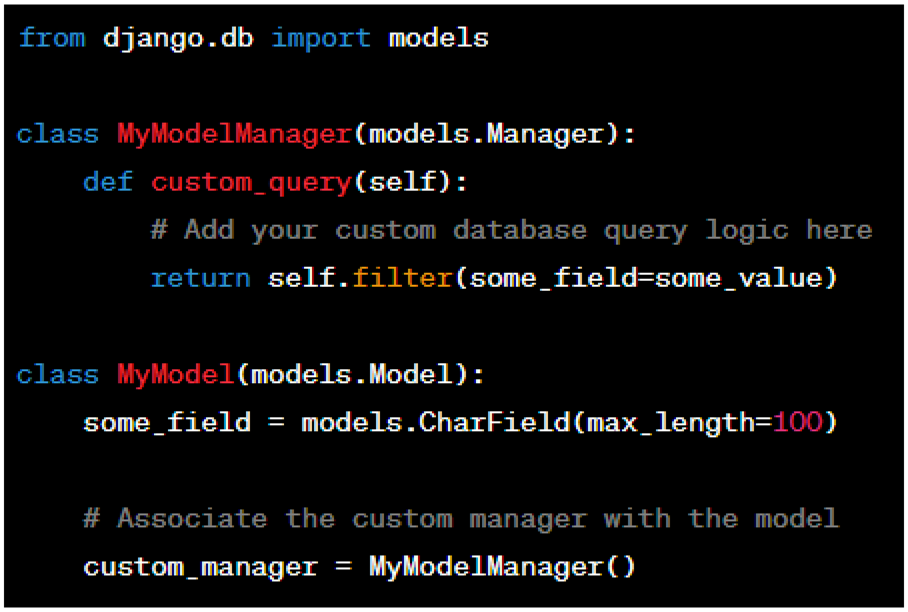
What is Django Manager?

In Django, a manager is an interface that allows you to perform various database queries and operations on a model. Managers are Python classes that provide a high-level API for querying the database and are associated with model classes. They enable you to retrieve, create, update, and delete objects in the database.

What Django Manager Provide?

Django provides a default manager for every model, which is called "objects" by convention. You can also create custom managers for your models to encapsulate common queries and operations that you frequently perform. Custom managers allow you to add custom methods to your models for retrieving specific sets of objects from the database.

Example



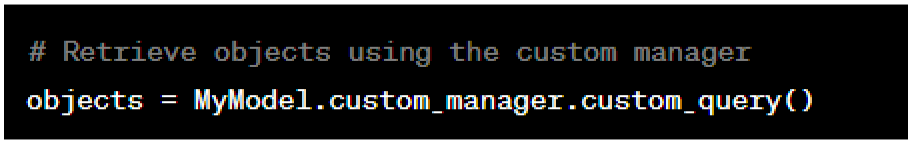
```python
from django.db import models

class MyModelManager(models.Manager):
    def custom_query(self):
        # Add your custom database query logic here
        return self.filter(some_field=some_value)

class MyModel(models.Model):
    some_field = models.CharField(max_length=100)

    # Associate the custom manager with the model
    custom_manager = MyModelManager()
```

With this custom manager, you can use the custom_query method to retrieve objects that meet specific criteria:



```python
# Retrieve objects using the custom manager
objects = MyModel.custom_manager.custom_query()
```

Conclusion:

Django's managers are a powerful feature that helps you encapsulate database-related logic and keep your code organized. They make it easier to perform common database operations and queries while adhering to the DRY (Don't Repeat Yourself) principle.