Tops Technology

Module 16) Python DB and Framework

Presented By : Nandni Vala

ORM and QuerySets :

1.Understanding Django’s ORM and how QuerySets are used to interact with the database.

- Django’s **Object-Relational Mapping (ORM)** allows developers to interact with databases using Python classes and methods, eliminating the need to write raw SQL. It maps database tables to Python models and provides an abstraction for performing database operations.
- A **QuerySet** is a collection of database queries to retrieve, filter, update, or delete data from the database. It is a Pythonic way to interact with the database and is lazy, meaning the database query is not executed until the data is needed.
- **Using QuerySets to Interact with the Database**

1. Retrieving Data

- **All Records**:
- books = Book.objects.all() # Fetches all records
- **Filtered Records**:
- cheap_books = Book.objects.filter(price__lt=10) # Books with price < 10

- **Single Record**:
- book = Book.objects.get(id=1) # Raises `DoesNotExist` if no match
- **Ordering**:
- books = Book.objects.order_by('title') # Ascending
- books = Book.objects.order_by('-title') # Descending
- **Creating Data**
- **Using create()**:
- Book.objects.create(title="Django ORM", author="John Doe", price=19.99)
- **Saving an Instance**:
- book = Book(title="Advanced ORM", author="Jane Doe", price=25.99)
- book.save()
- **Updating Data**
- **Single Record**:
- book = Book.objects.get(id=1)
- book.price = 20.99
- book.save()

- **Multiple Records**:
- Book.objects.filter(price__lt=15).update(price=15)
- **Deleting Data**
- **Single Record**:
- book = Book.objects.get(id=1)
- book.delete()
- **Multiple Records**:
- Book.objects.filter(price__lt=10).delete()
- **Advanced QuerySet Features**
- **Chaining Queries**: QuerySets can be chained for complex queries:
- books = Book.objects.filter(author="John").exclude(price__gt=20).order_by('title')
- **Aggregations**: Perform calculations like count, average, sum, etc.:from django.db.models import Avg
- avg_price = Book.objects.aggregate(Avg('price'))

- **Related Models**: Use QuerySets for relationships:
- authors_books = Book.objects.filter(author__name="John Doe")
- **Advantages of Django ORM and QuerySets**
- Simplifies database operations.
- Reduces the risk of SQL injection.
- Provides an abstraction layer for different databases.
- Encourages clean and maintainable code.