Volume 2, Issue 12, December 2024 **ISSN (E):** 2938-3757

DATABASE CAPABILITIES OF THE DJANGO FRAMEWORK

Tuhtanazarov Dilmurod Solijonovich
International Islamic Academy of Uzbekistan "Modern Information and Communication Technologies" Department, Associate Professor, PhD
dtuxtanazarov@gmail.com

Mahkamov Anvarjon Abdujabborovich
International Islamic Academy of Uzbekistan "Modern Information and Communication Technologies" Department, Associate Professor, PhD
mahkamovanvar2020@gmail.com

Abstract
In this article, the statistics of the Python programming language's Django framework and its capabilities in working with databases are discussed. The article explains Django ORM and the methods for creating models. It provides an overview of the field types in models. The process of registering models and working with the admin panel settings is explained. Additionally, the possibilities of performing migrations and make migrations, as well as working with migration files, are outlined.

Keywords: Database, Django, model, migrate, make migrations, field, register, admin, query set, framework.

Introduction

The Django framework is currently very popular worldwide for creating web applications, and it is used by many companies and developers. Django stands out for its convenience, security, and efficiency. This technology is a convenient tool for developing web applications quickly and efficiently and is applied across various sectors.

Django has remained one of the most popular frameworks for web development for several years. It is widely used among developers due to its ease of use and effectiveness. In 2023, Django ranks second among Python web frameworks (with Flask being in the first position). Django is often chosen for new projects because of its full framework structure, extended capabilities, and the large community that has contributed to its development.

Django technology is used in various fields. Significant steps have been taken in developing web applications based on Django, and its adoption is expanding globally. Many large and successful companies around the world use Django. Some popular platforms built with the Django framework include:

- **Instagram**: One of the most popular social network platforms built with Django.
- **Pinterest**: Initially built on Django, this platform has seen significant growth.
- **Spotify**: Django has been used to manage parts of the Spotify system.
- **Disqus**: One of the largest commenting systems on the internet, created using Django.

Web of Technology: Multidimensional Research Journal
webofjournals.com/index.php/4

- **YouTube (library and admin panel)**: Django is used for some parts of YouTube.
- **The Washington Post**: Django has been used to develop and manage a wide range of news and media content.

PROBLEM-SOLVING TECHNOLOGY. Django ORM (Object-Relational Mapping) is an object-based database management system that connects SQL queries to Python objects. With Django ORM, it is easy to access and manage the database. ORM transforms data into Python classes, and these classes correspond to the structures of the database tables.
Key concepts of Django ORM:

- **Model**: A model is used to create a table in the database through Django ORM. A model is a Python class where each attribute represents a column in the table.
- **Migration**: Django uses migrations to automatically manage changes in the database. With migrations, tables can be created, modified, or deleted.

```
from django.db import models
class Post(models.Model):
title = models.CharField(max_length=100)
text = models.TextField()
author = models.CharField(max_length=100)
date = models.DateField(auto_now=True)
```

In this example, the Post model creates a "news" table in the database, where the title, text, author, and creation date of each post are stored.
Creating and Applying Migrations:
python manage.py makemigrations
This command generates migration files based on changes made to the models.
python manage.py migrate
This command applies the generated migrations to update the database schema accordingly.
Using Django ORM, performing CRUD (Create, Read, Update, Delete) operations on the database is very simple.
To create a new object, we use the model as follows:

```
post = Book(title="News", text ="News text",  author="Alisherov Alisher", date="2024-12-25")
post.save()
```

To retrieve data from the database, we use Django ORM queries.

```
Get all posts:
posts = Post.objects.all()
Get filtered results:
posts = Post.objects.filter(author="Alisherov Alisher")
Get a single post:
post = Post.objects.get(id=1)
```

Update a specific post:

```
post = Post.objects.get(id=1)
post.title = "news 1"
post.save()
```

Delete post:

```
post = Post.objects.get(id=1)
post.delete()
```

It is also very convenient to perform complex queries through Django ORM. We use methods such as filter(), exclude(), annotate() and aggregate().
Retrieving data based on a specific condition:

```
posts = Post.objects.filter(author=" Alisherov Alisher ", date=2023)
```

To get results that do not match a given condition:

```
posts = Post.objects.exclude(author=" Alisherov Alisher ")
```

Make additional calculations:

```
from django.db.models import Count

authors = Post.objects.values('author').annotate(post_count=Count('id'))
```

Getting total values from the table:

```
from django.db.models import Avg
avg_date = Book.objects.aggregate(Avg('date'))
```

Django supports connecting to various databases. You can configure the connection in the DATABASES section of Django's settings.
For example, to work with a MySQL database:
In the settings.py file:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'mydatabase',
        'USER': 'myuser',
        'PASSWORD': 'mypassword',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

It also easily connects to other databases such as PostgreSQL and SQLite.

Web of Technology: Multidimensional Research Journal
webofjournals.com/index.php/4

Migrations allow you to make changes to your database. Django automatically generates SQL queries associated with your migrations. If a new field is added to your model or an old field is changed, Django detects these changes and reflects them to the database.
The process of creating and applying a migration:
- Creating a migration:

```
python manage.py makemigrations
```

- Apply migration:

```
python manage.py migrate
```

Django also provides a convenient admin panel for managing the database. To display a model in the admin panel, you need to register it in the admin.py file:

```
from django.contrib import admin
from .models import Post
admin.site.register(Post)
```

After that, Django will automatically create an admin panel and you can manage the database through its web interface.

CONCLUSION

Working with a database using Django ORM is very convenient and efficient. Through ORM, you can perform CRUD operations, make complex queries, and manage the database. Using Django, it is very easy to work with various databases such as MySQL, PostgreSQL, SQLite, and apply migrations. The Django administration panel allows you to visually manage the database.
Django ORM's powerful features and high level of automation provide developers with great convenience in working with the database.

References

[1] Ben Shaw, Saurabh Badhwar, Chris Guest, Bharath Chandra K S. Web Development with Django. Copyright © 2023 Packt Publishing. Second edition: May 2023
[2] Foster, D., & Walden, T. (2021). psycopg2 - PostgreSQL Database Adapter for Python. Retrieved from https://www.psycopg.org/docs/
[3] Ullman, J. D., & Widom, J. (2008). Database Systems: The Complete Book. Pearson Education.
[4] Patel, R. (2019). Python and SQL Integration: A Practical Guide. O'Reilly Media.
[5]https://www.sqlite.org/docs.html -SQLite Consortium. (2021). SQLite Documentation. Retrieved from