Django ORM – Inserting, Updating & Deleting Data

Last Updated : 12 Sep, 2023

Django lets us interact with its database models, i.e. add, delete, modify, and query objects, using a database-abstraction API called ORM(Object Relational Mapper). This article discusses all the functional operations we can perform using Django ORM.

Prerequisite: Django models

Django ORM

Django's Object-Relational Mapping (ORM) system, a fundamental component that bridges the gap between the database and the application's code. This article delves into the intricate realm of Python, Django, and their ORM, uncovering how this technology stack simplifies database interactions and accelerates the development process. For demonstration purposes, we will use the following Django models.

Python3

```
class Album(models.Model):
    title = models.CharField(max_length = 30)
    artist = models.CharField(max_length = 30)
    genre = models.CharField(max_length = 30)

    def __str__(self):
        return self.title

class Song(models.Model):
    name = models.CharField(max_length = 100)
    album = models.ForeignKey(Album, on_delete = models.CASCADE)

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

Django Shell

We can access the Django ORM by running the following command inside our project directory.

```
python manage.py shell
```

This brings us to an interactive Python console. Assuming that our models exist in

`myProject/albums/models.py`

we can import our models using the following command:

```
>>> from books.models import Song, Album
```

Django ORM Queries

Insert Data with Django ORM

To create an object of model Album and save it into the database, we need to write the following command:

```
>>> a = Album(title = "Divide", artist = "Ed Sheeran", genre = "Pop")
>>> a.save()
```

To create an object of model Song and save it into the database, we need to write the following command:

```
>>> s = Song(name = "Castle on the Hill", album = a)
>>> s.save()
```

Creating Data with Django ORM

Let us add 2 more Albums records for the sake of demonstration.

```
>>> a = Album(title = "Abbey Road", artist = "The Beatles", genre =
"Rock")
>>> a.save()
>>> a = Album(title = "Revolver", artist = "The Beatles", genre =
```

```
"Rock")
>>> a.save()
```

Retrieving Data with Django ORM

To retrieve all the objects of a model, we write the following command:

```
>>> Album.objects.all()
<QuerySet [<Album: Divide>, <Album: Abbey Road>, <Album: Revolver>]>
```

The output is a QuerySet, or a set of objects that match the query. Notice that the name printed is the output of the `__str__()` function. We can also filter queries using the functions `filter()`, `exclude()` and `get()`. The `filter()` function returns a QuerySet having objects that match the given lookup parameters.

```
>>> Album.objects.filter(artist = "The Beatles")
<QuerySet [<Album: Abbey Road>, <Album: Revolver>]>
```

The `exclude()` function returns a QuerySet having objects other than those matching the given lookup parameters.

```
>>> Album.objects.exclude(genre = "Rock")
<QuerySet [<Album: Divide>]>
```

The `get()` function returns a single object which matches the given lookup parameter. It gives an error when the query returns multiple objects.

```
>>> Album.objects.get(pk = 3)
<QuerySet [<Album: Revolver>]>
```

Update Data with Django ORM

We can modify an existing object as follows:

```
>>> a = Album.objects.get(pk = 3)
>>> a.genre = "Pop"
>>> a.save()
```

Deleting Data with Django ORM

To delete a single object, we need to write the following commands:

```
>>> a = Album.objects.get(pk = 2)
>>> a.delete()
>>> Album.objects.all()
<QuerySet [<Album: Divide>, <Album: Revolver>]>
```

To delete multiple objects, we can use `filter()` or `exclude()` functions as follows:

```
>>> Album.objects.filter(genre = "Pop").delete()
>>> Album.objects.all()
<QuerySet []>
```


Previous Article

Django Models

Next Article

Django Basic App Model - Makemigrations and Migrate

Similar Reads

How to Pull a Random Record Using Django's ORM?

Django's Object-Relational Mapping (ORM) is a powerful tool that allows developers to interact with the database using Python code instead of raw SQL...

3 min read

Implementing SQL LIKE Queries with Django ORM

When working with databases, one often comes across the necessity to do the search by pattern contained within the textual field. In SQL this can be done wit...

3 min read

Django ORM vs SQLAlchemy

For relational database newbies who are Python developers, Django ORM and SQLAlchemy are two heavyweights worth considering. As Object-Relational...

10 min read

Objection.js | SQL friendly ORM for Node.js

Node.js has plenty of object-relational mappers (for relational databases) that we can choose from few popular ones are: SequelizeMongoose or...

4 min read

SQLAlchemy ORM conversion to Pandas DataFrame

In this article, we will see how to convert an SQLAlchemy ORM to Pandas DataFrame using Python. We need to have the sqlalchemy as well as the panda...

4 min read

SQLAlchemy ORM - Creating Session

In this article, we will see how to create a session for SQLAlchemy ORM queries. Before we begin, let us install the required dependencies using pip: pip install...

3 min read

SQLAlchemy ORM - Query

In this article, we will see how to query using SQLAlchemy ORM in Python. To follow along with this article, we need to have sqlalchemy and anyone database...

10 min read

SQLAlchemy ORM - Declaring Mapping

In this article, we will see how to declare mapping using SQLAlchemy in Python. You will need a database (MySQL, PostgreSQL, SQLite, etc) to work with. Since...

4 min read

SQLAlchemy ORM - Adding Objects

In this article, we will discuss how to add objects in the SQLAlchemy ORM. The SQLAlchemy Object Relational Mapper presents a method of associating user-...

4 min read

Bulk insert with SQLAlchemy ORM in Python

In this article, we will see how to insert or add bulk data using SQLAlchemy in Python. SQLAlchemy is among one of the best libraries to establish...

1 min read

Article Tags : DBMS Python Web Technologies Django-basics +1 More

Practice Tags : python

OOAD
System Design Bootcamp
Interview Questions
Puzzles

School Subjects
Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos
DSA
Python
Java
C++
Web Development
Data Science
CS Subjects