Django ORM

Mastering Django ORM: A Comprehensive Guide to **Database Interactions** in Django

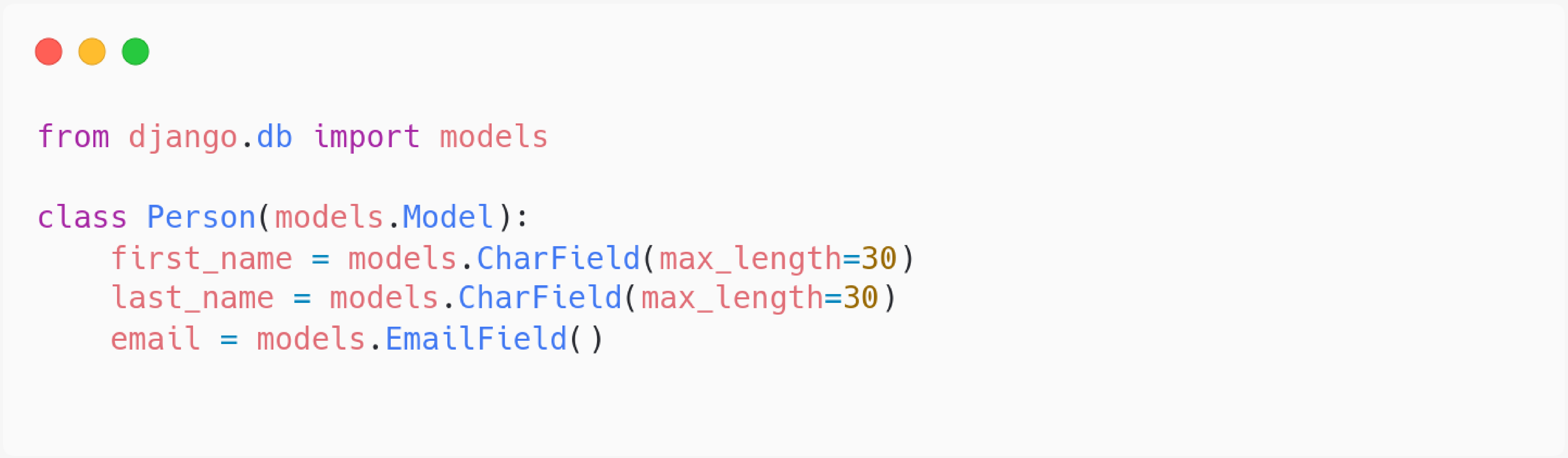
Ammar Munir

SWIPE

WHAT the Hell is **ORM** ?

In Django, **ORM** stands for Object-Relational Mapping. It is a technique that allows you to interact with your **database** using high-level object-oriented code, instead of writing raw SQL queries. Django's ORM provides a way to define your database schema and perform database operations using **Python classes and methods.**

Let's make it **simple**, create a **model**



```python
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField()
```

1. **Create** Migration.
2. **Apply** Migration to store that model in db

Swipe to Apply **CRUD** Operations using ORM

Ammar Munir

SWIPE

CREATE

```python
from .models import Person

person = Person(first_name='Ammar', last_name='Munir', email='munirammar0@gmail.com')
person.save()
```

READ

```python
from .models import Person

all_people = Person.objects.all()
```

UPDATE

```python
from .models import Person

person = Person.objects.get(id=1)
person.first_name = 'New Name'
person.save()
```

DELETE

```python
from .models import Person

person = Person.objects.get(id=1)
person.delete()
```

In this example, we **create** a person, **retrieve** all people from the database, **update** a person's name, and delete a person.

WHY **ORM** ?

Django's **ORM** provides a powerful and convenient way to interact with databases, abstracting away much of the complexity involved in working directly with **SQL**.

SWIPE

@ammar-munirr

Was it Helpful ?

Like Comment Share Save

Ammar Munir

Python | Django | Web Engineer