# Django insert data

## Prerequisites¶

- Django Models

## Django shell¶

- Run the command `python manage.py shell` open the django shell.
- Write django ORM queries to insert data into the database table.

## Django Model - Contact¶

- We will be using model `Contact` from Django Models.

my_app/models.py

```
from django.db import models

class Contact(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
    email = models.EmailField(max_length=255)
    phone = models.CharField(max_length=10, null=True)

    class Meta:
        db_table = "contact"
```

## insert data using - create()¶

- Open django shell and import the `Contact` model`
- write the below query to insert the data

```
from my_app.models import Contact

john = Contact.objects.create(
    first_name="John",
    last_name="D",
    email="john@example.com"
)
print(john)
# output: <Contact: Contact object (1)>
print(john.id)
# output: 1
```

- `create()` method make executes a insert query and insert the data into the database.

## insert data using - save()¶

- To use the `save()` method we need to create the object instance first.

```
from my_app.models import Contact

krish = Contact(
    first_name="Krish",
    last_name="C",
    email="krish@example.com"
)
print(krish)
# output: <Contact: Contact object (None)>
print(krish.id)
# output: None
# call .save() to insert it into the database.
krish.save()
print(krish)
# output: <Contact: Contact object (2)>
print(krish.id)
# output: 2
```

- Data inserted only if we call `save()` on the object otherwise it won't be inserted into the database.

## insert data using - get_or_create()¶

- `get_or_create()` is used to retrieve the record if exists otherwise it inserts the record into the database.
- It makes two database queries to if object does not exists in the database.

```
from my_app.models import Contact

irish, created = Contact.objects.get_or_create(
    first_name="Irish",
    last_name="W",
    email="irish@example.com"
)
print(created)
# output: True
print(irish)
# output: Contact object (3)
```

- Above code, executes two queries `SELECT` and `INSERT` because the object does not exists in the database.
- Let's try the same code again.

```
from my_app.models import Contact

irish, created = Contact.objects.get_or_create(
    first_name="Irish",
    last_name="W",
    email="irish@example.com"
)
print(created)
# output: False
print(irish)
# output: Contact object (3)
```

- Above code only executes `SELECT` query. Because the object already exists

## bulk insert using - bulk_create()¶

- In some cases, we may need to insert the data in bulk.
- In such cases, most of the people use `create()` method inside the loop. It will result in multiple database queries which takes more execution time.
- The best way to do it is to use `bulk_create()`
- Let's see how we can do that in the below code.

```
from my_app.models import Contact

contacts = [
    Contact(
        first_name="Berry",
        last_name="Allen",
        email="berry@example.com"
    ),
    Contact(
        first_name="Harrison",
        last_name="Wells",
        email="harrison@example.com"
    ),
    Contact(
        first_name="Oliver",
        last_name="Queen",
        email="oliver@example.com"
    ),
]

# use bulk_create() method to do bulk insert
results = Contact.object.bulk_create(contacts)
print(results)
# output:
# [<Contact: Contact object (4)>,
#  <Contact: Contact object (5)>,
#  <Contact: Contact object (6)>]
```

## References¶

- https://docs.djangoproject.com/en/dev/topics/db/queries/