Python

Full stack Skills Bootcamp

Introduction to Django ORM and QuerySets

What is Django ORM

Django ORM (Object-Relational Mapping) connects Python code with the database.

- QuerySets are lists of objects from your database, allowing you to interact with data easily.

With QuerySets, you can:

- Retrieve all objects from a table,
- Filter objects based on specific conditions,
- Order objects by certain fields from your database.

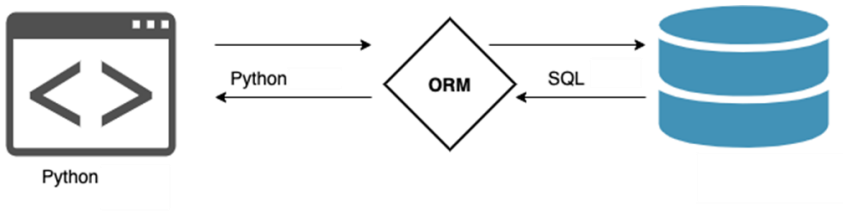
Django Shell

- Use the Django shell to interact with your models directly.
- Run the following command to open the shell:

```bash
python manage.py shell
```

Displaying All Objects

- Use modelname.objects.all() to show all objects in the Post model.
- Make sure to import the model first if it’s not already imported.




```python
from blog.models import Post
Post.objects.all()
```

Creating & Updating Object

You can create new records in the database using modelname.objects.create().

For example:

- Each post needs to have an author, which must be a user from the User model.
- You can create as many posts as you want repeating the same method, so each time you call Post.objects.create(), a new record is added to the database.

You can update one or more records in the database using **update()** method.

- Alternatively, if you want to update a specific object, retrieve it, modify it and call **save()** again.

```python
from django.contrib.auth.models import User
me = User.objects.get(username='ola')
Post.objects.create(author=me, title='Sample title', text='Test')
```

```python
Post.objects.filter(author=me).update(title='Updated Title')
```

```python
post = Post.objects.get(id=1)
post.title = 'Updated Post Title'
post.save()
```

Filtering QuerySets

- Use .filter() to get specific objects based on conditions.
- For example, find posts authored by a specific user.

```python
Post.objects.filter(author=me)
```

- You can filter using operations like **contains** to match parts of a field's value.
- Use **__contains** to find posts with a specific word in their table.

```python
Post.objects.filter(title__contains='title')
```

- Filter posts by the **published_date** to get only those published before the current time.

```python
from django.utils import timezone
Post.objects.filter(published_date__lte=timezone.now())
```

Ordering Objects (Sorting)

QuerySets allow you to order objects by a specific field.

For example:

- Use **.order_by()** to sort posts by the **created_date** field.

```python
Post.objects.order_by('created_date')
```

Reversing order:

- To reverse the order of the QuerySet, add a '–' before the field name.

```python
Post.objects.order_by('-created_date')
```

Additional QuerySet Operations

delete(): Deletes records from the database.

count(): Counts the number of records returned by a QuerySet.

exists(): Checks if a QuerySet contains any results.

aggregate(): Performs calculations like sum, average, etc., on a set of objects.

values(): Returns a QuerySet of dictionaries, where each dictionary is a record from the database with field names as keys.

distinct(): Removes duplicate records from the QuerySet.

Exiting Django Shell

Once you're done working in the Django shell, type **exit()** to close it.

```
python

>>> exit()
```

Dynamic data in templates

Update **views.py** to Pass Data to Template.

For example:

- In Blog/views.py you need to fetch the posts from the database using a QuerySet and pass them to the template.

```
from django.shortcuts import render
from django.utils import timezone
from .models import Post

def post_list(request):
    posts = Post.objects.filter(published_date__lte=timezone.now()).order_by('published_date
    return render(request, 'blog/post_list.html', {'posts': posts})
```

Update template to Display Data

Update **.html** to display dynamic data.

For example:

- Now, update your template (post_list.html) to loop through the posts QuerySet and display the data dynamically.

```
<body>
    <h1>Blog Posts</h1>
    {% for post in posts %}
        <div>
            <h2>{{ post.title }}</h2>
            <p>{{ post.text }}</p>
            <p><em>Published on: {{ post.published_date }}</em></p>
        </div>
    {% empty %}
        <p>No posts available.</p>
    {% endfor %}
</body>
```