SDEV 2401

Rapid Backend App Development

Databases, Models, Migrations and the Admin - 1

Expectations - What I expect from you

- No Late Assignments

- No Cheating

- Be a good classmate

- Don’t waste your time

- Show up to class

WE ARE ESSENTIAL TO ALBERTA | NAIT

Agenda

On the right is what we will cover today.

- Django ORM Fundamentals
- Defining a Model
- Available Core Fields in Django Models
- Making and Applying Migrations
- Registering Models with the Admin
- Creating and Querying Data
- Filtering and Updating Records
- Deleting Records
- Conclusion

WE ARE ESSENTIAL TO ALBERTA | NAIT

Django ORM Fundamentals

- Django’s ORM (Object Relational Mapper) lets you interact with your database using Python classes and objects.
- You don’t need to write raw SQL for most operations.
- Key concepts: Model, Migration, QuerySet, Admin.

WE ARE ESSENTIAL TO ALBERTA | NAIT

Defining a Model

- A model is a Python class that represents a table in your database.
- Example: Let’s track books in a library.

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=100)
    published_year = models.IntegerField()

    def __str__(self):
        return f"{self.title} by {self.author}"
```

WE ARE ESSENTIAL TO ALBERTA | NAIT

Available Core Fields in Django Models

Here's a brief overview of some core field types you can use in Django models (look at the docs Django documentation for more details):

- `CharField` : A short text field, used for small strings like names or titles.
- `TextField` : A large text field, used for longer strings like descriptions or comments.
- `IntegerField` : A field for storing integers.
- `FloatField` : A field for storing floating-point numbers.
- `BooleanField` : A field for storing boolean values (True/False).
- `DateField` : A field for storing dates.
- `DateTimeField` : A field for storing date and time values.
- `EmailField` : A field for storing email addresses, which includes validation to ensure the value is a valid email format.
- `URLField` : A field for storing URLs, which includes validation to ensure the value is a valid URL format.

WE ARE ESSENTIAL TO ALBERTA | NAIT

Making and Applying Migrations

- After creating or changing a model, generate a migration:

```
python manage.py makemigrations
```

- Apply the migration to update the database:

```
python manage.py migrate
```

- This creates the `Book` table in your database.

WE ARE ESSENTIAL TO ALBERTA | NAIT

Registering Models with the Admin

- To manage your data easily, register your model in `admin.py`:

```
from django.contrib import admin
from .models import Book

admin.site.register(Book)
```

- Now you can add, edit, and delete books via the Django admin interface.

WE ARE ESSENTIAL TO ALBERTA | NAIT

Creating and Querying Data

- Open the Django shell:

```
python manage.py shell
```

- Add a new book:

```
from library.models import Book
Book.objects.create(title="1984", author="George Orwell", published_year=1949)
```

- Query all books:

```
books = Book.objects.all()
for book in books:
    print(book)
```

WE ARE ESSENTIAL TO ALBERTA | NAIT

Filtering and Updating Records

- Filter books by author:

```
orwell_books = Book.objects.filter(author="George Orwell")
```

- Update a book's year:

```
book = Book.objects.get(title="1984")
book.published_year = 1950
book.save()
```

WE ARE ESSENTIAL TO ALBERTA | NAIT

Deleting Records

- Remove a book from the database:

```
book = Book.objects.get(title="1984")
book.delete()
```

WE ARE ESSENTIAL TO ALBERTA | NAIT

Conclusion

- Django ORM makes it easy to create, read, update, and delete records using Python.
- Use models to define your data structure, migrations to update your database, and the admin for easy management.
- Practice by building a model for another entity, like `Member` or `Loan`, and try CRUD operations in the shell.

WE ARE ESSENTIAL TO ALBERTA | NAIT

Example

Let’s go do an example together