PENJELASAN SOURCE CODE API_JADWAL IMSKIYAH

NAMA : PENDI SAPUTRA

NIM : 204855092

1. **admin.py**

Penggunaan Django Admin untuk mendaftarkan sebuah model bernama `jadwal`.

Django Admin adalah fitur bawaan Django yang memungkinkan administrator untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada data aplikasi. Dengan mendaftarkan model ke Django Admin, Anda dapat mengakses data model tersebut dari interface admin tanpa perlu membuat tampilan atau form khusus.

Untuk dapat menggunakan Django Admin, perlu melakukan registrasi model pada `admin.site` melalui `admin.site.register()`. Dalam kasus ini, model `jadwal` diimpor dari modul models (`from .models import jadwal`) dan kemudian didaftarkan dengan perintah `admin.site.register(jadwal)`.

Dengan begitu, model `jadwal` dapat diakses dari Django Admin interface pada URL `/admin` setelah administrator melakukan login dan mengeklik link untuk model tersebut. Di sana, administrator dapat melakukan operasi CRUD pada data yang terkait dengan model `jadwal`.

2. **apps.py**

definisi konfigurasi aplikasi Django yang disebut `JadwalImsyakConfig`. Konfigurasi ini digunakan untuk mengkonfigurasi aplikasi Django `jadwal_imsyak`.

`AppConfig` adalah kelas bawaan Django yang digunakan untuk mengatur konfigurasi aplikasi Django. Dalam konfigurasi ini, `default_auto_field` diatur sebagai `django.db.models.BigAutoField`, yang menunjukkan bahwa Django akan menggunakan tipe data `BigAutoField` sebagai primary key secara default ketika membuat model baru.

`name` diatur sebagai `jadwal_imsyak`, yang menunjukkan nama aplikasi Django yang akan digunakan. Nama aplikasi ini harus sama dengan nama folder aplikasi di mana file `apps.py` ditempatkan.

Setelah definisi konfigurasi ini, kita dapat mengintegrasikan aplikasi `jadwal_imsyak` ke dalam proyek Django secara keseluruhan dengan menambahkannya ke `INSTALLED_APPS` dalam file `settings.py`.

Perlu diingat bahwa konfigurasi ini hanya digunakan untuk mengatur aplikasi Django, dan tidak menambahkan model atau tampilan secara langsung. Untuk menambahkan model atau tampilan, kita harus membuat file yang sesuai dengan aturan yang berlaku di dalam folder aplikasi `jadwal_imsyak`.

3. **models.py**

Model Django yang disebut `jadwal` yang mendefinisikan struktur tabel dalam basis data. Model ini memiliki beberapa kolom atau atribut dengan tipe data yang berbeda seperti `CharField` untuk `tanggal` dan `TimeField` untuk waktu shalat.

`CharField` digunakan untuk menyimpan nilai karakter dengan panjang maksimum 50 karakter, sementara `TimeField` digunakan untuk menyimpan nilai waktu. Parameter `auto_created=False` digunakan untuk menonaktifkan pembuatan waktu secara otomatis saat data disimpan, sehingga waktu disimpan secara manual.

`def __str__(self)` digunakan untuk menentukan bagaimana objek model ini direpresentasikan sebagai string. Dalam hal ini, objek `jadwal` akan direpresentasikan sebagai string yang mengandung nilai kolom `tanggal`.

Dalam definisi model ini, `jadwal` adalah kelas yang mewarisi dari kelas `models.Model` yang disediakan oleh Django. Ini memberikan akses ke berbagai metode untuk membuat dan memanipulasi objek dalam model, seperti `objects.all()` untuk mendapatkan semua objek dalam model atau `save()` untuk menyimpan objek ke database.

Setelah definisi model ini, kita dapat membuat migrasi dan melakukan migrasi untuk membuat tabel dalam basis data. Setelah tabel dibuat, kita dapat menggunakan model ini untuk melakukan operasi CRUD pada data dalam tabel menggunakan ORM (Object Relational Mapping) Django.

4. **sereializers.py**

definisi serializer Django menggunakan Django REST framework untuk model `jadwal`. Serializer digunakan untuk mengubah objek model menjadi format yang dapat di-serialize seperti JSON atau XML, dan sebaliknya.

Dalam contoh ini, `jadwalserializer` adalah kelas serializer yang diturunkan dari kelas `serializers.ModelSerializer` yang disediakan oleh Django REST framework. `Meta` class di dalam `jadwalserializer` menyediakan informasi tentang model yang akan di-serialize, yaitu `jadwal`.

`fields = "__all__"` menunjukkan bahwa semua kolom dalam model `jadwal` akan di-serialize. Ini berarti ketika data `jadwal` di-serialize, semua kolom seperti `tanggal`, `imsak`, `subuh`, dll. akan termasuk dalam hasil serialisasi.

Setelah definisi serializer ini, kita dapat menggunakannya dalam view untuk mengkonversi data dari model `jadwal` menjadi format JSON dan mengirimkan sebagai respon HTTP.

5. **view.py**

Django REST framework untuk melakukan operasi CRUD pada data `jadwal` dalam database.

Dalam contoh ini, `readjadwal` dan `createjadwal` adalah fungsi view yang didekorasi dengan `@api_view(['GET'])` dan `@api_view(['POST'])` yang menunjukkan tipe metode HTTP yang digunakan dalam permintaan.

`readjadwal` berfungsi untuk membaca semua data `jadwal` dari database dan mengembalikan data dalam format JSON menggunakan serializer `jadwalserializer`. Pertama, fungsi ini memanggil `jadwal.objects.all()` untuk mendapatkan semua objek `jadwal` dari database. Kemudian, objek-objek ini diserialisasi menggunakan `jadwalserializer` dengan `many=True`, karena banyak objek yang akan di-serialize. Hasilnya kemudian dikirim kembali sebagai respons HTTP dengan menggunakan `Response(serializer.data)`.

`createjadwal` berfungsi untuk membuat data baru dalam tabel `jadwal` menggunakan data yang diberikan dalam permintaan HTTP POST. Pertama, fungsi ini membuat instance `jadwalserializer` menggunakan data dari permintaan HTTP dengan `serializer = jadwalserializer(data=request.data)`. Kemudian, ia memvalidasi data yang diberikan dengan memanggil `is_valid()`. Jika data valid, data baru disimpan ke dalam database dengan memanggil `serializer.save()`. Hasil dari serializer kemudian dikirim kembali sebagai respons HTTP dengan `Response(serializer.data)`.

Dalam rangka memungkinkan akses ke API dari sumber yang berbeda, anda juga perlu menambahkan middleware CORS.