INTRODUCTION TO APIS

DJANGO REST FRAMEWORK

What is an API?

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

What does API stand for?

API stands for **Application Programming Interface**. In the context of APIs, the word Application refers to any software with a distinct function. Interface can be thought of as a contract of service between two applications. This contract defines how the two communicate with each other using requests and responses. Their API documentation contains information on how developers are to structure those requests and responses.

How do APIs work?

API architecture is usually explained in terms of client and server. The application sending the request is called the **client**, and the application sending the response is called the **server**. So in the weather example, the bureau's weather database is the **server**, and the mobile app is the **client**.

There are four different ways that APIs can work depending on when and why they were created.

Types of APIs

SOAP APIs

These APIs use **Simple Object Access Protocol**. Client and server exchange messages using XML. This is a less flexible API that was more popular in the past.

RPC APIs

These APIs are called **Remote Procedure Calls.** The client completes a function (or procedure) on the server, and the server sends the output back to the client.

Types of APIs Cont'd

Websocket APIs

Websocket API is another modern web API development that uses JSON objects to pass data. A WebSocket API supports two-way communication between client apps and the server. The server can send callback messages to connected clients, making it more efficient than REST API.

REST APIs

These are the most popular and flexible APIs found on the web today. The client sends requests to the server as data. The server uses this client input to start internal functions and returns output data back to the client. Let's look at REST APIs in more detail below.

What are REST APIs?

REST stands for Representational State Transfer, it talks about the means of communication between these softwares. And REST is communicating over the web, meaning it is using HTTP protocol

REST defines a set of functions like **GET, PUT, DELETE,** etc. that clients can use to access server data. Clients and servers exchange data using HTTP.

The main feature of **REST API** is **statelessness. Statelessness** means that servers do not save client data between requests. Client requests to the server are similar to URLs you type in your browser to visit a website. The response from the server is plain data, without the typical graphical rendering of a web page.

What are the benefits of REST APIs?

REST APIs offer four main benefits:

1. Integration

APIs are used to integrate new applications with existing software systems. This increases development speed because each functionality doesn't have to be written from scratch. You can use APIs to leverage existing code.

2. Innovation

Entire industries can change with the arrival of a new app. Businesses need to respond quickly and support the rapid deployment of innovative services. They can do this by making changes at the API level without having to re-write the whole code.

What are the benefits of REST APIs?

3. Expansion

APIs present a unique opportunity for businesses to meet their clients' needs across different platforms. For example, maps API allows map information integration via websites, Android,iOS, etc. Any business can give similar access to their internal databases by using free or paid APIs.

4. Ease of maintenance

The API acts as a gateway between two systems. Each system is obliged to make internal changes so that the API is not impacted. This way, any future code changes by one party do not impact the other party.

Django Rest Framework

Django Rest Framework (DRF) is a package built on the top of Django to create web APIs. It provides the most extensive features of Django, Object Relational Mapper (ORM), which allows the interaction of databases in a Pythonic way.

Hence the Python object can't be sent over the network, so we need to translate Django models into the other formats like JSON, XML, and vice-versa. This process is known as **serialization**, which the Django REST framework made super easy.

JSON

JSON stands for JavaScript Object Notation.

JSON is a text format for storing and transporting data, is used to send data between computers

Note: it's not javascript, though its syntax look like js object or python dictionary.it is an independent technology on its own, but programming language like python or js have built in package to interact with json.
Python has a built in package for working with json.
In python you can import json. Which is the python library for working with json data
With this package we can convert JSON string to python dictionary or convert python object like list, dict to JSON string.

Key Terms

Endpoint

A traditional website consists of web pages with HTML, CSS, images, JavaScript, and more. There is a dedicated URL, such as example.com/1/, for each page. A web API also relies on URLs and a corresponding one might be example.com/api/1/, but instead of serving up web pages consumable by humans it produces API endpoints. An endpoints contains data, typically in the JSON format, and also a list of available actions (HTTP verbs).

SERIALIZERS

What is a serializer?

Serializers allow complex data such as querysets and model instances to be converted to native Python data types that can then be easily rendered into `JSON`, `XML` or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.

The serializers in REST framework work very similarly to Django's `Form` and `ModelForm` classes. We provide a `Serializer` class which gives you a powerful, generic way to control the output of your responses, as well as a `ModelSerializer` class which provides a useful shortcut for creating serializers that deal with model instances and querysets

Serializers Cont’d

Serializers in Django REST Framework are **responsible for converting objects into data types understandable by javascript and front-end frameworks**. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data.Serializers are used to represent the model data in JSON format and convert object instances to a more transferable format. It makes the process of parsing data from our API easy. On the other hand, Deserializers convert the JSON data into our model as an object instance.We will create the serializers.py file in the sample_app, converting a model object into JSON format before sending the response.