Django Rest Framework

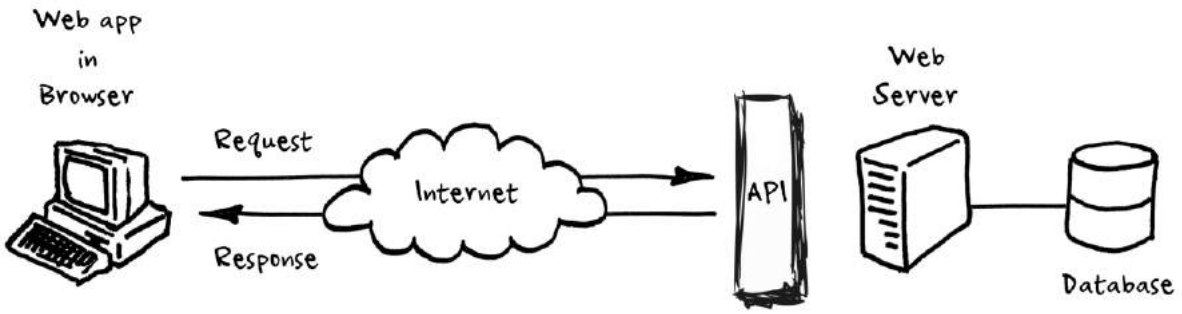
Notes By

Naveen

What is an API?

API stands for Application programming interface.

An **API** is a set of programming code that enables data exchange between one software product to another software product. It also contains the terms of this data exchange.



The software that needs to access information or functionality from another software, calls its API while specifying the requirements of how data/functionality must be provided. The other software returns data/functionality requested by the former application.

API TYPES BY RELEASE POLICIES

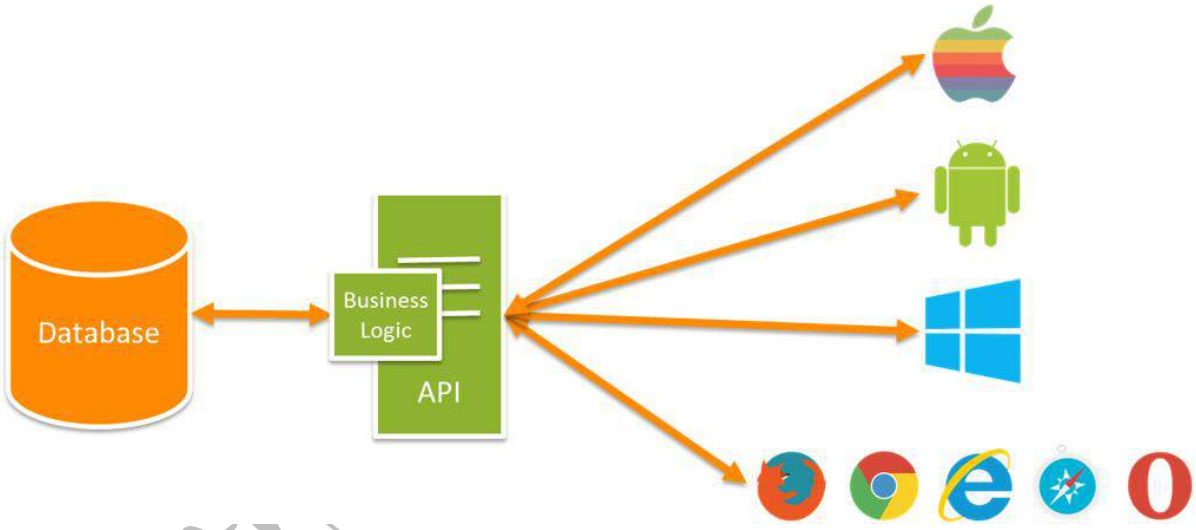
Private	Partner	Public or external
APIs are solely used within an organization Apps are mostly built for company employees. Common use cases are the integration of company systems/apps or development of new systems using existing resources	Openly promoted but available for known business partners End customers or business users are potential target audiences for such apps. Software integration between two organizations is the popular use case for these APIs	Available to any third-party developers Apps with public APIs are mostly designed for end customers This API release policy allows for increasing brand awareness and fostering external innovation

Web API

The Web API is an extensible framework for building HTTP based services that can be accessed in different applications on different platforms such as web, windows, mobile etc.

It works more or less the same way as Django MVT web application except that it sends data as a response instead of html view. It only supports HTTP protocol.

Web API supports RESTful applications and uses GET, PUT, POST, DELETE verbs for client communications.

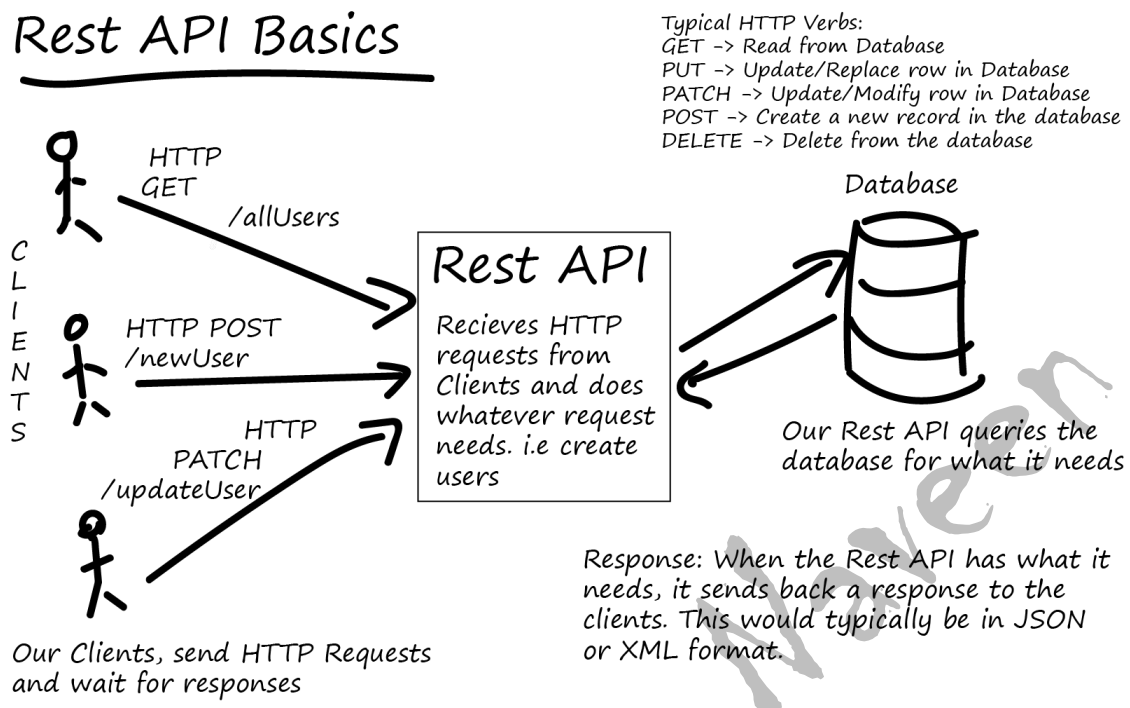


What is REST?

REST stands for **Representational state transfer.**

It is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other.

REST-compliant systems, often called RESTful systems.



HTTP request methods

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource.

Verb	Objective	Usage
GET	Retrieve items from resource	links
POST	Create new item in resource	forms
PUT	Replace existing item in resource	forms
PATCH	Update existing item in resource	forms
DELETE	Delete existing item in resource	forms/ links

HTTP status codes

Status codes are issued by a server in response to a client's request made to the server.

This is a list of **Hypertext Transfer Protocol** (HTTP) response status codes.

There are five classes defined by the standard:

- **1xx informational response** .
- **2xx successful .**
- **3xx redirection .**
- **4xx client side error .**
- **5xx server side error.**

Note : refer to https://httpstatuses.com/

Integration of two applications

Integration means connecting one application with another application which are running in same server or in different server.

Example :

1) amazon is using ICICI,SBI,KOTAK for payments.

2) makemytrip is using Flights,Hotels,Trains,Buses,Cabs,Visa

3) In Mobile Swiggy app is using Hotels,banks

Integration is of 2 types.

1) **Homogeneous Integration** : The 2 applications which are designed in same language like in python.

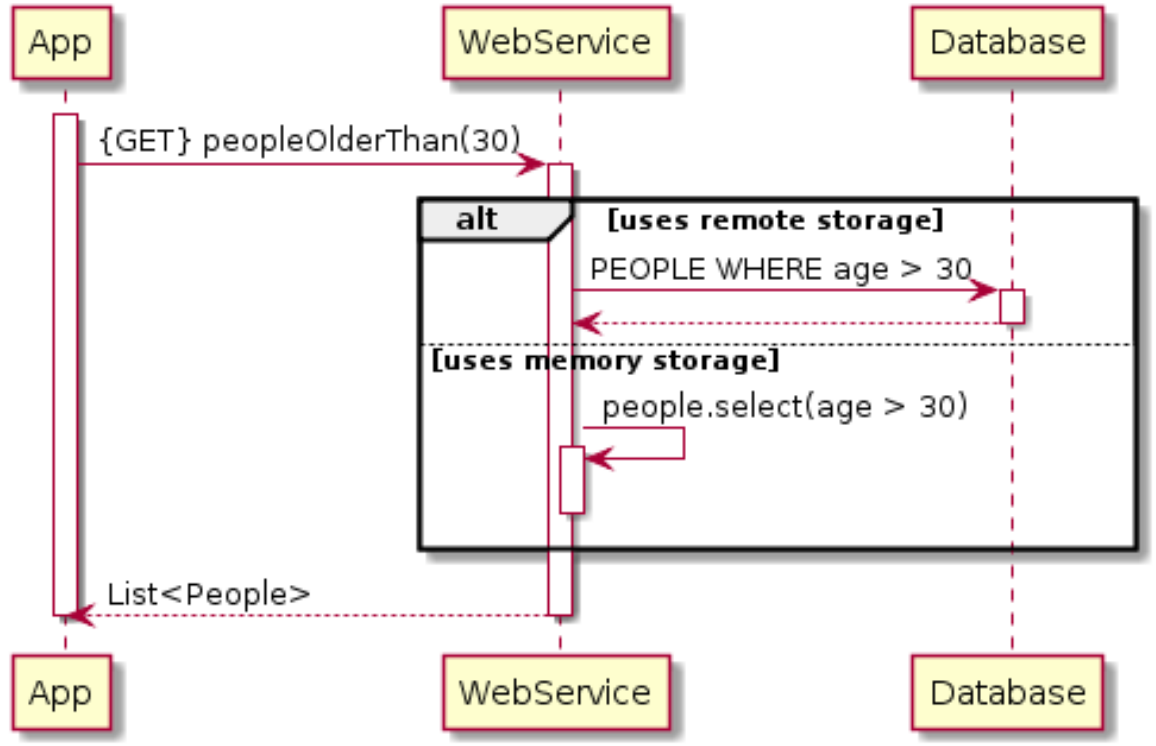
2) **Heterogeneous Integration** : The 2 applications which are designed in 2 different languages like 1 in python the other in Java.

What Is a Web Service?

A web service is an application or data source that is accessible via a standard web protocol (HTTP or HTTPS).

Unlike web applications, web services are designed to communicate with other programs, rather than directly with users.

While web services can provide data in different formats like XML and JSON. These standard text-based formats can be easily recognized and parsed by another program that receives the data.



Provider Application : The application which provides the service to other application

Consumer Application : The application which consume the service from provider application.

JSON

(JavaScript Object Notation)

JSON is a lightweight data-interchange format.

JSON is easy for humans to read and write. It is easy for machines to parse and generate.

JSON is based on a subset of the **JavaScript** Programming Language.

JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others.

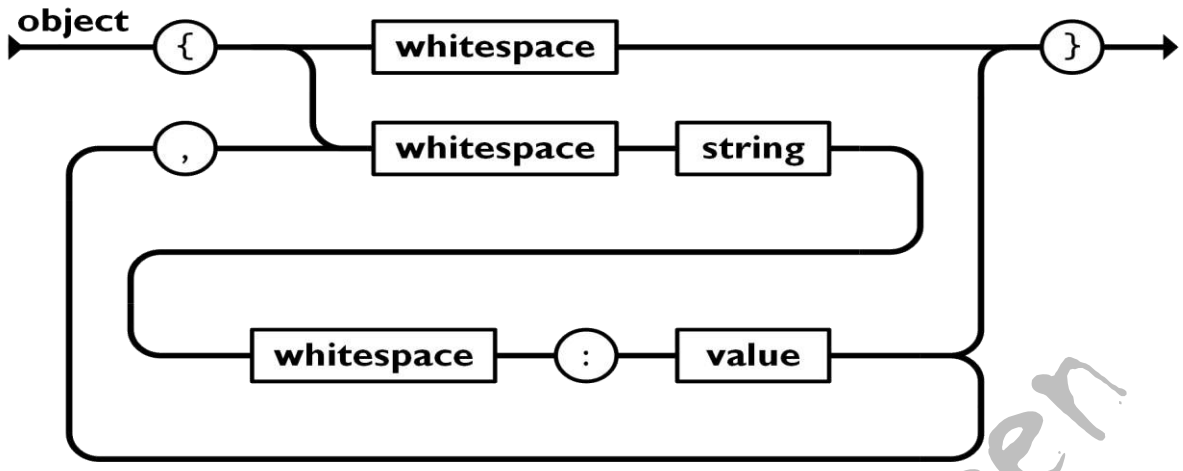
In JSON, they take on these forms:

Object

An *object* is an unordered set of name/value pairs.

An object begins with **{***left brace* and ends with **}***right brace*.

Each name is followed by **:***colon* and the name/value pairs are separated by **,***comma*.

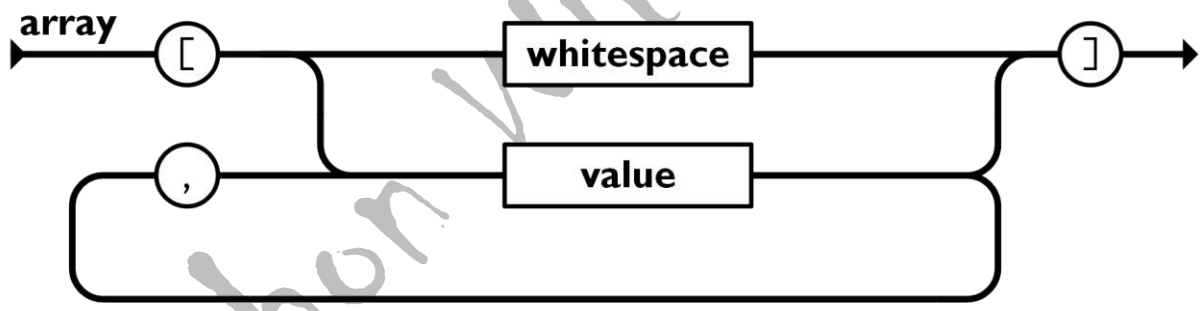


Array

An *array* is an ordered collection of values.

An array begins with **[***left bracket* and ends with **]***right bracket*.

Values are separated by **,***comma*.



Python comes with a built in package called json for encoding and decoding JSON data.

Example : import json

The process of encoding JSON is usually called **serialization**.

This term refers to the transformation of data into a *series of bytes* (hence *serial*) to be stored or transmitted across a network.

Naturally, **deserialization** is the mutual process of decoding data that has been stored or delivered in the JSON standard.

Serializing JSON

The json library exposes the **dump()** method for writing data to files.

There is also a **dumps()** method for writing to a Python string.

Deserializing JSON

In the json library, you'll find **load()** and **loads()** for turning JSON encoded data into Python objects.

Python	JSON
dict	object
list, tuple	array
str	string
int, float	number
True	true
False	false
None	null

Json Data

```
data = {

   "employee": {  "name": "Ravi",   "company": "Sathya"  }

}
```

A Simple Serialization Example

Write the above json code into a file and the extension of the file must be .json

Example

```
import json

file = open("sample.json","w")
json.dump(data,file)
```

Note that dump() takes two positional arguments:

(1) the data object to be serialized, and

(2) the file-like object to which the bytes will be written.

A Simple De-Serialization Example

Just open the .json file in read mode.

You can easily deserialize that with loads(), which naturally loads from a string.

Example

```
import json
file = open("sample.json","r")
str_data = file.read()
# converting str to dict type(json type)
json_data = json.loads(str_data)
print(json_data)
print(type(json_data))
```

Django web application Example's

1) In this example the Django application is giving html response, this response can understand by any web browser(For User not for application).

urls.py

```
path('index/',views.openMainPage, name="index")
```

views.py

```
from django.http import HttpResponse
def openMainPage(request):
    html_str = "<html><body><h1>Welcome to Python with Naveen</h1></body></html>"
    return HttpResponse(html_str)
```

Output

2) In this example the Django application is giving JSON response to browser in HttpResponse(For application not for user).

Note : In this we are using python built-in package "json"

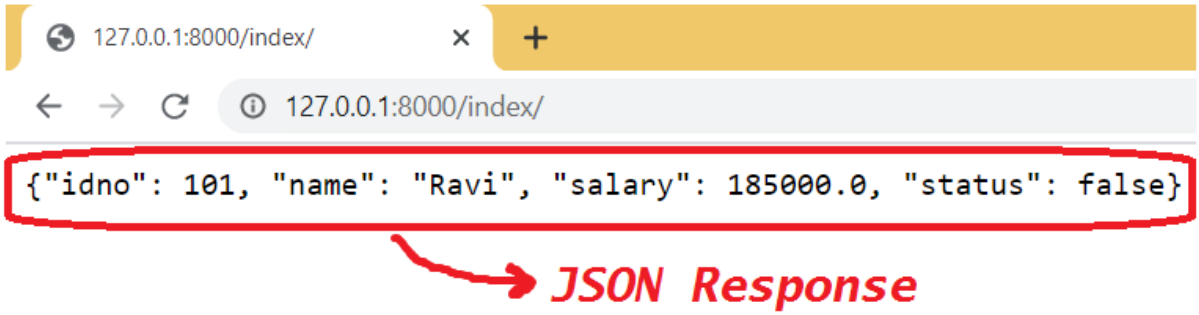
urls.py

```
path('index/',views.openMainPage, name="index")
```

views.py

```
from django.http import HttpResponse
import json

def openMainPage(request):

    # dict type data
    emp_info = {"idno":101,"name":"Ravi","salary":185000.00,
"status":False}

    # converting dict to json
    json_data = json.dumps(emp_info)

    return HttpResponse(json_data,content_type="application/json")
```

Output



3) In this example the Django application is giving JSON response to browser in HttpResponse(For application not for user).

Note: In this we are using django built in "JsonResponse"

urls.py

```
path('index/',views.openMainPage, name="index")
```

views.py

```
from django.http import JsonResponse

def openMainPage(request):

    # dict type data
    emp_info = {"idno":101,"name":"Ravi","salary":185000.00,
"status":False}

    return JsonResponse(emp_info)
```

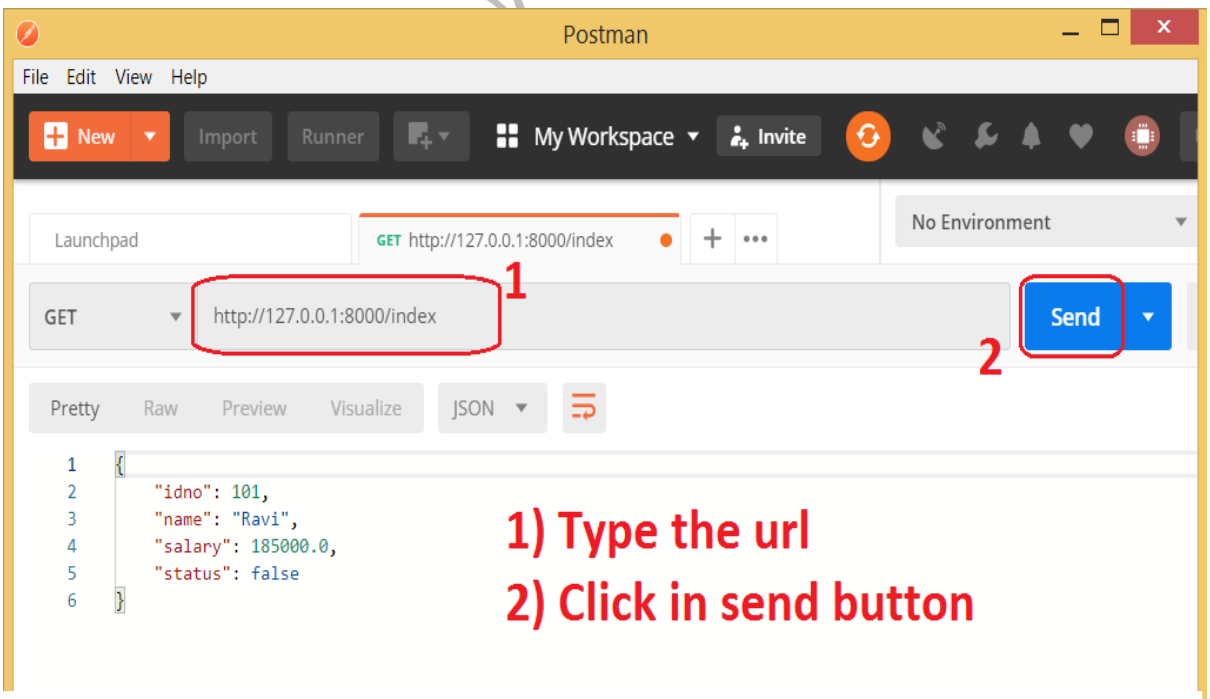
Output

Note : To see the output of 2 and 3rd examples we need to use a application

In this Example I am using python program to access the 2nd example.

```
import requests
url = "https://127.0.0.1:800/index/"
response = requests.request("GET", url)
json_data = response.json()
print(response.text)
```

In this Example I am using Postman to access the 3rd example.



What is Postman? (https://www.postman.com/)

Postman is a collaboration platform for API development.

Postman's features simplify each step of building an API and streamline collaboration so you can create better APIs—faster.

Use Cases

API-First Development

Release reliable services by building your API before deploying code.

Application Development

Eliminate dependencies and reduce time to production by having front-end and back-end teams work in parallel.

Automated Testing

Automate manual tests and integrate them into your CI/CD pipeline to ensure that any code changes won't break the API in production.

Exploratory Testing

Explore the API by sending it different kinds of data to see what values are returned.

Developer Onboarding

Quickly get consumers up to speed on what your API can do and how it works.

Developer Portals

Publish API documentation to help internal and external consumers adopt your APIs.

4) In this example the Django application is giving JSON response to Postman using Class based view.

Note : In this we are using python built-in package "json"

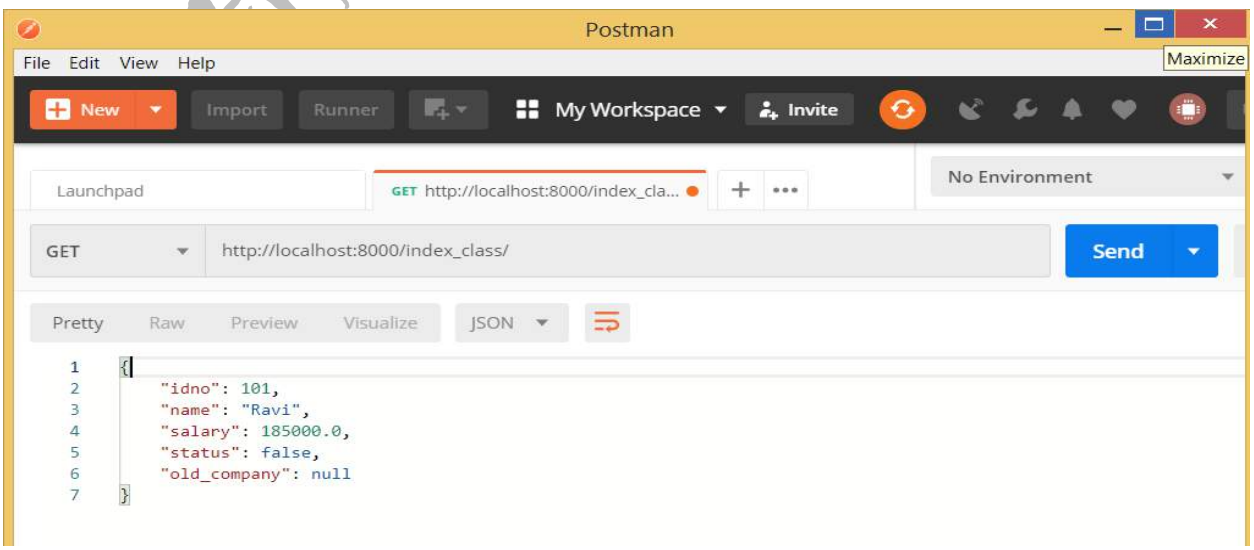
urls.py

```
path('index_class/',views.OpenMainPage.as_view(),name="index_class")
```

views.py

```
from django.views.generic import View
import json
class OpenMainPage(View):
    def get(self,request):
        emp_info = {"idno": 101, "name": "Ravi", "salary": 185000.00, "status": False,"old_company":None}
        json_data = json.dumps(emp_info)
        return HttpResponse(json_data,content_type="application/json")
```

Output



5) In this example the Django application is giving JSON response to Postman using Class based view.

Note: In this we are using django built in "JsonResponse"

urls.py

```
path('index_class/',views.OpenMainPage.as_view(),name="index_class")
```

views.py

```
from django.views.generic import View
from django.http import JsonResponse
class OpenMainPage(View):
    def get(self,request):
        emp_info = {"idno": 101, "name": "Ravi", "salary": 185000.00, "status": False,"old_company":None}
        return JsonResponse(emp_info)
```

Output