

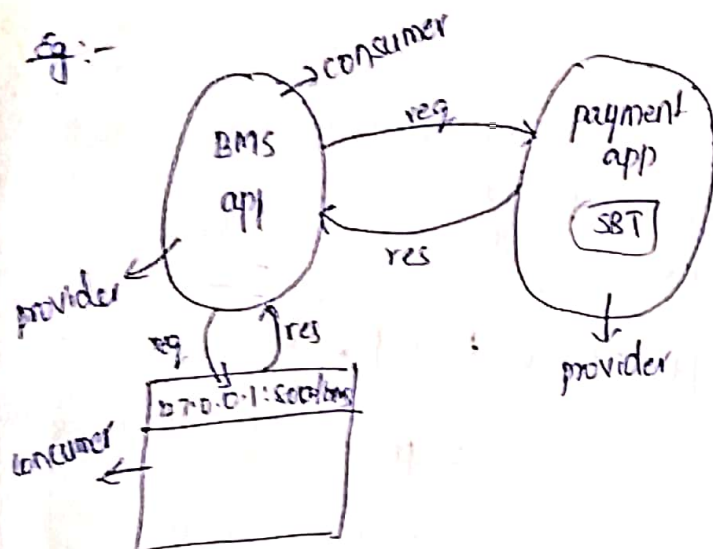
Web service provider:-

The application which is providing web services is called web service provider

Web service consumer:-

The application which is consuming the web services which are provided by other application is called web service consumer

eg:-



* Here when browser & booking show app are communicating then bms application providing services to browser so it is like a provider and browser acting like a consumer

* For payment purpose bms interacting to bank application so it is acting like a consumer now and bank provides services to bms so it acts like a provider

JSON & XML:-

→ JSON & XML are called universal languages for sharing the data between the multiple applications

→ If data is available in the form of JSON or XML that data can be understood by all the other languages like Java, Python, .Net...

→ JSON :- Java Script Object Notation

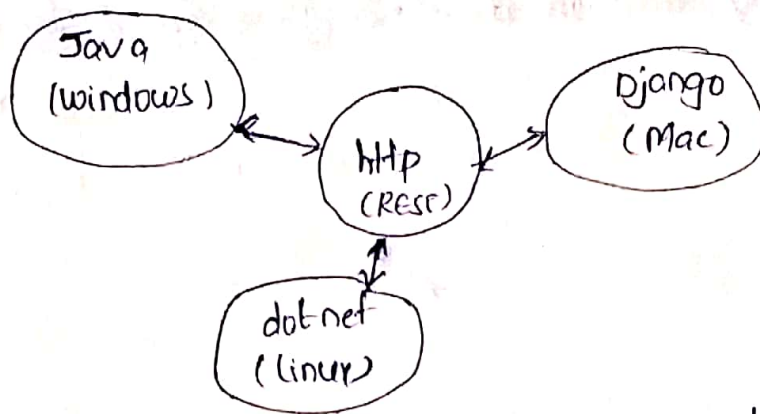
→ XML :- EXtensible Markup Language

→ HTTP :- Hyper text transfer protocol

→ HTTP protocol is acting like a universal protocol to sharing the data state (behaviour) along with request what we are to perform via HTTP methods

eg:- GET, PUT, POST, DELETE.....

→ HTTP protocol is communicating to all the applications irrespective of languages (Java, .Net, Python.....), irrespective of platforms



* create a django project its containing multiple views to converting dictionary to string, dictionary to json

step-1: project name: first project

step-2: application name: first app

step-3: open views.py

from django.shortcuts import render

from django.http import HttpResponse

def home(request):

msg = 'Welcome to home page'

return HttpResponse(msg)

creating function based view for converting dictionary data into string format

by using format method

def empview(request):

emp = { 'eno': 10,

'ename': 'shrinivas',

'esal': 10000,

'eaddr': 'Hyd'

}

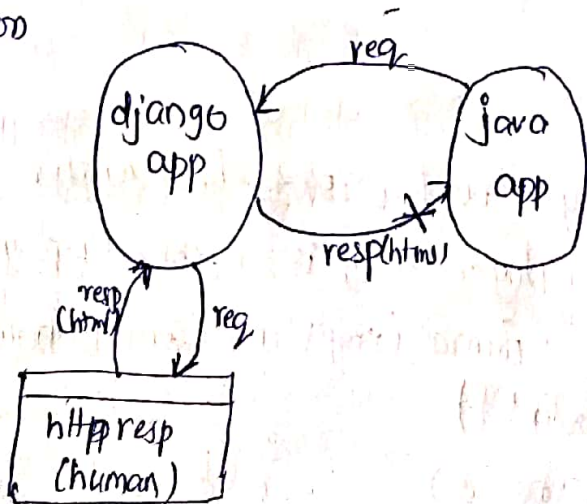
response = "<h1>Employee number is {eno},
 Employee name is {ename},
 Employee salary is {esal},
 Employee address is {eaddr}" . format(emp['eno'], emp['ename'], emp['esal'], emp['eaddr'])

return HttpResponse(response)

creating fbn for converting dictionary data into json format by using
dumps() of json module

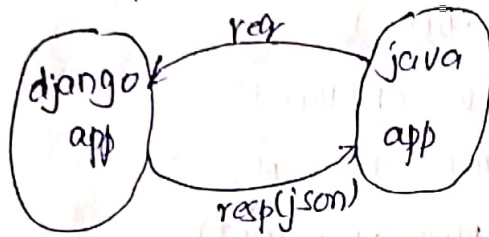
```
import json  
def jsonview(request):  
    emp = {  
        'eno': 20,  
        'ename': 'Virat',  
        'esal': 20000,  
        'eaddr': 'Delhi'  
    }  
    json_data = json.dumps(emp)  
    return HttpResponse(json_data, content_type='application/json')
```

- when ever we are sending the request from the browser as httprequest object server application will send back "http response" object in the form of ~~http~~ 'html response' format
- Always browser displays html format data only by default
- If we want to display another type data then we need to use the MIME (Multipurpose Internet Mail Extension) type attribute with name as 'content_type'
- Browser can easily understand the html response so that ~~any~~^{end} user can easily read the data, when we are displaying on browser
- it has a drawback if our application response with communicate to other application



* Here my java application will send the http request the my django app will send the html response to back (java app), then my java app will not understand this html response directly

- same way if any other application will send the request, it is also not understanding the html response
- That's why in our rest api we develop the code as should be any software application can able to understanding easily irrespective of languages (java, python,) or platforms (window, linux,)
- For that we required "json" response to make the communication between multiple applications



create fbr to converting dictionary data into json without dump method
by using JsonResponse()

```
from django.http import JsonResponse
def jsonview2(request):
```

```
    emp = {
        'eno' : 30
        'ename' : 'Rohit'
        'esal' : 30000
        'eaddr' : 'Mumbai'
    }
```

```
    return JsonResponse(emp)
```

→ Here JsonResponse() is return directly json response output by taking input as 'dict', without 'mime' type attribute (content-type)

NOTE :- json mime-type value is 'content-type = 'application/json'

xml mime-type value is 'content-type = 'application/xml'

html mime-type value is 'content-type = 'text/html'

step 4:- open project url.py and create the required url

```
from django.conf.urls import url
```

```
from django.contrib import admin
```

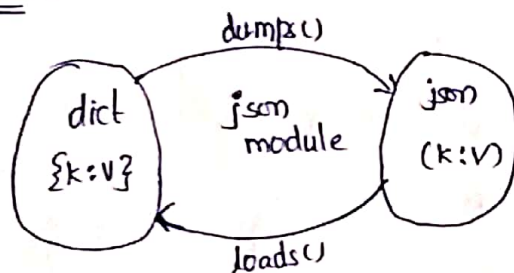
```
from django.urls import path
```

```
from django.views import views
```

```
urlpatterns = [  
    path('admin/', admin.site.urls),  
    path('home/', views.home),  
    path('empdata/', views.empview),  
    path('jsondata/', views.jsonview),  
    path('jsondata2/', views.jsonview2),  
]
```

step 5:- execute runserver command

JSON Module :-



dumps() :- To convert python dictionary object data into json data type we required to use dumps() method

loads() :- To convert json datatype into dictionary object type the we use loads() method.

→ Both methods are available in "json" module, it is coming along with python software, when we are installing

```
import json  
j = json.dumps(dict)  
d = json.loads(j)
```

* using command prompt api (API) checking till now we are sending the request from the browser only

a) it is possible to send http request from command prompt

Q:- yes, it is possible to testing the api's from the command prompt by using common like ~~the~~ http clients from

Ex:-

curl or http command line module (third party module)

* we need to install http module

* `cmd > pip install httpie`

* after installing third party module now we convert from command prompt

* when we are testing apis using command prompt our url should be containing prefix as http followed by required url

`cmd > http URL`

Ex:- http http://127.0.0.1:8000/home

NOTE:- if the server stops then command will not work. Exception like
http error connection error

Views concepts:-

Two types of views support in Django

1. function based view (fbv)

2. class based view (cbv)

* fbv created by def keyword followed by view name and it takes the request object as a parameter

`def vn(request)`

* cbv create by class keyword followed by class name and extend by predefined class `class cn(View):`