Date : 30th - Nov- 2020
Morning Session : 9am – 11.00 PM
By ~ Akash

Topics: REST API WITH NODEJS

```
////Setup to setup data
>download mongodb (https://www.mongodb.com/try/download/community)
> in c drive
    make one folder by the name of "data"
    inside data make one folder by the name of "db"

////Step to run mongodb

open cmd
> c:\programefile\mongodb\4.4\server\bin
   mongod (start mongodb server 27017)
> c:\programefile\mongodb\4.4\server\bin
    (mongo)
27017
```

SQL	mongodb
Database	Database
tables	collections
row	document

Check all database

Syn: show dbs

To use the database
Syn : Use databasaename

To check collections
Syn: Show collections

To find the all the records

Syn: db.collectionname.find() == select * from table

Create new database

Syn: use database

find nested Record:

```
//find nested record
db.hotels.find({"type.roomtype":"1",city_name:"Mumbai"},{name:1}).pretty()

///
db.hotels.find({cost:{$lt:5000}},{name:1,cost:1}).pretty()

db.hotels.find({cost:{$lt:5000,$gt:3000}},{name:1,cost:1}).pretty()

db.hotels.find({$and:[{city_name:"Mumbai"},{"type.roomtype":"4"}]},{name:1,city_name:1,_id:0}).pretty()
db.hotels.find({$or:[{city_name:"Mumbai"},{"type.roomtype":"4"}]},{name:1,city_name:1,_id:0}).pretty()
```

Update Mongodb:

```
//update in mongodb
db.users.update({_id:10},
    {
        $set:{
            name:'John',
            city:'NewDelhi'
        }
    },
    {
        upsert:true
    },
    {
        multi:true
    }
)
```

REmove:

```
//remove
db.collectionname.remove({})

db.users.remove({"name" : "Keev"})

//delete collection
db.collectionname.drop()
```

RESTAPI:

For RESTAPI packages to install

Body-parser, chai, chai-http, cors, ejs, express, mocha, mongodb

Chai , chai-http, mocha for testing

Body-parser , cors, ejs, express, mongodb for development

Representational State Transfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. REST API is a way of accessing the web services in a simple and flexible way without having any processing.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses the least bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API used only HTTP requests.

Working

A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE. After that a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image or JSON. But now JSON is the most popular format being used in Web Services.

In **HTTP** there are five methods which are commonly used in a REST based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

GET: The HTTP GET method is used to **read** (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

POST: The POST verb is most-often utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.
NOTE: POST is neither safe nor idempotent.

PUT: It is used for **updating** the capabilities. However, PUT can also be used to **create** a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. On successful update, return 200 (or 204 if not returning any content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation. PUT is not a safe operation but it's idempotent.

PATCH: It is used for **modify** capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource. This resembles PUT, but the body contains a set of instructions describing how a resource currently residing on the server should be modified to produce a new version. This means that the PATCH body should not just be a modified part of the resource, but in some kind of patch language like JSON Patch or XML Patch. PATCH is neither safe nor idempotent.

DELETE: It is used to **delete** a resource identified by a URI. On successful deletion, return HTTP status 200 (OK) along with a response body.

*** While coming for tomorrow class install postman in ur laptop/desktop**