***Express.js(expressjs.com)***

***Middleware*** functions are functions that have access to the [request object](http://expressjs.com/4x/api.html#req) (req), the [response object](http://expressjs.com/4x/api.html#res) (res), and the next middleware function in the application’s request-response cycle. The next middleware function is commonly denoted by a variable named next.

Middleware functions can perform the following tasks:

* Execute any code.
* Make changes to the request and the response objects.
* End the request-response cycle.
* Call the next middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.

The following figure shows the elements of a middleware function call:

|  |  |
| --- | --- |
| http://expressjs.com/images/express-mw.png | HTTP method for which the middleware function applies.  Path (route) for which the middleware function applies.  The middleware function.  Callback argument to the middleware function, called "next" by convention.  HTTP [response](http://expressjs.com/en/4x/api.html#res) argument to the middleware function, called "res" by convention.  HTTP [request](http://expressjs.com/en/4x/api.html#req) argument to the middleware function, called "req" by convention. |

Request object: it contains all the parameter send from the browser or frontend

Response object: it is used to send the data from the server to browser or front end

**Http** : is a protocol use to establish a communication between client and server

There are mainly 3 different ways to send data from client to server

1. **data appended to url**

Example:

<http://localhost:9998?city=banglore>

<http://localhost:9998/banglore>

1. **data appended to request body**

Example:

<http://localhost:9998/test>

body : {

“city” : banglore

}

1. **appending data to request headers as key value pairs**

**Http methods:**

1) GET

2) POST

1. PUT
2. DELETE

Retriving data from different types of request at server side

**Get:**

By using get method we can send data in two ways

1. Path params
2. Query params

Data or values send through get method can be visible in the url

**Path params :**

**Client:**

Path params is used to append values to url path directly.

“/ ” (slash ) is used to append the values to the url path.

Syntax: / followed by value

Example : /banglore

Example :

<http://localhost:9998/banglore>

In the above url “banglore ” is the path param value appended to url.

If we want to send multiple values through path param see following example

<http://localhost:9998/banglore/svuniversity>

**Server :**

How to retrieve values of path params at server side?

Url values are mapped with service path

-🡪 <http://localhost:9998/testpath/banglore/svuniversity> ( url is also named as web service)

🡪 app.get(‘/testpath/:city/:collage’,function(req,res){

Banglore is stored in city and svuniversity is stored in collage

By using city and collage key word we retrieve the values from req.params object like

req.params.city

req.params.collage

**Example :**

app.get(‘/testpath/:city/:collage’,function(req,res){

var city = req.params.city; //path param values are stored in req.params object.

res.send(city) // “banglore” is send as response to client

})

**Query params:**

Difference between the path and query params are

1) path params don’t have key .we have to append the value directly to url

2) Query params contains key and values in the url.

**Client :**

Syntax: http://localhost:port/testpath?key=value&key=value

Example: <https://localhost:9998?city=banglore&collage=university>

**Server:**

Key and value pairs are stored in the req.query object.

We need to access the values of the req.query object using key present in the url

Example:

app.get(‘/testpath’,function(req,res){

var city = req.query.city;//retrieving data by using key “city ”from req.query object

res.send(req.query.collage)

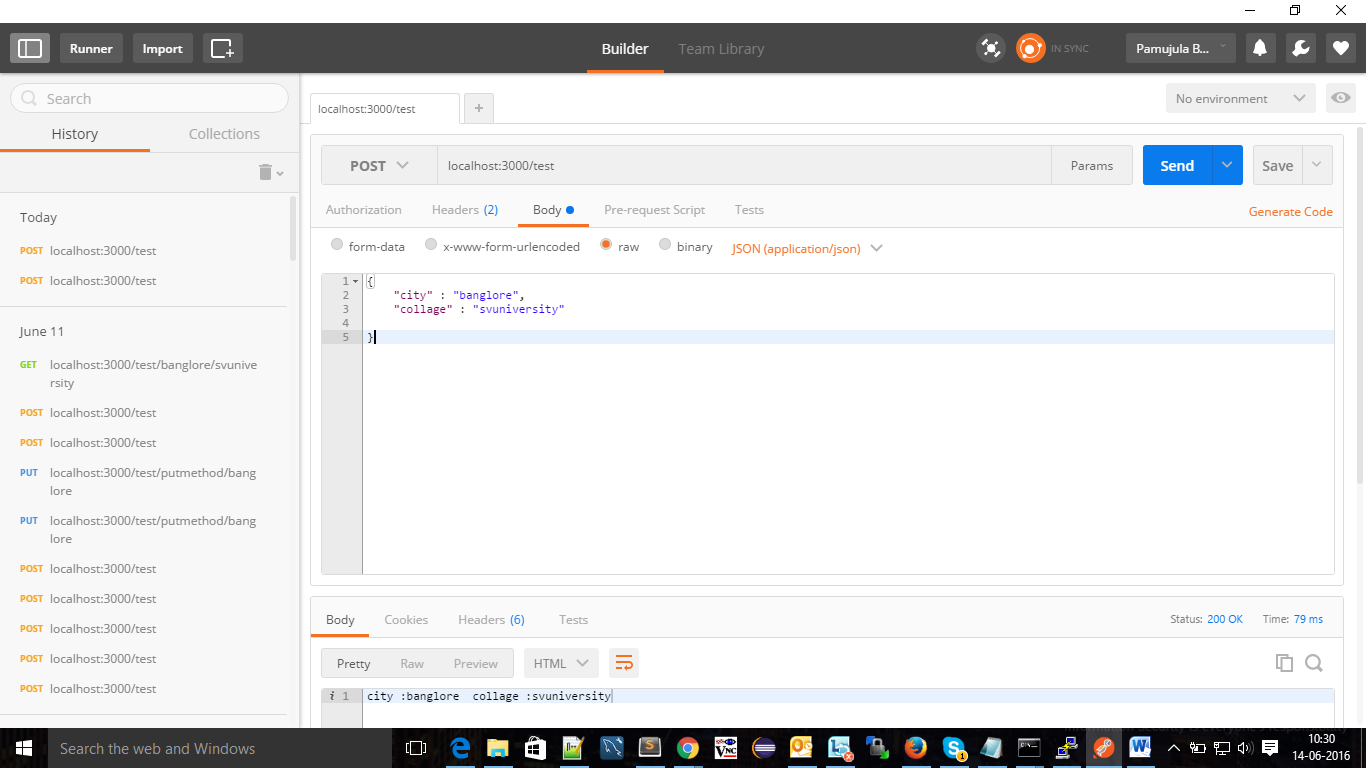
})

POST: Is used when we want to send data through the body of the request. Data or values can not be visible in the url as data goes in the body part.

NOTE: when sending data throught POST method ,we need to configure header values as follows

IN POSTMAN:

* Select headers
* Next in keys column enter ‘content-type’ without quotes
* Next in values column enter ‘application/json’ without quotes
* Select “body”
* Next select “raw”
* Next enter data in body part as shown below



**PUT** : is similar to post method ,it is considered as modified version of data already exist in server

To update any values present in the server ,we send data through put method

We can send data through url,body,headers

**Delete** : Is used to delete data on server

We can send data through url,body,headers