****

JS Promises

Node.JS

- **Introduction**
- **Modules**
- **Http server**
- **Http get**
- **Http post**
- **Express**
- **Express get**
- **Express post**
- **Express url parameters**
- **Imports and exports in express**
- **Routing in express**

****

==========

Promise:-

==========

- Promises are used to handle asynchronous operations in Javascript.
- Promises are special Javascript objects.
- Promises have two states
    - Success (resolve)
    - Failure (reject)
- Promises can be created using the object of 'Promise' class.
- Promise is the predefined class in Javascript.
- Promises can be consumed using 'then()'

```
//Eg01
//creating promise
let myPromise = new Promise((resolve, reject)=>{
    resolve('Tomorrow i will be at home')
})
//consume promise
myPromise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})

//Eg02
let myPromise = new Promise((resolve,reject)=>{
    setTimeout(()=>{
        resolve('Success')
    },5000)
})
myPromise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})
```

```
//Eg03
let myPromise = new Promise((resolve, reject)=>{
    reject('Failure')
    resolve('Success')
})
myPromise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})


//Eg04
let myPromise = new Promise((resolve, reject)=>{
    setTimeout(()=>{
        resolve('Success')
    },6000)
    setTimeout(()=>{
        reject('Failure')
    },5000)
})
myPromise.then((posRes)=>{
    console.log(posRes)
},(errRes)=>{
    console.log(errRes)
})
```

==========

async and await

==========

- Netscape released the above keywords in ES9.
- Above keywords are used to increase code readability.
- These keywords increase application performance.

```
//Eg05
let myPromise = new Promise((resolve, reject)=>{
    resolve('Hello')
})
async function myFun(){
    let res = await myPromise
    console.log(res)
}
myFun()

//Eg06
function add(num) {
    let myPromise = new Promise((resolve, reject) => {
        resolve(num + 5)
    })
    return myPromise
}
function sub(num) {
    let myPromise = new Promise((resolve, reject) => {
        resolve(num - 3)
```

```
    })
    return myPromise
}
async function myFun() {
    let res1 = await add(5)
    let res2 = await sub (res1)
    console.log(res2)
}
myFun()

//Eg07 Solution for Callback hell ?
function add(num) {
    return new Promise((resolve, reject) => {
        resolve(num + 5)
    })
}
function sub(num) {
    return new Promise((resolve, reject) => {
        resolve(num - 3)
    })
}
function mul(num) {
    return new Promise((resolve, reject) => {
        resolve(num * 4)
    })
}
function div(num) {
    return new Promise((resolve, reject) => {
        resolve(num / 2)
    })
}
async function myFun(){
    let addRes = await add(5)
    let subRes = await sub(addRes)
    let mulRes = await mul(subRes)
    let divRes = await div(mulRes)
    console.log(divRes)
}
myFun()
```

==========

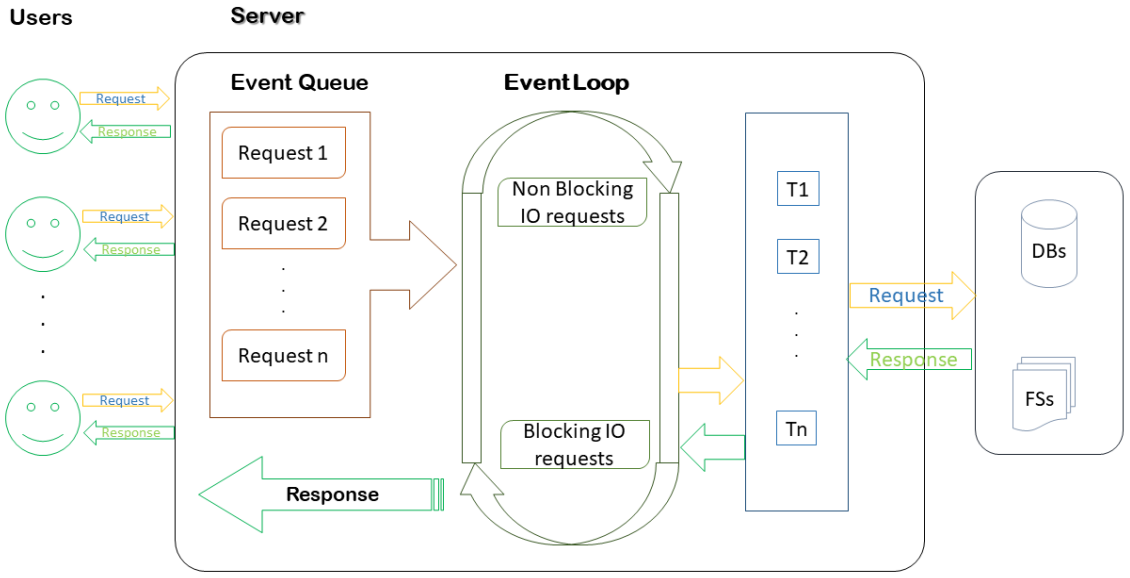
Node.js

==========

Introduction:-

- Node.JS is open source, cross platform, Javascript runtime environment.
- Node applications can be developed using either Javascript or Typescript.
- Node.JS was released by Ryan Dahl on 27th May 2009, at netscape.
- Latest version of Node.JS is 20.5.1 09-08-2023
- Current stable version is 18.17.1 08-08-2023

The applications (servers) developed by Node.JS are called 'Single Threaded Event Loop' applications.



1 Modules in NodeJS

- Node supports modules
- Predefined modules
- Custom modules

2 http

- This is the native module.
- This module is available along with 'Node Engine'.
- This module is used to develop http servers.

3 url

- This is native module.
- This module is used to read get parameters in http servers.

4 query-string

- This is native module.
- This module is used to read the post parameters in http servers.

5 fs

- This is native module.
- fs stands for 'File system'.
- This module is used to interact with flat files.
- Eg:- txt, xml, json, etc

6 express

- This is third party module.
- This module is used to develop 'ReST APIs' (web services.)
- ReST API:- Representational State Transfer Application Programing Interface.

7 # mysql

- This is third party module.
- This module is used to interact with My Sql database.
-

8 # mongodb

- This is third party module.
- This module is used to interact with mongodb without schema.
- [Note:- rules and regulations of db are called as schema].
-

9 # mongoose

- This is third party module.
- This module is used to interact with mongodb with schema.
-

10 # mssql

- This is third party module.
- This module is used to interact SQL Server.
-

11 # multer

- This is third party module.
- This module is used to upload images to server.
-

12 # socket.io

- This is third party module.
- This module is used to develop chat applications.
-

13 # jwt-simple

- This is third party module.
- This module is used to generate tokens for authentication purpose.
- This system is technically called as Token based authentication system. •

14 # body-parser

- This is third party module.
- This module is used to set MIME type.
-

15 # cluster

- This is third party module.
- This module is used to implement child process in http server.
- implementing child process is called as load balancing.

16 # express-cluster

- This third party module.
- This module is used to implement load balancing in ReST APIs.

17 cookie-parser

- This is third party module.
- This module is used to work with cookies.

- We can download all third party modules by using either 'npm' or 'yarn' tool.
- npm stands for 'Node Packaging Manager'.
- npm is an integrated tool for NodeJS.
- yarn is the latest tool used to download 'Node modules'
- yarn is faster as compared to npm.
- All node modules will be downloaded to the 'node_modules' folder in the current path.
- we can start node server in 3 ways
  >node server
  >npm start
  >nodemon server (monitoring / watch mode)

Environmental setup

1. Download and install nodejs
   https://nodejs.org/en/
2. Download and install git
   https://git-scm.com/
3. Download and install VSCode
   https://code.visualstudio.com/
4. Download and install postman (DO NOT SIGN IN OR SIGNUP)
   https://www.postman.com/downloads/
5. Install yarn tool using following command
   >npm install -g yarn
   npm :- node packaging manager
   -g :- global installation

- Create a folder rename it as 'NodeJS'
- in that folder create one more folder 'modules eg'
- open 'modules eg' folder in VSCode
- create server.js file there

****
Download follwoing modules using yarn tool
****

1. express
2. mysql
3. mongodb@2.2.32
4. multer
5. jwt-simple

>npm init -y

1. express >yarn add express --save

2. mysql >yarn add mysql --save
3. mongodb@2.2.32 >yarn add mongodb@2.2.32 --save
4. multer
5. jwt-simple
>yarn add multer jwt-simple --save

==========
Implementing HTTP server
==========
- 'http' is the predefined module used to create http servers.
- http is the native module, so no need to download it.
- 'require()' is used to import.
- Eg let http = require('http')
- 'createServer()' is the predefined function in the http module.
- createServer is used to create http server.
- the argument to createServer is the arrow function.
- to this arrow function there are two arguments, 'req' and 'res'.
- request and response objects provided by node engine respectively.
- req object is used to store client data.
- res object is used to send response to client.
- 'writeHead(-,-)' is the predefined function in res object.
- writeHead function is used to set the MIME type.
- First argument to writeHead function is status code (200 - ok)
- Second argument to the writeHead function is JSON object.
- JSON key is 'content-type' and the value is 'text/html'.
- 'write(-)' is the predefined function in res object.
- write function is used to append response to res object.
- 'end()' is the predefined function in res object used to lock the response.

```
let http = require('http')
let server = http.createServer((req, res) => {
    //set MIME type
    res.writeHead(200, { 'content-type': 'text/html' })
    res.write('<h1> Welcome to http server</h1>')
    res.end()
})
//assign port number
server.listen(8080)
console.log('Server listening port no 8080')
/*
    start server
    >node server
     url http://localhost:8080
*/
```

==========
HTTP get parameters
==========
- 'url' is the predefined module in node.
- url module is used to read get parameters in http server.

```
//import http module
let http = require('http')
//import url module
let url = require('url')
let server = http.createServer((req, res) => {
    let obj = url.parse(req.url, true).query
    //set MIME type
    res.writeHead(200, { 'content-type': 'text/html' })
    let uname = obj.uname
    let upwd = obj.upwd
    if (uname == 'admin' && upwd == 'admin')
        res.write('<h1>Login Success</h1>')
    else
        res.write('<h1>Login Failed</h1>')
    res.end()
})
//assign port no
server.listen(8080)
console.log("Server listening port no 8080")

//url :- http://localhost:8080/?uname=admin&upwd=admin
```

==========

HTTP post parameters

==========

- 'querystring' is the predefined module in nodejs.
- querystring module is used to read post parameters in http server

server.js

```
//import http module
let http = require('http')
//import query string module
let qs = require('querystring')
//create server
let server = http.createServer((req, res) => {
    //set MIME type
    res.writeHead(200, { 'content-type': 'text/html' })
    let body = ""
    //listen post parameters
    req.on("data", (result) => {
        body = body + result
    })
    //end call back to node engine
    req.on("end", () => {
        obj = qs.parse(body)
        //read post parameters
        let uname = obj.uname
        let upwd = obj.upwd
        if (uname === 'admin' && upwd === 'admin')
            res.write("<h1 style = 'color:Green'>Login Success</h1>")
        else
            res.write("<h1 style = 'color:red'>Login Failed</h1>")
        res.end()
```

```
    })
})
//assign port no
server.listen(8080)
console.log("Server Listening port no 8080")
```

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <link rel="stylesheet" href="style.css">
    </head>
    <body>
        <form action = "http://localhost:8080" method="post" class = box>
            <h1>Login</h1>
            <input type="text" placeholder="Enter Username" name="uname">
            <input type="password" placeholder="Enter Pasword" name = "upwd">
            <input type="submit" value="Login">
        </form>
    </body>
</html>
```

style.css

```
body
{
    background: radial-gradient(circle, white, black);
    font-family: sans-serif;
}
h1
{
    color: white;
    text-transform: uppercase;
    font-weight: normal;
}
.box
{
    background-color: black;
    width: 300px;
    margin: 50px auto;
    padding: 40px;
    border-radius: 20px;
    text-align: center;
}
input
{
    margin: 20px auto;
    text-align: center;
    padding: 14px 10px;
    width: 200px;
    border-radius: 24px;
    background: none;
}
input[type = "text"], input[type = "password"]
{
```

```
    border: 2px solid skyblue;
    color : lightyellow;
}
input[type = "submit"]
{
    border: 2px solid burlywood;
    color:white;
    cursor: pointer;
}
```

==========

Express

==========

- Download express module
> yarn add express --save

```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express() //where app is rest object
//create get request
app.get("/", (req, res) => {
    console.log('Default get request')
    res.send({ 'message': 'Default get request' })
    //res.json({'message':'Default get request'})
})
app.get("/fetch", (req, res) => {
    res.json({ 'message': 'fetch get request' })
})
app.post("/", (req, res) => {
    res.send({ 'message': 'default post request' })
})
//create one more post request
app.post("/login", (req, res) => {
    res.send({ 'message': 'login post request' })
})
//create port
let port = 8080
//assign port no
app.listen(port, () => {
    console.log('Server listening port no ', port)
})

/*
    Test urls with postman
    http://localhost:8080           Default GET
    http://localhost:8080/fetch     fetch GET
    http://localhost:8080           Default POST
    http://localhost:8080/login     login POST
*/
```

==========
Reading get parameters in express
==========

//url :- http://localhost:8080/login/?uname=admin&upwd=admin

```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//create rest api
app.get("/login", (req, res) => {
    //query is the predefined key in req object
    //query is used to read get parameters
    let uname = req.query.uname
    let upwd = req.query.upwd
    if (uname == 'admin' && upwd == 'admin')
        res.json({ 'login': 'Success' })
    else
        res.json({ 'login': 'Failed' })
})
//assign port no
app.listen(port, () => {
    console.log('Server listenig port no ', port)
})
```

==========
Reading Express Post parameters
==========
Download express and body-parser modules

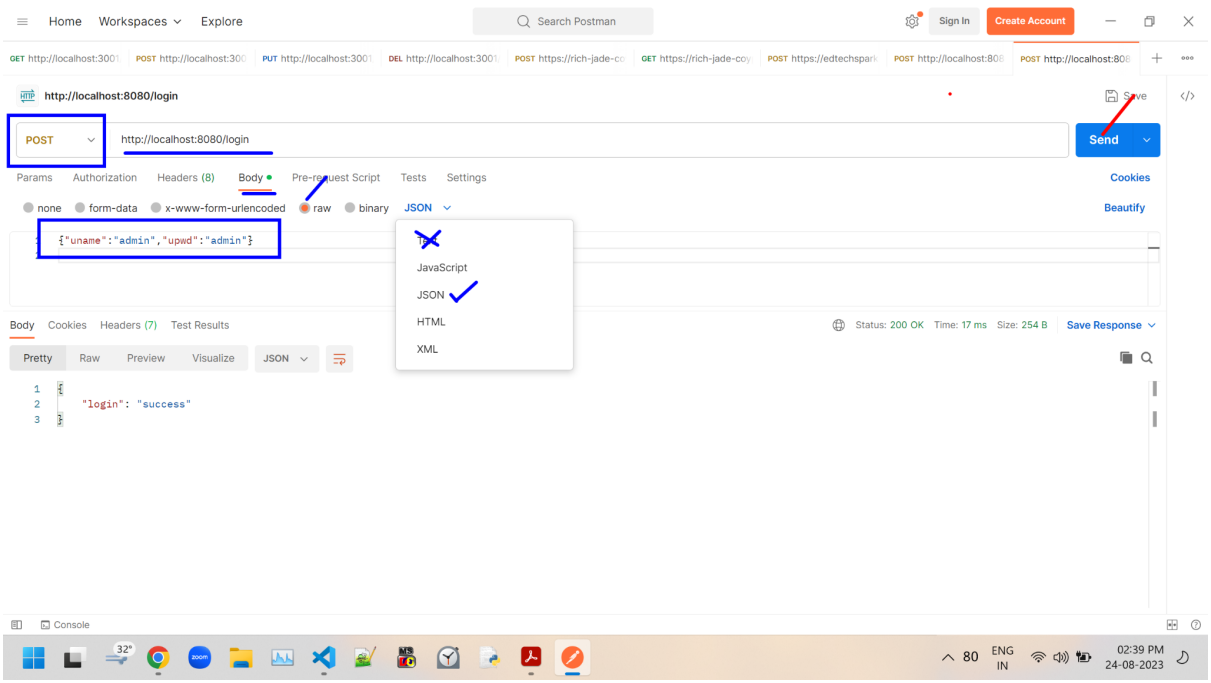
```
//initialyse project
//>npm init -y
//download express module
//Download express and body-parser modules
//>yarn add express body-parser --save
//import modules
let express = require('express')
let bodyparser = require('body-parser')
//create rest object
let app = express()
//create port
let port = 8080
//set JSON as MIME type
app.use(bodyparser.json())
//front end encoading form data
app.use(bodyparser.urlencoded({ extended: false }))
//create rest api
```

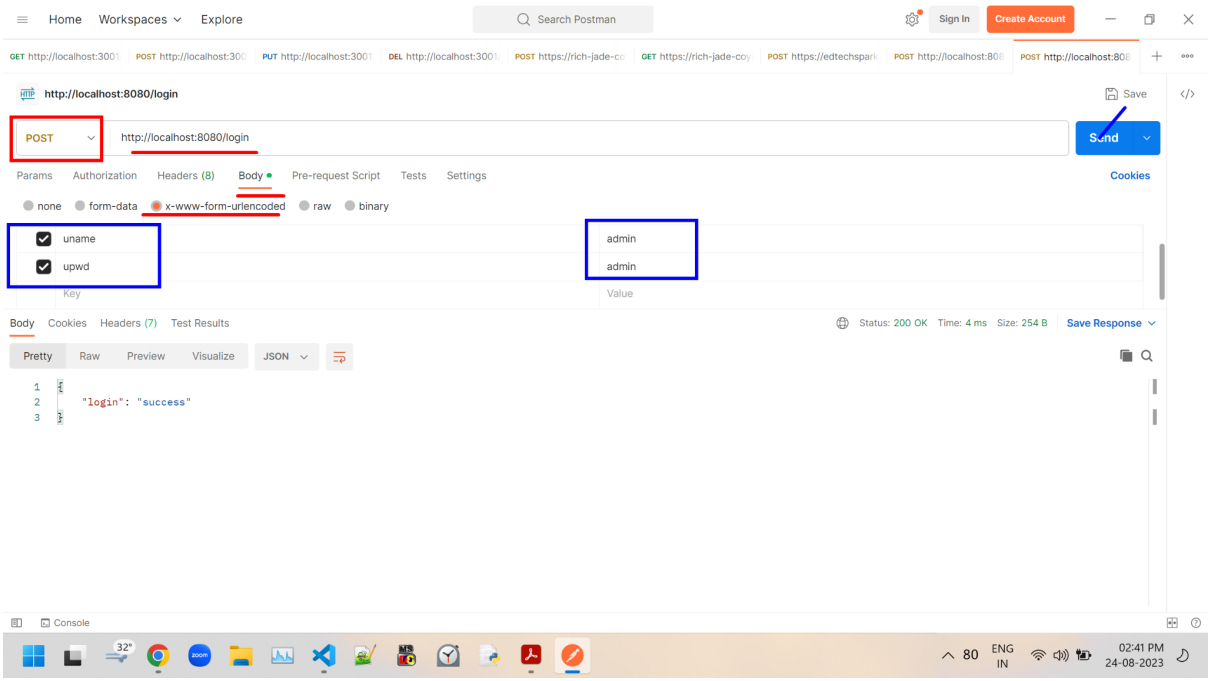
```
app.post("/login", (req, res) => {
    //client parameters are stored in body part of req
    let uname = req.body.uname
    let upwd = req.body.upwd
    if (uname == 'admin' && upwd == 'admin')
        res.send({ 'login': "success" })
    else
        res.send({ 'login': 'failed' })
})
//assign port no
app.listen(port,()=>{
    console.log('Server listening port no ', port)
})

/*
    Start server
    >node server
    Test Rest api in postman with url
    http://localhost:8080/login
    in postman
    1.  req     -> post
        body    -> raw
                -> text <-> json
                {"uname":"admin","upwd":"admin"}

        2.  req     -> post
            body    -> x-www...
                    -> key and values

    Figs
*/
```





==========

Reading url parameters from express

==========

```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//url:http://localhost:8080/login/admin/admin
app.get("/login/:uname/:upwd",(req, res)=>{
    //params is the predefined key used to read parameters from url
    let uname = req.params.uname
    let upwd = req.params.upwd
    if(uname === 'admin' && upwd === 'admin')
        res.json({'login':'success'})
    else
        res.json({'login':'failed'})
})
//assign port no
app.listen(port,()=>{
    console.log(`Server listening port no :- ${port}`)
})
```

Imports and exports

- module is the predefined object in node.
- exports is the predefined key in module object.
- exports key is used to export (JSON object or function)

Eg01
<>
config
- db_config.js
- server.js

db_config.js
```
module.exports = {
    "host": "localhost",
    "user": "root",
    "password": "root",
    "database": "nodedb",
    "table": "products"
}
```

server.js
```
//initialyse project
//>npm init -y
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//create port
let port = 8080
//import db_config
let obj = require('./config/db_config')
app.get("/", (req, res) => {
    res.json(obj)
})
//assign port no
app.listen(port, () => {
    console.log("Server listening port no ", port)
})
```

Eg02
<>
config
- my_fun.js
- server.js

my_fun.js
```
module.exports = (arg1, arg2) => {
    if (arg1 == 'admin' && arg2 == 'admin')
        return "Login success"
    else
        return "Login Failed"
}
```

server.js
```
//initialyse project
//>npm init -y
```

```
//download express module
//>yarn add express --save
//import express module
let express = require('express')
//create rest object
let app = express()
//import my_fun
let my_fun = require("./config/my_fun")
app.get("/login", (req, res) => {
    res.send(my_fun(req.query.uname, req.query.upwd))
})
//assign port no
app.listen(8080)
console.log("Server listening port no 8080")

//url :- http://localhost:8080/login?uname=admin&upwd=admin
```

==========

Modules in Express

==========

<>

- login
  - login.js
- logout
  - logout.js
- server.js

//initialyse project
//>npm init -y
//download express module
//>yarn add express --save

***login.js**

```
//import express module
let express = require('express')
//create router instance
let router = express.Router()
//create get request
router.get("/", (req, res) => {
    res.json({ 'message': 'Welcome to login module' })
})
//create one more get request
router.get("/login/:uname/:upwd", (req, res) => {
    //here we are reading url parameters using params
    let uname = req.params.uname
    let upwd = req.params.upwd
    if (uname == 'admin' && upwd == 'admin')
        res.json({ 'login': 'success' })
```

```
    else
        res.json({ 'login': 'failed' })
})
//export router
module.exports = router
```

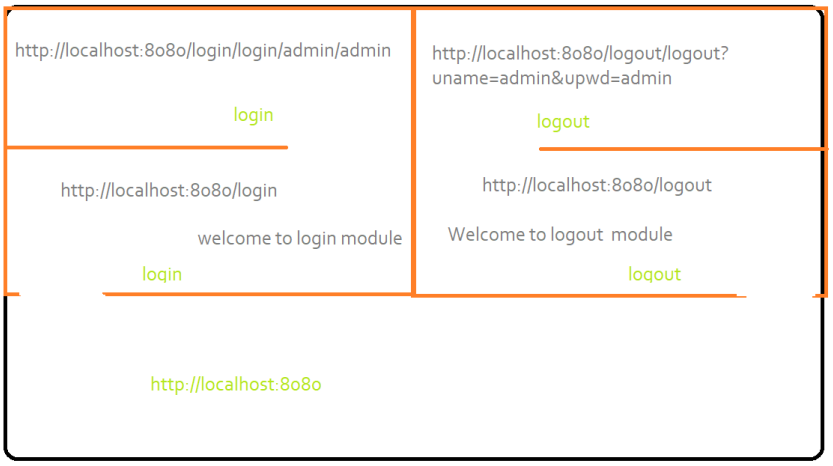
logout.js

```
//import express module
let express = require('express')
//create router instance
let router = express.Router()
//create get request
router.get("/",(req,res)=>{
    res.json({'message':'Welcome to logout  module'})
})
//create one more get request
//URL :- http://localhost:8080/logout/logout/?uname=admin&upwd=admin
router.get("/logout",(req,res)=>{
    //here we are reading get parameters using query
    let uname = req.query.uname
    let upwd = req.query.upwd
    if(uname == 'admin' && upwd == 'admin')
        res.send({'logout' : 'Success'})
    else
        res.send({'logout' : 'Failed'})
})
//export router
module.exports = router
```

server.js

```
//import modules
let express = require('express')
let login = require('./login/login')
let logout = require('./logout/logout')
//create rest object
let app = express()
//use modules
app.use("/login", login)
app.use("/logout", logout)
//assign port no
app.listen(8080)
```

```
console.log("Server listening port no 8080")
/*
http://localhost:8080/login
http://localhost:8080/login/login/admin/admin
http://localhost:8080/logout
http://localhost:8080/logout/logout?uname=admin&upwd=admin
*/
```



For Practice only

VCS:-

- Version control system
- Eg clearcase, vss, cvs, github, etc.

Why VCS ?
- Control versions -> safty(updates)
- Can perform parallel development.
- Development time reduced.
- Insreased Productivity.

Github
- opensource reposiotry by microsoft.
- 'git' is the tool used to work with github.

Github operations
1. Installation
2. Create Local Repository.
3. Add user credentials to repository.
4. Create files to repository.
5. Commit changes to repository.
6. Generate Branches.
7. Merge Branches.

1. Installation
download from
https://git-scm.com/downloads
Follow onscreen messages and install.

2. Create local repository
- new folder -> test
- rt clk -> git bash here
- $git init
- 'test' folder converted to local repository
- $git config --list -> list of git repository properties

3. Add user credentials to repository.
- add name
$ git config --global user.name '<any username>'
- add email
$ git config --global user.email '<any email>'

4. Create files to local repository
- $ touch <>
- $ touch sample.txt
- sample.txt is unsaved / untracked / uncommited
- $ git status
- gives files in red colour.

5. commit changes
- $ git add .
- $ git commit

        - press 'I' for insert mode
        - type message 'initial commit'
        - press 'Esc'
        - type ':wq'
    - from second commit onwords
        - git add .
        - git commit -m <relavent message>
    - check log
        - $ git log

6. Generate Branches
    - git branch 'demo'
        - by default all files from master are available here.
    - $ git checkout demo

7. Merge branches
    - $ git checkout master
    - $ git merge demo