MongoDB

--start mongodb

1.to start MongoDB data base

path>mongod --dbpath="D:\Vittal\MongoDBData"

2. to get mongodb console

path>mongo

connection syntact from nodeJS

URL : mongodb://localhost:27017/TodoApp

mongodb://hostName:port/port{27017}/databaseName

-->if MongoDB is running from the machine then the above connection will successfull otherwise through error.

-->If MongoDB running, if "TodoApp" database is not there in the MongoDB even thoug it will connect and this is the feaute of MongoDB

-->Until you insert the collection in the database , it will not list in the MongoDB

OBJECT Destructuring in ES6

Pull out OBJECT and assigning to a viable

var user ={name:"Vittal",age:28};

var {name} =user //-->var name = user.name;

console.log(name);

MONGOOSE:

1. Mongoose is a third party npm component and act as ORM(Object Relational Mapping)

2. It can be used for user authention , passowrd validation , property validation but writing all these boilerpalting instead we

use Mongoose and it comes up with lot many features

3. it helps us to structure the data.

4. Search for Mongoose validator :http://mongoosejs.com/docs/validation.html

5. Default type casting does exist in Mongoose. If Default schema of data model's object is string then it can accept value like

boolean values which default wrapped in double quotes and also accept int values which wrapped in double quotes as well.

For example:

var TodoModel = Mongoose.model({

text:{

type:String,

}

});

but text an accept values like true/false and numbuers as well but which are wrapped in double quotes.

var NewTodoInsertData = new TodoModel({

text:true / text:123 / text:"some string"

});

NewTodoInsertData.save().callbackPromiseFn();

BODY-PARSER

1. this is npm component and it parses the body of request which comes from cleint and send to server.

2. Converts basically JSON string from client req object to JSON object to server.

TESTING

convert object id to string, while testing /todos/:id route

\_id.toHexString()

Deleting Records:

Todo.findOneAndRemove({anyKey:val});delete and returns the deleted doc

Todo.findByIdAndRemove({\_id:val});delete and returns the deleted doc

OR

Todo.findByIdAndRemove('id in string');delete and returns the deleted doc

Todo.remove({});delete all docs

MongoDB URL work around

https://www.udemy.com/the-complete-nodejs-developer-course-2/learn/v4/questions/1777812

Questions to be asked

1. While unit testing rest api, we operating the DB transanction on application database but it could not be actual scenario rt

2. Code coverage report

3. When we have to use catch() block and when we have to use then() method with err & res params, while handling promises

with then

user.save().then((user)=>{

res.send(user);

},(err)=>{

res.status(400).send(err);

});

with catch block

user.save().then((user)=>{

res.send(user);

}).catch((err)=>{

res.status(400).send(err);

)}

While running test command in windows, update the following line in package.JSON

"test": "export NODE\_ENV=test || \"SET NODE\_ENV=test\" && mocha server/\*\*/\*.test.js",

For mLab(online free mongodb) login to https://mlab.com/

with username/passowrd for the account

@@@ ARROW function @@@

()=>{}

Arrow function does not support this keyword

@@@@ INSTANCE METHOD and MODEL METHOD @@@

🡪Instance method, small u(user)

🡪instance method is responsible a document or records

UserSchema.**methods**.generateAuthToken = function(){

var user = this;

};

🡪Model method, capital letter U(User)

Model method is responsible for entire model or enitre schema

UserSchema.**statics**.findByToken = function(){

};

How to configure ES Lint

What is ESLint ?

A **Lint** or a **Linter** is a program that supports **linting** (verifying code quality). They are available for most languages like **JavaScript**, CSS, HTML, Python, etc.. ... **lint** is a tool that is used to mark the source code with some suspicious and non-structural (may cause bug)

ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code. In many ways, it is similar to JSLint and JSHint with a few exceptions:

* ESLint uses [**Espree**](https://github.com/eslint/espree) for JavaScript parsing.
* ESLint uses an AST to evaluate patterns in code.
* ESLint is completely pluggable, every single rule is a plugin and you can add more at runtime.

This linting is for NodeJS development or pure javascript development.

We configured ESLint with google standards from <https://github.com/google/eslint-config-google>

How to set up the environment

1. npm install eslint --save-dev

npm install –g eslint --save-dev

1. create a file called “.eslintrc.json” in root dir of folder

and following values to the file

{

"extends": ["eslint:recommended", "google"],

"parserOptions": {

"ecmaVersion": 6,

"sourceType": "module",

"ecmaFeatures": {

"jsx": false

}

},

"rules": {

"semi": 2

},

"env": {

"node": true

},

"root": true

}

1. Install google eslint plugin

npm install --save-dev eslint eslint-config-google –g

1. Run following command to generate lint report

Root\_project\_dir > eslint \*\*/\*.js > lint-report.txt

1. Explore the **lint-report.txt** file from root directory of project and you can see the list of linting report file wise.

*Happy linting to produce quality code….*