API Development

CREATING A SIMPLE CRUD API USING NODEJS

H.K. Sadeepa Diluk Jayatissa | SOFT 355 | 10516205 | April 2, 2016

Introduction

What is an API?

An API, a common slang used in modern computer programming society. And it's known to make the workload of the programmer easy. So what is this API really is

API stands for Application programming interface. In other words a set of standards, programming instructions (list of commands) and format of commands that one program can send to another program or web tool to request information or send information or update information. For example API can be taken as a building block. And programmer just has to put the right blocks together to make the process of the program easier.

The API is not a user interface, it's an interface between software's that can communicate between each other with or without the user's knowledge.

Description about my API

For this module we have to create a simple API that can do CRUD (Create, Read, Update and Delete) functions that is RESTful and can represent data in JSON format. So I chose NODE, Express and Mongoose to interact with MongoDB and passport for best practice and tools. So this API allow users to Authenticate, perform CRUD operations.

Installing necessary software and tools

For this I use Linux Mint 17.1 as the development environment because it's easy to install and configure those software's and tools

Install NODE

1. First open a terminal and enter these codes

```
curl -sL https://deb.nodesource.com/setup_5.x | sudo -E bash - sudo apt-get install -y nodejs
```

   this will download and install NODE automatically

2. Then install essential build tools to make easy to compile and install native add-ons from NPM

```
sudo apt-get install -y build-essential
```

3. to check the current node version type this in terminal

```
node –v or node --version
```

Installing NPM

Npm is a package manager that manage NODE's packages that needs for development using nodejs

4. install npm by using this command on terminal. When installing node needed paths and working version of npm already installed. So we can use npm to update npm itself

```
sudo npm install npm
```

or install npm using apt-get

```
sudo apt-get install npm
```

5. check the current version of npm

```
npm –v or npm – version
```

Install MongoDB and Mongoose

6. install MongoDB using following command in terminal

```
sudo apt-get install –y mongodb-org
```

7. then install Mongoose by following command

```
sudo apt-get install mongoose --save
```

8. then add following line to server.js to connect mongodb with the API

```
// Connect to the vinelocker MongoDB
mongoose.connect('mongodb://localhost:27017/vinelocker');
```

Install express using npm

```
npm install –g express-generator
```

Express is a java script framework that build java script projects automatically

Install postman

Postman is a powerful HTTP client that can use to test web services efficiently. Since the API doesn't have a user interface we'll use postman to create HTTP request to API to check it's functionality.

Creating the project

Using the express a template for nodejs project can be easily created. For that first must create a directory and switch to it

```
mkdr node
cd node
```

then using express create the project template

```
express vinelocker
```

this will create something like this on terminal

```
express vinelocker
create : vinelocker
    create : vinelocker/package.json
    create : vinelocker/app.js
    create : vinelocker/public
    create : vinelocker/public/javascripts
    create : vinelocker/public/images
    create : vinelocker/public/stylesheets
    create : vinelocker/public/stylesheets/style.css
    create : vinelocker/routes
    create : vinelocker/routes/index.js
    create : vinelocker/routes/users.js
    create : vinelocker/views
    create : vinelocker/views/index.jade
    create : vinelocker/views/layout.jade
    create : vinelocker/views/error.jade
    create : vinelocker/bin
    create : vinelocker/bin/www

    install dependencies:
     > cd vinelocker && npm install

    run the app:
     > SET DEBUG=vinelockerb:* & npm start
```

This is the file template of the project we develop

Open the package.json file and update following codes to add external dependencies to the project

```
{
  "name": "vinelocker",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
```

```
    "bcrypt-nodejs": "0.0.3",
    "body-parser": "^1.13.3",
    "cookie-parser": "~1.3.5",
    "debug": "~2.2.0",
    "express": "^4.13.4",
    "jade": "~1.11.0",
    "mongoose": "^4.4.12",
    "morgan": "~1.6.1",
    "serve-favicon": "~2.3.0"
  }
}
```

Then open a terminal in vinelocker folder and enter this

```
npm install
```

This will install dependencies that needed for the API

Create server.js

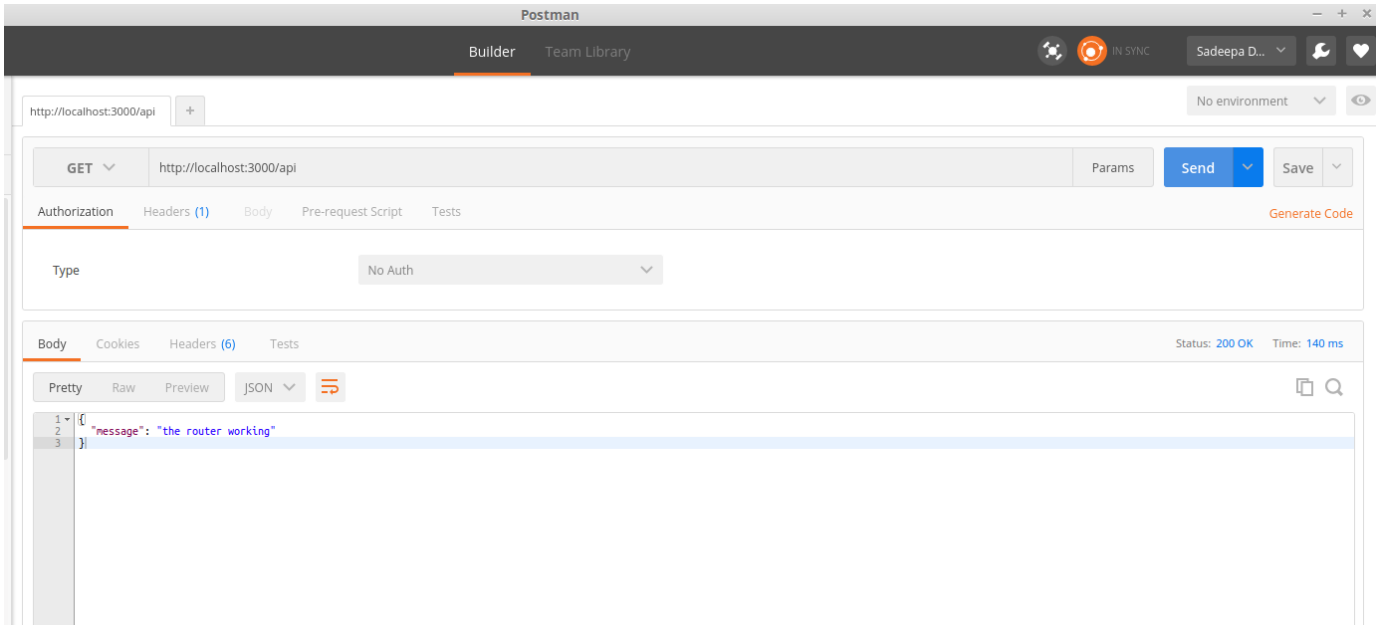
Rename the app.js to server.js and open it, then modify the code to following to check the connectivity of API

```
var express = require('express');
var app = express();
var port = process.env.PORT || 3000;
var router = express.Router();

// Initial dummy route for testing
// http://localhost:3000/api
router.get('/', function(req, res) {
  res.json({ message: router is working' });
});

app.use('/api', router);
```

save it and start the server by entering following to terminal

```
node server.js
```

open a web browser and enter following url or open postman and enter following url and send. It will show following message



This indicate the server is working

Create CRUD functions

Create a file named beer.js in controller and modify the following code in it

```
// Load required packages
var Beer = require('../models/beer');

// Create endpoint /api/beers for POSTS
exports.postBeers = function(req, res) {
  // Create a new instance of the Beer model
  var beer = new Beer();

  // Set the beer properties that came from the POST data
  beer.name = req.body.name;
  beer.type = req.body.type;
  beer.quantity = req.body.quantity;

  // Save the beer and check for errors
  beer.save(function(err) {
    if (err)
      res.send(err);

    res.json({ message: 'Beer added to the locker!', data: beer });
  });
};

// Create endpoint /api/beers for GET
exports.getBeers = function(req, res) {
  // Use the Beer model to find all beer
  Beer.find(function(err, beers) {
```

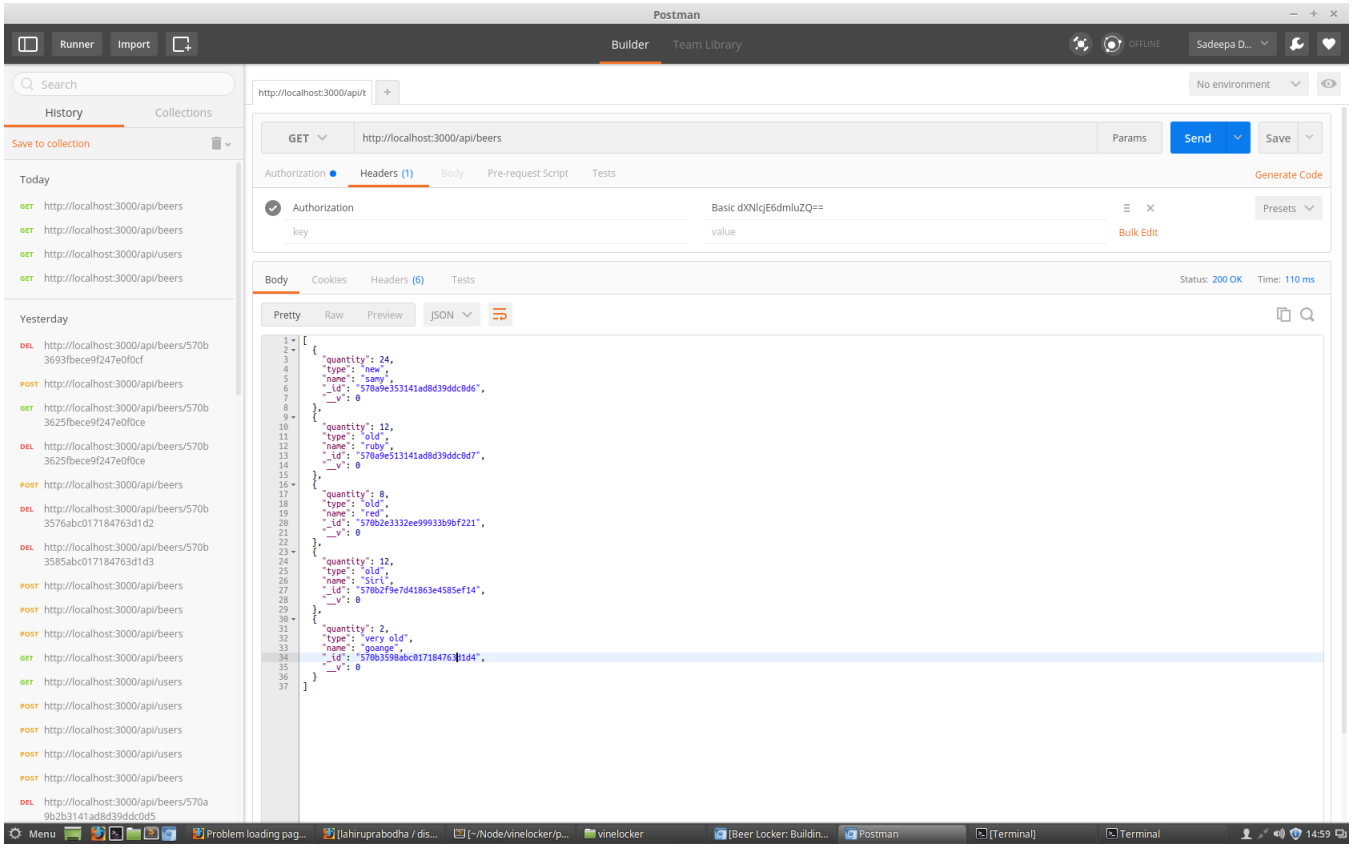
```
    if (err)
      res.send(err);

    res.json(beers);
  });
};

// Create endpoint /api/beers/:beer_id for GET
exports.getBeer = function(req, res) {
  // Use the Beer model to find a specific beer
  Beer.findById(req.params.beer_id, function(err, beer) {
    if (err)
      res.send(err);

    res.json(beer);
  });
};

// Create endpoint /api/beers/:beer_id for PUT
exports.putBeer = function(req, res) {
  // Use the Beer model to find a specific beer
  Beer.findById(req.params.beer_id, function(err, beer) {
    if (err)
      res.send(err);

    // Update the existing beer quantity
    beer.quantity = req.body.quantity;

    // Save the beer and check for errors
    beer.save(function(err) {
      if (err)
        res.send(err);

      res.json(beer);
    });
  });
};

// Create endpoint /api/beers/:beer_id for DELETE
exports.deleteBeer = function(req, res) {
  // Use the Beer model to find a specific beer and remove it
  Beer.findByIdAndRemove(req.params.beer_id, function(err) {
    if (err)
      res.send(err);

    res.json({ message: 'Beer removed from the locker!' });
  });
};
```

Also add another file named beer.js in models and enter following code

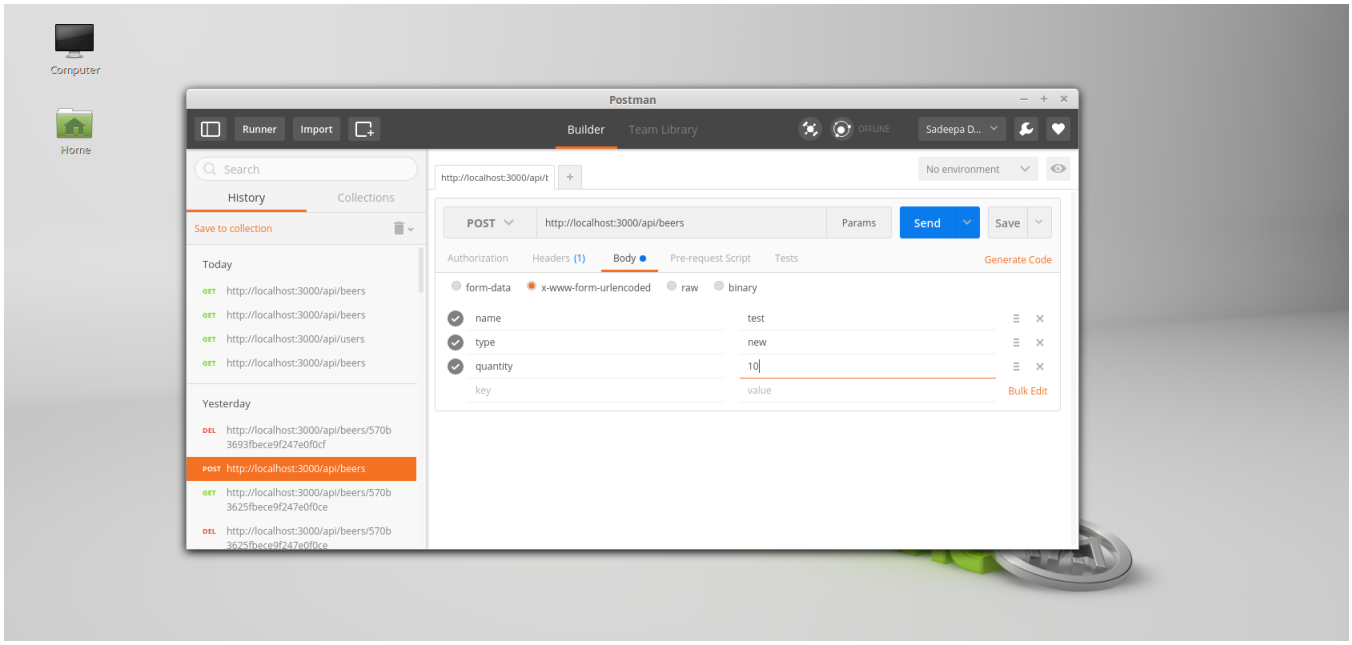
```
  // Load required packages
var mongoose = require('mongoose');

// Define our beer schema
var BeerSchema   = new mongoose.Schema({
  name: String,
  type: String,
  quantity: Number
});

// Export the Mongoose model
module.exports = mongoose.model('Beer', BeerSchema);
```

Now the REST API is ready to work with Postman

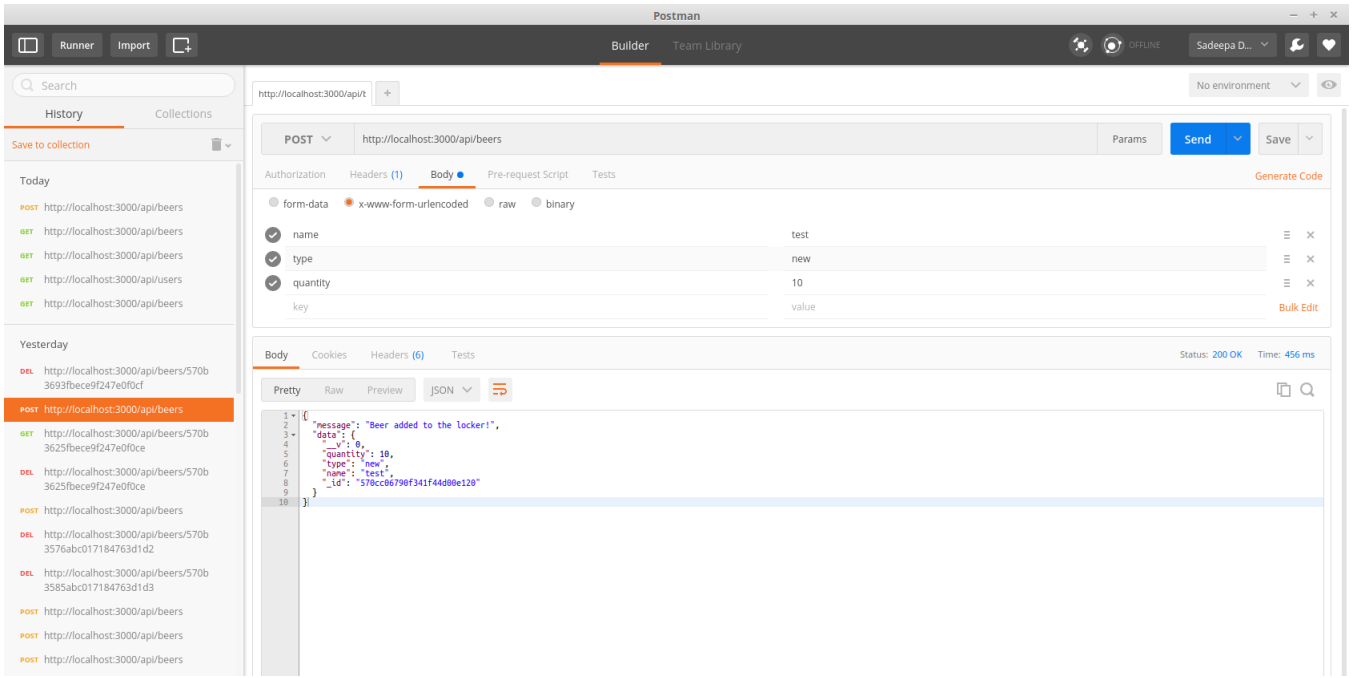
I already put some data to test GET method



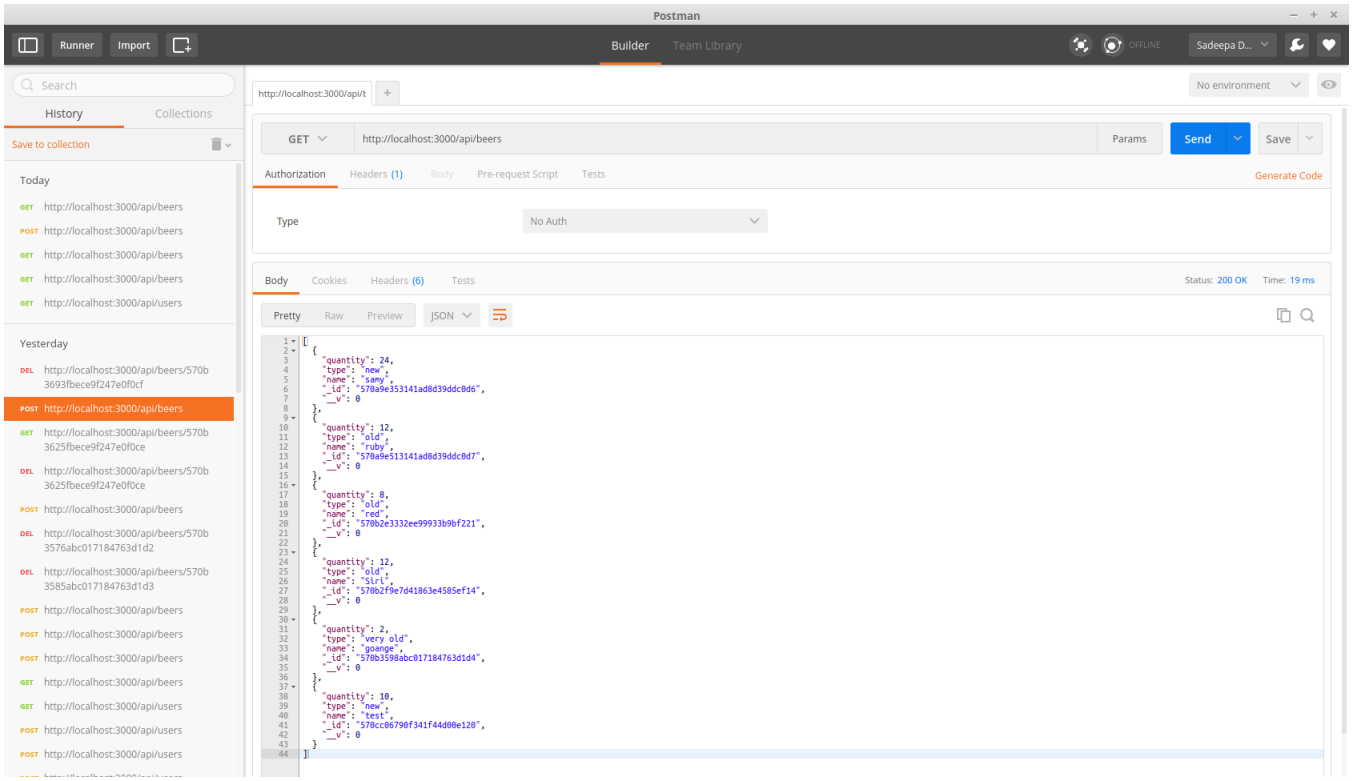
Insert values and keys to test POST method



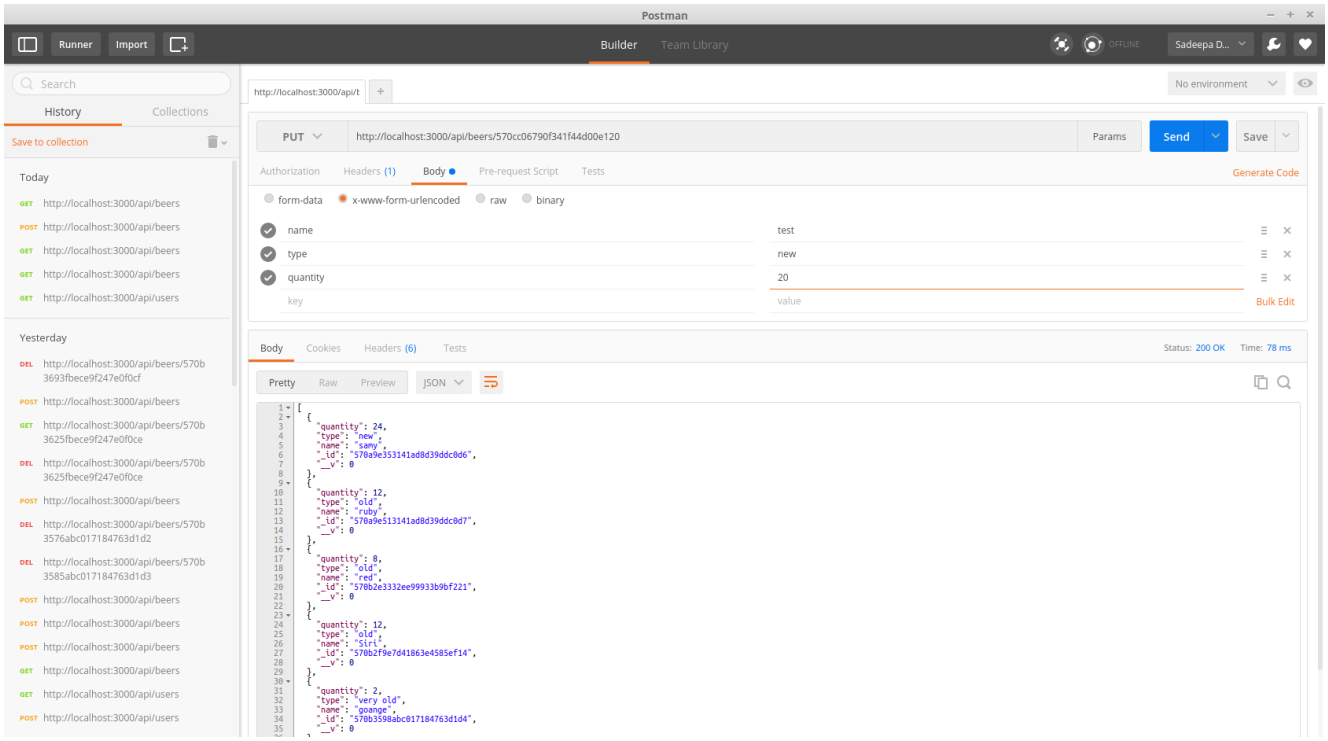
POST method success and output



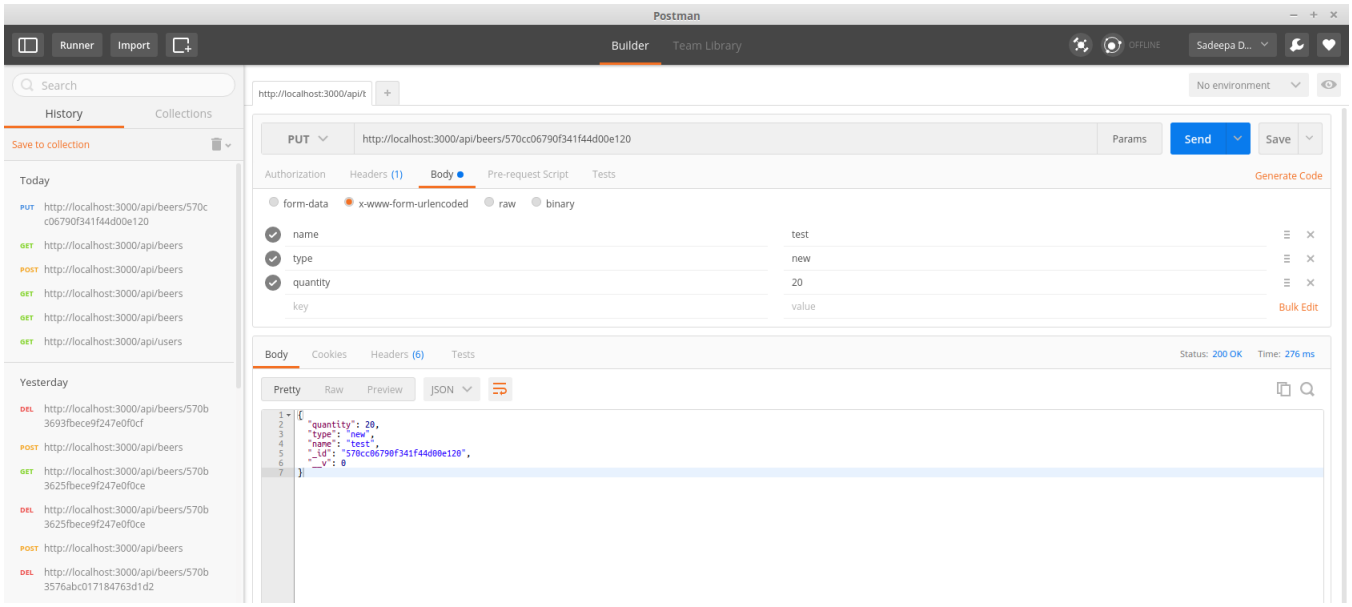
Newly added data in json format, at the bottom



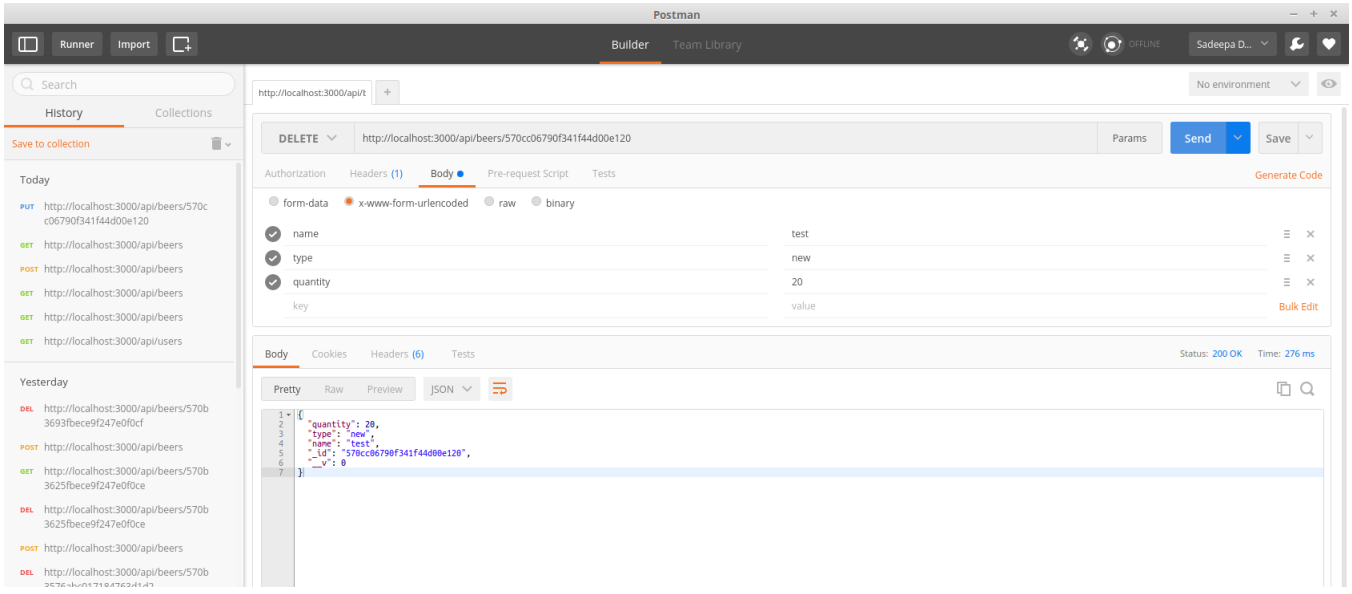
PUT method (Update current data)



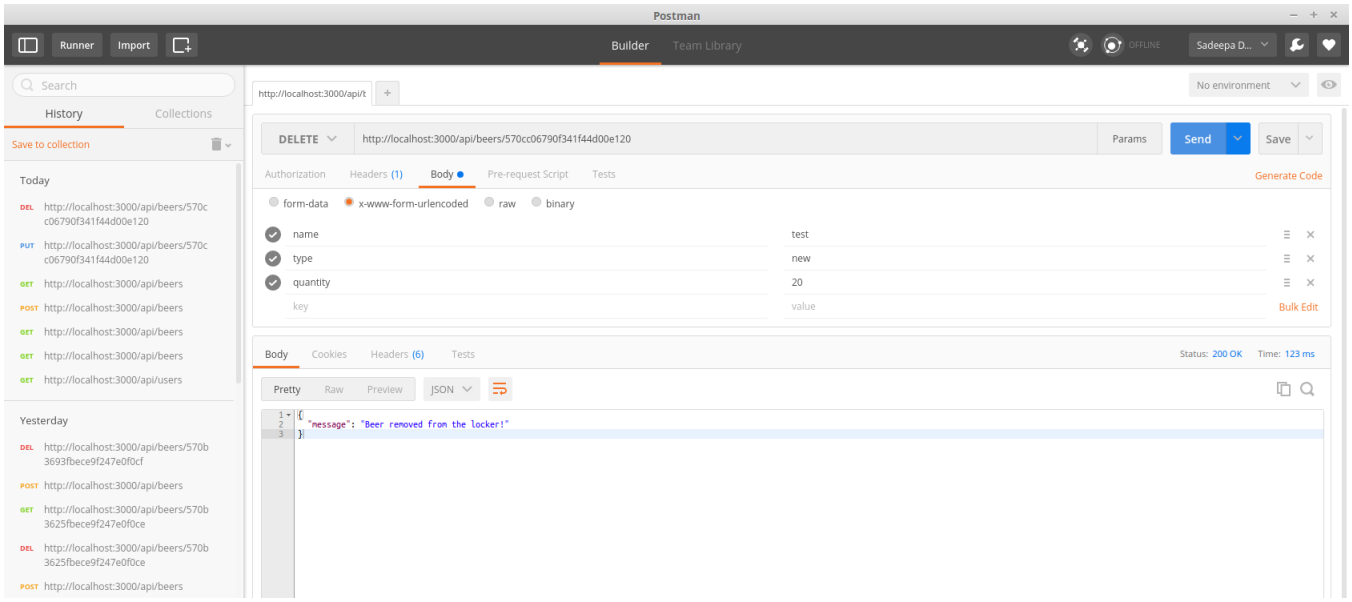
PUT method working and output



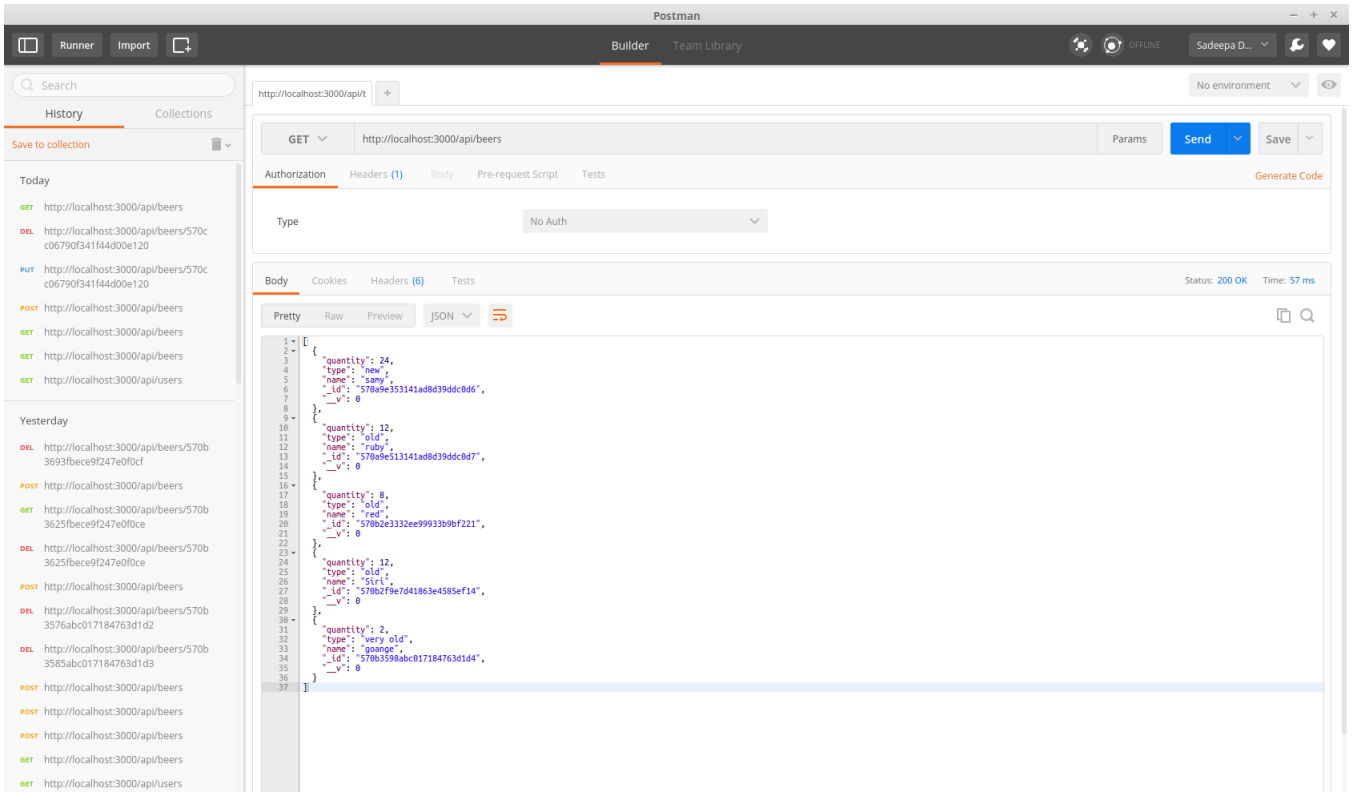
DELETE method testing



DELETE successful



GET method use to check the current DB(without deleted data)



to Create users, Create a file named users.js in controller and modify the following code in it

```
// Load required packages
var User = require('../models/user');

// Create endpoint /api/users for POST
exports.postUsers = function(req, res) {
  var user = new User({
    username: req.body.username,
    password: req.body.password
  });

  user.save(function(err) {
    if (err)
      res.send(err);

    res.json({ message: 'New beer user added to the locker room!' });
  });
};

// Create endpoint /api/users for GET
exports.getUsers = function(req, res) {
  User.find(function(err, users) {
    if (err)
      res.send(err);

    res.json(users);
  });
};
```

Same as above in model

```
// Load required packages
var mongoose = require('mongoose');
var bcrypt = require('bcrypt-nodejs');

// Define our user schema
```

```
var UserSchema = new mongoose.Schema({
  username: {
    type: String,
    unique: true,
    required: true
  },
  password: {
    type: String,
    required: true
  }
});

// Execute before each user.save() call
UserSchema.pre('save', function(callback) {
  var user = this;

  // Break out if the password hasn't changed
  if (!user.isModified('password')) return callback();

  // Password changed so we need to hash it
  bcrypt.genSalt(5, function(err, salt) {
    if (err) return callback(err);

    bcrypt.hash(user.password, salt, null, function(err, hash) {
      if (err) return callback(err);
      user.password = hash;
      callback();
    });
  });
});

UserSchema.methods.verifyPassword = function(password, cb) {
  bcrypt.compare(password, this.password, function(err, isMatch) {
    if (err) return cb(err);
    cb(null, isMatch);
  });
};
// Export the Mongoose model
module.exports = mongoose.model('User', UserSchema);
```

Check USER authentication with passport

Without username and password now clients can't do CRUD functions

IMPORTANT

Must create a new user or use the current username and password set by me to continue

User name = user1

Password = vine

Create a file named auth.js in controller and modify the following code in it

```
// Load required packages
var passport = require('passport');
var BasicStrategy = require('passport-http').BasicStrategy;
var User = require('../models/user');

passport.use(new BasicStrategy(
  function(username, password, callback) {
    User.findOne({ username: username }, function (err, user) {
      if (err) { return callback(err); }

      // No user found with that username
      if (!user) { return callback(null, false); }

      // Make sure the password is correct
      user.verifyPassword(password, function(err, isMatch) {
        if (err) { return callback(err); }

        // Password did not match
        if (!isMatch) { return callback(null, false); }

        // Success
        return callback(null, user);
      });
    });
  }
));

exports.isAuthenticated = passport.authenticate('basic', { session : false });
```

Install passport

```
npm install passport --save
npm install passport-http --save
```

without correct username and password following message will display