Mongodb Full stack application

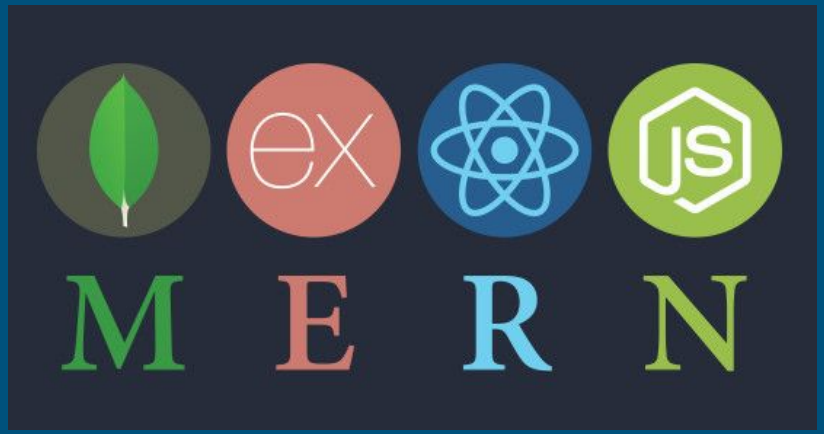
Robin Davies

Code

Full code is:

https://github.com/robin22d/ToDos_hogan

Popular Application Stacks

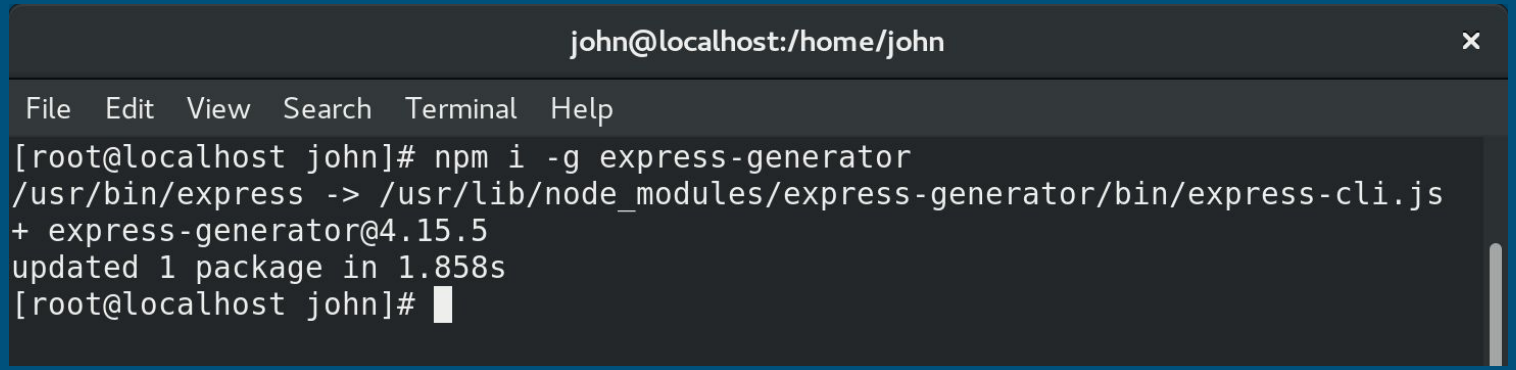


How does it work?

EXPRESS

Install express-generator

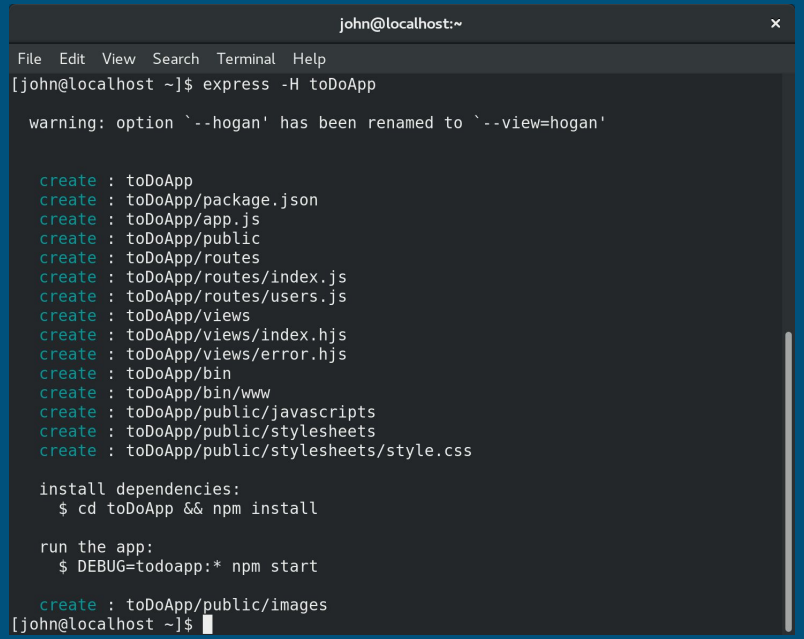
- Command:
  - npm i -g express-generator
    - ‘i’ means install
    - ‘-g’ means install globally



EXPRESS

Create template app

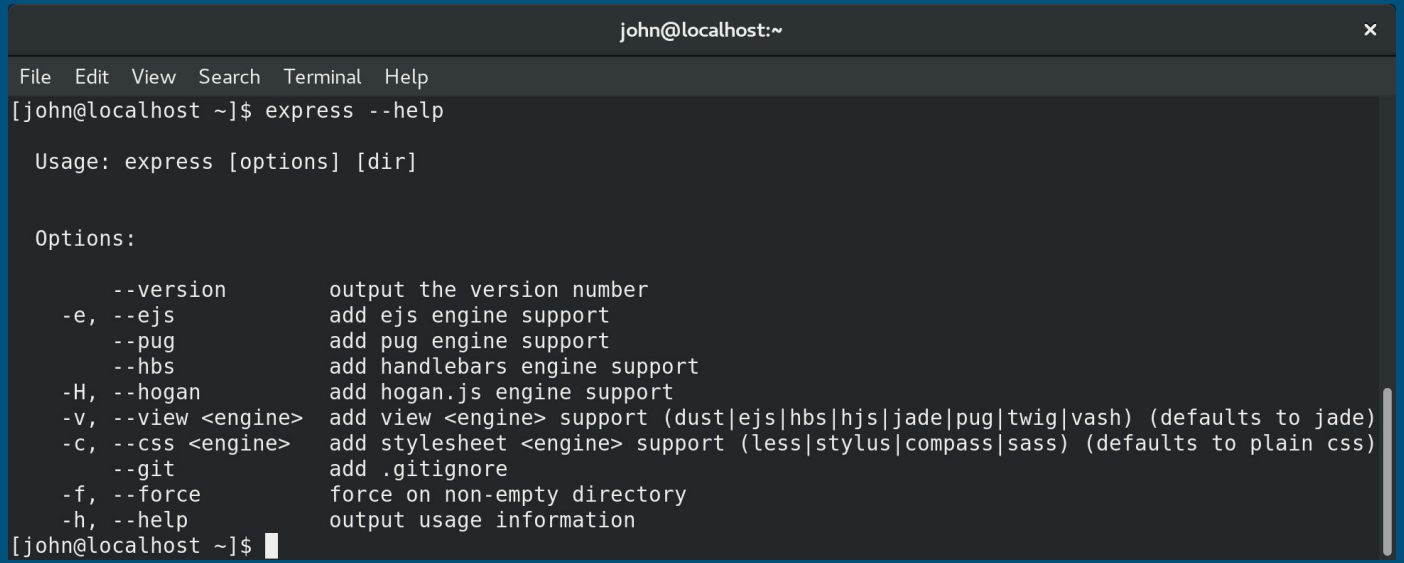
- Command:
  - express -H {Name_of_your_app}
    - ‘-H’ means use hogan as you view engine



EXPRESS

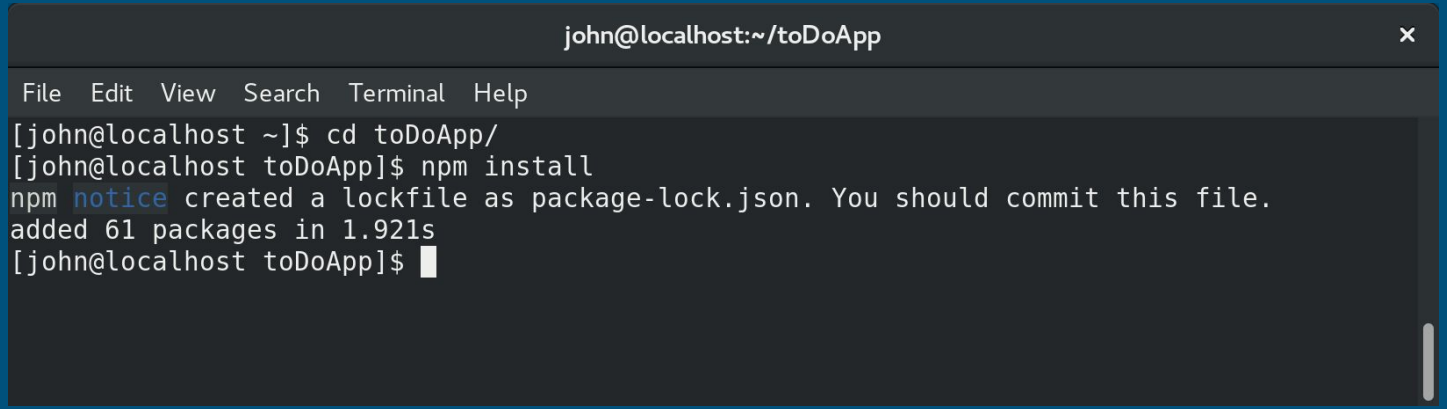
Express

- These are the view engines that can be used.
- We will initially suing hogan but then switching to react or angular.



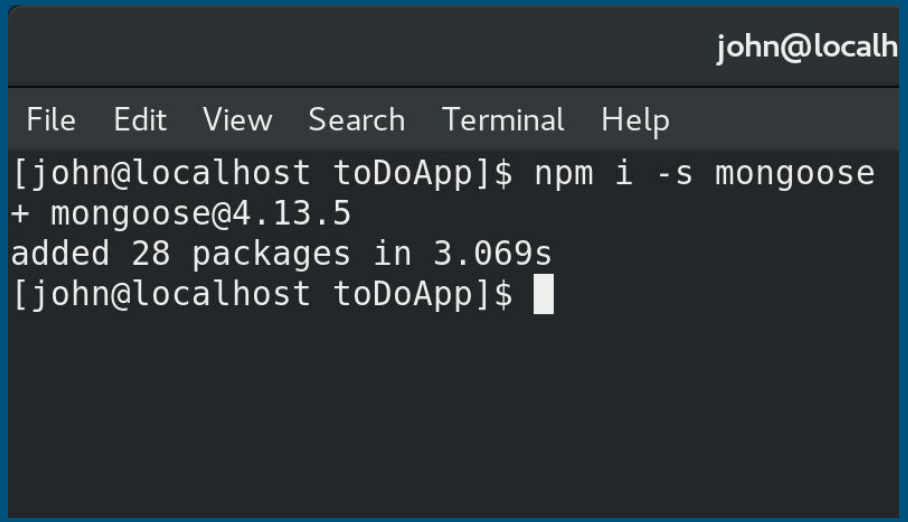
Install node modules

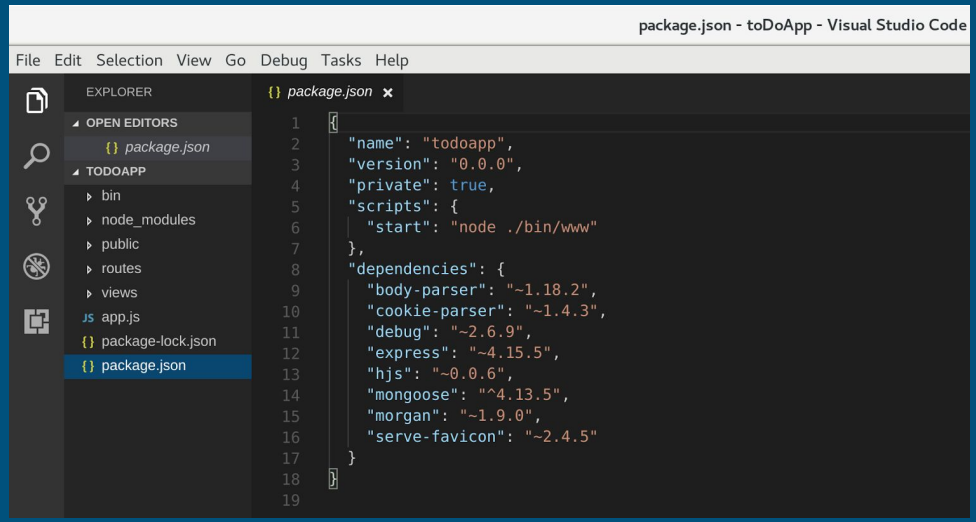
- Cd to app
  - cd {your_app}
- Install node modules
  - npm install
    - This will install the dependencies that are in your package.json file.



Add mongoose as a dependency

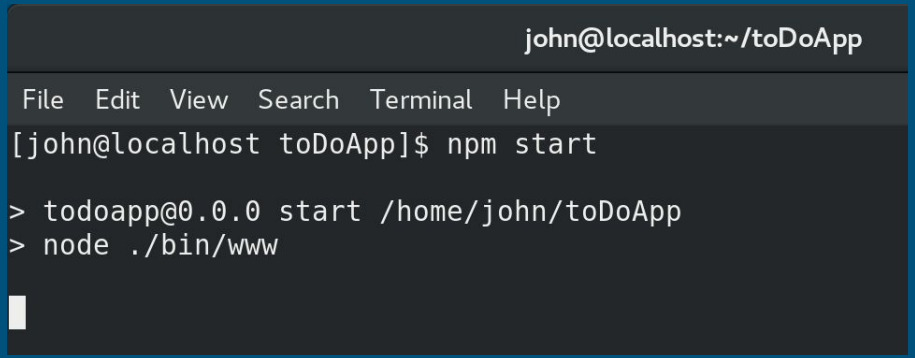
- Install mongoose
  - npm i -S mongoose
    - ‘-S’ saves mongoose to you dependencies in your package.json

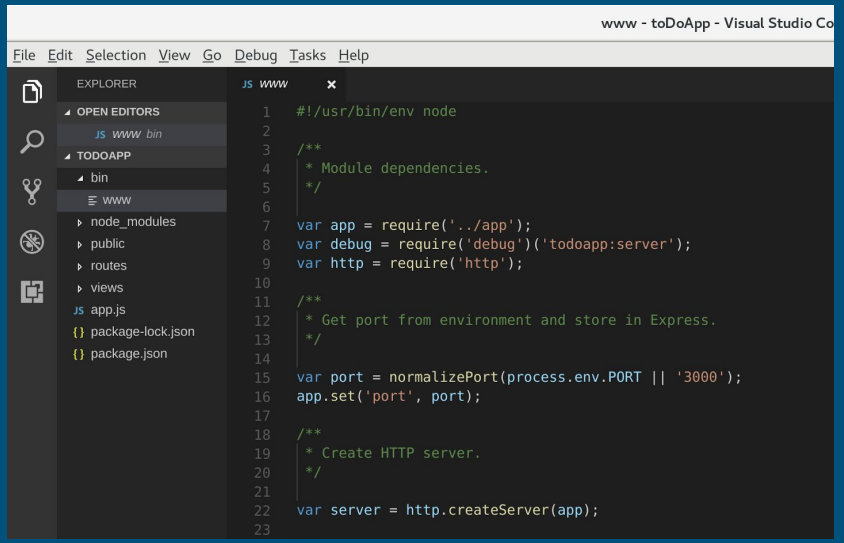




Check that it works

- Start the server
  - npm start
- As can be seen from the ‘/bin/www’ folder the app is running locally on port: 3000 so go to:
  - http://localhost:3000/



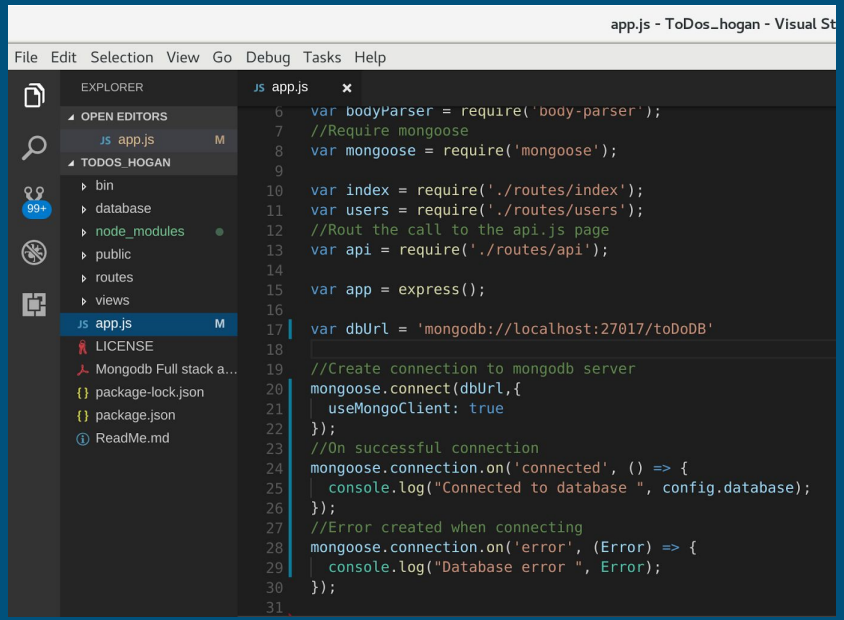


Check that it works

- Expresses template page should not be showing in the browser

Add mongoose to the server

- The mongoose driver will need to be added to the node server.



Code

Line 10 of app.js

```
var mongoose = require('mongoose');
```

Lines 17-31 of app.js

```
//the name of the database after the /
var dbUrl = 'mongodb://localhost:27017/toDoDB'

//Create connection to mongodb server
mongoose.connect(dbUrl,{
 useMongoClient: true
});
//On successful connection
mongoose.connection.on('connected', () => {
 console.log("Connected to database ", config.database);
});
//Error created when connecting
mongoose.connection.on('error', (Error) => {
 console.log("Database error ", Error);
});
```

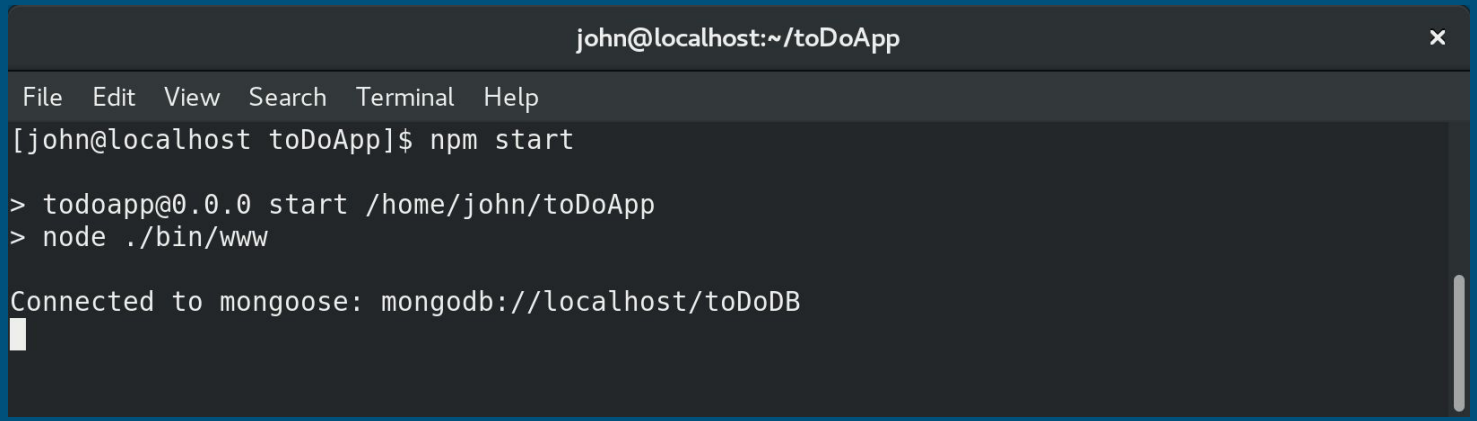
Start mongodb server

- To start mongo run the command
  - mongod

```
john@localhost:~
File  Edit  View  Search  Terminal  Help
[john@localhost ~]$ mongod
2017-12-02T17:11:51.614+0000 I CONTROL  [initandlisten] MongoDB starting : pid=6
922 port=27017 dbpath=/data/db 64-bit host=localhost.localdomain
2017-12-02T17:11:51.614+0000 I CONTROL  [initandlisten] db version v3.4.6
2017-12-02T17:11:51.614+0000 I CONTROL  [initandlisten] git version: c55eb86ef46
ee7aede3b1e2a5d184a7df4bfb5b5
2017-12-02T17:11:51.614+0000 I CONTROL  [initandlisten] OpenSSL version: OpenSSL
 1.1.0g-fips  2 Nov 2017
```

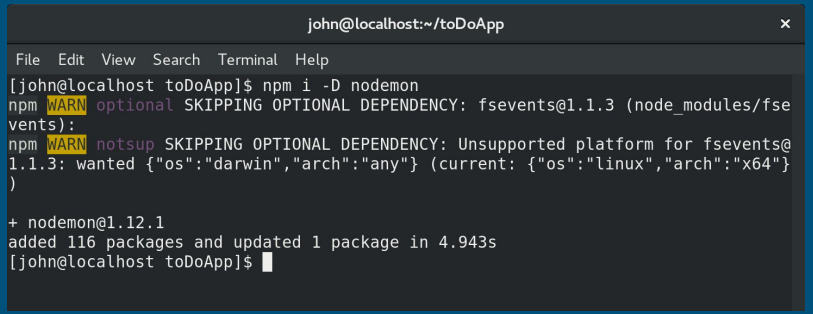
Rerun the server

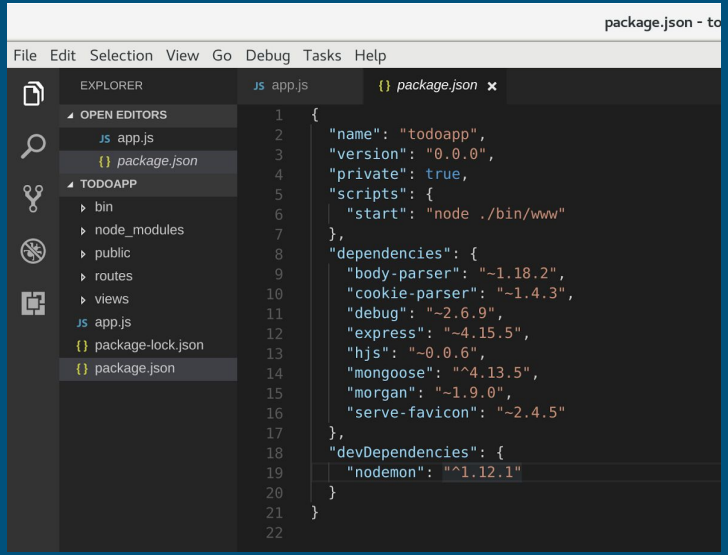
- Stop server
  - Ctrl + c
- Start server again
  - npm start



Adding nodemon

- This will mean that you will not have to keep restarting the server.
  - npm -i -g -D nodemon
    - ‘-D’ means save in dev dependencies
    - ‘-g’ means so it can be accessed globally





Restart server

- Run server with nodemon
  - nodemon

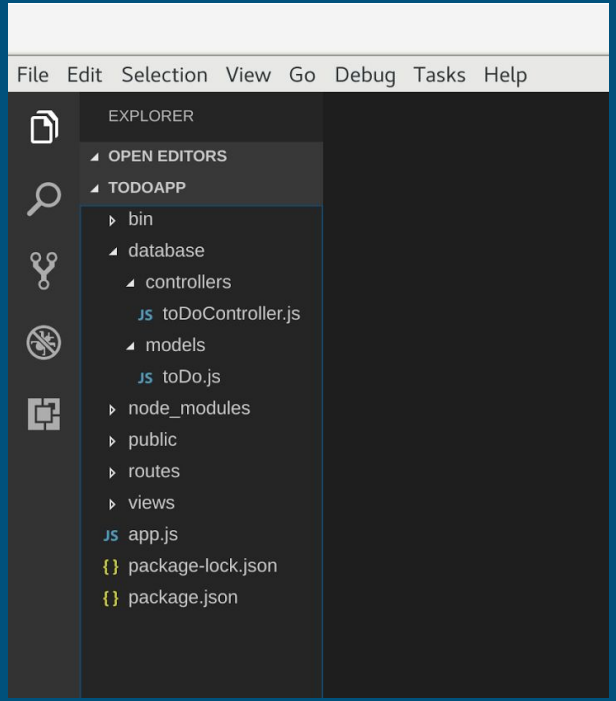
```
john@localhost:~/toDoApp                                    ×
File  Edit  View  Search  Terminal  Help
[john@localhost toDoApp]$ nodemon
[nodemon] 1.12.1
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/www`
Connected to mongoose: mongodb://localhost/toDoDB
```

Check server still runs

- Go to:
  - http://localhost:3000/

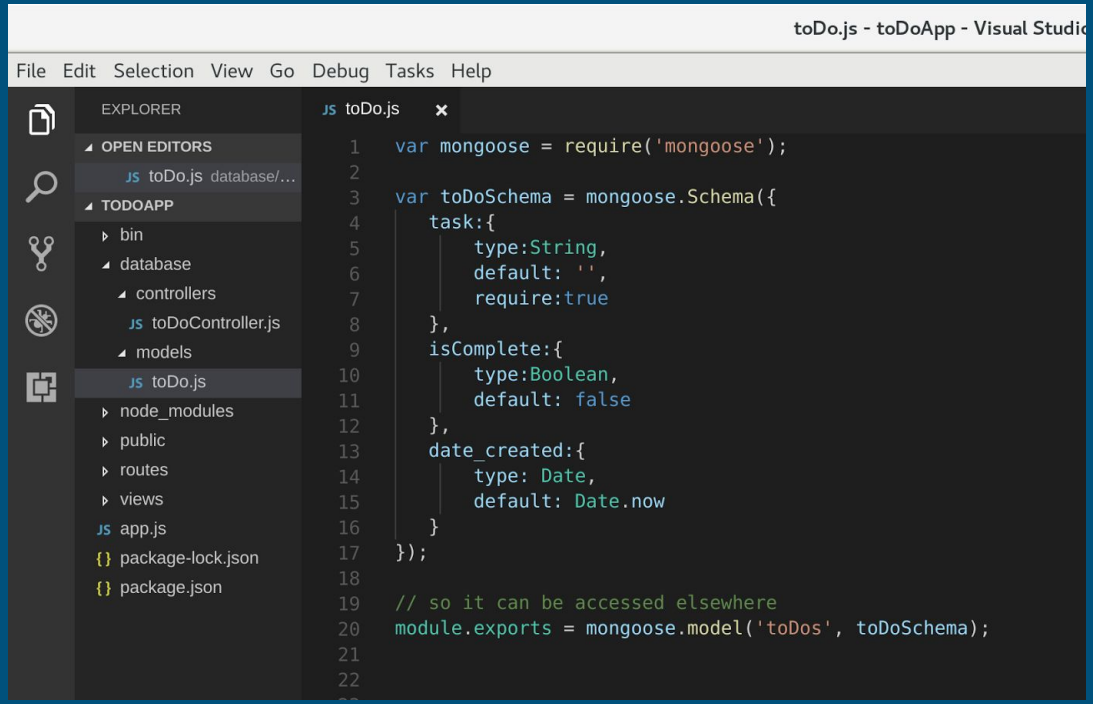
Create Database

- Create a folder for your database
  - Create a models folder
    - Create a models.js file
  - Create a controllers folder
    - Create a controller.js for the module



Create your schema

- Fill in your schema



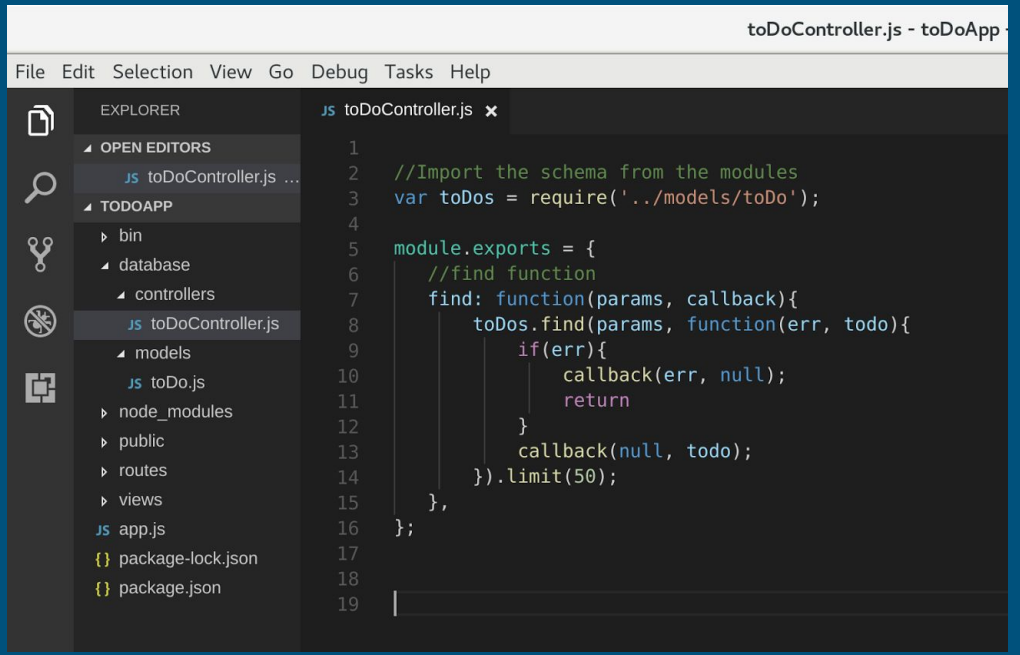
Code

All of ‘/database/modules/toDo.js’

```
var mongoose = require('mongoose');

var toDoSchema = mongoose.Schema({
  task:{
      type:String,
      default: '',
      require:true
  },
  isComplete:{
      type:Boolean,
      default: false
  },
  date_created:{
      type: Date,
      default: Date.now
  }
});

// so it can be accessed elsewhere
module.exports = mongoose.model('toDos', toDoSchema);
```

Create the controller

- Create a controller for the module
  - Require the module



Code

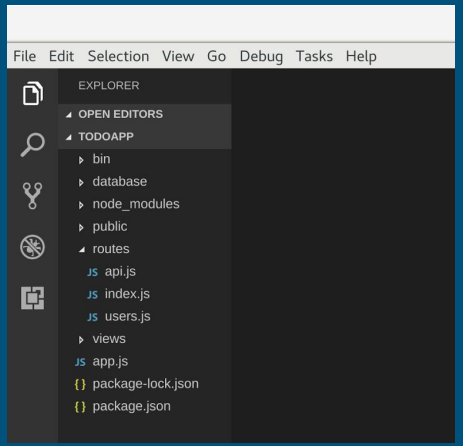
Lines 0 to 14 of
‘/database/controller/toDoController.js’

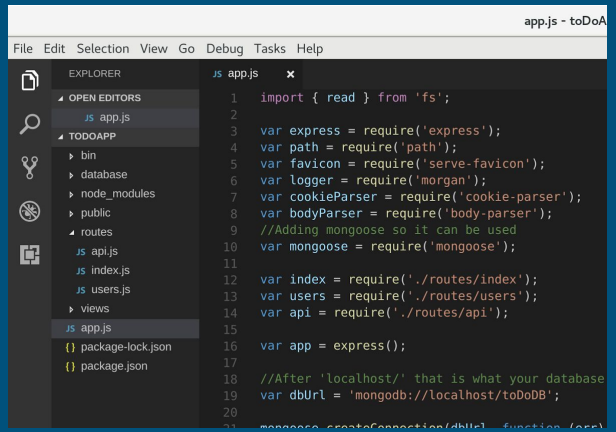
```
//Import the schema from the modules
var toDos = require('../models/todos.js');

module.exports = {
  //find function
  find: function(params, callback){
      toDos.find(params, function(err, todo){
          if(err){
              callback(err, null);
              return
          }
          callback(null, todo);
      }).limit(50);
  }
};
```

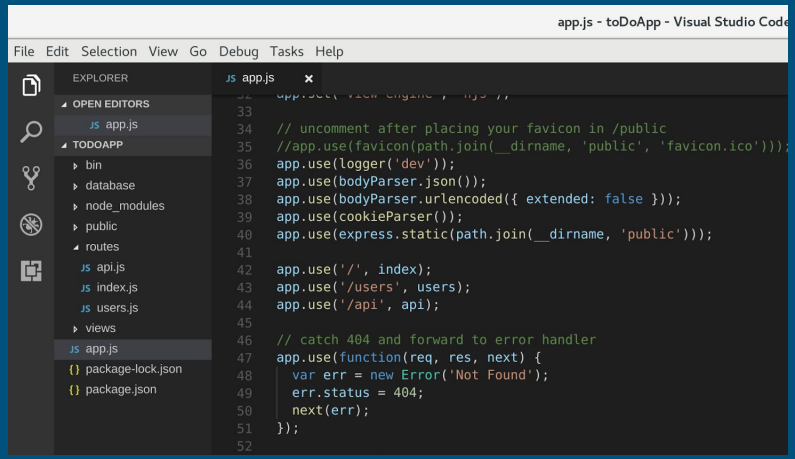
Creating the API

{ api }

- Create a ‘api.js’ file in the ‘/routes’ directory
- Make the file accessible from the server
  - Route the requests to the api







Code

Line 13 of ‘app.js’

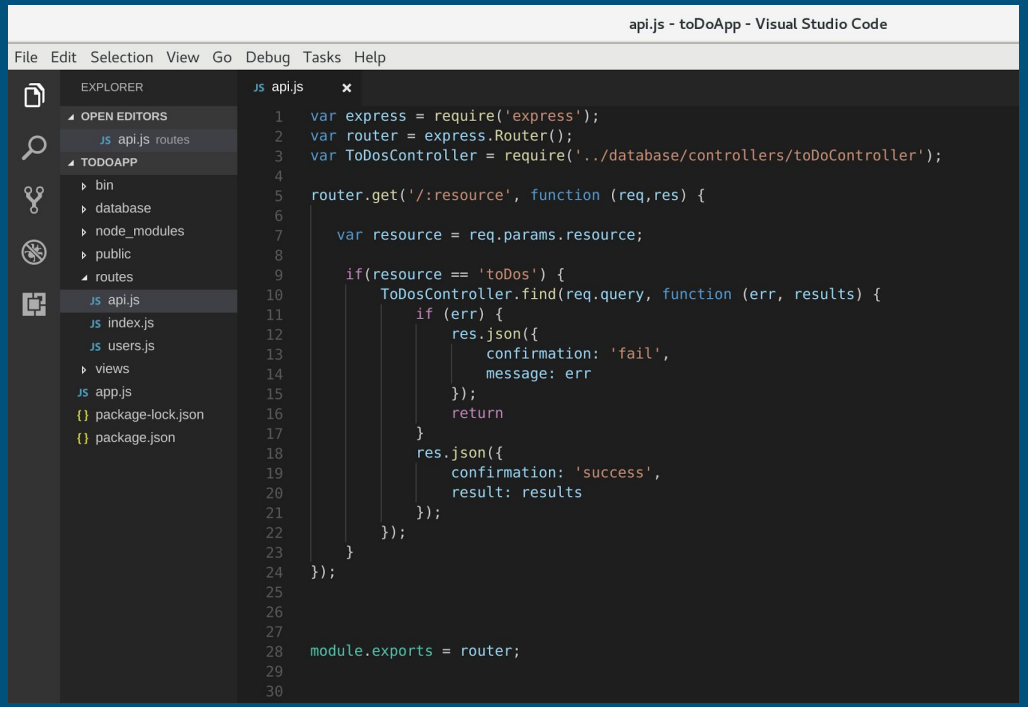
```
var api = require('./routes/api');
```

Line 47 of ‘app.js’

```
app.use('/api’ api);
```

Api get request

- Handle that api request
  - Require the controller



Code

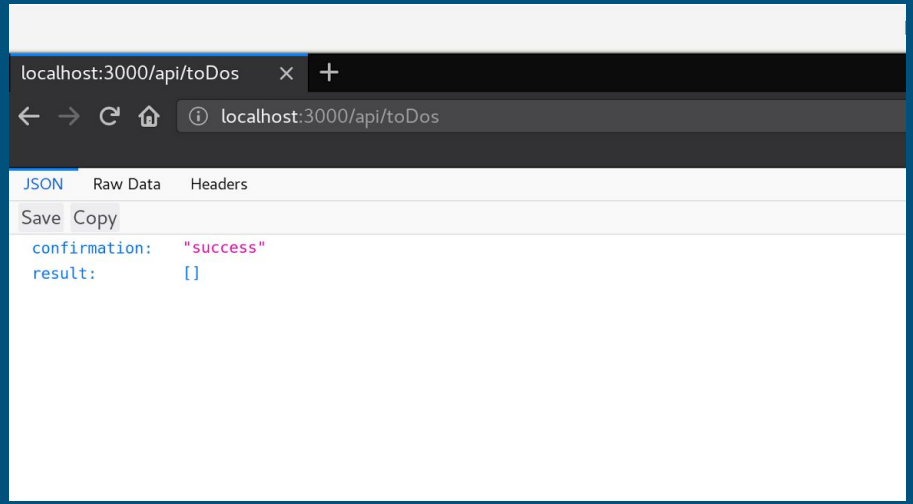
Lines 0 to 24 and 49 of
‘/routes/api.js’

{ api }

```
var express = require('express');
var router = express.Router();
var ToDosController = require('../database/controllers/toDoController.js');

router.get('/:resource', function (req,res) {

  var resource = req.params.resource;

  if(resource == 'toDos') {
      ToDosController.find(req.query, function (err, results) {
          if (err) {
              res.json({
                  confirmation: 'fail',
                  message: err
              });
              return
          }
          res.json({
              confirmation: 'success',
              result: results
          });
      });
  }
});

module.exports = router;
```

Check it works

- Go to
  - http://localhost:3000/api/toDos



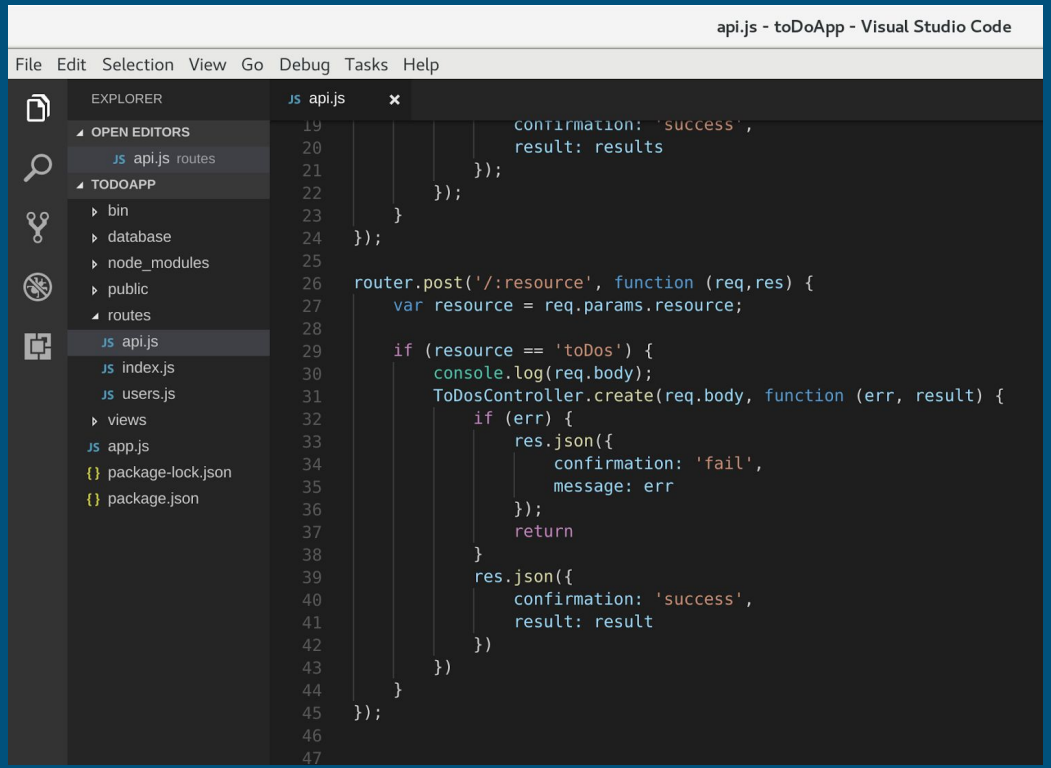
Response

- The api is now going to the mongo database and checking what is there and returning that as a json object.

Adding information

{ api }

- Handle the new request
  - Is a post rather than a get



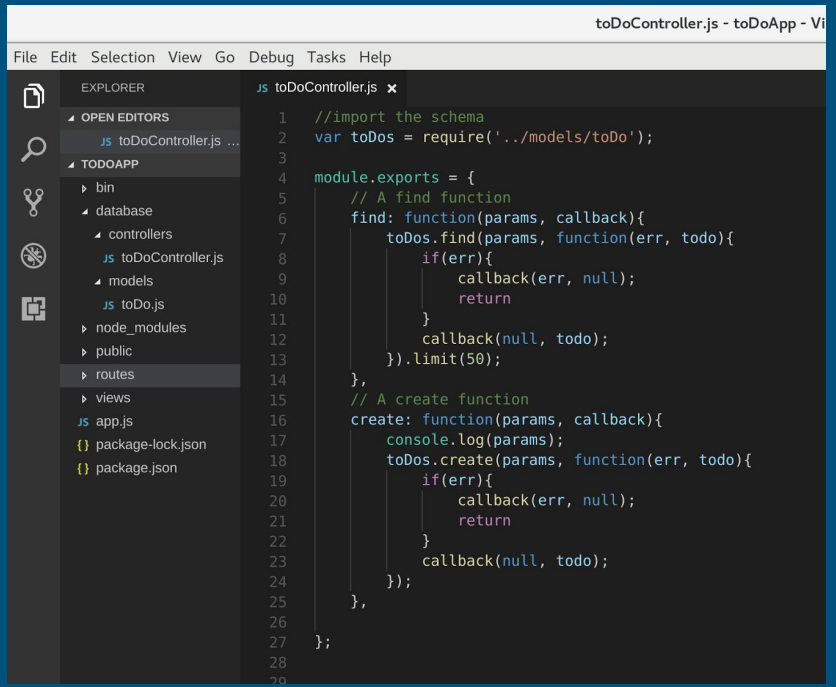
Code

Lines 26 to 45 of ‘/routes/api.js’

```
router.post('/:resource', function (req,res) {
  var resource = req.params.resource;

  if (resource == 'toDos') {
      console.log(req.body);
      ToDosController.create(req.body, function (err, result) {
          if (err) {
              res.json({
                  confirmation: 'fail',
                  message: err
              });
              return
          }
          res.json({
              confirmation: 'success',
              result: result
          })
      })
  }
});
```

Add controller

- Add a create method to the controller



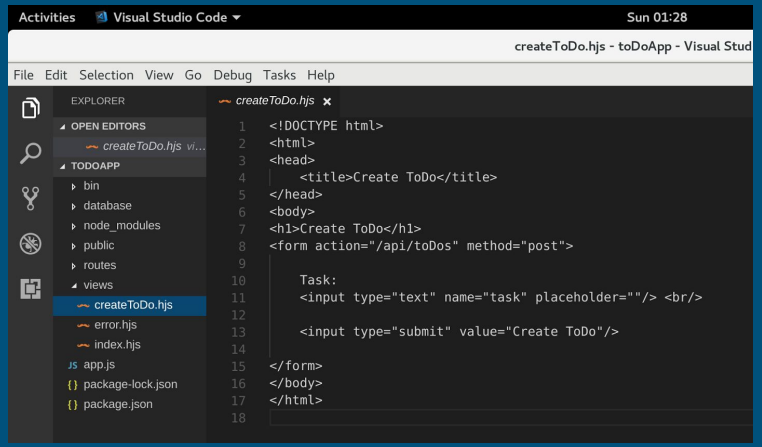
Code

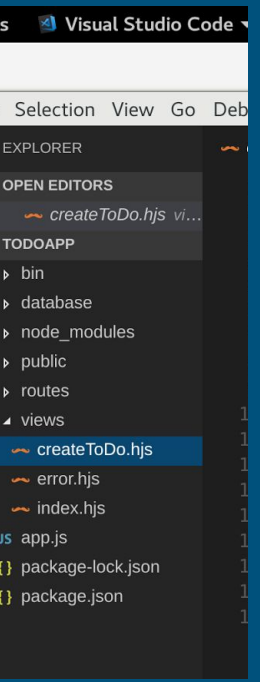
Lines 15 to 25 of
‘/controller/toDoController.js’

```
// A create function
  create: function(params, callback){
      console.log(params);
      toDos.create(params, function(err, todo){
          if(err){
              callback(err, null);
              return
          }
          callback(null, todo);
      });
  }
```

Send information

- Create a hjs file to send the new todo
  - This will be replaced with a react or angular client
- Add a form to the new file
  - This will send a post request to the api





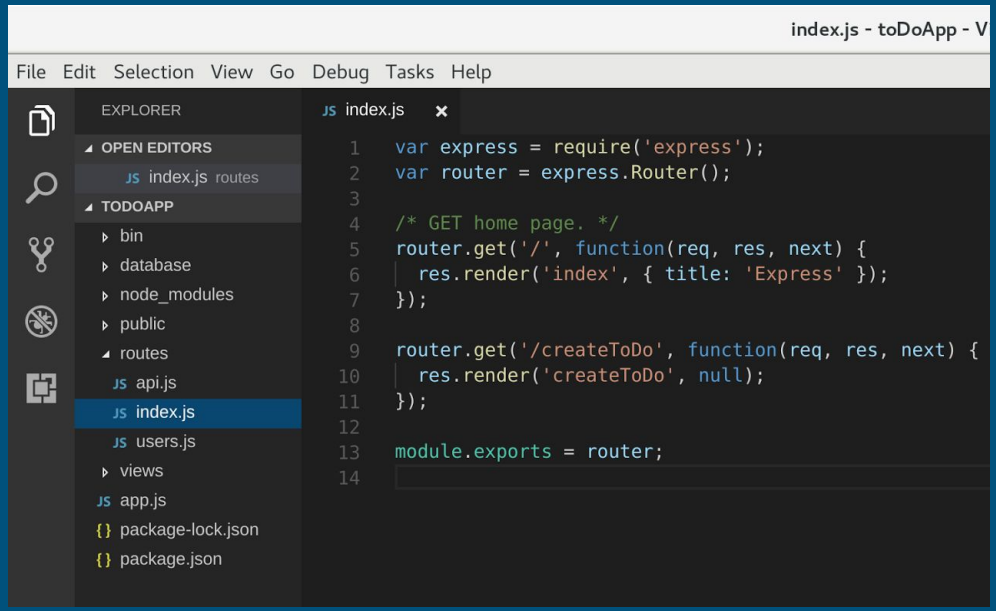
Code

All of ‘/view/createToDo.hjs’

```
<!DOCTYPE html>

<html>

<head>

        <title>Create ToDo</title>

</head>

<body>

<h1>Create ToDo</h1>

<form action="/api/toDos" method="post">

        Task:

        <input type="text" name="task" placeholder=""/> <br/>

        <input type="submit" value="Create ToDo"/>

</form>

</body>

</html>
```

Add new files to the route

- Handle the request when it is enter
  - Render the createToDo.hjs page



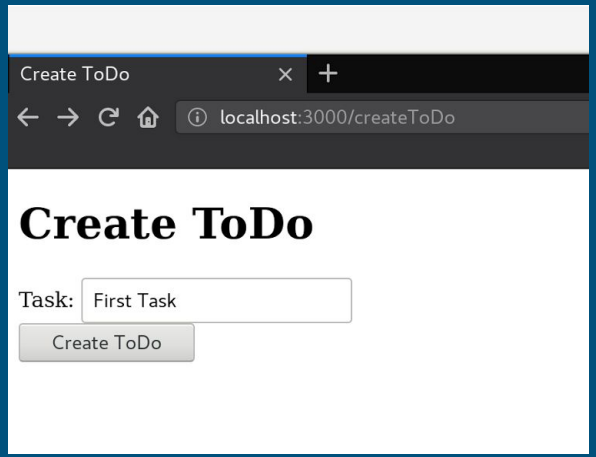
Code

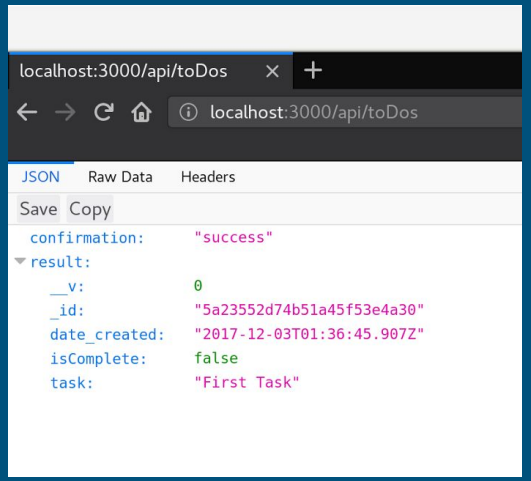
Lines x to x of ‘/routes/index.js’

```
router.get('/createToDo', function(req, res, next) {
 res.render('createToDo', null);
});
```

Test this works

- Go to:
  - http://localhost:3000/createToDo
- Add a value into the input box and submit the form
  - You will be redirected the the api with the information that was enter in it.





Check that is has been added to mongo

- Go to mongo by typing
  - mongo
- Check that your database has been created
  - Show dbs

```
16
17  //After localhost is the name of your database
18  var dbUrl = 'mongodb://localhost/toDoDB';
19
```

```
john@localhost:~
File  Edit  View  Search  Terminal  Help
> show dbs
admin        0.000GB
local        0.000GB
toDoDB       0.000GB
tvDatabase   0.000GB
>
```

Show collection

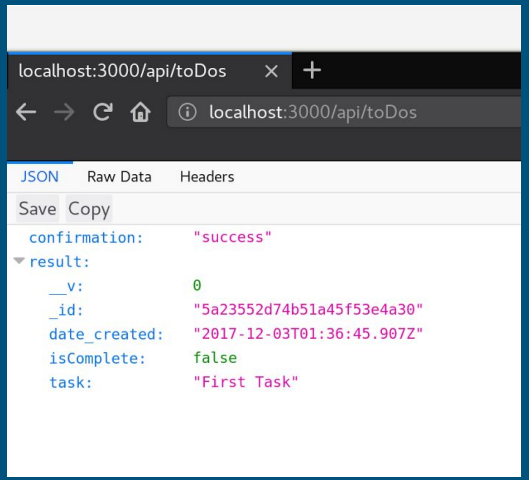
- Switch to your database
  - use {your_db}
- Show the collections in the database
  - Show collection

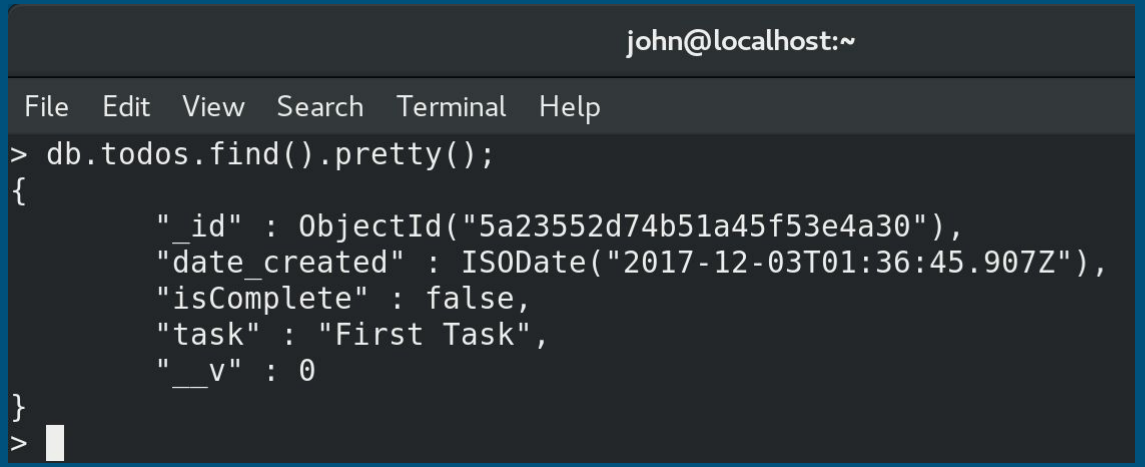
```
module.exports = {
    // A find function
    find: function(params, callback){
        toDos.find(params, function(err, todo){
            if(err){
                callback(err, null);
```

```
john@loca
File  Edit  View  Search  Terminal  Help
> show dbs
admin       0.000GB
local       0.000GB
toDoDB      0.000GB
tvDatabase  0.000GB
> use toDoDB
switched to db toDoDB
> show collections
todos
>
```

Collections content

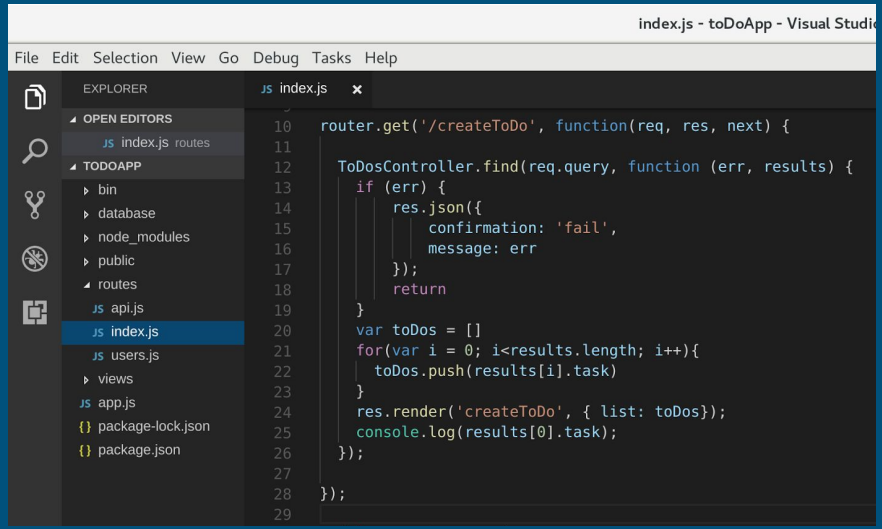
- To check what is in the collection type
  - db.{collection_name}.find().pretty();
    - ‘.find()’ gets everything in the collection
    - ‘.pretty()’ make it easier to read





Render List

- Add a get to the index.js file
  - The get will call the api and return the results



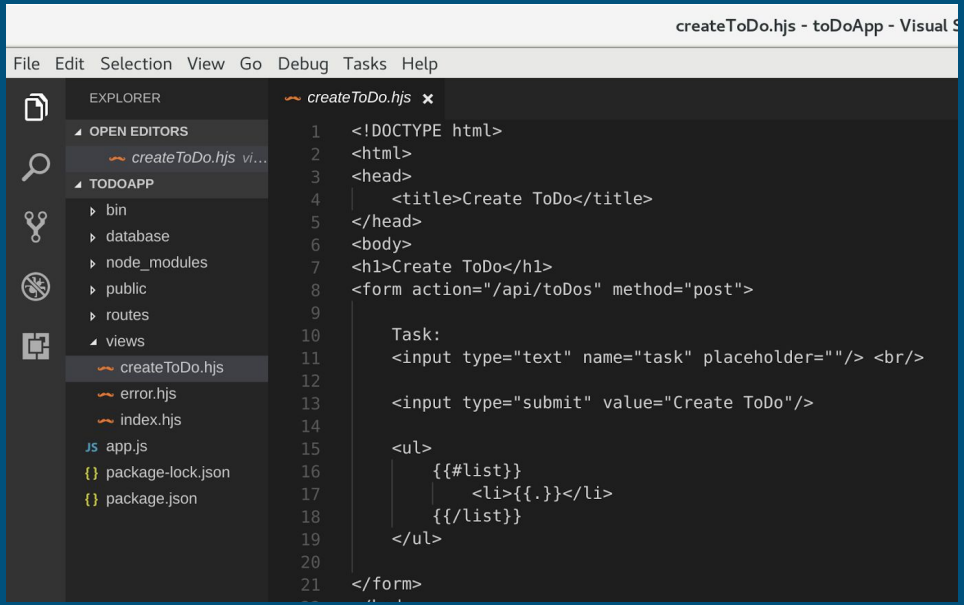
Code

Lines 3 and 10 to 28 of ‘/routes/index.js’

```
var ToDosController =
require('../database/controllers/toDoController
');


router.get('/createToDo', function(req, res, next) {

 ToDosController.find(req.query, function (err, results) {
   if (err) {
       res.json({
           confirmation: 'fail',
           message: err
       });
       return
   }
   var toDos = []
   for(var i = 0; i<results.length; i++){
     toDos.push(results[i].task)
   }
   res.render('createToDo', { list: toDos});
   console.log(results[0].task);
 });

});
```

Render the results

- Render the resolus in the html



Code

Lines 15 to 19 of ‘/routes/index.js’

```
<ul>
    {{#list}}
    <li>{{.}}</li>
    {{/list}}
</ul>
```

Check that it works

- Make sure the list is being rendered
- Add content to the list and go back to
  - http://localhost:3000/createToDo