

Mongodb Full stack application

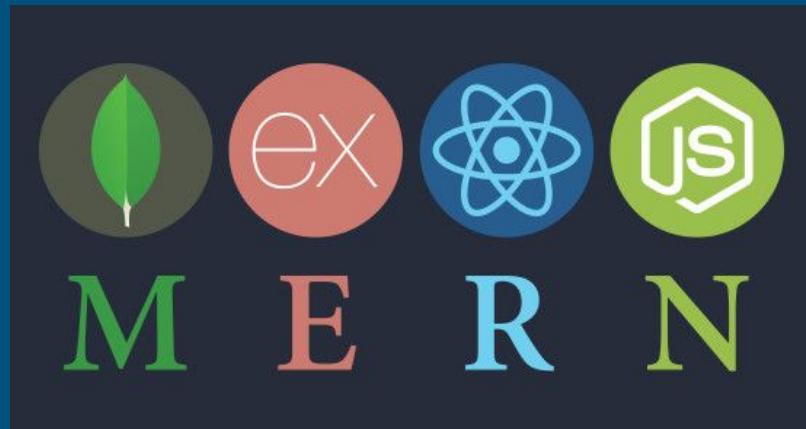
Robin Davies

Code

Full code is:

https://github.com/robin22d/ToDos_hogan

Popular Application Stacks

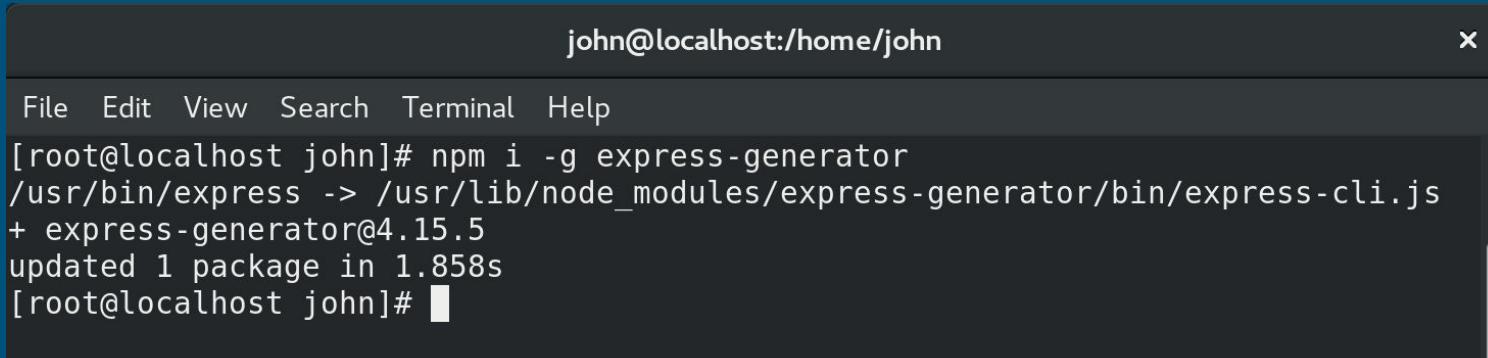


How does it work?



Install express-generator

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



The screenshot shows a terminal window with a dark background and light-colored text. The title bar reads "john@localhost:/home/john". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The command entered was `npm i -g express-generator`. The output shows the package being installed from `/usr/bin/express` to `/usr/lib/node_modules/express-generator/bin/express-cli.js`, with version `+ express-generator@4.15.5`. It also indicates that 1 package was updated in 1.858 seconds. The prompt at the end is `[root@localhost john]#`.

```
john@localhost:/home/john
File Edit View Search Terminal Help
[root@localhost john]# npm i -g express-generator
/usr/bin/express -> /usr/lib/node_modules/express-generator/bin/express-cli.js
+ express-generator@4.15.5
updated 1 package in 1.858s
[root@localhost john]#
```

Create template app

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

```
john@localhost:~
```

```
File Edit View Search Terminal Help
[john@localhost ~]$ express -H ToDoApp
warning: option `--hogan' has been renamed to `--view=hogan'

create : ToDoApp
create : ToDoApp/package.json
create : ToDoApp/app.js
create : ToDoApp/public
create : ToDoApp/routes
create : ToDoApp/routes/index.js
create : ToDoApp/routes/users.js
create : ToDoApp/views
create : ToDoApp/views/index.hjs
create : ToDoApp/views/error.hjs
create : ToDoApp/bin
create : ToDoApp/bin/www
create : ToDoApp/public/javascripts
create : ToDoApp/public/stylesheets
create : ToDoApp/public/stylesheets/style.css

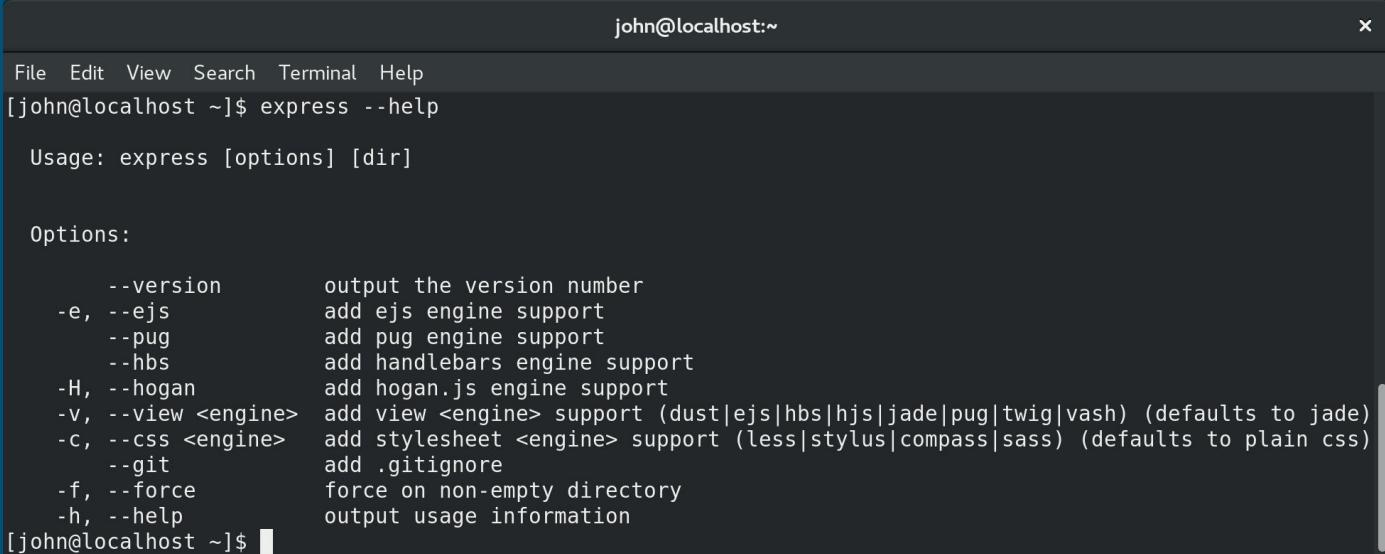
install dependencies:
$ cd ToDoApp && npm install

run the app:
$ DEBUG=todoapp:* npm start

create : ToDoApp/public/images
[john@localhost ~]$
```

Express

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



The screenshot shows a terminal window titled "john@localhost:~". The command [john@localhost ~]\$ express --help is run, displaying the usage and options for the express command. The usage is "Usage: express [options] [dir]". The options section lists various flags and their descriptions, including support for different view engines like jade, pug, handlebars, hogan.js, and less/stylus/compass/sass, as well as gitignore handling and force creation of non-empty directories.

```
john@localhost:~  
File Edit View Search Terminal Help  
[john@localhost ~]$ express --help  
  
Usage: express [options] [dir]  
  
Options:  
  
      --version      output the version number  
      -e, --ejs       add ejs engine support  
      --pug          add pug engine support  
      --hbs          add handlebars engine support  
      -H, --hogan     add hogan.js engine support  
      -v, --view <engine> add view <engine> support (dust|ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)  
      -c, --css <engine> add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)  
      --git          add .gitignore  
      -f, --force     force on non-empty directory  
      -h, --help      output usage information  
[john@localhost ~]$
```



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.

```
john@localhost:~/ToDoApp
File Edit View Search Terminal Help
[john@localhost ~]$ cd ToDoApp/
[john@localhost ToDoApp]$ npm install
npm notice created a lockfile as package-lock.json. You should commit this file.
added 61 packages in 1.921s
[john@localhost ToDoApp]$ █
```



Add mongoose as a dependency

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

```
john@localhost
File Edit View Search Terminal Help
[john@localhost ToDoApp]$ npm i -s mongoose
+ mongoose@4.13.5
added 28 packages in 3.069s
[john@localhost ToDoApp]$
```

```
package.json - ToDoApp - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER package.json
OPEN EDITORS package.json
TODOAPP
bin
node_modules
public
routes
views
app.js
package-lock.json
package.json
1
{
  "name": "todoapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.18.2",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.15.5",
    "hjs": "~0.0.6",
    "mongoose": "^4.13.5",
    "morgan": "~1.9.0",
    "serve-favicon": "~2.4.5"
  }
}
19
```



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/>

```
john@localhost:~/ToDoApp
File Edit View Search Terminal Help
[john@localhost ToDoApp]$ npm start

> todoapp@0.0.0 start /home/john/todoApp
> node ./bin/www
```

www - ToDoApp - Visual Studio Code

```
File Edit Selection View Go Debug Tasks Help
EXPLORER JS www x
OPEN EDITORS JS www bin
TODOAPP bin www
node_modules public routes views
JS app.js package-lock.json package.json
1 #!/usr/bin/env node
2
3 /**
4  * Module dependencies.
5 */
6
7 var app = require('../app');
8 var debug = require('debug')('todoapp:server');
9 var http = require('http');
10
11 /**
12  * Get port from environment and store in Express.
13 */
14
15 var port = normalizePort(process.env.PORT || '3000');
16 app.set('port', port);
17
18 /**
19  * Create HTTP server.
20 */
21
22 var server = http.createServer(app);
```



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.

A screenshot of the Visual Studio Code interface. The title bar says "app.js - ToDos_hogan - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The Explorer sidebar shows a file tree with "OPEN EDITORS" containing "app.js" and "TODOS_HOGAN" which includes "bin", "database", "node_modules", "public", "routes", and "views". Other files like "LICENSE", "MongoDb Full stack a...", "package-lock.json", "package.json", and "ReadMe.md" are also listed. The main editor area shows the code for "app.js":

```
app.js - ToDos_hogan - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

EXPLORER
OPEN EDITORS
  JS app.js M
  TODOS_HOGAN
    > bin
    > database
    > node_modules
    > public
    > routes
    > views
JS app.js M
LICENSE
MongoDb Full stack a...
{} package-lock.json
{} package.json
① ReadMe.md

var bodyParser = require('body-parser');
//Require mongoose
var mongoose = require('mongoose');

var index = require('./routes/index');
var users = require('./routes/users');
//Rout the call to the api.js page
var api = require('./routes/api');

var app = express();

var dbUrl = 'mongodb://localhost:27017/toDoDB'

//Create connection to mongodb server
mongoose.connect(dbUrl, {
  useNewUrlParser: 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);
});
```

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,{ 
  useNewUrlParser: 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);
});
```





MongoDB

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
```



MongoDB

Rerun the server

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

```
john@localhost:~/ToDoApp
File Edit View Search Terminal Help
[john@localhost ToDoApp]$ npm start
> todoapp@0.0.0 start /home/john/ToDoApp
> node ./bin/www
Connected to mongoose: mongodb://localhost/ToDoDB
```



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

```
john@localhost:~/ToDoApp
File Edit View Search Terminal Help
[john@localhost ToDoApp]$ npm i -D nodemon
npm [WARN] optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.1.3 (node_modules/fsevents):
npm [WARN] notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.1.3: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
+ nodemon@1.12.1
added 116 packages and updated 1 package in 4.943s
[john@localhost ToDoApp]$
```

```
package.json - todoapp
File Edit Selection View Go Debug Tasks Help
EXPLORER JS app.js {} package.json x
OPEN EDITORS JS app.js {} package.json
TODOAPP
  ▾ bin
  ▾ node_modules
  ▾ public
  ▾ routes
  ▾ views
  JS app.js
  {} package-lock.json
  {} package.json
1 {
  "name": "todoapp",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.18.2",
    "cookie-parser": "~1.4.3",
    "debug": "~2.6.9",
    "express": "~4.15.5",
    "hjs": "~0.0.6",
    "mongoose": "~4.13.5",
    "morgan": "~1.9.0",
    "serve-favicon": "~2.4.5"
  },
  "devDependencies": {
    "nodemon": "^1.12.1"
  }
}
```



Restart server

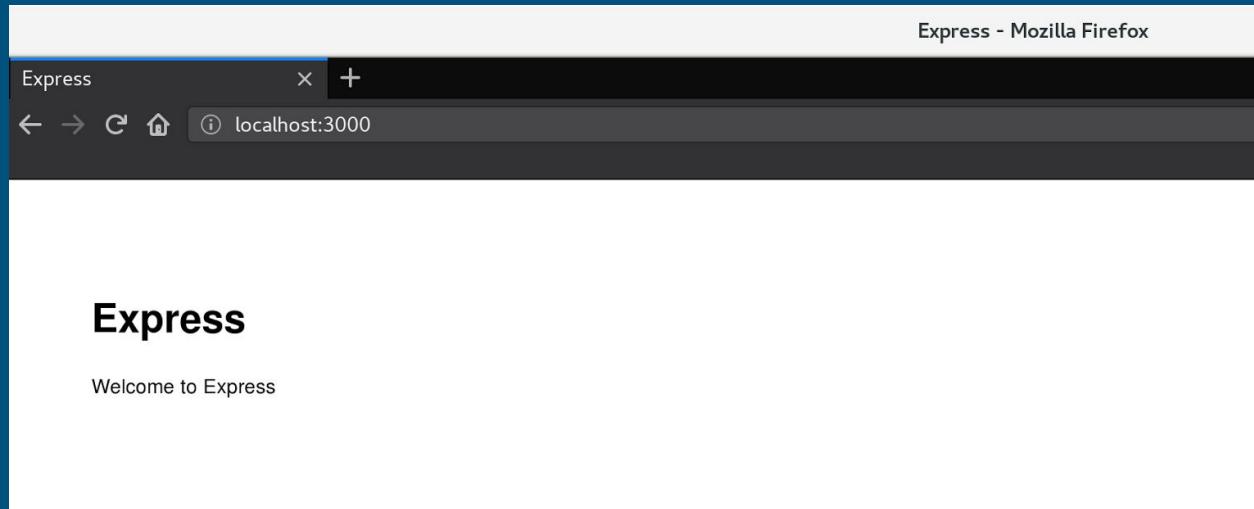
- Run server with nodemon
 - nodemon

```
john@localhost:~/ToDoApp x
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
```



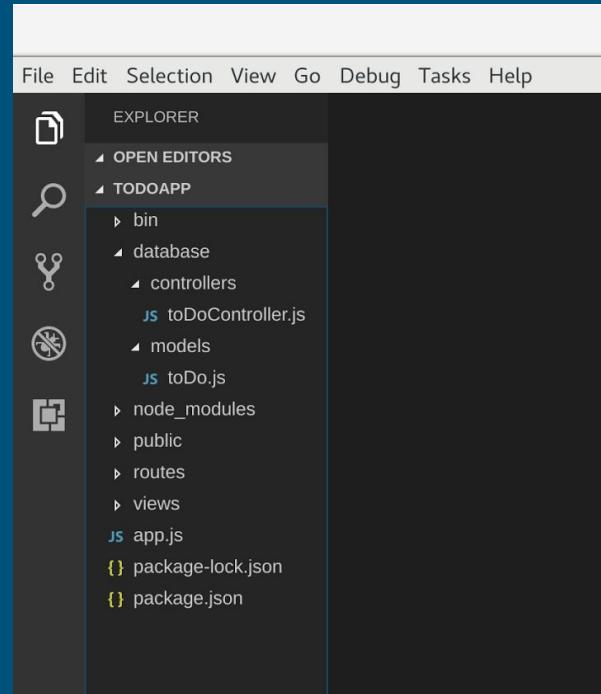
Check server still runs

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



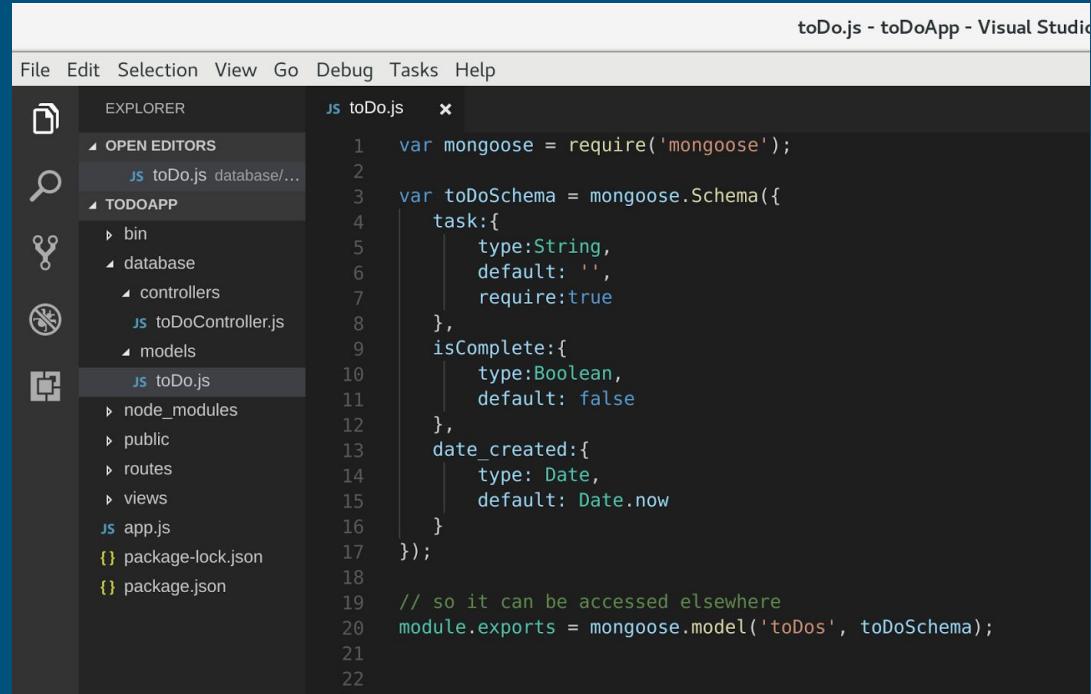
Create Database

- 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



```
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
JS ToDo.js database/...
TODOAPP
bin
database
controllers
JS ToDoController.js
models
JS ToDo.js
node_modules
public
routes
views
JS app.js
{} package-lock.json
{} package.json
JS 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);
```

Code

All of '/database/modules/toDo.js'

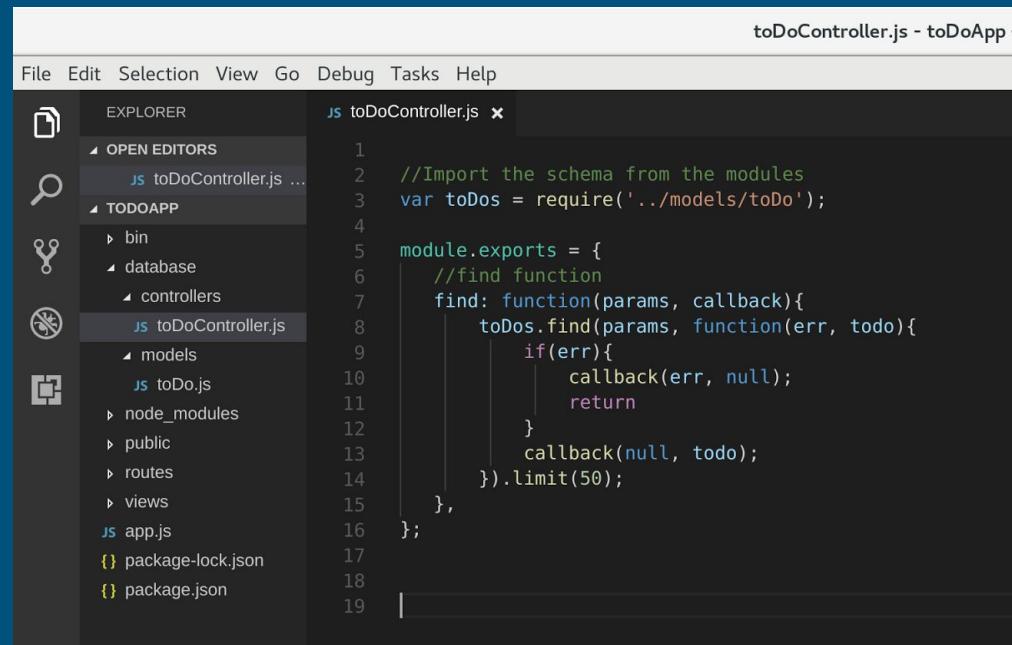
```
var mongoose = require('mongoose');

var ToDoSchema = mongoose.Schema({
  task: {
    type: String,
    default: '',
    required: 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



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows the project structure under "TODOAPP". The "ToDoController.js" file is currently selected.
- Code Editor:** On the right, the code for "ToDoController.js" is displayed. The code imports the "todo" schema and defines a module export with a "find" function that queries the database for todos.
- Status Bar:** At the bottom, it says "ToDoController.js - todoApp".

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

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);
  },
};
```

Code

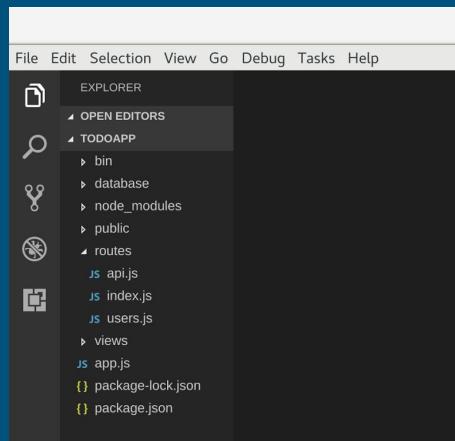
Lines 0 to 14 of
'/database/controller/todoController.js'

```
//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

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

A screenshot of the Visual Studio Code interface showing the code editor. The title bar says 'app.js - todoApp - Visual Studio Code'. The code editor displays the contents of the 'app.js' file. The code is as follows:

```
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TODOAPP
bin
database
node_modules
public
routes
views
app.js
api.js
index.js
users.js
app.js
package-lock.json
package.json

app.js - todoApp - Visual Studio Code

import { read } from 'fs';
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
//Adding mongoose so it can be used
var mongoose = require('mongoose');
var index = require('./routes/index');
var users = require('./routes/users');
var api = require('./routes/api');

var app = express();
//After 'localhost/' that is what your database
var dbUrl = 'mongodb://localhost/todoDB';
mongoose.connect(dbUrl, function(err)
```

A screenshot of the Visual Studio Code interface showing the code editor. The title bar says 'app.js - todoApp - Visual Studio Code'. The code editor displays the contents of the 'app.js' file. The code is as follows:

```
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
TODOAPP
bin
database
node_modules
public
routes
views
app.js
api.js
index.js
users.js
app.js
package-lock.json
package.json

app.js - todoApp - Visual Studio Code

import { read } from 'fs';
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
//Adding mongoose so it can be used
var mongoose = require('mongoose');
var index = require('./routes/index');
var users = require('./routes/users');
var api = require('./routes/api');

var app = express();
//After 'localhost/' that is what your database
var dbUrl = 'mongodb://localhost/todoDB';
mongoose.connect(dbUrl, function(err)
```

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

```
File Edit Selection View Go Debug Tasks Help
EXPLORER api.js
OPEN EDITORS routes
TODOAPP
bin
database
node_modules
public
routes
api.js
index.js
users.js
views
app.js
package-lock.json
package.json
var express = require('express');
var router = express.Router();
var ToDosController = require('../database/controllers/toDoController');

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;
```



Code

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

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

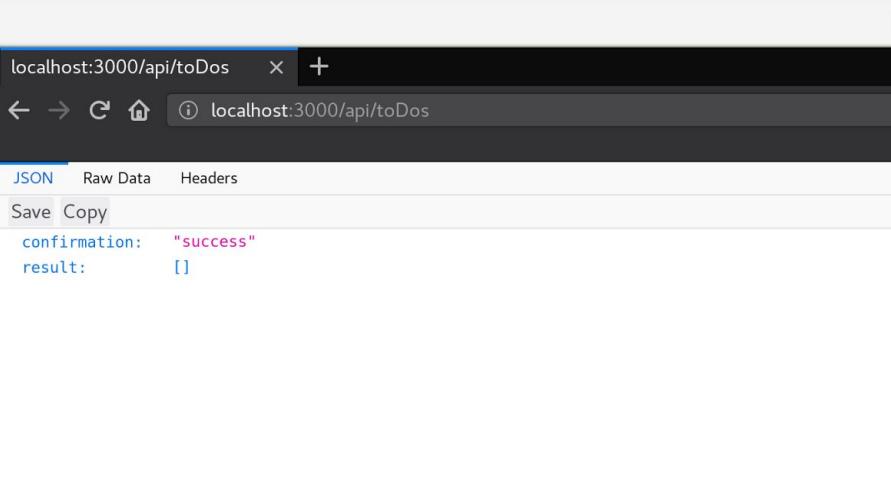
router.get('/:resource', function (req,res) {
    var resource = req.params.resource;

    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>



A screenshot of a browser developer tools Network tab. The tab title is "localhost:3000/api/toDos". The URL in the address bar is "localhost:3000/api/toDos". Below the address bar are tabs for "JSON", "Raw Data", and "Headers", with "JSON" being the active tab. There are "Save" and "Copy" buttons. The JSON response is displayed as follows:

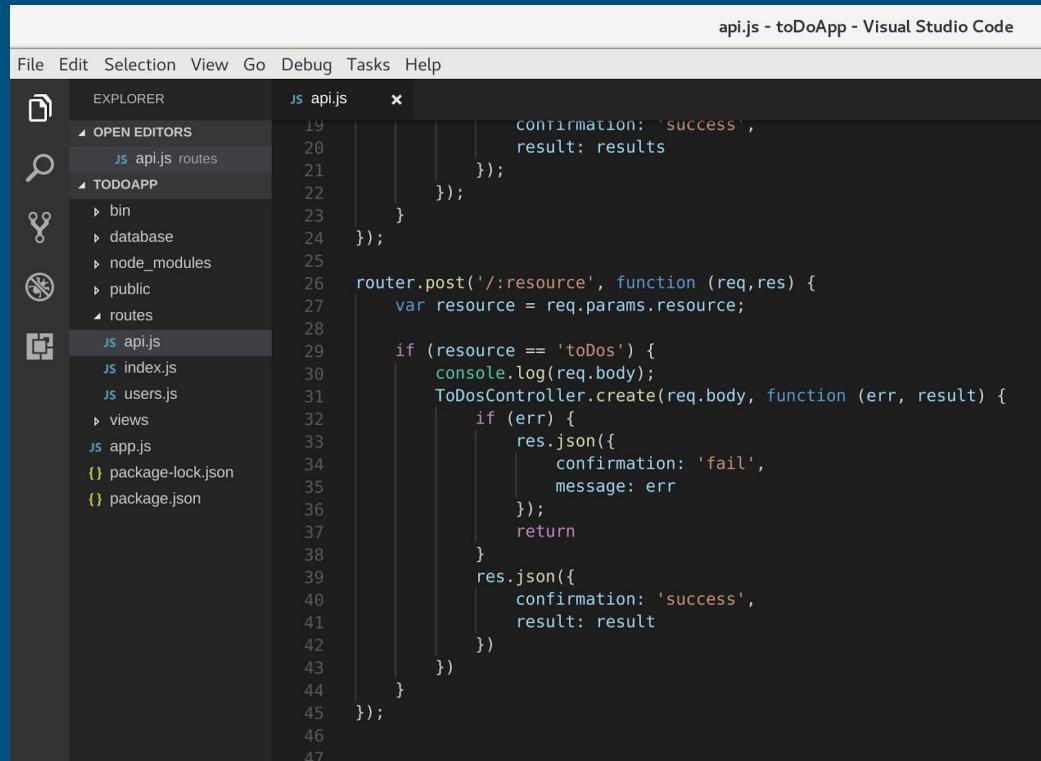
```
confirmation: "success"
result: []
```

Response

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

Adding information

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



The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** api.js - ToDoApp - Visual Studio Code
- File Menu:** File Edit Selection View Go Debug Tasks Help
- Explorer:** Shows the project structure:
 - OPEN EDITORS: api.js, routes
 - TODOAPP: bin, database, node_modules, public, routes
 - api.js (selected)
 - index.js, users.js, app.js, package-lock.json, package.json
- Code Editor:** The api.js file content is displayed, showing code for handling POST requests to '/:resource'. The code uses promises and the ToDosController.create method to handle the request and return a JSON response with confirmation and result fields.

```
api.js - Visual Studio Code
File Edit Selection View Go Debug Tasks Help
EXPLORER      api.js
OPEN EDITORS   routes
TODOAPP
  bin
  database
  node_modules
  public
  routes
  api.js
  index.js
  users.js
  views
  app.js
  package-lock.json
  package.json
19 |           confirmation: 'success',
20 |           result: results
21 |       });
22 |   });
23 | });
24 | });
25 |
26 router.post('/:resource', function (req,res) {
27     var resource = req.params.resource;
28
29     if (resource == 'toDos') {
30         console.log(req.body);
31         ToDosController.create(req.body, function (err, result) {
32             if (err) {
33                 res.json({
34                     confirmation: 'fail',
35                     message: err
36                 });
37             }
38             res.json({
39                 confirmation: 'success',
40                 result: result
41             });
42         });
43     }
44 });
45 });
46
47 
```

Code

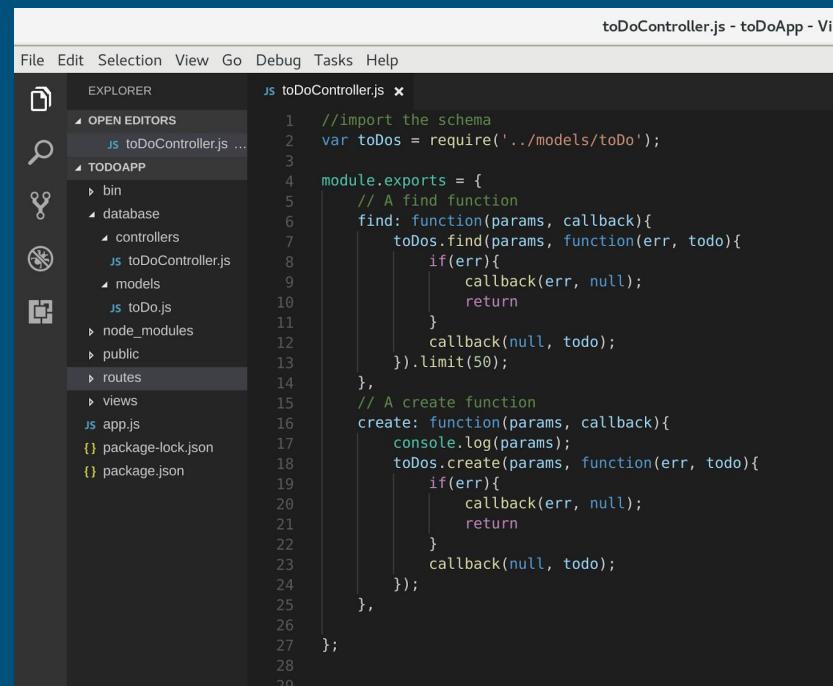
Lines 26 to 45 of '/routes/api.js'

```
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



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, it shows the project structure under "TODOAPP". The "routes" folder is currently selected.
- Code Editor:** On the right, the file "ToDoController.js" is open. The code defines two methods: "find" and "create".
- Code Content:**

```
1 //import the schema
2 var toDos = require('../models/todo');
3
4 module.exports = {
5   // A find function
6   find: function(params, callback){
7     toDos.find(params, function(err, todo){
8       if(err){
9         callback(err, null);
10      }
11      callback(null, todo);
12    }).limit(50);
13  },
14  // A create function
15  create: function(params, callback){
16    console.log(params);
17    toDos.create(params, function(err, todo){
18      if(err){
19        callback(err, null);
20      }
21      callback(null, todo);
22    });
23  },
24 25
26 27
28 29};
```

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

A screenshot of Visual Studio Code interface. The title bar says "Activities Visual Studio Code Sun 01:28". The code editor shows a file named "createToDo.hjs" with the following content:

```
<!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>
```

The Explorer sidebar on the left shows a project structure for "TODOAPP" with files like "bin", "database", "node_modules", "public", "routes", and "views". Inside "views", "createToDo.hjs" is selected. Other files shown include "error.hjs", "index.hjs", "app.js", "package-lock.json", and "package.json".

A screenshot of Visual Studio Code interface. The title bar says "Activities Visual Studio Code". The code editor shows a file named "createToDo.hjs" with the following content:

```
<!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>
```

The Explorer sidebar on the left shows a project structure for "TODOAPP" with files like "bin", "database", "node_modules", "public", "routes", and "views". Inside "views", "createToDo.hjs" is selected. Other files shown include "error.hjs", "index.hjs", "app.js", "package-lock.json", and "package.json".



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

The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists files and folders. In the main area, the file `index.js` is open, showing the following code:

```
File Edit Selection View Go Debug Tasks Help
EXPLORER
OPEN EDITORS
index.js routes
TODOAPP
bin
database
node_modules
public
routes
api.js
index.js
users.js
views
app.js
{} package-lock.json
{} package.json

index.js - toDoApp - V

var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.render('index', { title: 'Express' });
});

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

module.exports = router;
```



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.

Create ToDo

localhost:3000/createToDo

← → ⌛ ⌂ ⌃

Create ToDo

Task: First Task

Create ToDo

localhost:3000/api/toDos

localhost:3000/api/toDos

← → ⌛ ⌂ ⌃

JSON Raw Data Headers

Save Copy

```
confirmation: "success"
result:
  _v: 0
  _id: "5a23552d74b51a45f53e4a30"
  date_created: "2017-12-03T01:36:45.907Z"
  isComplete: false
  task: "First Task"
```



MongoDB

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
> █
```



MongoDB

Show collection

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

```
4 module.exports = {
5   // A find function
6   find: function(params, callback){
7     toDos.find(params, function(err, todo){
8       if(err){
9         callback(err, null);
```

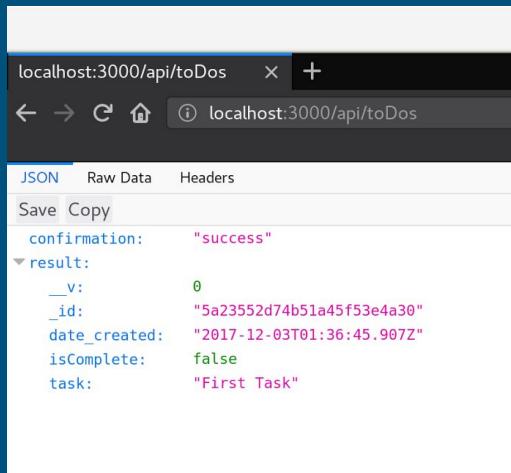
```
john@local: ~
File Edit View Search Terminal Help
> show dbs
admin          0.000GB
local          0.000GB
ToDoDB         0.000GB
tvDatabase     0.000GB
> use ToDoDB
switched to db ToDoDB
> show collections
todos
> █
```



MongoDB

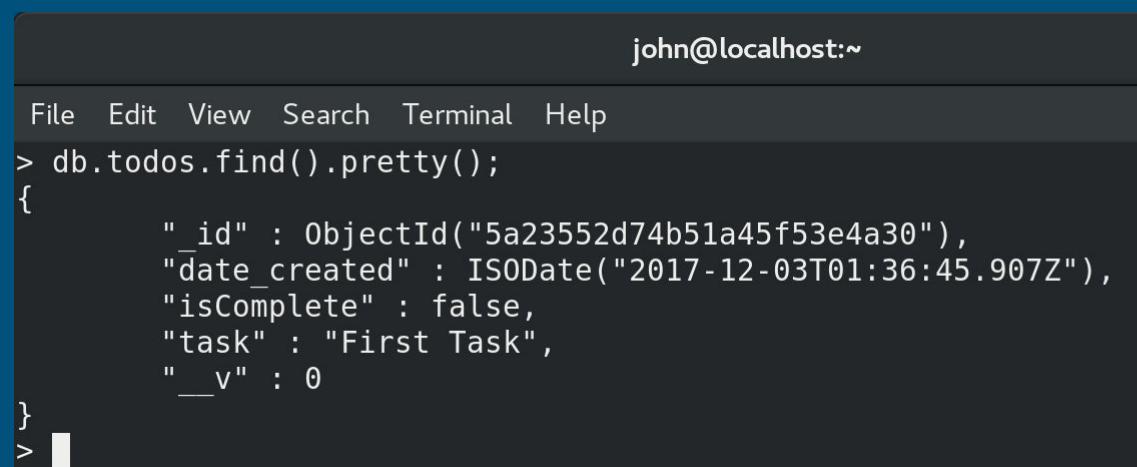
Collections content

- 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



A screenshot of a browser window showing a JSON response. The URL is `localhost:3000/api/toDos`. The response is:

```
JSON Raw Data Headers
Save Copy
confirmation: "success"
result:
  _v: 0
  _id: "5a23552d74b51a45f53e4a30"
  date_created: "2017-12-03T01:36:45.907Z"
  isComplete: false
  task: "First Task"
```



A screenshot of a terminal window showing the MongoDB shell. The prompt is `john@localhost:~`. The command entered is `> db.todos.find().pretty();`. The output is:

```
File Edit View Search Terminal Help
> db.todos.find().pretty();
{
    "_id" : ObjectId("5a23552d74b51a45f53e4a30"),
    "date_created" : ISODate("2017-12-03T01:36:45.907Z"),
    "isComplete" : false,
    "task" : "First Task",
    "__v" : 0
}> █
```



Render List

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

The screenshot shows the Visual Studio Code interface with the title bar "index.js - ToDoApp - Visual Studio Code". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The Explorer sidebar on the left lists project files: OPEN EDITORS (index.js routes), TODOAPP (bin, database, node_modules, public, routes, api.js, index.js, users.js, views, app.js), and package-lock.json, package.json. The main editor area displays the "index.js" code:

```
10 router.get('/createToDo', function(req, res, next) {
11
12   ToDosController.find(req.query, function (err, results) {
13     if (err) {
14       res.json({
15         confirmation: 'fail',
16         message: err
17       });
18     }
19     var toDos = []
20     for(var i = 0; i<results.length; i++){
21       toDos.push(results[i].task)
22     }
23     res.render('createToDo', { list: toDos});
24     console.log(results[0].task);
25   });
26
27
28 });

29 }
```



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

```
<!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"/>
        <ul>
            {{#list}}
                <li>{{.}}</li>
            {{/list}}
        </ul>
    </form>
```



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>

A screenshot of a web browser window titled "Create ToDo". The address bar shows "localhost:3000/createToDo". The main content area displays the heading "Create ToDo" and a form with a "Task:" input field and a "Create ToDo" button. Below the form, a list of tasks is shown:

- First Task
- Second Task
- Third Task