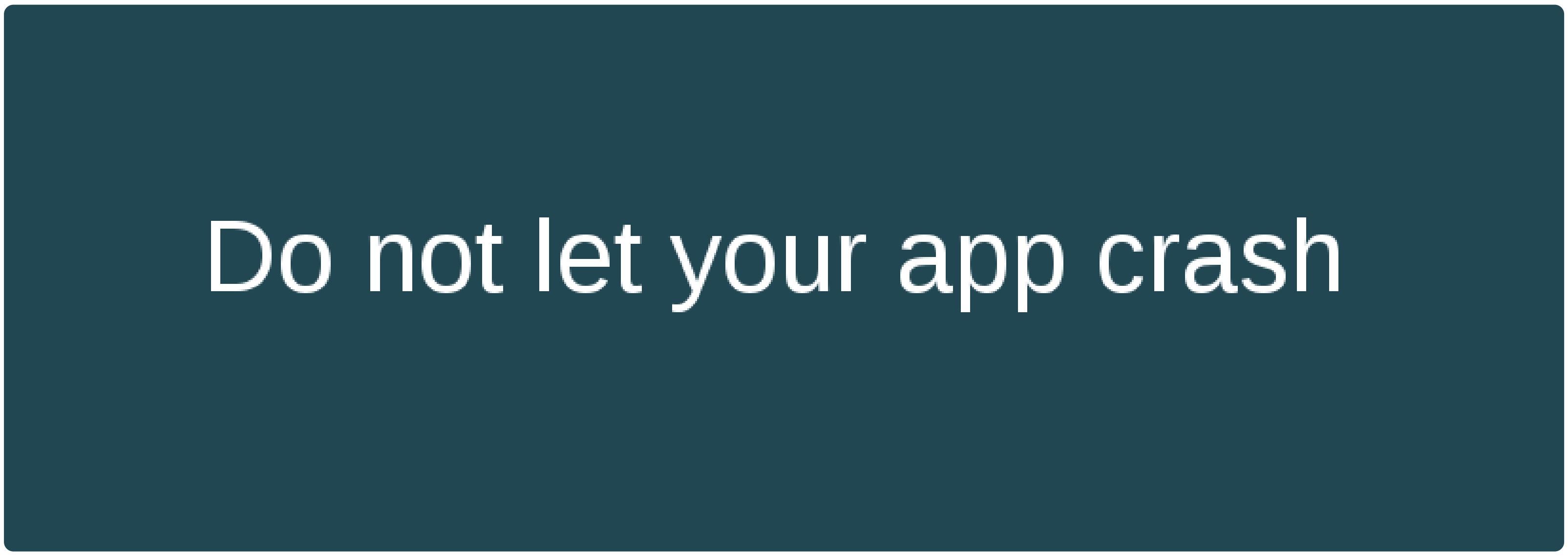
Error handling in Express using Middleware

2 Comments / Short-tips, Node.js tutorials / By Shahid



Express adopts middleware design pattern for the router handling. Express also provides us simple middleware which we can use to handle any run-time errors i.e in case of run-time errors your app will not stop, rather it will call the error handling middle-ware.

Related learning : How to write custom middle-ware in Express

Table of Contents

A quick picture

In general when we use Express middle-ware we pass three fields :

(req,res,next)

In case of error handler middle-ware we pass one extra parameter, which is **error** one.

(error,req,res,next)

Sample code to get started

Consider following Express.js code with middle-wares.

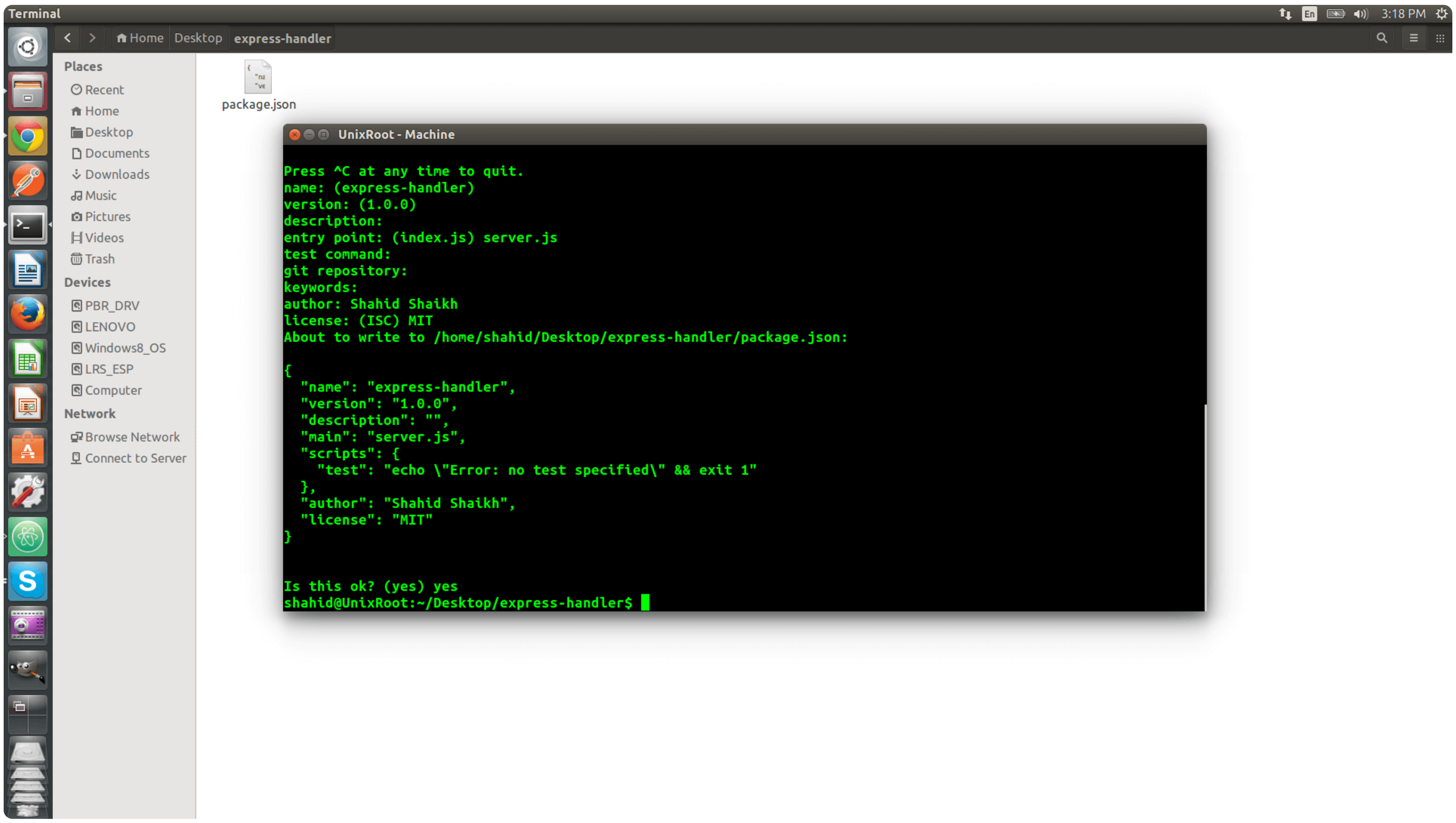
```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');

app.use(bodyParser);

app.use(function(req,res,next) {
  console.log("In second route");
  next(); // go to next route.
});

app.listen(3000);
```

Here is same Express.js code but with Error handling middle-ware.

```
var express = require('express');
var app = express();
var bodyParser = require('body-parser');

app.use(bodyParser);

app.use(function(req,res,next) {
  console.log("In second route");
  next(); // go to next route.
});

// Error handling middle-ware

app.use(function(err,req,res,next) {
  console.log("Error happens",err.stack);
});

app.listen(3000);
```

Sample project

Let's develop small project to prove this concept. Create new directory and generate **package.json** in it. Recommended way is to use **npm init** command.



```
Press ^C at any time to quit.
name: (express-handler)
version: (1.0.0)
description:
entry point: (index.js) server.js
test command:
git repository:
keywords:
author: Shahid Shaikh
license: (ISC) MIT
About to write to /home/shahid/Desktop/express-handler/package.json:

{
  "name": "express-handler",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Shahid Shaikh",
  "license": "MIT"
}


Is this ok? (yes) yes
shahid@UnixRoot:~/Desktop/express-handler$
```

Here is my **package.json**.

package.json
```
{
  "name": "express-handler",
  "version": "1.0.0",
  "description": "",
  "main": "server.js",
  "scripts": {
    "test": "echo "Error: no test specified" && exit 1"
  },
  "author": "Shahid Shaikh",
  "license": "MIT"
}
```

Let's install **Express** module. Run following command in termin

```
npm i --save express
```

You may need to provide **sudo** access to it. Let's code our Serv

Express Server

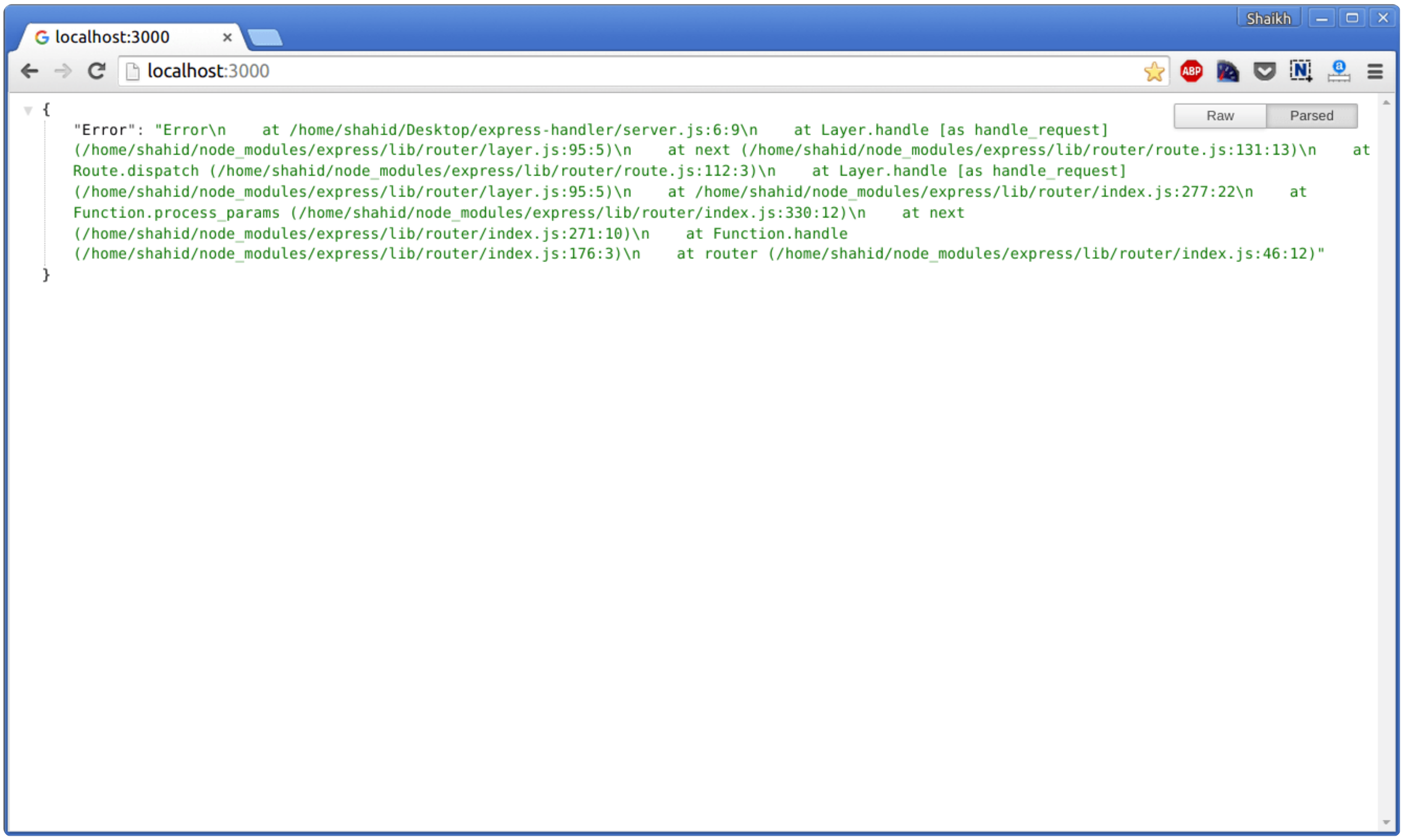
Here is simple Express Server with error handler middle-ware.

```
Server.js
var express = require('express');
var app = express();
var router = express.Router();

router.get('/',function(req,res) {
  res.send("Hello World!");
});

app.use('/',router);

app.use(function(err,req,res,next) {
  console.log(err.stack);
  res.status(500).send({"Error" : err.stack});
});

app.listen(3000);
```

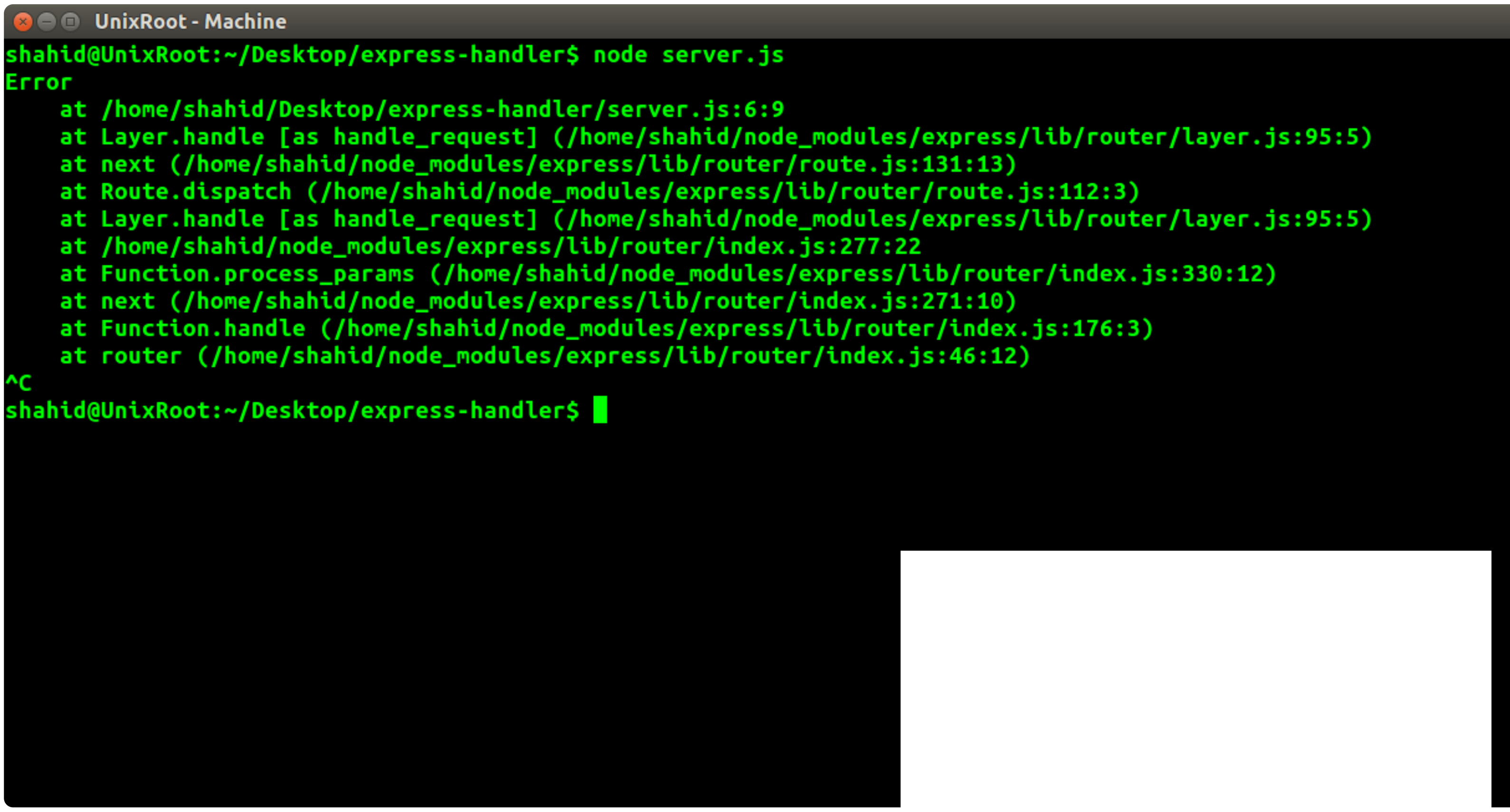
Run the app using **node Server.js** command and visit **localhost:3000** to view the app. However we need to validate whether our error handler logic is working or not. To generate the run-time error, we will throw random error and see whether it catches it or not.

```
Server.js
var express = require('express');
var app = express();
var router = express.Router();

router.get('/',function(req,res) {
  throw new Error();
  res.send("Hello World!");
});

app.use('/',router);

app.use(function(err,req,res,next) {
  console.log(err.stack);
  res.status(500).send({"Error" : err.stack});
});

app.listen(3000);
```

Run the app and hit **localhost:3000** url. You should see following error in console and in browser too.



```
{
    "Error": "Error\n    at /home/shahid/Desktop/express-handler/server.js:6:9\n    at Layer.handle [as handle_request] (/home/shahid/node_modules/express/lib/router/layer.js:95:5)\n    at next (/home/shahid/node_modules/express/lib/router/route.js:131:13)\n    at Route.dispatch (/home/shahid/node_modules/express/lib/router/route.js:112:3)\n    at Layer.handle [as handle_request] (/home/shahid/node_modules/express/lib/router/layer.js:95:5)\n    at /home/shahid/node_modules/express/lib/router/index.js:277:22\n    at Function.process_params (/home/shahid/node_modules/express/lib/router/index.js:330:12)\n    at next (/home/shahid/node_modules/express/lib/router/index.js:271:10)\n    at Function.handle (/home/shahid/node_modules/express/lib/router/index.js:176:3)\n    at router (/home/shahid/node_modules/express/lib/router/index.js:46:12)"
}
```



```
shahid@UnixRoot:~/Desktop/express-handler$ node server.js
Error
    at /home/shahid/Desktop/express-handler/server.js:6:9
    at Layer.handle [as handle_request] (/home/shahid/node_modules/express/lib/router/layer.js:95:5)
    at next (/home/shahid/node_modules/express/lib/router/route.js:131:13)
    at Route.dispatch (/home/shahid/node_modules/express/lib/router/route.js:112:3)
    at Layer.handle [as handle_request] (/home/shahid/node_modules/express/lib/router/layer.js:95:5)
    at /home/shahid/node_modules/express/lib/router/index.js:277:22
    at Function.process_params (/home/shahid/node_modules/express/lib/router/index.js:330:12)
    at next (/home/shahid/node_modules/express/lib/router/index.js:271:10)
    at Function.handle (/home/shahid/node_modules/express/lib/router/index.js:176:3)
    at router (/home/shahid/node_modules/express/lib/router/index.js:46:12)
^C
shahid@UnixRoot:~/Desktop/express-handler$
```

I did press CTRL+C

This works!

Conclusion

Express is no doubt first choice of Node.js developer for developing the web application. Express error handling middle-ware is very helpful in catching globally occurring errors without worrying about the Server to crash.