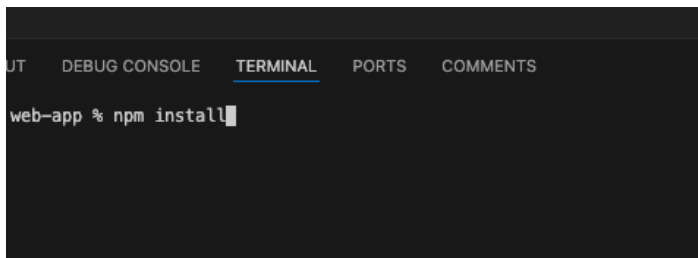


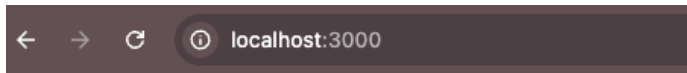
Installation and Configuration

1. Install Node.js 18 or higher.
2. We will use an express generator for ease of using the framework
3. Run `npm install -g express-generator` to install the generator globally
4. Let's make a folder, run command `mkdir web-app`
5. Since we are only making APIs we do not need a view engine. So run this command `express --no-view web-app`
 - a. It will install the express framework inside the created folder web-app
6. After installing, open your project folder web-app inside visual studio code and open terminal
 - a. In the terminal window run command `npm install`



```
web-app % npm install
```

7. Once you have installed all node modules successfully. Run another command to start the express web application `npm run start`
8. To verify that it is running successfully, go to a browser and type in localhost:3000. Then you should be able to see the express application running.

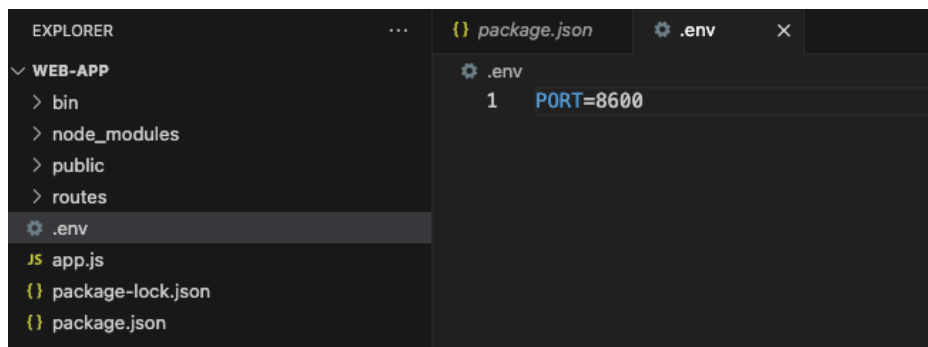


Express

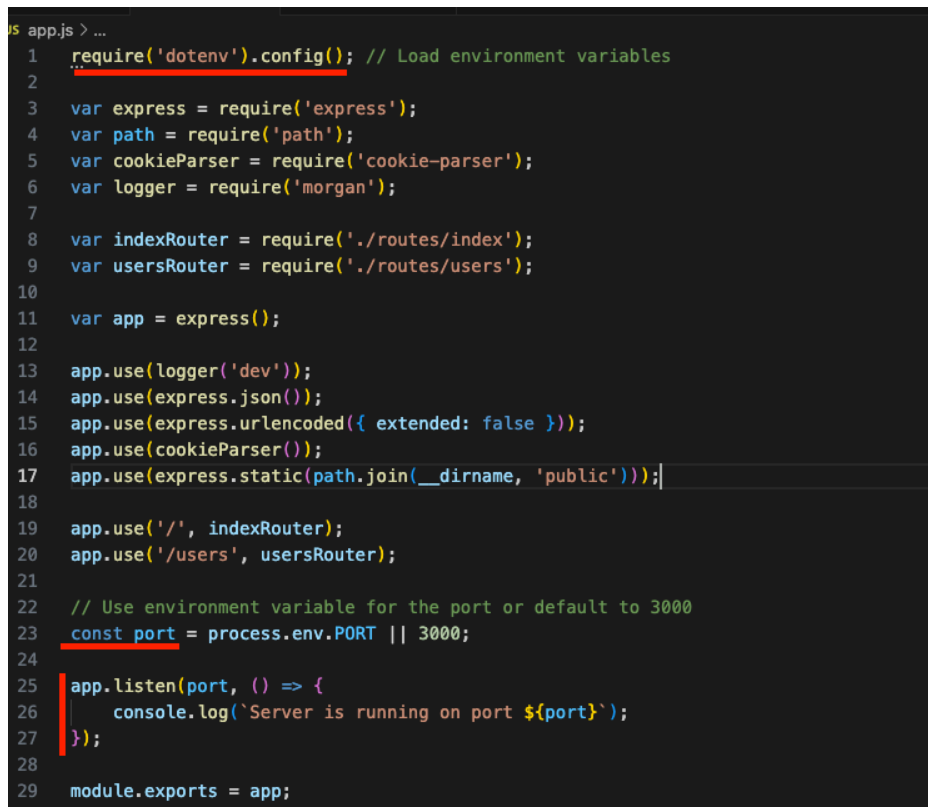
Welcome to Express

9. Now we need to create environment variables for our web application
10. First install the package dotenv `npm install dotenv`
11. In the root directory of your Express.js project, create a `.env` file. This file will store your environment variables.

12. Open the .env file and add a port value, you can choose any value above 4000.



13. Next is to edit app.js and add in the dotenv library.



14. Next is to update the start script of the package.json file.

```
} package.json > {} scripts > start
1  {
2    "name": "web-app",
3    "version": "0.0.0",
4    "private": true,
5    "scripts": {
6      "start": "node app.js"
7    },
8    "dependencies": {
9      "cookie-parser": "~1.4.4",
10     "debug": "~2.6.9",
11     "dotenv": "^16.4.5",
12     "express": "~4.16.1",
13     "morgan": "~1.9.1"
14   }
15 }
```

15. After doing those edits, re-run the application by running `npm run start` then update the app url to localhost:[port]

```
> web-app@0.0.0 start
> node app.js

Server is running on port 8600
GET / 200 10.082 ms - 176
GET /stylesheets/style.css 200 1.661 ms - 111
GET /favicon.ico 404 1.231 ms - 150
```

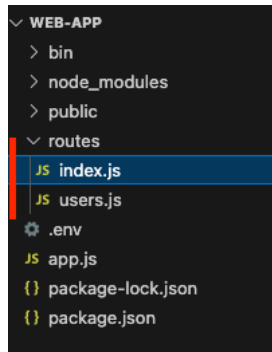
← → ↻ ⓘ localhost:8600

Express

Welcome to Express

Routing

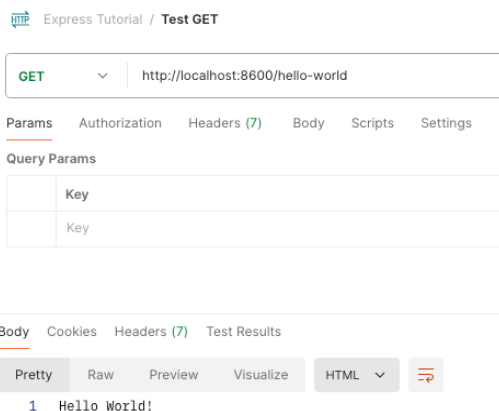
1. In the generate application, we already have router examples
2. Open the routes folder and you will see 2 routes already inside. The index and users routes.



3. To update or add new routes open one of them. Let us add a new route under the index.js file. Open the file and add the code.

```
routes > JS index.js > ...
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET home page. */
5  router.get('/', function(req, res, next) {
6    res.render('index', { title: 'Express' });
7  });
8
9  router.get('/hello-world', function(req, res, next) {
10    res.send('Hello World!');
11  });
12
13 module.exports = router;
14
```

4. Save the file and restart the express application, and your route is already available. Just use postman to test the route like shown here.



5. Now let us create a post route in the user.js file. Use the code below and test using postman.

```
routes > JS users.js > router.post('/create') callback
1  var express = require('express');
2  var router = express.Router();
3
4  /* GET users listing. */
5  router.get('/', function(req, res, next) {
6    res.send('respond with a resource');
7  });
8
9  router.post('/create', function(req, res, next){
10   res.send('this is a post request. ');
11 });
12
13 module.exports = router;
14
```



Express Tutorial / Test POST

POST



http://localhost:8600/users/create

Params

Authorization

Headers (8)

Body

Scripts

Settings

Query Params

	Key
	Key

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

HTML



1 this is a post request.

Connect to a Database

1. Let us install sqlite so that we can save and get data.
2. Run this command `npm install sqlite3`
3. Then edit app.js to include the newly added mongoose.

```
...require('dotenv').config(); // Load environment variables

var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
const sqlite3 = require('sqlite3').verbose();
```

```
// Connect to SQLite database (creates file if it doesn't exist)
const db = new sqlite3.Database('./database.sqlite', (err) => {
  if (err) {
    console.error('Error opening database ' + err.message);
  } else {
    console.log('Connected to the SQLite database.');
```

```
  });

  // Create users table if it doesn't exist
  db.run(`CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    age INTEGER NOT NULL
  );`);

  app.use('/', indexRouter);
  app.use('/users', usersRouter);
```

```
  // Use environment variable for the port or default to 3000
  const port = process.env.PORT || 3000;

  app.listen(port, () => {
    console.log(`Server is running on port ${port}`);
  });

  // Close the database connection when the application ends
  process.on('SIGINT', () => {
    db.close((err) => {
      if (err) {
        console.error('Error closing database ' + err.message);
      }
      console.log('Database connection closed.');
```

```
      process.exit(0);
    });
  });

  module.exports = app;
```

4. Next is to update the user route, so go to routes folder and edit users.js.

```
var express = require('express');
var router = express.Router();
const sqlite3 = require('sqlite3').verbose();

//Db Connection
const db = new sqlite3.Database('./database.sqlite');
```

5. Then update the POST route.

```
// POST route to create a new user
router.post('/', async (req, res) => {
  try {
    console.log(req.body);
    const { name, email, age } = req.body;

    db.run('INSERT INTO users (name, email, age) VALUES (?, ?, ?)', [name, email, age], function (err) {
      if (err) {
        res.status(500).json({ error: err.message });
        return;
      }
      res.status(201).json({ id: this.lastID });
    });
  } catch (error) {
    res.status(400).json({ message: error.message });
  }
});
```

6. Next is to update the GET route

```
// GET route to get all users
router.get('/', async (req, res) => {
  try {
    db.all('SELECT * FROM users', [], (err, rows) => {
      if (err) {
        res.status(500).json({ error: err.message });
        return;
      }
      if (rows.length > 0) {
        res.status(200).json(rows);
      } else {
        res.status(200).json({
          "message": "No data found."
        });
      }
    });
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
});
```

7. To test both APIs you need to use postman.

POST

 Express Tutorial / Users POST

POST

http://localhost:8600/users

Params Authorization Headers (9) Body Scripts Settings

☐ none

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	Key	Value
<input checked="" type="checkbox"/>	name	test1
<input checked="" type="checkbox"/>	email	tester@testers.com
<input checked="" type="checkbox"/>	age	20
	Key	Value

Body Cookies Headers (7) Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"id": 1

3

}

GET

 Express Tutorial / **Users GET**

GET



http://localhost:8600/users

Params

Authorization

Headers (7)

Body

Scripts

Settings

Query Params

Key	
Key	

Body

Cookies

Headers (7)

Test Results

Pretty

Raw

Preview

Visualize

JSON



```
1  [
2    {
3      "id": 1,
4      "name": "test1",
5      "email": "tester@testers.com",
6      "age": 20
7    },
8    {
9      "id": 2,
10     "name": "test2",
11     "email": "tester2@testers.com",
12     "age": 88
13   }
14 ]
```