

TP-08

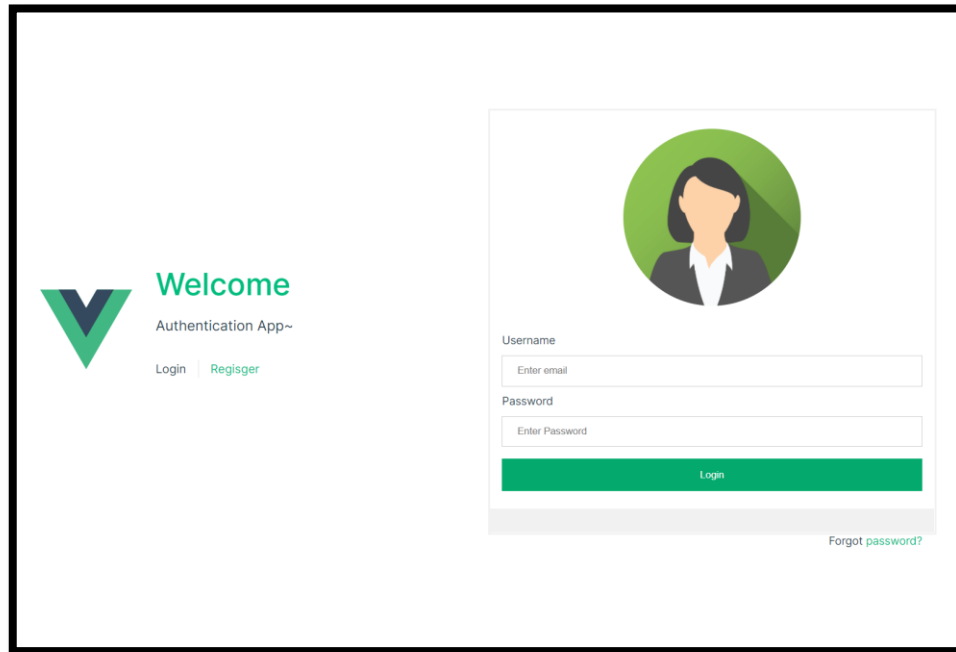
VueJS, NodeJS

(Authentication frontend and APIs)

TP07 Exercise

TP07.1: Auth screens in VueJS

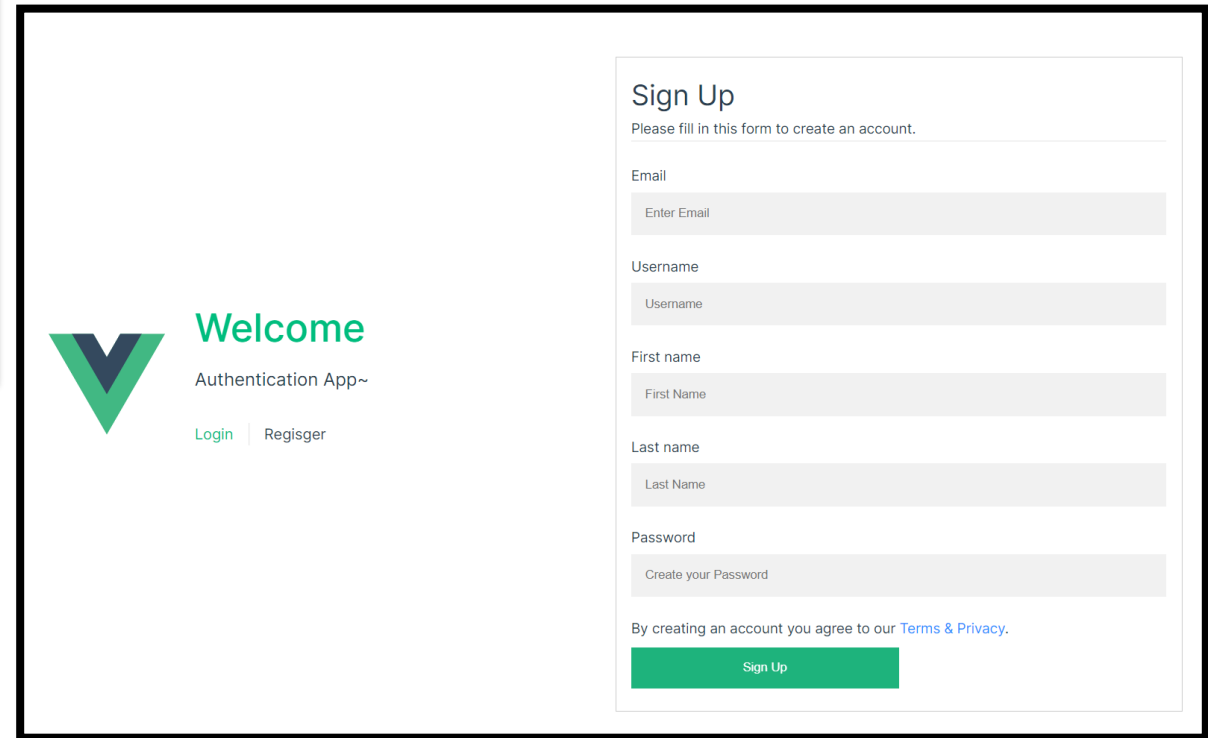
Create a simple login and register form in VueJS



The login screen features a green Vue.js logo on the left. To its right, the text 'Welcome' is displayed in green, followed by 'Authentication App~' in a smaller font. Below this, there are two links: 'Login' and 'Register'. The main form area contains a circular profile picture placeholder with a green background and a person icon. Below the profile picture, there are two input fields: 'Username' with the placeholder text 'Enter email' and 'Password' with the placeholder text 'Enter Password'. A green 'Login' button is positioned below the password field. At the bottom right of the form, there is a link for 'Forgot password?'.

By using VueJS router, we can access by

- **Login:** `http://localhost:3000/`
- **Register:** `http://localhost:3000/register`
- **Home:** `http://localhost:3000/home`

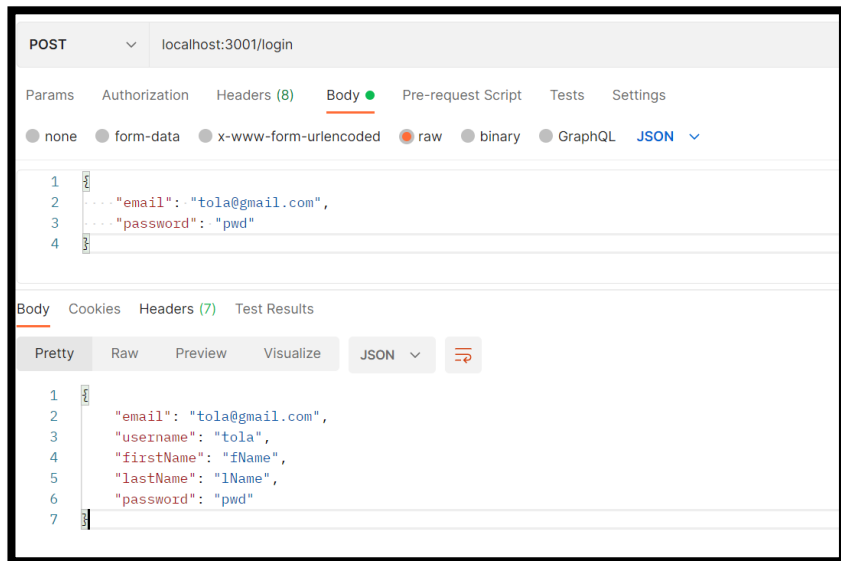


The sign up screen features the same green Vue.js logo and 'Welcome' text as the login screen. Below the logo, there are two links: 'Login' and 'Register'. The main form area is titled 'Sign Up' in green, followed by the instruction 'Please fill in this form to create an account.' Below this, there are four input fields: 'Email' with placeholder 'Enter Email', 'Username' with placeholder 'Username', 'First name' with placeholder 'First Name', and 'Last name' with placeholder 'Last Name'. Below these, there is a 'Password' section with a placeholder 'Create your Password'. At the bottom, there is a green 'Sign Up' button. Below the button, there is a line of text: 'By creating an account you agree to our [Terms & Privacy](#)'.

TP07 Exercise

TP07.2: NodeJS with ExpressJS

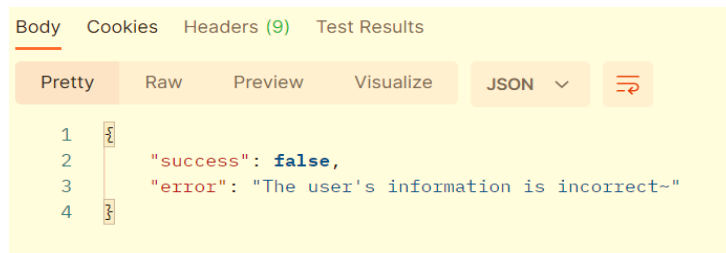
Create a nodejs APIs for authentication with ExpressJS. Use Json file as a database to store user's information.



```
POST localhost:3001/login

{
  "email": "tola@gmail.com",
  "password": "pwd",
  "username": "tola",
  "firstName": "fName",
  "lastName": "lName"
}
```

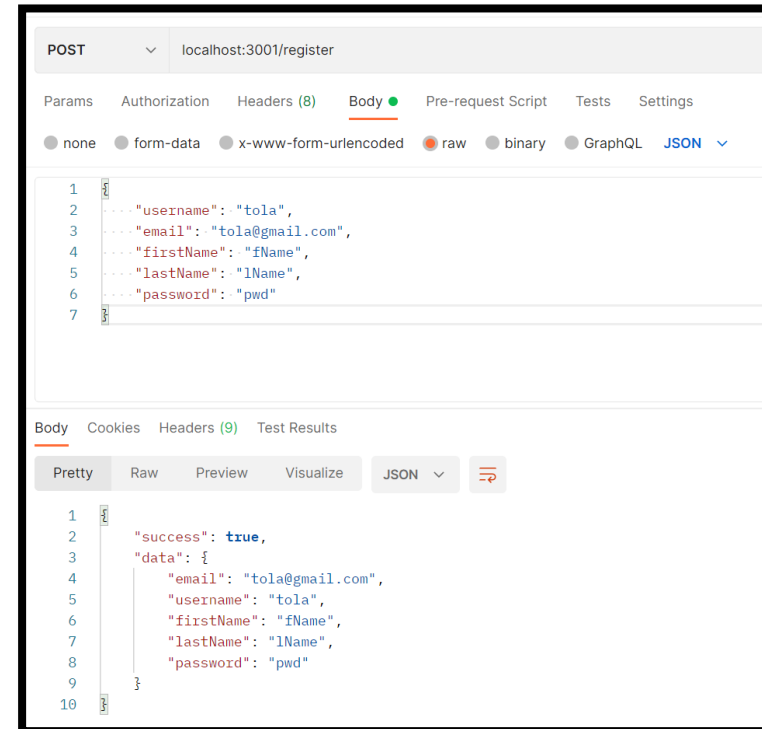
```
{
  "email": "tola@gmail.com",
  "username": "tola",
  "firstName": "fName",
  "lastName": "lName",
  "password": "pwd"
}
```



```
POST localhost:3001/register

{
  "email": "tola@gmail.com",
  "password": "pwd",
  "username": "tola",
  "firstName": "fName",
  "lastName": "lName"
}
```

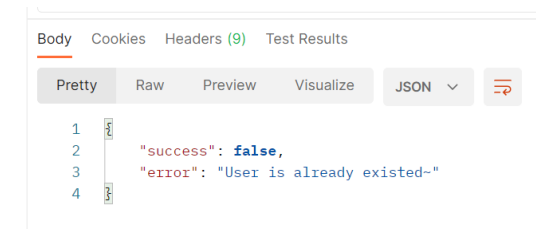
```
{
  "success": false,
  "error": "The user's information is incorrect-"
}
```



```
POST localhost:3001/register

{
  "email": "tola@gmail.com",
  "password": "pwd",
  "username": "tola",
  "firstName": "fName",
  "lastName": "lName"
}
```

```
{
  "success": true,
  "data": {
    "email": "tola@gmail.com",
    "username": "tola",
    "firstName": "fName",
    "lastName": "lName",
    "password": "pwd"
  }
}
```



```
POST localhost:3001/register

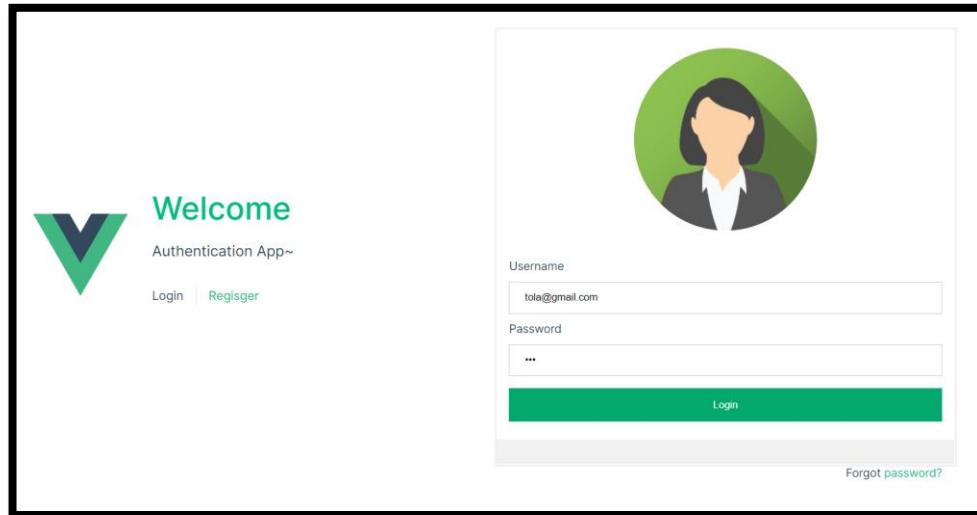
{
  "email": "tola@gmail.com",
  "password": "pwd",
  "username": "tola",
  "firstName": "fName",
  "lastName": "lName"
}
```

```
{
  "success": false,
  "error": "User is already existed-"
}
```

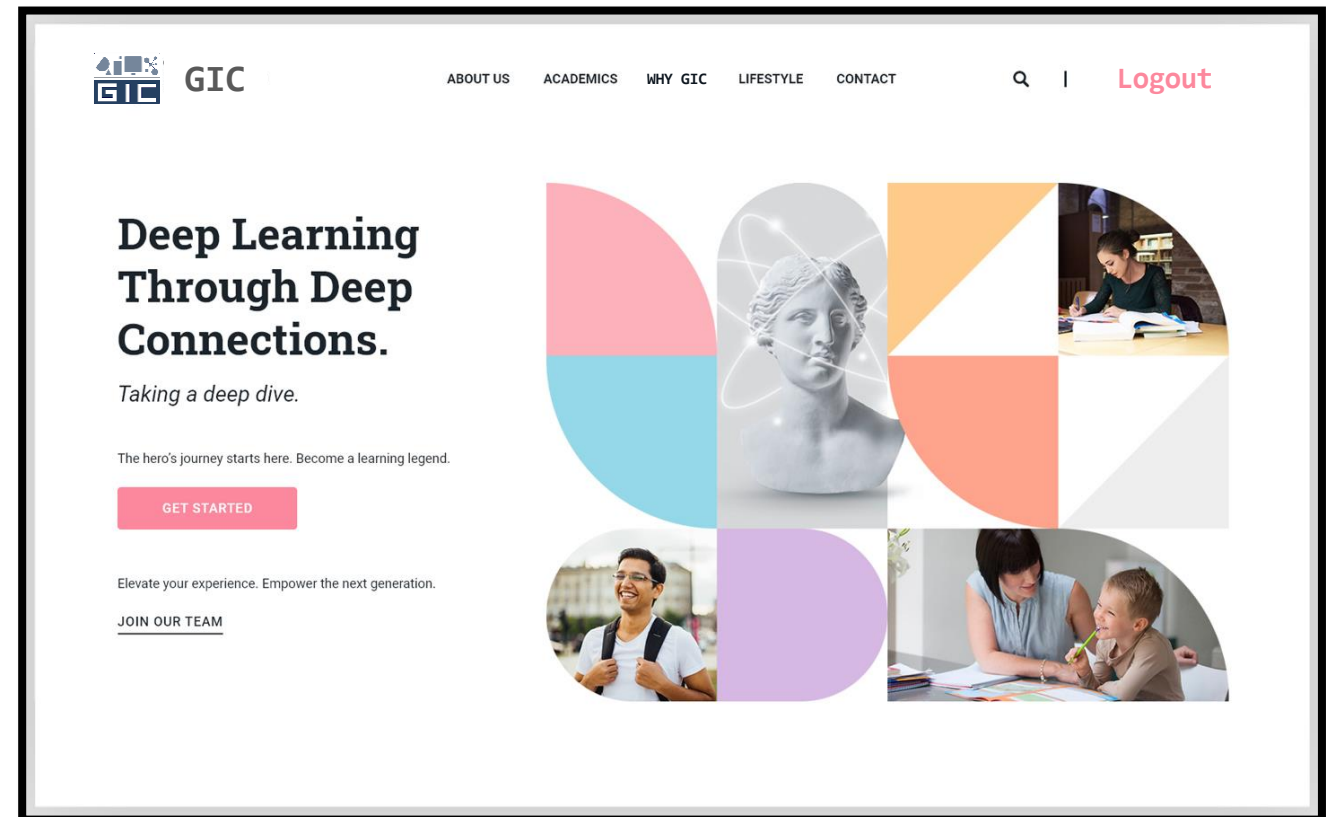
TP07 Exercise

TP07.3: UI and APIs Integration

Integrate Login/Register app with APIs

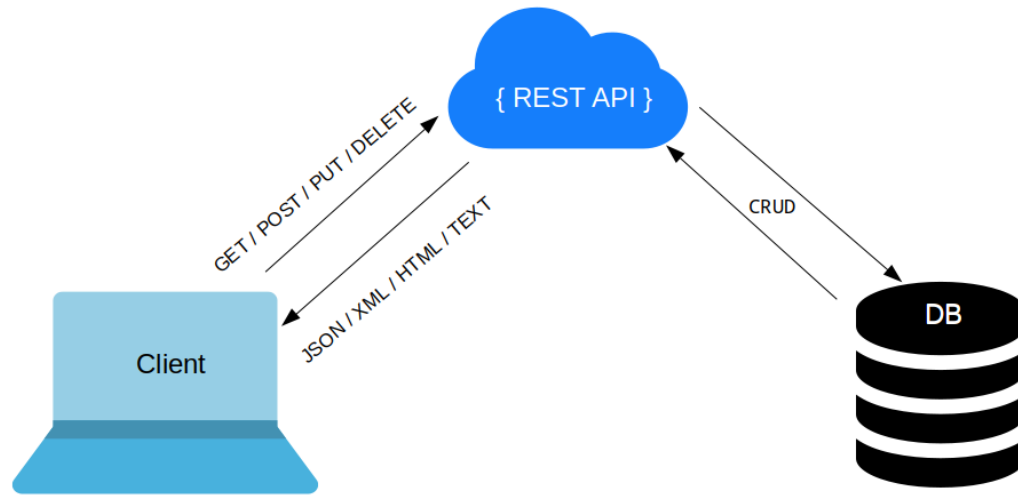


A user authentication interface mockup. On the left, a green 'V' logo is next to the text 'Welcome Authentication App~'. Below this are links for 'Login' and 'Register'. On the right, there is a circular profile picture placeholder with a green background and a person icon. Below the profile picture are input fields for 'Username' (containing 'tola@gmail.com') and 'Password' (containing three dots). A green 'Login' button is below the password field. At the bottom right, there is a link for 'Forgot password?'.

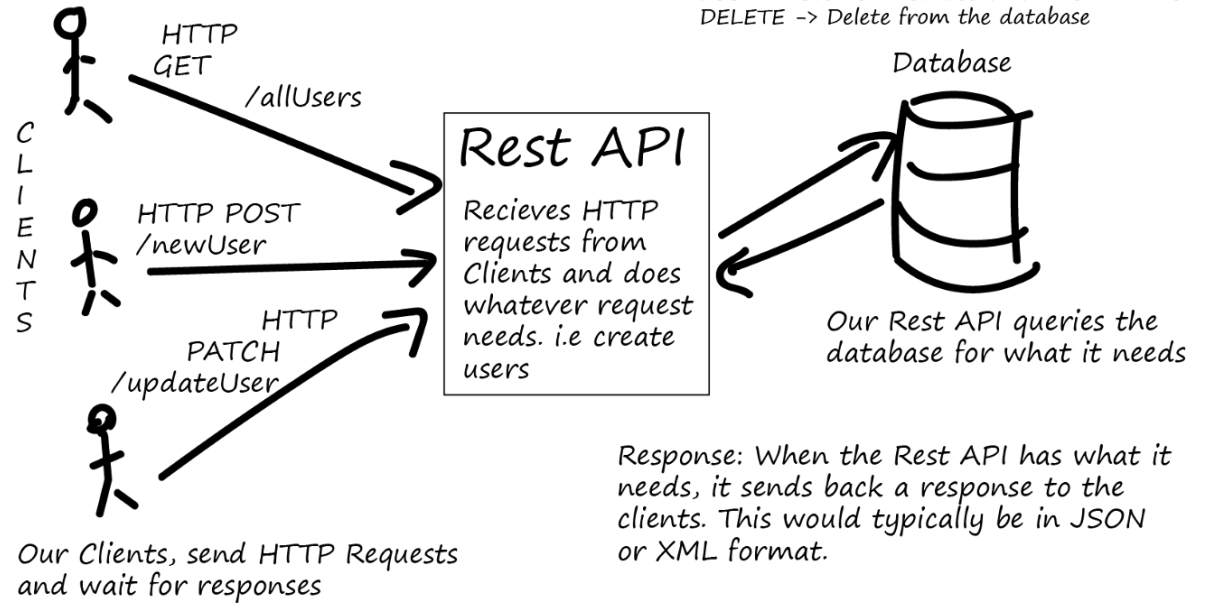


Getting to understand “ExpressJS”

Rest APIs



Rest API Basics



ExpressJS

```
const express = require('express') 4.17.3 )
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`);
})
```

```
$ node app.js
```

ExpressJS

Basic routing

```
app.get('/', (req, res) => {  
  res.send('Hello World!')  
})
```

Respond to POST request on the root route (/), the application's home page:

```
app.post('/', (req, res) => {  
  res.send('Got a POST request')  
})
```

Respond to a PUT request to the /user route:

```
app.put('/user', (req, res) => {  
  res.send('Got a PUT request at /user')  
})
```

Respond to a DELETE request to the /user route:

```
app.delete('/user', (req, res) => {  
  res.send('Got a DELETE request at /user')  
})
```


ExpressJS

Serving static files in Express

```
express.static(root, [options])
```

```
app.use(express.static('public'))
```

```
http://localhost:3000/images/kitten.jpg  
http://localhost:3000/css/style.css  
http://localhost:3000/js/app.js  
http://localhost:3000/images/bg.png  
http://localhost:3000/hello.html
```

```
const path = require('path')  
app.use('/static', express.static(path.join(__dirname, 'public')))
```

```
http://localhost:3000/static/images/kitten.jpg  
http://localhost:3000/static/css/style.css  
http://localhost:3000/static/js/app.js  
http://localhost:3000/static/images/bg.png  
http://localhost:3000/static/hello.html
```

ExpressJS

Middleware

```
var express = require('express');
var app = express();
```

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

```
app.get('/', function(req, res, next) {
  next();
})
```

Callback argument to the middleware function, called "next" by convention.

HTTP response argument to the middleware function, called "res" by convention.

HTTP request argument to the middleware function, called "req" by convention.

```
app.listen(3000);
```

```
const express = require('express')
const app = express()

const myLogger = function (req, res, next) {
  console.log('LOGGED')
  next()
}

app.use(myLogger)

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(3000)
```

```
const express = require('express')
const cookieParser = require('cookie-parser')
const cookieValidator = require('./cookieValidator')

const app = express()

async function validateCookies (req, res, next) {
  await cookieValidator(req.cookies)
  next()
}

app.use(cookieParser())

app.use(validateCookies)

// error handler
app.use((err, req, res, next) => {
  res.status(400).send(err.message)
})

app.listen(3000)
```

ExpressJS

Application-level middleware

```
const express = require('express')
const app = express()

app.use((req, res, next) => {
  console.log('Time:', Date.now())
  next()
})
```

```
app.use('/user/:id', (req, res, next) => {
  console.log('Request URL:', req.originalUrl)
  next()
}, (req, res, next) => {
  console.log('Request Type:', req.method)
  next()
})
```

```
function logOriginalUrl (req, res, next) {
  console.log('Request URL:', req.originalUrl)
  next()
}
```

```
function logMethod (req, res, next) {
  console.log('Request Type:', req.method)
  next()
}
```

```
const logStuff = [logOriginalUrl, logMethod]
app.get('/user/:id', logStuff, (req, res, next) => {
  res.send('User Info')
})
```

ExpressJS

Router-level middleware

```
const router = express.Router()
```

```
const express = require('express')
const app = express()
const router = express.Router()

// a middleware function with no mount path. This code is executed for every request to the router
router.use((req, res, next) => {
  console.log('Time:', Date.now())
  next()
})

// a middleware sub-stack shows request info for any type of HTTP request to the /user/:id path
router.use('/user/:id', (req, res, next) => {
  console.log('Request URL:', req.originalUrl)
  next()
}, (req, res, next) => {
  console.log('Request Type:', req.method)
  next()
})
```

Built-in middleware

- `express.static` serves static assets such as HTML files, images, and so on.
- `express.json` parses incoming requests with JSON payloads. **NOTE: Available with Express 4.16.0+**
- `express.urlencoded` parses incoming requests with URL-encoded payloads. **NOTE: Available with Express 4.16.0+**

Third-party middleware

```
const express = require('express')
const app = express()
const cookieParser = require('cookie-parser')

// load the cookie-parsing middleware
app.use(cookieParser())
```

I want more about ExpressJS

 <https://expressjs.com/>

Good luck 🍀