Express.js(API)

Wednesday, February 26, 2025 7:12 PM

Twitter Server App

- **Postman** :-it used for testing Api
- **npx nodemon src/index.js**:server automatically start after some changes in index.js file

Using middleware

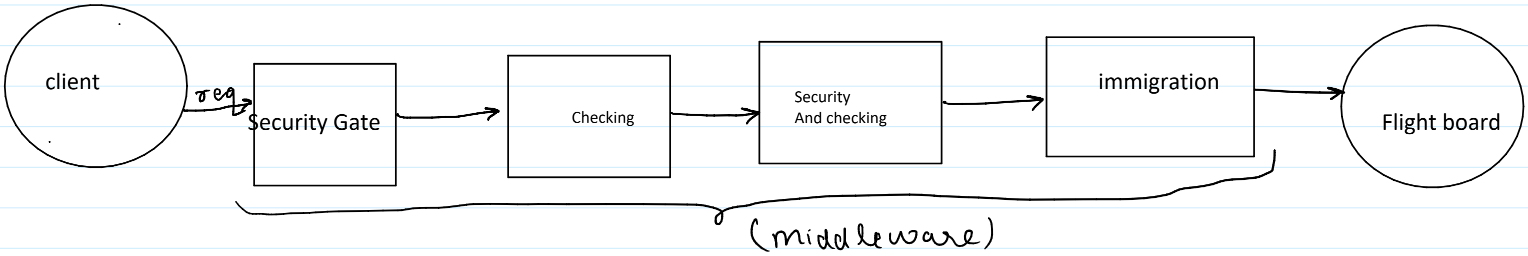
Express is a routing and middleware web framework that has minimal functionality of its own: An Express application is essentially a series of middleware function calls.

Middleware functions are functions that have access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle. The next middleware function is commonly denoted by a variable named next.

Middleware functions can perform the following tasks:

- Execute any code.
- Make changes to the request and the response objects.
- End the request-response cycle.
- Call the next middleware function in the stack.

From <https://expressjs.com/en/guide/using-middleware.html>



* f(req, res, next) → g(req, res, next) → h(req, res, next)

=Example of middleware

```
function mid1(req,res,next){
    console.log('mid1');
    next();
}
function mid2(req,res,next){
    console.log('mid2');
    next();
}
function mid3(req,res,next){
    console.log('mid3');
    next();
```

mid1()

↓

mid2()

↓

mid3()

↓

(req, res) => {}

```
}
```

```
app.get('/ping' ,mid1,mid2,mid3,(req,res)=>{
    return res.json({
        message : 'pong'
    });
});
```

client → /ping → mid1 (req, res, next) → mid2(req, res, next) → mid3 (req, res, next)

- **Npm I morgan**

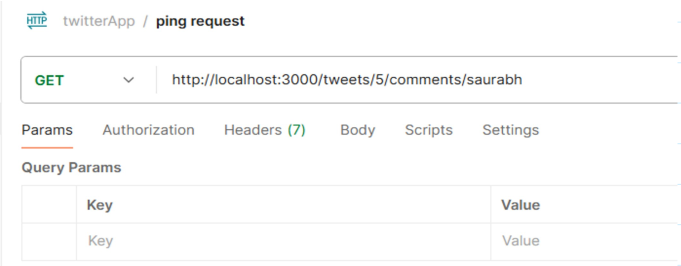
morgan(format, options)

Create a new morgan logger middleware function using the given format and options.
The format argument may be a string of a predefined name (see below for the names), a string of a format string, or a function that will produce a log entry.

The format function will be called with three arguments tokens, req, and res, where tokens is an object with all defined tokens, req is the HTTP request and res is the HTTP response. The function is expected to return a string that will be the log line, or undefined / null to skip logging.

From <https://www.npmjs.com/package/morgan#morganformat-options>

```
app.get('/tweets/:tweet_id/comments/:comment_id',(req,res)=>{
    console.log(req.params);
    return res.json({
        message:'tweet details'
    });
});
```



twitterApp / ping request

GET http://localhost:3000/tweets/5/comments/saurabh

Query Params

Key	Value
Key	Value

The URL encoding for a comma is "%2C".

Explanation:

- **Character:** Comma (,)
- **URL Encoding:** "%2C"

Key points about URL encoding:

- It replaces special characters in a URL with a percentage sign (%) followed by two hexadecimal digits representing the character's ASCII code.
- This ensures that the URL can be correctly interpreted by web servers.

From <https://www.google.com/search?q=url+encoding+of+comma&rlz=1C1UEAD_en-GBIN1121IN1121&oq=url+encoding+of+comma&gs_lcrp=EgZjaHJvbWUyBggAEEUYOdIBCTk5MjJqMGoxNagCCLACAQ&sourceid=chrome&ie=UTF-8>

```
import express from 'express';
import morgan from 'morgan';
//cerate a new express app/server object
const app =express();
app.use(morgan('combined'));
app.use(express.json());
app.use(express.json());
app.use(express.text());
//app.use(express.urlencoded());
function mid1(req,res,next){
    console.log('mid1');
    next();
}
function mid2(req,res,next){
    console.log('mid2');
    next();
}
function mid3(req,res,next){
    console.log('mid3');
    next();
}
app.use(commonMiddleware);
function commonMiddleware(req,res,next){
    console.log('commonMiddleware');
    next();
}

const middlewares =[mid1,mid2,mid3];
app.get('/ping' ,middlewares,(req,res)=>{
    console.log("query params: ",req.query);
    console.log("req body: ",req.body);
    return res.json({
        message : 'pong'
    });
});
app.post('/hello',[mid1,mid3] ,(req,res)=>{
    return res.json({
        message : 'world'
    });
});

app.get('/tweets/:tweet_id/comments/:comment_id',(req,res)=>{
    console.log(req.params);
    return res.json({
        message:'tweet details'
    });
});
//defining a PORT and attach it to the express app
app.listen(3000 ,() =>{
    console.log("server is running on port 3000");
})
```