NODE PROGRAM

EXPRESS.JS

NODE.JS VERSION: 5.1
LAST UPDATED: JAN 2016

EXPRESS

EXPRESS IS THE MOST POPULAR WEB APPLICATION FRAMEWORK FOR NODE

IT IS EASY TO WORK WITH AS IT TIES INTO NODE'S FUNCTIONAL PARADIGM

- DELIVER STATIC CONTENT (OR CONSIDER USING NGINX)
- MODULARIZE BUSINESS LOGIC

INSTALLING DEPENDENCY

```
$ npm install express --save
```

INSTALLING SCAFFOLDING

INSTALL EXPRESS.JS COMMAND-LINE GENERATOR:

```
$ npm install -g express-generator
```

USING THE GENERATOR

```
$ express todo-list-app
```

- `app.js`: MAIN FILE, HOUSES THE EMBEDDED SERVER AND APPLICATION LOGIC
- `public/`: CONTAINS STATIC FILES TO BE SERVED BY THE EMBEDDED SERVER
- `routes/`: HOUSES CUSTOM ROUTING FOR THE EMBEDDED

CONFIGURING EXPRESS

THE EXPRESS SERVER NEEDS TO BE CONFIGURED BEFORE IT CAN START

MANAGE CONFIGURATION VIA THE `set` METHOD:

```
var app = express();
app.set('port', process.env.PORT || 3000);
app.set('views', 'views'); // the directory the templates are stored in
app.set('view engine', 'jade');
```

NODE.JS MIDDLEWARE PATTERN

WHAT IS MIDDLEWARE

MIDDLEWARE PATTERN IS A SERIES OF PROCESSING UNITS CONNECTED TOGETHER, WHERE THE OUTPUT OF ONE UNIT IS THE INPUT FOR THE NEXT ONE. IN NODE.JS, THIS OFTEN MEANS A SERIES OF FUNCTIONS IN THE FORM:

```
function(args, next) {
  next(output) // error or real output
}
```

CONNECT MIDDLEWARE

EXAMPLE:

```
app.use(function middleware1(req, res, next) {
  // middleware 1
  next();
});
app.use(function middleware2(req, res, next) {
  // middleware 2
  next();
});
```

MIDDLEWARE ORDER

MIDDLEWARE ARE EXECUTED IN THE ORDER SPECIFIED:

```
app.use(express.logger('dev'));
app.use(express.basicAuth('test', 'pass'));
app.use(express.json());
```

CREATING MIDDLEWARE

CUSTOM MIDDLEWARE IS EASY TO CREATE:

```
app.use(function (req, res, next) {
  // modify req or res
  // execute the callback when done
  next();
});
```

CONNECT FRAMEWORK

EXPRESS LEVERAGES THE CONNECT FRAMEWORK TO PROVIDE MIDDLEWARE
FUNCTIONALITY.

MIDDLEWARES ARE USED TO MANAGE HOW A REQUEST SHOULD BE HANDLED.

MOST POPULAR AND USEFUL CONNECT/EXPRESS MIDDLEWARE

```
$ npm install <package_name> --save
```

- BODY-PARSER REQUEST PAYLOAD
- COMPRESSION GZIP
- CONNECT-TIMEOUT SET REQUEST TIMEOUT
- COOKIE-PARSER COOKIES
- COOKIE-SESSION SESSION VIA COOKIES STORE

CONNECT/EXPRESS MIDDLEWARE

- CSURF CSRF
- ERRORHANDLER ERROR HANDLER
- EXPRESS-SESSION SESSION VIA IN-MEMORY OR OTHER STORE
- METHOD-OVERRIDE HTTP METHOD OVERRIDE
- MORGAN SERVER LOGS
- RESPONSE-TIME

CONNECT/EXPRESS MIDDLEWARE

- SERVE-FAVICON FAVICON
- SERVE-INDEX
- SERVE-STATIC STATIC CONTENT
- VHOST

OTHER POPULAR MIDDLEWARE

- COOKIES AND KEYGRIP: ANALOGOUS TO `cookieParser`
- RAW-BODY
- CONNECT-MULTIPARTY, CONNECT-BUSBOY
- QS: ANALOGOUS TO `query`
- ST, CONNECT-STATIC ANALOGOUS TO `staticCache`

OTHER POPULAR MIDDLEWARE

- EXPRESS-VALIDATOR: VALIDATION
- LESS: LESS CSS
- PASSPORT: AUTHENTICATION LIBRARY
- HELMET: SECURITY HEADERS
- CONNECT-CORS: CORS
- CONNECT-REDIS

TEMPLATE ENGINE

SETTING THE `view engine` VARIABLE TO `jade` FOR INSTANCE, WOULD TRIGGER THE FOLLOWING FUNCTION CALL INTERNALLY

```
app.set('view engine', 'jade'); // shorthand

// does the same as the above
app.engine('jade', require('jade').__express);
```

TEMPLATE ENGINE

CUSTOM CALLBACKS CAN BE DEFINED TO PARSE TEMPLATES

```
app.engine([format], function (path, options, callback) {
  // template parsing logic goes here
});
```

NOTE: CUSTOM CALLBACKS ARE USEFUL IF THE TEMPLATE ENGINE DOESN'T EXPORT AN __EXPRESS FUNCTION

RUNNING EXPRESS

```
var http = require('http'),
    express = require('express');

var app = express();

// ...

var server = http.createServer(app);
server.listen(app.get('port'), function () {
  // Do something... maybe log some info?
});
```

DEMO

🖥️

RESTFUL API

[HTTPS://GITHUB.COM/AZAT-CO/REST-API-EXPRESS](https://github.com/azat-co/rest-api-express)

ALTERNATIVES

- SAILS
- LOOPBACK 👈
- METEOR
- HAPI
- RESTIFY

REGISTRY OF HAND-PICKED NODE FRAMEWORKS:

QUESTIONS AND EXERCISES

👍

WORKSHOP

🔨

```
$ npm i -g expressworks
$ npm i -g meanworks
```