

# Writing Express Middleware

*by* Robert Damphousse



<https://github.com/robertjd>

@robertjd\_

<https://stormpath.com/>

@goStormpath

# **This Talk Here:**

<https://github.com/robertjd/writing-express-middleware>

## What is middleware?

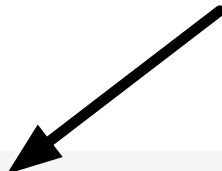
- They're just functions
- Let you modify the HTTP request or response
- They're called "filters" in other frameworks

This is middleware..

```
function logger(req,res,next){  
  console.log(new Date(), req.method, req.url);  
  next();  
}
```

This is middleware..

This is the function  
signature of ALL  
middleware functions



```
function logger (req, res, next) {  
  console.log(new Date(), req.method, req.url);  
  next();  
}
```

# This is middleware too!

<https://github.com/expressjs/serve-favicon>

```
function favicon(req, res, next){
  if (parseUrl(req).pathname !== '/favicon.ico') {
    next();
    return;
  }

  if ('GET' !== req.method && 'HEAD' !== req.method) {
    var status = 'OPTIONS' === req.method ? 200 : 405;
    res.writeHead(status, {'Allow': 'GET, HEAD, OPTIONS'});
    res.end();
    return;
  }

  if (icon) return send(req, res, icon);

  fs.readFile(path, function(err, buf){
    if (err) return next(err);
    icon = createIcon(buf, maxAge);
    send(req, res, icon);
  });
};
```

It's a function  
that does  
something  
awesome for  
your  
application!

Let's build one


We want to say hello, good bye, and  
log the requests



# Create a basic Express app

```
var express = require('express');  
  
var app = express();  
  
var server = app.listen(3000);
```

# Write our logger middleware

```
var express = require('express');  
  
var app = express();  
  
function logger(req, res, next) {  
  console.log(new Date(), req.method, req.url);  
  next();  
}  
  
var server = app.listen(3000);
```

# Attach it to the app


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

var app = express();

function logger(req, res, next) {
  console.log(new Date(), req.method, req.url);
  next();
}

app.use(logger);

var server = app.listen(3000);
```



# What happens??

CURL Shell:

```
$ curl http://localhost:3000/  
Cannot GET /  
$
```

Server shell:

```
Mon Mar 23 2015 11:05:04 GMT-0700 (PDT) 'GET' '/'
```


Why “Cannot GET /” ??

Cuz we haven’t defined any endpoints,  
we’re just calling next() and then Express  
says “nope.. nothing else to do!”

# Say Hello (write more middleware!)

...

```
function logger(req,res,next){  
  console.log(new Date(), req.method, req.url);  
  next();  
}
```



```
function hello(req,res,next){  
  res.write('Hello \n');  
  next();  
}
```

```
app.use(logger);
```



```
app.get('/hello',hello);
```

```
var server = app.listen(3000);
```

# What happens??

CURL Shell:

```
$ curl http://localhost:3000/hello  
Hello
```



Server shell:

```
Mon Mar 23 2015 11:23:59 GMT-0700 (PDT) 'GET' '/hello'
```

# What happens??

CURL Shell:

```
$ curl http://localhost:3000/hello
```

```
Hello
```



WTF, mate? Gimme back my shell!

Server shell:

```
Mon Mar 23 2015 11:23:59 GMT-0700 (PDT) 'GET' '/hello'
```




We need to end the response

# Say Goodbye (write more middleware!)

...

```
function hello(req,res,next){  
  res.write('Hello \n');  
  next();  
}
```



```
function bye(req,res,next){  
  res.write('Bye \n');  
  res.end();  
}
```

Aha!

```
app.use(logger);
```

```
app.get('/hello',hello,bye);
```

```
var server = app.listen(3000);
```

# What happens?? #winning

CURL Shell:

```
$ curl http://localhost:3000/hello
Hello
Bye
$
```

Server shell:

```
Mon Mar 23 2015 11:23:59 GMT-0700 (PDT) 'GET' '/hello'
```

## Other Use Cases

# Use middleware on a specific route

```
app.use(logger); No more!  
  
app.get('/hello', logger, hello, bye);  
  
app.get('/wassaaa', hello, bye);  
  
var server = app.listen(3000);
```

# What happens??

CURL Shell:

```
$ curl http://localhost:3000/hello
Hello
Bye
$ curl http://localhost:3000/wassaaa
Hello
Bye
$
```

Server shell:

```
Mon Mar 23 2015 11:23:59 GMT-0700 (PDT) 'GET' '/hello'
```

# Use middleware with routers

```
function logger(req,res,next){  
  console.log(new Date(), req.method, req.url);  
  next();  
}  
  
function hello(req,res,next){  
  res.write('Hello \n');  
  next();  
}  
  
function bye(req,res,next){  
  res.write('Bye \n');  
  res.end();  
}
```

```
var app = express();  
  
var apiRouter = express.Router();  
  
apiRouter.use(logger);  
  
app.use(hello,bye);  
  
app.use('/api',apiRouter);
```

# What happens??

CURL Shell:

```
$ curl http://localhost:3000/  
Hello  
Bye  
$ curl http://localhost:3000/api/hello  
Hello  
Bye  
$
```

Server shell:

*The sound.. of silence*

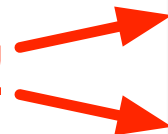


# Use middleware with routers, order matters!

```
function logger(req,res,next){  
  console.log(new Date(), req.method, req.url);  
  next();  
}  
  
function hello(req,res,next){  
  res.write('Hello \n');  
  next();  
}  
  
function bye(req,res,next){  
  res.write('Bye \n');  
  res.end();  
}
```

```
var app = express();  
  
var apiRouter = express.Router();  
  
apiRouter.use(logger);  
  
app.use('/api',apiRouter);  
  
app.use(hello,bye);
```

Swap!



# What happens??

CURL Shell:

```
$ curl http://localhost:3000/  
Hello  
Bye  
$ curl http://localhost:3000/api/hello  
Hello  
Bye  
$
```

Server shell:

```
Mon Mar 23 2015 14:48:59 GMT-0700 (PDT) 'GET' '/hello'
```

# What happens??

CURL Shell:

```
$ curl http://localhost:3000/  
Hello  
Bye  
$ curl http://localhost:3000/api/hello  
Hello  
Bye  
$
```

Server shell:

```
Mon Mar 23 2015 14:48:59 GMT-0700 (PDT) 'GET' '/hello'
```

Hmm?



So much middleware! Here are some cool ones you should know about..

# morgan

<https://github.com/expressjs/morgan>

HTTP request logger middleware for node.js

```
var express = require('express')
var fs = require('fs')
var morgan = require('morgan')

var app = express()

var fileName = __dirname + '/access.log', {flags: 'a'}

// create a write stream (in append mode)
var accessLogStream = fs.createWriteStream(fileName);

// setup the logger
app.use(morgan('combined', {stream: accessLogStream}))

app.get('/', function (req, res) {
  res.send('hello, world!')
})
```

# node-client-sessions <https://github.com/mozilla/node-client-sessions>

Implements sessions in encrypted tamper-free cookies.

```
app.use(sessions({
  cookieName: 'mySession',
  secret: 'blargadeeblargblarg',
  duration: 24 * 60 * 60 * 1000,
  cookie: {
    path: '/api',
    maxAge: 60000,
    ephemeral: false,
    httpOnly: true,
    secure: true
  }
}));
```

Lots of awesome options- do all the cookie things!

# helmet <https://github.com/helmetjs/helmet>

Helmet is really just a collection of 10 smaller middleware functions:

- `crossdomain` for serving `crossdomain.xml`
- `contentSecurityPolicy` for setting Content Security Policy
- `hidePoweredBy` to remove the X-Powered-By header
- `hpkp` for HTTP Public Key Pinning
- `hsts` for HTTP Strict Transport Security
- `ieNoOpen` sets X-Download-Options for IE8+
- `noCache` to disable client-side caching
- `noSniff` to keep clients from sniffing the MIME type
- `frameguard` to prevent clickjacking
- `xssFilter` adds some small XSS protections

You should  
use this!

# stormpath-express

(\*shameless self promo)

<https://github.com/stormpath/stormpath-express>

Showing you everything that an Express middleware module can be!

- Create user accounts.
- Edit user accounts.
- Store user data with each account.
- Create groups and roles.
- Assign users various permissions (groups, roles, etc.).
- Handle complex authentication and authorization patterns.
- Log users in via social login with Facebook and Google OAuth.
- Cache user information for quick access.
- Scale your application as you get more users.
- Securely store your users and user data in a central location.



**stormpath-sdk-angularjs**

(\*shameless self promo)

<https://github.com/stormpath/stormpath-sdk-angularjs>



(enough said)

The End

Thanks For Coming!