Glen R. Goodwin

@areinet

arei.net

github.com/arei

Glen R. Goodwin

@areinet

arei.net

github.com/arei

BUILDING AN AWESOME ENTERPRISE API SERVER

THE
AWESOME
ENGINEERING
COMPANY

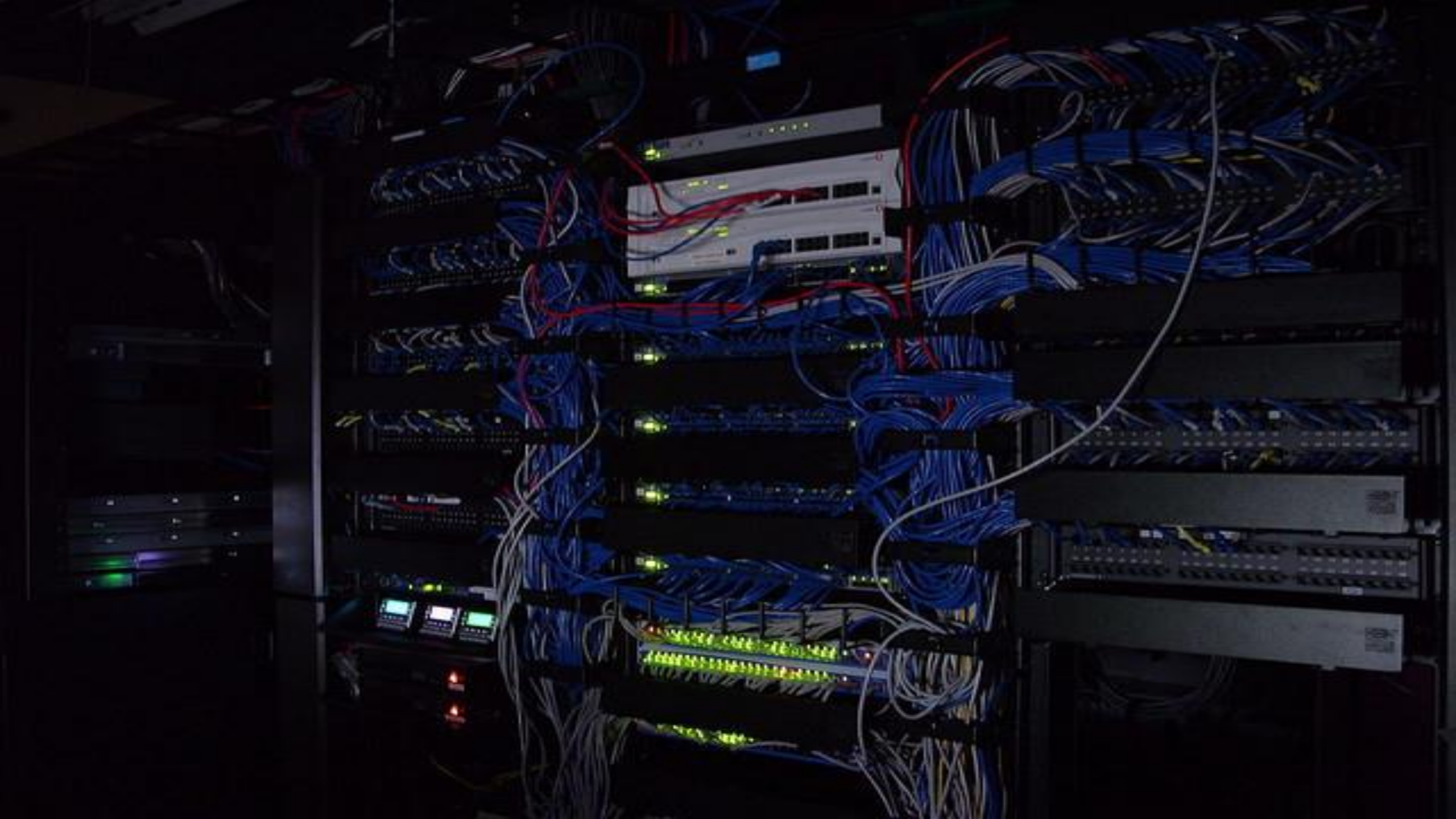
AWESOME LOG

AWESOME

CONFIG

AWESOME

SERVER



- Stable
- Secure
- Auditable
- Configurable
- Performant
- Scalable

- Stable
- Secure
- Auditable
- Configurable
- Performant
- Scalable

AWESOME LOG

```
> npm install @awesomeeng/awesome-log
+ @awesomeeng/awesome-log@3.0.1
added 2 packages from 2 contributors and audited 2 packages in 3.47s
found 0 vulnerabilities
```

```
const Log = require("@awesomeeng/awesome-log");
```

```
const Log = require("@awesomeeng/awesome-log");

Log.init();
```

```
const Log = require("@awesomeeng/awesome-log");
Log.init();

Log.start();
```

```
const Log = require("@awesomeeng/awesome-log");
Log.init();
Log.start();

Log.info("Hello there world!");

Log.error("Something terribly awful happened here.");

Log.debug("All work and no play…");
```

```
// our main application entry point
const Log = require("@awesomeeng/awesome-log");
Log.init();
Log.start();
Log.info("Log started at top of application!");

// some other module used later
const Log = require("@awesomeeng/awesome-log");
Log.warn("Logged from somewhere else!");
```

```
const Log = require("@awesomeeng/awesome-log");

Log.init({
    levels: "access,error,warn,info,debug,silly"
});

Log.start();
```

```
Log.init({
    writers: [{
        name: "MyConsoleWriter",
        type: "console",
        formatter: "csv",
    }],
});

Log.init({
    writers: [{
        name: "MyConsoleWriter",
        type: "console",
        formatter: "json"
    }],
});
```

```
Log.init({
    writers: [{
        name: "MyFileWriter",
        type: "file",
        formatter: "json",
        options: {
            filename: "logs/MyLogs.{YYYYMMDD}.log"
        }
    }],
});
```

```
Log.defineFormatter("my-custom-formatter","./My-Custom-Formatter.js");
Log.defineWriter("my-custom-writer","./My-Custom-Writer.js");

Log.init({
    writers: [{
        name: "MyCustomLogs",
        type: "my-custom-writer",
        formatter: "my-custom-formatter"
    }],
});
```

AwesomeLog Key Features

- Ready to go with zero initial configuration;
- Configure and Start once, Use Everywhere;
- Support for clean nested AwesomeLog usage;
- Customizable Log Levels;
- Configurable log entry field contents;
- Built-In Formatters: Default, JSON, JS, or CSV;
- Or add your own log formatters;
- Console and File writers;
- or add your own custom log writer;
- Colorized Console Logging;
- Log History;
- Pause/Resume;
- SubProcess Logging;

npm repo @awesomeeng/awesome-log

https://github.com/awesomeeng/awesome-log

AWESOME

CONFIG

```
> npm install @awesomeeng/awesome-config
+ @awesomeeng/awesome-config@1.0.0
added 3 packages from 2 contributors and audited 4 packages in 2.835s
found 0 vulnerabilities
```

```
const config = require("@awesomeeng/awesome-config");
```

```
const config = require("@awesomeeng/awesome-config");

config().init();
```

```
const config = require("@awesomeeng/awesome-config");
config().init();

config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});

config().start();
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();

let user = config.user.name;
console.log("Greetings "+name+" from "+config.user.location);
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();

let user = config.user.name;
console.log("Greetings "+name+" from "+config.user.location);
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();

let user = config.user.name;
console.log("Greetings "+name+" from "+config.user.location);
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();

config.user.name = “Sal”;
// Mutation Error

config().add("config.user.name = Sal");
// Mutation Error
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();
```

```
{
    "user": {
        "name": "Glen",
        "location": "Columbia, MD, USA"
    }
}
```

```
const config = require("@awesomeeng/awesome-config");
config().init();
config().add("./AwesomeConfig1.json");
config().add({
    user: {
        name: "Bob"
    }
});
config().start();
```

```
{
    "user": {
        "name": "Glen",
        "location": "Columbia, MD, USA"
    }
}
```

```
{
    user: {
        name: "Bob",
        location: "Columbia, MD, USA"
    }
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

```
/* API Server configuration */
{
    "server": {
        "scheme": "http",
        "hostname": "localhost",
        "port": 1234,
        "url": "${server.scheme}://${server.hostname}:${server.port}"
    }
}

[env:target=production]
server.scheme = "https"
server: {
    "hostname": "0.0.0.0",
    "port": 443
}
```

AwesomeConfig Key Features

- Add configuration from files, JSON, or javascript objects;
- Uses JSON notation or our custom notation;
- Globally accessible config, no passing config objects around;
- Configuration exposed as a plain JavaScript object;
- Namespace instances to isolate usage as needed;
- Configuration is immutable once started;
- Configuration Variables;
- Configuration Conditions;
- Configuration Placeholders;
- No reserved words.

npm repo @awesomeeng/awesome-config

https://github.com/awesomeeng/awesome-config

AWESOME

SERVER

```
> npm install @awesomeeng/awesome-server
+ @awesomeeng/awesome-server@1.2.1
added 1 package from 2 contributors and audited 9 packages in 1.533s
found 0 vulnerabilities
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

let server = new AwesomeServer();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});

await server.start();
```

```
> curl http://localhost:8080/test
Hello World!
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});
await server.start();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});

await server.start();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});
await server.start();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();

server.addHTTPServer({
    hostname: "localhost",
    port: 80
});
server.addHTTPSServer({
    hostname: "localhost",
    port: 443
});

server.route("GET","/test",(path,request,response)=>{
    return response.writeHTML("Hello World!");
});
await server.start();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",handler);

await server.start();
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/te

await server.start();
```

Specific Method: GET, POST, PUT, DELETE, HEAD, OPTIONS, CONNECT, TRACE, PATCH

Wildcard: *

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",har

await server.start();
```

Exact: "/test"
Starts With: "/test*"
Ends With: "*/test"
Contains: "*test*"
RegEx: "^/^\/[test]+$/"
Multiples: "/test|/test/*"

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/test",handler);

await server.start();
```

Function: (path,request,response)=>{ … }

Controller: class MyController extends AwesomeServer.AbstractController { … }

Filename: "./MyController.js“

Directory: "./MyControllers

```
const AwesomeServer = require("@awesomeeng/awesome-server");
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: "localhost",
    port: 80
});

server.route("GET","/users","./UsersController.js");
server.route("GET","/user","./UserController.js");
server.route("GET","/groups","./GroupsController.js");
server.route("GET","/group","./GroupController.js");
server.route("GET","/keys","./KeysController.js");
server.route("GET","/key","./KeyController.js");

await server.start();
```

AwesomeServer Key Features

- Easy to use;
- HTTP support;
- HTTPS support;
- HTTP/2 support including push routing for preloading;
- Or mix and match all three types of servers;
- Basic routing of Method X path Y into handler Z function;
- Advanced routing using Controllers;
- Controllers from classes, files, or whole directory trees;
- Support for serving static files or whole directories;
- Easy built-in redirects;
- async/await ready;
- Add your own custom servers beyond HTTP, HTTPS, or HTTP/2.

npm repo @awesomeeng/awesome-server

https://github.com/awesomeeng/awesome-server

An AWESOME KITTEH SERVER

- GET /kittens - return a list of all kitten ids
- GET /kittens/<id> - return a specific kitten image.
- Log of each request
- Local vs production configuration

> **npm install @awesomeeng/awesome-log**

+ @awesomeeng/awesome-log@3.0.1
added 2 packages from 2 contributors and audited 2 packages in 3.47s
found 0 vulnerabilities

> **npm install @awesomeeng/awesome-config**

+ @awesomeeng/awesome-config@1.0.0
added 3 packages from 2 contributors and audited 4 packages in 2.835s
found 0 vulnerabilities

> **npm install @awesomeeng/awesome-server**

+ @awesomeeng/awesome-server@1.2.1
added 1 package from 2 contributors and audited 9 packages in 1.533s
found 0 vulnerabilities

```
const AwesomeServer = require("@awesomeeng/awesome-server");

class KittenController extends AwesomeServer.AbstractController {
    get(path,request,response) {
    }
}
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

class KittenController extends AwesomeServer.AbstractController {
    get(path,request,response) {
        if (path==="") return getKittenIds(path,request,response);
        else return getSpecificKitten(path,request,response);
    }
}
```

```
const getKittenIds = function(path,request,response) {
    return new Promise((resolve,reject)=>{
        try {
            path = Path.resolve("./images")
            FS.readdir(path,async (err,files)=>{
                if (err) return await response.writeError(500,err);

                files = files.filter((filename)=>{
                    return filename.endsWith(".jpg");
                });
                files = files.map((filename)=>{
                    return Path.basename(filename,".jpg");
                });
                await response.writeJSON(files);

                resolve();
            });
        }
        catch (ex) {
            return reject(ex);
        }
    });
};
```

```
const getSpecificKitten = function(id,request,response) {
    return response.serve(200,"image/jpeg",Path.resolve("./"+id+".jpg"));
};
```

```
const getSpecificKitten = function(id,request,response) {
    return response.serve(200,"image/jpeg",Path.resolve("./"+id+".jpg"));
};
```

```
const getSpecificKitten = function(id,request,response) {
    return response.serve(200,"image/jpeg",Path.resolve("./"+id+".jpg"));
};
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");

class KittenController extends AwesomeServer.AbstractController {
    get(path,request,response) {
        if (path==="") return getKittenIds(path,request,response);
        else return getSpecificKitten(path,request,response);
    }
}

const getKittenIds = function(path,request,response) { ... };

const getSpecificKitten = function(id,request,response) { ... };

module.exports = KittenController;
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
const Log = require("@awesomeeng/awesome-log");
const config = require("@awesomeeng/awesome-config");
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
const Log = require("@awesomeeng/awesome-log");
const config = require("@awesomeeng/awesome-config");

Log.init();
Log.start();
Log.debug("Log started.");
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
const Log = require("@awesomeeng/awesome-log");
const config = require("@awesomeeng/awesome-config");

Log.init();
Log.start();
Log.debug("Log started.");

config().init();
config().add("./config.json");
config().start();
Log.debug("Config started.");
```

```
... requires ...
... logging ...
... config ...
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: config.server.hostname,
    port: config.server.port
});
server.route("*","*",(path,request,response)=>{
    Log.access("Request from "
        +request.origin+" to "
        +request.url.href+".");
});
server.route("*","/kittens|/kittens/*","./KittenController.js");
server.start();
Log.debug("Server started.");
```

```
... requires ...
... logging ...
... config ...
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: config.server.hostname,
    port: config.server.port
});
server.route("*","*",(path,request,response)=>{
    Log.access("Request from "
        +request.origin+" to "
        +request.url.href+".");
});
server.route("*","/kittens|/kittens/*","./KittenController.js");
server.start();
Log.debug("Server started.");
```

```
... requires ...
... logging ...
... config ...
let server = new AwesomeServer();
server.addHTTPServer({
    hostname: config.server.hostname,
    port: config.server.port
});
server.route("*","*",(path,request,response)=>{
    Log.access("Request from "
        +request.origin+" to "
        +request.url.href+".");
});
server.route("*","/kittens | /kittens/*","./KittenController.js");
server.start();
Log.debug("Server started.");
```

```
... requires ...

... log stuff ...

Log.start();

... config stuff ...

... server stuff ...

server.start();

...
```

```
... requires ...

... log stuff ...

await Log.start();

... config stuff ...

... server stuff ...

await server.start();

...
```

```
... requires ...

(async ()=>{
    ... log stuff ...

    await Log.start();

    ... config stuff ...

    ... server stuff ...

    await server.start();

    ...
})();
```

```
/* Our base, which is also our local server configuration */
{
	"server": {
		"scheme": "http",
		"hostname": "localhost",
		"port": 8080,
		"url": "${server.scheme}://${server.hostname}:${server.port}"
	}
}

/* our production environment settings */
[env:target=production]
{
	"server": {
		"hostname": "0.0.0.0",
		"port": 80
	}
}
```

```
/* Our base, which is also our local server configuration */
{
	"server": {
		"scheme": "http",
		"hostname": "localhost",
		"port": 8080,
		"url": "${server.scheme}://${server.hostname}:${server.port}"
	}
}

/* our production environment settings */
[env:target=production]
{
	"server": {
		"hostname": "0.0.0.0",
		"port": 80
	}
}
```

```
/* Our base, which is also our local server configuration */
{
	"server": {
		"scheme": "http",
		"hostname": "localhost",
		"port": 8080,
		"url": "${server.scheme}://${server.hostname}:${server.port}"
	}
}

/* our production environment settings */
[env:target=production]
{
	"server": {
		"hostname": "0.0.0.0",
		"port": 80
	}
}
```

```
const AwesomeServer = require("@awesomeeng/awesome-server");
const Log = require("@awesomeeng/awesome-log");

class KittenController extends AwesomeServer.AbstractController {
	get(path,request,response) {
		if (path==="") return getKittenIds(path,request,response);
		else return getSpecificKitten(path,request,response);
	}
}

const getKittenIds = function(path,request,response) {
	Log.debug("Received request to return Kitten Ids");
	...
};

const getSpecificKitten = function(id,request,response) {
	Log.debug("Received request to return specific Kitten "+id+".");
	...
};

module.exports = KittenController;
```

DEMO

npm repo @awesomeeng/awesome-log
npm repo @awesomeeng/awesome-config
npm repo @awesomeeng/awesome-server

github.com/awesomeeng
npmjs.com/org/awesomeeng

Glen R. Goodwin
Awesome Engineering