MONO

terrajs/mono

Sébastien CHOPIN

Co-author of Nuxt.js & MONO

Atinux @Atinux

ORION.sh

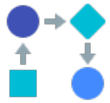
WEB API FRAMEWORK

Based on Express

terrajs/mono

WHY EXPRESS

FAST



MIDDLEWARE

MAINTAINED

PRODUCTION READY

WHY MONO

Give structure for developers

Directory structure & filenames to follow: the rest is up to you.

Enforce known best practices

Node.js best practices: github.com/i0natan/nodebestpractices

Focus on business logic

Included & Configured: Helmet • Body parser • Nodemon • Winston • Joi

MONO FEATURES

 Environment-based configuration

MONO FEATURES

 Environment-based configuration

 API Versioning

MONO FEATURES

 Environment-based configuration

 API Versioning

 Sessions with JWT

MONO FEATURES

 Environment-based configuration

 API Versioning

 Sessions with JWT

 Integrated ACL with IMPERIUM

 terrajs/mono

MONO FEATURES

 Environment-based configuration

 API Versioning

 Sessions with JWT

 Integrated ACL with IMPERIUM

 Routes validation with Joi

 terrajs/mono

MONO FEATURES

 Environment-based configuration

 API Versioning

 Sessions with JWT

 Integrated ACL with IMPERIUM

 Routes validation with Joi

 Logs with winston & morgan

 terrajs/mono

MONO FEATURES

- Environment-based configuration
- API Versioning
- Sessions with JWT
- Integrated ACL with IMPERIUM
- Routes validation with Joi
- Logs with winston & morgan

Automatic “init" methods discovery

MONO FEATURES

 Environment-based configuration

 API Versioning

 Sessions with JWT

 Integrated ACL with IMPERIUM

 Routes validation with Joi

 Logs with winston & morgan

 Automatic “init" methods discovery

 Automatic routes discovery

 terrajs/mono

MONO FEATURES

- Environment-based configuration
- API Versioning
- Sessions with JWT
- Integrated ACL with IMPERIUM
- Routes validation with Joi
- Logs with winston & morgan
- Automatic “init" methods discovery
- Automatic routes discovery

API testing

terrajs/mono

MONO FEATURES

- Environment-based configuration
- API Versioning
- Sessions with JWT
- Integrated ACL with IMPERIUM
- Routes validation with Joi
- Logs with winston & morgan
- Automatic “init" methods discovery
- Automatic routes discovery
- API testing

Extend with modules

terrajs/mono

INSTALLATION

npm install --save @terrajs/mono

package.json

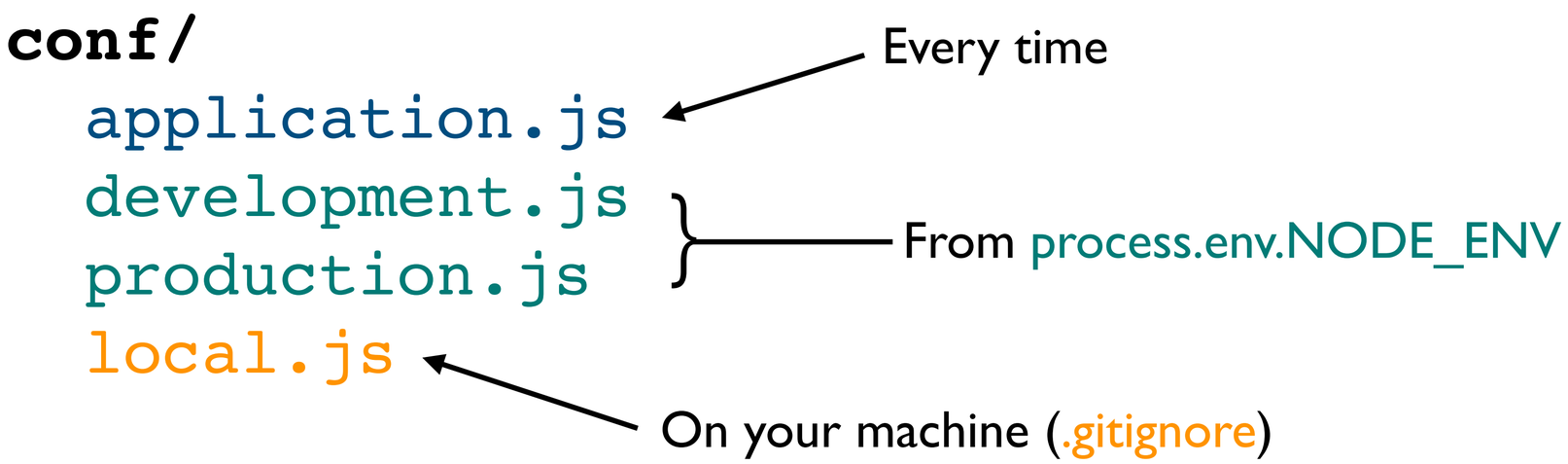
```
//...
"scripts": {
  "dev": "mono dev",
  "start": "mono"
}
//...
```

Use nodemon to watch & restart on file changes

DIRECTORY STRUCTURE

```
conf/
src/
test/
package.json
```

Environment-based configuration



Environment-based configuration

```
// conf/application.js
module.exports = {
  mono: {
    http: {
      port: 6000
    }
  }
}
```

🔭 Automatic routes discovery

Load all **src/**/*.routes.js** files

Automatic routes discovery

```
// src/users/users.routes.js
const validation = require('./users.validation')
const controller = require('./users.controller')

module.exports = [
  {
    method: 'GET',
    path: '/user',
    session: true,
    handler: controller.getUser
  },
  {
    method: 'PUT',
    path: '/user/hero',
    session: true,
    validation: validation.chooseHero,
    handler: controller.chooseHero
  }
]
```

```
// src/heroes/heroes.routes.js
const controller = require('./heroes.controller')

module.exports = [
  {
    method: 'GET',
    path: '/heroes',
    handler: controller.listHeroes
  }
]
```

- GET • /user
- PUT • /user/hero
- GET • /heroes

 Routes validation with Joi

```
// src/users/users.routes.js
const validation = require('./users.validation')
const controller = require('./users.controller')

module.exports = [
  // ...
  {
    method: 'PUT',
    path: '/user/hero',
    session: true,
    validation: validation.chooseHero,
    handler: controller.chooseHero
  }
]
```

```
// src/users/users.validation.js
const Joi = require('joi')

exports.chooseHero = {
    body: Joi.object().keys({
        hero: Joi.string().required()
    })
}
```

Routes validation with Joi

```
// src/users/users.validation.js
const Joi = require('joi')

exports.chooseHero = {
    body: Joi.object().keys({
        hero: Joi.string().required()
    })
}
```

```
$ curl -X PUT 'http://localhost:8000/user/hero' -H 'authorization: Bearer ...'
HTTP/1.1 400 Bad Request
{
  "code":"validation-error",
  "status":400,
  "context":[
    {
      "field": ["hero"],
      "location":"body",
      "messages":["\"hero\" is required"],
      "types"["any.required"]
    }
  ]
}
```

Sessions with JWT

```
// src/users/users.routes.js
// ...
const controller = require('./users.controller')

module.exports = [
  {
    method: 'GET',
    path: '/user',
    session: true,
    handler: controller.getUser
  }
  // ...
]
```

```
$ curl http://localhost:8000/user
HTTP/1.1 401 Unauthorized
{"code":"credentials-required","status":401,"context":{}}
```

```
$ curl http://localhost:8000/user -H "Authorization: Bearer eyJhbGc..."
HTTP/1.1 200 OK
{
  "_id":"5ad7c83f7e0e0222f9411e08",
  "username":"Atinux",
  "name":"Sébastien Chopin",
  "..."
}
```

Sessions with JWT

```
// src/users/users.routes.js
// ...
const controller = require('./users.controller')

module.exports = [
  {
    method: 'GET',
    path: '/user',
    session: true,
    handler: controller.getUser
  }
  // ...
]
```

```
// src/users/users.controller.js
const { HttpError } = require('@terrajs/mono')
const Users = require('./users.service')

exports.getUser = async (req, res) => {
  const user = await Users.getById(req.session.userId)
  if (!user) throw new HttpError('user-not-found', 404)

  res.json(user)
}
//...
```

Sessions with JWT

```
// src/github/github.controller.js
const { jwt } = require('@terrajs/mono')

exports.authenticate = async (req, res) => {
  // ...
  // Generate jwt token
  const token = await jwt.generateJWT({
    userId: user._id,
    username: user.username,
    githubId: user.githubId
  })

  // Returns jwt
  res.json({ token })
}
```

```
// conf/development.js
module.exports = {
  mono: {
    jwt: {
      secret: 'MY-JWT-SECRET'
    },
    // ...
  }
  // ...
}
```

Integrated ACL with IMPERIUM

Load all **src/**/*.acl.js** files

Integrated ACL with IMPERIUM

```
// src/users/users.acl.js
const { imperium } = require('@terrajs/mono')

const user = imperium.role('user', (req) => {
  return { user: req.session.userId }
})
// '@' means itself
user.can('manageUser', { user: '@' })

const admin = imperium.role('admin', (req) => {
  return req.session.role === 'admin'
})
// '*' means all: admin can manage all users
admin.is('user', { user: '*' })
```

```
// src/users/users.routes.js
module.exports = [
  {
    method: 'PUT',
    path: '/users/:userId',
    can: {
      action: 'manageUser',
      user: ':userId'
    },
    // ...
  }
]
```

Integrated ACL with IMPERIUM

```
// src/users/users.routes.js
module.exports = [
  {
    method: 'PUT',
    path: '/users/:userId',
    can: {
      action: 'manageUser',
      user: ':userId'
    },
    // ...
  }
]
```

/users/ I

```
{ can: 'manageUser', user: '1' }
```

✓ req.session.userId === ‘1’

✗ req.session.userId === ‘2'

```
$ curl -XPUT http://localhost:8000/users/1 -H "Authorization: Bearer ..."
HTTP/1.1 403 Forbidden
{"code":"invalid-perms","status":403,"context":{}}
```

✓ req.session.role === ‘admin’

API Versioning

```
// src/users/users.routes.js
module.exports = [
  {
    version: '*', // default
    method: 'GET',
    path: '/user',
    session: true,
    handler: controller.getUser
  }
]
```

- ✅ /user
- ✅ /v1/user
- ✅ /v15/user

- ❌ /v/user
- ❌ /15/user

```
$ curl http://localhost:8000/v/user
HTTP/1.1 404 Not Found
{"code":"not-found","status":404,"context":{"url":"/v/user"}}
```

API Versioning

```
// src/users/users.routes.js
module.exports = [
  {
    version: ['v1', 'v2'],
    method: 'GET',
    path: '/user',
    session: true,
    handler: controller.getUser
  }
]
```

- ✅ /v1/user
- ✅ /v2/user

- ❌ /user
- ❌ /v3/user

Automatic “init" methods discovery

Load all **src/**/*.init.js** files

Automatic “init" methods discovery

```
// src/users/users.init.js
const { db } = require('mono-mongodb')
const collection = db.collection('users')

module.exports = async () => {
  // Set index unique for usernames
    await collection.createIndex({ username: 1 }, { unique: true })
    await collection.createIndex({ slug: 1 })
}
```

 Logs with winston & morgan

```
const { log } = require('@terrajs/mono')

// Write on stdout
log.verbose('This is a verbose message')
log.debug('This is a debug message')
log.info('This is an information message')
log.warn('Warning, this feature will be removed soon')

// Write on stderr
log.error('An error appened')

// Profiling
log.profile('test')
setTimeout(() => log.profile('test'), 1000)
```

 Logs with winston & morgan

```
2018-05-17T08:34:17.743Z - verbose: [@terrajs/mono] This is a verbose message
2018-05-17T08:34:17.743Z - debug: [@terrajs/mono] This is a debug message
2018-05-17T08:34:17.743Z - info: [@terrajs/mono] This is an information message
2018-05-17T08:34:17.743Z - warn: [@terrajs/mono] Warning, this feature will be removed soon
2018-05-17T08:34:17.743Z - error: [@terrajs/mono] An error appened
2018-05-17T08:34:17.757Z - debug: [@terrajs/mono] Server running on http://localhost:8000
2018-05-17T08:34:17.757Z - info: [@terrajs/mono] Startup durationMs=28
2018-05-17T08:34:18.750Z - info: [@terrajs/mono] test durationMs=1007
```

Logs with winston & morgan

```
// conf/development.js
module.exports = {
  mono: {
    log: {
      level: 'info',
      // Log to console
      console: true,
      // Log to files
      files: [
        {
          filename: '/tmp/logs.txt'
        }
      ],
      // Log to http server
      http: ['...'],
      // Winston transports
      transports: [ ... ]
    }
  }
}
```

API testing

npm install --save-dev ava mono-test-utils

```
// package.json
"scripts": {
  // ...
  "test": "ava --serial --fail-fast test/",
},
```

API testing

```
// test/index.js
const test = require('ava')
const { start, stop, $get, $post } = require('mono-test-utils')

let ctx
// Start server
test.before('Start Mono app', async (t) => {
    ctx = await start(require('path').join(__dirname, '..'))
})

// ...

// Close server
test.after('Close Mono server', async (t) => {
    await close(ctx.server)
})
```

terrajs/mono

API testing

```
// test/index.js
test('Call GET - /example', async (t) => {
  const { stdout, stderr, statusCode, body } = await $get('/example')

  t.true(stdout[0].includes('GET /example'))
  t.is(stderr.length, 0)
  t.is(statusCode, 200)
  // Imagine that GET - /example returns { hello: 'world' }
  t.deepEqual(body.body, { hello: 'world' })
})
```

terrajs/mono

MONO UTILS

```
const { utils } = require('@terrajs/mono')
// Or
const { ok, cb, waitFor, ... } = require('@terrajs/mono/utils')
```

MONO UTILS

```
ok(promise: Object): Promise

cb(fn: Function, ...args: any[]): Promise

waitFor(ms: number): Promise

waitForEvent(emitter: EventEmitter, eventName: string, timeout: number = -1): Promise<Array>

asyncObject(obj: Object): Promise<Object>

asyncMap(array: Array, fn: Function): Promise<Array>

asyncForEach(array: Array, fn: Function): Promise<void>
```

terrajs/mono

MONO HttpError

```
// Optional, Mono sets global.HttpError
const { HttpError } = require('@terrajs/mono')

new HttpError(message [, statusCode] [, context])
```

MONO HttpError

```
// src/errors.routes.js
module.exports = [
  {
    method: 'GET',
    path: '/error',
    handler(req, res) {
      throw new HttpError('holy-error', 500, {
        room: 3
      })
    }
  }
]
```

```
$ curl http://localhost:8000/error
HTTP/1.1 500 Internal Server Error
{"code":"holy-error","status":500,"context":{"room":3}}
```

terrajs/mono

Extend with modules

A module can :

- Access the project config
- Read / Add routes
- Add Imperium roles
- Give helpers
- Use Mono hooks

MONO MODULES

MONO MONGODB

MONO REDIS

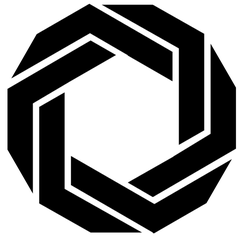
MONO NOTIFICATIONS

MONO ELASTICSEARCH

MONO IO

MONO MAIL

MONO PUSH



MONO DOC

More coming…

terrajs/mono

MONO HOLY

```
mono-holy/
   lib/
      index.js
   package.json
```

terrajs/mono

MONO HOLY

```
mono-holy/
   lib/
      index.js
   package.json
```

```
{
  "name": "mono-holy",
  "description": "Holy module for Mono",
  "version": "0.0.1",
  "main": "lib/"
}
```

MONO HOLY

```
mono-holy/
   lib/
      index.js
   package.json
```

```
module.exports = (context) => {
  // Do some magic here :)
  context.log('😇')
}
```

terrajs/mono

MONO HOLY

context {

log → *Log instance of the application*

conf → *Project configuration*

app → *Express app instance*

server → *Node.js server instance*

hook → *Mono hook instance*

}

MONO MODULE

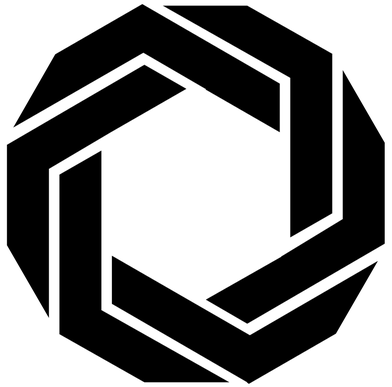
npm install -g create-mono-module

create-mono-module mono-holy

Answer the questions

npm run test

API DOCUMENTATION



MONO DOC

terrajs/mono

DEMO

npm install -g create-mono-app

create-mono-app holy-api

Answer the questions

npm run dev

MONO TEAM

Atinux

benjamincanac

gaetansenn

THANK YOU

Sébastien CHOPIN

Atinux @Atinux

nuxt

cmty

terrajs

terrajs/mono