简介

Express 是一个自身功能极简，完全是由路由和中间件构成一个的 web 开发框架：从本质上来说，一个 Express 应用就是在调用各种中间件。

故expresse 3.x依赖Connect内置了很多中间件，但这些都在4.x 版本移除了，也不依赖Connect，只内置了一个设置静态文件的中间件：express.static，若要使用，须自行添加中间件；这里是在github中比较受欢迎的第三方中间件。

基本使用

内置和第三方中间件的使用方法都一样，大体可分为三个步骤

1. 引入模块
2. 配置/挂载中间件
3. 调用

常用中间件示例

cookie-parser

解析cookie请求头、填充req.cookies与对象加密cookie名称。

```
// 1.引入模块
const cookieParser = require('cookie-parser')

// 2.挂载中间件
app.use(cookieParser())

// 3. 路由调用
app.get('/cookie', (req, res) => {
    if (req.cookies.isVisit) {
```

```
        res.send(req.cookies.isVisit)
    } else {
        res.cookie('isVisit', '欢迎再次访问', {
            maxAge: 5 * 1000,
            httpOnly: true
        })
        res.send('欢迎第一次访问')
    }
})
```

connect-flash and express-session

flash 是 session 中一个用于存储信息的特殊区域。消息写入到 flash 中，在跳转目标页中显示该消息；flash 需配合session一同使用的，以确保消息在目标页面中可用。

```
var flash = require('connect-flash');

app.use(flash());

app.use(function (req, res, next) {
    res.locals.errors = req.flash('error');
    res.locals.infos = req.flash('info');
    next();
});

app.use(session({
    name: config.session.key, // skey
    secret: config.session.secret, // 用来对session id相关的cookie进行签名
    saveUninitialized: false, // 是否自动保存未初始化的会话
    resave: true, // 强制写入session
    cookie: { // 设置cookie有效期
        maxAge: config.session.maxAge
    }
}))



app.get('/flash', (req, res) => {
    var fs = res.flash('error').toString()
    res.send(fs)
```

```
})

app.get('/session', (req, res) => {
    req.session.user = {
        name: 'admin',
        _id: 'efowc9vdjnegv',
        age: 20,
        skill: ['skateboard', 'coding', 'reading', 'talking', 'paly game', 'thinking',
'front-end']
    }
    res.send(req.session.user)
})
```

以上介绍了内置中间的使用，接下来是第三方中间件

express-formidable

处理表单和上传文件

```
const formidable = require('express-formidable')

app.use(require('express-formidable')({
    uploadDir: path.join(__dirname, 'public/img'), // 上传文件目录
    keepExtensions: true// 保留后缀
}))

app.post('/file', (req, req) => {
        let data = {
        name: req.fields.name,
        pwd: req.fields.pwd,
        files: req.files
    }
    res.send(data)
})
```

mongolass

```
// 1.引入模块
const Mongolass = require('mongolass')

// 2.实例化Mongolass
const mongolass = new Mongolass()

// 3. 建立连接
mongolass.connect('mongodb://localhost:27017/myblog')

// 4. 定义模块
const User = mongolass.model('users', {
    name: { type: 'string', required: true },
    age: { type: 'number', default: 18 }
})

// 创建索引
User.index({ name: 1 }, { unique: true }).exec()

// 5. 定义插件
mongolass.plugin('addCreatedAt', {
    afterFind: results => {
        results.forEach(item => {
            item.created_at = moment(objectIdToTimestamp(item._id)).format('YYYY-MM-DD
HH:mm')
        })
        return results
    },
    afterFindOne: result => {
        if (result) {
            result.created_at = moment(objectIdToTimestamp(result._id)).format('YYYY-MM-DD
HH:mm')
        }
        return result
    }
})

// 5. 查询数据库
User.getUserByName = (name) => {
    User.find({name: name})
        .select()
```

```
        .exec()
        .then(function (result) {
            console.log(result)
        })
        .catch(console.error)
}

User.createOne = (name, age) => {
    User.insertOne({ name: name, age: age })
        .exec()
        .then(console.log)
        .catch(console.error)
}

User.createOne('new User')
User.getUserByName('new User')
```