Koa

不仅仅是 generator

何翊宇(aka. 死马)

dead-horse

koa

CNPM

deadhorse_busi

2B码农死小马

异步之殇

```
// error in next tick
try {
  setImmediate(function () {
    throw new Error('error happened');
  });
} catch (err) {
  // 捕获不到异常
}
```

无法捕获异步中的异常

```
// get json by callback
function getJSON(file, callback) {
  fs.readFile(file, function (err, content) {
    if (err) return callback(err);
    try {
      content = JSON.parse(content);
    } catch (err) {
      return callback(err);
    }
    return callback(null, content);
  });
}
```

通过 callback 处理异常

```
// callback hell
getUser(function (err, user) {
  if (err) return callback(err);
  getArticle(user, function (err, article) {
    if (err) return callback(err);
    getComment(article, function (err, comment)) {
      if (err) return comment;
      // res = ...
      callback(null, res);
    });
  });
});
```

callback hell

- 重构代码 callbackhell.com
- async
- promise
- ES7 async function
- ES6 generator and co

```
// get json by co
co(function* getJSON(file) {
    var content = yield fs.readFile(file);
    content = JSON.parse(content);
}).catch(error);

// sync style in co
co(function* () {
  var user = yield getUser();
  var article = yield getArticle(user);
  var comment = yield getComment(article);
  // res = ...
}).catch(error);
```

generator and co

先来认识一下

generator

```
function IteratorFactory(items) {
  var iterator = {
    index: 0,
    max: items.length
  };

  iterator.next = function () {
    return this.index === this.max
      ? { value: undefined, done: true }
      : {value: items[this.index++], done: false};
  }

  return iterator;
};

var iterator = IteratorFactory([1, 2, 3]);
do {
  var ret = iterator.next();
  console.log(ret);
} while (!ret.done);
```

Iterator

```
function* GeneratorFunction(items) {
  var index = 0;
  var max = items.length;
  while (index < max) {
    yield items[index++];
  }
}

var generator = GeneratorFunction([1, 2, 3]);
do {
  var ret = generator.next();
  console.log(ret);
} while (!ret.done);
```

Generator

```
function IteratorFactory(items) {
  var iterator = {
    index: 0,
    max: items.length
  };

  iterator.next = function () {
    return this.index === this.max
      ? { value: undefined, done: true }
      : {value: items[this.index++], done: false};
  }

  return iterator;
}

function run(iterator) {
  var ret = iterator.next();
  if (ret.done) return;
  ret.value(function () {
    run(iterator);
  });
}

var count = 1;
function tick(done) {
  setTimeout(function () {
    console.log('tick %s after %s ms', count++, 1000);
    done();
  }, 1000)
}

run(IteratorFactory([tick, tick, tick]));
```

Iterator with Functions

```
function* GeneratorFunction(items) {
  var index = 0;
  var max = items.length;
  while (index < max) {
    yield items[index++];
  }
}

function run(generator) {
  var ret = generator.next();
  if (ret.done) return;
  ret.value(function () {
    run(generator);
  });
}

var count = 1;
function tick(done) {
  setTimeout(function () {
    console.log('tick %s after %s ms', count++, 1000);
    done();
  }, 1000)
}

run(GeneratorFunction([tick, tick, tick]));
```

Generator with Functions

```
var count = 1;
function tick(done) {
  setTimeout(function () {
    console.log('tick %s after %s ms', count++, 1000);
    done();
  }, 1000)
}

function* GeneratorFactory() {
  console.log('start run...');
  yield tick;
  console.log('tick 1 done');
  yield tick;
  console.log('tick 2 done');
  yield tick;
  console.log('tick 3 done');
}

function run(generator) {
  var ret = generator.next();
  if (ret.done) return;
  ret.value(function () {
    run(generator);
  });
}

run(GeneratorFactory());
```

Generator with Logic

```
function tick(time) {
  return function (done) {
    setTimeout(function () {
      done(null, time);
    }, time);
  }
}

function* GeneratorFunction() {
  var time;
  console.log('start run...');
  time = yield tick(500);
  console.log('tick 1 done after %s ms', time);
  time = yield tick(1000);
  console.log('tick 2 done after %s ms', time);
  time = yield tick(2000);
  console.log('tick 3 done after %s ms', time);
}

function run(generator, err, res) {
  var ret = generator.next(res);
  if (ret.done) return;
  ret.value(function (err, res) {
    run(generator, err, res);
  });
}

run(GeneratorFunction());
```

Minimal Implement of co

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

function tick(time) {
  return function (done) {
    setTimeout(function () {
      done(null, time);
    }, time);
  }
}

function* GeneratorFunction() {
  var time;
  console.log('start run...');
  time = yield tick(500);
  console.log('tick 1 done after %s ms', time);
  time = yield tick(1000);
  console.log('tick 2 done after %s ms', time);
  time = yield tick(2000);
  console.log('tick 3 done after %s ms', time);
}

co(GeneratorFunction).then();
```

co with Thunk

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

function tick(time) {
  return new Promise(function (resolve) {
    setTimeout(function () {
      resolve(time);
    }, time);
  });
}

function* GeneratorFunction() {
  var time;
  console.log('start run...');
  time = yield tick(500);
  console.log('tick 1 done after %s ms', time);
  time = yield tick(1000);
  console.log('tick 2 done after %s ms', time);
  time = yield tick(2000);
  console.log('tick 3 done after %s ms', time);
}

co(GeneratorFunction).then();
```

co with Promise

```
function tick(time) {
  return new Promise(function (resolve) {
    setTimeout(function () {
      resolve(time);
    }, time);
  });
}

async function asyncFunction() {
  var time;
  console.log('start run...');
  time = await tick(500);
  console.log('tick 1 done after %s ms', time);
  time = await tick(1000);
  console.log('tick 2 done after %s ms', time);
  time = await tick(2000);
  console.log('tick 3 done after %s ms', time);
}

asyncFunction();
```

ES7 Async Function

- Generator 本质上是一个迭代器
- Generator Function(function *) 提供动态构建 generator 的能力
- co 封装了 Generator，让其具有异步流程控制能力
- Async Function 是规范级别的解决方案

前戏结束

进入正题

什么是 Koa ?

- 提供最基础功能的 web 框架
- TJ Holowaychuk 出品
- 和 express 共享大部分的底层模块
- 通过 generator 提供更优的异步控制和异常处理

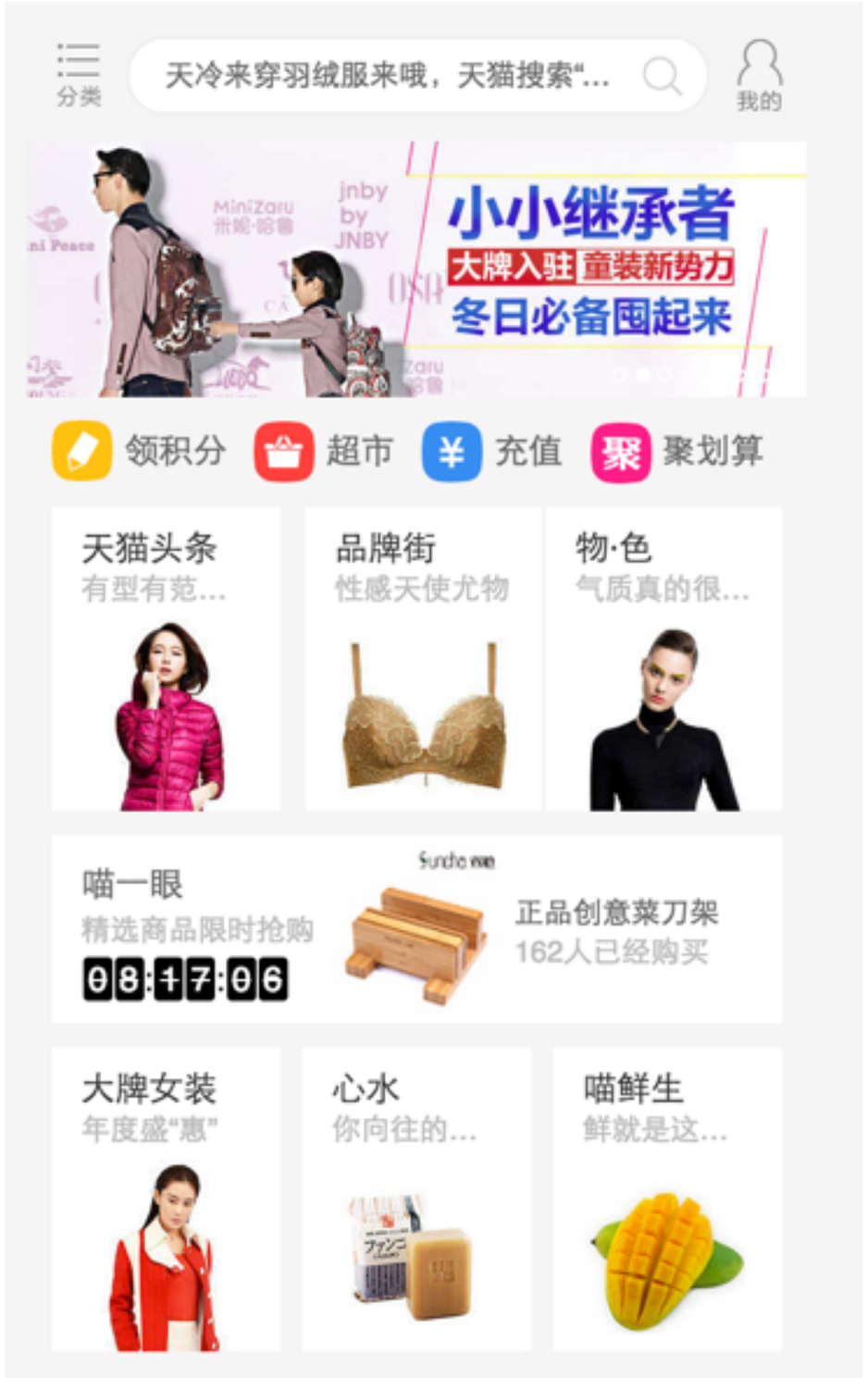
koa 是个玩具?


支付宝钱包
你好，何翊宇
退出
我的支付宝
安全中心
服务大厅
支付宝
我的支付宝
交易记录
会员保障
应用中心
首页
账户资产
账户设置
账户通
输入关键字，如"密码"
下午好，何翊宇 喝杯茶吧，让精神抖擞起来
账户名：156****1239
安全等级：高
上次登录时间：2014.10.25 10:06:57
账户余额
99999 元
充值
提现
转账
查看
余额宝
99999 元
转入
转出 | 管理 | 累计收益：920.04 元 [?]
信任宝
可用额度：
99999.00 元
管理 | 还款
金融机构为贷款提供方
招财宝
上周最高收益率 7%
上周平均收益率 5.82%
查看
更高收益多产品 | 管理
历史产品本息兑付率100%
其他账户
更多>
银行卡：3 张
阿里账户：2 个
购物账户：1 个
红包：1 个
集分宝：4 个

我的支付宝


天猫 TMALL.COM
搜索
2014秋装套装 | zara | iphone6 | 女毛衣 | 小米 | 魅族mx4 | 秋季童装 | 秋拖鞋
女神的新衣
总冠军今夜揭晓
商品服务分类
品牌街
喵鲜生
天猫会员
电器城
天猫超市
医药馆
淘宝旅行
天猫国际
精选市场
女装/内衣
男装/运动户外
女鞋/男鞋/箱包
化妆品/个人护理
手机数码/电脑办公
母婴玩具
零食/进口食品/酒
大家电/生活电器
家具建材
珠宝饰品/腕表眼镜
汽车/配件/车品
家纺/家饰/鲜花
医药保健
厨具/收纳/宠物
图书音像
11.11 购物狂欢节
双11来了
2014.11.11 COMING SOON
电脑平板
火热预售
双11手机预售
三星NOTE3降价
China unicom中国联通
让一切自由连通
联通宽带 不止50%off
服务更好 网速更快
购物车
热门品牌 HOT BRAND
大牌街
潮牌街
原创街
换一批
品牌库
自由新发现
WEEKLY
UNIQLO
new balance
品牌专享折扣
PHILIPS
MUJI
無印良品
L'ORÉAL
PARIS
BRAND OF THE DAY
www.tmall.com/go/market/promotion-act/shuangshiyiyushoudiannaobangong.php?...

天猫首页


分类
天冷来穿羽绒服来哦，天猫搜索“...
我的
小小继承者
大牌入驻 童装新势力
冬日必备囤起来
领积分
超市
充值
聚划算
天猫头条
有型有范...
品牌街
性感天使尤物
物·色
气质真的很...
喵一眼
精选商品限时抢购
08:17:06
正品创意菜刀架
162人已经购买
大牌女装
年度盛“惠”
心水
你向往的...
喵鲜生
鲜就是这...

天猫首页 H5

```
var koa = require('koa');
var app = koa();

app.use(function* () {
  this.body = 'hello koa';
});

app.listen(3000);
```

Hello Koa

```
app.use(function (req, res, next) {
  fs.readFile('package.json', function (err, content) {
    if (err) return next(err);
    try {
      content = JSON.parse(content);
    } catch (err) {
      return next(err);
    }
    content.user = 'dead_horse';
    res.json(content);
  });
});

app.use(function errorHandler (err, req, res, next) {
  res.status(500).send(err.message);
});
```

express 中的异常处理

```
app.use(function* (next) {
  try {
    yield next;
  } catch (err) {
    this.body = err.message;
    this.status = 500;
  }
});

app.use(function* () {
  var content = yield fs.readFile('package.json');
  var content = JSON.parse(content);
  content.user = 'dead_horse';
  this.body = content;
});
```

koa 中的异常处理

```
app.use(function (req, res) {
  var stream = fs.createReadStream('filename.txt');
  stream
  .on('error', onerror)
  .pipe(zlib.createGzip())
  .on('error', onerror)
  .pipe(res);

  res.once('close', function () {
    // 如果客户端终止了这个请求，可能导致 `fd` 泄漏
    // 需要 `unpipe` 来让上游关闭这个 `fd`
    stream.unpipe();
  });

  function onerror(err) {
    res.status(500).send(err.message);
  }
});
```

express 中的stream

```
// stream in koa
app.use(function* () {
  this.body = fs.createReadStream('filename.txt');
  this.body = this.body.pipe(zlib.createGzip());
});
```

koa 中的stream

koa 核心

- generator 带来更自然的异常处理
- 抽象的 context + 神奇的 setter
- “修饰器”模式的中间件

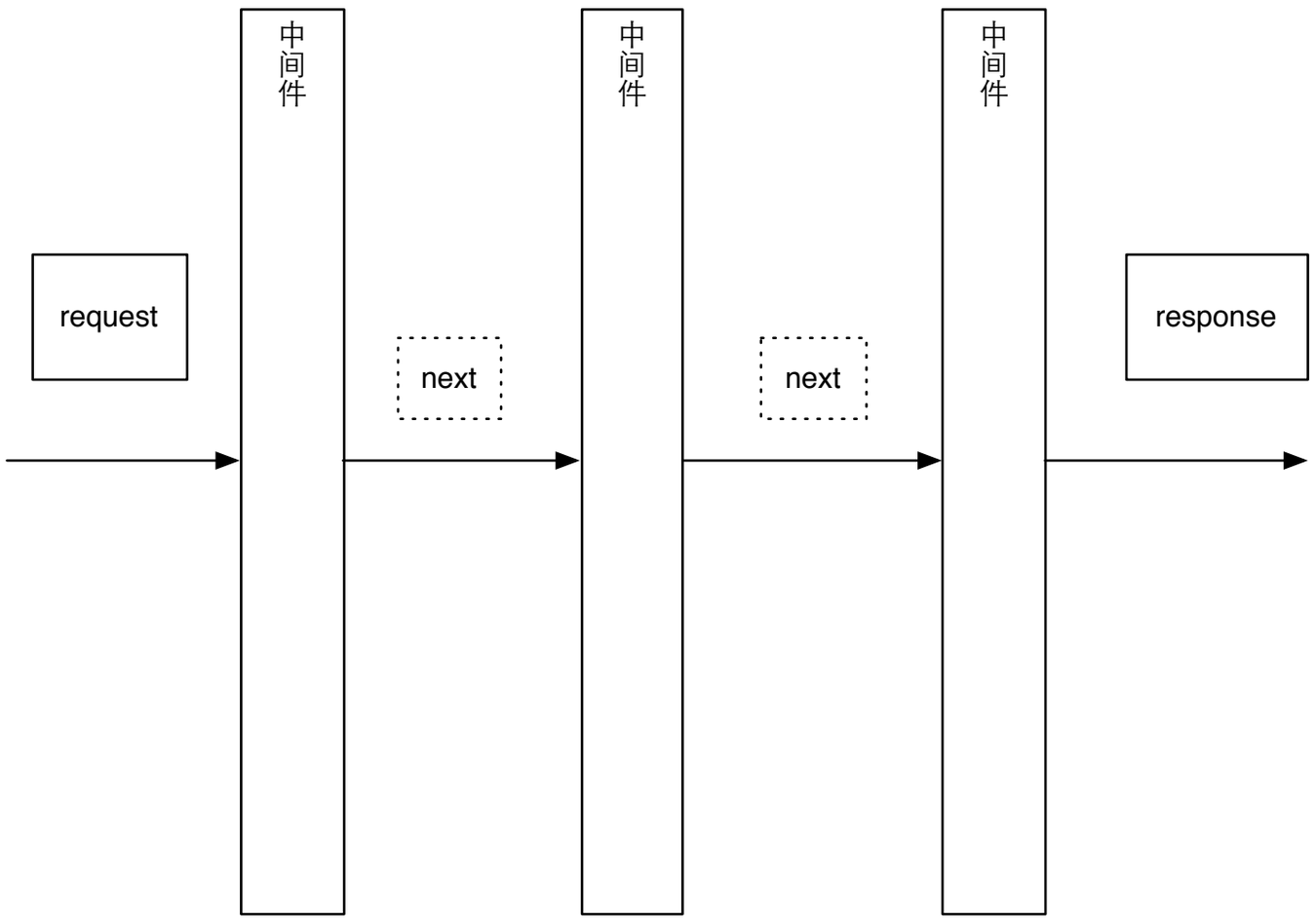
```
app.use(function* responseTime(next) {
  var start = Date.now();
  // 在进入下一个中间件前执行
  yield next;
  // 在执行完后面的所有中间件之后执行
  var used = Date.now() - start;
  this.set('X-Response-Time', used);
});

app.use(function* respond() {
  this.body = 'hello koa';
  this.status = 200;
});
```

koa 中间件

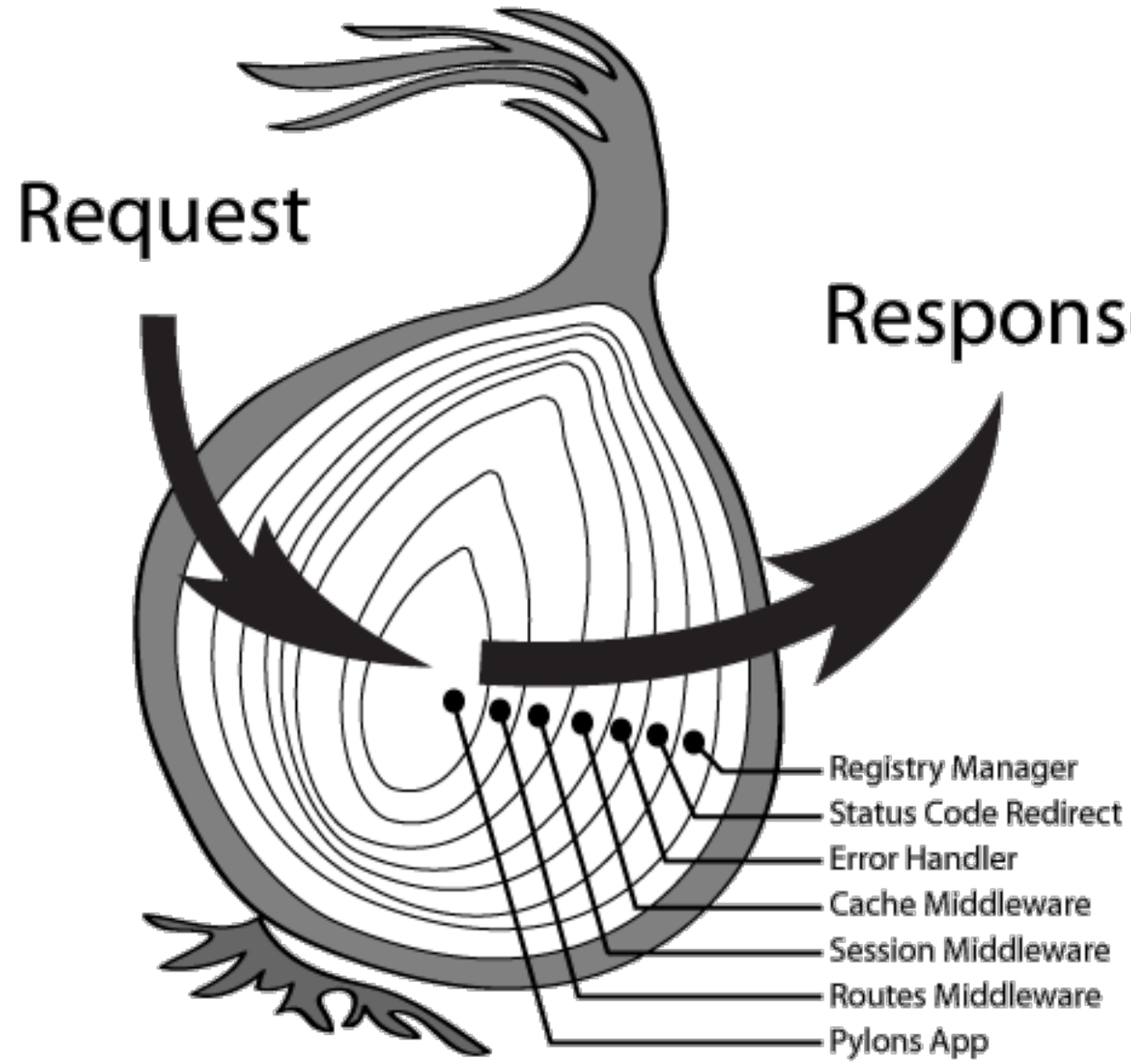
```
app.use(function* () {
  var start = Date.now();

  // yield next() => yield respond();
  this.body = 'hello koa';
  this.status = 200;

  var used = Date.now() - start;
  this.set('X-Response-Time', used);
});
```

koa 中间件

express



koa


Request
Response
Registry Manager
Status Code Redirect
Error Handler
Cache Middleware
Session Middleware
Routes Middleware
Pylons App

```
var koa = require('koa');
var app = koa();

app.use(function* responseTime(next) {
  var start = new Date;
  yield next;
  var ms = new Date - start;
  this.set('X-Response-Time', ms + 'ms');
});

app.use(function* logger(next) {
  var start = new Date;
  yield next;
  var used = new Date - start;
  console.log('%s %s %s %sms',
    this.method,
    this.originalUrl,
    this.status, used);
});

app.use(function* contentLength(next) {
  yield next;
  if (!this.body) return;
  this.set('Content-Length', this.body.length);
});

app.use(function* body(next) {
  yield next;
  if (this.path !== '/') return;
  this.body = 'Hello World';
});

app.listen(3000);
```

```
// 设置响应的 body
app.use(function* respond(next) {
  this.body = fs.createReadStream(this.path);
  yield next;
});
// 设置 etag
app.use(function* etag(next) {
  var stat = yield fs.stat(this.body.path);
  this.response.etag = require('etag')(stat);
  yield next;
});
// 将 body gzip
app.use(function* gzip(next) {
  this.body = this.body.pipe(zlib.createGzip());
});
```

修改响应

```
app.use(function* () {
  // 设置响应 body
  this.body = fs.createReadStream(this.path);

  // 设置 etag
  var stat = yield fs.stat(this.body.path);
  this.response.etag = require('etag')(stat);

  // 修改响应 body (gzip 压缩)
  this.body = this.body.pipe(zlib.createGzip());

  // koa 自带的最外层的中间件，向外写响应
  this.body.pipe(this.res);
});
```

修改响应

express-compression

VS

koa-compress

- koa 不仅仅是 generator + express
- koa 拥有自己的一套理念
- koa 十分精简，具有高度可扩展性

Q&A

Thanks

更多学习资料

- koajs/koa: 源码是最好的教程
- dead-horse/koa-step-by-step: 示例代码和一些简单的例子
- koa/examples: 各种示例
- koajs/kick-off-koa: 交互式学习教程
- koajs/workshop: TDD 式学习教程
- tj/co: 基于 generator 的异步流程控制解决方案
- jshttp: web 框架的基石