Application.js

Listen()

Use(fn)

use(fn) {

if (typeof fn !== 'function') throw new TypeError('middleware must be a function!');

if (isGeneratorFunction(fn)) { //判断是不是generator函数

//反对？？提示 depd模块

deprecate('Support for generators will be removed in v3. ' +

'See the documentation for examples of how to convert old middleware ' +

'https://github.com/koajs/koa/blob/master/docs/migration.md');

fn = convert(fn); //转化

}

debug('use %s', fn.\_name || fn.name || '-');

this.middleware.push(fn);

return this;

}

callback() {

const fn = compose(this.middleware);

if (!this.listeners('error').length) this.on('error', this.onerror);

const handleRequest = (req, res) => {

res.statusCode = 404;

const ctx = this.createContext(req, res);

const onerror = err => ctx.onerror(err);

const handleResponse = () => respond(ctx);

onFinished(res, onerror);

return fn(ctx).then(handleResponse).catch(onerror);

};

return handleRequest;

}

# 辅助函数

1.Is-generator函数

//主要从三点看。

//一点是function toString

//一点是[object GeneratorFunction] Object toString

//一点是从原型看(内部[[Prototype]]属性的值)

module.exports = function isGeneratorFunction(fn) {

if (typeof fn !== 'function') {

return false;

}

if (isFnRegex.test(fnToStr.call(fn))) { ///^\s\*(?:function)?\\*/

return true;

}

if (!hasToStringTag) { //判断环境认不认识Symbol 可能是判断es6和es5

var str = toStr.call(fn);

return str === '[object GeneratorFunction]';

}

// getProto Object.getPrototypeOf 获取原型

// GeneratorFunction function\*() {}的原型

return getProto(fn) === GeneratorFunction;

};

1. Only 返回koa的json对象

keys.reduce(function(ret, key){

if (null == obj[key]) return ret;

ret[key] = obj[key];

return ret;

}, {});

3.koa-compose 模块 洋葱圈核心方法 await next;调用下一个中间件的方法

类似栈 返回Promise对象

return function (context, next) {

// last called middleware #

let index = -1

return dispatch(0)

function dispatch (i) {

if (i <= index) return Promise.reject(new Error('next() called multiple times'))

index = i

let fn = middleware[i]

if (i === middleware.length) fn = next

if (!fn) return Promise.resolve()

try {

return Promise.resolve(fn(context, function next () {

return dispatch(i + 1)

}))

} catch (err) {

return Promise.reject(err)

}

}

}

4.koa-convert 模块 就是把转化成Promise对象，感觉