Egg.js 中使用 socket.io

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

一、 socket.io 中的命名空间

所谓命名空间，就是指在不同的域当中发消息只能给当前的域的 socket 收到。

```
io.of('/cart').on('connection', (socket) => {
    console.log("来了一个客户");
    //给客户端广播一个消息  发送给建立连接的用户

    let roomId = querystring.parse(socket.request.url.split("?")[1]).roomid;
    socket.join(roomId);

    socket.on("addCart",(clientData)=>{
        console.log(clientData);
         io.of('/cart').to(roomId).emit("serverMsg","this is addCart msg");
    })
});
```

客户端：

```
var socket = io.connect("http://192.168.0.13:8000/cart?roomid=20");
socket.on("serverMsg",function(serverData){
    console.log(serverData);
})
function addCart(){
    socket.emit("addCart","client AddCart");
}
```

二、 egg-socket.io 介绍

网址：https://eggjs.org/zh-cn/tutorials/socketio.html

Socket.IO 是一个基于 Node.js 的实时应用程序框架，在即时通讯、通知与消息推送，实时分析等场景中有较为广泛的应用。

WebSocket 的产生源于 Web 开发中日益增长的实时通信需求，对比基于 http 的轮询方式，它大大节省了网络带宽，同时也降低了服务器的性能消耗； socket.io 支持 websocket、polling 两种数据传输方式以兼容浏览器不支持 WebSocket 场景下的通信需求。

框架提供了 egg-socket.io 插件，增加了以下开发规约：

namespace: 通过配置的方式定义 namespace（命名空间）
middleware: 对每一次 socket 连接的建立/断开、每一次消息/数据传递进行预处理
controller: 响应 socket.io 的 event 事件
router: 统一了 socket.io 的 event 与 框架路由的处理配置方式

三、项目中集成 egg-socket.io

参考：https://eggjs.org/zh-cn/tutorials/socketio.html

官方 demo: https://github.com/eggjs/egg-socket.io/tree/master/example

1、安装 egg-socket.io

```
npm i egg-socket.io --save
```

2、config/plugin.js 中开启插件

```
exports.io = {
  enable: true,
  package: 'egg-socket.io',
};
```

3、config/config.default.js 中配置插件

```
exports.io = {
    init: { }, // passed to engine.io
    namespace: {
        '/': {
            connectionMiddleware: [],
            packetMiddleware: [],
        },
        '/example': {
            connectionMiddleware: [],
            packetMiddleware: [],
        },
    },
};
```

命名空间为 / 与 /example

4、路由中配置命名空间对应的控制器

```
// app.io.of('/')
app.io.route('chat', app.io.controller.chat.index);

// app.io.of('/chat')
app.io.of('/chat').route('chat', app.io.controller.chat.index);
```

5、配置中间件 以及 控制器

io/middleware/auth.js

```
'use strict';
module.exports = () => {
    return async (ctx, next) => {
        ctx.socket.emit('res', 'auth! server say');
        await next();
        console.log('disconnect!');
    };
};
```

io/controller/chat.js

```
'use strict';
module.exports = app => {
```

```
    class Controller extends app.Controller {
        async index() {
            const message = this.ctx.args[0];
            console.log('chat :', message + ' : ' + process.pid);
            this.ctx.socket.emit('res', "say");
        }
    }
    return Controller;
};
```