

node.js调试

用了几天node.js感觉很新奇,但是调试问题实在是愁煞人,开始的时候懒的学习调试方法,看看异常内容就可以了,但随着代码复杂程度的上升,并不是所有错误都是语法错误了,不调试搞不定了,只好搜搜资料,学习了一下怎么调试。

不用每次都重启服务的supervisor

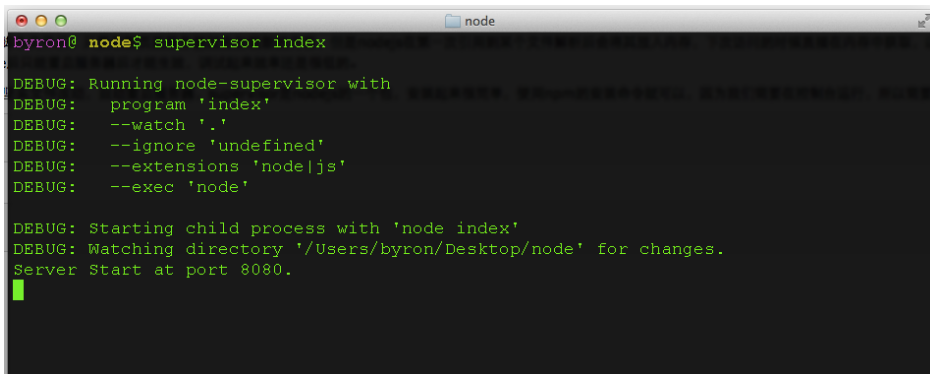
使用过PHP的同学肯定都清楚,修改了某个脚本文件后,只要刷新页面服务器就会加载新的内容,但是node.js在第一次引用到某个文件解析后会将其放入内存,下次访问的时候直接在内存中获取,以提高效率,但是这对我们开发造成一定困扰,修改了某个module后只能重启服务器后才能生效,调试起来效率还是很低的。

于是乎node.js中有了supervisor插件帮我们坚实文件改动,自动重启服务器,supervisor是node.js的一个包,安装起来很简单,使用npm的安装命令就可以,因为我们需要在控制台运行,所以需要安装在全局环境中

```
npm install -g supervisor
```

这样我们就可以使用supervisor启动脚本了

```
supervisor index
```

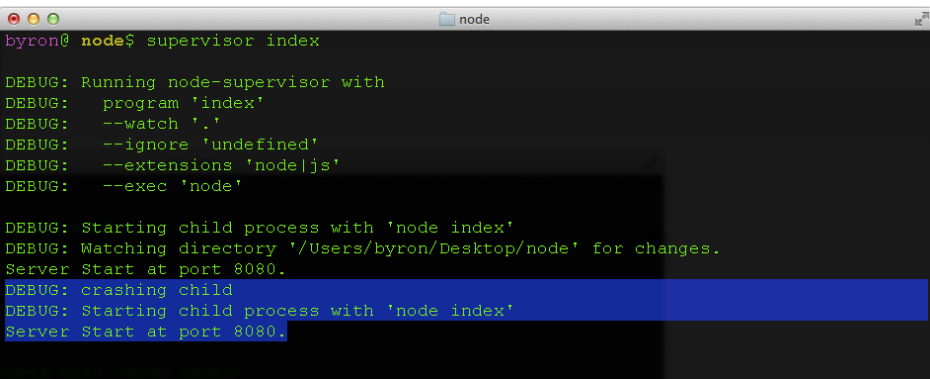


```
byron@ node$ supervisor index

DEBUG: Running node-supervisor with
DEBUG:   program 'index'
DEBUG:   --watch '.'
DEBUG:   --ignore 'undefined'
DEBUG:   --extensions 'node|js'
DEBUG:   --exec 'node'

DEBUG: Starting child process with 'node index'
DEBUG: Watching directory '/Users/byron/Desktop/node' for changes.
Server Start at port 8080.
```

当我们对文件做了改动的时候,可以看到控制台多了三行,服务器已经重启了



```
byron@ node$ supervisor index

DEBUG: Running node-supervisor with
DEBUG:   program 'index'
DEBUG:   --watch '.'
DEBUG:   --ignore 'undefined'
DEBUG:   --extensions 'node|js'
DEBUG:   --exec 'node'

DEBUG: Starting child process with 'node index'
DEBUG: Watching directory '/Users/byron/Desktop/node' for changes.
Server Start at port 8080.
DEBUG: crashing child
DEBUG: Starting child process with 'node index'
Server Start at port 8080.
```

原生控制台调试

node.js本身支持调试,在语句前面加debugger指令就可以添加一个断点

```
var server=require('./server'),
    router=require('./router'),
    requestHandlers=require('./requestHandlers');

debugger;
var handle={};
debugger;
handle['/']=handle['/start']=requestHandlers.start;
```

公告

阿里巴巴国际站招前端

luyong.sunly@alibaba-inc.com

昵称: Samaritans

园龄: 4年5个月

粉丝: 2046

关注: 12

+加关注

最新随笔

1. 前端零基础学习提纲
2. 阿里前端两年随想
3. JavaScript面向对象
4. 前端模块化
5. 你可能没注意的CSS单位
6. CSS 实现打字效果
7. direction和unicode-bidi
8. CSS3 滤镜
9. EDM制作要点
10. Outlook HTML渲染引擎

随笔分类(107)

- ASP.NET(5)
- CSS(21)
- ECMAScript5(5)
- HTML(9)
- JavaScript Core(22)
- JavaScript应用(8)
- jQuery(5)
- node.js(4)
- PHP step by step(2)
- Web综合(8)
- 设计模式(3)
- 我所理解的OOP(1)
- 杂谈(8)
- 自备JavaScript库(6)

随笔档案(108)

- 2016年2月 (1)
- 2016年1月 (1)
- 2015年4月 (1)
- 2015年3月 (1)
- 2014年11月 (10)
- 2014年1月 (8)
- 2013年12月 (13)
- 2013年11月 (9)
- 2013年10月 (18)
- 2013年9月 (15)
- 2013年8月 (22)
- 2013年4月 (3)
- 2012年11月 (3)
- 2012年10月 (2)
- 2012年9月 (1)

积分与排名

积分 - 298632

排名 - 412

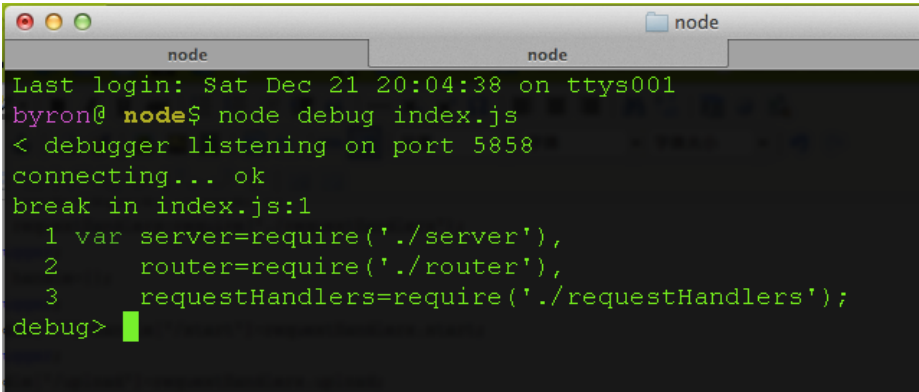
推荐排行榜

1. 找工作的一些感悟——前端小菜的成长(300)
2. 了解了这些才能开始发挥jQuery的威力(78)
3. 容易被忽略CSS特性(66)
4. JavaScript 正则表达式上——基本语法(66)
5. 简单理解Socket(60)

```
debugger;
handle['/upload']=requestHandlers.upload;
handle['/show']=requestHandlers.show;
debugger;
server.start(8080,router.route,handle);
```

在启动服务的时候添加debug 选项

```
node debug index.js
```



这时候输入一些指令就可以单步调试、到断点监视局部变量等，看个命令图，很多命令都有其缩写形式

node.js调试命令

命令	功能
run	执行脚本,在第一行暂停
restart	重新执行脚本
cont, c	继续执行,直到遇到下一个断点
next, n	单步执行
step, s	单步执行并进入函数
out, o	从函数中步出
setBreakpoint(), sb()	当前行设置断点
setBreakpoint('f()'), sb(...)	在函数f的第一行设置断点
setBreakpoint('script.js', 20), sb(...)	在 script.js 的第20行设置断点
clearBreakpoint, cb(...)	清除所有断点
backtrace, bt	显示当前的调用栈
list(5)	显示当前执行到的前后5行代码
watch(expr)	把表达式 expr 加入监视列表
unwatch(expr)	把表达式 expr 从监视列表移除
watchers	显示监视列表中所有的表达式和值
repl	在当前上下文打开即时求值环境
kill	终止当前执行的脚本
scripts	显示当前已加载的所有脚本

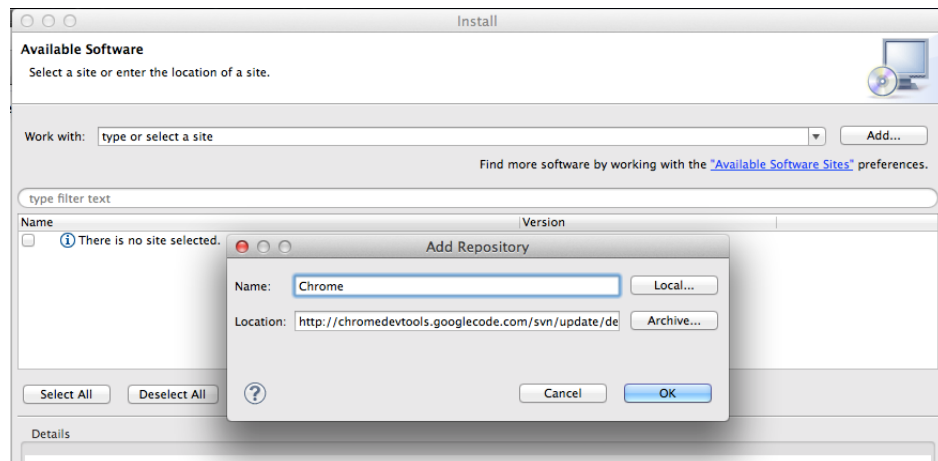
- 6. JavaScript prototype(55)
- 7. JavaScript 闭包究竟是什么(48)
- 8. CSS布局 ——从display , position , float属性谈起(45)
- 9. CSS清浮动处理 (Clear与BFC) (33)
- 10. JavaScript面向对象(33)

详细使用有兴趣同学可以自己摸索，我是没兴趣。。。太复杂了，看几个贴心的

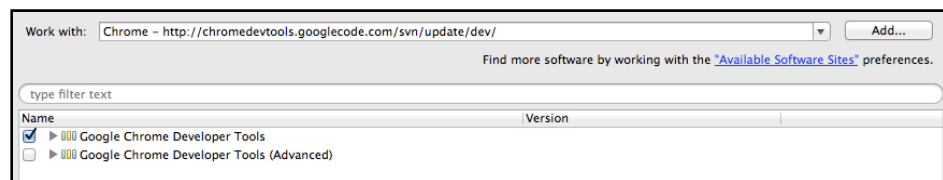
使用Eclipse调试

是的，Eclipse又威武了，连node.js也能调试，在Eclipse官网下载eclipse，然后 Help->Install New Software->Add

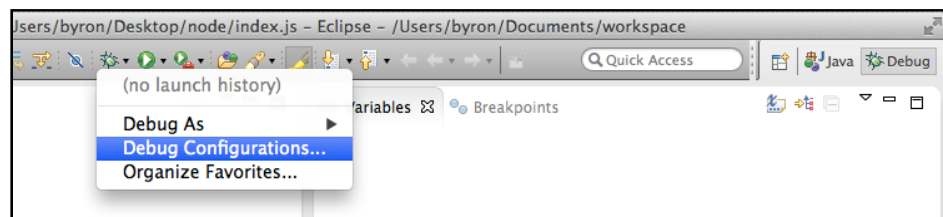
在弹出的窗口添加一个源，名字好记就行，地址是<http://chromedevtools.googlecode.com/svn/update/dev/>



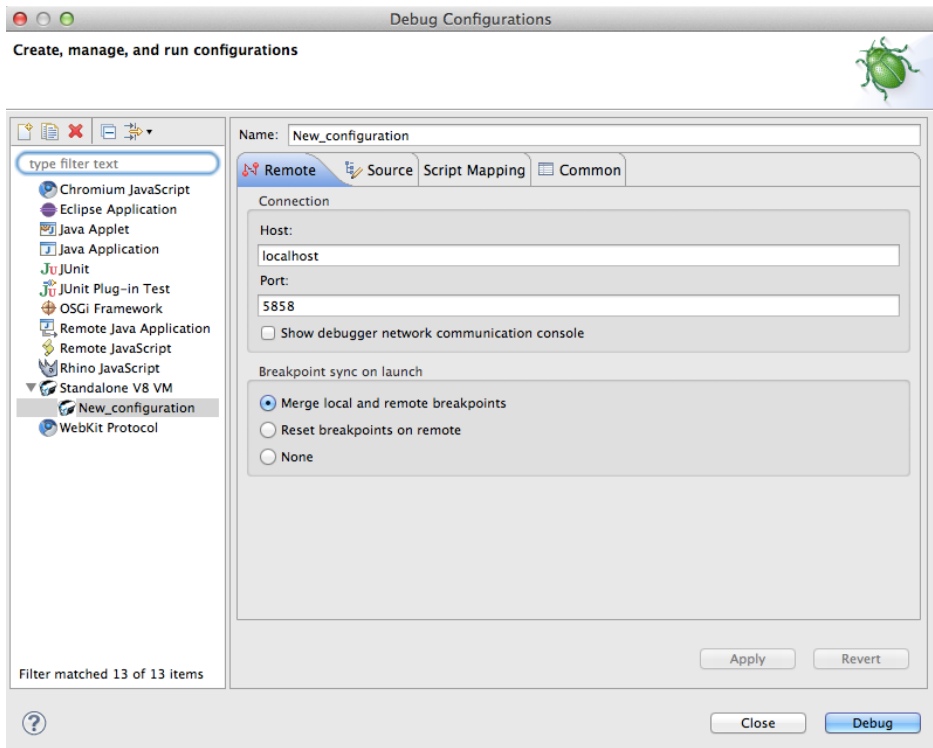
等一会儿后弹出选择界面，选中第一个



一路next到最后finish，下载完成后会提醒重启Eclipse，完成之后就可以调试node.js了，打开想调试的文件，切换Eclipse到调试视图，点击工具栏右边的小三角，选择Debug Configuration



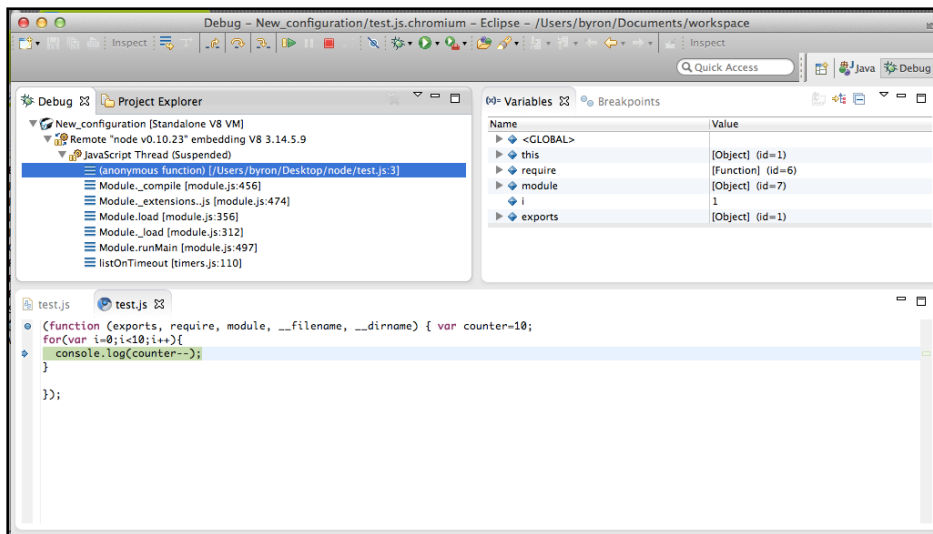
双击 Standard V8 VM 选项创建一个新的配置，填好相应参数



通过 --debug-brk选项在控制台启动node服务器

```
node --debug-brk=5858 test.js
```

点击Eclipse刚才界面的debug按钮，就可以像调试Java一样调试node.js了

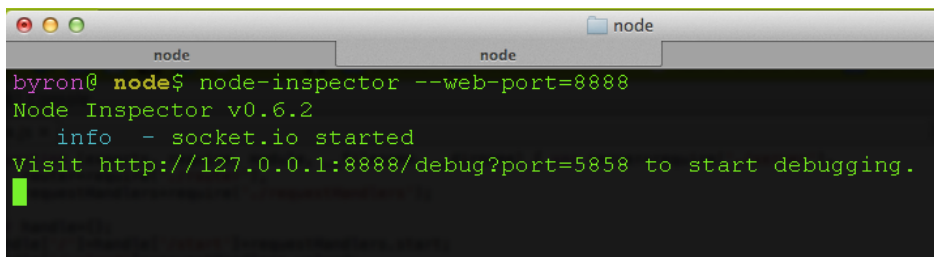


使用node-inspector调试

大部分node.js应用都是web应用，所以一些基于Chrome的在线调试工具应运而生，最出名的应该就是node-inspector了，这是一个node.js的模块，安装、使用相当的方便，首先使用npm把其安装在全局环境中

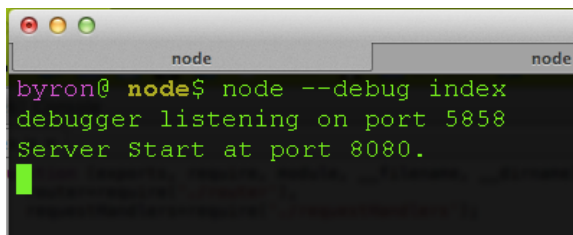
```
npm install -g node-inspector
```

node-inspector是通过websocket方式来转向debug输入输出的。因此，我们在调试要先启动node-inspector来监听node.js的debug调试端口。默认情况下node-inspector的端口是8080，可以通过参数--web-port=[port]来设置端口。



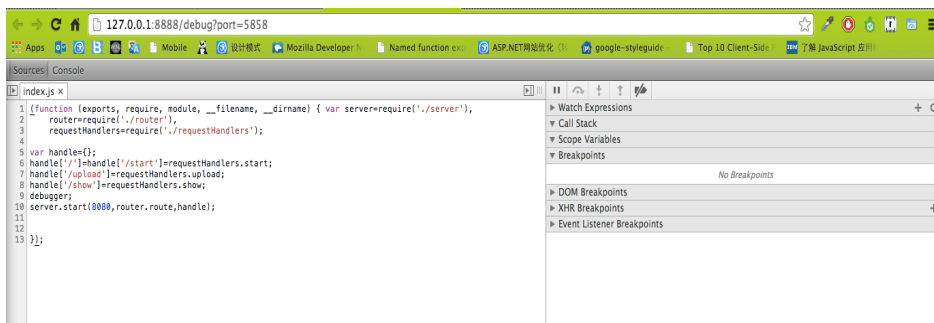
```
byron@ node$ node-inspector --web-port=8888
Node Inspector v0.6.2
info - socket.io started
Visit http://127.0.0.1:8888/debug?port=5858 to start debugging.
```

在启动node-inspector之后，我们可以通过--debug或--debug-brk来启动node.js程序。



```
byron@ node$ node --debug index
debugger listening on port 5858
Server Start at port 8080.
```

这时候就可以访问<http://127.0.0.1:8888/debug?port=5858> 使用浏览器调试了，看看界面，不用多说什么了吧



最后

参考：node.js开发指南

PS：个人觉得还是最后一种最方便

分类: [node.js](#)



Samaritans



关注 - 12
粉丝 - 2046

6 0

(请你对文章做出评价)

+加关注

« 上一篇：node.js module初步理解

» 下一篇：JavaScript 正则表达式上——基本语法

posted @ 2013-12-22 14:56 Samaritans 阅读(12752) 评论(7) 编辑 收藏

评论列表

#1楼 2013-12-22 20:07 {name:"代码屠夫"}

webstorm随便调

支持(0) 反对(0)

#2楼[楼主] 2013-12-22 22:15 Samaritans

@ {name:"代码屠夫"}

引用

webstorm随便调

没用过，有空试试

支持(0) 反对(0)

#3楼 2013-12-23 09:28 特雷西#1

你的头像！！！！

支持(0) 反对(0)

#4楼[楼主] 2013-12-23 09:43 Samaritans

@ 特雷西#1

引用

你的头像！！！！

帅吧

支持(0) 反对(0)

#5楼 2013-12-23 16:17 Arliang

@ dolphinX

@{name:"代码屠夫"}

node-inspector比webstorm好用，webstorm会有看不到对象值的情况

支持(0) 反对(0)

#6楼[楼主] 2013-12-23 16:24 Samaritans

@ Arliang

引用

@dolphinX

@{name:"代码屠夫"}

node-inspector比webstorm好用，webstorm会有看不到对象值的情况

是嘛，有空看看，都关注一下，谢谢了

支持(0) 反对(0)

#7楼 2015-11-16 15:07 bugong

mark

支持(0) 反对(0)

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库

【推荐】融云即时通讯云 - 豆果美食、Faceu等亿级APP都在用

ActiveReports

全方位的报表解决方案

终端用户

报表服务器

数据库服务器

立即下载

GrapeCity

最新IT新闻:

- 58到家陈小华：为什么O2O领域一定会出现百亿美元公司？
- Snapchat“碾压”Facebook：视频观看量一年内增长150%
- 刚挂牌新三板的爱尚鲜花承认刷单超3000万，但它只是冰山一角
- 太阳系或存在过超级地球：因无法摆脱太阳引力被吞噬

4/20/2016

- [三星全面开发虚拟现实 除了硬件业务还投资开发软件](#)
- » [更多新闻...](#)



最新知识库文章:

- [架构漫谈（九）：理清技术、业务和架构的关系](#)
- [架构漫谈（八）：从架构的角度看如何写好代码](#)
- [架构漫谈（七）：不要空设架构师这个职位，给他实权](#)
- [架构漫谈（六）：软件架构到底是要解决什么问题？](#)
- [架构漫谈（五）：什么是软件](#)
- » [更多知识库文章...](#)