[安装 2](#_Toc391203109)

[运行 2](#_Toc391203110)

[注意事项 2](#_Toc391203111)

[自动测试 2](#_Toc391203112)

[运行区域测试 3](#_Toc391203113)

[异步测试 3](#_Toc391203114)

[如何像jamsine一样扩展？ 4](#_Toc391203115)

[Webstorm中调试和运行jasmine-node测试 4](#_Toc391203116)

[使用SinonJs 6](#_Toc391203117)

[问题记录 7](#_Toc391203118)

[1. 在异步访问node.js服务器进程的测试中，测试代码中的sinon.stub不能影响到node.js服务器进程 7](#_Toc391203119)

[问题描述 7](#_Toc391203120)

[原因分析 8](#_Toc391203121)

[解决方案 8](#_Toc391203122)

[注意事项 8](#_Toc391203123)

[1. 如果不用异步测试，则function的形参不要为done！否则会造成测试一直等待，从而超时！ 8](#_Toc391203124)

# 安装

npm install jasmine-node -g

# 运行

在cmd中执行命令

可以全局运行jasmine-node，但是测试文件需要指定目录。因此，可以先进入测试文件所在目录，然后在执行命令。

如测试文件aSpec.js在D:\test目录内，则可输入一下命令：

jasmine-node D:\test\aSpec.js

或者

cd D:\test\

Jasmine-node aSpec.js

# 注意事项

测试文件要以\*spec.js或\*Spec.js命名

# 自动测试

加上参数：--auto

如果测试代码更改，则选中命令窗口（cmd窗口）时，会自动运行发生了更改的测试文件。

可以自动测试单个文或目录内的所有文件

如：jasmine --auto testSpec.js

jasmine –-auto script/test/

# 运行区域测试

可以运行目录内的所有测试。

如需要运行script/test目录中的测试，则可输入命令：

jasmine-node script/test/或者jasmine-node script/test

# 异步测试

调用done()，则测试会在done调用后执行。

官方说明：

Accepting a done callback in the specification will trigger jasmine-node to run the test asynchronously waiting until the done() callback is called.

An asynchronous test will fail after 5000 ms if done() is not called. This timeout can be changed by setting jasmine.getEnv().defaultTimeoutInterval or by passing a timeout interval in the specification.

**var** request **=** require('request');

it("should respond with hello world", **function**(done) {

request("http://localhost:3000/hello", **function**(error, response, body){

done();

}, 250); *// timeout after 250 ms*

});

or

**var** request **=** require('request');

jasmine.getEnv().defaultTimeoutInterval **=** 500;

it("should respond with hello world", **function**(done) {

request("http://localhost:3000/hello", **function**(error, response, body){

done();

}); *// timeout after 500 ms*

});

# 如何像jamsine一样加入custom matcher？

加入文件名包含helpers（不能为Helper）的文件（即原specHelper.js），将jasmine改为this,去掉global(window)

参考：

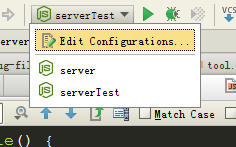
<http://stackoverflow.com/questions/21868851/jasmine-node-including-helper>

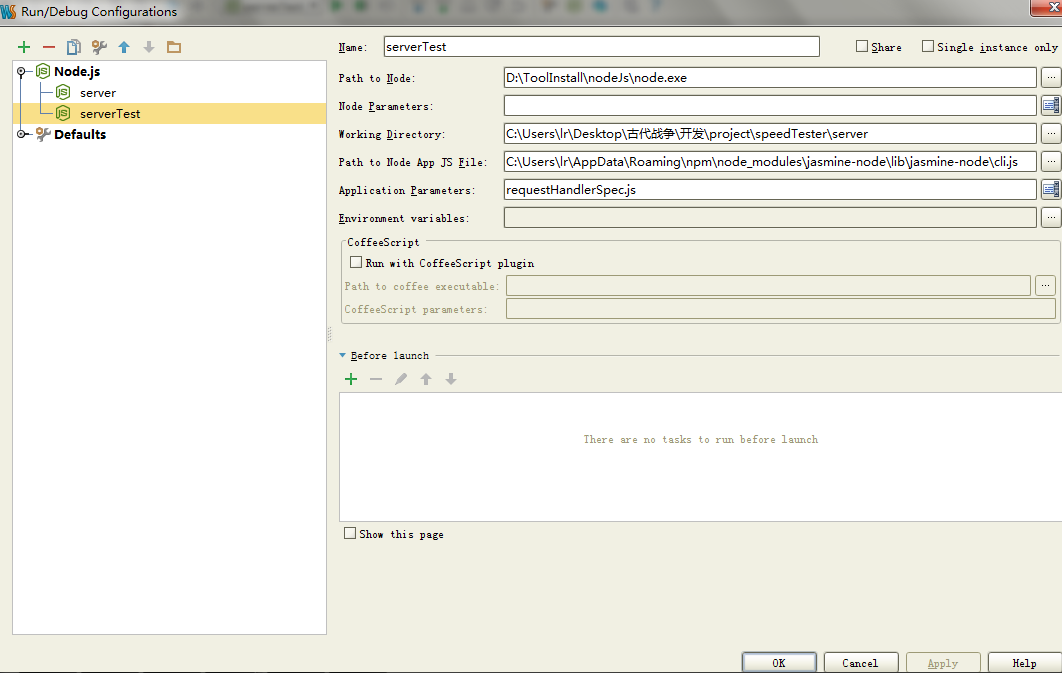
https://www.npmjs.com/package/jasmine-matchers

# Webstorm中调试和运行jasmine-node测试

可以在edit configuration中配置，配置cli.js文件。

然后可以运行或调试jasmine-node测试：





Notes:

* The working directory is the directory containing my test scripts.
* I have jasmine-node installed globally. If you installed it locally, your path to cli.js will be under your local node\_modules directory.
* In Application Parameters you specify the particular test script that you want to run under debug. You can use a dot '.' to run all tests in your working directory.

So now I just select this run configuration, and debug.

**测试代码异步访问nodejs服务器**

如果测试代码异步访问了nodejs服务器，则需要打开nodejs服务器进程。

如果要调试服务器端，则应该先调试nodejs服务器进程，然后再调试jasmine-node。

**为什么要配置cli.js文件？这是因为：**

In short, just debug jasmine-node:

node --debug-brk node\_modules/jasmine-node/lib/jasmine-node/cli.js spec/my\_spec.js

If you look at the source of the jasmine-node script, it just invokes cli.js, and I found I could debug that script just fine.

I wanted to use node-inspector to debug a CoffeeScript test. Just adding the --coffeeswitch worked nicely, e.g.

node --debug-brk node\_modules/jasmine-node/lib/jasmine-node/cli.js

# 使用SinonJs

使用npm安装sinonJs模块：npm install sinon

注意要修改sinon->collection.js->stub源码，使其能正确fake

# 问题记录

1. 在异步访问node.js服务器进程的测试中，测试代码中的sinon.stub不能影响到node.js服务器进程

### 问题描述

如测试代码为：

it("拷贝指定目录下的所有文件到输出目录下", function () {

var baseDir = "C:/baseDir/";

sandbox.stub(require("../config"), "fileServerBaseDir", baseDir);

request("http://localhost:8500/speedTester.html", function (error, response, body) {

expect(tool.copyFileSync.callCount).toEqual(2);

done();

});

});;

服务器代码为：

http.createServer(function (req, res) {

var baseDir = require("../config").fileServerBaseDir;

}).listen(config.port, function () {

console.log("Server listening for requests on port " + config.port);

});

则服务器的baseDir并不为"C:/baseDir/"，而是config本来的fileServerBaseDir！

### 原因分析

我调试了该代码，发现require会用到缓存Module.\_cache：

使用 require 函数引入外部模块时，Node.js 会根据模块的实际文件路径进行缓存，同一路径的模块只会加载一次。这就意味着同一进程的多个模块引入同一外部模块时，得到的是同一对象。即如果在测试代码中同样引入 Request 模块并对其 get 方法进行 Stub，就如同在要测试的模块中对 request.get 进行 Stub。

实质是stub(Module.\_cache)，这里测试代码使用了request模块进行集成测试，使得测试代码中的Module.\_cache和服务端的Module.\_cache并不相同！

这是因为测试代码和服务器位于不同的进程中，所以两者的Module.\_cache不相同！

### 解决方案

1. sandbox.Stub全局方法require，使其在require("../config")时返回fakeConfig

该方法未测试！

1. 将服务器要测试的代码抽离到模块中，直接在测试进程中引用该模块并测试。

抽离后，使得测试代码不用request模块进行集成测试，而是改为单元测试。这样测试代码与被测试的产品代码就都在一个进程中了，这样就可以进行stub了。

# 注意事项

1. 如果不用异步测试，则function的形参不要为done！否则会造成测试一直等待，从而超时！

如下面代码所示，会造成测试一直等待，从而超时：

it("should respond with hello world", **function**(done) {

});

应该删除“done”形参：

it("should respond with hello world", **function**() {

});

1. 如果require的路径错误，则在测试时会报“Exception loading”的错误。