精华 初探nodejs的c++ addon

• 发布于 3 年前 • 作者 DoubleSpout (/user/DoubleSpout) • 11043 次浏览

最近我打算把公司的接口服务前端换成node，可以减少负载服务器的数量，节约成本，同时也想把node作为公司的一门重要语言加入到项目中，以前都是.net和php。

node已经很快了，但是我想榨干它的极限，想让它更快一些，所以就想到了c++模块。于是对照着node官网的手册一步步的开始摸索c++模块和V8。

要写C++模块，必须先搭建一个环境，node-gyp命令是必须。所以我们得先安装node-gyp

```
npm install -g node-gyp  //当然要保证你的node版本是0.8.x
```

装好node-gyp后，我发现死活跑不起来官网的hello world例子，于是打开node-gyp的github页面，赫然印有需要 python 2.7.3 的要求，于是继续折腾py，具体py的安装例子参考：
CentOs 安装python (http://snoopyxdy.blog.163.com/blog/static/60117440201252711214619 9/)
python 2.7.3 下载 (http://www.python.org/download/releases/2.7.3/#download)
装好py之后，我们就完成了环境搭建工作了，可以跑官网的hello world例子了。
创建node-gyp文件：

```
{
  "targets": [
    {
      "target_name": "hello",
      "sources": [ "hello.cc" ]
    }
  ]
}
```

接着执行node-gyp configure然后执行node-gyp build如果报找不到binding.gyp的错误，只要把binding.gyp拷贝到build文件目录下即可

hello.cc (http://hello.cc):

```
#include <node.h>
#include <v8.h>
using namespace v8;
Handle<Value> Method(const Arguments& args) {
  HandleScope scope;
  return scope.Close(String::New("world"));
}

void init(Handle<Object> target) {
  target->Set(String::NewSymbol("hello"),
      FunctionTemplate::New(Method)->GetFunction());
}
NODE_MODULE(hello, init)
```

这是官网一个简单的helloworld的示例，node端调用方法为：

```
var hello = require('./addon/build/Release/hello.node').hello();
console.log(hello); //这里打印world字符串
```

成功跑起来例子之后，我打算从一个简单的模块入手，慢慢的熟悉如何写C++的addon，我们开发一个简单的但是实用的node验证模块，比如验证参数是否是数组，数字，字符串，大于0等等。因为这个模块在我项目中可能被大量用到，需要验证接口参数的大量合法性数据。
我们构建一个verify文件夹，用来做开发verify验证模块用，目录结构如下：

```
verify
/index.js
/package.json
/readme.md
/lib
    /verify.js
/addon
    /binding.gyp
    /SimpleVerify.cc
    /SimpleVerify.h
    /verify.cc
```

一个简单的模块目录构成，不知道写的对不对，反正我就这么弄了，哈哈。
我们直接看下3个c++代码：（依葫芦画瓢的，不要取笑啊）

verify.cc (http://verify.cc):

```
#define BUILDING_NODE_EXTENSION
#include <node.h>
#include "SimpleVerify.h"
using namespace v8;



void Init(Handle<Object> target) {

  target->Set(String::NewSymbol("isArray"),
           FunctionTemplate::New(SimpleV::isArray)->GetFunction());

  target->Set(String::NewSymbol("isNumber"),
           FunctionTemplate::New(SimpleV::isNumber)->GetFunction());
}

NODE_MODULE(verify, Init)
```

SimpleVerify.h

```
#ifndef SV_H
#define SV_H
#include <node.h>

class SimpleV {
 public:
  static v8::Handle<v8::Value> isArray(const v8::Arguments& args);
  static v8::Handle<v8::Value> isNumber(const v8::Arguments& args);

 private:
  SimpleV();
  ~SimpleV();
};

#endif
```

SimpleVerify.cc (http://SimpleVerify.cc)

```
#define BUILDING_NODE_EXTENSION
#include <node.h>
#include "SimpleVerify.h"

using namespace v8;

SimpleV::SimpleV() {};
SimpleV::~SimpleV() {};

Handle<Value> SimpleV::isArray(const Arguments& args) {//定义是否是数组
  HandleScope scope;
  return scope.Close(Boolean::New(args[0]->IsArray()));
}

Handle<Value> SimpleV::isNumber(const Arguments& args) {//定义是否是数字
  HandleScope scope;
  return scope.Close(Boolean::New(args[0]->IsNumber()));
}
```

代码很简单，看完官网的例子和初步了解V8手册以后很容易看懂，不多解释了。
目前SimpleV这个类只有判断是否是数组和是否是数字2个方法，这2个方法都可以在v8手册上看到，当然我打算继续完善这个模块，根据node-validate的功能山寨一把，全部搬成C++的。
估计有同学说，你费这么大劲搞c++的验证模块，到底效果如何？有用吗？
数据说明一切，下面我们针对这2种情况做一下简单的测试，测试代码如下：

回到顶部

```
var sv = module.exports = require('./lib/verify.js');

//利用纯js语法进行判断
var ia = Array.isArray;
console.time('js');
for(var j=0;j<1000;j++){
 ia([]);
 ia('111');
 isNaN(123);
 isNaN('abc');
}
console.timeEnd('js')


//利用c++模块进行判断
var ia2 = sv.isArray;
var in2 = sv.isNumber;
console.time('c++');
for(var j=0;j<1000;j++){
 ia2([]);
 ia2('111');
 in2(123);
 in2('abc');
}
console.timeEnd('c++');
```

测试环境：linux redhat虚拟机，不多介绍了，具体测试结果可能跟机器配置有关系
我们看一下多次测试的结果：

```
js: 10ms
c++: 3ms


js: 9ms
c++: 1ms


js: 7ms
c++: 2ms


js: 8ms
c++: 0ms


js: 6ms
c++: 0ms
```

我们可以看到尽管我们利用了js原生的isNaN和isArray这2个方法，但是从性能上来说，无疑C++验证的更快一些。
可能有同学说循环1000次是否有些过了，但是我最近打算用node做接口服务，所以同时有100个接口被查询，每个接口有5-10个参数被验证的情况是可能存在的，如果我们使用c++的addon验证模块可以至少快5ms，当然如果一些正则等等判断可能差距的更多。

最后补一个小教训，我在一开始先创建了binding.gyp，没有把SimpleVerify.cc文件包含进去，导致每次编译都通过，但是调用一直报错，搞了好一阵子才发现，给大家提个醒啊，不要像我这么笨了，哎~

感谢小型笨蛋的系列文章让我学到很多，javascript里有个C (http://cnodejs.org/topic/4f16442ccae1f4aa270010c5)
google v8开发手册地址 (http://bespin.cz/~ondras/html/classv8_1_1Value.html#a70d4afaccc7903e6a01f40a46ad04188)
博客原文，用C++写node（一） (http://snoopyxdy.blog.163.com/blog/static/60117440201210239134461 7/)
博客原文，用C++写node（四） (http://snoopyxdy.blog.163.com/blog/static/60117440201210309211468 2/)
里面有一些官网实例的代码分析，不知道对不对，希望指教啊
最后发现，不知觉我的粉丝有100啦，哈哈，大家多多关注啊

5 回复