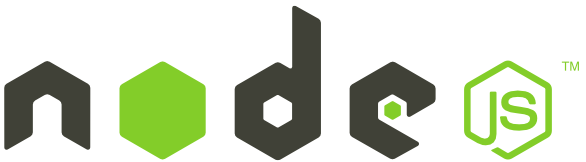
[Node.js开发Web后台服务](http://www.cnblogs.com/best/p/6204116.html)

**目录**

* [一、简介](http://www.cnblogs.com/best/p/6204116.html#_label0)
* [二、搭建Node.js开发环境](http://www.cnblogs.com/best/p/6204116.html#_label1)
  + [2.1、安装Node.js](http://www.cnblogs.com/best/p/6204116.html#_lab2_1_0)
  + [2.2、安装IDE开发Node.js插件](http://www.cnblogs.com/best/p/6204116.html#_lab2_1_1)
* [三、第一个Node.js程序](http://www.cnblogs.com/best/p/6204116.html#_label2)
* [四、NPM（Node.js包管理器）](http://www.cnblogs.com/best/p/6204116.html#_label3)
  + [4.3、安装模块](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_0)
  + [4.4、卸载模块](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_1)
  + [4.5、模块列表](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_2)
  + [4.6、更新模块](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_3)
  + [4.7、搜索模块](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_4)
  + [4.9、更换NPM 镜像](http://www.cnblogs.com/best/p/6204116.html#_lab2_3_5)
* [五、Express](http://www.cnblogs.com/best/p/6204116.html#_label4)
  + [5.2、Express框架核心特性](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_0)
  + [5.3、安装 Express](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_1)
  + [5.4、第一个Express框架实例](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_2)
  + [5.5、使用Nodeclipse开发Express项目](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_3)
  + [5.6、ejs基础](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_4)
  + [5.7、lodash](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_5)
  + [5.8、参数](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_6)
    - [5.8.1、URL中的参数占位](http://www.cnblogs.com/best/p/6204116.html#_label3_4_6_0)
    - [5.8.2、URL中的QueryString](http://www.cnblogs.com/best/p/6204116.html#_label3_4_6_1)
    - [5.8.3、HTTP正文中的参数](http://www.cnblogs.com/best/p/6204116.html#_label3_4_6_2)
  + [5.9、JSON](http://www.cnblogs.com/best/p/6204116.html#_lab2_4_7)
* [六、RESTful（表述性状态转移）](http://www.cnblogs.com/best/p/6204116.html#_label5)
* [七、示例下载](http://www.cnblogs.com/best/p/6204116.html#_label6)

**一、简介**

Node.js 是一个基于Google Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。Node.js 的包管理器 npm，是全球最大的开源库生态系统。



能方便地搭建响应速度快、易于扩展的网络应用，Node.js 使用事件驱动， 非阻塞I/O 模型而得以轻量和高效，非常适合在分布式设备上运行的数据密集型的实时应用。

官网：[https://nodejs.org/en/](https://nodejs.org/en/" \t "_blank)  
中文：[https://cnodejs.org/](https://cnodejs.org/%20" \t "_blank)、[http://nodejs.cn/](http://nodejs.cn/" \t "_blank)

API：[http://nodeapi.ucdok.com/#/api/](http://nodeapi.ucdok.com/" \l "/api/" \t "_blank)

简单说Node.js就是运行在服务器端的JavaScript，是现在流行的语言中能同时运行在前端与后台的程序语言，你可以把JavaScript想像成Java与C#。相关技术：

数据库：MongoDB，非关系型数据库，NoSQL（Not only SQL）

MVC框架：AngularJS

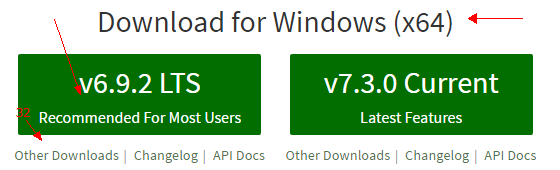
Web服务器：Express

模板引擎：jade、ejs、htmljs、swig、hogan.js

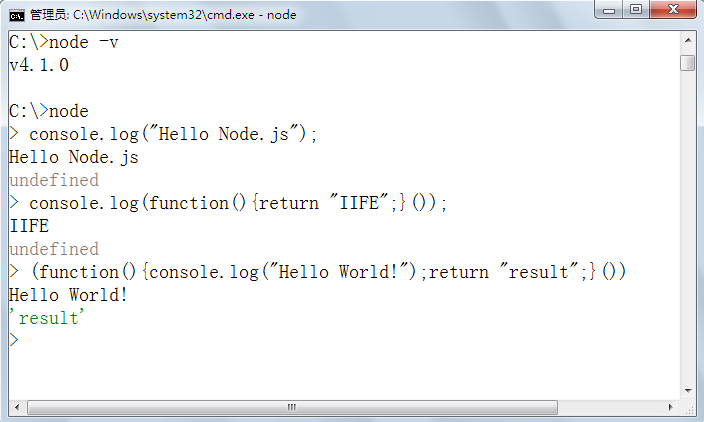
**二、搭建Node.js开发环境**

**2.1、安装Node.js**

[去官网下下载最新版本的](https://nodejs.org/en/)Node.js一步一步按提示安装即可，如果安装失败就手动安装，将Node.js的安装位置配置到环境变量的path中。



安装完成后启动命令行，测试：



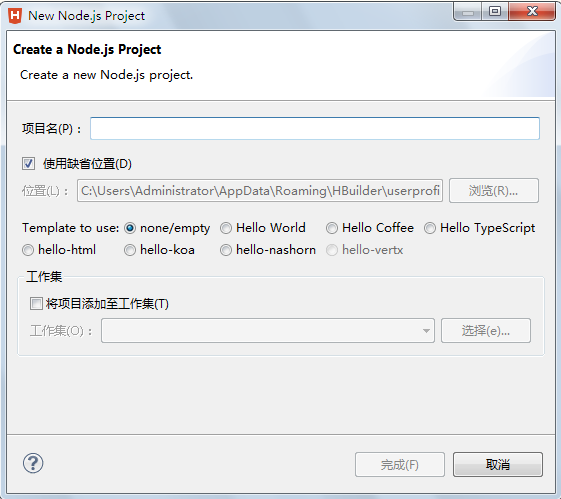
**2.2、安装IDE开发Node.js插件**

如果不使用IDE开发项目效率较低，在很多主流的集成开发环境（IDE）中都可以安装插件支持Node.js开发，如Eclipse，这里我们以HBuilder为例：

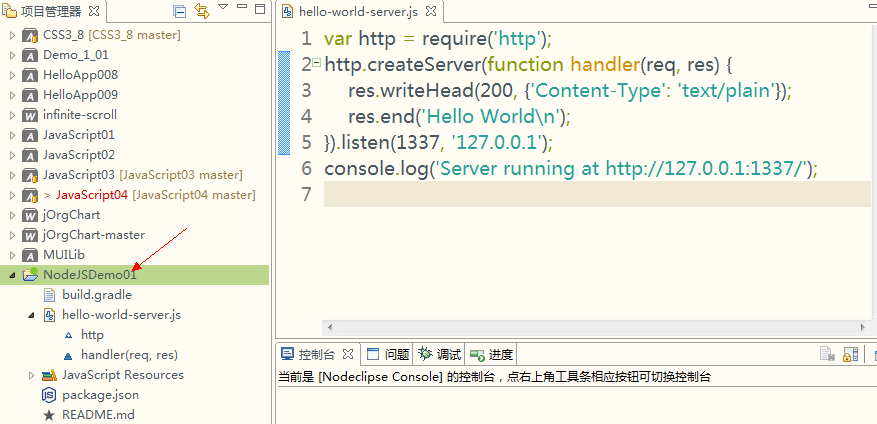
启动HBuilder->工具->插件安装



安装成功后就可以新建Node.js项目了：

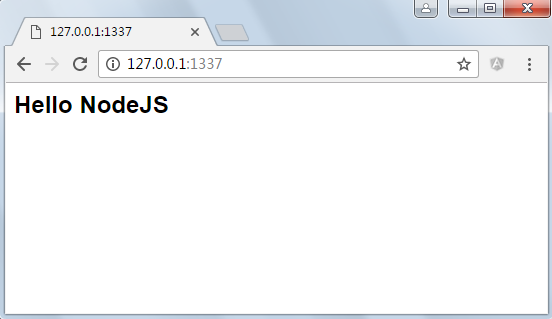


这里选择Hello World，新建好的项目如下：



hello-world-server.js文件就是一个简单的web服务器，右键选择“运行方式”->"Node Application"

控制台提示“Server running at http://127.0.0.1:1337/”在浏览器查看的效果如下：



**三、第一个Node.js程序**

在上面的示例中，我们是通过IDE完成编译与运行的，其实手动运行也可以，比如编写一段代码如下：

server.js

[复制代码](javascript:void(0);)

//依赖一个http模块，相当于java中的import，与C#中的using

var http = require('http');

//创建一个服务器对象

server = http.createServer(function (req, res) {

//设置请求成功时响应头部的MIME为纯文本

res.writeHeader(200, {"Content-Type": "text/plain"});

//向客户端输出字符

res.end("Hello World\n");

});

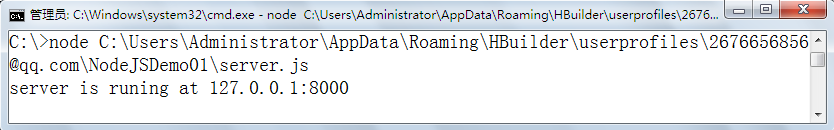
//让服务器监听本地8000端口开始运行

server.listen(8000,'127.0.0.1');

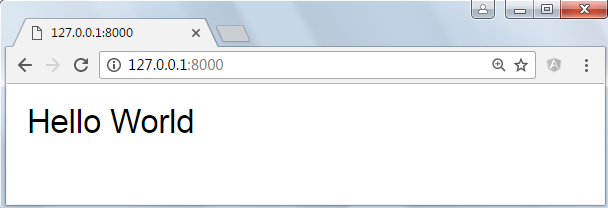
console.log("server is runing at 127.0.0.1:8000");

[复制代码](javascript:void(0);)

在node环境下解释运行：



运行结果：



引入 required 模块：我们可以使用 require 指令来载入 Node.js 模块。  
创建服务器：服务器可以监听客户端的请求，类似于TomCat、IIS、Apache 、Nginx 等 HTTP 服务器。  
接收请求与响应请求 服务器很容易创建，客户端可以使用浏览器或终端发送 HTTP 请求，服务器接收请求后返回响应数据。

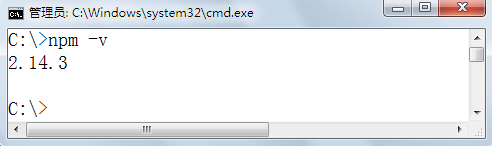
第一行请求（require）Node.js 自带的 http 模块，并且把它赋值给 http 变量。  
接下来我们调用 http 模块提供的函数： createServer 。这个函数会返回 一个对象，这个对象有一个叫做 listen 的方法，这个方法有一个数值参数， 指定这个 HTTP 服务器监听的端口号。

**四、NPM（Node.js包管理器）**

NPM是随同NodeJS一起安装的包管理工具，能解决NodeJS代码部署上的很多问题，常见的使用场景有以下几种：  
a)、允许用户从NPM服务器下载别人编写的第三方包到本地使用。  
b)、允许用户从NPM服务器下载并安装别人编写的命令行程序到本地使用。  
c)、允许用户将自己编写的包或命令行程序上传到NPM服务器供别人使用。

官网：<https://www.npmjs.com/>

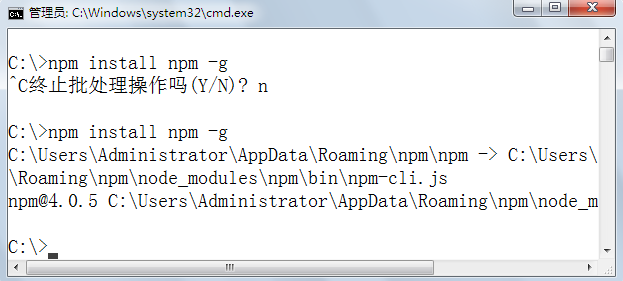
4.1、查看npm版本  
由于新版的nodejs已经集成了npm，所以之前npm也一并安装好了。同样可以通过输入 "npm -v" 来测试是否成功安装。命令如下，出现版本提示表示安装成功:



4.2、升级npm

如果你安装的是旧版本的 npm，可以很容易得通过 npm 命令来升级

npm install npm -g

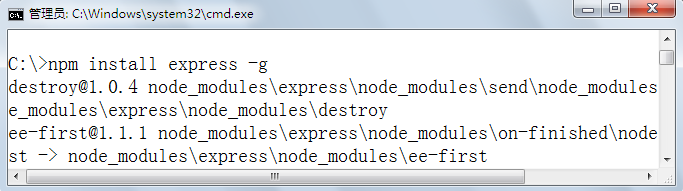


**4.3、安装模块**

npm install <Module Name> -参数

如果带参数-g表示全局安装，否则只是安装到某个目录下。

以下实例，我们使用 npm 命令安装常用的 Node.js web框架模块 express



**4.4、卸载模块**

我们可以使用以下命令来卸载 Node.js 模块。  
npm uninstall <Module Name>

如先使用安装指令安装bootstrap:

npm install bootstrap

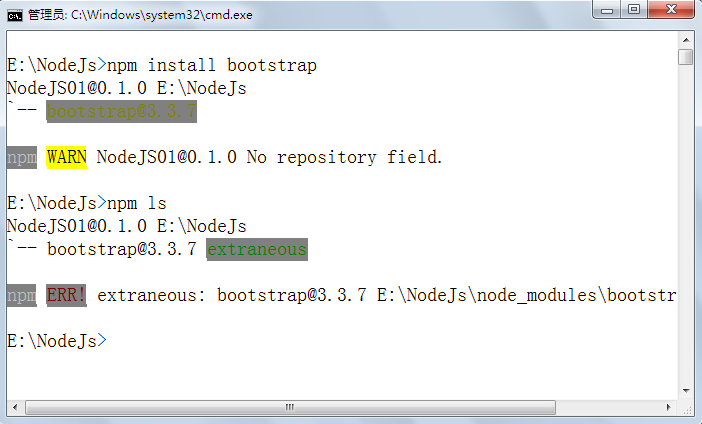
再使用卸载指令删除模块：

npm uninstall bootstrap

可以到 /node\_modules/ 目录下查看包是否还存在

**4.5、模块列表**

使用模块列表命令可以方便的看到当前项目中依赖的包：  
npm ls



**4.6、更新模块**

我们可以使用以下命令更新模块：  
npm update 模块名称  
npm up -g 模块名称

**4.7、搜索模块**

npm search 模块名称

4.8、NPM 常用命令  
除了本章介绍的部分外，NPM还提供了很多功能，package.json里也有很多其它有用的字段。  
除了可以在npmjs.org/doc/查看官方文档外，这里再介绍一些NPM常用命令。  
NPM提供了很多命令，例如install和publish，使用npm help可查看所有命令。  
NPM提供了很多命令，例如install和publish，使用npm help可查看所有命令。  
使用npm help <command>可查看某条命令的详细帮助，例如npm help install。  
在package.json所在目录下使用npm install . -g可先在本地安装当前命令行程序，可用于发布前的本地测试。  
使用npm update <package>可以把当前目录下node\_modules子目录里边的对应模块更新至最新版本。  
使用npm update <package> -g可以把全局安装的对应命令行程序更新至最新版。  
使用npm cache clear可以清空NPM本地缓存，用于对付使用相同版本号发布新版本代码的人。  
使用npm unpublish <package>@<version>可以撤销发布自己发布过的某个版本代码。

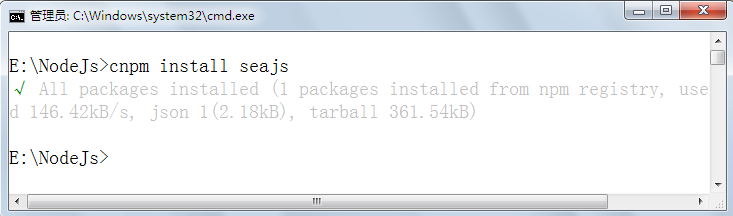
**4.9、更换NPM 镜像**

因为npm的服务器在国外，在网络状态不好的情况下引入一个模块会因为网络延迟而失败，可以更换成国内速度更快的镜像服务器，这里以使用淘宝 NPM 镜像（http://npm.taobao.org/）为例：

npm install -g cnpm --registry=https://registry.npm.taobao.org



这样就可以使用 cnpm 命令来安装模块了：  
$ cnpm install [name]



这是一个完整 npmjs.org 镜像，你可以用此代替官方版本(只读)，同步频率目前为 10分钟 一次以保证尽量与官方服务同步。

**五、Express**

Express 是一个简洁而灵活的 node.js Web应用框架, 提供了一系列强大特性帮助你创建各种 Web 应用，和丰富的 HTTP 工具。  
使用 Express 可以快速地搭建一个完整功能的网站。使用Node.js作为AngularJS开发Web服务器的最佳方式是使用Express模块。

Express官网： http://expressjs.com/

Express4.x API：http://expressjs.com/zh-cn/4x/api.html



**5.2、Express框架核心特性**

可以设置中间件来响应 HTTP 请求。

定义了路由表用于执行不同的 HTTP 请求动作。

可以通过向模板传递参数来动态渲染 HTML 页面。

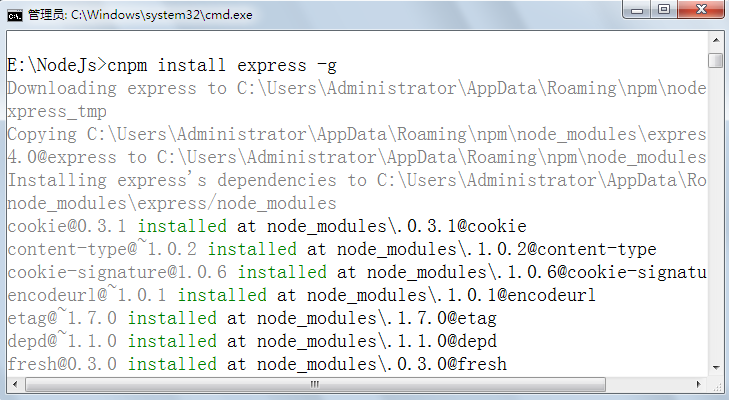
丰富的 HTTP 快捷方法和任意排列组合的 Connect 中间件，让你创建健壮、友好的 API 变得既快速又简单。

Express 不对 Node.js 已有的特性进行二次抽象，我们只是在它之上扩展了 Web 应用所需的基本功能。

**5.3、安装 Express**

安装 Express 并将其保存到依赖列表中：

npm install express --save



以上命令全局安装express。也可安装时指定安装中间件。

body-parser - node.js 中间件，用于处理 JSON, Raw, Text 和 URL 编码的数据。  
cookie-parser - 这就是一个解析Cookie的工具。通过req.cookies可以取到传过来的cookie，并把它们转成对象。  
multer - node.js 中间件，用于处理 enctype="multipart/form-data"（设置表单的MIME编码）的表单数据。

npminstallbody−parser−−save npm install cookie-parser --save  
$ npm install multer --save

默认这些模块都已经添加。

**5.4、第一个Express框架实例**

接下来我们使用 Express 框架来输出 "Hello World"。  
以下实例中我们引入了 express 模块，并在客户端发起请求后，响应 "Hello World" 字符串。

创建一个目录，如Project，进入命令行：

使用npm install express 导入express模块。

在目录下创建hello.js文件，如下所示：

[复制代码](javascript:void(0);)

//引入express模块

var express = require('express');

//创建一个app对象，类似一个web 应用（网站）

var app = express();

//接受指定路径的请求，指定回调函数

app.get('/', function (req, res){

res.send('Hello World');

});

//创建一个web服务器，可以认为就是web服务器对象

//监听8081端口，当监听成功时回调

var server = app.listen(8081, function () {

var host = server.address().address; //地址

var port = server.address().port; //端口

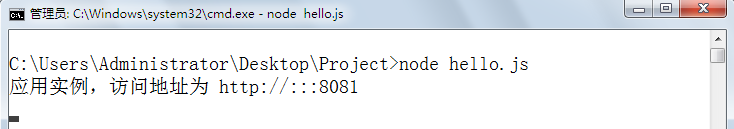
console.log("应用实例，访问地址为 http://%s:%s", host, port);

});

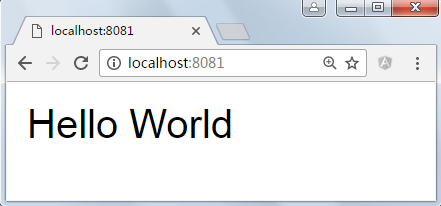
})

[复制代码](javascript:void(0);)

使用node执行js：

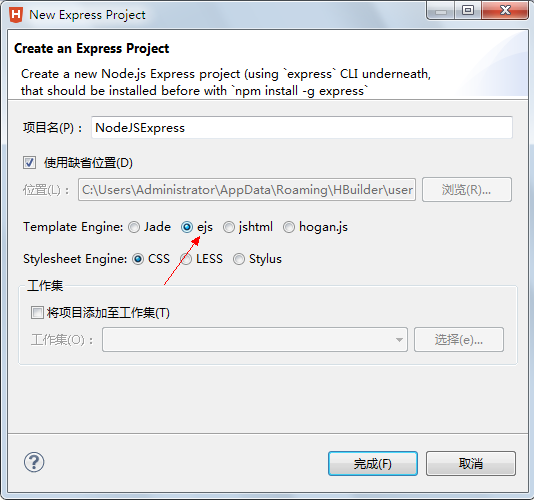


运行结果：

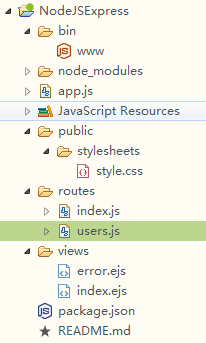


**5.5、使用Nodeclipse开发Express项目**

如果直接使用记事本效率会不高，nodeclipse插件可以方便的创建一个Express项目，步骤如下：



创建好的项目如下：



app.js是网站：

[复制代码](javascript:void(0);)

var express = require('express');

var path = require('path');

var favicon = require('serve-favicon');

var logger = require('morgan');

var cookieParser = require('cookie-parser');

var bodyParser = require('body-parser');

var index = require('./routes/index');

var users = require('./routes/users');

var app = express();

//指定视图引擎为ejs

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', index);

app.use('/users', users);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

var err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

[复制代码](javascript:void(0);)

bin\www是web服务器：

[复制代码](javascript:void(0);)

#!/usr/bin/env node

/\*\*

\* 依赖模块，导入

\*/

var app = require('../app');

var debug = require('debug')('nodejsexpress:server');

var http = require('http');

/\*\*

\* 从上下文环境中获得监听端口，如果空则3000

\*/

var port = normalizePort(process.env.PORT || '3000');

app.set('port', port);

/\*\*

\* 创建Web服务器

\*/

var server = http.createServer(app);

/\*\*

\* 开始监听

\*/

server.listen(port);

server.on('error', onError); //指定发生错误时的事件

server.on('listening', onListening); //当监听成功时的回调

/\*\*

\* 规范化端口

\*/

function normalizePort(val) {

var port = parseInt(val, 10);

if (isNaN(port)) {

// named pipe

return val;

}

if (port >= 0) {

// port number

return port;

}

return false;

}

/\*\*

\*错误事件监听

\*/

function onError(error) {

if (error.syscall !== 'listen') {

throw error;

}

var bind = typeof port === 'string'

? 'Pipe ' + port

: 'Port ' + port;

//错误处理

switch (error.code) {

case 'EACCES':

console.error(bind + ' requires elevated privileges');

process.exit(1); //结束程序

break;

case 'EADDRINUSE':

console.error(bind + ' is already in use');

process.exit(1);

break;

default:

throw error;

}

}

/\*\*

\* 当用户访问服务器成功时的回调

\*/

function onListening() {

var addr = server.address();

var bind = typeof addr === 'string'

? 'pipe ' + addr

: 'port ' + addr.port;

debug('Listening on ' + bind);

}

[复制代码](javascript:void(0);)

routers/index.js路由，有点类似控制器或Servlet：

[复制代码](javascript:void(0);)

var express = require('express');

var router = express.Router();

/\* 获得首页 \*/

router.get('/', function(req, res, next) {

res.render('index', { title: 'Express' });

});

module.exports = router;

[复制代码](javascript:void(0);)

views/index.ejs首页视图：

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<title><%= title %></title>

<link rel='stylesheet' href='/stylesheets/style.css' />

</head>

<body>

<h1><%= title %></h1>

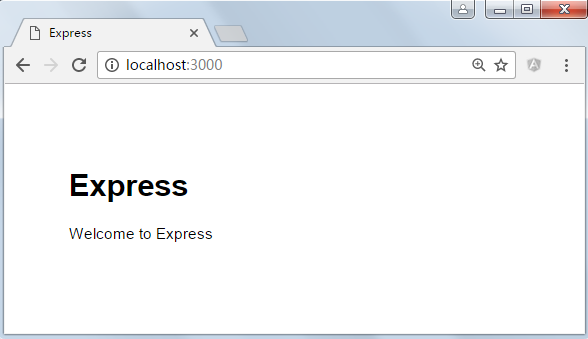
<p>Welcome to <%= title %></p>

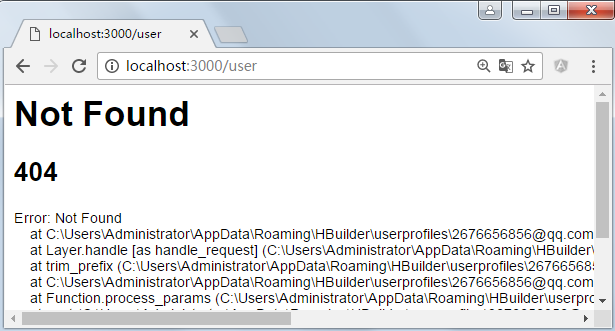
</body>

</html>

[复制代码](javascript:void(0);)

在www上右键选择“运行方式”->“Node Application”运行结果：





**5.6、ejs基础**

ejs是一个Express Web应用的模板引擎，在NodeJS开发中可以选择的模板引擎可能是所有Web应用开发中范围最广的，如jade、ejs、htmljs、swig、hogan.js，但ejs是最容易上手的，与jsp,asp,php的原始模板引擎风格很像。

官网：<http://www.embeddedjs.com/>

添加一个product.js路由：

[复制代码](javascript:void(0);)

var express = require('express');

var router = express.Router();

/\* 产品 \*/

router.get('/', function(req, res, next) {

var products=[];

products.push({name:"ZTE U880",price:899.8});

products.push({name:"HuWei 荣耀8",price:1899.8});

products.push({name:"iPhone 7 Plus 128G",price:5899.8});

//将product视图与指定的对象渲染后输出到客户端

res.render('product', { title: '天狗商城', pdts:products});

});

module.exports = router;

[复制代码](javascript:void(0);)

在views目录下添加product.ejs视图，这里是一个简单的MVC：

[复制代码](javascript:void(0);)

<!DOCTYPE html>

<html>

<head>

<title>

<%= title %>

</title>

<link rel='stylesheet' href='/stylesheets/style.css' />

</head>

<body>

<h1><%= title %> - 产品列表</h1>

<table border="1" width="80%">

<tr>

<th>序号</th>

<th>名称</th>

<th>价格</th>

</tr>

<%pdts.forEach(function(pdt,index){%>

<tr>

<td>

<%=index+1%>

</td>

<td>

<%=pdt.name%>

</td>

<td>

<%=pdt.price%>

</td>

</tr>

<%});%>

</table>

<ul>

<% for(var i=0; i<pdts.length; i++) {%>

<li>

<%=pdts[i].name%>

</li>

<% } %>

</body>

</html>

[复制代码](javascript:void(0);)

修改app，注册定义好的模块product：

[复制代码](javascript:void(0);)

var index = require('./routes/index');

var users = require('./routes/users');

var pdts = require('./routes/product');

var app = express();

//指定视图引擎为ejs

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', index);

app.use('/users', users);

app.use('/pdt', pdts);

[复制代码](javascript:void(0);)

运行结果：



**5.7、lodash**

这是一个具有一致接口、模块化、高性能等特性的 JavaScript 工具库。可以非常方便的操作json。

官网：<http://lodashjs.com/>

安装：

npm i -g npm

npm i --save lodash

安装时先用cd切换到当前项目下。

如果浏览器使用可以直接引入：

<script src="lodash.js"></script>

后台Node.js使用，可以引入模块：

[复制代码](javascript:void(0);)

//导入lodash模块

var \_= require('lodash');

var products=[];

products.push({name:"ZTE U880",price:899.8});

products.push({name:"HuWei 荣耀8",price:1899.8});

products.push({name:"iPhone 7 Plus 128G",price:5899.8});

//1、取出第一个元素

var obj1=\_.first(products);

console.log(obj1.name); //ZTE U880

//2、取出最后一个元素

var obj2=\_.last(products);

console.log(obj2.name); //iPhone 7 Plus 128G

//3、指定查找条件返回符合条件的索引

var obj3=\_.findIndex(products,function(obj){

return obj.price>=1000&&obj.name.indexOf("7")>0;

});

console.log(obj3); //2

//4、指定查找条件返回查找到的对象

var obj4=\_.find(products,function(obj){

return obj.price>=1000&&obj.name.indexOf("7")>0;

});

console.log(obj4); //{ name: 'iPhone 7 Plus 128G', price: 5899.8 }

//5、排序

var obj5=\_.orderBy(products,["price","name"],["desc","asc"]);

console.log(obj5);

//[ { name: 'iPhone 7 Plus 128G', price: 5899.8 },

//{ name: 'HuWei 荣耀8', price: 1899.8 },

//{ name: 'ZTE U880', price: 899.8 } ]

//6、查找价格为1899.8的产品的key

var obj6=\_.findKey(products,{price:1899.8});

console.log(obj6); //1

[复制代码](javascript:void(0);)

API的使用非常简单，但需要注意版本，可以现查现用，API地址：[https://lodash.com/docs/4.17.2](https://lodash.com/docs/4.17.2%20" \t "_blank)

**5.8、参数**

**5.8.1、URL中的参数占位**

Checks route params (req.params), ex: /user/:id

127.0.0.1:3000/index，这种情况下，我们为了得到index，我们可以通过使用req.params得到，通过这种方法我们就可以很好的处理Node中的路由处理问题，同时利用这点可以非常方便的实现MVC模式；

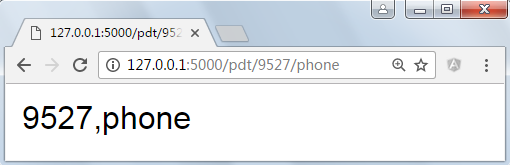
//获得产品根据Id

router.get('/:id/:category',function(request,res,next){

res.send(request.params.id+","+request.params.category);

});

运行结果：



**5.8.2、URL中的QueryString**

Checks query string params (req.query), ex: ?id=12

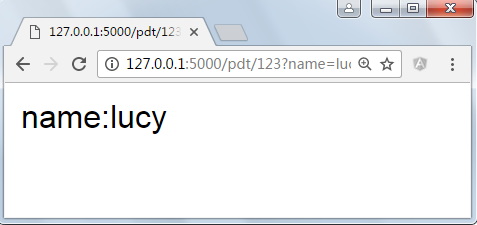
127.0.0.1:3000/index?id=12，这种情况下，这种方式是获取客户端get方式传递过来的值，通过使用req.query.id就可以获得，类似于PHP的get方法；

router.get('/:id',function(request,res,next){

res.send("name:"+request.query.name);

});

运行结果：



**5.8.3、HTTP正文中的参数**

 在post请求中获得表单中的数据。

Checks urlencoded body params (req.body), ex: id=

127.0.0.1：300/index，然后post了一个id=2的值，这种方式是获取客户端post过来的数据，可以通过req.body.id获取，类似于PHP的post方法；

http://images2015.cnblogs.com/blog/63651/201612/63651-20161222150412682-243382356.png

页面：

http://images.cnblogs.com/OutliningIndicators/ContractedBlock.gif View Code

代码：

[复制代码](javascript:void(0);)

router.post('/add',function(request,res,next){

var entity={name:request.body.name,price:request.body.price};

products.push(entity);

//将product视图与指定的对象渲染后输出到客户端

res.render('product', { title: '天狗商城', pdts:products,msg:"添加成功"});

});

[复制代码](javascript:void(0);)

结果：



**5.9、JSON**

如果需要Node.js向外提供返回JSON的接口，Express也是非常方便的，可以使用原来在浏览器中使用到的JSON对象，这是一个浏览器内置对象在服务可以直接使用：

将对象序列化成字符：

[复制代码](javascript:void(0);)

//对象

var rose={"name":"Rose","weight":"65"};

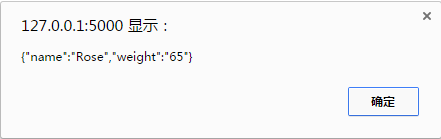
//序列化成字符串

var str=JSON.stringify(rose);

alert(str);

[复制代码](javascript:void(0);)

结果：



反序列化，将字符转换成对象：

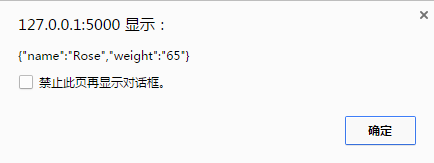
//将字符串转换成JavaScript对象

var markStr='{"name":"mark","weight":"188"}';

var mark=JSON.parse(markStr);

alert(mark.name+"，"+mark.weight);

结果：



Express已经封装了一个json方法，直接调用该方法就可以序列化对象：

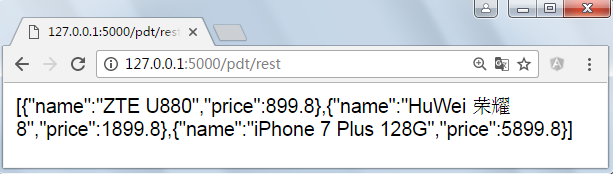
/\* 产品 \*/

router.get('/rest', function(req, res, next) {

res.json(products);

});

运行结果：



练习：完成一个图书管理的功能，图书包含（编号，名称，作者，图片，价格），实现：

a)、非AJAX的CRUD，使用Node.js+Express+ejs的动态技术。

b)、AJAX的CRUD，使用Node.js+Express+jQuery+HTML技术实现。

c)、使用RestFul风格的服务完成第个作业，get,post,delete,put请。

**六、RESTful（表述性状态转移）**

REST是英文Representational State Transfer的缩写，中文称之为“表述性状态转移”  
基于HTTP协议  
是另一种服务架构  
传递是JSON、POX（Plain Old XML）而不是SOAP格式的数据  
充分利用HTTP谓词（Verb）  
侧重数据的传输，业务逻辑交给客户端自行处理

REST是一种分布式服务架构的风格约束，像Java、.Net（WCF、WebAPI）都有对该约束的实现，使URL变得更加有意义，更加简洁明了，如：

http://www.zhangguo.com/products/1 get请求 表示获得所有产品的第1个

http://www.zhangguo.com/products/product post请求 表示添加一个产品

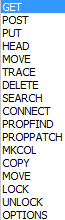
http://www.zhangguo.com/products/1/price get请求 表示获得第1个产品的价格

http://www.zhangguo.com/products/1 delete请求 删除编号为1的产品

REST设计需要遵循的原则  
网络上的所有事物都被抽象为资源（resource）；  
每个资源对应一个唯一的资源标识符（resource identifier）；  
通过通用的连接器接口（generic connector interface）对资源进行操作；  
对资源的各种操作不会改变资源标识符；  
所有的操作都是无状态的（stateless）

谓词  
GET  
表示查询操作，相当于Retrieve、Select操作  
POST  
表示插入操作，相当于Create，Insert操作  
PUT  
表示修改操作，相当于Update操作  
DELETE  
表示删除操作，相当于Delete操作

其它还有：



NodeJS+Express可以很容易的实现REST

application/x-www-form-urlencoded

multipart/form-data

application/json

res.setHeader('Content-Type', 'application/json;charset=utf-8');

示例代码cars.js：

[复制代码](javascript:void(0);)

var express = require('express');

var router = express.Router();

var \_= require('lodash');

var cars=[];

cars.push({id:201701,name:"BMW",price:190,speed:"210km/h",color:"白色"});

cars.push({id:201702,name:"BYD",price:25,speed:"160km/h",color:"红色"});

cars.push({id:201703,name:"Benz",price:300,speed:"215km/h",color:"蓝色"});

cars.push({id:201704,name:"Honda",price:190,speed:"170km/h",color:"黑色"});

cars.push({id:201705,name:"QQ",price:130,speed:"210km/h",color:"白色"});

/\* Get \*/

/\*获得所有汽车\*/

/\*url /cars/\*/

router.get('/', function(req, res, next) {

res.json(cars);

});

/\*Get\*/

/\*获得汽车通过id\*/

/\*url：/cars/:id \*/

router.get('/:id', function(req, res, next) {

//从路径中映射参数，转换成数字

var id=parseInt(req.params.id);

var car=\_.find(cars,{id:id});

res.json(car);

});

/\*Post\*/

/\*添加汽车\*/

/\*url：/cars/car \*/

router.post('/car', function(req, res, next) {

var car=req.body; //从请求正文中获得json对象

car.id=\_.last(cars).id+1; //将编号修改为最后一辆车的编号+1

cars.push(car); //将汽车对象添加到集合中

res.json(car); //将添加成功的车以json的形式返回

});

/\*Put\*/

/\*修改汽车\*/

/\*url：/cars/car \*/

router.put('/car', function(req, res, next) {

var car=req.body; //从请求正文中获得json对象

console.log(req.body);

var index=\_.findIndex(cars,{id:parseInt(car.id)}); //根据id获得车在集合中的下标

cars[index]=car; //替换原对象

//res.json(car); //将修改后的车以json的形式返回

res.send({status:"success", message:"更新成功!"});

});

/\*Delete\*/

/\*删除汽车\*/

/\*url：/cars/:id \*/

router.delete('/id/:id', function(req, res, next) {

//获得url中的编号参数

var id=parseInt(req.params.id);

var index=\_.findIndex(cars,{id:id}); //根据id获得车在集合中的下标

cars.splice(index,1); //在cars数组中删除下标从index开始的1条数据

res.send({status:"success", message:"删除成功!"});

});

module.exports = router;

[复制代码](javascript:void(0);)

示例代码app.js：

[复制代码](javascript:void(0);)

var express = require('express');

var path = require('path');

var favicon = require('serve-favicon');

var logger = require('morgan');

var cookieParser = require('cookie-parser');

var bodyParser = require('body-parser');

var index = require('./routes/index');

var users = require('./routes/users');

var pdts = require('./routes/product');

var task = require('./routes/task');

var cars = require('./routes/cars');

var app = express();

//指定视图引擎为ejs

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'ejs');

// uncomment after placing your favicon in /public

//app.use(favicon(path.join(\_\_dirname, 'public', 'favicon.ico')));

app.use(logger('dev'));

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

app.use('/', index);

app.use('/users', users);

app.use('/pdt', pdts);

app.use("/task",task);

app.use("/cars",cars);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

var err = new Error('Not Found');

err.status = 404;

next(err);

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

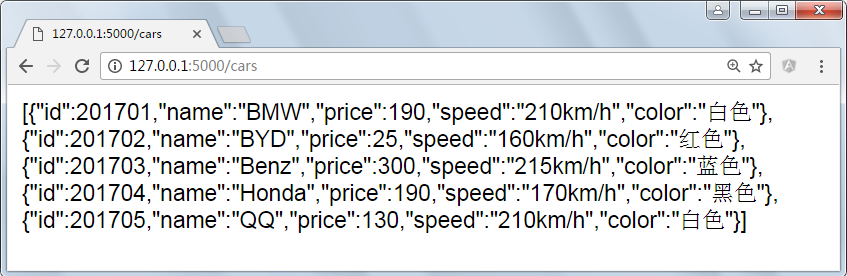
res.render('error');

});

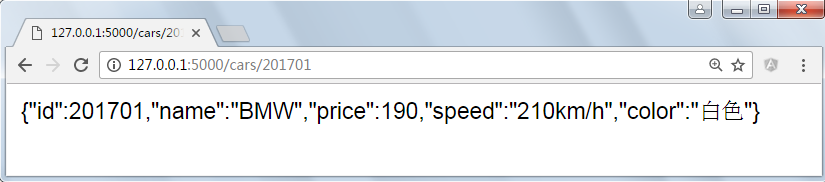
module.exports = app;

[复制代码](javascript:void(0);)

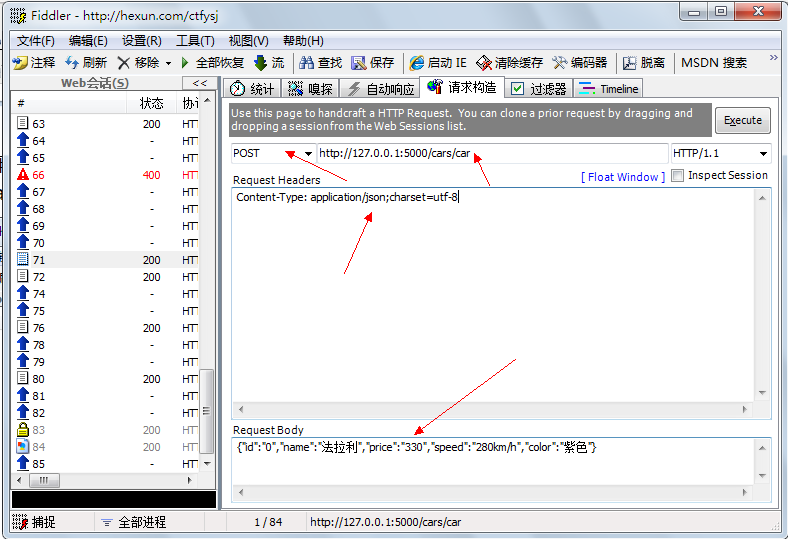
/\* Get \*/  
/\*获得所有汽车\*/  
/\*url /cars/\*/

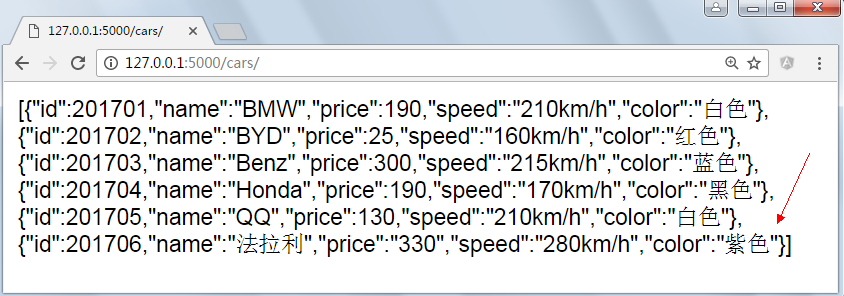


/\*Get\*/  
/\*获得汽车通过id\*/  
/\*url：/cars/:id \*/



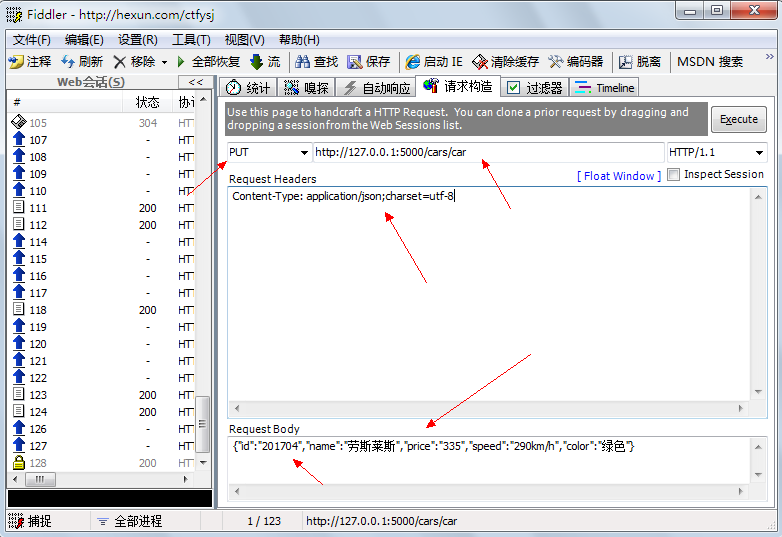
/\*Post\*/  
/\*添加汽车\*/  
/\*url：/cars/car \*/

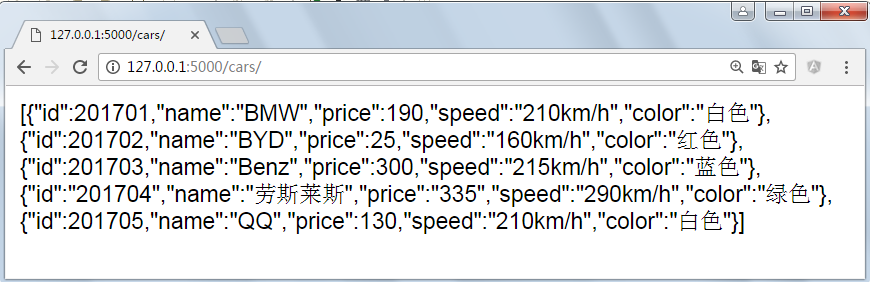




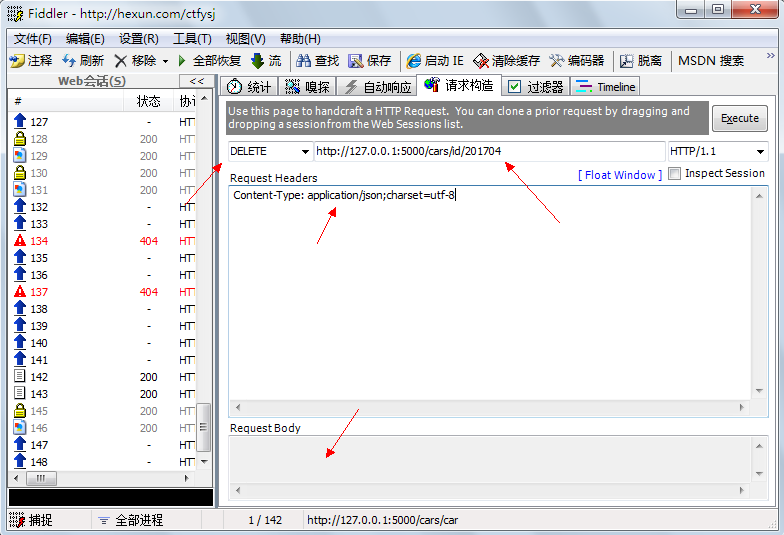
参数中的json格式一定要使用标准格式，注意引号，注意Content-Type，默认的Content-Type类型是：application/x-www-form-urlencoded

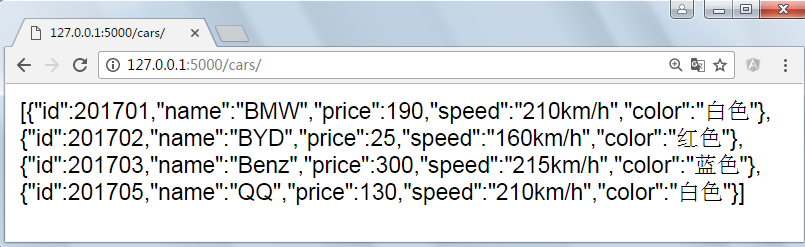
/\*Put\*/  
/\*修改汽车\*/  
/\*url：/cars/car \*/





/\*Delete\*/  
/\*删除汽车\*/  
/\*url：/cars/:id \*/





**七、示例下载**

git：[https://coding.net/u/zhangguo5/p/NodeJS001/git](https://coding.net/u/zhangguo5/p/NodeJS001/git" \t "_blank)

git：<https://coding.net/u/zhangguo5/p/NodeJSExpress/git>