# 实验5：http server

实验背景

利用已掌握的知识，成功安装node.js交互环境，并完成入门实例。如图所示：

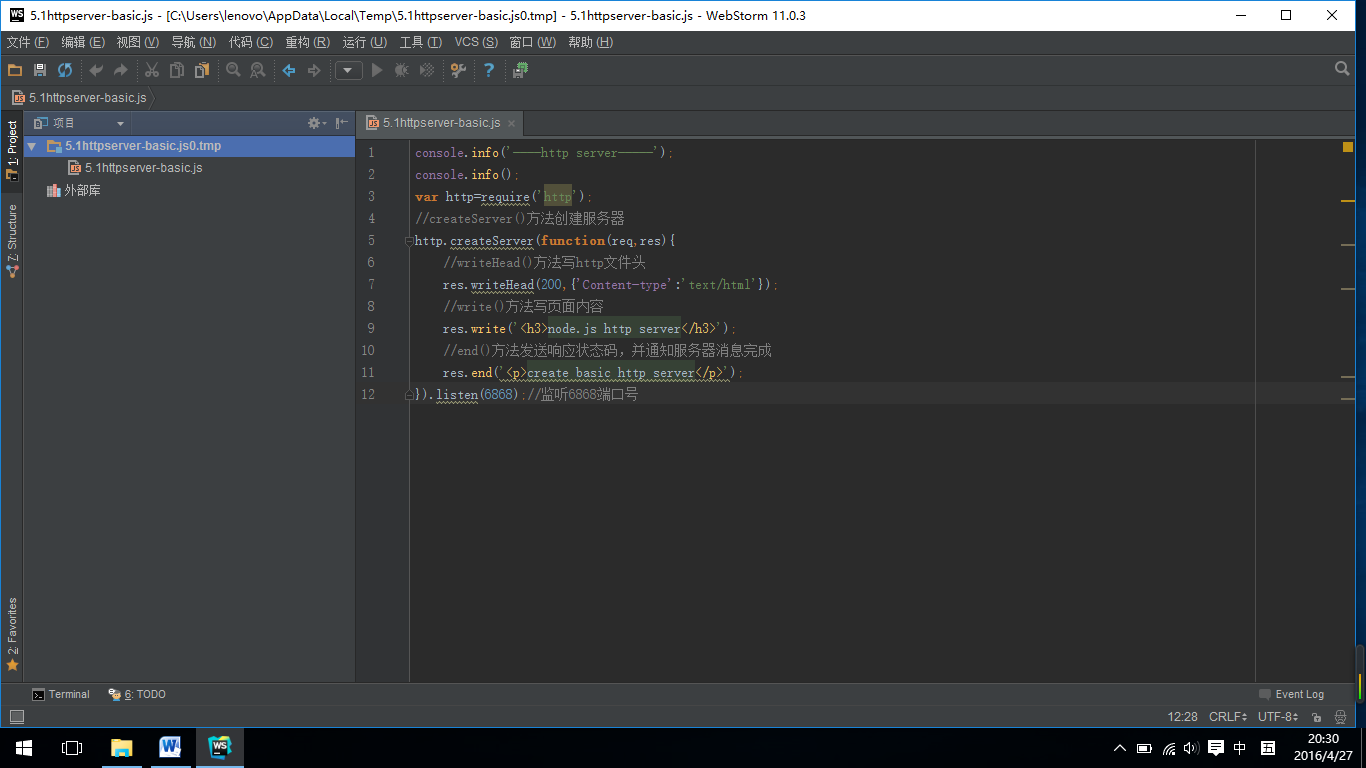


图1-1

实验目的

1．了解http和https

2．掌握http头文件

3．掌握http server

实验分析

本次实验要完成的有三个方面的知识。第一：完成案例《5.1 httpserver-basic》？第二：完成案例《5.2 httpserver-response》，第三：完成案例《5.3 req-statuscodes》。

实验步骤

1. 完成案例《5.1 httpserver-basic》

1.创建httpserver-basic.js文件，设计代码如下：

console.info('----http server-----');

console.info();

var http=require('http');

//createServer()方法创建服务器

http.createServer(function(req,res){

//writeHead()方法写http文件头

res.writeHead(200,{'Content-type':'text/html'});

//write()方法写页面内容

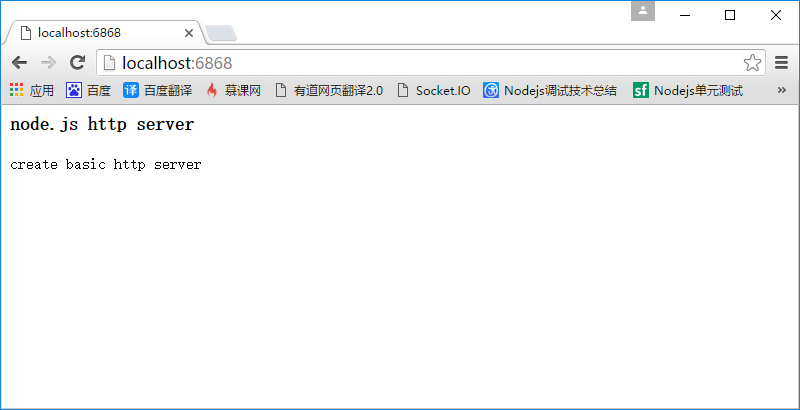
res.write('<h3>node.js http server</h3>');

//end()方法发送响应状态码，并通知服务器消息完成

res.end('<p>create basic http server</p>');

}).listen(6868);//监听6868端口号

2、运行调试代码：



1. 完成案例《5.2 httpserver-response》，

1.创建httpserver-response.js文件，设计代码如下：

console.info('-------创建http服务器测试request-------');  
console.info();  
**var** http=require('http');  
http.createServer(**function**(req,res){  
 res.writeHead(200,{'content-type':'text/html'});  
 res.write('<h3>这是node.js创建的 http服务器</h3>');  
 res.end('创建HTTP服务器对客户请求的响应！');  
}).listen(6868);

2.创建httpserver-request.js文件，设计代码如下：

console.info('----创建http客户端-----');  
console.info();  
**var** http=require('http');  
**var** options={  
 hostname:'localhost',  
 port:6868,  
 path:'/',  
 method:'post'  
};  
**var** req=http.request(options,**function**(res){  
 console.log('status:'+res.statusCode);  
 res.setEncoding('UTF-8');  
 res.on('data',**function**(chunk){  
 console.log('body:'+chunk);  
 });  
});  
req.on('error',**function**(e){  
 console.log('problem with request:'+ e.message);  
});  
req.end();

3、运行调试代码：

"C:\Program Files (x86)\JetBrains\WebStorm 11.0.3\bin\runnerw.exe" E:\nodejs\node.exe 5.2httpserver-request.js

----创建http客户端-----

status:200

body:<h3>这是node.js创建的 http服务器</h3>

body:创建HTTP服务器对客户请求的响应！

进程已结束,退出代码0

1. 完成案例《5.3 req-statuscodes》

1.创建req-statuscodes.js文件，设计代码如下：

console.info('------http status code-------');  
console.info();  
**var** http=require('http');//引用http模块  
console.log('now start http server');  
console.info();  
http.createServer(**function**(req,res){  
 //获取URL查询字段  
 **var** status=req.url.substr(1);  
 //获取响应状态码  
 **var** status\_codes=status.substring(12);  
 //验证响应状态码集合是否有效  
 **if**(!http.STATUS\_CODES[status\_codes]){  
 status\_codes='404';  
 }  
 //写http文件头  
 res.writeHeader(status\_codes,{'content-type':'text/plain;charset=utf-8'});  
 //发送响应状态码，并通知服务器消息完成  
 res.end(http.STATUS\_CODES[status\_codes]);  
}).listen(6868);

2.创建res-statuscodes.js文件，设计代码如下：

console.info('----http code----');  
console.info();  
**var** http=require('http');//引入http模块  
**var** querystring=require('querystring');//引入querystring模块  
//定义响应状态码数组  
**var** status\_codes=**new** Array();  
status\_codes[0]='201';  
status\_codes[1]='202';  
status\_codes[2]='203';  
status\_codes[3]='401';  
status\_codes[4]='404';  
//定义查询字段  
**var** postData=**new** Array();  
**for**(**var** n=0;n<5;n++){  
 postData[n]=status\_codes[n];  
}  
//模似http客户端向http服务器连续发出request请求  
**for**(**var** i=0;i<5;i++){  
 **var** options={  
 hostname:'localhost',  
 port:6868,  
 path:'/'+postData[i],  
 header:{  
 'Content-Type':'application/x-www-form-urlencoded',  
 'content-length':postData.length  
 }  
 };  
 **var** req=http.request(options,**function**(res){  
 console.log('status\_code:'+res.statusCode);  
 console.log('headers:'+JSON.stringify(res.headers));  
 console.info();  
 // res.setEncoding('uft-8');  
 res.on('data',**function**(chunk){  
 console.log('body:'+chunk);  
 console.info();  
 });  
 });  
 req.on('error',**function**(e){  
 console.log('problem with request:'+ e.message);  
 console.info();  
 });  
 req.write('\n');  
 req.end();  
}

3、运行调试代码：

"C:\Program Files (x86)\JetBrains\WebStorm 11.0.3\bin\runnerw.exe" E:\nodejs\node.exe 5.3req-statuscodes.js

----http code----

status\_code:404

headers:{"content-type":"text/plain;charset=utf-8","date":"Wed, 27 Apr 2016 12:43:39 GMT","connection":"close","transfer-encoding":"chunked"}

body:Not Found

status\_code:404

headers:{"content-type":"text/plain;charset=utf-8","date":"Wed, 27 Apr 2016 12:43:39 GMT","connection":"close","transfer-encoding":"chunked"}

body:Not Found

status\_code:404

headers:{"content-type":"text/plain;charset=utf-8","date":"Wed, 27 Apr 2016 12:43:39 GMT","connection":"close","transfer-encoding":"chunked"}

body:Not Found

status\_code:404

headers:{"content-type":"text/plain;charset=utf-8","date":"Wed, 27 Apr 2016 12:43:39 GMT","connection":"close","transfer-encoding":"chunked"}

body:Not Found

status\_code:404

headers:{"content-type":"text/plain;charset=utf-8","date":"Wed, 27 Apr 2016 12:43:39 GMT","connection":"close","transfer-encoding":"chunked"}

body:Not Found

进程已结束,退出代码0

扩展练习

1.思考并实现，完成案例《5.4http-response-writeheader》

参考学习资料：详见案例《5.4http-response-writeheader》源文件

实验思考

1.思考并实现，《5.4 http-response》。