好的，书接上回，在安装完成Node.js后。我们就可以开始使用它搭建简单的服务端程序了。

Node.js模块和包的使用

工程化的开发离不开项目之间的协作。在开始搭建服务端之前，我们有必要先简单认识一下Node.js的模块(Moudles)机制，这是Node.js进行项目间的封装和依赖调用的途径。

简单来说，就是可以开发我们自己的模块，打成包(package)发布出去，供他人使用。也可以在需要调用模块内容时，获取包含该模块的包并将引入到代码中。

那么引入其他模块时需要使用的语句是`require`，他的语法是：

```
require(模块名称)
```

他会返回该模块向外导出(exports)的对象或对象集合。

通常，需要定义一个变量来接受返回的对象，例如

```
var http = require('http')
```

有些模块是Node.js提供的模块，如`http`,`fs`,`path`这些，称作**核心模块**
还有更多的是由用户编写的模块，像我们即将使用的`express`,`body-parser`等，称作**文件模块**

文件模块并不直接存在于我们安装的Node.js中，需要我们在控制台中使用npm(Node Package Menager)工具进行安装，常用语法是

```
npm install <package-name>
```

例如，安装接下来会用到的`express`模块：

```
npm install express
```

（除了使用包名还可以使用url进行安装）

模块机制和npm的使用还有很多内容值得了解，不过这不是这篇博客的重点，也不需要在一开始就刨根问底。总之，就先开始主要内容吧！

---

http搭建服务端程序

使用Node.js搭建服务端程序，是它能够接收并处理http请求。

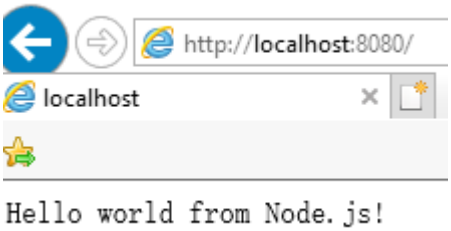
首先，需要引入`http`模块。

```
var http = require('http')
```

接着，创建可以处理请求的服务端程序，对8080端口开放：

```
http.createServer((req,res)=>{
    console.log('Node.js received a request!'); //控制台打印接受到请求
    res.end('Hello world from Node.js!');       //响应'Hello world from Node.js!'
}).listen(8080)
```

打开浏览器，访问本地的8080端口：



使用Express框架

http模块可以搭建服务端程序，但是对于路由，请求预处理，中间件集成等需求，仅使用它就会显得有些无力。

这时，Express框架就出现了。他是目前在Node.js中使用最广泛地服务端web应用框架。

在引用Express模块之前，首先要在npm中安装它，因为他不是Node.js内置的核心模块：

```
npm install express
```

安装成功后可以看到这样的提示：

```
+ express@4.17.1
added 50 packages from 37 contributors in 3.314s
```

（@后面跟着的，是express模块的版本号）

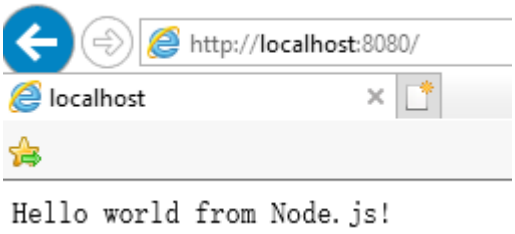
接着，使用express重新编写默认路径的服务：

```
var express = require('express')//引入express模块

let app = express()

//创建get请求服务，路由为默认路径
app.get('/',(req,res)=>{
    console.log('Node.js received a request')
    res.end('Hello world from Node.js!')
})

//设置服务端监听端口为8080，成功后回调在控制台上打印提示
let server = app.listen(8080,()=>{
    console.log('The server is listening on port : 8080')
})
```

执行程序：

```
> node .\server-express.js
The server is listening on port : 8080
```

发送请求：



要编写其他url路径下的服务，只需要更改方法中第一个参数即可：

```
app.get('/hello',(req,res)=>{
    res.end('hello')
})
```

处理请求的参数

在能接收到请求之后，还有一个重要的问题是能够接受到请求中的参数并正确解析他们。

下面重点总结一下`get`和`post`两种请求的参数获取方法

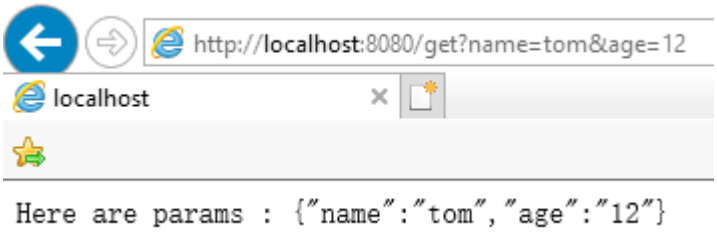
get请求

对于get请求，其参数是直接拼接在url后面的，解析也相对容易。在express框架中，get请求的参数全部都被放置在了`request`对象的`query`中。

如下面服务：

```
app.get('/get',(req,res)=>{
    res.end('Here are params : ' + JSON.stringify(req.query))
})
```

发送get请求：



post请求

然而对于post请求，情况就没有那么简单了。

post请求可以以很多种形式发送数据，这取决于post请求的请求头`Content-type`的内容

一般来说，常见的有三种：

- multipart/form-data
- application/x-www-form-urlencoded

- application/json

处理这些请求的参数，需要用到中间件的支持，为了处理这三种格式，我们需要引入这些中间件：

- `body-parser`，用于处理`application/x-www-form-urlencoded`和`application/json`两种格式
- `connect-multiparty`，用于处理`multipart/form-data`格式

先分别在文件夹中使用npm安装他们

```
npm install body-parser
npm install connect-multiparty
```

接着在代码中引入他们：

```
var bodyParser = require('body-parser')
var multiparty = require('connect-multiparty')
```

处理 application/x-www-form-urlencoded 格式

为了他们能够在express处理post请求的时候发挥作用，需要进行注册

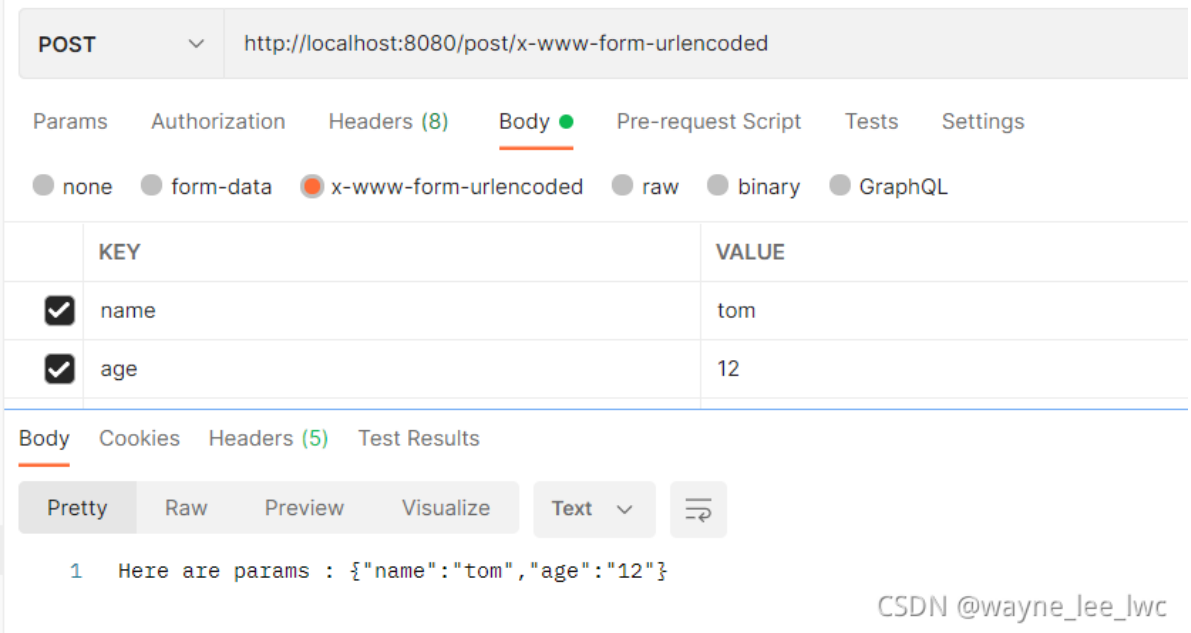
以接受`x-www-form-urlencoded`为例。
一种方式是直接在post方法中作为第二个参数传入：

```
app.post('/post/x-www-form-urlencoded',bodyParser.urlencoded({extended:true}),
(req,res)=>{
    console.log('x-www-form-urlencoded')
    res.end('Here are params : ' + JSON.stringify(req.body))
})
```

这种方式只能在该方法中使用中间件。如果需要所有方法都使用中间件，可以使用另一种方式将其全局注册：

```
app.use(bodyParser.urlencoded({extended:true}))
```

全局注册后，就不需要在post方法中传入了。
测试结果：



处理 application/json 格式

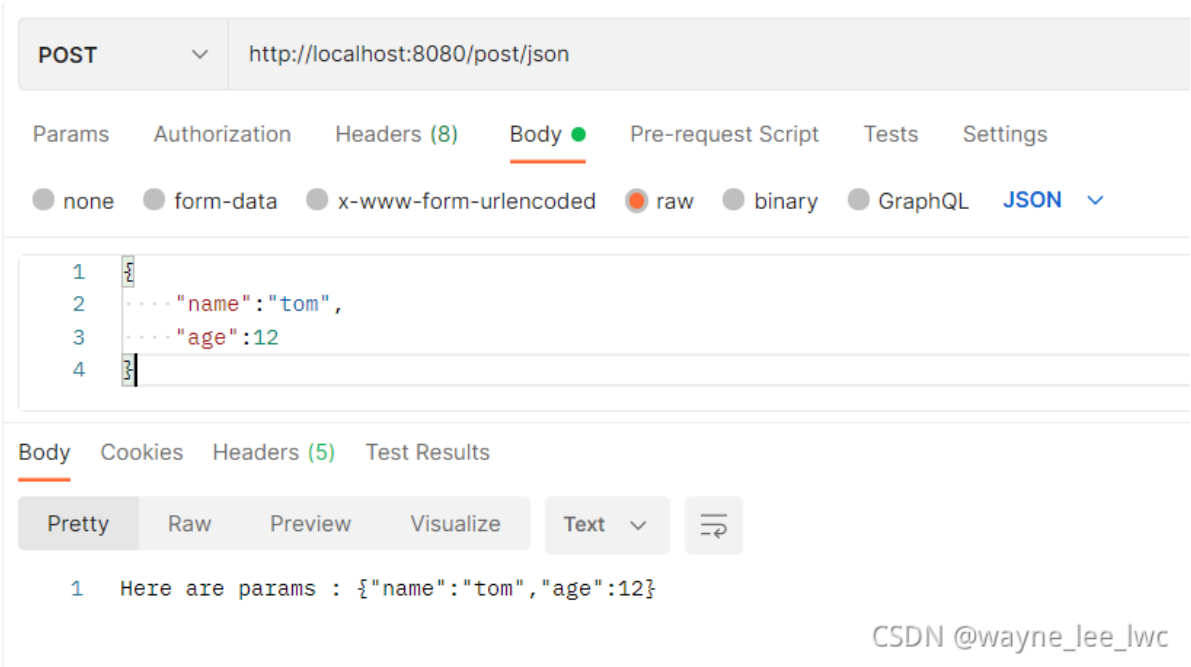
对于`application/json`类型，全局注册语句如下：

```
app.use(bodyParser.json())
```

测试服务代码：

```
app.post('/post/json',(req,res)=>{
    console.log('json')
    res.end('Here are params : ' + JSON.stringify(req.body))
})
```

测试结果：



处理 multipart/form-data 格式

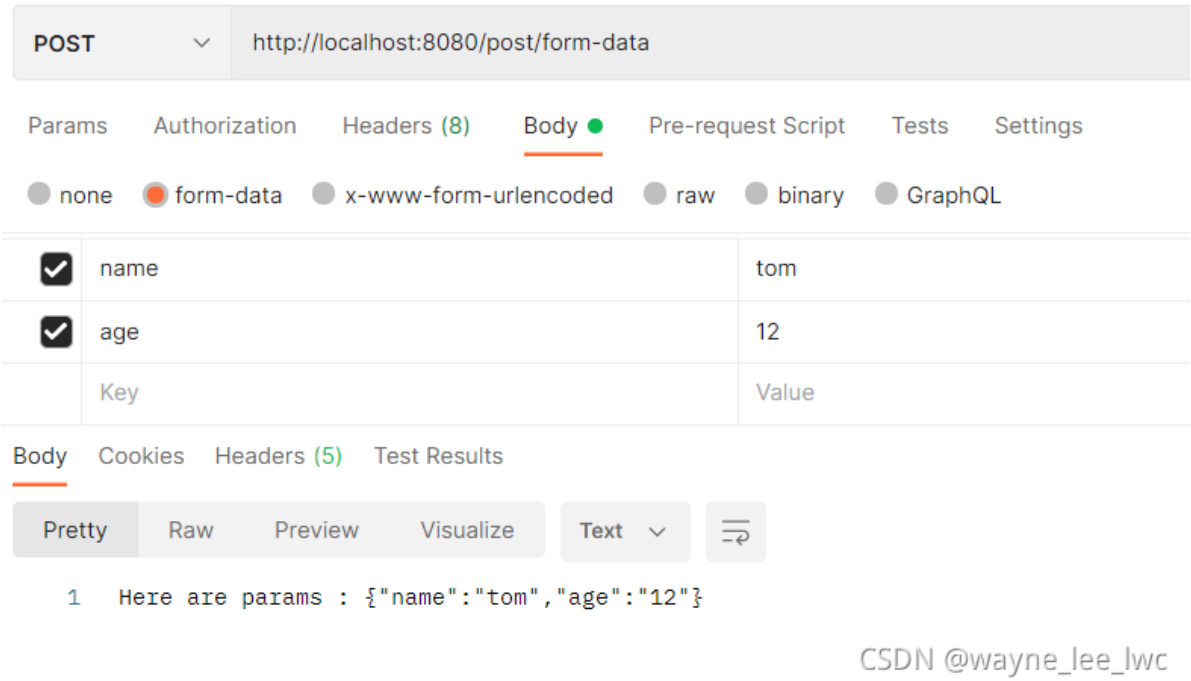
全局注册语句如下：

```
app.use(multiparty())
```

测试服务如下：

```
app.post('/post/form-data',(req,res)=>{
    console.log('form-data')
    res.end('Here are params : ' + JSON.stringify(req.body));
})
```

测试结果：



测试完整代码：

```
var express = require('express')
var bodyParser = require('body-parser')
var multiparty = require('connect-multiparty')

let app = express()

//处理 x-www-form-urlencoded
app.use(bodyParser.urlencoded({
    extended:true
}));

//处理 application/json
app.use(bodyParser.json())

//处理 mutipart/form-data
app.use(multiparty())

app.get('/get',(req,res)=>{
    res.end('Here are params : ' + JSON.stringify(req.query))
})

app.post('/post/x-www-form-urlencoded',(req,res)=>{
    console.log('x-www-form-urlencoded')
    res.end('Here are params : ' + JSON.stringify(req.body))
})

app.post('/post/json',(req,res)=>{
    console.log('json')
    res.end('Here are params : ' + JSON.stringify(req.body))
})

app.post('/post/form-data',(req,res)=>{
    console.log('form-data')
```

```
    res.end('Here are params : ' + JSON.stringify(req.body));
})

let server = app.listen(8080,()=>{
    console.log('The server is listening on port : 8080')
})
```

参考资料

- 菜鸟教程
- 《深入浅出Node.js》
- 《Learn NodeJS in 1 Day》
- 《The node craftsman book》
- 《MERN Projects for Beginners》

其中的英文原著，博主整理在了 **书单列表(booklist)** 小项目中，欢迎小伙伴们光临并查看下载，点点star就是对博主最大的鼓励！
github仓库
gitee仓库

往期内容

- 【Node.js】下载安装及简单使用