

课程名称

第1天课堂笔记（本课程共8天）

前端与移动开发学院

http://web.itcast.cn

# 目录

[目录 2](#_Toc13236)

[一、 3](#_Toc29963)

# 一、Express框架

Express框架是后台的Node框架，所以和jQuery、zepto、yui、bootstrap都不一个东西。

Express在后台的受欢迎的程度，和jQuery一样，就是企业的事实上的标准。

● 原生Node开发，会发现有很多问题。比如：

■ 呈递静态页面很不方便，需要处理每个HTTP请求，还要考虑304问题

■ 路由处理代码不直观清晰，需要写很多正则表达式和字符串函数

■ 不能集中精力写业务，要考虑很多其他的东西

我们自己可以把第一天的作业，就是那个静态文件服务给封装成为模块。封装的越多，就自己做出了类似Express的东西。

● EXPRESS的哲学是在你的想法和服务器之间充当薄薄的一层。这并不意味着他不够健壮，或者没有足够的有用特性，而是尽量少干预你，让你充分表达自己的思想，同时提供一些有用的东西。

英语官网：http://expressjs.com/

中文官网：<http://www.expressjs.com.cn/>

整体感知，Express框架。

安装Express框架，就是使用npm的命令。

|  |
| --- |
| 1. npm install **--save** express |

--save参数，表示自动修改package.json文件，自动添加依赖项。

路由能力：

|  |
| --- |
| 1. var express = require("express"); 2. var app = express(); 3. app.get("/",function(req,res){ 4. res.send("你好"); 5. }); 6. app.get("/haha",function(req,res){ 7. res.send("这是haha页面，哈哈哈哈哈哈"); 8. }); 9. app.get(/^\/student\/([\d]{10})$/,function(req,res){ 10. res.send("学生信息，学号" + req.params[0]); 11. }); 12. app.get("/teacher/:gonghao",function(req,res){ 13. res.send("老师信息，工号" + req.params.gonghao); 14. }); 15. app.listen(3000); |

静态文件伺服能力：

|  |
| --- |
| 1. app.use(express.static("./public")); |

模板引擎：

|  |
| --- |
| 1. var express = require("express"); 2. var app = express(); 3. app.set("view engine","ejs"); 4. app.get("/",function(req,res){ 5. res.render("haha",{ 6. "news" : ["我是小新闻啊","我也是啊","哈哈哈哈"] 7. }); 8. }); 9. app.listen(3000); |

我们学习的是Express4.X，和Express3.X差别非常大。

# 二、路由

当用get请求访问一个网址的时候，做什么事情：

|  |
| --- |
| 1. app.get("网址",function(req,res){ 3. }); |

当用post访问一个网址的时候，做什么事情：

|  |
| --- |
| 1. app.post("网址",function(req,res){ 3. }); |

如果想处理这个网址的任何method的请求，那么写all

|  |
| --- |
| 1. app.all("/",function(){ 3. }); |

这里的网址，不分大小写，也就是说，你路由是

|  |
| --- |
| 1. app.get("/AAb",function(req,res){ 2. res.send("你好"); 3. }); |

实际上小写的访问也行。

所有的GET参数，? 后面的都已经被忽略。 锚点#也被忽略

你路由到/a ， 实际/a?id=2&sex=nan 也能被处理。

正则表达式可以被使用。正则表达式中，未知部分用圆括号分组，然后可以用req.params[0]、[1]得到。

req.params类数组对象。

|  |
| --- |
| 1. app.get(**/^\/student\/([\d]{10})$/**,function(req,res){ 2. res.send("学生信息，学号" + **req.params[0]**); 3. }); |

冒号是更推荐的写法。

|  |
| --- |
| 1. app.get("/student/:id",function(req,res){ 2. var id = req.params["id"]; 3. var reg= /^[\d]{6}$/; //正则验证 4. if(reg.test(id)){ 5. res.send(id); 6. }else{ 7. res.send("请检查格式"); 8. } 9. }); |

表单可以自己提交到自己上。

|  |
| --- |
| 1. app.get("/",function(req,res){ 2. res.render("form"); 3. }); 4. app.post("/",function(req,res){ 5. //将数据添加进入数据库 6. res.send("成功"); 7. }); |

适合进行 RESTful路由设计。简单说，就是一个路径，但是http method不同，对这个页面的使用也不同。

/student/345345

get 读取学生信息

add 添加学生信息

delete 删除学生新

# 三、中间件

如果我的的get、post回调函数中，没有next参数，那么就匹配上第一个路由，就不会往下匹配了。

如果想往下匹配的话，那么需要写**next()**

|  |
| --- |
| 1. app.get("/",function(req,res,**next**){ 2. console.log("1"); 3. **next();** 4. }); 5. app.get("/",function(req,res){ 6. console.log("2"); 7. }); |

下面两个路由，感觉没有关系：

|  |
| --- |
| 1. app.get("**/:username/:id**",function(req,res){ 2. console.log("1"); 3. res.send("用户信息" + req.params.username); 4. }); 5. app.get("**/admin/login**",function(req,res){ 6. console.log("2"); 7. res.send("管理员登录"); 8. }); |

但是实际上冲突了，因为admin可以当做用户名 login可以当做id。

解决方法1：交换位置。 也就是说，express中所有的路由（中间件）的顺序至关重要。

匹配上第一个，就不会往下匹配了。 具体的往上写，抽象的往下写。

|  |
| --- |
| 1. app.get("/admin/login",function(req,res){ 2. console.log("2"); 3. res.send("管理员登录"); 4. }); 5. app.get("/:username/:id",function(req,res){ 6. console.log("1"); 7. res.send("用户信息" + req.params.username); 8. }); |

解决方法2：

|  |
| --- |
| 1. app.get("/:username/:id",function(req,res,**next**){ 2. var username = req.params.username; 3. //检索数据库，如果username不存在，那么next() 4. if(检索数据库){ 5. console.log("1"); 6. res.send("用户信息"); 7. }else{ 8. **next();** 9. } 10. }); 11. app.get("/admin/login",function(req,res){ 12. console.log("2"); 13. res.send("管理员登录"); 14. }); |

路由get、post这些东西，就是中间件，中间件讲究顺序，匹配上第一个之后，就不会往后匹配了。next函数才能够继续往后匹配。

app.use()也是一个中间件。与get、post不同的是，他的网址不是精确匹配的。而是能够有小文件夹拓展的。

比如网址： http://127.0.0.1:3000/admin/aa/bb/cc/dd

|  |
| --- |
| 1. app.use("/admin",function(req,res){ 2. res.write(req.originalUrl + "\n"); // /admin/aa/bb/cc/dd 3. res.write(req.baseUrl + "\n"); // /admin 4. res.write(req.path + "\n"); // /aa/bb/cc/dd 5. res.end("你好"); 6. }); |

如果写一个/

|  |
| --- |
| 1. //当你不写路径的时候，实际上就相当于"/"，就是所有网址 2. app.use(function(req,res,next){ 3. console.log(new Date()); 4. next(); 5. }); |

app.use()就给了我们增加一些特定功能的便利场所。

实际上app.use()的东西，基本上都从第三方能得到。

● 大多数情况下，渲染内容用res.render()，将会根据views中的模板文件进行渲染。如果不想使用views文件夹，想自己设置文件夹名字，那么app.set("views","aaaa");

● 如果想写一个快速测试页，当然可以使用res.send()。这个函数将根据内容，自动帮我们设置了Content-Type头部和200状态码。send()只能用一次，和end一样。和end不一样在哪里？能够自动设置MIME类型。

● 如果想使用不同的状态码，可以：

res.status(404).send('Sorry, we cannot find that!');

● 如果想使用不同的Content-Type，可以：

res.set('Content-Type', 'text/html');

# 四、GET请求和POST请求的参数

● GET请求的参数在URL中，在原生Node中，需要使用url模块来识别参数字符串。在Express中，不需要使用url模块了。可以直接使用req.query对象。

● POST请求在express中不能直接获得，必须使用body-parser模块。使用后，将可以用req.body得到参数。但是如果表单中含有文件上传，那么还是需要使用formidable模块。

Node中全是回调函数，所以我们自己封装的函数，里面如果有异步的方法，比如I/O，那么就要用回调函数的方法封装。

错误：

|  |
| --- |
| 1. res.reder("index",{ 2. "name" : student.getDetailById(234234).name 3. }); |

正确:

|  |
| --- |
| 1. student.getDetailByXueHao(234234,function(**detail**){ 2. res.render("index",{ 3. "name" : **detail**.name 4. }) 5. }); |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |

|  |
| --- |
|  |