link: null
title: 珠峰架构师成长计划
description: Express是一个简洁、灵活的node.js Web应用开发框架,是目前最流行的基于Node.js的Web开发框架.
它提供一系列强大的功能，比如：
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=96 sentences=99, words=566

1. Express介绍

Express是一个简洁、灵活的node.js Web应用开发框架,是目前最流行的基于Node.js的Web开发框架. 它提供一系列强大的功能，比如：

- 模板解析
- 静态文件服务
- 中间件
- 路由控制

还可以使用其他模块来帮助你创建各种Web和移动设备应用

2. 使用express

本地安装

```
$ npm install express
```

获取、引用 通过变量 app我们就可以调用 express的各种方法

```
var express = require('express');
var app = express();

app.listen(3000);
```

思考: express的本质上是什么，是如何工作的

3. get请求

根据请求路径来处理客户端发出的GET请求 语法

```
app.get(path,function(request, response));
```

- 第一个参数 path为请求的路径
- 第二个参数为处理请求的回调函数，有两个参数分别是
    - request 代表请求信息
    - response 代表响应信息

```
var express = require('express');
var app = express();
app.get('/',function(req,res){
    res.end('welcome to  首页');
});
app.get('/about',function(req,res){
 res.end('欢迎来到关于我们');
})
app.listen(3000);
```

4.curl客户端使用方法

- 指定请求头

```
curl -H 'content-type:application/json;charset=utf-8' http://localhost:8080/users
```

- 指定方法名

```
curl -X POST http://localhost:8080/users
```

- 指定请求体

```
curl --data "name=zfpx&age=8" http://localhost:8080/users
```

5. all

app.all()函数可以匹配所有的HTTP动词 路由中的星号能匹配所有的路径 语法

```
app.all(path,function(request, response));
```

```
var express = require('express');
var app = express();
app.all("*",function(req,res){
 res.send("404");
})
app.listen(3000);
```

6. 获取请求参数

- req.host 返回请求头里取的主机名(不包含端口号)
- req.path 返回请求的URL的路径名

```
app.get('/',function(req,res){
   res.end('欢迎来到首页'+req.host+" "+req.path);
});
```

7.获得查询字符串

```
app.get('/',function(req,res){
   res.send(req.query);
});
```

8. params路径参数

req.params可以用来获取请求URL中的参数值

```
app.get('/:id/:name',function(req,res){
   res.send(req.params.id+" "+req.params.name);
});
```

9. 中间件

中间件就是处理HTTP请求的函数，用来完成各种特定的任务 比如检查用户是否登录、检测用户是否有权限访问等，它的特点是:

- 一个中间件处理完请求和响应可以把相应数据再传递给下一个中间件
- 回调函数的 next参数,表示接受其他中间件的调用，函数体中的next(),表示将请求数据传递给下一个中间件
- 还可以根据路径来区分进行返回执行不同的中间件

```
var express = require('express');
var app = express();
var path = require('path');

app.use(function(req,res,next){
 res.setHeader('Content-Type','text/plain;charset=utf-8');
 next();
});

app.get('/',function(req,res){
 res.end('首页');
});
app.get('/about',function(req,res){
 res.end('关于我们');
});

app.listen(3000);
```

编写一个请求日志中间件，不管客户端访问什么路径，都在控制台打印出 方法名 路径

10. send

send方法向浏览器发送一个响应信息，并可以智能处理不同类型的数据 并在输出响应时会自动进行一些设置，比如HEAD信息、HTTP缓存支持等等。 语法

```
res.send([body|status], [body])
```

示例 1.当参数为一个String时，Content-Type默认设置为"text/html"。

```
res.send('Hello World');
```

2.当参数为Array或Object时，Express会返回一个JSON

```
res.send({ user: 'tobi' });
res.send([1,2,3]);
```

3.当参数为一个Number时，并且没有上面提到的任何一条在响应体里，Express会帮你设置一个响应体，比如：200会返回字符"OK"

```
res.send(200);
res.send(404);
res.send(500);
```

11.模板

在nodejs中使用express框架，它默认的是ejs和jade渲染模板

```
npm install ejs
```

使用ejs模板

```

app.set('view engine', 'ejs');
app.set('views',path.join(__dirname,'views'));
res.render('index');
```

模板使用html后缀

```

app.set( 'view engine', 'html' );
app.set('views',path.join(__dirname,'views'));

app.engine( '.html', require( 'ejs' ).__express );
```

- 参数 view就是模板的文件名
- 在渲染模板时 locals可为其模板传入变量值
- callback用来处理返回的渲染后的字符串

```
res.render(view, [locals], callback);
```

```
var tmpl = '{{name}}{{age}}';
var data = {name:'zfpx',age:30};
var html= tmpl.replace(/\{\{(\w+)\}\}/g,function(input,group){
    return data[group];
})

console.log(html);
```

12. 静态文件服务器

如果要在网页中加载静态文件（css、js、img），就需要另外指定一个存放静态文件的目录，当浏览器发出非HTML文件请求时，服务器端就会到这个目录下去寻找相关文件

```
app.use(express.static(path.join(__dirname,'/')));
```

13. 重定向

redirect方法允许网址的重定向，跳转到指定的url并且可以指定status，默认为302方式。 语法

```
res.redirect([status], url);
```

```
res.redirect("http://www.baidu.com");
```

14. post请求

post方法 根据请求路径来处理客户端发出的Post请求 语法

```
app.post(path,function(req, res));
```

```
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended:true}));
app.post('/login',function(req,res){
        console.log(req.body.username);
});
```

15. 作业:注册登陆实战

实现一个注册登录的功功，描述如下

16. 中间件原理

```

var app = function(req,res){
   var i=0;

   function next(){
       var fn = app.routes[i++];
       if(fn)
        fn(req,res,next);
   }
    next();
}

app.routes = [];

app.use = function(fn){

    app.routes.push(fn);
}

app.use(function(req,res,next){
    console.log(req.url);
    console.log(1);
    next();
});
app.use(function(req,res,next){
    console.log(2);
    res.end('ok');
    next();
});

var http = require('http');
var server = http.createServer(app);
server.listen(9090);
```

17. params原理

```

var path = '/users/:name/:age';

var url = '/users/zfpx/8';

var paramNames = [];
var regStr = path.replace(/:(\w+)/g,function(matchedStr,group1){
    paramNames.push(group1);
    return '(\\w+)';
});
console.log(regStr);
var reg = new RegExp(regStr);
var result = url.match(reg);

console.log(result);
var params = {};

for(var i=0;i1];
}
console.log(params);
```

资源