```
var express = require('express');
// app 是一个请求监听的回调函数，每当请求到来的时候会执行此函数
var app = express();
//使用中间件
// next 是一个函数 如果调用它表示此中间件执行完毕，可以继续向下
//可以把公共的代码写在中间件里
/**
 * 路由和中间件的区别和联系
 * 1 . 他们在同一个数组中
 * 2. 中间件不匹配路径和方法名，路由要匹配路径和方法名
 * 3. 中间件多了 next 参数，它能决定是否继续
 */
/**
 * 中间件的功能
 * 1. 增加公共的方法和属性
 * 2. 进行公共的处理
 */
app.use(function(req,res,next){

res.setHeader('Content-Type','text/plain;charset=u
```

```
tf-8');
    //next();
});
//当客户端通过GET的请求方式，访问/路径的时候
// get post delete put head patch
app.get('/',function(req,res){
    res.end('首页');
});

app.get('/login',function(req,res){
    res.end('get 登陆');
})

app.post('/login',function(req,res){
    res.end('post 登陆');
});;

app.get('/user',function(req,res){
    res.end('用户主页');
});
//定制404提示，当请求的路径没有路由能处理的时候，返回你请求的页面不存在
```

```
// all 匹配所有的方法 * 匹配所有的路径
app.all('*',function(req,res){
    res.end('你请求的页面不存在');
});
//在当前的本机服务器上监听 9090 端口
app.listen(9090);
/*var server = require('http').createServer(app);
server.listen(9090);*/
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var express = require('express');
var app = express();
/**
 * 1. 中间件可以给 req res 赋值
 * 2. 中间件里可以结束响应
 */

//中央
app.use(function(qqq,rrr,ccc){
   console.log(qqq.url);
    qqq.money = 100;
    ccc();
});
```

```
//省里
app.use(function(req,res,next){
    req.money -= 30;
    next();
});
//县里
app.use(function(req,res,next){
    console.log('县里');
    req.money -= 30;
    //如果next 没有参数，会继续执行后续的路由和正常中间件，如果传了参数，表示出错了，会掉过正常的路由和中间件，交由错误处理中间件来处理
    next('错误：失火了');
});
//村里
app.use(function(req,res,next){
    console.log('村里');
    req.money -= 40;
    res.end();
    //next();
});
app.get('/money',function(req,res){
```

```
    console.log('农民收到'+req.money);
    res.end();
});
//错误处理中间件 有四个参数
app.use(function(err,req,res,next){
    console.log(1,err);
    res.end('error');
    next();
});

app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
/**
 * 参数的处理
 * 请求行 method url(pathname+query)
 * 请求头 headers
 * 请求体 body
 */
var express = require('express');
var http  = require('http');
//console.log(http.STATUS_CODES);
var url = require('url');
```

```
var app = express();
// app.use(function (req,res,next) {
//     var urlObj = url.parse(req.url,true);
//     req.path = urlObj.pathname;
//     req.query = urlObj.query;
//     next();
// });
/**
 * typeof number object
 * Object.prototype.toString.call Array Object
 * obj.constructor
 * instanceOf
 */
var util = require('util');
/**
app.use(function(req,res,next){
    res.send = function(params){
        var type = typeof params;//获得参数类型
        switch (type){//判断参数类型
            case 'object'://如果是对象

res.setHeader('Content-Type','application/json;
```

```
charset=utf-8');
                res.end(JSON.stringify(params));
                break;
            case 'string':

res.setHeader('Content-Type','text/html;
charset=utf-8');
                res.end(params);
                break;
            case 'number':

res.setHeader('Content-Type','text/plain;
charset=utf-8');
                res.statusCode = params;
                res.end(http.STATUS_CODES[params]);
                break;
            default:
                res.end(util.inspect(params));
        }
    }
    next();
});
```

```
 **/
app.get('/',function(req,res){
   /* console.log('method=',req.method);
    var urlObj = url.parse(req.url,true);
    console.log('pathname=',urlObj.pathname);
    console.log('query=',urlObj.query);*/
    console.log(req.path);
    console.log(req.query);
    //First argument must be a string or Buffer
    /**
     * 参数是对象 Content-Type:application/json; charset=utf-8
     * 参数是字符串 Content-Type:text/html; charset=utf-8
     * 参数是数字 把数字当成了状态码 响应体是对应的描述短语 Content-Type:text/plain; charset=utf-8
     */
   // res.status(404).send('页面不存在');
    res.sendStatus(404);
});
//获取某个用户的信息  路径可以写正则
/*app.get(/^\/users\/(\w+)$/,function(req,res){
```

```
    res.end('1');
});*/
//获取某个用户的信息 :id 是占位符 匹配一个字符串
/users/1
app.get('/users/:id',function(req,res){
    //params 是 express 帮我们添加的对象属性，属性名就是
占位符，值是实际请求的字符串占位符对应的部分
    res.end('id='+req.params.id);
});
app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var users =[{id:1,name:'张三'},{id:2,name:'李四'}];
var express = require('express');
var path = require('path');
var app = express();
//设置模板引擎 用来添加文件后缀的
app.set('view engine','html');
//console.log(app.get('view engine'));
//设置模板存放目录 指定一个模板的绝对路径
app.set('views',path.join(__dirname,'views'));
//设置对于html 类型的模板使用ejs 来进行渲染
app.engine('.html',require('ejs').__express);
```

```
// http://localhost:9090/users/1
/**
 * 1. 定义路由
 * 2. 取得 ID
 * 3. 取得 ID 对应的 user 对象
 * 4. 渲染模板
 */
app.get('/users/:id',function(req,res){
    var id = req.params.id;
    var user = users.find(function(user){
        return user.id == id;
    });
    //用数据渲染模板
    // 1.模板的相对路径
    // 2 参数是渲染模板数据对象  模板里的变量要从数据对象的属性中取值
    res.render('user',user);
})
app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var express = require('express');
var app = express();
```

```
app.use(function(req,res,next){
    console.log(req.path);
    next();
});
app.get('/',function(req,res){
    //重定向 告诉客户端重新向新的URL 地址发起请求
    res.redirect('/user');
});
app.get('/user',function(req,res){
    res.send('用户');
});
app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var express = require('express');
var path = require('path');
var bodyParser = require('body-parser');
var app = express();
/**
 * 对服务器发起请求的时候 ../ ./ / 都一样
/node_modules/bootstrap/dist/css/bootstrap.css
 *
 */
```

```
//设置模板引擎
app.set('view engine','html');
//resolve 从当前路径出发，得到一个绝对路径
console.log(path.resolve('views'),__dirname);
app.set('views',path.resolve('views'));
app.engine('.html',require('ejs').__express);
var fs = require('fs');
app.use(function(req,res,next){
    res.locals.title = '珠峰培训';
    next();
});
/**
 * 静态文件中间件
 * 当客户端访问服务器一个静态文件的时候，服务器返回这个静态文件
 */
//静态文件根目录
// 1.取出请求路径 2. 到根目录下找找有没有这个文件
// 3.如果找到的话返回给客户端 4.没找到 next 的
/*function static(staticRootDir){
   //staticRootDir = F:\201609node\1.express\public
   //req.path=/bootstrap/dist/css/bootstrap.css
```

```
    //filename=
F:\201609node\1.express\public\bootstrap\dist\css\
bootstrap.css
    return function(req,res,next){
        //先得到静态文件绝对路径
        var filename =
path.join(staticRootDir,req.path);
        //判断文件是否存在
        fs.exists(filename,function(exists){
          if(exists){//如果存在，读出来发送给客户端
             fs.createReadStream(filename).pipe(res);
          }else{//如果不存在，继续执行
              next();
          }
        });
    }
}*/
//参数就是public的绝对路径
app.use(express.static(path.resolve('public')));
//使用请求体中间件，把请求体字符串转成对象挂载到
req.body上
app.use(bodyParser.urlencoded({extended:true}));
```

```
var querystring  = require('querystring');
app.use(function(req,res,next){
   var result  =""
   req.setEncoding('utf8');//设置编码之后 data 就成为了字符串
   req.on('data',function(data){
       result+=data;
   })
    // username=zfpx&password=8
   req.on('end',function(){
        req.body = querystring.parse(result);
         next();
   })
});
/*app.get('/node_modules/bootstrap/dist/css/bootstrap.css',function(req,res){
    //path must be absolute or specify root to res.sendFile

//res.sendFile('./public/bootstrap/dist/css/bootstrap.css',{root:__dirname});
```

```
res.sendFile(path.resolve('public/bootstrap/dist/c
ss/bootstrap.css'));
});*/
var users = [];
/*app.use(function(req,res,next){
    //重定向要求客户端重新向新的地址发起请求
    res.redirect = function(path){
        res.statusCode = 302;
        //在响应头重指定新的地址
        res.setHeader('Location',path);
        res.end();
    }
    next();
});*/
//当客户端通过GET /reg 发起请求的时候
app.get('/reg',function(req,res){
    res.render('reg',{title:'用户注册'});
/*
    var username = req.query.username;
    if(username){ //表示提交表单
        var password = req.query.password;
        //把用户名和密码封装成对象保存到数组中
```

```
            users.push({username,password});
            //redirect 会触发URL 地址的改变
            res.redirect('/login');
            //res.render('login');render 则不会
        }else{//获取空白表单
                //模板的路径 views 的父目录+reg+view engine 后缀
            //render 方法自带end 获取模板 渲染模板为html 发送给客户端并结束响应 end
            /!**
             * 如果不传回调函数 先渲染模板 然后直接返回给客户端并结束响应
             * 如果传回调函数 只管渲染模板，把渲染后的HTML 字符串传给回调函数
             *!/
            res.render('reg',{title:'用户注册'});
        }*/
});
//客户端在发送 post 请求的时候，会把输入的表单转成查询字符串，并且放在请求体里发送给服务器
app.post('/reg',function(req,res){
    var user = req.body;
```

```
    console.log(user);
    //把用户名和密码封装成对象保存到数组中
    users.push(user);
    //redirect 会触发URL 地址的改变
    res.redirect('/login');
    //res.render('login');render 则不会
});

app.get('/login',function(req,res){
    var username = req.query.username;
    if(username){
        var password = req.query.password;
        var user = users.find(function(user){
            return user.username == username &&
user.password == password;
        });
        if(user){
            res.redirect('/welcome');
        }else{
            res.redirect('back');
        }
    }else{
```

```
        //模板的路径 views 的父目录+reg+view engine 后
缀
        res.render('login',{title:'用户登陆'});
    }

});
app.get('/welcome',function(req,res){
    //模板的路径 views 的父目录+reg+view engine 后缀
    res.render('welcome',{});
});
app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var http = require('http');
var url = require('url');
var util = require('util');
var querystring = require('querystring');
http.createServer(function(req,res){
    var urlObj = url.parse(req.url,true);
    var pathname = urlObj.pathname;
    if(pathname == '/write'){
        //通过响应头写入 cookie Cookie:name=zfpx;
age=8
```

```
//res.setHeader('Set-Cookie',['name=zfpx','age=9']
);
        res.setHeader('Set-Cookie','name=zfpx;
age=9')
      res.end();
    }else if(pathname == '/read'){
        //Cookie:name=zfpx; age=8
        var cookie = req.headers.cookie;
        var cookies = querystring.parse(cookie,'; ');
        //console.log(cookie);//name=zfpx; age=8
        res.end(util.inspect(cookies));
    }else{
        res.end('404');
    }
}).listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var express = require('express');
var cookieParser = require('cookie-parser');
var app = express();
app.use(cookieParser());
/*app.use(function (req, res, next) {
```

```
    res.cookie = cookie(name, val, options)
    {
        var opt = options || {};
        var value = encodeURIComponent(val);
        var pairs = [name + '=' + value];
        if (null != opt.maxAge) {
            var maxAge = opt.maxAge - 0;
            if (isNaN(maxAge)) throw new
Error('maxAge should be a Number');
            pairs.push('Max-Age=' +
Math.floor(maxAge));
        }
        if (opt.domain) {
            pairs.push('Domain=' + opt.domain);
        }
        if (opt.path) {
            pairs.push('Path=' + opt.path);
        }
        if (opt.expires) pairs.push('Expires=' +
opt.expires.toUTCString());
        if (opt.httpOnly) pairs.push('HttpOnly');
        if (opt.secure) pairs.push('Secure');
```

```
        return pairs.join('; ');
    }
    next();
});*/
app.get('/write', function (req, res) {
    //写入cookie
    //设置域名 设置域名之后，只有向这些域名发起请求是才会发送cookie
    //res.cookie('name','zfpx',{domain:'zf1.cn'});
    //Set-Cookie:name=zfpx; Path=/read1
    //设置了path之后，只有向这个 path 发起请求才会发送cookie
    //res.cookie('name','zfpx',{path:'/read1'});
    //res.cookie('age','8',{path:'/read1'});
    //指定失效的绝对事件和相对时间
    ///res.cookie('name','zfpx',{expires:new Date(Date.now()+10*1000)});
    res.cookie('name', 'zfpx', {maxAge: 10 * 1000});
    res.cookie('age', '8', {maxAge: 10 * 1000});
    res.send('write ok');
});
```

```
app.get('/read1', function (req, res) {
    res.send(req.cookies);
});
app.get('/read2', function (req, res) {
    // req.headers.cookie +querystring =>
req.cookies
    res.send(req.cookies);
});
app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
var express = require('express');
/**
 * 使用session 中间件
 *
 * @type {session}
 */
var session = require('express-session');
var app = express();
//使用了session 中间件之后，req.session 就是此客户端在
服务器端对应的数据对象
app.use(session({
```

```
    resave:true,//每次客户端访问服务器的时候都重新保
存 session 对象
    secret:'zfpx',//秘密 加密
    saveUninitialized:true//保存未初始化的 session
}));

app.get('/visit',function(req,res){
   var visit =
req.session.visit?req.session.visit+1:1;
   req.session.visit = visit;
   res.send(`顾客，这是你的第${visit}次访问`);
});

app.listen(9090);
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/*/
```