link: null
title: 珠峰架构师成长计划
description: express.static是 Express 内置的唯一一个中间件。是基于 serve-static 开发的，负责托管 Express 应用内的静态资源。
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=39 sentences=56, words=347

1. 静态文件中间件

```
express.static(root, [options])
```

express.static是 Express 内置的唯一一个中间件。是基于 serve-static 开发的，负责托管 Express 应用内的静态资源。

- root 参数指的是静态资源文件所在的根目录。
- options 对象是可选的，支持以下属性：

属性 描述 类型 默认值 dotfiles 控制点文件服务，可选值为allow,deny,'ignore' String "ignore" etag 控制etag生成 Boolean true extensions 设置文件后缀名补充 Boolean false index 设置目录访问的返回，设置为false可以禁止目录访问 Mixed "index.html" lastModified 根据文件修改时间设置Last-Modified报头 Boolean true maxAge 设置Cache-Control报头的缓存控制时间，单位为毫秒 Number 0 redirect 当路径名是目录时，重定向到包含结尾/的目录 Boolean true setHeaders 函数用于为文件设置HTTP头 Function

关于此中间件的细节，请参考 通过 Express 托管静态资源文件 (http://www.expressjs.com.cn/starter/static-files.html)。

2. body-parser中间件

body-parser是非常常用的一个express中间件，作用是对post请求的请求体进行解析。使用非常简单，以下两行代码已经覆盖了大部分的使用场景。

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
```

2.1 http报文头

```
POST / HTTP/1.1
Host: 127.0.0.1:8080
Content-Type: text/html; charset=utf8
Content-Encoding: gzip

zfpx
```

- Content-Type：请求报文主体的类型、编码。常见的类型有text/plain、application/json、application/x-www-form-urlencoded。常见的编码有utf8、gbk等。
- Content-Encoding：声明报文主体的压缩格式，常见的取值有gzip、deflate、identity。 报文主体：这里是个普通的文本字符串zfpx。

2.2 body-parser工作

- 处理不同类型的请求体：比如text、json、urlencoded等，对应的报文主体的格式不同。
- 处理不同的编码：比如utf8、gbk等。
- 处理不同的压缩类型：比如gzip、deflate等。

2.3 处理内容类型

2.3.1 处理text/plain

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

var options = {
    hostname: '127.0.0.1',
    port: '3000',
    path: '/',
    method: 'POST',
    headers: {
        'Content-Type': 'text/plain',
        'Content-Encoding': 'identity'
    }
};

var client = http.request(options, (res) => {
    res.pipe(process.stdout);
});

client.end('zfpx');
```

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

var parse = function (req, done) {
    let arr = [];
    req.on('data', data => {
        arr.push(data);
    });

    req.on('end', () => {
        let result = Buffer.concat(arr);
        done(result);
    });
};

var server = http.createServer(function (req, res) {
    parse(req, (chunks) => {
        res.end(chunks)
    });
});

server.listen(3000);
```

2.3.2 处理application/json

```
'Content-Type': 'application/json',
client.end( JSON.stringify({name:zfpx}) );
```

```
var json = JSON.parse( chunks.toString() );
```

2.3.3 处理application/x-www-form-urlencoded

```
'Content-Type': 'form/x-www-form-urlencoded',
client.end( querystring.stringify({name:'zfpx}) );
```

```
var body = querystring.parse( chunks.toString() );
```

2.4 处理不同编码

```
var iconv = require('iconv-lite');
var encoding = 'gbk';
'Content-Type': 'text/plain; charset=' + encoding,
var buff = iconv.encode('zfpx', encoding);
```

```
var contentType = require('content-type');
var obj = contentType.parse(req.headers['content-type']);
var charset = obj.parameters.charset;
var body = iconv.decode(chunks, charset);
```

2.5 处理不同压缩类型

```
var zlib = require('zlib');
'Content-Encoding': 'gzip'
var buff = zlib.gzipSync('zfpx');
client.end(buff);
```

```
var contentEncoding = req.headers['content-encoding'];
if(contentEncoding === 'gzip') {
  stream = zlib.createGunzip();
  req.pipe(stream);
}
```