新人分享

——nodejs运行机制及RSS爬虫实例分享

shirlyzhang

nodejs是一个平台

让Javascript运行在浏览器之外的平台

为什么适合开发服务器端程序呢？

- 模块机制
- 异步IO，事件轮询机制

1 模块系统遵循commonJS规范

方便的定义和引入

```
var http = require('http');
var user = require('./routes/user')
```

得到模块的导出对象module.exports

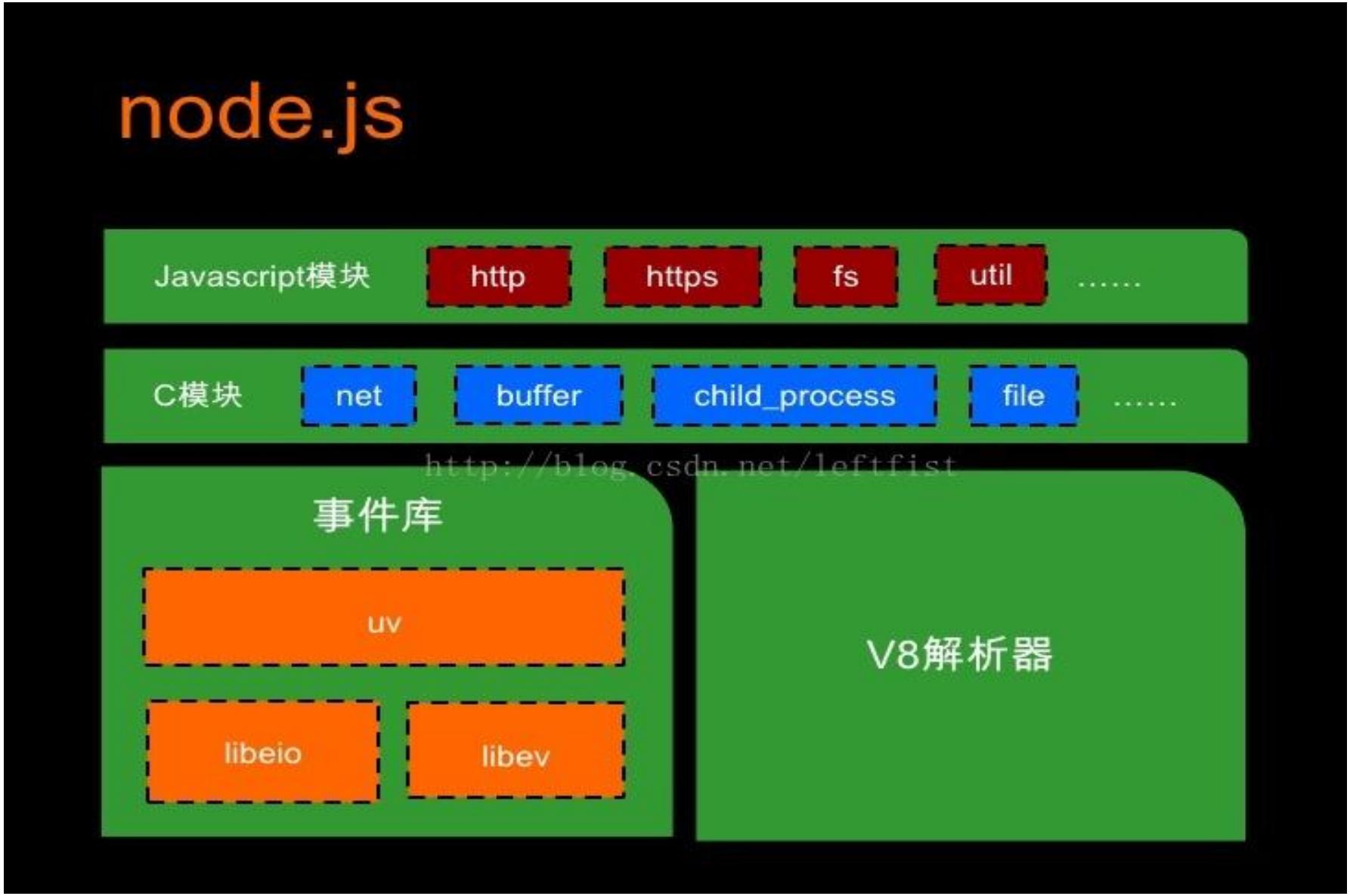
require的一个重要行为就是它缓存了module.exports的值并且在未来再次调用require时返回同样的值。
它依据被require文件的绝对路径来进行缓存。

如果你想要你的模块返回不同的值，
你应该导出一个在再次调用时能返回不同值的函数

内置封装了很多模块供js调用

- 如文件，网络通信等CoreJavascript中没有或者不完善的功能

- net、 http、fs、stream等



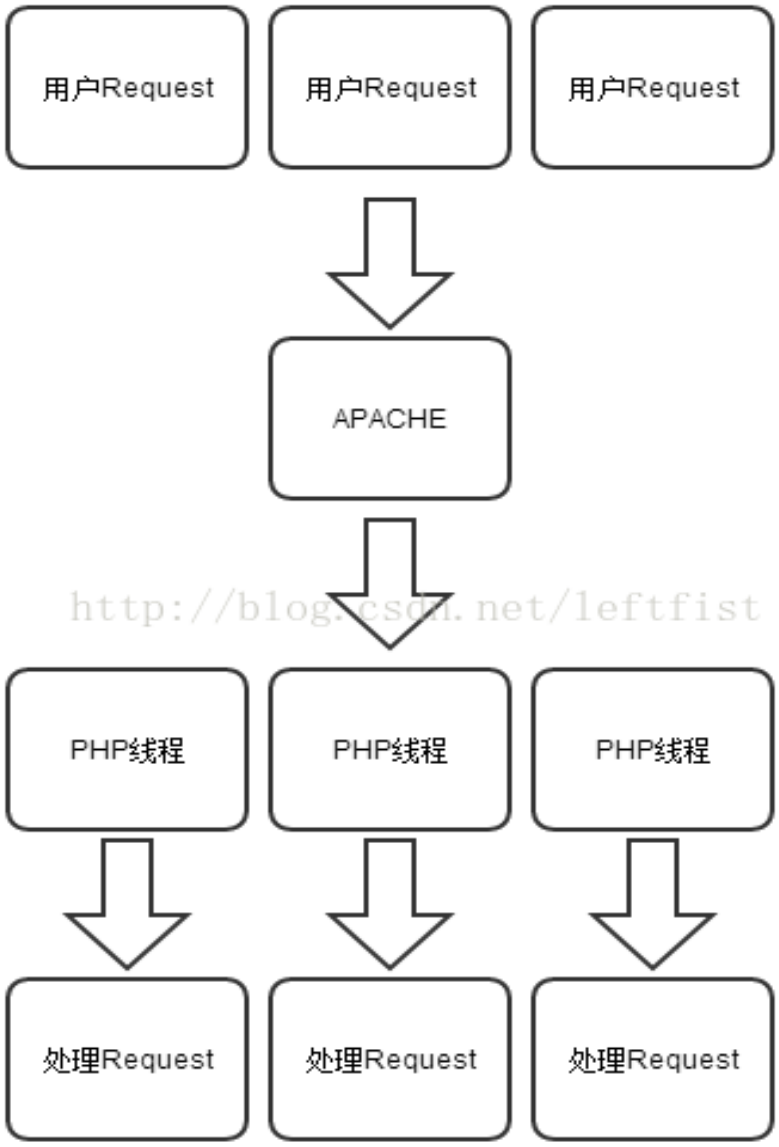
不断更新的第三方模块

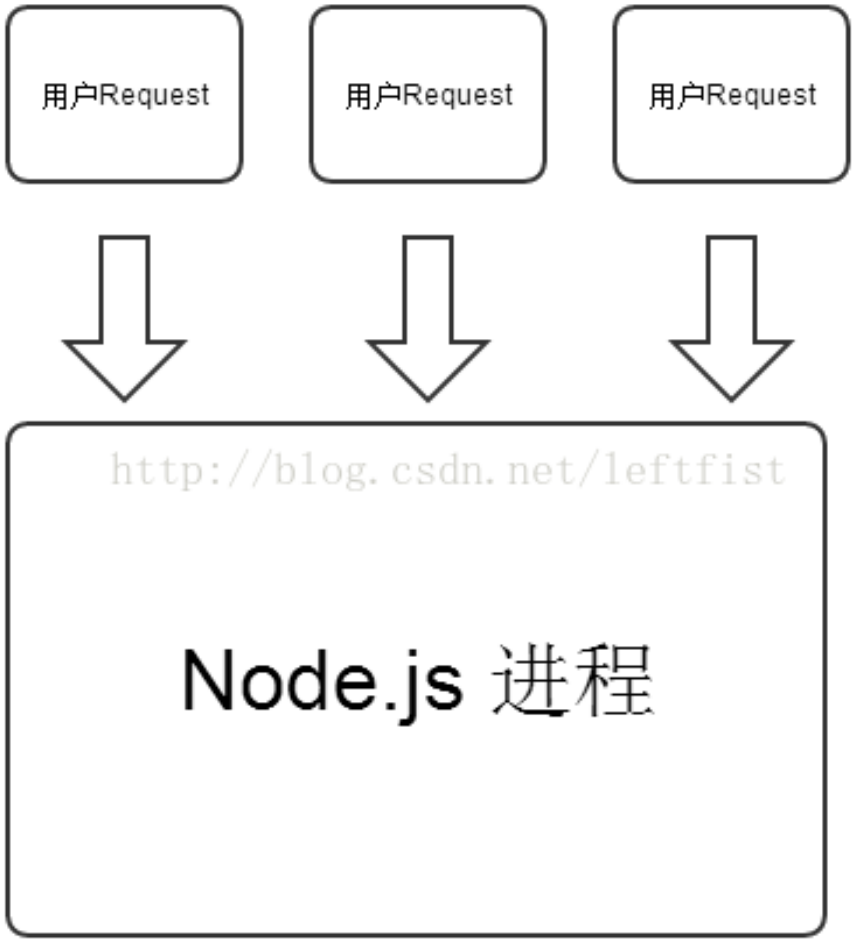
- 如基于http上的MVC框架--express模块
- Web Socket服务--Socket.io模块
- HTTP请求调用—Request模块
- 数据库驱动—Mongoose模块
- 异步流程控制—Async模块

2 非阻塞模式

异步IO与事件轮询机制

Nodejs是单线程的





好处就是

1）简单
2）高性能，避免了频繁的线程切换开销
3）占用资源小，在大负荷情况下，
　　对内存占用仍然很低
3）线程安全，没有死锁这些问题

单线程如何处理并发？

Node最大的并发量就是1，
但是v8执行js代码的速度非常快，
在调用堆栈执行非常快的情况下，
同一微小的时刻，也无须处理多个请求。

异步IO、回调方式

遇到IO、数据库等费时的请求，
会异步执行，不阻塞等待，
而是等它们执行完后通知主线程，
主线程在执行相对应的回调函数，

```
var fs = require("fs");
fs.readFile("./testfile", "utf8", function(error, file) {
    if (error) throw error;
    console.log("我读完文件了！ ");
});
console.log("我不会被阻塞！ ");
```

采用事件轮询机制实现

shirlyzhang

- 主线程上运行eventloop线程
- 运行到某个事件，先注册事件
- 不停的询问内核这些事件是否已经分发
- 当事件分发时，对应的回调函数就会被触发
- 如果没有事件触发，则继续执行其他代码
- 直到有新事件时，再去执行对应的回调函数

我们可以利用http模块方便的架起
一个服务器监听端口，接收各种请求消息，
传递给置于其中的处理函数。

```
1  var http = require('http');
2  server = http.createServer(function (request, response) {
3        res.writeHeader(200, {"Content-Type": "text/plain"});
4        res.end("Hello World\n");
5  })
6  server.listen(8080);
7  console.log("http server run in port: 8080");
```

我们要考虑的就是有请求过来后，function(request, response){}中应该怎么响应，怎么根据不同的请求给出相应的response。

你最原始的nodejs服务器程序可能是这样的

shirlyzhang

```
1  if (request.method === “POST”) {
2      postHandle();
3  }
4  else {
5      getHandle();
6  }
```

既然nodejs的特色之一是各种第三方模块功能，那我们可以不用麻烦关注如何路由不同的请求

```
var express = require('express'),
      routes = require('./routes');
var app = express();
//环境变量
app.set('views', __dirname + '/views');
app.use(app.router);
app.use(express.static(path.join(__dirname, 'public')));

// 开发模式
if ('development' == app.get('env')) {
    app.use(express.errorHandler());
}

// 路径解析
app.get('/', routes.index);
app.get('/users', routes.list);

// 启动及端口
http.createServer(app).listen(app.get('port'), function(){
    console.log('Express server listening on port '
            + app.get('port'));
});
```

这样，我们重点关注的就是routes内相应的处理逻辑了

Nodejs借助事件驱动，非阻塞 I/O 模型变得轻量和高效，非常适合：

1、io操作多，cpu操作少的场合

2、数据简单，但是数据访问频繁的场合

下面具体分享下使用nodejs做的
抓取rss新闻的网络爬虫程序的逻辑

RSS(简易信息聚合)

- 基于XML标准
- 一种描述和同步网站内容的格式

```
▼<rss version="2.0">
  ▼<channel>
    <title>IMWeb: web developer center/pastest个人主页</title>
    <link>http://imweb.io/user/pastest</link>
    <language>zh-cn</language>
    <description>IMWeb: web developer center/pastest个人主页</description>
   ▼<item>
      <title>test code display</title>
      <link>http://imweb.io/topic/54b0a0829c6d622f55ba5a57</link>
      <guid>http://imweb.io/topic/54b0a0829c6d622f55ba5a57</guid>
     ▼<description>
        <div class="markdown-text"><p>这是一个普通段落：</p> <pre class="prettyprint "><code>这是一个
        class=&quot;footer&quot;&gt; &amp;copy; 2004 Foo Corporation &lt;/div&gt;</code></pre></div>
      </description>
      <author>pastest</author>
      <pubDate>Sat, 10 Jan 2015 03:46:10 GMT</pubDate>
    </item>
  </channel>
</rss>
```

- 分析要抓取的站点，获取站点的RSS链接
- 根据链接发起请求
- 各站点编码格式不一样 GBK,UTF-8等等，所以需要进行编码
- 动态解析拉取到的rss文章信息
- 格式化存入数据库
- 根据rss文章信息中的文章链接，请求文章具体内容
- 得到新闻页面的所有元素
- 在全文中查找新闻的正文
- 在正文中找到需要的图片

需要抓取的站点信息放到一个json文件中

```
{

    "channel":[
        {
            "from":"baidu",
            "name":"civilnews",
            "work":false,        //false 则不抓取
            "title":"百度国内最新新闻",
            "link":"http://news.baidu.com/n?cmd=4&class=
            "typeId":1
        },{
            "from":"netEase",
            "name":"rss_gn",
            "title":"网易最新新闻",
            "link":"http://news.163.com/special/00011K6L
            "typeId":2
        }
    ]
}
```

request(url,[option]);
这个是可以发送http请求的函数

```
1  var request = require('request')
2  var req = request(feed, {timeout: 10000, pool: false});
```

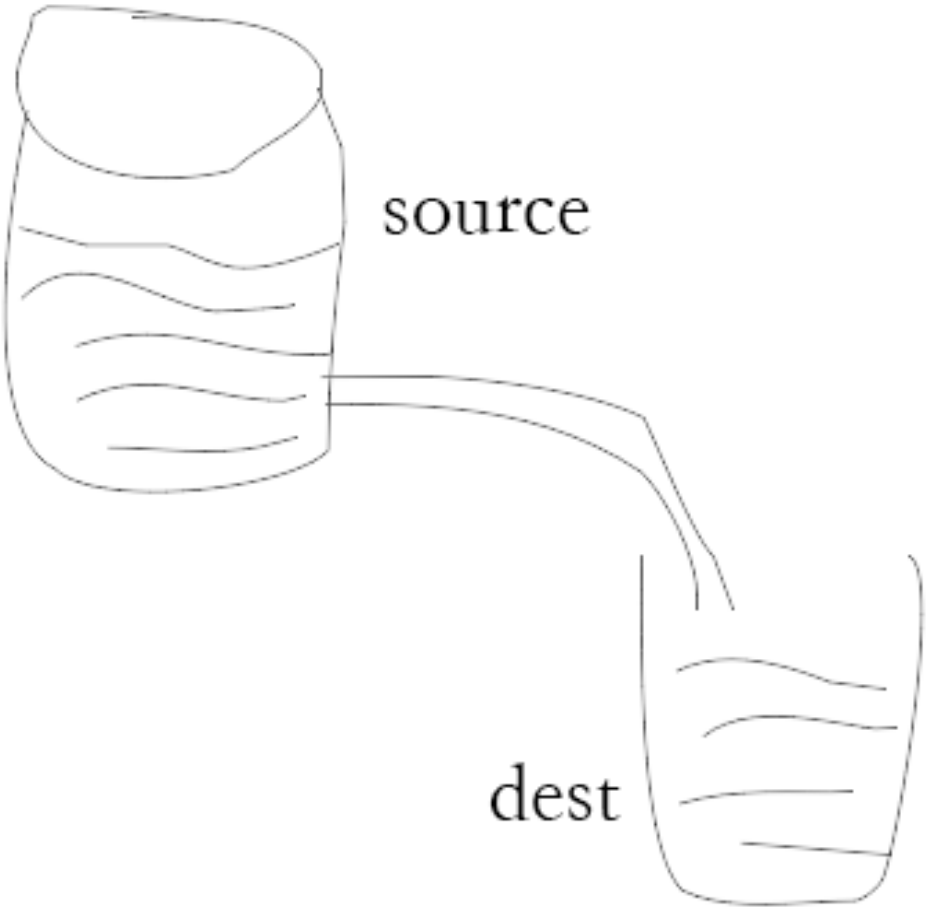
req 需要监听几个状态 response， error。

```
1  req.on('error', done);
2  req.on('response', function(res) {
3      //...
4  }
```

把非utf8的编码转换成utf8

```
var Iconv = require('iconv').Iconv;
charset = getParams(res.headers['content-type'] || '').charset;

iconv = new Iconv(charset, 'utf-8');
iconv.on('error', done);
stream = this.pipe(iconv);
```



Stream在nodejs中是EventEmitter的实现，
并且有多种实现形式，例如：
http responses request
fs read write streams
tcp sockets

.pipe()方法监听res的'data'、'end'、'error'等事件，
这样转换就可以边接收边进行转换

保存文章信息结构

```
1  var FeedParser = require('feedparser')
2  var feedparser = new FeedParser();
3
4  stream.pipe(feedparser);
5  feedparser.on('error', done);
6  feedparser.on('end', function(err){
7      //  postService.savePost(posts);
8  });
9  feedparser.on('readable', function() {
10     var post;
11     while (post = this.read()) {
12         posts.push(transToPost(post));
13     }
14 });
```

然后可以从存入的文章信息中获取文章url，拉取文章的具体内容

```
1 var req = request(url, {timeout: 10000, pool: false});
```

中文乱码问题解决

转换的技巧在于，使用buffer去转换，
把收到的数据全部变成buffer字节数据，
然后再统一转换为GBK

```
req.on('response', function(res) {
    var bufferHelper = new BufferHelper();
    res.on('data', function (chunk) {
        bufferHelper.concat(chunk);
    });
    res.on('end',function(){
        var result = iconv.decode(bufferHelper.toBuffer(),'GBK');
        calback(result);
    });
});
```

正文截取

html文件源码操作的神器 cheerio

它能让我像使用jquery一样方便快捷地操作抓取到的源码

```
1  var $ = cheerio.load(htmlData);
2  var context = $('.content').html();
3  var img = $('.content').find('img')[0];
```