简要说明一下两大板块（网页+爬虫）的代码实现过程

初始化数据库

通过 node.js 连接 mysql 数据库：

```
const mysql = require("mysql");
const pool = mysql.createPool({
    host: 'localhost',
    user: 'root',
    password: '密码',
    database: 'crawl1'
});
```

封装的访问模块：

```
const query = function (sql, sqlparam, callback) {
    pool.getConnection(function (err, conn) {
        if (err) {
            callback(err, null, null);
        } else {
            conn.query(sql, sqlparam, function (qerr, vals, fields) {
                conn.release(); //释放连接
                callback(qerr, vals, fields); //事件驱动回调
            });
        }
    });
};
const query_noparam = function (sql, callback) {
    pool.getConnection(function (err, conn) {
        if (err) {
            callback(err, null, null);
        } else {
            conn.query(sql, function (qerr, vals, fields) {
                conn.release(); //释放连接
                callback(qerr, vals, fields); //事件驱动回调
            });
        }
    });
};
exports.query = query;
exports.query_noparam = query_noparam;
```

创建 fetches 表：

```
✨ Active Connection
-- Active: 1688877588249@@127.0.0.1@3306@crawl MySQL
▷ Execute
CREATE TABLE `fetches` (
  `id_fetches` int(11)  NOT NULL AUTO_INCREMENT,
  `url` varchar(200) DEFAULT NULL,
  `source_name` varchar(200) DEFAULT NULL,
  `source_encoding` varchar(45) DEFAULT NULL,
  `title` varchar(200) DEFAULT NULL,
  `keywords` varchar(200) DEFAULT NULL,
  `author` varchar(200) DEFAULT NULL,
  `publish_date` date DEFAULT NULL,
  `crawltime` datetime DEFAULT NULL,
  `content` longtext,
  `createtime` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id_fetches`),
  UNIQUE KEY `id_fetches_UNIQUE` (`id_fetches`),
  UNIQUE KEY `url_UNIQUE` (`url`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

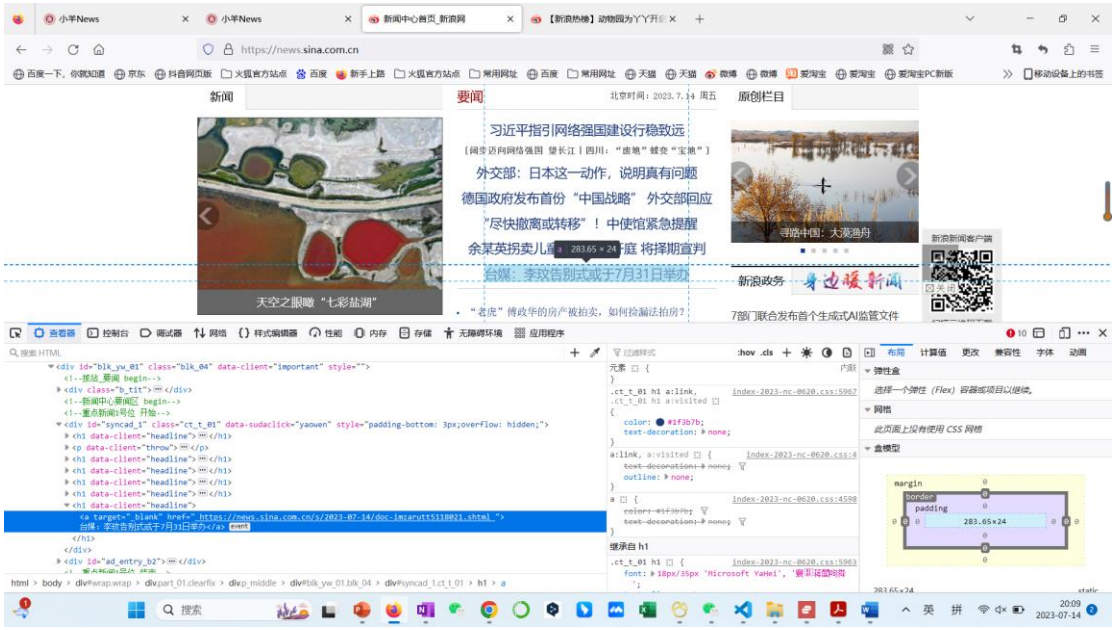
爬虫部分

爬虫板块我爬取了五个网站的新闻信息，分别是：

1、 新浪新闻　http://news.sina.com.cn/
2、 上海交通大学新闻学术网　https://news.sjtu.edu.cn/
3、 网易娱乐　https://ent.163.com/
4、 网易新闻　https://news.163.com/
5、 中国青年网　http://www.youth.cn/

下面以**新浪新闻**为例进行介绍，不逐一介绍全部的五个网站

首先分析将要爬取的新闻网页。在网站上右击“检查”和“查看页面源代码”可以查看要爬取的网页的源码，利用代码栏左上角的“查看器”按钮，在网页上选择想要审视的要素，就可以看到该要素对应的 html 源代码。利用这个方式，我们可以确定要爬取的内容的格式。




```
<!DOCTYPE html>
<!-- [ published at 2023-07-14 15:50:58 ] -->
<!-- LLTJ_MT:name ="界面新闻" -->

<html>
<head>
<meta charset="utf-8"/>
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge"/>
<meta name="sudameta" content="urlpath:s/; allCIDs:51924,257,51895,200856,56264,258,38790">

<title>台媒：李玟告别式或于7月31日举办_新浪新闻</title>
<meta name="keywords" content="台媒：李玟告别式或于7月31日举办" />
<meta name="tags" content="" />
<meta name="description" content="据台媒ETtoday新闻云7月14日报道，知情人士透露，李玟告别式传出将于7月31日在香港殡仪馆举行，并于隔天（8月1日..._新浪网" />
<link rel="mask-icon" sizes="any" href="//www.sina.com.cn/favicon.svg" color="red">
<meta property="og:type" content="news" />
<meta property="og:title" content="台媒：李玟告别式或于7月31日举办" />
<meta property="og:description" content="台媒：李玟告别式或于7月31日举办" />
<meta property="og:url" content="https://news.sina.com.cn/s/2023-07-14/doc-imzarutt5118021.shtml" />
<meta property="og:image" content="" />
<meta name="weibo: article:create_at" content="2023-07-14 15:50:57" />

<meta property="article:published_time" content="2023-07-14T15:50:57+08:00" />
<meta property="article:author" content="界面" />
```

新浪新闻的主页是我们本次爬取的种子页面，我们将解析出这个页面上的新闻页面超链接并对这些新闻页面进行爬取，根据需要爬取的数据修改 format 内容，以及用正则表达式匹配新闻 url。

使用的工具包：

```
var fs = require('fs');
var myRequest = require('request');
var myCheerio = require('cheerio');
var myIconv = require('iconv-lite');
require('date-utils');
var mysql = require('./mysql.js');//mysql在相同文件夹下的mysql.js
var schedule = require('node-schedule');//定时

var source_name = "新浪新闻";
var source = "新浪新闻";
var domain = 'http://news.sina.com.cn/';
var myEncoding = "utf-8";
var seedURL = 'http://news.sina.com.cn/';
```

其中，request 库是一个简单的客户端，用于发送 http 请求。cheerio 库是为服务器端实现的一个快速敏捷的 Jquery 核心库，在这个项目里用来解析 html。iconv-lite 库可以将编码格式由 GBK 转为 UTF-8，转换后的 html 就可以用cheerio 解析。fs 库是 file system library 的意思，是用来读取本地文件的。date-utils 库用于转换日期格式。node-schedule 库用来完成定时器功能。

构建访问模块

```
//防止网站屏蔽我们的爬虫
var headers = {
    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.65 Safari/537.36 Safari/537.36'
}

//request模块异步fetch url
function request(url, callback) {
    var options = {
        url: url,
        encoding: null,
        //proxy: 'http://x.x.x.x:8080',
        headers: headers,
        timeout: 10000 //
    }
    myRequest(options, callback)
};
```

定时执行

```
//! 定时执行
//var rule = new schedule.RecurrenceRule();
//var times = [0,2,4,6,8,10,12,14,16,18,19,22]; //每2小时自动执行
//var times2 = 14; //定义在第几分钟执行
//rule.hour = times;
//rule.minute = times2;

//定时执行httpGet()函数
//schedule.scheduleJob(rule, function() {
  //  seedget();
//});
```

爬取种子页面

在爬取种子页面时，本质上就是爬取新闻网页链接地址，供我们之后爬取新闻内容使用。网页链接一般在标签内。

重要的几点如下：

1、使用**seedURL_format = "$('a')"**选择所有的链接地址

2、检验新闻链接是否符合新闻 URL 的正则表达式，即判断改链接是否是一个有效的新闻地址（有可能出现 404 页面）。例如一个标准的新闻网址 https://news.sina.com.cn/gov/xlxw/2022-07-08/doc-imizirav2474084.shtml

3、值得注意的是，链接有些是 http，有些是 https，需要分类处理；

4、对每个有效的链接发起 request；

```
//function seedget() {
    request(seedURL, function (err, res, body) { //读取种子页面
        try {
            //用iconv转换编码
            var html = myIconv.decode(body, myEncoding);
            // console.log(html);
            // 准备用cheerio解析html
            var $ = myCheerio.load(html, { decodeEntities: true });
        } catch (e) { console.log('读种子页面并转码出错：' + e) };
        var seedurl_news;
        try {
            seedurl_news = eval(seedURL_format);
        } catch (e) { console.log('url列表所处的html块识别出错：' + e) };
        seedurl_news.each(function (i, e) { //遍历种子页面里所有的a链接
            var myURL = "";
            try {
                //得到具体新闻url
                var href = "";
                href = $(e).attr("href");
                if (href == undefined) return;
                if (href.toLowerCase().indexOf('https://') >= 0 || href.toLowerCase().indexOf('http://') >= 0) myURL = href; //http://开头的
                else if (href.startsWith('//')) myURL = 'https:' + href; ////开头的
                else myURL = seedURL.substr(0, seedURL.lastIndexOf('/') + 1) + href; //其他

            } catch (e) { console.log('识别种子页面中的新闻链接出错：' + e) }

            if (!url_reg.test(myURL)) return; //检验是否符合新闻url的正则表达式
            if (not_url_reg.test(myURL)) return;
            console.log(myURL);

            var fetch_url_Sql = 'select url from fetches where url=?';
            var fetch_url_Sql_Params = [myURL];
            mysql.query(fetch_url_Sql, fetch_url_Sql_Params, function (qerr, vals, fields) {
                // console.log(vals)
                if (!vals) {
                    console.log('vals=NULL')
                }
                else if (vals.length > 0) {
                    console.log('URL duplicate!')
                }
                else {
                    // window.addEventListener('load', newsGet(myURL), false);
                    newsGet(myURL); //读取新闻页面
                }
            });
        });
    });
//}
```

爬取内容页面

在内容页面，我们需要爬取的就是新闻具体内容，内容包括 title、author、keywords 等。因为不同的内容位于不同的标签内，并且有不同的 name，在之后我们要根据我们所需要爬取的内容，使用选择器选择我们要爬取的元素。爬取新浪新闻网需要用到的匹配 format，通过选择器，我们可以选择我们想要获取的元素内容。
常用的选择器如下：

- 元素选择器　$('title')
- 类选择器　$('.class')
- id 选择器　$('#author')
- 属性选择器　$('meta[name="author"]')

```
var source_name = "新浪新闻";
var source = "新浪新闻";
var domain = 'http://news.sina.com.cn/';
var myEncoding = "utf-8";
var seedURL = 'http://news.sina.com.cn/';

var seedURL_format = "$('a')";
var keywords_format = " $('meta[name=\"keywords\"]').eq(0).attr(\"content\")";
var title_format = "$('title').text()";
var date_format = " $('meta[property=\"article\:published\_time\"]').eq(0).attr(\"content\")";
var author_format = "$('.show_author').text()";
var content_format = "$('.article').text()";
var desc_format = " $('meta[name=\"description\"]').eq(0).attr(\"content\")";
var source_format = "$('.source').text()";

var url_reg = /.*\/(\d{4})-(\d{2})-(\d{2})\/.*(\d{7}).shtml/;
var not_url_reg = /.*http.*http.*/;

var regExp = /((\d{4}|\d{2})(\-|\/|\.)\d{1,2}\3\d{1,2})|(\d{4}年\d{1,2}月\d{1,2}日)/
let n = 0
```

newsGet 函数

该函数的作用为读取某一个有效 URL 的新闻内容，也是代码的核心部分。主要作用如下：

- 用 iconv 转换编码
- 用 cheerio 解析 html_news
- 判断是否是有效页面
- 通过选择器，读取元素内容
- 匹配日期格式
- 判断改页面是否已经被爬取过（查询数据库，判断数据库中是否已经存在）；
- 写入数据库

```
function newsGet(myURL) { //读取新闻页面
    request(myURL, function (err, res, body) { //读取新闻页面
        try {
            var html_news = myIconv.decode(new Buffer(body), myEncoding); //用iconv转换编码
            // console.log(html_news);
            //准备用cheerio解析html_news
            var $ = myCheerio.load(html_news, { decodeEntities: true });
            myhtml = html_news;
        } catch (e) {
            console.log('读新闻页面并转码出错：' + e);
            return;
        };

        console.log("转码读取成功:" + myURL);
        //动态执行format字符串，构建json对象准备写入文件或数据库
        var fetch = {};
        fetch.title = "";
        fetch.content = "";
        fetch.publish_date = (new Date()).toFormat("YYYY-MM-DD");
        //fetch.html = myhtml;
        fetch.url = myURL;
        fetch.source_name = source_name;
        fetch.source = source;
        fetch.source_encoding = myEncoding; //编码
        fetch.crawltime = new Date();

        if (keywords_format == "") fetch.keywords = source_name; // eval(keywords_format);  //没有关键词就用sourcename
        else fetch.keywords = eval(keywords_format);
        // console.log("&&&&&keywords: " + $('meta[name="keywords"]').attr("content"));

        if (title_format == "") fetch.title = ""
        else fetch.title = eval(title_format); //标题

        fetch.summary = fetch.title;

        // console.log(date_format);                //debug:$('meta[name="weibo: article:create_at"]').eq(0).attr("content")
        if (date_format != "") {
            // console.log("####\n");                    //debug:进入if
            // console.log(typeof date_format);          //debug:String
            // console.log(typeof eval(date_format));    //debug:undefined
            // var date = document.querySelector('meta[name=\"weibo: article:create_at\"]').getAttribute('content');
            // fetch.publish_date = date;
            fetch.publish_date = eval(date_format);
        }//刊登日期
        // console.log('@@@@@date: ' + $('meta[property="article:published_time"]').attr("content"));
        // console.log("#####description: " + $('meta[name="description"]').attr("content"));

        if (fetch.publish_date) {
            fetch.publish_date = regExp.exec(fetch.publish_date)[0];
            fetch.publish_date = fetch.publish_date.replace('年', '-')
            fetch.publish_date = fetch.publish_date.replace('月', '-')
            fetch.publish_date = fetch.publish_date.replace('日', '')
            fetch.publish_date = new Date(fetch.publish_date).toFormat("YYYY-MM-DD");
        }


        if (author_format == "") fetch.author = source_name; //eval(author_format);  //作者
        else fetch.author = eval(author_format);

        if (content_format == "") fetch.content = "";
        else fetch.content = eval(content_format).replace("\r\n" + fetch.author, ""); //内容,是否要去掉作者信息自行决定
        console.log("@@@@@@@content: " + fetch.content);

        //if (source_format == "") fetch.source = fetch.source_name;
        //else fetch.source = eval(source_format).replace("\r\n", ""); //来源

        if (desc_format == "") fetch.desc = fetch.title;
        else fetch.desc = eval(desc_format);
        if (fetch.desc) fetch.desc.replace("\r\n", ""); //摘要

        // console.log("content: " + fetch.content);
        if (fetch.content) {
            // var filename = source_name + "_" + (new Date()).toFormat("YYYY-MM-DD") +
            //     "_" + myURL.substr(myURL.lastIndexOf('/') + 1) + ".json";
            // ////存储json
            // fs.writeFileSync(filename, JSON.stringify(fetch));

            var fetchAddSql = 'INSERT INTO fetches(url,source_name,source_encoding,title,' +
                'keywords,author,source,publish_date,crawltime,summary,content) VALUES(?,?,?,?,?,?,?,?,?,?,?)';
            var fetchAddSql_Params = [fetch.url, fetch.source_name, fetch.source_encoding,
            fetch.title, fetch.keywords, fetch.author, fetch.source,fetch.publish_date,
            fetch.crawltime.toFormat("YYYY-MM-DD HH24:MI:SS"), fetch.summary,fetch.content
            ];

            //执行sql，数据库中fetch表里的url属性是unique的，不会把重复的url内容写入数据库
            mysql.query(fetchAddSql, fetchAddSql_Params, function (qerr, vals, fields) {
                if (qerr) {
                    console.log(qerr);
                    // con.on('error', function(err) {
                    //     console.log("[mysql error]",err);
                    //   });
                    return;
                    // console.log('qerr\n')

                }
            }); //mysql写入
        } else console.log("404 page not found.");
    });
}
```

学习心得：

在爬取不同的新闻页面时，代码的框架能够直接复用，但我们需要根据不同页面的实际情况，分析种子页面和新闻页面，使用选择器选择我们想要读取的 html 标签，从而获取内容。

网页部分

网站框架

- 网站后端使用 node.js express 框架
- 网站前端引入了少量 bootstrap 组件

Express 脚手架

Express 脚手架是一个规范的项目模版。通过使用脚手架，我们能够快速并且规范地使用 express 框架服务。
Express 脚手架目录如下：

- bin/：www 文件所在文件目录，我们通过运行 www 文件启动 Express；
- public/：通常用来存放浏览器拉取的公共资源，如图片等；
- routes/：index.js 所在文件目录，定义 Express 的响应路由；
- views：存放 Express 动态渲染的 HTML 页面，但在此文件后缀为.ejs，但作用与 HTML 相同；

Bootstrap

Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架，Bootstrap 包含了许多可重用的组件，用于创建图像、下拉菜单、导航、警告框、弹出框等等，能够帮助我们快速搭建 Web 前端网页。

Express 网站后端

数据库查询

- 选择查找的 column；
- 选择要匹配 like 的关键词；
- 选择升序和降序；
- 选择 LIMIT offset 查找范围；

```
let search = request.query.search;
let column = request.query.column;
const fetchSql = "select id_fetches,url,source_name,title,author,publish_date,summary as summary_ " +
    "from fetches where " + column + " like '%" + search + "%' order by publish_date desc LIMIT 10 " + "offset " + offset;
mysql.query_noparam(fetchSql, async function (err, result, fields) {
```

请求路由

```
router.post('/process_post', function (request, response, next) {
    console.log(request.body)
    let search = request.body.search;
    const fetchSql = "select id_fetches,url,source_name,title,author,publish_date,summary as summary_ " +
        "from fetches where title like '%" + search + "%' order by publish_date desc LIMIT 10";
    mysql.query_noparam(fetchSql, async function (err, result, fields) {
        response.render('main', {column: "title", search: search, content: result});
    });
});
```

网页渲染

- 通过 response.render 可以渲染 HTML 页面;
- 控制语句<% ？？？ %>;
- 渲染语句<%= ？？？ %>;

```
<ul>
    <% for(var i = 0;i < content.length;i++){ %>
        <li>
            <a href="/article?id=<%= content[i].id_fetches %>" class="contentBox" target="_blank">
                <strong><%= content[i].title %></strong>
                <p><%= content[i].summary_ %></p>
                <p><%= new Date(content[i].publish_date).toISOString().substring(0, 10) %></p>
            </a>
        </li>
    <% } %>
</ul>
```

网页前端

Boostrap 组件

组件引入

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>小羊News</title>
    <link rel="icon" href="images/icon.svg">
    <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css">
    <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js"></script>
    <script src="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>
    <link rel="stylesheet" href="stylesheets/style_main.css">
```

导航栏

```
<ul class="nav navbar-nav navbar-right">
    <li><a href="/"><strong>搜索</strong></a></li>
```

下拉菜单

```
<li class="dropdown">
    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
        <strong>时间热度统计</strong>
        <b class="caret"></b>
    </a>
    <ul class="dropdown-menu">
        <li><a target="_blank" href="/wordCloud">词云图</a></li>
        <li class="divider"></li>
        <li><a target="_blank"
               href="/linChart?column=<%= column %>&search=<%= search %>">折线图</a></li>
        <li class="divider"></li>
        <li><a target="_blank"
               href="/bar?column=<%= column %>&search=<%= search %>">直方图</a></li>
        <li class="divider"></li>
        <li><a target="_blank"
               href="/pie?column=<%= column %>&search=<%= search %>">饼图</a></li>
    </ul>
</li>
```

分页管理

```
<div class="pageControl">
    <ul class="pagination">
        <li class="page-item"><a class="page-link" href="/">Previous</a>
        </li>
        <li class="page-item"><a class="page-link"
                                 href="/process_page?column=<%= column %>&page=1&search=<%= search %>">1</a>
        </li>
        <li class="page-item"><a class="page-link"
                                 href="/process_page?column=<%= column %>&page=2&search=<%= search %>">2</a>
        </li>
        <li class="page-item"><a class="page-link"
                                 href="/process_page?column=<%= column %>&page=3&search=<%= search %>">3</a>
        </li>
        <li class="page-item"><a class="page-link" href="/">Next</a>
        </li>
    </ul>
</div>
```

新闻来源分布

```
<center>
<div id="main" style="width: 800px;height:400px;background-color: #81acb5 ;opacity: 0.8;"></div>
  <script type="text/javascript">
  var myChart = echarts.init(document.getElementById('main'));
  var option = {
      title: {
          text: '小羊News 来源分布'
      },
      tooltip: {},
      legend: {
          data:['条目数']
      },
      xAxis: {
          data: ["新浪新闻","交大新闻学术网","网易娱乐","网易新闻","中国青年网"]
      },
      yAxis: {},
      series: [{
          name: '数量',
          type: 'bar',
          color:' #3491bd',
          data: [172, 43, 57, 93, 87]
      }]
  };
  // 使用刚指定的配置项和数据显示图表。
  myChart.setOption(option);
  </script>
</center>
```

课上讲过的 Echarts

使用 Echarts 绘制图表（四种图的代码区别不大）；

- 词云图（绘制词云图前，需要使用 node segment 进行中文分词，并统计词频；）
- 折线图
- 直方图
- 饼图

```
<body>
<!--为ECharts准备一个具备大小（宽高）的Dom-->
<div id="main" style="width: 1500px;height:700px;"></div>
<script type="text/javascript">
```

异步加载

```
<script type="text/javascript">
    // 基于准备好的dom，初始化echarts实例
    var myChart = echarts.init(document.getElementById('main'), 'light');
    $.get("/get_data?column=<%= column %>&search=<%= search %>", function (data)
        myChart.setOption({
            title: {
                text: '时间热度分析'
            },
            legend: {
                orient: 'vertical',
                left: 'right',
                data: JSON.parse(data).name
            },
            series: [
                {
                    name: '热度分析',
                    type: 'pie',
                    radius: '55%',
                    roseType: 'angle',
                    data: JSON.parse(data)
                }
            ]
        })
    });
</script>
</body>
</html>
```

中文分词

```
router.get('/get_words_json', function (request, response) {
    const fetchSql = "select title,publish_date  from fetches order by publish_date desc LIMIT 1000";
    mysql.query_noparam(fetchSql, async function (err, result, fields) {
        // console.log(result)
        let Text = "";
        for (let i = 0; i < result.length; i++) {
            Text += result[i]['title'];
        }
        // 开始分词
        let words = segment.doSegment(Text, {stripPunctuation: true})
        // 统计词频
        let countResult = count(words)
        // console.log(countResult);
        response.write(JSON.stringify(countResult));
        response.end();
    });

    // response.render('wordCloud');
});
```

统计词频

```
function count(words) {
    let map = new Map();
    let keys = [];
    for (let i = 0; i < words.length; i++) {
        let word = words[i]
        if (map.has(word['w'])) {  // 如果有该key值
            map.set(word['w'], map.get(word['w']) + 1);
        } else {
            map.set(word['w'], 1);   // 如果没有该key值
            keys.push(word['w']);
        }
    }
    // console.log(keys)
    let result = [];
    for (let i = 0; i < keys.length; i++) {
        result.push({name: keys[i], value: map.get(keys[i])});
    }
    return result;
}

module.exports = router;
```

学习心得：

在本次 Project 中，主要使用 node.js 和 html 进行实现。node.js 中的 express 是一个使用非常方便的后端框架，并且 express 提供的脚手架已经能够被直接访问了，我们只需要添加新的路由信息和相应的处理函数即可快速搭建一个简单的 Web 网站，本次的 Web 网站前端由六个主要的 html 组成。