link: null
title: 珠峰架构师成长计划
description: db.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=52 sentences=224, words=1537

1.抓取数据

```
const request=require('request-promise');
const cheerio=require('cheerio');
const debug=require('debug')('juejin:task:read');
```

```
exports.tagList=async function (uri) {
    debug('读取文章标签列表');
    let options={
        uri,
        transform: function (body) {
            return cheerio.load(body);
        }
    }
    return request(options).then($ => {
            let tags= [];
            $('.item').each((i,item) => {
                let tag=$(item);
                let image=tag.find('div.thumb').first();
                let title=tag.find('.title').first();
                let subscribe=tag.find('.subscribe').first();
                let article=tag.find('.article').first();
                let name=title.text().trim();
                tags.push({
                    image: image.data('src').trim(),
                    name,
                    url:`https://juejin.im/tag/${encodeURIComponent(title.text().trim())}`,
                    subscribe: Number(subscribe.text().match(/(\d+)/)[1]),
                    article:Number(article.text().match(/(\d+)/)[1])
                });
                debug(`读取文章标签:${name}`);
            });
           return tags.slice(0,1);
    });
}
```

```
exports.articleList=async function (uri) {
    debug('读取博文列表');
    let options={
        uri,
        transform: function (body) {
            return cheerio.load(body);
        }
    }
    return request(options).then(async $ => {
        let articleList=[];
        let items =$('.item .title');
        for (let i=0;ilet article=$(items[i]);
            let href = article.attr('href').trim();
            let title=article.text().trim();
            let id=href.match(/\/(\w+)$/)[1];
            href='https://juejin.im'+href;
            let articleDetail = await readArticle(id,href);
            articleList.push({
                href,
                title,
                id,
                content:articleDetail.content,
                tags:articleDetail.tags
            });
            debug(`读取文章列表:${title}`);
        }
        return articleList;
    });
}
```

```
async function readArticle(id,uri) {
    debug('读取博文');
    let options={
        uri,
        transform: function (body) {
            return cheerio.load(body);
        }
    }
    return request(options).then($ => {
        let article=$('.main-container');
        let title=article.find('h1').text().trim();
        let content=article.find('.article-content').html();
        let tags=article.find('.tag-list-box>div.tag-list>a.item');
        tags=tags.map((index,item) => {
            let href = $(item).attr('href');
            return href? href.slice(4):href;
        })
        tags=Array.prototype.slice.call(tags);
        debug(`读取文章详情:${title}`);
        return {
            id,
            title,
            content,
            tags
        };
    });
}
```

```
let read = require('./read');
let write = require('./write');
(async function () {
    let tagUrl = 'https://juejin.im/subscribe/all';

    let tags = await read.tags(tagUrl);

    await write.tags(tags);
    let allAricles = {};

    for (tag of tags) {
        let articles = await read.articleList(tag.href);
        articles.forEach(article => allAricles[article.id] = article);
    }

    await write.articles(Object.values(allAricles));
    process.exit();
})()
```

2 表结构

字段 类型 说明 id int(11) 标签名称 name varchar(255) 标签名称 image varchar(255) 标签图片 url varchar(255) url地址 subscribe int(11) 订阅数 article int(11) 文章数

```
+
| Field     | Type          | Null | Key | Default | Extra          |
+
| id        | int(11)       | NO   | PRI | NULL    | auto_increment |
| name      | varchar(255)  | NO   |     | NULL    |                |
| image     | varchar(255)  | NO   |     | NULL    |                |
| url       | varchar(255)  | NO   |     | NULL    |                |
| subscribe | int(11)       | YES  |     | NULL    |                |
| article   | int(11)       | YES  |     | NULL    |                |
+
```

字段 类型 说明 id varchar(255) 文章ID title varchar(255) 文章名称 href varchar(255) 文章连接 content longtext 文章内容

```
+
| Field   | Type          | Null | Key | Default | Extra |
+
| id      | varchar(255)  | NO   | PRI | NULL    |       |
| title   | varchar(255)  | NO   |     | NULL    |       |
| content | longtext      | YES  |     | NULL    |       |
| href    | varchar(255)  | YES  |     | NULL    |       |
+
```

字段 类型 说明 article_id varchar(255) 文章ID tag_id int(11) 标签ID

```
+
| Field      | Type          | Null | Key | Default | Extra |
+
| article_id | varchar(255)  | NO   | PRI | NULL    |       |
| tag_id     | int(11)       | NO   | PRI | NULL    |       |
+
```

3. 写入数据库

db.js

```
const mysql=require('mysql');
var Promise = require('bluebird');
const connection = mysql.createConnection({
    host:            '127.0.0.1',
    port:            3306,
    database:        'juejin',
    user:            'root',
    password:        ''
});
connection.connect();
module.exports={
    query:Promise.promisify(connection.query).bind(connection),
    end:connection.end
}
```

crawl/task/write.js

```
const {query,end}=require('../db');
const debug=require('debug')('juejin:task:write');
```

```
exports.tagList=async function (tagList) {
    debug('保存文章标签列表');
    for (tag of tagList) {
        let oldTags=await query(`SELECT 1 FROM tags WHERE name=? LIMIT 1 `,[tag.name]);
        if (Array.isArray(oldTags)&&oldTags.length>0) {
            let oldTag=oldTags[0];
            await query(`UPDATE tags SET name=?,image=?,url=? WHERE id=?`,[tag.name,tag.image,tag.url,oldTag.id]);
        } else {
            await query(`INSERT INTO tags(name,image,url) VALUES(?,?,?)`,[tag.name,tag.image,tag.url]);
        }
    }
}
```

```
exports.articleList=async function (articleList) {
    debug('写入博文列表');
    debugger;
    for (article of articleList) {
        let oldArticles = await  query(`SELECT 1 FROM articles WHERE id=? LIMIT 1 `,article.id);
        if (Array.isArray(oldArticles)&&oldArticles.length>0) {
            let oldArticle=oldArticles[0];
            await query(`UPDATE articles SET title=?,content=?,href=? WHERE id=?`,[article.title,article.content,article.href,oldArticle.id]);
        } else {
            await query(`INSERT INTO articles(id,title,href,content) VALUES(?,?,?,?)`,[article.id,article.title,article.href,article.content]);
        }
        await query(`DELETE FROM article_tag WHERE article_id=? `,[article.id]);
        const where="('"+article.tags.join("','")+"')";
        const sql=`SELECT id FROM tags WHERE name IN ${where}`;
        let tagIds = await query(sql);
        for (row of tagIds) {
            await query(`INSERT INTO article_tag(article_id,tag_id) VALUES(?,?)`,[article.id,row.id]);
        }
    }
}
```

4. 建立web服务器查看数据

```
let express=require('express');
const path=require('path');
const {query}=require('../db');
const cronJob=require('cron').CronJob;
const debug=require('debug')('crawl:server');
const {spawn}=require('child_process');
let app=express();
app.set('view engine','html');
app.set('views',path.resolve('views'));
app.engine('html',require('ejs').__express);
app.get('/',async function (req,res) {
    let {tagId}=req.query;
    let tags=await query(`SELECT * FROM tags`);
    tagId=tagId||tags[0].id;
    let articles=await query(`SELECT a.* from articles a inner join article_tag  t on a.id = t.article_id WHERE t.tag_id =? `,[tagId]);
    res.render('index',{
        tags,articles
    });
});
app.get('/detail/:id',async function (req,res) {
    let id=req.params.id;
    let articles = await query(`SELECT * FROM articles WHERE id=? `,[id]);
    res.render('detail',{article:articles[0]});
});
app.listen(8080);
let job=new CronJob('*/5 * * * *',function () {
    debug('开始执行定时任务');
    let update= spawn(process.execPath,[path.resolve(__dirname,'update/index.js')]);
    update.stdout.pipe(process.stdout);
    update.stderr.pipe(process.stderr);
    updaste.on('close',function (code) {
        console.log('更新任务，代码=%d',code);
    });
});
job.start();

process.on('uncaughtException',function (err) {
    console.error('uncaughtException: %s',erro.stack);
});
```

```
<%- include header.html%>
<div class="container">
         <div class="row">
         <div class="col-md-2">
           <ul class="list-group">
              <%tags.forEach(tag=>{%>
                  <li class="list-group-item text-center">
                       <a href="/?tagId=">
                       <img style="width:25px;height:25px;" src=""/>
                       <%=tag.name%>
                   a>
                 li>
              <%})%>
           ul>
         div>
         <div class="col-md-10">
             <ul class="list-group">
              <%articles.forEach(article=>{%>
                  <li class="list-group-item">
                       <a href="/detail/">
                       <%=article.title%>
                   a>
                 li>
              <%})%>
           ul>
         div>
       div>
    div>
<%- include footer.html%>
```

```
<%- include header.html%>
    <div class="container">
         <div class="row">
         <div class="col-md-12">
             <div class="panel">
             <div class="panel-heading">
                 <h1 class="text-center"><%- article.title%>h1>
             div>
             <div class="panel-body">
                 <%- article.content%>
             div>
           <div>
         div>
       div>
    div>
<%- include footer.html%>
```

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="https://cdn.bootcss.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-
BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
    <title>博客列表title>
head>
<body>
<nav class="navbar navbar-default">
  <div class="container-fluid">

    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false">
        <span class="sr-only">Toggle navigationspan>
        <span class="icon-bar">span>
        <span class="icon-bar">span>
        <span class="icon-bar">span>
      button>
      <a class="navbar-brand" href="#">博客列表a>
    div>

    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li><a href="/">首页a>li>
      ul>
    div>
  div>
nav>
```

```
body>
html>
```

```
<%include header.html%>
    <div class="row">
        <div class="col-md-4 col-md-offset-4">
            <form method="POST">
                <input type="email" name="email" class="form-control" placeholder="请输入邮箱进行登录">
            form>
        div>
    div>
<%include footer.html%>
```

```
        .tag {
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
        }

                            关注
                                文章

                                type="checkbox" name="subscribe" value="
                               "> 关注
```

5. 订阅发布

```
app.get('/login', async function (req, res) {
    res.render('login', { title: '登录' });
});
app.post('/login', async function (req, res) {
    let { email } = req.body;
    let oldUsers = await query(`SELECT * FROM users WHERE email=?`, [email]);
    if (Array.isArray(oldUsers) && oldUsers.length > 0) {
        req.session.user = oldUsers[0];
        res.redirect('/');
    } else {
        let result = await query(`INSERT INTO users(email) VALUES(?)`, [email]);
        req.session.user = {
            id: result.insertId,
            email
        }
        res.redirect('/');
    }
});
app.get('/subscribe', async function (req, res) {
    let tags = await query(`SELECT * FROM tags`);
    let user = req.session.user;
    let selectedTags = await query(`SELECT tag_id from user_tag WHERE user_id = ?`, [user.id]);
    let selectTagIds = selectedTags.map(item => item['tag_id']);
    tags.forEach(item => {
        item.checked = selectTagIds.indexOf(item.id) != -1 ? 'true' : 'false';
    });
    res.render('subscribe', { title: '请订阅你感兴趣的标签', tags });
});
app.post('/subscribe', async function (req, res) {
    let { tags } = req.body;
    let user = req.session.user;
    await query(`DELETE FROM user_tag WHERE user_id=?`, [user.id]);
    for (let i = 0; i < tags.length; i++) {
        await query(`INSERT INTO user_tag(user_id,tag_id) VALUES(?,?)`, [user.id, parseInt(tags[i])])
    }
    res.redirect('/');
});
```

String.fromCodePoint() 静态方法返回使用指定的代码点序列创建的字符串。

```
let str = '时';
console.log(String.fromCodePoint('0x65F6'))
content = content.replace(//&#x(\w+?);/g, function (matched, point) {
   return String.fromCodePoint(`0x${point}`);
});
```