[1. Lodash 1](#_Toc7274)

[2. Pg-promise 1](#_Toc9542)

[3. Bluebird 1](#_Toc19627)

[4. request 1](#_Toc18117)

[5. Follow-redirects 4](#_Toc9485)

[6. https/http 4](#_Toc3105)

[Follow-redirects 5](#_Toc23051)

[问题： 5](#_Toc3080)

如何在字符串中动态地添加某变量。

Var name=”jack”;

Var prompt=`My name is ${name}`;

Console.log(prompt); //My name is jack

使用require(“”)来获取某个json文件，此时直接已经解析成变量。

# Lodash

函数：defaultsDeep(object, [sources])

对对象进行深度复制，对于sources中有object中不存在的，则复制到object中。

# Pg-promise

Node访问postgresql的模块

# 3. Bluebird

Promise.config()

u1.c1566.g1567.k1438553095041.pz

# 4. request

参考：<https://www.npmjs.com/package/request>

提交请求时，需要注意header中的content-type字段。不同的content-type，其做request请求时提交的请求参数和参数的组织形式不同。

Application/json

## Request.jar

管理是否使用cookie，可以不使用全局cookie，定制自己的cookie

对于post,get, delete, head,等函数，其接受的参数的形式为两种：

(url, options, callback)

或者：

(options,callback)

对于JSONObject中的参数有：

form: 该参数为json对象，不能使用JSON.stringify

formData: 同上

json:

body: 该参数必须是string或者buffer

RequestBodyStream：

headers:

proxy:

qs:

aws:

hawk:

httpSignature:

auth:为json对象。{user:””, pass:””, sendImmediately:””, bearer:””}

multipart:

time:

oauth:

ca:

agentClass:

agentOptions

forever:

requestBodyStream: 已经弃用，需要将request object传给stream.pipe

注意当request收到来自服务器的应答后会触发response事件，response的arguments是http.IncomingMessage的一个实例

Get

Put

Post

表单数据时，content-type支持application/x-www-form-urlencoded 和multipart/form-data、multipart/related

Content-type支持application/x-www-form-urlencoded 时：

request.post('http://service.com/upload', {form:{key:'value'}})

request.post('http://service.com/upload').form({key:'value'})

request.post({url:'http://service.com/upload', form: {key:'value'}}, function(err,httpResponse,body){ //处理响应})

'X-Requested-With': 'XMLHttpRequest',？

对象的toString()和JSON.stringify()区别

'Accept-Encoding':'gzip, deflate, sdch'

注意：

Ajax.Request

Post

Request.post(“url”,{form:formJsonData,},function(err, res, body){});

Request.post({url:””, body:bodyData,},function(err, res, body){});

Request.post({headers:headers, url:””, body:bodyData,},function(err, res, body){});

Body:”mes=heydude”

Form:{mes:”heydude”}

## http认证

<http://www.open-open.com/lib/view/open1435301679966.html>

## OAuth登陆

# 5. Follow-redirects

对于爬取大量网站的信息，由于请求引发的情况的多样性，适合使用该包，提供处理redirect的情况，同时获取数据后，对于response对象提供监听事件，可以动态的处理数据。如：

Http.request(url,function(response){

Response.on(“data”,function(){});

}).on(“error”,function(){});

# 6. https/http

7. Buffer

编码

8. BufferHelper

# Follow-redirects

该模块封装处理了转发

# mocha

Nodejs中的測試框架

https://cnodejs.org/topic/516526766d38277306c7d277

# 问题：

函数以参数形式传递和使用export的区别

# Pm2

# request模块爬虫数据实例

爬取数据实例：

介绍：爬取携程国际火车票（欧铁），相关网址：<http://rails.ctrip.com/international/PassFamilyIndex.aspx>

提交表单后跳转到实际页面，举例如下。

<http://rails.ctrip.com/international/OutiePTPList.aspx?departureDate=2017-3-22&starttime=&adult=1&child=0&youth=0&seniors=0&searchType=0&pageStatus=0&passHolders=0&from=FRPAR&to=ITFLR&arriveDate=>

此时可以查看返回的文件：<http://webresource.ctrip.com/ResTrainOnline/R9/Outie/JS/OutiePtpList.js?2017_3_9_16_40_48.js>。该文件中存在ajax请求。

$.ajax('./Ajax/QueryOutiePTPProd.ashx', {

method: 'POST',

async: true,

context: {

QueryParam: $.stringifyJSON(OutiePTPProcess.packetqueryparam())

},

onsuccess: function (result) {//实现省略}

Form表单提交的数据(即上述ajax请求中的OutiePTPProcess.packetqueryparam()返回的json字段。：

var QueryParam={

StartTime:"06:00",

BackTime:"06:00",

StartDate:"2017-3-23",

BackDate:"2017-03-26",

StartCityCode:"FRPAR",

ArriveCityCode:"ITFLR",

PassengerType:{

AdultCount:"2",

YouthCount:"0",

ChildCount:"0",

OldCount:"0"

},

PassHolders:"0",

LastStartDate:"",

StartCityName:"巴黎",

ArrivalCityName:"佛罗伦萨",

TrvalType:"1",

PageLoadGUID:data

};

使用nodejs的模块request，进行模拟ajax请求。

Var request=require(“request”);

Var headers={

User-Agent:””,

Cookie:””

};

var context={QueryParam: JSON.stringify(QueryParam)};

Request.post(url,{headers: headers, form: context, gzip: true}, function(err, res, body){

console.log(body); //此时的body即是所要的json数据。

});

注意：

提交请求时，提交的表单信息格式要根据其原网站设计传递的参数类型来传递，因为原系统提交的信息为JSON.stringify参数，故模拟传递时也要传stringify格式数据。

爬取数据时，产生输入图中验证码

南航：

正常请求网页，获取json数据，网页地址为http://b2c.csair.com/B2C40/query/jaxb/direct/query.ao

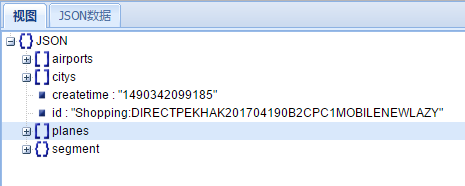
提交的参数为

json:{"depcity":"PEK", "arrcity":"DLC", "flightdate":"20170413", "adultnum":"3", "childnum":"0", "infantnum":"0", "cabinorder":"0", "airline":"1", "flytype":"0", "international":"0", "action":"0", "segtype":"1", "cache":"1", "preUrl":"", "isMember":""}

提交的参数中：flytype并非常量，不同的城市不同，暂未找到该值的来源。

提交的城市码信息中，北京市的城市码为PEK,而不是BJS

正常返回的数据格式



异常情况，其返回需要验证的response，格式如下

{"needverify": "true"}

此时浏览器发起另外的请求http://b2c.csair.com/B2C40/GenerateVerifyCodeServlet.do?r=0.14767951674547386，该请求为图片验证，

参数r的值为Math.random()生成。

## 参考文件：

Google的re-captcha

Google使用用户的上网行为来进行辨别

Github项目：<https://github.com/andelf/fuck12306>

注意图片验证请求，其响应位于客户端中，若错误，则生成新的图片验证，否则再次提交请求。此时的请求相当于新的请求。

问题：

如何

当正常请求时，返回需要验证的数据，如图片验证，此时进行 提交后，

## Hotel-api程序问题

问题描述：

创建了两个worker程序，四大航空公司每个worker请求一次的时间间隔为60s。两个worker的创建时间相近，可能存在两个worker进行同一个网站的请求的时间间隔在5s内，可能造成请求频繁，此时需要将做些改动：

方法1. 两个worker进程请求同一个网站的时间限制间隔为30s

方法2. 创建两个worker进程的时间相隔30s