

今日国内油价查询

产品亮点：实时查询国内当天的汽油柴油的价格。

API列表

查询今日国内油价

API基本信息

接口地址：<http://:80//gnyj>

描述：查询今日国内油价

认证方式：basic auth

方法：get

类型：路由/映射

请求参数：无

返回参数

名称	类型	说明
error_code	int	返回码，0为查询成功
reason	string	返回说明
result	string	返回结果集
city	string	城市
0h	string	0号柴油
98h	string	98号汽油
95h	string	95号汽油
92h	string	92号汽油

认证类型为basic auth的curl命令调用示例

```
curl -X GET -H "Authorization:Basic ZGVtbzAxOjVRC2hIR1cwXg==" "http://{ip}/{访问路径}/gnyj"
```

其中header中的key为"Authorization",内容为"Basic"+一个空格+base64加密(username:password)。

加密用到的认证参数请根据应用市场中的<已购资产>中的资产信息获取。

成功返回报文样例

```
{
  "result": [
    {
      "city": "北京",
      "0h": "6.82",
```

```
"98h": "8.57",
"95h": "7.59",
"92h": "7.13"
},
{
  "city": "河南",
  "0h": "6.76",
  "98h": "8.28",
  "95h": "7.62",
  "92h": "7.14"
},
{
  "city": "河北",
  "0h": "6.78",
  "98h": "8.35",
  "95h": "7.53",
  "92h": "7.12"
},
{
  "city": "天津",
  "0h": "6.78",
  "98h": "8.75",
  "95h": "7.53",
  "92h": "7.12"
},
{
  "city": "山西",
  "0h": "6.84",
  "98h": "8.35",
  "95h": "7.65",
  "92h": "7.09"
},
{
  "city": "山东",
  "0h": "6.77",
  "98h": "8.35",
  "95h": "7.63",
  "92h": "7.11"
},
{
  "city": "上海",
  "0h": "6.76",
  "98h": "8.25",
  "95h": "7.55",
  "92h": "7.10"
},
{
  "city": "福建",
  "0h": "6.77",
  "98h": "8.30",
  "95h": "7.58",
  "92h": "7.10"
},
{
  "city": "江苏",
  "0h": "6.74",
  "98h": "8.54",
  "95h": "7.56",
```

```
"92h": "7.11"  
},  
{  
  "city": "安徽",  
  "0h": "6.81",  
  "98h": "8.43",  
  "95h": "7.60",  
  "92h": "7.10"  
},  
{  
  "city": "浙江",  
  "0h": "6.76",  
  "98h": "8.28",  
  "95h": "7.56",  
  "92h": "7.11"  
},  
{  
  "city": "江西",  
  "0h": "6.82",  
  "98h": "8.62",  
  "95h": "7.62",  
  "92h": "7.10"  
},  
{  
  "city": "广西",  
  "0h": "6.84",  
  "98h": "8.56",  
  "95h": "7.77",  
  "92h": "7.20"  
},  
{  
  "city": "湖南",  
  "0h": "6.84",  
  "98h": "8.34",  
  "95h": "7.54",  
  "92h": "7.09"  
},  
{  
  "city": "广东",  
  "0h": "6.79",  
  "98h": "8.73",  
  "95h": "7.75",  
  "92h": "7.15"  
},  
{  
  "city": "湖北",  
  "0h": "6.77",  
  "98h": "8.52",  
  "95h": "7.65",  
  "92h": "7.14"  
},  
{  
  "city": "云南",  
  "0h": "6.85",  
  "98h": "8.49",  
  "95h": "7.81",  
  "92h": "7.28"  
},
```

```
{
  "city": "贵州",
  "0h": "6.88",
  "98h": "8.57",
  "95h": "7.67",
  "92h": "7.26"
},
{
  "city": "海南",
  "0h": "6.86",
  "98h": "9.90",
  "95h": "8.76",
  "92h": "8.25"
},
{
  "city": "重庆",
  "0h": "6.85",
  "98h": "8.57",
  "95h": "7.61",
  "92h": "7.20"
},
{
  "city": "四川",
  "0h": "6.84",
  "98h": "8.40",
  "95h": "7.73",
  "92h": "7.23"
}
],
"reason": "success!",
"error_code": 0
}
```

错误码参照

错误码	说明
10020	接口维护
10021	接口停用

代码示例分享

java:

```
package com.springboot.demo.controller;

import com.alibaba.fastjson.JSONObject;
import com.springboot.demo.service.TokenService;
import com.springboot.demo.utils.Base64Util;
import com.springboot.demo.utils.HttpUtil;
import io.swagger.annotations.Api;
import lombok.extern.slf4j.Slf4j;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
```

```

import java.util.HashMap;
import java.util.Map;

@Slf4j
@RestController
@Api(value = "调用API接口")
public class CallApiController {
    // basicauth的username
    public static String basic_user = "showcase";
    // basicauth的password
    public static String basic_password = "dw$40m&ke";
    public static void main(String[] args) {
        queryGnyj();
    }

    /**
     * 查询油价情况
     *
     */
    public static void queryGnyj() {
        String url = "http://{ip}/{访问路径}/gnyj";
        Map<String,String> headers = new HashMap<>();
        String token = String.format("Basic %s", basicAuthToken(basic_user,
basic_password));
        headers.put("Authorization", token);
        try {
            String response = HttpUtil.sendGet(url,headers);
            System.out.println("调用结果: " + response);
            JSONObject jsonObject = JSONObject.parseObject(response);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static String basicAuthToken(String userName, String password) {
        String base64Str = String.format("%s:%s", userName, password);
        return Base64Util.getBase64(base64Str);
    }
}

```

```

package com.springboot.demo.utils;

import lombok.extern.slf4j.Slf4j;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

import java.io.UnsupportedEncodingException;

@Slf4j
public class Base64Util {
    /**
     * 加密
     * @param str
     * @return
     */
}

```

```

public static String getBase64(String str) {
    byte[] b = null;
    String s = null;
    try {
        b = str.getBytes("utf-8");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
        log.error("base64 加密处理错误: {}", e);
    }
    if (b != null) {
        s = new BASE64Encoder().encode(b);
    }
    return s;
}

/**
 * 解密
 * @param s
 * @return
 */
public static String getFromBase64(String s) {
    byte[] b = null;
    String result = null;
    if (s != null) {
        BASE64Decoder decoder = new BASE64Decoder();
        try {
            b = decoder.decodeBuffer(s);
            result = new String(b, "utf-8");
        } catch (Exception e) {
            log.error("base64 解密处理错误: {}", e);
        }
    }
    return result;
}
}

```

页面调试样例

The screenshot displays a web application testing tool interface. On the left, the 'Request API' section is active, showing a GET request to 'http://192.168.1.49 /defaultapi/gmj'. The authentication method is set to 'Basic Auth' with the username 'showcase' and password 'dW940m8Xk'. Below this, there is a 'Headers' section with the label '请求头参数 Headers'. On the right, the 'Response Results' section shows a successful response with a status code of 200 and a response time of 287 ms. The response body is a JSON array of city names in Chinese, such as '北京', '天津', '上海', etc., each followed by its coordinates in a structured format.

调用

1. 在已购资产管理中，可以查看到买家已经订购的API商品信息

资产名称	资产类型	初次购买时间	到期时间	使用情况	操作
simpleWeather	API	2021-07-16 14:38:10	-	0/100	查看详情 ...
kafka-operator	Operator	2021-07-16 14:13:46	2021-08-15 14:16:52	-	查看详情 ...
07-16-01	Operator	2021-07-16 13:59:35	2021-08-15 13:59:42	-	查看详情 ...

2. API的认证方式调用说明

每个API上都会展示对应的认证类型和访问凭证，以basic auth为例，会展示username 和password,买家需要使用这对凭证进行API的调用。需要首先对username和password进行base64的加密，然后获取到一个字符换token，然后调用API的时候传入key为Authorization，值为token的header即可。具体的代码示例可以参考API列表中的代码使用示例章节，加密部分参考如下：

```
public static String basicAuthToken(String userName, String password) {
    String base64Str = String.format("%s:%s", userName, password);
    return Base64Util.getBase64(base64Str);
}

public static String getBase64(String str) {
    byte[] b = null;
    String s = null;
    try {
        b = str.getBytes("utf-8");
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
        log.error("base64 加密处理错误: {}", e);
    }
    if (b != null) {
        s = new BASE64Encoder().encode(b);
    }
    return s;
}
```

3.java使用sdk方式调用

HTTP SDK工具类，用来向服务端发送HTTP请求。如果提供认证参数，它能够在内部将请求消息进行签名处理，然后向API网关发送进行验证和调用，目前支持的认证类型有basic-auth, jwt, oauth2, hmac-auth, 支持的调用方式method有get, post, delete, patch, put, options, head。

请求API网关Restful服务的访问地址

网关地址访问格式：<http://{网关ip}:80/{api网关访问路径}>

- {ip} API网关的ip地址
- 默认的访问端口为 "80"，可以不填写
- 请求的context-path可以根据API网关的访问路径填写

工具包的下载地址

- 包下载：<https://github.com/icarhunter/api-gateway-sdk/>

HTTP Client SDK 使用方式

使用编程方式调用

```
import com.apigateway.sdk.HttpCaller;
import com.apigateway.sdk.dto.AuthParam;
import com.apigateway.sdk.dto.HttpReturn;
import com.apigateway.sdk.exception.HttpCallerException;
...

```

(1) 直接调用方式 POST 参数为json 认证为basic-auth，其中认证参数根据应用市场中的<已购资产>中的资产信息获取，如下的authType为<访问控制方式>如: basic-auth，如下的username和password在<访问凭证>的详情中获取。

```
String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1", "value1");
//设置json的请求参数
JSONObject requestJ = new JSONObject();
requestJ.put("key1", "value1");
requestJ.put("key2", "value2");
//设置认证参数，比如basic-auth类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("basic-auth");
//basic-auth 用户名
authParam.setUsername("ccc");
//basic-auth 密钥
authParam.setPassword("2sHG0~j&c");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType = "application/json;charset=utf-8";
try {
    HttpReturn httpReturn =
HttpCaller.doPost(url,headerParam,contentype, new
ContentBody(requestJ.toJSONString()),authParam);
    log.info(JSONObject.toJSONString(httpReturn));
} catch (HttpCallerException e) {
    e.printStackTrace();
    // error process
}

```

(2) 直接调用方式 POST 参数为Map 认证为jwt，其中认证参数根据应用市场中的<已购资产>中的资产信息获取，如下的authType为<访问控制方式>如: jwt，如下的key、secret和algorithm(如: HS256)在<访问凭证>的详情中获取。

```
String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1", "value1");
//设置Map的请求参数
Map<String, String> paramsMap = new HashMap<>();
paramsMap.put("key1", "value1");
paramsMap.put("key2", "value2");
//设置认证参数，比如jwt类型
AuthParam authParam = new AuthParam();

```



```

//认证类型
authParam.setAuthType("jwt");
//jwt 参数key
authParam.setKey("key");
//jwt 秘钥或者是私钥
authParam.setSecretOrPrivateKey("secret");
//加密算法, 包括: HS256, HS384, HS512, RS256, ES256
authParam.setAlgorithm("HS256");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType = "application/x-www-form-urlencoded;charset=utf-8";
try {
    HttpReturn httpReturn = HttpClient.doPost(url,
        headerParam,contentType,paramsMap,authParam);
    Log.info(JSONObject.toJSONString(httpReturn));
} catch (HttpClientException e) {
    e.printStackTrace();
    // error process
}

```

(3) 直接调用方式 POST 参数为xml 认证为hmac-auth, 其中认证参数根据应用市场中的<已购资产>中的资产信息获取, 如下的authType为<访问控制方式>如: hmac-auth, 如下的username、secret和algorithm(如: hmac-sha1)在<访问凭证>的详情中获取。

```

String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1","value1");
//设置xml的请求参数
Document document = DocumentHelper.createDocument();
Element root = document.addElement("root1");
Element key1 = root.addElement("key1");
Element key2 = root.addElement("key2");
key1.addText("value1");
key2.addText("value2");
String output = document.asXML();
//设置认证参数, 比如hmac-auth类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("hmac-auth");
//hmac-auth 用户名
authParam.setHmacUserName("username");
//hmac-auth 秘钥
authParam.setHmacSecret("secret");
//加密算法, hmac-sha1, hmac-sha256, hmac-sha384, hmac-sha512, 默认值是: hmac-
sha1
authParam.setHmacAlgorithm("hmac-sha1");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType="application/xml;charset=utf-8";
try {
    HttpReturn httpReturn = HttpClient.doPost(url,
        headerParam,contentType,new ContentBody(output),authParam);
    Log.info(JSONObject.toJSONString(httpReturn));
} catch (HttpClientException e) {
    e.printStackTrace();
    // error process
}

```

(4) 直接调用方式GET 认证为oauth2, 其中认证参数根据应用市场中的<已购资产>中的资产信息获取, 如下的authType为<访问控制方式>如: oauth2, 如下的clientId和clientSecret在<访问凭证>的详情中获取, scope参数在没有特殊说明时为空串即可。

```
String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1","value1");
//设置Map的请求参数 参数会拼接在url上, 参数也可以在url上拼接完成, 这里空Map即可
Map<String, String> paramsMap = new HashMap<>();
paramsMap.put("key1", "value1");
paramsMap.put("key2", "value2");
//设置认证参数, 比如oauth2类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("oauth2");
//oauth2 clientId
authParam.setClientId("clientId");
//oauth2 clientSecret
authParam.setClientSecret("clientSecret");
//oauth2 scope 根据情况可以不填
authParam.setScope("scope");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType="";
try {
    HttpReturn httpReturn = HttpClient.doGet(url,
        headerParam,contentType,paramsMap,authParam);
    log.info(JSONObject.toJSONString(httpReturn));
} catch (HttpClientException e) {
    e.printStackTrace();
    // error process
}
}
```

(5) 直接调用方式Put/Patch 注意参数类型可以参考Post中的详细demo

```
//Put 参数Map
HttpReturn httpReturn = HttpClient.doPut(url,
    headerParam,contentType,paramsMap,authParam);
//Put 参数json
HttpReturn httpReturn = HttpClient.doPut(url,headerParam,contentType, new
    ContentBody(requestJ.toJSONString()),authParam);
//Put 参数xml
HttpReturn httpReturn = HttpClient.doPut(url,headerParam,contentType, new
    ContentBody(output),authParam);
//Patch 参数Map
HttpReturn httpReturn = HttpClient.doPatch(url,
    headerParam,contentType,paramsMap,authParam);
//Patch 参数json
HttpReturn httpReturn = HttpClient.doPatch(url,headerParam,contentType, new
    ContentBody(requestJ.toJSONString()),authParam);
//Patch 参数xml
HttpReturn httpReturn = HttpClient.doPatch(url,headerParam,contentType, new
    ContentBody(output),authParam);
```

(6) 直接调用方式Head/Options 注意参数类型可以参考Get中的详细demo

```

//head
HttpReturn httpReturn = HttpClient.doHead(url,
    headerParam,contentType,paramsMap,authParam);
//options
HttpReturn httpReturn = HttpClient.doOptions(url,
    headerParam,contentType,paramsMap,authParam);

```

(7) 直接调用方式Delete 认证为oauth2，其中认证参数根据应用市场中的<已购资产>中的资产信息获取，如下的authType为<访问控制方式>如：oauth2，如下的clientId和clientSecret在<访问凭证>的详情中获取，scope参数在没有特殊说明时为空串即可。

```

String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1","value1");
//设置Map的请求参数
Map<String, String> paramsMap = new HashMap<>();
//设置认证参数，比如oauth2类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("oauth2");
//oauth2 clientId
authParam.setClientId("clientId");
//oauth2 clientSecret
authParam.setClientSecret("clientSecret");
//oauth2 scope 根据情况可以不填
authParam.setScope("scope");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType="";
try {
    HttpReturn httpReturn = HttpClient.doDelete(url,
        headerParam,contentType,paramsMap,authParam);
    log.info(JSONObject.toJSONString(httpReturn));
} catch (HttpClientException e) {
    e.printStackTrace();
    // error process
}

```

(8) 直接调用方式Get 下载文件

```

String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1","value1");
//设置Map的请求参数 参数会拼接在url上，参数也可以在url上拼接完成，这里空Map即可
Map<String, String> paramsMap = new HashMap<>();
paramsMap.put("key1", "value1");
paramsMap.put("key2", "value2");
//设置认证参数，比如oauth2类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("oauth2");
//oauth2 clientId
authParam.setClientId("clientId");
//oauth2 clientSecret
authParam.setClientSecret("clientSecret");

```

```

//oauth2 scope 根据情况可以不填
authParam.setScope("scope");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType="";
response.reset();
try {
    HttpReturn httpReturn = HttpClient.doGet(url,
        headerParam,contentType,paramsMap,authParam);
    httpReturn.getHeaderMap().get("fileName");
    log.info(JSONObject.toJSONString(httpReturn));
    response.setHeader("Content-Disposition", "attachment;

filename=\""+httpReturn.getHeaderMap().get("fileName")+"\";charset=utf-8");
    //取文件流
    byte[] bytes = httpReturn.getResponseBytes();
    OutputStream out = response.getOutputStream();
    InputStream br = new ByteArrayInputStream(bytes);
    byte[] buf = new byte[1024];
    int len = 0;
    while ((len = br.read(buf)) > 0)
        out.write(buf, 0, len);
    br.close();
    out.close();
} catch (HttpClientException e) {
    e.printStackTrace();
    // error process
}
}

```

(9) 直接调用方式Post 上传文件

```

String url="http://{ip}:80/{path}";
//设置header参数
Map<String, String> headerParam = new HashMap<>();
headerParam.put("header1","value1");
//设置Map的请求参数
Map<String, String> paramsMap = new HashMap<>();
//设置认证参数, 比如oauth2类型
AuthParam authParam = new AuthParam();
//认证类型
authParam.setAuthType("oauth2");
//oauth2 clientId
authParam.setClientId("clientId");
//oauth2 clientSecret
authParam.setClientSecret("clientSecret");
//oauth2 scope 根据情况可以不填
authParam.setScope("scope");
//设置请求content-type 也可以不填写 例如String contentType = ""
String contentType="";
//上传的文件
MultipartHttpServletRequest params = ((MultipartHttpServletRequest)
request);
paramsMap.put("key1", params.getParameter("key1"));
paramsMap.put("key2", params.getParameter("key2"));
List<MultipartFile> files = ((MultipartHttpServletRequest)
request).getFiles("fileKey");
MultipartFile file = null;
try {

```

```

        for (int i = 0; i < files.size(); ++i) {
            file = files.get(i);
            if (!file.isEmpty()) {
                InputStream in = file.getInputStream();
                HttpReturn httpReturn = HttpClient.doPost(url, headerParam,
"",
                paramsMap, file.getName(), in, ContentEncoding.none,
null);

                log.info(httpReturn.toString());
                return JSONObject.toJSONString(httpReturn);
            } else {
                log.info("You failed to upload " + i + " because the file was
empty.");
            }
        }
    } catch (HttpClientException e) {
        e.printStackTrace();
        // error process
    } catch (Exception e) {
        e.printStackTrace();
        // error process
    }
}

```

(10) 使用Builder的方式构造调用参数, 然后进行调用

```

import com.apigateway.sdk.HttpParameters;
import com.apigateway.sdk.HttpCaller;
import com.apigateway.sdk.HttpCallerException;

HttpParameters.Builder builder = HttpParameters.newBuilder();

builder.requestURL("http://ip:80/{path}") // 设置请求的URL
    .method("POST") // 设置调用方式
    .contentType("application/x-www-form-urlencoded;charset=utf-8"); //设置请求
content-type

// 设置请求参数
builder.putParamsMap("key1", "value1");
builder.putParamsMap("key2", "{\"a\":value1}"); // json format value
builder.putParamsMap("key3", "value1","value2","value3");//设置数组参数

//设置请求调用方式
builder.method("POST");

//设置透传的HTTP Headers
builder.putHeaderParamsMap("header1", "value1");
builder.putHeaderParamsMap("header2", "value2");
//设置认证参数 例如jwt
AuthParam authParam = new AuthParam();
authParam.setAuthType("jwt");
authParam.setKey("value1");
authParam.setSecretOrPrivateKey("value2");
authParam.setAlgorithm("HS256");
builder.authParam(authParam);
//进行调用 返回结果
String result = null;
try {

```

```
    HttpReturn res = HttpClienter.invokeReturn(builder.build()); //然后在res里获取相关的信息
    res.getResponseStr();//获取响应的文本串。
    res.responseBytes;//获取响应二进制数据，比如图片
} catch (HttpClienterException e) {
    // error process
}
```