介绍

针对uni.request请求做了一次封装，使用思维参考axios

引入与使用

```
import MsbUniRequest from '@/common/plugins/msbUniRequest';

//初始实例化
const msbRequest = new MsbUniRequest();
msbRequest.baseUrl = 'http://0.0.0.0:8080/msb/';//请求前缀

//项目中使用
//示例1
msbRequest.method({
  url:'/getGoods',
  method:'GET',
  data:{pageIndex:1,length:10}
});
//示例2 效果与示例1相同
msbRequest.get('/getGoods',{pageIndex:1,length:10})

```

API

method(option)

自定义请求，同uni.request

get(url,params,header)

get请求，接收3个参数

- url　　请求地址
- params　　请求参数
- header　　针对当前请求头设置特定的请求头。传了此参数request拦截器会失效

post(url, data, header)

post请求 接收3个参数

- url　　请求地址
- data　请求参数
- header　　针对当前请求头设置特定的请求头。传了此参数request拦截器会失效

put(url, data, header)

put请求，参数同post

delete(url, data, header)

delete请求，参数同post

use(hookName, callback)

实例hook注入器 ，接收2个参数
hookName　　要注入的拦截器，支持request / success / error 3个拦截器
callback 拦截器具体见拦截器说明

属性

baseUrl 请求地址前缀

拦截器

request (config):config => {}

请求前拦截，每次发起请求前会进入当前hook;
config参数是uni.request 的option 对象。可以重新对config进行编辑，最后返回新对象即可;
uni.request的第一个参数option都可以重置;

success (response) : data =>{}

请求成功结果拦截 请求200状态后会进入当前hook
response 是uni.request请求成功的response，在这可以做一些统一返回结果出来，一些业务错误都可以在这处理，最后把处理好的结果返回即可;

error（error) : error => {}

请求错误拦截，请求非200状态都会进入当前hook；
error 是uni.request的error 这这可以做统一请求错误处理

```

// 示例
 const msbReq = new MsbUniRequest();
 msbReq.baseUrl = 'xxxx'
 msbReq.use('request', (option)=>{
    // option 返回请求配置
    .....这里可以对option做一系列操作
   return option //最后返回一个正确的请求配置
 })
 msbReq.use('success', (res)=>{
     //res 返回请求结果
      let newRes = ..... //这里可以对请求结果做统一处理
     return newRes
 })

 msbReq.use('error', (error)=>{
     //error 返回错误结果
      let newError = ..... //这里可以对请求错误做统一处理
     return newError
 })
```