**‘’//并发请求：同时进行多个请求，统一处理返回值**

 //axios.all()  参数数组 请求

  //axios.spread()  在axios.all多个请求完成时候，把返回数据分割处理，在把每一个返回值统一处理

  axios.all(

      [

          axios.get('/data.json'),

          axios.get('/city.json'),

      ]

  ).then(

      axios.spread((dataRes,cityRes)=>{

            console.log(dataRes,cityRes)

      }) //回调函数

  )

  },

**axios创建实例**

  //后端接口地址有多个，并且超时时长不一样

  created(){

    let instance = axios.create({

        baseURL:'http:/localhost:8080',

        timeout:1000

    })

    let instance = axios.create({

        baseURL:'http://localhost:9090',

        timeout:5000

    })

    instance.get('/data.json').then(res=>{

        console.log(res)

    })

  }

**axios实例相关配置**

  created(){

      axios.create({

          // axios常用配置

          baseURL:'http://localhost:8080',//请求域名,基本地址

          timeout:100,//超时时间

          url:'/data.json',//请求路径

          method:'get,put,put,patch,delete',//请求方法

          headers:{  //设置请求头 接口需要登录鉴权，用token 识别当前登录人信息

                token:'' //

          },

          params:{}, //请求参数拼接在url

          data:{} //把请求参数放到请求体里

      })

      axios.get('/data/json',{

          params:{

          }

      })

       //优先级有低到高

      //1.axios 全局配置

      axios.defaults.timeout=1000

      axios.defaults.baseURL='http://localhost:8080'

      //2.axios 实例配置

      let instance = axios.create() //create不添加参数，实例配置就是全局配置

      instance.defaults.timeout =3000 //创建完实例 修改配置

      //3.axios 请求配置 数据多修改超时时长5000

      instance.get('/data.json',{

          timeout:5000

      })

  }

**axios实例常用参数配置具体使用方法**

//实际开发

//有两种请求接口： 或者俩个域名

//http:localhost:9090

//http:localhost:9091

//创建两个实例 instance instance1

let instance = axios.create({

    baseURL:'http://localhost:9090',

    timeout:1000

})

let instance1 = axios.create({

    baseURL:'http://localhost:9091',

    timeout:3000

})

//请求方法 用到哪些参数 ： baseUrl,timeout,url,method

instance.get('/contactList').then(res=>{

    console.log(res)

})

//数据量很大 无法满足接口 需要单独设置timeout:5000

//baserUrl,timeout:5000,url,method,params

instance1.get('/orderList',{

    timeout:5000

}).then(res=>{

    console.log(res)

})

**axios拦截器**

//拦截器：在请求或者响应被处理前拦截它们

//两种 1、请求拦截器    2、响应拦截器

//请求拦截器  有两个回调参数， 请求前config，请求后 err

axios.interceptors.request.use(config=>{

    //在发送请求前做些什么

    return config

},err=>{

    //请求错误游览器没有到达后端 404

    return Promise.reject(err)  //请求成功用Promise.resolve  请求错误Promise.reject

 })

 //响应拦截器 两个回调函数 1、请求成功 res  2、请求失败

 axios.interceptors.response.use(res=>{

     //请求成功

     return res

 },err=>{

     //响应错误差一个人 没有这个人http 500 到了返回都错误 响应错误

     //响应错误做些什么

     return Promise.reject(err)

 })

 axios.get().then(res=>{}).catch(err=>{})

//取消拦截器 (只需要了解)  ------begin---

 let interceptors = axios.interceptors.request.use

 (config=>{

     config.headers={

         auth:true

     }

     return config

 })

 axios.interceptors.request.eject(interceptors) //eject 声明一个变量 放拦截器 取消掉

//------end---

//例子 登录状态（token:''）  实际开发中一般都是给实例添加拦截器，添加属性  需要登录接口  ---- begin

let instance = axios.create()

instance.interceptors.request.use(config=>{

    config.headers.token='字符串'  //发送请求自动添加 不用手动添加

    return config

})

//---end---

//不需要登记 用他去访问不需要登录的接口

let newInstance = axios.create({})

//移动端开发

let instance\_phone = axios.create({})

instance\_phone.interceptors.request.use(config=>{

    //$('#model').show() 请求前显示

    return res

})

instance\_phone.interceptors.response.use(res=>{

    //$('#model').hid() 请求后隐藏

    return res

})