实验二十一　HTTP 请求及响应处理

一、实验目的

1. 了解 DevEco Studio 的使用
2. 学习 ArkTS 语言，HTTP 请求及响应处理
3. 编写代码
4. 编译运行
5. 在模拟器上运行

二、实验原理

1. 鸿蒙开发原理
2. ArkTS，ArkUI 开发原理
3. 鸿蒙应用运行原理

三、实验仪器材料

1. 计算机实训室电脑一台
2. DevEco Studio 开发环境及鸿蒙手机模拟器

四、实验步骤

HTTP 协议相关的资料需要提前了解。

HTTP 菜鸟教程：

https://www.runoob.com/http/http-tutorial.html

HTTP 协议详解：

https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Guides/Overview

本实验学习三种发起 HTTP 请求的方法：

- 使用 Network Kit

  https://developer.huawei.com/consumer/cn/doc/harmonyos-references-V5/network-api-V5

- 使用 Remote Communication Kit

  https://developer.huawei.com/consumer/cn/doc/harmonyos-guides-V5/remote-communication-introduction-V5

- 使用 ohos/axios 库

https://gitee.com/openharmony-sig/ohos_axios

https://gitcode.com/openharmony-sig/ohos_axios

(1) 先看 Network Kit（网络服务），这个包主要提供以下功能：

- **HTTP 数据请求**：通过 HTTP 发起一个数据请求。
- **WebSocket 连接**：使用 WebSocket 建立服务器与客户端的双向连接。
- **Socket 连接**：通过 Socket 进行数据传输。
- **网络连接管理**：网络连接管理提供管理网络一些基础能力，包括 WiFi/蜂窝/Ethernet 等多网络连接优先级管理、网络质量评估、订阅默认/指定网络连接状态变化、查询网络连接信息、DNS 解析等功能。
- **MDNS 管理**：MDNS 即多播 DNS（Multicast DNS），提供局域网内的本地服务添加、移除、发现、解析等能力。

使用网络管理模块的相关功能时，需要请求相应的权限:

权限名	说明
ohos.permission.GET_NETWORK_INFO	获取网络连接信息。
ohos.permission.INTERNET	允许程序打开网络套接字，进行网络连接。

（2）Remote Communication Kit（远场通信服务）是华为提供的 HTTP 发起数据请求的 NAPI 封装。应用通过 Remote Communication Kit 可便捷快速地向服务器发起数据请求。

使用 Remote Communication Kit 的主要业务流程如下：

- 应用客户端**创建会话**。
- 应用客户端**发起请求**。
- 应用客户端**接收请求结果，处理相应业务**。

应用在使用 Remote Communication Kit 能力前，需要检查是否已经获取对应权限。如未获得授权，需要声明对应权限。

Remote Communication kit 所需权限有(除取消网络请求，关闭会话，其余请求都需要权限)：

- ohos.permission.INTERNET：用于应用的权限，决定是否允许应用访问互联网。
- ohos.permission.GET_NETWORK_INFO：用于获取设备网络信息的 API 。

基础能力包括：

- 使用fetch发送网络请求
- 使用get发送网络请求
- 使用post发送网络请求
- 使用put发送网络请求
- 使用head发送网络请求
- 使用delete发送网络请求
- 使用cancel取消网络请求
- 使用close关闭会话

（3）第三方库 ohos/axios

Axios ，是一个基于 promise 的网络请求库，可以运行 node.js 和浏览器中。本库基于 Axios 原库 v1.3.4 版本进行适配，使其可以运行在 OpenHarmony，并沿用其现有用法和特性:

- http 请求
- Promise API
- request 和 response 拦截器
- 转换 request 和 response 的 data 数据
- 自动转换 JSON data 数据

下载安装

ohpm install @ohos/axios

需要权限

ohos.permission.INTERNET

我们主要通过实例来学习。

1. 打开 DevEco Studio，点击 Create Project 创建工程

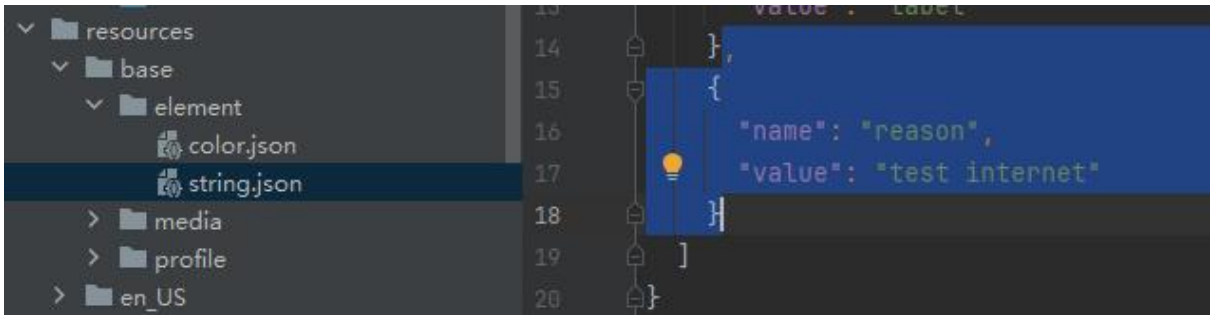
设置项目名称为 NetworkDemo。

2. Network Kit HTTP 数据请求

场景：应用通过 HTTP 发起一个数据请求，支持常见的 GET、POST、OPTIONS、HEAD、PUT、DELETE、TRACE、CONNECT 方法。

接口：HTTP 数据请求功能主要由 http 模块提供。使用该功能需要申请 ohos.permission.INTERNET 权限。

在 module.json5 配置文件的 requestPermissions 标签中声明权限。首先在 entry>src>main>resources>base>elemnt>string.json 文件中加上字符串 reason:




```
{
    "name": "reason",
    "value": "test internet"
}
```

然后在 module.json5 文件中加上代码：


```
"requestPermissions":[
```

```
      {
        "name" : "ohos.permission.INTERNET",
        "reason": "$string:reason",
        "usedScene": {
          "abilities": [
            "EntryAbility"
          ],
          "when":"inuse"
        }
      }
    ]
```

配置改动了，提示需要 sync now 就点击一下。

@ohos.net.http 数据请求的 API 文档：
https://developer.huawei.com/consumer/cn/doc/harmonyos-references-V5/js-apis-http-V5
在 Index.ets 中加入基础代码：

```
@Entry
@Component
struct Index {
  @State message: string = 'HTTP 测试';
  @State url: string = '';
  @State result: string = '';

  build() {
    Column() {
```

```
      Text(this.message)
        .fontSize(30)
        .margin('2%')

      TextInput({placeholder: '请输入 URL'})
        .width('75%')
        .margin({top: 10})
        .enableKeyboardOnFocus(false)
        .onChange((value: string) => {
          this.url = value;
        })

      Button('发送 GET 请求')
        .width('60%')
        .height(50)
        .margin({top: 10})
        .onClick(()=> {

        })

      TextArea({placeholder: '结果输出在这里', text: this.result})
        .width('75%')
```

```
        .height('50%')
        .margin({top: 30})
        .maxLines(50)
    }
    .height('100%')
    .width('100%')
  }
}
```

页面效果：输入一个 URL，发起 GET 请求，得到的结果输出在 TextArea 中。

然后，在按钮的 onClick 事件中添加代码，先尝试 Promise 的方式：

```
        .onClick(()=> {
          let httpRequest = http.createHttp();
          let promise = httpRequest.request(this.url, {
            method: http.RequestMethod.GET,
            connectTimeout: 60000,
            readTimeout: 60000,
            header: 'application/json'
```

```
        });
        promise.then((data:http.HttpResponse) => {
          console.info('Result:' + data.result);
          console.info('code:' + data.responseCode);
          console.info('type:' + JSON.stringify(data.resultType));
          console.info('header:' + JSON.stringify(data.header));
          console.info('cookies:' + data.cookies); // 自 API version 8 开始支持 cookie
          console.info('header.content-Type:' + data.header);
          console.info('header.Status-Line:' + data.header);
          this.result = data.result.toString();
        }).catch((err:Error) => {
          console.info('error:' + JSON.stringify(err));
        });
      })
```

当然，前面需要导入：

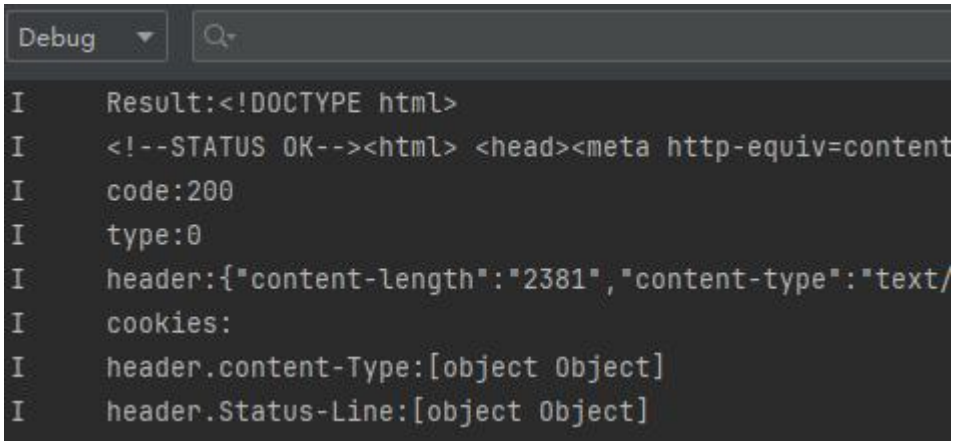
```
import { http } from '@kit.NetworkKit';
```

查看文档，可以了解 request 接口开发步骤：

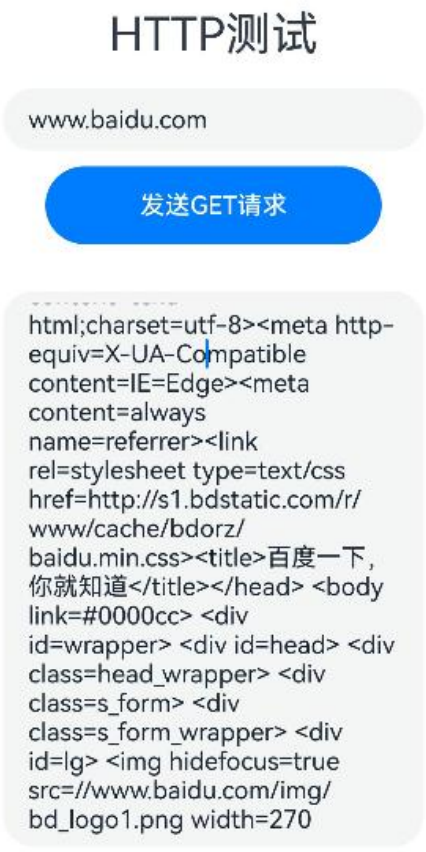
- 从@kit.NetworkKit 中导入 http 命名空间。
- 调用 createHttp()方法，创建一个 HttpRequest 对象。
- 调用该对象的 on()方法，订阅 http 响应头事件，此接口会比 request 请求先返回。可以根据业务需要订阅此消息。
- 调用该对象的 request()方法，传入 http 请求的 url 地址和可选参数，发起网络请求。
- 按照实际业务需要，解析返回结果。
- 调用该对象的 off()方法，取消订阅 http 响应头事件。

- 当该请求使用完毕时，调用 destroy()方法主动销毁。

红色字体的部分是我们实现了的。在文本框中输入“www.baidu.com”，然后点击按钮，先查看 Log：



界面：



再尝试通过 callback 方式作为异步方法，删除掉 onClick 中的代码，替换为：

```
        .onClick(()=> {
```

```
        let httpRequest = http.createHttp();
        httpRequest.request(this.url, (err: Error, data: http.HttpResponse) => {
          if (!err) {
            console.info('Result:' + data.result);
            console.info('code:' + data.responseCode);
            console.info('type:' + JSON.stringify(data.resultType));
            console.info('header:' + JSON.stringify(data.header));
            console.info('cookies:' + data.cookies);
            this.result = data.result.toString();
          } else {
            console.info('error:' + JSON.stringify(err));
          }
        });
      })
```

尝试 www.qq.com，Log：

界面：

查看代码，注意这里用的是 callback 的方式。

3. Remote Communication Kit HTTP 数据请求

首先，添加权限，前面已经添加了 ohos.permission.INTERNET，现在再添加 ohos.permission.GET_NETWORK_INFO，到 module.json5 中添加：

```
    {
        "name" : "ohos.permission.GET_NETWORK_INFO",
        "reason": "$string:reason",
        "usedScene": {
          "abilities": [
            "EntryAbility"
          ],
          "when":"inuse"
        }
```

```
    }
```

注意添加逗号，以及有提示 sync now 就点击一下同步。

修改 Index.ets 文件，首先导入：

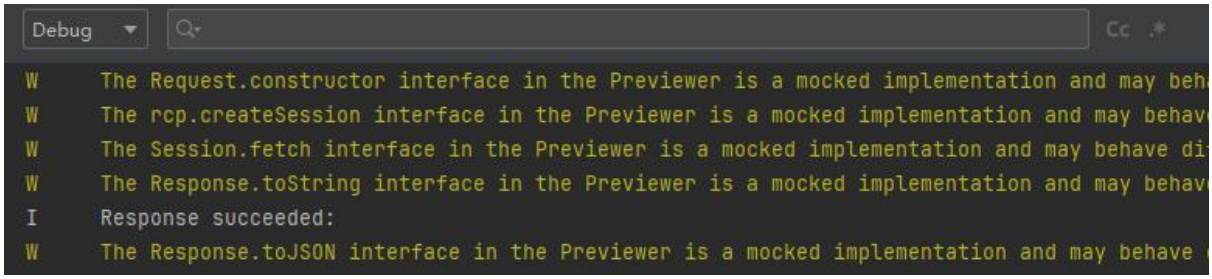
```
import { rcp } from '@kit.RemoteCommunicationKit';
import { BusinessError } from '@kit.BasicServicesKit';
```

修改 onClick 事件中的代码：

```
          .onClick(()=> {
            let kHttpServerAddress = this.url;
            // 创建 Request 对象
            const request = new rcp.Request(kHttpServerAddress, "GET");
            // 创建会话
            const session = rcp.createSession();
            // 发起请求，并处理返回结果
            session.fetch(request).then((rep: rcp.Response) => {
              console.info(`Response succeeded: ${rep}`);
              if (rep!=null) {
                this.result = `${rep}`;
              }
            }).catch((err: BusinessError) => {
              console.error(`Response err: Code is ${err.code}, message is ${JSON.stringify(err)}`);
```

```
        });
    })
```

在 Previewer 上：（注意要加上 http://或 https://，否则认为 URL 错误）



看起来 Previewer 不支持。

在模拟器上运行：（注意要加上 http://或 https://，否则认为 URL 错误，导致 jscrash）

再尝试使用 get 方法，onClick 中的代码改为：

```
        .onClick(()=> {
          let kHttpServerAddress = this.url;
          // 创建 Request 对象
          const session = rcp.createSession();
          session.get(kHttpServerAddress).then((response) => {
            console.info(`Response succeeded: ${response}`);
            this.result = `${response}`;
          }).catch((err: BusinessError) => {
            console.error(`Response err: Code is ${err.code}, message is ${JSON.stringify(err)}`);
          });
        })
```

效果是一样的。

遇到一个疑问，输入百度网址的时候，Log 打出来是这样的，很短：

```
Response succeeded: <!DOCTYPE html>
```

不清楚是什么原因。是不是需要定义一下 header?

网上有很多开放的 API 资源，我们尝试使用网站 wanandroid.com 上的 openapis:

https://www.wanandroid.com/openapis

尝试访问：

https://www.wanandroid.com/article/list/0/json

得到结果：

再尝试一下 POST 请求，用登录作为例子，在 wanandroid 上注册一个账号，然后修改 Index.ets 的代码为：

```
import { rcp } from '@kit.RemoteCommunicationKit';
import { BusinessError } from '@kit.BasicServicesKit';


@Entry
@Component
struct Index {
  @State message: string = 'HTTP 测试';
  @State url: string = '';
  @State result: string = '';
```

```
  build() {
    Column() {
      Text(this.message)
        .fontSize(30)
        .margin('2%')

      TextInput({text: $$this.url, placeholder: '请输入 URL'})
        .width('75%')
        .margin({top: 10})
        .enableKeyboardOnFocus(false)
        .onChange((value: string) => {
          this.url = value;
        })

      Button('发送 POST 请求')
        .width('60%')
        .height(50)
        .margin({top: 10})
        .onClick(()=> {
          // 定义头信息，其他参数请看 API 参考
          const requestHeaders: rcp.RequestHeaders = {
            'accept': 'application/json'
```

```
        }

        // 在此处携带头信息
        const session = rcp.createSession({ headers: requestHeaders});
        // 定义请求的 URL，地址请开发者自行定义
        const requestURL = "https://www.wanandroid.com/user/login";
        this.url = requestURL;

        // 定义 requestContent，请求部分的正文内容
        const requestContent: rcp.RequestContent = {
            "username": "hechu",
            "password": "abcd1234"
        }
        session.post(requestURL, requestContent).then((response) => {
          // 请求成功处理，可利用 Response.toJSON 将响应转换成 JSON 格式
          console.info(`Response succeeded: ${response}`);
          this.result = `${response}`;
        }).catch((err: BusinessError) => {
          // 请求失败处理
          console.error(`Response err: Code is ${err.code}, message is ${err.message}`);
        });
      })
```

```
        TextArea({placeholder: '结果输出在这里', text: this.result})
            .width('75%')
            .height('50%')
            .margin({top: 30})
            .maxLines(50)
    }
    .height('100%')
    .width('100%')
  }
}
```

注意红色字体的代码：

- text: $$this.url 是把 this.url 和文本输入框的值双向绑定

- requestContent 是 POST 带上去的数据，这里根据文档，输入了用户名和密码（错误的密码。即使输入是正确的，因为密码不能明文传送，所以这样发送也不会得到正确的结果。）

```
https://www.wanandroid.com/user/login

方法：POST
参数：
        username，password
```

结果：

```
Response succeeded: {"data":null,"errorCode":-1,"errorMsg":"账号密码不匹配！"}
```

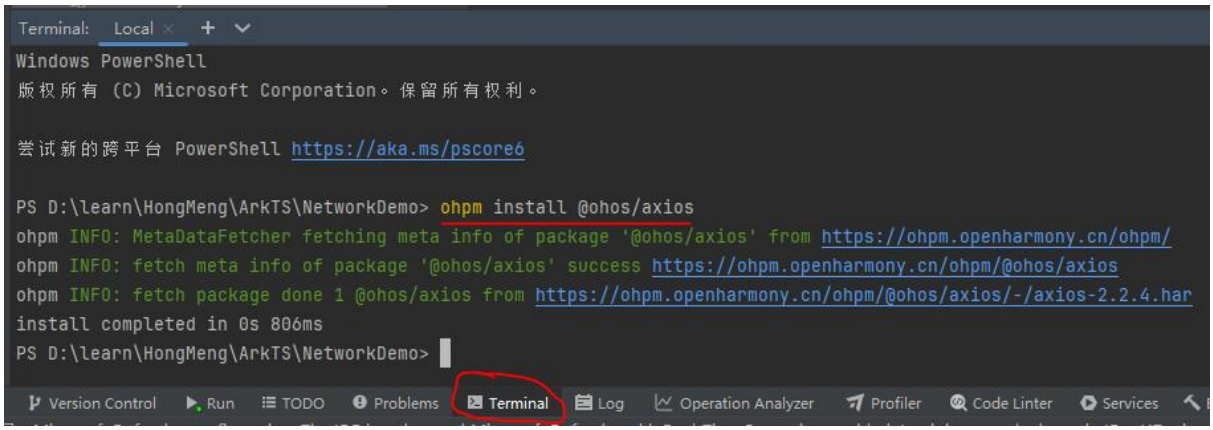
POST 发送成功，也得到了正确的返回结果。

4. @ohos/axios HTTP 数据请求

首先，前面已经添加了 ohos.permission.INTERNET，无需再添加。

然后，通过 ohpm 安装@ohos/axios 库，打开 Terminal，输入命令：

> ohpm install @ohos/axios



把 Index.ets 中的代码改为：

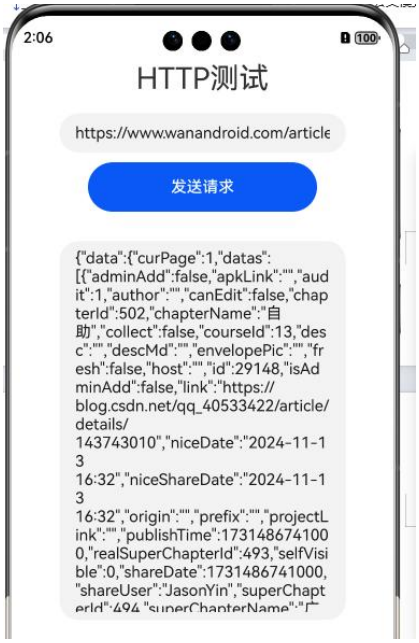
```
import axios, { AxiosError, AxiosResponse } from '@ohos/axios';


@Entry
@Component
struct Index {
  @State message: string = 'HTTP 测试';
  @State url: string = 'https://www.wanandroid.com/article/list/0/json';
  @State result: string = '';

  build() {
```

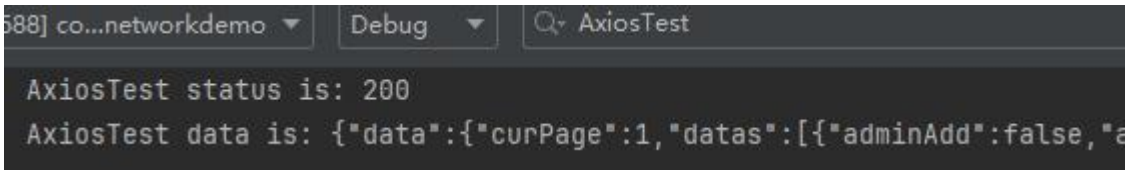
```
    Column() {
      Text(this.message)
        .fontSize(30)
        .margin('2%')

      TextInput({text: $$this.url, placeholder: '请输入 URL'})
        .width('75%')
        .margin({top: 10})
        .enableKeyboardOnFocus(false)

      Button('发送请求')
        .width('60%')
        .height(50)
        .margin({top: 10})
        .onClick(()=> {
          const instance = axios.create();
          instance.get<string, AxiosResponse<string>, null>(this.url, {
            connectTimeout: 20000
          }).then((res: AxiosResponse) => {
            console.info(`AxiosTest status is: ${res ? res.status : ''}`);
            console.info(`AxiosTest data is: ${res ? JSON.stringify(res.data) : ''}`) ;
            this.result = `${res ? JSON.stringify(res.data) : 'nothing'}`
```

```
            }).catch((err: AxiosError) => {
                console.info(`AxiosTest err is: ${err.message}`);
            })
        })

        TextArea({placeholder: '结果输出在这里', text: this.result})
            .width('75%')
            .height('50%')
            .margin({top: 30})
            .maxLines(50)
    }
    .height('100%')
    .width('100%')
  }
}
```

在 Previewer 上也可以运行：

在模拟器上运行：





也可以使用通用的 request 方法，在 method 中设定是 get, post 或其他的方法，修改代码：

```
        .onClick(()=> {
            const instance = axios.create();
            instance.request<string, AxiosResponse<string>, null>({
                url: this.url,
                method: 'get',
                connectTimeout: 20000
            }).then((res: AxiosResponse) => {
                console.info(`AxiosTest status is: ${res ? res.status : ''}`);
```

结果是一样的。

上面的代码中，也可以将 instance.request 直接改为 axios：

结果也是一样的。

再尝试发送 POST。首先把 url 改为：

```
@State url: string = 'https://www.wanandroid.com/user/login';
```

在@Entry 前面加一个 interface:

```
interface UserInfo {
  username: string
  password: string
}
```

因为 POST 中的 data 必须有一个对应的类型，用来保存账号和密码（注意修改成你自己的）。

把 onClick 事件中的部分代码改为：

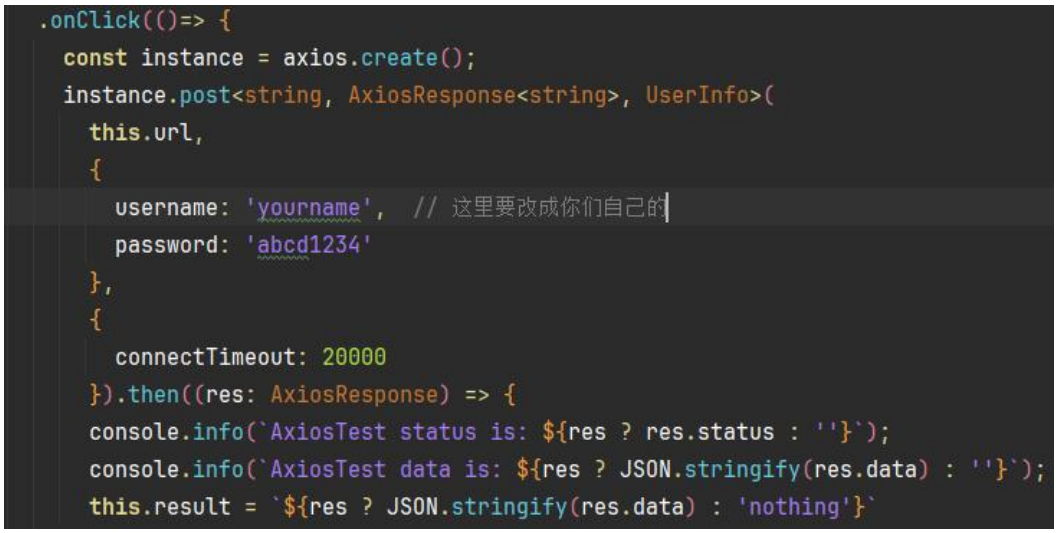
```
        .onClick(()=> {
          const instance = axios.create();
          instance.request<string, AxiosResponse<string>, UserInfo>({
            url: this.url,
            method: 'post',
            data: {
              username: 'your name',
```

```
                password: 'abcd1234'
            },
            connectTimeout: 20000
        }).then((res: AxiosResponse) => {
            console.info(`AxiosTest status is: ${res ? res.status : ''}`);
        })
    })
```

刷新 Previewer，点击按钮，结果：

可以看到，POST 请求发送成功。

也可以直接调用 post:



```
.onClick(()=> {
  const instance = axios.create();
  instance.post<string, AxiosResponse<string>, UserInfo>(
    this.url,
    {
      username: 'yourname',  // 这里要改成你们自己的
      password: 'abcd1234'
    },
    {
      connectTimeout: 20000
    }).then((res: AxiosResponse) => {
    console.info(`AxiosTest status is: ${res ? res.status : ''}`);
    console.info(`AxiosTest data is: ${res ? JSON.stringify(res.data) : ''}`);
    this.result = `${res ? JSON.stringify(res.data) : 'nothing'}`
```

代码：

```
.onClick(()=> {
  const instance = axios.create();
  instance.post<string, AxiosResponse<string>, UserInfo>(
    this.url,
    {
      username: 'yourname',
      password: 'abcd1234'
    },
    {
      connectTimeout: 20000
  }).then((res: AxiosResponse) => {
    console.info(`AxiosTest status is: ${res ? res.status : ''}`);
```

到目前为止，我们尝试了三种发送 HTTP 请求的方法：

- Network Kit 的 HTTP 方法

- Remote Communication Kit 的方法
- 第三方库：@ohos/axios 库

大家可以根据自己的喜好选择其中一种。看起来官方是推荐使用 Remote Communication Kit。

五、实验注意事项

1. 注意教师的操作演示。

2. 学生机与教师机内网连通，能接收和提交实验结果。

3. 按实验要求输入测试数据，并查看输出结果是否符合实验结果。

六、思考题

1. 通过这个实验，你学到了什么？