1. **AsyncHttpHarmony功能介绍**

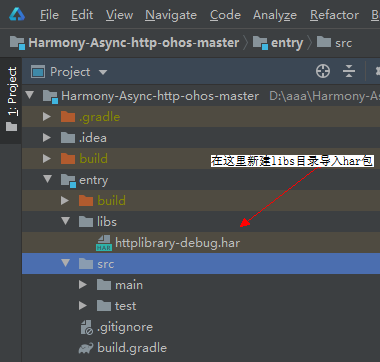
我们在做鸿蒙应用开发过程，用java原生的HttpsURLConnection实现网络请求很难高效的达到预期的效果。我们需要处理数据解析，子线程通知UI线程更新。这里我们就基于三方原生的网络框架AsyncHttpClient二次封装更加高效实现网络请求及数据解析。同时鸿蒙为我们提供了一个TaskDispatcher类派发同步任务达到更新UI的效果

1. **AsyncHttpHarmony使用方法**
   1. **为应用添加httplibrary-debug.har包依赖**

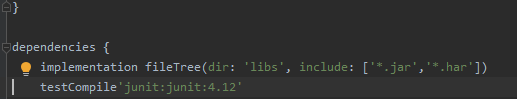
在应用模块中调用HAR，常用的添加依赖的方式包括如下两种。

* 方式一：依赖本地HAR

第一步：将httplibrary-debug.har复制到entry\libs目录下即可（由于build.gradle中已经依赖的libs目录下的\*.har，因此不需要在做修改）。

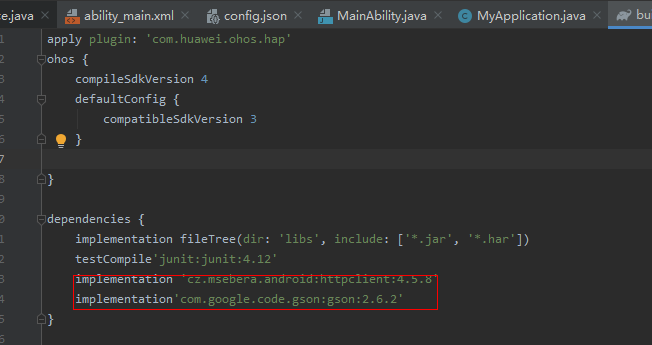


查看工程目录中build.gradle下的\*.har是存在



第二步：除了依赖以上har之外还需要添加外部依赖用来实现Header类的引入，引入方式如下图，引入完之后同步下就可以使用。

implementation 'cz.msebera.android:httpclient:4.5.8'  
implementation'com.google.code.gson:gson:2.6.2'



以上操作无误之后就可以进行编码了！

* 1. **主页面的布局文件**

定义一个Text文本用来显示请求返回的数据，定义一个text实现请求点击事件

<?xml version="1.0" encoding="utf-8"?>  
<DirectionalLayout  
 xmlns:ohos="http://schemas.huawei.com/res/ohos"  
 ohos:height="match\_parent"  
 ohos:width="match\_parent"  
 ohos:orientation="vertical">  
  
 <Text  
 ohos:id="$+id:tvResult"  
 ohos:height="match\_content"  
 ohos:width="match\_content"  
 ohos:background\_element="$graphic:background\_ability\_main"  
 ohos:layout\_alignment="horizontal\_center"  
 ohos:text="数据显示"  
 ohos:text\_size="50"  
 ohos:top\_margin="180vp"  
 />  
 <Text  
 ohos:id="$+id:tvRequest"  
 ohos:height="match\_content"  
 ohos:width="match\_content"  
 ohos:background\_element="$graphic:background\_ability\_main"  
 ohos:layout\_alignment="horizontal\_center"  
 ohos:text="get请求"  
 ohos:text\_size="50"  
 ohos:top\_margin="80vp"  
 />  
  
</DirectionalLayout>

* 1. **MainAbilitySlice代码如下**

核心代码是initListener，其中声明了一个 AsyncHttpClient对象，设置请求参数，然后调用get方法获取ulr返回结果，然后通过TaskDispatcher类派发同步任务达到更新UI的效果，代码如下：

package com.huawei.asynchttpharmonyos.slice;  
  
import com.example.httplibrary.utils.AsyncHttpClient;  
import com.example.httplibrary.utils.JsonHttpResponseHandler;  
import com.example.httplibrary.utils.RequestParams;  
import com.huawei.asynchttpharmonyos.ResourceTable;  
import cz.msebera.android.httpclient.Header;  
import ohos.aafwk.ability.AbilitySlice;  
import ohos.aafwk.content.Intent;  
import ohos.agp.components.Component;  
import ohos.agp.components.Text;  
import ohos.hiviewdfx.HiLog;  
import ohos.hiviewdfx.HiLogLabel;  
  
public class MainAbilitySlice extends AbilitySlice {  
  
 private Text tvRequest,tvResult;  
  
 private static final String *TAG* = "MainAbilitySlice";  
  
 private static final HiLogLabel *label*=new HiLogLabel(HiLog.*DEBUG*,0x00100,"async-http");  
 @Override  
 public void onStart(Intent intent) {  
 super.onStart(intent);  
 super.setUIContent(ResourceTable.*Layout\_ability\_main*);  
 initView();  
 initListener();  
 }  
  
 private void initView() {  
 tvResult = (Text) findComponentById(ResourceTable.*Id\_tvResult*);  
 tvRequest = (Text) findComponentById(ResourceTable.*Id\_tvRequest*);  
 }  
  
 private void initListener() {  
 tvRequest.setClickedListener(new Component.ClickedListener() {  
 @Override  
 public void onClick(Component component) {  
 String url="https://apis.juhe.cn/simpleWeather/query";  
 String key="32becf485f7f174d4385957b62f28f61";

//这里获取AsyncHttpClient实例， 这个类提供了get post delete put 请求对外的接口方法  
 AsyncHttpClient client=new AsyncHttpClient();

//这里是我们包装参数的实体类  
 RequestParams params=new RequestParams();  
 params.put("city","西安");  
 params.put("key",key);

/这里是实现get请求的方，JsonHttpResponseHandler会重写请求成功的onSuccess和onFailure两个方法，两个方法内部做具体业务逻辑  
 client.get(url,params, new JsonHttpResponseHandler() {  
 @Override  
 public void onSuccess(int statusCode, Header[] headers, String responseString) {  
 super.onSuccess(statusCode, headers, responseString);  
 HiLog.*error*(*label*,"zel-onSuccess:"+responseString,responseString);  
 // 通知主线程更新UI  
 getUITaskDispatcher().asyncDispatch(new Runnable() {  
 @Override  
 public void run() {

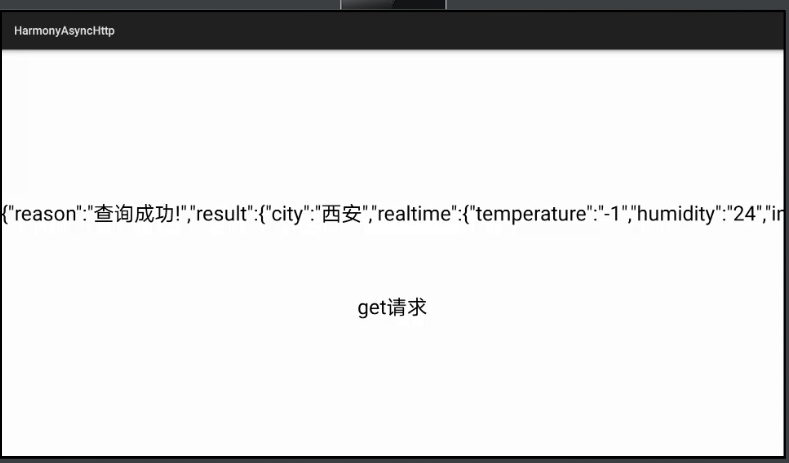
// 这里具体业务Text文本显示请求数据  
 tvResult.setText(responseString);  
 }  
 });  
 }  
  
 @Override  
 public void onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {  
 super.onFailure(statusCode, headers, responseString, throwable);  
 HiLog.*error*(*label*,"zel-onFailure:"+responseString,responseString);  
 }  
 });  
  
 }  
 });  
 }  
  
 @Override  
 public void onActive() {  
 super.onActive();  
 }  
  
 @Override  
 public void onForeground(Intent intent) {  
 super.onForeground(intent);  
 }  
}

1. **AsyncHttpHarmonyDemo实现效果**

请求前：



点击get请求之后



1. **AsyncHttpHarmony核心方法说明**

AsyncHttpClient ：通过实例对象的get方法发起get请求实现内部两个方法（请求成功失败方法）

1：onSuccess(int statusCode, Header[] headers, String responseString) {}

其中第三个参数就是返回的数据

2：onFailure(int statusCode, Header[] headers, String responseString, Throwable throwable) {}

responseString返回请求失败的json数据

AbilityContext：由AbilitySlice继承过来的getUITaskDispatcher()返回一个TaskDispatcher实例来分发任务，TaskDispatcher是一个任务分发器，它是Ability分发任务的基本接口，隐藏任务所在线程的实现细节，主要特点是在UI线程上运行的任务默认以高优先级运行

通过syncDispatch（）派发同步任务达到更新UI的效果

最后在手机模拟器上运行，结果如下：