项目 3

电 子 词 典

本项目通过鸿蒙系统开发工具 DevEco Studio，基于 Java 开发一款 SQLite 的跨设备电子词典，实现对用户输入的单词在线翻译。

3.1 总体设计

本部分包括系统架构和系统流程。

3.1.1 系统架构

系统架构如图 3-1 所示。

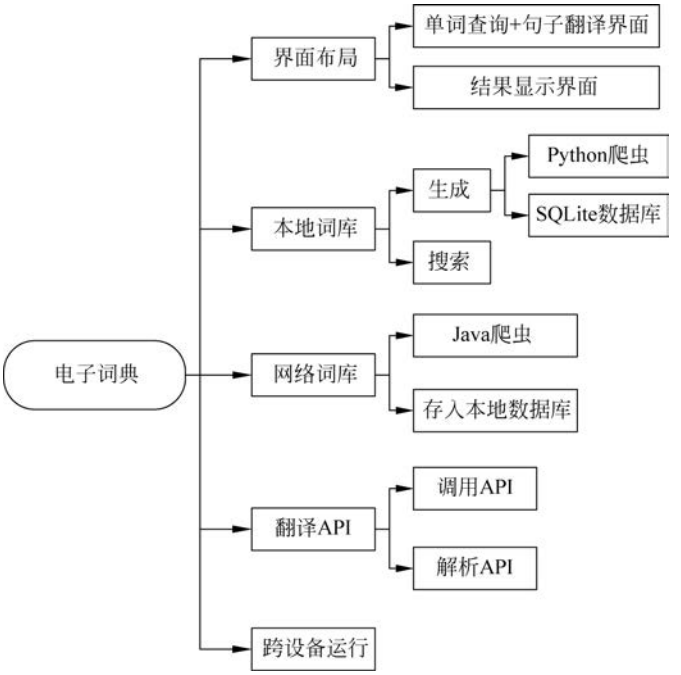

图 3-1 系统架构

3.1.2　系统流程

系统流程如图 3-2 所示。

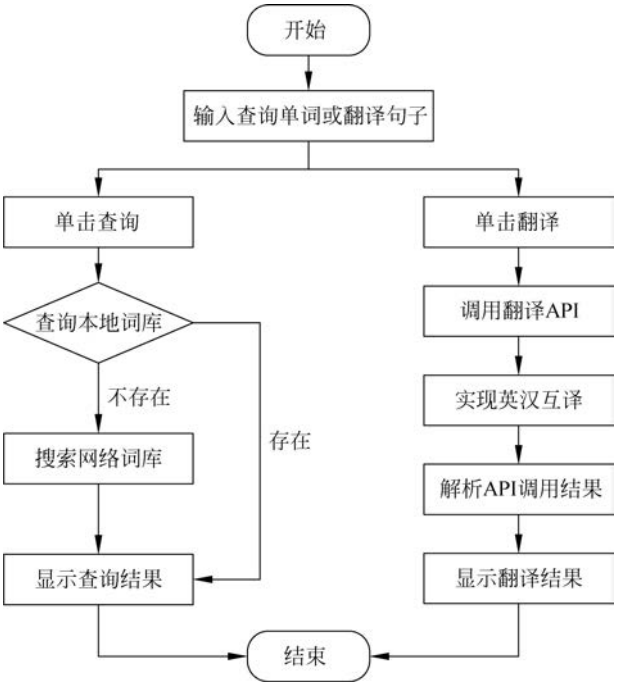

图 3-2　系统流程

为提高搜索效率，使用 Python 爬取（requests 库）部分高频词汇保存在本地 SQLite 数据库中，生成本地词库。

用户查询单词时，先查询本地词库，若存在，直接显示查询结果；若不存在，则使用 Java 爬虫（Jsoup 库），进行网络词库的搜索与显示。

用户翻译句子时，调用有道翻译 API，自动检测语言实现英汉互译，使用 Google 开发的 Gson 库解析 API 调用结果并显示翻译。

3.2　开发工具

本项目使用 DevEco Studio 开发工具，安装过程如下。

（1）注册开发者账号，完成注册并登录。在官网下载 DevEco Studio 并安装。

（2）模板类型选择 Empty Ability，设备类型选择 Tablet，语言类型选择 Java，单击 Next 后填写相关信息。

（3）创建后的应用目录结构如图 3-3 所示。

（4）在 src/main/java 目录下进行基于 SQLite 跨设备电子词典的应用开发。

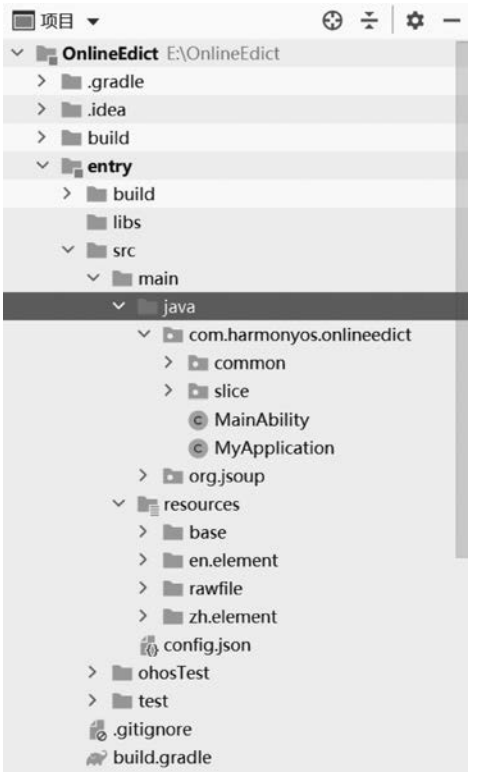

图 3-3　应用目录结构

3.3　开发实现

本部分包括界面设计和程序开发，下面分别给出各模块的功能介绍及相关代码。

3.3.1　界面设计

本部分包括图片导入、界面布局和完整代码。

1. 图片导入

将选好的图片导入 project 中，图片文件(. png 格式)保存在 src/main/resources/base/media 文件夹下，如图 3-4 所示。

2. 界面布局

本项目基于 SQLite 跨设备电子词典的界面布局如下。

1) 主界面(ability_main_tablet. xml)

主界面由名言警句、翻译图标、输入文本框、查询图标、书籍图片和翻译结果文本显示(隐藏)组件构成。

(1) 名言警句：由水平布局的 Image 元素(study. png)和 Text 元素组成。

（2）翻译图标：由 Image 元素(translation. png)组成。

（3）输入文本框：由 TextField 元素组成，其中 TextField 元素以 border. xml 为边框。

（4）查询图标：由 Image 元素(search. png)组成。

（5）书籍图片：由 Image 元素(image. png)组成。

（6）翻译结果文本显示：由隐藏的 Text 元素组成，其中 Text 元素以 trans_border. xml 为边框。

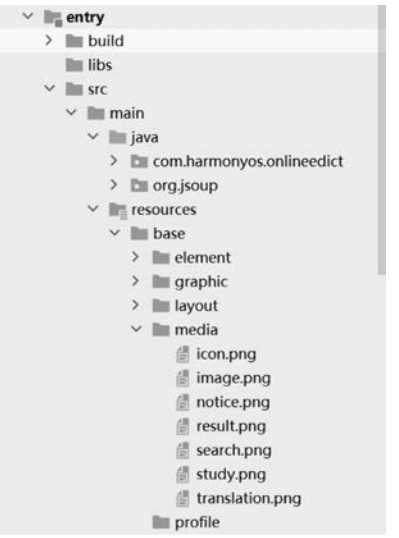

图 3-4　图片导入

2）本地词库结果显示界面(tablet_search_result. xml)

本地词库结果显示界面由名言警句、词库提示和查询结果文本显示组件构成。

（1）名言警句：由水平布局的 Image 元素(result. png)和 Text 元素组成。

（2）词库提示：由水平布局的 Image 元素(notice. png)和 Text 元素组成。

（3）查询结果文本显示：由 Text 元素组成。

3）网络词库结果显示界面(tablet_search_webresult. xml)

网络词库结果显示界面由名言警句、词库提示和查询结果文本显示组件构成。

（1）名言警句：由水平布局的 Image 元素(result. png)和 Text 元素组成。

（2）词库提示：由水平布局的 Image 元素(notice. png)和 Text 元素组成。

（3）查询结果文本显示：由 Text 元素组成。

4）手表输入查询单词的主界面(ability_main_wearable. xml)

手表输入查询单词的主界面由名言警句、输入文本框和查询图标组件构成。

（1）名言警句：由竖直布局的 Image 元素(study. png)和 Text 元素组成。

（2）输入文本框：由 TextField 元素组成，其中 TextField 元素以 border. xml 为边框。

（3）查询图标：由 Image 元素(search. png)组成。

5）手表查询单词的结果显示界面(wearable_search_result. xml)

手表查询单词的结果显示界面由名言警句和查询结果文本显示组件构成。

（1）名言警句：由竖直布局的 Image 元素(result. png)和 Text 元素组成。

（2）查询结果文本显示：由 Text 元素组成。

3. 完整代码

界面设计完整代码请扫描二维码文件 3 获取。

文件 3

3.3.2　程序开发

本部分包括生成本地词库、提取 HAP 私有路径下的本地词库、响应查询按钮和翻译按钮单击事件、搜索本地词库、解析网络词库数据、异步搜索网络词库、跳转查询单词结果显示界面、访问翻译 API、异步调用翻译 API、解析 API 调用结果、跨设备运行和完整代码。

(1) 生成本地词库(word.py)。使用Python从中国教育在线官网爬取英语四级考试词汇手册,作为查询单词操作中的高频词汇,存储在dict.sqlite数据库中(存储形式为单词、词性、词义),如图3-5所示。并将此数据库导入project的entry/src/main/resources/rawfile目录作为初始化本地词库,如图3-6所示。

数据库结构 编辑杂注 执行 SQL 浏览数据

表(T): words 在所有列中过滤

	id	word	type	meanings
	过滤	过滤	过滤	过滤
1	1	a	art.	一(个);每一(个)
2	2	abandon	vt.	丢弃;放弃,抛弃
3	3	ability	n.	能力;能耐,本领
4	4	able	a.	有能力的;出色的
5	5	abnormal	a.	不正常的;变态的
6	6	aboard	ad.	在船(车)上;上船
7	7	about	prep.	关于;在…周围
8	8	above	prep.	在…上面;高于
9	9	abroad	ad.	(在)国外;到处
10	10	absence	n.	缺席,不在场;缺乏
11	11	absent	a.	不在场的;缺乏的
12	12	absolute	a.	绝对的;纯粹的
13	13	absolutely	ad.	完全地;绝对地
14	14	absorb	vt.	吸收;使专心
15	15	abstract	a.	抽象的
16	16	abstract	n.	摘要
17	17	abundant	a.	丰富的;大量的
18	18	abuse	vt.	滥用;虐待
19	19	abuse	n.	滥用
20	20	academic	a.	学院的;学术的
21	21	academy	n.	私立中学;专科院校
22	22	accelerate	vt.	(使)加快,促进

1 - 22 / 2616 转到: 1

图3-5 SQLite数据库

(2) 提取HAP私有路径下的本地词库(MyDict.java+MainAbilitySlice.java)。在project中entry/src/main/java/com目录下新建common包,处理查询单词相关操作。在common包内新建MyDict类,读取本地词库SQLite文件。

关键代码包括MyDict构造方法(初始化数据库路径DBPath和字典路径dictPath)及读取数据方法(读取dict.sqlite文件的字节流并以4KB大小依次输出),相关代码如下。

```
public MyDict(AbilityContext context) {
    this.context = context;
    dictPath = new File(context.getDataDir().
toString() + "/MainAbility/databases/db");
    if(!dictPath.exists()){
        dictPath.mkdirs();
    }
```

图3-6 数据库导入

```
        dbPath = new File(Paths.get(dictPath.toString(),"dict.sqlite").toString());
    }
    private void extractDB() throws IOException {
        //读取 dict.sqlite 文件的字节流
        Resource resource = context.getResourceManager().getRawFileEntry("resources/rawfile/
    dict.sqlite").openRawFile();
        if(dbPath.exists()) {
            dbPath.delete();
        } //每次需复制一遍
          //输出
        FileOutputStream fos = new FileOutputStream(dbPath);
        byte[] buffer = new byte[4096];                    //每次读取 4KB
        int count = 0;
        while((count = resource.read(buffer)) >= 0) {
            fos.write(buffer,0,count);
        }
        resource.close();
        fos.close();
    }
```

在 MainAbilitySlice.java 中声明 MyDict 类型的成员变量，并进行初始化。

```
myDict = new MyDict(this);
try {
    myDict.init();
}catch (IOException e) {
    terminateAbility();
}
```

(3) 响应查询按钮和翻译按钮单击事件(MainAbilitySlice.java)。在 MainAbilitySlice.java 中将查询按钮和翻译按钮均设置为可单击，并定义相关接听器。

```
imageSearch.setClickable(true);
imageSearch.setClickedListener(new Component.ClickedListener() {
...});
transresult.setClickable(true);
transresult.setClickedListener(new Component.ClickedListener() {
...});
```

(4) 搜索本地词库(WordData.java＋MyDict.java＋MainAbilitySlice.java)。在 common 包中新建 WordData 类，并定义词性和词义两个 String 类型的变量。

```
public class WordData {
    public String type;                                //词性
    public String meanings;                            //词义
}
```

通过 MyDict.java 使用鸿蒙关系数据库 RDBStore，配置相关路径并初始化回调方法。在初始化方法中读取数据，打开本地词库(SQLite 数据库)。

```
private RdbStore store;                                //数据库引擎
private StoreConfig config = StoreConfig.newDefaultConfig("dict.sqlite");
```

```
//数据库路径
private static final RdbOpenCallback callback = new RdbOpenCallback() {
    @Override
    public void onCreate(RdbStore rdbStore) {
    }
    @Override
    public void onUpgrade(RdbStore rdbStore, int i, int i1) {
    }
};                                                          //回调
public void init() throws IOException {
    extractDB();
    //打开数据库
    DatabaseHelper helper = new DatabaseHelper(context);
    store = helper.getRdbStore(config,1,callback,null);
}
```

首先,在 MyDict.java 中定义搜索本地词库的方法,返回 WordData 类型的数组列表。其次,进行大小写转换,确保搜索关键词为小写单词。最后,执行用于查询操作的 SQL 语句,查询单词对应的词性和词义,保存在 WordData 数组列表中进行返回。

```
public ArrayList<WordData> searchLocalDict(String word) {
    word = word.toLowerCase();                              //转换为小写字母
    String[] args = new String[]{word};                     //当前要查询的单词
    ResultSet resultSet = store.querySql("select * from words where word = ?",args);
                                                            //返回值
    ArrayList<WordData> result = new ArrayList<>();
    while (resultSet.goToNextRow()) {                       //逐条读入
        WordData wordData = new WordData();
        wordData.type = resultSet.getString(2);             //获取 type 的值
        wordData.meanings = resultSet.getString(3);         //获取中文解释
        result.add(wordData);
    }
    resultSet.close();                                      //关闭数据库
    return result;
}
```

在 MainAbilitySlice.java 中查询按钮的接听器内,调用 searchLocalDict 方法并输入框内 TextField 对象的文本内容,根据返回结果判断直接显示操作或继续搜索网络词库。

```
imageSearch.setClickedListener(new Component.ClickedListener() {
    @Override
    public void onClick(Component component) {
        ArrayList<WordData> result = myDict.searchLocalDict(textfieldWord.getText());
        if(result.size() > 0) {                             //查询到结果
                showSearchResult(result, 1);
        } else {
        ...}
});
```

(5) 解析网络词库数据(SearchWordCallback.java+MyDict.java)。鸿蒙开发工具要

求访问网络必须在非界面线程(非主线程)中进行操作,并且异步进行。在 common 包中创建 SearchWordCallback 接口用来接收异步搜索的单词结果。

```
public interface SearchWordCallback {
    //定义列表接收搜索结果
    void onResult(List<WordData> result);
}
```

在 MyDict 类中定义异步搜索网络词库的方法,利用上述接口回调查询结果。首先,进行大小写转换,确保搜索关键词为小写单词。然后,使用 MyDict 中异步搜索的封装类开始线程。

```
public void searchWebDict(String word, SearchWordCallback callback) {
    word = word.toLowerCase();                          //转换为小写字母
    //异步搜索
    new AsyncSearchWord(word,store,callback).start();  //开启线程
}
```

首先,在继承 Thread 的 AsyncSearchWord 类中编写构造方法,初始化查询单词、存储数据库、回调三个变量。其次,重写 run 方法,解析 HTML 的 Java 开源工具——Jsoup 库。最后,将其源码直接导入 project 中,保存在 entry/src/main/java/org/jsoup 文件夹下,如图 3-7 所示。

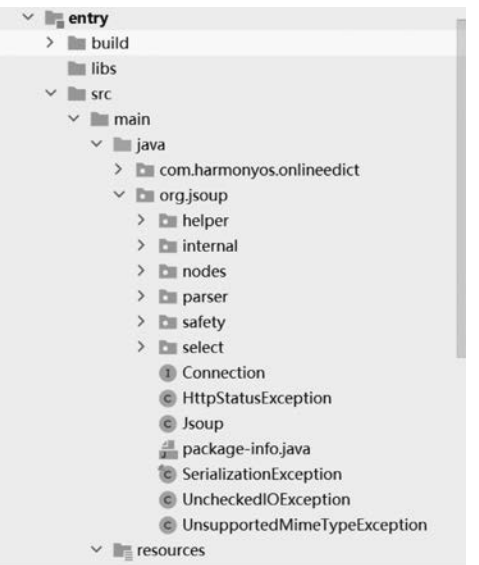

图 3-7　Jsoup 库导入

(6) 异步搜索网络词库(MyDict.java)。在 AsyncSearchWord 类的 run 中,使用 Jsoup 库的 connect 方法访问相应 URL,得到 HTML 形式的搜索结果。通过解析网页标签,获取每个词性及其对应的词义,以 WordData 类型进行存储并添加在列表中。解析完毕后将网络词库中的单词信息使用 SQL 语句保存在本地词库,最后定义有关回调。

```
@Override
public void run() {
    try {
        //获取搜索结果(HTML 形式)
        Document doc = Jsoup.connect("https://www.iciba.com/word?w=" + word).get();
                                                        //通过 HTTPS 协议获取 Web 数据
        Elements ulElements = doc.getElementsByClass("Mean_part__UI9M6");
        //将网络单词信息保存在本地的 SQL 语句
        String insertSQL = "insert into words(word, type, meanings) values(?,?,?);";
        List<WordData> wordDataList = new ArrayList<>(); //创建 List 对象
        for (Element ulElement: ulElements) {
            //获取单词的每个词性和对应词义
```

```
                Elements liElements = ulElement.getElementsByTag("li");
                //对每个词性进行迭代
                for (Element liElement:liElements) {
                    WordData wordData = new WordData();
                    //获取词性
                    Elements iElements = liElement.getElementsByTag("i");
                    for (Element iElement:iElements) {
                        //获取当前词性
                        wordData.type = iElement.text();
                        break;
                    }
                    //获取词义
                    Elements divElements = liElement.getElementsByTag("div");
                    for (Element divElement:divElements) {
                        //获取当前词义
                        wordData.meanings = divElement.text();
                        break;
                    }
                    wordDataList.add(wordData);
                    //将数据保存在本地数据库
                    store.executeSql(insertSQL,new String[]{word,wordData.type,wordData.meanings});
                }
                break;
            }
            if (callback != null) {
                callback.onResult(wordDataList);
            }                                                    //回调
        } catch (Exception e) {
        }
    }
```

在 MainAbilitySlice.java 中编写内部类 SearchWordCallbackImpl，实现上述定义的 SearchWordCallback 接口。其中需要实现 onResult 方法，进行搜索结果的展示。

```
private class SearchWordCallbackImpl implements SearchWordCallback {
    @Override
    public void onResult(List<WordData> result) {
        showSearchResult(result, 2);
    }
}
```

实现上述查询按钮的接听器时，若本地词库搜索结果为空，调用 searchWebDict 方法进行网络词库的搜索。

```
myDict.searchWebDict(textfieldWord.getText(),new SearchWordCallbackImpl());
```

此处搜索网络词库需要访问权限，在 config.json 的 module 中增加请求权限。

```
"reqPermissions": [
  {
    "name": "ohos.permission.INTERNET",
```

```
    "reason": "internet",
    "usedScene": {
      "ability": ["com.harmonyos.onlineedict.MainAbility"],
      "when": "always"
    }
  }
],
```

(7) 跳转查询单词结果显示界面(TabletSearchResultAbilitySlice.java＋TabletSearchResultWebAbilitySlice.java＋MainAbilitySlice.java)。在 Slice 包中新建 TabletSearchResultAbilitySlice 类和 TabletSearchResultWebAbilitySlice 类,分别用于本地词库和网络词库的查询单词结果显示,各自在类中重现 onStart 方法。首先,将界面分别设置为 tablet_search_result.xml 和 tablet_search_webresult.xml,并将搜索结果文本清空。其次,通过 Intent 变量传递内容,获取单词的词性和词义。最后,通过 for 循环进行输出显示。

```
public class TabletSearchResultAbilitySlice extends AbilitySlice {
    private Text textSearchResult;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_tablet_search_result);
        textSearchResult = (Text)findComponentById(ResourceTable.Id_text_search_result);
if (textSearchResult != null) {
            textSearchResult.setText("");                    //清空
            //获取词性
            ArrayList<String> typeList = intent.getStringArrayListParam("typeList");
            //获取词义
            ArrayList<String> meaningList = intent.getStringArrayListParam("meaningList");
            for (int i = 0; i < typeList.size(); i++) {
                textSearchResult.append(typeList.get(i) + " " + meaningList.get(i) + "\r\n");
            }
            if (typeList.size() == 0) {                      //未搜索到数据
                textSearchResult.setText("当前单词无查询结果,请检查输入!");
            }
        }
    }
}
public class TabletSearchResultWebAbilitySlice extends AbilitySlice {
    private Text textSearchResult;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_tablet_search_webresult);
        textSearchResult = (Text)findComponentById(ResourceTable.Id_text_search_result);
if (textSearchResult != null) {
            textSearchResult.setText("");                    //清空
            //获取词性
            ArrayList<String> typeList = intent.getStringArrayListParam("typeList");
            //获取词义
```

```
                ArrayList<String> meaningList = intent.getStringArrayListParam("meaningList");
                for (int i = 0; i < typeList.size(); i++) {
                    textSearchResult.append(typeList.get(i) + " " + meaningList.get(i) + "\r\n");
                }
                if (typeList.size() == 0) {                    //未搜索到数据
                    textSearchResult.setText("当前单词无查询结果,请检查输入!");
                }
            }
        }
    }
```

在 MainAbilitySlice.java 中编写通用的查询结果显示方法 showSearchResult。首先，定义 Intent 类型的变量用于传递当前数据到新的 Slice。其次，将词性词义分别添加到对应 String 类型的数组列表中进行传递。最后，根据参数 i 判断传递到本地词库或网络词库对应的 Slice。

```
public void showSearchResult(List<WordData> result, int i) {
    Intent intent = new Intent();                              //传递当前数据到新的 Slice
    ArrayList<String> typeList = new ArrayList<>();            //词性列表
    ArrayList<String> meaningList = new ArrayList<>();         //词义列表
    for (WordData wordData:result) {
        typeList.add(wordData.type);                           //添加词性
        meaningList.add(wordData.meanings);                    //添加词义
    }
    intent.setStringArrayListParam("typeList", typeList);
    intent.setStringArrayListParam("meaningList", meaningList);
    if (i == 1){
        present(new TabletSearchResultAbilitySlice(), intent);
    }
    else if (i == 2){
        present(new TabletSearchResultWebAbilitySlice(), intent);
    }
}
```

(8) 访问翻译 API(MainAbilitySlice.java)。本项目的翻译功能选择有道翻译平台，调用翻译 API 时同样需要请求网络权限，分别在 config.json 中增加获取网络连接信息的 ohos.permission.GET_NETWORK_INFO 权限和允许程序打开网络套接字、进行网络连接的 ohos.permission.INTERNET 权限。

```
"reqPermissions": [
  {
    "name": "ohos.permission.INTERNET",
    "reason": "internet",
    "usedScene": {
      "ability": ["com.harmonyos.onlineedict.MainAbility"],
      "when": "always"
    }
  },
  {
```

```
    "name": "ohos.permission.GET_NETWORK_INFO",
    "reason": "访问翻译 API",
    "usedScene": {
      "ability": ["com.harmonyos.onlineedict.MainAbility"],
      "when": "always"
    }
  }
],
```

在 MainAbilitySlice.java 中编写私有方法 selfTranslate，使用当前网络打开 URL 链接。首先，调用 NetManager.getInstance(Context)获取网络管理的实例对象和 NetManager.getDefaultNet()获取默认的数据网络，同时，检查是否存在缺省异常。其次，调用 NetHandle.openConnection()打开 URL 链接，使用 GET 方法，通过 URL 链接实例访问网站。访问成功后，通过 InputStream 获取 HTTP，对请求返回的内容和 BufferedReader 获取到的输入流进行读取，存储到 StringBuilder 文件中。

```
private String selfTranslate(String word) {
    NetManager netManager = NetManager.getInstance(null);
    if (!netManager.hasDefaultNet()) {
        return "";
    } //检查是否存在缺省异常
    NetHandle netHandle = netManager.getDefaultNet();
    //可以获取网络状态的变化
    //NetStatusCallback callback = new NetStatusCallback() {
    //重写需要获取网络状态变化的 override 函数
    };
    //netManager.addDefaultNetStatusCallback(callback);
    String resText = "";
    //通过 openConnection 获取 URLConnection
    HttpURLConnection connection = null;
    try { String urlString = String.format("https://fanyi.youdao.com/translate?&doctype =
json&type = AUTO&i = %s", word);
        URL url = new URL(urlString);
        URLConnection urlConnection = netHandle.openConnection(url,
                java.net.Proxy.NO_PROXY);
        if (urlConnection instanceof HttpURLConnection) {
            connection = (HttpURLConnection) urlConnection;
        }
        connection.setRequestMethod("GET");
        connection.connect();
        //可进行 URL 的其他操作
        InputStream in = connection.getInputStream();
        //对获取到的输入流进行读取
        BufferedReader reader = new BufferedReader(new InputStreamReader(in));
        StringBuilder response = new StringBuilder();
        String line;     while ((line = reader.readLine()) != null) {
            response.append(line);
        }
        HiLog.info(LOG_LABEL, response.toString());
```

```
            resText = response.toString();
        } catch(IOException e) {
            e.printStackTrace();
        } catch (Throwable throwable) {
            throwable.printStackTrace();
        } finally {      if (connection != null){
                connection.disconnect();
            }
        }
        return parseResult(resText);
    }
```

完善上述翻译按钮的接听器功能，当单击翻译按钮时隐藏书籍图片、显示搜索结果，并用 String 类型的变量 query 存储输入框内 TextField 对象的文本内容。

```
transresult.setClickedListener(new Component.ClickedListener() {
    @Override
    public void onClick(Component component) {
        image.setVisibility(Component.HIDE);                    //搜索后隐藏书籍图片
        textSearchResult.setVisibility(Component.VISIBLE); //显示搜索结果
        String query = textfieldWord.getText();
        HiLog.info(LOG_LABEL, query);
    }
});
```

(9) 异步调用翻译 API(MainAbilitySlice.java)。在上述翻译按钮的接听器实现中使用任务分发器 TaskDispatcher 接口，选择全局并发任务分发器 GlobalTaskDispatcher 进行异步派发任务 asyncDispatch。

```
TaskDispatcher taskDispatcher = getGlobalTaskDispatcher(TaskPriority.DEFAULT);
taskDispatcher.asyncDispatch(new Runnable() {
    @Override
    public void run() {
        }
});
```

使用 EventHandler 机制处理线程间通信，用户在当前线程上投递 InnerEvent 事件到异步线程上处理。每个 EventHandler 和指定的 EventRunner 事件循环器所创建的新线程绑定，并且该新线程内部有一个事件队列。EventHandler 可以投递指定的 InnerEvent 事件到它的事件队列。EventRunner 从事件队列里循环取出事件，通过所在线程执行 processEvent 回调。

创建 EventHandler 的子类——selfEventHandler 内部类，在子类中重写实现方法 processEvent 来处理事件。判断 eventId，进行对应事件的处理操作(取出 InnerEvent 事件参数，将调用翻译 API 的返回结果显示输出)。

```
class selfEventHandler extends EventHandler {
    public selfEventHandler(EventRunner runner) throws IllegalArgumentException {
        super(runner);
    }
    @Override
```

```
    protected void processEvent(InnerEvent event) {
        super.processEvent(event);
        switch (event.eventId){
            case 1:
                String result = (String)event.object;
                transResulttext.setText(result);
                break;
            default:
                break;
        }
    }
}
```

重写异步派发任务 run 的方法，将调用 selfTranslate 方法得到的返回值定义为 InnerEvent 事件的 object，并通过 EventHandler 投递 InnerEvent 事件。在翻译按钮的接听器实现完成后，通过 EventRunner 得到主线程，提交给 EventHandler 机制处理。

```
if(transresult != null){
    transresult.setClickable(true);
    transresult.setClickedListener(new Component.ClickedListener() {
        @Override
        public void onClick(Component component) {
            image.setVisibility(Component.HIDE);                    //翻译后隐藏书籍图片
            textSearchResult.setVisibility(Component.VISIBLE);  //显示翻译结果
            String query = textfieldWord.getText();
            HiLog.info(LOG_LABEL, query);
            //selfTranslate(query);
            TaskDispatcher taskDispatcher = getGlobalTaskDispatcher(TaskPriority.DEFAULT);
            taskDispatcher.asyncDispatch(new Runnable() {
                @Override
                public void run() {
                    String result = selfTranslate(query);
                    InnerEvent evt = InnerEvent.get(1);
                    evt.object = result;
                    eventHandler.sendEvent(evt);
                }
            });
        }
    });
    //present(new TabletTransAbilitySlice(), intent);
    eventRunner = EventRunner.getMainEventRunner();                 //得到主线程
    eventHandler = new selfEventHandler(eventRunner);
}
```

(10) 解析 API 调用结果(build.gradle＋MainAbilitySlice.java)。调用翻译 API 的返回结果是 Json 类型的数据，本项目使用 Google 开发的 Gson 库进行解析。需要在 entry 目录下的 build.gradle 中增加外部依赖，引入外部 Gson 库，文件位置如图 3-8 所示。

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar', '*.har'])
    testImplementation 'junit:junit:4.13'
```

```
    ohosTestImplementation 'com.huawei.ohos.testkit:runner:1.0.0.100'
    implementation 'com.google.code.gson:gson:2.8.6'
}
```

增加外部依赖后及时进行项目同步，便可找到导入的外部库，如图 3-9 所示。

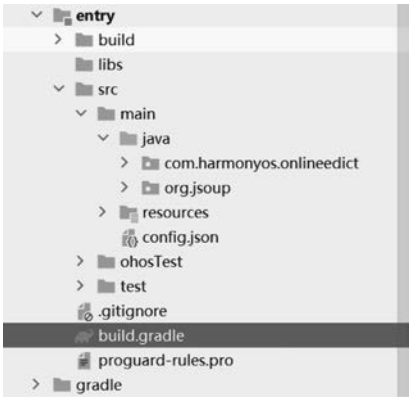

图 3-8　build.gradle 文件位置

图 3-9　外部库

在 MainAbilitySlice.java 中编写私有方法 parseResult，用来解析 API 调用结果。首先，调用 JsonParser 类的 parseString 静态方法，并得到其中 JsonObject 对象。然后，通过对 JsonObject 进行结构分析，解析出在 tgt 标签下的翻译结果。

```
private String parseResult(String transApiResponseText) {
    String result = "";
    JsonElement jsonElement = JsonParser.parseString(transApiResponseText);
    JsonObject jsonObject = jsonElement.getAsJsonObject();
    int errorCode = jsonObject.get("errorCode").getAsInt();
    //if(errorCode != 0)
        //throw new Exception("");
    JsonArray translateResult = jsonObject.get("translateResult").getAsJsonArray();
    JsonObject jsonObject1 = translateResult.get(0).getAsJsonArray().get(0).getAsJsonObject();
    result = jsonObject1.get("tgt").getAsString();
    return result;
}
```

(11) 跨设备运行(config.json＋MainAbilitySlice.java)。考虑在线电子词典 App 的实际应用场景，设计在常用电子产品平板和便携式移动设备手表上运行。同时，考虑到界面大小局限性，仅在手表上实现查询单词功能。

在 config.json 中对设备类型进行扩展，便于项目跨设备运行。

```
"deviceType": [
  "tv",
  "wearable",
  "tablet",
  "phone"
],
```

在 MainAbilitySlice.java 的主线程 onStrat 方法中判断设备类型，加载对应的界面布局。

```
@Override
public void onStart(Intent intent) {
    super.onStart(intent);
    //判断设备类型
    if (DeviceInfo.getDeviceType().equals("wearable")) {
        super.setUIContent(ResourceTable.Layout_ability_main_wearable);
    } else if (DeviceInfo.getDeviceType().equals("tablet")) {
        super.setUIContent(ResourceTable.Layout_ability_main_tablet);
    } else {
        super.setUIContent(ResourceTable.Layout_ability_main);
    }
...
}
```

在 Slice 包中新建 WearableSearchResultAbilitySlice 类，在类中重现 onStart 方法。首先，将界面设置为 wearable_search_result.xml，并将搜索结果文本清空。其次，通过 Intent 变量传递，获取单词的词性和词义。最后，通过 for 循环进行输出显示。

```
public class WearableSearchResultAbilitySlice extends AbilitySlice {
    private Text textSearchResult;
    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_wearable_search_result);
        textSearchResult = (Text)findComponentById(ResourceTable.Id_text_search_result);
if (textSearchResult != null) {
            textSearchResult.setText("");                          //清空
            //获取词性
            ArrayList<String> typeList = intent.getStringArrayListParam("typeList");
            //获取词义
            ArrayList<String> meaningList = intent.getStringArrayListParam("meaningList");
            for (int i = 0; i < typeList.size(); i++) {
                textSearchResult.append(typeList.get(i) + " " + meaningList.get(i) + "\r\n");
            }
            if (typeList.size() == 0) {                            //未搜索到数据
                textSearchResult.setText("当前单词无查询结果,请检查输入!");
            }
        }
    }
}
```

在 MainAbilitySlice.java 中修改通用的查询结果显示方法 showSearchResult，通过 Intent 类型的变量传递当前数据到不同设备对应的新的 Slice。

```
public void showSearchResult(List<WordData> result, int i) {
    Intent intent = new Intent();                                  //传递当前数据到新的 Slice
    ArrayList<String> typeList = new ArrayList<>();                //词性 List
    ArrayList<String> meaningList = new ArrayList<>();             //词义 List
    for (WordData wordData:result) {
        typeList.add(wordData.type);                               //添加词性
        meaningList.add(wordData.meanings);                        //添加词义
    }
```

```
        intent.setStringArrayListParam("typeList", typeList);
        intent.setStringArrayListParam("meaningList", meaningList);
        if (DeviceInfo.getDeviceType().equals("wearable")) {
            present(new WearableSearchResultAbilitySlice(), intent);
        }
        else {
            if (i == 1){
                present(new TabletSearchResultAbilitySlice(), intent);
            }
            else if (i == 2){
                present(new TabletSearchResultWebAbilitySlice(), intent);
            }
        }
    }
```

(12) 程序开发完整代码请扫描二维码文件 4 获取。

文件 4

3.4 成果展示

在平板端打开基于 SQLite 的跨设备电子词典 App,应用初始界面如图 3-10 所示。

在文本框中输入要查询的单词(输入一个本地词库中存在的单词 discipline),单击查询按钮,界面跳转到结果显示页,输出显示查询单词 discipline 的两个词性及对应的词义,并显示本地词库的相关提示,如图 3-11 所示。

图 3-10 应用初始界面

图 3-11 本地词库单词查询

返回主界面，继续在文本框中输入要查询的单词(输入一个本地词库中不存在的单词synthetic)，单击查询按钮。界面跳转到结果显示页，输出显示查询单词 synthetic 的两个词性及对应的词义，并显示网络词库的相关提示，如图 3-12 所示。

图 3-12　网络词库单词查询

返回主界面，继续在文本框中输入此单词 synthetic(经过查询网络词库后，此单词已经加入本地词库)，单击查询按钮，界面跳转到结果显示页，输出显示查询单词 synthetic 的两个词性及对应的词义，并显示本地词库的相关提示，说明此单词已成功存储到本地词库中，如图 3-13 所示。

在文本框中输入要翻译的句子(输入英文句子 I have completed my homework.)，单击翻译按钮，界面下方书籍图片隐藏，出现翻译结果，输出显示所输入英文句子的中文翻译，如图 3-14 所示。

继续在文本框中输入要翻译的句子(输入中文句子：我希望能在这门课程中取得好成绩)，单击翻译按钮，界面下方书籍图片隐藏，出现翻译结果，输出显示所输入中文句子的英文翻译，如图 3-15 所示。

在手表端打开基于 SQLite 的跨设备电子词典 App，应用初始界面如图 3-16 所示。

在文本框中输入要查询的单词(输入一个本地词库中存在的单词 shift)，单击查询按钮，界面跳转到结果显示页，输出显示查询单词 shift 的两个词性及对应的词义，如图 3-17 所示。

返回主界面，继续在文本框中输入要查询的单词(输入一个本地词库中不存在的单词premium)，单击查询按钮，界面跳转到结果显示页，输出显示查询单词 premium 的两个词性及对应的词义，如图 3-18 所示。

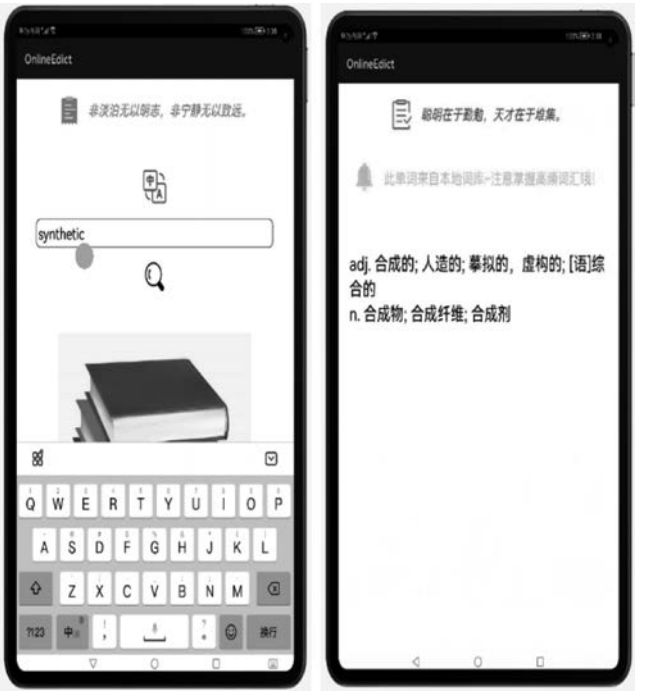

图 3-13　网络词库单词二次查询

图 3-14　英文句子翻译

图 3-15　中文句子翻译

图 3-16　应用初始界面

图 3-17　本地词库单词查询

图 3-18　网络词库单词查询