实验八：网络访问

【实验目的】

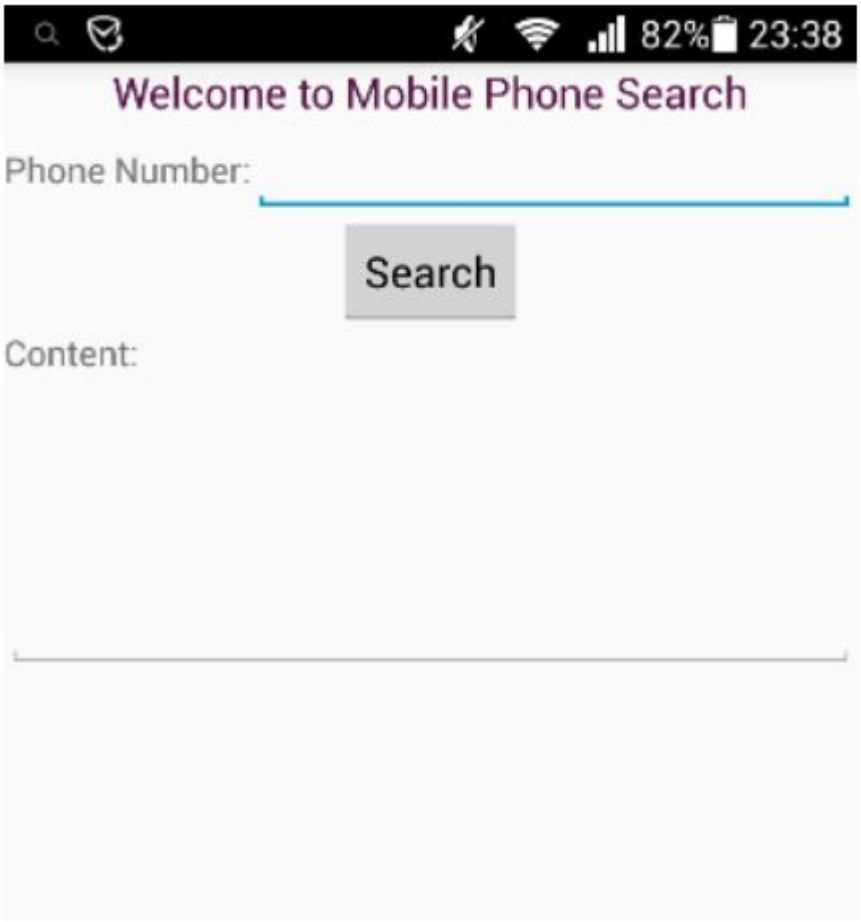
（1）熟练使用 HttpURLConnection 访问 WebService

（2）熟练使用多线程以及 Handler 更新 UI

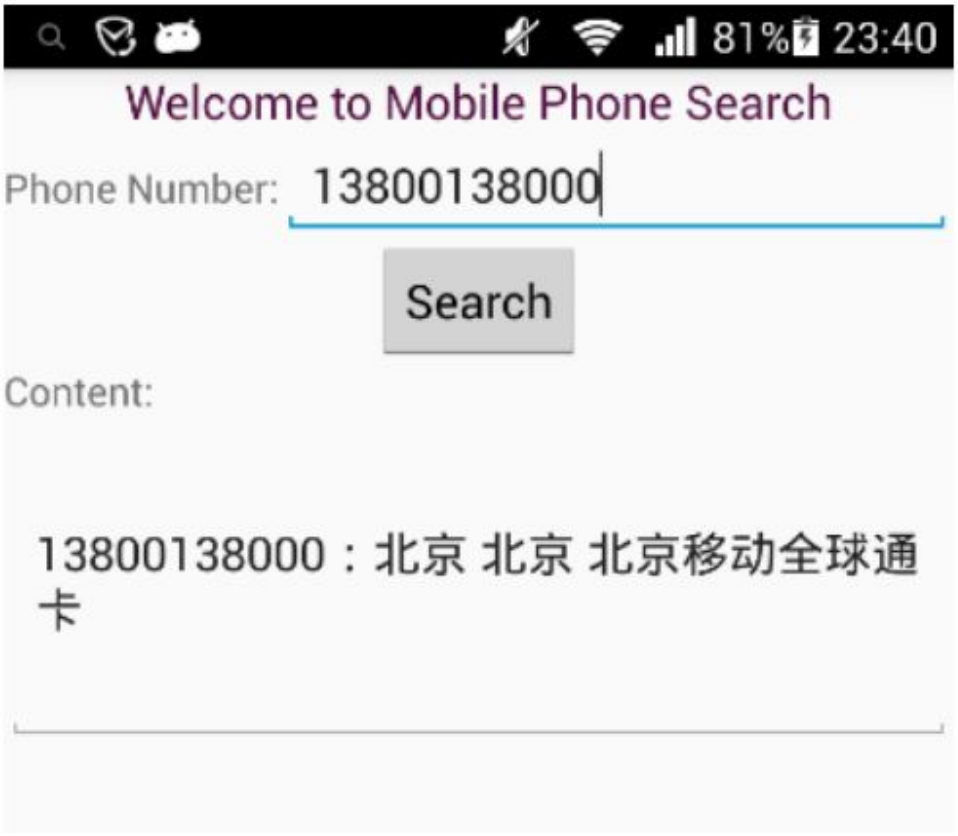
（3）熟悉使用 XmlPullParser 解析 xml 文档数据

【实验内容】

（1）程序界面如下图所示：



（2）当输入手机号码后，点击 Search 按钮后，能够查询手机归属地：



【参考内容】

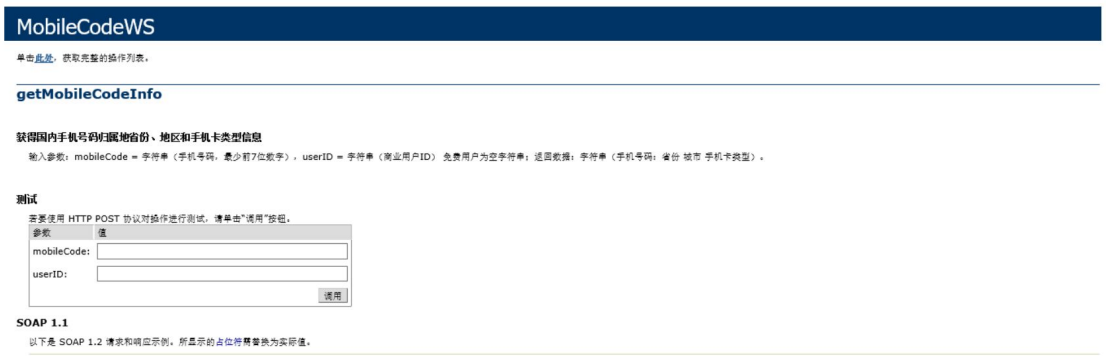
（1）实验中需要用到的 WebService 地址为：

http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx?op=getMobileCodeInfo

（2）有兴趣的同学可以在网站上看一下如何使用免费 WebService：

http://www.webxml.com.cn/zh_cn/web_services.aspx?offset=1

在浏览器打开实验需要的 WebService 网站可以看到截图如下：



可以看到，查询需要用到两个参数 mobileCode 和 userID，如果使用免费的

WebService，其中 userID 值为空，输入后点击调用，查看返回值：

```
<?xml version="1.0" encoding="UTF-8"?>
<string xmlns="http://WebXml.com.cn/">13800138000：北京 北京 北京移动全球通卡</string>
```

可以看到其中返回数据为 xml 文件格式，我们需要用 XmlPullParser 提取我们需要的信息。

（3）如何使用 Android 网络访问呢？在这里我们使用 HttpURLConnection 实现网络访问：

首先定义我们需要用到的 WebService 地址：

```
private static final String url = "http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx/"
  + "getMobileCodeInfo";
```

使用 HttpURLConnection 建立一个 http 连接：

```
HttpURLConnection connection = null;
try {
    // create a connection use url
  connection = (HttpURLConnection)((new URL(url.toString()).openConnection()));
```

其中新建一个 URL 对象，然后打开连接即可，并设置我们访问时候的时间设置，以及访问的方法为 POST：

```
  // set method
connection.setRequestMethod("POST");
connection.setConnectTimeout(8000);
connection.setReadTimeout(8000);
```

【注意】有的时候连接服务器网络不通畅，需要把相关时间值设大一些（建议设置成 40000）。

最后将我们需要请求的字段以流的形式写入到 connection 之中：

```
DataOutputStream out = new DataOutputStream(connection.getOutputStream())

  // use post method to post our data
out.writeBytes("mobileCode=" + phone_number.getText().toString() +
  "&userID=");

//Log.d("test", phone_number.getText().toString());
```

实际上这一步相当于我们将需要的参数提交到网络连接，并且请求网络数据（类似于 html 中的表单操作，将 post 数据提交到服务器）

最后将我们提取到的数据转化成为字符串:

```
    // get response data
InputStream in = connection.getInputStream();
BufferedReader reader =  new BufferedReader(new InputStreamReader(in));
StringBuilder response = new StringBuilder();

String line;

while ((line = reader.readLine()) != null) {
  response.append(line);
}
```

最后我们看一下 response 字符串形式：

```
<?xml version="1.0" encoding="utf-8"?><string xmlns="http://WebXml.com.cn/">13800138000：北京 北京 北京移动全球通卡</string>
```

这样我们就明白了，实际上就是将网站中的 xml 数据转化为字符串而已，以便于我们下一步进行 xml 数据解析。

用完了记得关掉 connection 连接：

```
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.disconnect();
    }
}
```

（4）Android4.0 以后， http 请求需要开启子线程，然后由子线程执行请求：

```
private void sendRequestWithHttpURLConnection() {
    new Thread(new Runnable() {
        @Override
        public void run() {
            HttpURLConnection connection = null;
            try {
                // create a connection use url
                connection = (HttpURLConnection)((new URL(url.toString()).openConnection()));

                // set method
                connection.setRequestMethod("POST");
                connection.setConnectTimeout(8000);
                connection.setReadTimeout(8000);

                DataOutputStream out = new DataOutputStream(connection.getOutputStream());
```

可以发现，原来我们请求的代码都是写在子线程之中的，然后使用 XmlPullParser 进行解析从而得到我们需要的数据。然后可能有些同学会认为，最后将解析后的数据直接传递到 UI 之中就行了，真的是这样吗？

如果按正常思路应该是这样的，但是会发现报错，为什么呢？原来在子线程之中，是不能够直接修改 UI 的，必须要通过一个“中间人”，这个中间人就是 Handler。

【注意】加载 Handler 所在包的时候，不要加载错了。

空间为 android.os.handler。注意与 java 包里的 handler 区分开。

```
// handler for message to modify ui
private Handler handler = new Handler() {
    public void handleMessage(Message message) {
        switch (message.what) {
            case UPDATE_CONTENT:
                //Log.d("test", message.obj.toString());
                content.setText(message.obj.toString());
                break;

            default:
                break;
        }
    }
};
```

在 Handler 中，有一个 handleMessage 函数，负责对子线程传递的消息进行分发和处理，分发就是消息类型控制的，子线程传递回来的数据，在这里由 Handler 负责修改 UI。

Message 是什么？可以认为是一种消息机制，负责在不同线程之间进行交互处理的，我们先定义消息类型：

```
private static final int UPDATE_CONTENT = 0;
```

然后将我们需要的内容通过消息传递回来：

```
    Message message = new Message();

    message.what = UPDATE_CONTENT;
    message.obj = parseXMLWithPull(response.toString());
```

（5）如何使用 XmlPullParser 实现 xml 文件的解析呢?

```
private String parseXMLWithPull(String xml) {
  String str = "";
  try {
      // use pull to parse xml
    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
    XmlPullParser parser = factory.newPullParser();

    parser.setInput(new StringReader(xml));
```

首先获取 XmlPullParser 对象实例，然后设置需要解析的字符串，最后就是逐个逐个 Tag 进行内容上的处理（Tag 可以参考 html 的相关信息）。

```
int eventType = parser.getEventType();

while (eventType != XmlPullParser.END_DOCUMENT) {
  switch (eventType) {
    case XmlPullParser.START_TAG:
      if ("string".equals(parser.getName())) {
        str = parser.nextText();
      }
      break;

    case XmlPullParser.END_TAG:
      break;

    default:
      break;
  }
  eventType = parser.next();
```

最后就能够得到我们需要的数据，在这里可以发现，实际上 Pull 解析格式是逐个逐个 Tag 进行事件触发的，遇到不同的 Tag 会触发不同的处理条件。

（6）最后给程序加上相应的权限（修改 Mainfest 文件）：

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

【扩展】

Android 中解析 xml 文档有很多种方法，XmlPullParser 是其中比较常用的一种，实际上使用 Sax 与 Dom 解析也能够达到同样的效果。

使用 Sax 和 Dom 方法其中一种方法，重新改写本实验中解析 xml 部分 （具体可以百度）。