Android移动应用开发基础教程

讲授：葛新

第7章 网络和数据解析

本章主要内容：

- 使用WebView
- 基于HTTP协议的网络访问方法
- 解析XML格式数据
- 解析JSON数据

7.1 使用WebView

- WebView控件用于在Android应用中代替浏览器来显示网页。下面通过具体的实例说明如何使用WebView显示网页。
- 首先在AndroidManifest.xml中申明网络访问权限

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.xbg.usewebview">
<uses-permission android:name="android.permission.INTERNET"/>
……
```

为主活动布局添加一个WebView控件

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
……
    <WebView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/webView" />
</RelativeLayout>
```

MainActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        WebView webView= (WebView) findViewById(R.id.webView);
        WebSettings ws=webView.getSettings();
        ws.setJavaScriptEnabled(true);                          //启用JavaScript
        webView.setWebViewClient(new WebViewClient());//使页面导航保持在WebView中
        webView.loadUrl("http://developer.android.google.cn");    //载入网页
```

7.2 基于HTTP协议的网络访问方法

本节主要内容：

1. 使用HttpURLConnection
2. 使用OkHttp

7.2.1 使用HttpURLConnection

- 使用HttpURLConnection的基本步骤
- 第1步：调用URL对象的openConnection()方法获得HttpURLConnection实例对象。例如：
  - URL url=new URL("https://developer.android.google.cn ");
  - HttpURLConnection con=(HttpURLConnection)url.openConnection();
- 第2步：设置HTTP请求方法。例如：
  - con.setRequestMethod("GET");
- 常用HTTP请求方法主要有GET和POST两种（注意大写）。GET方法一般用于仅仅希望从服务器返回数据，POST则可向服务器提交数据。

- 第3步：设置请求相关参数。例如，可设置连接和请求的超时设置（单位为毫秒）。
  - con.setConnectTimeout(6000);
  - con.setReadTimeout(6000);
- 如果是采用POST方式，则需要使用DataOutputStream来添加需要向服务器提交的数据。例如：
  - con.setRequestMethod("POST");
  - con.setDoOutput(true);
  - DataOutputStream out=new DataOutputStream(con.getOutputStream());
  - out.writeBytes("id=admin&pwd=123");
- 向服务器提交的数据采用键值对的方式表示，键值对之间用&符号分隔。

- 第4步：处理返回结果。调用HttpURLConnection对象的getInputStream()方法，获得服务器返回结果的InputStream，从中可获取服务器返回结果。例如：

```
InputStream in=con.getInputStream();
reader =new BufferedReader(new InputStreamReader(in));
StringBuilder result=new StringBuilder();
String s;
s=reader.readLine();
while(s!=null){
    result.append(s);
    s=reader.readLine();
}
```

7.2.2 使用OkHttp

- HttpURLConnection 将服务器响应结果封装在InputStream中，需要编程从中读取结果。
- OkHttp是Square公司开发的一个开源HTTP访问项目，使用起来非常简单。
- OkHttp主页地址为http://square.github.io/okhttp，从中可了解OkHttp详细信息。目前，OkHttp最新版本为3.8.0。

使用OkHttp步骤

- 第1步：需要修改项目的app/build.gradle文件，添加OkHttp库编译信息。例如：

```
dependencies {
.....
    compile 'com.android.support:appcompat-v7:25.3.1'
    testCompile 'junit:junit:4.12'
    compile 'com.squareup.okhttp3:okhttp:3.8.0'
}
```

- Gradle在构建项目时，可自动下载需要的OkHttp相关的库文件。

使用OkHttp步骤

- 第2步：创建OkHttpClient对象。例如：
  OkHttpClient okClient=new OkHttpClient();
- 第3步：创建Request.Builder来创建Request对象。例如：
  Request.Builder builder=new Request.Builder();
  builder.url("https://developer.android.google.cn ");
  Request request=builder.build();
- 默认OkHttp使用GET方法完成Http请求。如果要使用POST方法向服务器提交数据，则需要创建RequestBody对象来封装数据。例如：

```
RequestBody requestBody=new FormBody.Builder()
        .add("id","admin")
        .add("password","123")
        .build();
builder.post(requestBody);
```

使用OkHttp步骤

- 第4步：调用Request对象的execute()方法执行请求，返回结果封装在Response对象中。例如：

  Response response=okClient.newCall(request).execute();

- 第5步：获得字符串形式的返回结果。例如：

  String result=response.body().string();

7.3 解析XML格式数据

- XML已成为一种常用的数据交换格式。应用的配置、应用之间交换数据或者是网络数据传输，都会用到XML格式。http://www.w3school.com.cn/xml/index.asp提供了一个XML简略教程，读者可访问学习。
- 在使用HttpURLConnection、OKHttp等执行HTTP请求时，就可使用XML格式来封装数据。再使用Pull或DOM等常见XML解析方式，即可获得服务器返回的具体数据。

本节主要内容：

1. 准备XML数据
2. DOM解析方式
3. Pull解析方式

7.3.1 准备XML数据

- 在学习如何解析从服务器获得的XML数据之前，先做一些准备工作，准备好服务器端的XML数据。
- 本书采用Windows 10自带的IIS作为Web服务器，在服务器中创建的XML文件getxml.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<users>
    <user>
        <id>admin</id>
        <password>123</password>
    </user>
    <user>
        <id>jike</id>
        <password>456</password>
    </user>
</users>
```

7.3.2 DOM解析方式

- DOM 将XML文档看作是一个树形结构，每个标签作为一个节点。DOM解析会遍历XML文档的树形结构，以获得节点和节点文本。
- 读者可访问http://www.w3school.com.cn/xmldom/index.asp了解XML DOM详细内容。
- 下面通过一个实例说明如何在Android应用中获取并解析XML文档。（实例项目：源代码\07\ParseXml）

实例关键步骤：申明网络访问权限

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.administrator.parsexml">
    <uses-permission
            android:name="android.permission.INTERNET"/>
    ……
</manifest>
```

实例关键步骤： 修改app/build.gradle，添加OkHttp编译信息

```
dependencies {
    ......
    compile 'com.squareup.okhttp3:okhttp:3.8.0'
}
```

实例关键步骤：为主活动布局添加控件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout   ……
    <Button
        android:text="获取XML文件"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/btGetXml" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        android:id="@+id/tvXml" />
    <Button
        android:text="使用DOM解析"
```

实例关键步骤：修改MainActivity.java

```
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvXml= (TextView) findViewById(R.id.tvXml);
        Button btGetXml= (Button) findViewById(R.id.btGetXml);
        btGetXml.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//点击按钮时通过HTTP请求获取XML文档
                new Thread(new Runnable() {
                    @Override
                    public void run() {
                        doUrlGet();
                    }
                }).start();
            }
        });
```

实例关键步骤：修改MainActivity.java

```
Button btDomXml=(Button) findViewById(R.id.btDomXml);
        btDomXml.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//点击按钮时解析XML文档
                TextView tvDomResult= (TextView) findViewById(R.id.tvDomResult);
                tvDomResult.setText(domXml());
            }
        });
```

实例关键步骤：修改MainActivity.java

```
private void doUrlGet(){//使用OkHttp获取XML文档
        try {
            OkHttpClient okClient=new OkHttpClient();
            Request.Builder builder=new Request.Builder();
            builder.url("http://192.168.0.104/getxml.xml");
            Request request=builder.build();
            Response response=okClient.newCall(request).execute();
            showResult(response.body().string());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

实例关键步骤：修改MainActivity.java

```
private void showResult(final String result){
        runOnUiThread(new Runnable() {//返回主线程
            @Override
            public void run() {
                tvXml.setText(result);//在TextView中显示XML文档
            }
        });
```

实例关键步骤：修改MainActivity.java

```
private String domXml(){//使用DOM解析XML文档
    try{
        String xmlData=tvXml.getText().toString();
        DocumentBuilderFactory factory=DocumentBuilderFactory.newInstance();
        DocumentBuilder builder=factory.newDocumentBuilder();
        InputSource data= new InputSource(new ByteArrayInputStream(xmlData.getBytes("UTF-8")));
        Document document=builder.parse(data);
        Element root=document.getDocumentElement();
        NodeList nodes=root.getElementsByTagName("user");
        String result="";
```

实例关键步骤：修改MainActivity.java

```
……
            for (int i=0;i<nodes.getLength();i++){
                Element user=(Element)nodes.item(i);
                Element id=(Element)user.getElementsByTagName("id").item(0);
                Element password=(Element) user.getElementsByTagName("password").item(0);
                result+="id="+id.getTextContent();
                result+="\npassword="+password.getTextContent();
                result+="\n";
            }
            return result;
        }catch (Exception e){
            e.printStackTrace();
            return "";
        }
```

小结：使用DOM解析XML文档的步骤

1. 创建DocumentBuilderFactory对象。
2. 创建DocumentBuilder对象。
3. 将XML文档封装到InputSource对象中。
4. 使用DocumentBuilder对象解析InputSource获得表示XML文档的Document对象。
5. 调用Document对象的相关方法获取XML文档各个节点及其文本。

7.3.3 Pull解析方式

- Pull解析方式将XML文档作为输入“流”来处理，依次读取每个标签，根据标签类型来处理相应数据。
- 使用Pull解析XML文档的步骤主要包括：

  1、创建一个XmlPullParser对象作为解析器。

  例如：

  ```
  XmlPullParserFactory xmlFactory=XmlPullParserFactory.newInstance();
  XmlPullParser xmlPullParser=xmlFactory.newPullParser();
  ```

  2、将XML文档设置为解析器的输入。

  例如：

  ```
  xmlPullParser.setInput(new StringReader(xmlData));
  ```

3、获得事件类型。

- Pull根据标签的类型（开始标签、结束标签）来觉得事件类型。解析XML文档主要用到3种事件类型：END_DOCUMENT（文档结束）、STAR_TAG（开始标签）和END_TAG（结束标签）。例如：

```
int event=xmlPullParser.getEventType();          //获得当前事件类型
event=xmlPullParser.next();                      //获得下一个事件类型
```

- 调用next()方法时，输入流指针前进到下一个标签位置，知道文档结束。

4、获取当前节点数据

- 如果事件类型不是文档结束，则可调用相应方法获取当前标签数据。例如：
  - String nodeName=xmlPullParser.getName(); //获得标签名称
  - String text=xmlPullParser.nextText() //获得标签的文本内容

实例项目：源代码\07\ParseXml

```
protected void onCreate(Bundle savedInstanceState) {
……
        Button btPullXml=(Button) findViewById(R.id.btPullXml);
        btPullXml.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {//点击按钮时解析XML文档
                TextView tvPullResult= (TextView) findViewById(R.id.tvPullResult);
                tvPullResult.setText(pullXml());
            }
        });
    }
```

```
private String pullXml(){
    try{
        String xmlData=tvXml.getText().toString();
        XmlPullParserFactory xmlFactory=XmlPullParserFactory.newInstance();
        XmlPullParser xmlPullParser=xmlFactory.newPullParser();
        xmlPullParser.setInput(new StringReader(xmlData));
        int event=xmlPullParser.getEventType(); //获得当前事件类型
        String result="",nodeName="";
        while(event!=xmlPullParser.END_DOCUMENT){
            nodeName=xmlPullParser.getName();//获得标签名称
            if(event==xmlPullParser.START_TAG){
                if(nodeName.equals("id"))
                    result+="id="+xmlPullParser.nextText()+"\n";//获得标签文本进行处理
                else if(nodeName.equals("password"))
                    result+="password="+xmlPullParser.nextText()+"\n";
            }
            event=xmlPullParser.next();//获得下一个事件类型
        }
        return result;
```

7.4 解析JSON数据

- JSON主要以键值对的方式表示数据。例如：

```
{
        "jike":"极客学院",
        "users":[{"id":"admin","password":"123"},{"id":"jike","password":"456"}]
}
```

- 最外围的花括号表示这是一个JSON格式的对象数据，该对象有两个键：jike和users。jike的值时一个字符串，users的值是一个数组，数组有两个对象。
- JSON与XML相比更简洁，可以节省网络传输时间。
- 使用org.json包提供的JSONArray、JSONObject等类可轻松完成JSON数据解析。

下面的代码可用于解析前面的这个JSON字符串

```
try {
        JSONObject json=new JSONObject(data);
        String result="jike="+json.getString("jike")+"\n";    //获得指定键的值
        JSONArray users=json.getJSONArray("users");       //获得指定键的数组
        for(int i=0;i<users.length();i++){
            JSONObject item=users.getJSONObject(i);       //获得一个数组元素
            result+="user"+(i+1)+" id="+item.getString("id")+"     ";//获取键值
            result+="password="+item.getString("password")+"\n";
        }
        return result;
    } catch (Exception e) {.....  }
```