一、在Android应用中的XML文件来源

1、本地xml文件

本地XML文件可以放在应用根目录assets文件夹、res/xml、res/raw、SDcard卡、应用的data目录等；

除res/xml可直接通过getXml(int id)获取XML文档，返回一个解析器对象（XmlResourceParer：XmlResourceParer是XmlPullParser的子类），其它位置情况都可以获取XML文档，返回一个Inputstream对象，进行读取数据，获取方法分别如下：

a.在res/xml目录下（推荐使用）：

XmlResourceParser xmlParser = this.getResources().getXml(R.xml.XXX);

b.在res/xml、res/raw目录下：

InputStream inputStream = this.getResources().openRawResource(R.xml.XXX);

c.在assets文件夹下(本人测试发现通过此方法获取的XML文档不能带有首行：<?xml version="1.0" encoding="utf-8"?>，否则解析报错，具体原因未查明，知道原因请回复交流)：

InputStream inputStream = getResources().getAssets().open(fileName);

d.在应用指定目录下(SDcard，应用data目录等):

[java] view plaincopyprint?

01.// path路径根据实际项目修改，此次获取SDcard根目录

02.String path = Environment.getExternalStorageDirectory().toString();

03.File xmlFlie = new File(path+fileName);

04.InputStream inputStream = new FileInputStream(xmlFlie);

// path路径根据实际项目修改，此次获取SDcard根目录

String path = Environment.getExternalStorageDirectory().toString();

File xmlFlie = new File(path+fileName);

InputStream inputStream = new FileInputStream(xmlFlie);

2、通过url得到的xml文件

很多时候需要解析xml文件都用于客户端与服务器之间的数据交互，比如解析google天气预报信息，或自己项目内定的一些XML数据结构，其中通过URL，使用DefaultHTTPClient get请求获取XML文件方法如下：

[java] view plaincopyprint?

01./\*\*

02. \* 读取url的xml资源 转成String

03. \* @param url

04. \* @return 返回 读取url的xml字符串

05. \*/

06.public String getStringByUrl(String url) {

07. String outputString = "";

08. // DefaultHttpClient

09. DefaultHttpClient httpclient = new DefaultHttpClient();

10. // HttpGet

11. HttpGet httpget = new HttpGet(url);

12. // ResponseHandler

13. ResponseHandler<String> responseHandler = new BasicResponseHandler();

14.

15. try {

16. outputString = httpclient.execute(httpget, responseHandler);

17. outputString = new String(outputString.getBytes("ISO-8859-1"), "utf-8"); // 解决中文乱码

18.

19. Log.i("HttpClientConnector", "连接成功");

20. } catch (Exception e) {

21. Log.i("HttpClientConnector", "连接失败");

22. e.printStackTrace();

23. }

24. httpclient.getConnectionManager().shutdown();

25. return outputString;

26.}

/\*\*

\* 读取url的xml资源 转成String

\* @param url

\* @return 返回 读取url的xml字符串

\*/

public String getStringByUrl(String url) {

String outputString = "";

// DefaultHttpClient

DefaultHttpClient httpclient = new DefaultHttpClient();

// HttpGet

HttpGet httpget = new HttpGet(url);

// ResponseHandler

ResponseHandler<String> responseHandler = new BasicResponseHandler();

try {

outputString = httpclient.execute(httpget, responseHandler);

outputString = new String(outputString.getBytes("ISO-8859-1"), "utf-8"); // 解决中文乱码

Log.i("HttpClientConnector", "连接成功");

} catch (Exception e) {

Log.i("HttpClientConnector", "连接失败");

e.printStackTrace();

}

httpclient.getConnectionManager().shutdown();

return outputString;

}

二、XML文件的解析方式

能够运用在Android系统上解析XML文件的常用有三种方式：DOM、SAX和PULL，其中DOM解析XML是先把XML文件读进内存中，再通过接口获取数据，该方法使用相对小的XML文件，移动设备往往受硬件性能影响，如果XML文件比较大使用DOM解析往往效率跟不上；SAX和PULL都是采用事件驱动方式来进行解析，在Android中的事件机制是基于回调函数。

本例旨在考虑简单方便性，综合考虑选择了PULL解析，PULL解析器是一个开源项目，Android平台已经内置了PULL解析器，同时Android系统本身也是使用PULL解析器来解析各种XML文档。

1、事件回调类型

PULL解析XML文件时，回调XmlResourceParser内定义表示文档开头结束和节点开头结束的数值（事件回调类型），表示如下：

a.读取到XML文档开头（声明）返回：XmlPullParser.START\_DOCUMENT（0）

b.读取到XML文档结束返回：XmlPullParser.END\_DOCUMENT （1）

c.读取到XML节点开始返回：XmlPullParser.START\_TAG （2）

d.读取到XML节点结束返回：XmlPullParser.END\_TAG （3）

e.读取到XML文本返回：XmlPullParser.TEXT （4）

2、XmlPullParser有几个主要方法(更多查阅Android APIs):

a.XmlPullParser.getEventType() : Returns the type of the current event (START\_TAG, END\_TAG, TEXT, etc.) 【获取当前事件回调类型】

b.XmlPullParser.getName():For START\_TAG or END\_TAG events, the (local) name of the current element is returned when namespaces are enabled.【获取当前节点名字】

c.XmlPullParser.getAttributeValue(int index):Returns the given attributes value.【根据id获取节点属性值】

d.XmlPullParser.getAttributeValue(String namespace, String name):Returns the attributes value identified by namespace URI and namespace localName.【根据name获取节点属性值】

e.XmlPullParser.netxText():If current event is START\_TAG then if next element is TEXT then element content is returned or if next event is END\_TAG then empty string is returned, otherwise exception is thrown.【回调节点START\_TAG时，通过此方法获取节点内容】

3、实际编码中如何使用

在实际编码中，主要根据事件回调类型，结合被解析的XML结构进行解析提取数据，PULL解析XML文件的主要模式如下，更具体使用看本文提供的例子：

[java] view plaincopyprint?

01.try {

02. //开始解析事件

03. int eventType = parser.getEventType();

04.

05. //处理事件，不碰到文档结束就一直处理

06. while (eventType != XmlPullParser.END\_DOCUMENT) {

07. //因为定义了一堆静态常量，所以这里可以用switch

08. switch (eventType) {

09. case XmlPullParser.START\_DOCUMENT:

10. // 不做任何操作或初开始化数据

11. break;

12.

13. case XmlPullParser.START\_TAG:

14. // 解析XML节点数据

15. // 获取当前标签名字

16. String tagName = parser.getName();

17.

18. if(tagName.equals("XXXTAGXXX")){

19.

20. // 通过getAttributeValue 和 netxText解析节点的属性值和节点值

21.

22. }

23. break;

24.

25. case XmlPullParser.END\_TAG:

26. // 单节点完成，可往集合里边添加新的数据

27. break;

28. case XmlPullParser.END\_DOCUMENT:

29.

30. break;

31. }

32.

33. // 别忘了用next方法处理下一个事件，不然就会死循环

34. eventType = parser.next();

35. }

36.} catch (XmlPullParserException e) {

37. e.printStackTrace();

38.}catch (IOException e) {

39. e.printStackTrace();

40.}

三、运用例子

本例主要是解析我国省市的XML文件，文件结构简单，本例采用资源数据地址：http://www.csw333.com/CityScene\_I/getPlace.php，关于XML的结构基础之类的就不再累赘，实际项目中，根据XML文件的节点结果来进行变换，主要修改case XmlPullParser.START\_TAG部分代码。

1.本例在获取XML文件的省份中，比较了获取三个不同位置的XML文件资源的读取方式；

2.再获取url上的XML文件时，返回的字符串进行utf-8转码，避免中文乱码（本例采用的url资源数据需要转码）；

3.针对实际项目中，若xml文件过大，可以使用多线程进行解析，避免ANR，若解析时间较长，可以添加滚动提示;

4.本例存在个Bug，根据例子使用的XML文件结构把直辖市也编进省份中，例子目的是解析XML文件，实际项目是需要避免；

5.主要代码：

a.各种获取XML文件资源方式

[java] view plaincopyprint?

01./\*\*

02. \* 同样删除首行，才能解析成功，

03. \* @param fileName

04. \* @return 返回xml文件的inputStream

05. \*/

06.public InputStream getInputStreamFromAssets(String fileName){

07. try {

08. InputStream inputStream = getResources().getAssets().open(fileName);

09. return inputStream;

10. } catch (IOException e) {

11. e.printStackTrace();

12. }

13. return null;

14.}

15.

16./\*\*

17. \* 读取XML文件，xml文件放到res/xml文件夹中，若XML为本地文件，则推荐该方法

18. \*

19. \* @param fileName

20. \* @return : 读取到res/xml文件夹下的xml文件，返回XmlResourceParser对象（XmlPullParser的子类）

21. \*/

22.public XmlResourceParser getXMLFromResXml(String fileName){

23. XmlResourceParser xmlParser = null;

24. try {

25. //\*/

26. // xmlParser = this.getResources().getAssets().openXmlResourceParser("assets/"+fileName); // 失败,找不到文件

27. xmlParser = this.getResources().getXml(R.xml.provinceandcity);

28. /\*/

29. // xml文件在res目录下 也可以用此方法返回inputStream

30. InputStream inputStream = this.getResources().openRawResource(R.xml.provinceandcity);

31. /\*/

32. return xmlParser;

33. } catch (Exception e) {

34. e.printStackTrace();

35. }

36. return xmlParser;

37.}

38.

39./\*\*

40. \* 读取url的xml资源 转成String

41. \* @param url

42. \* @return 返回 读取url的xml字符串

43. \*/

44.public String getStringByUrl(String url) {

45. String outputString = "";

46. // DefaultHttpClient

47. DefaultHttpClient httpclient = new DefaultHttpClient();

48. // HttpGet

49. HttpGet httpget = new HttpGet(url);

50. // ResponseHandler

51. ResponseHandler<String> responseHandler = new BasicResponseHandler();

52.

53. try {

54. outputString = httpclient.execute(httpget, responseHandler);

55. outputString = new String(outputString.getBytes("ISO-8859-1"), "utf-8"); // 解决中文乱码

56.

57. Log.i("HttpClientConnector", "连接成功");

58. } catch (Exception e) {

59. Log.i("HttpClientConnector", "连接失败");

60. e.printStackTrace();

61. }

62. httpclient.getConnectionManager().shutdown();

63. return outputString;

64.}

65.

66./\*\*

67. \* 解析SDcard xml文件

68. \* @param fileName

69. \* @return 返回xml文件的inputStream

70. \*/

71.public InputStream getInputStreamFromSDcard(String fileName){

72. try {

73. // 路径根据实际项目修改

74. String path = Environment.getExternalStorageDirectory().toString() + "/test\_xml/";

75.

76. Log.v("", "path : " + path);

77.

78. File xmlFlie = new File(path+fileName);

79.

80. InputStream inputStream = new FileInputStream(xmlFlie);

81.

82. return inputStream;

83. } catch (IOException e) {

84. e.printStackTrace();

85. }

86. return null;

87.}

b.多线程解析：

[java] view plaincopyprint?

01./\*\*

02. \* 多线程加载网络端的xml，若xml文件过大也需要用该方式加载

03. \*/

04.Handler mHandler = new Handler();

05.Runnable mRunnable = new Runnable() {

06. public void run() {

07. if(!isFinishParser){

08.

09. mHandler.postDelayed(mRunnable, 1000);

10. }else{

11. showView.setText(provinceStr);

12. mHandler.removeCallbacks(mRunnable);

13. }

14. }

15.};

16.

17./\*\*

18. \* 比较耗时操作新建一个线程，避免UI线程ANR

19. \*/

20.public void parserWhitThread(){

21. new Thread(){

22. @Override

23. public void run() {

24. provinceandcityStr = getStringByUrl(provinceAndCityUrl);

25. provinceArray = ProvincePullParse.Parse(provinceandcityStr);

26. for(Province pro : provinceArray){

27. provinceStr += pro.getProvinceId() + " : " +pro.getProvinceName()+"\n";

28. }

29. isFinishParser = true;

30. }

31. }.start();

32.}

c.从XML文件中解析出城市：

[java] view plaincopyprint?

01.public static ArrayList<City> ParseXml(XmlPullParser parser){

02. ArrayList<City> CityArray = new ArrayList<City>();

03. City CityTemp = null;

04. int provinceId = 0;

05. int cityId;

06. String cityName;

07.

08. try {

09. //开始解析事件

10. int eventType = parser.getEventType();

11.

12. //处理事件，不碰到文档结束就一直处理

13. while (eventType != XmlPullParser.END\_DOCUMENT) {

14. //因为定义了一堆静态常量，所以这里可以用switch

15. switch (eventType) {

16. case XmlPullParser.START\_DOCUMENT:

17. break;

18.

19. case XmlPullParser.START\_TAG:

20.

21. //给当前标签起个名字

22. String tagName = parser.getName();

23. // Log.d("", "====XmlPullParser.START\_TAG=== tagName: " + tagName);

24.

25. if(tagName.equals("province")){

26.

27. provinceId = Integer.parseInt(parser.getAttributeValue(0));

28. }else if(tagName.equals("item")){

29. CityTemp = new City();

30. }else if(tagName.equals("id")){

31. cityId = Integer.parseInt(parser.nextText());

32. parser.next();

33. cityName = parser.nextText();

34.

35. Log.v("", "id getText: "+cityId);

36. Log.v("", "name getText: "+cityName);

37. Log.e("", "=========================");

38.

39. CityTemp.setProvinceId(provinceId);

40. CityTemp.setCityId(cityId);

41. CityTemp.setCityName(cityName);

42.

43. CityArray.add(CityTemp);

44. }

45. break;

46.

47. case XmlPullParser.END\_TAG:

48. break;

49. case XmlPullParser.END\_DOCUMENT:

50. break;

51. }

52.

53. //别忘了用next方法处理下一个事件，忘了的结果就成死循环#\_#

54. eventType = parser.next();

55. }

56. } catch (XmlPullParserException e) {

57. // TODO Auto-generated catch block

58. e.printStackTrace();

59. }catch (IOException e) {

60. // TODO Auto-generated catch block

61. e.printStackTrace();

62. }

63.

64. return CityArray;

65.}

books.xml

**<?xml** version="1.0" encoding="utf-8"**?>**

1. **<books>**
2. **<book>**
3. **<id>**1001**</id>**
4. **<name>**Thinking In Java**</name>**
5. **<price>**80.00**</price>**
6. **</book>**
7. **<book>**
8. **<id>**1002**</id>**
9. **<name>**Core Java**</name>**
10. **<price>**90.00**</price>**
11. **</book>**
12. **<book>**
13. **<id>**1003**</id>**
14. **<name>**Hello, Andriod**</name>**
15. **<price>**100.00**</price>**
16. **</book>**
17. **</books>**

BookParser.java

1. **package** com.scott.xml.parser;
3. **import** java.io.InputStream;
4. **import** java.util.List;
6. **import** com.scott.xml.model.Book;
8. **public** **interface** BookParser {
9. /\*\*
10. \* 解析输入流 得到Book对象集合
11. \* @param is
12. \* @return
13. \* @throws Exception
14. \*/
15. **public** List<Book> parse(InputStream is) **throws** Exception;
17. /\*\*
18. \* 序列化Book对象集合 得到XML形式的字符串
19. \* @param books
20. \* @return
21. \* @throws Exception
22. \*/
23. **public** String serialize(List<Book> books) **throws** Exception;
24. }

PullBookParser.java

1. **package** com.scott.xml.parser;
3. **import** java.io.InputStream;
4. **import** java.io.StringWriter;
5. **import** java.util.ArrayList;
6. **import** java.util.List;
8. **import** org.xmlpull.v1.XmlPullParser;
9. **import** org.xmlpull.v1.XmlSerializer;
11. **import** android.util.Xml;
13. **import** com.scott.xml.model.Book;
15. **public** **class** PullBookParser **implements** BookParser {
17. @Override
18. **public** List<Book> parse(InputStream is) **throws** Exception {
19. List<Book> books = **null**;
20. Book book = **null**;
22. //      XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
23. //      XmlPullParser parser = factory.newPullParser();
25. XmlPullParser parser = Xml.newPullParser(); //由android.util.Xml创建一个XmlPullParser实例
26. parser.setInput(is, "UTF-8");               //设置输入流 并指明编码方式
28. **int** eventType = parser.getEventType();
29. **while** (eventType != XmlPullParser.END\_DOCUMENT) {
30. **switch** (eventType) {
31. **case** XmlPullParser.START\_DOCUMENT:
32. books = **new** ArrayList<Book>();
33. **break**;
34. **case** XmlPullParser.START\_TAG:
35. **if** (parser.getName().equals("book")) {
36. book = **new** Book();
37. } **else** **if** (parser.getName().equals("id")) {
38. eventType = parser.next();
39. book.setId(Integer.parseInt(parser.getText()));
40. } **else** **if** (parser.getName().equals("name")) {
41. eventType = parser.next();
42. book.setName(parser.getText());
43. } **else** **if** (parser.getName().equals("price")) {
44. eventType = parser.next();
45. book.setPrice(Float.parseFloat(parser.getText()));
46. }
47. **break**;
48. **case** XmlPullParser.END\_TAG:
49. **if** (parser.getName().equals("book")) {
50. books.add(book);
51. book = **null**;
52. }
53. **break**;
54. }
55. eventType = parser.next();
56. }
57. **return** books;
58. }
60. @Override
61. **public** String serialize(List<Book> books) **throws** Exception {
62. //      XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
63. //      XmlSerializer serializer = factory.newSerializer();
65. XmlSerializer serializer = Xml.newSerializer(); //由android.util.Xml创建一个XmlSerializer实例
66. StringWriter writer = **new** StringWriter();
67. serializer.setOutput(writer);   //设置输出方向为writer
68. serializer.startDocument("UTF-8", **true**);
69. serializer.startTag("", "books");
70. **for** (Book book : books) {
71. serializer.startTag("", "book");
72. serializer.attribute("", "id", book.getId() + "");
74. serializer.startTag("", "name");
75. serializer.text(book.getName());
76. serializer.endTag("", "name");
78. serializer.startTag("", "price");
79. serializer.text(book.getPrice() + "");
80. serializer.endTag("", "price");
82. serializer.endTag("", "book");
83. }
84. serializer.endTag("", "books");
85. serializer.endDocument();
87. **return** writer.toString();
88. }
89. }

然后再对MainActivity做以下更改：

1. readBtn.setOnClickListener(**new** View.OnClickListener() {
2. @Override
3. **public** **void** onClick(View v) {
4. **try** {
5. InputStream is = getAssets().open("books.xml");
6. //                  parser = new SaxBookParser();
7. //                  parser = new DomBookParser();
8. parser = **new** PullBookParser();
9. books = parser.parse(is);
10. **for** (Book book : books) {
11. Log.i(TAG, book.toString());
12. }
13. } **catch** (Exception e) {
14. Log.e(TAG, e.getMessage());
15. }
16. }
17. });

private void SerializeObject(string filename)

{

XmlSerializer **serializer** =

new XmlSerializer(typeof(OrderedItem));

// Create an instance of the class to be serialized.

OrderedItem i = new OrderedItem();

// Set the public property values.

i.ItemName = "Widget";

i.Description = "Regular Widget";

i.Quantity = 10;

i.UnitPrice = (decimal) 2.30;

// Writing the document requires a TextWriter.

TextWriter writer = new StreamWriter(filename);

// Serialize the object, and close the TextWriter.

**serializer**.Serialize(writer, i);

writer.Close();

}

// This is the class that will be serialized.

public class OrderedItem

{

public string ItemName;

public string Description;

public decimal UnitPrice;

public int Quantity;

}