Mobile Computing

XML Data Manipulation using Android

Introduction

- XML stands for EXtensible Markup Language.
- XML was designed to describe data.
- XML is a software- and hardware-independent tool for carrying information.
- XML is a popular format for sharing data on the internet.
- Websites that frequently update their content, such as news sites or blogs, often provide an XML feed so that external programs can keep abreast of content changes.
- Uploading and parsing XML data is a common task for network-connected apps.

XML Parser

- Any appliance which needs to interpret XML should have a built-in XML parser.
- An XML parser converts an XML document into an XML DOM object - which can then be manipulated with the development language.
- Android SDK has several built-in XML parsers.

XML Parsers in Android

- **XmlPullParser**, which is an efficient and maintainable way to parse XML on Android. Historically Android has had two implementations of this interface:
- **KXmlParser** via XmlPullParserFactory.newPullParser().
- **ExpatPullParser**, via Xml.newPullParser().

Sample XML Document

```
- <employees>
  - <person id="1392">
      <name>John Smith</name>
      <dob>1974-07-25</dob>
      <start-date>2004-08-01</start-date>
      <salary currency="USD">35000</salary>
    </person>
  - <person id="1395">
      <name>Clara Tennison</name>
      <dob>1968-03-15</dob>
      <start-date>2003-05-16</start-date>
      <salary currency="USD">27000</salary>
    </person>
  </employees>
```

Instantiate the Parser

- The following snippet, a parser is initialized to not process namespaces, and to use the provided InputStream as its input. It starts the parsing process with a call to nextTag()

```
XmlPullParser parser = Xml.newPullParser();

parser.setFeature(
XmlPullParser.FEATURE_PROCESS_NAMESPACES,
false);

parser.setInput(in, null);

parser.nextTag();
```

Read the XML Data

- The following function looks for elements tagged “employee“ (i.e. < employee >) as a starting point for recursively processing the document.
- If a tag isn't an entry tag, it skips it.
- Once the whole document has been recursively processed, the function returns a List containing the entries (including nested data members) it extracted from the document.
- This List is then returned by the parser.

```
private List readTags(XmlPullParser parser) throws
XmlPullParserException, IOException {
    List entries = new ArrayList();

    parser.require(XmlPullParser.START_TAG, ns,
“department");
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() !=
XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        // Starts by looking for the entry tag
        if (name.equals(“employee")) {
            entries.add(readEmployee(parser));
        } else {
            skip(parser);
        }
    }
    return entries;
}
```

```
private Entry readEmployee(XmlPullParser parser) throws
XmlPullParserException, IOException {
    parser.require(XmlPullParser.START_TAG, ns, “employee");
    String id= null;

    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        if (name.equals(“id")) {
             parser.require(XmlPullParser.START_TAG, ns, “id");
            id= parser.getText();
             parser.nextTag();
            parser.require(XmlPullParser.END_TAG, ns, “id");
        } else {
            skip(parser);
        }
    }
    return new Employee(id);
}
```

```
private void skip(XmlPullParser parser) throws
XmlPullParserException, IOException {
    if (parser.getEventType() != XmlPullParser.START_TAG){
        throw new IllegalStateException();
    }
    int depth = 1;
    while (depth != 0) {
        switch (parser.next()) {
        case XmlPullParser.END_TAG:
            depth--;
            break;
        case XmlPullParser.START_TAG:
            depth++;
            break;
        }
    }
 }
```

Exercise

- Create an application that fetches weather data from **openweathermap.org** in XML data and display it.

Reading XML from Res Folder

```
InputStream ins = getResources().openRawResource( getResources().
                getIdentifier("FILENAME_WITHOUT_EXTENSION", "raw", getPackageName()));
```

```
InputStream ins = getResources ( getResources().
                getIdentifier("FILENAME_WITHOUT_EXTENSION", "raw", getPackageName()));
```