FPT Software

ANDROID TRAINING LESSON 10

Version 0.1

Agenda

- **What is REST**?
- **URLConnection**
- **Standard Java SAX parsing**
- **JSON libraries**

What is REST?

- REST stands for **Re**presentational **S**tate **T**ransfer. (It is sometimes spelled "ReST".) It relies on a stateless, client-server, cacheable communications protocol -- and in virtually all cases, the HTTP protocol is used.
- REST is *an architecture style* for designing networked applications, rather than using complex mechanisms such as CORBA, RPC or SOAP to connect between machines.
- REST uses HTTP for all four CRUD (Create/Read/Update/Delete) operations.
- REST is not a "standard". There will never be a W3C recommendation for REST, for example

What is REST?

- **REST as Lightweight Web Services**
  - Much like Web Services, a REST service is:
    - Platform-independent (you don't care if the server is Unix, the client is a Mac, or anything else),
    - Language-independent (C# can talk to Java, etc.),
    - Standards-based (runs on top of HTTP), and
    - Can easily be used in the presence of firewalls.
  - Like Web Services, REST offers no built-in security features, encryption, session management, QoS guarantees, etc. But also as with Web Services, these can be added by building on top of HTTP:
    - For security, username/password tokens are often used.
    - For encryption, REST can be used on top of HTTPS (secure sockets).
    - ... etc.

What is REST?

- How simple is REST?
  - Using Web Services and SOAP, the request would look something like this:

```
<?xml version="1.0"?>
<soap:Envelopexmlns:soap="http://www.w3.org/2001/12/soap-
    envelope"soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:body pb="http://www.acme.com/phonebook">
<pb:GetUserDetails>
<pb:UserID>12345</pb:UserID>
</pb:GetUserDetails>
</soap:Body>
</soap:Envelope>
```

  - with REST, the query will probably look like this:
    - http://www.acme.com/phonebook/UserDetails/12345
  - REST can easily handle more complex requests, including multiple parameters. In most cases, you'll just use HTTP GET parameters in the URL.
    - For example:
    - http://www.fsoft.com/phonebook/UserDetails?firstName=John&lastName=Doe
  - REST services might use XML in their ***responses*** (as one way of organizing structured data), REST *requests* rarely use XML.

HttpURLConnection

1. Obtain a new HttpURLConnection by calling URL.openConnection() and casting the result to HttpURLConnection.
2. Prepare the request (URI).
3. Optionally upload a request body
4. Read the response. Response headers typically include metadata such as the response body's content type and length, modified dates and session cookies. The response body may be read from the stream returned by getInputStream().
5. Disconnect. Once the response body has been read, the HttpURLConnection should be closed by calling disconnect(). Disconnecting frees all resources held by a connection.

HttpURLConnection

```
//Fetching  URL
URL url=new URL("http://www.somesite.fi");
HttpURLConnection
    urlConnection=(HttpURLConnection)url.openConnection();
InputStream in=new
    BufferedInputStream(urlConnection.getInputStream());
String all="";
int character;
while((character=stream.read())!=-1)
{
    all+=(char)character;
}
urlConnection.disconnect();
```

HttpURLConnection

- It’s wise to fetch data in separate thread, because touching the UI thread is forbidden in Android
- To update UI from background task, you need Android specific classes like Handler.
- AsyncTask is a class that wraps threading.

XML parsing

- **Create a Parsing Class**
  - The class extends DefaultHandler, which provides the SAX parsing tools.

```
public class DataHandler extends DefaultHandler {
    //class declaration goes here
  }
```

XML parsing

- **Add the Standard SAX Methods**

```
//start of the XML document
public void startDocument () {
       Log.i("DataHandler", "Start of XML document");
}
//end of the XML document
public void endDocument () {
      Log.i("DataHandler", "End of XML document");
}
//opening element tag
public void startElement (String uri, String name, String qName, Attributes atts) {
      //handle the start of an element
 }
 //closing element tag
public void endElement (String uri, String name, String qName) {
      //handle the end of an element
}
//element content
public void characters (char ch[], int start, int length) {
       //process the element content
}
```

XML parsing

- **Provide the Data to the Application Context**

```
//create a parser
SAXParserFactory parseFactory =
    SAXParserFactory.newInstance();
SAXParser xmlParser = parseFactory.newSAXParser();
//get an XML reader
 XMLReader xmlIn = xmlParser.getXMLReader();
//instruct the app to use this object as the handler
 XMLHandler myXMLHandler = new XMLHandler();
 xmlIn.setContentHandler(myXMLHandler);
```

JSON Parsing

- Android already contains the required JSON libraries
  - org.json.JSONArray;
  - org.json.JSONObject;

JSON Parsing

- Create a JSON Object

```
private JSONObject jObject;
```

- Define JSON object in a String ,

```
private String jString = "{\"menu\":    {\"id\": \"file\",
    \"value\": \"File\", \"popup\": { \"menuitem\":    [
    {\"value\": \"New\", \"onclick\": \"CreateNewDoc()\"},
      {\"value\": \"Open\", \"onclick\": \"OpenDoc()\"},
      {\"value\": \"Close\", \"onclick\": \"CloseDoc()\"}]}}}";
```

- Convert jString to the jObject ,

```
jObject = new JSONObject(jString);
```

JSON Parsing

- Extract the content from jObject
  - Extract the menu object by creating a new menu object:

    ```
    JSONObject menuObject = jObject.getJSONObject("menu");
    ```

  - Extract its atrtibutes:

    ```
    String attributeId = menuObject.getString("id");
    String attributeValue = menuObject.getString("value");
    ```

  - since “popup” is not plainly a String, extract it to an object:

    ```
    JSONObject popupObject = menuObject.getJSONObject("popup");
    ```

- popup contains an array of “menuitem”, extract it to a JSONArray:

```
JSONArray menuitemArray = popupObject.getJSONArray("menuitem");
//Since it contains 3 items lets put it in a for loop.
for (int i = 0; i < 3; i++) {
    System.out.println(menuitemArray.getJSONObject(i)
    .getString("value").toString());
    System.out.println(menuitemArray.getJSONObject(i).getString(
    "onclick").toString());
}
```

Homework

- Create GUI for parsing XMLWithSAX project
- Suggest:
  - List view displays CD title
  - Click on listItem to show detail of CD
- Research library Twiter4j add to dashboard project

Thank you!