PENGEMBANGAN APLIKASI PERANGKAT BERGERAK (MOBILE)

JSON

K Candra Brata
andra.course@gmail.com

Mobille App Lab 2015-2016
REST Web Service

Interoperability has Highest Priority
Exposing the Existing Function on the network

Web services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.

Interoperability

Web services allow various applications to talk to each other and share data and services among themselves. For example, a VB or .NET application can talk to Java web services and vice versa.

Standardized Protocol

Web services use standardized industry standard protocol for the communication.

Low Cost of Communication

Web services HTTP protocol, so you can use your existing low-cost internet for implementing web services.
Web Service

A software system designed to support interoperable machine-to-machine interaction over a network.

There are 2 most popular web service formats that often used service-based application development

- SOAP (Simple Object Access Protocol (SOAP)).
- REST (REpresentational State Transfer).
SOAP (Simple Object Access Protocol)

SOAP is a protocol specification for exchanging structured information in the implementation of web services in computer networks.

It uses XML Information Set for its message format, and relies on other application layer protocols, most notably Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), for message negotiation and transmission.
SOAP (Simple Object Access Protocol)

POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 299
SOAPAction: "http://www.w3.org/2003/05/soap-envelope"

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock/Surya">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
REST (REpresentational State Transfer)

Representational State Transfer (REST) is an architectural style that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

In the REST architectural style, data and functionality are considered resources and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web.

In the REST architectural style you can use XML or JSON message to perform data communication, but usually we prefer to use JSON because it’s more lightweight than XML.
XML Vs JSON

JSON Example

```json
{"employees": [  {"firstName": "John", "lastName": "Doe"},  {"firstName": "Anna", "lastName": "Smith"},  {"firstName": "Peter", "lastName": "Jones"} ]
}
```

The following XML example also defines an employees object with 3 employee records:

XML Example

```xml
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```
REST usage

- OpenWeatherMap API example

**XML mode:**
http://api.openweathermap.org/data/2.5/weather?q=Malang\&mode=xml\&appid=2de143494c0b295cca9337e1e96b00e0

**JSON mode**
http://api.openweathermap.org/data/2.5/weather?q=Malang\&mode=json\&appid=2de143494c0b295cca9337e1e96b00e0
<current>
  <city id="1636722" name="Malang">
    <coord lon="112.63" lat="-7.98"/>
    <country>ID</country>
  </city>
  <temperature value="300.15" min="300.15" max="300.15" unit="kelvin"/>
  <humidity value="78" unit="%"/>
  <pressure value="1009" unit="hPa"/>
  <wind>
    <speed value="2.6" name="Light breeze"/>
    <gusts/>
    <direction value="250" code="WSW" name="West-southwest"/>
  </wind>
  <clouds value="20" name="few clouds"/>
  <visibility value="5000"/>
  <precipitation mode="no"/>
  <weather number="721" value="haze" icon="50n"/>
  <lastupdate value="2015-11-16T20:00:00"/>
</current>
Web Services in Android

- Android has ability to consume HTTP web service response using some methods.

1. Using HTTPClient from Apache package → Deprecated since android API Ver. 14

2. HttpURLConnection from Java Net package

3. Another external HTTP API library :
   - GSON : https://github.com/google/gson
JavaScript Object Notation

Why use JSON?
There are tons of reasons why you would want to use JSON:
- It's human readable... *(if it's properly formatted)*
- It's compact because it doesn't use a full markup structure, unlike XML.
- It's easy to parse, especially in JavaScript
- A gazillion JSON libraries are available for most programming languages.
- The data structure is easy to understand *even if you're an idiot.*
A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.
**JSON (String)**

**String** is a sequence of zero or more Unicode (UTF-8) characters, wrapped in double quotes, using backslash escapes.

A character is represented as a single character string. A string is very much like a C or Java string.

- Sequence of 0 or more Unicode characters.
- No separate character type.
- A character is represented as a string with a length of 1.
- Wrapped in "double quotes".
- Backslash escapement.
JSON (String)

String

"Any UNICODE character except " or \ or control character"

"quotation mark"

"reverse solidus"

"solidus"

"backspace"

"formfeed"

"newline"

"carriage return"

"horizontal tab"

"4 hexadecimal digits"
JSON (Number)

- Integer
- Real
- Scientific

- No octal or hex
- No NaN or Infinity
- Use null instead
JSON (Number)

number

- digit 1-9
- .
- digit
- e
- E
- +
- -
Objects are unordered containers of key/value pairs

- Objects are wrapped in `{ }`
- `,` separates key/value pairs
- `:` separates keys and values
- Keys are strings
- Values are JSON values.

- `struct`, `record`, `hashtable`, `object`
JSON (Object)
{
    "name" : "Jack B. Nimble",
    "at large" : true,
    "grade" : "A",
    "format" : {
        "type" : "rect",
        "width" : 1920,
        "height" : 1080,
        "interlace" : false,
        "framerate" : 24
    }
}
Arrays are **ordered sequences** of values

Arrays are wrapped in `[ ]`

, separates values

JSON does not talk about indexing.

An implementation can start array indexing at 0 or 1.
JSON (Array)

array
[
  value,
]

[Image of a JSON array diagram]
["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

[
[0, -1, 0],
[1, 0, 0],
[0, 0, 1]
]
JSON (Arrays vs Objects)

- Use objects when the key names are arbitrary strings.
- Use arrays when the key names are sequential integers.
- Don't get confused by the term Associative Array.
```json
{
"routes" : [
    {
        "bounds" : {
            "northeast" : {
                "lat" : 37.8079996,
                "lng" : -122.4074334
            },
            "southwest" : {
                "lat" : 37.7881005,
                "lng" : -122.4203553
            }
        },
        "copyrights" : "Map data ©2015 Google",
        ...
    }
]}
```
Generating JSON Data
Web Server as Web Service Using PHP
<?php
$arr = array(
    'a' => 1, 'b' => 2,
    'c' => 3, 'd' => 4
);
header('Content-Type: application/json');
echo json_encode($arr);
?>
```php
<?php
class Box {
    public $length = 0;
    public $color = "";
    public $alive = false;
}
$danbo = new Box();
$danbo->length = 2;
$danbo->color = "brown";
$danbo->alive = true;
header('Content-Type: application/json');
echo json_encode($danbo);
```
Android JSON Parser

Handle JSON on Android Client
JSON String to Array/Object

JSON String to JSON Object (starts with 

```
JSONObject object = new JSONObject(jsonString);
```

JSON String to JSON Array (starts with [])

```
JSONArray object = new JSONArray(jsonString);
```
Getting JSON Object Attributes

- **get(String name)**
  This method just returns the value but in the form of Object type, returns another JSONObject.

- **getBoolean(String name)**
  This method returns the boolean value specified by the key

- **getDouble(String name)**
  This method returns the double value specified by the key

- **getInt(String name)**
  This method returns the integer value specified by the key

- **getLong(String name)**
  This method returns the long value specified by the key
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:background="#F44f33"
android:orientation="vertical">

<TextView
android:id="@+id/header"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:text="REST Connection"
android:textColor="#ffffff"
android:textSize="25sp" />

<Button
android:id="@+id/getButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:layout_marginTop="50dp"
android:text="Button" />

<ListView
android:id="@+id/listView"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:drawSelectorOnTop="true"/>

</LinearLayout>
public class MainActivity extends AppCompatActivity {
    private ProgressDialog progressDialog;
    private Button b1;
    private ListView lvnama;
    private String webcontent;
    private ArrayList<String> listnama = new ArrayList<String>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        b1 = (Button) findViewById(R.id.getButton);
        lvnama = (ListView) findViewById(R.id.listView);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (checkInternetConenction() == true) {
                    Toast.makeText(getApplicationContext(), " Connected ", Toast.LENGTH_SHORT).show();
                    readStream("http://YOUR_HOST_ADDRESS/papb/testjson.php");
                } else {
                    Toast.makeText(getApplicationContext(), " Not Connected ", Toast.LENGTH_LONG).show();
                }
            }
        });
    }
}
private boolean checkInternetConnection() {
    // get Connectivity Manager object to check connection
    ConnectivityManager connec = (ConnectivityManager) getSystemService(getBaseContext().CONNECTIVITY_SERVICE);

    // Check for network connections
    if ( connec.getNetworkInfo(0).getState() == android.net.NetworkInfo.State.CONNECTED ||
         connec.getNetworkInfo(0).getState() == android.net.NetworkInfo.State.CONNECTING ||
         connec.getNetworkInfo(1).getState() == android.net.NetworkInfo.State.CONNECTING ||
         connec.getNetworkInfo(1).getState() == android.net.NetworkInfo.State.CONNECTED )
    {
        return true;
    }
    else if ( connec.getNetworkInfo(0).getState() == android.net.NetworkInfo.State.DISCONNECTED ||
              connec.getNetworkInfo(1).getState() == android.net.NetworkInfo.State.DISCONNECTED )
    {
        return false;
    }
    return false;
}
private void readStream (String urlStr) {

    progressDialog = ProgressDialog.show(this, "", "Please Wait..!! ");
    final String url = urlStr;

    new Thread() {
        public void run() {
            InputStream in = null;

            Message msg = Message.obtain();
            msg.what = 1;

            try {
                StringBuilder sb = new StringBuilder();
                in = openHttpConnection(url);
                BufferedReader reader = new BufferedReader(new InputStreamReader(in,"UTF-8"));

                String nextLine = "";
                while ((nextLine = reader.readLine()) != null) {
                    sb.append(nextLine);
                    webcontent = sb.toString();
                    in.close();
                }
            } catch (IOException e) {
                e.printStackTrace();
            }

            messageHandler2.sendMessage(msg);
        } start();
    }
}
private InputStream openHttpConnection(String urlStr) {
    InputStream in = null;
    int resCode = -1;

    try {
        URL url = new URL(urlStr);
        URLConnection urlConn = url.openConnection();

        if (!(urlConn instanceof HttpURLConnection)) {
            throw new IOException("URL is not an Http URL");
        }
        HttpURLConnection httpConn = (HttpURLConnection) urlConn;
        httpConn.setAllowUserInteraction(false);
        httpConn.setInstanceFollowRedirects(true);
        httpConn.setRequestMethod("GET");
        httpConn.connect();
        resCode = httpConn.getResponseCode();

        if (resCode == HttpURLConnection.HTTP_OK) {
            in = httpConn.getInputStream();
        }
    }
    catch (MalformedURLException e) {
        e.printStackTrace();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
    return in;
}
private Handler messageHandler2 = new Handler() {
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (!webcontent.equals("null")) {
            try {
                JSONObject jObj = new JSONObject(webcontent);
                listnama.add(jObj.getString("a"));
                listnama.add(jObj.getString("b"));
                listnama.add(jObj.getString("c"));
                listnama.add(jObj.getString("d"));

                ArrayAdapter<String> myadapter =
                        new ArrayAdapter<String>(getApplicationContext(),
                        android.R.layout.simple_list_item_1, listnama);
                lvnama.setAdapter(myadapter);
            } catch (JSONException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
        progressDialog.dismiss();
    }
};
Tugas kelompok.

Kelompok terdiri dari 4 orang (Gabungan 2 kelompok).

Buat Aplikasi Android yang inovatif dengan topik: (BEBAS).

-Native app.
-Menggunakan Web service (Client-Server).
-Menggunakan Map (Location Based)
Thanks!

QUESTIONS?

JOIN !!