Trabajo DADMC

WebServices

Autor: Jorge Cañas Estévez

Desarrollo de Aplicaciones para Dispositivos Móviles y de Consumo

Introducción

1. Descripción de Servicios Web
2. Acceso a Servicios Web desde Android
    1. Google App Engine – Servlet y Endpoints
    2. SOAP
    3. HTTP POST / GET y REST
3. Ejemplo de uso de Servicio Web
4. Bibliografía

Descripción de Servicios Web

- Hoy en día existen cientos de servicios web que usan distintos **protocolos/estándares** para intercambiar datos entre aplicaciones sin importar los lenguajes que utilicen.
  - **HTTP, SMTP, FTP**. Protocolos de capa de transporte.
  - **XML**. Formato estándar para intercambio de datos.
  - **JSON**. Intercambio de datos en formato texto ligero.
  - **WSDL** (Web Service Description Language). Interfaz pública para servicios Web. Descripción en XML de requisitos para realizar comunicación.
  - **REST** (Representational State Transfer). Usa protocolo HTTP para proporcionar API con los métodos básicos (GET, POST, PUT, DELETE, etc.) para comunicar WS y cliente.

Descripción de Servicios Web

- **SOAP.** protocolo que tiene una estructura llamada sobre, que contiene una cabecera y un cuerpo. Está descrito en XML y es el lenguaje de comunicación que usa.
- Al igual que existe gran cantidad de protocolos y estándares para comunicar aplicaciones, existe un gran variedad para crear éstas. Se pueden crear servlet, SOAP, REST, etc.
- Una vez generadas estas aplicaciones su acceso desde Android se hará usando alguno de los protocolos antes mencioados.

Acceso a Servicios Web desde Android. Google App Engine – Endpoints y Servlet

- Una forma de generar un servicio Web es a través del **SDK de Google App Engine (GAE)**. La ventaja de este SDK es que se pueden generar aplicaciones en **Java, PHP, Phyton o Go** y una vez generada se puede desplegar en los servidores de Google donde se contará con una serie de **cuotas gratuitas** lo que supone un gran ahorro de tiempo en el mantenimiento del servidor, dinero para adquirir o alquilar uno y otra serie de beneficios.

- En GAE se pueden montar Servlet (HTTP) o Endpoints (se genera una serie de librerías y clases que hay que añadir a la aplicación Android), cuyo acceso se puede asegurar usando OAuth 2.0.

- Las librerías y clases del Endpoint contienen acceso a los métodos con los que interactuar.

Acceso a Servicios Web desde Android SOAP

- Para acceder a servicios de tipo SOAP en Android se usa la librería ksoap2 de Google que facilitará el trabajo a la hora de construir la petición. Esta librería se puede descargar y adjuntar al proyecto en la carpeta libs, pero en su lugar se va a añadir el repositorio al fichero build.gradle fuera de la carpeta app quedando así:

```
buildscript {
    repositories {
        maven { url 'http://ksoap2-android.googlecode.com/svn/m2-repo' }
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.3.0'
    }
}

allprojects {
    repositories {
        maven { url 'http://ksoap2-android.googlecode.com/svn/m2-repo' }
        jcenter()
    }
}
```

Acceso a Servicios Web desde Android SOAP

- A continuación se modifica el archivo build.gradle dentro de la carpeta app y se añade una nueva dependencia correspondiente a la versión deseada del ksoap2 de android. En este caso se ha usado la versión 3.4.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'com.android.support:design:23.1.1'
    compile 'com.google.code.ksoap2-android:ksoap2-android:3.4.0'
}
```

Acceso a Servicios Web desde Android SOAP

- Una vez se tiene accesible la librería se puede comenzar a hacer uso de esta.
- Existen dos versiones, la 1.1 y la 1.2. La primera usa XML simple, pero esto llevó a muchas ambigüedades por lo que se creó la versión 1.2 que hace uso de XML InfoSet donde se describe un modelo abstracto de datos del XML.
- En este ejemplo el uso de la versión 1.1 o 1.2 no repercute en ningún aspecto, por lo que se obviará.

Acceso a Servicios Web desde Android SOAP

- El servicio que se va a usar es el convertidor de temperatura de la página webservicex.net. Para hacer el cambio pide una temperatura (double), la unidad en la que está la temperatura y a qué unidad se quiere cambiar. La unidad puede ser Celsius, Fahrenheit, Rankine, Reamur y kelvin.

```
private final String NAMESPACE = "http://www.webserviceX.NET/";
private final String METHOD_NAME = "ConvertTemp";
private final String URL = "http://www.webservicex.net/ConvertTemperature.asmx";
private final String SOAP_ACTION = "http://www.webserviceX.NET/ConvertTemp";
```

- Para usar los servicios SOAP hay que tener en cuenta que se hace una llamada asíncrona al servidor. En este caso se ha optado por implementar una tarea asíncrona, donde la información principal a definir para SOAP es la siguiente:

Acceso a Servicios Web desde Android SOAP

- Función doInBackground de la tarea asíncrona.

```
protected String doInBackground(Object... params) {
    Log.i(DEBUGTAG, "doInBackground");
    try {
        //Creación del objeto Soap
        SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
        //Definición de propiedades temperatura, Unidad origen y destino
        PropertyInfo Temperature = new PropertyInfo();
        Temperature.setName("Temperature");
        Temperature.setValue(String.valueOf(params[0]));
        Temperature.setType(Double.class);
        request.addProperty(Temperature);

        PropertyInfo FromUnit = new PropertyInfo();
        FromUnit.setName("FromUnit");
        FromUnit.setValue((String) params[1]);
        Log.i(DEBUGTAG, (String) params[1]);
        FromUnit.setType(String.class);
        request.addProperty(FromUnit);

        PropertyInfo ToUnit = new PropertyInfo();
        ToUnit.setName("ToUnit");
        ToUnit.setValue((String) params[2]);
        Log.i(DEBUGTAG, (String) params[2]);
        ToUnit.setType(String.class);
        request.addProperty(ToUnit);
```

Acceso a Servicios Web desde Android SOAP

- Función dentro del doInBackground de un AsyncTask

```
        //Creación del sobre soap con la version 1.1
        SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
        //Compatibilidad con .NET
        envelope.dotNet = true;
        envelope.setOutputSoapObject(request);
        //Creación de la clase transporte y añadido del sobre Soap
        HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
        androidHttpTransport.call(SOAP_ACTION, envelope);
        //Recoger respuesta del servidor y devolverla al onPostExecute
        SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
        String webResponse = response.toString();
        Log.i(DEBUGTAG, webResponse);
        //tvValueDestiny.setText(webResponse);
        return webResponse;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "";
}
```

Acceso a Servicios Web desde Android SOAP

- Una vez finaliza el doInBackground simplemente se muestra el resultado en un TextView, con lo que se terminaría el acceso al servicio Web y el mostrar su información.

```
@Override
protected void onPostExecute(String result) {
    Log.i(DEBUGTAG, "postExecute "+result);
    if (result != "") {
        tvValueDestiny.setText(result);
    } else {
        tvValueDestiny.setText("Error");
    }
}
```

Acceso a Servicios Web desde Android HTTP POST/GET y REST

- Otra forma de acceso a servicios web es a través del protocolo HTTP del que hacen uso muchas páginas y servicios como los denominados REST y Servlet.
- REST es más flexible que SOAP ya que el resultado se puede enviar en cualquier formato como JSON o XML, en lugar de solo el último como ocurre en SOAP.
- Los Servlet harían prácticamente la misma función que los servicios REST ya que son también bastante flexibles.

Acceso a Servicios Web desde Android HTTP POST/GET y REST

- Para usar un servicio basado en HTTP hay que tener en cuenta que a partir de la API 23 de Android, la librería para apache queda obsoleta, por lo que hay que hacer uso de HttpUrlConnection. También, al igual que con SOAP, hay que hacer uso de un método asíncrono por lo que se puede usar un hilo por ejemplo.

```
//Se coge el valor de la temperatura
if (etValue.getText().toString().equals("")){
    tempValue = 0;
}else{
    tempValue = Integer.valueOf(etValue.getText().toString());
}
//Se crea un hilo para que no se bloquee
new Thread((Runnable) () -> {
        //Función que realiza la petición POST
        final String temp = getTemp(tempValue, tempPOSTValue[tempOriginChoice],
                tempPOSTValue[tempDestinyChoice]);
        hdnl.post(() -> { tvValueDestiny.setText(temp); });
}).start();
```

Acceso a Servicios Web desde Android HTTP POST/GET y REST

- El método que llama al servicio web es el siguiente (parte 1):

```
//Dirección del servicio web
private static final String URL_POST =
        "http://www.webserviceX.NET/ConvertTemperature.asmx/ConvertTemp";
public String getTemp(int Temperature, String FromUnit, String ToUnit){
    Log.i(DEBUGTAG,"getTemp");
    URL url;
    HttpURLConnection connection = null;
    try{
        //Definición de los parámetros
        String urlParameters =
                "Temperature="+ URLEncoder.encode(String.valueOf(Temperature), "UTF-8")+
                "&FromUnit="+URLEncoder.encode(FromUnit,"UTF-8")+
                "&ToUnit="+URLEncoder.encode(ToUnit,"UTF-8");
        Log.i(DEBUGTAG, urlParameters);
        //Se crea la conexión y se define la cabecera
        url = new URL(URL_POST);
        connection = (HttpURLConnection)url.openConnection();
        connection.setRequestMethod("POST");
        connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");
        connection.setRequestProperty("Content-Length", "" +
                Integer.toString(urlParameters.getBytes().length));

        //Se define que se envía y recibe información
        connection.setDoInput(true);
        connection.setDoOutput(true);
```

Acceso a Servicios Web desde Android HTTP POST/GET y REST

- El método que llama al servicio web es el siguiente (parte 2):

```
        //Se envía la petición con los parámetros definidos
        DataOutputStream wr = new DataOutputStream (
                connection.getOutputStream ());
        wr.writeBytes (urlParameters);
        wr.flush ();
        wr.close ();

        //Se obtiene la respuesta, como devuelve un XML hay que parsearlo, para ello se crea
        //un lector DOM que lee la respuesta del servidor
        InputStream is = connection.getInputStream();
        BufferedReader rd = new BufferedReader(new InputStreamReader(is));
        DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder dbilder = dbFactory.newDocumentBuilder();
        Document doc = dbilder.parse(is);
        //Como solo hay 1 elemento que es la temperatura se lee directamente
        String response = doc.getDocumentElement().getTextContent();
        rd.close();
        Log.i(DEBUGTAG, response);
        return response;

    } catch (IOException e) {
        e.printStackTrace();
    }catch (Exception e){
        e.printStackTrace();
    }
    return null;
}
```

Ejemplo de uso de Servicio Web

- Demostración de la librería SOAP y protocolo HTTP

Bibliografía

- HttpUrlConnect
  - http://developer.android.com/intl/es/reference/java/net/HttpURLConnection.html
  - http://stackoverflow.com/questions/2793150/using-java-net-urlconnection-to-fire-and-handle-http-requests
- Ksoap2
  - http://helpdev.com.br/2015/07/10/android-studio-gradle-adicionando-ksoap2-como-dependencia-adding-ksoap-dependency-to-gradle-project/
  - http://www.itcsolutions.eu/2011/03/03/how-to-consume-web-services-from-android-applications-using-ksoap2/