Tema 8. Servicios web

Prof. Manuel Castillo

Programación de Dispositivos Móviles

Escuela Profesional de Ciencias de la Computación

Facultad de Ciencias

Universidad Nacional de Ingeniería

Objetivos

- Conocer los fundamentos de un servicio web.
- Saber cómo funciona un servicio SOAP.
- Saber cómo conectar a un servicio SOAP desde Android.
- Conocer el formato de intercambio de información JSON.
- Conocer cómo se monta un servicio REST.
- Saber cómo conectar a un servicio REST desde Android.

Índice de contenido

1. ¿Qué es un servicio web? (I)

- Es una aplicación web que permite la gestión de información de manera remota.
- Está basada en el protocolo *HTTP*.
- Permite el uso de uno o varios formatos de intercambio.
- Tiene manejo de sesión.
- Tiene distintos métodos de funcionamiento.

1. ¿Qué es un servicio web? (II)

- Dispone de dos agentes principales.
  - Cliente/consumidor: es el agente que realiza las consultas de información.
  - Servidor: es el agente que gestiona las peticiones de los clientes para gestionar la información.
- Los dos protocolos principales son:
  - SOAP: muy estricto, sólo funciona en base a *XML*, está muy estructurado, muy utilizado con tecnologías *.NET* y Java.
  - REST: muy flexible y liviano, funciona con cualquier formato no sólo *XML*, permite el uso de *JSON*, típico de las empresas de Internet.

2. Servicios basados en SOAP

- Soap es uno de los protocolos más extendidos de los servicios web, sobre todo dentro de la empresa.
- Funciona en base a una *URL* principal y una serie de métodos (comandos) de funcionamiento.
- El intercambio de información se realiza mediante ficheros *XML*, tanto en una dirección como en otra.
- Siempre se realizan peticiones *HTTP POST*.
- Es un protocolo complejo y pesado.
- Dar permiso para acceso a Internet:
  - *<uses-permission android:name="android.permission.INTERNET" />*

2. Servicios basados en SOAP

2.1. Ejemplo - Propiedades

- Utilizar las librerías *kSOAP2* (http://ksoap2.sourceforge.net/).

```
Public class main_activity extends AppCompactAcitivity {
    private final String NAMESPACE =
            "http://www.w3schools.com/webservices/";
    private final String URL =
            "http://www.w3schools.com/webservices/tempconvert.asmx";
    private final String SOAP_ACTION =
            "http://www.w3schools.com/webservices/CelsiusToFahrenheit";
    private final String METHOD_NAME = "CelsiusToFahrenheit";
    private String TAG = "PGGURU";
    private static String celcius;
    private static String fahren;
    Button b;
    TextView tv;
    EditText et;
```

Método en el servidor

2.1. Ejemplo - *ASyncTask*


```
private class AsyncCallWS extends AsyncTask<String, Void, Void> {
        @Override
        protected Void doInBackground(String... params) {
            Log.i(TAG, "doInBackground");
            getFahrenheit(celcius);
            return null;
        }
        @Override
        protected void onPostExecute(Void result) {
            Log.i(TAG, "onPostExecute");
            tv.setText(fahren + "º F");
        }
```

En segundo plano lanza el SOAP

Nos devuelve la temperatura

2.1. Trabajo con *ASyncTask*


```
    @Override
    protected void onPreExecute() {
        Log.i(TAG, "onPreExecute");
        tv.setText("Calculating...");
    }
    @Override
    protected void onProgressUpdate(Void... values) {
        Log.i(TAG, "onProgressUpdate");
    }
}
```

Coloca un texto mientras carga

2.1. Ejemplo – método getFharenheit


```
public void getFahrenheit(String celsius) {
        //Create request
        SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
        //Property which holds input parameters
        PropertyInfo celsiusPI = new PropertyInfo();
        //Set Name
        celsiusPI.setName("Celsius");
        //Set Value
        celsiusPI.setValue(celsius);
        //Set dataType
        celsiusPI.setType(double.class);
        //Add the property to request object
        request.addProperty(celsiusPI);
        //Create envelope
        SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(
                SoapEnvelope.VER11);
```

Objeto SOAP con url y método

Datos que quiero pasar con el nombre, valor y tipo de dato

Validamos la propiedad

Configuración de empaquetar para .NET

2.1. Código en segundo plano

```
envelope.dotNet = true;
//Set output SOAP object
envelope.setOutputSoapObject(request);
//Create HTTP call object
HttpTransportSE androidHttpTransport = new HttpTransportSE(URL);
try {
    //Invole web service
    androidHttpTransport.call(SOAP_ACTION, envelope);
    //Get the response
    SoapPrimitive response = (SoapPrimitive) envelope.getResponse();
    //Assign it to fahren static variable
    fahren = response.toString();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Incorpora la petición

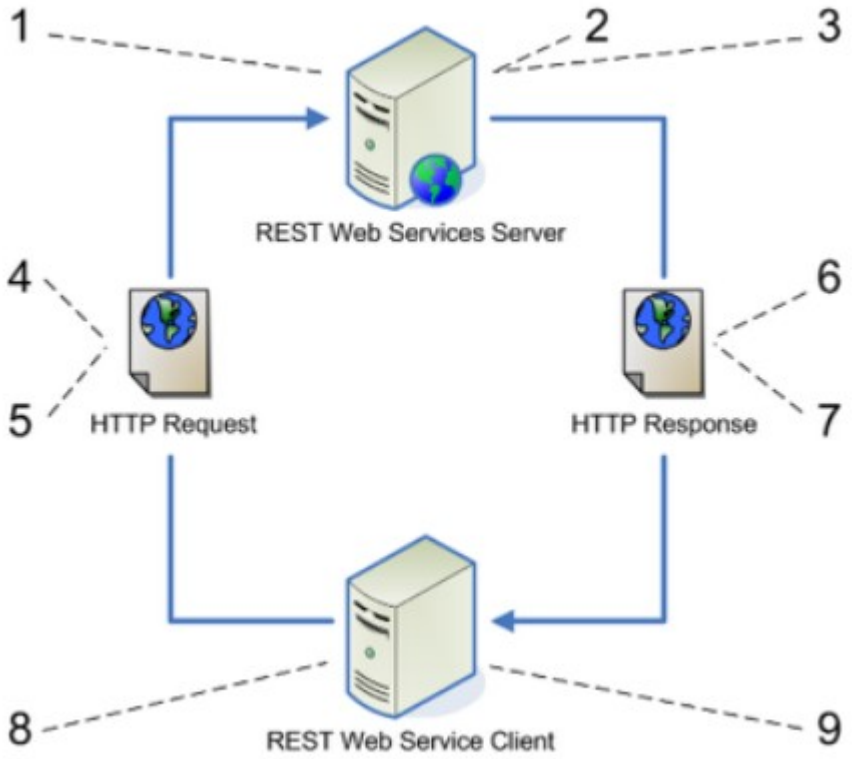
Abrimos el protocolo HTTP

Realiza la conexión al servidor y guardamos en fahren la respuesta

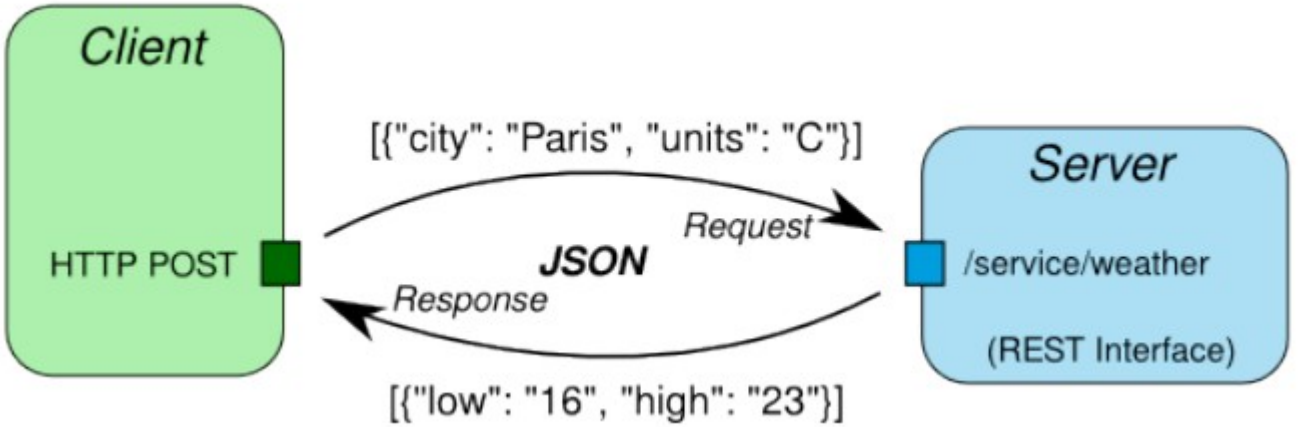
3. Servicios basados en REST

- *REST* es uno de los protocolos más populares de los servicios web.
- Funciona en base a una *URL* principal y una serie de parámetros indicados por *URL*.
- El intercambio de información se realiza en distintos formatos, *XML* y *JSON* son los más populares.
- Se utiliza directamente el protocolo *HTTP*, por medio de sus operaciones *GET*, *POST*, *PUT* y *DELETE*
- Es un protocolo simple de intercambio de información.
- Permisos:
  - *<uses-permission android:name="android.permission.INTERNET" />*
  - *<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />*

3. Servicios basados en REST



3.1. JSON/REST/HTTP



3.1. Ejemplo - onCreate


```
protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_my);
       etResponse = (EditText) findViewById(R.id.etResponse);
      tvIsConnected = (TextView) findViewById(R.id.tvIsConnected);
       if(isConnected()){
          tvIsConnected.setBackgroundColor(0xFF00CC00);
          tvIsConnected.setText("Estás conectado");
      } else{
          tvIsConnected.setText("No estás conectado");
      }
       new HttpAsyncTask().execute
                ("http://hmkcode.appspot.com/rest/controller/get.json");
}
```

Verificación de conexión

Llamada a *AsyncTask* llamando a la url

2.1. Ejemplo – método isConected

```
public boolean isConnected(){
    ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Activity.CONNECTIVITY_SERVICE);
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
    if (networkInfo != null && networkInfo.isConnected())
       return true;
    else
       return false;
 }z
```

Llamamos al ConnectivityManager y obtenemos el servicio de conectividad

Verificamos si esa información está conectada o no

2.1. Ejemplo - *ASyncTask*

```
private class HttpAsyncTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        return GET(urls[0]);
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        Toast.makeText(getBaseContext(), "¡Recibido!",
                Toast.LENGTH_LONG).show();
        JSONObject json = null;
```

Llamada vía get a la dirección

Inicializamos el objeto JSON

2.1. Código en ASyncTask

```
try {
        json = new JSONObject(result);
        etResponse.setText(json.toString(1));
        JSONArray articles = json.getJSONArray("articleList");
        StringBuffer sb=new StringBuffer();
        String str=sb.append("Resultados:")
              .append(articles.length())
              .append(":nombres:")
              .append(articles.getJSONObject(0).names())
              .append(":url:"+articles.getJSONObject(0).getString("url")).
                    toString();
        etResponse.setText(str);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

Capturamos el objeto JSON y convertimos

Propiedades de los artículos: nombre, url

Estable el String

2.1. Ejemplo – Método GET


```
public static String GET(String url){
    InputStream inputStream = null;
    String result = "";
    try {
        HttpClient httpclient = new DefaultHttpClient();
        HttpResponse httpResponse = httpclient.execute(new HttpGet(url));
        inputStream = httpResponse.getEntity().getContent();
        if(inputStream != null)
            result = convertInputStreamToString(inputStream);
        else
            result = "Sin trabajo!";
    } catch (Exception e) {
        Log.d("InputStream", e.getLocalizedMessage());
    }
    return result;
}
```

Funciones de Apache

Lanzamos un objeto GET

Llamamos a la función para convertir el inputStream a una cadena de caracteres

2.1. Ejemplo – Método convertir a String


```
private static String convertInputStreamToString
        (InputStream inputStream) throws IOException {
    BufferedReader bufferedReader = new BufferedReader(
            new InputStreamReader(inputStream));
    String line = "";
    String result = "";
    while((line = bufferedReader.readLine()) != null)
       result += line;

    inputStream.close();
    return result;
}
```

Lee de línea en línea y se va agregando

4. Servicios basados en JSON

- *JSON* es un formato de intercambio serializable de datos sencillo.
- Está basado en la definición de objetos Javascript.
- Básicamente es un objeto con más propiedades con valores que son otros objetos.
- La definición se realiza dentro de unas "*{}*".
- Los "*[]*" permiten la gestión de listados.
- El elemento básico es la propiedad
  - Se define como un conjunto de pares propiedad:valor separados por "*,*"
- Es un formato muy liviano, ideal para compartir mucha información en muy poco espacio.

4.1. Ejemplo

Diferentes propiedades con su valor, String entre comillas

```
{
  "arguments" : { "number" : 10 },
  "url" : "http://localhost:8080/restty-tester/collection",
  "method" : "POST",
  "header" : {
    "Content-Type" : "application/json"
  },
  "body" : [
    {
      "id" : 0,
      "name" : "name 0",
      "description" : "description 0"
    },
    {
      "id" : 1,
      "name" : "name 1",
      "description" : "description 1"
    }
  ],
  "output" : "json"
}
```

Body tiene dos objetos como propiedades

Body tiene dos objetos como propiedades