Acceso a Servicios Web SOAP en Android

Begoña Feito Peláez

Índice

¿Qué es SOAP? (Simple Object Access Protocol)

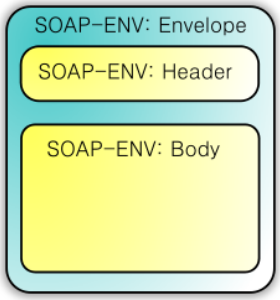
Es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio del intercambio de datos en formato **XML**. Es uno de los más utilizados en los servicios Web. Puede ser utilizado sobre cualquier protocolo de transporte como **HTTP**, SMTP, TCP o JMS. Además permite cualquier modelo de programación.

La especificación del protocolo SOAP indica que consiste en 3 partes:

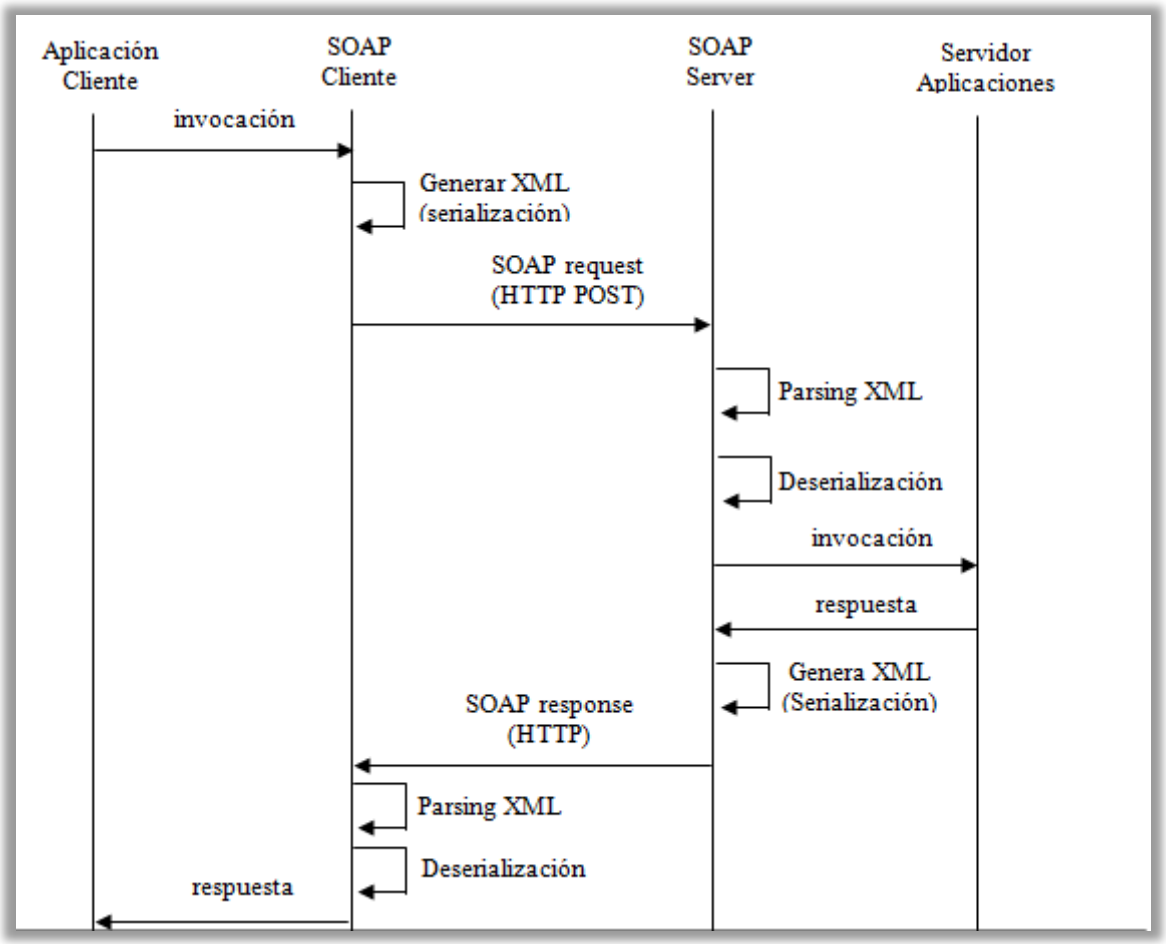
- **El constructor SOAP ENVELOPE** que define un framework para expresar qué hay en un mensaje y a quién está dirigido.
- **Las reglas de codificación** que definen un mecanismo de serialización para intercambiar instancias de tipos de datos.
- **La representación SOAP RPC** (Remote Procedure Call) que define una metodología que puede ser usada para representar invocaciones a procedimientos remotos y sus respuestas.

Se basa en cinco pasos básicamente, definir la petición (request), configurar un sobre (envelope; que define el mensaje y cómo procesarlo), definir el canal de transporte, hacer la llamada y recoger los datos.

Normalmente, un mensaje SOAP consta de tres partes diferenciadas:



- **Envelope:** Es el elemento principal, como el sobre en el que mandamos el mensaje. Consta de las otras dos partes:
- **Header:** Se utiliza para pasar información relacionada con la aplicación que se va a procesar. Esta parte es opcional, pero el servicio web puede filtrar los mensajes por el contenido de esta parte.
- **Body:** Actúa como contenedor para la información que se envía al receptor del mensaje. Los usos típicos son proveer un mecanismo simple de intercambio de información con el receptor del mensaje SOAP. En esta parte del mensaje es donde se encuentran las invocaciones RPC o bien el resultado de la invocación



Arquitectura de funcionamiento de la comunicación mediante SOAP

Ejemplo de cómo pueden ser utilizados los procedimientos SOAP:

Un mensaje SOAP podría ser enviado a un sitio Web que tiene habilitado Web service, para realizar la búsqueda de algún precio en una base de datos, indicando los parámetros necesitados en la consulta. El sitio podría retornar un documento formateado en XML con el resultado: precios, localización, características. Teniendo los datos de respuesta en un formato estandarizado "parseable", este puede ser integrado directamente en un sitio Web o aplicación externa.

Ventajas del uso de SOAP

- **Protocolo abierto:** Esta construido sobre tecnologías también abiertas como XML y HTTP.
- **Simplicidad:** La especificación de SOAP está bien definida y es sumamente simple.
- **Independiente de plataformas y lenguajes:** El uso de XML y HTTP hace a SOAP completamente independiente de plataformas, sistemas operativos o lenguajes de programación.
- **Interoperabilidad:** Uno de los objetivos de SOAP es eliminar las dificultades que separan las plataformas de la programación distribuida. Para esto SOAP se basa en atributos de otros protocolos exitosos para la WEB: simplicidad, flexibilidad, independencia de plataforma y basado en texto.
- **Firewalls:** Al usar HTTP como transporte puede pasar fácilmente a través de los firewalls, los cuales dejan pasar habitualmente el tráfico que les llega por el puerto HTTP.
  Es de destacar que SOAP permite que los administradores configuren los firewalls para bloquear selectivamente los mensajes SOAP usando SOAP headers específicos (el header SOAPACTION, puede ser usado por los firewalls y otros filtros para conocer el requerimiento que llega sin necesidad de parsear el mensaje).

- **Transporte:** La versión 1.0 de SOAP mandataba el uso de HTTP como transporte, pero en la versión 1.1 se flexibilizó esta posición permitiendo el uso de protocolos de transporte como FTP, SMTP, además de HTTP y sus variantes (HTTPS). Puede pensarse que en algún momento se usen otros protocolos como ser IIOP, RMI, etc. y de esta manera aumentar su usabilidad e interoperabilidad.
- **Bajo acoplamiento:** Los sistemas distribuidos basados en SOAP tienen un bajo acoplamiento, por lo cual pueden ser fácilmente mantenidos dado que pueden ser modificados independientemente de otros sistemas.
- **Escalabilidad:** Al usar el protocolo HTTP como transporte, los sistemas distribuidos construidos sobre SOAP son sumamente escalables

Inconvenientes

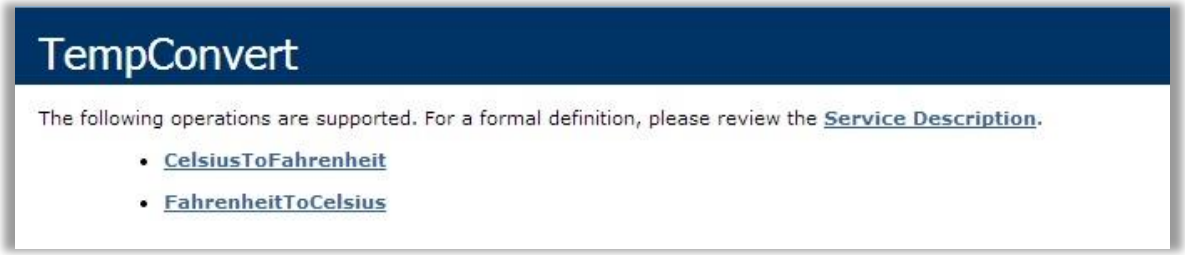
- **Performance:** No es una forma de comunicación compacta, dado que los mensajes están codificados en XML, o sea en un formato ASCII. Esto hace que el transporte sea ineficiente a través de la red, en particular en los casos de grandes conjuntos de datos. Lo contrario sucede con los formatos binarios de datos como mecanismo de comunicación entre aplicaciones, que son usados por ejemplo por CORBA (CDR), DCOM (NDR), etc.
- **Parsing:** Como está basado en XML (formato ASCII) el parsing requiere más recursos de CPU, que otros protocolos basados en formato binario.
- **Serialización/Deserialización:** Como está basado en XML (formato ASCII) el marshalling/unmarshalling requiere más recursos de CPU, que otros protocolos basados en formatos binarios.
- **Serialización :** Todos los datos son serializados y pasados por valor y no por referencia, esto puede provocar problemas de sincronización si múltiples copias de un objeto fuesen pasadas en el mismo momento.
- **Semántica:** SOAP no define la semántica de los mensajes, lo cual significa que la aplicación cliente y la aplicación servidor deben acordar la semántica de los mensajes.

Ejemplo Práctico

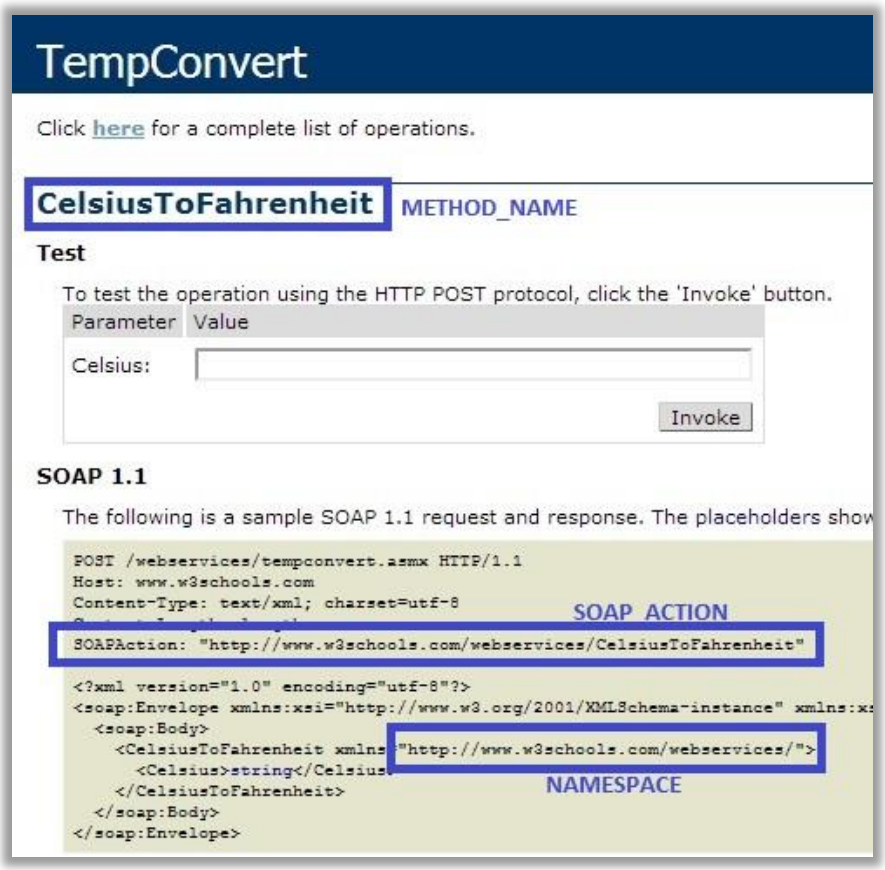
Aquí nos centraremos en el caso concreto de Android. Para ello lo único que necesitamos a mayores en nuestro proyecto es la librería ksoap2-android (que podéis descargar aquí: http://ksoap2-android.googlecode.com/svn/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/3.0.0/ksoap2-android-assembly-3.0.0-jar-with-dependencies.jar).

Como ejemplo práctico voy a utilizar un servicio web ya hecho que encontré en: http://www.w3schools.com/webservices/tempconvert.asmx. Un servicio web SOAP no es más que una aplicación online con una serie de métodos públicos que devuelven un XML según dicho estándar. En este caso el web service es muy sencillo: convierte los grados Celsius a Fahrenheit y viceversa.

Al entrar en la URL anterior os encontraréis con la pantalla principal, en la que se pueden ver los métodos accesibles y una descripción del servicio web en sí.



Esos dos puntos son los métodos disponibles. Si pinchamos sobre ellos aparecerá algo así:

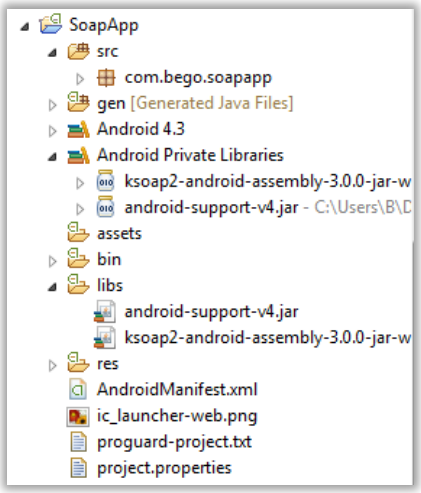


Los datos marcados en las imágenes, junto con la URL del servicio, se corresponden con cuatro constantes que definiremos en nuestro proyecto para trabajar con mayor comodidad.

Si ponemos un valor en la parte que pone Test y clicamos en Invoke, accederemos a otra ventana en la que nos muestra la respuesta en formato XML.

Esto es lo que nos devuelve el web service al invocar el método CelsiusToFahrenheit.

Una vez creado el proyecto, tenemos que añadir la librería para SOAP (descargada previamente). Para ello, podemos arrastrarla hasta la carpeta *libs* de nuestro proyecto. Si la versión del plugin de Android para Eclipse es antigua, quizás haya q ir las propiedades del proyecto y en el apartado *Java Build Path / Libraries* seleccionar *Add External JARs…* y buscarla.



En la imagen vemos que ya está añadida. Ahora podemos pasar a programar la aplicación.

Para asegurarnos de que nuestra aplicación funcione en cualquier versión de Android tendremos que realizar la conexión al servicio en una tarea asíncrona (AsyncTask), ya que a partir de la versión 3.0 no se pueden realizar operaciones de larga duración directamente en el hilo principal. Si lo hiciéramos, nos saltaría una excepción de tipo *NetworkOnMainThread*.

Tendremos que implementar dos métodos propios de las AsyncTask:

- ***doInBackground(String…params).*** Aquí realizaremos la petición SOAP, crearemos el objeto de transporte, enviaremos la petición y recogeremos los datos.
- ***onPostExecute(String result).*** Procederemos a mostrar el resultado.

Una vez creadas las AsyncTask necesarias, empezaremos a codificar su método doInBackground(). Lo primero será definir las constantes mencionadas anteriormente (solo por comodidad):

```
final String NAMESPACE = "http://www.w3schools.com/webservices/";
final String URL="http://www.w3schools.com/webservices/tempconvert.asmx";
final String METHOD_NAME = "CelsiusToFahrenheit";
final String SOAP_ACTION = "http://www.w3schools.com/webservices/CelsiusToFahrenheit";
```

Ahora crearemos la petición al método mediante un SoapObjetc al que le pasamos el namespace y el nombre del método al que va dirigida.

```
SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
```

Mediante *addProperty("key", value)* añadiremos a la petición los parámetros de entrada necesarios (El nombre de la clave viene determinado por el web service y se distingue entre mayúsculas y minúsculas). En este caso un String con el número a convertir.

```
request.addProperty("Celsius", value.getText().toString());
```

Lo siguiente será crear el sobre (envelope) en el que mandaremos nuestra petición. Para ello crearemos un objeto SoapSerializationEnvelope indicando la versión de SOAP que vamos a utilizar.

```
SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER11);
envelope.dotNet = true;
envelope.setOutputSoapObject(request);
```

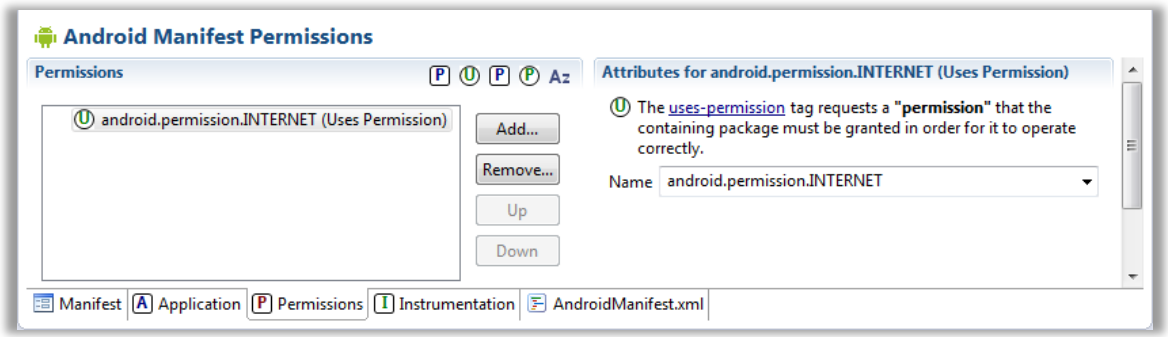
La propiedad *dotNet* indica que se trata de un servicio web .NET, y mediante *setOutputSoapObject()* asociamos la petición.

Hecho esto tenemos que crear el objeto que realizará la comunicación vía HTTP con el servidor y que llamará al servicio web mediante el método *call(SOAP_ACTION, envelope)*. Luego podemos guardar el resultado obtenido, en este caso con un objeto SoapPrimitive, pero esto puede variar dependiendo de lo que nos interese.

```
HttpTransportSE transporte = new HttpTransportSE(URL);

try{
   transporte.call(SOAP_ACTION, envelope);
   SoapPrimitive resultado_xml = (SoapPrimitive)envelope.getResponse();
   resObj = resultado_xml.toString();
}catch(Exception e){
   Toast.makeText(getApplicationContext(), e.getMessage(),
         Toast.LENGTH_SHORT).show();
}
```

Ahora lo único que quedaría por hacer, en éste supuesto, es mostrar el resultado. Esto lo haríamos en el método *onPostExecute().* Otra cosa importante que no se nos puede olvidar es dar permiso para el acceso a internet en el Manifest.



Por último, decir que si tuviéramos que enviar un objeto al método web (como por ejemplo un objeto Cliente), deberíamos modificar la clase del objeto para que SOAP sepa cómo serializar los objetos. Habría que implementar la interface KvmSerializable y los siguientes métodos:

- getProperty(int indice)
- getPropertyCount()
- getPropertyInfo(int indice, HashTable ht, PropertyInfo info)
- setProperty(int indice, Objetc valor)

Imaginemos que tenemos una base de datos de clientes, por ejemplo, de los que almacenamos un id, un nombre y un telefono (en ese orden), el método *getProperty(0)* nos devolvería el valor del id de dicho cliente. El método *gePropertyCount()* devolvería el número de atributos del objeto (o columnas de la tabla); *getPropertyInfo(int indice, HashTable ht, PropertyInfo info)* nos daría, según el índice recibido como parámetro, el tipo y nombre del atributo correspondiente. Por último, *setProperty(int indice, Objetc valor)* que se encarga de asignar el valor de cada atributo según el índice y el valor recibido como parámetro.

Bibliografía

Definición

http://es.wikipedia.org/wiki/Simple_Object_Access_Protocol

Ejemplo práctico

http://www.sgoliver.net/blog/?p=2571

http://www.sgoliver.net/blog/?p=2594

Librería:

http://ksoap2-android.googlecode.com/svn/m2-repo/com/google/code/ksoap2-android/ksoap2-android-assembly/3.0.0/ksoap2-android-assembly-3.0.0-jar-with-dependencies.jar

Web Service

http://www.w3schools.com/webservices/tempconvert.asmx

Otras consultas:

http://www.negomobile.es/es/node/27

http://www.ingens-networks.com/blog/post/2012/05/07/Consumiento-Web-Services-SOAP-en-Android.aspx

http://androidsensei.net/lo-retro-junto-con-lo-nuevo-comunicando-una-aplicacion-android-con-un-webservice-soap/