# 4 SOAP Y WSDL

En este capítulo se van  se va a ver la arquitectura de Servicios Web SOAP y el lenguaje de descripción de Servicios WSDL. Este estudio será necesario para poder realizar comparaciones entre REST y SOAP.

## *4.1 SOAP*

Actualmente un sin fin de empresas se han decantado por el desarrollo de aplicaciones que puedan trabajar sobre ***Internet*** porque permite la distribución global de la información. Las tecnologías más usadas para el desarrollo de estas aplicaciones, han sido hasta hace poco ***CORBA***, ***COM*** y ***EJB.*** Cada una proporciona un marco de trabajo basado en la activación de objetos remotos mediante la solicitud de ejecución de servicios de aplicación a un servidor de aplicaciones.

Estas tecnologías han demostrado ser muy efectivas para el establecimiento de sitios Web, sin embargo presentan una serie de desventajas, como son total incompatibilidad e interoperabilidad entre ellas, total dependencia de la máquina servidora sobre la que corren, así como el lenguaje de programación.

Esto ha llevado a la necesidad de considerar un nuevo modelo de computación distribuida de objetos que no sea dependiente de plataformas, modelos de desarrollo ni lenguajes de programación. Por todos estos motivos surge el concepto de ***SOAP*** *(Simple Object Access Protocol).*

### 4.1.1 Concepto de SOAP

La funcionalidad que aporta **SOAP** es la de proporcionar un mecanismo simple y ligero de intercambio de información entre dos puntos usando el lenguaje ***XML.*** **SOAP** no es más que un mecanismo sencillo de expresar la información mediante un modelo de empaquetado de datos modular y una serie de mecanismos de codificación de datos. Esto permite que **SOAP** sea utilizado en un amplio rango de servidores de aplicaciones que trabajen mediante el modelo de comunicación ***RPC*** *(Remote Procedure Call).*

**SOAP** consta de tres partes:

- El ***SOAP envelope*** que define el marco de trabajo que determina **qué** se puede introducir en un mensaje, **quién** debería hacerlo y si esa operación es **opcional u obligatoria.**

- Las ***reglas de codificación SOAP*** que definen el **mecanismo de serialización** que será usado para encapsular en los mensajes los distintos tipos de datos.

- La representación **SOAP RPC** que define un modo de funcionamiento a la hora de realizar llamadas a procedimientos remotos y la obtención de sus resultados.

### 4.1.2 Objetivos de SOAP

A la hora de realizar el diseño de **SOAP** se han tenido en cuenta una serie de consideraciones con el fin de cumplir una serie de objetivos claros, objetivos que le darán el potencial que reside en **SOAP** y que le harán tan atractivo. Estos objetivos son:

- Establecer un protocolo estándar de invocación a servicios remotos que esté basado en protocolos estándares de uso frecuente en ***Internet***, como son ***HTTP*** *(Hiper Text Transport Protocol)* para la transmisión y ***XML*** *(eXtensible Markup Language)* para la codificación de los datos.

- Independencia de plataforma hardware, lenguaje de programación e implementación del servicio Web.

El logro de estos objetivos ha hecho de SOAP un protocolo extremadamente útil, ya que el protocolo de comunicación ***HTTP*** es el empleado para la conexión sobre ***Internet,*** por lo que se garantiza que cualquier cliente con un navegador estándar pueda conectarse con un servidor remoto. Además, los datos en la transmisión se empaquetan o ***serializan*** con el lenguaje ***XML***, que se ha convertido en algo imprescindible en el intercambio de datos ya que es capaz de salvar las incompatibilidades que existían en el resto de protocolos de representación de datos de la red.

Por otra parte, los servidores Web pueden procesar las peticiones de usuario empleando tecnologías tales como ***Servlets***, páginas ***ASP*** *(Active Server Pages),* páginas ***JSP*** *(Java Server Pages)* o sencillamente un servidor de aplicaciones con invocación de objetos mediante ***CORBA, COM o EJB***.

Un ejemplo típico de diseño de un **servicio Web** utilizando las ventajas de **SOAP** podría ser el mostrado en la Figura [6].

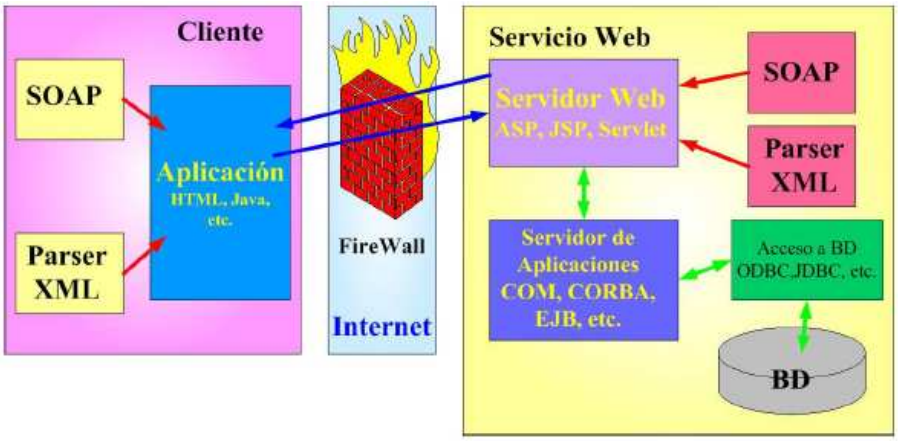


Figura 6: Ejemplo de uso de SOAP

La ***especificación SOAP (SOAP Specification 1.1)*** indica que las aplicaciones deben ser independientes del lenguaje de desarrollo, por lo que las aplicaciones cliente y servidor pueden estar escritas con ***HTML, DHTML, Java, Visual Basic*** o cualquier otra herramienta o lenguaje disponibles.

### 4.1.3 Un ejemplo sencillo de mensajes SOAP

Supongamos que tenemos un servicio Web el cual contiene una relación de productos junto con una serie de características de éstos, y supongamos que le queremos hacer una consulta acerca del precio de uno de los productos cuyo código es **RHAT**, el código relativo a la petición de consulta en lenguaje **SOAP** sería:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
                    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

   <SOAP-ENV:Body>

        <getQuote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
                <symbol>RHAT</symbol>
        </getQuote>

   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Como respuesta a esta petición vendría de vuelta el siguiente mensaje en el que se le dice que el precio del producto pedido es *4.12*:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
                    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

   <SOAP-ENV:Body>

        <Quote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
                <Price>4.12</Price>
        </Quote>

   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

#### 4.1.3.1 Partes de un mensaje SOAP

Un mensaje **SOAP** no es más que un documento en formato **XML** que está constituido por tres partes bien definidas que son: el ***SOAP envelope,*** el ***SOAP header*** de carácter opcional y el ***SOAP body.*** Cada uno de estos elementos contiene lo siguiente:

• El ***envelope*** es el elemento más importante y de mayor jerarquía dentro del documento **XML** y representa al mensaje que lleva almacenado dicho documento.

• El ***header*** es un mecanismo genérico que se utiliza para añadir características adicionales al mensaje **SOAP**. El modo en la que se añadan cada uno de los campos dependerá exclusivamente del servicio implementado entre cliente y servidor, de forma que cliente y servidor deberán estar de acuerdo con la jerarquía con la que se hayan añadido los distintos campos. De esta forma será sencillo separar entre sí los distintos datos a transmitir dentro del mensaje.

• El ***body*** es un contenedor de información en el cual se almacenarán los datos que se quieran transmitir de lado a lado de la comunicación. Dentro de este campo, **SOAP** define un elemento de uso opcional denominado ***Fault*** utilizado en los mensajes de respuesta para indicar al cliente algún error ocurrido en el servidor.

Un ejemplo de uso del ***header***, donde se indica un cierto parámetro útil para el servicio Web que le indica como debe procesar el mensaje sería:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
                xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
   SOAP- ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

        <SOAP-ENV:Header>
                <t:Transaction   xmlns:t="some-URI"   SOAP-ENV:mustUnderstand="1">
                        5
                </t:Transaction>
        </SOAP-ENV:Header>

        <SOAP-ENV:Body>
                <getQuote xmlns= "http://namespaces.cafeconleche.org/xmljava/ch2/">
                        <symbol>RHAT</symbol>
                </getQuote>
        </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Un ejemplo de uso del nuevo elemento ***body*** sería el siguiente, en el que se ha detectado un error en la aplicación que corre sobre el servidor que provoca que no opere convenientemente, por lo que se le indica al cliente:

```
<?xml version="1.0"?>
<SOAP-ENV:Envelope
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      SOAP-ENV:encodingStyle= "http://schemas.xmlsoap.org/soap/encoding/">

      <SOAP-ENV:Body>

            <SOAP-ENV:Fault>
                  <faultcode>SOAP-ENV:Server</faultcode>

                  <faultstring>Server Error</faultstring>

            </SOAP-ENV:Fault>

      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

El atributo *encodingStyle* se utiliza para indicar las reglas de *serialización* utilizadas en el mensaje **SOAP.** No existe un formato de codificación por defecto, sino que existen una serie de posibles formatos a utilizar. El valor de este atributo es una lista ordenada de una o más ***URIs*** que identifican la regla o reglas de serialización que pueden ser utilizadas en el mensaje, en el orden en el que se han de aplicar. De entre todas las posibles, las más utilizadas son:

- *http://schemas.xmlsoap.org/soap/encoding/* y
- *http://my.host/encoding/restricted http://my.host/encoding/***.**

Todo mensaje **SOAP** debe tener un elemento ***Envelope*** asociado a la dirección de red *http://schemas.xmlsoap.org/soap/envelope/.* Si un mensaje recibido por una aplicación **SOAP** contiene en este elemento un valor distinto al anterior la aplicación trataría dicho mensaje como erróneo.

### 4.1.4 Serialización

A la hora de introducir los datos en un mensaje **SOAP** existen una serie de normas a tener en cuenta. De esta forma **SOAP** define una serie de tipos básicos que serán empaquetados de forma directa y una serie de mecanismos para empaquetar tipos complejos y estructurados formados por elementos simples.

En un inicio, **SOAP** consideró un conjunto de tipos como tipos simples con el fin de realizar un *mapeo* directo entre el propio documento **SOAP** y tipos básicos de **Java.** Ahora bien, actualmente este conjunto ha aumentado notablemente de forma que existe

un número bastante grande de tipos que gozan de esta situación. De entre todos ellos destacan los indicados en la Figura [7].

| TIPOS SOAP | TIPOS JAVA |
|---|---|
| SOAP-ENC:int | Java.lang.Integer |
| SOAP-ENC:long | Java.lang.Long |
| SOAP-ENC:short | Java.lang.Short |
| SOAP-ENC:string | Java.lang.String |
| SOAP-ENC:bolean | Java.lang.Boolean |
| SOAP-ENC:flota | Java.lang.Float |
| SOAP-ENC:double | Java.lang.Double |
| SOAP-ENC:byte | Java.lang.Byte |

Tabla 2 Mapeo entre tipos SOAP y JAVA

Además de todos estos tipos sencillos, **SOAP** implementa una serie de mecanismos para *serializar* tipos complejos con elementos simples en su interior. De esta forma tenemos:

• ***Structs***

Un ***struct*** no es más que un elemento que contiene un conjunto de elementos *hijos* almacenados cada uno de ellos en un campo propio. Un ejemplo podría ser:

```
<Quote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
  <Symbol>RHAT</Symbol>
  <Price>4.12</Price>
</Quote>
```

En términos de **Java** tendríamos un objeto de tipo ***Quote*** dentro del cual hay dos elementos sencillos: uno de tipo *String* llamado ***Symbol*** y el otro de tipo *double* y de nombre ***Price.*** Es decir:

```
public class Quote {

  public String Symbol();
  public double Price();
}
```

• ***Arrays***

Un ***array*** en un mensaje **SOAP** es representado mediante un elemento cuyo tipo es ***SOAP-ENC:Array.***

Aunque en Java sea obligatorio que dentro de un array haya únicamente elementos del mismo tipo, SOAP no presenta esta restricción, sino que es posible albergar elementos de distintos tipos, así un ejemplo sería:

```
<Bid xsi:type="SOAP-ENC:Array">
  <Symbol xsi:type="xsd:token">RHAT</Symbol>
  <Price xsi:type="xsd:double">4.12</Price>
  <Account xsi:type="xsd:string">777-7777</Account>
</Bid>
```

Si quisiésemos restringir explícitamente el tipo de los elementos almacenados en el ***array*** podríamos hacerlo de la forma:

```
<Bid xsi:type="SOAP-ENC:Array"     SOAP-ENC:arrayType="xsd:double[3]">
  <Price>4.52</Price>
  <Price>0.35</Price>
  <Price>34.68</Price>
</Bid>
```

En **SOAP** también es posible implementar ***arrays*** bidimensionales, incluso ***arrays*** que almacenan *strings*. Un ejemplo de esto sería:

```
<SOAP-ENC:Array     SOAP-ENC:arrayType="xsd:double[3,2]">
  <SOAP-ENC:double>1.1</SOAP-ENC:double>
  <SOAP-ENC:double>1.2</SOAP-ENC:double>
  <SOAP-ENC:double>2.1</SOAP-ENC:double>
  <SOAP-ENC:double>2.2</SOAP-ENC:double>
  <SOAP-ENC:double>3.1</SOAP-ENC:double>
  <SOAP-ENC:double>3.2</SOAP-ENC:double>
</SOAP-ENC:Array>
```

Este mismo Array que hemos explicado su realización en **SOAP**, en tecnología **Java** sería de la manera:

```
double[][] array = {
```

```
  {1.1, 1.2},
  {2.1, 2.2},
  {3.1, 3.2}
}
```

O un ejemplo algo más complicado sería un A*rray* que contiene a su vez dos *arrays* cada uno de los cuales contiene a su vez un A*rray* de S*trings:*

```
<SOAP-ENC:Array SOAP-ENC:arrayType="xsd:string[][2]">
  <item href="#array-1"/>
  <item href="#array-2"/>
</SOAP-ENC:Array>

<SOAP-ENC:Array id="array-1" SOAP-ENC:arrayType="xsd:string[2]">
  <item>r1c1</item>
  <item>r1c2</item>
  <item>r1c3</item>
</SOAP-ENC:Array>

<SOAP-ENC:Array id="array-2"   SOAP-ENC:arrayType="xsd:string[2]">
  <item>r2c1</item>
  <item>r2c2</item>
</SOAP-ENC:Array>
```

### 4.1.5 Protocolo HTTP mediante SOAP

Una de las funcionalidades con las que cuenta **SOAP** y que lo hace extremadamente atractivo es el envío de mensajes **SOAP** vía protocolo **HTTP,** todo ello realizado de forma directa por medio de las librerías de **SOAP.**

Este hecho provoca que existen ciertas diferencias entre un mensaje **SOAP** normal y uno enviado haciendo uso del protocolo **HTTP.** De esta forma deben cumplirse dos aspectos:

• El ***HTTP header*** del mensaje de petición al servicio Web debe contener un campo ***SOAPAction.*** El campo ***SOAPAction*** alerta a los *servidores Web* y *firewalls* por los que pase de que el mensaje contiene documento **SOAP,** esto facilita a dichos *firewalls* el filtrado de las peticiones **SOAP** sin necesidad de tener que explorar el contenido del ***body*** del mensaje.

• Si la petición **SOAP** al servicio Web falla, el servidor debe devolver en el mensaje de respuesta un error del tipo **HTTP 500 Internal Server Error** en vez de un **200 OK** que se envía cuando todo ha ido bien.

Un ejemplo de todo esto sería el siguiente extracto de código en el que lanza una petición **HTTP** de tipo **POST** contra un ***Servlet*** que corre en la dirección Web *www.ibiblio.org* bajo el control del usuario de nombre *elharo:*

```
POST /xml/cgi-bin/SOAPHandler HTTP/1.1
Content-Type: text/xml; charset="utf-8"
Content-Length: 267
SOAPAction: "http://www.ibiblio.org/#elharo"

<?xml version="1.0"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >

  <SOAP-ENV:Body>
        <getQuote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
              <symbol>RHAT</symbol>
        </getQuote>
  </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

El servidor envía la respuesta de vuelta al cliente y después cierra la conexión. Como cualquier otra respuesta **HTTP** el mensaje **SOAP** empieza con el **código de retorno HTTP.** Suponiendo que la petición tuvo éxito y no ocurrió ningún error en el servidor, el mensaje de respuesta sería:

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: 260

<?xml version="1.0"?>
<SOAP-ENV:Envelope
        xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

      <SOAP-ENV:Body>
        <Quote xmlns="http://namespaces.cafeconleche.org/xmljava/ch2/">
              <Price>4.12</Price>
        </Quote>
      </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

## 4.2 WSDL

En este apartado se verá el lenguaje de descripción de Servicios Web **WSDL 1.1** [5], cuya especificación completa [WSDL 1.1] puede encontrarse en el W3C.

El esquema ***XML*** por sí solo **no puede describir** totalmente un **Servicio Web**. A continuación se verá un ejemplo en el que se supone que se ha creado un servicio Web Calculadora. Este servicio Web expone los métodos sumar y restar. Ambos métodos aceptan dos enteros y devuelven un único entero con el resultado; sumar devuelve la suma de los dos enteros y restar devuelve su diferencia.

En un esfuerzo para describir cómo interacciona un cliente con el servicio Web se define un **esquema para los mensajes** que se intercambiarán entre el cliente y el servidor. El esquema contiene una definición de un tipo de complejo para los mensajes de petición y repuesta para los métodos sumar y restar. Recordemos que el objetivo último es que los desarrolladores no tengan que investigar en las definiciones del esquema intentando descifrar cómo interaccionar con el servicio Web. En lugar de ello se quiere describir el servicio de forma que una herramienta pueda descifrarlo y crear un proxy por el cliente.

Además de la información que proporciona el esquema, ¿Qué más necesita conocer el cliente para invocar los métodos que expone el Servicio Web Calculadora? Como el cuerpo de un mensaje de SOAP puede contener cualquier cosa que no invalide el XML los mensajes de SOAP se pueden combinar para disponer de una amplia variedad de patrones de intercambio de mensajes. Los patrones de intercambio de mensajes para el Servicio Web Calculadora son bastante inmediatos pero una asociación formal entre los mensajes de petición Sumar y Restar y sus mensajes de respuesta asociados eliminarían cualquier posible ambigüedad.

Una descripción formal de los patrones de mensaje resulta aún más importante en el caso de Servicio Web más complejos. Algunos servicios podrían aceptar una petición pero no enviar la respuesta correspondiente devuelta al cliente. Otros podrían solamente enviar mensajes al cliente. Además, el esquema no contiene información sobre cómo acceder al Servicio Web. Como SOAP es independiente del protocolo, se intercambiarán los mensajes entre el cliente y el servidor de numerosas formas. ¿Cómo se sabe si hay que enviar un mensaje mediante HTTP, SMTP o cualquier otro protocolo de transporte? Más aún, ¿cómo se sabe la dirección la que hay que enviar el mensaje?

El lenguaje de descripción de servicios Web (WSDL, Web Service Description Language) es un dialecto basado en XML sobre el esquema que describe un servicio Web. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el servicio Web. WSDL es extensible y se pude utilizar para describir, prácticamente, cualquier servicio de red, incluyendo SOAP sobre HTTP e incluso protocolos que no se basan en XML como DCOM sobre UDP.

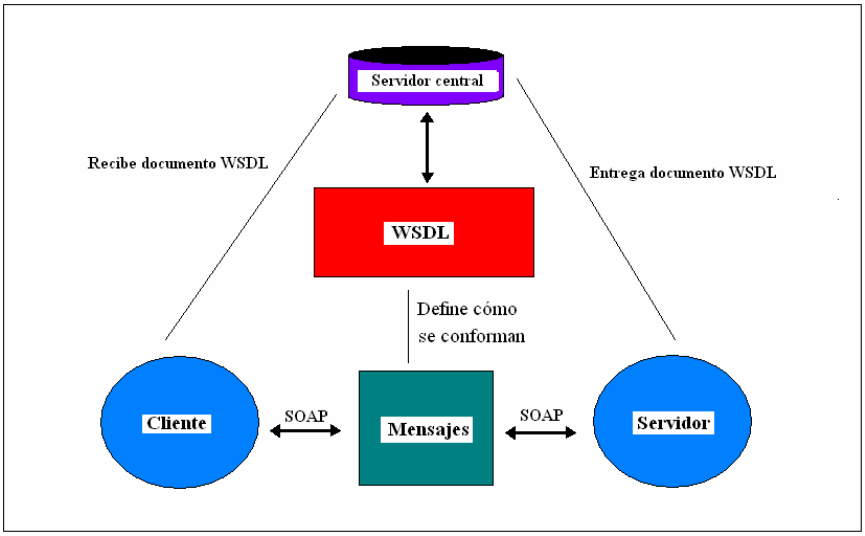


Figura 7: Esquema WSDL

Dado que los protocolos de comunicaciones y los formatos de mensajes están estandarizados en la comunidad del Web, cada día aumenta la posibilidad e importancia de describir las comunicaciones de forma estructurada. WSDL afronta esta necesidad definiendo una gramática XML que describe los servicios de red como colecciones de puntos finales de comunicación capaces de intercambiar mensajes. Las definiciones de servicio de WSDL proporcionan documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles que toman parte en la comunicación entre aplicaciones.

Los documentos WSDL definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio. Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red:

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type**: conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.

A continuación se detalla un poco más en profundidad cada uno de estos elementos

Elemento **Types:**

El elemento Types contiene información de esquema referenciado en el documento WSDL. El sistema de tipos predeterminado que admite WSDL es de esquema de XML. Si se usa esquema de XML para definir los tipos que contiene el elemento Types, el elemento schema aparecerá inmediatamente como elemento hijo. Se pueden utilizar otros sistemas de tipos por extensión. Si se desea utilizar otro sistema de tipo puede aparecer un elemento de extensibilidad bajo el elemento Types. El nombre de este elemento debería identificar el sistema de tipos utilizados. En este capítulo se limitará a tratar el esquema de XML porque es el sistema de tipos dominante en los documentos WSDL.

Elemento **message**:

El elemento Message proporciona una abstracción común para el paso de mensajes entre el cliente y el servidor. Como puede utilizar múltiples formatos de de definición de esquema en documento WSDL es necesario disponer de un mecanismo común para identificar los mensajes. El elemento Message proporciona este nivel común de abstracción al que se hará referencia en otras partes del documento WSDL.

Puede aparecer, y normalmente aparecerán, múltiples elementos Message en un documento WSDL, uno para cada mensaje que se comunica entre el cliente y el servidor. Cada mensaje contiene uno o más elementos "Part" que describen las piezas del contenido del mensaje. Un ejemplo de una parte es el cuerpo de un mensaje de SOAP o un parámetro que forma parte de una cadena de petición, un parámetro codificado en el cuerpo del mensaje de SOAP o todo el cuerpo de un mensaje de SOAP.

Elemento **PortType**:

El elemento PorType contiene un conjunto de operaciones abstractas que representan los tipos de correspondencia que pueden producirse entre el cliente y el servidor. Para los Servicios Web de estilo RPC se puede pensar en un PorType como una definición en donde cada método se puede definir como una operación.

Un tipo puerto se compone de un conjunto de elementos Operation que define una determinada acción. Los elementos Operation se componen de mensajes definidos en el documento WSDL. WSDL define cuatro tipos de operaciones denominadas tipo operaciones:

- **Request-response** (petición-respuesta): Comunicación del tipo RPC en la que le cliente realiza una petición y el servidor envía la correspondiente respuesta.
- **One-way** (un-sentido): Comunicación del estilo documento en la que el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.

- **Solicit-response** (solicitud-respuesta): La contraria a la operación petición-respuesta. El servidor envía una petición y el cliente le envía de vuelta una respuesta.
- **Notification** (Notificación): La contraria a la operación un-sentido el servidor envía una comunicación del estilo documento al cliente.

Elemento **Binding**:

El elemento Binding contiene las definiciones de la asociación de un protocolo como SOAP a un determinado BindingType. Las definiciones Binding especifican detalles de formatos del mensaje y el protocolo. Por ejemplo, la información de asociación especifica si se puede acceder a una instancia de un PortType de forma RPC.

Las definiciones Binding también indican el número de comunicaciones de red que se requieren para realizar una determinada acción. Por ejemplo, una llamada RPC de SOAP sobre HTTP podría involucrar un intercambio de comunicación HTTP, pero esa misma llamada sobre SMTP podría involucrar dos intercambios de comunicaciones de SMTP discretas.

La asociación se logra utilizando elementos de extensión. Cada protocolo tiene su propio conjunto de elementos de extensión para especificar los detalles del protocolo y el formato de los mensajes. Para un determinado protocolo los elementos de extensión se suelen utilizar para decorar las acciones individuales de una operación y la propia operación con la información de asociación del protocolo. A veces los elementos de extensión se utilizan en el propio nivel PortType.

Elemento **Service**:

Un servicio es un grupo de puertos relacionados y se definen en el elemento Service. Un puerto es un extremo concreto de un Servicio Web al que se hace referencia por una dirección única. Los puertos que se definen en determinado servicio son independientes. Por ejemplo, la salida de un puerto que no puede utilizarse como una entrada de otro.

Elemento **Extensibilidad :**

Los elementos de Extensibilidad se utilizan para representar determinadas tecnologías. Por ejemplo, se puede utilizar los elementos de extensibilidad para especificar el idioma en que se utiliza en el esquema de los elementos Types.

El esquema para un determinado conjunto de elementos de extensibilidad se debe definir dentro de distintos espacios de nombres que WSDL. La definición de los propios elementos puede contener un atributo wsdl:required que indique un valor boolean si el atributo Requiered se establece a “TRUE”. En una definición de elementos una asociación que haga referencia a ese conjunto concreto de electos de extensibilidad tiene que incluir dicho elemento.

Lo más habitual es que los elementos de extensibilidad se utilicen para especificar especificación de asociación. La especificación WSDL define conjunto de elementos de extensibilidad para la asociación SOAP, HTTP GET, HTTP POS, MIME. Sin embargo, la especificación sólo define las asociaciones para dos de los cuatro tipos de operaciones. Un sentido y petición repuesta.

## *4.3 Conclusiones*

En este capítulo se ha visto la arquitectura de Servicios Web SOAP y el lenguaje de descripción de Servicios WSDL. En el siguiente capítulo se estudiará ampliamente REST, que se presenta como una alternativa a SOAP en el mundo de los Servicios Web.