Unidad 6

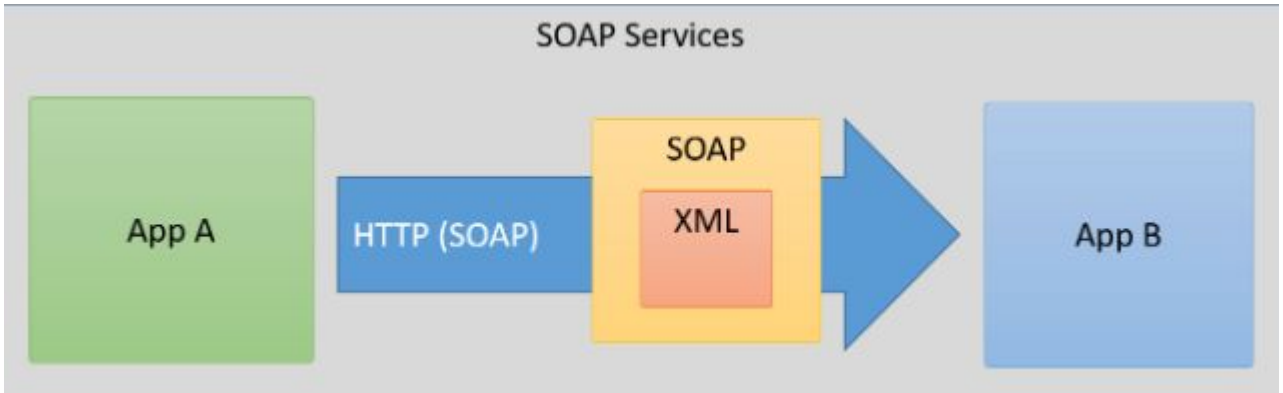
Servicios web.
Intercambio de información: SOAP

1. Servicios web SOAP

En ocasiones, las aplicaciones que desarrolles necesitarán compartir información con otras aplicaciones.

Un **servicio web** (servicios SOAP) (Web Service) es un método que permite que dos equipos intercambien información a través de una red informática.

Al utilizar servicios web, el servidor puede ofrecer un punto de acceso a la información que quiere compartir. De esta forma, controla y facilita el acceso a la misma por parte de otras aplicaciones.



1.1 Características

Existen numerosos protocolos que permiten la comunicación entre ordenadores a través de una red: FTP, HTTP, SMTP, POP3, TELNET, etc. En todos estos protocolos se definen un servidor y un cliente.

> El **servidor** es la máquina que está esperando conexiones (escuchando) por parte de un cliente.
> El **cliente** es la máquina que inicia la comunicación. Cada uno de estos protocolos tiene asignado además un puerto (TCP o UDP) concreto, que será el que utilicen normalmente los equipos servidores.

Cada uno de los protocolos que hemos nombrado ha sido creado para un fin específico:

- FTP: transferencias de archivos.
- HTTP para páginas web.
- SMTP y POP3 para correo electrónico.

TELNET para acceso remoto.

1.1 Características

No han sido diseñados para transportar peticiones de información genéricas entre aplicaciones, como solicitar el PVP de un producto. Sin embargo, desde hace tiempo existen otras soluciones para este tipo de problemas, una de las más populares es RPC.

Protocolo RPC: se creó para permitir a un sistema acceder de forma remota a funciones o procedimientos que se encuentren en otro sistema. El cliente se conecta con el servidor y le indica qué función debe ejecutar. El servidor la ejecuta y le devuelve el resultado obtenido.

Por ejemplo, podemos crear en el servidor RPC una función que reciba un código de producto y devuelva su PVP.

RPC usa su propio puerto (1024-5000), pero normalmente solo a modo de directorio. Los clientes se conectan a él para obtener el puerto real del servicio que les interesa. Este puerto no es fijo; se asigna de forma dinámica.

1.1 Características

Los servicios web se crearon para permitir el intercambio de información al igual que RPC pero sobre la base del protocolo HTTP. En lugar de definir su propio protocolo para transportar las peticiones de información, utilizan HTTP para este fin. **La respuesta obtenida no será una página web, sino la información que se solicitó.**

De esta forma pueden funcionar sobre cualquier servidor web; y, lo que es aún más importante, utilizando el puerto 80 reservado para este protocolo.

Por tanto, cualquier ordenador que pueda consultar una página web, podrá también solicitar información de un servicio web. Si existe algún cortafuegos en la red, tratará la petición de información igual que lo haría con la solicitud de una página web

1.1 Características

Existen al menos dos cuestiones que debería resolver un servicio web para poder funcionar correctamente:

- **Cómo se transmite la información.** Si se va a usar HTTP para las peticiones y las respuestas, el cliente y el servidor tendrán que ponerse de acuerdo en la forma de enviar unas y otras. Es decir, ¿cómo hace el cliente para indicar que quiere conocer el PVP del artículo con código X?, y también, ¿cómo envía el servidor la respuesta obtenida?

- **Cómo se publican las funciones a las que se puede acceder en un servidor determinado**. Este punto es opcional, pero muy útil. Es decir, el cliente puede saber que la función del servidor que tiene que utilizar se llama getPVPArticulo, y que debe recibir como parámetro el código del artículo. Pero si no lo sabe, sería útil que hubiera un mecanismo donde pudiera consultar las funciones que existen en el servidor y cómo se utiliza cada una.

1.1 Características

Cada uno de los métodos que podemos utilizar hoy en día para crear un servicio web responde a estas preguntas de formas distintas:

- **Cómo se transmite la información.** → Protocolo SOAP → Utiliza **lenguaje XML** → Intercambia información

- **Cómo se publican las funciones a las que se puede acceder en un servidor determinado**. → **Lenguaje WSDL** → Describir servicios web → indica cómo se debe acceder a un servicio y utilizarlo

1.2 Intercambio de información: SOAP

SOAP es un protocolo que indica cómo deben ser los mensajes que se intercambien el servidor y el cliente, cómo deben procesarse éstos, y cómo se relacionan con el protocolo que se utiliza para transportarlos de un extremo a otro de la comunicación.
En el caso de los servicios web, este protocolo será HTTP.

SOAP utiliza **XML** para componer los mensajes que se transmiten entre el cliente (que genera una petición) y el servidor (que envía una respuesta) del servicio web.

NOTA: Aunque nosotros vamos a utilizar HTTP para transmitir la información, SOAP **no requiere el uso de un protocolo concreto** para transmitir la información. SOAP se limita a definir las reglas que rigen los mensajes que se deben intercambiar el cliente y el servidor.

1.2 Intercambio de información: SOAP

Ejemplo: Si implementamos un servicio web para informar sobre el precio de los artículos que se venden en la tienda web, una petición de información para el artículo con código ‘KSTMSDHC8GB’ podría ser de la siguiente forma:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1=""
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
  <soap:Body>
    <ns1:getPVP>
      <param0 xsi:type="xsd:string">KSTMSDHC8GB</param0>
    </ns1:getPVP>
  </soap:Body>
</soap:Envelope>
```

1.2 Intercambio de información: SOAP

Ejemplo: Y su respectiva respuesta:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:ns1=""
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
>
  <soap:Body>
    <ns1:getPVPResponse>
      <return xsi:type="xsd:string">10.20</return>
    </ns1:getPVPResponse>
  </soap:Body>
</soap:Envelope>
```

1.2 Intercambio de información: SOAP

SOAP y PHP

En PHP se pueden utilizar 3 implementaciones de SOAP:

- **PHP5 SOAP** es la implementación de SOAP que se incluye con PHP a partir de la versión 5 del lenguaje. En versiones anteriores se tenía que recurrir a otras opciones para trabajar con SOAP. Es una extensión nativa (escrita en lenguaje C) y por tanto más rápida que las otras posibilidades. Como veremos más adelante, su gran inconveniente es que **no permite la generación automática del documento WSDL una vez programado el servidor SOAP correspondiente.**

- **NuSOAP** es un conjunto de clases programadas en PHP que ofrecen muchas funcionalidades para utilizar SOAP. Al contrario que PHP5 SOAP, funcionan también con PHP4, y además **permite generar automáticamente el documento WSDL correspondiente a un servicio web**.

- **PEAR::SOAP** es un paquete PEAR que permite utilizar SOAP con PHP a partir de su versión 4. Al igual que NuSOAP, también está programado en PHP.

1.3 Descripción del servicio: WSDL

Una vez que hayas creado un servicio web, puedes programar el correspondiente cliente y comenzar a utilizarlo. Si el servicio lo has creado tú, sabrás cómo acceder a él: en qué URL está accesible, qué parámetros recibe, cuál es la funcionalidad que aporta y qué valores devuelve.

Si lo que quieres es que el servicio web sea accesible a aplicaciones desarrolladas por otros programadores, deberás indicarles cómo usarlo, es decir, crear un **documento WSDL** que describa el servicio.

WSDL es un lenguaje basado en XML que utiliza unas reglas determinadas para generar el documento de descripción de un servicio web. Una vez generado, ese documento se suele poner a disposición de los posibles usuarios del servicio.

Normalmente se accede al documento WSDL añadiendo `?wsdl` a la URL del servicio.

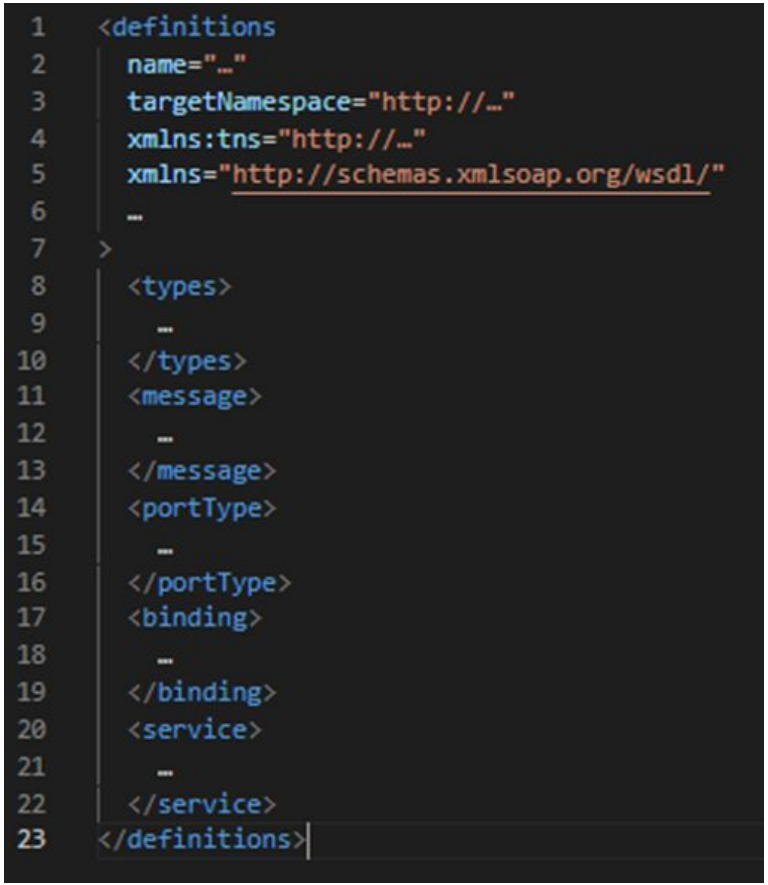
El espacio de nombres de un documento WSDL es http://schemas.xmlsoap.org/wsdl/ aunque en un documento WSDL se suelen utilizar también otros espacios de nombres.

1.3 Descripción del servicio: WSDL

Estructura del documento WSDL

El objetivo de cada una de las secciones del documento es el siguiente:

- **`types`**. Incluye las definiciones de los tipos de datos que se usan en el servicio.
- **`message`.** Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.
- **`portType`.** Cada portType es un grupo de funciones que implementa el servicio web. Cada función se define dentro de su portType como una operación (operation).
- **`binding`.** Define cómo va a transmitirse la información de cada portType.
- **`service`**. Contiene una lista de elementos de tipo port. Cada port indica dónde (en qué URL) se puede acceder al servicio web.

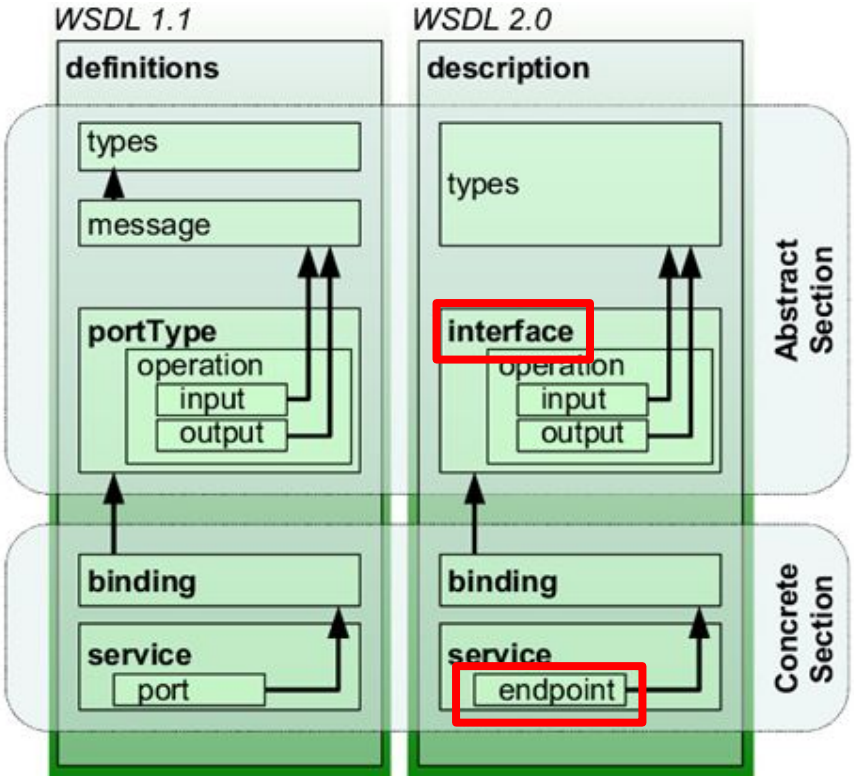


1.3 Descripción del servicio: WSDL

Estructura del documento WSDL

Las secciones anteriores son las correspondientes a la versión 1.1 de WSDL.

En la versión 2.0, los conceptos y nomenclatura cambian ligeramente; por ejemplo, en lo que en 1.1 es un portType se denomina interface en la versión 2.0



1.3 Descripción del servicio: WSDL

1.3.1 `types` Incluye las definiciones de los tipos de datos que se usan en el servicio.

En el ejemplo: se definen **2 tipos de datos** usando XML Schema: `dirección y usuario.`

En muchas ocasiones no será necesario definir tipos propios, y por tanto en el documento WSDL **no habrá** sección `types`; será suficiente con utilizar alguno de los tipos propios de XML Schema, como xsd:string, xsd:float o xsd:boolean.

```
<types>
  <xsd:schema targetNamespace="http://localhost/DWEES/unidad6">
   <xsd:complexType name="direccion">
    <xsd:all>
      <xsd:element name="ciudad" type="xsd:string"/>
      <xsd:element name="calle" type="xsd:string"/>
      <xsd:element name="numero" type="xsd:string"/>
      <xsd:element name="piso" type="xsd:string"/>
      <xsd:element name="CP" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
   <xsd:complexType name="usuario">
    <xsd:all>
      <xsd:element name="id" type="xsd:int"/>
      <xsd:element name="nombre" type="xsd:string"/>
      <xsd:element name="direccion" type="tns:direccion"/>
      <xsd:element name="email" type="xsd:string"/>
    </xsd:all>
   </xsd:complexType>
  </xsd:schema>
</types>
```

1.3 Descripción del servicio: WSDL

1.3.1 `types` Incluye las definiciones de los tipos de datos que se usan en el servicio.

Definición de clases en WSDL

En WSDL, las clases se definen utilizando los tipos complejos de XML Schema. Al utilizar `all` dentro del tipo complejo, estamos indicando que la clase contiene esos miembros, aunque no necesariamente en el orden que se indica (si en lugar de all hubiésemos utilizando `sequence`, el orden de los miembros de la clase debería ser el mismo que figura en el documento).

- Los métodos de la clase forman parte de la lógica de la aplicación y no se definen en el documento WSDL
- Aunque en WSDL se puede usar cualquier lenguaje para definir los tipos de datos, es aconsejable usar XML Schema, indicándolo dentro de la etiqueta `types` e incluyendo el espacio de nombres correspondiente en el elemento `definitions`.

```
<definitions
  name="WSDLusuario"
  targetNamespace="http://localhost/DWEES/Unidad6"
  xmlns:tns="http://localhost/DWEES/Unidad6"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  …
>
```

1.3 Descripción del servicio: WSDL

1.3.1 `types` Incluye las definiciones de los tipos de datos que se usan en el servicio.

Definición de arrays en WSDL

Al definir un array en WSDL, se debe tener en cuenta que:

- El atributo `arrayType` se utiliza para indicar qué elementos contiene el array.
- Se debe añadir al documento el espacio de nombres SOAP `encodingc:`

```
<definitions
  name="WSDLusuario"
  targetNamespace="http://localhost/DWEES/Unidad6"
  xmlns:tns="http://localhost/DWEES/Unidad6"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  …
>
```

- El nombre del array debería ser `ArrayOfXXX`, dónde XXX es el nombre del tipo de elementos que contiene el array.

1.3 Descripción del servicio: WSDL

1.3.2 message Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.

Después del tipo, el siguiente paso es indicar cómo se agrupan estos tipos para formar los parámetros de entrada y de salida.

Veamos un ejemplo:
Siguiendo con los usuarios que acabamos de definir, podríamos crear en el servicio web una función `getUsuario` para dar acceso a los datos de un usuario.
Como parámetro de esa función vamos a pedir el id del usuario, y como valor de salida se obtendrá un objeto usuario.
Por tanto, debemos definir los siguientes mensajes:

1.3 Descripción del servicio: WSDL

1.3.2 message Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.

Como podemos ver, para cada función del servicio web se crea un mensaje (`message`) para los parámetros de entrada y otro para los de salida. Dentro de cada mensaje, se incluirán tantos elementos `part` como sea necesario. Cada mensaje contendrá un atributo `name` que debe ser único para todos los elementos de este.

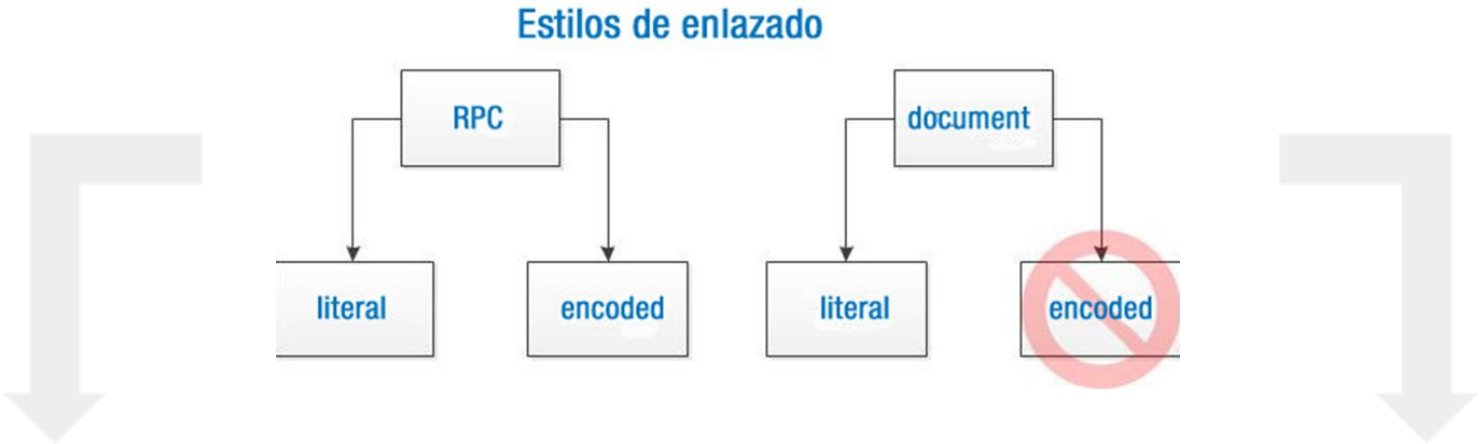
```
1  <message name="getUsuarioRequest">
2    <part name="id" type="xsd:int"/>
3  </message>
4  <message name="getUsuarioResponse">
5    <part name="getUsuarioReturn" type="tns:usuario"/>
6  </message>
```

Es aconsejable que el nombre del mensaje con los parámetros de entrada acabe en Request y los de salida en Response.

1.3 Descripción del servicio: WSDL

1.3.2 message Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.

En un documento WSDL podemos especificar 2 estilos de enlazado: `document` o RPC




```
17  <SOAP-ENV:Body>
18    <ns1:getPVP>
19      <param0 xsi:type="xsd:string">KSTMSDHC8GB</param0>
20    </ns1:getPVP>
21  </SOAP-ENV:Body>
```

```
10  <SOAP-ENV:Body>
11    <producto>
12      <codigo>KSTMSDHC8GB</codigo>
13    </producto>
14  </SOAP-ENV:Body>
```

1.3 Descripción del servicio: WSDL

1.3.2 message Define conjuntos de datos, como la lista de parámetros que recibe una función o los valores que devuelve.

Además, cada estilo de enlazado puede ser de tipo encoded o literal. La combinación document/encoded no se utiliza

Al indicar encoded, estamos diciendo que vamos a usar un conjunto de reglas de codificación, como las que se incluyen en el propio protocolo SOAP (espacio de nombres http://schemas.xmlsoap.org/soap/encoding/), para convertir en XML los parámetros de las peticiones y respuestas.

El ejemplo anterior de RPC es en realidad RPC/encoded. Un ejemplo de un mensaje SOAP con estilo RPC/literal sería:

```
24    <SOAP-ENV:Body>
25      <ns1:getPVP>
26        <param0>KSTMSDHC8GB</param0>
27      </ns1:getPVP>
28    </SOAP-ENV:Body>
```

Trabajaremos únicamente con estilo enlazado RPC/encoded.

1.3 Descripción del servicio: WSDL

1.3.3 `portType` Cada portType es un grupo de funciones que implementa el servicio web.

Un portType contiene una lista de funciones, indicando para cada función (`operation`) la lista de parámetros de entrada y de salida que le corresponden

El documento WSDL contendrá un único portType (a no ser que estés generando un servicio web bastante complejo).

Cada `portType` debe contener un atributo name con el nombre (único para todos los elementos portType).

Cada elemento `operation` también debe contener un atributo name, que se corresponderá con el nombre de la función que se ofrece.

```
31  <portType name="usuarioPortType">
32    <operation name="getUsuario">
33      <input message="tns:getUsuarioRequest"/>
34      <output message="tns:getUsuarioResponse"/>
35    </operation>
36  </portType>
```

1.3 Descripción del servicio: WSDL

1.3.3 `portType` Cada portType es un grupo de funciones que implementa el servicio web.

Además, en función del tipo de operación de que se trate, contendrá:

Un elemento `input` para indicar funciones que no devuelven valor (su objetivo es sólo enviar un mensaje al servidor).

Un elemento `input` y otro `output`, en este orden, para el caso más habitual: funciones que reciben algún parámetro, se ejecutan, y devuelven un resultado.

```
31  <portType name="usuarioPortType">
32    <operation name="getUsuario">
33      <input message="tns:getUsuarioRequest"/>
34      <output message="tns:getUsuarioResponse"/>
35    </operation>
36  </portType>
```

Normalmente, los elementos `input` y `output` contendrán un atributo `message` para hacer referencia a un mensaje definido anteriormente.

1.3 Descripción del servicio: WSDL

1.3.4 `binding` Define cómo va a transmitirse la información de cada portType.

En el elemento `binding` es dónde debes indicar que el estilo de enlazado en tu documento sea `RPC/encoded`.

Es posible crear documentos WSDL con varios elementos binding pero la mayoría contendrá solo uno.

Para cada una de las funciones (`operation`) del portType que acabamos de crear, se deberá indicar cómo se codifica y transmite la información.

Para el `portType` anterior, podemos crear un elemento `binding` como el siguiente:

```
<binding name="usuarioBinding" type="tns:usuarioPortType">
  <soap:binding
      style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"
  />
  <operation name="getUsuario">
    <soap:operation
    soapAction="http://localhost/DWEES/Unidad6/getUsuario.php?getUsuario"
    />
    <input>
     <soap:body
        use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost/DWEES/Unidad6"
      />
    </input>
    <output>
     <soap:body
        use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://localhost/DWEES/Unidad6"
      />
    </output>
  </operation>
</binding>
```

1.3 Descripción del servicio: WSDL

1.3.4 `binding` Define cómo va a transmitirse la información de cada portType.

Fíjate que el atributo `type` hace referencia al `portType` creado anteriormente.

El siguiente elemento indica el tipo de codificación (RPC) y, mediante una URL correspondiente, el protocolo de transporte a utilizar (HTTP).

```
39  <binding name="usuarioBinding" type="tns:usuarioPortType">
40    <soap:binding
41        style="rpc"
42        transport="http://schemas.xmlsoap.org/soap/http"
43    />
44    <operation name="getUsuario">
45      <soap:operation
46      soapAction="http://localhost/DWEES/Unidad6/getUsuario.php?getUsuario"
47      />
48      <input>
49       <soap:body
50          use="encoded"
51          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
52          namespace="http://localhost/DWEES/Unidad6"
53        />
54      </input>
55      <output>
56       <soap:body
57          use="encoded"
58          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
59          namespace="http://localhost/DWEES/Unidad6"
60        />
61      </output>
62    </operation>
63  </binding>
```

1.3 Descripción del servicio: WSDL

1.3.4 `binding` Define cómo va a transmitirse la información de cada portType.

Obviamente, deberás añadir el correspondiente espacio de nombres al elemento raíz:

```
67
68  <definitions
69      name="WSDLusuario"
70      targetNamespace="http://localhost/DWEES/Unidad6"
71      xmlns:tns="http://localhost/DWEES/Unidad6"
72      xmlns="http://schemas.xmlsoap.org/wsdl/"
73      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
74      xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
75      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
76      …
77  >
```

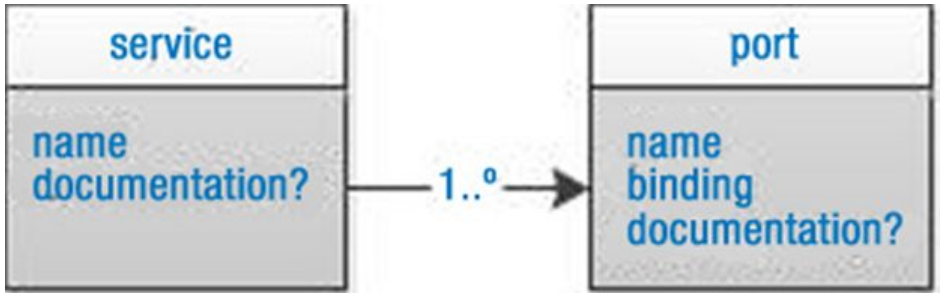
El elemento soap:operation debe contener un atributo `soapAction` con la URL para esa función (`operation`) en particular. Dentro de él habrá un elemento `input` y otro `output`

1.3 Descripción del servicio: WSDL

1.3.5 `service` Contiene una lista de elementos de tipo port.

Normalmente solo encontraremos un elemento `service` en cada documento WSDL

En él, hará referencia al `binding` anterior, utilizando un elemento `port` y se indicará la URL en la que se puede acceder al servicio.



1.3 Descripción del servicio: WSDL

1.3.5 `service` Contiene una lista de elementos de tipo port.

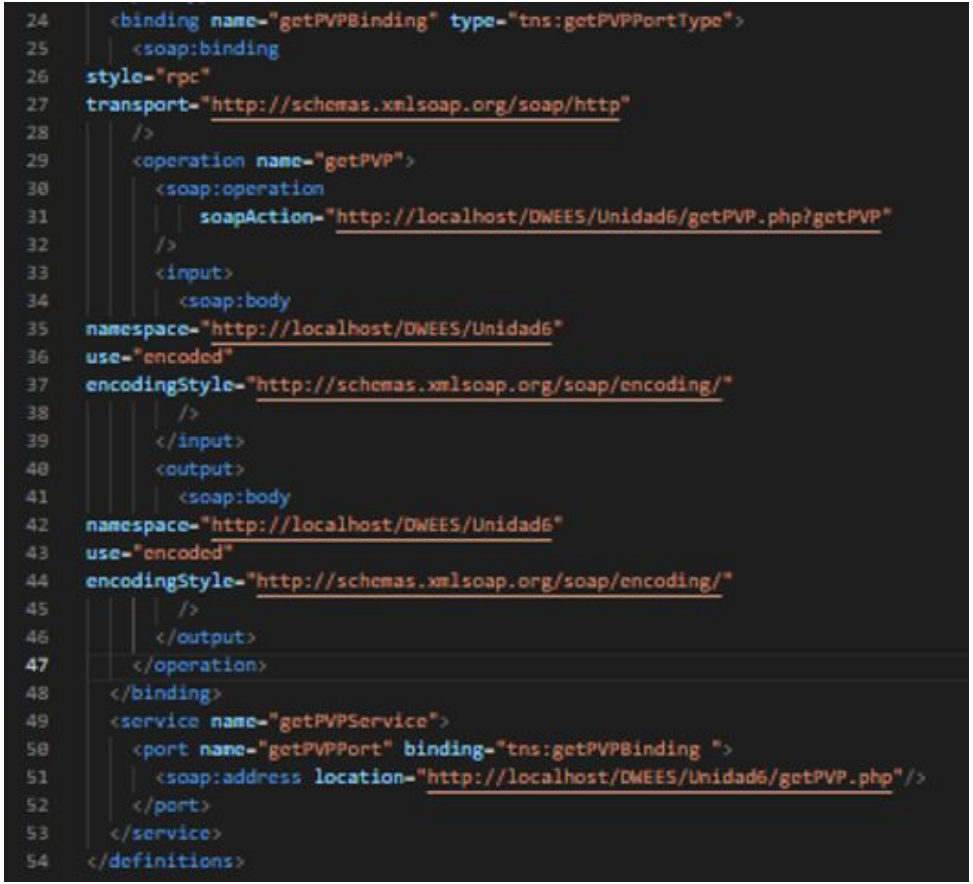
Por ejemplo:

```
<service name="usuario">
  <port name="usuarioPort" binding="tns:usuarioBinding">
    <soap:address
        location="http://localhost/DWEES/Unidad6/getUsuario.php"
    />
  </port>
</service>
```

1.3 Descripción del servicio: WSDL

```xml
<?xml version="1.0" encoding="UTF-8"?>
<definitions
  name="WSDLgetPVP"
  targetNamespace="http://localhost/DWEES/Unidad6"
  xmlns:tns="http://localhost/DWEES/Unidad6"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>
  <message name="getPVPRequest">
    <part name="codigo" type="xsd:string"/>
  </message>
  <message name="getPVPResponse">
    <part name="PVP" element="xsd:float"/>
  </message>
  <portType name="getPVPPortType">
    <operation name="getPVP">
      <input message="tns:getPVPRequest"/>
      <output message="tns:getPVPResponse"/>
    </operation>
  </portType>
```

1.3 Descripción del servicio: WSDL



```
<binding name="getPVPBinding" type="tns:getPVPPortType">
  <soap:binding
style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"
  />
  <operation name="getPVP">
    <soap:operation
      soapAction="http://localhost/DWEES/Unidad6/getPVP.php?getPVP"
    />
    <input>
      <soap:body
namespace="http://localhost/DWEES/Unidad6"
use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
    </input>
    <output>
      <soap:body
namespace="http://localhost/DWEES/Unidad6"
use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
      />
    </output>
  </operation>
</binding>
<service name="getPVPService">
  <port name="getPVPPort" binding="tns:getPVPBinding ">
    <soap:address location="http://localhost/DWEES/Unidad6/getPVP.php"/>
  </port>
</service>
</definitions>
```

2. Extensión PHP SOAP

Gracias a PHP SOAP puedes utilizar y crear de forma sencilla servicios web en tus aplicaciones

Para poder usar la extensión primero hay que comprobar si ya se encuentra disponible (por ejemplo, consultando la salida obtenida por la función `phpinfo`).

Normalmente, la extensión SOAP formará parte de PHP y podrás utilizarla directamente. Si no fuera así, debes instalarla y habilitarla en el fichero php.ini

Las dos funciones principales que vamos a utilizar en nuestras aplicaciones son:

`SoapCliente` (te permite comunicarte con un servicio web) y `SoapServer` (crear tus propios servicios)

soap

Soap Client	enabled
Soap Server	enabled

Directive	Local Value	Master Value
soap.wsdl_cache	1	1
soap.wsdl_cache_dir	/tmp	/tmp
soap.wsdl_cache_enabled	1	1
soap.wsdl_cache_limit	5	5
soap.wsdl_cache_ttl	86400	86400

2.1 Utilización de un servicio web

Vamos a ver cómo crear en PHP una aplicación que se comunique con un servicio web para obtener información. Vamos a verlo a través de un ejemplo.

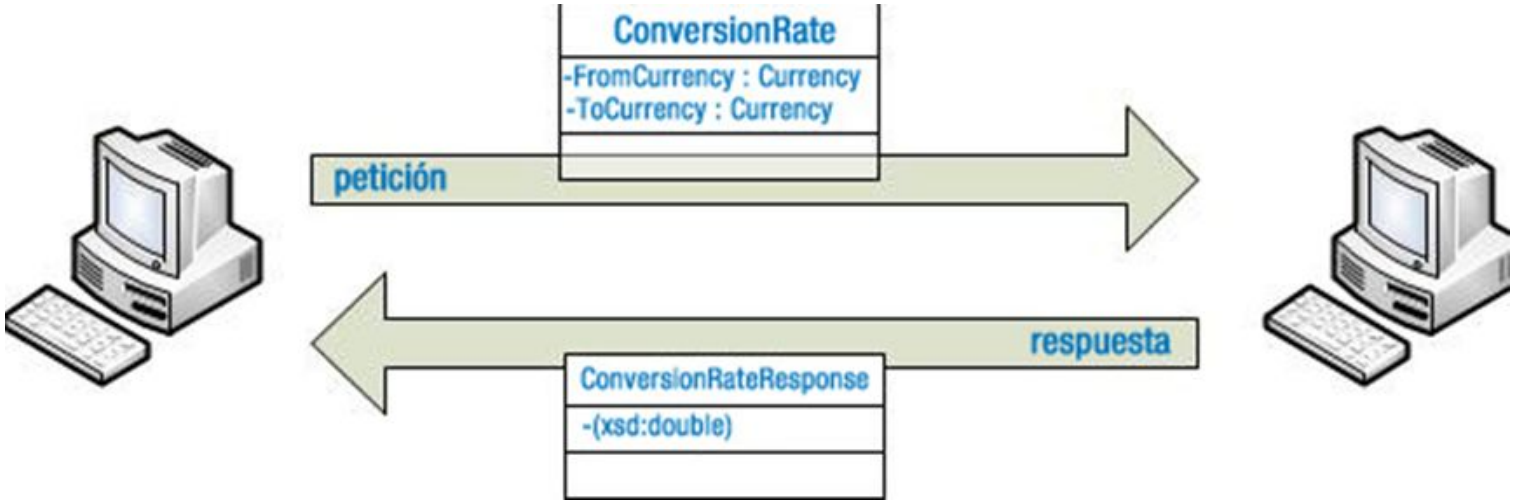
EJEMPLO:

Imagínate que has finalizado la aplicación de tienda web y lleva un tiempo funcionando. Un día la empresa necesita comenzar a vender en el extranjero y quiere dar la posibilidad de mostrar los precios de los productos en dólares.

Lo primero que necesitas saber es la tasa de conversión entre euros y dólares. Y para tener esa información, decides usar un servicio web que te la ofrezca en tiempo real.

2.1 Utilización de un servicio web

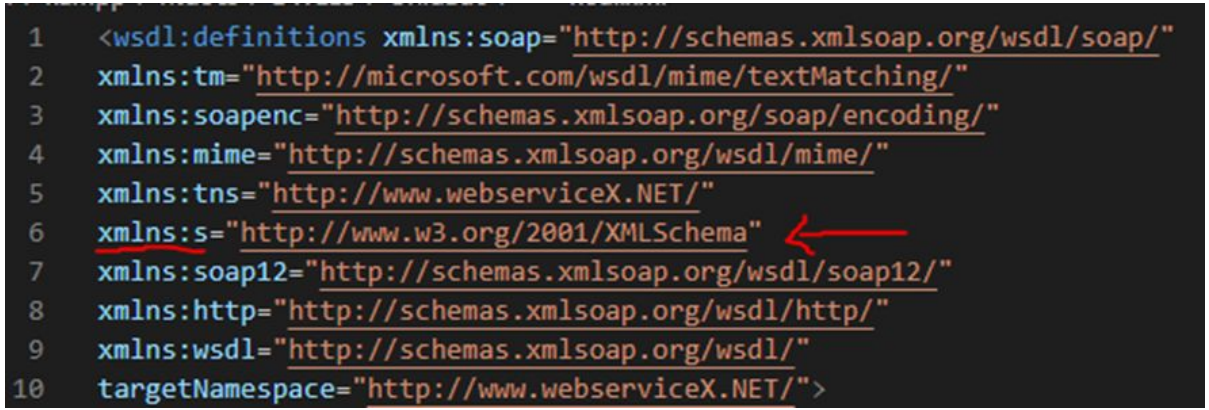
En el documento WSDL del Classroom podemos observar:



2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- El alias del espacio de nombres correspondiente al XML Schema que utiliza el documento es s.



2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- El tipo `Currency` debe ser un string de 3 caracteres de los que se listan en el documento, correspondiente a las siglas de una divisa. (líneas 22-176 del documento)

```
22    <s:simpleType name="Currency">
23    <s:restriction base="s:string">
24    <s:enumeration value="AFA"/>
25    <s:enumeration value="ALL"/>
```

2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- El tipo `ConversionRate` es una secuencia de dos elementos `Currency`

```
14   <s:element name="ConversionRate">
15   <s:complexType>
16   <s:sequence>
17   <s:element minOccurs="1" maxOccurs="1" name="FromCurrency" type="tns:Currency"/>
18   <s:element minOccurs="1" maxOccurs="1" name="ToCurrency" type="tns:Currency"/>
19   </s:sequence>
20   </s:complexType>
21   </s:element>
```

2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

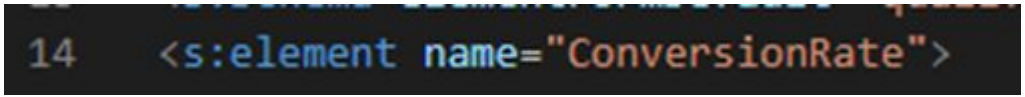
- El tipo `ConversionRateResponse` es un double

```
177   <s:element name="ConversionRateResponse">
178   <s:complexType>
179   <s:sequence>
180   <s:element minOccurs="1" maxOccurs="1" name="ConversionRateResult" type="s:double"/>
181   </s:sequence>
182   </s:complexType>
183   </s:element>
```

2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- El estilo de enlazado es document/literal (recuerda que nosotros vimos el RPC/encoded solamente) por lo que los elementos tipo message tienen un formato distinto. Sin embargo, con base a su contenido (fíjate en los elementos que terminan en Soap) se puede deducir también que:
    - El nombre de la función a la que debes llamar es `ConversionRate`.



    - Como parámetro de entrada le tienes que pasar un elemento de tipo `ConversionRate` (dos string) y devolverá un elemento `ConversionRateResponse` (un double)

2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- Como parámetro de entrada le tienes que pasar un elemento de tipo `ConversionRate` (dos string) y devolverá un elemento `ConversionRateResponse` (un double)

```
187   <wsdl:message name="ConversionRateSoapIn">
188   <wsdl:part name="parameters" element="tns:ConversionRate"/>
189   </wsdl:message>
190   <wsdl:message name="ConversionRateSoapOut">
191   <wsdl:part name="parameters" element="tns:ConversionRateResponse"/>
192   </wsdl:message>
```

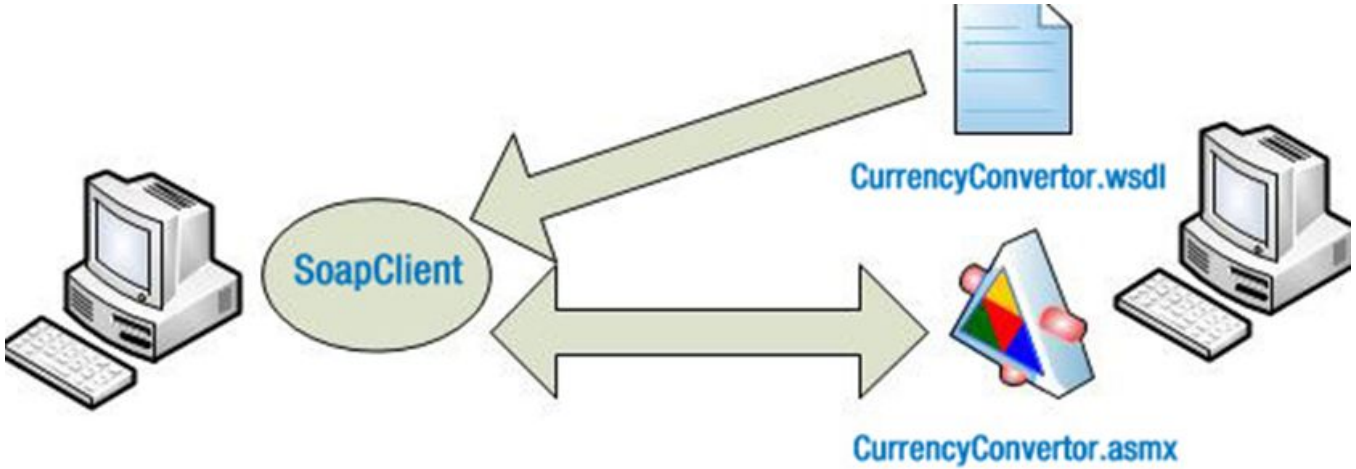
```
207   <wsdl:portType name="CurrencyConvertorSoap">
208   <wsdl:operation name="ConversionRate">
209   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
210   <br><b>Get conversion rate from one currency to another currency
211   <b><br><p><b><font color='#000080' size='1' face='Verdana'>
212   <u>Differenct currency Code and Names around the world</u>
213   </font></b></p><blockquote><p>
214   <font face='Verdana' size='1'>AFA-Afghanistan Afghani<br>ALL-Albanian
215   <wsdl:input message="tns:ConversionRateSoapIn"/>
216   <wsdl:output message="tns:ConversionRateSoapOut"/>
217   </wsdl:operation>
218   </wsdl:portType>
```

2.1 Utilización de un servicio web

En el documento WSDL del Classroom podemos observar:

- La URL para acceder al servicio es http://www.webservicex.net/CurrencyConvertor.asmx

```
281    <wsdl:service name="CurrencyConvertor">
282    <wsdl:port name="CurrencyConvertorSoap" binding="tns:CurrencyConvertorSoap">
283    <soap:address location="http://www.webservicex.net/CurrencyConvertor.asmx"/>
284    </wsdl:port>
```



2.1 Utilización de un servicio web

Con la información anterior para utilizar el servicio desde PHP creas un nuevo objeto de la clase `SoapCliente`. Como el servicio tiene un documento WSDL asociado, en el constructor le indicas dónde se encuentra:

```
2   $cliente = new SoapClient(
3     "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL"
4   );
```

Y para realizar la llamada a la función `ConversionRate`, incluyes los parámetros en un array:

```
1   <?php
2   $cliente = new SoapClient(
3     "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL"
4   );
5   $parametros = array("FromCurrency" => "EUR", "ToCurrency" => "USD");
6   $tasa = $cliente->ConversionRate($parametros);
7
```

2.1 Utilización de un servicio web

La llamada devuelve un objeto de una clase predefinida en PHP llamada `StdClass`. Para utilizar el valor devuelto, puedes hacer:

```php
<?php
$cliente = new SoapClient(
  "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL"
);
$parametros = array("FromCurrency" => "EUR", "ToCurrency" => "USD");
$tasa = $cliente->ConversionRate($parametros);

print("Resultado: ".$tasa->ConversionRateResult);

?>
```

2.1 Utilización de un servicio web

La llamada devuelve un objeto de una clase predefinida en PHP llamada `StdClass`. Para utilizar el valor devuelto, puedes hacer:

```php
<?php
$cliente = new SoapClient(
  "http://www.webservicex.net/CurrencyConvertor.asmx?WSDL"
);
$parametros = array("FromCurrency" => "EUR", "ToCurrency" => "USD");
$tasa = $cliente->ConversionRate($parametros);

print("Resultado: ".$tasa->ConversionRateResult);

?>
```

El constructor de la clase `SoapCliente` puede usarse de dos formas: indicando un documento WSDL, como en el caso anterior, o sin indicarlo. En el primer caso, la extensión SOAP examina la definición del servicio y establece las opciones adecuadas para la comunicación, con lo cual el código necesario para utilizar un servicio es bastante simple.

En el segundo caso, si no indicas en el constructor un documento WSDL, el primer parámetro debe ser `null`, y las opciones para comunicarse con el servicio las tendrás que establecer en un array que se pasa como segundo parámetro.

2.1 Utilización de un servicio web

Si estás usando un documento WSDL para acceder al servicio web, la clase `SoapCliente` implementa dos métodos que muestran parte de la información que contiene; concretamente, los tipos de datos definidos por el servicio, y las funciones que ofrece.

Para conocer esta información, una vez creado el objeto, debes utilizar los métodos `__getTypes` y `__getFunctions` respectivamente.

```
11  $cliente = new CurrencyConvertor();
12  print_r($cliente->__getTypes());
13  print_r($cliente->__getFunctions());
14
```

2.1 Utilización de un servicio web

El resultado obtenido es:

```
Array (
  [0] => struct ConversionRate {
            Currency FromCurrency;
            Currency ToCurrency;
  }
  [1] => string Currency
  [2] => struct ConversionRateResponse {
            double ConversionRateResult;
  }
)
Array (
  [0] => ConversionRateResponse ConversionRate
            (ConversionRate $parameters)
  [1] => ConversionRateResponse ConversionRate
            (ConversionRate $parameters)
)
```

Donde la función ConversionRate aparece duplicada, dado que el servicio web ofrece dos versiones de la misma: una para SOAP 1.1 y otra para SOAP 1.2

2.2 Creación de un servicio web

En PHP SOAP, para crear un servicio web (`servicio.php`), debes utilizar la clase `SoapServer`. Veamos un ejemplo:

El código anterior crea un servicio web con dos funciones: suma y resta. Cada función recibe 2 parámetros y devuelve un valor. Además:

- `addFunction` se encarga de publicar en el servicio la función que se le pase como parámetro.
- `handle` es el encargado de procesar las peticiones, recogiendo los datos que se reciban utilizando POST por HTTP

```php
<?php

function suma($a,$b){ return $a+$b; }
function resta($a,$b){ return $a-$b; }

$uri="http://localhost/DWEES/Unidad6";
$server = new SoapServer(null,array('uri'=>$uri));
$server->addFunction("suma");
$server->addFunction("resta");
$server->handle();

?>
```

2.2 Creación de un servicio web

Para consumir este servicio, necesitas escribir el siguiente código:

```php
<?php

$url="http://localhost/DWEES/Unidad6/servicio.php";
$uri="http://localhost/DWEES/Unidad6";
$cliente = new SoapClient(null,array('location'=>$url,'uri'=>$uri));

$suma = $cliente->suma(2,3);
$resta = $cliente->resta(2,3);
print("La suma es ".$suma);
print("<br />La resta es ".$resta);

?>
```

El documento que hemos creado no incluye un documento WSDL para describir sus funciones.

Sabemos que existen los métodos suma y resta, y los parámetros que debes utilizar con ellos, porque conoces el código interno del servicio. Un usuario que no tuviera esta información, **no sabría cómo consumir el servicio.**

2.2 Creación de un servicio web

Al igual que sucedía con `SoapClient` al programar un cliente, cuando utilizas `SoapServer` puedes crear un servicio **sin** documento WSDL asociado (como en el caso anterior), o indicar el documento WSDL correspondiente al servicio; pero antes deberás haberlo creado.

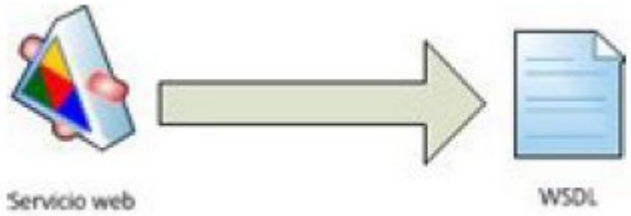
El primer parámetro del constructor indica la ubicación del WSDL correspondiente. El segundo parámetro es una colección de opciones de configuración del servicio. Si existe el primer parámetro, ya no hace falta más información. PHP5 SOAP utiliza la información del documento WSDL para ejecutar el servicio. Si, como en el ejemplo, no existe WSDL, deberás indicar en el segundo parámetro al menos la opción uri, con el espacio de nombres destino del servicio.

2.2 Creación de un servicio web

Creación de un documento WSDL

Para crear un documento WSDL de descripción del servicio, tendrás que seguir los pasos vistos anteriormente.

NOTA: Al programar un servicio web, es importante cambiar en el fichero `php.ini` la directiva `soap.wsdl_cache_enabled` a 0. En caso contrario, con su valor por defecto (1), los cambios que realices en los ficheros WSDL no tendrán efecto de forma inmediata.



2.2 Creación de un servicio web

Creación de un documento WSDL

El elemento raíz del documento será:

```
<definitions
  name="WSDLCalcula"
  targetNamespace="http://localhost/DWEES/Unidad6"
  xmlns:tns="http://localhost/DWEES/Unidad6"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soap-enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

En este caso no necesitas definir ningún tipo nuevo, por lo que no tendrás sección `types`.
Los elementos `message` necesarios para la suma (los de la resta son similares) serán:

```
12    <message name="sumaRequest">
13      <part name="a" type="xsd:float"/>
14      <part name="b" type="xsd:float"/>
15    </message>
16    <message name="sumaResponse">
17      <part name="resultado" type="xsd:float"/>
18    </message>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

El `portType` (no se incluye el `operation` de la resta, que es equivalente):

```
20    <portType name="CalculaPortType">
21      <operation name="suma">
22        <input message="tns:sumaRequest"/>
23        <output message="tns:sumaResponse"/>
24      </operation>
25    </portType>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

El `portType` (no se incluye el `operation` de la resta, que es equivalente):

```
20    <portType name="CalculaPortType">
21      <operation name="suma">
22        <input message="tns:sumaRequest"/>
23        <output message="tns:sumaResponse"/>
24      </operation>
25    </portType>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

Suponiendo que el servicio web está en el fichero `calcula.php`, la parte de la operación suma correspondiente al binding será:

```
27  <binding name="CalculaBinding" type="tns:CalculaPortType">
28    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
29    <operation name="suma">
30      <soap:operation
31        soapAction="http://localhost/DWEES/Unidad6/calcula.php?method=suma"
32      />
33      <input>
34       <soap:body
35           namespace="http://localhost/DWEES/Unidad6"
36           use="encoded"
37           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
38        />
39      </input>
40      <output>
41       <soap:body
42           namespace="http://localhost/DWEES/Unidad6"
43           use="encoded"
44           encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
45        />
46      </output>
47    </operation>
48  </binding>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

Y para finalizar, el elemento `service`:

```
50  <service name="Calcula">
51    <port name="CalculaPort" binding="tns:CalculaBinding">
52      <soap:address location="http://localhost/DWEES/Unidad6/calcula.php"/>
53    </port>
54  </service>
```

2.2 Creación de un servicio web

Creación de un documento WSDL

En vez de utilizar funciones para la lógica interna del servicio web, como la suma y la resta del ejemplo anterior, es aconsejable definir una clase que implemente los métodos que queramos publicar en el servicio

```php
class Calcula {
    public function suma($a, $b){ return $a+$b; }
    public function resta($a, $b){ return $a-$b; }
}
```

Al hacerlo de esta forma, en lugar de añadir una a una las funciones, podemos añadir la clase completa al servidor utilizando el método `setClass` de `SoapServer`.

```php
<?php

require_once('Calcula.php');

$server = new SoapServer(null, array('uri'=>''));
$server->setClass('Calcula');
$server->handle();
```

2.2 Creación de un servicio web

Aunque como ya hemos dicho, PHP5 SOAP no genera el documento WSDL de forma automática para los servicios que crees, existen algunos mecanismos que nos permiten generarlo, aunque siempre es aconsejable revisar los resultados obtenidos antes de publicarlos.
Una de las formas más sencillas es utilizar la librería **`WSDLDocument.`**

https://code.google.com/p/wsdldocument/

Esta librería revisa los comentarios que hayas añadido al código de la clase que quieras publicar (debe ser una clase, no funciones aisladas) y genera como salida el documento WSDL correspondiente.

Para que funcione correctamente, es necesario que los comentarios de las clases sigan un formato específico: el mismo que utiliza la herramienta de documentación **`PHPDocumentor`**

2.2 Creación de un servicio web

PHPDocumentor es una herramienta de código libre para generación automática de documentación, similar a Javadoc (para el lenguaje Java).

Si comentamos el código de nuestras aplicaciones siguiendo unas normas, `PHPDocumentor` es capaz de generar, a partir de los comentarios que introduzcamos en el código mientras programamos, documentación en diversos formatos (HTML, PDF, XML).

http://www.phpdoc.org/

Los comentarios se deben ir introduciendo en el código distribuidos en bloques, y utilizando ciertas marcas específicas como `@param` para indicar un parámetro y `@return` para indicar el valor devuelto por una función.
Por ejemplo, la clase `Calcula` comentada según estas normas quedaría:

```php
class Calcula {
    /**
     * Suma dos números y devuelve el resultado
     *
     *  @param float $a
     *  @param float $b
     *  @return float
     */
    public function suma($a, $b){
        return $a+$b;
    }

    /**
     * Resta dos números y devuelve el resultado *
     * @param float $a
     * @param float $b
     * @return float
     */
    public function resta($a, $b){
        return $a-$b;
    }
}
```

2.2 Creación de un servicio web

No es necesario instalar `WSDLDocument`, basta con descargarlo en tu equipo y descomprimir el archivo. Su contenido es un único fichero con código en PHP.

Para generar el documento WSDL a partir de la clase `Calcula` anterior, debes crear un nuevo fichero con el siguiente código:

```php
require_once("Calcula.php");
// Ruta a WSDLDocument require_once("WSDLDocument.php");

$wsdl = new WSDLDocument( "Calcula",
"http://localhost/dwes/ut6/servicio.php",
"http://localhost/dwes/ut6" );

echo $wsdl->saveXml();

?>
```

Es decir, crear un nuevo objeto de la clase WSDLDocument, e indicar como parámetros:

- El nombre de la clase que gestionará las peticiones al servicio.
- La URL en que se ofrece el servicio.
- El espacio de nombres destino.

El método `saveXML` obtiene como salida el documento WSDL de descripción del servicio

2.2 Creación de un servicio web

Cuando esté listo, publícalo con tu servicio. Para ello, copia el fichero obtenido en una ruta accesible vía web (por ejemplo, en la misma ruta en la que se encuentre la clase que gestiona el servicio), e indica la URL en que se encuentra cuando instancies la clase `SoapServer`.

```php
$server = new SoapServer("http://localhost/dwes/ut6/calcula.wsdl");
```

Si añades a la URL del servicio el parámetro POST wsdl, verás el fichero de descripción del servicio.
En nuestro caso la URL sería `http://localhost/dwes/ut6/calcula.php?wsdl`.

En conclusión

Podríamos decir que **un servicio web es un mecanismo mediante el cual dos aplicaciones web se pueden comunicar y se pueden transmitir datos**. Normalmente una de las aplicaciones es la proveedora de servicios y la otra aplicación es la consumidora de servicios.

- Los servicios web se pueden implementar de 2 formas: SOAP y REST
  SOAP utiliza exclusivamente el formato XML
  REST puede trabajar tanto con XML como con JSON (Lo veremos en U.8)

Ejemplos:

SOAP: servicios financieros, pasarelas de pago, servicios de telecomunicaciones

REST: en redes sociales, proveer servicios a móviles. Google, Facebook, Twitter, LinkedIn, Tripadvisor son ejemplos de empresas.