Web services. SOAP y WSDL

SOAP

Protocolo que permite desarrollar software en cualquier lenguaje y este ser interpretado por cualquier otro sistema o aplicación a través de una red mediante XML. SOAP establece reglas para enviar y recibir Remote Procedure Calls (RPC), como las de los mensajes HTTP request y HTTP response, por lo que no está ligado a ningún sistema operativo o lenguaje de programación.

Los servicios web SOAP se describen utilizando WSDL (Web Services Description Language), que es un lenguaje basado en XML para describir la interfaz de un servicio web.

Estructura mensaje SOAP

PETICIÓN

Sintaxis

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <!-- Cabecera (opcional) -->
    </soap:Header>
    <soap:Body>
        <!-- Cuerpo de la petición -->
    </soap:Body>
</soap:Envelope>
```

Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
    soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

    <soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </soap:Body>
</soap:Envelope>
```

- `<?xml version="1.0" encoding="UTF-8"?>`: Esta declaración indica la versión XML utilizada (1.0) y la codificación de caracteres (UTF-8) del documento.

- `<soap:Envelope>`: Es el elemento raíz de la estructura SOAP. Define el sobre que contiene toda la información de la solicitud o respuesta SOAP.
    - `xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"`: El atributo xmlns:soap establece el espacio de nombres asociado con el prefijo "soap" y define la ubicación de la especificación SOAP utilizada.
    - `soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"`: Este atributo opcional especifica el estilo de codificación utilizado en la respuesta SOAP. En este caso, se utiliza el estilo de codificación "http://www.w3.org/2003/05/soap-encoding".
- `<soap:Body>`: Contiene el cuerpo de la solicitud o respuesta SOAP. Es el contenedor principal para los datos específicos de la operación SOAP.
    - `xmlns:m="http://www.example.org/stock"`: El atributo xmlns:m establece un espacio de nombres personalizado asociado con el prefijo "m". En este caso, se utiliza el espacio de nombres "http://www.example.org/stock".
- `<m:GetStockPrice>`: Representa la operación específica que se está solicitando en el servicio web. En este ejemplo, se solicita el precio de una acción.
- `<m:StockName>IBM</m:StockName>`: Es el elemento que contiene el nombre de la acción para la cual se solicita el precio. En este caso, se utiliza "IBM" como ejemplo.

Ejemplo con SOAP:FAULT

Se utiliza en el cuerpo de la respuesta SOAP para transmitir información sobre un error específico que ha ocurrido. Es opcional y se incluye cuando se necesita informar sobre un fallo en la respuesta SOAP.

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
    soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

    <soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPrice>
            <m:StockName>IBM</m:StockName>
        </m:GetStockPrice>
    </soap:Body>

    <soap:Fault>
        <soap:Code>
            <soap:Value>soap:Receiver</soap:Value>
        </soap:Code>
        <soap:Reason>
            <soap:Text xml:lang="en">Invalid StockName</soap:Text>
        </soap:Reason>
        <soap:Detail>
            <m:ErrorDetails>Error occurred while retrieving stock
price</m:ErrorDetails>
        </soap:Detail>
    </soap:Fault>

</soap:Envelope>
```

- `<soap:Fault>`: Representa un mensaje de error en una respuesta SOAP. Contiene información detallada sobre el error.
    - `<soap:Code>`: Define el código de error asociado al mensaje. En este caso, se utiliza soap:Value para establecer el valor del código como "soap:Receiver".
    - `<soap:Reason>`: Proporciona una explicación textual del motivo del error. soap:Text se utiliza para especificar el texto del motivo en el idioma indicado (xml:lang). En este ejemplo, el texto del motivo es "Invalid StockName".
    - `<soap:Detail>`: Proporciona detalles adicionales sobre el error. Aquí se puede agregar información específica del servicio web o del contexto del error. En este caso, se utiliza <m:ErrorDetails> para indicar que se produjo un error al recuperar el precio de la acción.

RESPUESTA

Sintaxis

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
    soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

    <soap:Header>
        <!-- Aquí se pueden incluir elementos adicionales en el encabezado -->
    </soap:Header>

    <soap:Body>
        <!-- Aquí se incluye el contenido de la respuesta -->
    </soap:Body>

</soap:Envelope>
```

Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
    soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

    <soap:Body xmlns:m="http://www.example.org/stock">
        <m:GetStockPriceResponse>
            <m:Price>34.5</m:Price>
        </m:GetStockPriceResponse>
    </soap:Body>

</soap:Envelope>
```

- `<?xml version="1.0" encoding="UTF-8"?>`: Esta declaración indica la versión XML utilizada (1.0) y la codificación de caracteres (UTF-8) del documento.
- `<soap:Envelope>`: Es el elemento raíz de la estructura SOAP. Define el sobre que contiene toda la información de la solicitud o respuesta SOAP.
    - `xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"`: El atributo xmlns:soap establece el espacio de nombres asociado con el prefijo "soap" y define la ubicación de la especificación SOAP utilizada.
    - `soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"`: Este atributo opcional especifica el estilo de codificación utilizado en la respuesta SOAP. En este caso, se utiliza el estilo de codificación "http://www.w3.org/2003/05/soap-encoding".
- `<soap:Body>`: Contiene el cuerpo de la solicitud o respuesta SOAP. Es el contenedor principal para los datos específicos de la operación SOAP.
    - `xmlns:m="http://www.example.org/stock"`: El atributo xmlns:m establece un espacio de nombres personalizado asociado con el prefijo "m". En este caso, se utiliza el espacio de nombres "http://www.example.org/stock".
- `<m:GetStockPriceResponse>`: Representa la respuesta a la solicitud GetStockPrice.
    - `<m:Price>34.5</m:Price>`: que contiene el valor del precio del stock, en este caso, 34.5.

Elementos necesarios

- **Envelope (Envoltorio)** : Es el elemento raíz del mensaje SOAP. Contiene todos los demás elementos y define el espacio de nombres SOAP utilizado en el mensaje.
- **Header (Cabecera)** : Es un elemento opcional que contiene información adicional relacionada con la transacción SOAP. Puede incluir información de autenticación, seguridad u otros datos relevantes.
- **Body (Cuerpo)** : Es el elemento principal que contiene el mensaje RPC. Aquí se incluyen los detalles específicos de la operación que se está realizando.
- **Fault (Error)** : Es un elemento opcional que se utiliza para representar errores o excepciones en caso de que ocurra algún problema durante la ejecución de la operación. En caso de que sea hijo de body representa un error específico relacionado con la operación dentro del body.

Bibliotecas y extensiones SOAP

son herramientas que simplifican y agilizan el desarrollo de aplicaciones web basadas en SOAP, proporcionando funciones y clases predefinidas para trabajar con los aspectos específicos de este protocolo de comunicación. Facilitan la creación de clientes y servidores SOAP, la manipulación de mensajes y la interacción con servicios web SOAP.

- Abstracción del protocolo: Proporcionan una capa de abstracción sobre los detalles técnicos del protocolo SOAP, permitiendo a los desarrolladores trabajar con objetos y métodos que representan los elementos SOAP subyacentes, como el encabezado, el cuerpo y los elementos de datos.
- Generación de clientes y servidores: Permiten generar automáticamente el código necesario para crear clientes y servidores SOAP.
- Manipulación de mensajes SOAP: Proporcionan funciones y métodos para manipular y modificar mensajes SOAP,
- Soporte para WSDL: Permite generar automáticamente el código cliente o servidor a partir de un archivo WSDL, lo que simplifica el proceso de desarrollo y garantiza la coherencia entre el cliente y el servidor SOAP.

Algunas bibliotecas o extensiones:

- **PHP SOAP**: Es una extensión incorporada en PHP que permite la comunicación y el consumo de servicios web SOAP. Proporciona funciones para crear clientes y servidores SOAP, enviar y recibir mensajes SOAP, y manejar la serialización y deserialización de datos SOAP.
- **NUSOAP**: Es una biblioteca PHP que ofrece una implementación completa de SOAP tanto para crear clientes como para desarrollar servicios web SOAP. Es fácil de usar y proporciona una interfaz orientada a objetos para trabajar con los elementos SOAP.
- **PEAR::SOAP**: Es una biblioteca basada en PEAR (PHP Extension and Application Repository) que permite crear y consumir servicios web SOAP. Proporciona una interfaz simple para trabajar con SOAP y facilita la creación de clientes y servidores SOAP.

PHP SOAP

En PHP SOAP, para crear un servicio web, hay que utilizar la clase SoapServer.

```
 class SoapServer {
/* Métodos */
public addFunction(mixed $functions): void
public addSoapHeader(SoapHeader $object): void
public __construct(mixed $wsdl, array $options = ?)
public fault(
    string $code,
    string $string,
    string $actor = ?,
    string $details = ?,
    string $name = ?
): void
public getFunctions(): array
public handle(string $soap_request = ?): void
public setClass(string $class_name, mixed $... = ?): void
public setObject(object $object): void
public setPersistence(int $mode): void
}
```

- SoapServer::addFunction — Añade una o más funciones al controlador de peticiones SOAP
- SoapServer::addSoapHeader — Añade un encabezado SOAP a la respuesta
- SoapServer::__construct — Constructor de SoapServer
- SoapServer::fault — SoapServer indica que ocurrió un fallo
- SoapServer::getFunctions — Devuelve una lista de las funciones definidas
- SoapServer::handle — Controla la petición SOAP
- SoapServer::setClass — Define la clase que controla las peticiones SOAP
- SoapServer::setObject — Define el objecto que será usado para controlar las peticiones SOAP
- SoapServer::setPersistence — Establece el modo de persistencia de SoapServer

Al igual que sucedía con SoapClient al programar un cliente, cuando se utiliza SoapServer se puede crear un servicio sin documento WSDL asociado (como en el caso anterior), o indicar el documento WSDL correspondiente al servicio.

PHPDOCUMENTOR

Documentación oficial

Es una herramienta de código libre que se utiliza para la generación automática de documentación en PHP. Se asemeja a Javadoc, que se utiliza en el lenguaje de programación Java. PHPDocumentor analiza los comentarios agregados al código fuente de una clase y genera documentación en varios formatos, como HTML, PDF y XML.

Para que PHPDocumentor funcione correctamente, es necesario seguir un formato específico en los comentarios de las clases. Los comentarios deben estar distribuidos en bloques y utilizar marcas específicas, como "@param" para indicar un parámetro y "@return" para indicar el valor devuelto por una función.

> **NOTA** existe una extensión para VSCode que ayuda a generar los comentarios de manera más eficiente. PHPDoc

```php
/**
 * Suma dos números enteros.
 *
 * @param int $a El primer número.
 * @param int $b El segundo número.
 * @return int La suma de los dos números.
 */
function sumar($a, $b) {
    return $a + $b;
}
```

En este ejemplo, el comentario PHPDoc describe brevemente la función y utiliza las marcas "@param" para indicar los parámetros de entrada y "@return" para indicar el valor devuelto por la función.

WSDL

Imagina que has creado una aplicación o función especial que puede ser útil para otros desarrolladores. ¡Eso es genial! Pero ahora necesitas decirles cómo utilizarla correctamente, ¿verdad?

Aquí es donde entra en juego el WSDL (Web Services Description Language). Es como un lenguaje especial basado en XML que nos ayuda a crear un documento de descripción de nuestro servicio web. Este documento es una guía detallada que explica cómo interactuar con nuestro servicio, como un manual de instrucciones.

Una vez que tengamos nuestro documento WSDL listo, lo compartiremos con los posibles usuarios de nuestro servicio. Lo colocaremos en algún lugar accesible y generalmente se puede acceder agregando "?wsdl" a la URL del servicio. Así, los programadores interesados pueden obtener toda la información necesaria para utilizar nuestro servicio de manera correcta y eficiente.

Por cierto, el documento WSDL tiene un espacio de nombres especial llamado http://schemas.xmlsoap.org/wsdl. Es como una dirección única para identificar nuestro documento WSDL en el vasto mundo de los servicios web. Pero también podemos usar otros espacios de nombres para aprovechar características y estándares adicionales.

Estructura WSDL

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  targetNamespace="http://example.com/namespace"
>

  <types>
    <!-- Definición de tipos de datos XML o esquemas XSD -->
  </types>

  <message name="NombreDelMensaje">
    <!-- Definición de partes del mensaje -->
  </message>

  <portType name="NombreDelPortType">
    <!-- Definición de operaciones -->
  </portType>

  <binding name="NombreDelBinding" type="tns:NombreDelPortType">
    <!-- Definición de protocolo de transporte (por ejemplo, SOAP) y detalles relacionados -->
  </binding>

  <service name="NombreDelServicio">
    <port name="NombreDelPuerto" binding="tns:NombreDelBinding">
      <!-- Definición de la dirección del servicio -->
    </port>
  </service>

</definitions>
```

- **Definición** : Es el elemento principal de todos los documentos WSDL. Define el nombre del servicio web, declara múltiples espacios de nombres utilizados en el resto del documento y contiene todos los elementos de servicio descritos aquí.
- **Tipos de datos** : Los tipos de datos que se utilizarán en los mensajes se definen en forma de esquemas XML
- **Mensaje** : Es una definición abstracta de los datos, presentados como un mensaje completo o como argumentos que se asignarán a una invocación de método.
- **Operación** : Es la definición abstracta de la operación para un mensaje, como nombrar un método, una cola de mensajes o un proceso empresarial, que aceptará y procesará el mensaje.
- **Tipo de puerto** : Es un conjunto abstracto de operaciones asignadas a uno o más puntos finales, que define la colección de operaciones para una unión; la colección de operaciones, al ser abstracta, se puede asignar a múltiples transportes a través de varias uniones.
- **Unión** : Es un conjunto abstracto de operaciones asignadas a uno o más puntos finales, que define la colección de operaciones para una unión; la colección de operaciones, al ser abstracta, se puede asignar a múltiples transportes a través de varias uniones.
- **Puerto** : Es una combinación de una unión y una dirección de red, que proporciona la dirección de destino de la comunicación del servicio.
- **Servicio** : Es una colección de puntos finales relacionados que abarcan las definiciones del servicio en el archivo; los servicios asignan la unión al puerto e incluyen cualquier definición de extensibilidad.

WSDL tipos de datos

Etiqueta definitions

La etiqueta "definitions" se utiliza en WSDL para envolver y definir el conjunto completo de elementos que conforman la descripción de un servicio web. Proporciona un contenedor global para todos los elementos relacionados con el servicio, como los tipos de datos, mensajes, operaciones, etc.

```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
</definitions>
```

- name

Este atributo define el nombre del servicio, en este caso, "HelloService".

- targetNamespace

Este atributo especifica el espacio de nombres al que pertenece el documento WSDL. Proporciona una identificación única para el servicio web. En este ejemplo, el espacio de nombres es "http://www.examples.com/wsdl/HelloService.wsdl".

- xmlns

Este espacio de nombres define el espacio de nombres predeterminado para los elementos en el documento WSDL. En este caso, "http://schemas.xmlsoap.org/wsdl/" se utiliza como espacio de nombres predeterminado.

- xmlns:soap (opcional)

Este espacio de nombres está asociado con el protocolo SOAP (Simple Object Access Protocol) y se utiliza para definir el formato de mensajes SOAP en el servicio web.

- xmlns:tns (opcional)

Este espacio de nombres se utiliza para especificar el espacio de nombres del servicio web en sí. En este caso, el espacio de nombres es "http://www.examples.com/wsdl/HelloService.wsdl".

- xmlns:xsd (opcional)

Este espacio de nombres se utiliza para referenciar el lenguaje XML Schema (XSD), que se utiliza para definir los tipos de datos en el documento WSDL.

Etiqueta types

La etiqueta "types" en WSDL se utiliza para definir y agrupar los tipos de datos utilizados en la descripción de un servicio web. Proporciona un lugar centralizado donde se pueden definir los tipos de datos complejos, como clases o estructuras, que se utilizan en los mensajes de entrada y salida de las operaciones del servicio.

Estructura usando clases

```
<types>
    <xsd:schema targetNamespace="http://ejercicios.soap.es/usuarios/">
        <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>
```

- xsd:schema targetNamespace="http://ejercicios.soap.es/usuarios/"

Esta etiqueta define un esquema XML utilizando el lenguaje XML Schema (XSD). El atributo targetNamespace especifica el espacio de nombres objetivo del esquema.

- xsd:complexType name="direccion"

Define un tipo complejo llamado "direccion". Un tipo complejo en XML Schema se utiliza para definir estructuras de datos con múltiples elementos.

- xsd:all

Indica que los elementos dentro de xsd:complexType deben aparecer todos en cualquier orden.Todos los elementos enumerados dentro de xsd:all son opcionales y no tienen una secuencia específica. Se garantiza que cada elemento se puede repetir como máximo una vez. Esta construcción es adecuada cuando no hay restricciones de orden en los elementos y todos los elementos son opcionales.

```
<types>
    <xsd:schema targetNamespace="http://ejercicios.soap.es/usuarios/">
        <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:all>
        </xsd:complexType>
    </xsd:schema>
</types>
```

- xsd:sequence

Define que los elementos deben aparecer en un orden secuencial específico dentro de un tipo complejo. Los elementos enumerados dentro de xsd:sequence deben seguir el orden especificado, y pueden ser obligatorios u opcionales. Cada elemento se puede repetir cualquier número de veces. Esta construcción es adecuada cuando el orden de los elementos es importante y deben aparecer en una secuencia particular.

- xsd:choice

Indica que solo uno de los elementos enumerados dentro de xsd:choice puede estar presente en un tipo complejo. Los elementos dentro de xsd:choice son mutuamente excluyentes y solo uno de ellos puede aparecer en la instancia del XML. Esta construcción es útil cuando solo se permite una opción específica entre varios elementos posibles.

- xsd:element name="ciudad" type="xsd:string"

Define un elemento llamado "ciudad" de tipo "xsd:string" dentro del tipo complejo "direccion". Esto significa que "ciudad" es una cadena de texto.

- xsd:complexType name="usuario"

Define otro tipo complejo llamado "usuario".

- xsd:element name="id" type="xsd:int"

Define un elemento llamado "id" de tipo "xsd:int" (entero) dentro del tipo complejo "usuario".

- xsd:element name="nombre" type="xsd:string"

Define un elemento llamado "nombre" de tipo "xsd:string" (cadena de texto) dentro del tipo complejo "usuario".

- xsd:element name="direccion" type="tns:direccion"

Define un elemento llamado "direccion" de tipo "tns:direccion" (refiriéndose al tipo complejo "direccion" definido anteriormente) dentro del tipo complejo "usuario". Esto indica que el elemento "direccion" es de tipo "direccion".

Arrays

En WSDL, cuando necesitamos definir un array, no existe un tipo base adecuado en XML Schema para hacerlo directamente. En su lugar, podemos utilizar el tipo Array definido en el esquema de codificación SOAP.

```
<xsd:complexType name="ArrayOfUsuario">
  <xsd:complexContent>
    <xsd:restriction base="soapenc:Array">
      <xsd:attribute ref="soapenc:arrayType"
                     wsdl:arrayType="tns:Usuario[]" />
    </xsd:restriction>
  </xsd:complexContent>
</xsd:complexType>
```

En el ejemplo anterior, se define un tipo complejo llamado ArrayOfUsuario, que representa un array de elementos de tipo Usuario. Para ello, se utiliza la restricción xsd:restriction con la base soapenc:Array, que indica que estamos definiendo un array. Dentro de esta restricción, se utiliza el atributo ref para hacer referencia al atributo arrayType, que especifica el tipo de array que estamos definiendo. En este caso, el atributo arrayType tiene el valor tns:Usuario[], lo que indica que estamos definiendo un array de elementos de tipo Usuario.

Es importante tener en cuenta que este ejemplo hace referencia a los espacios de nombres soapenc y tns, los cuales deben estar definidos en el documento WSDL para ser utilizados correctamente. Además, este ejemplo asume que ya se ha definido previamente el tipo complejo Usuario en la sección types del documento WSDL.

Sin embargo, en muchos casos, no será necesario definir tipos de array personalizados en el documento WSDL. En su lugar, podemos utilizar los tipos propios de XML Schema, como xsd:string, xsd:float o xsd:boolean, según sea necesario. Esto es especialmente útil cuando los arrays contienen tipos de datos simples y no necesitamos definiciones más complejas.

Message

La sección message en un documento WSDL sirve para definir los mensajes que se utilizan en un servicio web. Un mensaje en WSDL representa una operación específica que se puede invocar en el servicio web.

La sección message consta de uno o más elementos message que definen los mensajes de entrada y salida de una operación. Cada mensaje puede contener uno o más elementos part, que representan los parámetros individuales o los valores de retorno de la operación.

La sección message es importante porque permite describir de manera precisa la estructura y el contenido de los mensajes que se intercambian entre el cliente y el servicio web. Al definir los mensajes en el documento WSDL, se establece un contrato claro entre el cliente y el servicio web sobre los datos que se deben enviar y recibir.

Cada message se identifica mediante un atributo name único, que proporciona un nombre descriptivo para el mensaje. Es común utilizar una convención de nomenclatura que incluye los sufijos "Request" y "Response" al final de los nombres de los mensajes para indicar si son mensajes de entrada o salida, respectivamente.

La sección message es una parte esencial en la descripción de un servicio web, ya que define los tipos y la estructura de los datos que se transmiten entre el cliente y el servicio. Esta información es utilizada por

herramientas y aplicaciones para generar automáticamente el código necesario para interactuar con el servicio web.

```
<message name="getUsuarioRequest">
  <part name="id" type="xsd:int"/>
</message>
<message name="getUsuarioResponse">
  <part name="getUsuarioReturn" type="tns:usuario"/>
</message>
```

En el ejemplo anterior, se definen dos mensajes: getUsuarioRequest y getUsuarioResponse. El mensaje getUsuarioRequest representa los parámetros de entrada de la función getUsuario, que en este caso es el parámetro id de tipo xsd:int. El elemento part dentro del mensaje especifica el nombre (name) y el tipo (type) del parámetro de entrada.

Por otro lado, el mensaje getUsuarioResponse representa el valor de salida de la función getUsuario, que es un objeto usuario. El elemento part dentro del mensaje especifica el nombre (name) del valor de salida y el tipo (type) del objeto usuario.

Es importante destacar que es común utilizar el sufijo "Request" al final del nombre del mensaje que representa los parámetros de entrada, y el sufijo "Response" para el mensaje que representa los parámetros de salida. Esto ayuda a tener una nomenclatura clara y consistente en el documento WSDL.

Estilos de enlazado

En WSDL, el estilo de enlace se refiere a cómo se transmiten los mensajes dentro de las solicitudes y respuestas SOAP en un servicio web. Hay dos estilos de enlace comunes: "document" y "RPC". Además, cada estilo de enlace puede ser de tipo "encoded" o "literal". La combinación de estilo de enlace "document/encoded" no se utiliza comúnmente.

- **encoded** : significa que se utilizará un conjunto de reglas de codificación para convertir los parámetros de las peticiones y respuestas en XML. Estas reglas de codificación se especifican en el espacio de nombres http://schemas.xmlsoap.org/soap/encoding/ y forman parte del protocolo SOAP. Esto implica que los valores de los parámetros se codificarán de acuerdo con estas reglas antes de ser transmitidos.
- **literal** : implica que los valores de los parámetros se transmitirán directamente en XML sin necesidad de aplicar una codificación específica.

PortType

En un documento WSDL, las funciones que se crean en un servicio web se conocen como operaciones. Para agrupar estas operaciones de manera organizada, se utilizan los elementos portType. Un portType en WSDL contiene una lista de funciones (operaciones) y para cada función se especifica la lista de parámetros de entrada y salida que le corresponden.

```
<portType name="usuarioPortType">
  <operation name="getUsuario">
    <input message="tns:getUsuarioRequest"/>
```

```
    <output message="tns:getUsuarioResponse"/>
  </operation>
</portType>
```

Se define un portType llamado "usuarioPortType" que contiene una operación llamada "getUsuario". Esta operación tiene un parámetro de entrada (input) llamado "getUsuarioRequest" y un parámetro de salida (output) llamado "getUsuarioResponse". Los atributos "message" en los elementos input y output hacen referencia a mensajes previamente definidos en el documento WSDL.

Normalmente, a menos que se esté desarrollando un servicio web muy complejo, el documento WSDL contendrá un único portType que agrupe todas las operaciones.

Cada portType debe tener un atributo name con un nombre único. Cada operación (elemento operation) también debe tener un atributo name que corresponde al nombre de la función que se ofrece. Dependiendo del tipo de operación, se pueden tener diferentes configuraciones:

- Para funciones que no devuelven un valor, se utiliza solo el elemento input para enviar un mensaje al servidor.
- Para funciones que reciben parámetros y devuelven un resultado, se utilizan los elementos input y output en ese orden.
- En casos raros, se pueden encontrar funciones con solo un## RESUMEN

junto con el elemento se utiliza para describir y configurar un servicio web específico en el documento WSDL. Proporciona la información necesaria para acceder al servicio, como el binding utilizado y la URL del punto de conexión. parámetro de salida (el servidor envía una notificación al cliente) o con los parámetros output e input en ese orden (el servidor solicita información al cliente). Estos casos son menos comunes.

Al definir una función (operación), es importante tener en cuenta el orden de los elementos input y output. Por lo general, los elementos input y output hacen referencia a mensajes definidos previamente mediante el atributo message.

La estructura de portType y las operaciones dentro de él son esenciales para describir las funcionalidades ofrecidas por un servicio web en el documento WSDL.

Binding

El elemento en un documento WSDL se utiliza para indicar cómo se codifica y transmite la información para cada función (operation) definida en el portType. Especifica el estilo de enlazado que se utilizará y otros detalles relacionados con el protocolo de transporte.

```
<binding name="usuarioBinding" type="tns:usuarioPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="getAlumno">
    <soap:operation soapAction="http://ejercicios.soap.es/usuarios/Usuario.php?
getUsuario" style="rpc" />
    <input>
      <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

```
    </input>
    <output>
      <soap:body use="encoded"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
    </output>
  </operation>
</binding>
```

Dentro del elemento binding, se utiliza el elemento soap:binding para especificar el estilo y el protocolo de transporte. En este caso, se establece el estilo como "rpc" y el transporte como HTTP utilizando la URL http://schemas.xmlsoap.org/soap/http.

A continuación, se define la operación "getAlumno" dentro del elemento operation. El atributo name indica el nombre de la operación que se corresponde con la función en el portType.

El elemento soap:operation contiene el atributo soapAction, que especifica la URL asociada a esa función en particular. Proporciona información sobre cómo se debe procesar la operación.

Dentro del elemento input y output, se utiliza el elemento soap:body para indicar el estilo de enlazado con el atributo use establecido como "encoded". El atributo encodingStyle especifica el espacio de nombres de codificación correspondiente, en este caso http://schemas.xmlsoap.org/soap/encoding/.

Service

El elemento service se utiliza para definir un servicio en el documento WSDL. Normalmente, solo habrá un elemento service en cada documento WSDL.

Dentro del elemento service, se especifica un nombre para el servicio utilizando el atributo name. Este nombre identifica de manera única el servicio dentro del contexto del documento WSDL.

El elemento port se utiliza para hacer referencia al elemento binding definido previamente, estableciendo la asociación entre el servicio y el estilo de enlazado. El atributo name en el elemento port proporciona un nombre para el puerto del servicio.

Dentro del elemento port, se utiliza el elemento soap:address para indicar la ubicación o URL en la que se puede acceder al servicio. El atributo location en el elemento soap:address especifica la URL del punto de conexión del servicio web.

```
<service name="usuario">
    <port name="usuarioPort" binding="tns:usuarioBinding">
        <soap:address location="http://ejercicios.soap.es/usuarios/Usuario.php" />
    </port>
</service>
```