



Servicios Web

Descripción Soap

Descripción Wsdl

Servicios Web



- El esquema XML por sí solo no puede describir totalmente un Servicio Web. Supongamos 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.

Servicios Web



- 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 mensaje de petición y respuesta para los métodos sumar y restar. 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.

Servicios Web



- 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:

Servicios Web



- 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.



- Elementos
 - 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.



- 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 de disponer de un mecanismo común de identificar los mensajes. El elemento Message proporciona este nivel común de abstracción al que se hará referencia en otras partes del documento WSDL.



- Elemento portType
 - El elemento portType 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 portType como una definición de internas en donde cada método se puede definir como una operación.



- Elemento porType
- Un tipo puerto se compone de un conjunto de electos operation que define una determinada acción. Los electos 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.



- Rpc
 - **Llamada a Procedimiento Remoto**) es un **protocolo** que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos. El protocolo es un gran avance sobre los **sockets** usados hasta el momento. De esta manera el programador no tenía que estar pendiente de las comunicaciones, estando éstas encapsuladas dentro de las RPC.



- Rpc
 - Las RPC son muy utilizadas dentro del **paradigma cliente-servidor**. Siendo el cliente el que inicia el proceso solicitando al servidor que ejecute cierto procedimiento o función y enviando éste de vuelta el resultado de dicha operación 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.



- La asociación de 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.



- 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.



- 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.



- Elementos de 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.



- En Java SE 6.0 se incluye **JAX-WS** (Java API for XML Web Services). El siguiente cuadro muestra las APIs disponibles en Java SE 6.0 y sus paquetes correspondientes.
- **Paquete API** javax.xml.ws Núcleo del API JAX-WS
javax.xml.soap API para crear y construir mensajes SOAP
javax.jws Metadatos para la construcción de Servicios Web



- Es necesario importar nuestras a la clase
- importamos `javax.jws.WebService`, lo cual es necesario para marcar la clase con la anotación `@WebService`
 - Con esto habilitamos la clase para que pueda ser publicada como un Servicio Web



- El siguiente paso es generar los *artefactos de despliegue* para dicha clase, para ello utilizamos la herramienta **wsgen**, incluida en Java SE 6.0, a la cual le pasamos como argumento la clase
- `wsgen -cp . Paquete.clase`

Jaws



- Si creamos un *Endpoint* dentro de una aplicación que actuará como host del servicio web.



utiliza el método **publish** de la clase `javax.xml.ws.Endpoint` para crear y publicar un *Endpoint* para una instancia de la clase en la dirección `http://localhost:8080/`. Esto último gracias a que la clase `Endpoint` utiliza el servidor web ligero de Sun que está incluido en Java SE dentro del paquete `com.sun.net.httpserver`.

jaws



- **import javax.xml.ws.Endpoint;**
- Import nombre de la clase
- Compilamos el endPoint



- Pero primero necesitamos generar las clases asociadas al cliente. Para ello utilizaremos la herramienta **wsimport** incluida en Java SE 6.0, a la cual le pasamos como argumento la ubicación del contrato del servicio, es decir el documento WSDL



- para invocar la operación de nuestro Servicio Web primero creamos el *localizador* del servicio (con la clase de servicio creada) y mediante éste obtenemos un *puerto* (de tipo clase) por medio del cual después invocamos al método del servicio —tras del cual se invoca la operación del Servicio Web el cual a su vez invoca el método de la clase



- package hello;
- import javax.jws.WebService;

@WebService

```
public class Hello {  
    private String message = new String("Hola ");  
    public String sayHello(String name) {  
        String msg = message + name + ".";  
        System.out.println(msg);  
        return msg;  
    }  
}
```

jaws



- `javac hello/Hello.java`
- `wsgen -cp . hello.Hello`



- **WsGen**
- Nuevo es una herramienta que funciona de la misma manera que Se necesita un archivo (EJB-JAR, WAR, JAR cliente, o EAR) y genera todos los archivos necesarios de servicios web:
- Crea específicas de su proveedor de servicios web para el despliegue de archivos del lado del servidor y, en caso necesario, el lado del cliente (Eje utilizará su propio formato WSDD).
- Genera y compila el lado del cliente artefactos (Servicios y Port Ataduras aplicación clases

jaws



```
import javax.xml.ws.Endpoint;  
import hello.Hello;
```

```
public class SimpleHelloWS {  
    public static void main(String[] args) {  
        String endpointAddress = "http://localhost:8080/hello";  
        Endpoint.publish(endpointAddress, new hello.Hello());  
        System.out.println("El Servicio Web Hello se esta ejecutando...");  
        System.out.println("WSDL: " + endpointAddress + "?wsdl");  
    }  
}
```

jaws



- `javac SimpleHelloWS.java`
- `java SimpleHelloWS`
- Arranca el servidor

jaws



- Tenemos que crear los ficheros que permiten establecer la comunicación
`wsimport http://localhost:8080/hello?wsdl`



```
public class Client
{ public static void main(String[] args)
{ String arg = null; String result = null;
  if (args.length > 0)
  arg = args[0]; }
else { arg = "Anonimo";
try {
hello.HelloService service = new hello.HelloService();
hello.Hello port = service.getHelloPort();
  result = port.sayHello(arg); }
}
```

jaws



```
catch (Exception ex) {  
    result = ex.toString();  
    finally{ System.out.println(result);  
}  
}  
}  
}
```

jaws



- `javac Client.java`
- `java Client Yo Hola Yo.`

Cliente anónimo



```
Símbolo del sistema
01/04/2008 08:48      854 SimpleHelloWS.class
01/04/2008 08:26      383 SimpleHelloWS.java
29/03/2008 23:28      857 SimpleHelloWS2.class
29/03/2008 23:25      392 SimpleHelloWS2.java
25/09/2007 00:13        26.112 tnameserv.exe
25/09/2007 00:13    122.880 unpack200.exe
25/09/2007 00:13        25.600 wsgen.exe
25/09/2007 00:13        25.600 wsimport.exe
25/09/2007 00:13        25.600 xjc.exe
      54 archivos      2.109.261 bytes
      5 dirs 71.501.692.928 bytes libres

C:\jdk1.6\bin>java Client
Hola Anonimo.

C:\jdk1.6\bin>
```

Servicio anónimo



```
C:\> Símbolo del sistema - java SimpleHelloWS
Microsoft Windows [Versión 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. Reservados todos los derechos.

C:\Users\pepe>cd\

C:\>cd jdk1.6\bin

C:\jdk1.6\bin>java SimpleHelloWS
El Servicio Web Hello se esta ejecutando...
WSDL: http://localhost:8080/hello?wsdl
Hola Anonimo.
-
```

Servicio Usuario



```
Símbolo del sistema - java SimpleHelloWS
Microsoft Windows [Versión 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. Reservados todos los derechos.

C:\Users\pepe>cd\

C:\>cd jdk1.6\bin

C:\jdk1.6\bin>java SimpleHelloWS
El Servicio Web Hello se esta ejecutando...
WSDL: http://localhost:8080/hello?wsdl
Hola Anonimo.
Hola Pedro.
Hola Pepito.
-
```

Cliente Usuario



```
CA: Símbolo del sistema
C:\jdk1.6\bin>java Client Alumno
Hola Alumno.
C:\jdk1.6\bin>_
```