**Jaxb**

Java Architecture for XML Binding (JAXB) permite a los desarrolladores Java asignar clases de Java a representaciones XML. JAXB proporciona dos características principales: la capacidad de serializar las referencias de objetos Java a XML y la inversa, es decir, deserializar XML en objetos Java, estas operaciones son conocidas como marshalling (Java a XML) y unmarshalling (XML a Java). Esta conversión es utilizada por los servicios web cuando en los mensajes son enviados objetos en lugar de datos nativos.

JAXB es una parte de la plataforma Java SE y una de las APIs de la plataforma Java EE, y es parte del Java Web Services Development Pack (JWSDP). También es uno de los fundamentos para WSIT. JAXB es parte de la versión 1.6 SE.

JAXB 1.0 fue desarrollado bajo el Java Community Process como JSR 31.1 A fecha de 2006, JAXB 2.0 está siendo desarrollado bajo JSR 222.2 Las implementaciones de referencia para estas especificaciones están disponibles bajo la licencia de código abierto CDDL en java.net.

Sitio oficial: [https://jaxb.java.net/](https://www.google.com/url?q=https://jaxb.java.net/&sa=D&ust=1503593628401000&usg=AFQjCNGwix3nN48aeXTr41-HSU_YoVCDEA" \t "_blank)

**Faults**

El elemento Fault de SOAP es utilizado para indicar mensajes de error, al momento de escribir un servicio web puede ser de utilidad que, en caso de producirse excepciones, la información de las mismas esté disponible para el cliente con el mayor detalle posible.

Como los clientes pueden ser escritos en diferentes plataformas utilizando diferentes lenguajes es necesaria la existencia de un mecanismo standard independiente de plataforma para comunicar los errores.

La especificación de SOAP (disponible en [http://www.w3.org/TR/soap/](https://www.google.com/url?q=http://www.w3.org/TR/soap/&sa=D&ust=1503593628402000&usg=AFQjCNH5R0nDdJ9LnDQsNQremREmyxSK2Q" \t "_blank)) define un forma independiente de plataforma para describir los errores dentro de un mensaje SOAP utilizando el elemento Fault.

Las faltas (Faults en inglés) SOAP  son generadas por los servidores de web services para reportar errores de la lógica de negocio o condiciones inesperadas.

En JAX-WS, las Java Exceptions (java.lang.Exception) que son arrojadas por los Servicios Web desarrollados en Java son mapeadas a faltas SOAP y enviadas al cliente para comunicar la razón de la falla.

**Tipos de Faults**

Las faltas SOAP puede ser de los siguientes tipos:

* Modeled — Se mapea a una excepción que es arrojada explícitamente desde la lógica de negocio del código Java y mapeada a las definiciones wsdl:fault en el archivo WSDL, cuando el servicio es deployado. En este caso las faltas SOAP son predefinidas.
* Unmodeled — Se mapea a una excepción (por ejemplo, java.lang.RuntimeException) que es generada en run-time cuando no está definida una falta de lógica de negocio en el WSDL. En este caso, las  Excepciones Java son representadas como faltas SOAP genéricas,  javax.xml.ws.soap.SOAPFaultException.

Las faltas son devueltas al cliente solo si un mensaje request/response está en uso. Si una operación de un Web Service está configurada como OneWay, la falta SOAP no es devuelta al cliente.

**Sub elementos del elemento Fault**

El elemento Fault de SOAP tiene los siguientes sub-elementos:

|  |  |
| --- | --- |
| Sub Elemento | Descripción |
| <faultcode> | Un código para identificar la falta. |
| <faultstring> | Una explicación interpretable por un humano de la falta. |
| <faultactor> | Información acerca de quien causó que ocurriera la falta. |
| <detail> | Contiene información del error específico de aplicación relativo al Body element. |

**Códigos SOAP Fault**

Los valores de fault code definidos debajo deben ser utilizados en el elemento faultcode cuando se describe una falta:

|  |  |
| --- | --- |
| Error | Descripción |
| VersionMismatch | Se encontró un namespace inválido para el elemento envoltorio SOAP. |
| MustUnderstand | Un subelemento del elemento Header, con el atributo mustUndersatnd configurado en “1”, no fué entendido. |
| Client | El mensaje está  formado incorrectamente o contiene información incorrecta. |
| Server | Hubo un problema con el servidor. |

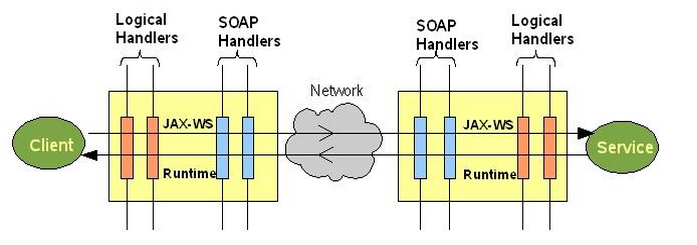
**SOAP message Handlers**

En ocasiones los servicios Web o sus clientes pueden necesitar acceder al mensaje SOAP para procesamiento adicional de los mensajes request o response. En  estos casos se pueden crear lo que se denomina SOAP message Handlers para habilitar a los web services o sus clientes a realizar este procesamiento adicional en los mensajes SOAP. Un SOAP message Handler provee un mecanismo para interceptar los mensajes SOAP tanto en el request como en el response de un Web Service.

Jax-WS soporta dos tipos de SOAP message handlers: SOAP handlers y Logical handlers. Los SOAP Handlers pueden acceder al mensaje SOAP entero, incluyendo el Header y el Body del mensaje. Los Logical Handlers pueden acceder solo al payload del mensaje, y no pueden cambiar ninguna información que sea específica del protocolo (como headers) en un mensaje.

**Orden de ejecución de los Handlers**

En la siguiente imagen se observa el orden del flujo de ejecución de los Handlers tanto desde el Cliente como hacia los web services.



**Rest**

La Transferencia de Estado Representacional o REST por sus siglas en Inglés (Representational State Transfer) es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

Si bien el término REST se refiere a un conjunto de principios de arquitectura, en la actualidad se utiliza en un sentido más amplio para describir cualquier interfaz web simple que utiliza XML o JSON y HTTP, sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de [servicios web](https://www.google.com/url?q=http://es.wikipedia.org/wiki/Servicio_web&sa=D&ust=1503593628414000&usg=AFQjCNEKXdgSJvaZ3H3Shbldof0YvzjNgw" \t "_blank) SOAP.

A partir de la versión 1.1 en adelante, JAX-RS es una parte oficial de [Java EE](https://www.google.com/url?q=http://es.wikipedia.org/wiki/Java_EE&sa=D&ust=1503593628414000&usg=AFQjCNFMHWyuIcRa58U8bAdqlD-vp7Aa2w) 6.

**Recursos**

Un concepto clave en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso - URI). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos.

La petición puede ser transmitida por cualquier número de conectores (por ejemplo clientes, servidores, cachés, túneles, etc.) pero cada uno lo hace sin "ver más allá" de su propia petición (lo que se conoce stateless (sin estado), otra restricción de REST, que es un principio común con muchas otras partes de la arquitectura de redes y de la información) Así, una aplicación puede interactuar con un recurso conociendo el identificador del recurso y la acción requerida, no necesitando conocer si existen cachés, proxys, cortafuegos, túneles o cualquier otra cosa entre ella y el servidor que guarda la información. La aplicación, sin embargo, debe comprender el formato de la información devuelta (la representación), que es por lo general un documento HTML o XML, aunque también puede ser una imagen o cualquier otro contenido.

**Jax-RS**

JAX-RS: Java API for RESTful Web Services es una especificación para la creación de servicios web basados en el ​​estilo arquitectónico Representational State Transfer (REST). JAX-RS utiliza anotaciones, introducidas en Java SE 5, para simplificar el desarrollo y despliegue de los clientes y puntos finales de los servicios web. A partir de la versión 1.1 en adelante, JAX-RS es parte oficial de Java EE 6.

JAX-RS cuenta con una implementación de referencia llamada Jersey, se pueden observar detalles de la misma en su página oficial:

[https://jersey.java.net/](https://www.google.com/url?q=https://jersey.java.net/&sa=D&ust=1503593628416000&usg=AFQjCNGnw9MvRsyEbzJULa7W9u3yrzJYHQ" \t "_blank)

Se puede obtener la especificación completa en:

[https://jcp.org/en/jsr/detail?id=339](https://www.google.com/url?q=https://jcp.org/en/jsr/detail?id%3D339&sa=D&ust=1503593628416000&usg=AFQjCNGHCsaxYfQ4-f6cR1dgFkqD9HSEaQ)

**WADL - Web Application Description Language**

Así como en SOAP contamos con el WSDL para obtener la definición de los métodos de un Web Service en REST contamos con el WADL. Para obtener el WADL  de un servicio REST se debe apuntar a la URL del recurso agregando /application.wadl.

Ej.: [http://localhost:8080/JavaWS-Clase4-RestCrud/recursos/application.wadl](https://www.google.com/url?q=http://localhost:8080/JavaWS-Clase4-RestCrud/recursos/application.wadl&sa=D&ust=1503593628418000&usg=AFQjCNGQ5HeLqJmjpHmwSVQIBnqK6uE04g" \t "_blank)

Esto nos dará como resultado un XML como el siguiente:

