**SOAP - Simple  Object  Access  Protocol**

É um protocolo baseado em XML que permite às aplicações trocarem informações estruturadas nas implementações de Web Services. Os dados são Transferidos no formato XML, encapsulados pelo protocolo SOAP. A sintaxe de uma mensagem SOAP é bem simples e contem algumas regras, tais como a utilização de XML, o uso dos namespace específicos da mensagem SOAP, seu namespace deve utilizar a codificação definida pela w3c, não deve conter referencia DTD e nem ter instruções de processamento XML.

* O elemento **ENVELOPE**é o elemento raiz de uma mensagem SOAP e corresponde à descrição da mensagem e do que deve ser processado.
* O elemento **HEADER**é opcional e tem como função estender as funcionalidades das mensagens SOAP.
* O elemento **BODY**é obrigatório e contém a informação propriamente dita.

O elemento **fault** SOAP tem os seguintes sub elementos:

* **faultcode**: um código para a identificação da falha.
* **faultstring**: Uma explicação legível humana da falha.
* **faultactor**: Informações sobre quem causou a falha acontecer.
* **detail**: Mantém aplicação específica informações de erro relacionado com o elemento do corpo.

**Regras de sintaxe XML:**  
  
• Deve ser codificada em XML.  
• Deve usar o SOAP Encoding namespace. (Não contém um valor default, deve-se, portanto, configurá-lo). Exemplo: soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">  
• Deve usar o SOAP Envelope namespace. Tal namespace DEVE ser o seguinte: "xmlns:soap="http://www.w3.org/2003/05/soap-envelope/". Se for utilizado um namespace diferente, será gerado um erro e a mensagem será descartada.  
• Não deve conter instruções de processamento XML.   
• Não pode conter referência a uma DTD.

**WSDL - WEB SERVICES DESCRIPTION LANGUAGE**

Define como as interfaces dos Web services podem ser representadas. É uma linguagem baseada em XML utilizada para descrever e localizar Web Services.

A especificação WSDL define três aspectos de um web service : o que faz o serviço, como ele se comunica e onde o encontrar:

1. O tópico "o que" de um documento WSDL, denominado **interface**, especifica quais operações o serviço suporta e define o formato das mensagens que são enviadas e recebidas pelo serviço.
2. O "como" de um documento WSDL, denominado **ligação**, mapeia a interface abstrata para um conjunto concreto de protocolos. A ligação especifica os detalhes técnicos de como se comunicar com o webservice.
3. A parte "onde" de um documento WSDL descreve o**local da implementação** de um web service específico (seu ponto final)."

**Descrições de Interface do serviço do WSDL**

* <**definitions**>: elemento padrão de todos os documentos WSDL (raiz). Permite efetuar descrições sobre schemas e namespace.
* <**types**>: aqui deverão ser descritos os tipos de dados suportados pelo serviço em questão.
* <**message**>: aqui devem ser especificados os padrões de entrada e saída de dados dos web services.
* <**portType**>: aqui devem ser descritos os agrupamentos lógicos das operações. São as operações executadas pelo web service.
* <**binding**>: aqui devem ser apresentados os protocolos de comunicação que os web services utilizam.
* <**operation**>: região que permite a especificação das assinaturas dos métodos disponibilizados.
* **<service>** - coleção de portas (endpoints) suportadas pelo serviço.

Pessoal, no WSDL 1.1 tínhamos 4 tipos básicos de operação: Unidirecional, Request-Response, Solicit Response e Notification. A partir do WSDL 2 essas definições mudaram:

* **Apenas-Entrada** (**In-Only**): É equivalente ao ***unidirecional***. Um padrão de troca de mensagem unidirecional onde o consumidor envia uma mensagem ao provedor que fornece apenas uma resposta de estado.
* **Apenas-Entrada Robusto** (**Robust In-Only**): Este padrão é para a troca de mensagens confiável unidirecional. O consumidor inicia com uma mensagem a qual o provedor responde com o estado. Se a resposta for um estado, a troca está completa, mas se for uma falha, o consumidor deve responder com um estado.
* **Entrada-Saída** (**In-Out**): É equivalente à **solicitação-resposta**. Um padrão de troca de mensagens bidirecional onde o consumidor inicia com uma mensagem, o provedor responde com uma mensagem ou uma falha e o consumidor responde com um estado.
* **Entrada e Saída-Opcional** (**In Optional-Out**): Um padrão de troca de mensagens bidirecional onde a resposta do provedor é opcional.

1. Apenas Saída
2. Apenas Saída Robusto
3. Saída Entrada
4. Saída Entrada-Opcional

**Segue abaixo os 4 tipos possíveis de chamada.**

**WSDL supports four basic types of operations. These are:**  
**One-way -** The service receives a message. This would mean that only the**<input>**element shown earlier will be present.   
  
**Request-Response -** The service receives a message and sends a response. Thus the operation will have an <input> element, followed by an <output> element as given in the <portType> syntax. The <fault> element is optional. The request-response pattern is most commonly used in SOAP services.   
  
**Solicit-Response -** The service (not the client) initiates communication by sending a message and receives a response. Hence, this operation will have an <output> element followed by an <input> element. The <fault> element is optional.   
  
**Notification -** The service sends a message and seeks no response. The operation will therefore, have a single <output> element only.

**UDDI**

Podemos comparar o UDDI com uma lista telefônica, da seguinte maneira:

* **Páginas Brancas**: contêm informações sobre nomes, endereços, números de telefone, além de outras informações sobre os fornecedores do serviço.
* **Páginas Amarelas**: contêm listagens comerciais baseadas nos tipos desses negócios, de maneira organizada por categoria específica ou regiões demográficas.
* **Páginas Verdes**: são usadas para indicar os serviços oferecidos por cada negócio, incluindo todas as informações técnicas envolvidas na interação com o serviço. Resumindo, explica como fazer a comunicação com eles.

**CORBA**

Ao contrário dos objetos tradicionais, os objetos em sistemas distribuídos possuem uma característica de dualidade: um estado dinâmico, tipicamente alocado em memória volátil (em tempo de execução), e um estado persistente, que não pode ser destruído após o encerramento do programa que os criou e que pode ser usado para reconstruir o estado dinâmico, devendo ser armazenado em memória não volátil, seja em sistema de arquivos ou banco de dados. A arquitetura CORBA, para prover a persistência, define o *Persistent Object Service*(POS) como sendo responsável por armazenar o estado persistente dos objetos, utilizando quatro elementos:

* Objetos Persistentes (Persistent Object (POs))
* Gerenciador de Objetos Persistentes (Persistent Objects Manager (POM))
* Serviços de Persistência de Dados (Persistent Data Services (PDSs))
* Base de Dados (Datastores)