# Web Services - SOAP



## Interoperabilidade entre Web Services

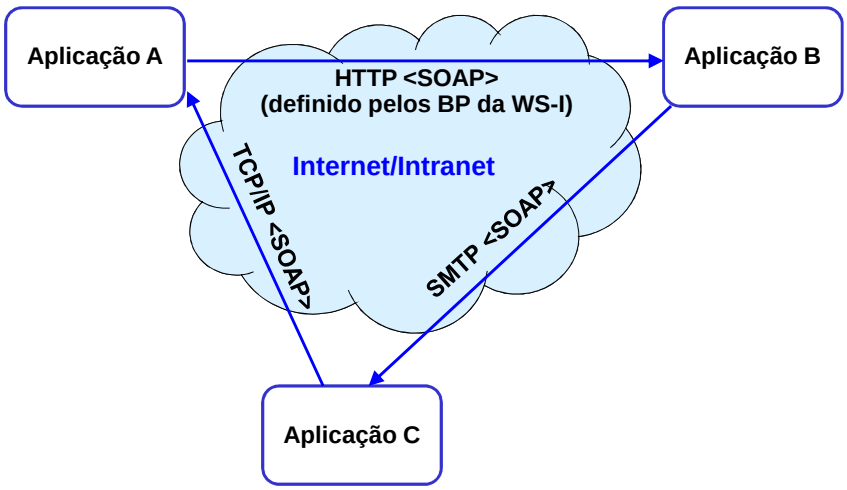
- A organização WS-I (*Web Services Interoperability*) define um conjunto de especificações, não proprietárias, sobre Web Services, designadas por *Basic Profile* (BP);
- Estes *Basic Profiles* são um conjunto de regras que definem como as aplicações podem usar as tecnologias Web Service, facilitando a interoperabilidade, essencialmente em questões relacionadas com SOAP, WSDL e UDDI;
- Para mais informação veja (www.ws-i.org)

## SOAP

- Embora originalmente SOAP fosse um acrónimo de *Simple Object Access Protocol*, hoje em dia não é mais que um nome para designar um protocolo de troca de mensagens usando XML;
- O SOAP 1.1 e 1.2 são standard de facto utilizados em tecnologias de *Web Services;*
- Um documento SOAP XML é designado por *SOAP message* ou *SOAP envelope* e obedece a um *XML Schema.*
- As *SOAP messages* são trocadas entre aplicações em rede;
- As mensagens SOAP podem ser também transportadas por SMTP (*Simple Mail Transfer Protocol*), FTP (*File Tranfer Protocol*) e TCP/IP (*Transmission Control Protocol/Internet Protocol*);
- Os *Basic Profile* da WS-I definem que em Web Services se usa SOAP via HTTP (*HyperText Transfer Protocol*);

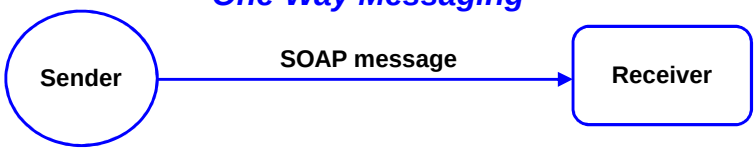


## Interoperabilidade entre aplicações com SOAP

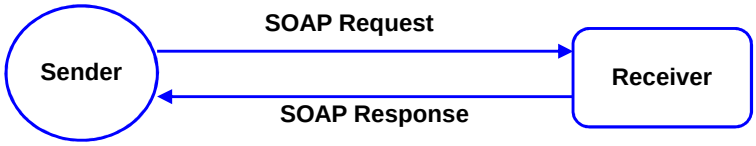


## *One-Way* versus *Request/Response Messaging*

### *One-Way Messaging*




### *Request/Response Messaging*




**O protocolo SOAP define como as mensagens são estruturadas e processadas, de forma independente de linguagens de programação, de plataformas *middleware* e de sistemas operativos.**



## Estrutura básica de uma mensagem SOAP

```
<?xml version="1.0" encoding="utf-8" ?>          não obrigatório
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
    <soap:Header>
        <! -- Dados do Header -- >
    </soap:Header>                                opcional
    <soap:Body>
        <! -- Dados da mensagem a ser trocada entre aplicações -- >
    </soap:Body>
</soap:Envelope>
```

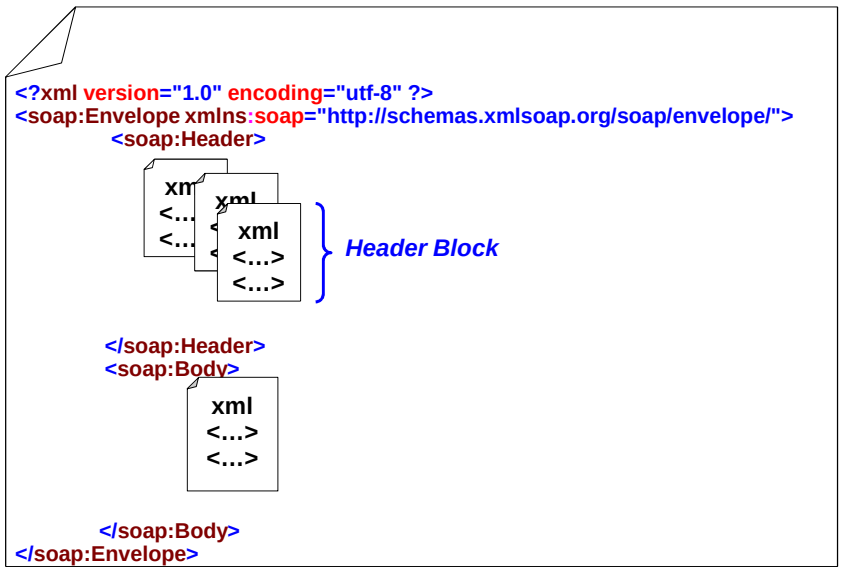
**Uma mensagem SOAP deve estar conforme com o *schema SOAP 1.1*, que requer que os elementos e atributos sejam totalmente *qualified* (uso de prefixos ou *namespaces* por omissão).**
**No caso do SOAP 1.2 o prefixo é soap12.**

## Estrutura básica de uma mensagem SOAP

- Não existe limitação para o tipo de XML transportado no elemento *Body*, tornando por isso o SOAP bastante flexível;
- Os dados em *Body* podem ser formados por qualquer elemento XML arbitrário, por exemplo, uma ordem de compra (*purchaseOrder*), ou um elemento que define os argumentos de chamada de uma operação (*Procedure Call*);
- O *Header,* que é opcional, pode conter um ou mais elementos XML com informação adicional relacionada com a mensagem: *Security credentials, transaction IDs, routing instructions, debugging information, XML digital signature, etc.*
- Por exemplo, podemos colocar no *Header* um elemento que contenha um identificador único da mensagem para efeitos de *debugging* ou *logging*.



## Estrutura básica de uma mensagem SOAP



## Exemplo de uma mensagem SOAP

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:myns="http://SD/HDR">
  <soap:Header>
     <myns:message-id>11d1def553e4:b1c5fa</myns:message-id>
  </soap:Header>
  <soap:Body>
    <po:purchaseOrder orderDate="2004-09-22"
                xmlns:po="http://SD/PO">
       <po:accountName>ISEL</po:accountName>
       <po:accountNumber>9955</po:accountNumber>
       <po:address>
           <po:name> AMAZON.COM</po:name>
           <po:street>1850 Mercer Drive</po:street>
           <po:city>Lexington</po:city>
       </po:address>
       <po:book>
            <po:title>Web Services</po:title>
            <po:quantity>2</po:quantity>
            <po:price>24.99</po:price>
       </po:book>
    </po:purchaseOrder>
  </soap:Body>
</soap:Envelope>
```



## Exemplo de uma mensagem SOAP em .NET

```
[WebService (Description="Soma dois inteiros", Namespace="http://DEETC.SD")]
public class Soma : System.Web.Services.WebService {
     [WebMethod]
     public int add(int op1, int op2)  {
          return op1+op2;
     }
}
```

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:xsd="http://www.w3.org/2001/XMLSchema"
       xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
    <add   xmlns="http://DEETC.SD">
        <op1>5</op1>
        <op2>72</op2>
    </add>
</soap:Body>
</soap:Envelope>
```

caso não se defina o atributo *Namespace* na classe o que aparece no SOAP Body é *http://tempuri.org/*

## Mensagem SOAP de resposta em .NET

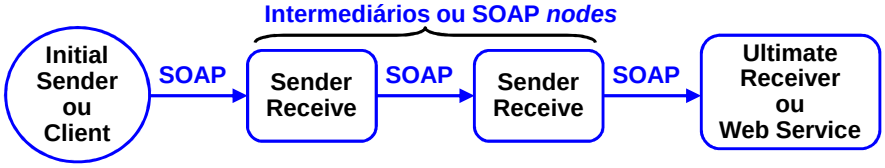
```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
    <addResponse  xmlns="http://DEETC.SD">
        <addResult>77</addResult>
    </addResponse>
</soap:Body>
</soap:Envelope>
```

Todos os elementos de uma mensagem SOAP devem ser *namespace-qualified* porque o XML schema do SOAP 1.1 e 1.2 especificam o atributo elementFormDefault="qualified"



## *SOAP Headers*

- A especificação SOAP define regras para o processamento dos *Header Blocks* ao longo do *message path* (caminho ou rota da mensagem entre o emissor inicial e o último receptor, incluindo o processamento feito pelos intermediários);
- As regras SOAP especificam que os nós intermediários devem processar determinados *Header Blocks* e o que deve ser feito com os mesmos depois de terem sido processados;




- Cada intermediário recebe a mensagem, processa um ou mais *Header Blocks* e pode acrescentar ou remover outros *Header Blocks*. Os intermediários não devem modificar o conteúdo do elemento <Body>, embora não seja proibido pela especificação WS-I BP 1.0

## Composição de *Headers*

Usar os *headers* que forem necessários

WS-Addressing

WS-Security

WS-Reliable Messaging

```
<S:Envelope … >
 <S:Header>
  <wsa:ReplyTo xmlns:wsa="">
    <wsa:Address>http://business456.com/User12</wsa:Address>
  </wsa:ReplyTo>
  <wsa:To>http://fabrikam123.com/Traffic</wsa:To>
  <wsa:Action>http://fabrikam123.com/Traffic/Status</wsa:Action>
  <wssec:Security>
    <wssec:BinarySecurityToken
          ValueType="wssec:X509v3"
          EncodingType="wssec:Base64Binary">
      dWJzY3JpYmVyLVBlc…..eFw0wMTEwMTAwMD
    </wssec:BinarySecurityToken>
  </wssec:Security>
  <wsrm:Sequence>
    <wsu:Identifier>http://fabrikam123.com/seq1234</wsu:Identifier>
    <wsrm:MessageNumber>10</wsrm:MessageNumber>
  </wsrm:Sequence>
 </S:Header>
 <S:Body>
   <app:TrafficStatus xmlns:app="http://highwaymon.org/payloads">
    <road>520W</road><speed>3MPH</speed>
   </app:TrafficStatus>
 </S:Body>
</S:Envelope>
```



## Exemplo de *Soap Headers*

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/enevelope/"
   xmlns:proc="http://DEETC.SD/processed-by" >
  <soap:Header>
    <proc:processed-by>
      <node>
        <time-in-millis>1013694680000</time-in-millis>
        <identity>http://www.DEPcustomer.com</identity>
      </node>
      <node>
        <time-in-millis>1013694680010</time-in-millis>
        <identity>http://www.DEPsales.com</identity>
      </node>
    </proc:processed-by>
  </soap:Header>
  <soap:Body>
    <!-- dados especificos da aplicação, por exemplo Purchase order -->
  </soap:Body>
</soap:Envelope>
```

Cada intermediário vai acrescentando um *header block* com o tempo de processamento e a sua identidade

## Atributo *Actor*

- Como é que um intermediário sabe quais os *header blocks* que deve processar ?
- O atributo *actor* especifica o papel (*role*) que um determinado intermediário deve fazer;
- Um *actor* pode desempenhar vários papéis, pelo que a escolha da palavra *actor* pode confundir. Em SOAP 1.2 este atributo mudou para *role;*
- O atributo *actor* está definido no mesmo *namespace* de *Envelope*, *Header* e *Body* ("http://schemas.xmlsoap.org/soap/envelope/");
- O atributo *actor* usa um URI (Uniform Resource Identifier) para identificar o papel que o nó deve fazer para processar o *header block*;
- Em adição, aos URIs dependentes da aplicação, o SOAP identifica dois papéis standard: *next* e *ultimate receiver*.
- *next* significa que o proximo nó deve processar o *header* . É identificado pelo URI "http://schemas.xmlsoap.org/soap/actor/next".
- *ultimate receiver* significa que o último receptor deve processar a mensagem. Não tem URI definido. A ausência do atributo *actor* no *header block* significa este papel.



## Atributo *actor* para definir *o papel (role)* dos intermediários

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
  xmlns:soap="http://schemas.xmlsoap.org/soap/enevelope/"
  xmlns:proc="http://DEETC.SD/processed-by" >
  xmlns:mid="http://DEETC.SD/message-id" >
  <soap:Header>
    <mid:messageID  soap:actor="http://www.deetc.isel.pt/logger" >
       11d1def534ea:b1c5fa:f3bf4d7:-8000
    </mid:messageID>
    <proc:processed-by>
      <node>
        <time-in-millis>1013694680000</time-in-millis>
        <identity>http://www.DEPcustomer.com</identity>
      </node>
    </proc:processed-by>
  </soap:Header>
  <soap:Body>
    <!-- dados especificos da aplicação, por exemplo Purchase order -->
  </soap:Body>
</soap:Envelope>
```

O papel *logger* pode ser, por exemplo, acrescentar o header block <proc:processed-by>

Todos os nós por onde passa a mensagem e que identificam que podem realizar o papel de *logger* ("http://www.deetc.isel.pt/logger") processam o *Header Block messageID*.

## atributo *mustUnderstand*

- Os *Header Blocks* podem ter um atributo que define se o *header block* tem de ser forçosamente processado;

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/enevelope/"
   xmlns:proc="http://DEETC.SD/processed-by" >
  <soap:Header>
     <proc:processed-by
           soap:actor="http://schemas.xmlsoap.org/soap/actor/next"
           soap:mustUnderstand="true" >
       <node>
          <time-in-millis>1013694680000</time-in-millis>
          <identity>http://www.DEPcustomer.com</identity>
       </node>
     </proc:processed-by>
  </soap:Header>
  <soap:Body>
    <!-- dados especificos da aplicação, por exemplo Purchase order -->
  </soap:Body>
</soap:Envelope>
```

o próximo nó tem de processar o *header block* <proc:processed-by>

valor "true" ou "false" ou de acordo com WS-I "1" ou "0"



## Tratamento de Falhas – SOAP 1.1 *faults*

- O que acontece se um nó não consegue interpretar (*Understand*) e processar um *header block,* por exemplo, mensagem mal formatada, erros de versão, ou outro tipo de erro especifico da aplicação ?
- O SOAP suporta a existência de mensagens de erro (*SOAP fault messages*);
- O elemento <Body> pode conter uma mensagem com dados específicos da aplicação ou uma *fault message* (nunca as duas em simultâneo);
- No modo *Request/Response* um receptor é obrigado, no caso de erro, a enviar para o emissor uma mensagem *SOAP fault*.
- Quando uma *fault message* é gerada, o elemento <Body> deve conter unicamente um elemento <Fault>. O elemento <Fault> contém um elemento <faultcode> e um elemento <faultstring> obrigatórios. Opcionalmente pode conter os elementos <faultactor> e <detail>.

## SOAP 1.1 fault message

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:myns="http://DEETC.SD/Faults" >
  <soap:Body>
    <soap:Fault>
        <faultcode> soap:Client</faultcode>
        <faultstring xml:lang="pt" >Formato do ISBN Inválido</faultstring>
        <faultactor>http://www.xyz.com/ValidaPedidos</faultactor>
        <detail>
            <myns:InvalidISBN>
              <received-value>19318224-D</received-value>
              <conformance-rules>
                    Formato ISBN standard: D-DDD-DDDDD-D
                    em que D representa digitos entre 0 e 9
              </conformance-rules>
            </myns:InvalidISBN>
        </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```



## Standard *fault codes*

- Embora seja permitido usar códigos arbitrários devem só ser usados os seguintes 4 códigos:
- **soap:Client** (sender em SOAP 1.2) – significa que o emissor causou o erro, por exemplo, mensagem <Body> mal formatada ou com falta de elementos, falta de um *header* especifico etc.
- **soap:Server** (receiver em SOAP 1.2) – Significa que o receptor teve uma falha (por exemplo falta de conexão a uma base de dados ou divisão por zero). Neste caso o emissor pode reenviar a mensagem SOAP novamente, pois este erro não se refere ao conteúdo da mensagem.
- **soap:VersionMismatch** – O receptor não reconhece o *namespace* associado aos elementos <Envelope>,<Header>,<Body> e <Fault>. Não se aplica a *header blocks*, versão XML ou a elementos específicos da aplicação existentes em <Body>
- **soap:MustUnderstand** – Usado quando um nó não consegue processar um *header block* que lhe é destinado (através do atributo *actor*) e o *header block* tem o atributo **mustUnderstand** a true.

## Estrutura de uma *fault message* SOAP 1.2

```
<env:Envelope   xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header> . . . </env:Header>
 <env:Body>
   <env:Fault>
      <env:Code> (required)
          <env:Value> (required)
              env:VersionMismatch | env:MustUnderstand | Sender | DataEncodingUnknown | Receiver
          </env:Value>
          <env:Subcode> (optional)
               <env:Value> Bad arguments </env:Value> (required)
          </env:Subcode>
      </env:Code>
      <env:Reason> (required)
           <env:Text xml:lang="en-US"> Processing error </env:Text> (required)
           <env:Text xml:lang="pt"> Erro de processamento </env:Text> (optional)
            ...
      </env:Reason>
      <env:Node> (optional)
           http://enterprise.com/processingNode
      </env:Node>
      <env:Role> (optional)
              http://enterprise.com/processingRole
      </env:Role>
      <env:Detail ...> (optional)
        ... (any XML fragment)
      </env:Detail>
   </env:Fault>
 </env:Body>
</env:Envelope>
```



## Exemplo de *Fault Code* em *WCF*

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
            xmlns:a="http://www.w3.org/2005/08/addressing">
  <s:Header>
    <a:Action s:mustUnderstand="1">
              http://www.w3.org/2005/08/addressing/soap/fault
    </a:Action>
    <a:RelatesTo>urn:uuid:66d4c7a8-dc0a-4305-b229-180221dfae1c</a:RelatesTo>
  </s:Header>
  <s:Body>
    <s:Fault>
      <s:Code>
        <s:Value>s:Sender</s:Value>
        <s:Subcode>
          <s:Value>a:DestinationUnreachable</s:Value>
        </s:Subcode>
      </s:Code>
      <s:Reason>
        <s:Text xml:lang="pt-PT">
          The message with To 'http://localhost:8081/ServiceOla' cannot be processed
          at the receiver, due to an AddressFilter mismatch at the EndpointDispatcher.
          Check that the sender and receiver's EndpointAddresses agree.
        </s:Text>
      </s:Reason>
    </s:Fault>
  </s:Body>
</s:Envelope>
```

## Modos das mensagens SOAP

- Excepto o caso das mensagens *fault* o SOAP não especifica o conteúdo do elemento <Body>. Desde que o <Body> tenha XML bem formado os dados específicos da aplicação podem ser qualquer coisa, podendo inclusive ser vazio;
- No entanto, o modo da mensagem pode ser, ***RPC*** ou ***Document*** com estilos de codificação ***Encoded*** ou ***Literal***;
- Assim, o SOAP suporta 4 modos de mensagens:
  - **Document/Literal** — Usado actualmente na maioria das plataformas
  - **RPC/Literal**
  - **Document/Encoded**
  - **RPC/Encoded**
- Devido a problemas de interoperabilidade e maior complexidade, os modos *Encoded* (RPC e Document) não são suportados por Web Services conformes com a WS-I BP 1.0;
- O termo *Literal* significa que um fragmento XML pode ser validado através do seu *XML Schema*;



## Mensagem Modo Document/Literal

Modo, por omissão, usado pelo .NET

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
        xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
        xmlns:myns="http://DEETC.SD/HDR">
   <soap:Header>
      <myns:message-id>11d1def553e4:b1c5fa</myns:message-id>
   </soap:Header>
   <soap:Body>
      <po:purchaseOrder orderDate="2004-09-22"   xmlns:po="http://DEETC.SD/PO">
         <po:accountName>ISEL</po:accountName>
         <po:accountNumber>9955</po:accountNumber>
         <po:address>
            <po:name> AMAZON.COM</po:name>
            <po:street>1850 Mercer Drive</po:street>
            <po:city>Lexington</po:city>
         </po:address>
         <po:book>
            <po:title>Web Services</po:title>
            <po:quantity>2</po:quantity>
            <po:price>24.99</po:price>
         </po:book>
      </po:purchaseOrder>
   </soap:Body>
</soap:Envelope>
```

O elemento <Body> contém um fragmento XML que contém dados específicos da aplicação e que pode ser validado através de um *namespace* e de um *XML schema*;

## Modo RPC/Literal

- Permite que mensagens SOAP modelem chamadas a *Procedures* ou métodos com parâmetros e valores de retorno;
- O conteúdo do elemento <Body> para uma Chamada (RPC Request) é sempre formatado como uma estrutura com: Nome do método e os parâmetros de chamada;
- O conteúdo do elemento <Body> para uma Resposta (RPC Response) contém o valor de retorno, eventuais parâmetros de saída ou um *message fault*;
- É importante reter que o *RPC/Literal* e o *Document/Literal* são muitas vezes indistinguíveis pois a diferença pode estar só no facto de não existir um *namespace* e um *XML schema* associado aos elementos do <Body>
- O .NET não suporta este modo.



## Mensagens Modo RPC Literal

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:myns="http://DEETC.SD/RPCLiteral" >
  <soap:Body>
     <myns:getBookPrice>
        <isbn>0-321-14300-8</isbn>
     </myns:getBookPrice
  </soap:Body>
</soap:Envelope>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:myns="http://DEETC.SD/RPCLiteral" >
  <soap:Body>
     <myns:getBookPriceResponse>
        <result>24.99</result>
     </myns:getBookPriceResponse>
  </soap:Body>
</soap:Envelope>
```

## *RPC/Literal* ou *Document /Literal* ?

```
<?xml version="1.0" encoding="utf-8" ?>
<soap:Envelope
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
    <add   xmlns="http://DEETC.SD">
        <op1>5</op1>
        <op2>72</op2>
    </add>
 </soap:Body>
</soap:Envelope>
```

**Mensagem SOAP em .NET**

**Embora possa parecer, numa primeira análise, *RPC/Literal*, note que é em modo Documento/Literal , pois tem no <Body> um elemento XML com *namespace*.**



## *RPC/Encoded em .NET*

- **Embora este modo não esteja conforme com o WS-I BP 1.0, existem no entanto *Web Services* que funcionam neste modo, por exemplo o disponível pela Amazon 3.0;**
- **Em .NET é possível aceder e também realizar este tipo de *Web Services*.**
- **Para realizar um Web Service RPC/Encoded basta colocar o atributo [System.Web.Services.Protocols.SoapRpcServiceAttribute] antes da classe que define o Web Service, tornando todas as mensagens RPC/Encoded ou no caso de só pretendermos um método coloca-se antes do método o atributo [System.Web.Services.Protocols.SoapRpcMethodAttribute]**
- **Em termos simples o RPC/Encoded é semelhante ao Document/Literal com a diferença que usa um atributo no <Body> para definir o encodingStyle e nos vários elementos que definem os argumentos é indicado o tipo de dados de acordo com o XML schema xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**

## Mensagem *RPC/Encoded em .NET*

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
   xmlns:tns="http://DEETC.SD"
   xmlns:types="http://DEETC.SD/encodedTypes"
   xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
 <soap:Body soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <tns:add>
       <op1 xsi:type="xsd:int">710</op1>
       <op2 xsi:type="xsd:int">100</op2>
    </tns:add>
 </soap:Body>
</soap:Envelope>
```