

# SOC

LECTURE NOTES

Service Oriented computing  
Computing

---

By, **Dr. Neila Ben Lakhel**  
Ph.D from Tokyo Institute of Technology  
Assistant Professor @ ENICARTHAGE

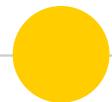
## Chapter 3. Web Services-Core functionality and standards

# Chap3

Today's lecture

< 97 >

**<https://neilabenlakhal.github.io/>**



## Where to find Web services ?

- API directory: <https://www.programmableweb.com/>
- A directory of free web services : <https://www.free-web-services.com/>
- Collection of public APIs: <https://explore.postman.com/templates>
- Financial web services <http://www.xignite.com/>
- SMS, voice, phone and address verification APIs <http://www.cdyne.com>
- Ebay shopping API :  
<https://developer.ebay.com/devzone/shopping/docs/Concepts/ShoppingAPIGuide.html>
- List of weather related and financial web services :  
<http://sofa.uqam.ca/soda/webservices.php>

<https://www.xignite.com/product/global-interest-rates#/DeveloperResources/Code/GetLatestRate>

The screenshot shows a web browser window with the URL <https://www.xignite.com/product/global-interest-rates#/DeveloperResources/Code/GetLatestRate>. The page content includes:

- A sidebar on the left with links: GetFederalRate, GetFederalRates, GetFHLBankRates, GetStateRate, GetStateRates, Charts, DrawRateChart, DrawRateChartCustom, DrawRateChartPreset, DrawYieldCurve, DrawYieldCurveCustom, DrawYieldCurvePreset, GetChartDesign, Utilities, GetRateDescription, and GetRateStatistics.
- A main area with sections for REST API (HTTP GET request example), WSDL (Web Service Definition Language), and Sample Code (links to ASPX/ASP.NET, C# (CSharp), Java/Axis, Perl/SoapLite, PHP 5, PHP/NuSoap, Python 3, Ruby, and VB.NET).

The screenshot shows the Xignite Rates API documentation page at <https://www.xignite.com/product/global-interest-rates#/DeveloperResources/Code/GetLatestRate>. The page features:

- The Xignite logo and navigation menu with links to Home, Why Xignite, Products, Customers, Resources, Blog, and Contact Us.
- A search bar and user account links for Free Trial and Login.
- A banner image of a person working on a laptop.
- A main title "XigniteRates" and tabs for Data Coverage and API List and Documentation.
- An "API Test Form" section for the GetLatestRate operation, which returns the most current value for an interest rate.
- Information about intra-day rates and their availability.
- A "Request" button to interact with the API.
- Links to other API operations like GetLatestRateFamily, GetRate, GetRateAsOfTime, GetRateFamily, GetRateFamilyTable, GetTodaysRate, Average Rates, GetAverageRate, GetAverageRates, and GetRateMovingAverage.
- Subscription requirements for certain operations.

[https://documenter.getpostman.com/view/8854915/Szf26WHn?version=latest&\\_ga=2.190467816.212368221.1602582773-908133927.1602582773](https://documenter.getpostman.com/view/8854915/Szf26WHn?version=latest&_ga=2.190467816.212368221.1602582773-908133927.1602582773)

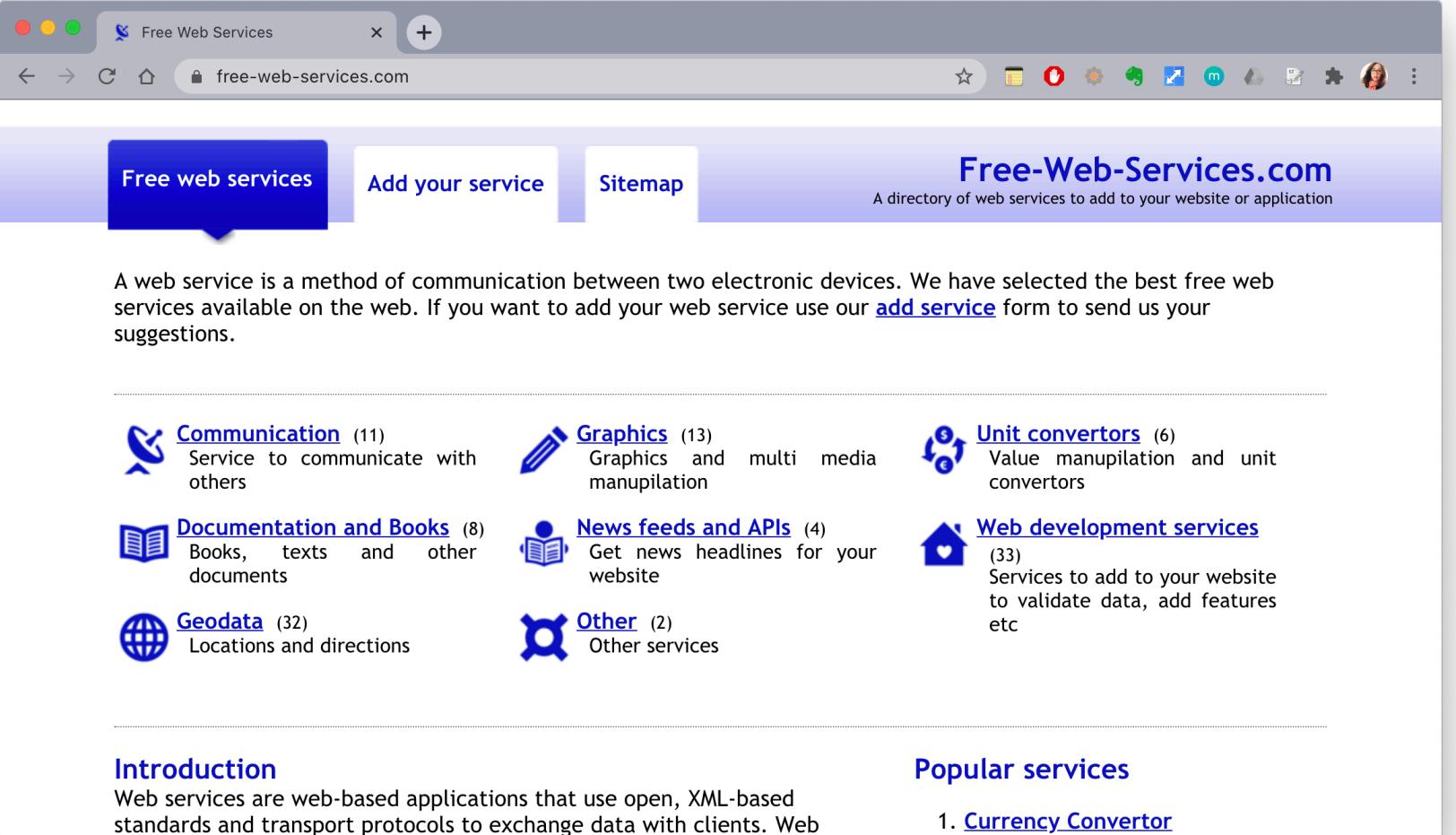
The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Build Beta', 'Browse', 'Explore' (which is selected), and 'Analyze'. A 'Sign in' button is also visible. Below the navigation, the main title 'Public SOAP APIs' is displayed, followed by a subtitle 'List of some public SOAP APIs.' A descriptive text block explains that it's a collection of publicly available SOAP APIs that do not require authentication. To the right, a sidebar titled 'PUBLIC SOAP APIs' lists various categories: Introduction, Numbers, Calculator, Continents, Countries, Currencies, Languages, Book ISBN Numbers, and Temperature. The main content area shows two API endpoints: 'POST NumberToWords' and 'POST NumberToDollars'. Each endpoint has its URL, a brief description, and example code for PHP. The 'NumberToWords' endpoint URL is <https://www.dataaccess.com/webserviceserver/NumberConversion.wsdl>. Its description states: 'Returns the word corresponding to the positive number passed as parameter. Limited to quadrillions.' The example PHP code for 'NumberToWords' is:

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://www.dataaccess.com/webserviceserver/NumberConversion.wsdl');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
    'follow_redirects' => TRUE
));
$request->setHeader(array(
    'Content-Type' => 'text/xml; charset=utf-8'
));
```

The 'NumberToDollars' endpoint URL is <https://www.dataaccess.com/webserviceserver/NumberConversion.wsdl>. Its description states: 'Returns the non-zero dollar amount of the passed number.' The example PHP code for 'NumberToDollars' is:

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
```

<https://www.free-web-services.com/>



The screenshot shows a web browser window with the title bar "Free Web Services" and the URL "free-web-services.com". The page has a blue header bar with the text "Free Web Services", "Add your service", and "Sitemap". To the right, it says "Free-Web-Services.com" and "A directory of web services to add to your website or application". Below the header, there's a main content area with a paragraph about web services and a "add service" form. There are several categories listed with icons: Communication (11), Graphics (13), Unit converters (6), Documentation and Books (8), News feeds and APIs (4), Web development services (33), Geodata (32), and Other (2). At the bottom, there are sections for "Introduction" and "Popular services".

Free web services Add your service Sitemap **Free-Web-Services.com**  
A directory of web services to add to your website or application

A web service is a method of communication between two electronic devices. We have selected the best free web services available on the web. If you want to add your web service use our [add service](#) form to send us your suggestions.

---

 [Communication](#) (11)  
Service to communicate with others

 [Graphics](#) (13)  
Graphics and multi media manipulation

 [Unit converters](#) (6)  
Value manipulation and unit convertors

 [Documentation and Books](#) (8)  
Books, texts and other documents

 [News feeds and APIs](#) (4)  
Get news headlines for your website

 [Web development services](#) (33)  
Services to add to your website to validate data, add features etc

 [Geodata](#) (32)  
Locations and directions

 [Other](#) (2)  
Other services

---

**Introduction**  
Web services are web-based applications that use open, XML-based standards and transport protocols to exchange data with clients. Web

**Popular services**

1. [Currency Convertor](#)

<https://www.programmableweb.com/apis/directory>

The screenshot shows the ProgrammableWeb API Directory homepage. At the top, there's a navigation bar with links for 'API DIRECTORY' and 'API NEWS'. A search bar is prominently displayed, showing 'Search over 23,730 APIs and much more'. Below the search bar, there are links for 'LEARN ABOUT APIs', 'WHAT IS AN API?', 'TUTORIALS', and 'CORONAVIRUS'. On the right side of the header, there's a button for 'ADD APIs & MORE' and social media sharing icons for RSS, Facebook, Twitter, and LinkedIn. The main content area features a large heading 'Search the Largest API Directory on the Web'. Below this, there's a search input field with placeholder text 'Search Over 23,730 APIs' and a 'SEARCH APIs' button. To the right of the search area, there's a box for the 'Coronavirus Developer Resource Center' which includes links to 'COVID-19 APIs, SDKs, coverage, open source code and other related dev resources'. Further down, there's a section titled 'Today in APIs' with a description of 'Latest news about the API economy and newest APIs, delivered daily:' followed by an 'Email Address' input field and a 'SUBSCRIBE' button. At the bottom of the page, there are filters for 'By Category' and 'Include Deprecated APIs', along with columns for 'API Name', 'Description', 'Category', 'Followers', and 'Versions'.

<https://developer.ebay.com/devzone/shopping/docs/Concepts/ShoppingAPIGuide.html>

The screenshot shows a web browser window with the title "Users Guide - eBay Shopping". The URL in the address bar is "developer.ebay.com/devzone/shopping/docs/Concepts/ShoppingAPIGuide.html". The page content includes the eBay Developers Program logo, a navigation bar with links to Home, All Docs, Support, Knowledge Base, Forums, and Resources, and a search bar. Below the navigation bar, there are links to Features Guide, Users Guide, Making a Call, API Reference, Tutorials, and Release Notes. A list of benefits for the Shopping API is provided, followed by a link to "eBay Shopping API Benefits". The main content area contains sections for "What Calls are Supported?", "What Formats are Supported?", and "Shopping API Web Services Protocols". A table lists the supported protocols: Input (URL, XML, JSON, SOAP) and Output (URL, XML, JSON, SOAP). A "Make a call" button is located at the bottom left of the content area.

ebay developers program  
Shopping API

Home All Docs Support Knowledge Base Forums Resources

Search text

Features Guide Users Guide Making a Call API Reference Tutorials Release Notes

- Easy to retrieve live eBay data
- Fast and flexible
- Simple to authenticate (no token required)
- Easy to embed in a web page or app
- Even easier to make affiliate commissions

For more features of the Shopping API, see [eBay Shopping API Benefits](#).

## What Calls are Supported?

See the [API Reference](#).

## What Formats are Supported?

The eBay Shopping API currently supports these formats:

### Shopping API Web Services Protocols

Input	Output
URL	URL, XML, JSON, SOAP
XML	XML
JSON	JSON
SOAP	SOAP

## Make a call



## **How to use these web services**

### **Step 1: Test them using testing tools**



SOAPUI API testing tool (<https://www.soapui.org/downloads/soapui/>)

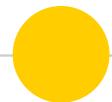


POSTMAN (<https://www.postman.com/downloads/>)

- IDE integrated testing tool (ECLIPSE)

### **Step 2: Write a Web service client**

### **Test the client, then integrate it with your implemented Business Process.**



# Demo

## ○ The Number Conversion Web Service:

<https://www.dataaccess.com/webservicesserver/NumberConversion.wso>

- Provides functions that convert **numbers** into **words** or dollar amounts.

The screenshot shows a web browser window with four tabs open: 'https://www.dataaccess.com/v...', 'Index of /nusoap/myservices', 'localhost/mywebservices/Data...', and 'Number Conversion Service'. The current view is the 'Number Conversion Service' page, which has a blue header bar with the text 'Number Conversion Service'. Below the header, there is descriptive text about the service and a list of available operations.

The Number Conversion Web Service, implemented with Visual DataFlex, provides functions that convert numbers into words or dollar amounts.

The following operations are available. For a formal definition, please review the [Service Description](#).

- [NumberToWords](#)

Returns the word corresponding to the positive number passed as parameter. Limited to quadrillions.

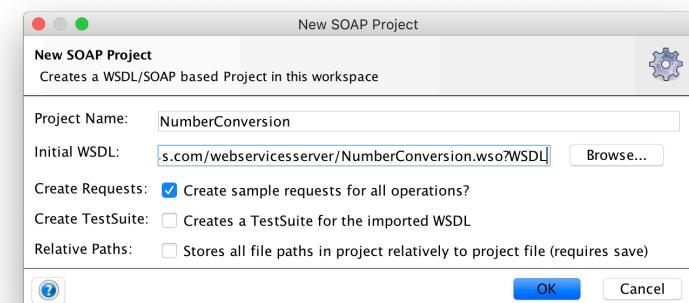
- [NumberToDollars](#)

Returns the non-zero dollar amount of the passed number.



# How to test with SOAPUI ?

- In SoapUI, click **Create SOAP Project** or **select File > New SOAP Project**

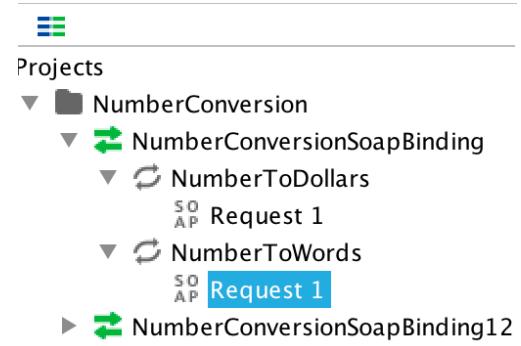


- In the dialog box, specify the following URL in the Initial WSDL field:

<https://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSDL>

- Leave the default settings and click **OK**

- SoapUI will load the specified WSDL and parse its contents into this object model:



Detailed explanation <https://www.Soapui.Org/docs/soap-and-wsdl/working-with-wsdl/>

# SOAPUI ( API testing tool)

The screenshot shows the SOAPUI application interface. At the top, there is a toolbar with icons for Empty, SOAP, REST, Import, Save All, Forum, Trial, Preferences, and Proxy. The Endpoint Explorer tab is selected. Below the toolbar, the Navigator panel shows a project structure under 'NumberConversion' with 'NumberConversionSoapBinding' containing 'NumberToDollars' and 'NumberToWords' requests, and 'NumberConversionSoapBinding12' and 'webservice'. The main workspace displays 'Request 1' for 'https://www.dataaccess.com/webservicesserver/NumberConversion.wsdl'. The request XML is:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Header></soap:Header>
<soap:Body>
<web:NumberToWords>
<web:ubiNum>12000</web:ubiNum>
</web:NumberToWords>
</soap:Body>
</soap:Envelope>
```

The response XML is:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webserviceserver">
<m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>
</m:NumberToWordsResponse>
</soap:Body>
</soap:Envelope>
```

On the left, the Request Properties panel shows the following settings:

Property	Value
Name	Request 1
Description	
Message Size	305
Encoding	UTF-8
Endpoint	https://www.dat...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Ty...	No Authorization
WSS-Password T...	
WSS TimeToLive	
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false

At the bottom, there are tabs for SoapUI log, http log, jetty log, error log, wsrm log, and memory log.

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Endpoint Explorer Search Forum Online Help

**Navigator**

Projects  
 ▾ NumberConversion  
 ▾ NumberConversionSoapBinding  
   ▼ □ NumberToDollars  
     SO AP Request 1  
   ▼ □ NumberToWords  
     SO AP Request 1  
 ▷ □ NumberConversionSoapBinding12  
 ▷ □ webservice

**Request Properties**

Property	Value
Name	Request 1
Description	
Message Size	305
Encoding	UTF-8
Endpoint	https://www.dat...
Timeout	
Bind Address	
Follow Redirects	true
Username	
Password	
Domain	
Authentication Ty...	No Authorization
WSS-Password T...	
WSS TimeToLive	
SSL Keystore	
Skip SOAP Action	false
Enable MTOM	false

**Properties**

**Request 1**

POST https://www.dataaccess.com/webservicesserver/NumberConversion.wso  
 Accept-Encoding: gzip,deflate  
 Content-Type: text/xml; charset=UTF-8  
 SOAPAction: ""  
 Content-Length: 305  
 Host: www.dataaccess.com  
 Connection: Keep-Alive  
 User-Agent: Apache-HttpClient/4.5.5 (java/12.0.1)

<soapenv:Envelope  
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"  
 xmlns:web="http://www.dataaccess.com/webservicesserver/  
   <soapenv:Header/>  
   <soapenv:Body>  
     <web:NumberToWords>  
       <web:ubiNum>12000</web:ubiNum>  
     </web:NumberToWords>  
   </soapenv:Body>  
</soapenv:Envelope>

**Raw**

HTTP/1.1 200 OK  
 Cache-Control: private, max-age=0  
 Content-Type: text/xml; charset=utf-8  
 Content-Encoding: gzip  
 Vary: Accept-Encoding  
 Server: Server  
 Web-Service: DataFlex 19.1  
 Access-Control-Allow-Origin: http://www.dataaccess.com  
 Access-Control-Allow-Methods: GET, POST  
 Access-Control-Allow-Headers: content-type  
 Access-Control-Allow-Credentials: true  
 Strict-Transport-Security: max-age=31536000  
 Date: Thu, 15 Oct 2020 10:38:17 GMT  
 Content-Length: 315

<?xml version="1.0" encoding="utf-8"?>  
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope">  
<soap:Body>  
  <m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/NumberConversion.wso">  
    <m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>  
  </m:NumberToWordsResponse>  
</soap:Body>  
</soap:Envelope>

**Logs**

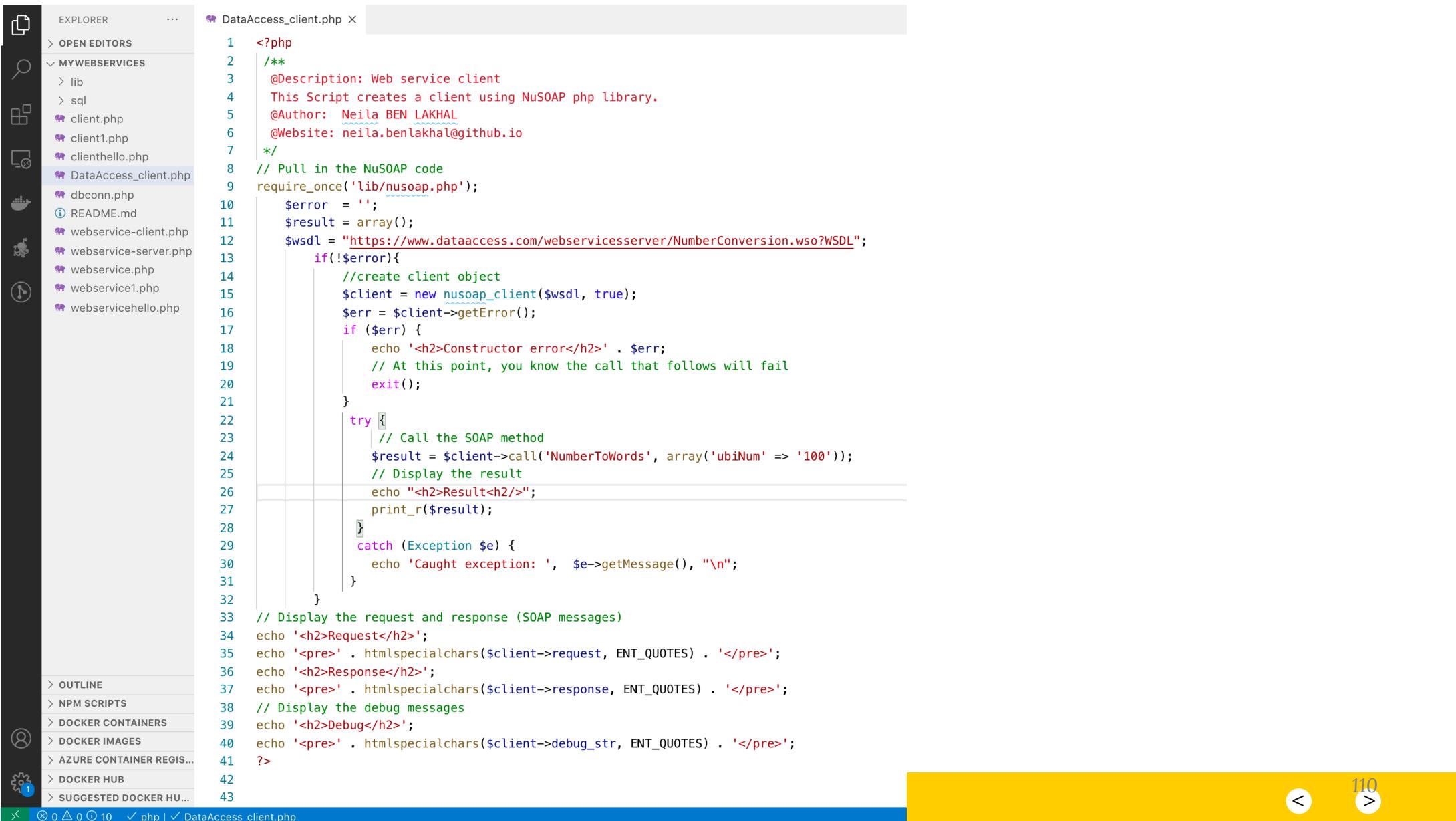
Thu Oct 15 11:37:46 CET 2020:DEBUG: << "Content-Length: 315[\r][\n]"  
Thu Oct 15 11:37:46 CET 2020:DEBUG: << "[\r][\n]"  
Thu Oct 15 11:37:46 CET 2020:DEBUG: << "[0x1f][0x8b][0x8][0x0][0x0][0x0][0x0][0x4][0x0][0xed][0xbd][0x7][0x1c][0x96]%"&/m[0xca][J[0xf5][0xd7][0xe]

SoapUI log http log jetty log error log wsrm log memory log

Inspector

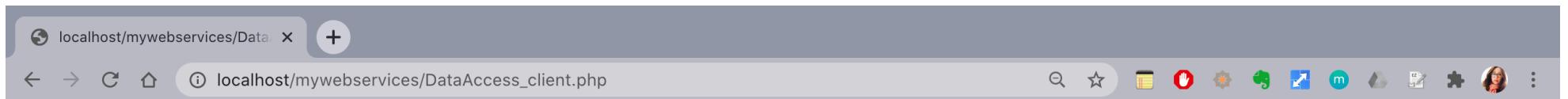
## Step 2

- Write a client
- We will use **NUSOAP library** to write a .php client



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists several files under 'MYWEBSERVICES' and other sections like 'OUTLINE' and 'NPM SCRIPTS'. The main area displays a PHP script named 'DataAccess\_client.php'. The code uses the NuSOAP library to call a WSDL endpoint for number conversion. It includes comments explaining the purpose and structure of the code.

```
1 <?php
2 /**
3  * @Description: Web service client
4  * This Script creates a client using NuSOAP php library.
5  * @Author: Neila BEN LAKHAL
6  * @Website: neila.benlakhal@github.io
7 */
8 // Pull in the NuSOAP code
9 require_once('lib/nusoap.php');
10 $error = '';
11 $result = array();
12 $wsdl = "https://www.dataaccess.com/webservicesserver/NumberConversion.wso?WSDL";
13 if(!$error){
14     //create client object
15     $client = new nusoap_client($wsdl, true);
16     $err = $client->getError();
17     if ($err) {
18         echo '<h2>Constructor error</h2>' . $err;
19         // At this point, you know the call that follows will fail
20         exit();
21     }
22     try {
23         // Call the SOAP method
24         $result = $client->call('NumberToWords', array('ubiNum' => '100'));
25         // Display the result
26         echo "<h2>Result</h2>";
27         print_r($result);
28     }
29     catch (Exception $e) {
30         echo 'Caught exception: ', $e->getMessage(), "\n";
31     }
32 }
33 // Display the request and response (SOAP messages)
34 echo '<h2>Request</h2>';
35 echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
36 echo '<h2>Response</h2>';
37 echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';
38 // Display the debug messages
39 echo '<h2>Debug</h2>';
40 echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';
41 ?>
42
43
```



## Result

Array ( [NumberToWordsResult] => one hundred )

## Request

```
POST /webservicesserver/NumberConversion.wso HTTP/1.0
Host: www.dataaccess.com
User-Agent: NuSOAP/0.9.5 (1.123)
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: ""
Content-Length: 468

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"

```

## Response

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Length: 344
Content-Type: text/xml; charset=utf-8
Server: Server
Web-Service: DataFlex 19.1
Access-Control-Allow-Origin: http://www.dataaccess.com
Access-Control-Allow-Methods: GET, POST
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Credentials: true
Strict-Transport-Security: max-age=31536000
Date: Thu, 15 Oct 2020 21:08:08 GMT

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/">
      <m:NumberToWordsResult>one hundred </m:NumberToWordsResult>
    </m:NumberToWordsResponse>
  </soap:Body>
</soap:Envelope>
```

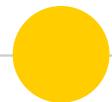
## SOAP Specification



## What is SOAP ?

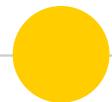
SOAP Version 1.2 (SOAP) is a lightweight protocol intended for exchange of structured information(XML) in a decentralized, distributed environment.





# What is SOAP?

- In Web services, **service requesters** and **service providers** communicate with each other in the form of XML messages. If you would like your software to use a web service, it must send a request message and it should receive a response. These messages conform to **SOAP** :
- SOAP: a **standard** developed initially in 1998.
  - Originally standing for a **Simple Object Access Protocol**.
  - Then, was changed to **SOA Protocol**.
  - Later, both acronyms was dropped
- Microsoft and IBM initiative.
  - **SOAP 1.1, W3C submitted in 2000.**
  - **SOAP 1.2, W3C recommendation 2003.**



# La naissance de SOAP

## ○ L'idée en elle-même n'était pas nouvelle :

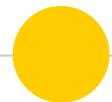
- Fortement inspirée de COM Internet Services (CIS) qui supportaient l'acheminement des appels DCOM à travers le protocole TCP utilisant le port 80
- Appel RPC d'une procédure distante placé sur une machine Windows.
- Dans le temps c'était une idée ingénieuse qui a permis la survie de plusieurs projets freinée par la mise en place de firewalls.

## ○ Du moment que DCOM le permettaient, pourquoi SOAP ?

- La complexité du déploiement d'une application DCOM
- La complexité de l'outil de configuration dcomcfg.exe
- Très difficile de faire en sorte que les appels DCOM contournent les firewalls.

(excerpt from :Building Web Applications with ADO.NET and XML Web Services)

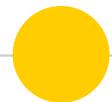
What makes these new Web services interesting is SOAP. SOAP is just a protocol specification that defines how to invoke methods on servers, typically over HTTP. It provides an extensible way for applications to communicate using simple XML messages over the Web. Regardless of operating system, brand of parser, or language, the messages themselves are the standard. The SOAP specification goes on to mandate a specific XML vocabulary that defines parameters, return values, and exceptions. This means that two software developers with completely different backgrounds, platform preference, and love for software companies can coexist peacefully in the world — and work together efficiently. Clients written in Visual Basic .NET can easily invoke services running CORBA on Unix boxes, while Windows Scripting Host (WSH) clients can call mainframe code and access DB2. Even your Macintosh boxes can access Perl Web services running on Red Hat Linux. The list goes on and on.



# Rôles de SOAP

● Permettre à des méthodes de toute nature – sur n'importe quelle plateforme, dans n'importe quel langage – de communiquer.

- La spécification SOAP est disponible dans 60+ langages sur 20+ plateformes.
- Assurer la communication entre applications d'une même entreprise (Intranet)
- Assurer les échanges inter-entreprises
  - Ne pas imposer une API ou un runtime
  - Ne pas imposer un modèle de programmation
  - Permettre l'utilisation de plusieurs modèles conjointement



# Fonctionnement de SOAP

Un message SOAP est envoyé d'un expéditeur (client) à un destinataire (serveur), directement ou via des intermédiaires.

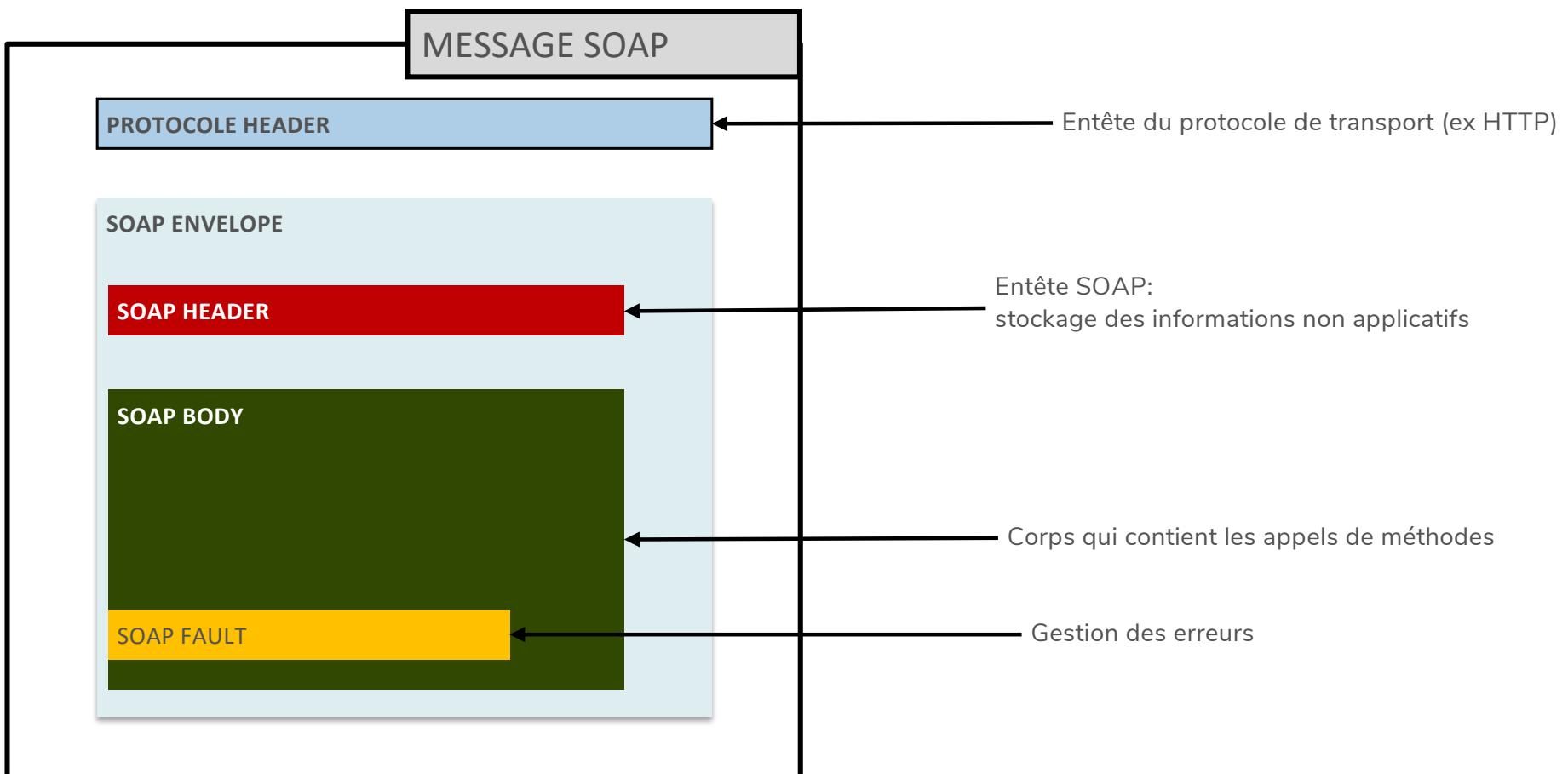
- Fonctionnement côté Client (simplifié)
  - Ouverture d'une connexion HTTP et envoie d'une requête
  - Requête SOAP est un document XML décrivant :
    - une méthode à invoquer sur une machine distante
    - Les paramètres de la méthode
- Fonctionnement côté Serveur (simplifié)
  - Récupère la requête
  - Exécute la méthode avec les paramètres
  - Renvoie une réponse SOAP (document XML) au client



## Structure d'un message SOAP 1/2

- Un message SOAP est un document XML constitué d'une **enveloppe** composée de 2 parties :
  - Enveloppe / **<envelope>** (Élément racine obligatoire), il contient:
    - Entête / **<header>** ( Élément optionnel)
      - Contient des entrées non applicatives (Ex: Transactions, sessions, ...)
      - Peuvent être adressées à certains intermédiaires
      - Leur traitement peut être marqués comme obligatoire
    - Corps / **<body>** (Élément obligatoire)
      - Contient les entrées applicatives du message :Nom d'une méthode, valeurs des paramètres, valeur de retour.
- Comme les messages SOAP ne peuvent pas être envoyés en lots, l'enveloppe contient un seul message,
- La requête et la réponse ont la même structure .

## Structure d'un message SOAP 2/2





## SOAP request/response : sample

```
Request Response

1<soapenv:Envelope
2  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3  xmlns:web="http://www.dataaccess.com/webservicesserver/"
4    <soapenv:Header/>
5<soapenv:Body>
6  <web:NumberToWords>
7    <web:ubiNum>12000</web:ubiNum>
8  </web:NumberToWords>
9</soapenv:Body>
10</soapenv:Envelope>
```



The screenshot shows a web browser window titled "Request 1". The address bar contains the URL "https://www.dataaccess.com/webservicesserver/NumberConversion.wso". Below the address bar is a toolbar with various icons. The main content area of the browser displays a SOAP response. At the top of this area are two tabs: "Request" and "Response", with "Response" being the active tab. The response XML is shown in a code editor-like format:

```
Request Response

1<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
2<soap:Body>
3<m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/">
4  <m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>
5</m:NumberToWordsResponse>
6</soap:Body>
7</soap:Envelope>
```



## SOAP envelope 1/3

### ◉ Règles de syntaxe - Un message SOAP :

- DOIT être codé en utilisant le langage XML
- La spécification impose que les balises, les sous balises ainsi que tous les attributs contenus dans l'enveloppe SOAP doivent **explicitement** être associés à un **namespace**, de manière à supprimer toute ambiguïté.
- Ces **namespaces** doivent être définis correctement.
- NE DOIT PAS contenir ni une référence DTD ni des instructions de traitement XML.
- Toutes les balises XML associées à SOAP doivent avoir :
- Le préfixe **soap:** ou **soap-env:** pour SOAP 1.1.
- Le préfixe **env:** pour SOAP 1.2.



## SOAP envelope 2/3

### SOAP envelope

- C'est l'élément de racine du message SOAP.
- L'enveloppe SOAP sert de conteneur aux autres éléments du message SOAP, elle est définie au début par la balise `<soap:Envelope>` et se termine par la balise `</soap:Envelope>`.
- Le prologue XML contient une déclaration XML `<?xml version="1.0" encoding="UTF-8"?>` spécifiant la version de XML et l'encodage des caractères du message XML.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
    ...
</soap:Envelope>
```



## SOAP envelope 3/3

Par convention, la spécification SOAP définit plusieurs **namespaces** :

- **soap-env:** ou **soap:** comme préfixe associé à l'URI <http://schemas.xmlsoap.org/soap/envelope/> pour définir le **namespace** de l'enveloppe dans la version 1.1.
- **env:** comme préfixe associé à l'URI <http://www.w3.org/2003/05/soap-envelope/> pour définir le **namespace** de l'enveloppe dans la version 1.2.
- L'attribut **encodingStyle** associé à l'URI <http://www.w3.org/2001/12/soap-encoding> pour la définition des formats de types de données.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
    xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding">
    ...
</soap:Envelope>
```



## SOAP header 1/3

### SOAP header :

- L'en-tête est un élément facultatif dans un message SOAP.
- Si présent, il doit être le premier élément dans l'enveloppe SOAP.
- Il sert à communiquer des informations pour :
  - Authentifier l'émetteur,
  - Définir des règles de sécurité et des algorithmes de chiffrement à utiliser pour la lecture du message,
  - Donner le contexte d'une transaction,
  - Etc.
- La spécification SOAP n'indique pas comment ajouter ce type d'information dans l'entête.



## SOAP Header 2/3

● L'en-tête d'un message SOAP commence avec la balise `<soap:Header>` et se termine avec la balise `</soap:Header>`,

- Note : on peut aussi trouver: `<soap-env:Header></soap-env:Header>`.

● 2 attributs associés à l'en-tête SOAP :

- `soap:mustUnderstand` : cet attribut prend la valeur 1 ou 0:
  - La valeur 1 : le récepteur doit reconnaître l'information présente dans l'en-tête, son traitement est obligatoire.
  - La valeur 0 : l'en-tête peut être ignoré par le récepteur.
- `soap:actor` : cet attribut identifie le destinataire de l'élément fils. Comme un message SOAP peut traverser plusieurs composants avant d'atteindre le destinataire final du message, il peut être nécessaire d'adresser des informations aux composants intermédiaires qui composent le chemin vers le destinataire final. C'est par l'intermédiaire de cet attribut que cela est rendu possible, en lui donnant comme valeur une URL qui correspond à ou aux destinataires de l'élément.

Note : En l'absence de cet attribut, l'élément fils est à destination du destinataire final du message SOAP.

### ● L'attribut mustUnderstand

- La valeur 1 : le récepteur doit reconnaître l'information présente dans l'en-tête, son traitement est obligatoire.

### ● Exemple :

```
<soap-env:header>
    <t:transaction xmlns:t="some-uri"
        soap-env:mustunderstand="1">
        5
    </t:transaction>
</soap-env:header>
```



## **SOAP body**

- Le corps SOAP <soap:Body> est un élément obligatoire dans le message SOAP.
  - Il contient l'information destinée au receveur.
- Le contenu du corps SOAP est utilisé pour spécifier un appel de méthode à un ordinateur distant avec les valeurs de paramètres.
  - Par exemple, la demande du solde d'un compte bancaire.
- Le corps du message SOAP commence avec la balise <soap:Body> et se termine avec la balise </soap:Body>.

## SOAP body (request)

Exemple : L'extrait suivant représente un corps SOAP qui fait appel à une méthode appelée **NumberToWords()**:

- L'élément `<NumberToWords>` fournit le nom de la méthode à appeler.
- L'élément `<ubiNum>` est un paramètre qui est passé dans la méthode.



The screenshot shows a window with two tabs at the top: "Request" (highlighted in blue) and "Response". The "Request" tab contains the following XML code:

```
1<soapenv:Envelope
2  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
3  xmlns:web="http://www.dataaccess.com/webservicesserver/"
4    <soapenv:Header/>
5<soapenv:Body>
6  <web:NumberToWords>
7    <web:ubiNum>12000</web:ubiNum>
8  </web:NumberToWords>
9</soapenv:Body>
10</soapenv:Envelope>
```



## SOAP body (réponse)

Exemple : L'extrait suivant représente un corps SOAP qui est la réponse à l'appel à la méthode appelée **NumberToWords()**:

- L'élément **<NumberToWordsResponse>** fournit le nom de la méthode qui a été appelée.
- L'élément **< NumberToWordsResult>** contient la valeur de retour.

The screenshot shows a SOAP UI tool window. The title bar says "SOAP Request 1". The address bar contains the URL "https://www.dataaccess.com/webservicesserver/NumberConversion.wso". Below the address bar, there are two tabs: "Request" and "Response", with "Response" being the active tab. The response content is displayed in a code editor-like area with line numbers:

```
1<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
2  <soap:Body>
3    <m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/">
4      <m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>
5    </m:NumberToWordsResponse>
6  </soap:Body>
7</soap:Envelope>
```



## SOAP body

- Exemple : L'extrait suivant représente un corps SOAP qui fait appel à une méthode appelée `checkAccountBalance()`:

```
<soap:Body>
  <checkAccountBalance>
    <accountNumber xsi:type="xsd:int">1234567890</accountNumber>
  </checkAccountBalance>
</soap:Body>
```

- L'élément `<checkAccountBalance>` fournit le nom de la méthode à appeler : `checkAccountBalance`.
- L'élément `accountNumber` est un paramètre qui est passé dans la méthode `checkAccountBalance`.
- Les attributs `xsi` et `xsd` définissent les espaces de noms qui vont être utilisés dans le corps du message.
  - La définition de `xsi` permet d'utiliser `xsi:type` dans le corps du message, le `xsd:int` signifie que cette valeur est de type entier. 1234567890 est la valeur donnée au paramètre.



## Body: Syntaxe

○ La partie **Body** encapsule un unique tag de méthode qui porte le nom de la méthode elle-même:

- <ns1:maMethode ... > ou <ns1:maMethodeReponse ... > (le même nom suivi du mot « Response » dans le cas du message de réponse).
- Exp: Méthode GetPrice() → élément <m:GetPrice> dans la requête et <m:GetPriceResponse> dans la réponse.

○ Le tag de la méthode reçoit typiquement le namespace correspondant au nom du WS, pour assurer l'unicité.

- Exp: <m:GetPrice>, <m:GetPriceResponse> ont le namespace **m**.

○ Le tag de méthode encapsule à son tour un certain nombre de paramètres, tel que le tag <param1 ... > dans l'enveloppe d'une requête. En présence de plusieurs paramètres, leur ordre d'apparition n'a pas d'importance.

○ Dans le message de réponse, on ne trouve qu'un seul tag de paramètre représentant la valeur de retour de la méthode (généralement appelé <return>).

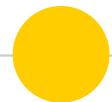


## Types de données /encoding

- Les messages SOAP peuvent supporter des encodages de données complexe et c'est l'une des forces de ce protocole:

Exemple : Le dialogue résultant de l'appel de **getEmployeeDetails** : Le client envoie **un entier (l'identification de l'employé)** et reçoit un tableau de chaînes de caractères à 2 éléments contenant : le nom de l'employé et le numéro de téléphone

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<soap-env:envelope ...xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/1999/XMLSchema">
    <soap-env:body>
        <ns1:getemployeeDetails xmlns:ns1="urn:mysoapservices">
            <param1 xsi:type="xsd:int">1016577</param1>
        </ns1:getemployeeDetails>
    </soap-env:body>
</soap-env:envelope>
```

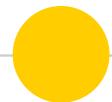


## Types de données /encoding

- Le client reçoit un tableau de chaînes de caractères à deux éléments contenant : le nom de l'employé et le numéro de téléphone
- Dans la balise <return> du message de réponse, nous avons le type de la structure complexe (le tableau de chaines), à savoir :

- xsi:type="ns2:Array" ns2:arrayType="xsd:string[2]"

```
<SOAP-ENV:Envelope ...>
  <SOAP-ENV:Body><ns1:getEmployeeDetailsResponse xmlns:ns1="urn:MySoapServices"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/">
        xsi:type="ns2:Array"
        ns2:arrayType="xsd:string[2]">
          <item xsi:type="xsd:string">Bill Posters</item>
          <item xsi:type="xsd:string">+1-212-7370194</item>
        </return>
      </ns1:getEmployeeDetailsResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```



## SOAP <Fault>

○ L'élément optionnel SOAP <**Fault**> est utilisé pour indiquer les messages d'erreur.

- Si un élément <**Fault**> est présent, il doit apparaître comme un élément enfant de <**Body**> .
- Un élément <**Fault**> ne peut apparaître qu'une fois dans un message SOAP.

élément	Description
< <b>faultcode</b> >	Contient un code qui identifie l'erreur. SOAP définit 4 codes d'erreur: <ul style="list-style-type: none"><li>•VersionMismatch (namespace du bloc Envelope non valide),</li><li>•MustUnderstand (un élément fils du bloc Header qui devait absolument être compris ne l'a pas été),</li><li>•Client (message mal formatée ou non valide),</li><li>•Server (problème lors du traitement du message).</li></ul>
< <b>faultstring</b> >	Contient un texte décrivant brièvement l'erreur pour un humain.
< <b>faultactor</b> >	Contient le composant (i.e., son URL) qui a généré l'erreur.
< <b>detail</b> >	Contient des informations spécifiques à l'application et concernant l'erreur.

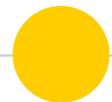


## Exemple d'un bloc <fault>

Request

Response

```
1<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
2  <soap:Body>
3    <soap:Fault>
4      <faultcode>soap:Server</faultcode>
5      <faultstring>Request parse error: unterminated start tag 'web:ubiNum'</faultstring>
6      <detail/>
7    </soap:Fault>
8  </soap:Body>
9 </soap:Envelope>
```



## Exemple Requête HTTP/SOAP

- SOAP utilise un protocole de transport pour véhiculer les messages SOAP de l'émetteur au récepteur
- Protocoles utilisés : HTTP, SMTP, FTP, ...

Request Response

```
POST https://www.dataaccess.com/webservicesserver/NumberConversion.wso HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml; charset=UTF-8
SOAPAction: ""
Content-Length: 305
Host: www.dataaccess.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.5 (Java/12.0.1)

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.dataaccess.com/webservicesserver/">
  <soapenv:Header/>
  <soapenv:Body>
    <web:NumberToWords>
      <web:ubiNum>12000</web:ubiNum>
    </web:NumberToWords>
  </soapenv:Body>
</soapenv:Envelope>
```

Raw XML

Request Response

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: text/xml; charset=utf-8
Content-Encoding: gzip
Vary: Accept-Encoding
Server: Server
Web-Service: DataFlex 19.1
Access-Control-Allow-Origin: http://www.dataaccess.com
Access-Control-Allow-Methods: GET, POST
Access-Control-Allow-Headers: content-type
Access-Control-Allow-Credentials: true
Strict-Transport-Security: max-age=31536000
Date: Thu, 15 Oct 2020 10:38:17 GMT
Content-Length: 315

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webservicesserver/">
      <m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>
    </m:NumberToWordsResponse>
  </soap:Body>
</soap:Envelope>
```

Raw XML



## Exemple Requête HTTP/SOAP

● Méthode de type **POST** (Envoi de données au service situé à l'URL spécifiée)

● **Content-Type** : renseigne sur le type MIME de l'entité transportée, doit être obligatoirement **text/xml**, pour un message SOAP. À cette valeur suit l'attribut **charset** (encoding).

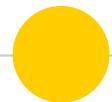
● **SOAPAction** : SOAP introduit dans l'en-tête HTTP POST le nouveau champ de requête **SOAPAction**, pour indiquer la destination du message SOAP.

- La spécification SOAP 1.1 précise que ce champ est obligatoire dans une requête HTTP véhiculant un message SOAP.
- soit contient un URL (qui peut être aussi une chaîne de caractères vide) ;
- soit ne contient aucune valeur (mais le champ est présent quand même).

```
Request Response

POST https://www.dataaccess.com/webservicesserver/NumberConversion.wso HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: text/xml;charset=UTF-8
SOAPAction: ""
Content-Length: 305
Host: www.dataaccess.com
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.5.5 (Java/12.0.1)

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://www.dataaccess.com/webservicesserver/">
<soapenv:Header/>
<soapenv:Body>
<web:NumberToWords>
<web:ubiNum>12000</web:ubiNum>
</web:NumberToWords>
</soapenv:Body>
</soapenv:Envelope>
```



## Exemple Réponse HTTP/SOAP

- La réponse HTTP véhicule les codes de retour HTTP.
- Dans la liaison SOAP/HTTP, les codes 2xx indiquent que le message SOAP a été reçu.
- Si code 500, message en erreur, le corps SOAP doit contenir l'élément <fault>.

Request Response

HTTP/1.1 200 OK  
Cache-Control: private, max-age=0  
Content-Type: text/xml; charset=utf-8  
Content-Encoding: gzip  
Vary: Accept-Encoding  
Server: Server  
Web-Service: DataFlex 19.1  
Access-Control-Allow-Origin: http://www.dataaccess.com  
Access-Control-Allow-Methods: GET, POST  
Access-Control-Allow-Headers: content-type  
Access-Control-Allow-Credentials: true  
Strict-Transport-Security: max-age=31536000  
Date: Thu, 15 Oct 2020 10:38:17 GMT  
Content-Length: 315

XML Raw

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <m:NumberToWordsResponse xmlns:m="http://www.dataaccess.com/webserviceserver/">
      <m:NumberToWordsResult>twelve thousand</m:NumberToWordsResult>
    </m:NumberToWordsResponse>
  </soap:Body>
</soap:Envelope>
```

Request Response

HTTP/1.1 500 Internal Server Error  
Cache-Control: private, max-age=0  
Content-Length: 346  
Content-Type: text/xml; charset=utf-8  
Server: Server  
Web-Service: DataFlex 19.1  
Access-Control-Allow-Origin: http://www.dataaccess.com  
Access-Control-Allow-Methods: GET, POST  
Access-Control-Allow-Headers: content-type  
Access-Control-Allow-Credentials: true  
Strict-Transport-Security: max-age=31536000  
Date: Thu, 15 Oct 2020 20:36:09 GMT  
Connection: close



## Les Styles de message

### Style RPC

- Appels de procédures distants, Degreeé élevé d'automatisation
- Paramètres proches des types des langages de programmation
- <soap:Body> contient un élément avec le nom de la méthode ou de procédure distante invoquée. Cet élément contient à son tour un élément pour chaque paramètre de cette procédure.

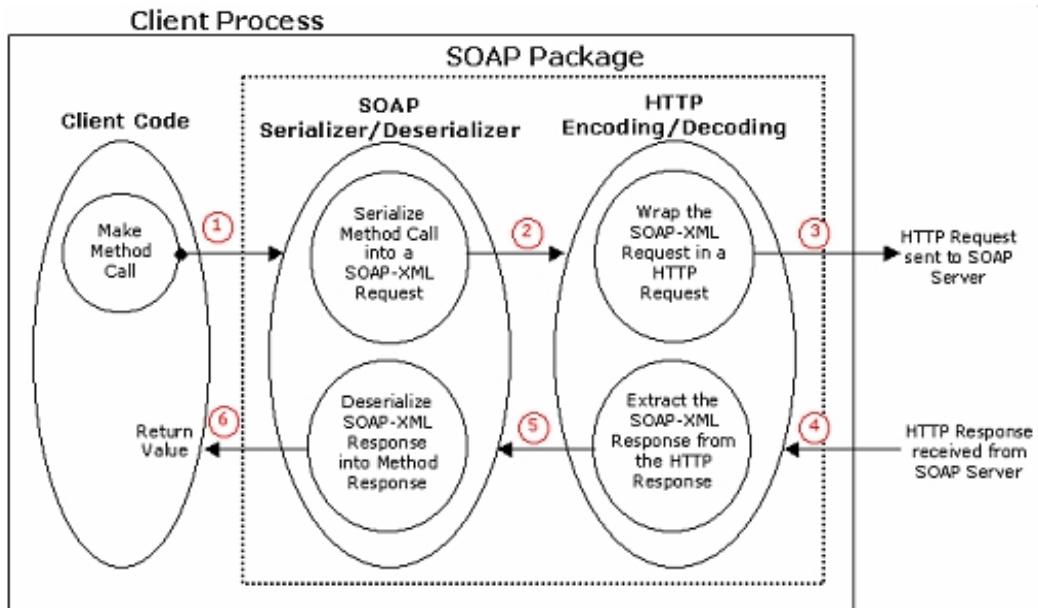
### Style DOC (Document)

- Echanges de messages conformes à des schémas arbitraires (Ex: Demande d'achat).
- Plus d'expressivité , Encouragé par .Net
- < soap:Body> contient un ou plusieurs éléments enfant appelé parties. Pas de règles de formatage SOAP pour ce que le <soap:Body> contient, l'expéditeur et le destinataire se mettent d'accord sur le contenu.



## Architecture technique côté client

- Les messages envoyés au serveur seront des requêtes SOAP-XML enveloppées dans des requêtes HTTP.
- De même, les réponses du serveur sont des réponses HTTP qui renferment des réponses SOAP-XML.
- Du côté client, pour ne pas prendre en charge la sérialisation SOAP et l'encodage HTTP, nous utilisons un package SOAP spécifique.
- Nous invoquons ensuite le service, simplement en invoquant la méthode appropriée du package SOAP (typiquement en spécifiant l'URL du service, le nom du service et tous les paramètres requis).
- **Le premier travail d'un package est de sérialiser l'invocation de ce service en requête SOAP. Il doit ensuite encoder ce message dans une requête HTTP et l'envoyer à l'URL spécifiée.**



- Le code client crée un appel de service en invoquant la méthode appropriée du package SOAP (1).
- Le serialiseur SOAP du package SOAP convertit cette invocation en requête SOAP et l'envoie à l'encodeur HTTP (2).
- L'encodeur HTTP enveloppe le message SOAP dans une requête HTTP et l'envoie au serveur SOAP (3).
- La réponse est reçue par le module d'encodage/décodage HTTP du serveur SOAP(4);
- ce module décode et extrait la réponse SOAP qui la remet au deserialiseur SOAP (5).
- Le deserialiseur SOAP deserialise le message et passe le résultat au code client (6) comme valeur de retour de l'invocation originale (1).



## Architecture technique côté serveur

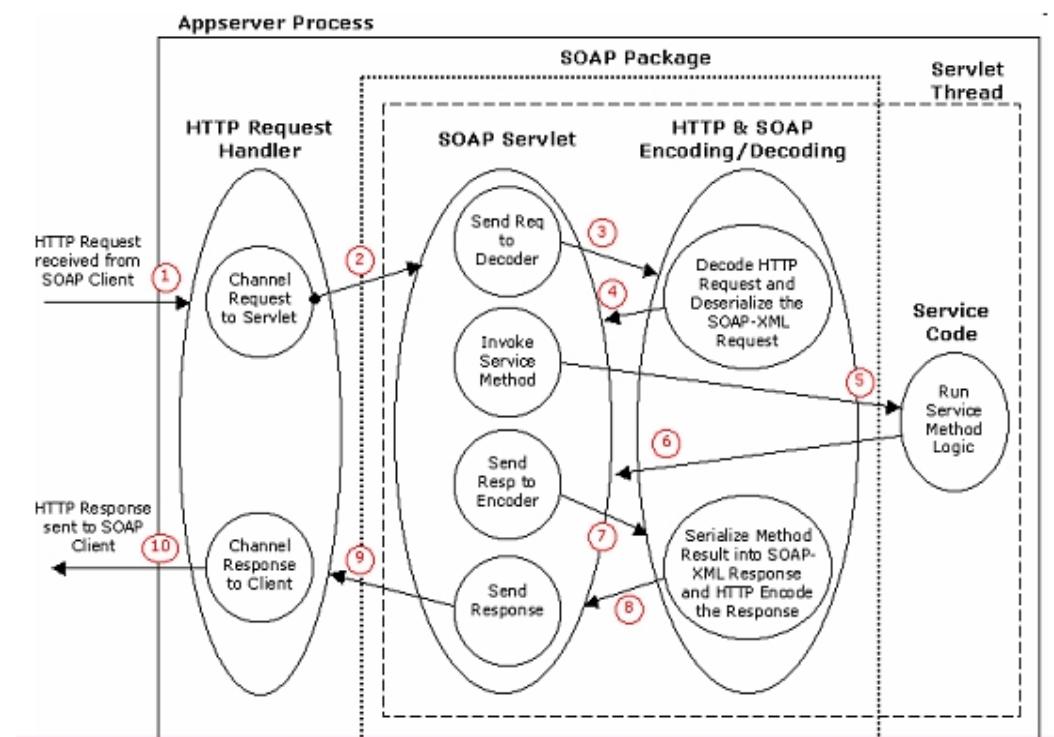
- Il est important de se rappeler que le choix du langage, de plate-forme et de package SOAP pour consommer des services web du côté client est **entièrement indépendant** du langage, de la plate-forme et du package SOAP utilisés par le côté serveur pour fournir des services web.
- De cette façon, le même service web basé sur SOAP (déployé par exemple sur UNIX, écrit en Java et exploitant Apache SOAP pour Java) peut être consommé par tout type de client écrit pour n'importe quelle plate-forme, dans n'importe quel langage, exploitant n'importe quel package SOAP applicable à cette combinaison langage/plate-forme. C'est l'une des grandes forces de la technologie SOAP.



## Architecture technique côté serveur

- Nous avons besoin d'un process "listener" (Le Listener est le process serveur qui est en attente de connexion client).
- Nous avons également besoin d'une implémentation du service lui-même. A part cela, nous nous reposons sur un package SOAP de la même façon que du côté client.
- Le listener est souvent implémenté au travers d'une servlet qui s'exécute comme une application web sur un appserver (comme c'est le cas lorsque nous utilisons Apache SOAP du côté serveur).
- Le serveur sera configuré pour passer toutes les requêtes destinées à une certaine URL (l'URL du service SOAP) à une servlet particulier (appelons-la servlet SOAP).
- Le travail de la servlet SOAP est d'extraire le message XML-SOAP de la requête HTTP, de le déserialiser (de ce fait, séparer le nom de la méthode et les paramètres fournis), et d'invoquer la méthode du service en conséquence. Le résultat de la méthode est alors sérialisé, encodé HTTP et renvoyé au demandeur.

Le serveur d'application web reçoit une requête HTTP du Client SOAP destinée à l'URL de service SOAP(1) et, en conséquence de quoi, le passe à la servlet SOAP (2). La servlet SOAP utilise les fonctionnalités de décodage SOAP et HTTP incluses dans le package pour extraire les détails de l'appel des services (3 et 4), c'est-à-dire le nom et les paramètres de la méthode. Une fois muni de ceux-ci, la servlet SOAP peut invoquer la méthode (5 et 6), encoder la réponse (7 et 8) et fournir la réponse HTTP au handler de requêtes HTTP (9), qui à son tour, répond au client SOAP (10)



## Un peu d'historique

- Septembre 1999 : SOAP 0.9  
Spécifications par Microsoft et DevelopMentor
- Décembre 1999 : SOAP 1.0  
Soumission des spécifications à l'IETF  
Association de UserLand
- Mai 2000 : SOAP 1.1 – Soumission au W3C  
Nombreuses associations : IBM, HP, Lotus, Compaq, Intel ...  
XIDL : rapprochement de Corba
- Septembre 2000
- Groupe de travail W3C pour la standardisation de SOAP  
Corba/Soap Interworking RFP => SCOP

## Implémentations de SOAP

● Voir <http://www.Soapware.org> pour une liste exhaustive

JAXM, JAX-RPC

Apache SOAP 2.2

Apache AXIS, AXIS2

SoapRMI

SOAPDirect

InstantXML

.Net

**NUSOAP**

**PHP5 SOAP API**

Sun metro/ Glassfish

...