



# Ambientes de Desenvolvimento Avançados

<http://www.dei.issep.ipp.pt/~jtavares/ADAV/ADAV.htm>

## Aula 18

### Engenharia Informática

2006/2007

José António Tavares  
jrt@isep.ipp.pt

1



## Web services standards



2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

2

## Antes disso ...

---

### WS em Java

Java

+

Tomcat (servidor WEB)  
<http://tomcat.apache.org/>

+

Axis  
<http://ws.apache.org/axis/>

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

3

## Consumo de WS em Java

---

### 1. WSDL2Java

```
java org.apache.axis.wsdl.WSDL2Java  
http://localhost:8080/axis/CalcService.jws?WSDL
```

```
java org.apache.axis.wsdl.WSDL2Java  
http://localhost/MyFirstWebService/service1.asmx?WSDL
```

### 2. Escrever a aplicação cliente usando as classes proxy geradas ...

Exemplo ...

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

4

```

import local host. *;
import org. apache. axis. AxisFault;
public class CalcClient
{
    public static void main(String[] args) {
        try { // Make a service
            CalcServiceService service = new CalcServiceServiceLocator();
            CalcService port = service.getCalcService();
            // Make the actual calls to the three methods
            double Csum = port.sum(21.00, 23.10);
            double Cdif = port.diff(21.00, 0.10);
            double Cmul = port.mul(23.10, 0.10);
            // Output the results
            System.out.println("21.00 + 23.10 = " + Csum);
            System.out.println("21.00 - 0.10 = " + Cdif);
            System.out.println("23.10 x 0.10 = " + Cmul);
        }
        catch (AxisFault af) {
            System.err.println("An Axis Fault occurred: " + af);
        }
        catch (Exception e) {
            System.err.println("Exception caught: " + e);
        }
    }
}

```



**Web services standards**

Web Services

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

6

## Wiring Standards

- Wiring is the fabric that connects electrical components;
- Plumbing is essentially the same whether it is for gas or water systems;
- Standards on this level are important for components to be connectable at all;
- Software component wiring and the emerging world of XML-based standards – promise of an universal adapter.

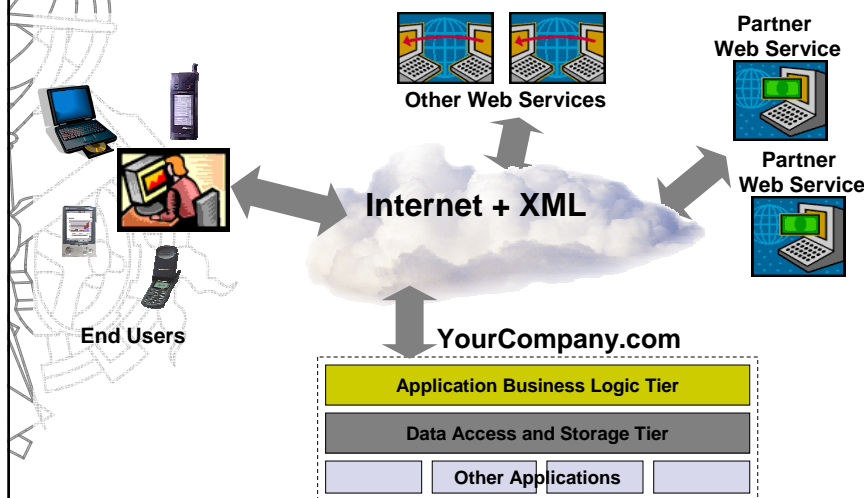
2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

7

## Web Services Overview

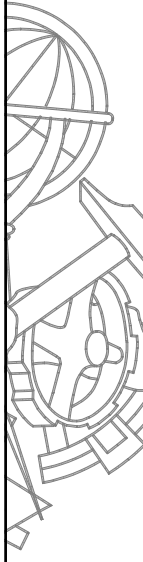
### Application Model



2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

8

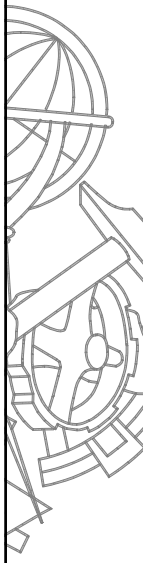


## Elements of the Web Service

- Description of the Service
  - UDDI
  - WSDL
  - WSCL
- The Wiring
  - SOAP
- The Host
  - Web Server
- The Actual Service
  - The Service Code

2006/2007 ADAV

Ambientes de Desenvolvimento Avançados 9



## Web Server Issues

- Handles the requests and invokes the functions
- Since HTTP is used, SOAP invocations can be as a result of a
  - Get request
  - Post request
  - Or a SOAP request (mime type SOAP)
- IIS 5.0 (.NET implementation)
- Apache (SOAP Module Built by IBM)
- Tomcat (Java Server)

2006/2007 ADAV

Ambientes de Desenvolvimento Avançados 10

## Hosting Environments for Web Services

---

- Microsoft .NET
- IBM Websphere
- Apache Axis - <http://xml.apache.org/axis>

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

11

## Web Services Standards Stack

---

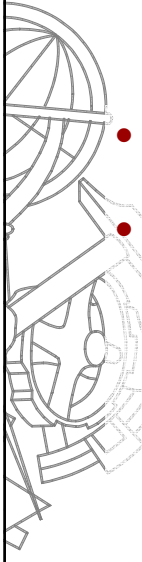
<b>Directory: Publish &amp; Find Services:</b>	<b>UDDI</b>
<b>Description: Formal Service Descriptions:</b>	<b>WSDL</b>
<b>Wire Format: Service Interactions:</b>	<b>SOAP</b>
<b>Universal Data Format:</b>	<b>XML</b>
<b>Ubiquitous Communications:</b>	<b>Internet</b>

**Simple, Open, Broad Industry Support**

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

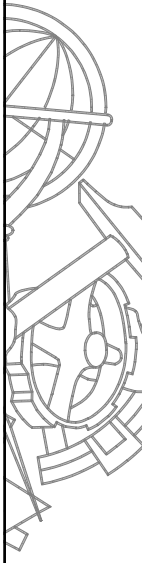
12



## Examining HTTP

- HTTP is a protocol for requesting and receiving a file from a Web server.
- Your Web browser uses HTTP to access Web pages. How does this work?
  1. You enter <http://www.yahoo.com/games.htm> into your Web browser
  2. Your Web browser makes a request to Yahoo's Web server for the file games.htm.
  3. The Web server then sends back this file and your Web browser renders the HTML in the file, displaying it in the browser.


2006/2007 ADAV 13  
Ambientes de Desenvolvimento Avançados



## HTTP Requests

- An HTTP request, in its simplest form, looks like the following:  
`GET /file/ename HTTP/1.1`
- So, if we wanted to get [www.yahoo.com/games.htm](http://www.yahoo.com/games.htm), the Web browser makes a connection to Yahoo's Web server on (typically) port 80 and then sends the command:  
`GET /games.htm HTTP 1.1`


2006/2007 ADAV 14  
Ambientes de Desenvolvimento Avançados



## HTTP Response

- When the Web server receives the request, it replies with an *HTTP response*, which has the following format:  
HTTP/1.1 *STATUS\_CODE*  
Content-type: *content\_type*  
*Contents of requested file...*

2006/2007 ADAV 15  
Ambientes de Desenvolvimento Avançados



## HTTP Response

- For example, the Yahoo Web server, after receiving the HTTP request for /games.htm, might respond with:  
HTTP/1.1 200 OK  
Content-type: text/html  
<html >  
<head><title>Yahoo  
Games! </title></head>  
<body>  
...

2006/2007 ADAV 16  
Ambientes de Desenvolvimento Avançados



## HTTP Request/Response



2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

17

## Web Services Standards Stack

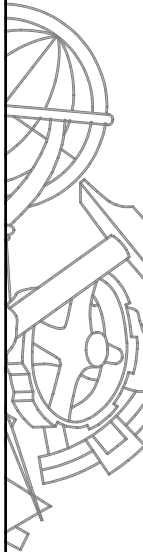
<b>Directory: Publish &amp; Find Services:</b>	<b>UDDI</b>
<b>Description: Formal Service Descriptions:</b>	<b>WSDL</b>
<b>Wire Format: Service Interactions:</b>	<b>SOAP</b>
<b>Universal Data Format:</b>	<b>XML</b>
<b>Ubiquitous Communications:</b>	<b>Internet</b>

**Simple, Open, Broad Industry Support**

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

18



## Introduction to XML

---

- XML stands for eXtensible Markup Language.  
<http://www.w3.org/TR/REC-xml/>
- XML is a way of specifying *self-describing* data in a *human-readable* format.
- XML's structure is like that of HTML – it consists of nested tags, or *elements*. Elements can contain text data, attributes, and other elements.
- XML is ideal for describing *hierarchical* data.



## The Components of XML

---

- XML document can be comprised of the following components:
  - A Root Element (*required*)
  - Elements
  - Attributes

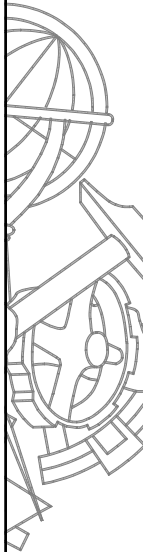
## An Example XML Document

```
<?xml version="1.0" encoding="UTF-8" ?>
<books>
  <!-- Each book is represented by a <book> element -->
  <book price="34.95">
    <title>Teach Yourself ASP 3.0 in 21 Days</title>
    <authors>
      <author>Mitchell</author>
      <author>Atkinson</author>
    </authors>
    <year>1999</year>
  </book>
</books>
```

## An Example XML Document

- All XML Documents must have precisely one *root element*.
- The first element in the XML document is referred to as the root element.
- For the books example, the root element is `<books>`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<books>
  ...
</books>
```



## An Example XML Document

---

- XML documents can contain an arbitrary number of *elements* within the root element. Elements have a *name*, can have an arbitrary number of attributes, and can contain either text content or further elements.
- The generic syntax of an element is given as:

```
<elementName attribute1="value1" ...  
  attributeN="valueN" >  
  text content or further elements  
</elementName>
```

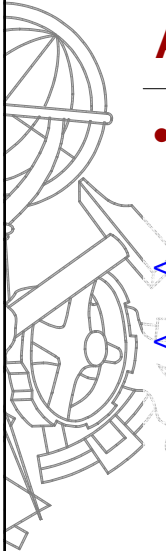


## An Example XML Document

---

- For example, in the books XML document, the <book> element contains an attribute named price and three *child elements*: <title>, <authors>, and <year>.

```
<book price="34.95">  
  <title>...</title>  
  <authors>  
    ...  
  </authors>  
  <year>...</year>  
</book>
```

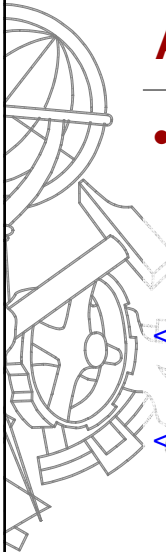


## An Example XML Document

- The <title> and <year> elements contain text content.

```
<title>Teach Yourself ASP 3.0 in 21 Days</title>  
<year>1999</year>
```

2006/2007 ADAV 25  
Ambientes de Desenvolvimento Avançados

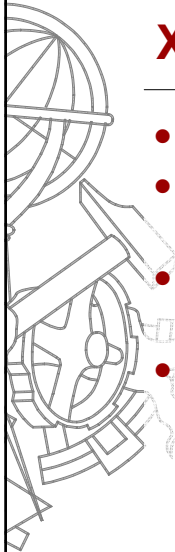


## An Example XML Document

- The <authors> element contains two <author> child elements. The <author> elements contain text content, specifying the author(s) of the book:

```
<authors>  
  <author>Mitchell</author>  
  <author>Atkinson</author>  
</authors>
```

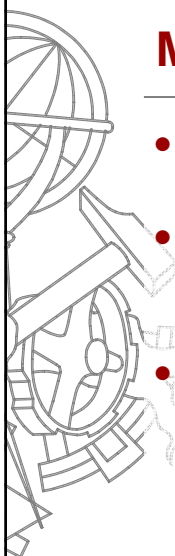
2006/2007 ADAV 26  
Ambientes de Desenvolvimento Avançados



## XML Formatting Rules

---

- XML is case sensitive.
- XML Documents must adhere to specific formatting rules.
- An XML document that adheres to such rules is said to be *well-formed*.
- These formatting rules are presented over the next few slides.



## Matching Tags

---

- XML elements must consist of both an *opening tag* and a *closing tag*.
- For example, the element `<book>` has an opening tag - `<book>` - and must also have a matching closing tag - `</book>`.
- For elements with no contents, you can use the following shorthand notation:  
`<elementName optionalAttributes />`



## Elements Must be Properly Nested

---

- Examples of properly nested elements:

```
<book price="39.95">  
  <title>XML and ASP.NET</title>  
</book>
```

- Example of improperly nested elements:

```
<book price="39.95">  
  <title>XML and ASP.NET  
</book> </title>
```



## Attribute Values Must Be Quoted

---

- Recall that elements may have an arbitrary number of attributes, which are denoted as:  
*<elementName attribute1="value1" ... attributeN="valueN"> ... </elementName>*
- Note that the attribute values *must be delimited by quotation marks*. That is, the following is malformed XML:

```
<book price=39.95> ... </book>
```

## Illegal Characters for Text Content

- Recall that elements can contain text content or other elements. For example, the <title> element contains text, providing the title of the book:

```
<title>XML for ASP.NET</title>
```

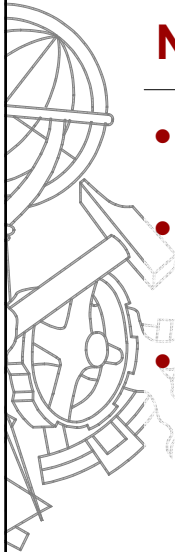
- There are five characters that cannot appear within the text content: <, >, &, ", and '.

## Replacing the Illegal Characters with Legal Ones

- If you need to use any of those four illegal characters in the text portion of an element, replace them with their equivalent legal value:

Replace This	With This
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos;





## Namespaces

---

- Element names in an XML document can be invented by the XML document's creator.
- Due to this naming flexibility, when computer programs work with different XML documents there may be *naming conflicts*.
- For example, appending two XML documents together that use the same element name for different data representations leads to a naming conflict.



## Naming Conflicts

---

- For example, if the following two XML documents were to be merged, there would be a naming conflict:

```
<bug>  
  <date>2003-05-22</date>  
  <description>The button, when clicked,  
  raises a GPF exception.</description>  
</bug>
```

```
<bug>  
  <genus>Homoptera</genus>  
  <species>Pseudococci dae</species>  
</bug>
```



## Solving Name Conflicts with Namespaces

---

- A namespace is a *unique* name that is “prefixed” to each element name to uniquely identify it.
- To create namespaces, use the xmlns attribute:

```
<namespace-prefix:elementName xmlns:namespace-prefix="namespace">
```

E. g. :

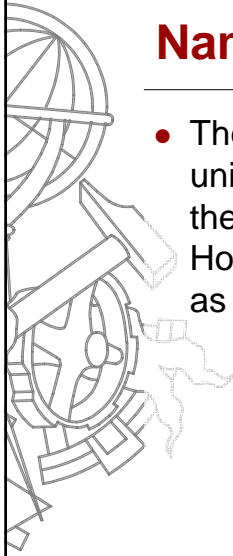
```
<animal:bug xmlns:animal="http://www.bugs.com/">
```



## Namespaces

---

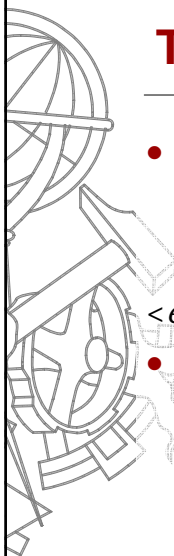
- Note that the xmlns namespace attribute contains two parts:
  1. The *namespace-prefix* and,
  2. The namespace value.
- The namespace-prefix can be any name. The namespace-prefix is then used to prefix element names to indicate that the element belongs to a particular namespace.
- The namespace for an element automatically transfers to its children elements, unless explicitly specified otherwise.



## Namespaces

---

- The namespace value must be a globally unique string. Typically, URLs are used since they are unique to companies/developers. However, *any* string value will suffice, so long as it is unique.



## The Default Namespace

---

- To save typing in the namespace-prefix for every element, a *default namespace* can be specified using `xmlns` *without* the *namespace-prefix* part, like:

```
<elementName xmlns="namespace" >
```

- Since messages to Web services are XML-formatted (as we'll see in a bit), it is important that namespaces are used to remove any ambiguity between potential naming conflicts.



## Defining XML Document Structure

---

- Relational databases employ *schemas*, which explicitly define the structure and properties of the data a table can hold.
- XML allows for its structure to be defined using either one of two technologies:
  1. DTDs (**D**ocument **T**ype **D**efinition)
  2. XSD (**X**ML **S**chema **D**efinition)
- XSD is the method used to define the structure of the XML messages passed to and from a Web service.



## XML Schemas (XSD)

---

- XSD files are XML-formatted files themselves, and define the structure of some other XML document.
- This technology is useful with Web services, because it provides a means for the Web service and client to both know the structure of the message being passed to and from.

## XSD Example

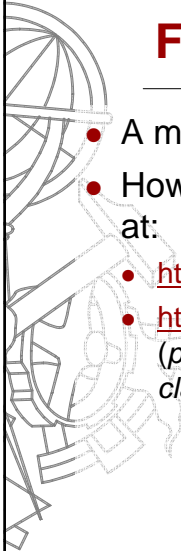
- Given the following XML document:

```
<books>
  <book>
    <title>ASP.NET Data Web Controls</title>
    <author>Mitchell</author>
    <year>2003</year>
  </book>
</books>
```

## XSD Example

- We can describe the previous XML file with the following XSD file:

```
<xsd:schema xmlns:xsd="...">
  <xsd:element name="books">
    <xsd:complexType>
      <xsd:element name="book">
        <xsd:complexType>
          <xsd:element name="title" type="xsd:string" />
          <xsd:element name="author" type="xsd:string" />
          <xsd:element name="year" type="xsd:integer" />
        </xsd:complexType>
      </xsd:element>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```




## For More XSD Information...

---

- A mastery of XSD is not required for this course.
- However, if you are interested, you can learn more at:
  - <http://www.w3schools.com/schema>
  - <http://datawebcontrols.com/classes/xmlfor.net/Session2.zip>  
(presentation on XML schemas from the XML for .NET class)

2006/2007
ADAV  
Ambientes de Desenvolvimento Avançados
43



## Web Services Standards Stack

---

<b>Directory: Publish &amp; Find Services:</b>	<b>UDDI</b>
<b>Description: Formal Service Descriptions:</b>	<b>WSDL</b>
<b>Wire Format: Service Interactions:</b>	<b>SOAP</b>
<b>Universal Data Format:</b>	<b>XML</b>
<b>Ubiquitous Communications:</b>	<b>Internet</b>

Simple, Open, Broad Industry Support

2006/2007
ADAV  
Ambientes de Desenvolvimento Avançados
44

## What is SOAP?

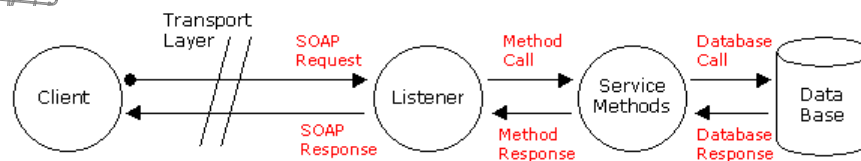
- **SOAP** – (Simple **O**bject **A**ccess **P**rotocol)  
<http://www.w3c.org/TR/soap>
- SOAP is an XML-based messaging protocol.
- simple one-way messaging
- performing RPC-style (Remote Procedure Call) request-response dialogues
- Not tied to any particular operating system or programming language

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

45

## What is SOAP?



The SOAP developer's approach

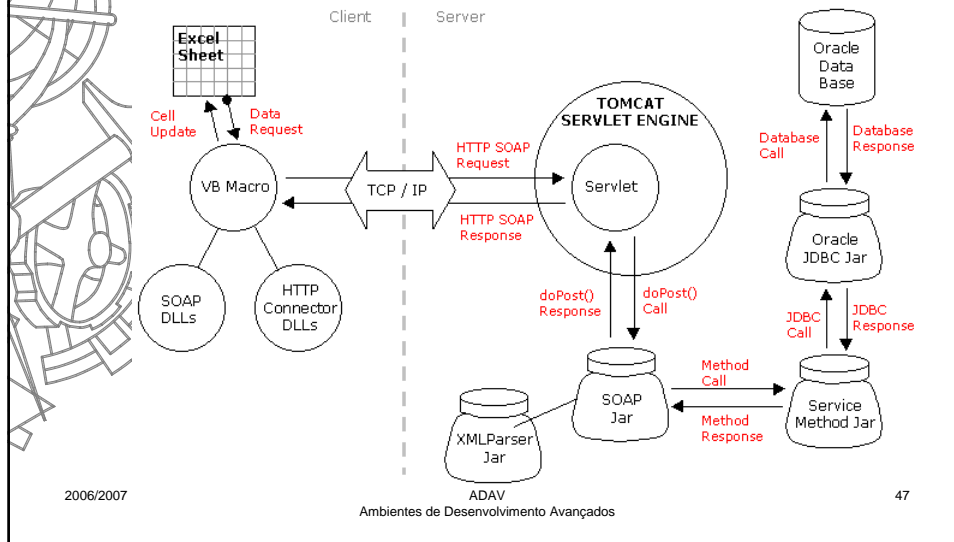
- encapsulate the database request logic for the service in a method
- the service method, decodes the incoming SOAP request

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

46

## What is SOAP?



## SOAP Messages

- A valid SOAP Message is a well-formed XML document
- It should use the SOAP Envelope and SOAP Encoding namespaces and have the following form:
  - An XML Declaration
  - A SOAP Envelope
    - A SOAP Header
    - A SOAP Body



# SOAP Messages

---

- A SOAP-encoded RPC dialogue contains both
  - a request message
  - a response message.

**Method Signature**

```
int doubleAnInteger ( int numberToDouble );
```

2006/2007
ADAV  
Ambientes de Desenvolvimento Avançados
49

# Message Structure

---

The diagram illustrates the nested structure of a SOAP message. It consists of the following layers from outermost to innermost:

- SOAP Message**: The complete SOAP message.
- Headers**: Protocol binding headers.
- SOAP Envelope**: `<Envelope>` encloses payload.
- SOAP Header**: `<Header>` encloses headers.
- Headers**: Individual headers.
- SOAP Body**: `<Body>` contains SOAP message name.
- Message Name & Data**: XML-encoded SOAP message name & data.

2006/2007
ADAV  
Ambientes de Desenvolvimento Avançados
50

- A SOAP message is an XML document – it has a mandatory SOAP envelope, an optional SOAP header, and a mandatory SOAP body.
- The envelope is the top element of the XML document representing the message.
- The header is a generic mechanism for adding features to a SOAP message in a decentralized manner without prior agreement between the communicating parties.
- SOAP defines a few attributes that can be used to indicate who should deal with a feature and whether it is optional or mandatory.
- The Body is a container for mandatory information intended for the ultimate recipient of the message.
- SOAP defines one element for the body, which is the Fault element used for reporting errors.

## SOAP Messages

### Request

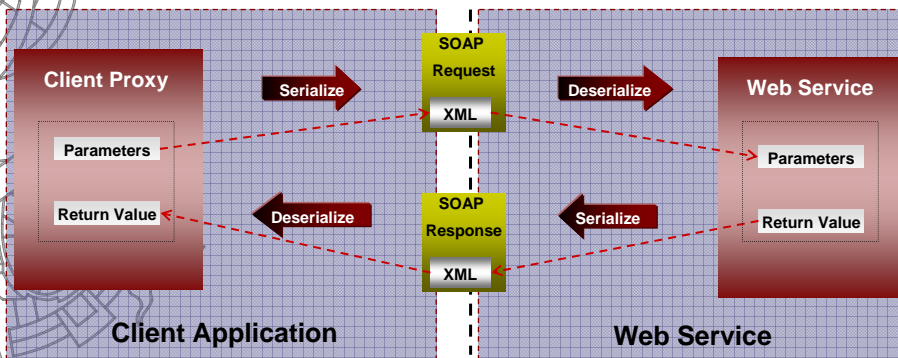
```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doubleInteger xmlns:ns1="urn:MySoapServices">
      <param1 xsi:type="xsd:int">123</param1>
    </ns1:doubleInteger>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

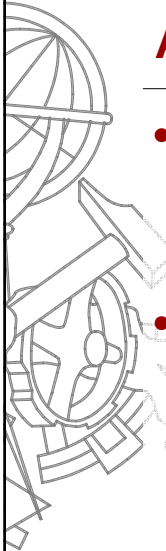
# SOAP Messages

## Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:doubleIntegerResponse xmlns:ns1="urn:MySoapServices"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <return xsi:type="xsd:int">246</return>
    </ns1:doubleIntegerResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

## How it works





## An Example

- Imagine that we had a Web service with an Add() method, that took two integer inputs and returned their sum.
- Let's look at what the request and response SOAP messages would look like if Add(5, 8) was called from the client...

2006/2007 ADAV 55  
Ambientes de Desenvolvimento Avançados



## The Request SOAP Message

```
<soap:Envelope namespace>  
  <soap:Body>  
    <Add xmlns="Web Service Namespace">  
      <a>5</a>  
      <b>8</b>  
    </Add>  
  </soap:Body>  
</soap:Envelope>
```

2006/2007 ADAV 56  
Ambientes de Desenvolvimento Avançados

## The Response SOAP Message

```
<soap:Envelope namespaces>  
  <soap:Body>  
    <AddResponse xmlns="Web Service Namespace">  
      <AddResult>13</AddResult>  
    </AddResponse>  
  </soap:Body>  
</soap:Envelope>
```

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

57

```
<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="Web Service Namespace">  
      <a>5</a>  
      <b>8</b>  
    </Add>  
  </soap:Body>  
</soap:Envelope>
```



Client



Server



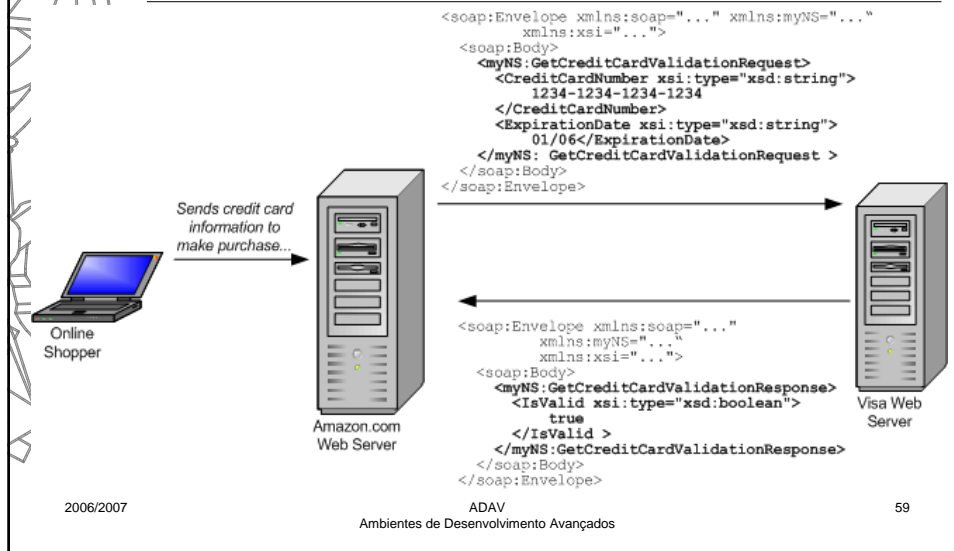
```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  
  <soap:Body>  
    <AddResponse xmlns="Web Service Namespace">  
      <AddResult>13</AddResult>  
    </AddResponse>  
  </soap:Body>  
</soap:Envelope>
```

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

58

## SOAP Interaction Other Example



## SOAP Example

- WS Calculadora

## Web Services Standards Stack

Directory: Publish & Find Services:	UDDI
Description: Formal Service Descriptions:	WSDL
Wire Format: Service Interactions:	SOAP
Universal Data Format:	XML
Ubiquitous Communications:	Internet

Simple, Open, Broad Industry Support

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

61

## WSDL

- **WSDL - Web Services Description Language**
- XML schema for describing Web Services
  1. Service interface definition
    - Abstract semantics for Web Service
  2. Service implementation definition
    - Concrete end points and network addresses where Web Service can be invoked
- Clear delineation between abstract and concrete messages

<http://www.w3c.org/TR/wsdl>

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

62

## WSDL - Overview

- WSDL is a simple XML grammar for describing how to communicate with a Web service
  - It defines the messages (both abstract and concrete) that are sent to and from a service
  - It defines logical collections of messages (“port type”, “interface”)
  - It defines how a given “port type” is bound to particular wire protocols
  - It defines where the service is located

2006/2007 ADAV 63  
 Ambientes de Desenvolvimento Avançados

## WSDL - Overview

```

<definitions>
  <types> <!-- XML Schema --> </types>
  <message name="getQuote_In" />
  <message name="getQuote_Out" />
  <portType name="StockQuoteServiceInterface">
    <operation name="getQuote">
      <input message="getQuote_In" />
      <output message="getQuote_Out" />
    </operation>
  </portType>
  <binding name="StockQuoteServiceBinding" type="StockQuoteServiceInterface">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
    ...
  </binding>
  <service name="StockQuoteService">
    <port name="StockQuoteServicePort" binding="StockQuoteServiceBinding">
      <soap:address location="http://www.acme.com/services/stockquote" />
    </port>
  </service>
</definitions>
  
```

Definition of data types

Definition of messages

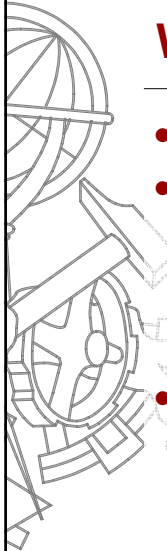
Definition of port type

Definition of the bindings

Definition of the service

2006/2007 ADAV 63  
 Ambientes de Desenvolvimento Avançados






## WSDL Overview

- WSDL is extensible.
- WSDL was created by IBM and Microsoft
  - The intent was to create something that worked, not something that was complete
  - Creating a formal Web Services “data model” was *not* a priority
- WSDL is RDF-compatible (*not* RDF-compliant)

2006/2007 ADAV 65  
Ambientes de Desenvolvimento Avançados



## The Web Service’s WSDL Document

- All Web services contain a WSDL file that very precisely spells out the Web service’s methods, their input and output parameters, how the Web service can be invoked (HTTP-GET/HTTP-POST/SOAP), and other such information.
- The Web service description page contains a link to the Web service’s WSDL file (go to the first page and click the “Service Description” link)

2006/2007 ADAV 66  
Ambientes de Desenvolvimento Avançados

# Web Service's WSDL Document

- The WSDL document is an XML-formatted document.

```

<?xml version="1.0" encoding="utf-8" ?>
- <definitions xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:s="http://www.w3.org/2001/XMLSchema"
  xmlns:s0="http://tempuri.org/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  targetNamespace="http://tempuri.org/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
- <types>
  - <s:schema elementFormDefault="qualified"
    targetNamespace="http://tempuri.org/">
    - <s:element name="Add">
      - <s:complexType>
        - <s:sequence>
          <s:element minOccurs="1" maxOccurs="1" name="x"
            type="s:int" />
          <s:element minOccurs="1" maxOccurs="1" name="y"
            type="s:int" />
        </s:sequence>
      </s:complexType>
    </s:element>
    - <s:element name="AddResponse">
      - <s:complexType>

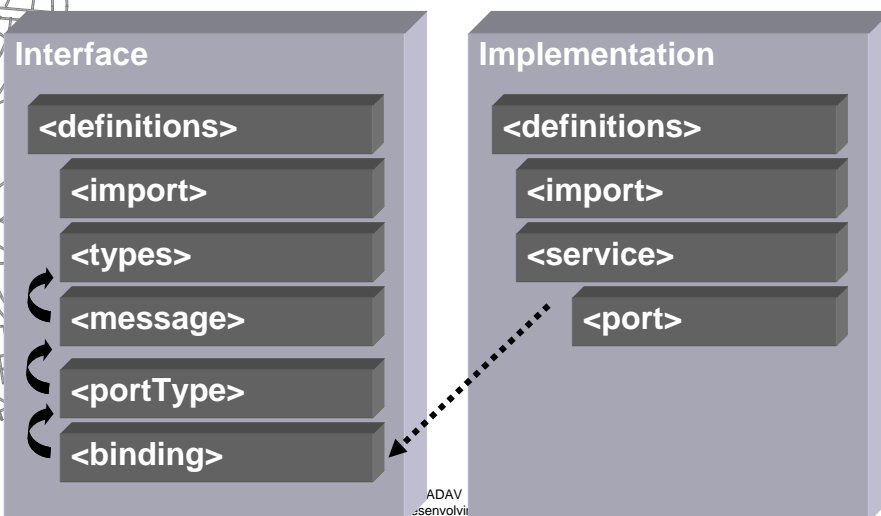
```

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

67

# WSDL WSDL Schema



# WSDL

## WSDL Schema

### Interface

**<definitions>**

**<import>**

**<types>**

**<message>**

**<portType>**

**<binding>**

- **<definitions>**  
are root node of WSDL
- **<import>**  
allows other entities for inclusion
- **<types>**  
are data definitions - xsd
- **<message>**  
defines parameters of a Web Service function
- **<portType>**  
defines input and output operations
- **<binding>**  
specifies how each message is sent over the wire

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

69

# WSDL

## WSDL Schema

### Implementation

**<definitions>**

**<import>**

**<service>**

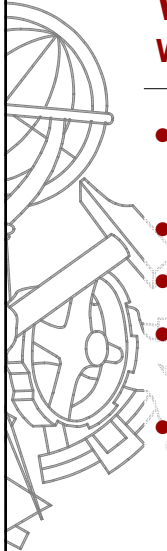
**<port>**

- **<service>**  
specifies details about the implementation
- **<port>**  
contains the address itself

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

70



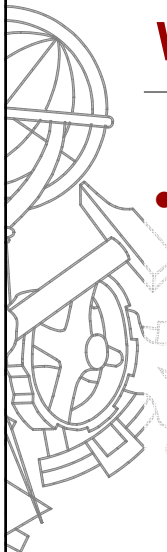
## WSDL

### WSDL Elements

---

- Open – allows for other namespaces and thus highly extensible
- Ability to import other schemas & WSDL
- Provides “recipe” for Web Services
- Provides both interface and implementation details
- Allows for separation of the two

2006/2007 ADAV 71  
Ambientes de Desenvolvimento Avançados



## WSDL Example

---

- WSDL do WS Calculadora

2006/2007 ADAV 72  
Ambientes de Desenvolvimento Avançados

## Web Services Standards Stack

<b>Directory: Publish &amp; Find Services:</b>	<b>UDDI</b>
<b>Description: Formal Service Descriptions:</b>	<b>WSDL</b>
<b>Wire Format: Service Interactions:</b>	<b>SOAP</b>
<b>Universal Data Format:</b>	<b>XML</b>
<b>Ubiquitous Communications:</b>	<b>Internet</b>

**Simple, Open, Broad Industry Support**

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

73

## What is UDDI?

- UDDI is a project started by Microsoft, IBM, and Ariba and is being publicly launched September 6th.
- The goal of UDDI is to accelerate the adoption of B2B commerce by providing businesses with a standard way to programmatically describe their own Web Services while making it easier to discover the Web Service capabilities of their trading partners and customers.
- UDDI does this through a set of specifications for service description and through the shared operation of a Business Registry on the web.
- UDDI is based on XML and SOAP and is a core component of Microsoft's .NET strategy.
- Over 30 companies, including Sun Microsystems, have joined the project.

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

74



## UDDI Overview

---

- **UDDI = Universal Description, Discovery, and Integration**  
<http://www.uddi.org>
- Industry Initiative to address discovery
  - A registration database for Web Services
- Specifications
  - Schema for service providers and descriptions
  - API for publishing and searching
  - Developed on industry standards (XML, HTTP, TCP/IP, SOAP)
  - Applies to both XML and non-XML services
- Implementation
  - Public and private instances of specification

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

75



## UDDI Overview

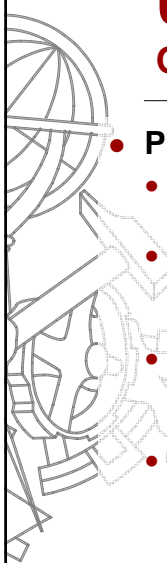
---

- A number of companies (Microsoft, IBM, HP, etc.) run global UDDI Business Registries, which are a database of Web services.
- These directories can be searched via a Web site interface. For example, see:  
<http://uddi.microsoft.com/> or  
<https://uddi.ibm.com/ubr/registry.html>

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

76

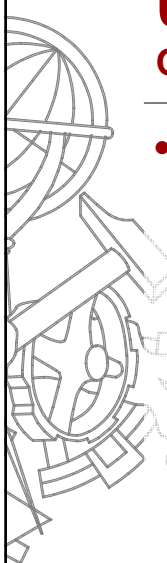


## UDDI Overview

---

- **Publishing activities**
  - All interactions that involve publishing require an authenticated connection.
  - Before you can start publishing data to a UDDI registry, you must first have visited the registry web site and selected a specific *UDDI Operator* that you'll use to manage your data.
  - Once you have selected an operator (there are many operators, each which manage a portion of the overall data), you'll sign up for publishing credentials.
  - You'll need these credentials (typically a user name and password pair) to connect to the UDDI publishing server before using any of the UDDI publishing API.

2006/2007 ADAV 77  
Ambientes de Desenvolvimento Avançados



## UDDI Overview

---

- **Registering your organization**
  - One of the first tasks is to register your master organization information.
  - This is simply data about your business or organization, including the name of the organization, and optional description and contact information.
  - Several of the objects provided in the SDK are useful for registering data about your organization.

2006/2007 ADAV 78  
Ambientes de Desenvolvimento Avançados

# UDDI

## Overview

- **Registering software that uses .Net services**
  - The main purpose of the UDDI registry is to let others know you exist and that you have .Net services exposed that can be used to interact with your business.
  - Common examples of these services will be for business document interchange – for documents such as purchase orders, invoices, shipping notices, etc.
  - Depending on the tools and products you use to manage your exposed services, you'll want to use the UDDI registry to advertise technical information about your service.

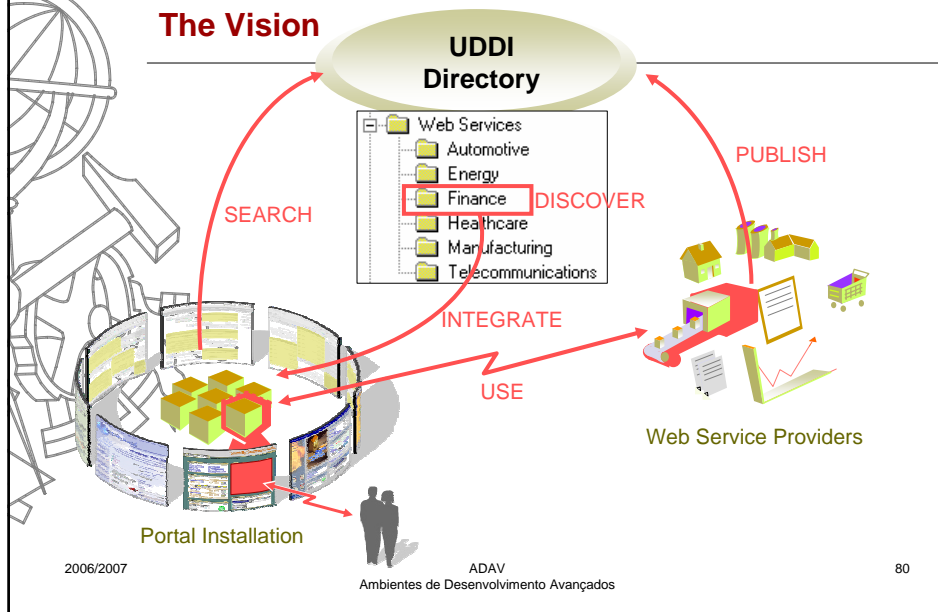
2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

79

# UDDI

## The Vision



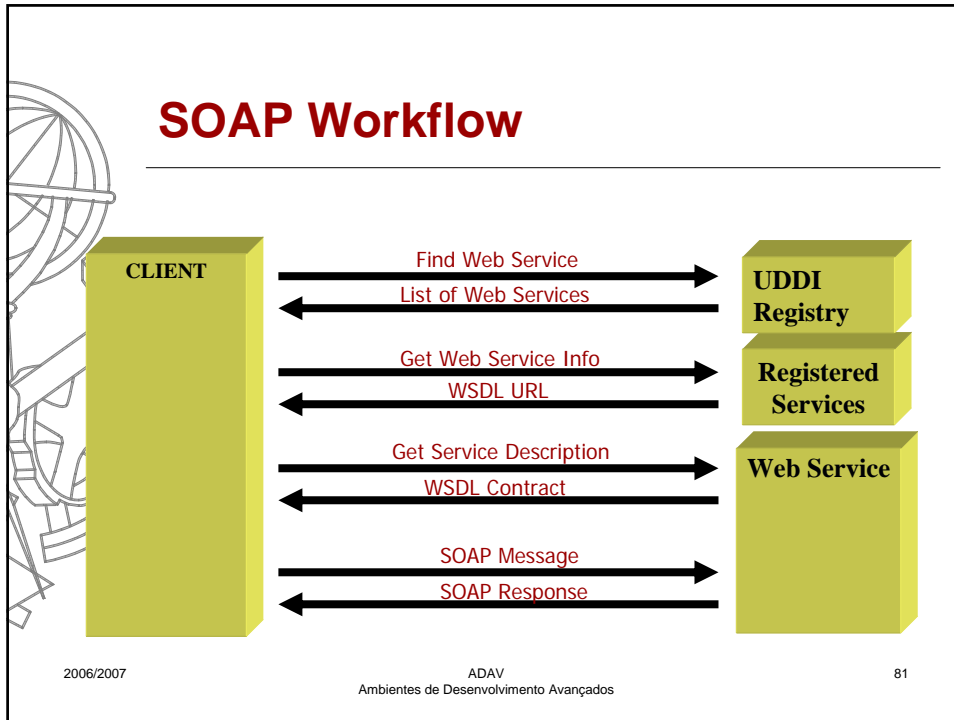
2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

80



## SOAP Workflow



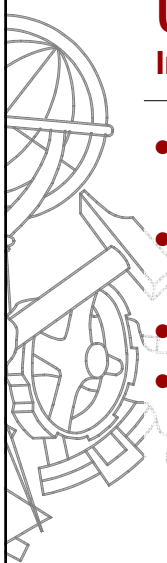
## UDDI Providers, Services And Bindings

- **Providers**
  - Examples: Accounting Department, Corporate Application Server
  - Name, Description, Contact Information
  - Categorization and Identification Information
- **Services**
  - Examples: Purchase Order services, Payroll services
  - Name, Description(s)
  - Categorization Information
- **Bindings**
  - Description(s), access points, parameters
  - Examples: Access Point (<http://...>) for Web Service

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

82



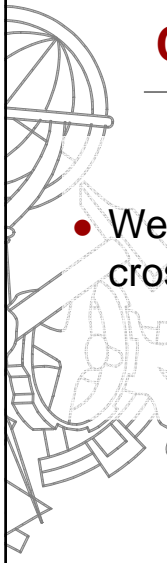
## UDDI

### Important UDDI Features

---

- Neutral in terms of protocols – as a registry, it can contain pointers to *anything*
- Can search by business, service, Web Service (tModel), binding
- Usage of Globally Unique Identifiers (GUIDs)
- Specification allows public and private nodes

2006/2007 ADAV 83  
Ambientes de Desenvolvimento Avançados

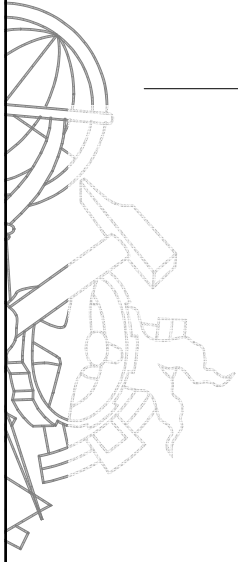


## Conclusion

---

- Web services standards are the future of cross-platform interop

2006/2007 ADAV 84  
Ambientes de Desenvolvimento Avançados



---

**Questões**

**?**

2006/2007

ADAV  
Ambientes de Desenvolvimento Avançados

85