



# **CSI02 – Arquitetura Orientada a Serviços**

## **Fundamentos de Orientação a Serviços**

**Instituto Tecnológico de Aeronáutica - ITA**  
Divisão de Ciência da Computação – IEC

Prof. Inaldo Capistrano Costa



## Roteiro da apresentação

- **Introdução**
  - **Histórico**
  - **O que são Web Services?**
  - **Por que Web Services?**
- **Tecnologias envolvidas**
  - **SOA**
  - **SOAP**
  - **XML**
  - **WSDL**
  - **UDDI**
  - **Vídeo-aula “Construção de WS com Eclipse”**
  - **Aula prática usando Netbeans e .Net**
- **Web Semântica**
  - **Anotações Semânticas em serviços**
  - **SA-WSDL**
    - **Vídeo-aula SA-WSDL com WSMO**
- **Exemplos WS**
- **Produtos disponíveis**
- **Links para padrões**
- **Considerações finais**
- **Referências**



## Histórico

- *Imagine a person trying to start two programs at the same time, on two separate computers, and wanting them to communicate with each other.*
- *There are a few different ways that two programs on two different computers can handle their communications with each other. We will start with client/server, which is probably the best known way.*
- *The client/server model solves the rendezvous problem by determining that one side (the server) must start execution, and wait indefinitely for the other side (the client) to contact it.*
  - *Server—a program that waits for incoming communication requests from a client. When the server receives a communication from the client, the server provides some useful service to the client, and sends the client the results.*
  - *Client—a program that initiates communication with the server. The client has need of some service that the server can provide.*



## Histórico

- *Servers are concerned with:*
  - *Authentication: Verifying that the client is who it claims to be*
  - *Authorization: Determining whether the given client is permitted to access any of the services the server supplies*
  - *Providing services*
  - *Data security: Guaranteeing that a client is not able to access data that the client is not permitted to access, preventing data from being stolen*
  
- *Servers typically come in two different versions:*
  1. *Stateless—server does not save any information about the status of ongoing interaction with clients*
  
  2. *Stateful—server saves information about the status of ongoing interactions with clients*



### ➤ *WHAT IS MIDDLEWARE?*

*The following definition from Techopedia (2016) is more comprehensive:*

*Middleware is a software layer situated between applications and operating systems. Middleware is typically used in distributed systems where it simplifies software development by doing the following:*

- *Hides the intricacies of distributed applications*
- *Hides the heterogeneity of hardware, operating systems, and protocols*
- *Provides uniform and high-level interfaces used to make interoperable, reusable and portable applications*
- *Provides a set of common services that minimizes duplication of efforts and enhances collaboration between applications*



### ➤ WHAT IS MIDDLEWARE?

*The following definition from Apprenda (2016):*

*A simple middleware definition: software that connects computers and devices to other applications. It can also be referred to as the slash or connecting point in client/server. Another way to define middleware is to say that it is software that acts as a liaison between applications and networks. The term is often used in the context of cloud computing, such as public or private cloud.*



## Histórico

- *Aplicação de duas camadas. Aplicações cliente-servidor.*
- *Busca de escalabilidade e tolerância maior a falhas, criou-se a arquitetura de 3 camadas.*
- *Camada de negócios, entre cliente e dados.*
- *Primeiros sistemas middleware foram os RPC (Remote Procedure Call), baseados na programação estruturada. Possibilitou execução de uma mesma aplicação em plataformas diferentes.*
- *Com POO, foram criados CORBA, DCOM e RMI, cada um com suas vantagens e desvantagens. Mas nenhuma delas se comunica com a outra. Aconselhados para Intranet.*
- *Até chegar em Web services.*

## Histórico

- *Primeiros sistemas middleware foram os RPC (Remote Procedure Call), baseados na programação estruturada. Possibilitou execução de uma mesma aplicação em plataformas diferentes.*

### Remote Procedure Call

- Basic RPC operation



Figure 4-3 Basic RPC model





## Histórico

### CORBA

Solução aberta de objetos distribuídos.

Desenvolvida pelo consórcio OMG (Object Management Group) para se tornar padrão de mercado.

Primeira vantagem é que cliente e servidor podem ser feitos em qualquer linguagem, em função do alto nível de abstração conseguido com o IDL (Interface Definition Language), que possui o mapeamento dos métodos.

Comunicação é feita através de ORBs (Object Request Broker), responsáveis pela tradução de solicitação e resposta.

Aconselhável para aplicações que necessitam alta performance

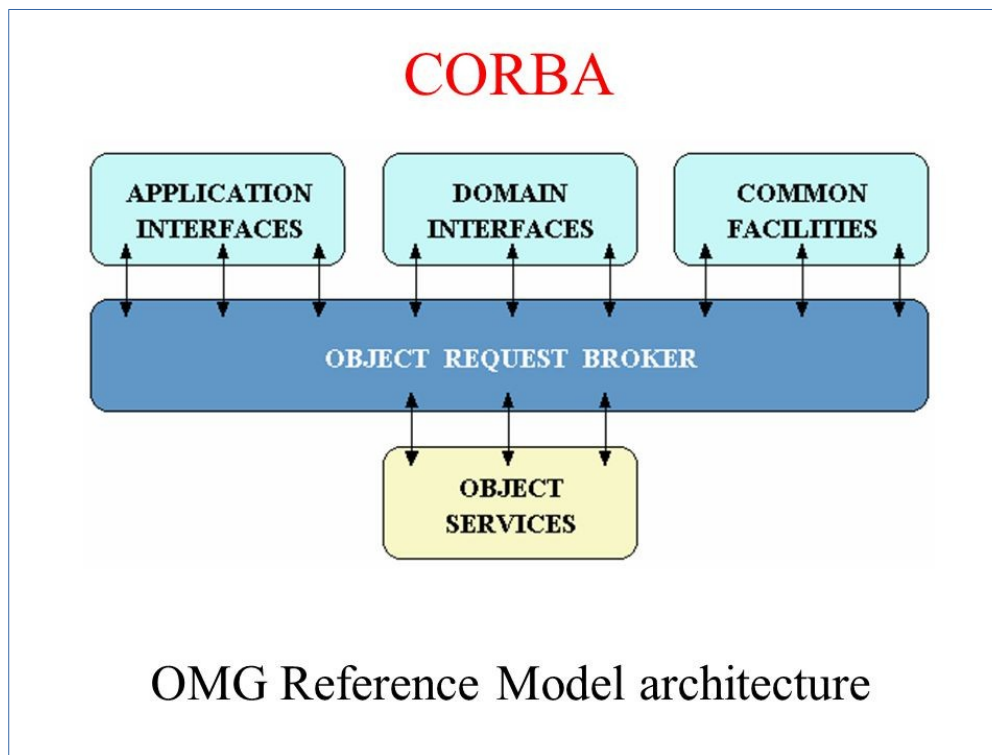
Principal desvantagem é o alto grau de complexidade na implementação de pequenas soluções.

## Histórico

### CORBA

Solução aberta de objetos distribuídos.

Desenvolvida pelo consórcio OMG (Object Management Group) para se tornar padrão de mercado.



## Histórico

### RMI

Remote Method Invocation

Habilita comunicação entre aplicações Java-Java, que usam JVM possivelmente em plataformas diferentes.

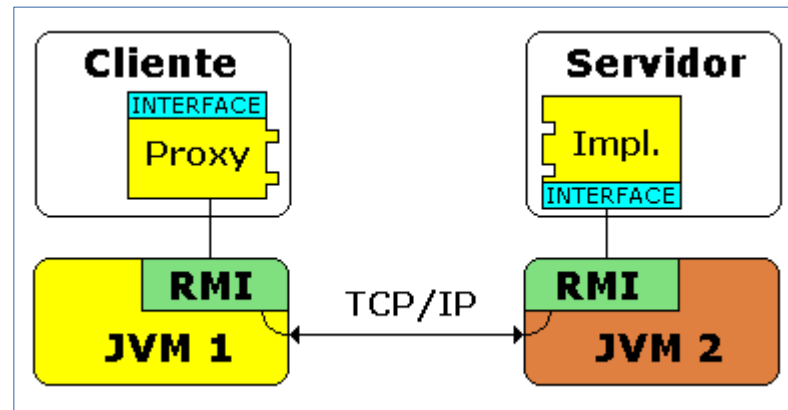
Aplicação Java armazena apenas as referências dos objetos remotos.

Desenvolvimento de aplicações ficou mais simples.

Suporte ao Garbage Collection

Desvantagens

Os dois lados devem ser aplicações Java.



## Histórico

### DCOM

Distributed Component Object Model

Desenvolvida pela Microsoft para distribuir objetos pela rede.

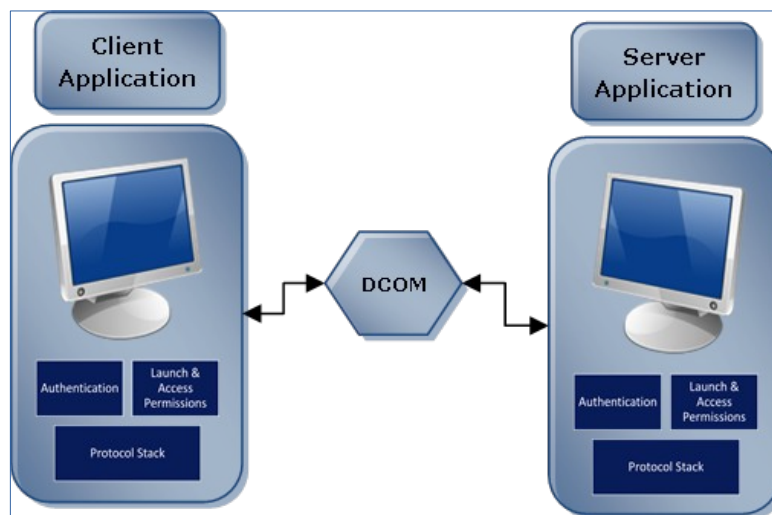
Servidor DCOM publica métodos para clientes escritas em IDL, similar ao C++.

Usa protocolo ORPC (Object Remote Procedure Call) que usa o PING para permanecer com os objetos ativos para o cliente.

DCOM suporta o GC.

Implantado nas empresas Apple, Unix e VMS.

Desvantagem – proprietário.



## O que são Web Services

- “Um sistema de software identificado através de uma URI (Identificador Universal de Recursos) cujas interfaces públicas e interconexões são descritas em XML. Sua definição é publicada de modo a poder ser “descoberta” por outros sistemas de software. Web Services podem interagir com outros sistemas ou Web Services do modo prescrito em sua definição, utilizando mensagens baseadas no padrão XML produzidas através de protocolos de Internet.” Ref: glossários do W3C
- Web service é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os Web Services são componentes que permitem às aplicações enviar e receber dados em formato XML. Cada aplicação pode ter a sua própria “linguagem”, que é traduzida para uma linguagem universal, o formato XML.



## O que são *Web Services*

Resumindo, Um Web Service é:

Qualquer serviço que é disponibilizado através da web.

Qualquer serviço que possibilita duas aplicações de computador trocarem dados.

Principalmente, mas não exclusivamente baseado em:

XML para codificação de dados

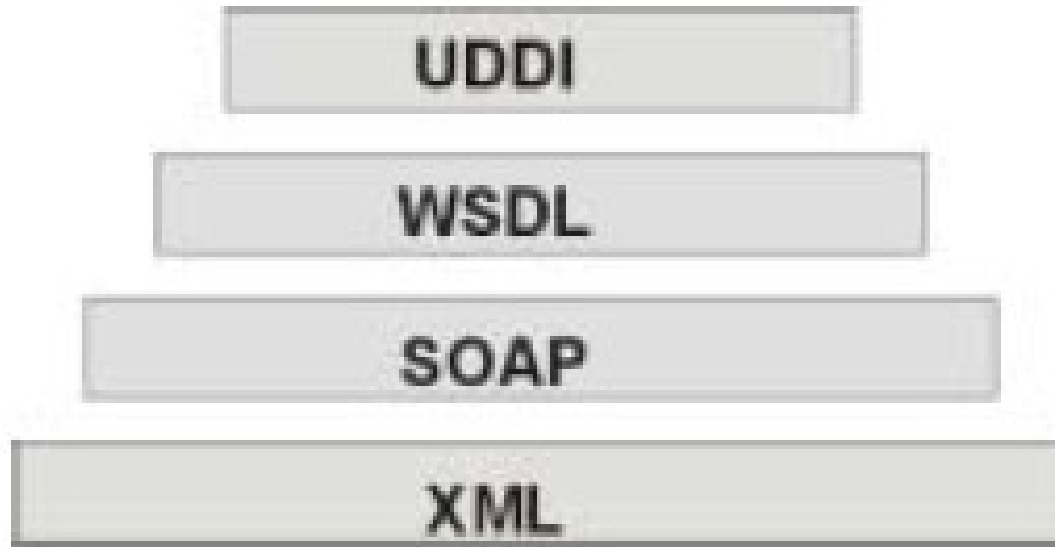
HTTP para transporte de dados

Um documento XML transmitido remotamente e mapeado para um programa executável.





## O que são *Web Services* (SOA)



Comunicação entre aplicações de WS usam 4 camadas que empacotam a requisição e a resposta entre o servidor e o cliente.

XML – possibilita estabelecer objetos, métodos, parâmetros, dados e tipos de dados

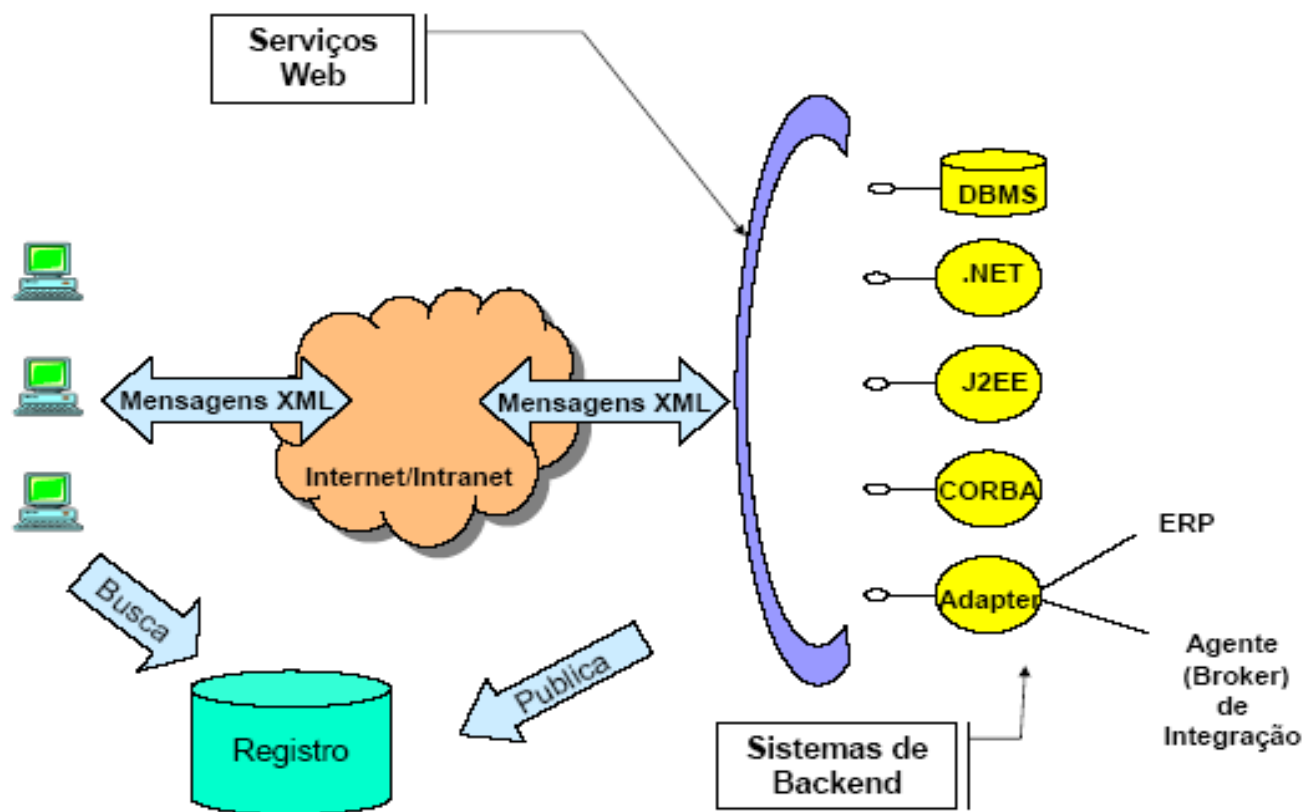
SOAP – protocolo que estabelece a comunicação entre os ambientes

WSDL – descreve o serviço (XML)

UDDI – representa service broker. Contém as descrições dos WS.

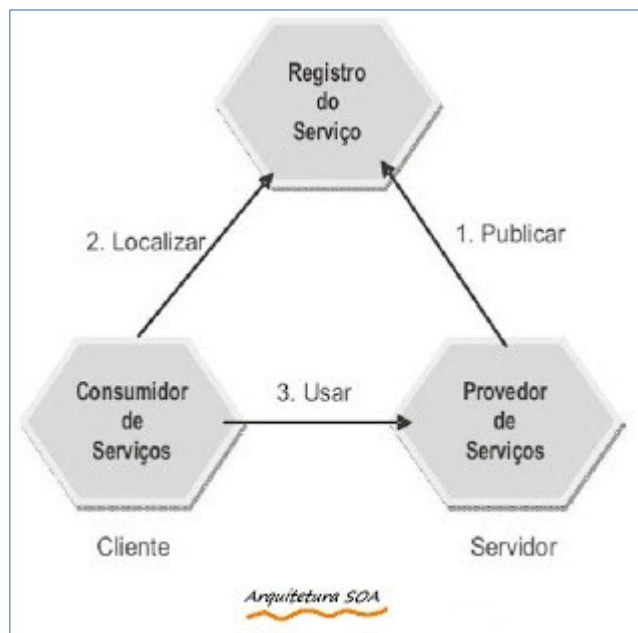
## O que são *Web Services*

### Modelo de Serviços Web



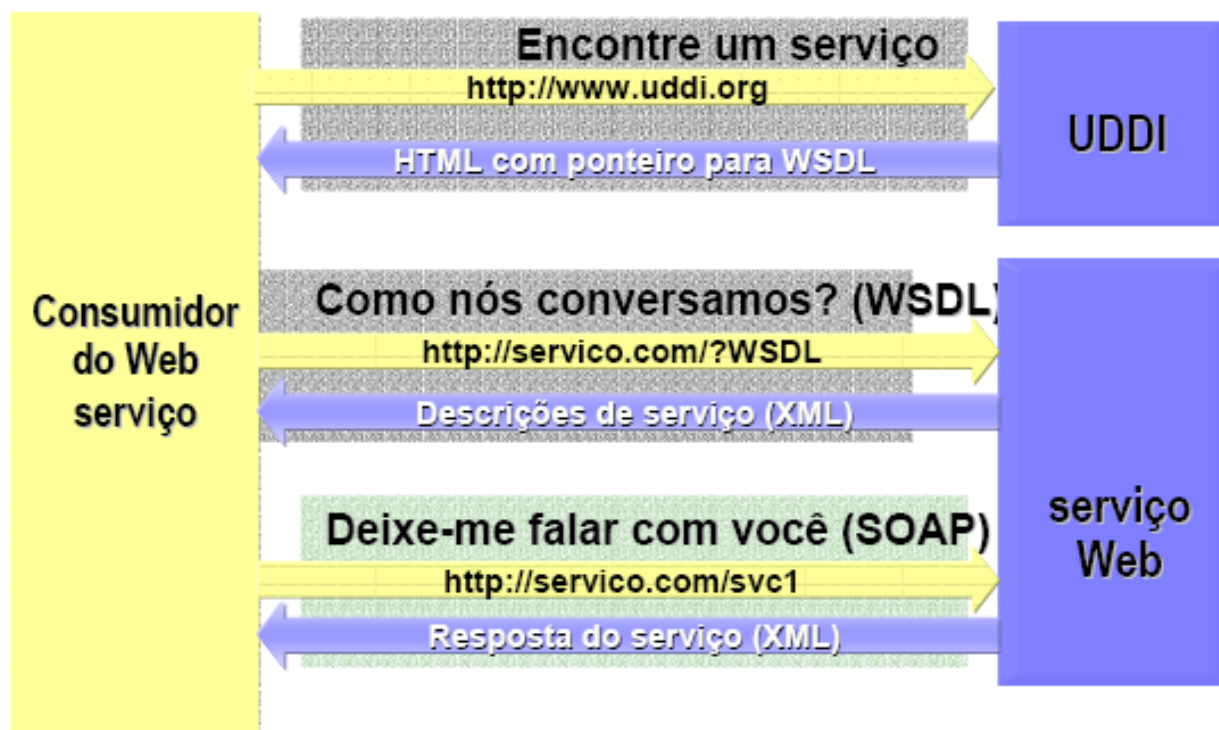


# O que são *Web Services*



## O que são *Web Services*

### Protocolos de Serviços Web





## Por que *Web Services*?

- Dois fatores chave:
  - ubiqüidade
  - facilidade de uso
- Interoperável:
  - Neutro em relação a SO e linguagem
  - Integração Java & .NET : simples e barata
- Todo mundo dá suporte ou irá dar a Serviços Web:
  - Necessário dar suporte a Serviços Web para facilitar Integração
- Não-invasivos(impactante):
  - Baseados em protocolos ubiqüos: HTTP/SMTP
  - Complementam tecnologias já existentes



## Por que *Web Services*?

### ➤ INTEGRAÇÃO

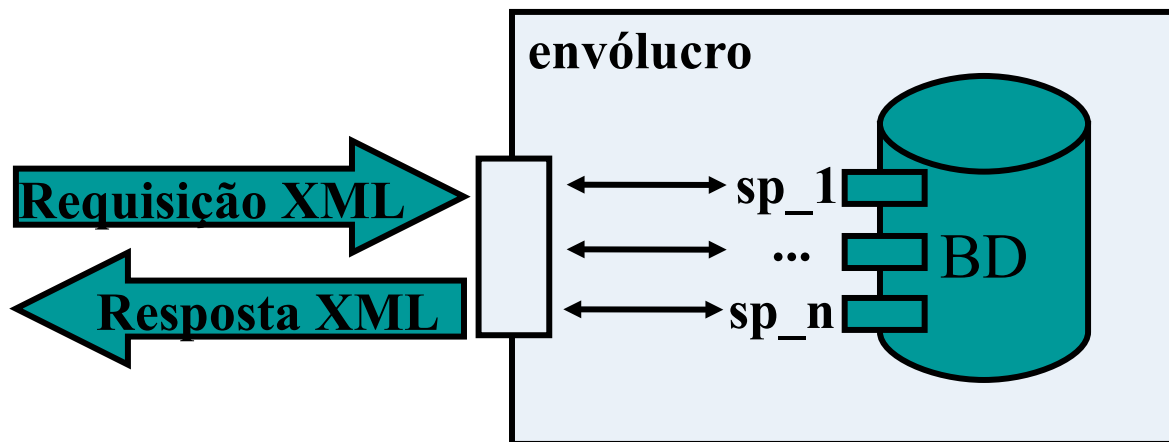
- Interna:
  - Dados como nome, endereço, telefone, matrícula, etc, todas as áreas da empresa utilizam. Assim, estes dados podem ser disponibilizados (em um formato comum) para que sistemas desenvolvidos por outras áreas possam utilizar em seus aplicativos específicos.
- Externa:
  - Gestão da cadeia de suprimentos. Esta integração é baseada na interação que existe entre várias empresas durante os diversos processos de fabricação e a logística.



## O que precisam?

	<b>Web Services</b>	<b>CORBA</b>	<b>RMI</b>
Comunicação	<b>SOAP</b>	GIOP	JRMP
Descrição	<b>WSDL</b>	CORBA IDL	Java
Registro	<b>UDDI</b>	Serviço de nomes	RMI Registry

## Exemplo de *Web Services*





## Exemplo de Web Services

GlobalWeather Web Service - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço <http://www.webservicex.com/globalweather.aspx?op=GetWeather>

Google  Go

### GlobalWeather

Click [here](#) for a complete list of operations.

#### GetWeather

Get weather report for all major cities around the world.

**Test**

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
CityName:	<input type="text" value="Porto Alegre"/>
CountryName:	<input type="text" value="Brazil"/>

**SOAP**

The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with the actual values.

```
POST /globalweather.aspx HTTP/1.1
Host: www.webservicex.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://www.webserviceX.NET/GetWeather"

<?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>
    <GetWeather xmlns="http://www.webserviceX.NET">
      <CityName>string</CityName>
      <CountryName>string</CountryName>
    </GetWeather>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Content-Length: length

```
<?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>
    <GetWeatherResponse xmlns="http://www.webserviceX.NET">
      <GetWeatherResult>string</GetWeatherResult>
    </GetWeatherResponse>
  </soap:Body>
</soap:Envelope>
```

Concluído

Internet

Concluído

Meus documentos web service clima - P... Aula\_de\_WebServic... Webservice de CEP ... GlobalWeather Web ... Maniezo Webdeve... <http://www.webs...> PT 00:50



## Tecnologías Envolvidas

- XML
- SOA
- SOAP
- WSDL
- UDDI



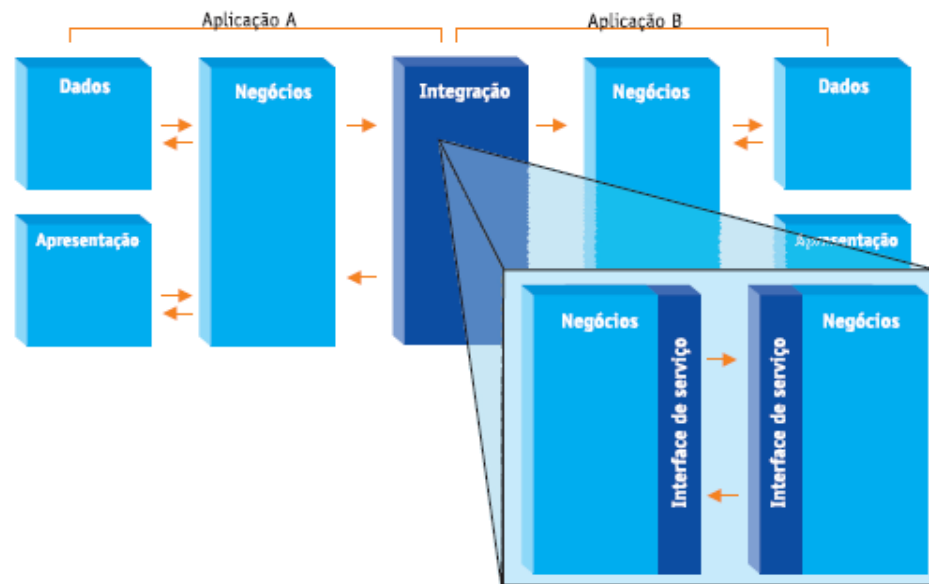
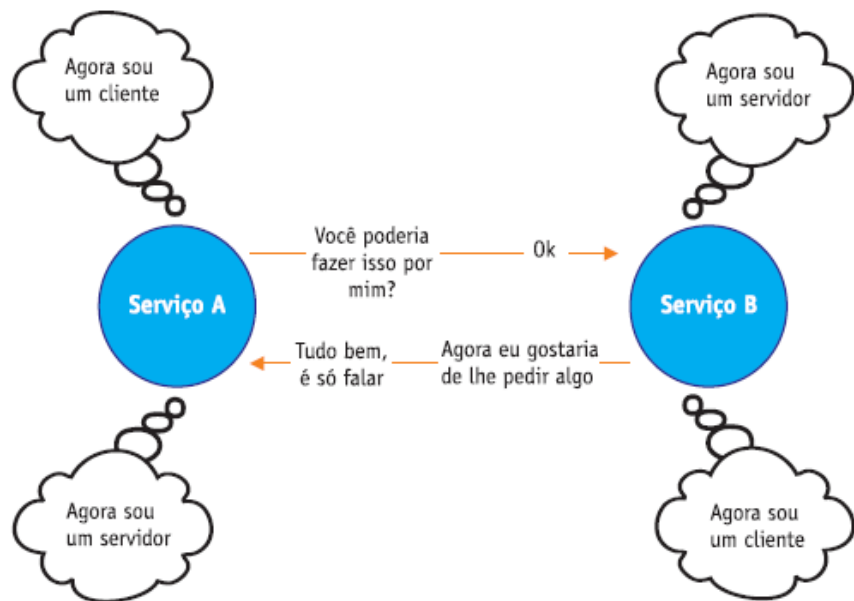


## XML

- XML (Extensible Markup Language) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

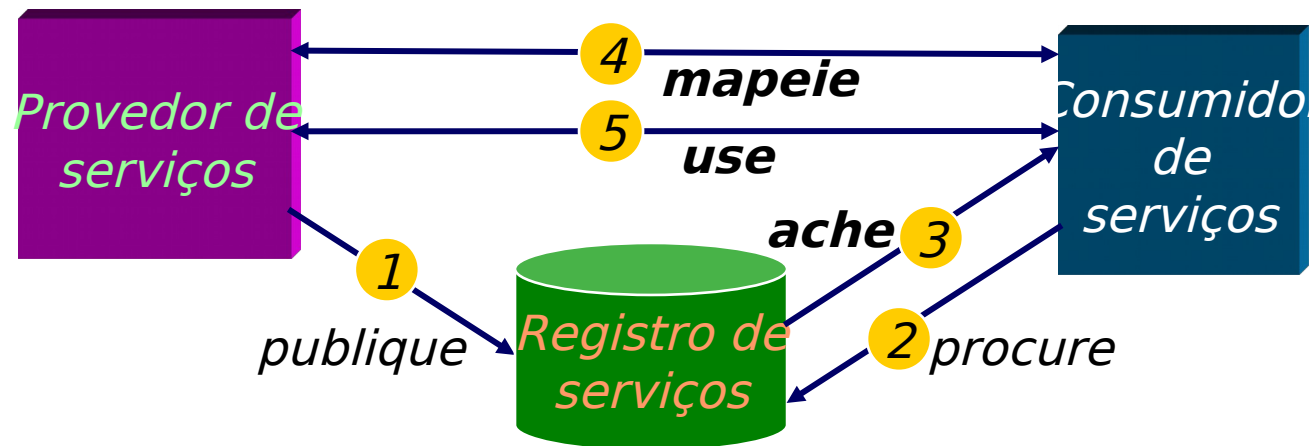
# SOA

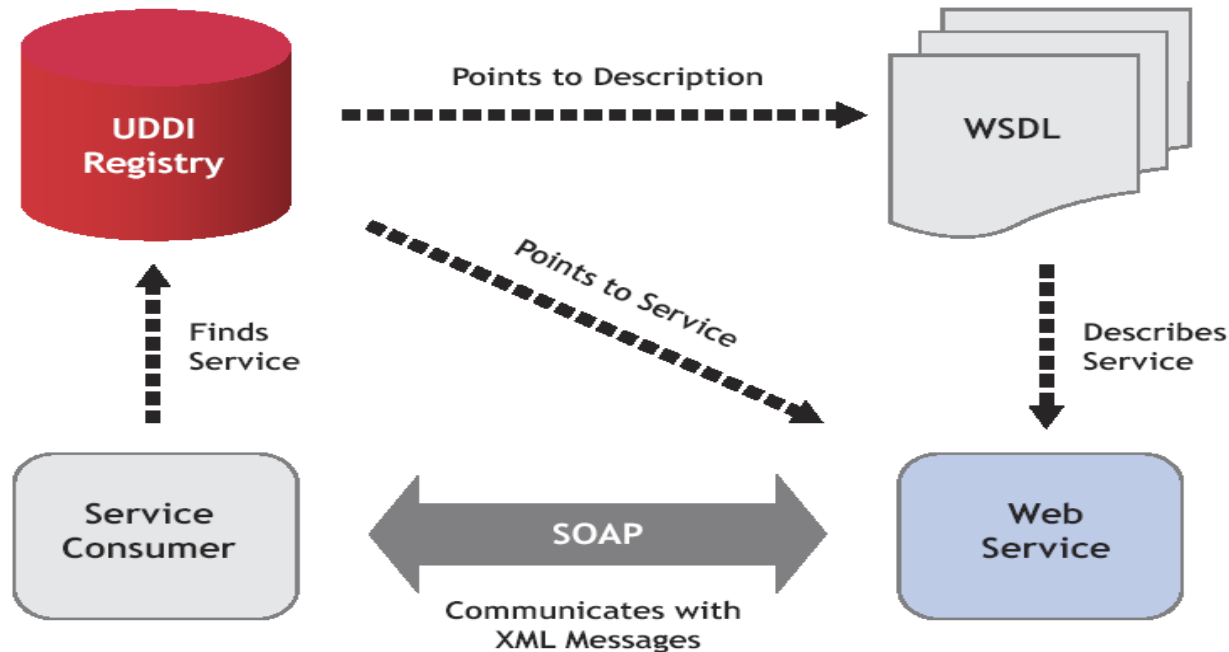
- A arquitetura orientada a serviços (SOA) descreve uma categoria de aplicativos compostos formados pelos componentes **provedor de serviço e consumidor de serviço**.
- Uma SOA é um modelo de projeto com um conceito profundamente amarrado à questão do encapsulamento de aplicação.
- A **arquitetura** resultante estabelece essencialmente um **paradigma de projeto**, no qual web services são os blocos de construção chave.
- **Serviços fracamente acoplados**
- **Vantagens: interoperabilidade, redução de custos e reuso**



# SOA

- Service Oriented Architecture
  - Novas palavras para um conceito antigo:
  - Arquitetura para aplicações baseadas em serviços (que usam serviços como componentes básicos)
  - Um serviço é a implementação de uma funcionalidade de negócio bem definida; aplicações com arquitetura SOA **combinam serviços para oferecer outros serviços**
- A principal forma de **implementar SOAs** hoje é através de **Web Services**
  - O motivo? **A interoperabilidade!**
- Find-Bind-Execute (Ache, Mapeie, Use) – paradigma central de um SOA





*A arquitetura é baseada em três componentes:*

- ✓ *UDDI – Universal Description Discovery and Integration*
- ✓ *WSDL – Web Service Description Language*
- ✓ *SOAP – Simple Object Application Protocol*

# SOA

Importância da arquitetura:

Um estudo de caso na CAIXA ECONÔMICA FEDERAL (CEF)

- *“Nada será como antes na relação da Caixa Econômica Federal com seus clientes. Agora ao entrar em uma agência para abrir uma conta corrente, apenas um processo colocará a disposição do usuário: número de conta, talão de cheques, acesso ao internet banking, informações de cadastro e uma linha de crédito aprovada. Estas facilidades fazem parte do novo processo de abertura de conta corrente de pessoa física, qe está sendo implantado nas agências do Banco, com base na arquitetura SOA considerada última etapa evolutiva do mercado mundial de software.”*



# Simple Object Access Protocol -SOAP

## SOAP - Características

- Protocolo de comunicação baseado em XML;
- Iniciado em 1999 pela W3C;
- Permite que mensagens sejam trocadas entre computadores heterogêneos;
- Pode passar por firewalls;



## SOAP

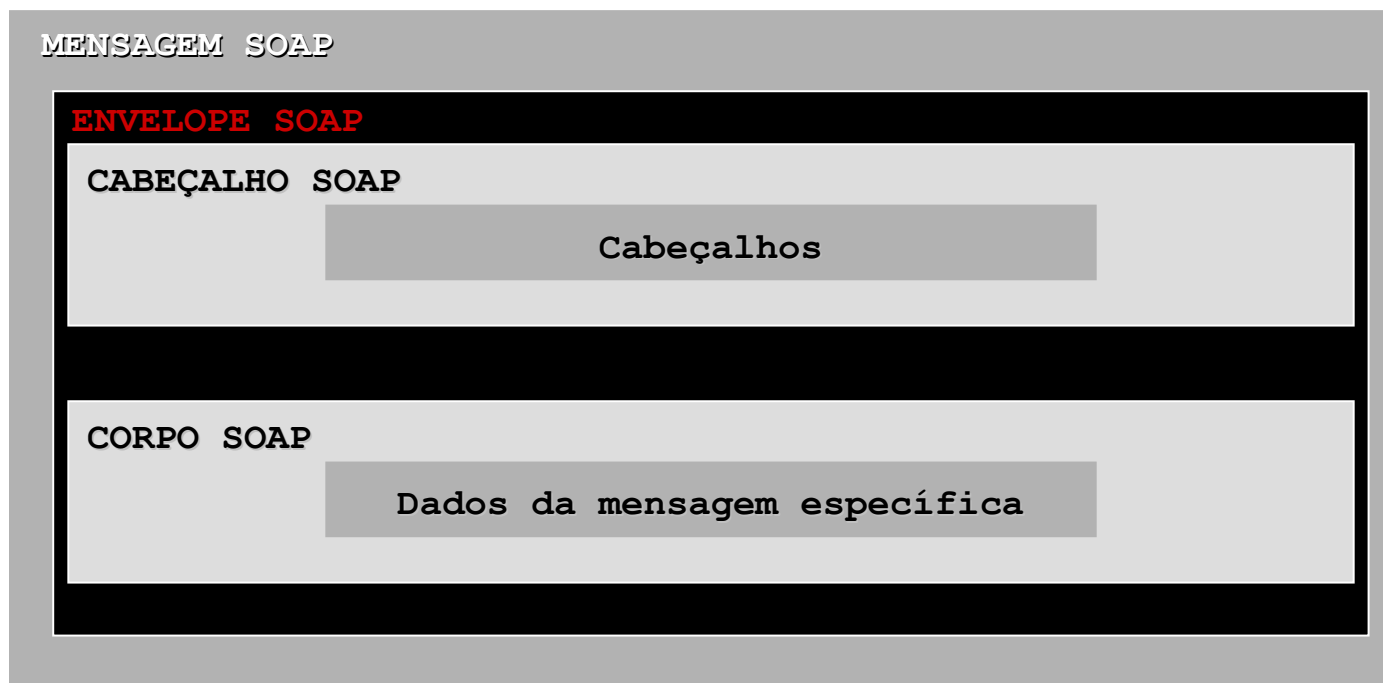
### **SOAP - Objetivo**

- O objetivo atrás do trabalho SOAP é ser um protocolo que é neutro para qualquer coisa:
  - Transporte
  - Linguagem de Programação
  - Sistema Operacional
  - Etc.

# SOAP

## ESTRUTURA GERAL

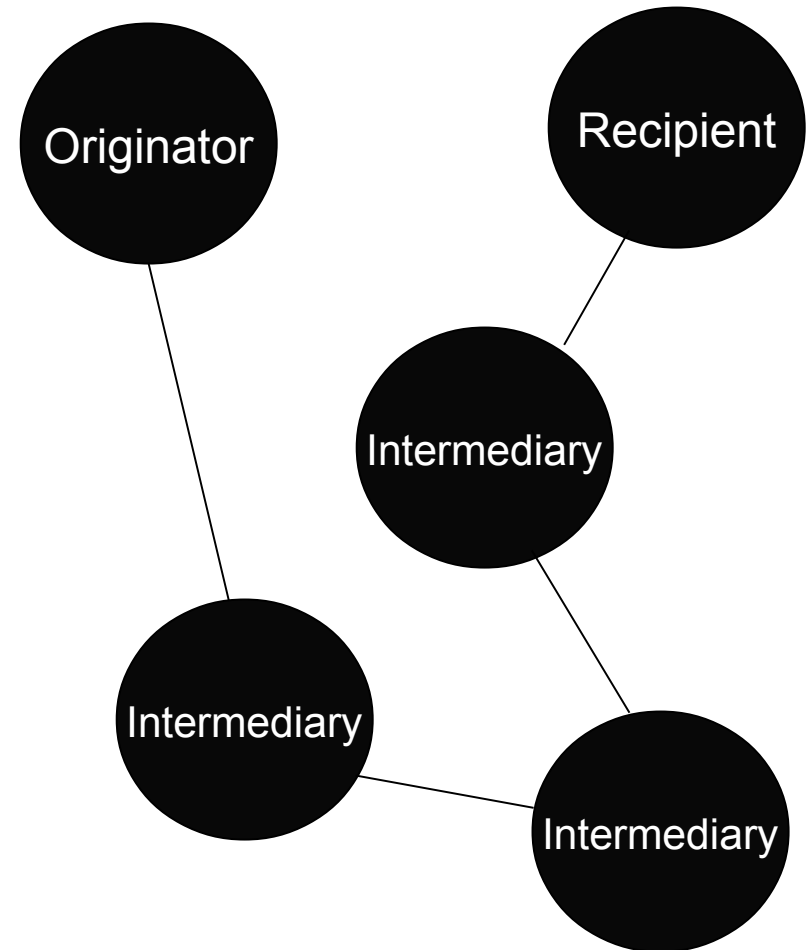
- Envelope (Envelope):
  - Define o conteúdo da mensagem
  - OBRIGATÓRIO estar associada com o namespace do envelope SOAP:
    - <http://www.w3.org/2001/06/soap-envelope>
- Cabeçalho (Header) (é opcional):
  - contém informação de controle e processamento.
- Corpo (Body):
  - contém informação da chamada e da resposta





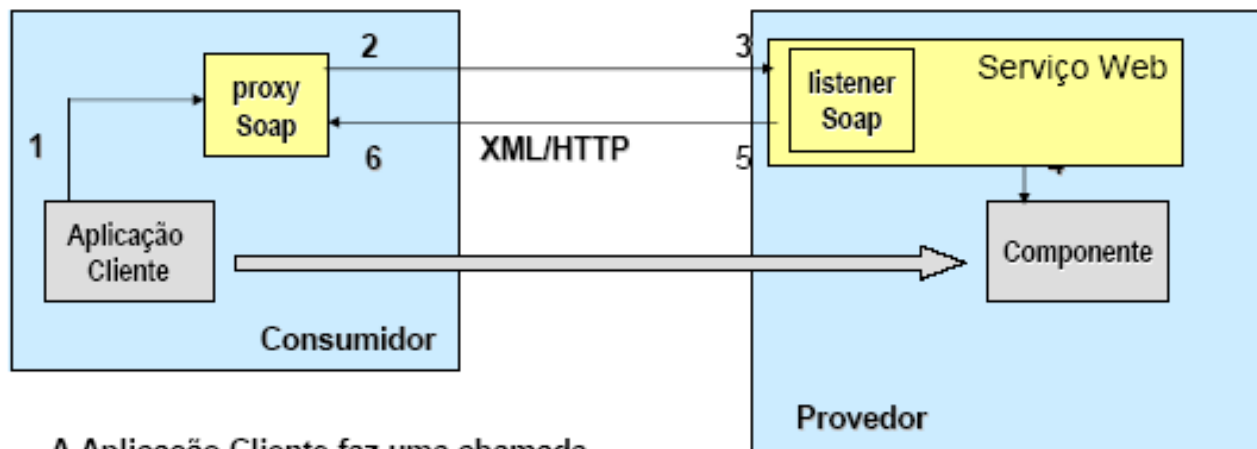
## SOAP- Funcionamento

- SOAP assumes messages have an *originator*, one or more *ultimate receivers*, and zero or more *intermediaries*.
- The reason is to support distributed message processing.
- That is, we can go beyond client-server messaging.



# SOAP

## SOAP



1. A Aplicação Cliente faz uma chamada
2. O Proxy intercepta a chamada, constrói e transmite a mensagem de requisição XML
3. O listener Soap recebe, analisa gramaticalmente e valida a requisição
4. O Listener chama a mensagem do componente
5. O Listener pega o resultado da chamada, constrói e transmite a msg. XML de resposta
6. O Proxy recebe, analisa gramaticalmente a resposta, e retorna o resultado ao cliente

**Este processo é transparente para o cliente e o componente**



## Exemplo 1- SOAP

```
<?xml version="1.0"?>
  <soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
    </m:GetPrice>
  </soap:Body>
</soap:Envelope>
```

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
<m:Price>1.90</m:Price>
    </m:GetPriceResponse>
  </soap:Body>
</soap:Envelope>
```



# SOAP [1/3] - Travel Reservation

SOAP Version 1.2 Part 0: Primer (Second Edition) - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://www.w3.org/TR/soap12-part0/#Example

Google

W3C Recommendation

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Áke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>New York</p:departing>
        <p:arriving>Los Angeles</p:arriving>
        <p:departureDate>2001-12-14</p:departureDate>
        <p:departureTime>late afternoon</p:departureTime>
        <p:seatPreference>aisle</p:seatPreference>
      </p:departure>
      <p:return>
        <p:departing>Los Angeles</p:departing>
        <p:arriving>New York</p:arriving>
        <p:departureDate>2001-12-20</p:departureDate>
        <p:departureTime>mid-morning</p:departureTime>
        <p:seatPreference/>
      </p:return>
    </p:itinerary>
    <q:lodging
      xmlns:q="http://travelcompany.example.org/reservation/hotels">
      <q:preference>none</q:preference>
    </q:lodging>
  </env:Body>
</env:Envelope>
```

Concluído

# SOAP [2/3] - Resposta da solicitação

SOAP Version 1.2 Part 0: Primer (Second Edition) - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://www.w3.org/TR/soap12-part0/#Example

## Example 2

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Áke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itineraryClarification
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>
          <p:airportChoices>
            JFK LGA EWR
          </p:airportChoices>
        </p:departing>
      </p:departure>
      <p:return>
        <p:arriving>
          <p:airportChoices>
            JFK LGA EWR
          </p:airportChoices>
        </p:arriving>
      </p:return>
    </p:itineraryClarification>
  </env:Body>
</env:Envelope>
```

SOAP message sent in response to the message in [Example 1](#)

# SOAP [3/3] – Escolha do Aeroporto

W3C Recommendation

SOAP Version 1.2 Part 0: Primer (Second Edition) - Mozilla Firefox

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://www.w3.org/TR/soap12-part0/#Example

Example 3

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <m:reservation
      xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
      <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
    </m:reservation>
    <n:passenger xmlns:n="http://mycompany.example.com/employees"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
      env:mustUnderstand="true">
      <n:name>Åke Jógvan Øyvind</n:name>
    </n:passenger>
  </env:Header>
  <env:Body>
    <p:itinerary
      xmlns:p="http://travelcompany.example.org/reservation/travel">
      <p:departure>
        <p:departing>LGA</p:departing>
      </p:departure>
      <p:return>
        <p:arriving>EWR</p:arriving>
      </p:return>
    </p:itinerary>
  </env:Body>
</env:Envelope>
```

Response to the message in [Example 2](#) continuing a conversational message exchange

Concluído



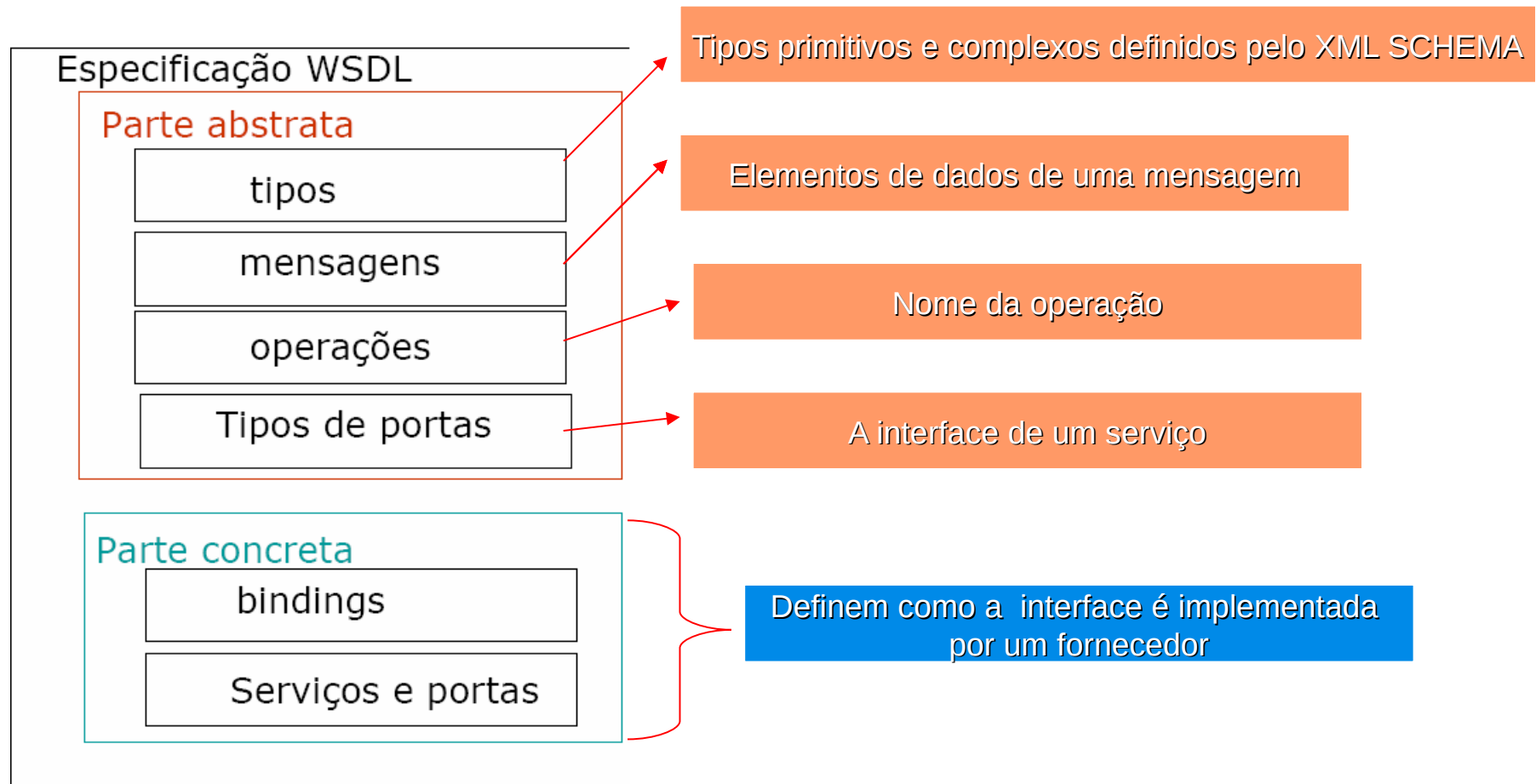
## WSDL

### O que é um Arquivo WSDL?

- Um arquivo WSDL- Web Service Description Language, é descrição de serviço Web através de um documento XML.
- Criado por Microsoft, IBM, Ariba.

## WSDL

### O que WSDL Descreve?





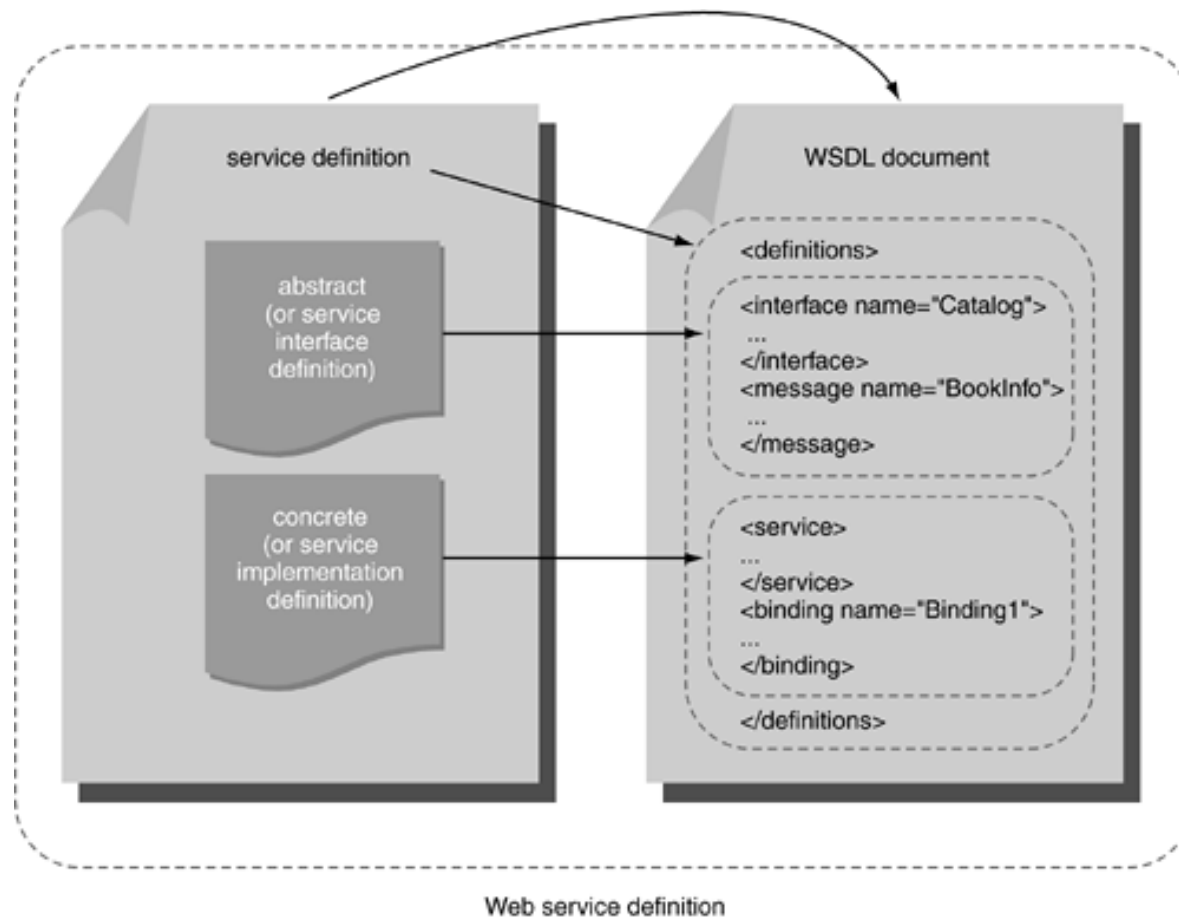


## Exemplo: WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="BookstoreService"
  targetNamespace="http://mybooks.org/wsdl"
  xmlns:tns="http://mybooks.org/wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <types>...</types>
  <message name="BookstoreIF_getPrice">
    <part name="String_1" type="xsd:string"/>
  </message>
  <message name="BookstoreIF_getPriceResponse">
    <part name="result" type="xsd:decimal"/>
  </message>
  <portType name="BookstoreIF">
    <operation name="getPrice" parameterOrder="String_1">
      <input message="tns:BookstoreIF_getPrice"/>
      <output message="tns:BookstoreIF_getPriceResponse"/>
    </operation>
  </portType>
  <binding ... > ...</binding>
  <service ... > ... </service>
</definitions>
```

*Informa onde está o serviço (endpoint)*

# WSDL



## WSDL

### Exemplo de Definição de Interface

```
<types>
  ...
</types>

<message name="MensagemPedido">
  <part name="NomeProduto" type="xs:string"/>
  <part name="Quantidade" type="xs:integer"/>
</message>

<message name="MensagemRetornoPedido">
  <part name="IdPedido" type="xs:integer"/>
</message>

<portType name="TipoPortaAquisicao">
  <operation name="pedirProdutos">
    <input message = "MensagemPedido"/>
    <output message = "MensagemRetornoPedido"/>
  </operation>
</portType>
```



## WSDL

### Exemplo de Implementação de Serviço

```
<binding name="LigacaoAquisicao" type="tns:TipoPortaAquisicao">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="pedirProdutos">
    <soap:operation
      soapAction="http://exemplo.com/pedirProdutos"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>

<service name="ServicoAquisicao">
  <port name="PortaAquisicao" binding="tns:LigacaoAquisicao">
    <soap:address location="http://exemplo.com/aquisicao"/>
  </port>
</service>
```



## Exemplo Completo

- Cenário : Um web service para reserva de hotel.
  - Funcionalidade
    - *CheckAvailability*. To check availability, the client must specify a **check-in date**, a **check-out date**, and **room type**. The Web service will **return a room rate (a floating point number in USD\$)** if such a room is available, or a zero room rate if not. If any input data is invalid, the service should return an error. **Thus, the service will accept a checkAvailability message and return a checkAvailabilityResponse or invalidDataFault message**



## WSDL - Element Type

```
<types>
  <xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://greath.example.com/2004/schemas/resSvc"
    xmlns="http://greath.example.com/2004/schemas/resSvc">

    <xs:element name="checkAvailability" type="tCheckAvailability"/>
    <xs:complexType name="tCheckAvailability">
      <xs:sequence>
        <xs:element name="checkInDate" type="xs:date"/>
        <xs:element name="checkOutDate" type="xs:date"/>
        <xs:element name="roomType" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>

    <xs:element name="checkAvailabilityResponse" type="xs:double"/>

    <xs:element name="invalidDataError" type="xs:string"/>

  </xs:schema>
</types>
```



## WSDL - Interface Element

```
<interface name = "reservationInterface" >

  <fault name = "invalidDataFault"
    element = "ghns:invalidDataError"/>

  <operation name="opCheckAvailability"
    pattern="http://www.w3.org/2006/01/wsdl/in-out"
    style="http://www.w3.org/2006/01/wsdl/style/iri"
    wsdlx:safe = "true">
    <input messageLabel="In"
      element="ghns:checkAvailability" />
    <output messageLabel="Out"
      element="ghns:checkAvailabilityResponse" />
    <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
  </operation>

</interface>
```

## WSDL - Binding Element

```
<binding name="reservationSOAPBinding"
  interface="tns:reservationInterface"
  type="http://www.w3.org/2006/01/wsdl/soap"
  wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP">

  <fault ref="tns:invalidDataFault"
    wsoap:code="soap:Sender"/>

  <operation ref="tns:opCheckAvailability"
    wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

</binding>
```

O elemento binding especifica um formato concreto de mensagem e protocolo da interface e deve suportar as mensagens e erros da interface.

Como acessar o serviço



## WSDL - Service Element

```
<service name="reservationService"
  interface="tns:reservationInterface">

  <endpoint name="reservationEndpoint"
    binding="tns:reservationSOAPBinding"
    address ="http://greath.example.com/2004/reservation"/>

</service>
```

### Onde acessar o servico

```
<?xml version="1.0" encoding="utf-8" ?>
<description
  xmlns="http://www.w3.org/2006/01/wsdl"
  targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
  xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
  xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
  xmlns:wsoap= "http://www.w3.org/2006/01/wsdl/soap"
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
  xmlns:wsdlix= "http://www.w3.org/2006/01/wsdl-extensions">
```



# Arquitetura de Web Services: camadas

- Camada de transporte (protocolo largamente utilizado)
  - Principais: HTTP (POST), FTP, SMTP
- Camada de mensagens
  - SOAP
- Camada dados ou serviços
  - XML (formato de mensagens)
  - XML-RPC
- Camada de descrição de serviços
  - WSDL
- Camada de descoberta (registro)
  - UDDI, ebXML

**Descoberta**

**Descrição**

**Dados**

**Mensagens**

**Transporte**

## UDDI

- Universal Description Discovery and Integration
  - Publicação e Descoberta de serviços
  - Criado por Microsoft, IBM, Ariba
  - Mantido pela OASIS ([www.oasis-open.org](http://www.oasis-open.org))
    - “Consórcio que dirige o desenvolvimento, convergência e adoção de *e-business patterns*”
    - OASIS UDDI Specification Committee
- Especificações
  - Esquema para descrição de provedores de serviços
  - API SOAP para publicação e descoberta de serviços
  - Baseado nos padrões da Web
    - XML, HTTP, TCP/IP, SOAP
  - Suporta serviços XML e outros modelos



## UDDI

### Como o UDDI é usado?

- analistas de negócios:
  - Para procurar por serviços
    - similar a máquinas de busca
- Desenvolvedores:
  - Para publicar serviços
  - Para escrever software que usa os serviços descobertos
- Incorporados em *toolkits*:
  - para automatizar a publicação de serviços

# UDDI

Dados dentro do UDDI são categorizados em 3 tipos

White Pages
Yellow Pages
Green Pages

- white pages (páginas brancas)
  - info básicas de contato sobre empresas
  - permite associação de Ids únicos aos empresários
- yellow pages (páginas amarelas)
  - info geral de classificação sobre a empresa ou Serviço que ela oferece (e.g. NAICS: códigos de indústria, UNSPC: produtos & Serviços clasf, ISO: geog)
- green pages (páginas verdes)
  - info técnica sobre um Serviço Web
  - referência a uma especificação externa (e.g. WSDL)
  - endereço para chamada do Serviço

# UDDI

## Estrutura

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





## UDDI

### Descrições de Serviço

- Identificador único programável para um dado tipo qualquer de Serviço Web
- Usado por entidades padrão e desenvolvedores para “publicar” como estes serviços trabalham
- Usado como uma assinatura por web sites que implementam aquelas interfaces
- Armazenados no UDDI como “*tModels*”

## UDDI

### ➤ Registros Públicos

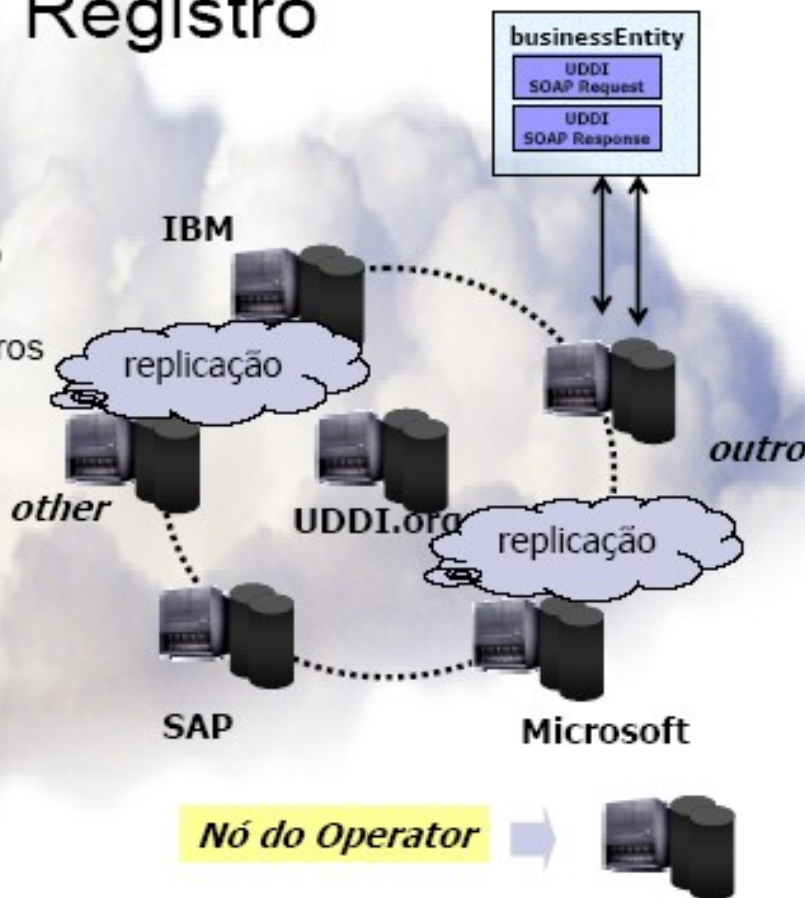
- conteúdo inserido no UBR (Universal Business Registry) é feito em um único nó que se torna proprietário mestre do conteúdo
- conteúdo pode ser acessado de qualquer nó
- qualquer negócio pode configurar um nó operador
- empresas podem configurar:
  - nós privados
  - *clouds* (nuvens) privadas
- caiu em desuso:
  - Microsoft, IBM, Outras, cancelaram o registro público alegando (Microsoft) falta de maturidade da tecnologia, entre outros problemas.



# UDDI

## Operação de Registro

- Peer nodes (websites)
- Empresas se registram em qualquer dos nós
- Registros das empresas são replicados diariamente
- Conjunto completo de registros "cadastrados" disponível em todos os nós
- Conjunto comum de APIs SOAP suportados por todos os nós





## UDDI

### Registros Privados

- Descoberta Direta
- Incluem funcionalidades adicionais de segurança
- Serviços acessíveis apenas de dentro da organização ou de grupos de parceiros confiáveis
- Podem ser usados para operações internas ou B2B
- Permite que as empresas forneçam serviços personalizados para clientes
- Empresas estão adotando registros privados mais rápido que os públicos

# UDDI

## Tipos de Registros Privados

- e-marketplace UDDI
  - hospedado por uma indústria, lista negócios e serviços de membros do consórcio
- portal UDDI
  - publica Serviços Web de empresas
  - só busca (*find*) é disponível externamente
- partner catalog UDDI
  - disponível apenas para empresas membro
  - *publish* e *find* restritos a usuários autorizados
  - reside atrás de um firewall
- internal UDDI
  - disponível apenas para uma única empresa
  - reside atrás de um firewall

# UDDI

## ➤ API de busca (Inquiry)

- Busca coisas
  - find\_business
  - find\_service
  - find\_binding
  - find\_tModel
- Obtém detalhes sobre coisas
  - get\_businessDetail
  - get\_serviceDetail
  - get\_bindingDetail
  - get\_tModelDetail

## ➤ API de Publicação (Publish)

- Salva coisas
  - save\_business
  - save\_service
  - save\_binding
  - save\_tModel
- Deleta coisas
  - delete\_business
  - delete\_service
  - delete\_binding
  - delete\_tModel
- Segurança...
  - get\_authToken
  - discard\_authToken

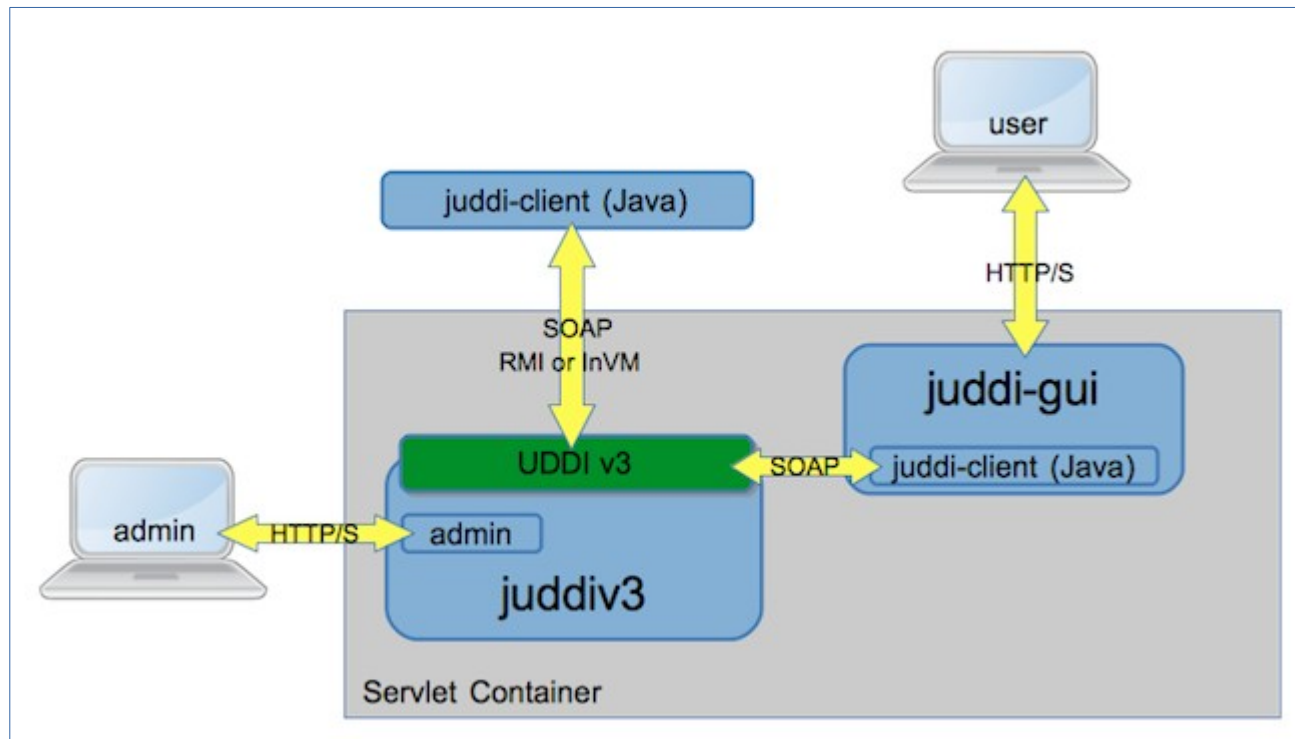
The screenshot shows a web browser window titled "Add a new Service - Microsoft Internet Explorer". The address bar shows the URL: `http://localhost:1010/uddi/uddiPublishService?action=addobject-service`. The page header features the IBM logo and the text "IBM UDDI Registry" and "Universal Description, Discovery and Integration". A left sidebar contains a menu with links: "UDDI Register", "UDDI Login", "UDDI Logout", "UDDI Publish", and "UDDI Find". The main content area is titled "Add a new Service" and includes a note: "The fields indicated with an asterisk (\*) are required; other fields are optional. If you don't wish to add the service please satisfied with the information you have entered press the Continue button." Below this, there is a section "Enter the Name of your service" with a text input field labeled "Name\*" containing the text "Stock Quote". At the bottom of the form are three buttons: "Cancel", "Continue", and "Help".

## UDDI

### Limitações do UDDI

- Ainda evoluindo
- Registros Públicos – confiabilidade de dados?
  - não há data da “última-atualização”
  - não há verificações de validade
- Qualidade-do-Serviço de um Serviço Web?
  - com que frequência ele pode ser acessado?
  - escalabilidade?
  - suporte técnico?
  - Etc.

# UDDI





<https://juddi.apache.org/>

Apache jUDDI ▾ Downloads ▾ Documentation ▾ Subprojects ▾ Misc ▾ Modules ▾ Project Documentation ▾

Fork me on GitHub



Apache  / Introduction

Version: 3.3.6 | Last Published: 05 Dec 2018

APACHE JUDDI

Welcome

Live Demos ▾

Prod UI 

Prod SVC 

Sandbox UI 





Sandbox SVC 

DOWNLOADS

Releases

Source Code

## Introduction

jUDDI (pronounced “Judy”) is an open source Java implementation of [OASIS](#)  the [Universal Description, Discovery, and Integration \(UDDI\)](#)  specification for (Web) Services. The jUDDI project includes [Scout](#) . Scout is an implementation of the [JSR 93 - Java™ API for XML Registries 1.0 \(JAXR\)](#) .

## Features

- Open Source
- Platform Independent
- Use with any relational database that supports ANSI standard SQL (MySQL, Oracle, DB2, Sybase, Derby etc.)
- Deployable on any Java application server that supports the Servlet 2.3 specification
- jUDDI registry supports a clustered deployment configuration.
- Easy integration with existing authentication systems
- Supports InVM embeddable mode



## Exemplo de Web Service: calculadora usando SOA

➤ Python  
<https://wiki.python.org.br/WebService>

➤ Java  
<https://www.devmedia.com.br/desenvolvendo-e-usando-web-services-em-java/37261>



**Prof. Inaldo Capistrano Costa - ITA**

[inaldo@ita.br](mailto:inaldo@ita.br)

