Centro de Ciências Exatas e Naturais
Departamento de Sistemas e Computação

SISTEMAS DISTRIBUÍDOS

MARCOS RODRIGO MOMO
marcos.rodrigomomo@gmail.com

Blumenau, outubro 2024.

Roteiro aulas Web Service

- ✓ Introdução a *Web Services*
- ✓ Padrões para desenvolvimento
- ✓ Web Service SOAP
- ✓ Atividades

Introdução aos Web Services

- Surgiram com a evolução dos modelos de computação distribuída
- RMI, DCOM e o CORBA, no entanto estas tecnologias tiveram mais sucesso na integração de softwares em ambientes de redes locais e homogêneas
- Quando a Internet avançou para o mercado corporativo surgiu a necessidade de integrar aplicações além das redes locais, ou seja, em ambientes heterogêneos
- É neste contexto que surge o a tecnologia que chamamos

Introdução aos Web Services

- É proveniente de grande empresas como IBM e Microsoft entre outras pertencentes ao W3C
- Une tecnologias para desenvolvimento de aplicações web e um padrão para o desenvolvimento de *web services (JSP, ASP e PHP ao XML)*
- Genericamente, *web services* são uma tecnologia para integração de sistemas, empregado principalmente em ambientes heterogêneos

Introdução aos Web Services

- Através de *web services* podemos desenvolver softwares para interagir com outros softwares não importando a linguagem de programação em que foram desenvolvidos, a plataforma ou hardware
- A única premissa é que para se comunicar com os *web services* a troca de dados tem de ser feita no formato XML
- Os *web services*

Exemplos de Código XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<c
<titulo>Death Magnetic</titulo>
<artista>Metallica</artista>
<ano>2008</ano>
</cd>
```

Catalogando CDs e Receita para pão no formato XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<receita nome="Pão Caseiro" tempo_de_preparo="10 minutos" tempo_assar="40 minutos">
  <titulo>Pão Caseiro Simples</titulo>
  <ingredientes>
    <ingrediente quantidade="3" unidade="xícaras">Farinha</ingrediente>
    <ingrediente quantidade="7" unidade="gramas">Fermento</ingrediente>
    <ingrediente quantidade="1" unidade="xícaras" estado="morna">Água</ingrediente>
    <ingrediente quantidade="1" unidade="xícaras">Açúcar</ingrediente>
    <ingrediente quantidade="1" unidade="colheres de chá">Sal</ingrediente>
  </ingredientes>
  <instrucoes>
    <passo>Misture o fermento com o açúcar e espere aproximadamente cinco minutos.</passo>
    <passo>Misture todos os ingredientes, e dissolva bem.</passo>
    <passo>Cubra com um pano e deixe por uma hora em um local morno.</passo>
    <passo>Misture novamente, modele o formato e coloque na forma. Asse em forno médio.</passo>
  </instrucoes>
</receita>
```

Padrões para o desenvolvimento Web Services

- Existem dois padrões definidos:
  - SOAP – padrão que seguiremos
  - REST ou RESTFull

	WS-*	REST
Descrição do Serviço	WSDL	Nenhum padronizado
Descoberta	UDDI	Não possui
Comunicação	Síncrona/Assíncrona	Síncrona
Transações	Sim	Não
Estado*	Com estado	Sem estado
Protocolos	Vários	Somente HTTP
Multilinguagem	SIM	SIM
Bibliotecas XML	SIM	Não necessário
Segurança	HTTPS / WS-Security	HTTPS
Formato de Mensagem	SOAP/XML	Não padronizada

* Entende-se por estado que um *web service* em REST por padrão não possui nativamente mecanismos para determinar o estado anterior e sim apenas uma representação do estado atual.

Fonte: **http://saloon.inf.ufrgs.br/twiki-**

Padrões SOAP
(Simple Objeject Access Protocol)

- SOAP é um protocolo-padrão para transmissão de dados dentro da arquitetura de *web services* proposta pela W3C.

- É baseado no XML e segue o modelo:
  -

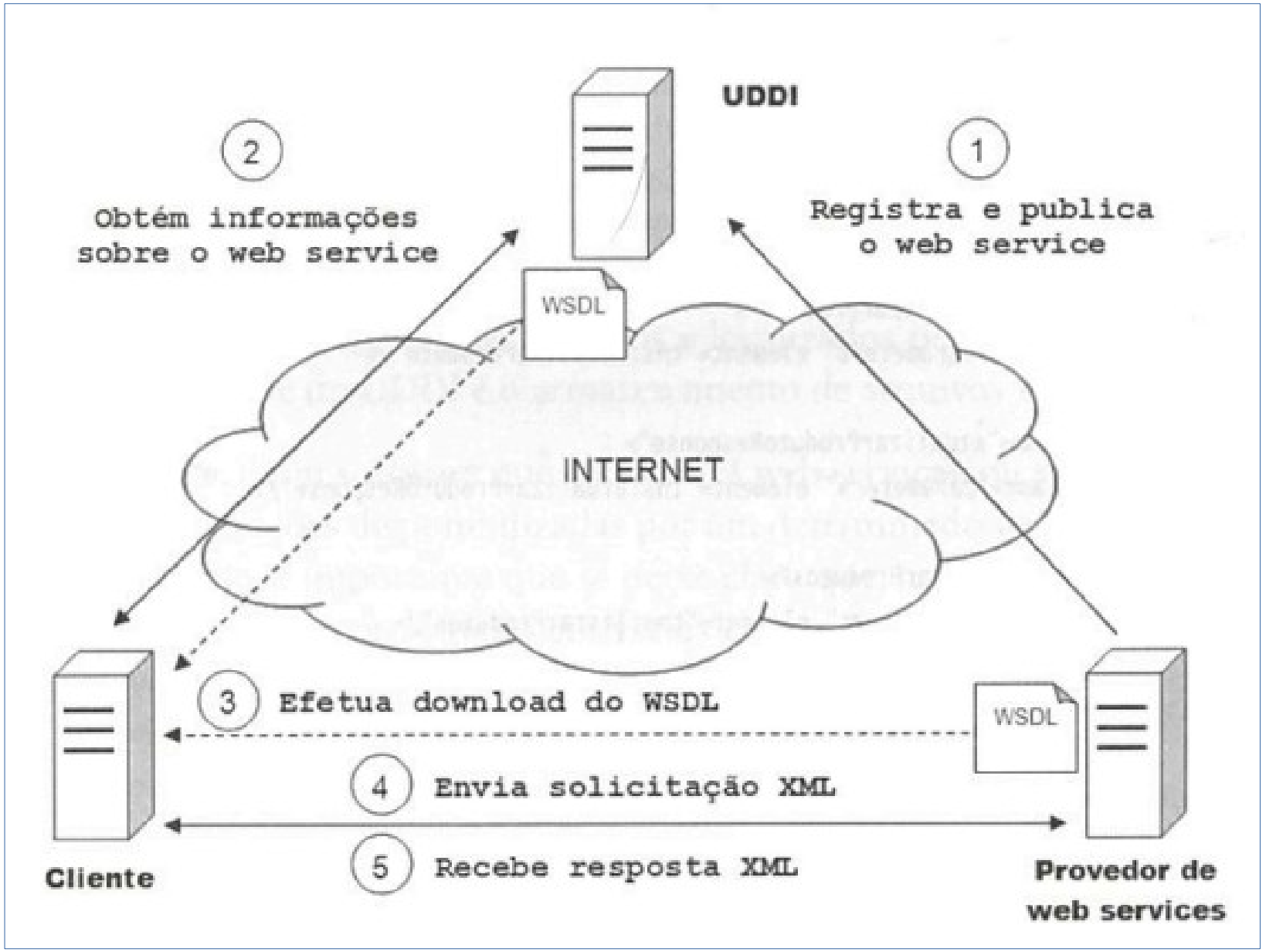
Arquitetura *Web Services* - SOAP

- WSDL (*Web Services Description Language*):
  - é um arquivo do tipo XML cuja finalidade é descrever detalhadamente um *web service,*
  - Especifica: operações e o formato de entrada e saída

Arquitetura *Web Services* - SOAP

- UDDI (Universal Description, Discovery and Integration):
  - a UDDI atende tanto o cliente quanto o provedor de *web services.*
  - Fornece meios para que os *web services* sejam registrados e publicados.
  - Permite que os *web services* sejam pesquisados e localizados pelos clientes
  - Também armazena WSDL

Arquitetura *Web Services* - SOAP



Arquitetura *Web Services* - SOAP

- Cliente:
  - É um software que consumirá um *web services.*
  - Utilizará as operações disponibilizadas por um determinado *web service*

- Provedor de *web services*:
  - É o componente que corresponde a um servidor de aplicações ou um web *container*.
  - Pode também armazenar arquivos WSDL.

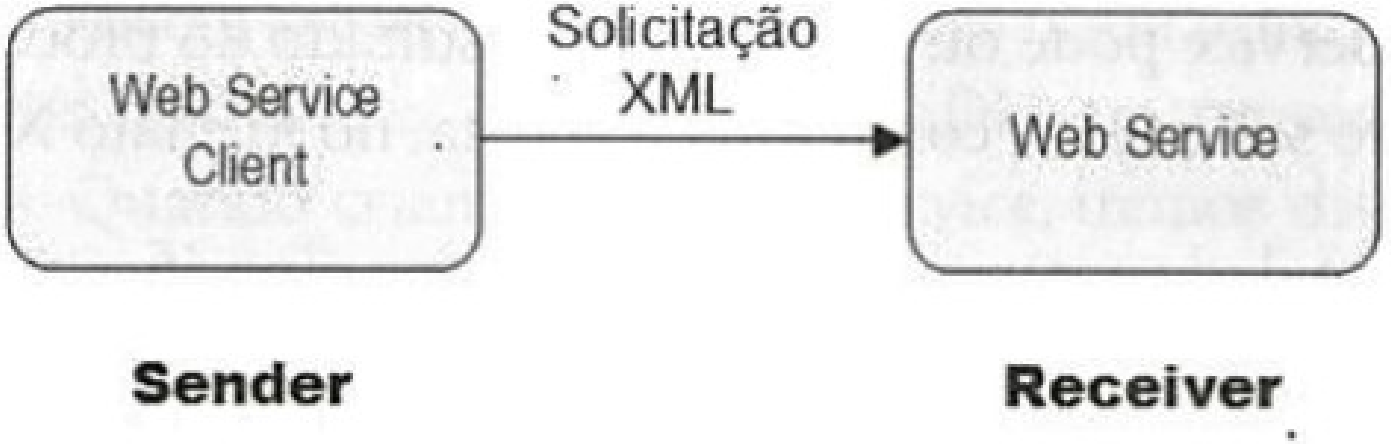
Funcionamento prático dos *Web Services* SOAP

Linguagem XML

- Todas as informações que são trocadas entre qualquer uma das partes são enviadas e recebidas por meio de mensagens no formato XML
- A arquitetura projetada pela W3C especifica que as solicitações e respostas XML possam trafegar por meio de qualquer protocolo como HTTP, FTP, SMTP, TCP puro etc.
- Na prática o que vemos é a utilização de XML, trafegando somente sobre HTTP

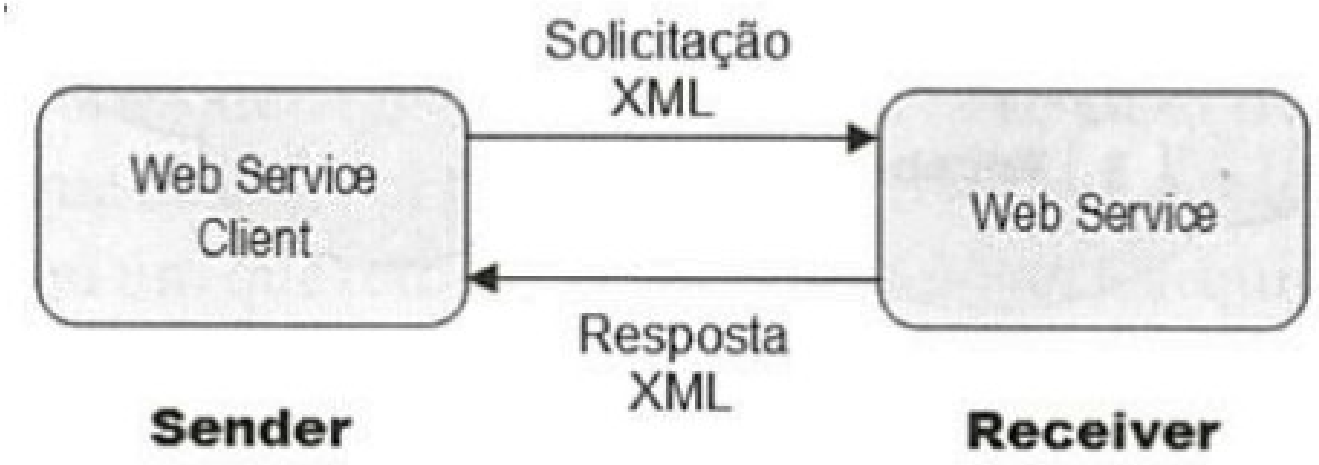
Envio de mensagens unilateral

- *One-Way-Messaging*:
  - O cliente envia a solicitação sem se preocupar com a resposta.
  - O W*eb Service* executará o processamento solicitado e não enviará resposta ao cliente



Envio de mensagens bilateral

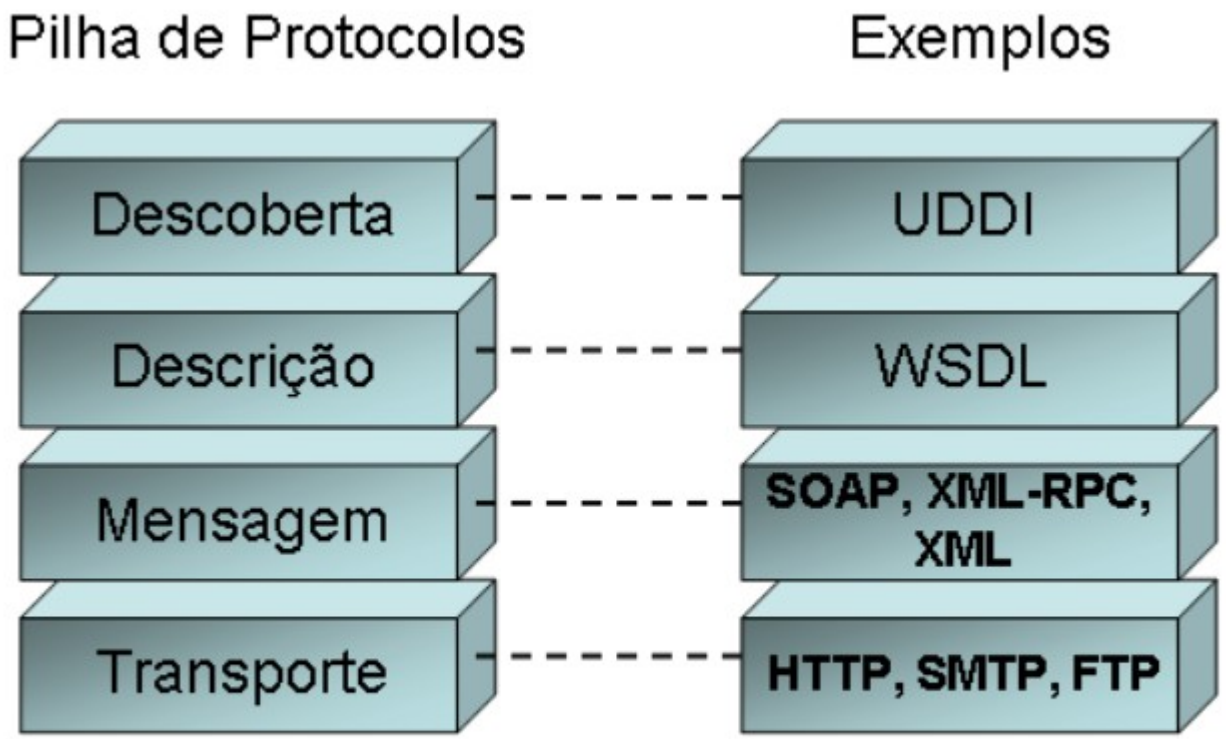
- *Request-Response Messaging*:
  - O cliente faz a solicitação ao *Web Service*
  - *Web Service* executa um processamento
  - Ao final envia o resultado ao cliente
  - A comunicação pode ser síncrona ou assíncrona



Atividades

1) O que são os Web Services?
2) Quais os objetivos e vantagens?
3) Descreva os protocolos usados em SOAP?
4) Descreva dois exemplos de aplicação Web Service:

Estrutura em camadas



Conjunto mais utilizado: HTTP + SOAP + WSDL + UDDI

Estrutura em camadas

- Nível de transporte
  - Responsável por transmitir os dados entre as aplicações
  - Pode utilizar protocolos já disponíveis pela internet (HTTP, SMTP, FTP, etc.)
- Nível de mensagem
  - Define o formato das mensagens enviadas entre as aplicações
  - Obs.:
    - Formato XML puro não garante interoperabilidade entre aplicações

Estrutura em camadas

- Nível de descrição
  - Representa a interface de um Web Service aos clientes
    - **Nome das operações disponíveis**
    - **Parâmetros**
    - **Protocolos do nível de transporte utilizados**
    - **Etc**
- Nível de descoberta
  - Registros onde é possível publicar um Web Service

Atividades práticas com Web Service

Utilizando Eclipse

ATIVIDADES DE LAB: Criação de *web services* *Calculadora*

- Nesta atividade criaremos
  - Um *web services*
    - *Apache Tomcat 7.0*
    - *Apache AXIS2 Framework*
  - Um cliente que irá utilizá-lo

- A aplicação baseada *web services* consiste em uma calculadora que terá as quatro operações: soma, subtração, divisão e multiplicação
  - *Web services* com quatro *web methods*

Criação de *web services*

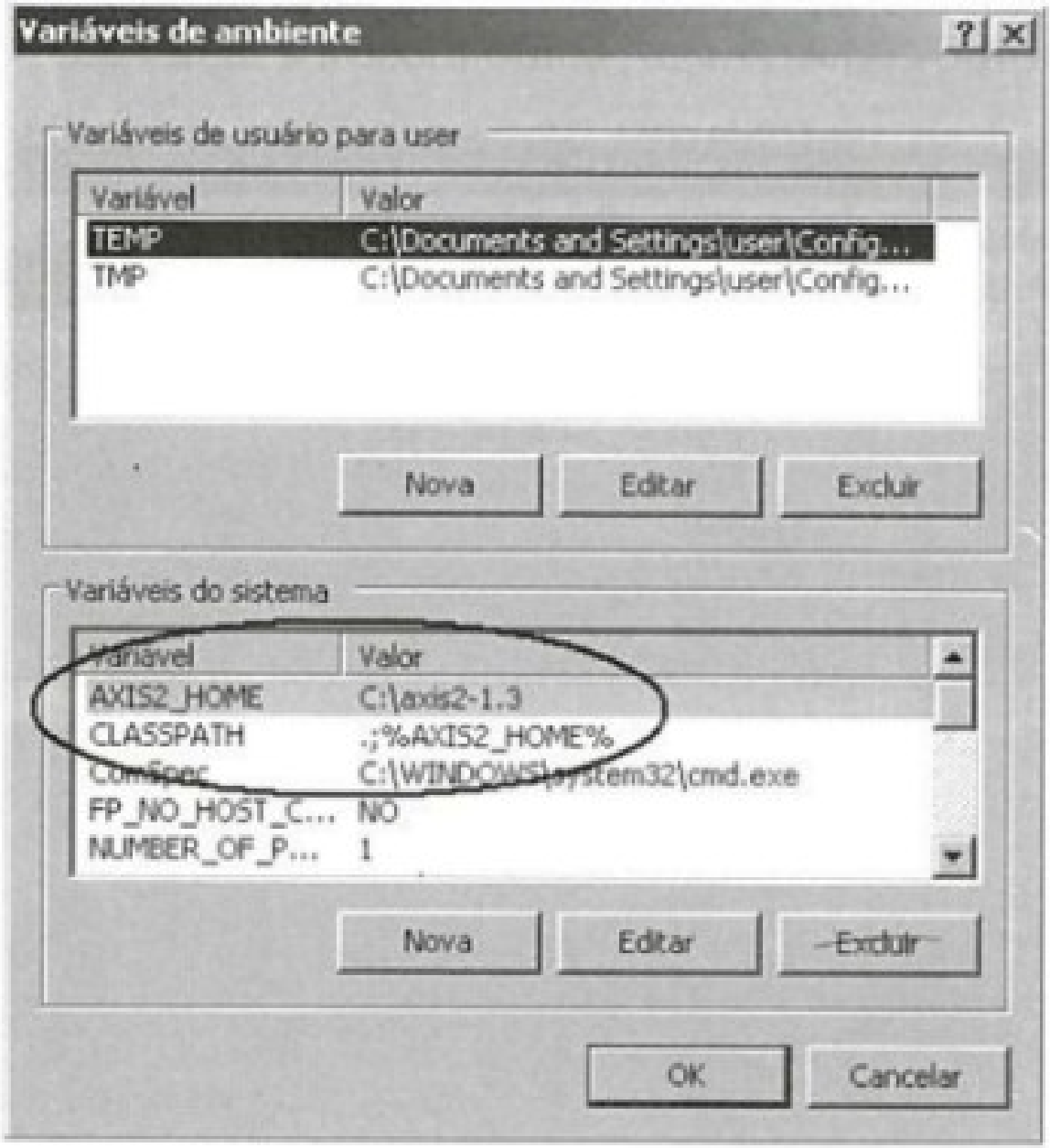
- Estas aplicações rodarão no *Tomcat* e serão criadas utilizando o framework *Apache Axis2*
- Configuração do ambiente:
  - IDE de desenvolvimento
  - Tomcat
  - Framework *Apache Axis2*
    - http://axis.apache.org/axis2/java/core/
- Baixar os dois arquivos:
- 1) axis xxxx.2-bin
- e 2) axisxxxx.2-war

Configuração do framework Apache Axis2

1. Descompacte o arquivo `axis2-1.3-bin.zip` e coloque a pasta descompactada na raiz do sistema operacional. O resultado será `C:\axis2-1.3`.

2. Agora teremos de acessar a janela "Variáveis de Ambiente" do sistema operacional. Para isso, clique com o botão direito do mouse sobre o ícone **Meu computador**, localizado na área de trabalho do Windows e, em seguida, na opção **Propriedades**. Depois, vá até a aba **Avançado** e clique no botão **Variáveis de ambiente**.

3. Crie a variável de ambiente `AXIS2_HOME` apontando para o caminho `C:\axis2-1.3`, em **Variáveis do sistema**, como exibe a figura 2.2.



4. Adicione a variável de ambiente `AXIS2_HOME` ao conteúdo da variável de ambiente `CLASSPATH`. Basta acrescentá-la ao final do conteúdo de `CLASSPATH` da seguinte forma: `%AXIS2_HOME%`. Utilize um ponto e vírgula (;) para separá-la do conteúdo já existente.

5. Da mesma forma, como descrito no item 4, adicione ao conteúdo da variável de ambiente `Path` o seguinte valor: `%AXIS2_HOME%\bin`. Não se esqueça do ponto e vírgula.

6. Descompacte o arquivo `axis2-1.3-war.zip` e coloque o arquivo descompactado `axis2.war` dentro da pasta `webapps` do Tomcat (`C:\Tomcat 6.0\webapps`). Depois, pare o serviço do Tomcat e inicie-o novamente. Será criada uma pasta `axis2` dentro da pasta `webapps` (`C:\Tomcat 6.0\webapps\axis2`).

7. Para verificar se a instalação e configuração realizadas tiveram sucesso, acesse o seguinte endereço: `http://localhost:8080/axis2/`. Se o conteúdo da página for igual ao exibido na figura 2.3 tanto o Tomcat quanto o Axis2 estão funcionando corretamente.


Axis 2 - Home - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8080/axis2/
Most Visited Getting Started Latest Headlines Hotmail gratuito Personalizar links Windows Media Windows
Axis 2 - Home
The Apache Software Foundation
http://www.apache.org/
AXIS2
Welcome!
Welcome to the new generation of Axis. If you can see this page you have successfully deployed the Axis2 Web Application. However, to ensure that Axis2 is properly working, we encourage you to click on the validate link.
Services
View the list of all the available services deployed in this server.
Validate
Check the system to see whether all the required libraries are in place and view the system information.
Administration
Console for administering this Axis2 installation.
Copyright © 1999-2006, The Apache Software Foundation
Licensed under the Apache License, Version 2.0
Done

Criação do Web Service

- Criar projeto chamado CalculadoraWS
- Criar o pacote chamado org.ws.calc
- Criar a classe chamada Calculadora.java

Classe Calculadora

- Terá quatro métodos:
  - Somar
  - Subtrair
  - Multiplicar
  - Dividir

```java
package org.ws.calc;

public class Calculadora {

        // Efetua a soma de dois valores
        public double soma(double i, double j) {
            double resultadoSoma;
            resultadoSoma = i + j;
            return resultadoSoma;
        }

        //Efetua a subtraçáo de dois valores
        public double subtracao(double i, double j){
            double resultadoSubtracao;
            resultadoSubtracao = i - j;
            return resultadoSubtracao;
        }

        //Efetua a multiplicaçáo de dois valores
        public double multiplicacao(double i, double j) {
            double resultadoMultiplicacao;
            resultadoMultiplicacao = i * j;
            return resultadoMultiplicacao;
        }

        //Efetua a divisáo de dois valores
        public double divisao(double i, double j) {
            double resultadoDivisao;
            resultadoDivisao = i / j;
            return resultadoDivisao;
        }
    }
```

Criar os arquivos XML

- No projeto CalculadoraWS criar na raiz uma pasta chamada resources
  - Dentro da pasta resources criar outra pasta com nome META-INF
  - Nesta pasta criar o arquivo XML chamado services.xml
    - O arquivo services.xml terá as informações sobre o web service tais como:
      - o nome, a descrição e a forma de como o web service poderá ser invocado pelo cliente e a classe que será a base para a construção do web service

Arquivo services.xml

- Os valores que devemos alterar de acordo com o projeto são:
  - Calculadora
  - http://org.ws/
  - http://org.ws/xsd
  - Org.ws.calc

Criar o arquivo build.xml

- Criar o arquivo build.xml na raiz do projeto CalculadoraWS
- O build é um arquivo ANT, para:
  - Criar o descritor do web service ou WSDL
  - Criar o próprio web service
    - Um arquivo com a extensão .aar
    - Posteriormente, este arquivo deverá ser colocado no web container

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="Calculadora" basedir=".">
    <property environment="env"/>
    <property name="AXIS2_HOME" value="C:/opt/axis2-1.3"/
>
    <property name="build.dir" value="build"/>
    <path id="axis2.classpath">
        <fileset dir="${AXIS2_HOME}/lib">
            <include name="*.jar"/>
        </fileset>
    </path>
```

build.xml

```
<target name="compile.service">
        <mkdir dir="${build.dir}"/>
        <mkdir dir="${build.dir}/classes"/>
        <javac debug="on" fork="true"
destdir="${build.dir}/classes"
            srcdir="${basedir}/src"
Classpathref="axis2.classpath">
        </javac>
    </target>
```

build.xml

```
<target name="generate.wsdl" depends="compile.service">
        <taskdef name="java2wsdl"
Classname="org.apache.ws.java2wsdl.Java2WSDLTask"
            Classpathref="axis2.classpath"/>
        <java2wsdl className="org.ws.calc.Calculadora"
            outputLocation="${build.dir}"
targetNamespace="http://org.ws/"
            schemaTargetNamespace="http://org.ws/xsd">
            <classpath>
                <pathelement path="${axis2.classpath}"/>
                <pathelement location="${build.dir}/classes"/>
            </classpath>
        </java2wsdl>
    </target>
```

build.xml

```
<target name="generate.service" depends="compile.service">
        <copy toDir="${build.dir}/classes" failonerror="false">
            <fileset dir="${basedir}/resources">
                <include name="**/*.xml"/>
            </fileset>
        </copy>
        <jar destfile="${build.dir}/Calculadora.aar">
            <fileset excludes="**/Test.class" dir="${build.dir}/classes"/>
        </jar>
    </target>
    <target name="clean">
        <delete dir="${build.dir}"/>
    </target>
</project>
```

build.xml