

# Web Service com Tomcat e Axis 1.4

*Por Valter Henrique*

A implementação do WS (Web Service) consiste em duas partes.

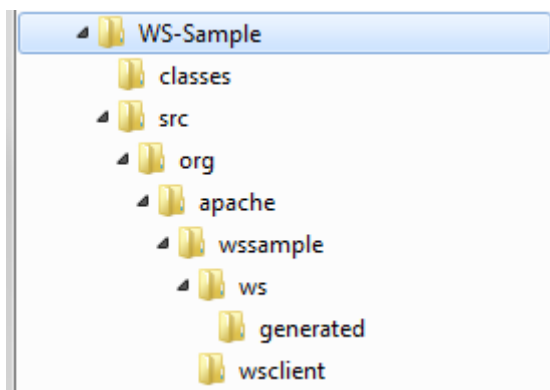
Primeiramente iremos fazer a parte do WS, uma calculadora, bem simples.

E colocaremos esta calculadora disponível no WS.

E depois iremos fazer a parte do cliente que irá utilizar esta calculadora, mas antes é necessário que este projeto tenha uma estrutura, tal como segue abaixo:

---

Estrutura do projeto



---

Parte do WEB Service

Calculadora

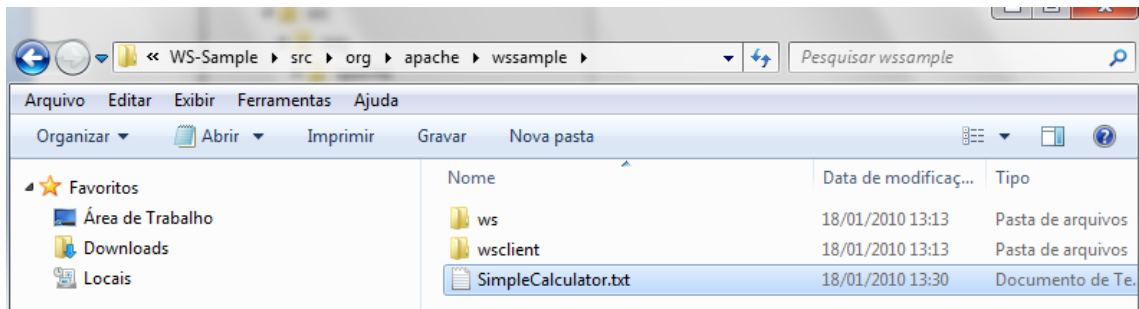
Primeiro precisamos fazer uma classe : “SimpleCalculator”, ela é uma classe totalmente independente do WS, mas iremos disponibilizar ela no WS posteriormente e sim você poderá fazer isto com suas aplicações também futuramente,rs.

```
package org.apache.wssample;

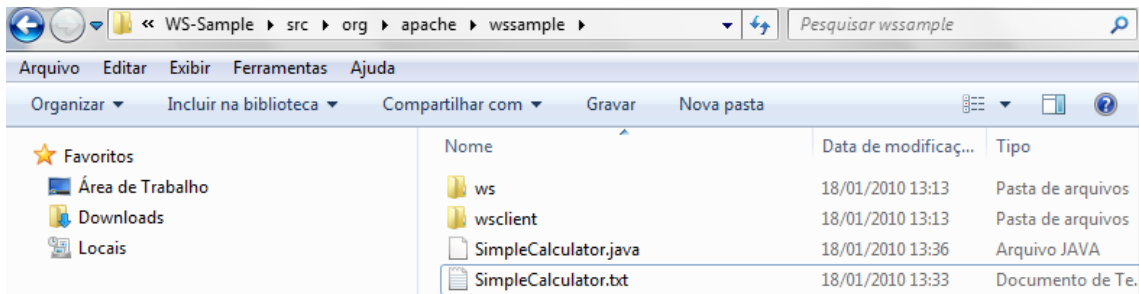
public class SimpleCalculator {

    public int add(int a, int b) {
        return a + b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
    public int multiply(int a, int b) {
        return a * b;
    }
}
```

\*Para quem tem dúvidas de como fazer estas classe:  
Crie um novo arquivo de texto no caminho :  
C:\WS-Sample\src\org\apache\wssample  
Desta forma:



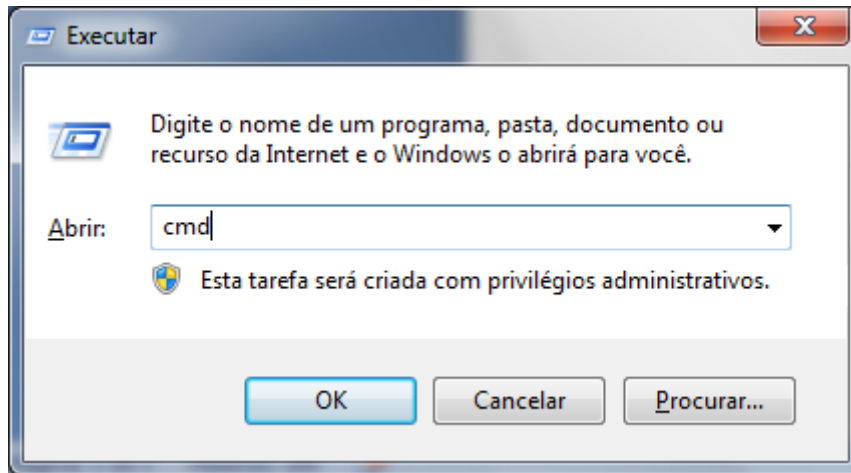
Abra o arquivo "SimpleCalculator.txt" e insira o código acima.  
Salve ele como SimpleCalculator.java ficando assim:



Podendo apagar o SimpleCalculator.txt agora, tudo bem? x)

Vamos compilar a classe acima com o comando no prompt :

- Aperte WINDOWS + R, ou botão iniciar -> executar:
- 

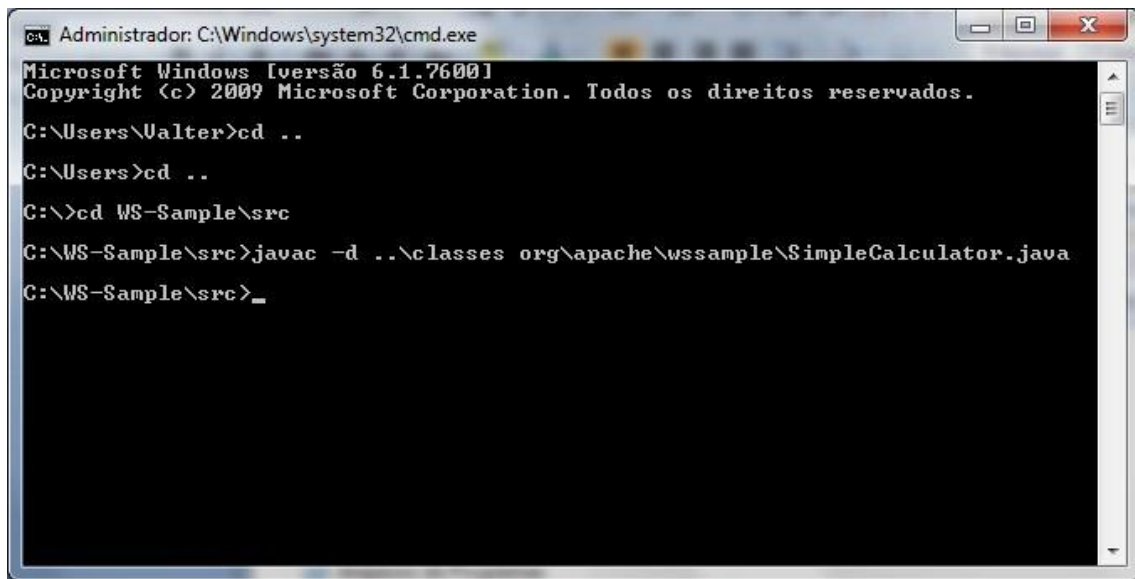


- O caminho destacado de amarelo é a pasta aonde você deve estar no momento, lembre-se disso.

Copie e cole o seguinte comando(destacado de verde):

```
WS-Sample\src> javac -d ../classes org\apache\wssample\SimpleCalculator.java
```

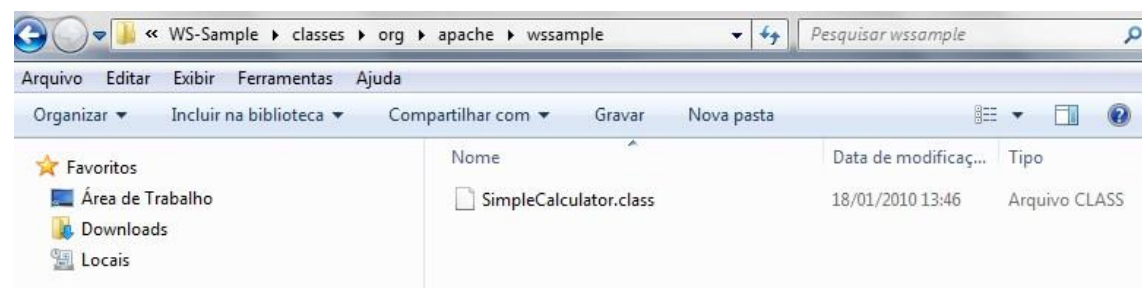
- Ficando desta maneira:



```
Administrador: C:\Windows\system32\cmd.exe
Microsoft Windows [versão 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Valter>cd ..
C:\Users>cd ..
C:\>cd WS-Sample\src
C:\WS-Sample\src>javac -d ..\classes org\apache\wssample\SimpleCalculator.java
C:\WS-Sample\src>_
```

Isto fará com que seja gerado o .class do SimpleCalculator, preservando a sua estrutura, o arquivo .class estará na pasta : C:\WS-Sample\classes\org\apache\wssample



## Interface do Web Service

Devemos determinar a interface que este WS terá.

Iremos usar apenas dois métodos através do nosso serviço, os métodos “add()” e “subtract()”.

Poderia ser qualquer número aqui, mas estamos fazendo isso para enfatizar que ela poderia ser da maneira que quiséssemos, ok?

```
package org.apache.wssample.ws;

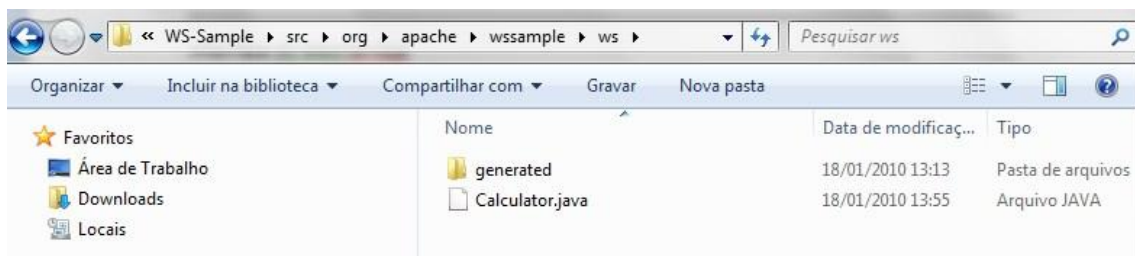
public interface Calculator {

    int add (int x, int y);

    int subtract(int x, int y);

}
```

Que deve estar neste caminho: C:\WS-Sample\src\org\apache\wssample\ws



Note, esta na pasta ws, pois ele é o nosso ws.

Compile usando o comando:

```
WS-Sample\src>javac -d ..\classes org\apache\wssample\ws\Calculator.java
```

Ficando:

```
C:\WS-Sample\src>javac -d ..\classes org\apache\wssample\ws\Calculator.java
C:\WS-Sample\src>_
```

---

## Java2WSDL – Gerando o arquivo WSDL

Axis possui uma ferramenta chamada Java2WSDL, o qual gera o arquivo WSDL, para o WS, usar a classe do Java.

Para isso precisamos da interface da calculadora, pois isso fizemos o passo acima, para gerar o Java2WSDL, precisamos do “Calculator.class”, (NÃO DO Calculator.java, ok? )

Para o comando de geração iremos precisar informar alguns parâmetros, são eles:

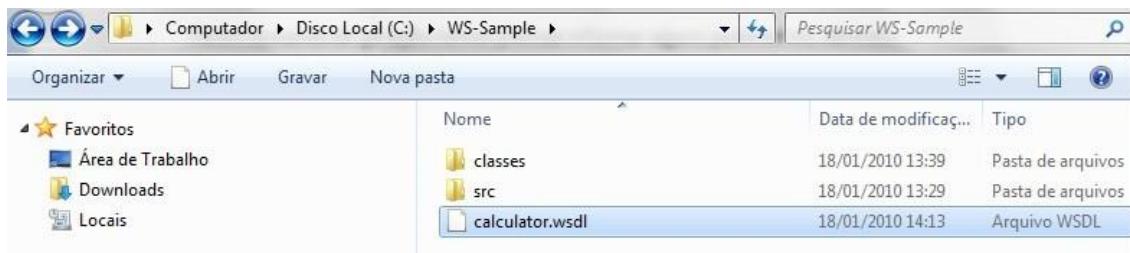
- o – nome para o arquivo WSDL -> (usaremos) calculator.wsdl
- n – o caminho para o namespace -> urn:org.apache.calculator
- l – url para o web service -> http://localhost:8080/axis/services/calculator

```
WS-Sample\classes> java org.apache.axis.wsdl.Java2WSDL -o
..\calculator.wsdl -n urn:org.apache.calculator -l
http://localhost:8080/axis/services/calculator
org.apache.wssample.ws.Calculator
```

Ficando desta maneira:

```
C:\WS-Sample\classes>java org.apache.axis.wsdl.Java2WSDL -o ..\calculator.wsdl -
n urn:org.apache.calculator -l http://localhost:8080/axis/services/calculator or
g.apache.wssample.ws.Calculator
log4j:WARN No appenders could be found for logger (org.apache.axis.i18n.ProjectR
esourceBundle).
log4j:WARN Please initialize the log4j system properly.
C:\WS-Sample\classes>_
```

Este comando irá gerar um arquivo chamado “calculator.wsdl” dentro da pasta principal do projeto:



## WSDL2Java – Gerando as classes para o web service do lado do servidor e do cliente

O Axis possui uma ferramenta chamada WSDL2Java, que gera o lado do servidor e do cliente, usando o arquivo WSDL.

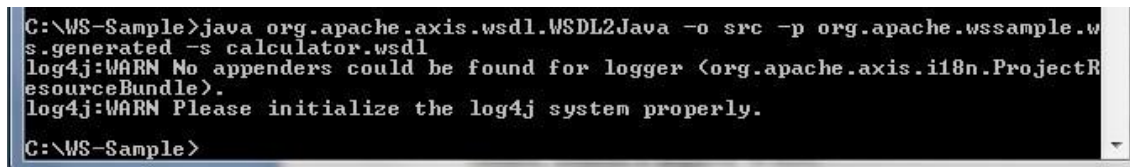
Estas classes são necessárias para implantar no WS e poder acessar o serviço por um cliente Java.

Como nosso passo anterior, precisamos informar alguns parâmetros e eles são:

- o – pasta de saída -> src
- p – local das classes que serão geradas-> org.apache.wssample.ws.generated
- s – gerar o lado do servidor também

```
WS-Sample> java org.apache.axis.wsdl.WSDL2Java -o src -p  
org.apache.wssample.ws.generated -s calculator.wsdl
```

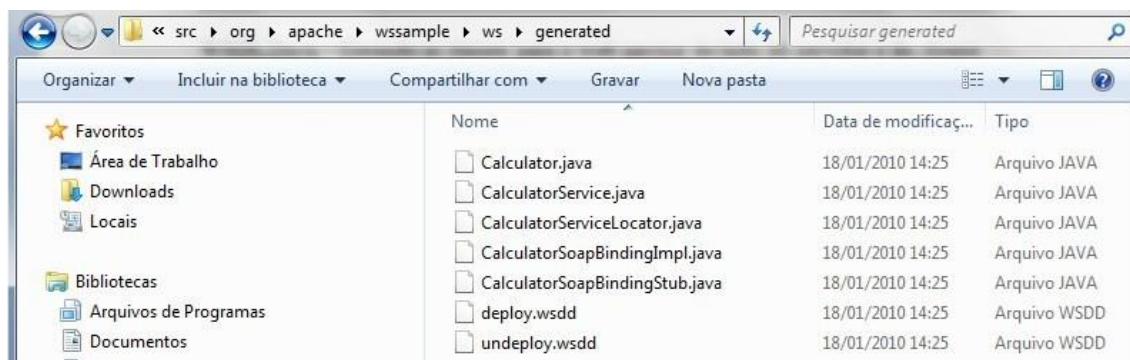
Ficando:



Os arquivos gerados deverão estar em :

C:\WS-Sample\src\org\apache\wssample\ws\generated

Desta forma:



Arquivos gerados:

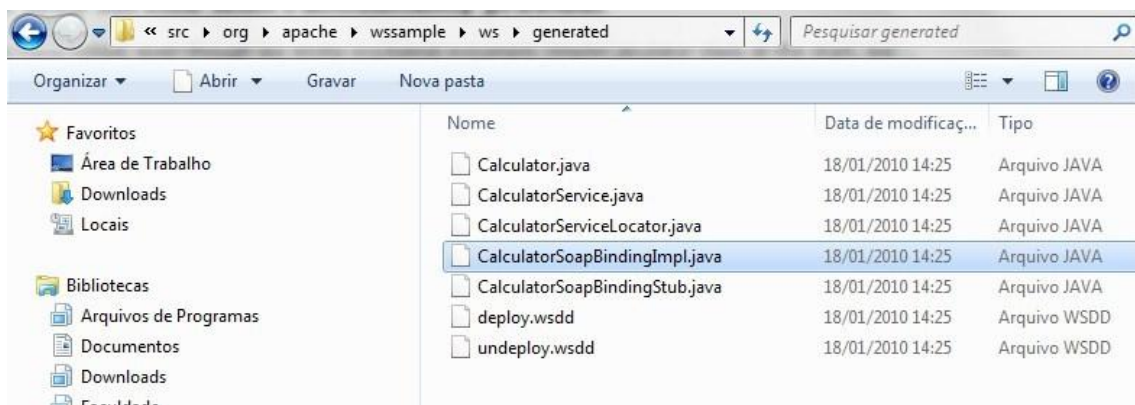
- Calculator.java
- CalculatorService.java
- CalculatorServiceLocator.java
- CalculatorSoapBindingImpl.java
- CalculatorSoapBindingStub.java
- deploy.wsdd
- undeploy.wsdd

---

Ligando o Web Service com a funcionalidade do provedor

Você deve ter notado que mesmo quando escrevemos `org.apache.wssample.SimpleCalculator` class no início do tutorial, nós não usamos ele até o momento, agora iremos ligar ele ao WS.

Há uma classes chamada “CalculatorSoapBindingImpl” dentro de:  
`C:\WS-Sample\src\org\apache\wssample\ws\generated`



Abra-o e edite ele desta forma:

```
package org.apache.wssample.ws.generated;

import org.apache.wssample.SimpleCalculator;

public class CalculatorSoapBindingImpl
    implements org.apache.wssample.ws.generated.Calculator{

    private SimpleCalculator calc = new SimpleCalculator();

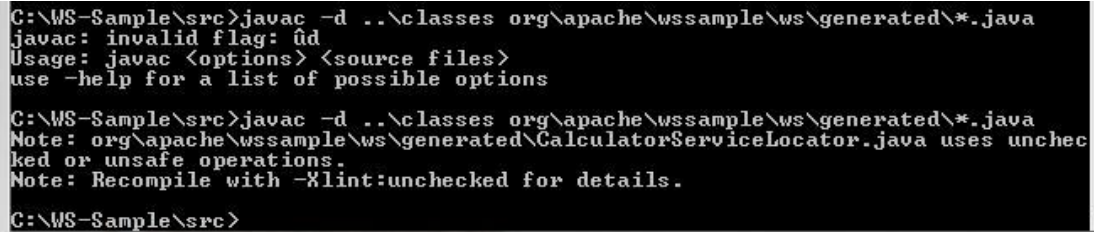
    public int add(int a, int b) throws java.rmi.RemoteException {
        return calc.add(a, b);
    }

    public int subtract(int from, int x) throws
java.rmi.RemoteException {
        return calc.subtract(from, x);
    }
}
```

Compilando os arquivos gerados no passo anterior:

```
WS-Sample\src> javac -d ..\classes org\apache\wssample\ws\generated\*.java
```

Ficando:



```
C:\WS-Sample\src>javac -d ..\classes org\apache\wssample\ws\generated\*.java
javac: invalid flag: ùd
Usage: javac <options> <source files>
use -help for a list of possible options

C:\WS-Sample\src>javac -d ..\classes org\apache\wssample\ws\generated\*.java
Note: org\apache\wssample\ws\generated\CalculatorServiceLocator.java uses unchec
ked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

C:\WS-Sample\src>
```

- Neste caso, se colar o comando poderá não dar certo, então recomendo digitar mesmo no prompt, é o mesmo comando como pode-se ver, mas às vezes pode apresentar problemas,ok?

Note que foi no arquivo “CalculatorSoapBindingImpl.java”, que pudemos editar e usar o pacote “import org.apache.wssample.SimpleCalculator;” que nada mais que é a nossa classe primária, lembra-se?

Esta aqui:

```
package org.apache.wssample;

public class SimpleCalculator {

    public int add(int a, int b) {
        return a + b;
    }
    public int subtract(int a, int b) {
        return a - b;
    }
    public int multiply(int a, int b) {
        return a * b;
    }
}
```

Alteramos o arquivo CalculatorSoapBindingImpl.java, para que use os métodos da classe SimpleCalculator.Simples, não?

---

Empacotando as classes necessárias

Agora vamos criar um arquivo jar, com todas aquelas classes que criamos, para podermos implantar no nosso WS.

Use o comando para criar o jar:

```
WS-Sample\classes>jar cvf ../calculatorServerSide.jar
org\apache\wssample\*.class org\apache\wssample\ws\*.class
org\apache\wssample\ws\generated\*.class
```

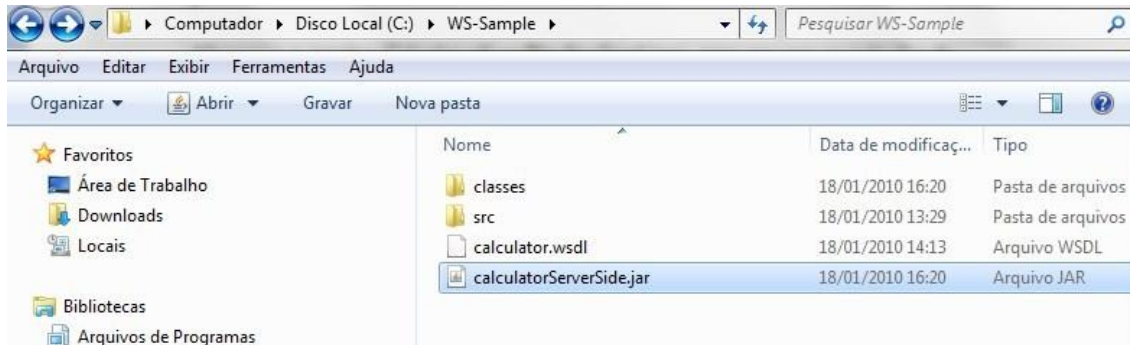


Ficando desta forma:

```
C:\WS-Sample\classes>jar cvf ..\calculatorServerSide.jar org\kamal\wssample\*.class
ass org\kamal\wssample\ws\*.class org\kamal\wssample\ws\generated\*.class
org\kamal\wssample\*.class : no such file or directory
org\kamal\wssample\ws\*.class : no such file or directory
org\kamal\wssample\ws\generated\*.class : no such file or directory
added manifest
C:\WS-Sample\classes>
```

O arquivo .jar gerado deve estar em :

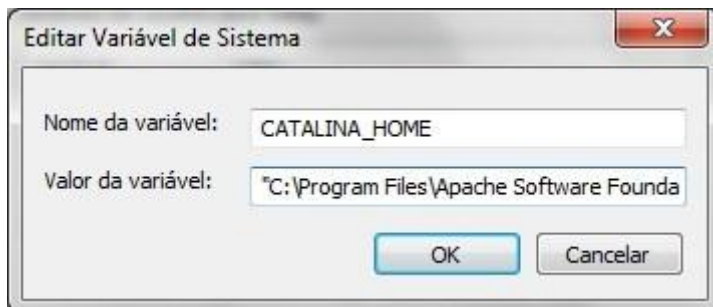
C:\WS-Sample



Agora copie este .jar gerado para a pasta:

%CATALINA\_HOME%\webapps\axis\WEB-INF\lib.

Por exemplo a minha variável de sistema CATALINA\_HOME, esta em:



Valor da variável: "C:\Program Files\Apache Software Foundation\Tomcat 6.0";

- Possui aspas simples pois há espaços no nome das pastas e isso faz com que o sistema não reconheça, então é com aspas sim, ok?

No fim eu irei copiar o .jar para :

C:\Program Files\Apache Software Foundation\Tomcat 6.0\webapps\axis\WEB-INF\lib

Ou pode-se usar este comando também:

```
WS-Sample>copy calculatorServerSide.jar "%CATALINA_HOME%\webapps\axis\WEB-INF\lib"
```



Agora vamos criar um outro arquivo .jar, para ser usado no lado do cliente.

Para a classe do cliente iremos utilizar as classes geradas pelo WSDL2Java.

- Os quais estão em :

C:\WS-Sample\classes\org\apache\wssample\ws\generated

Apenas não iremos compilar o arquivo “CalculatorSoapBindingImpl class”, pois ele já foi compilado no passo acima, ok?

```
WS-Sample\classes>jar cvf ..\calculatorClientSide.jar  
org\apache\wssample\ws\generated\CalculatorSoapBindingStub.class  
org\apache\wssample\ws\generated\CalculatorServiceLocator.class  
org\apache\wssample\ws\generated\CalculatorService.class  
org\apache\wssample\ws\generated\Calculator.class
```

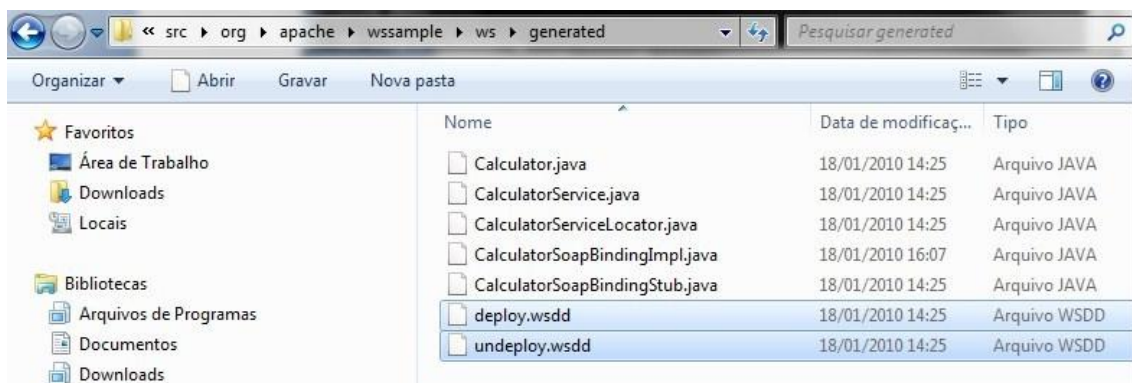
Ficando desta forma:

```
C:\WS-Sample\classes>jar cvf ..\calculatorClientSide.jar org\apache\wssample\ws\generated\CalculatorSoapBindingStub.class org\apache\wssample\ws\generated\CalculatorServiceLocator.class org\apache\wssample\ws\generated\CalculatorService.class org\apache\wssample\ws\generated\Calculator.class  
added manifest  
adding: org/apache/wssample/ws/generated/CalculatorSoapBindingStub.class(in = 5706) (out= 2694)(deflated 52%)  
adding: org/apache/wssample/ws/generated/CalculatorServiceLocator.class(in = 3746) (out= 1749)(deflated 53%)  
adding: org/apache/wssample/ws/generated/CalculatorService.class(in = 446) (out= 241)(deflated 45%)  
adding: org/apache/wssample/ws/generated/Calculator.class(in = 261) (out= 194)(deflated 25%)  
C:\WS-Sample\classes>
```

Registrando o web service com axis

Axis contém uma ferramenta para registrar o web service, se chama “AdminClient”.

Olhe em: C:\WS-Sample\src\org\apache\wssample\ws\generated



Estes dois arquivos “deploy.wsdd” e “undeploy.wsdd”, são descritores de implantação do WS.

**Nota: O Tomcat tem que estar ligado para estes comandos funcionarem**

```
WS-Sample\src>java org.apache.axis.client.AdminClient  
org\apache\wssample\ws\generated\deploy.wsdd
```

```
C:\WS-Sample\src>java org.apache.axis.client.AdminClient org\apache\wssample\ws\  
generated\deploy.wsdd  
log4j:WARN No appenders could be found for logger <org.apache.axis.i18n.ProjectR  
esourceBundle>.  
log4j:WARN Please initialize the log4j system properly.  
Processing file org\apache\wssample\ws\generated\deploy.wsdd  
<Admin>Done processing</Admin>  
  
C:\WS-Sample\src>_
```

Este comando ira implantar o web services no axis. Agora reinicie o tomcat(desligue e ligue). Para verificar se a implantação foi feita corretamente, acesse:

<http://localhost:8080/axis/services/calculator?wsdl>

- Muda as portas 8080 para a porta que você configurou na sua máquina no Tomcat

Ao acessar este endereço você verá o arquivo wsdl compeltto, a definição completa do serviço que implantamos.

Você verá algo assim:

O documento XML não está associado a estilos. A estrutura do documento é representada abaixo.

```
<?xml version='1.0' encoding='utf-8'>  
<wsdl:definitions targetNamespace='urn:org.apache.calculator'>  
  <!--  
    WSDL created by Apache Axis version: 1.4  
    Built on Apr 22, 2006 (06:55:48 PDT)  
  -->  
  <wsdl:message name='addRequest'>  
    <wsdl:part name='in0' type='xsd:int'/>  
    <wsdl:part name='in1' type='xsd:int'/>  
  </wsdl:message>  
  <wsdl:message name='subtractResponse'>  
    <wsdl:part name='subtractReturn' type='xsd:int'/>  
  </wsdl:message>  
  <wsdl:message name='subtractRequest'>  
    <wsdl:part name='in0' type='xsd:int'/>  
    <wsdl:part name='in1' type='xsd:int'/>  
  </wsdl:message>  
  <wsdl:message name='addResponse'>  
    <wsdl:part name='addReturn' type='xsd:int'/>  
  </wsdl:message>  
  <wsdl:portType name='Calculator'>  
    <wsdl:operation name='add' parameterOrder='in0 in1'>  
      <wsdl:input message='impladdRequest' name='addRequest'/>  
      <wsdl:output message='impladdResponse' name='addResponse'/>  
    </wsdl:operation>  
    <wsdl:operation name='subtract' parameterOrder='in0 in1'>  
      <wsdl:input message='implsubtractRequest' name='subtractRequest'/>  
      <wsdl:output message='implsubtractResponse' name='subtractResponse'/>  
    </wsdl:operation>  
  </wsdl:portType>  
  <wsdl:binding name='calculatorSoapBinding' type='impl:Calculator'>  
    <wsdl:soap:binding style='rpc' transport='http://schemas.xmlsoap.org/soap/http'/>  
    <wsdl:operation name='add'>  
      <wsdl:soap:operation soapAction=''>  
        <wsdl:input name='addRequest'>  
          <wsdl:soap:body encodingStyle='http://schemas.xmlsoap.org/soap/encoding/' namespace='urn:org.apache.calculator' use='encoded'/>  
        <wsdl:input>
```

(Tem muito mais código para baixar, tah?rs)

Agora tudo no Web Service(lado do servidor) está completado e nosso web service esta implantado com sucesso!

## Web Service Cliente

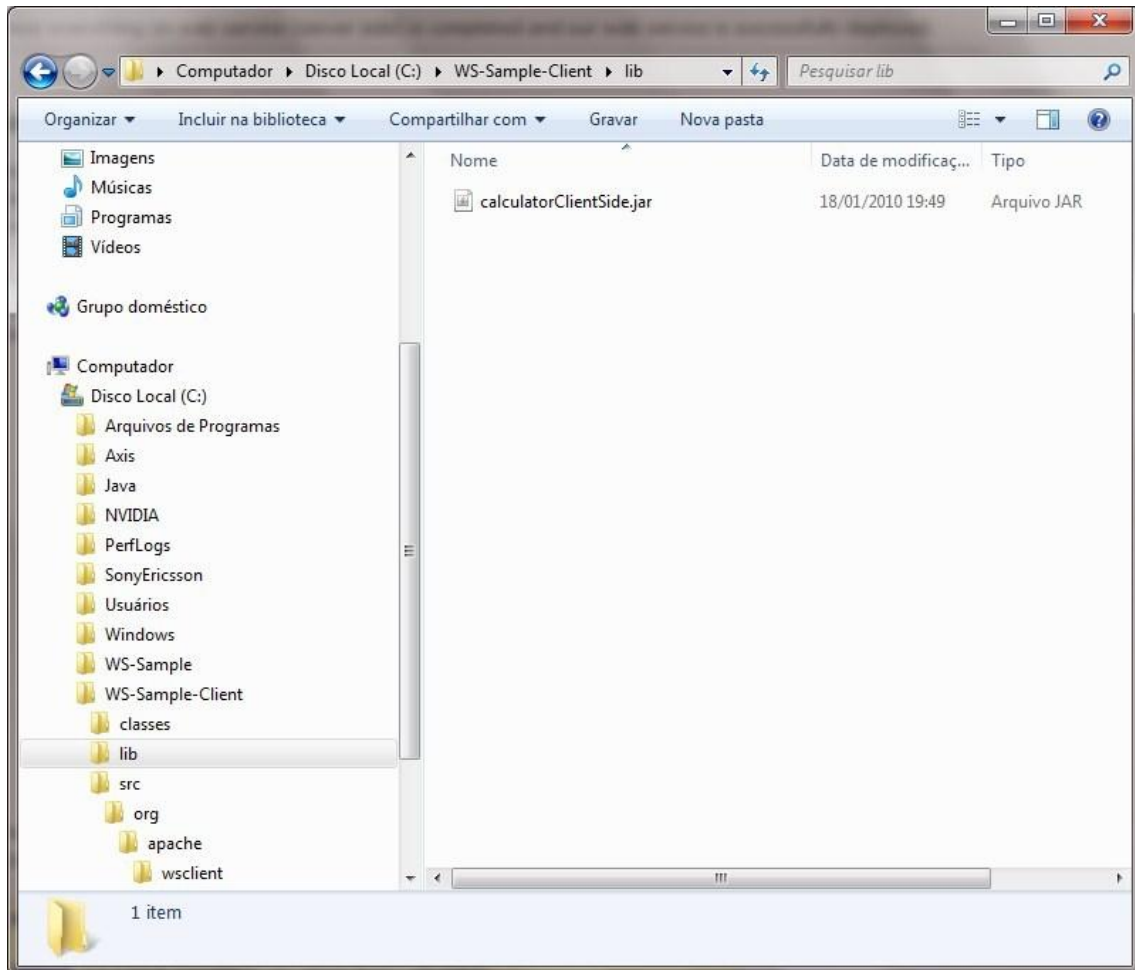
Agora iremos escrever um código para o cliente acessar este web Service e usar os serviços que são servidos(digamos assim,rs).

Lembra-se do .jar do cliente que fizemos?

Então foi exatamente para isso, vamos lá então:

Para o lado do cliente iremos criar uma outra estrutura, ok?

Para ficar mais organizado o nosso projeto, ficando desta maneira:



Cole o “calculatorClientSide.jar” em “WS-Sample-Client\lib”

Nós vamos criar o cliente exatamente como está no código abaixo.

Desde que nosso web service possui dois métodos, lembra-se?

O add() e o subtract().

A classe cliente irá usar estes serviços, irá chamar por estes métodos.

```
package org.apache.wsclient;

import org.apache.wssample.ws.generated.Calculator;
import org.apache.wssample.ws.generated.CalculatorService;
import org.apache.wssample.ws.generated.CalculatorServiceLocator;

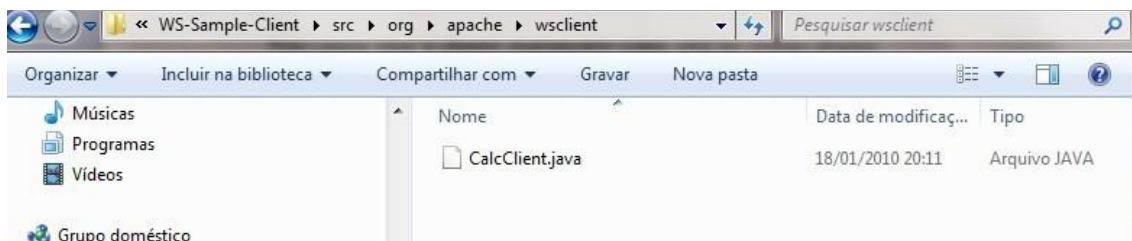
public class CalcClient {

    public static void main(String[] args) throws Exception {
        CalculatorService service = new CalculatorServiceLocator();
        Calculator calc = service.getcalculator();

        System.out.println("15 + 6 = " + calc.add(15, 6));
        System.out.println("15 - 6 = " + calc.subtract(15, 6));
    }
}
```

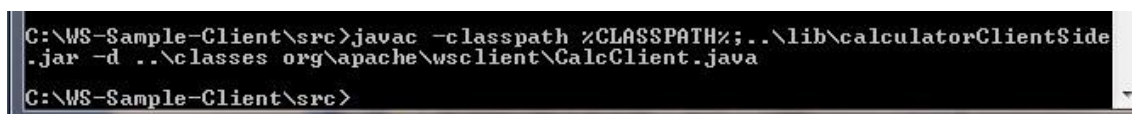
A classe acima não usou uma única classe que escrevemos para a calculadora em sim, ou seja, apenas algumas classes que a ferramenta do axis a WSDL2Java gerou. Nós não precisamos das classes do lado do servidor, apenas estamos solicitando serviços daquelas classes do servidor.

Este código deve ficar em : C:\WS-Sample-Client\src\org\apache\wsclient  
Assim:



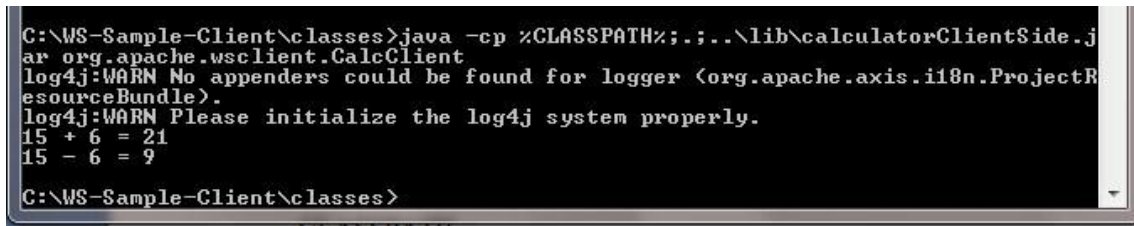
Usamos este comando para compilar:

```
WS-Sample-Client\src> javac -classpath %CLASSPATH%;..\lib\calculatorClientSide.jar -d  
..\classes org\apache\wsclient\CalcClient.java
```



E por fim podemos rodar o nosso serviço (ufa,rs):

WS-Sample-Client\classes>java -cp %CLASSPATH%;..\lib\calculatorClientSide.jar  
org.apache.wsclient.CalcClient



```
C:\WS-Sample-Client\classes>java -cp %CLASSPATH%;..\lib\calculatorClientSide.jar  
org.apache.wsclient.CalcClient  
log4j:WARN No appenders could be found for logger <org.apache.axis.i18n.ProjectResourceBundle>.  
log4j:WARN Please initialize the log4j system properly.  
15 + 6 = 21  
15 - 6 = 9  
C:\WS-Sample-Client\classes>
```

Você deverá ver:

```
15 + 6 = 21  
15 - 6 = 9
```

Nosso cliente do web service, CalcClient acessou o web service e recebeu os resultados das operações feitas pela classe SimpleCalculator, aonde SimpleCalculator esta rodando no lado do servidor, interessante não é? Demais! x)

É isso,rs.

Espero ter ajudado você a entender melhor um pouco sobre web service e a usá-lo com o Tomcat e o Axis, ok?

Qualquer dúvida mande um email para : [mascaranegrayoko@hotmail.com](mailto:mascaranegrayoko@hotmail.com)

Fonte : <http://lkamal.blogspot.com/2008/07/web-service-axis-tutorial-client-server.html#comment-form>

---

Apenas caso ajuda, minhas variáveis de ambiente estão configuradas desta maneira:

CATALINA\_HOME

C:\Program Files\Apache Software Foundation\Tomcat 6.0

CLASSPATH

C:\Axis\lib\axis.jar;C:\Axis\lib\commons-discovery-0.2.jar;C:\Axis\lib\commons-logging-1.0.4.jar;C:\Axis\lib\jaxrpc.jar;C:\Axis\lib\log4j-1.2.8.jar;C:\Axis\lib\saaj.jar;C:\Axis\lib\wsdl4j-1.5.1.jar;%JAVA\_HOME%;.

JAVA\_HOME

C:\Java\jdk1.6.0\_18

PATH

%JAVA\_HOME%\bin;