

FIAP GRADUAÇÃO

# DIGITAL BUSINESS ENABLEMENT

Prof. Alexandre C. de Jesus

#03 – WEB SERVICES SOAP - IMPLEMENTAÇÃO

# PERCURSO

---



Aula Inaugural

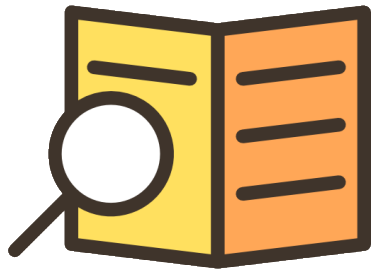


Web Services SOAP



## #03 - AGENDA

---



- Web Service Provider
  - Configuração do eclipse para Axis 2
  - Criação do projeto
  - Gerar um serviço web
- Web Service Requester
  - Interface Texto
    - Criação do projeto
    - Geração das classes de acesso ao web service
    - Exemplo de acesso ao web service

- **JAX WS** – JAVA API FOR XML WEB SERVICES
- API do **JAVA EE** para criação de web services;
- **Implementações:**
  - Apache Axis (JAX-RPC)
  - **Apache Axis 2 (JAX-WS, JAX-RS)**
  - Apache CXF (JAX-WS, JAX-RS)
  - Java SE Runtime (a partir do Java 6) (JAX-WS)



# WEB SERVICE PROVIDER

# CONFIGURAÇÃO DO ECLIPSE

- Faça o download do Apache Axis 2 através do link:

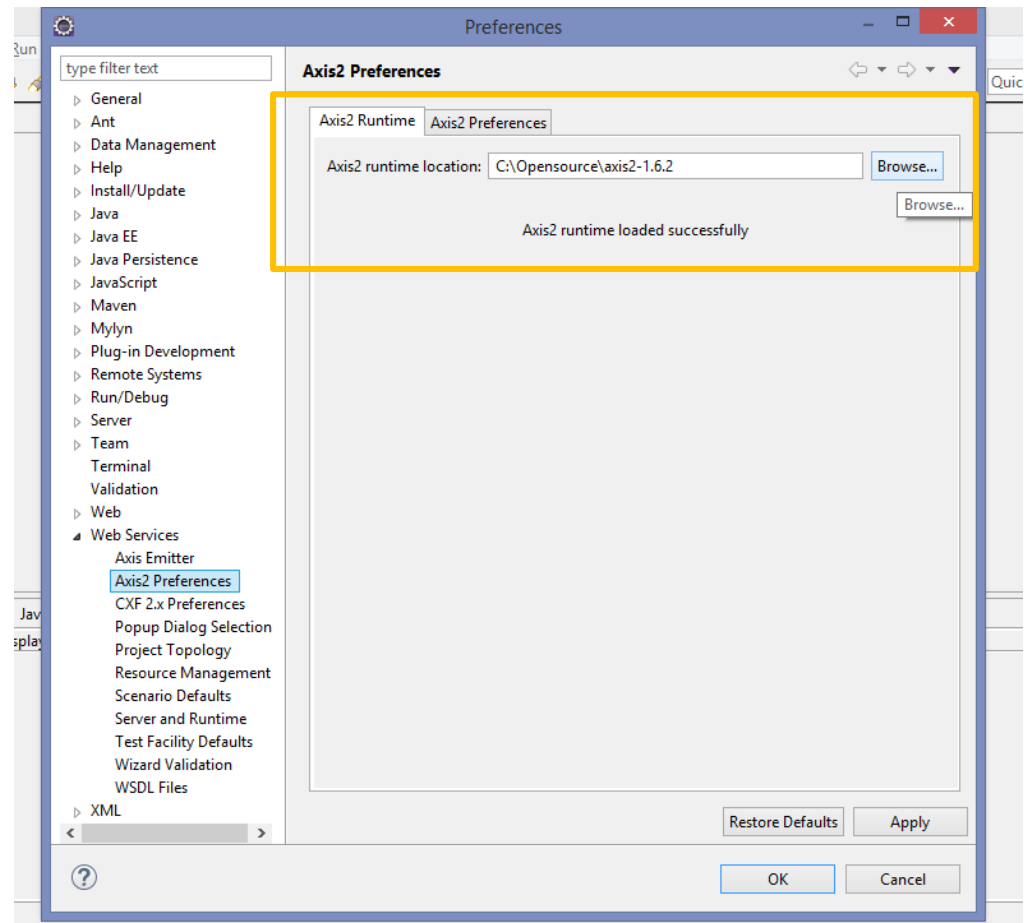
LINK

Binary distribution

[axis2-1.7.9-bin.zip](#)

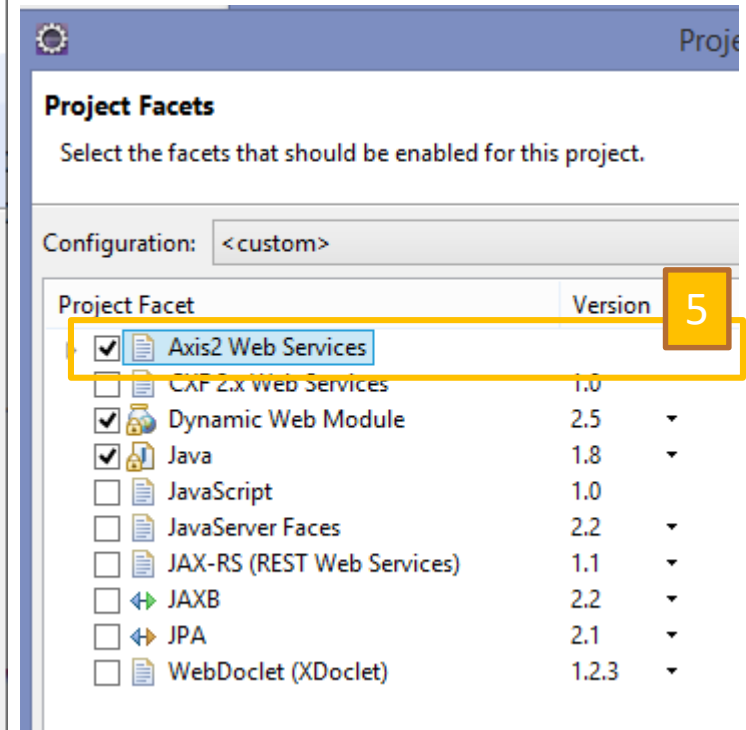
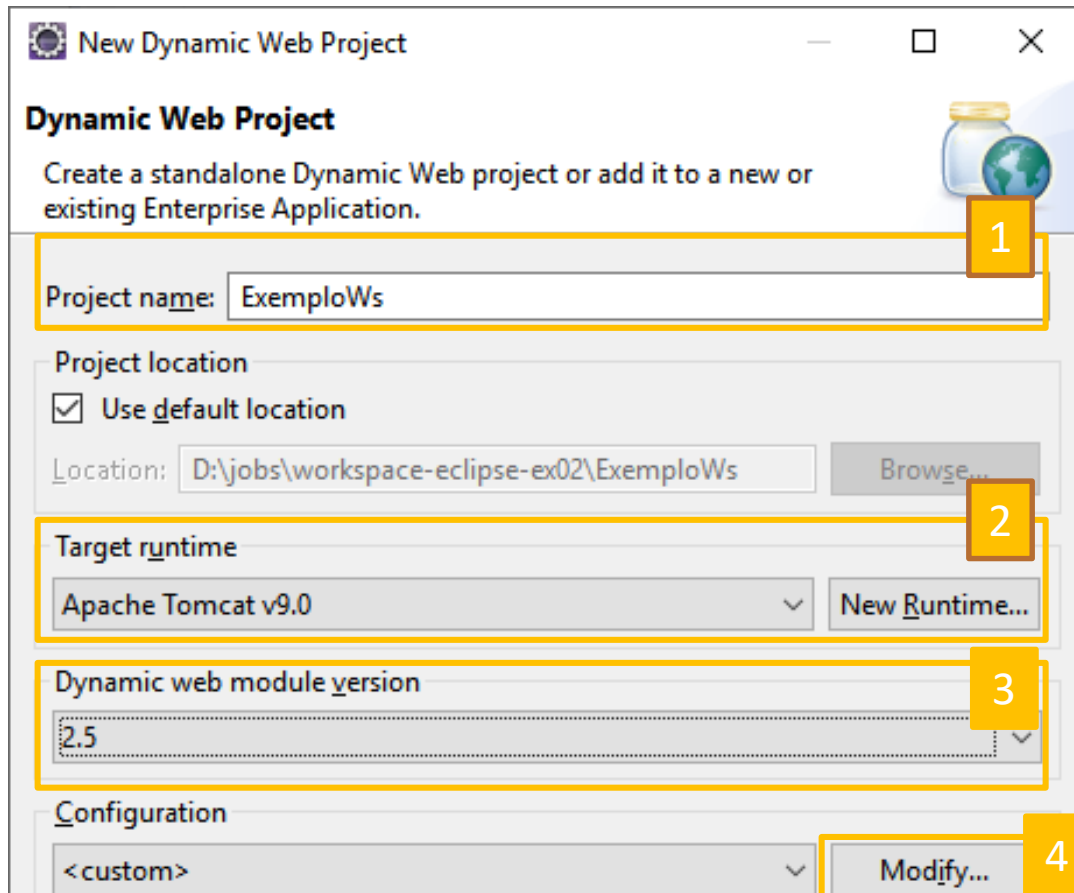
<http://axis.apache.org/axis2/java/core/download.cgi> escolha a opção Binary.

No Eclipse em **Windows** >  
**Preferences**, procure por **Axis2**  
**Preferences** e configure a  
localização do apache axis 2.



# CRIANDO O PROJETO

- 1 - Crie um **Dynamic Web Project**;
- 2 - Configure o Target Runtime: **Apache Tomcat 9**;
- 3 - Mude o **Dynamic web module version: 2.5**;
- 4 e 5 - Em configuration clique em : **Modify** > Marque: **Axis 2 Web Services**;



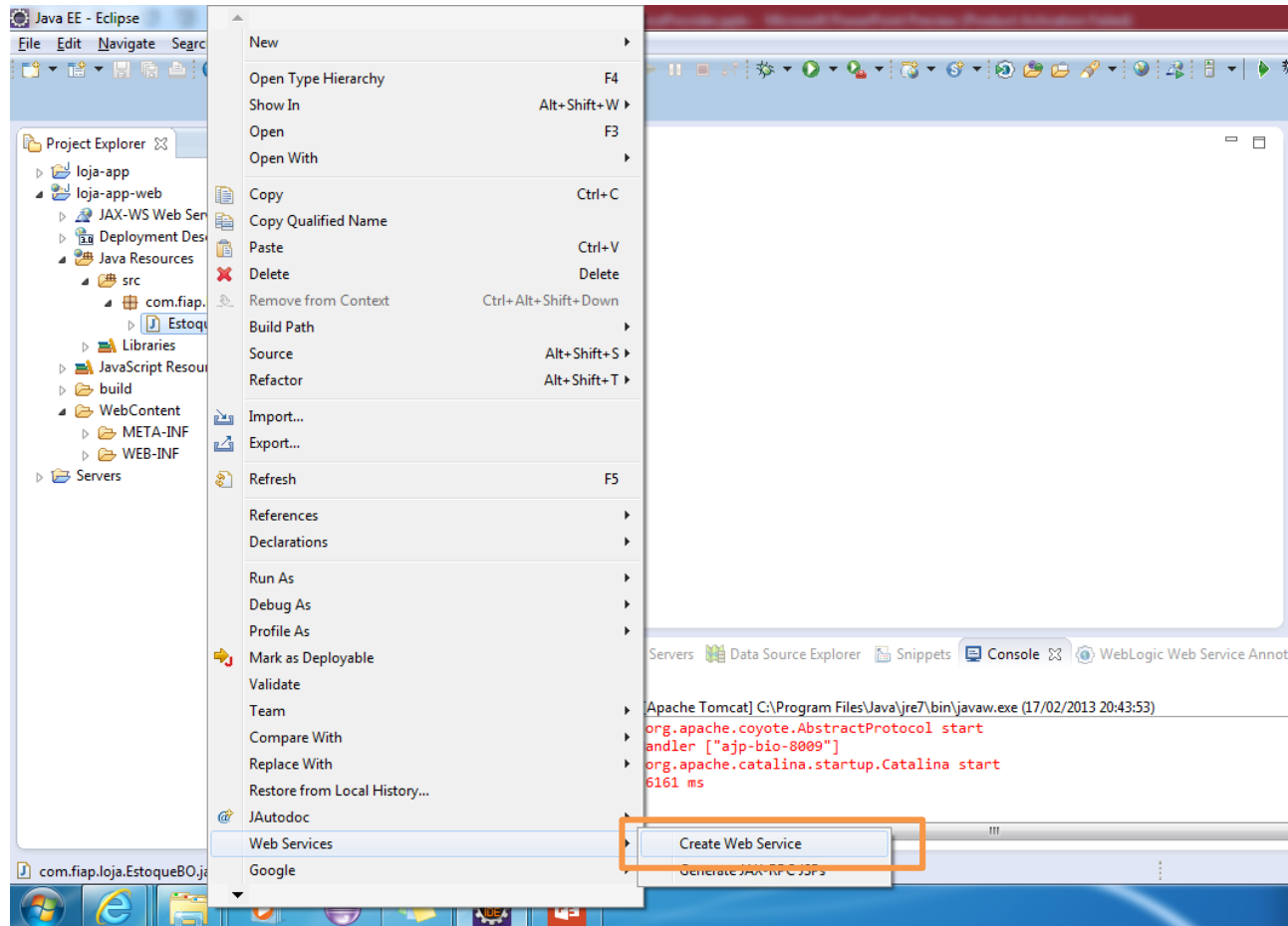


- Crie a classe de Serviço chamada EstoqueBasico, com o seguinte método.

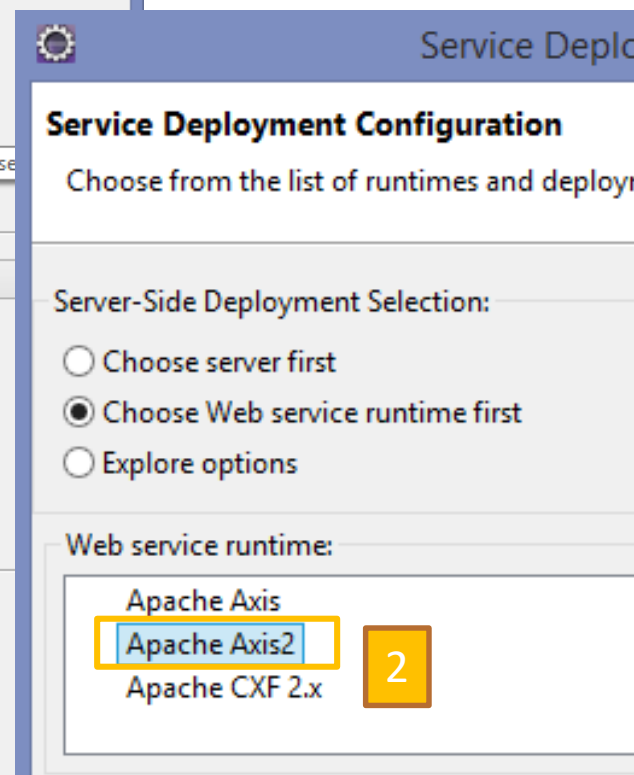
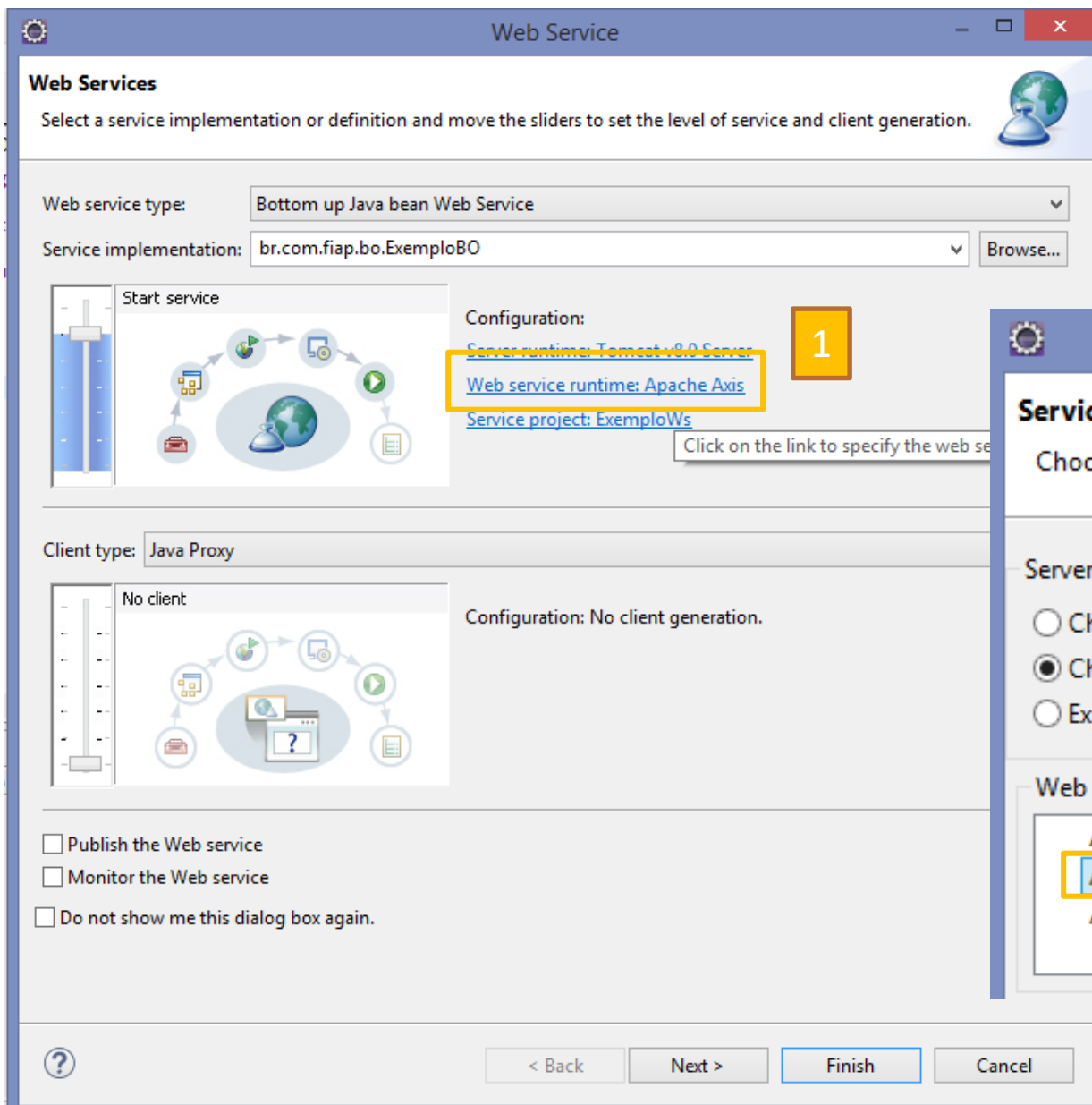
```
package br.com.fiap.estoque;  
  
public class EstoqueBasico {  
  
    public int soma(int nr1,int nr2) {  
        return (nr1+nr2);  
    }  
  
}
```

# GERANDO UM SERVIÇO WS

- Clique com o botão direito na classe de serviço e escolha **Web Services> Create Web Service**;

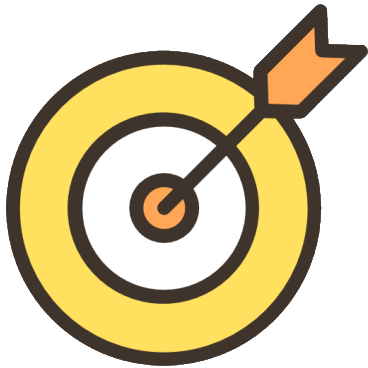


# GERANDO UM SERVIÇO WS



# PRÁTICA!

---

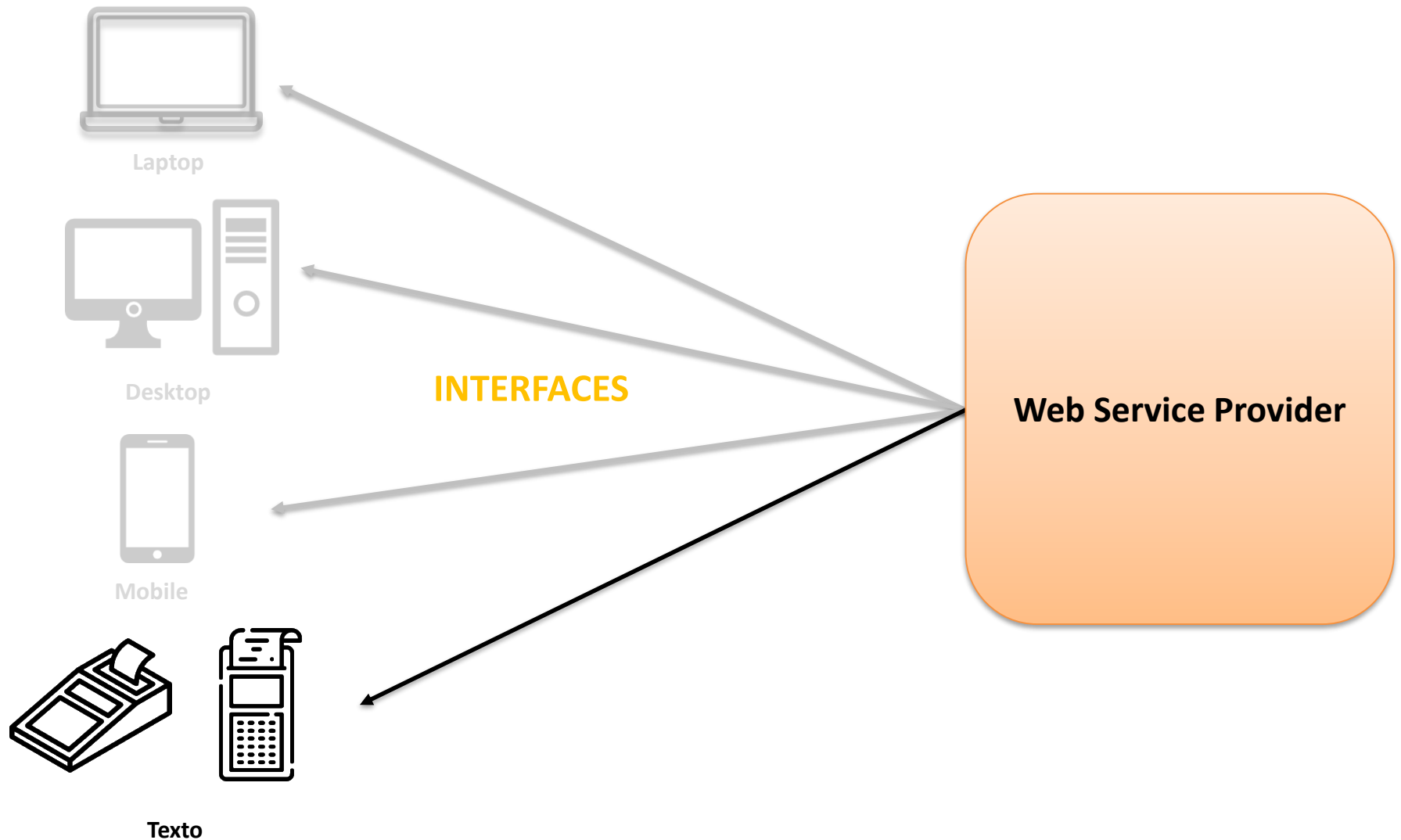


1. Criar um projeto **Java Web Application** com o **Axis 2**;
2. Desenvolver um **Web Services Provider** para obter informações a respeito de **produtos** de uma loja;
3. Gerar um **AxisFault** caso o produto não esteja cadastrado (`throw new AxisFault("Produto Não Cadastrado")`);
4. Teste o serviço com o **SOAP UI**;



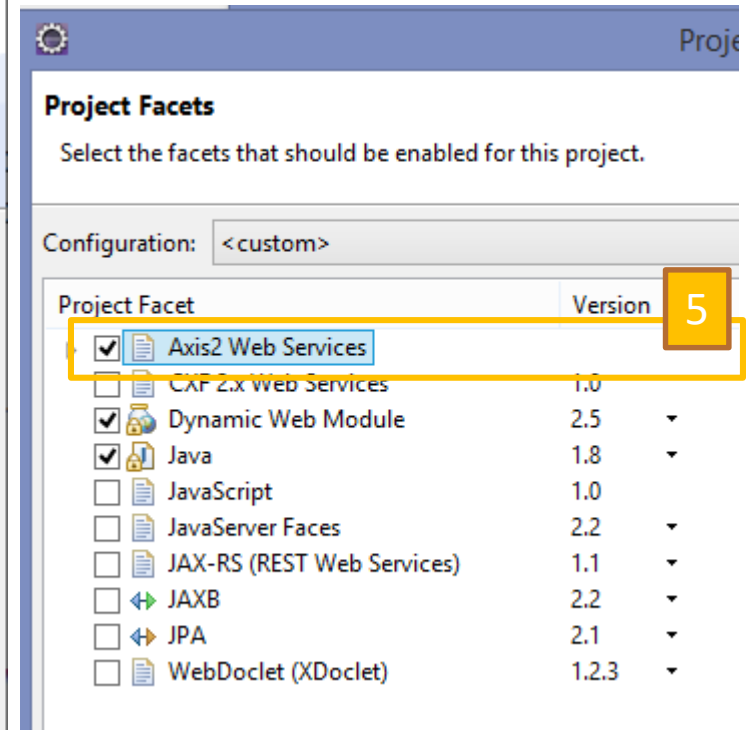
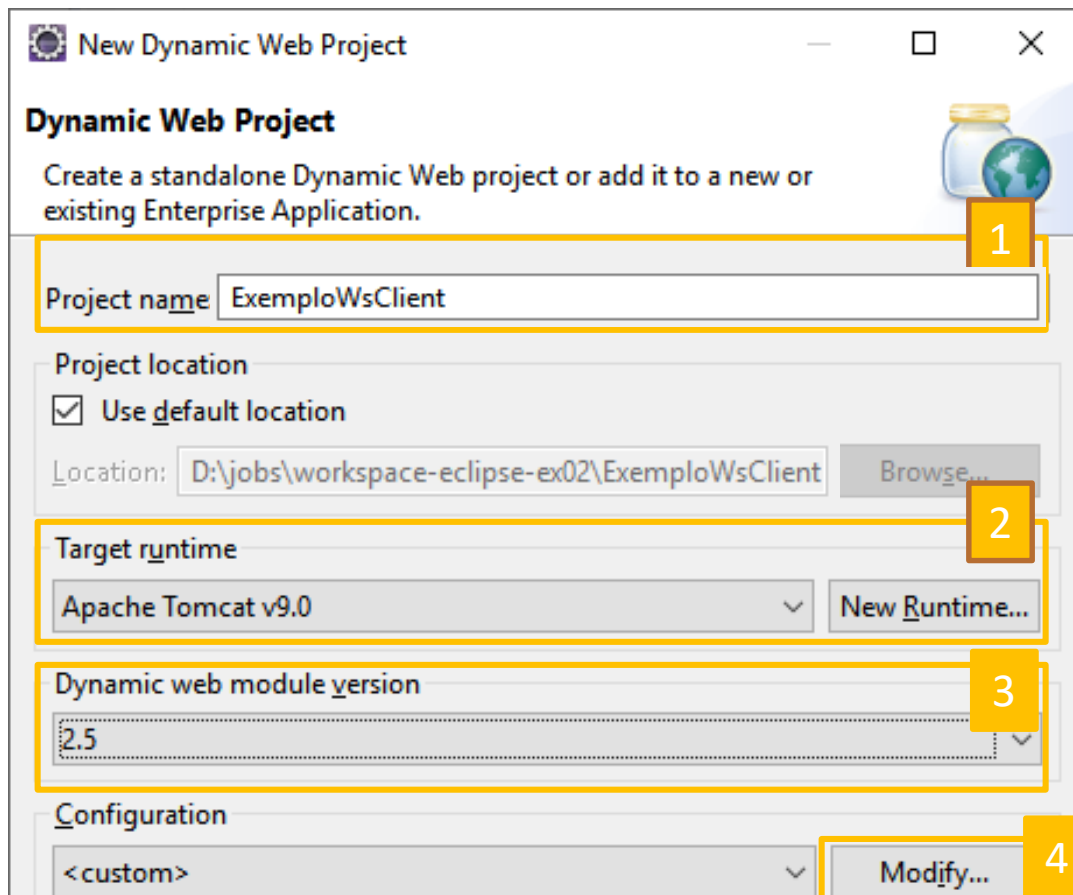


# WEB SERVICE REQUESTER



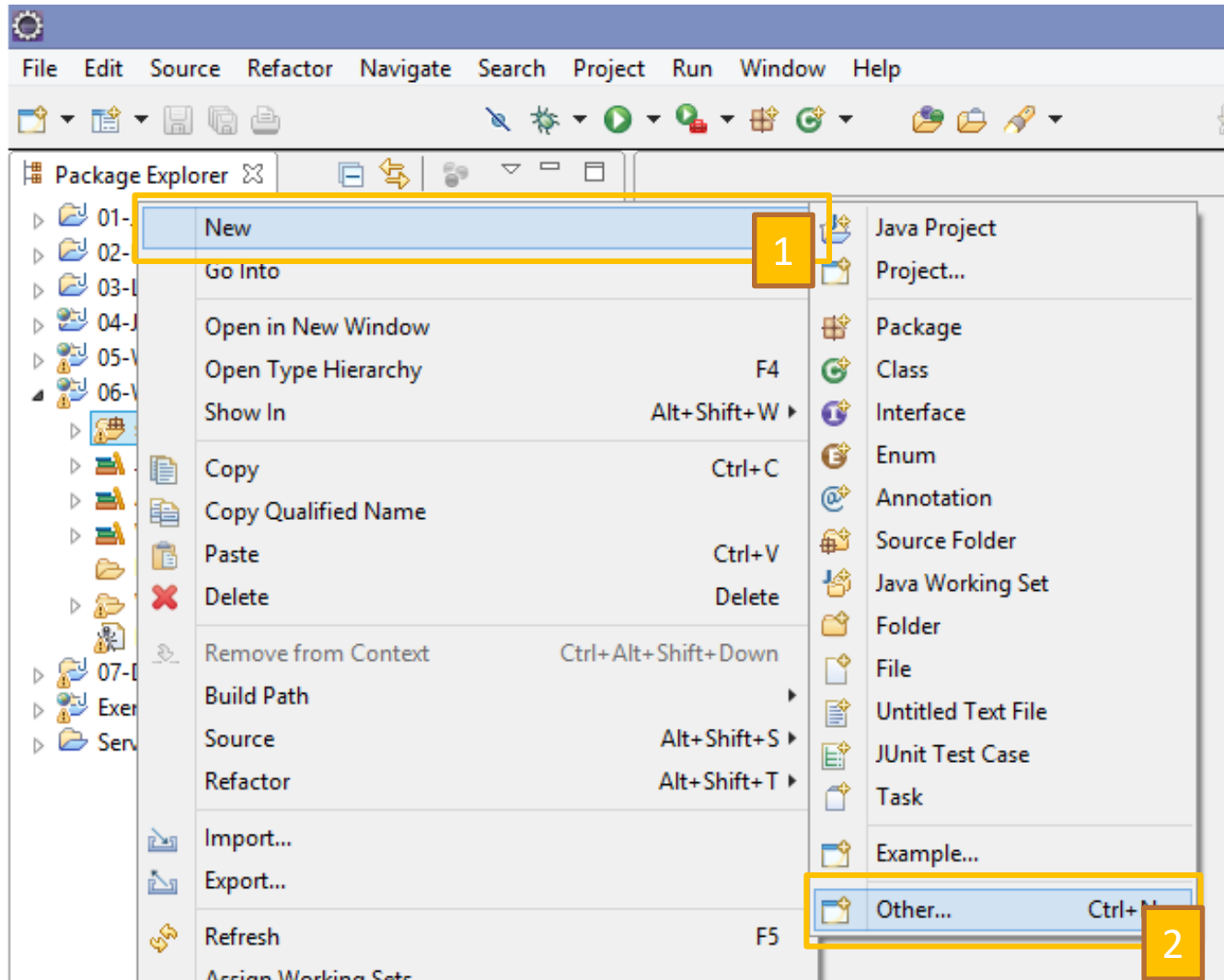
# PROJETO – WS REQUESTER

- Crie um **Dynamic Web Project**;
- Configure o Target Runtime: **Apache Tomcat 9**;
- Mude o **Dynamic web module version: 2.5**;
- Em configuration clique em : **Modify** > Marque: **Axis 2 Web Services**;



# CRIANDO O WEB SERVICE CLIENT

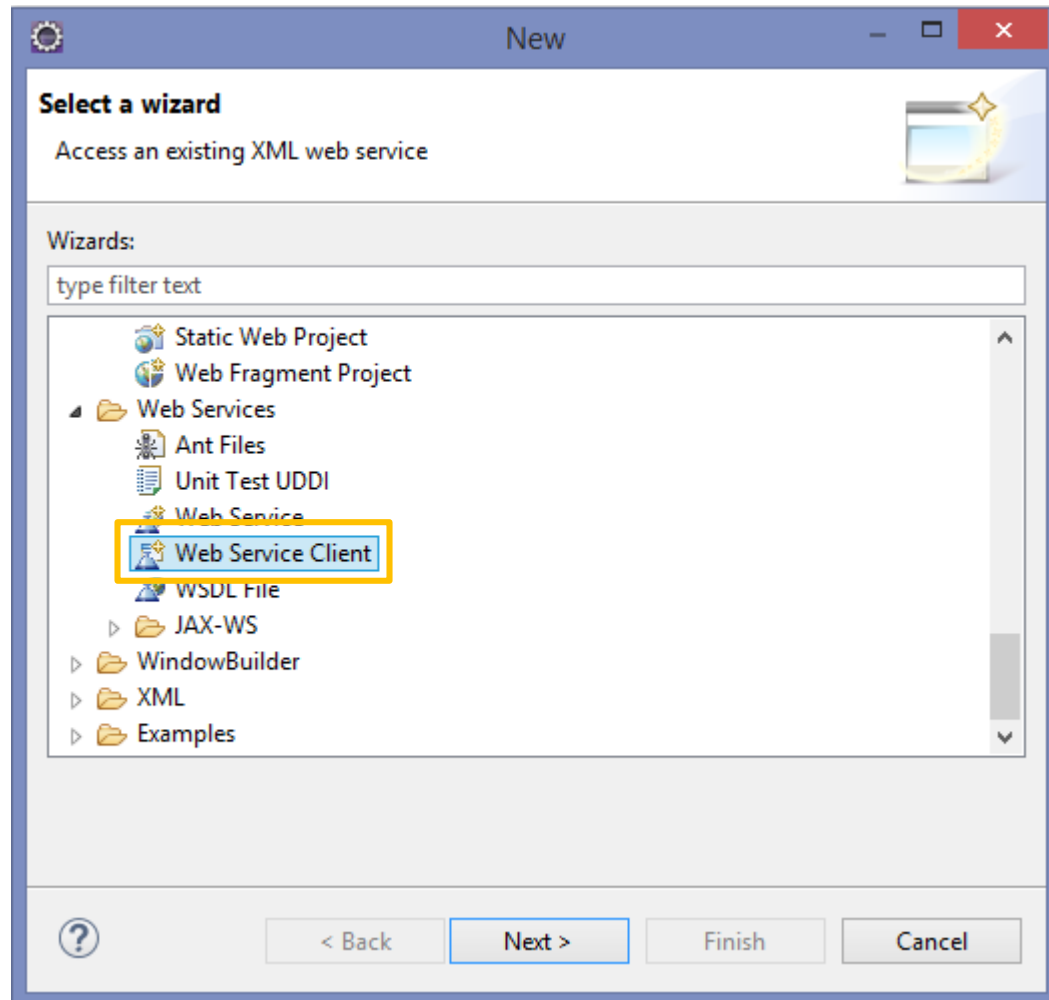
- Clique com o botão direito do mouse no projeto e escolha **New > Other...**





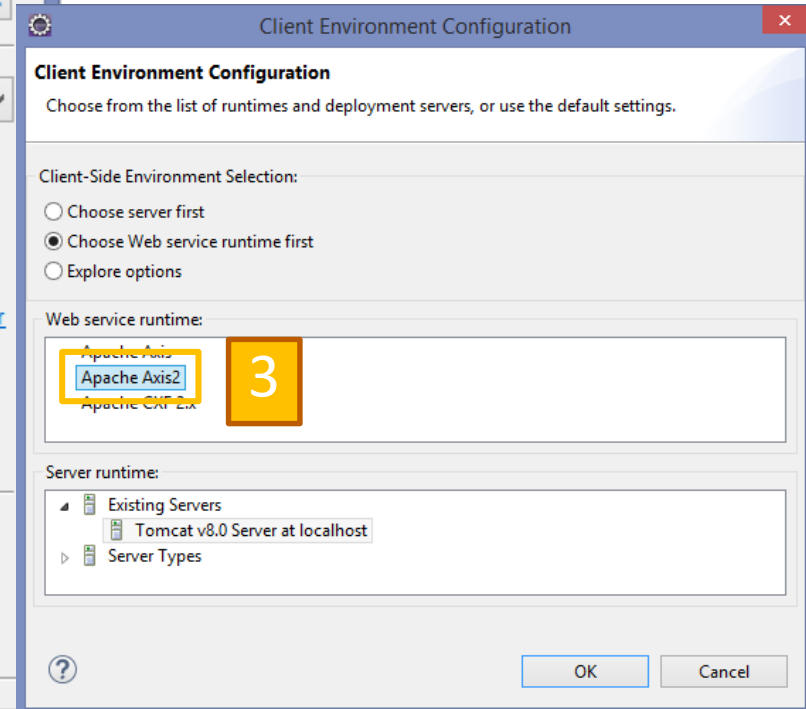
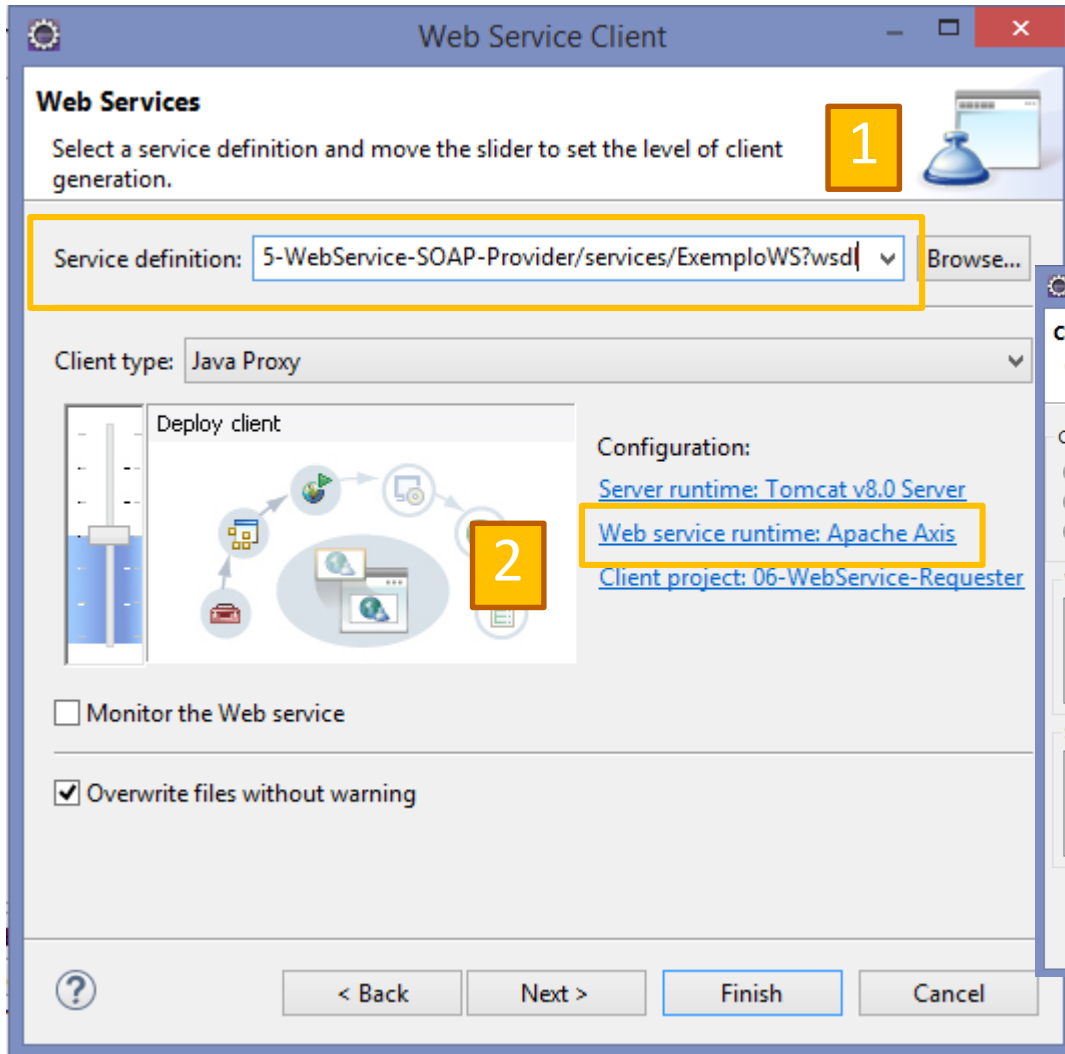
# CRIANDO O WEB SERVICE CLIENT

- Escolha a opção **Web Service Client**



# CRIANDO O WEB SERVICE CLIENT

- Coloque o **endereço do WSDL** e ajuste o **Apache Axis 2**;



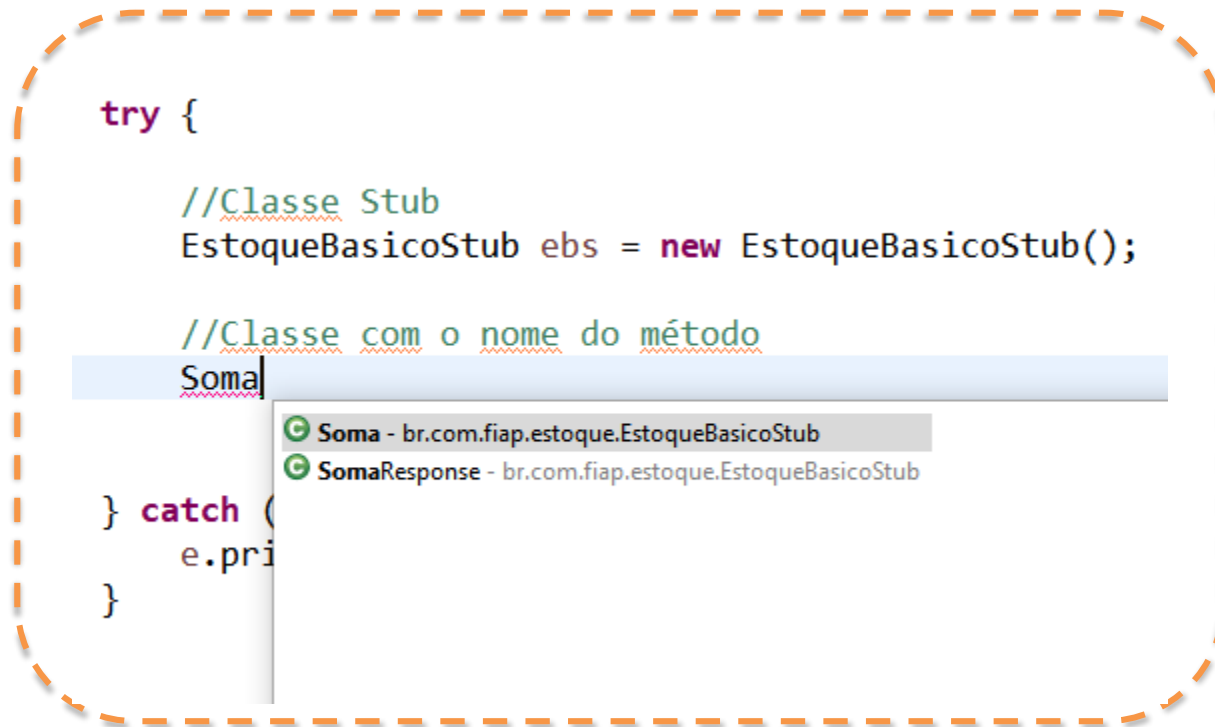
- Crie instancias das classes das seguintes formas:

**Stub** : Utilizada para acessar o método e receber o parâmetro(s) da Classe com o nome do método.

**Classe com nome do método** : Utilizada para setar o parâmetro principal que a Stub vai receber.

**Classe com nome do método e Response** : Utilizada para receber o retorno de Stub e gerar o objeto resultante.

```
try {  
    //Classe Stub  
    EstoqueBasicoStub ebs = new EstoqueBasicoStub();  
  
    //Classe com o nome do método  
    Soma  
} catch (e.pri  
}
```

A screenshot of a code editor with a dashed orange border. The code is in Java and shows a try-catch block. Inside the try block, there is a comment '//Classe Stub' followed by 'EstoqueBasicoStub ebs = new EstoqueBasicoStub();'. Then there is a comment '//Classe com o nome do método' followed by 'Soma'. The word 'Soma' is highlighted with a blue bar, and a dropdown menu is open below it, showing two suggestions: 'Soma - br.com.fiap.estoque.EstoqueBasicoStub' and 'SomaResponse - br.com.fiap.estoque.EstoqueBasicoStub'. The catch block is partially visible, showing 'catch (e.pri' and a closing brace '}'.

```
package br.com.fiap.estoque;

import java.rmi.RemoteException;

import org.apache.axis2.AxisFault;

import br.com.fiap.estoque.EstoqueBasicoStub.Soma;
import br.com.fiap.estoque.EstoqueBasicoStub.SomaResponse;

public class ConsultaBasica {

    public static void main(String[] args) {

        try {

            //Classe Stub
            EstoqueBasicoStub ebs = new EstoqueBasicoStub();

            //Classe com o nome do método
            Soma sm = new Soma();
            sm.setNr1(100);
            sm.setNr2(23);

            //Stub recebendo o parâmetro e já retornando para a response.
            SomaResponse smr = ebs.soma(sm);

            //Response retornando o Obj resultante.
            System.out.println("O Resultado da Operação é :" + smr.get_return());

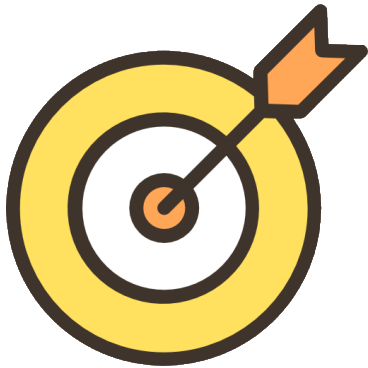
        } catch (AxisFault e) {
            e.printStackTrace();
        } catch (RemoteException e) {
            e.printStackTrace();
        }

    }

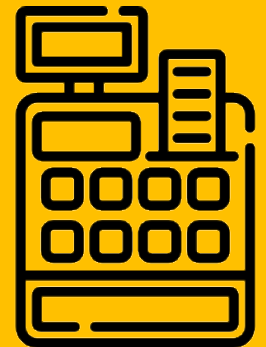
}
```

## PRÁTICA! INTERFACE TEXTO

---



1. Criar um projeto para o **Web Services Requester**;
2. Criar as **classes de acesso** ao Web Service do exercício anterior;
3. Implemente uma classe com o método **main** para que o **usuário possa informar o código** de um produto e o programa **acesse o web service** para obter a resposta;



## VOCÊ APRENDEU...

---

- Implementar um **web service provider** com Axis 2;
- Criar um **web service requester** com **interface texto**;



**Copyright © 2013 – 2020**  
**Prof. Alexandre C. de Jesus**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*“Nosso maior medo não deve ser o fracasso, mas  
ser bem-sucedidos em algo que não importa”*