

FIAP GRADUAÇÃO

DIGITAL BUSINESS ENABLEMENT

Prof. THIAGO T. I. YAMAMOTO

#05 – WEB SERVICES SOAP



thiagoyama



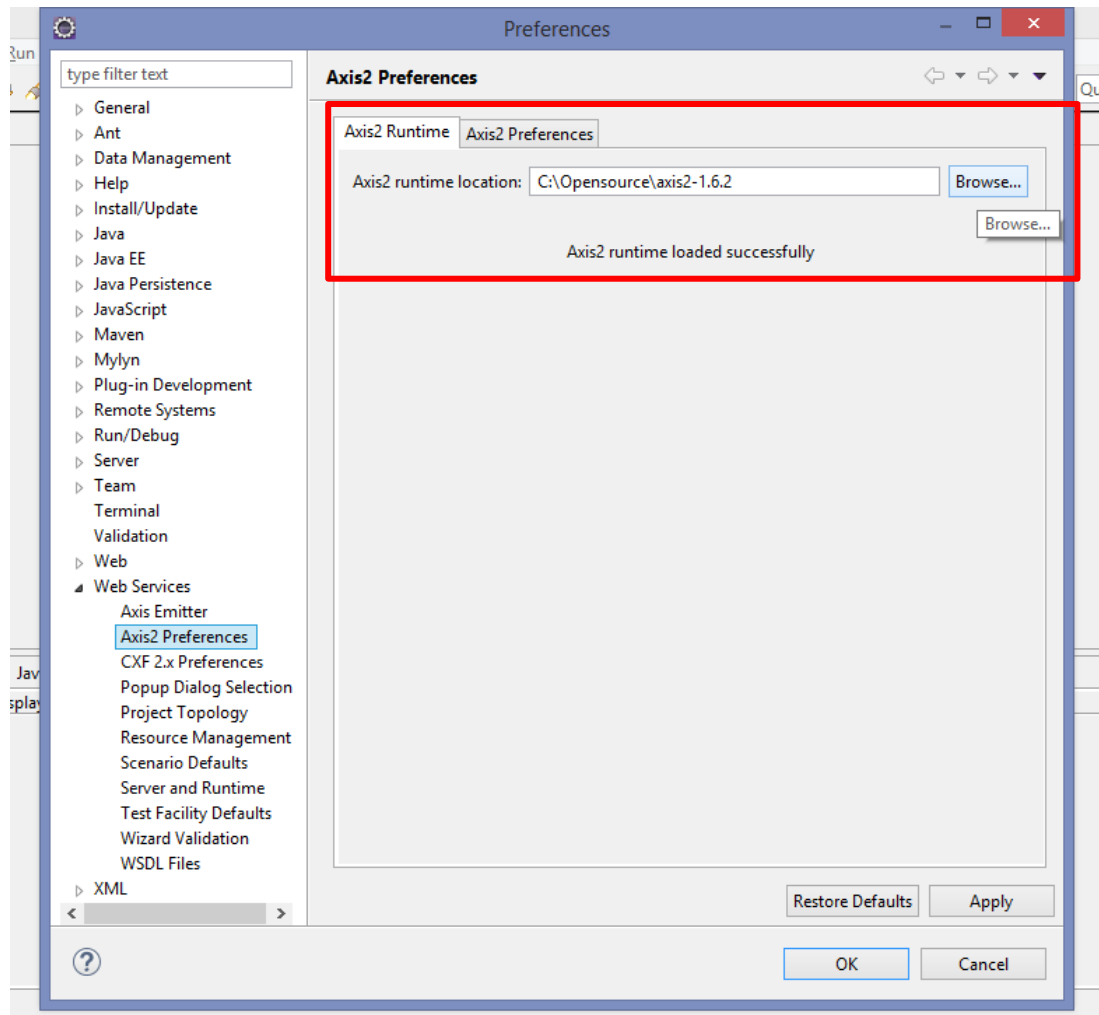
thiagoyama@gmail.com

- Desenvolvimento de Web Service Provider
- Desenvolvimento de Web Service Requester
- Integração Java Web Services com Java GUI (Graphical User Interface) – Desktop utilizando SWT (Standard Widget Toolkit)

WEB SERVICE PROVIDER

CONFIGURANDO O SERVIDOR

- Faça o download do Apache Axis 2 através do link: <http://axis.apache.org/axis2/java/core/download.cgi>
- Em windows -> preferences, procure por Axis2 Preferences e configure a localização do apache axis 2:



CONFIGURANDO O SERVIDOR

- Crie um Dynamic Web Project;
- Configure o Target Runtime: Apache Tomcat 9;
- Dynamic web module version: 2.5;
- Configuration: Modify -> Marque: Axis 2 Web Services

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

Project Facets

Select the facets that should be enabled for this project.

Configuration:

| Project Facet | Version |
|--|---------|
| <input checked="" type="checkbox"/> Axis2 Web Services | |
| <input type="checkbox"/> CXF 2.x Web Services | 1.0 |
| <input checked="" type="checkbox"/> Dynamic Web Module | 2.5 |
| <input checked="" type="checkbox"/> Java | 1.8 |
| <input type="checkbox"/> JavaScript | 1.0 |
| <input type="checkbox"/> JavaServer Faces | 2.2 |
| <input type="checkbox"/> JAX-RS (REST Web Services) | 1.1 |
| <input type="checkbox"/> JAXB | 2.2 |
| <input type="checkbox"/> JPA | 2.1 |
| <input type="checkbox"/> WebDoclet (XDoclet) | 1.2.3 |

GERANDO UM SERVIÇO WS

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays a project named 'loja-app' with a sub-project 'loja-app-web'. Inside 'loja-app-web', there is a 'src' folder containing a package 'com.fiap.loja' with a class 'EstoqueBO.java'. The 'Web Services' menu is open, showing options like 'New', 'Open Type Hierarchy', 'Copy', 'Paste', 'Delete', 'Build Path', 'Source', 'Refactor', 'Import...', 'Export...', 'Refresh', 'References', 'Declarations', 'Run As', 'Debug As', 'Profile As', 'Mark as Deployable', 'Validate', 'Team', 'Compare With', 'Replace With', 'Restore from Local History...', 'JAutodoc', 'Web Services', and 'Google'. The 'Web Services' menu is highlighted, and a sub-menu is visible with the options 'Create Web Service' and 'Generate JAX-RPC JSPs'. The 'Create Web Service' option is highlighted with a red rectangle. The background shows the Eclipse IDE with a toolbar, a console window at the bottom displaying logs for Apache Tomcat, and a taskbar at the bottom with various application icons.

Java EE - Eclipse

File Edit Navigate Search

Project Explorer

- loja-app
 - loja-app-web
 - JAX-WS Web Service
 - Deployment Descriptor
 - Java Resources
 - src
 - com.fiap.loja
 - EstoqueBO.java
 - Libraries
 - JavaScript Resources
 - build
 - WebContent
 - META-INF
 - WEB-INF
 - Servers

Web Services

- Create Web Service
- Generate JAX-RPC JSPs

Console

```
Apache Tomcat] C:\Program Files\Java\jre7\bin\javaw.exe (17/02/2013 20:43:53)
org.apache.coyote.AbstractProtocol start
andler ["ajp-bio-8009"]
org.apache.catalina.startup.Catalina start
6161 ms
```

GERANDO UM SERVIÇO WS

Web Service

Select a service implementation or definition and move the sliders to set the level of service and client generation.

Web service type: Bottom up Java bean Web Service

Service implementation: br.com.fiap.bo.ExemploBO

Start service

Configuration:
[Server runtime: Tomcat v8.0 Server](#)
[Web service runtime: Apache Axis](#)
[Service project: ExemploWs](#)

Click on the link to specify the web service

Client type: Java Proxy

No client

Configuration: No client generation.

☐ Publish the Web service
☐ Monitor the Web service
☐ Do not show me this dialog box again.

< Back Next > **Finish** Cancel

Service Deployment Configuration

Choose from the list of runtimes and deployment options

Server-Side Deployment Selection:

☐ Choose server first
☒ Choose Web service runtime first
☐ Explore options

Web service runtime:

Apache Axis
[Apache Axis2](#)
Apache CXF 2.x

- Criar um projeto Java Web Application com o nome “estoque-ws”
- Desenvolver um Web Services Provider para obter informações a respeito de produtos de uma loja.
- A classe responsável por prover as informações do produto será a classe “**com.fiap.inventario.Estoque**” com o método “**ProdutoTO buscarProduto(String codProduto)**”, este método deverá retornar a descrição do produto, quantidade em estoque e preço unitário do mesmo
- Gerar um AxisFault caso o produto não esteja cadastrado (throw new AxisFault(“Produto Não Cadastrado”))
- Gerar javadoc das classes
- Teste o serviço com o SOAP UI

TESTE COM SOAP UI

The screenshot displays the SoapUI 1.6 application window. The left sidebar shows a project tree with 'Test' > 'Calculo' > 'Request 1' selected. The main area is titled 'Request: [Request 1] - Calculo#calculo' and shows the SOAP request XML. Below the XML, the 'Request Properties' table is visible. The right side shows the SOAP response XML. At the bottom, a log window displays the execution details.

Request Properties

| Property | Value |
|--------------------|------------------------|
| Name | Request 1 |
| Encoding | UTF-8 |
| Endpoint | http://localhost:80... |
| Enable MTOM/Inline | false |
| Username | |
| Password | |
| Domain | |
| WSS-Passport Type | |
| Inline Response | false |

SOAP Request XML:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2001/XMLSchema-instance" xmlns:cal="http://schemas.xmlsoap.org/Calculo/">
  <soapenv:Header/>
  <soapenv:Body>
    <cal:calculo soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <a xsi:type="xsd:int">3</a>
      <b xsi:type="xsd:int">4</b>
    </cal:calculo>
  </soapenv:Body>
</soapenv:Envelope>
```

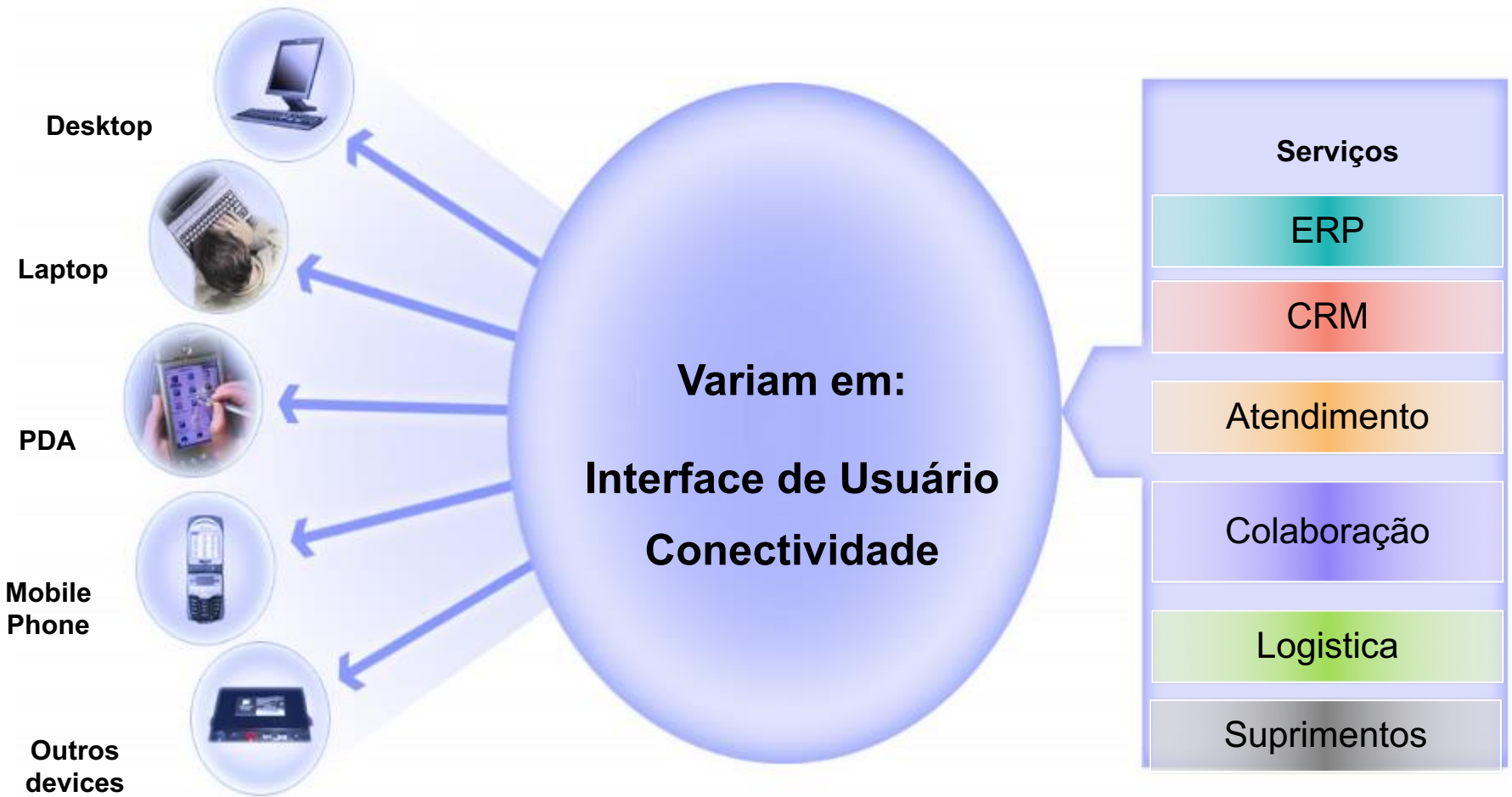
SOAP Response XML:

```
<?xml version='1.0' encoding='UTF-8'>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="http://schemas.xmlsoap.org/Calculo/">
  <soapenv:Body>
    <ns1:calculoResponse soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <calculoReturn xsi:type="xsd:int">7</calculoReturn>
    </ns1:calculoResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Log:

```
Sat Mar 03 01:49:52 BRT 2007:INFO:Importing binding {http://calculo.fiap.com}CalculoSoapBinding
Sat Mar 03 01:49:52 BRT 2007:INFO:importing endpoint http://localhost:8080/TAFIAPCalculoWS/services/Calculo
Sat Mar 03 01:49:52 BRT 2007:INFO:importing operation calculo
Sat Mar 03 01:50:17 BRT 2007:INFO:Got response for [Calculo.calculo:Request 1] in 507ms (458 bytes)
```

WEB SERVICE REQUESTER - CONSOLE



WEB SERVICE REQUESTER (CLIENT)

- Crie um Dynamic Web Project;
- Configure o Target Runtime: Apache Tomcat 9;
- Dynamic web module version: 2.5;
- Configuration: Modify -> Marque: Axis 2 Web Services

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name:

Project location

☒ Use default location

Location:

Target runtime

Dynamic web module version

Configuration

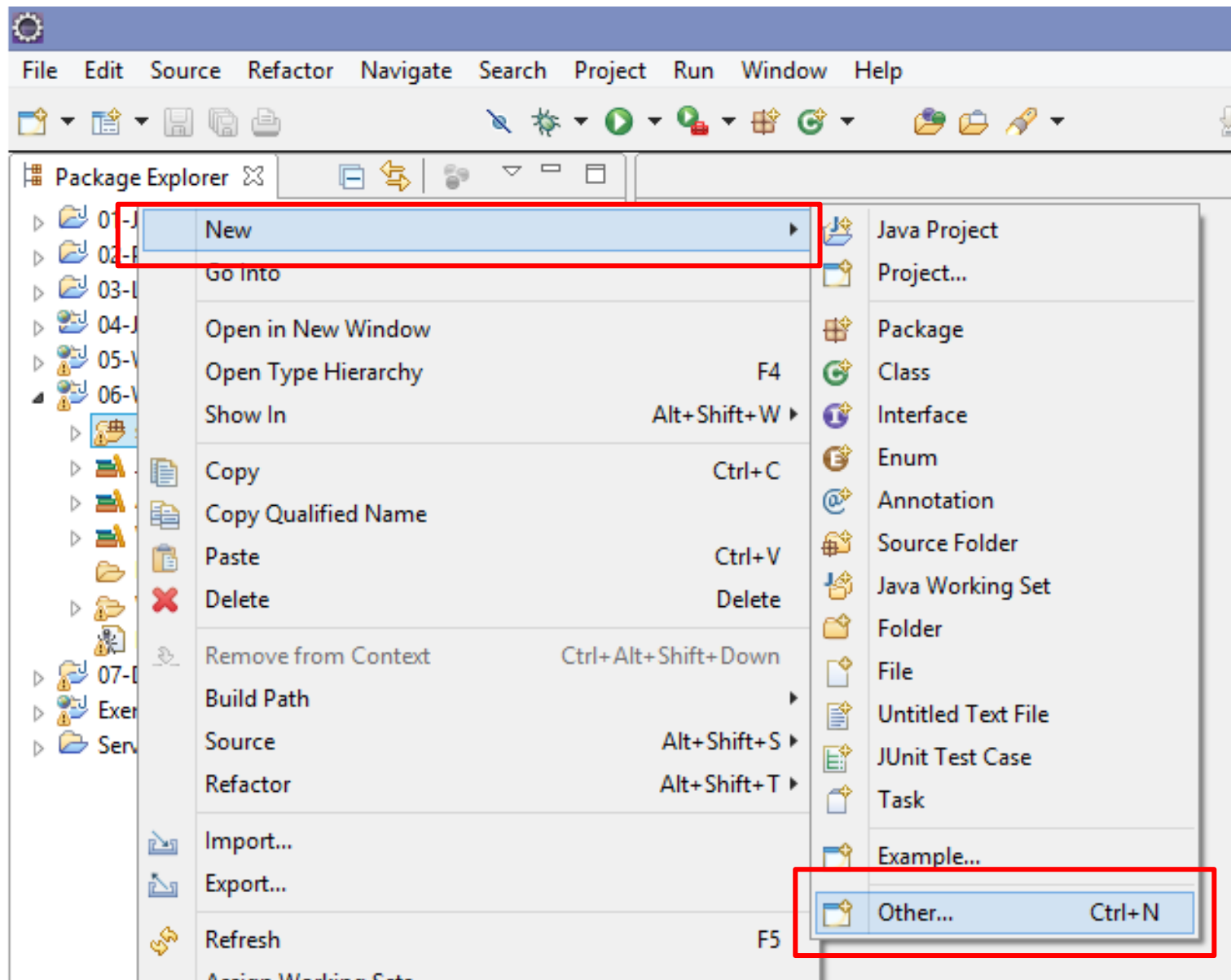
Project Facets

Select the facets that should be enabled for this project.

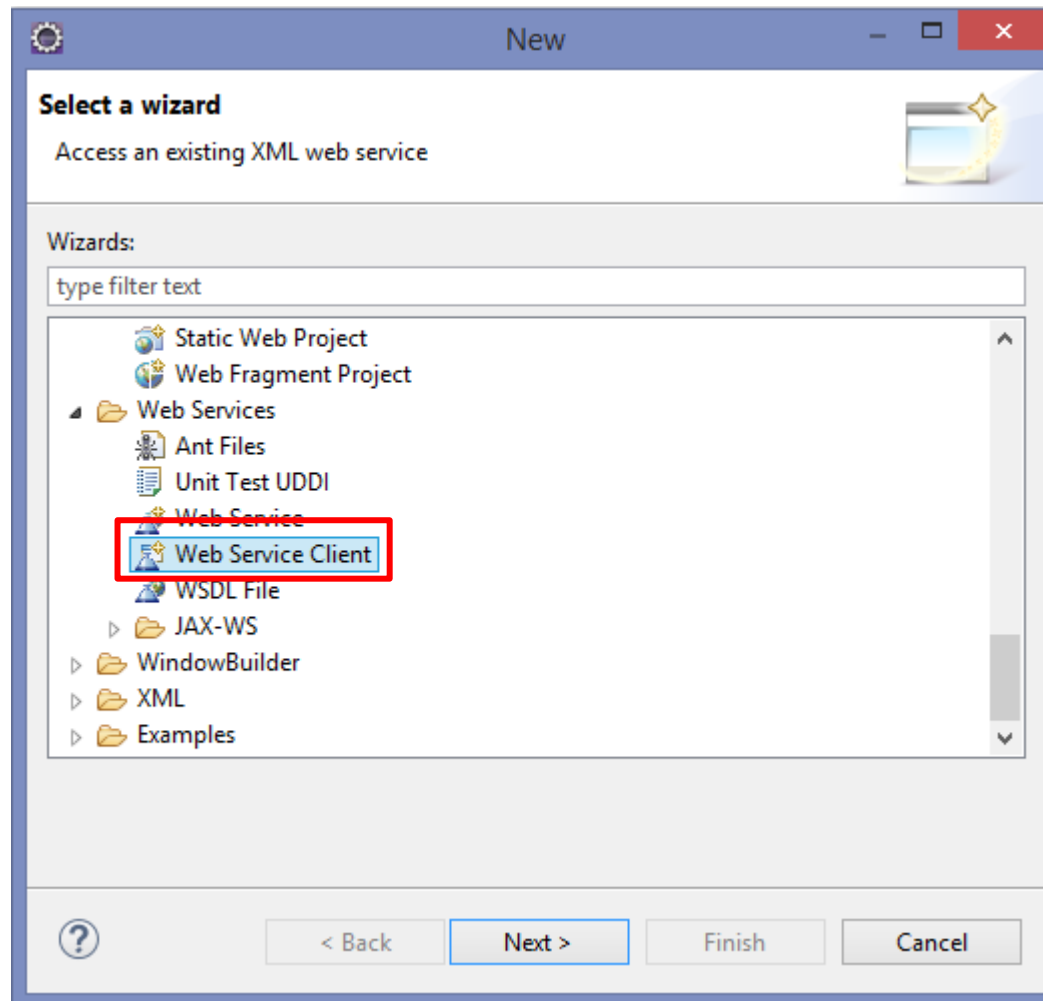
Configuration:

| Project Facet | Version |
|--|---------|
| <input checked="" type="checkbox"/> Axis2 Web Services | |
| <input type="checkbox"/> CXF 2.x Web Services | 1.0 |
| <input checked="" type="checkbox"/> Dynamic Web Module | 2.5 |
| <input checked="" type="checkbox"/> Java | 1.8 |
| <input type="checkbox"/> JavaScript | 1.0 |
| <input type="checkbox"/> JavaServer Faces | 2.2 |
| <input type="checkbox"/> JAX-RS (REST Web Services) | 1.1 |
| <input type="checkbox"/> JAXB | 2.2 |
| <input type="checkbox"/> JPA | 2.1 |
| <input type="checkbox"/> WebDoclet (XDoclet) | 1.2.3 |

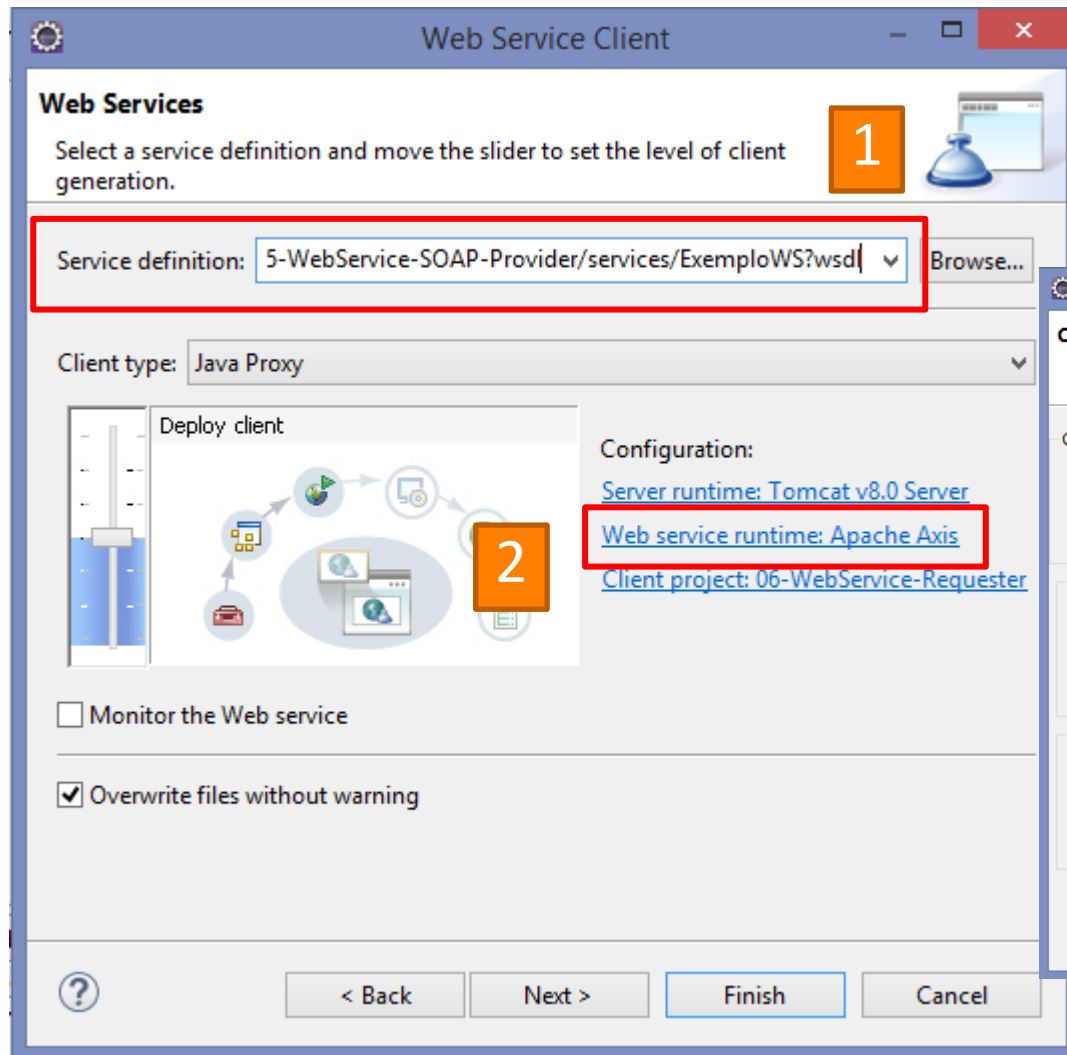
CRIANDO O WEB SERVICE CLIENT



❑ CRIANDO O WEB SERVICE CLIENT



! CRIANDO O WEB SERVICE CLIENT



The 'Web Service Client' dialog box is shown. It has a title bar with a gear icon and standard window controls. The main area is titled 'Web Services' and contains instructions: 'Select a service definition and move the slider to set the level of client generation.' A red box labeled '1' highlights the 'Service definition' dropdown menu, which currently shows '5-WebService-SOAP-Provider/services/ExemploWS?wsdl'. Below this, the 'Client type' is set to 'Java Proxy'. A 'Deploy client' section features a slider and a diagram of a client-server interaction, with a red box labeled '2' highlighting the diagram. To the right of the diagram, the 'Configuration' section lists: 'Server runtime: Tomcat v8.0 Server', 'Web service runtime: Apache Axis' (highlighted with a red box), and 'Client project: 06-WebService-Requester'. At the bottom, there are checkboxes for 'Monitor the Web service' (unchecked) and 'Overwrite files without warning' (checked). Navigation buttons at the bottom include '< Back', 'Next >', 'Finish', and 'Cancel'.

Web Services

Select a service definition and move the slider to set the level of client generation.

Service definition: 5-WebService-SOAP-Provider/services/ExemploWS?wsdl

Client type: Java Proxy

Deploy client

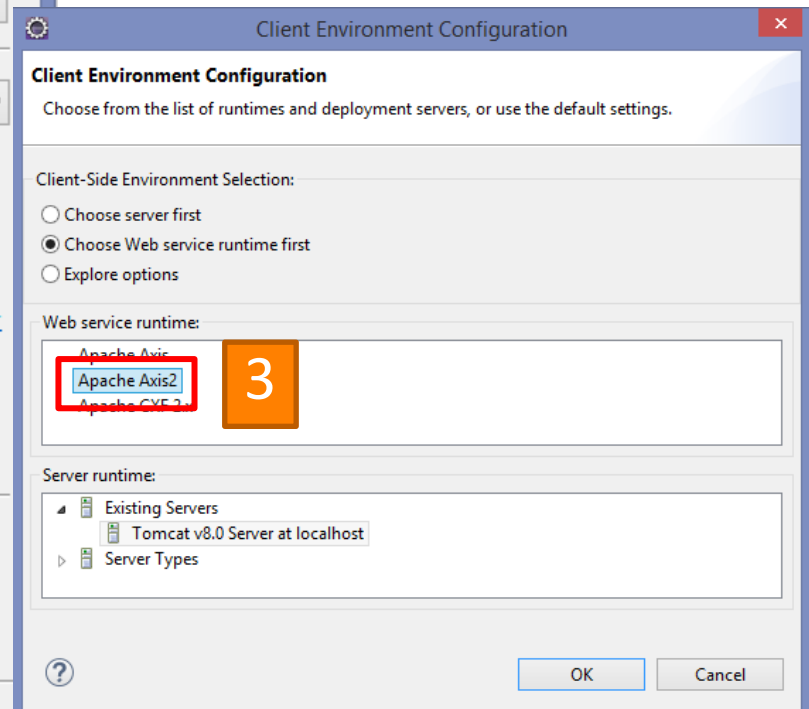
Configuration:

- Server runtime: Tomcat v8.0 Server
- Web service runtime: Apache Axis
- Client project: 06-WebService-Requester

☐ Monitor the Web service

☒ Overwrite files without warning

< Back Next > Finish Cancel



The 'Client Environment Configuration' dialog box is shown. It has a title bar with a gear icon and standard window controls. The main area is titled 'Client Environment Configuration' and contains instructions: 'Choose from the list of runtimes and deployment servers, or use the default settings.' Under 'Client-Side Environment Selection', there are three radio buttons: 'Choose server first', 'Choose Web service runtime first' (selected), and 'Explore options'. Below this, the 'Web service runtime' section shows a list with 'Apache Axis', 'Apache Axis2' (highlighted with a red box), and 'Apache CXF 2'. A red box labeled '3' highlights the 'Apache Axis2' option. The 'Server runtime' section shows a tree view with 'Existing Servers' (containing 'Tomcat v8.0 Server at localhost') and 'Server Types'. At the bottom right, there are 'OK' and 'Cancel' buttons.

Client Environment Configuration

Choose from the list of runtimes and deployment servers, or use the default settings.

Client-Side Environment Selection:

- ☐ Choose server first
- ☒ Choose Web service runtime first
- ☐ Explore options

Web service runtime:

- Apache Axis
- Apache Axis2
- Apache CXF 2

Server runtime:

- Existing Servers
 - Tomcat v8.0 Server at localhost
- Server Types

OK Cancel


```
View.java
1 package br.com.fiap.main;
2
3 import br.com.fiap.bo.ExemploWSStub;
4
5
6
7
8
9 public class View {
10
11     public static void main(String[] args) throws Exception {
12
13         ExemploWSStub stub = new ExemploWSStub();
14
15         //Cadastrar um carro
16         CarroTO param = new CarroTO();
17         param.setModelo("GOL");
18         param.setPreco(123);
19
20         Cadastrar cadastrar = new Cadastrar();
21         cadastrar.setCarro(param);
22         stub.cadastrar(cadastrar);
23
24         //Listar um carro
25         Listar listar = new Listar();
26
27         ListarResponse response = stub.listar(listar);
28
29         CarroTO[] vetor = response.get_return();
30
31         for (CarroTO carroTO : vetor) {
32             System.out.println(carroTO.getModelo());
33             System.out.println(carroTO.getPreco());
34         }
35     }
36
37 }
```

EXERCÍCIO – WS REQUESTER

- Acessar e executar o wsdl via SOAP UI no serviço:
 - » <http://192.168.60.200:8080/estoque-ws/services/Estoque>
- Criar um projeto Java Application com o nome “loja-central-app”
- Esta aplicação será um Web Services Requester
- Desenvolver a classe “com.fiap.loja.TerminalBuscaPreco” como Java Console em que o usuário informe o código do produto e a aplicação faça a pesquisa de produto
- A aplicação pesquisará a descrição dos produtos, quantidade em estoque e valor unitário utilizando o serviço provido pelo Web Service Provider no servidor



WEB SERVICE REQUESTER – USO DE COMPLEX TYPE COM COLEÇÕES DE DADOS

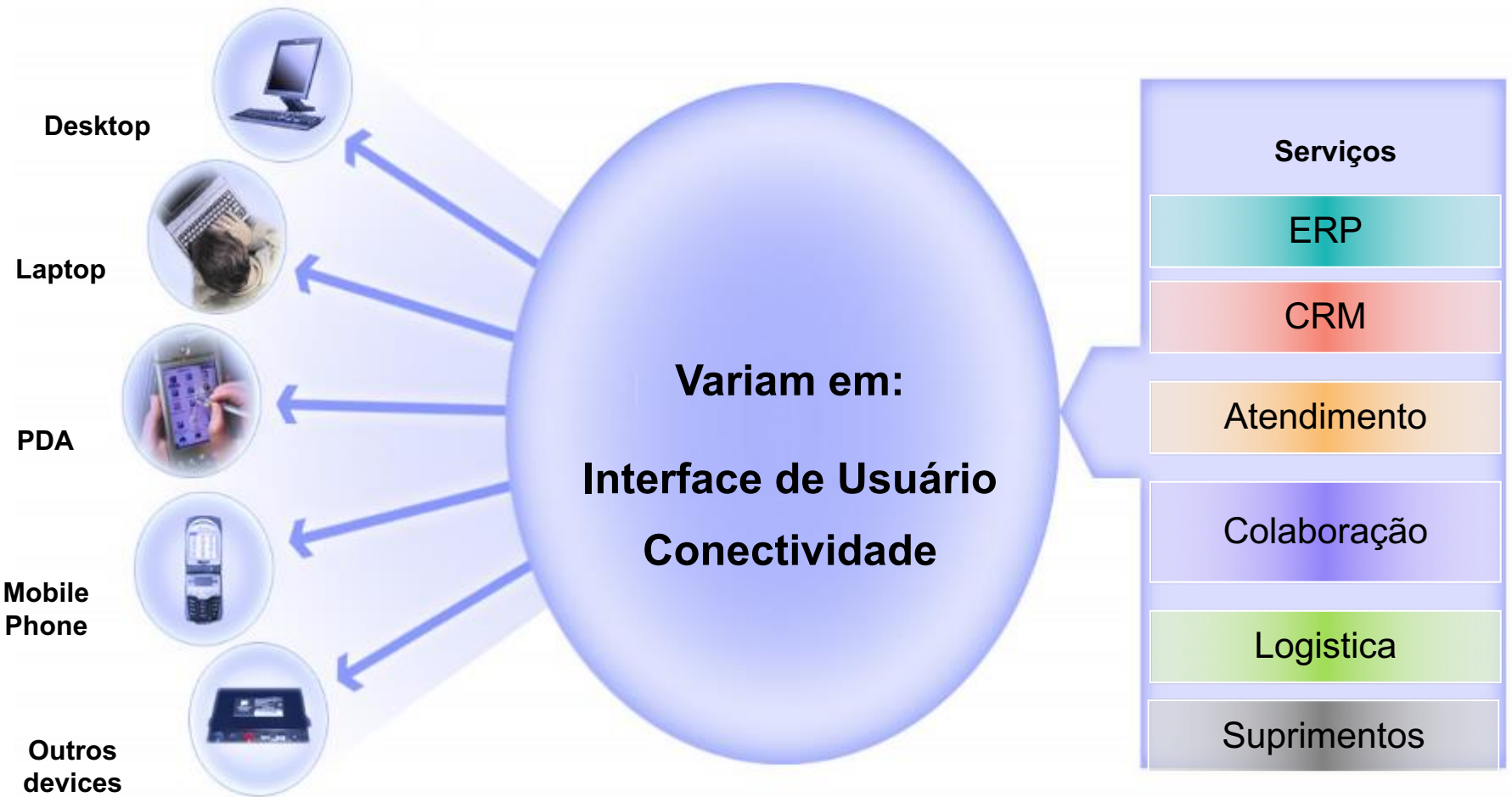
- Complex Type são objetos de estrutura de dados que são repassados para o Service Requester a partir de uma requisição processada pelo Service Provider (maiores informações na aula de Web Services Provider)
- A implementação de Apache Axis 1 utiliza arrays para a passagem de coleção de dados
- Caso você utilize uma lista como ArrayList, é necessário converter o valor para Array[] antes de repassar os dados, através do método **toArray** do objeto ArrayList
- Afinal, o que é Apache Axis?
- Apache Axis é uma implementação do protocolo SOAP que permite a criação e publicação de WebServices (WebService Provider) e a criação de clientes de acesso (WebService Requester) de forma simplificada e também automática, com o auxílio das ferramentas disponíveis no próprio Apache Axis. Considerado um framework.

//...

```
List<ProdutoTO> listaProdutos = new ArrayList<ProdutoTO>();  
listaProdutos = Arrays.asList(response.get_return());
```

```
for(ProdutoTO e : listaProdutos) {  
    System.out.println(e.getDescricao());  
    System.out.println(e.getQuantidade ());  
}
```

WEB SERVICE REQUESTER - DESKTOP



Para desenvolver aplicações desktop no eclipse podemos utilizar Swing, ATW e etc..

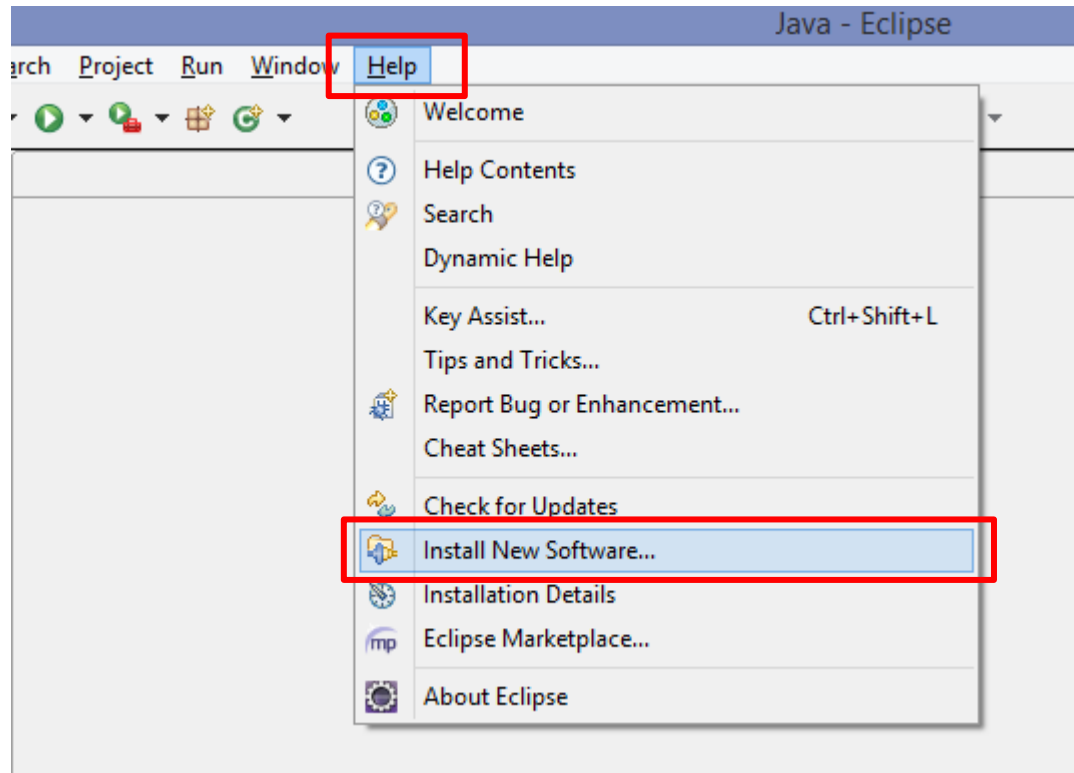
Nós vamos utilizar o **SWT: The Standard Widget Toolkit**

Para isso, precisamos instalar um plugin. (Na fiap já está instalado).



INSTALAÇÃO DO PLUGIN

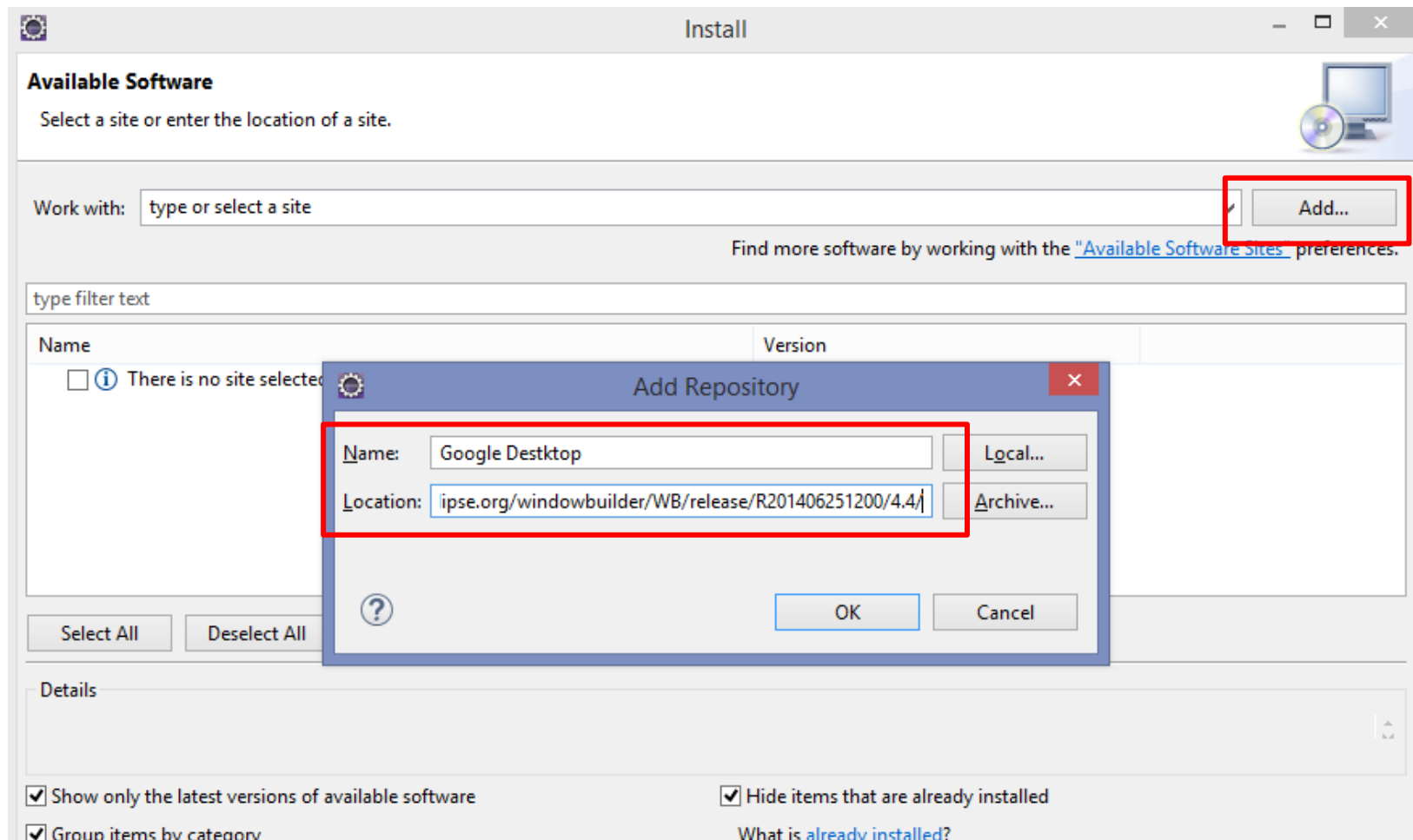
- Acesse a opção: Help -> Install New Software...



INSTALAÇÃO DO PLUGIN

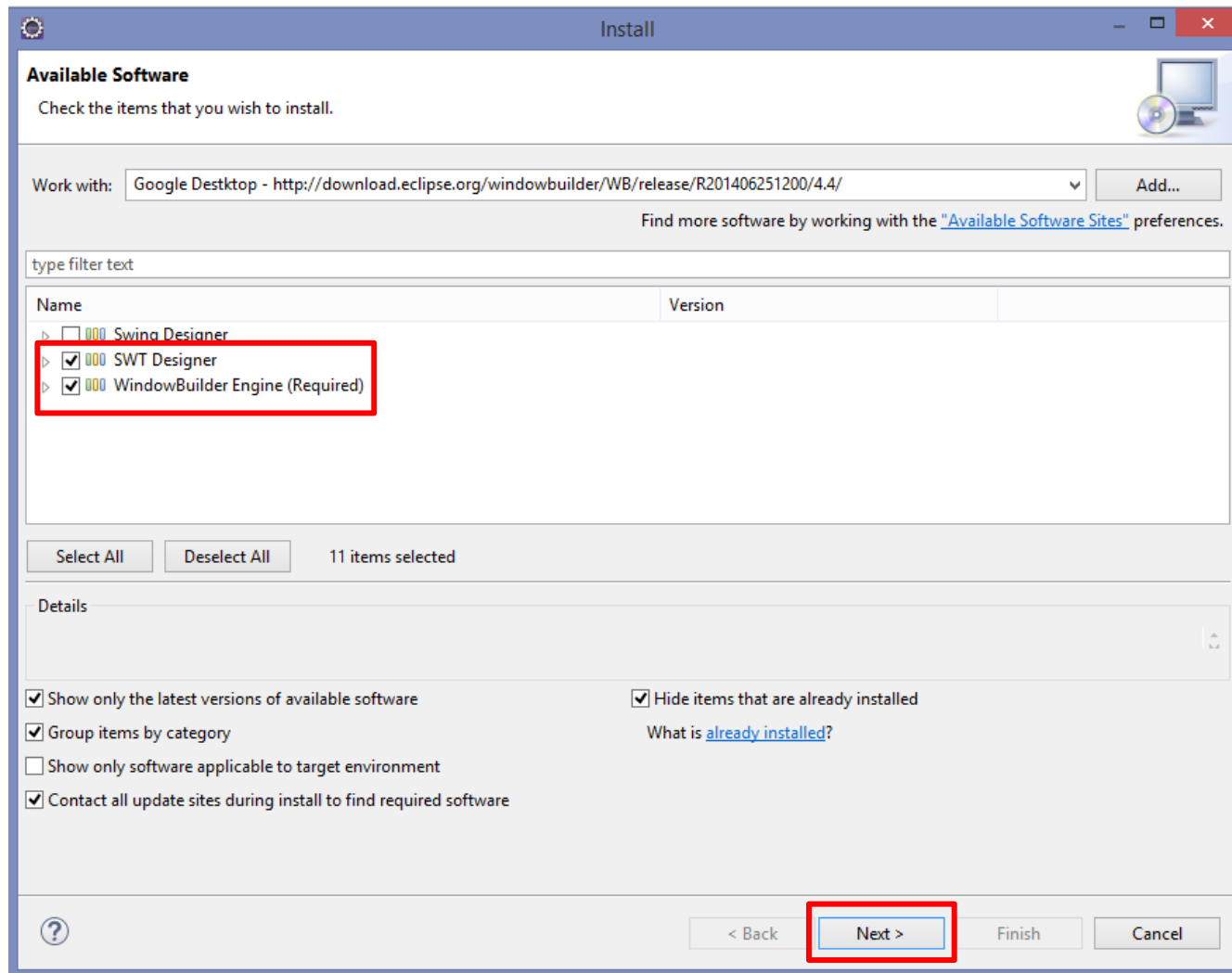
- Clique no botão Add..
- Adicione um nome ao Repository e o seguinte Location:

<http://download.eclipse.org/windowbuilder/WB/integration/4.7/>



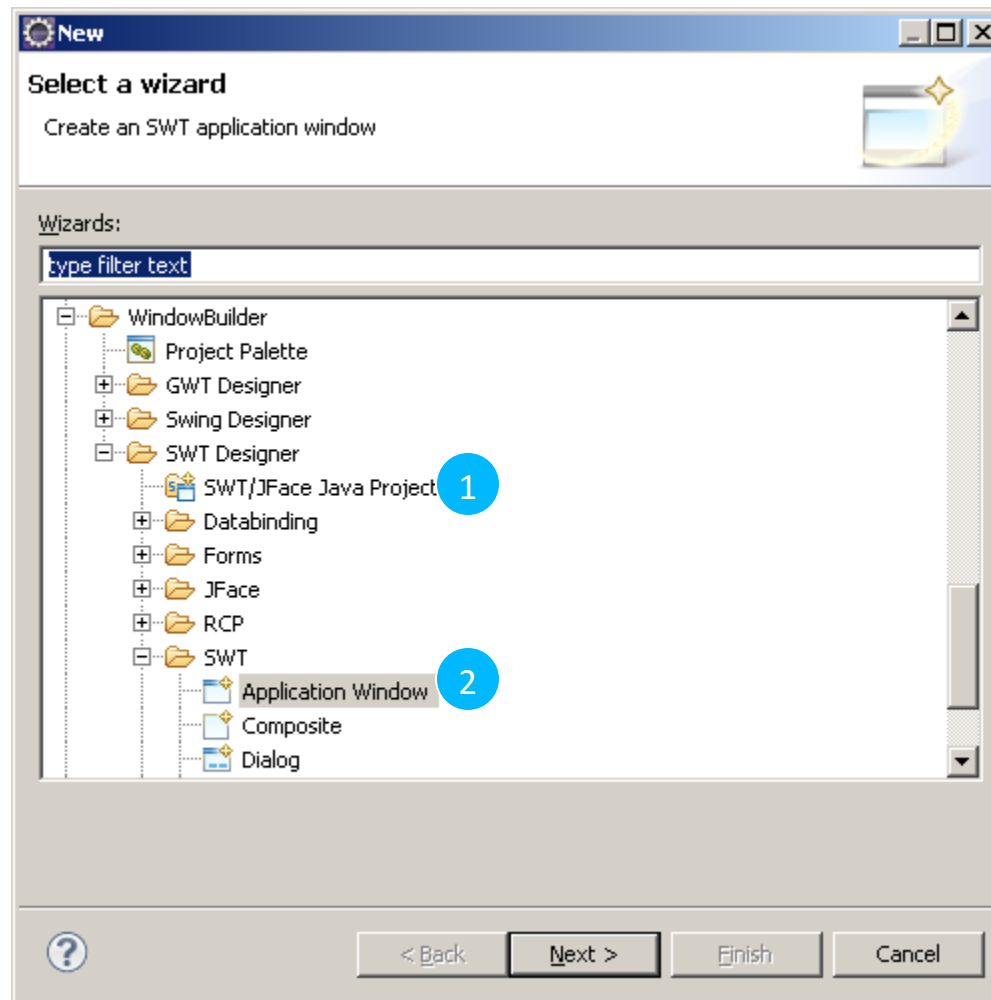
INSTALAÇÃO DO PLUGIN

- Marque as opções: **SWT Designer** e **WindowBuilder Engine** e clique em next. Depois next e depois aceite os termos e finish.

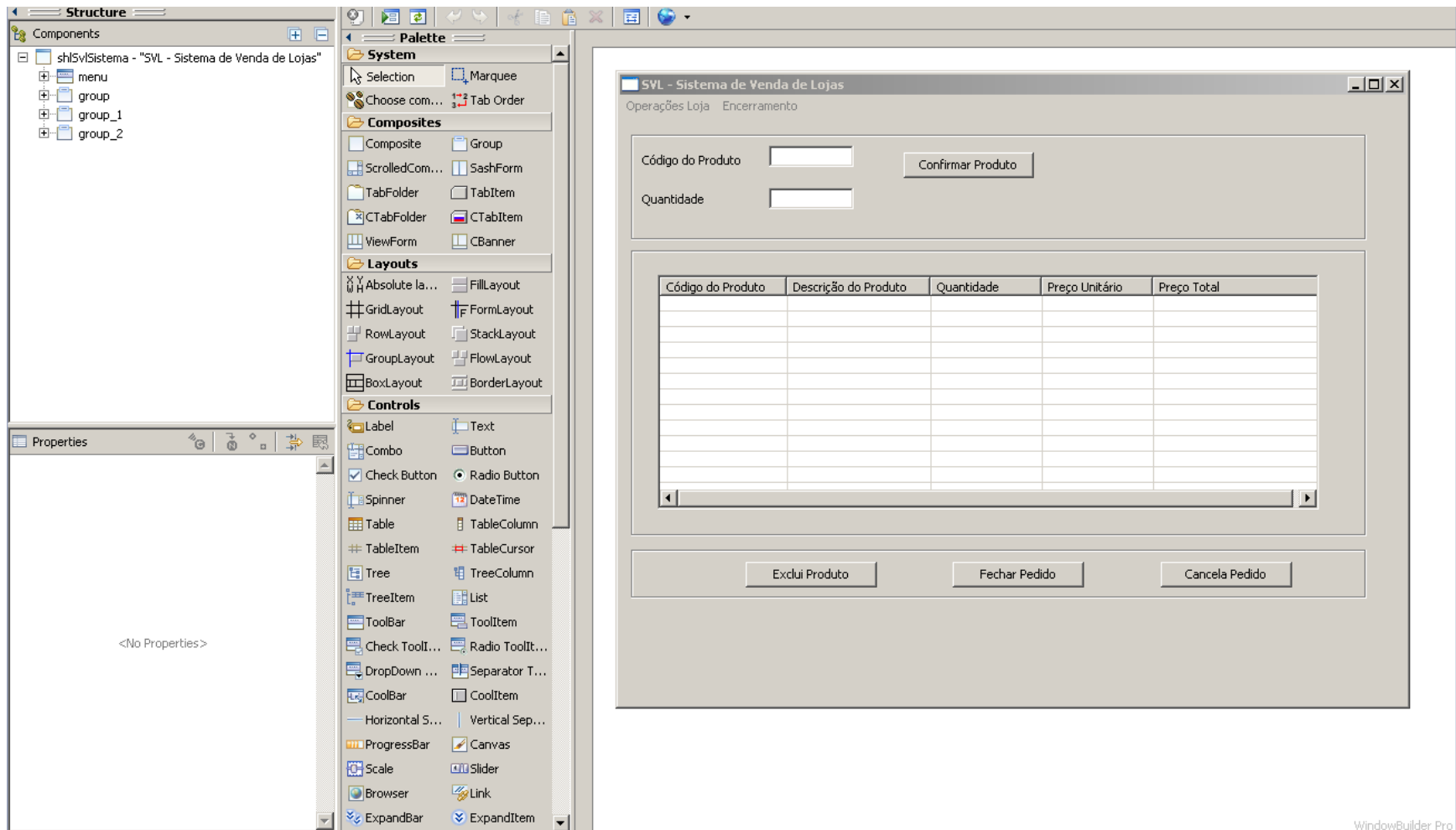


SWT - STANDARD WIDGET TOOLKIT

- Alternativa ao AWT/Swing
- Usa bibliotecas nativas do sistema operacional
- Desenvolvimento acelerado com o Google Window Builder Plugin



BARRA DE FERRAMENTAS



- Controls são objetos que permitem a interação com a interface preparada para o usuário
- Estes controls podem ser instanciados visualmente utilizando o WindowBuilder
- **Label**: Apresenta dados na tela
 - Propriedade **Font** permite alterar aparência visual do objeto
 - Propriedade **Foreground** permite alterar cor do texto
 - Propriedade **Image** permite você incluir uma imagem no objeto
 - Selecione o objeto -> botão direito -> Autosize Control para ajustar texto
- **Text**: Permite apresentação e edição de dados
 - Propriedade **Variable** permite definir um nome para o objeto.
 - Prefixo padrão de nome **“txt”**
 - Propriedade **Editable** permite definir se o campo é de consulta ou para preenchimento de dados
 - Método **getText()** permite recuperar o texto digitado pelo usuário
 - Método **setText()** permite alterar o conteúdo texto de um objeto na tela

- Controls são objetos que permitem a interação com o a interface preparada para o usuário
- Estes controls podem ser instanciados visualmente utilizando o WindowBuilder
- **Table**: Apresenta dados em formato de tabela
 - Propriedade **Variable** define o nome do objeto a ser instanciado.
 - Prefixo padrão de nome **“tbl”**.
- **TableItem**: Filho de Table. Representa as colunas de uma tabela
 - Propriedade **Text** define nome das colunas
 - Para incluir um novo item de forma programática basta instanciar um objeto do tipo TableItem (conforme abaixo)
 - A seqüência das colunas deve obedecer a seqüência de inserção

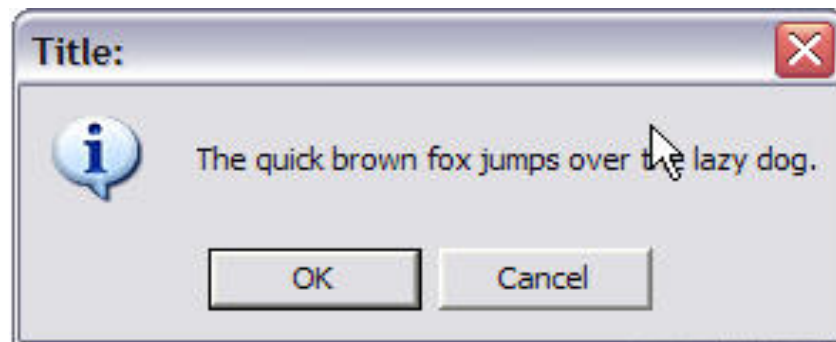
```
TableItem item = new TableItem(table, 0);  
item.setText(new String[] {"1", "Calça"});
```

- **Button**: Ação de usuário por meio de botão
- Prefixo padrão de nome **“btn”**.
- Para incluir uma ação no botão selecione o objeto -> botão direito -> Add Event Handler -> Selection -> widgetSelected para incluir uma ação no momento em que o usuário clicar no objeto

```
Button b = new Button(shell,SWT.PUSH);  
b.setText(“OK”);  
b.addSelectionListener(new SelectionAdapter() {  
    public void widgetSelected(SelectionEvent e) {  
        MessageDialog.openInformation(shell, “SWT”, “Ola Mundo!”);  
    }  
});
```

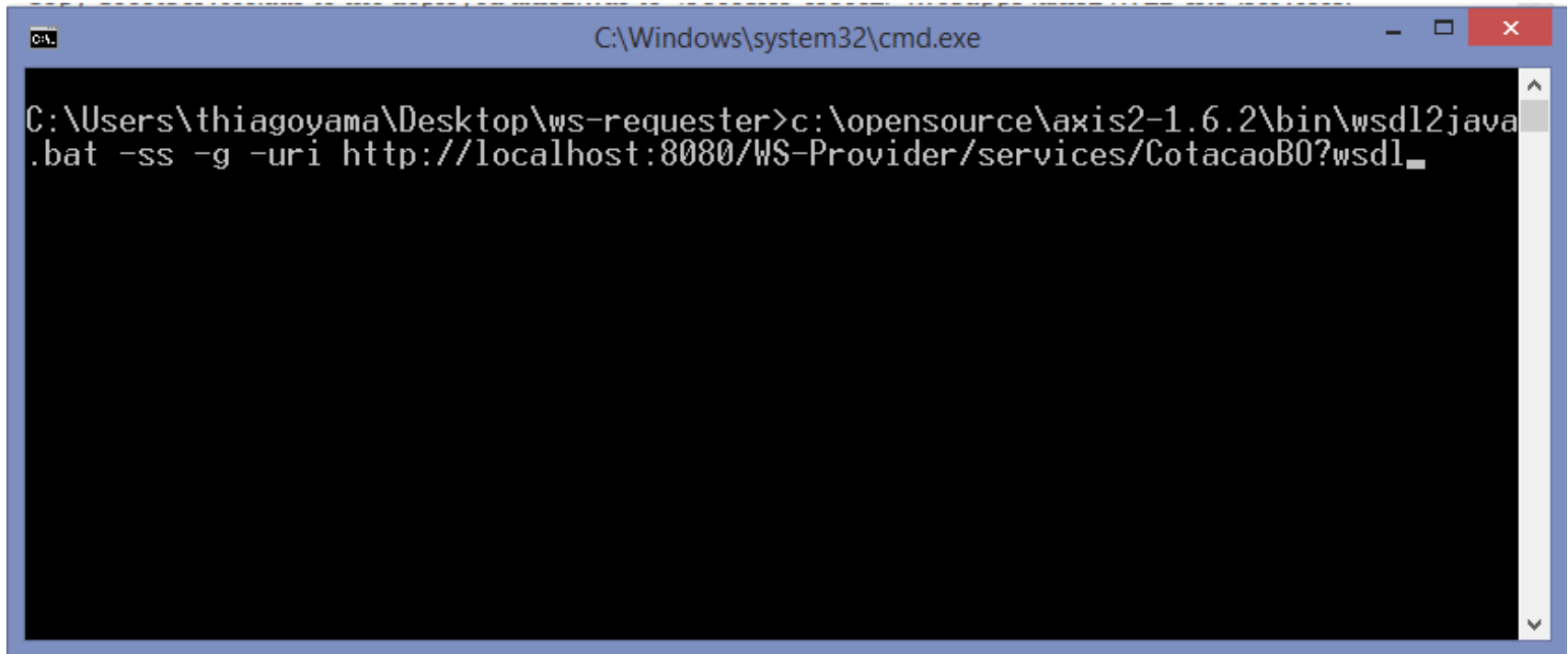

I MENSAGEM DE CONFIRMAÇÃO

```
MessageBox mb = new MessageBox(shell, SWT.OK|SWT.CANCEL);  
mb.setMessage("Clique OK caso queira encerrar a aplicação");  
int result = mb.open();  
if ( result == SWT.OK) {  
    System.out.println("OK foi pressionado");  
    System.exit(0); /* encerra programa  
}  
if (result == SWT.CANCEL)  
    System.out.println("cancela foi pressionado");
```



WEB SERVICE REQUESTER

- Para gerar as classes de acesso ao web service, vamos utilizar uma ferramenta do axis2:



```
C:\Windows\system32\cmd.exe

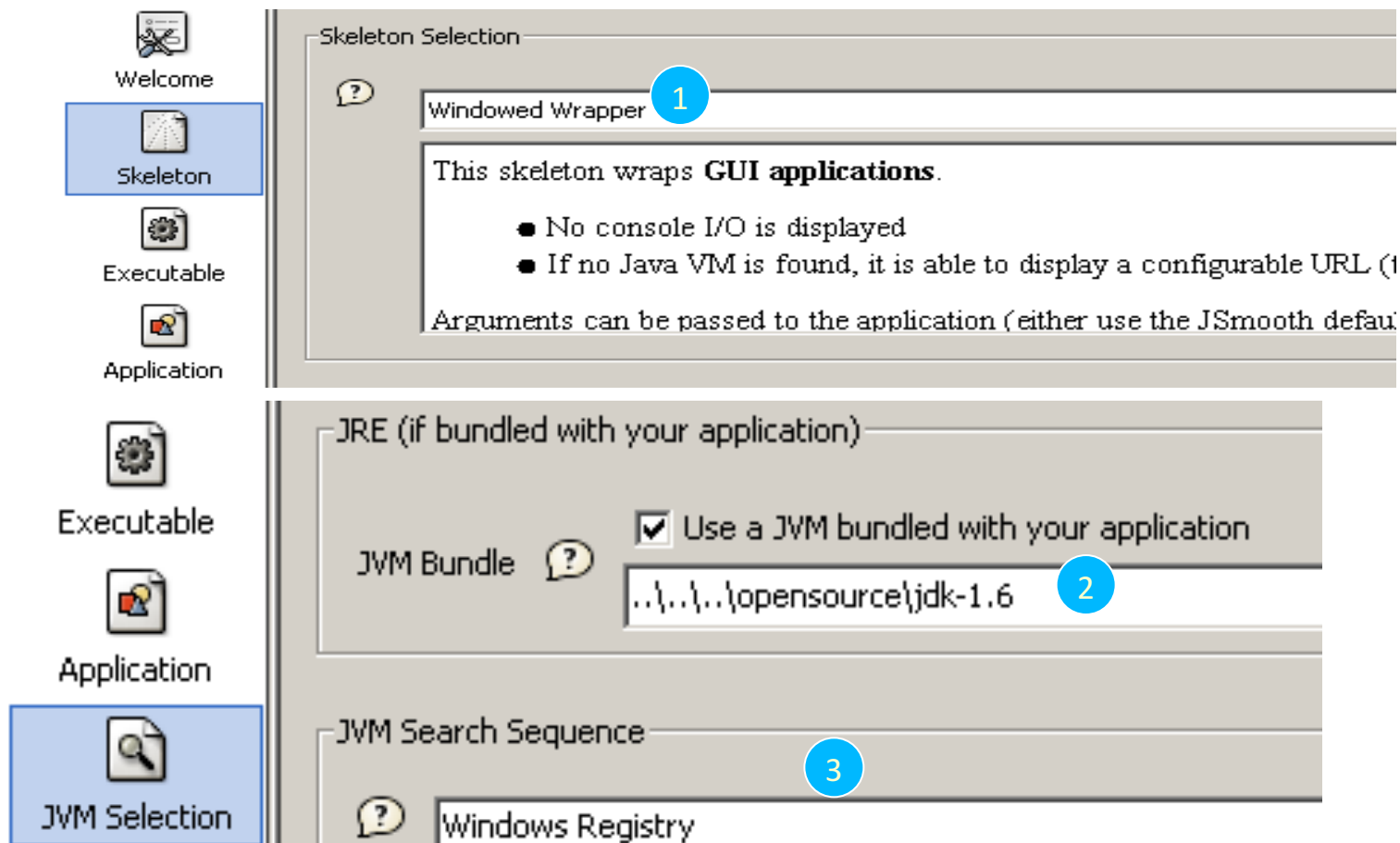
C:\Users\thiagoyama\Desktop\ws-requester>c:\opensource\axis2-1.6.2\bin\wsdl2java
.bat -ss -g -uri http://localhost:8080/WS-Provider/services/CotacaoB0?wsdl_
```

EXERCÍCIO WS REQUESTER - DESKTOP FIAP

- Desenvolva um Web Services Requester com Java Desktop para ser utilizado na aplicação de terminal de caixa para pesquisar informações a respeito dos produtos da loja que estão sendo vendidos. A aplicação pesquisará a descrição dos produtos, quantidade em estoque e valor unitário utilizando o serviço provido pelo Web Service Provider no servidor.

DEPLOYMENT

- Deployment de SWT segue as mesmas regras de aplicações Java (utilizar o Runnable Jar File)
- Para os casos de aplicações em 64bits é necessário adequar a JVM conforme figura abaixo
- Gerar executável no drive D:



EXERCÍCIO – APLICAÇÃO DESKTOP

- Criar um projeto SWT/JFace Java Project com o nome “loja-gui” com a classe “com.fiap.loja.Caixa” em que o usuário informe o código do produto
- Esta classe deverá consumir um Web Services para pesquisar informações a respeito dos produtos no momento em que o usuário pressionar “Pesquisar” e incluir o produto na tabela depois de pressionar o “Confirmar Compra”.
- Após realizar a pesquisa, a aplicação deverá apresentar uma mensagem na tela informando que a pesquisa ocorreu com sucesso
- Deverá haver um botão para encerrar a aplicação. O usuário deverá ter a possibilidade de fazer uma última confirmação antes de encerrar a aplicação



Registro de Caixa

FIAP

Código do Produto:

Produto:

Quantidade:

| Código | Nome do Produto | Quantidade |
|--------|-----------------|------------|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Copyright © 2013 - 2018 Prof. Me. Thiago T. I. Yamamoto

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”*