

MBA⁺

***MBA em Desenvolvimento
de Aplicações Java - SOA e
Internet das Coisas***

MBA⁺

Mule

Laboratório 5

Prof. André Pereira, MSc



“Só porque alguém tropeça e erra o caminho, não quer dizer que está perdido para sempre”



FIAP

Agenda

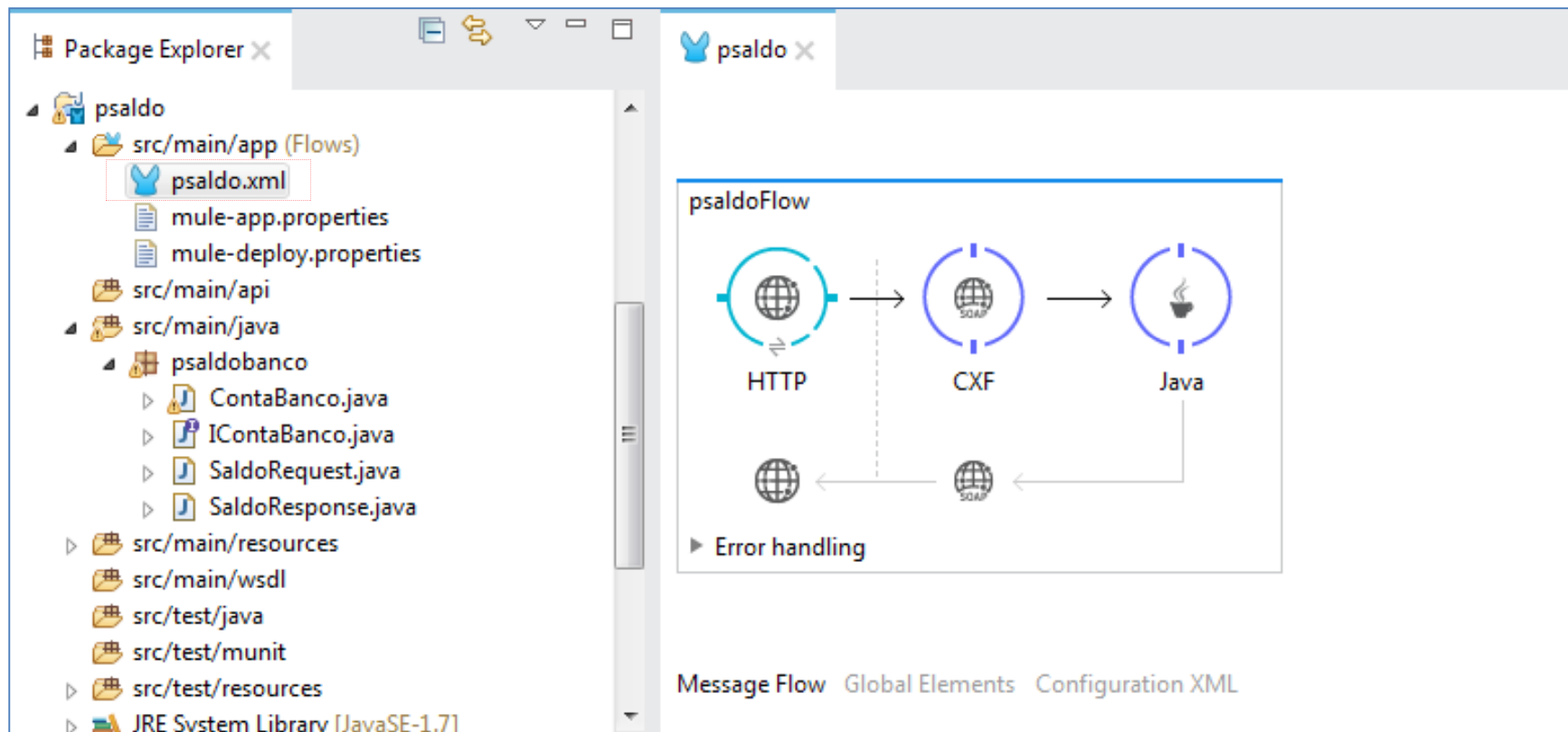
Agenda

- **O Que será feito nesse Lab.**
 - **Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”.**
 - **Testando o serviço “Proxy” no Mule Runtime e SoapUI.**

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

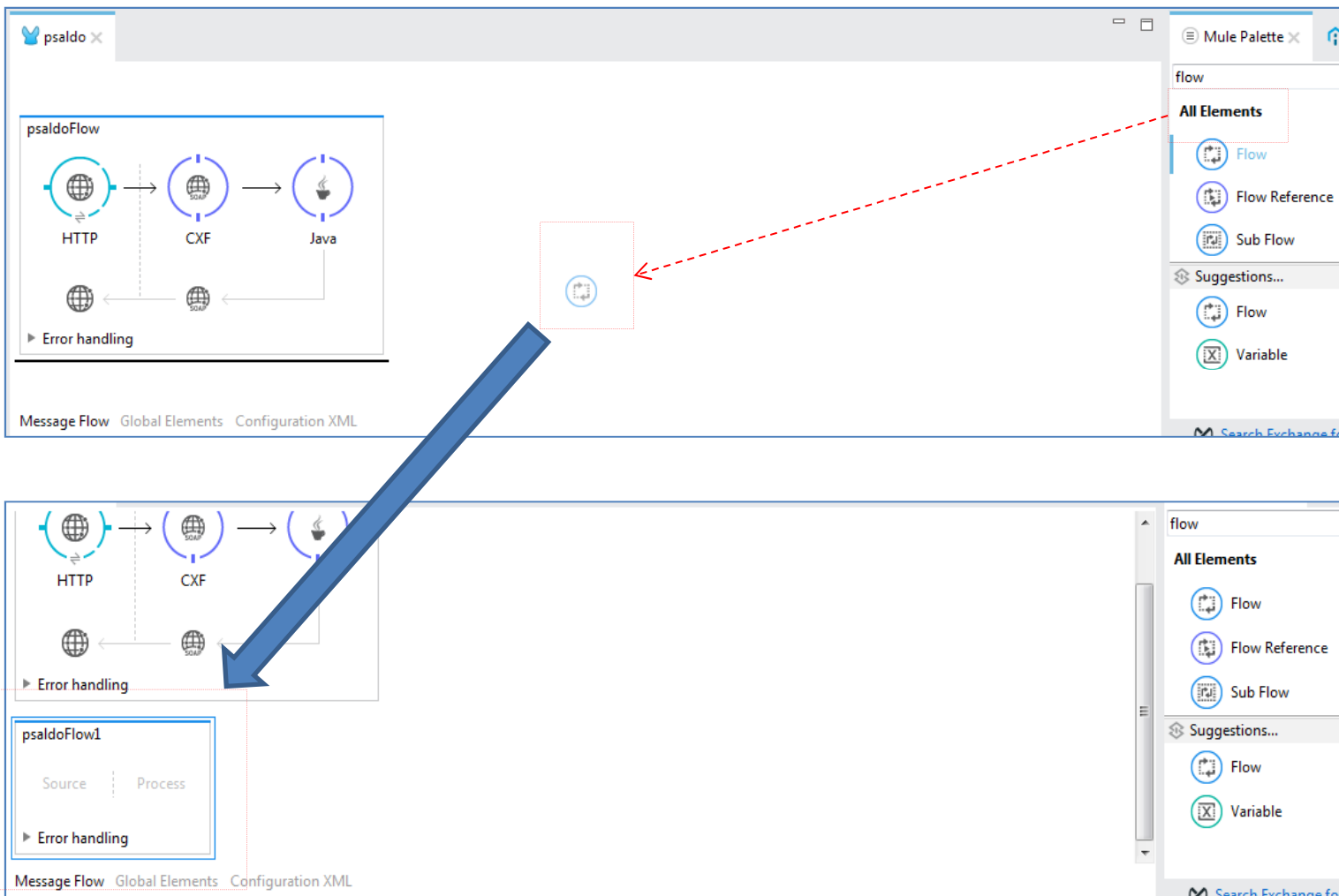
1) Vamos aumentar o projeto “Saldo” criando um serviço de “Proxy” contendo dados alterados. Se o mesmo não estiver aberto, abra o projeto “psaldo.xml”.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

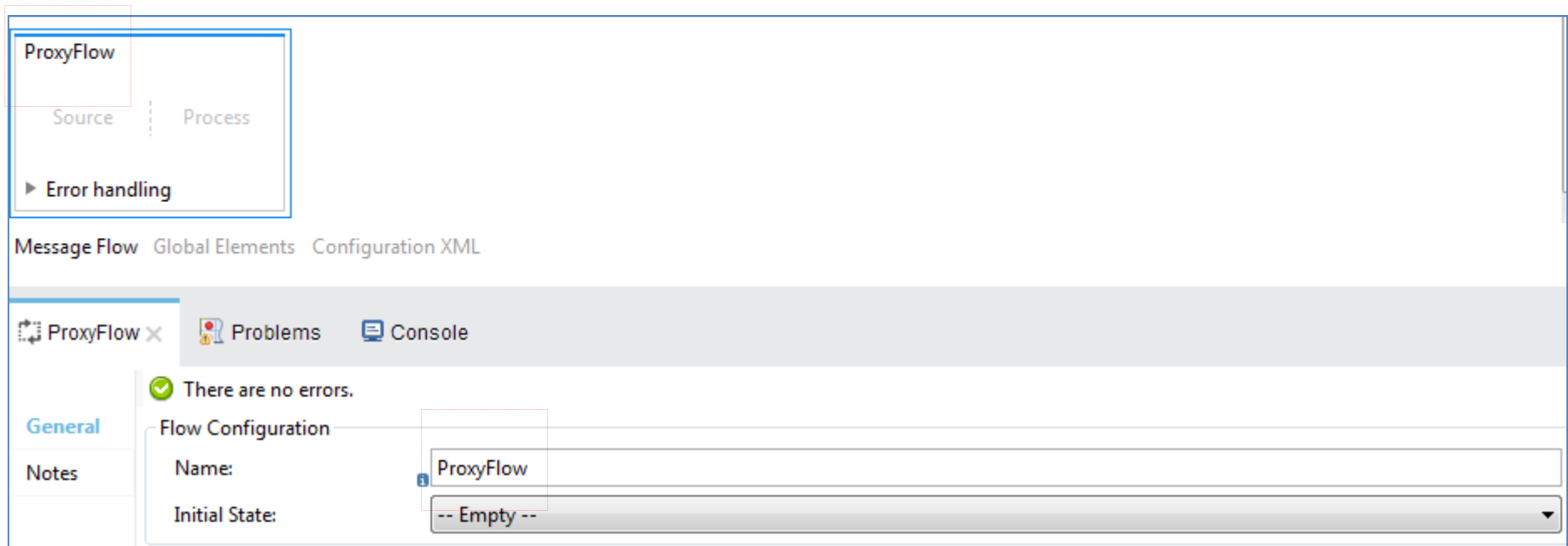
2) Vá até a paleta e escolha o componente “Flow”. Clique e arraste até a área de trabalho conforme a figura abaixo.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

3) Troque o nome do novo fluxo para “ProxyFlow” conforme a figura abaixo.

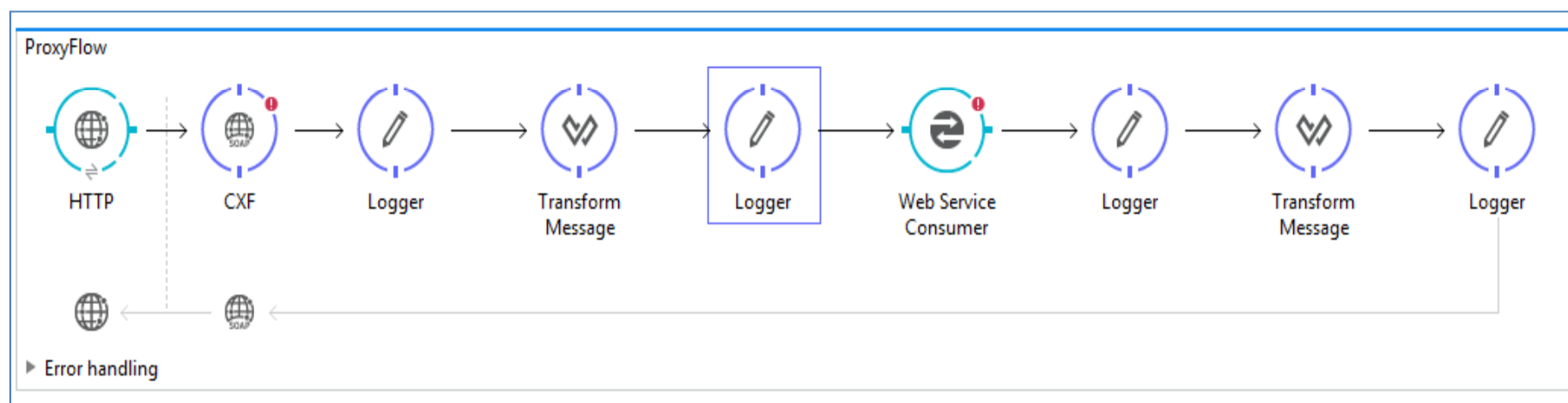


Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

4) Vamos criar a lógica do proxy. Arraste as seguintes quantidades de componentes:

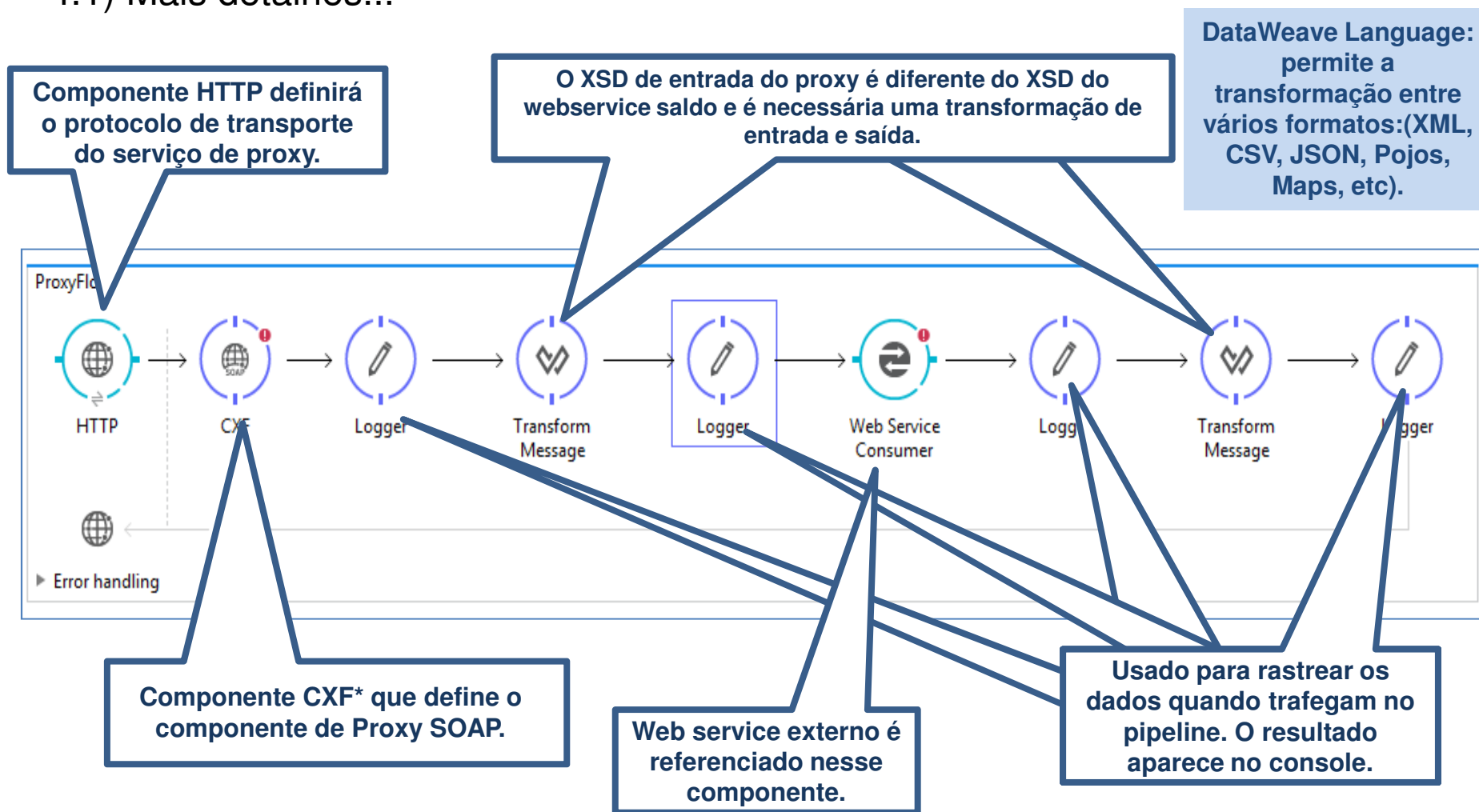
- 1 componente “HTTP”
- 4 componentes “Logger”
- 2 componentes “Transform Message”
- 1 componente “Web Service Consumer”

para o “ProxyFlow” de acordo com a sequencia abaixo:



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

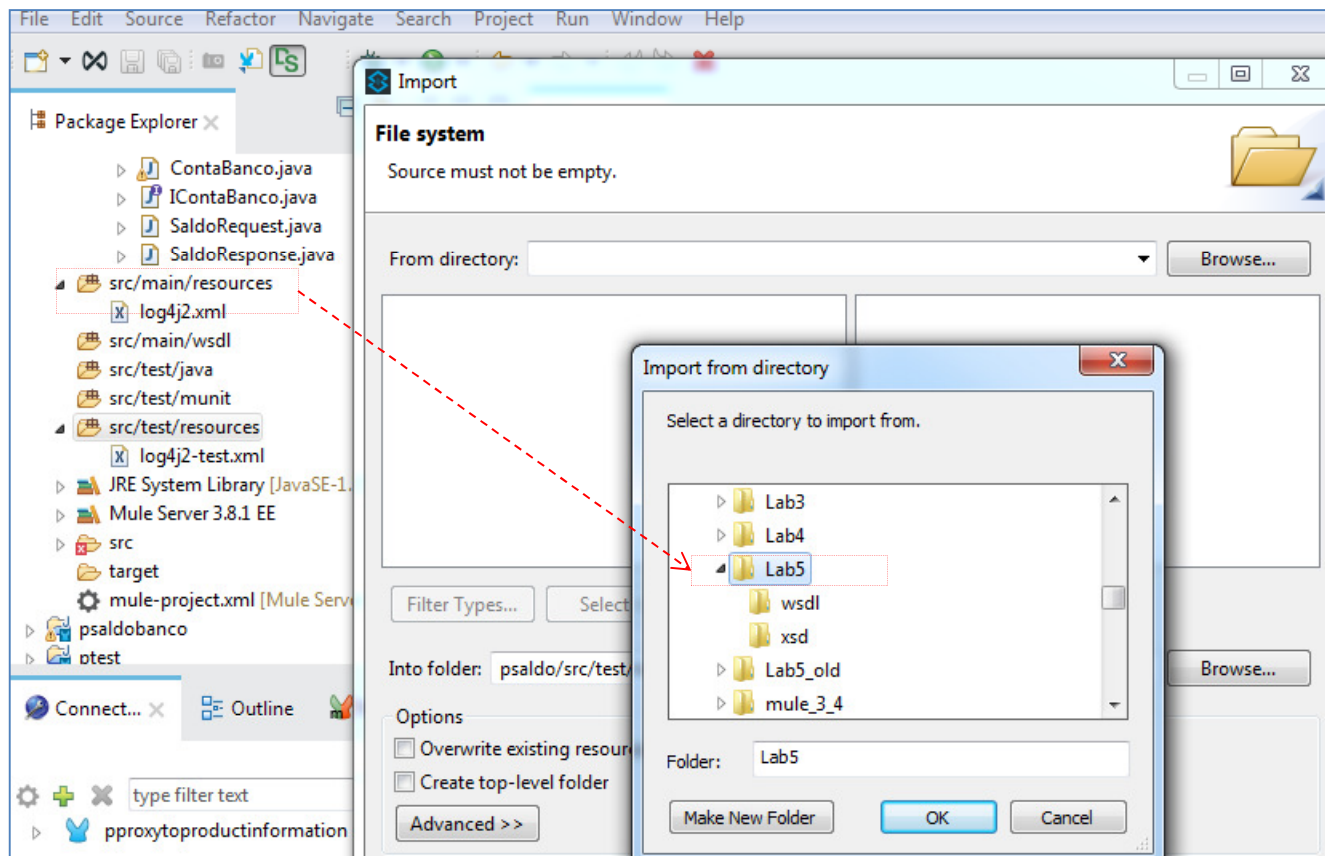
4.1) Mais detalhes...



Apache CXF* (projetos "Celtix" and "XFire".) é um framework open source que dá suporte a criação e consumo de Web Services e está de acordo com JAX-WS e JAX-RS. Suporta: SOAP, XML/HTTP, RESTful HTTP ou CORBA. Um Componente Proxy permite implementar padrões WS* tais como: WS-Security ou WS-Addressing.

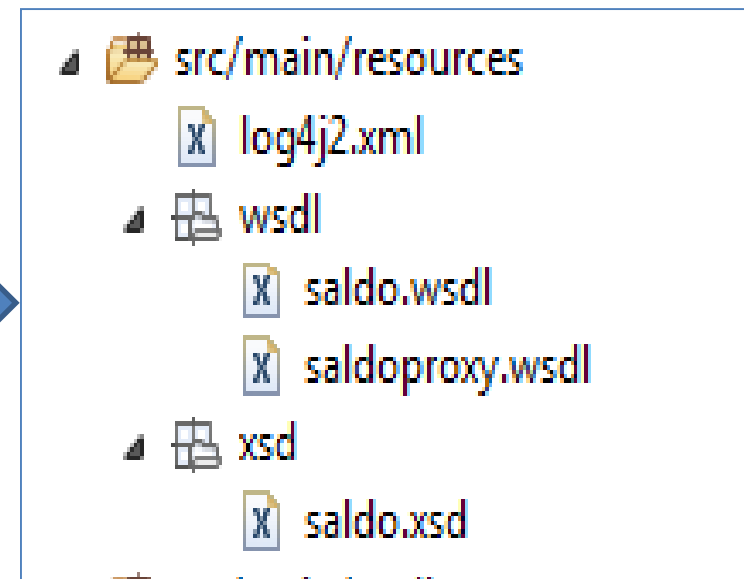
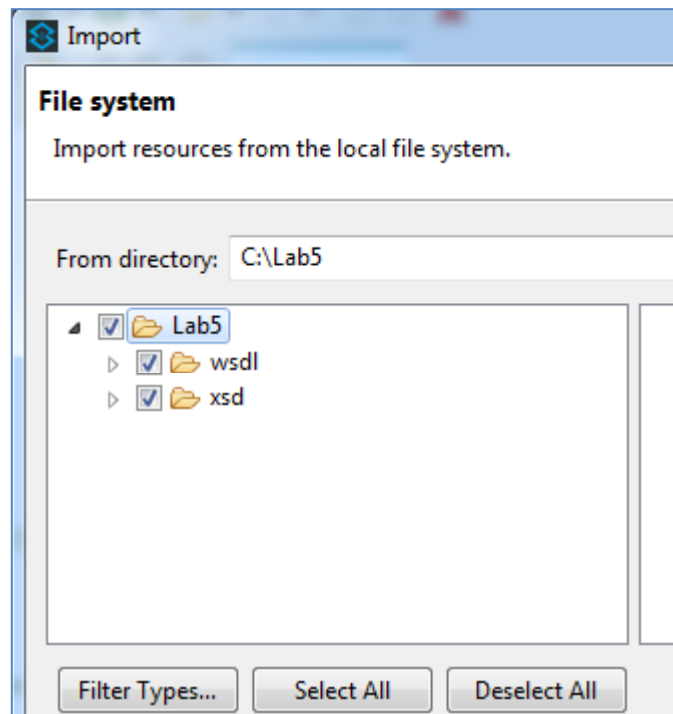
Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

5) O novo serviço de proxy, utilizará como base o mesmo wsdl gerado pelo serviço “saldo”, contudo, ele também utilizará um novo arquivo de xsd para utilizar como metadado. Como esse metadado é diferente do metadado original pertencente ao serviço “saldo”, necessitará das transformações. Primeiramente importe o wsdl e xsd que se encontra na pasta Lab5 conforme a figura abaixo:



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

6) Os arquivos serão importados para o pacote: “src/main/resources”.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

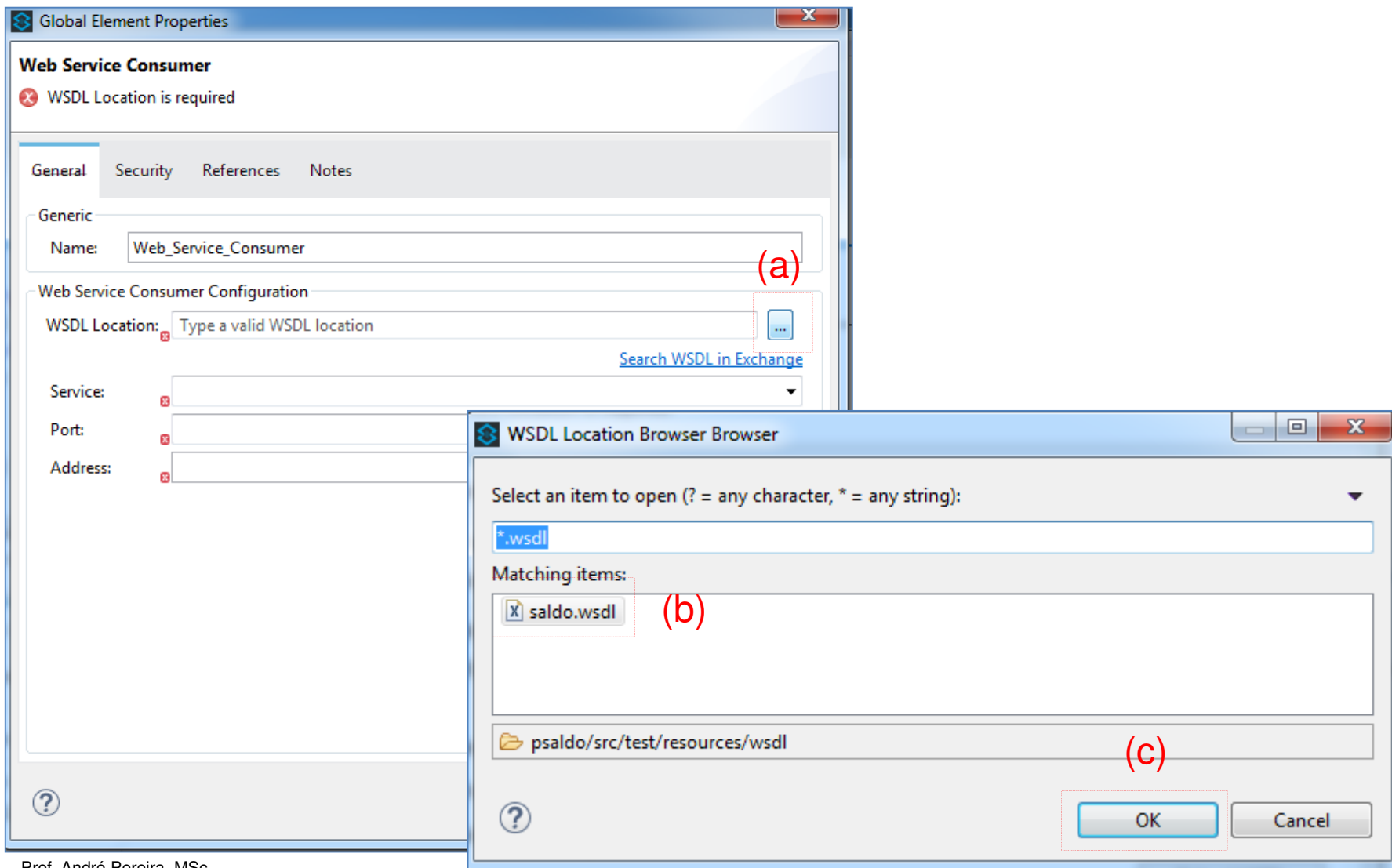
7) Agora começaremos a configurar cada componente. Clique em “Web Service Consumer” e no clique no sinal de “+ verde” (a).

The screenshot displays the ProxyFlow configuration interface. The main canvas shows a sequence of components: HTTP, CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, and Transform Message. The 'Web Service Consumer' component is highlighted with a red box. Below the canvas, the 'Web Service Consumer' configuration panel is open, showing the 'Display Name' field with the value 'Web Service Consumer' and a red '(a)' next to it. The panel also shows 'Basic Settings' with 'Connector Configuration' and 'Operation' fields, and a 'Reload WSDL' button.

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

8) Na tela “Global Element Properties”, clique em (a) para inserir o wsdl importado. Selecione “saldo.wsdl” (b) e depois clique em “OK”. (c)



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”



9) Os demais campos serão automaticamente preenchidos. Aguarde um pouco e depois clique em “OK”.

The screenshot shows the 'Global Element Properties' dialog box for a 'Web Service Consumer'. The 'General' tab is selected. The 'Generic' section has a 'Name' field with the value 'Web_Service_Consumer'. The 'Web Service Consumer Configuration' section contains the following fields: 'WSDL Location' with 'wsdl/saldo.wsdl', 'Service' with 'IContaBancoService', 'Port' with 'IContaBancoPort', and 'Address' with 'http://localhost:8081/saldo'. There is a 'Reload WSDL' button and a 'Search WSDL in Exchange' link. The 'OK' and 'Cancel' buttons are at the bottom right.

Field	Value
Name	Web_Service_Consumer
WSDL Location	wsdl/saldo.wsdl
Service	IContaBancoService
Port	IContaBancoPort
Address	http://localhost:8081/saldo

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

10) Selecione a operação “getSaldo” (a) e depois salve.

The screenshot displays the Apache Camel IDE interface. At the top, a message flow diagram titled "ProxyFlow" shows a sequence of components: HTTP, CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, and Transform Message. The "Web Service Consumer" component is highlighted with a red box. Below the flow diagram, the "Error handling" section is visible. The bottom panel shows the configuration for the "Web Service Consumer" component. The "Display Name" is "Web Service Consumer". Under "Basic Settings", the "Connector Configuration" is set to "Web_Service_Consumer". The "Operation" is set to "getSaldo", which is marked with a red letter "(a)". A "Reload WSDL" button is located at the bottom right of the configuration panel.

ProxyFlow

HTTP → CXF → Logger → Transform Message → Logger → Web Service Consumer → Logger → Transform Message

Error handling

Message Flow Global Elements Configuration XML

Web Service Consumer x Problems Console

General

Notes

Display Name: Web Service Consumer

Basic Settings

Connector Configuration: Web_Service_Consumer

Operation: (a) getSaldo

Reload WSDL

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

11) Selecione o componente “CXF” depois selecione “Proxy service” (a).

The screenshot displays the Apache Camel IDE interface. At the top, a message flow diagram titled "ProxyFlow" is shown. It consists of a sequence of components: HTTP, CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, and Transform Message. The CXF component is highlighted with a red dashed box. Below the diagram, the "Message Flow" tab is selected, showing the "Global Elements" and "Configuration XML" sections. In the bottom left, the "CXF" component is selected in the "Problems" tab. The "General" configuration panel for the CXF component is open, showing the "Display Name" as "CXF". Under the "Generic" section, the "Operation" dropdown menu is open, displaying a list of options: "Proxy service", "Simple service", "JAX-WS client", "Proxy client", and "Simple client". The "Proxy service" option is highlighted in blue. A red dashed box is drawn around the "Operation" dropdown and its list. A red label "(a)" is placed next to the "Proxy service" option.

ProxyFlow

HTTP → CXF → Logger → Transform Message → Logger → Web Service Consumer → Logger → Transform Message

Message Flow Global Elements Configuration XML

CXF Problems Console

General

Interceptors

Security

Advanced

Notes

Metadata

Display Name: CXF

Generic

Config Reference:

Operation:

Please select an operation

- Proxy service
- Simple service
- JAX-WS client
- Proxy client
- Simple client

(a)

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

12) Abaixo, preencha as informações:

- Namespace: `http://psaldobanco/`
- Service: `IContaBancoService`

De acordo com a figura abaixo.

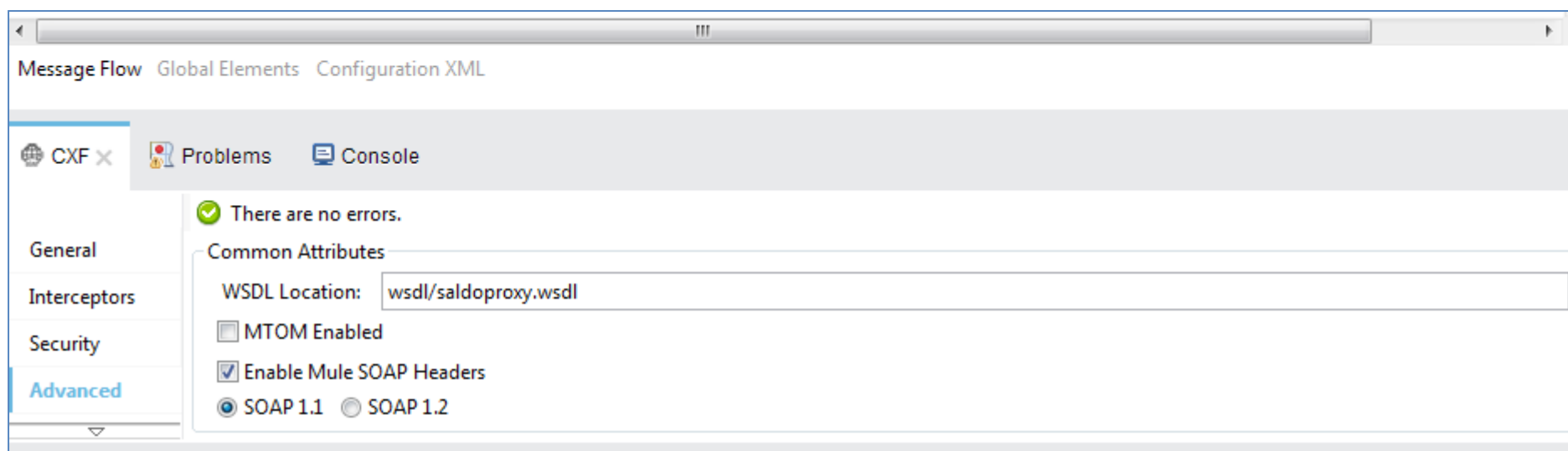
The screenshot shows the CXF IDE configuration window with the following details:

- General Tab:** Contains fields for Binding ID, Port, Namespace, Service, and Service Class. The Namespace is set to `http://psaldobanco/` and the Service is set to `IContaBancoService`. A red box highlights these two fields.
- Validation:** A checkbox for "Validation Enabled" is present.
- Buttons:** A green plus icon and a magnifying glass icon are visible, along with the text "Generate from WSDL".
- Proxy-Service:** A section at the bottom of the configuration window.

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”



13) Depois vá até a aba “Advanced” e insira no campo “WSDL Location” o valor: “wsdl/saldoproxy.wsdl” conforme figura abaixo. Não esqueça de salvar.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

14) Selecione o componente HTTP. Depois adicione um novo “Connector Configuration” clicando no sinal de “+ verde” (a).

The screenshot displays the Apache Camel IDE interface. At the top, a message flow diagram titled "ProxyFlow" is shown. The flow consists of the following components in sequence: HTTP (highlighted with a red box), CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, and Transform Message. Below the main flow, an "Error handling" section is visible. At the bottom, the "HTTP" component's configuration panel is open. The "General" tab is selected, showing the "Display Name" as "HTTP". Under "General Settings", the "Connector Configuration" dropdown is set to "HTTP_Listener_Configuration". A red box highlights the green "+" icon and the "Add" button next to this dropdown. The "Basic Settings" section shows the "Path" as "/" and the "Allowed Methods" field is empty.

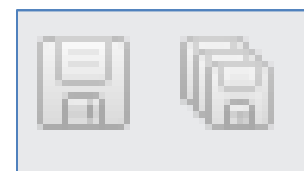
Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

15) Em “Global Element Properties”, preencha os seguintes valores:

- Name: HTTP_Listener_Configuration_Proxy
- Host: localhost
- Port: 8085
- Base Path: saldo/proxy

NÃO ESQUEÇA DE SALVAR!!!

De acordo com a figura abaixo.

The screenshot shows a 'Global Element Properties' dialog box with the title 'HTTP Listener Configuration' and the subtitle 'Create reusable HTTP listener'. It has three tabs: 'General', 'TLS/SSL', and 'Notes'. The 'General' tab is active. Under the 'Generic' section, the 'Name' field is filled with 'HTTP_Listener_Configuration_Proxy'. Under the 'URL Configuration' section, the 'Protocol' is set to 'HTTP (Default)' (selected with a radio button), 'Host' is 'localhost', 'Port' is '8085', and 'Base Path' is 'saldo/proxy'. At the bottom, there are 'OK' and 'Cancel' buttons. A red dashed box highlights the 'Name', 'Protocol', 'Host', 'Port', and 'Base Path' fields.

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

16) O Próximo passo é a configuração dos componentes de transformação. Observem abaixo que antes e depois da invocação do Webservice “saldo”, acontece uma transformação. Repare que na ida os tipos são diferentes e na volta, existem campos à mais.



Primeira transformação

```
<xs:complexType name="saldoRequest">
  <xs:sequence>
    <xs:element minOccurs="0" name="agencia" type="xs:string"/>
    <xs:element minOccurs="0" name="conta" type="xs:string"/>
    <xs:element minOccurs="0" name="name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="getSaldoResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="return" type="tns:saldoResponse"/>
  </xs:sequence>
</xs:complexType>
```



```
<xs:complexType name="saldoRequest">
  <xs:sequence>
    <xs:element minOccurs="0" name="agencia" type="xs:integer"/>
    <xs:element minOccurs="0" name="conta" type="xs:integer"/>
    <xs:element minOccurs="0" name="name" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="getSaldoResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="return" type="tns:saldoResponse"/>
  </xs:sequence>
</xs:complexType>
```

Segunda transformação

```
<xs:complexType name="saldoResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="agencia" type="xs:string"/>
    <xs:element minOccurs="0" name="conta" type="xs:string"/>
    <xs:element minOccurs="0" name="date" type="xs:dateTime"/>
    <xs:element minOccurs="0" name="name" type="xs:string"/>
    <xs:element name="saldo" type="xs:double"/>
  </xs:sequence>
</xs:complexType>
```



```
<xs:complexType name="saldoResponse">
  <xs:sequence>
    <xs:element minOccurs="0" name="agenciaConta" type="xs:string"/>
    <xs:element minOccurs="0" name="nomeAgencia" type="xs:string"/>
    <xs:element minOccurs="0" name="date" type="xs:dateTime"/>
    <xs:element minOccurs="0" name="name" type="xs:string"/>
    <xs:element name="saldo" type="xs:double"/>
    <xs:element name="currency" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

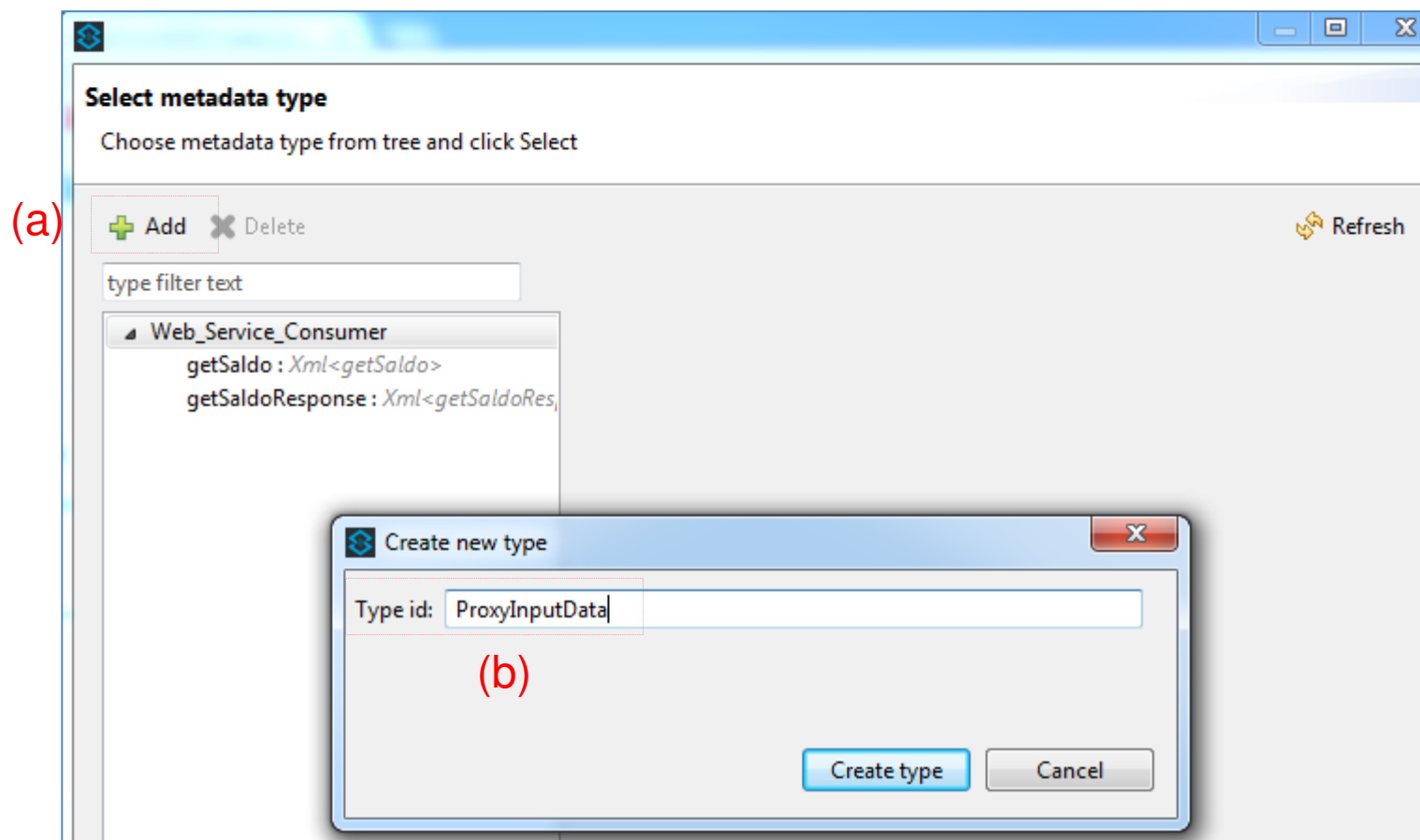
17) Clique no primeiro componente “Transform Message”. Vamos definir o metadado de entrada (que vem do proxy para a chamada do webservice). Clique em (a).

The screenshot displays the Apache Camel IDE interface. At the top, a message flow diagram titled "ProxyFlow" shows a sequence of components: HTTP, CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, and Transform Message. The first "Transform Message" component is highlighted with a red box. Below the flow diagram, the "Error handling" section is visible. The bottom panel shows the configuration for the selected "Transform Message" component. On the left, the "Input" section has "Payload : Unknown" and a red box labeled (a) pointing to the "Define metadata" link. The "Output" section shows the payload structure: `Xml<getSaldo>` with an argument `arg0 : Xml<getSaldo>` containing `agencia : String` and `conta : String`. The "Output Payload" tab on the right shows the XML structure:

```
1 <?xml version='1.0'>
2 <output application/java>
3 <---
4 {
5 }
```


Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

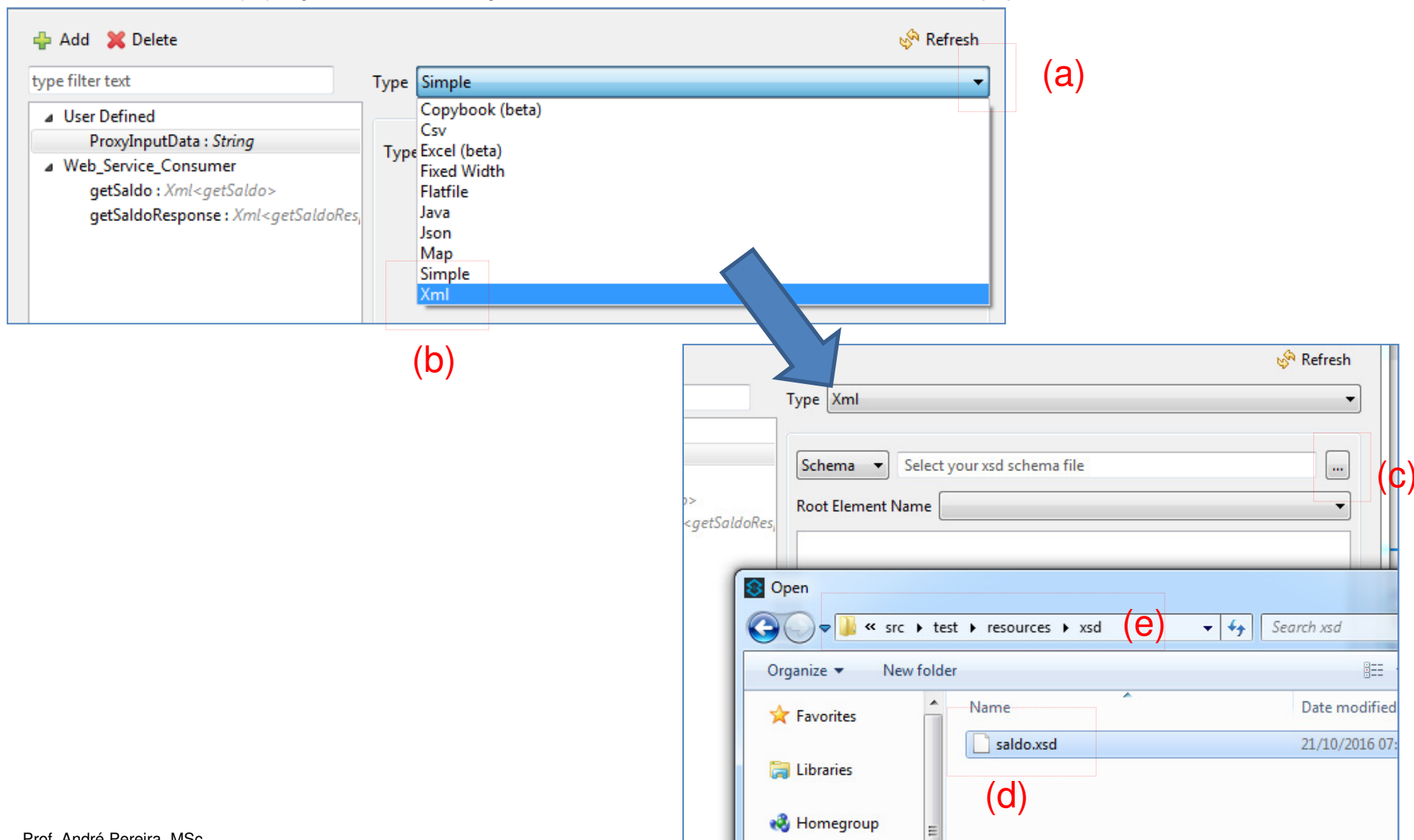
18) Clique no “+ verde” (a) para adicionar um novo metadado. Depois coloque o nome de “ProxyInputData” (b).



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

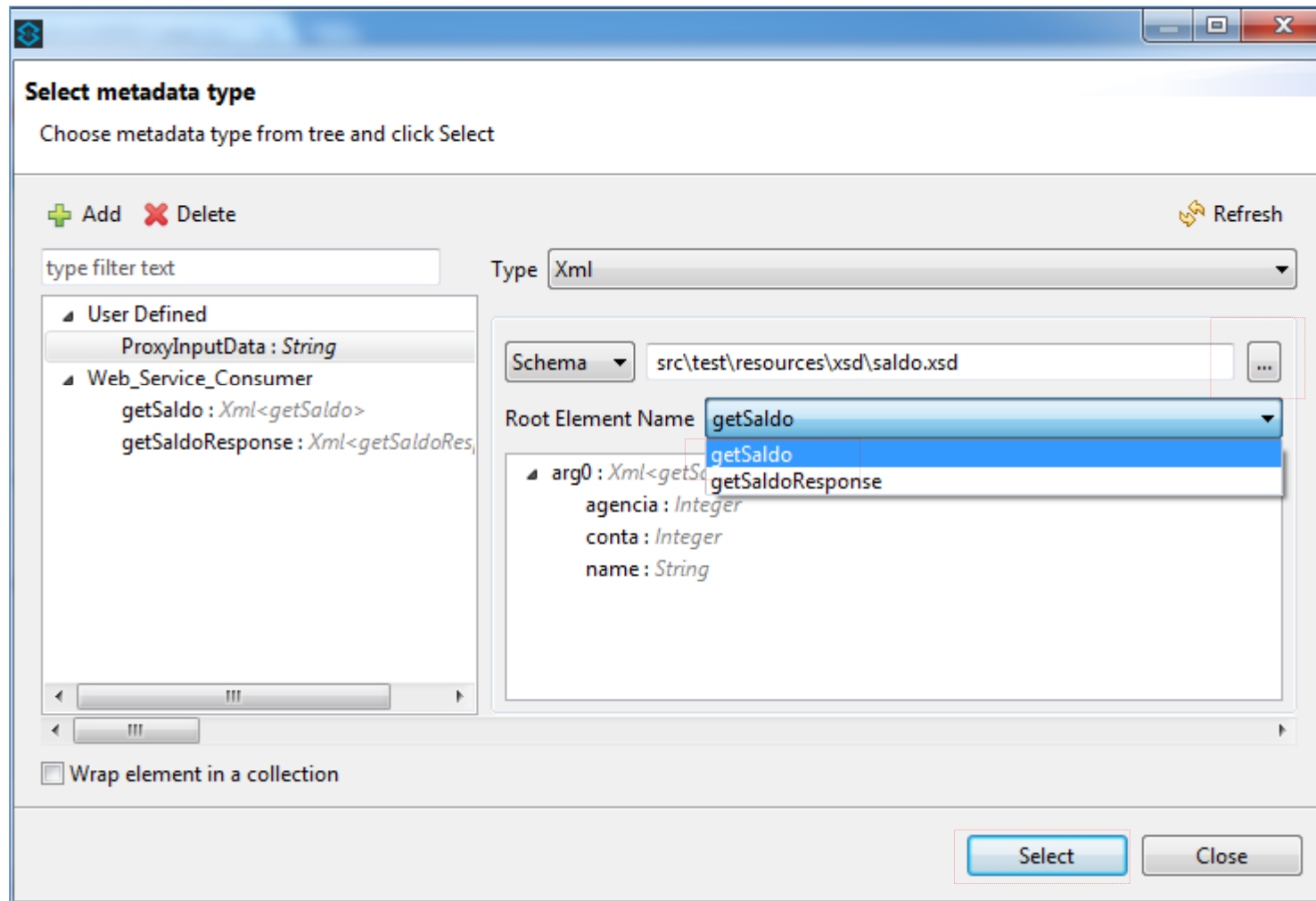
19) Clique em (a) e escolha “Xml” (b). Depois em (c) e escolha o arquivo “saldo.xsd” (d) que está na pasta “src/test/resources/xsd” (e).



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

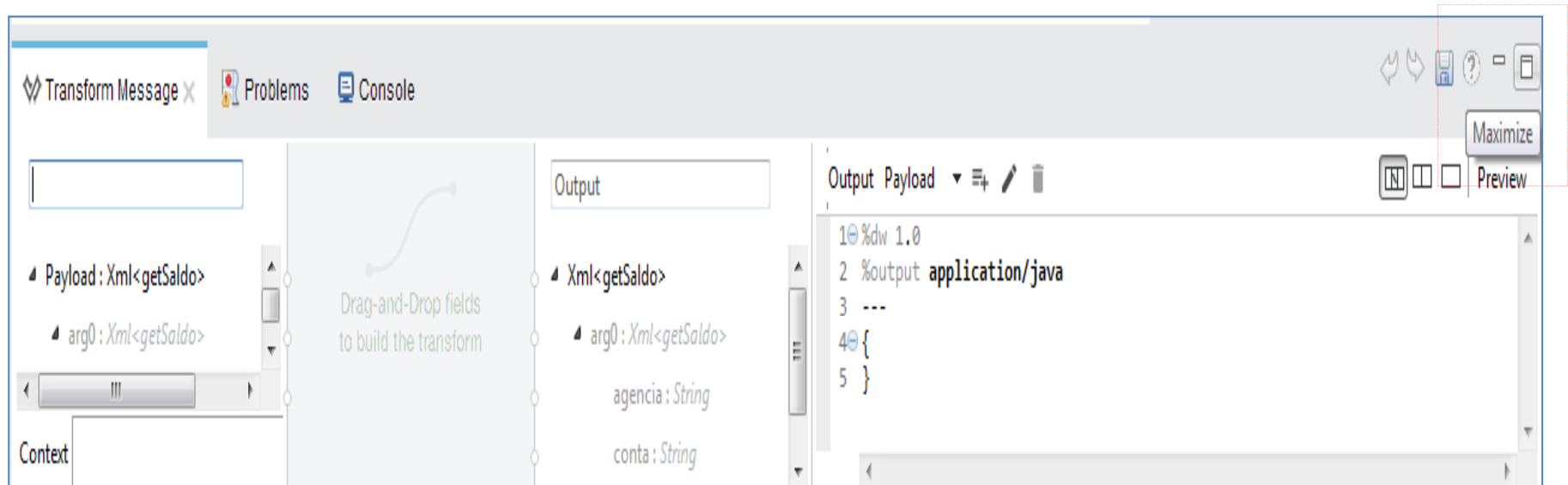
20) Clique em (a) e depois selecione “getSaldo”. Depois clique no botão “Select”.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

21) Maximize a tela. Clique em (a).



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

22) Clique em cada campo do lado (a) e arraste-o e solte-o em cima do campo do lado (b). Por exemplo, clique em “**agencia: Integer**” arraste-o até “**agencia: String**” e solte-o. Faça para os demais. Reparem que o Mule já faz a conversão automática (c). Depois restaure a tela (d) e salve o seu trabalho.

The screenshot displays the MuleSoft IDE's 'Transform Message' window. The interface is divided into three main sections: 'Input', a central transformation canvas, and 'Output'.
(a) Input: The 'Payload' is an XML document with the root element 'getSaldo'. It contains an 'arg0' element with three child elements: 'agencia' (type Integer), 'conta' (type Integer), and 'name' (type String).
(b) Output: The 'Payload' is the same XML document, but the types have been changed: 'agencia' is now String, 'conta' is now String, and 'name' remains String.
(c) Transformation: The central canvas shows a visual mapping of the input fields to the output fields. Lines connect 'agencia: Integer' to 'agencia: String', 'conta: Integer' to 'conta: String', and 'name: String' to 'name: String'. Below the canvas, the 'Output Payload' is shown in a code editor with the following XML structure:

```
1 <?xml version='1.0'>
2 <output application/xml>
3 <?namespace ns0 http://psaldobanco/>
4 <---
5 {
6   ns0#getSaldo: {
7     arg0: {
8       agencia: payload.ns0#getSaldo.arg0.agencia as :string,
9       conta: payload.ns0#getSaldo.arg0.conta as :string,
10      name: payload.ns0#getSaldo.arg0.name
11     }
12   }
13 }
```


(d) Actions: On the right side of the IDE, there is a 'Restore' button and a 'Preview' button, both of which are highlighted with red boxes.

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

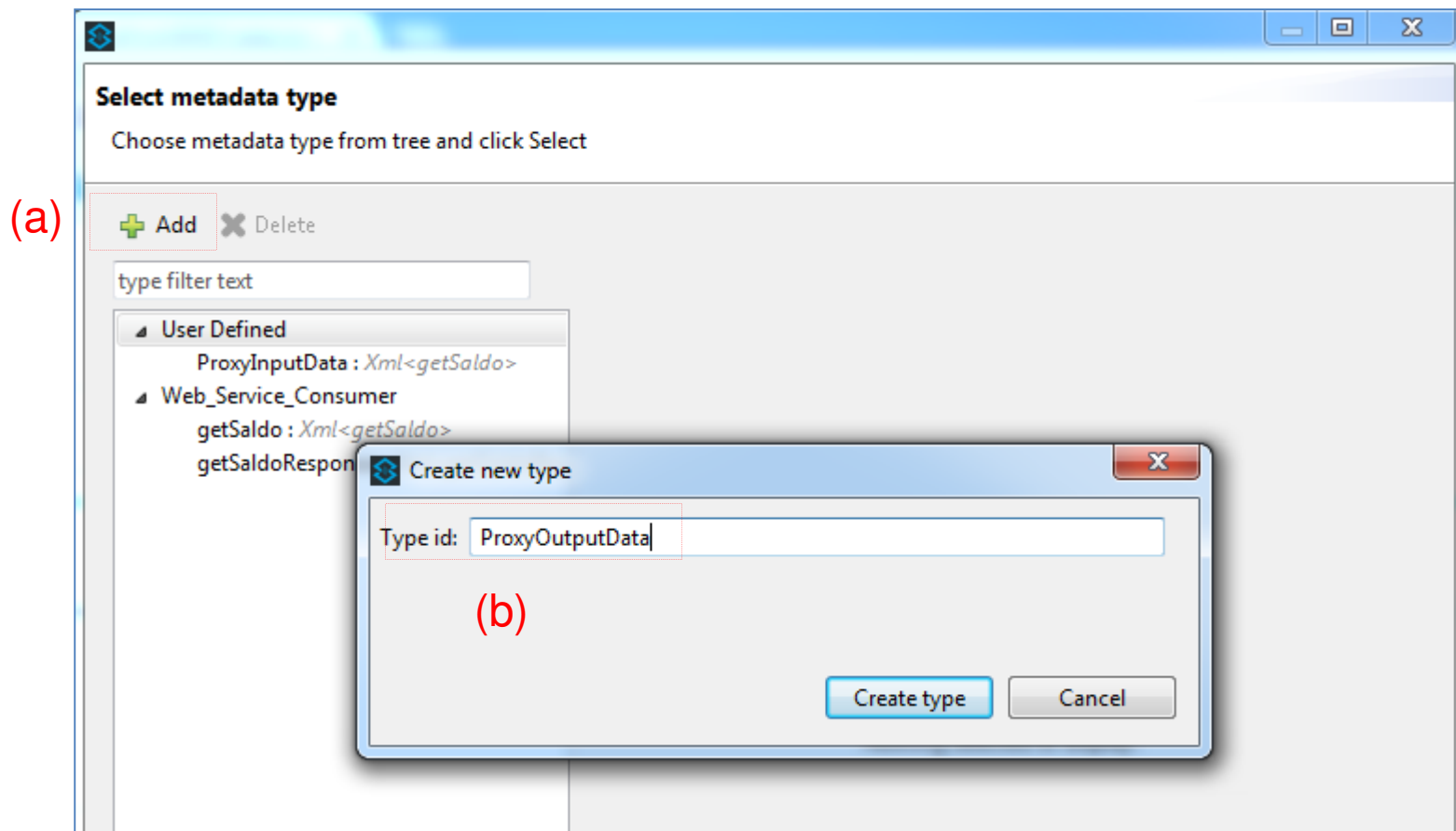
23) Clique no segundo componente “Transform Message”. Vamos definir o metadado de saída (que vem do webservice para o proxy). Clique em (a).

The screenshot displays the Apache Camel IDE interface. At the top, a message flow diagram shows a sequence of components: CXF, Logger, Transform Message, Logger, Web Service Consumer, Logger, Transform Message, and Logger. The second Transform Message component is highlighted with a red box. Below the diagram, the 'Message Flow' tab is active, showing the configuration for the selected Transform Message component. The 'Input' section shows the payload: `Xml<getSaldoResponse>`. The 'Output' section, labeled with a red (a), shows an 'Unknown' status with a 'Define metadata' link. The 'Output Payload' section on the right shows the following XML structure:

```
1 %dw 1.0
2 %output application/java
3 ---
4 {
5 }
```

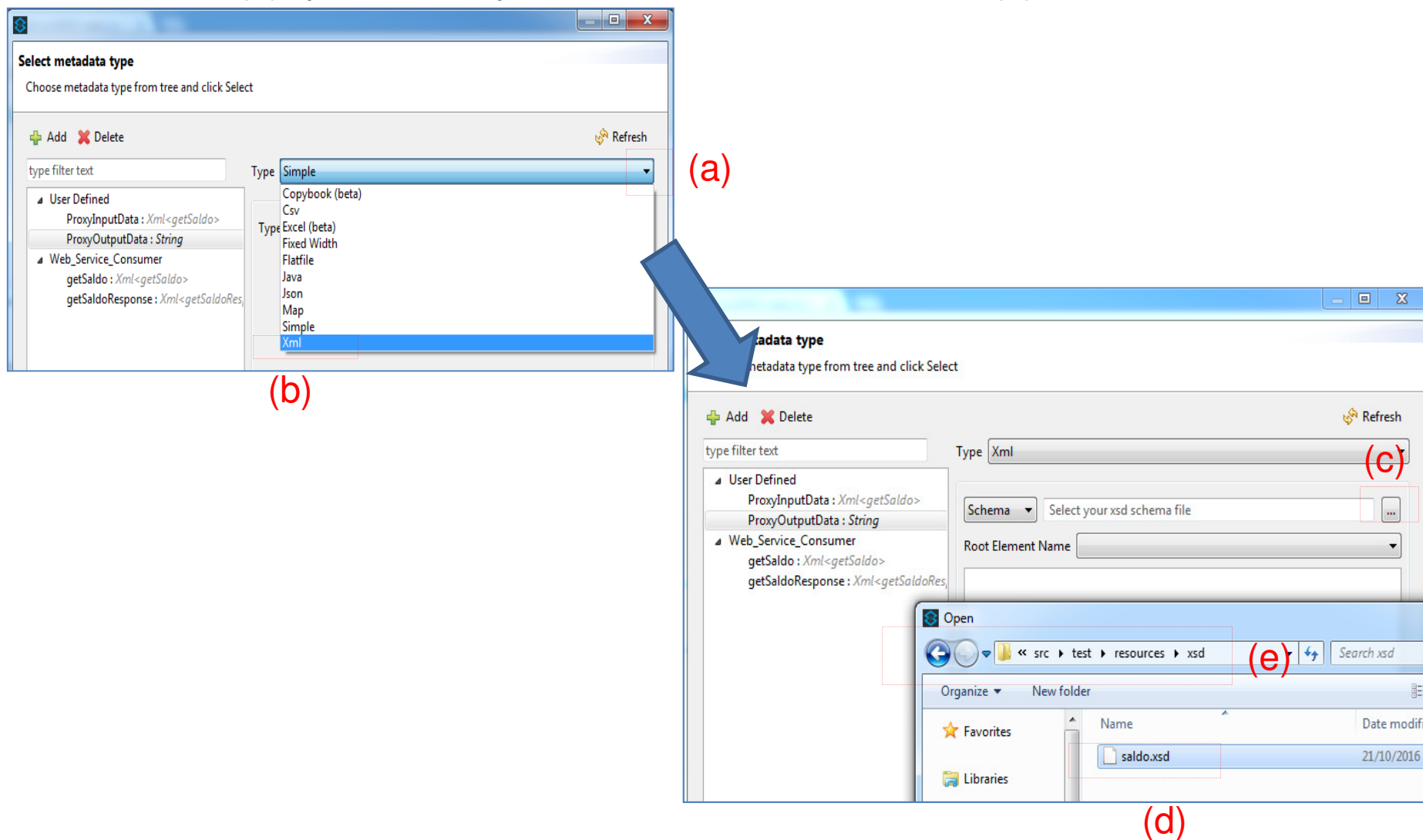
Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

24) Clique no “+ verde” (a) para adicionar um novo metadado. Depois coloque o nome de “ProxyOutputData” (b).



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

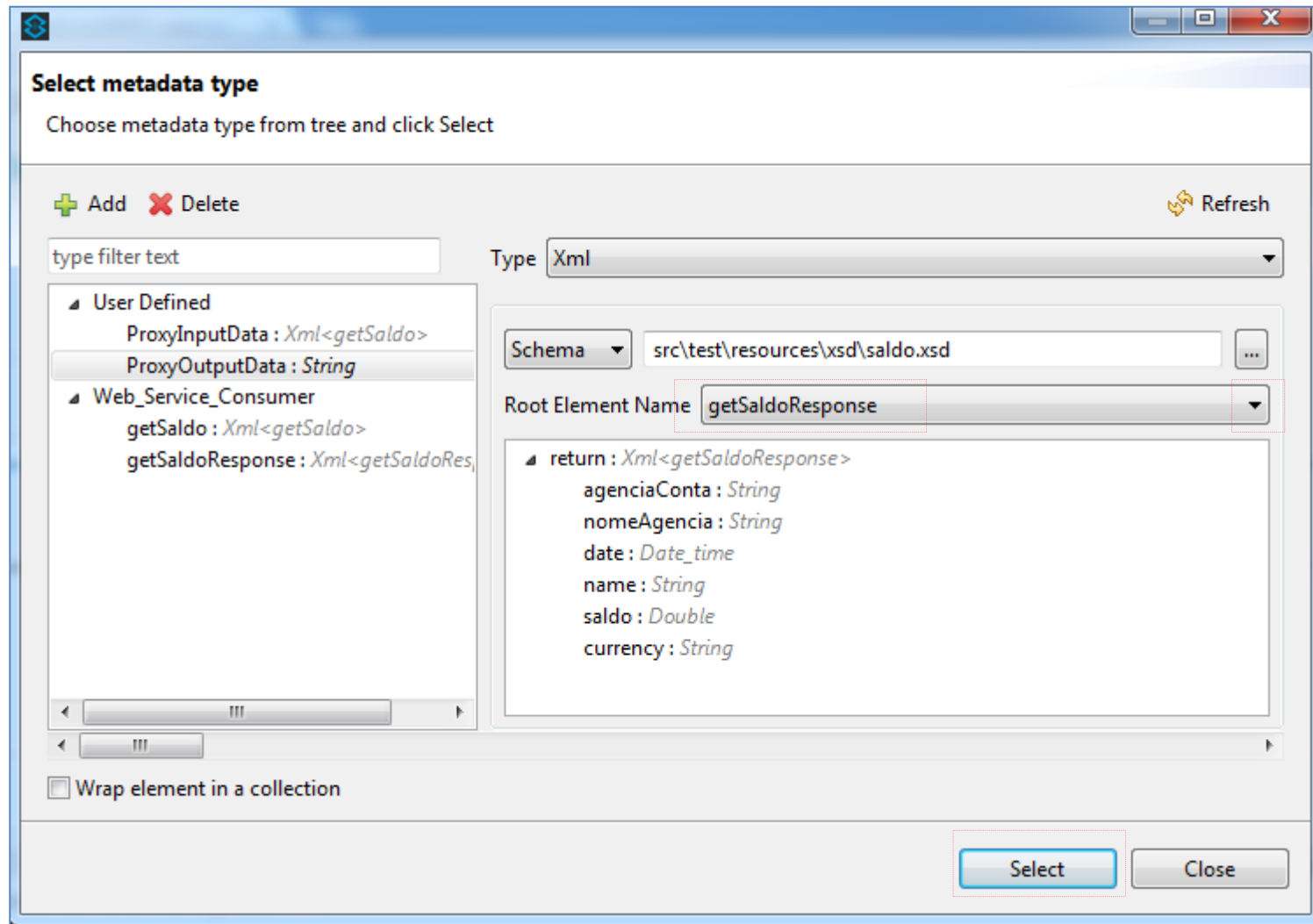
25) Clique em (a) e escolha “Xml” (b). Depois em (c) e escolha o arquivo “saldo.xsd” (d) que está na pasta “src/test/resources/xsd” (e).



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

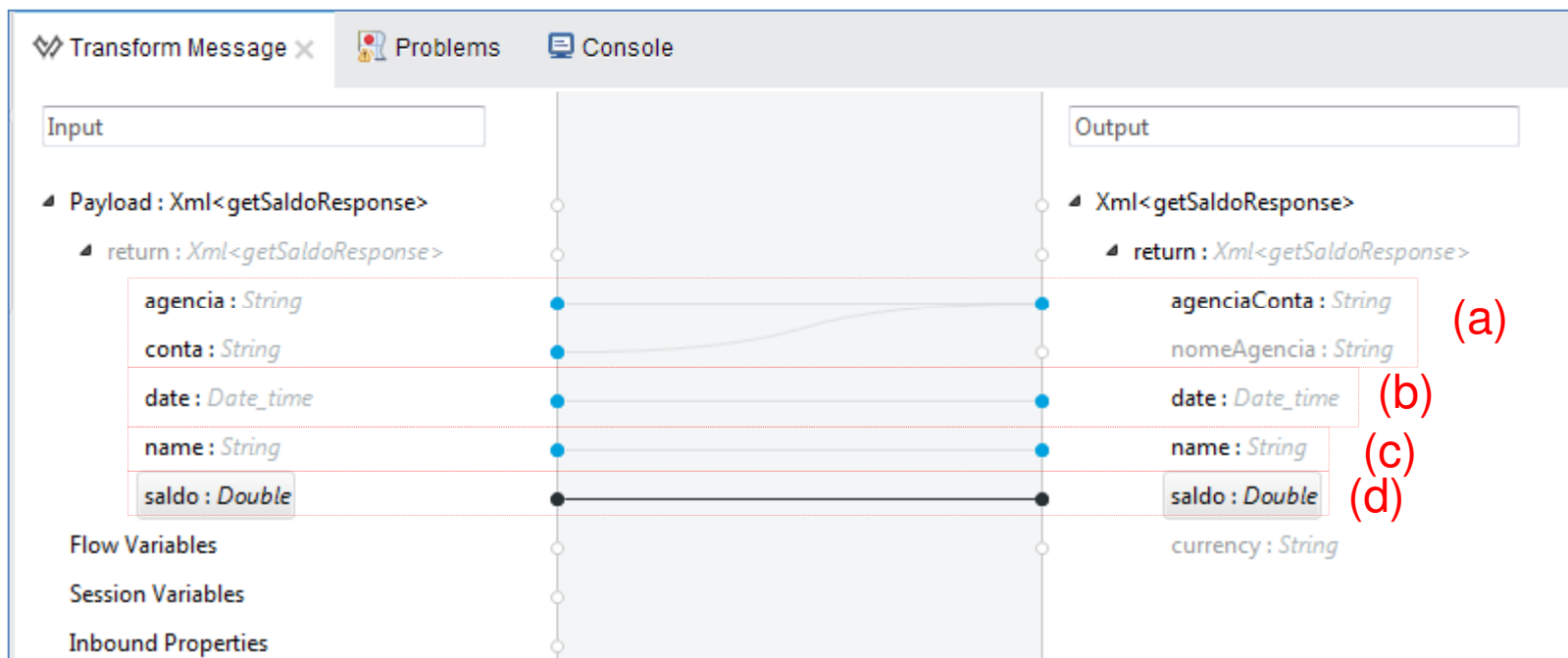
FIAP

26) Clique em (a) e depois selecione “getSaldoResponse”. Depois clique no botão “Select”.



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

27) Maximize a tela e comece a trabalhar. Primeiramente associe os seguintes campos: “agencia” e “conta” com “agenciaConta” (a), “date” com “date” (b), “name” com “name” (c) e “saldo” com “saldo” (d) conforme figura abaixo.



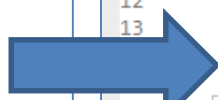
Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”



28) Depois faça a alteração:

- `agenciaConta: payload.ns0#getSaldoResponse.return.agencia ++ '/' ++ payload.ns0#getSaldoResponse.return.conta,`
- `nomeAgencia: 'Agencia Rua do Sol',`
- `date: payload.ns0#getSaldoResponse.return.date,`
- `name: payload.ns0#getSaldoResponse.return.name,`
- `saldo: payload.ns0#getSaldoResponse.return.saldo,`
- `currency: 'Reais'`

```
Output Payload
1 %dw 1.0
2 %output application/xml
3 %namespace ns0 http://psaldobanco/
4 ---
5 {
6   ns0#getSaldoResponse: {
7     return: {
8       agenciaConta: payload.ns0#getSaldoResponse.return.agencia
9 ++ payload.ns0#getSaldoResponse.return.conta,
10       date: payload.ns0#getSaldoResponse.return.date,
11       name: payload.ns0#getSaldoResponse.return.name,
12       saldo: payload.ns0#getSaldoResponse.return.saldo
13     }
14   }
15 }
```

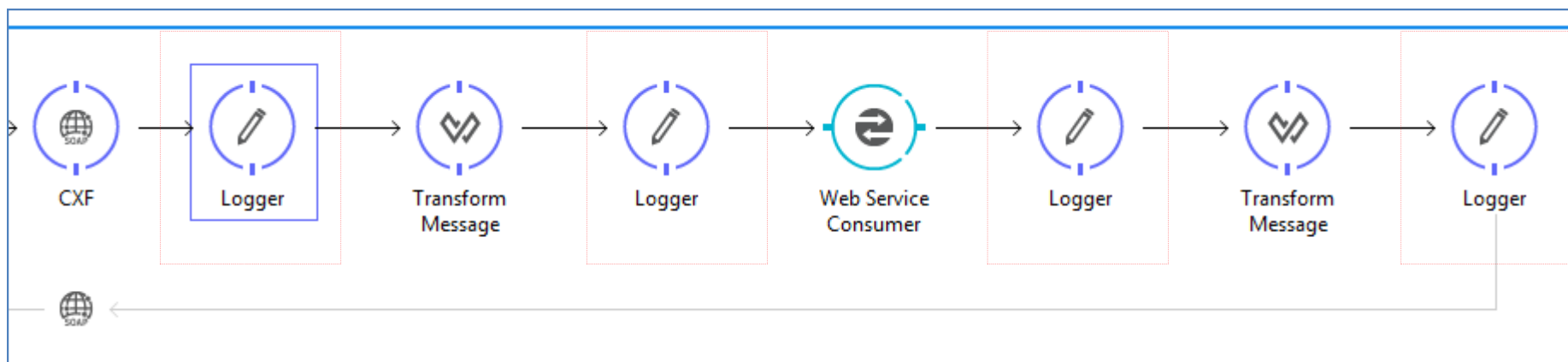


```
Output Payload
1 %dw 1.0
2 %output application/xml
3 %namespace ns0 http://psaldobanco/
4 ---
5 {
6   ns0#getSaldoResponse: {
7     return: {
8       agenciaConta: payload.ns0#getSaldoResponse.return.agencia
9 ++ '/' ++ payload.ns0#getSaldoResponse.return.conta,
10       nomeAgencia: 'Agencia Rua do Sol',
11       date: payload.ns0#getSaldoResponse.return.date,
12       name: payload.ns0#getSaldoResponse.return.name,
13       saldo: payload.ns0#getSaldoResponse.return.saldo,
14       currency: 'Reais'
15     }
16   }
17 }
```

Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

29) Restaure a tela e salve o seu trabalho. Agora vamos trabalhar com os componentes “Logger”.

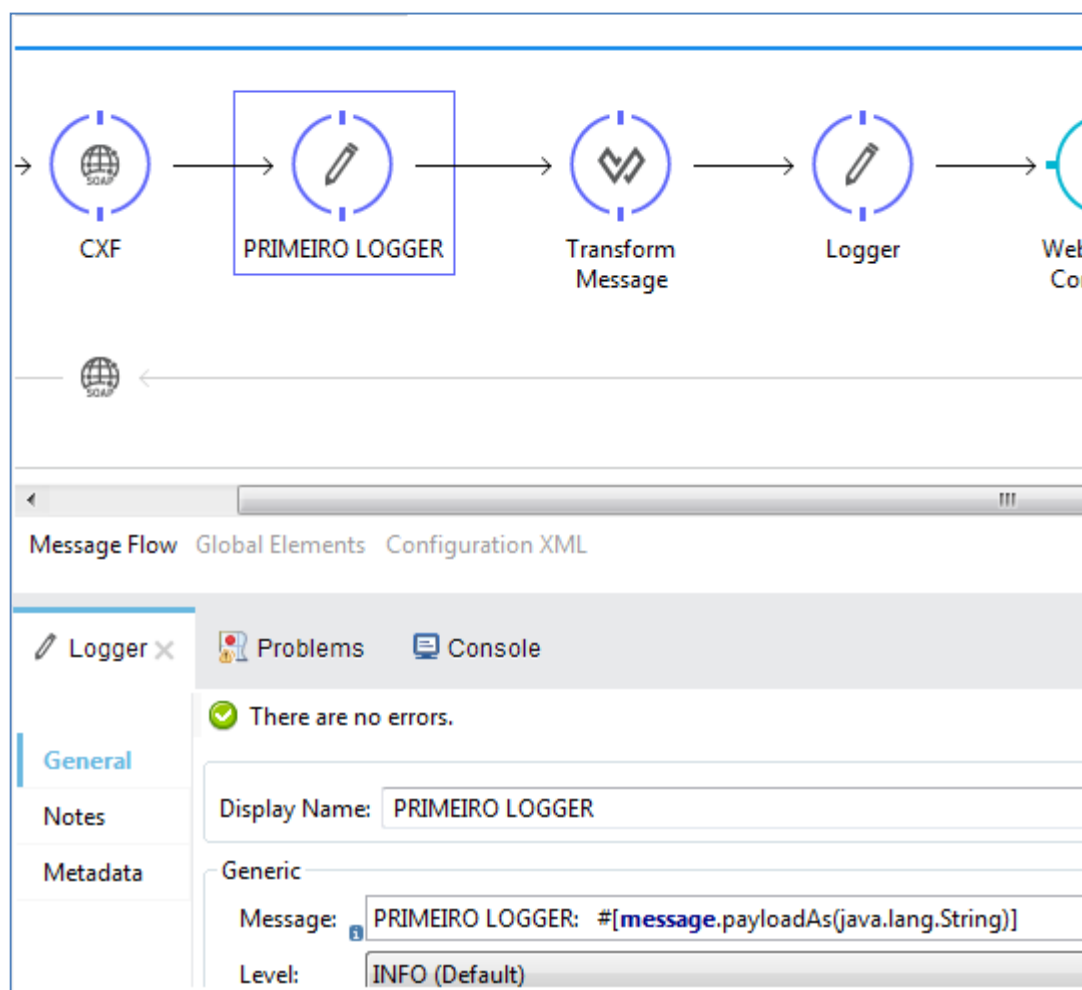


Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

30) Clique no primeiro componente “Logger” e insira as seguintes informações:

Display Name: **PRIMEIRO LOGGER**

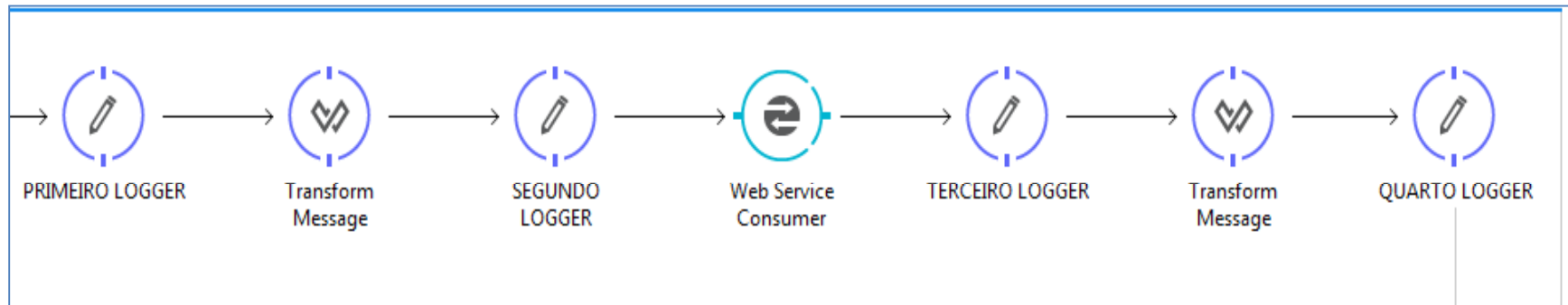
Message: **PRIMEIRO LOGGER: #[message.payloadAs(java.lang.String)]**



Criação de um serviço “Proxy” com transformação de dados para o serviço de “Saldo”

FIAP

31) Repita o procedimento para os demais. Salve seu trabalho. Está pronto seu serviço de “Proxy”.

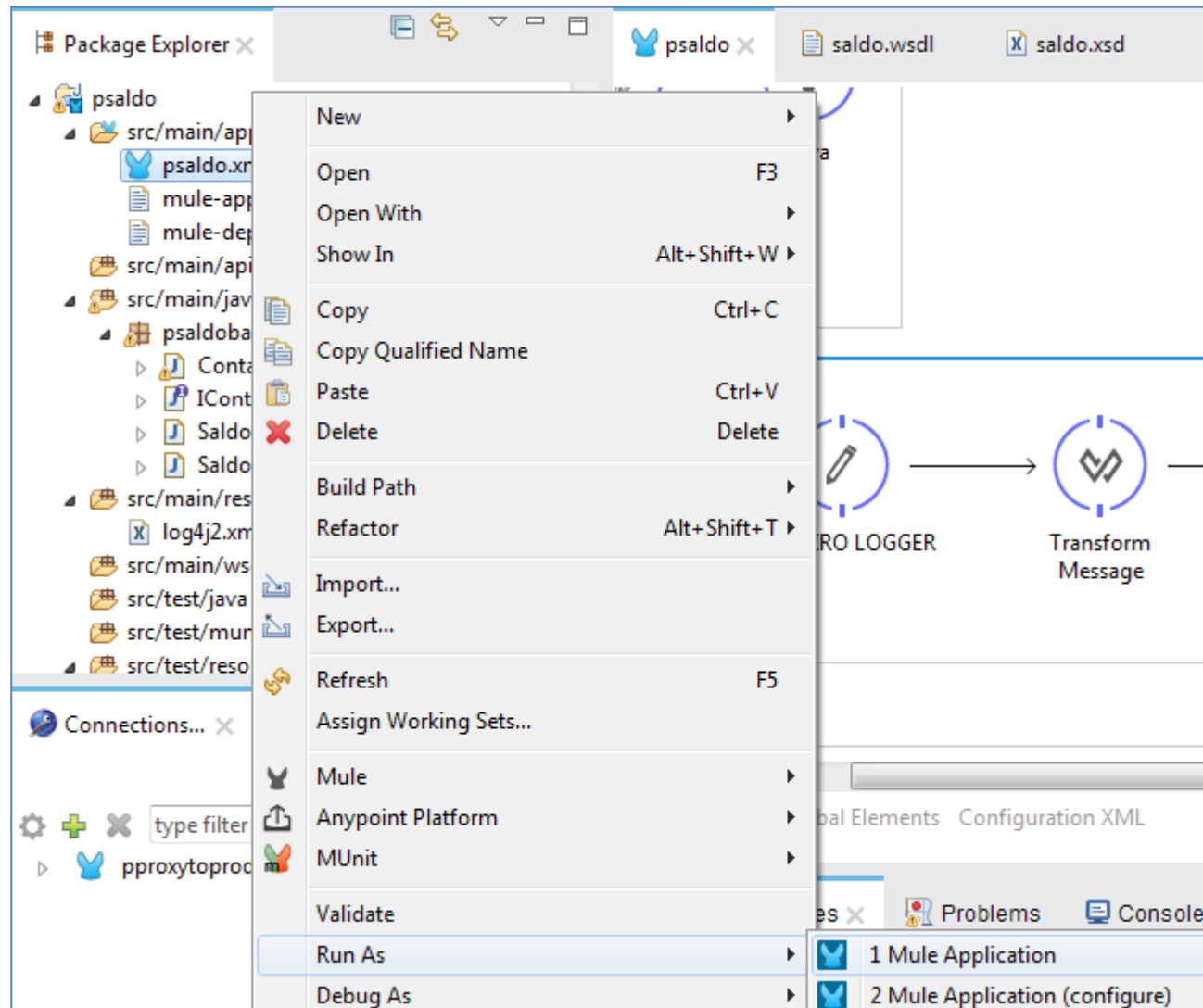


Testando o serviço “Proxy” no Mule Runtime e SoapUI

Testando o serviço “Proxy” no Mule Runtime e SoapUI



32) Clique com o botão direito em “psaldo.xml” e depois “Run As”->”Mule Application”.



Testando o serviço “Proxy” no Mule Runtime e SoapUI



33) Na tela “Console”, abaixo, o “runtime” do Mule executará e sua aplicação será implantada “Deployed”.

```
psaldo [Mule Applications] C:\Program Files (x86)\Java\jdk1.7.0_79\bin\javaw.exe (21 de out de 2016 09:28:35)
*****
* default                                * DEPLOYED                                *
*****

*****
* - - + APPLICATION + - -                * - - + DOMAIN + - -                * - - + STATUS + - - *
*****
* psaldo                                * default                                * DEPLOYED                                *
*****
```

Testando o serviço “Proxy” no Mule Runtime e SoapUI



34) Abra uma janela do “Browser” e coloque o endereço: “<http://localhost:8087/saldo/proxy?wsdl>”.

The screenshot shows a web browser window with the address bar displaying `localhost:8087/saldo/proxy?wsdl`. The page content displays the XML WSDL for the service. The XML is as follows:

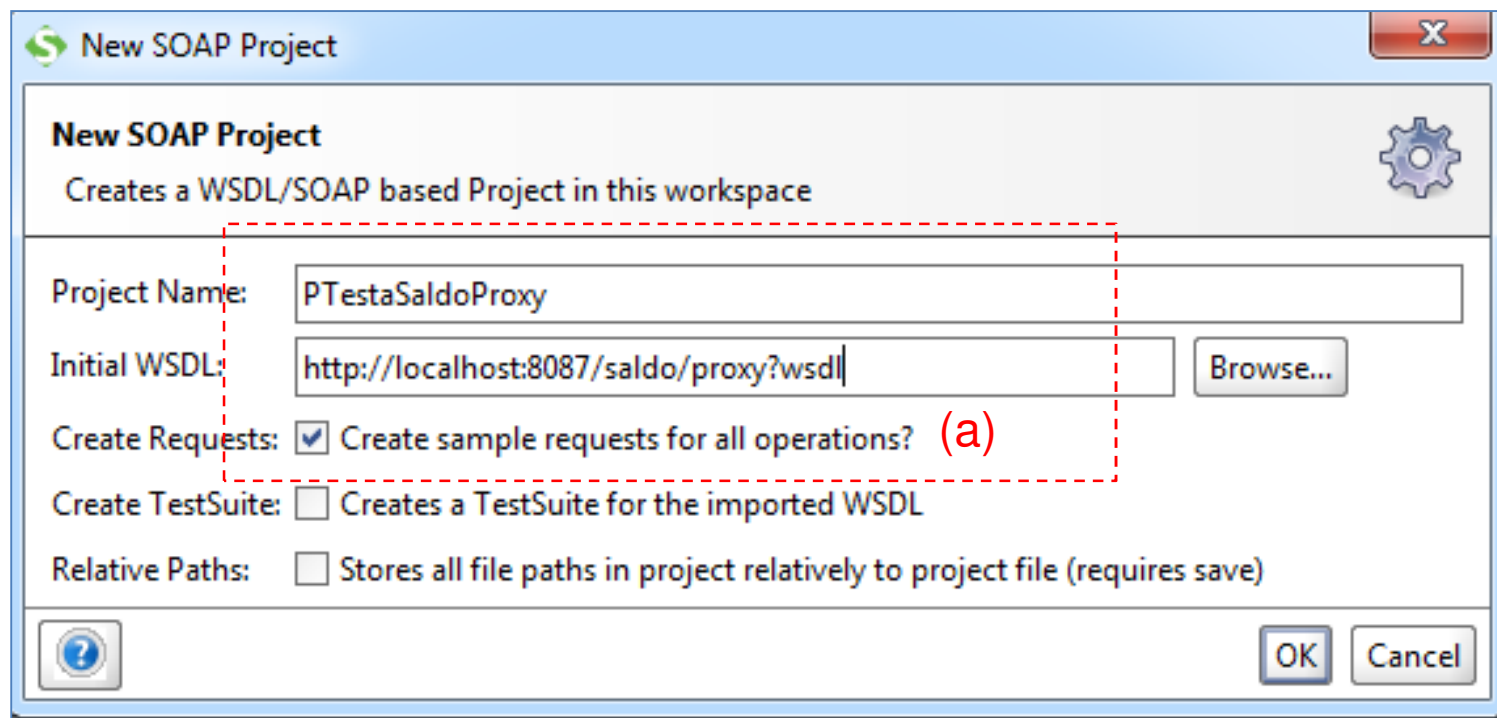
```
<?xml version='1.0' encoding='UTF-8'?>
<wsdl:definitions xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:ns1="http://schemas.xmlsoap.org/soap/http" name="IContaBancoService" targetNamespace="http://psaldobanco/">
  <wsdl:types>
    <xsd:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:tns="http://psaldobanco/" elementFormDefault="qualified">
      <xs:element name="getSaldo" type="tns:getSaldo"/>
      <xs:element name="getSaldoResponse" type="tns:getSaldoResponse"/>
      <xs:complexType name="getSaldo">
        <xs:sequence>
          <xs:element minOccurs="0" name="arg0" type="tns:saldoRequest"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="saldoRequest">
        <xs:sequence>
          <xs:element minOccurs="0" name="agencia" type="xs:integer"/>
          <xs:element minOccurs="0" name="conta" type="xs:integer"/>
          <xs:element minOccurs="0" name="name" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="getSaldoResponse">
        <xs:sequence>
          <xs:element minOccurs="0" name="return" type="tns:saldoResponse"/>
        </xs:sequence>
      </xs:complexType>
      <xs:complexType name="saldoResponse">
        <xs:sequence>
          <xs:element minOccurs="0" name="agencia" type="xs:string"/>
          <xs:element minOccurs="0" name="conta" type="xs:string"/>
          <xs:element minOccurs="0" name="date" type="xs:dateTime"/>
          <xs:element minOccurs="0" name="name" type="xs:string"/>
          <xs:element name="saldo" type="xs:double"/>
        </xs:sequence>
      </xs:complexType>
    </xsd:schema>
  </wsdl:types>

```

Testando o serviço “Proxy” no Mule Runtime e SoapUI



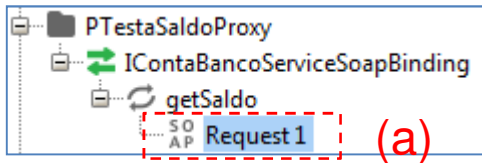
35) Agora abra o SoapUI e crie um novo projeto chamado “PTestaSaldoProxy”. Copie o endereço do wsdl para o campo “Initial WSDL” e marque na opção de criação de requisições automáticas (a) conforme figura abaixo e depois clique “OK”.



Testando o serviço “Proxy” no Mule Runtime e SoapUI



36) Abra o request (a) , preencha os campos (b) e execute o teste (c).



(c)

Request 1

http://localhost:8087/saldo/proxy

XML

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <psal:getSaldo>
      <!--Optional:-->
      <arg0>
        <!--Optional:-->
        <agencia>3120</agencia>
        <!--Optional:-->
        <conta>22340</conta>
        <!--Optional:-->
        <name>Ze Banana</name>
      </arg0>
    </psal:getSaldo>
  </soapenv:Body>
</soapenv:Envelope>
```

(b)

XML

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns0:getSaldoResponse xmlns:ns0="http://psaldobanco/">
      <return>
        <agenciaConta>3120/22340</agenciaConta>
        <nomeAgencia>Agencia Rua do Sol</nomeAgencia>
        <date>2016-10-21T09:33:31.878-02:00</date>
        <name>Ze Banana</name>
        <saldo>5000.0</saldo>
        <currency>Reais</currency>
      </return>
    </ns0:getSaldoResponse>
  </soap:Body>
</soap:Envelope>
```

Testando o serviço “Proxy” no Mule Runtime e SoapUI



37) Volte para o console e veja o que foi “loggado”:

The screenshot shows the Mule IDE console with the following content:

```
psaldo [Mule Applications] C:\Program Files (x86)\Java\jdk1.7.0_79\bin\javaw.exe (21 de out de 2016 09:28:35)

</ns0:getSaldo>
INFO 2016-10-21 09:33:31,933 [[psaldo].HTTP_Listener_Configuration_Proxy.worker.01] org.mule.api.processor.LoggerMessageProcessor: TERCEIRO LOGGER: <?xml version="1.0" encoding="UTF-8">
INFO 2016-10-21 09:33:32,064 [[psaldo].HTTP_Listener_Configuration_Proxy.worker.01] org.mule.api.processor.LoggerMessageProcessor: QUARTO LOGGER: <?xml version='1.0' encoding='UTF-8'>
<ns0:getSaldoResponse xmlns:ns0="http://psaldobanco/">
  <return>
    <agenciaConta>3120/22340</agenciaConta>
    <nomeAgencia>Agencia Rua do Sol</nomeAgencia>
    <date>2016-10-21T09:33:31.878-02:00</date>
    <name>Ze Banana</name>
    <saldo>5000.0</saldo>
    <currency>Reais</currency>
  </return>
</ns0:getSaldoResponse>
```

Two red dashed boxes highlight specific parts of the output:

- The first box highlights the XML response structure, specifically the `<return>` block containing account details like `<agenciaConta>`, `<nomeAgencia>`, `<date>`, `<name>`, `<saldo>`, and `<currency>`.
- The second box highlights the logger messages, specifically the lines for `TERCEIRO LOGGER` and `QUARTO LOGGER`.



Source: http://www.123rf.com/photo_18476654_doodle-style-quitting-time-or-end-of-work-day-illustration-in-vector-format-includes-text-blowing-wh.html

MBA⁺

Copyright © **2016** Prof. André Pereira

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