

MBA⁺

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

MBA⁺

Mule

Laboratório 3

Prof. André Pereira, MSc

Frases
de CINEMA

Filme: Gênio Indomável
Ano: 1997
Quem diz: Will Hunting

**"Muita gente perde tanto quanto você,
mas levanta a cabeça e tenta de novo."**

frasesdecinema.com.br



FIAP

Agenda

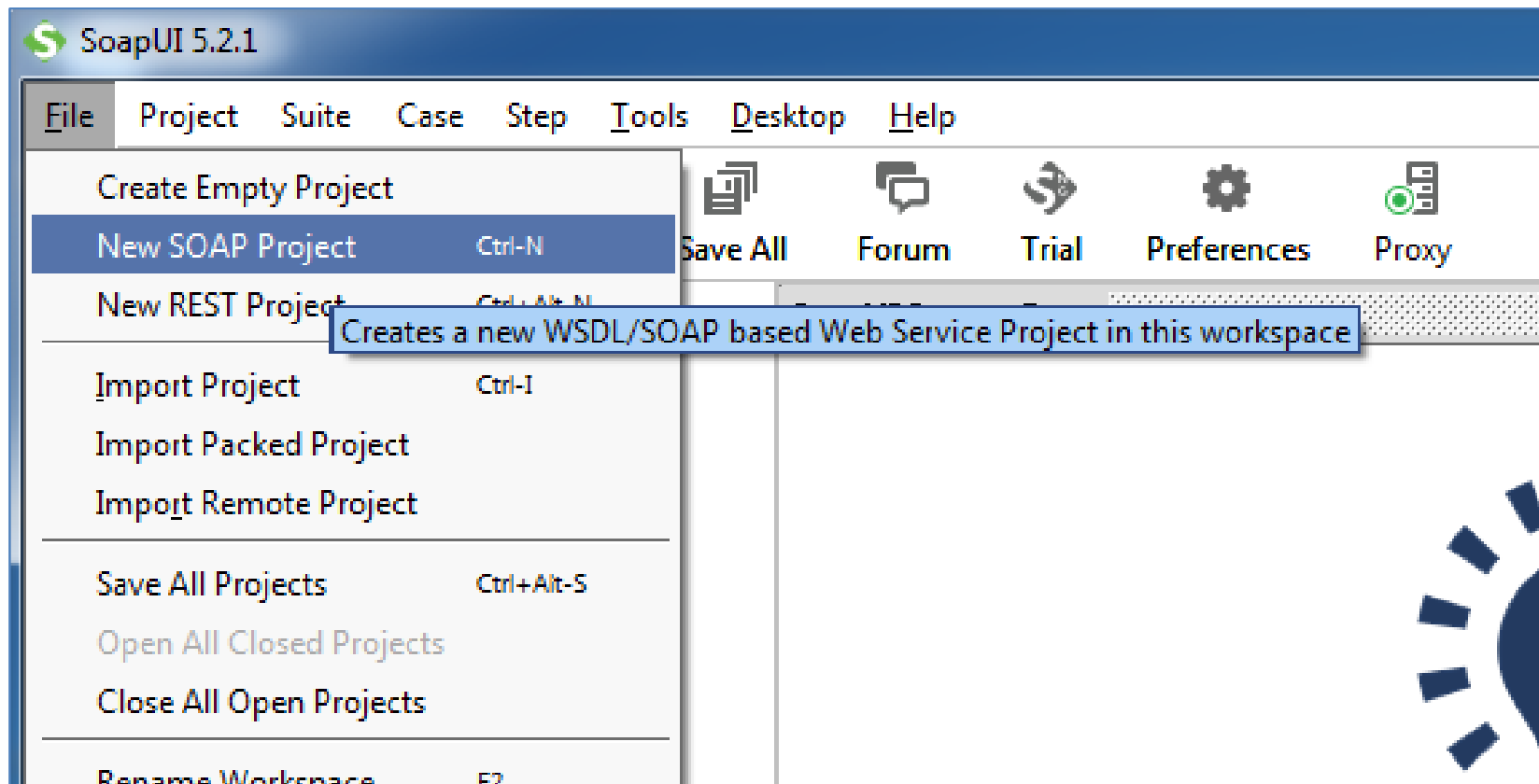
| Agenda

- **O Que será feito nesse Lab.**
 - **Criação do serviço do legado “MOCK”.**
 - **Criação de um serviço “proxy” no Anypoint Studio.**
 - **Testando o serviço “Proxy” no Mule Runtime.**

Criação do serviço do legado “MOCK”

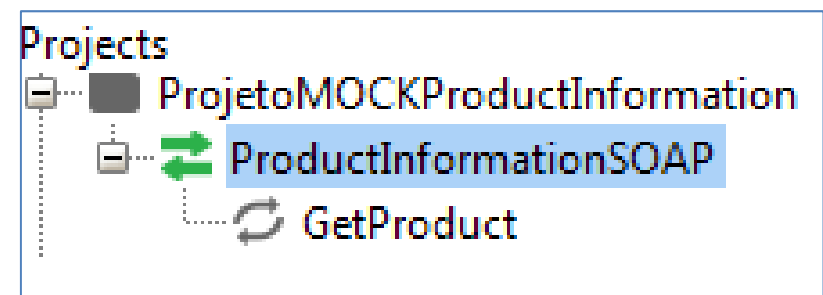
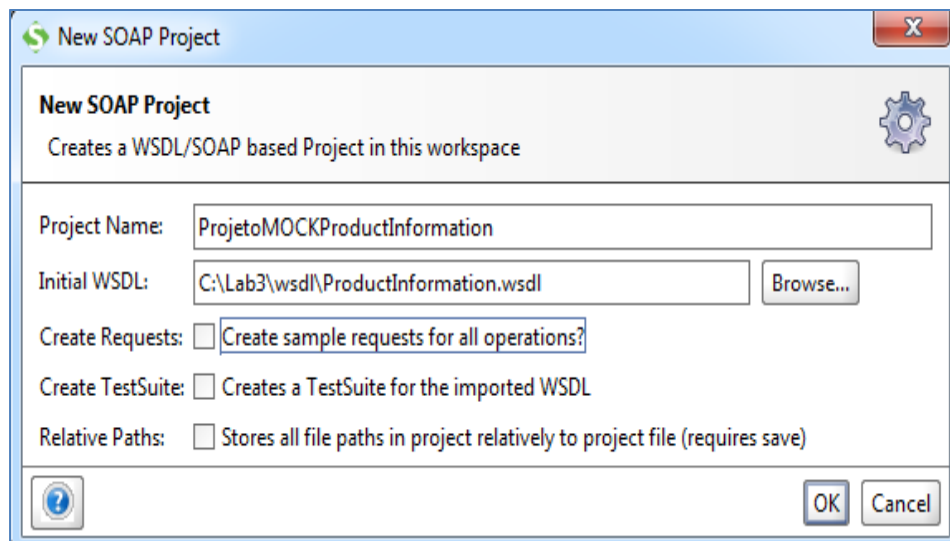
Criação do serviço do legado “MOCK”

1) Abrir o SoapUI e depois ir em “**File**”->“**New SOAP Project**”.



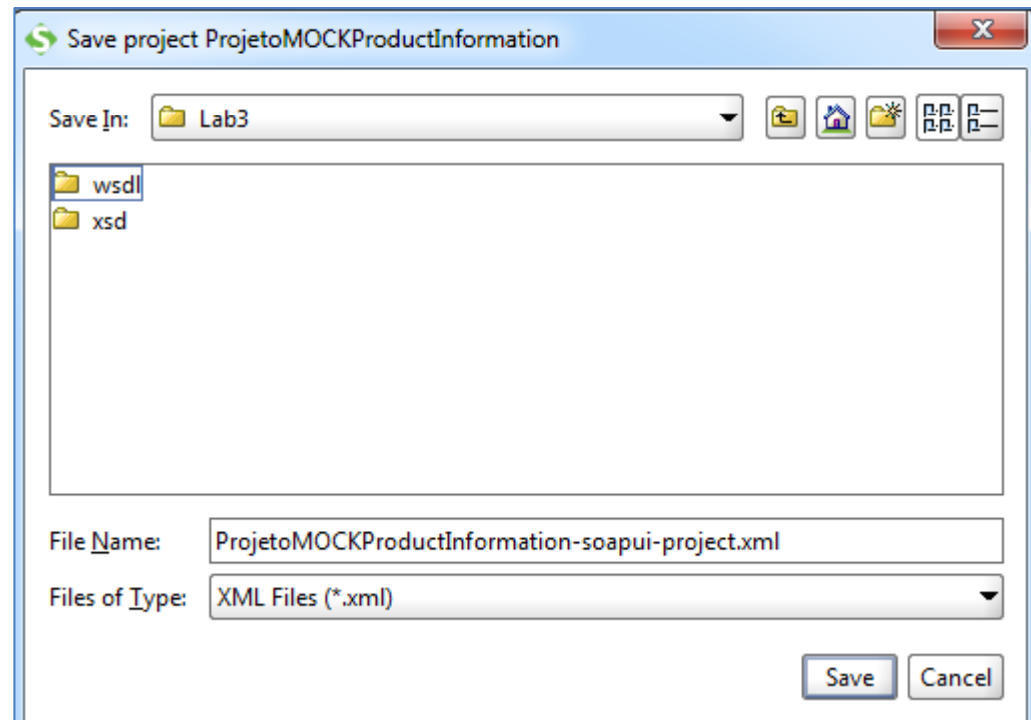
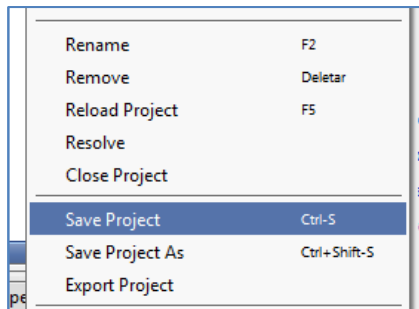
Criação do serviço do legado “MOCK”

2) Colocar o nome: “**ProjetoMOCKProductInformation**”. Depois, em “**Initial WSDL**”, clicar no botão “**Browse**” e procurar pelo arquivo: “**ProductInformation.wsdl**”. Deixe todas as opções desmarcadas e clique em “OK”. Observe que o projeto foi criado.



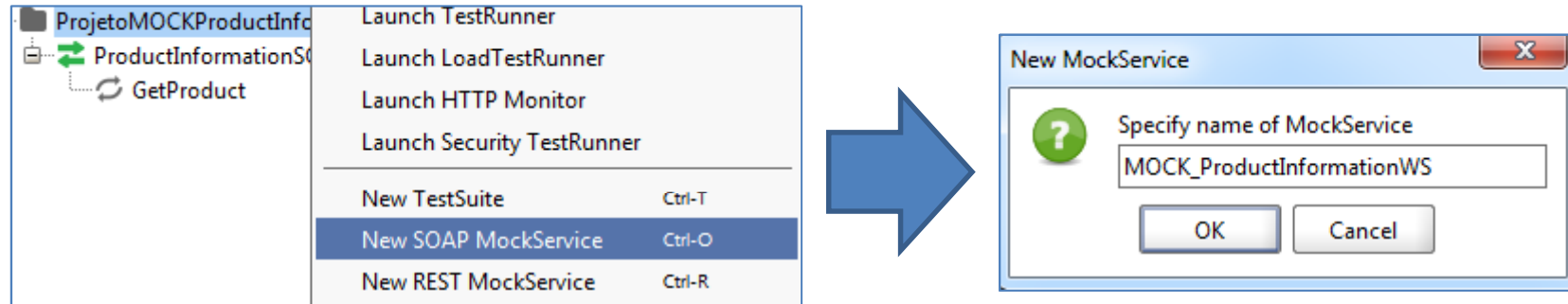
Criação do serviço do legado “MOCK”

3) Salve o projeto (Clique com o botão direito no projeto e escolha “Save Project”)...



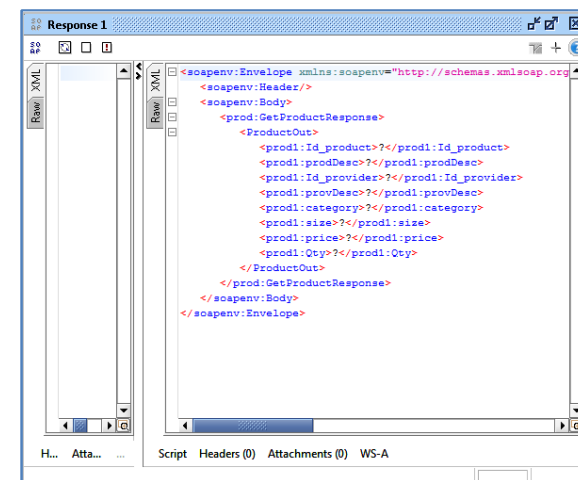
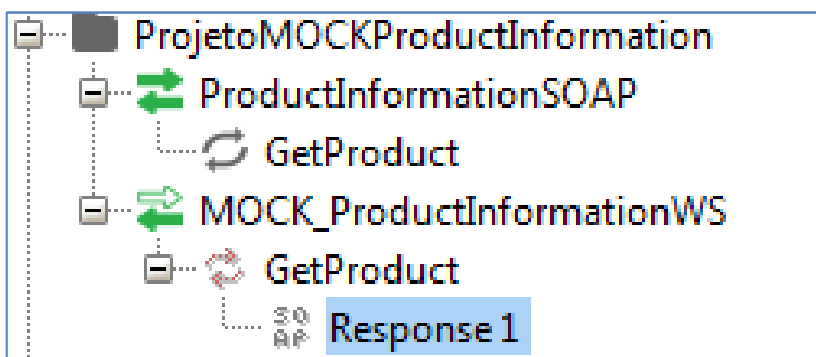
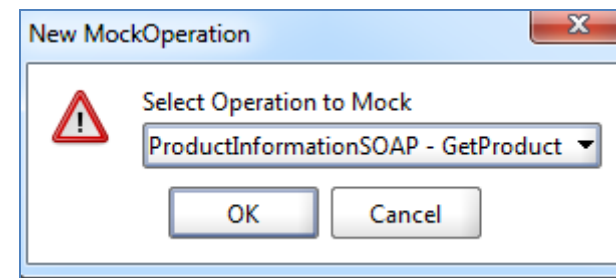
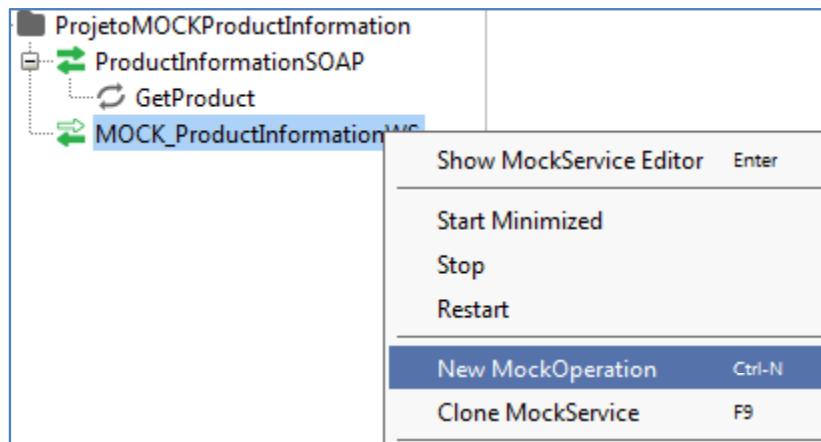
Criação do serviço do legado “MOCK”

4) Clique com o botão direito no projeto “**ProjetoMOCKProductInformation**” e escolha: “**New MockService**”. Coloque o nome: “**MOCK_ProductInformationWS**”.



Criação do serviço do legado “MOCK”

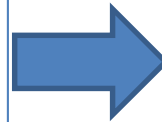
5) Clique no serviço “MOCK” criado com o botão direito para criar uma operação “MOCK”. Escolha a operação “**GetProduct**”.



Criação do serviço do legado “MOCK”

6) Preencha os dados da resposta automática. Não esqueça de salvar.

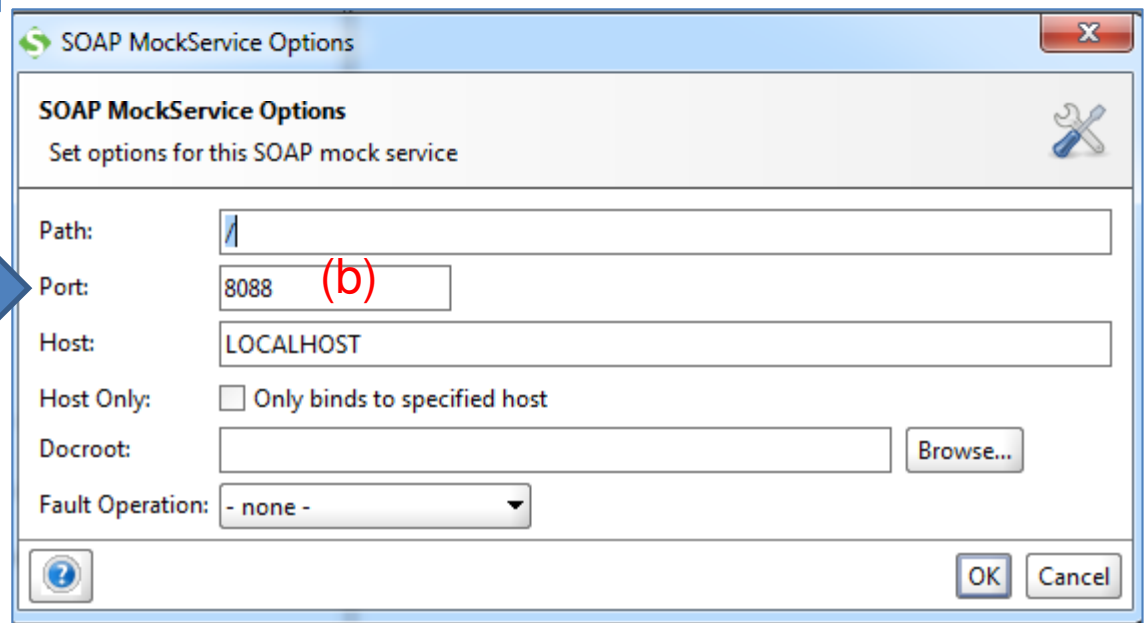
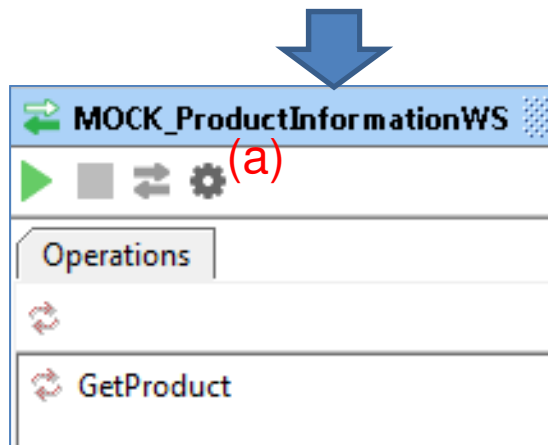
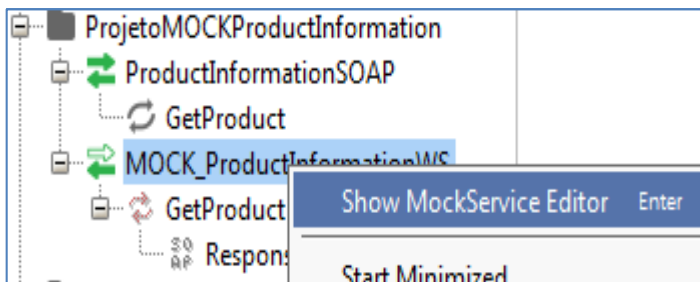
```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <prod:GetProductResponse>
      <ProductOut>
        <prod1:Id_product?/></prod1:Id_product>
        <prod1:prodDesc?/></prod1:prodDesc>
        <prod1:Id_provider?/></prod1:Id_provider>
        <prod1:provDesc?/></prod1:provDesc>
        <prod1:category?/></prod1:category>
        <prod1:size?/></prod1:size>
        <prod1:price?/></prod1:price>
        <prod1:Qty?/></prod1:Qty>
      </ProductOut>
    </prod:GetProductResponse>
  </soapenv:Body>
</soapenv:Envelope>
```



```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <prod:GetProductResponse>
      <ProductOut>
        <prod1:Id_product>001</prod1:Id_product>
        <prod1:prodDesc>CAMISA SOCIAL</prod1:prodDesc>
        <prod1:Id_provider>05</prod1:Id_provider>
        <prod1:provDesc>BRASIL MODAS</prod1:provDesc>
        <prod1:category>MODA MASCULINA</prod1:category>
        <prod1:size>G</prod1:size>
        <prod1:price>150</prod1:price>
        <prod1:Qty>5</prod1:Qty>
      </ProductOut>
    </prod:GetProductResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

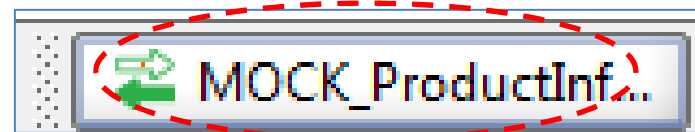
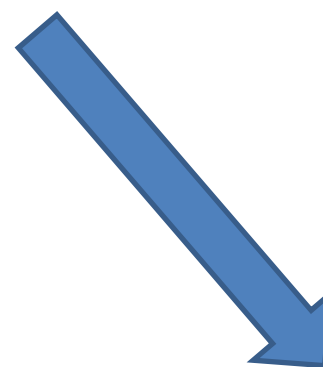
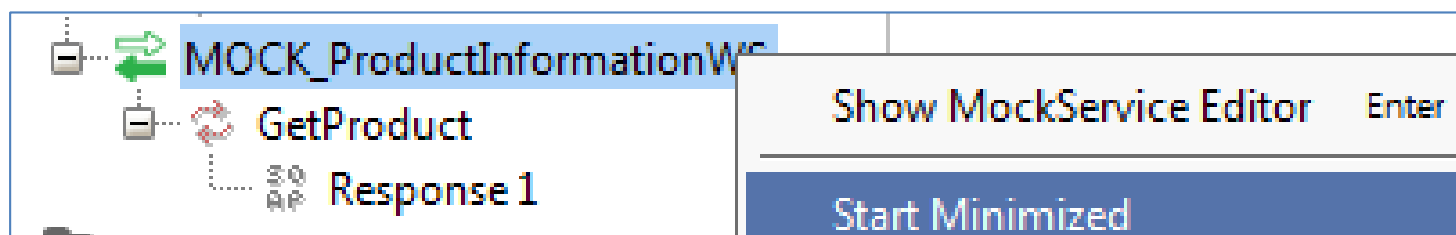
Criação do serviço do legado “MOCK”

8) Vamos alterar a porta do serviço. Clique com o botão direito em “”, depois clique no ícone de opções (a) e coloque o endereço 8088 (b). Depois dê “OK” e feche tudo.



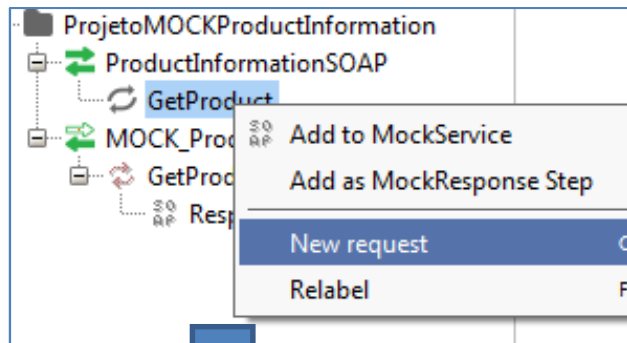
Criação do serviço do legado “MOCK”

9) Vamos agora colocar o MOCK para funcionar. Basta clicar com o botão direito no MOCK criado e depois escolher “Start Minimized”. Ele executará abaixo:

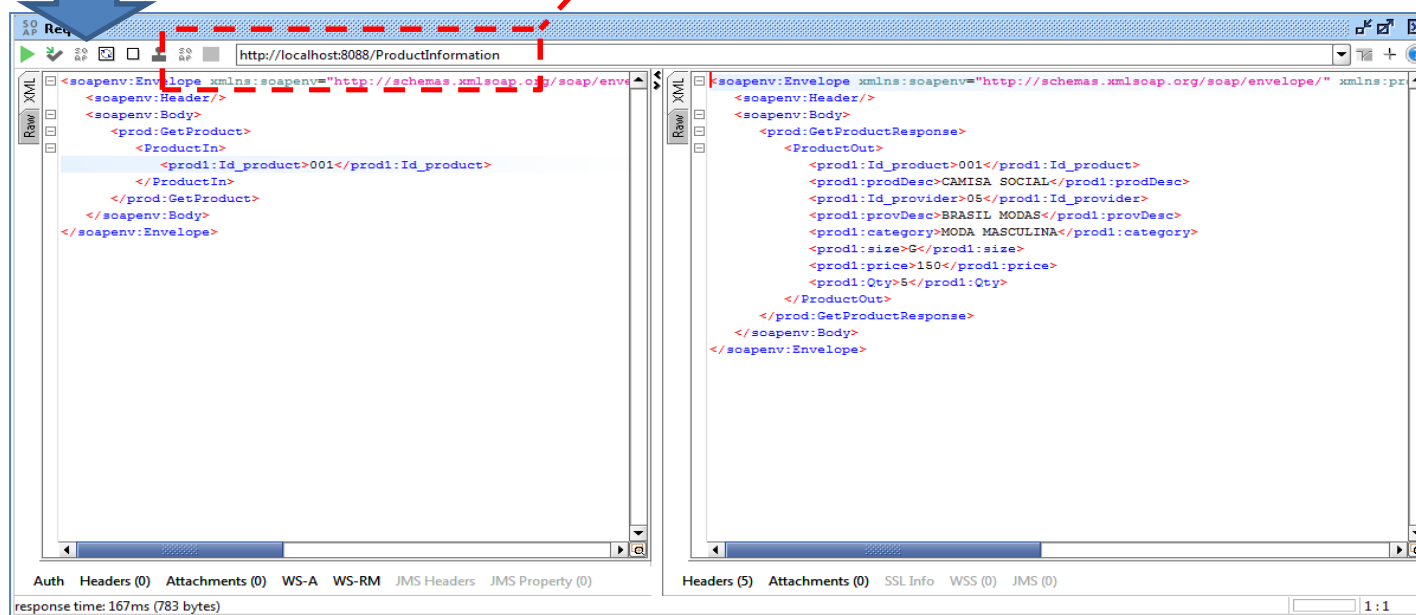


Criação do serviço do legado “MOCK”

10) Para testá-lo, vamos criar dois “requests”, uma para cada operação. Primeiramente clique com o botão direito em “**GetProduct**” e depois em “New Request”, coloque o mesmo nome para o “request”. Depois clique no botão “verde” :

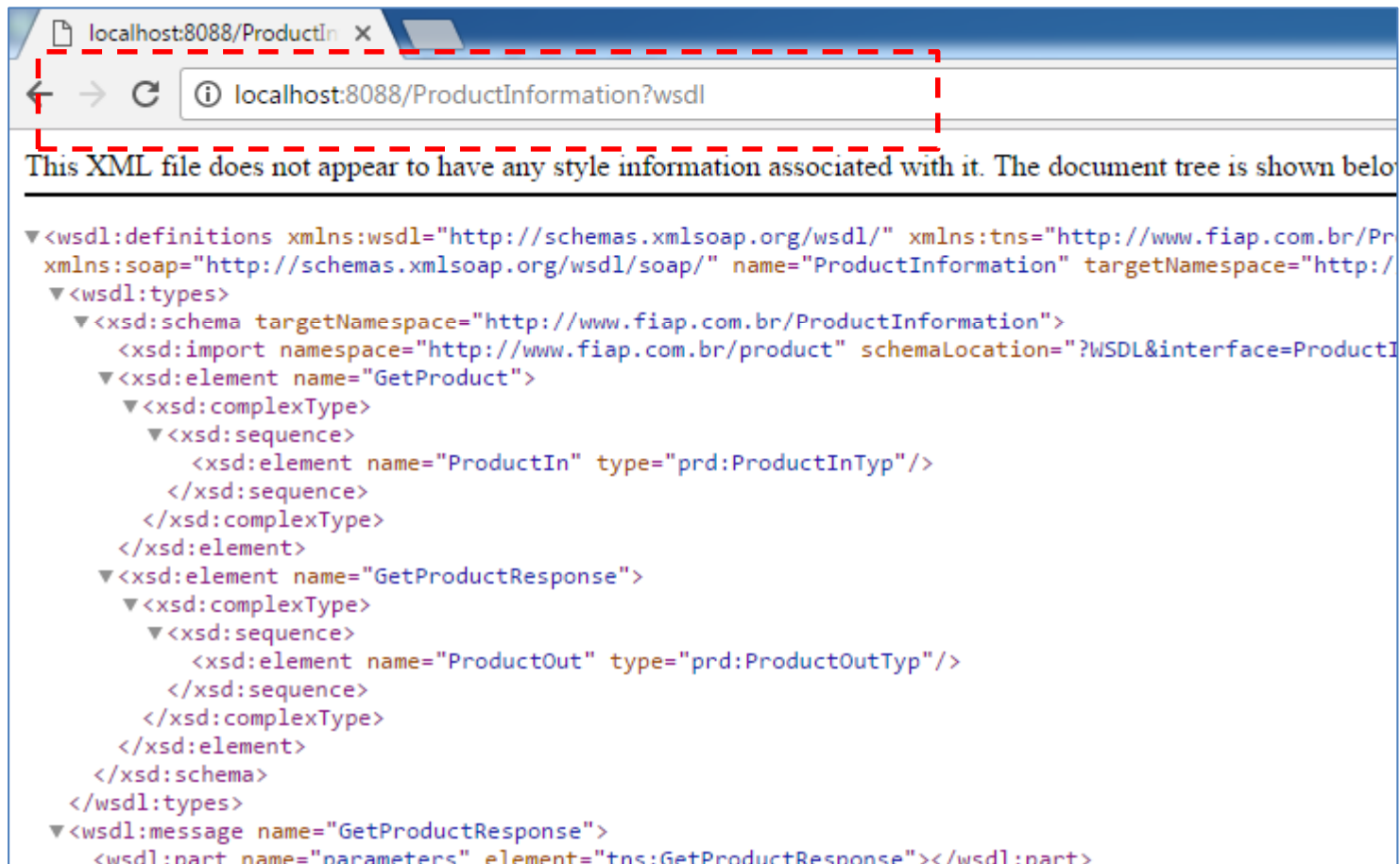


Lembre-se de alterar a porta para 8088



Criação do serviço do legado “MOCK”

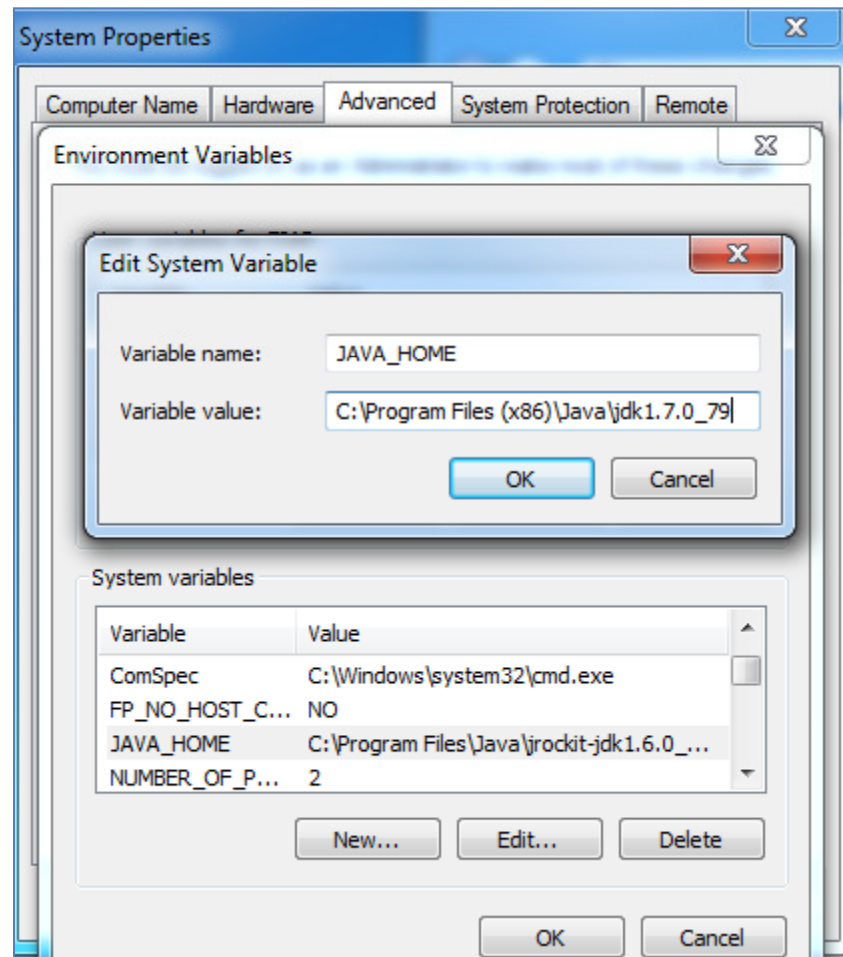
11) Copie a URL “***http://localhost:8088/ProductInformation?wsdl***” , abra um browser e coloque a mesma. Você verá o WSDL gerado. Deixa essa tela aberta que você irá precisar desse endereço.



Criação de um serviço “proxy” no Anypoint Studio.

Criação de um serviço “proxy” no Anypoint Studio

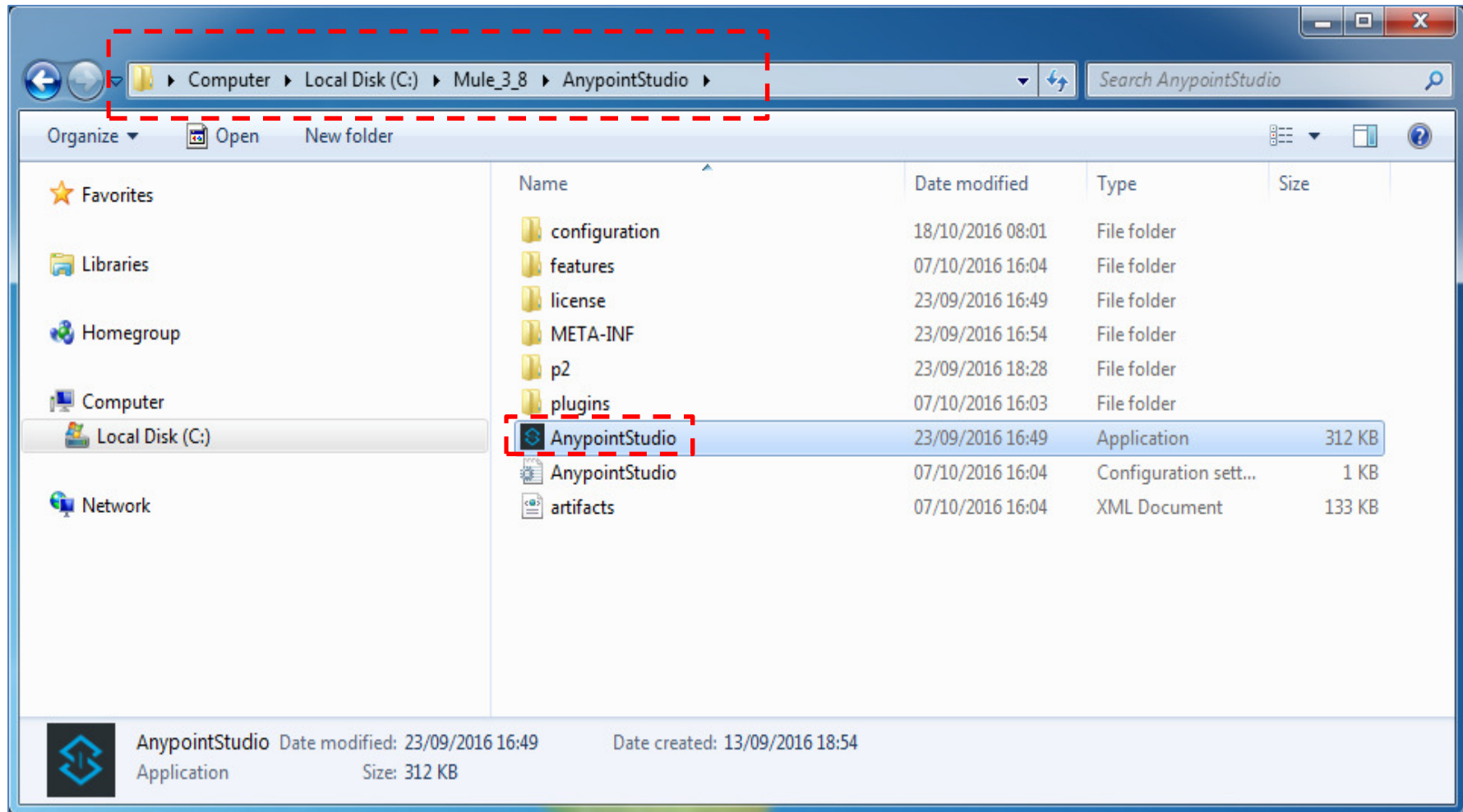
12) Inicialmente, temos que checar a versão de java a ser utilizada. Temos que configurar a variável de ambiente `java_home` para: `C:\Program Files (x86)\Java\jdk1.7.0_79` conforme figura abaixo.



Criação de um serviço “proxy” no Anypoint Studio

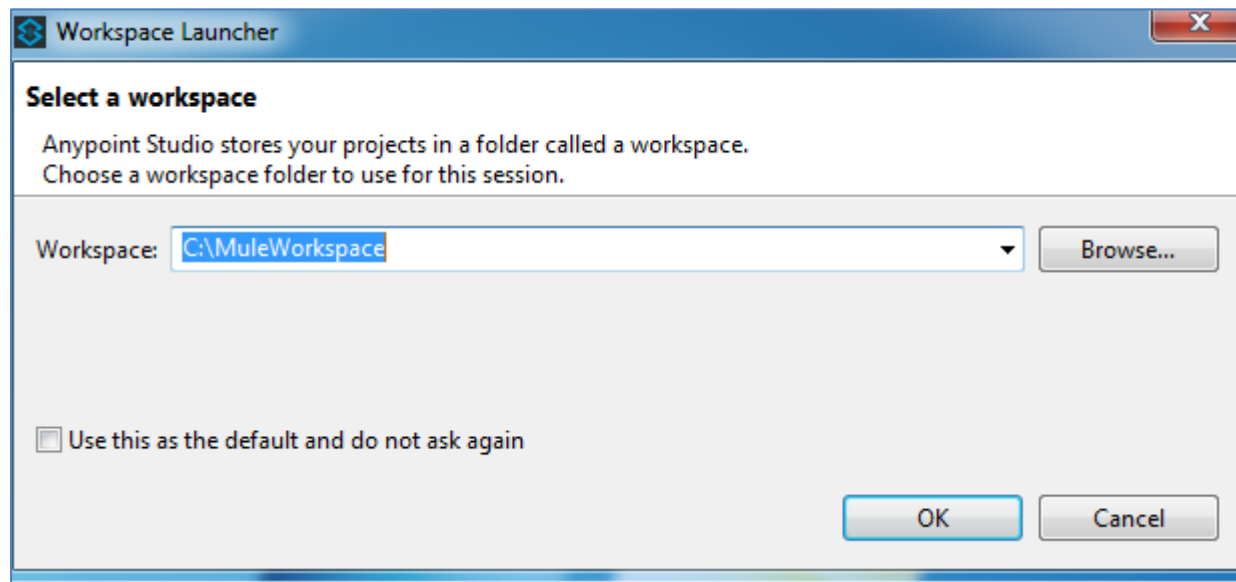


13) Inicie o Mule Anypoint Studio que se encontra no caminho: **c:\Mule_3_8\AnypointStudio**.



Criação de um serviço “proxy” no Anypoint Studio

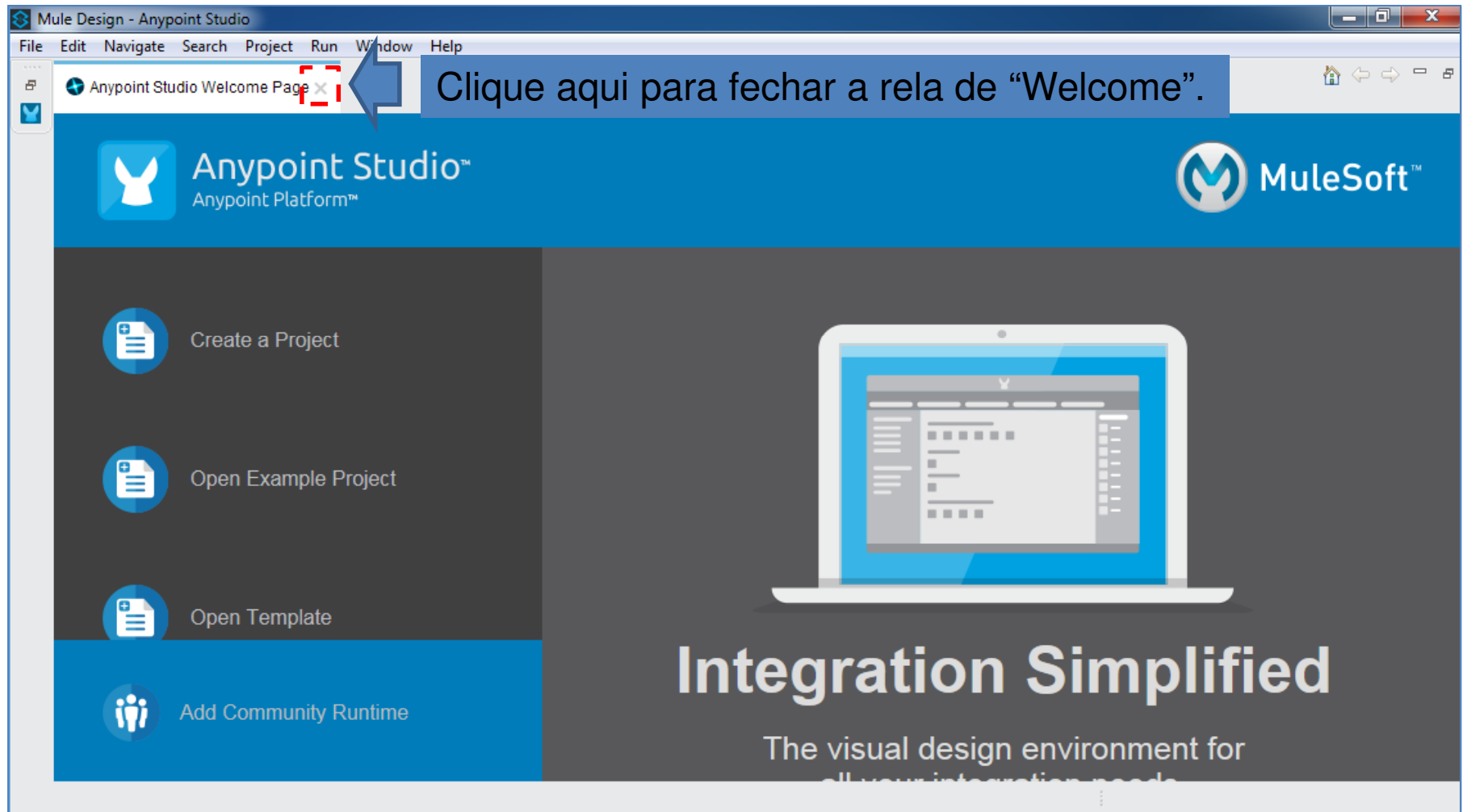
13.1) Ao abrir o Mule Anypoint Studio e confirme o local de “workspace” para a ferramenta que é baseada no IDE Eclipse. Coloque o nome que quiser. No exemplo, foi estabelecido o nome: **“C:\MuleWorkspace”**.



Criação de um serviço “proxy” no Anypoint Studio



14) A tela de “Welcome” aparece. Feche a tela.



Criação de um serviço “proxy” no Anypoint Studio

15) A tela eclipse do Mule é formada pelas seguintes partes:

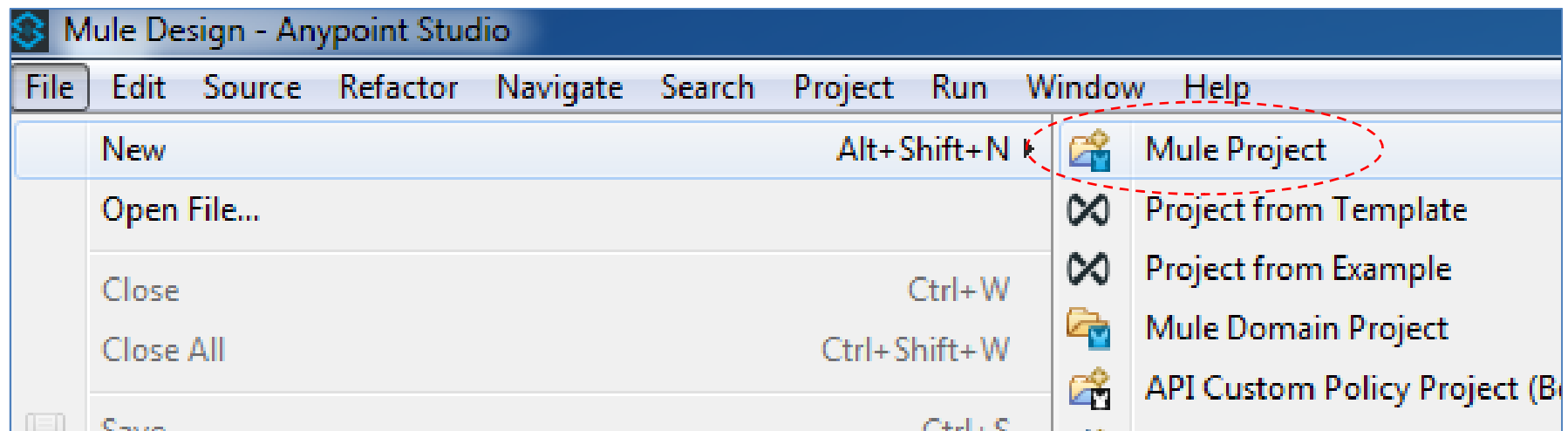
The screenshot displays the Anypoint Studio Mule Design interface. The central canvas shows a message flow starting with an HTTP connector, followed by a CXF connector, then a Transform Message component, a Web Service Consumer, and another Transform Message. The interface is divided into several panels:

- Package Explorer:** Located on the left, it shows the project structure, including folders like 'basic_tutorial', 'papitest', 'pproduct', and 'pproductwithtransformation'. A callout box points to it with the text: "Package Explorer: Estrutura do projeto".
- Canvas:** The central workspace where the message flow is designed. A callout box points to it with the text: "Canvas: onde os fluxos são desenhados".
- Paleta:** Located on the right, it contains a search bar and a list of connectors (Ajax, Amazon S3, Amazon SQS). A callout box points to it with the text: "Paleta: componentes à disposição para a construção de fluxos."
- Properties Editor:** Located at the bottom, it shows the configuration for the selected CXF component. A callout box points to it with the text: "Proprieties Editor: Para configuração dos componentes." (Note the typo in the original image).
- Console:** Located at the bottom right, it displays the execution logs of the message flow. A callout box points to it with the text: "Console: Onde a execução do fluxo é acompanhada através de mensagens".

Criação de um serviço “proxy” no Anypoint Studio



16) Como o Mule, podemos criar aplicações diversas. Vamos começar criando um web service que será de “proxy” para um webservice “externo”. Clique em “File”->”Mule Project”.



Criação de um serviço “proxy” no Anypoint Studio



17) Coloque o nome do projeto: “**PProxyToProductInformation**” escolha o Runtime “Mule Server 3.8.1. EE” e depois clique em “Finish”.

The screenshot shows the 'New Mule Project' dialog box in Anypoint Studio. The 'Project Settings' section is active, showing the 'Project Name' as 'PProxyToProductInformation'. The 'Runtime' section shows 'Mule Server 3.8.1 EE' selected. The 'Maven Settings' section is collapsed. The 'Finish' button is highlighted with a red dashed circle.

Project Settings
Create a Mule project in the workspace or in an external location.

Project Name: PProxyToProductInformation

Runtime
Mule Server 3.8.1 EE
Mule Server 3.8.0 CE

Compatibility: ☁ = CloudHub 🏢 = On Premises

Maven Settings
☐ Use Maven - (Maven is currently disabled, [configure Maven](#))

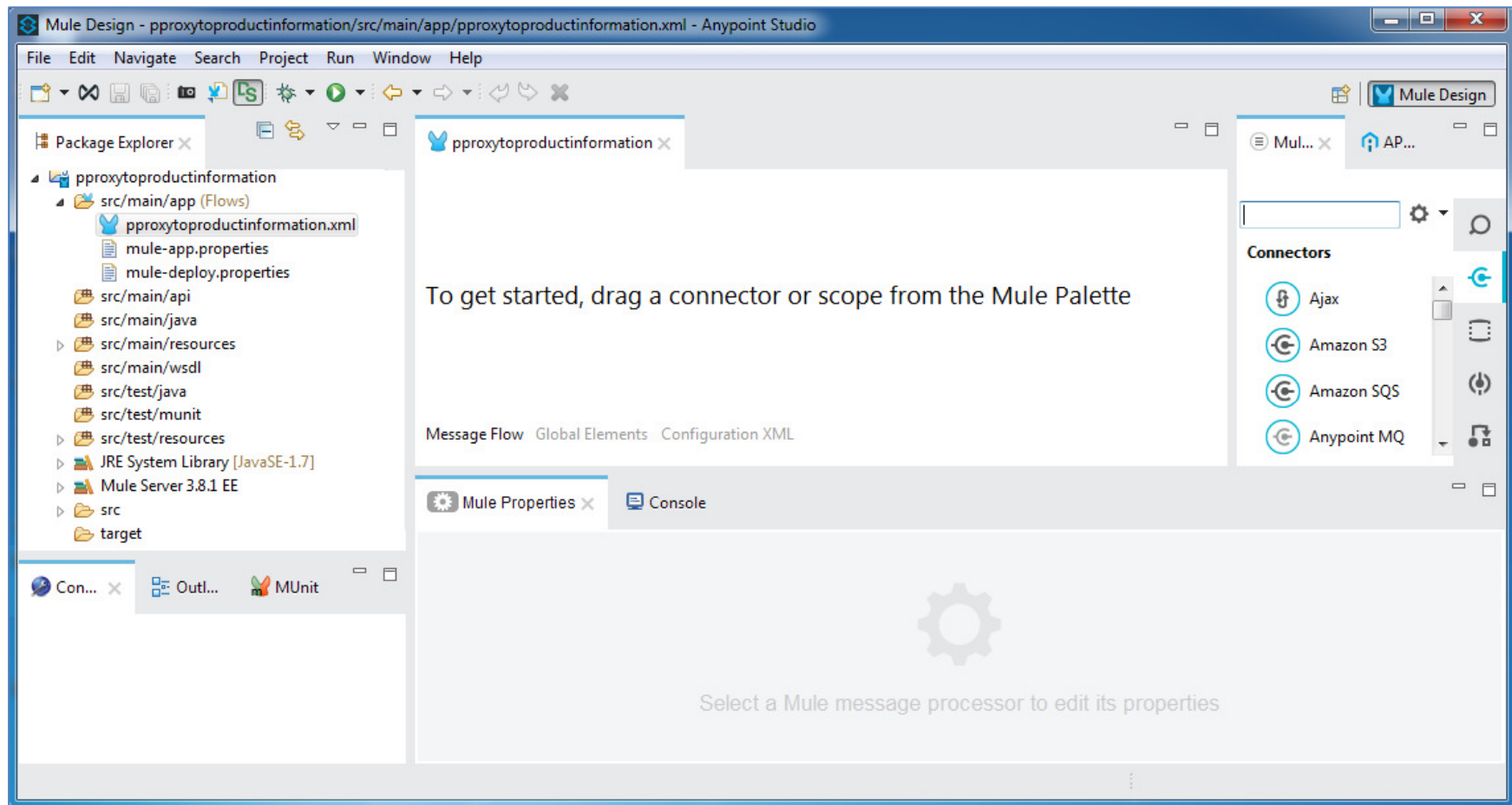
Group Id: com.mycompany
Artifact Id: pproxytoproductinformation
Version: 1.0.0-SNAPSHOT

< Back Next > **Finish** Cancel

Criação de um serviço “proxy” no Anypoint Studio



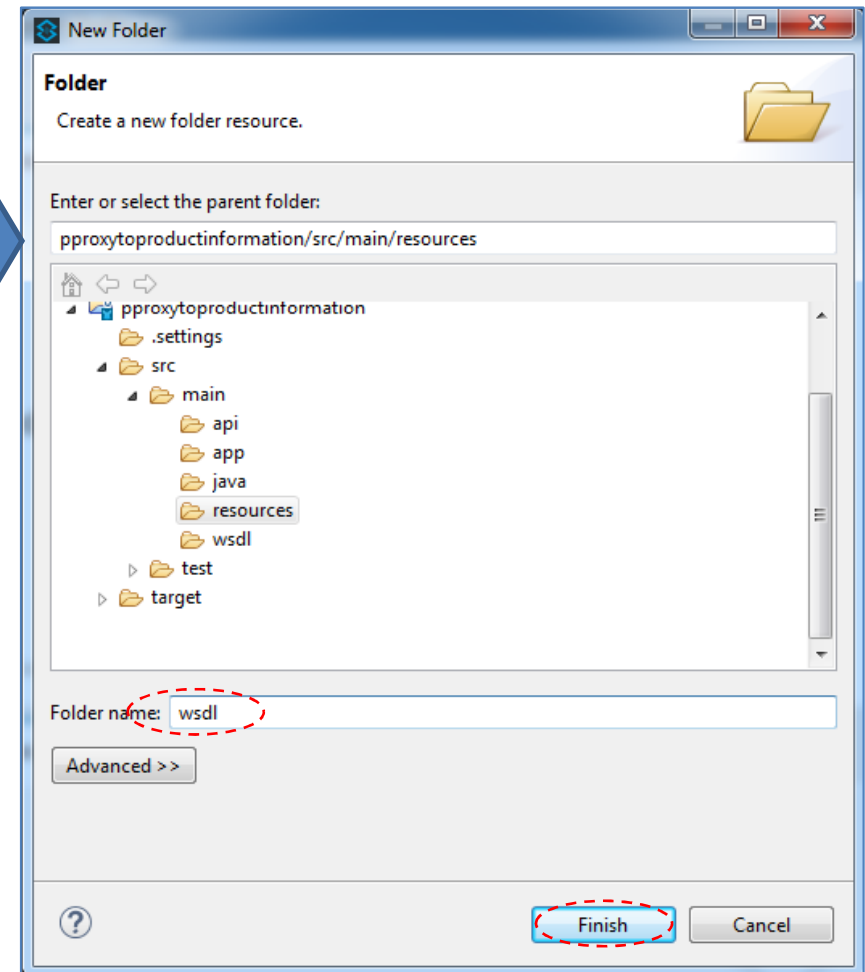
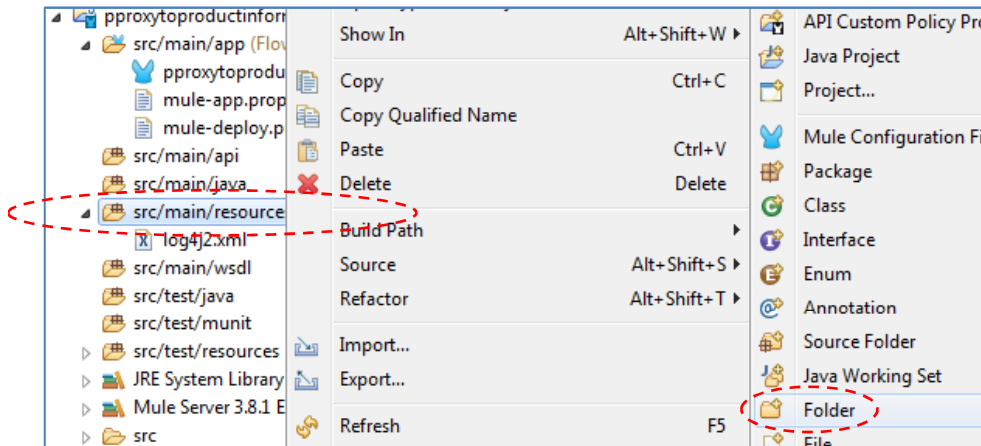
18) A estrutura do projeto aparecerá pronta para a construção do fluxo.



Criação de um serviço “proxy” no Anypoint Studio

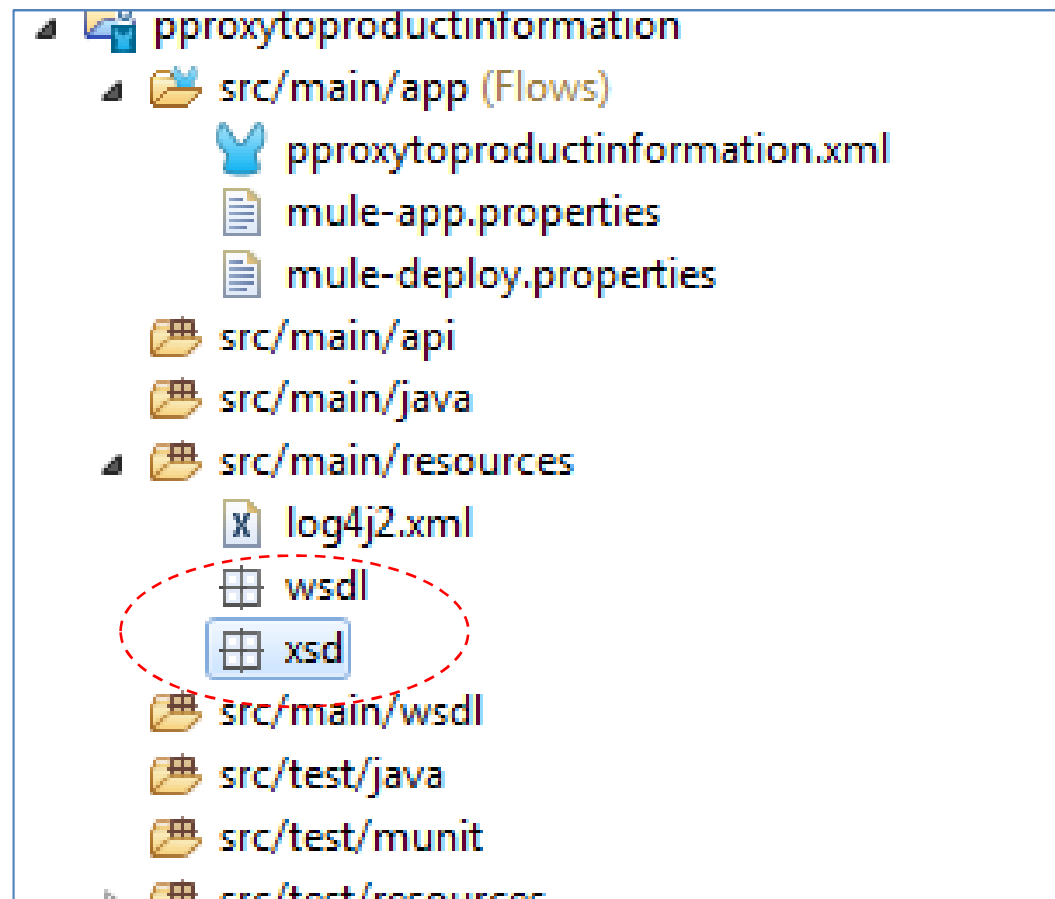


19) Vamos importar os wsdl e xsds que precisaremos para o exemplo. Clique com o botão direito em “src/main/resources”->”New”->”Folder”. Depois coloque o nome de “wsdl”.



Criação de um serviço “proxy” no Anypoint Studio

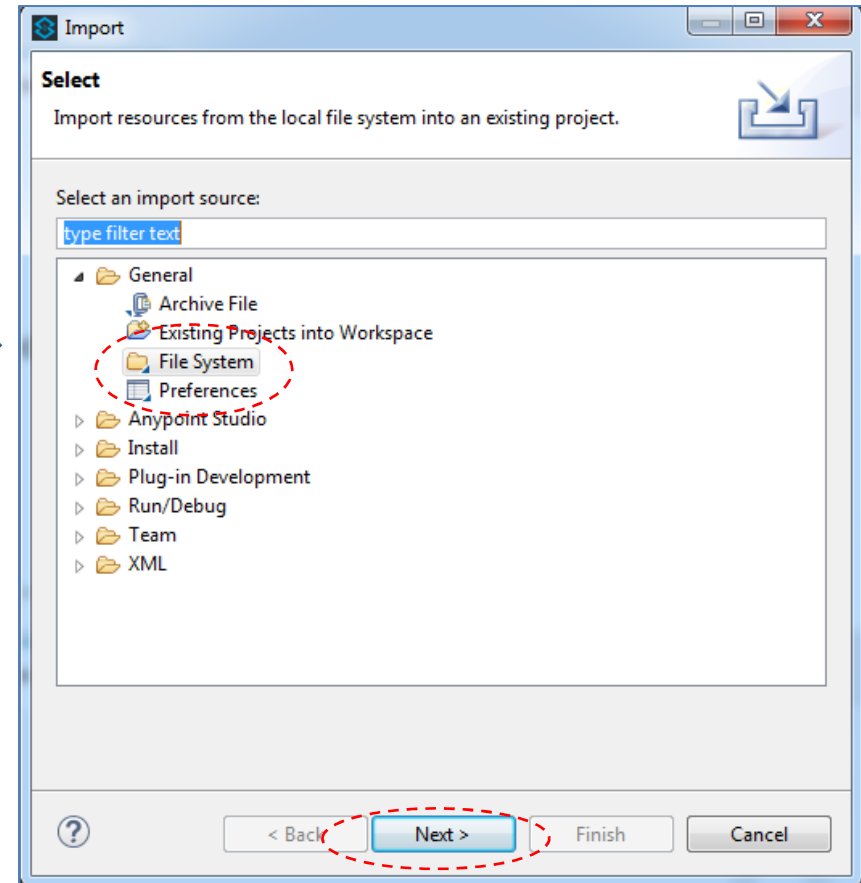
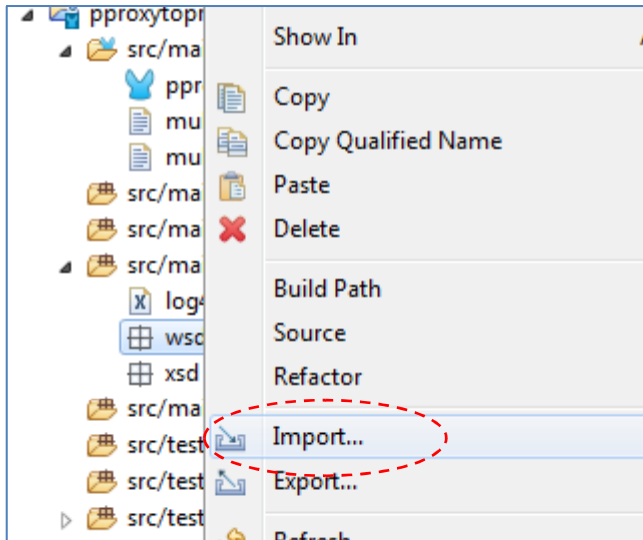
20) Repita o passo anterior e crie uma pasta “xsd” conforme a figura abaixo:



Criação de um serviço “proxy” no Anypoint Studio

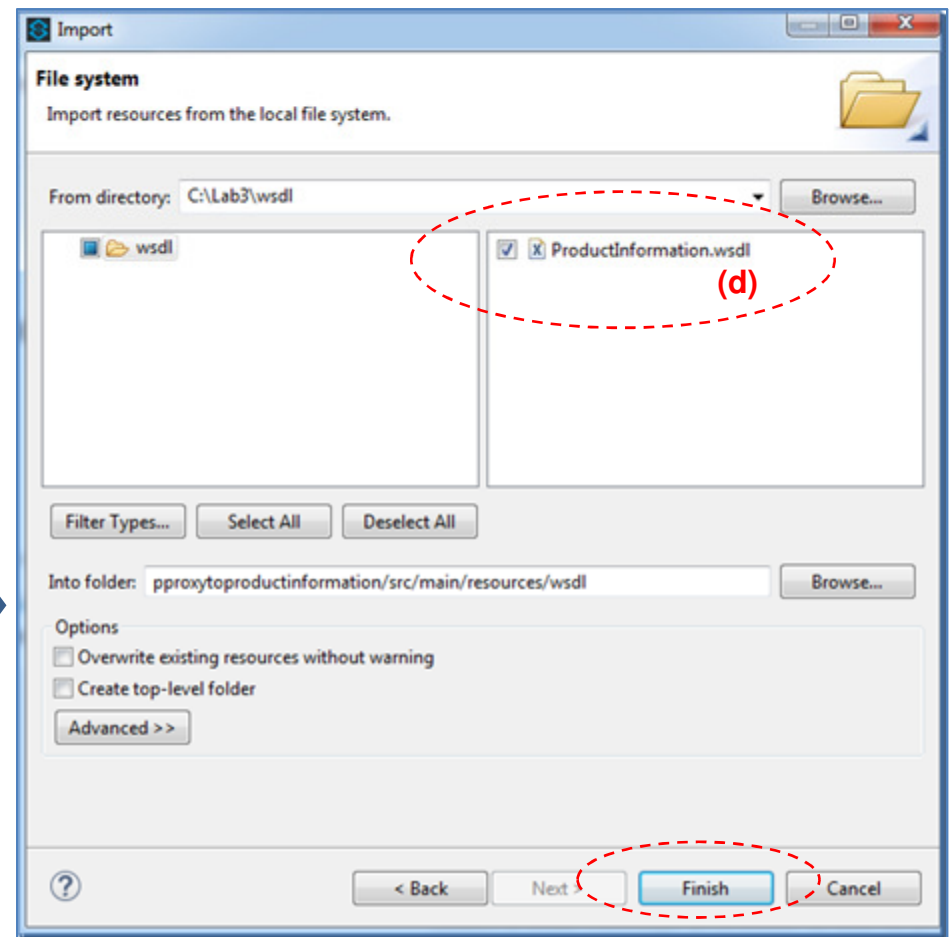
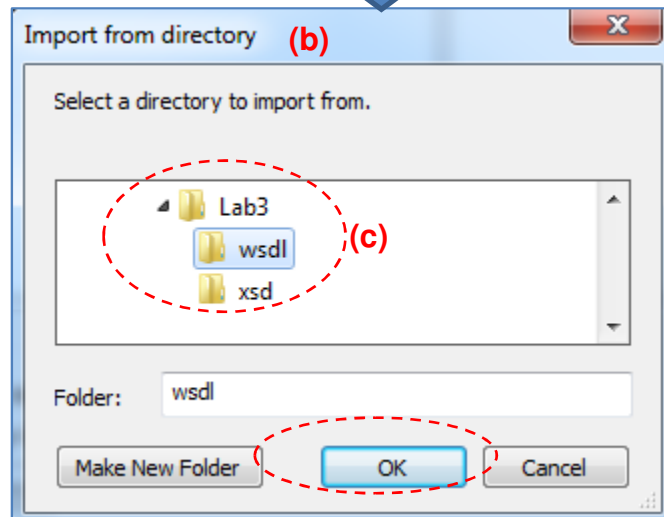
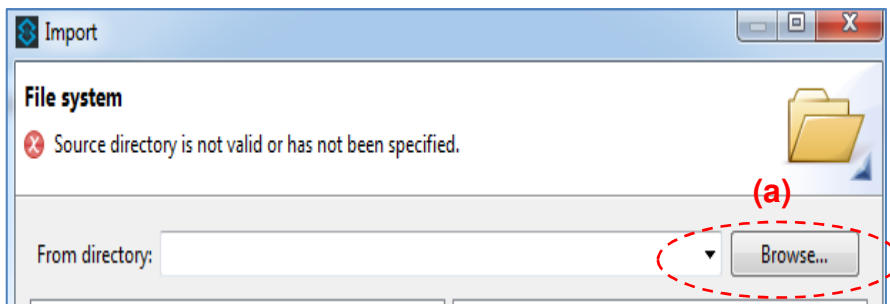


21) Vamos importar os arquivos necessários. Clique com o botão direito em “wsdl”->”Import”. Depois escolha “File System” e clique em “Next”.



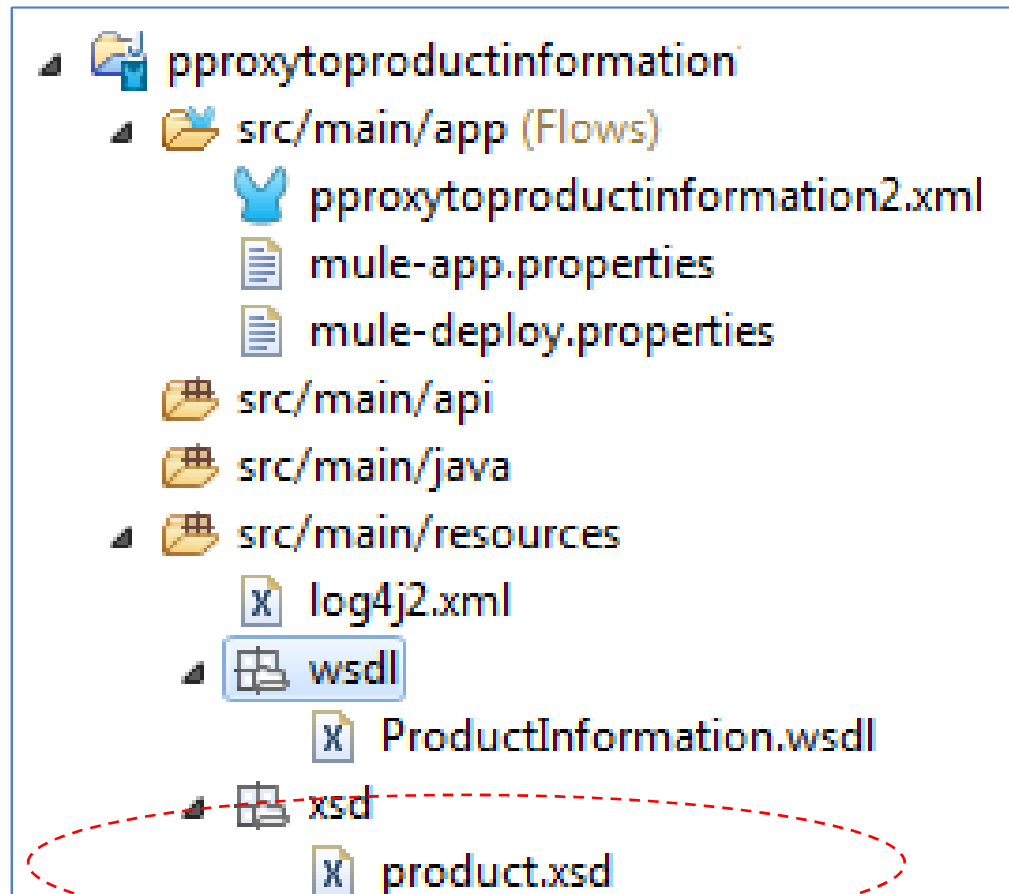
Criação de um serviço “proxy” no Anypoint Studio

22) Na próxima tela, clique em “browse” (a), depois na tela de import (b), selecione o diretório “c:\Lab3\wsdl” (c) e clique “OK”. No final escolha o arquivo wsdl indicado na figura (d) e clique em “Finish”.



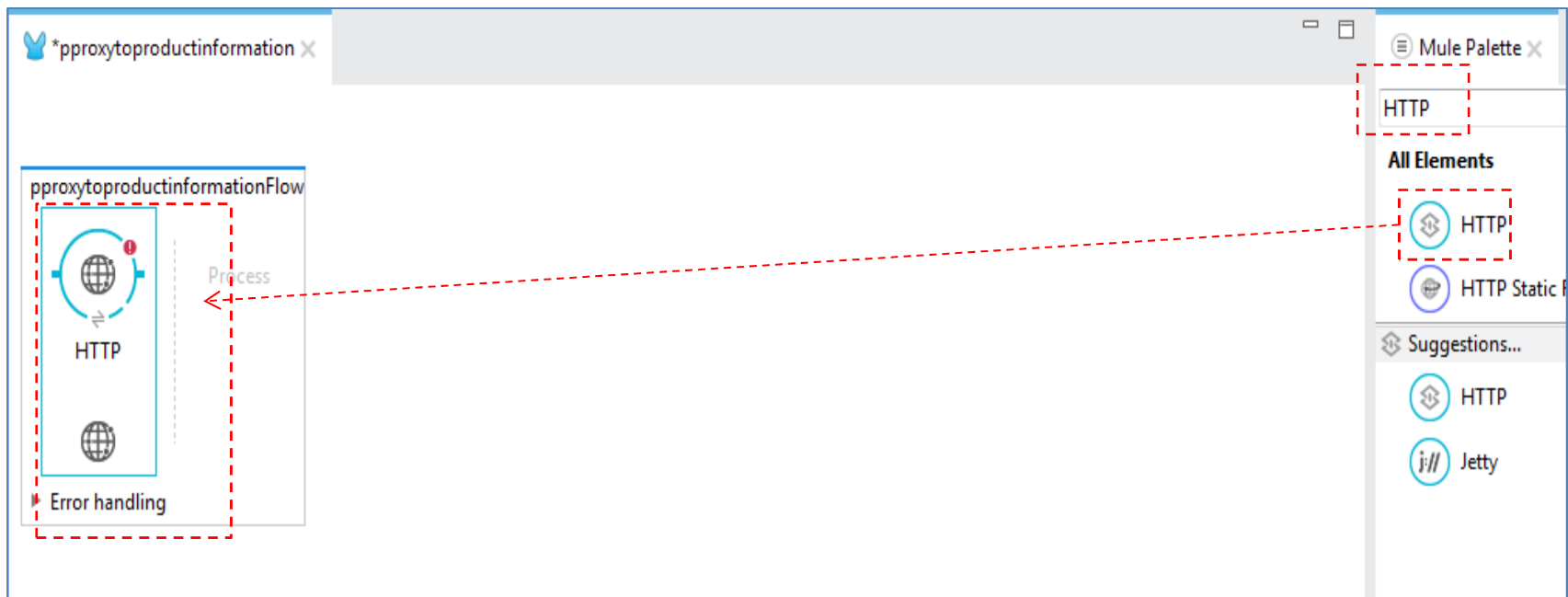
Criação de um serviço “proxy” no Anypoint Studio

23) Repita o procedimento para importar os xsd conforme a figura abaixo:



Criação de um serviço “proxy” no Anypoint Studio

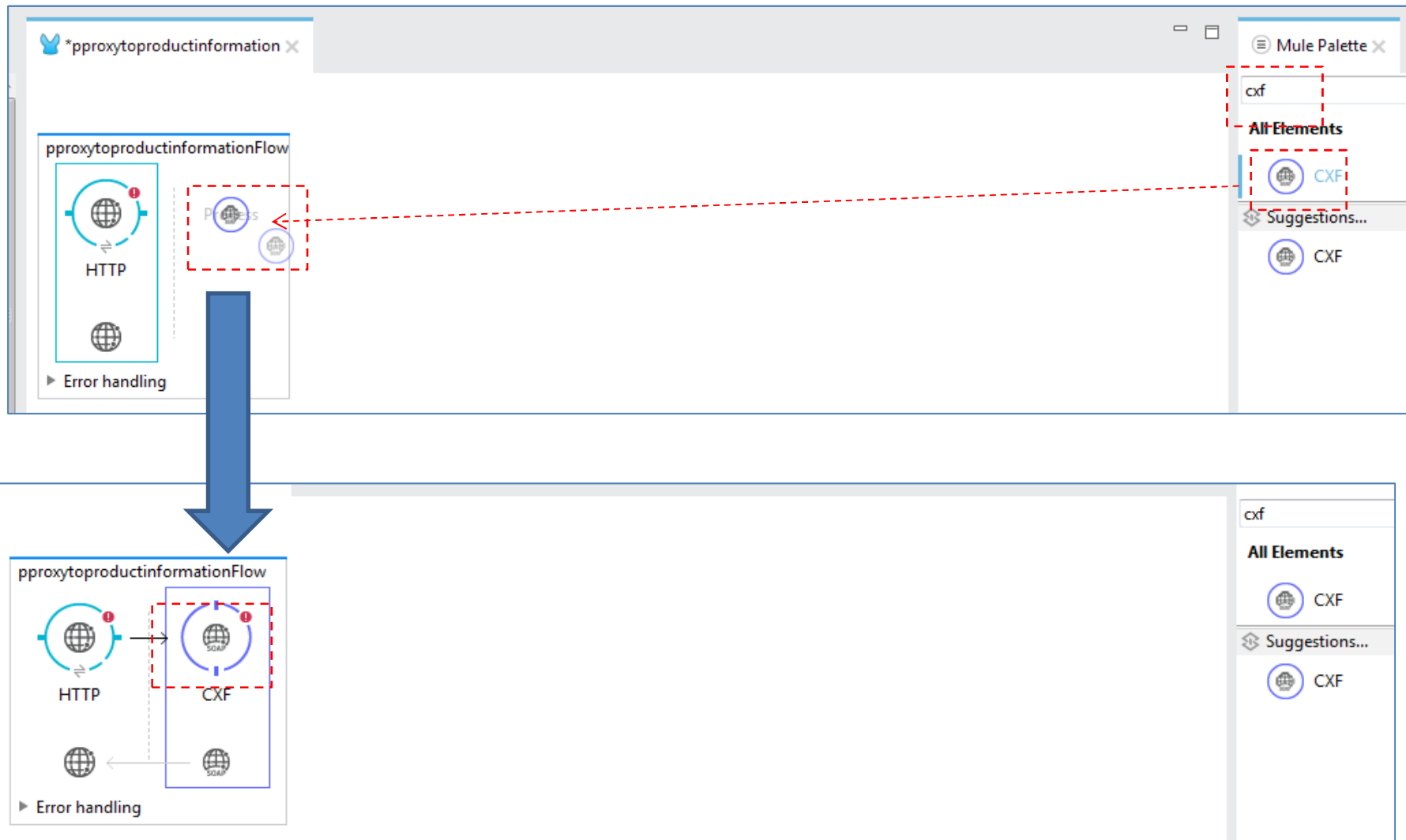
24) Vá na paleta ao lado, procure o componente HTTP, clique e arraste-o até a área de fluxo de mensagens (Canvas) conforme abaixo.



Criação de um serviço “proxy” no Anypoint Studio

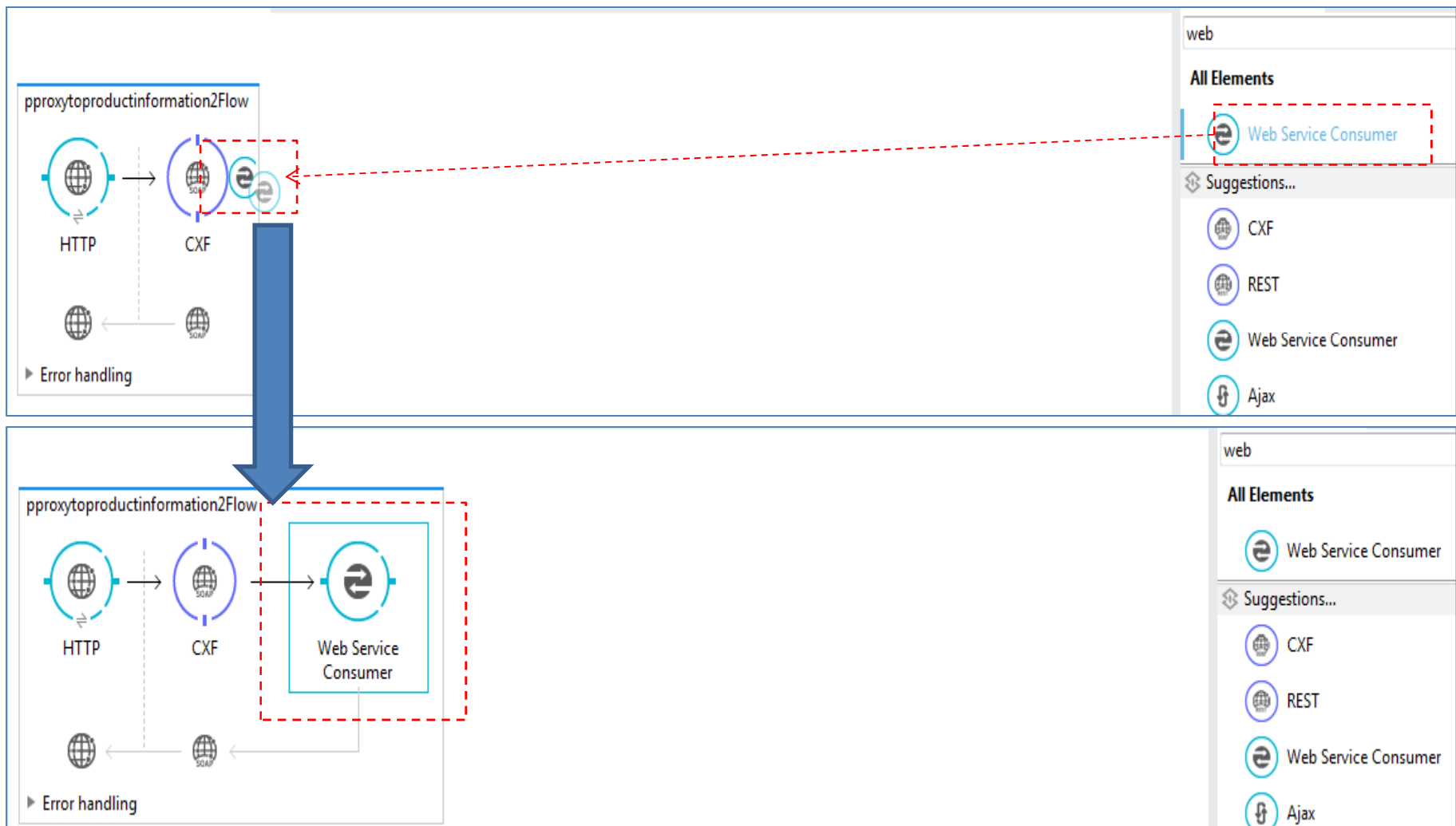


25) Vá na paleta ao lado, procure o componente CXF, clique e arraste-o até ao lado direito do componente HTTP já colocado conforme figuras abaixo.



Criação de um serviço “proxy” no Anypoint Studio

26) Vá na paleta ao lado, procure o componente “Web Service Consumer”, clique e arraste-o até ao lado direito do componente “CXF” já colocado conforme figuras abaixo.



Criação de um serviço “proxy” no Anypoint Studio

27) Primeiramente vamos conectar o webservice “externo”. Clique em cima do componente “Web Service Consumer”. Abaixo, aparecerá as propriedades desse componente (a). Clique no sinal de “+” verde para configurar o conector. (b)

The screenshot displays the Anypoint Studio interface. At the top, a message flow diagram titled 'pproxytoproductioninformationFlow' shows a sequence of components: HTTP, CXF, and Web Service Consumer. The Web Service Consumer component is highlighted with a red dashed box and labeled (a). Below the diagram, the 'Web Service Consumer' component is selected in the left-hand pane. The 'General' tab is active, showing the 'Display Name' as 'Web Service Consumer'. Under 'Basic Settings', the 'Connector Configuration' dropdown is open, and a green plus sign (+) is visible next to it, indicating the option to add a new configuration. This area is also highlighted with a red dashed box and labeled (b). The 'Operation' dropdown is also visible. A 'Reload WSDL' button is located at the bottom right. A warning message 'Attribute 'config-ref' is required' is displayed at the top of the configuration pane.

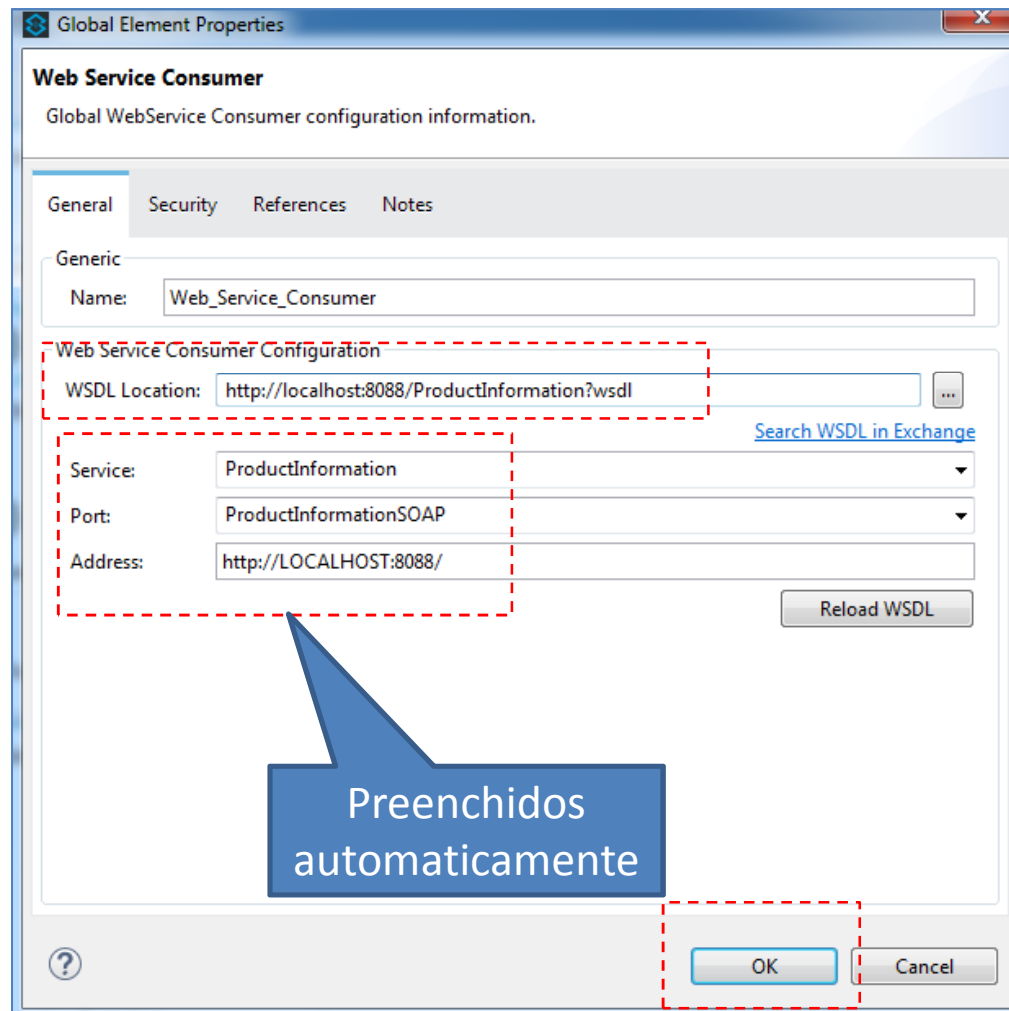
Criação de um serviço “proxy” no Anypoint Studio



28) No campo “WSDL Location”, coloque o wsdl:

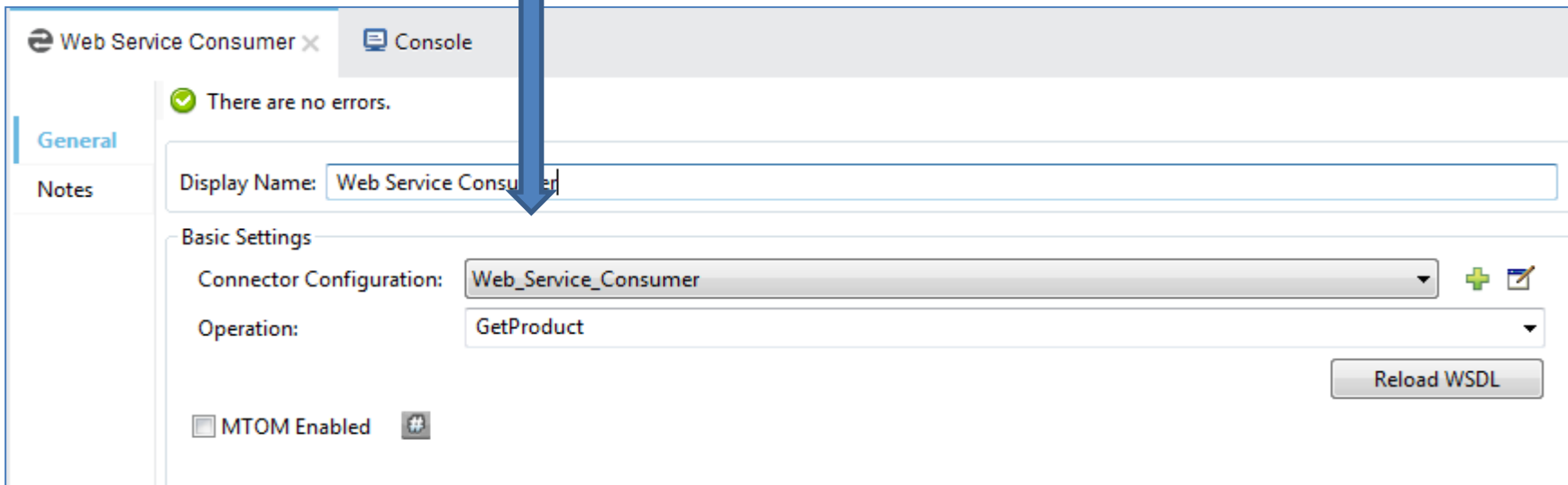
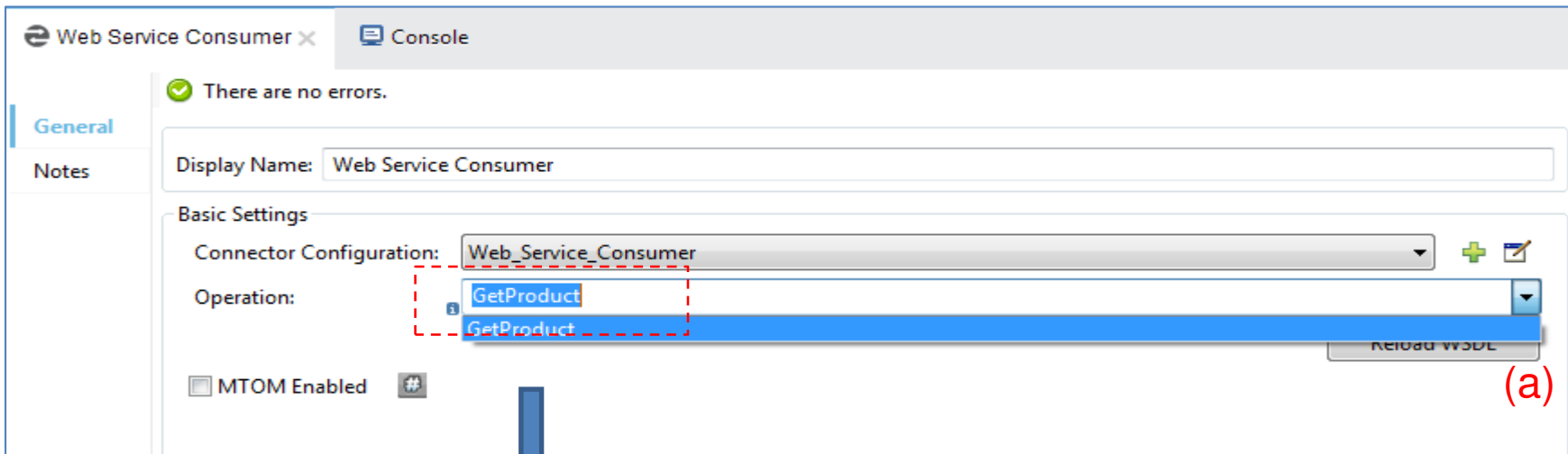
“http://localhost:8088/ProductInformation?wsdl” .

Aguarde alguns segundos e os demais campos serão preenchidos automaticamente. Clique em “OK”.



Criação de um serviço “proxy” no Anypoint Studio

29) Escolha a operação “GetProduct” (a) e depois salve a aplicação. Pronto, o cliente webservice está configurado.



Criação de um serviço “proxy” no Anypoint Studio

30) Agora, vamos configurar o “proxy”. Clique no componente “CXF” (a), depois selecione o tipo de operação do componente, no caso, escolha “Proxy service” (b).

The screenshot displays the Anypoint Studio interface. At the top, a message flow diagram titled 'pproxytoproductinformationFlow' shows a sequence of components: HTTP, CXF (labeled with a red '(a)'), and Web Service Consumer. The CXF component is highlighted with a red dashed box. Below the diagram, the 'Message Flow' tab is active, and the 'CXF' component is selected in the left-hand pane, also highlighted with a red dashed box. The right-hand pane shows the configuration for the CXF component. The 'Display Name' is 'CXF'. Under the 'Generic' section, the 'Operation' dropdown menu is open, showing a list of options: '---- Select an operation ----', 'JAX-WS service', 'Proxy service' (highlighted with a red dashed box and labeled with a red '(b)'), 'Simple service', 'JAX-WS client', 'Proxy client', and 'Simple client'. A red dashed box also highlights the 'Please select an operation' text above the dropdown.

Criação de um serviço “proxy” no Anypoint Studio



31) Abaixo, coloque os seguintes valores nos seguintes campos:

- Namespace: “http://www.fiap.com.br/ProductInformation”
- Service: “ProductInformation”

Estes itens devem estar de acordo com o WSDL que será referenciado na sequência.

CXF x Problems Console

General

Interceptors

Security

Advanced

Notes

Metadata

There are no errors.

Inbound Attributes

Binding ID:

Port:

Namespace: http://www.fiap.com.br/ProductInformation

Service: ProductInformation

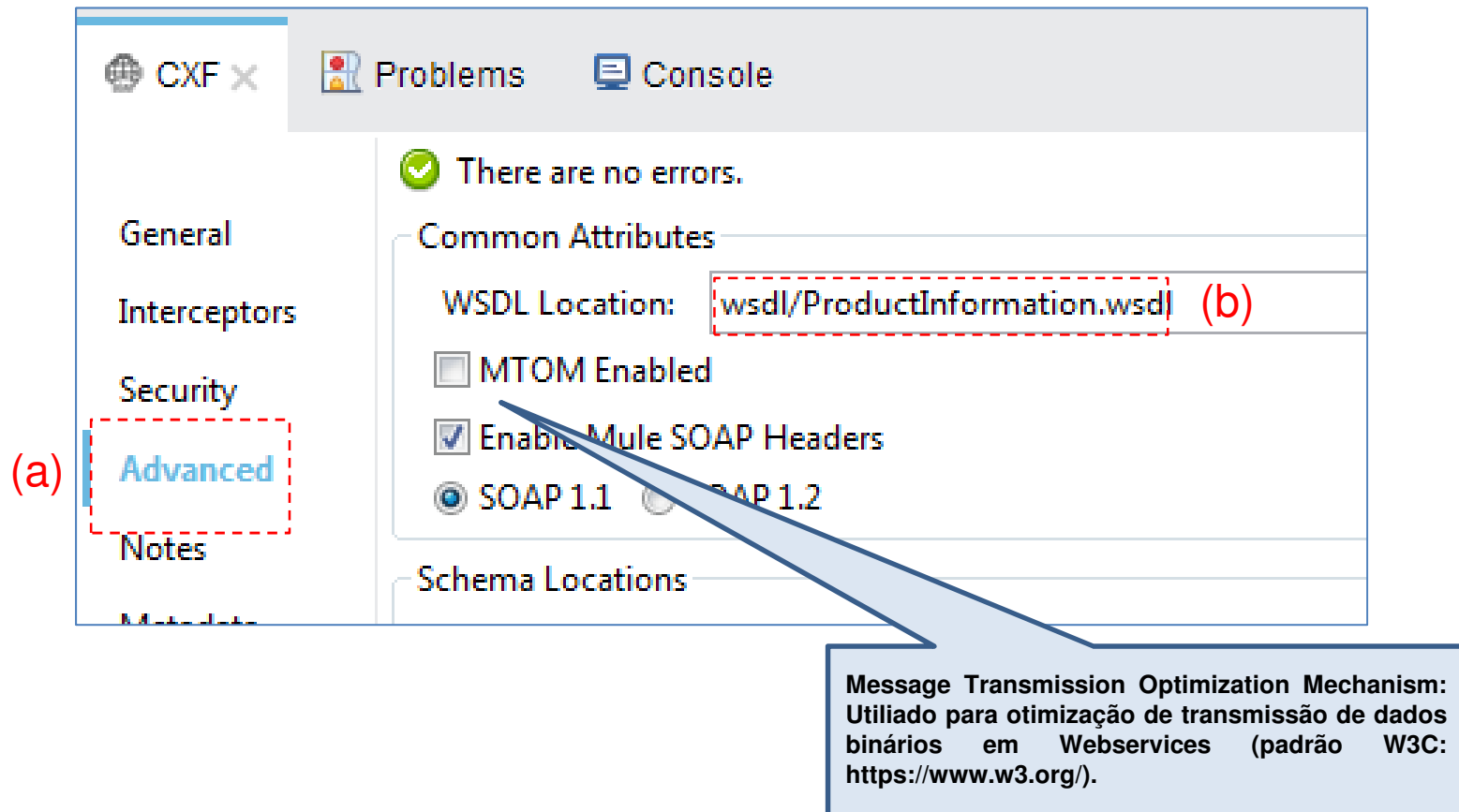
Service Class:

+ Generate from WSDL

☐ Validation Enabled

Criação de um serviço “proxy” no Anypoint Studio

32) Clique na aba “Advanced” (a), depois coloque em “WSDL Location” (b) o valor: “wsdl/ProductInformation.wsdl” que é o wsdl que o serviço de proxy atenderá.



Criação de um serviço “proxy” no Anypoint Studio

33) Agora, vamos configurar o “HTTP”. Clique no componente “HTTP” (a), depois clique no ícone “+” verde para adicionar uma nova configuração do conector (b).

The screenshot displays the Anypoint Studio interface. At the top, a message flow diagram titled 'pproxytoproductinformationFlow' is shown. It contains three components: 'HTTP' (a globe icon), 'CXF' (a circle with 'SOAP' inside), and 'Web Service Consumer' (a circle with a double 'e' inside). Arrows indicate a flow from HTTP to CXF, and from CXF to Web Service Consumer. The 'HTTP' component is highlighted with a red dashed box and labeled '(a)'. Below the diagram, the 'Error handling' section is visible. The bottom panel shows the 'HTTP' connector configuration. The 'General' tab is selected, and a red dashed box highlights the 'Connector Configuration' dropdown menu, which has a green plus icon next to it, labeled '(b)'. The configuration fields include 'Path' (set to '/'), 'Allowed Methods', 'Status Code', 'Reason', and a 'Disable properties' checkbox.

Criação de um serviço “proxy” no Anypoint Studio

34) Na tela “HTTP Listener Configuration” preencha os campos abaixo (a) com as seguintes informações:

- Host: localhost
- Port: 8085
- Base Path: productInfo/proxy

As demais informações,
Deixe como estão. Depois
clique em “OK”

NÃO ESQUEÇA DE SALVAR!!!



Global Element Properties

HTTP Listener Configuration
Create reusable HTTP listener

General TLS/SSL Notes

Generic

Name: HTTP_Listener_Configuration

URL Configuration

Protocol: ☒ HTTP (Default) ☐ HTTPS

Host: localhost

Port: 8085

Base Path: productInfo/proxy

Threading Profile Settings
Define threading profile behavior

HTTP x Console

There are no errors.

General

Advanced

Notes

Metadata

Display Name: HTTP

General Settings

Connector Configuration: HTTP_Listener_Configuration

Basic Settings

Path: /

Criação de um serviço “proxy” no Anypoint Studio



35) Agora sua primeira aplicação Mule está pronta. Basta testar.

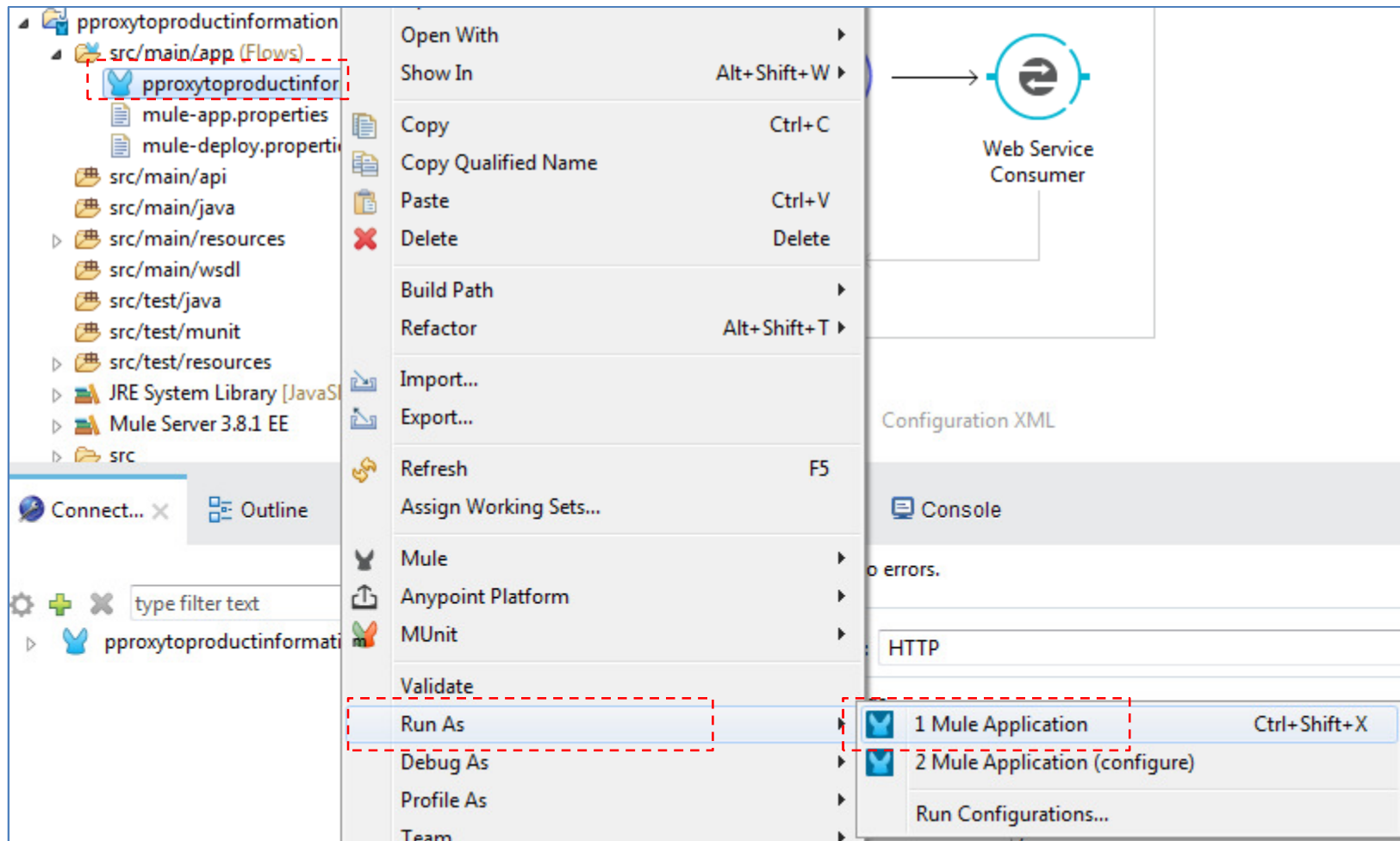
The screenshot displays the Anypoint Studio interface. At the top, a message flow diagram titled 'pproxytoproductinformationFlow' is shown. It consists of three main components: an HTTP connector (represented by a globe icon), a CXF connector (represented by a globe icon with 'SOAP' text), and a Web Service Consumer (represented by a circular icon with a double arrow). Arrows indicate the flow from HTTP to CXF, and then to the Web Service Consumer. Below the main flow, there is an 'Error handling' section with a globe icon. Below the diagram, there are tabs for 'Message Flow', 'Global Elements', and 'Configuration XML'. The 'HTTP' connector is selected, and its configuration panel is visible on the right. The panel has tabs for 'General', 'Advanced', 'Notes', and 'Metadata'. The 'General' tab is active, showing a green checkmark and the text 'There are no errors.' Below this, the 'Display Name' is set to 'HTTP'. Under 'General Settings', the 'Connector Configuration' is set to 'HTTP_Listener_Configuration'. Under 'Basic Settings', the 'Path' is set to '/' and the 'Allowed Methods' field is empty.

Testando o serviço “Proxy” no Mule Runtime

Testando o serviço “Proxy” no Mule Runtime



36) Para testar, clique com o botão direito em “pproxytoproductinformation.xml” e vá até “Run As”->“Mule Application”.



Testando o serviço “Proxy” no Mule Runtime



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

The screenshot shows the Mule IDE interface with the 'Console' tab selected. The console output displays the Mule runtime startup logs, including the deployment status of the application. The application name 'pproxytoproductinformation' is listed with the status 'DEPLOYED'.

```
pproxytoproductinformation. [Mule Applications] C:\Program Files (x86)\Java\jdk1.7.0_79\bin\javaw.exe (20 de out de 2016 08:02:05)
INFO 2016-10-20 08:02:17,070 [main] org.mule.module.launcher.StartupSummaryDeploymentListener:
*****
*      - - + DOMAIN + - -      *      - - + STATUS + - - *
*****
* default                      * DEPLOYED                      *
*****

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

Testando o serviço “Proxy” no Mule Runtime



38) Abra uma nova aba no browser e acesse:
“<http://127.0.0.1:8085/productInfo/proxy?wsdl>”

The screenshot shows a web browser window with two tabs. The active tab is titled "127.0.0.1:8085/productInfo/proxy?wsdl". The address bar shows the URL "127.0.0.1:8085/productInfo/proxy?wsdl". The main content area displays a message: "This XML file does not appear to have any style information associated with it. The document tree is shown below." Below this message is the XML document tree for the WSDL. The XML is a WSDL 1.1 document with the following structure:

```
<?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:prdt="http://www.fiap.com.br/product" name="ProductInformation" targetNamespace="http://www.fiap.com.br/ProductInformation" >
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.fiap.com.br/ProductInformation" >
      <xsd:import namespace="http://www.fiap.com.br/product" schemaLocation="http://127.0.0.1:8085/productInfo/product.xsd" />
      <xsd:element name="GetProduct">
        <xsd:complexType base="xsd:string">
          <xsd:sequence base="xsd:string" maxOccurs="1" minOccurs="1">
            <xsd:element name="ProductIn" type="prdt:ProductInType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GetProductResponse">
        <xsd:complexType base="xsd:string">
          <xsd:sequence base="xsd:string" maxOccurs="1" minOccurs="1">
            <xsd:element name="ProductOut" type="prdt:ProductOutType" />
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="GetProductRequest">
    <wsdl:part element="tns:GetProduct" name="parameters" />
  </wsdl:message>
  <wsdl:message name="GetProductResponse">
    <wsdl:part element="tns:GetProductResponse" name="parameters" />
  </wsdl:message>
  <wsdl:port name="ProductInfoProxy" type="wsdl:http" binding="wsdl:http:productInfoProxy" />
  <wsdl:binding name="productInfoProxy" type="wsdl:http" >
    <wsdl:operation name="GetProduct" binding="wsdl:http:productInfoProxy" />
  </wsdl:binding>
</wsdl:definitions>
```

Testando o serviço “Proxy” no Mule Runtime



39) Reparem que é o mesmo serviço. A diferença está no “endpoint” diferente. Esse é o objetivo de um “proxy” de um ESB: Desacoplamento.

```
localhost:8088/ProductInformation?wsdl

This XML file does not appear to have any style information associated with it. The document tree is shown below.

<?xml version='1.0' encoding='UTF-8'>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://www.fiap.com.br/ProductInformation" xmlns:prd="http://www.fiap.com.br/product" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" name="ProductInformation" targetNamespace="http://www.fiap.com.br/ProductInformation">
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.fiap.com.br/ProductInformation">
      <xsd:import namespace="http://www.fiap.com.br/product" schemaLocation="?WSDL&interface=ProductInformationSOAP&part=product.xsd"/>
      <xsd:element name="GetProduct">
        <xsd:complexType base="xsd:string">
          <xsd:sequence>
            <xsd:element name="ProductIn" type="prd:ProductInType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GetProductResponse">
        <xsd:complexType base="xsd:string">
          <xsd:sequence>
            <xsd:element name="ProductOut" type="prd:ProductOutType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
</wsdl:definitions>
<wsdl:message name="GetProductResponse">
  <xsd:sequence>
    <xsd:element name="ProductOut" type="prd:ProductOutType"/>
  </xsd:sequence>
</wsdl:message>
```

```
localhost:8088/ProductIn x 127.0.0.1:8085/productin x
127.0.0.1:8085/productInfo/proxy?wsdl

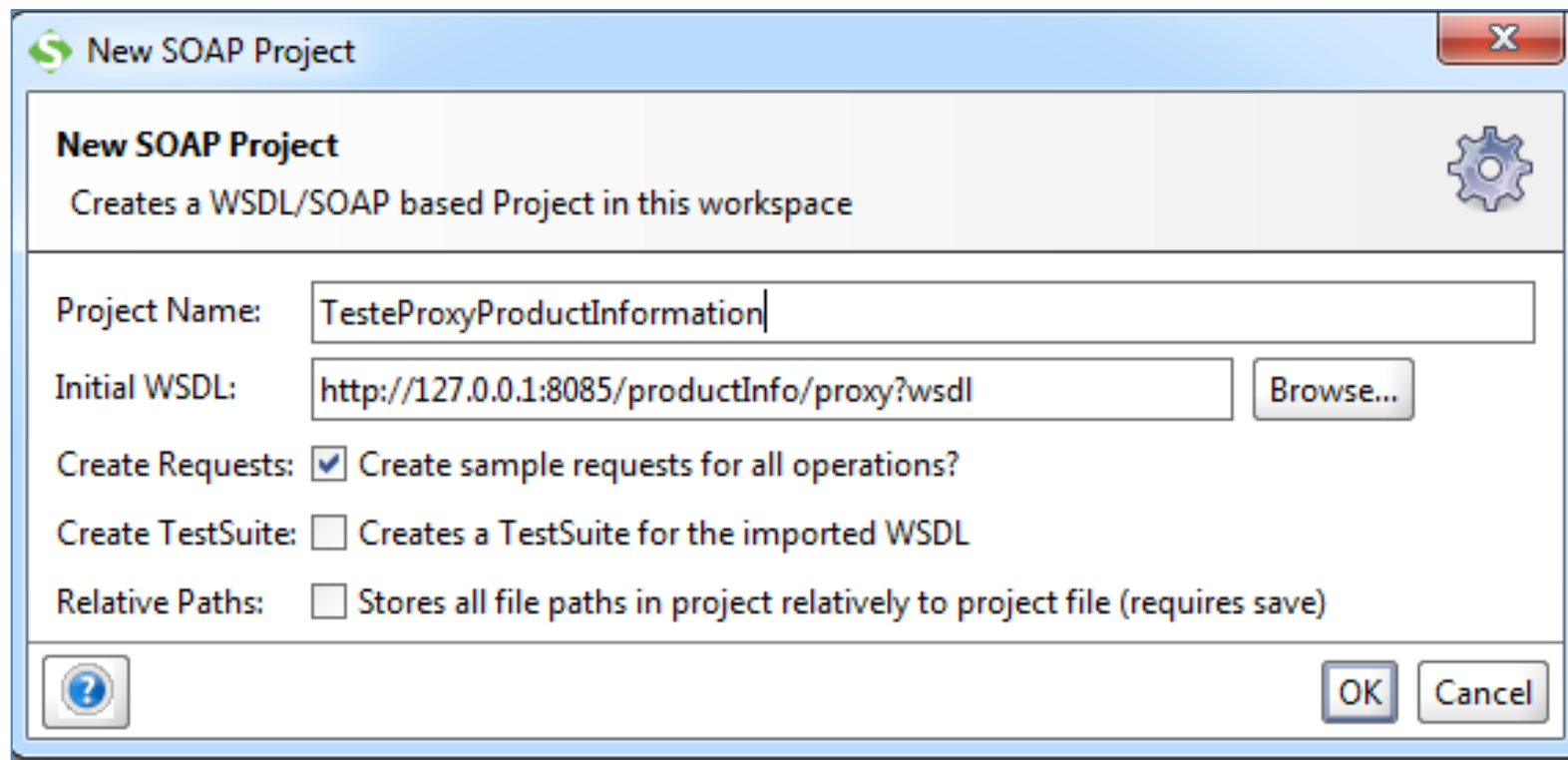
This XML file does not appear to have any style information associated with it. The document tree is shown below.

<?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:tns="http://www.fiap.com.br/ProductInformation" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:prd="http://www.fiap.com.br/product" name="ProductInformation" targetNamespace="http://www.fiap.com.br/ProductInformation">
  <wsdl:types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="http://www.fiap.com.br/ProductInformation">
      <xsd:import namespace="http://www.fiap.com.br/product" schemaLocation="http://127.0.0.1:8085/productInfo/proxy?xsd=../xsd/product.xsd"/>
      <xsd:element name="GetProduct">
        <xsd:complexType base="xsd:string">
          <xsd:sequence>
            <xsd:element name="ProductIn" type="prd:ProductInType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GetProductResponse">
        <xsd:complexType base="xsd:string">
          <xsd:sequence>
            <xsd:element name="ProductOut" type="prd:ProductOutType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
</wsdl:definitions>
<wsdl:message name="GetProductResponse">
  <xsd:sequence>
    <xsd:element name="ProductOut" type="prd:ProductOutType"/>
  </xsd:sequence>
</wsdl:message>
```

Testando o serviço “Proxy” no Mule Runtime



40) Agora, volte ao SoapUI e crie um novo projeto: “**TesteProxyProductInformation**” e coloque o wsdl do proxy: “**http://127.0.0.1:8085/productInfo/proxy?wsdl**”.



Testando o serviço “Proxy” no Mule Runtime



41) Salve e realize o teste.

The screenshot displays the Mule Studio IDE with a project named 'ProjetoMockProductInformation'. The left sidebar shows a project tree with folders 'ProductInformationSOAP' and 'TesteProxyProductInformation'. Under 'ProductInformationSOAP', there is a 'GetProduct' operation with a 'Request 1' message. Under 'TesteProxyProductInformation', there is a 'GetProduct' operation with a 'Request 1' message. The main workspace shows the 'Request 1' message in the 'Raw' view, which is a SOAP request. The URL bar indicates the endpoint 'http://127.0.0.1:8085/productInfo/proxy'. The request XML is as follows:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Header/>
  <soapenv:Body>
    <prod:GetProduct>
      <ProductIn>
        <prod1:Id_product>001</prod1:Id_product>
      </ProductIn>
    </prod:GetProduct>
  </soapenv:Body>
</soapenv:Envelope>
```

The response XML is also shown in the 'Raw' view, which is a SOAP response. The response XML is as follows:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <prod:GetProductResponse xmlns:prod1="http://www.fiap.com.br/product" xmlns:pr>
      <ProductOut>
        <prod1:Id_product>001</prod1:Id_product>
        <prod1:prodDesc>CAMISA SOCIAL</prod1:prodDesc>
        <prod1:Id_provider>05</prod1:Id_provider>
        <prod1:provDesc>BRASIL MODAS</prod1:provDesc>
        <prod1:category>MODA MASCULINA</prod1:category>
        <prod1:size>G</prod1:size>
        <prod1:price>150</prod1:price>
        <prod1:Qty>5</prod1:Qty>
      </ProductOut>
    </prod:GetProductResponse>
  </soap:Body>
</soap:Envelope>
```



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).