

# SWIFT MT/FIN Encoder

**Version 1.0**

©2011, Harry Liu (HarryLiu123@yahoo.com)

**SWIFT MT/FIN Encoder** is used to decode SWIFT MT/FIN data into XML format, or encode XML data into SWIFT MT/FIN format. It is built up on **SWIFT SDK** which provides the basic parsing. **SWIFT MT/FIN Encoder** also provides flexible extensions for schema validation, cross-fields semantic validation and text validation, through XSLT, Schematron, JavaScript, JRuby, java, etc, or some combinations of them, you can choose depending on your own preference and expertise.

**SWIFT MT/FIN Encoder** seamlessly integrates with the powerful Open Source platform **OpenESB** and **GlassFishESB** (JBI based). It can be used in different ways, e.g. in interactive **SWIFT Encoder Tester**, hook-up with dozens of **Open JBI components** (e.g. ftp, file, database, smtp, imap, pop3, http, soap, rest, jms, ldap, mq, msmq, snmp, scheduler etc), in BPEL engine, POJO engine.

**SWIFT MT/FIN Encoder** can also run outside **OpenESB / GlassFishESB**, e.g. in Java EE, Java SE.

Potentially, **SWIFT MT/FIN Encoder** can also be used in SeeBeyond/SUN/Oracle **Java CAPS** platform (the prior generations of **OpenESB**) in certain ways. Actually **SWIFT SAI** is a product built up on **Java CAPS**, combining with **SWIFT SDK**.

This PDF document (<http://swiftencoder.appspot.com/SwiftEncoder.pdf>) is a step-by-step tutorial to **SWIFT MT/FIN Encoder**.

There are also several screen casts showing the steps:

1. Basics <http://swiftencoder.appspot.com/SwiftEncoderBasics.html>
2. In BPEL <http://swiftencoder.appspot.com/SwiftEncoderInBPEL.html>
3. In POJO <http://swiftencoder.appspot.com/SwiftEncoderInPOJO.html>
4. In Java SE <http://swiftencoder.appspot.com/SwiftEncoderInJavaSE.html>
5. In Java EE <http://swiftencoder.appspot.com/SwiftEncoderInJavaEE.html>

There is also a page ([http://swiftencoder.appspot.com/MTXML\\_Tester.jsp](http://swiftencoder.appspot.com/MTXML_Tester.jsp)) which allows you to perform online decode/encode conversions between MT/FIN and XML.

## Table of Contents

1. Basics of SWIFT MT/FIN Encoder .....	4
2. Using SWIFT Encoder in BPEL .....	22
3. Using SWIFT Encoder in POJO.....	35
4. Using SWIFT Encoder in Java SE.....	46
5. Using SWIFT Encoder in Java EE.....	49

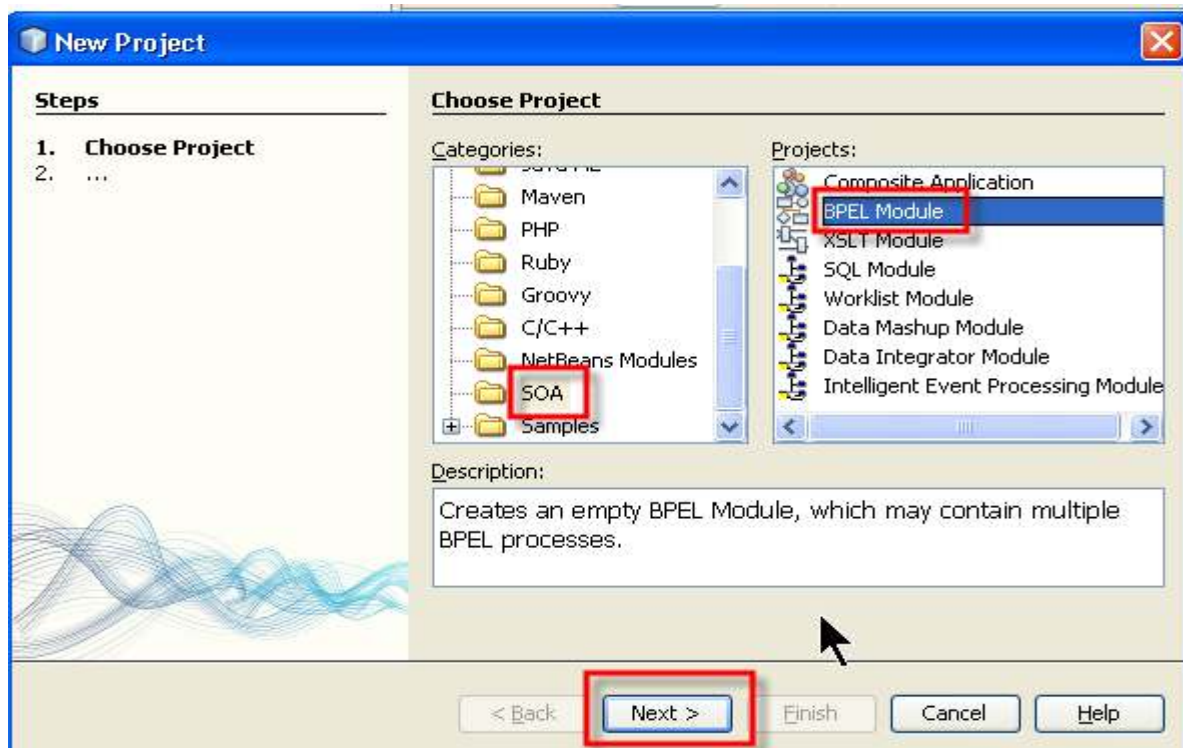
# 1. Basics of SWIFT MT/FIN Encoder

We'll go through basic functions of SWIFT Encoder, also practice SWIFT Encoder Tester. We'll show the extensions for cross-fields semantic validation and text validation with examples of XSLT, Schematron, java.

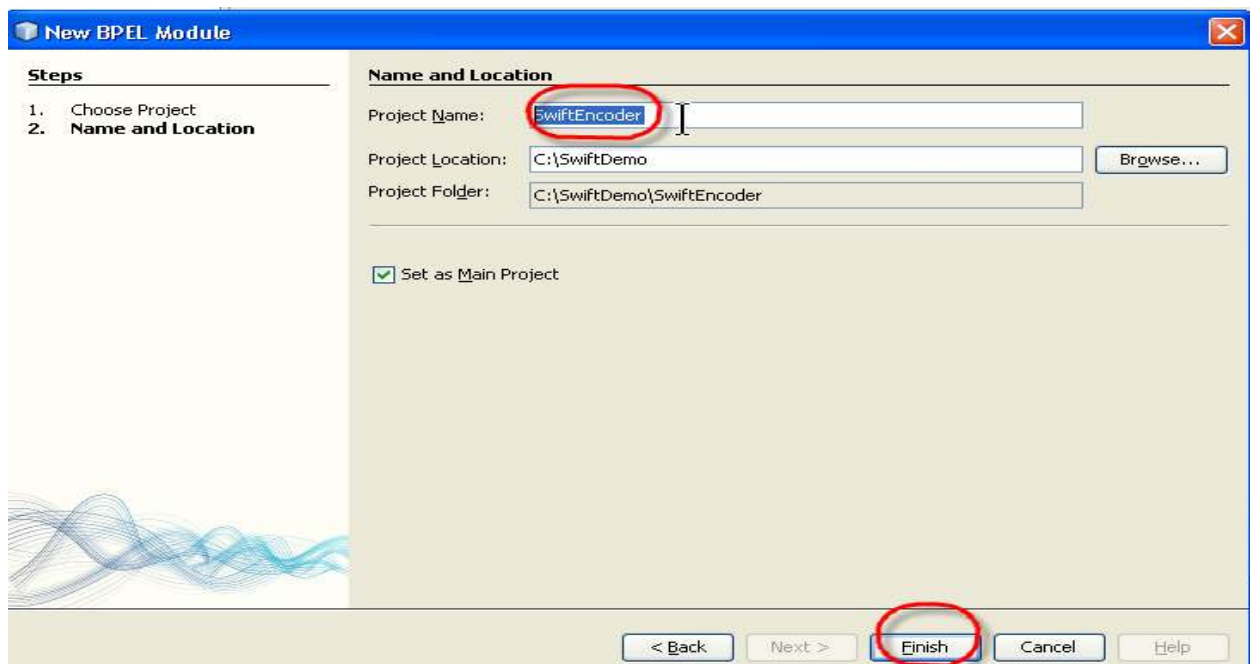
Here is a screen cast:

<http://swiftencoder.appspot.com/SwiftEncoderBasics.html>

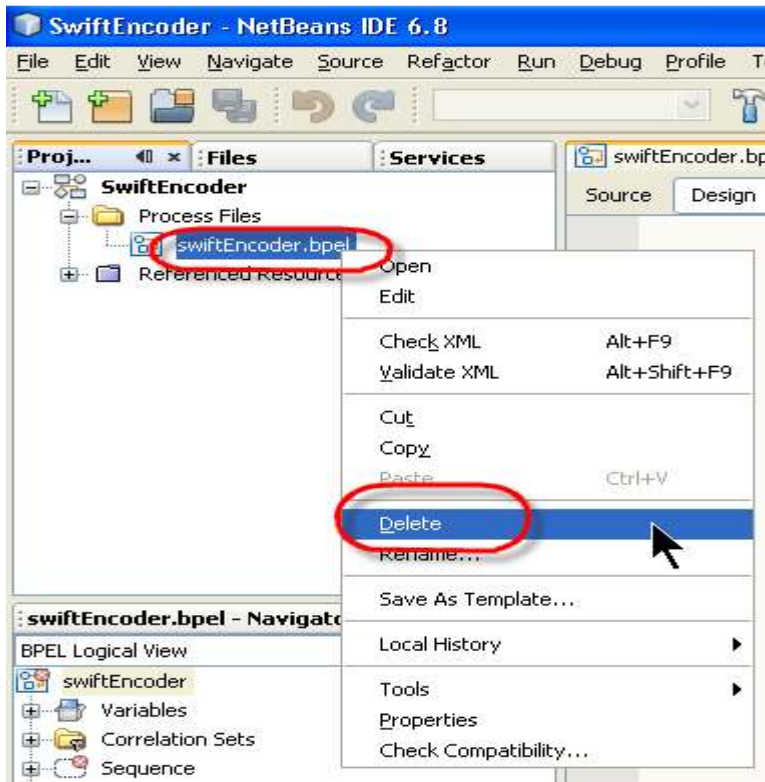
Create a project SOA/BPEL Module.



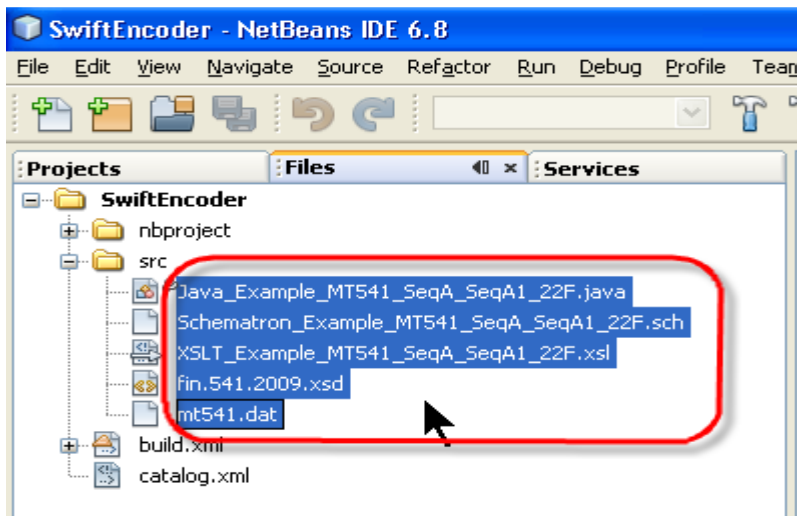
Name it as "SwiftEncoder"



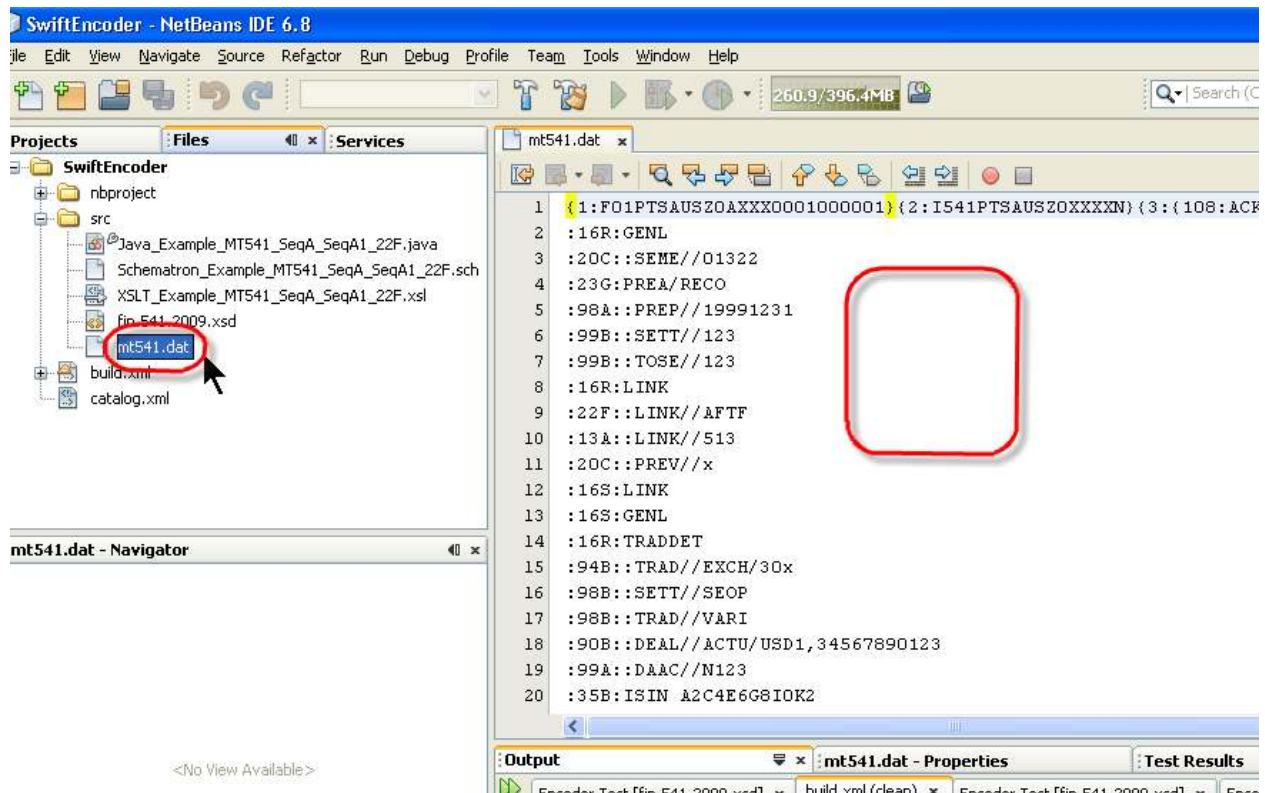
Delete the .bpel file because we don't need it.



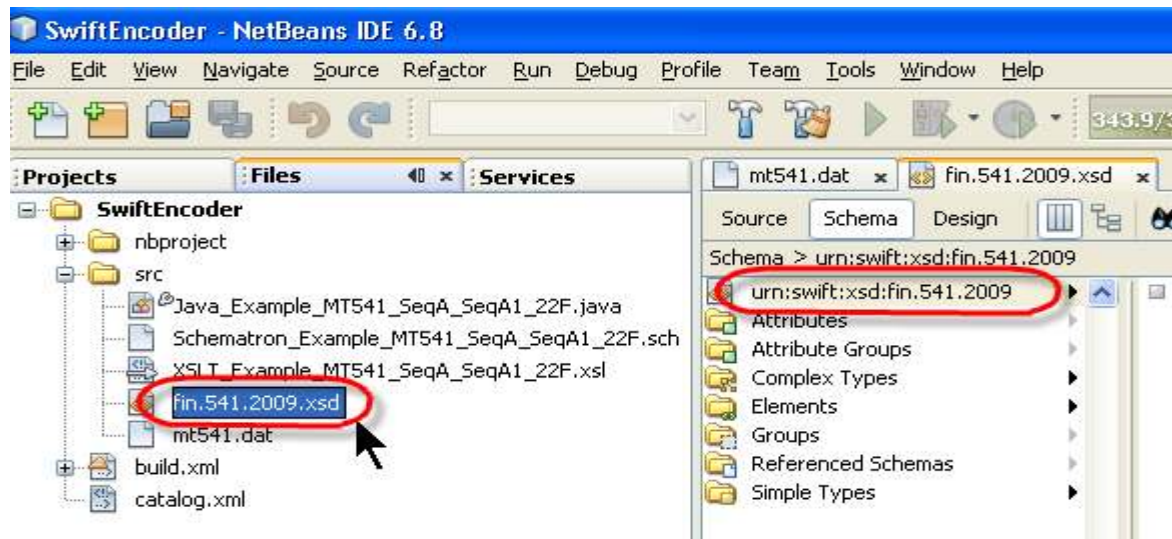
Copy some resource files into project, they will be used later.



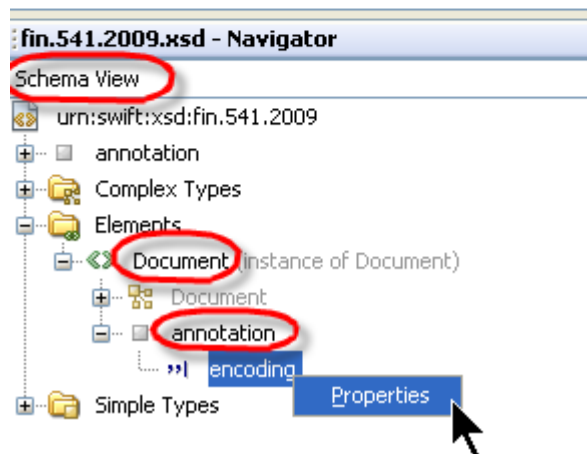
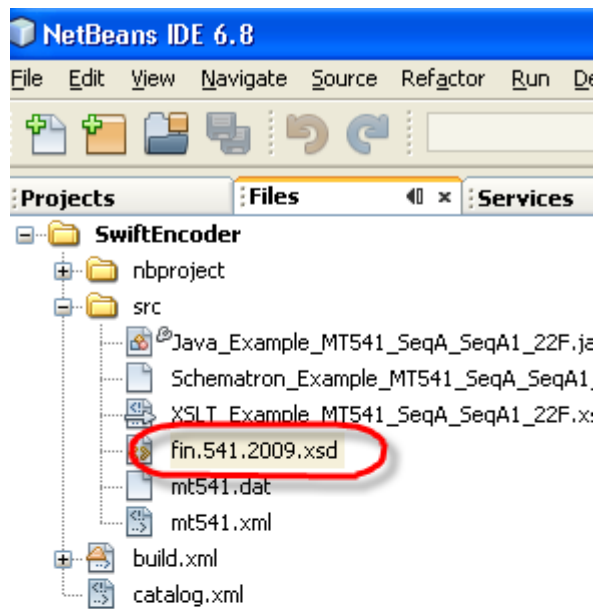
Here is an MT test file, [mt541.dat](#).



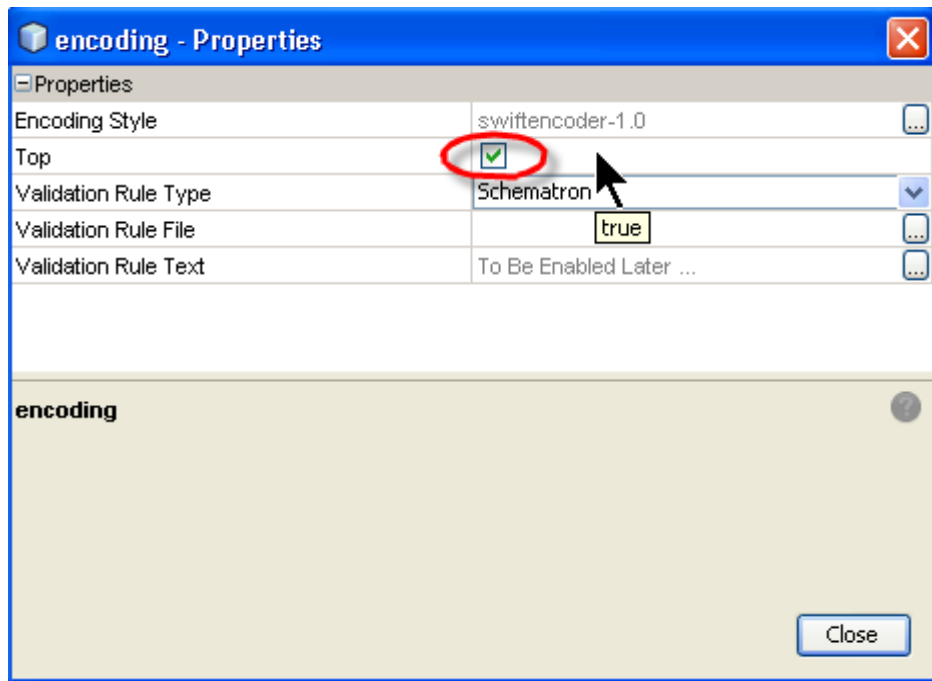
Here is a schema for [fin.541.2009.xsd](#).



Select schema file. In **Schema View**, navigate to **Document->annotation->encoding**. Right-click on “encoding”, select “Properties”.

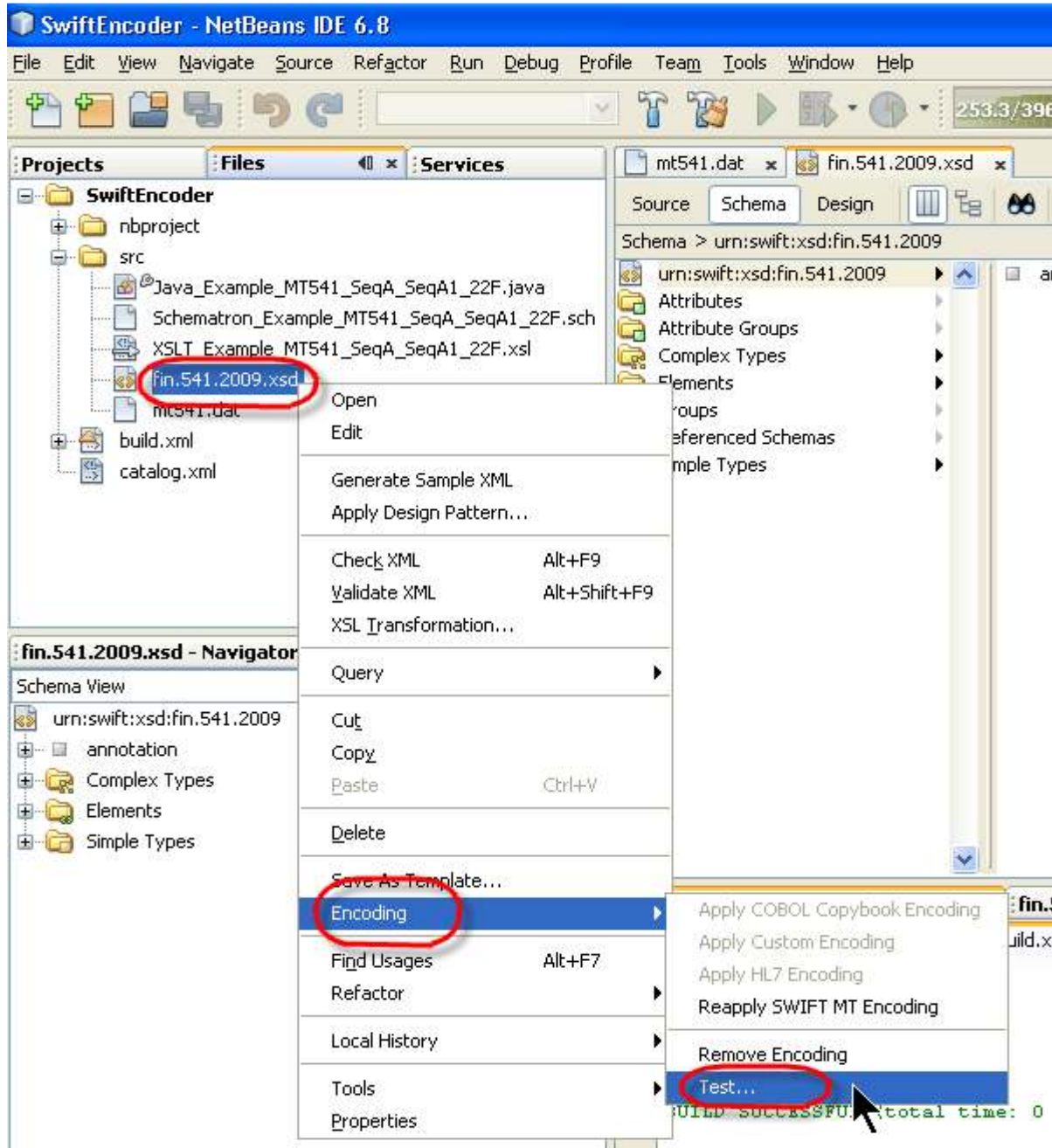


Check “Top” check box.

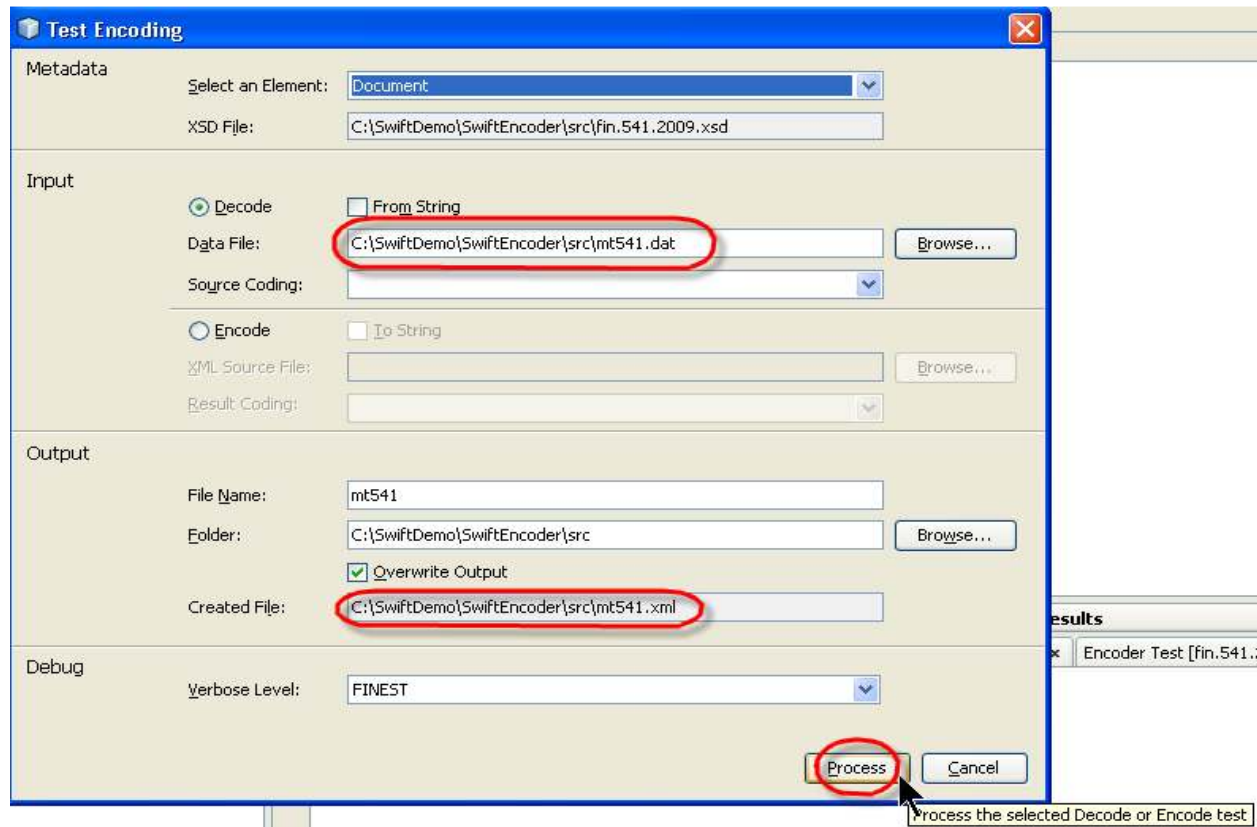


Save all.

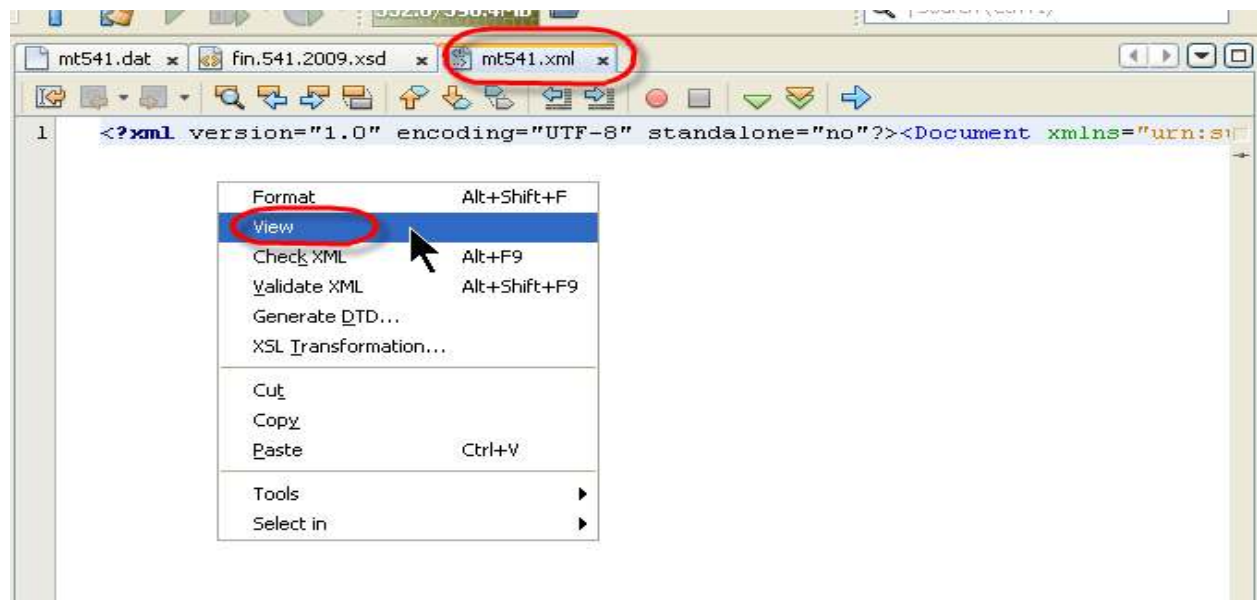
Right-click on schema “fin.541.2009.xsd”, choose “Encoding->Test”.



Test “Decode”. Select the input MT data file, mt541.dat. Specify the output XML file, mt541.xml. Click “Process” button.



Output mt541.xml is generated. Right-click, select “View”.



You can see view the XML file [mt541.xml](#), e.g. in a web-browser.

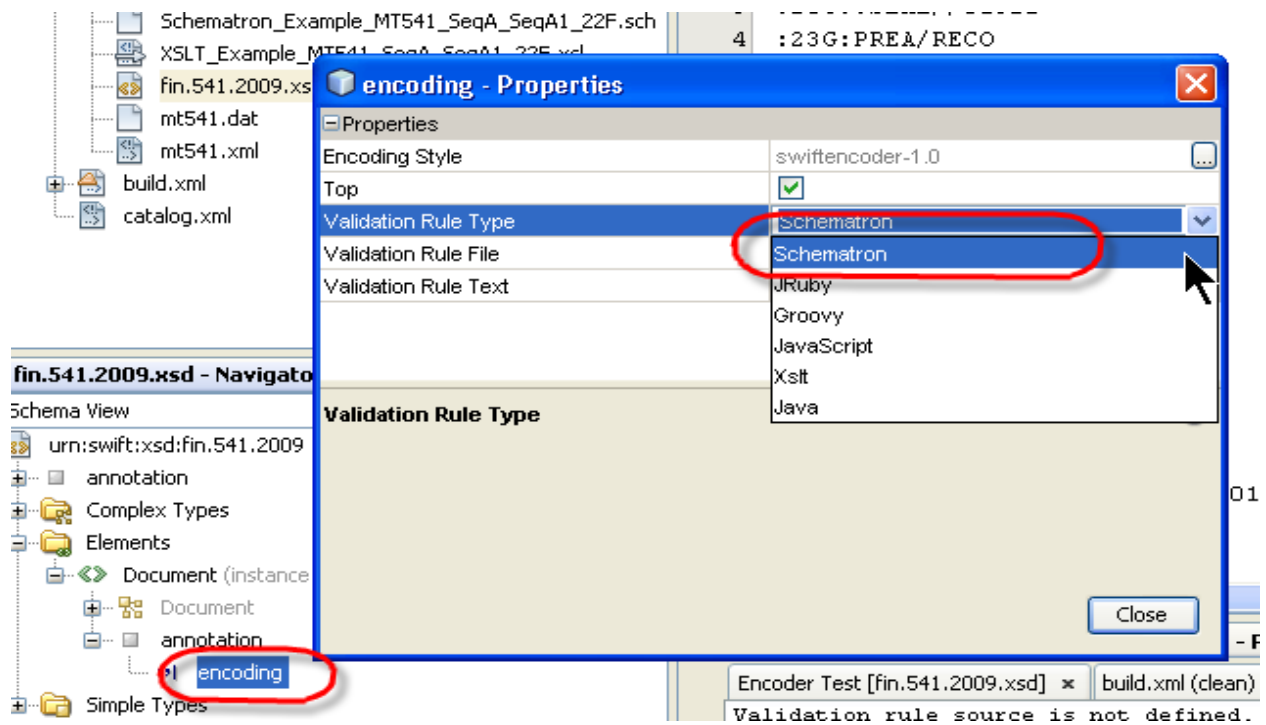


This XML file does not appear to have any style information associated with it.

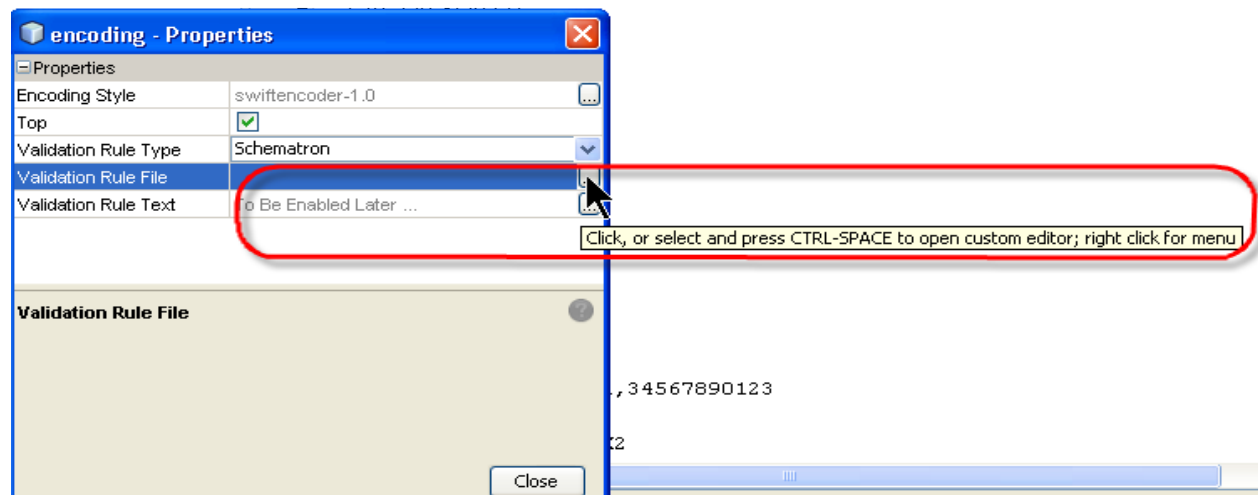
```
<Document xmlns="urn:swift:xsd:fin.541.2009">
  <MT541>
    <SeqA>
      <F16a_1>
        <!-- line=1 -->
        <F16R>GENL</F16R>
      </F16a_1>
      <F20a>
        <SEME>
          <!-- line=2 -->
          <F20C>
            <Reference>01322</Reference>
          </F20C>
        </SEME>
      </F20a>
      <F23a>
        <!-- line=3 -->
        <F23G>
          <Function>PREA</Function>
          <SubFunction>RECO</SubFunction>
        </F23G>
      </F23a>
      <F98a>
        <PREP>
          <!-- line=4 -->
          <F98A>
            <Date>19991231</Date>
          </F98A>
        </PREP>
      </F98a>
      <F99a>
        <SETT>
          <!-- line=5 -->
          <F99B>
            <Number>123</Number>
          </F99B>
        </SETT>
      </F99a>
    </SeqA>
  </MT541>
</Document>
```

Note: This sample MT data actually contains a semantic validation error, it is caught by default parsing/validation engine. It will be covered below.

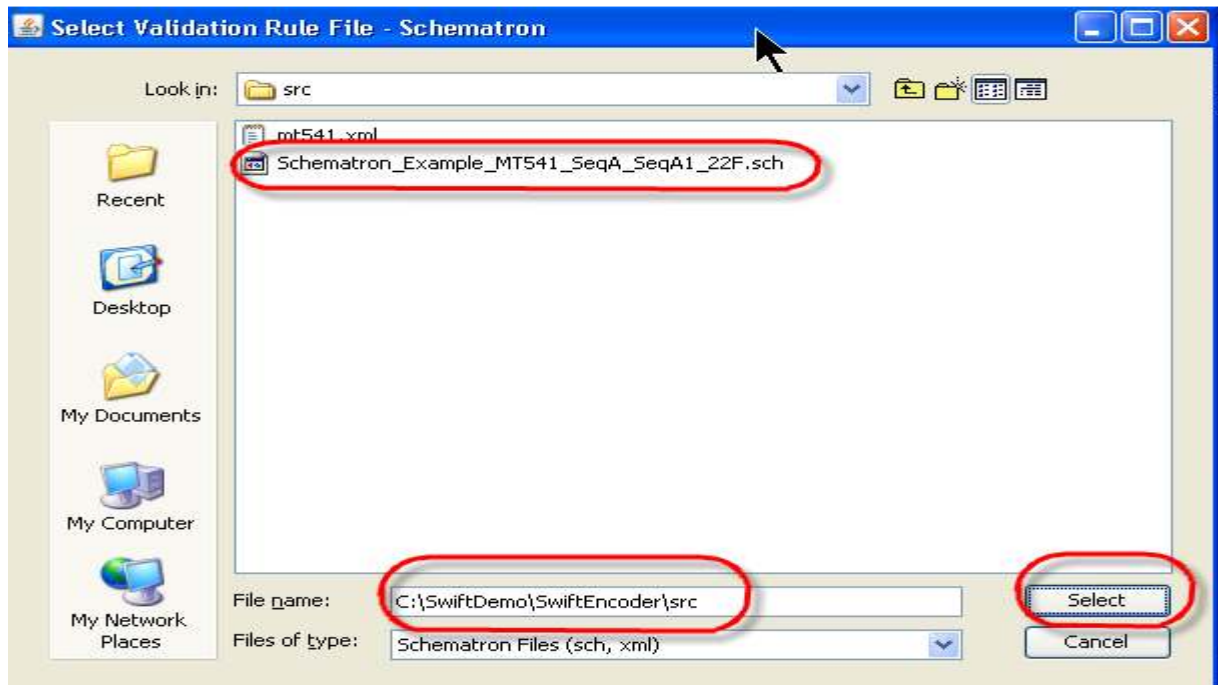
Some validation rule types are plugged. [Schematron](#), [XSLT](#), [JavaScript](#), [Java](#), etc ... I'll give some examples below.



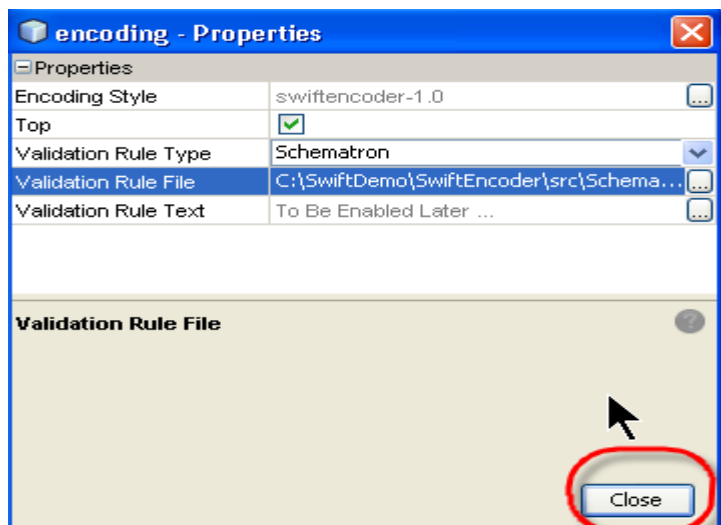
You provide the validation rules via files. For example, select “Schematron”, then choose file ...



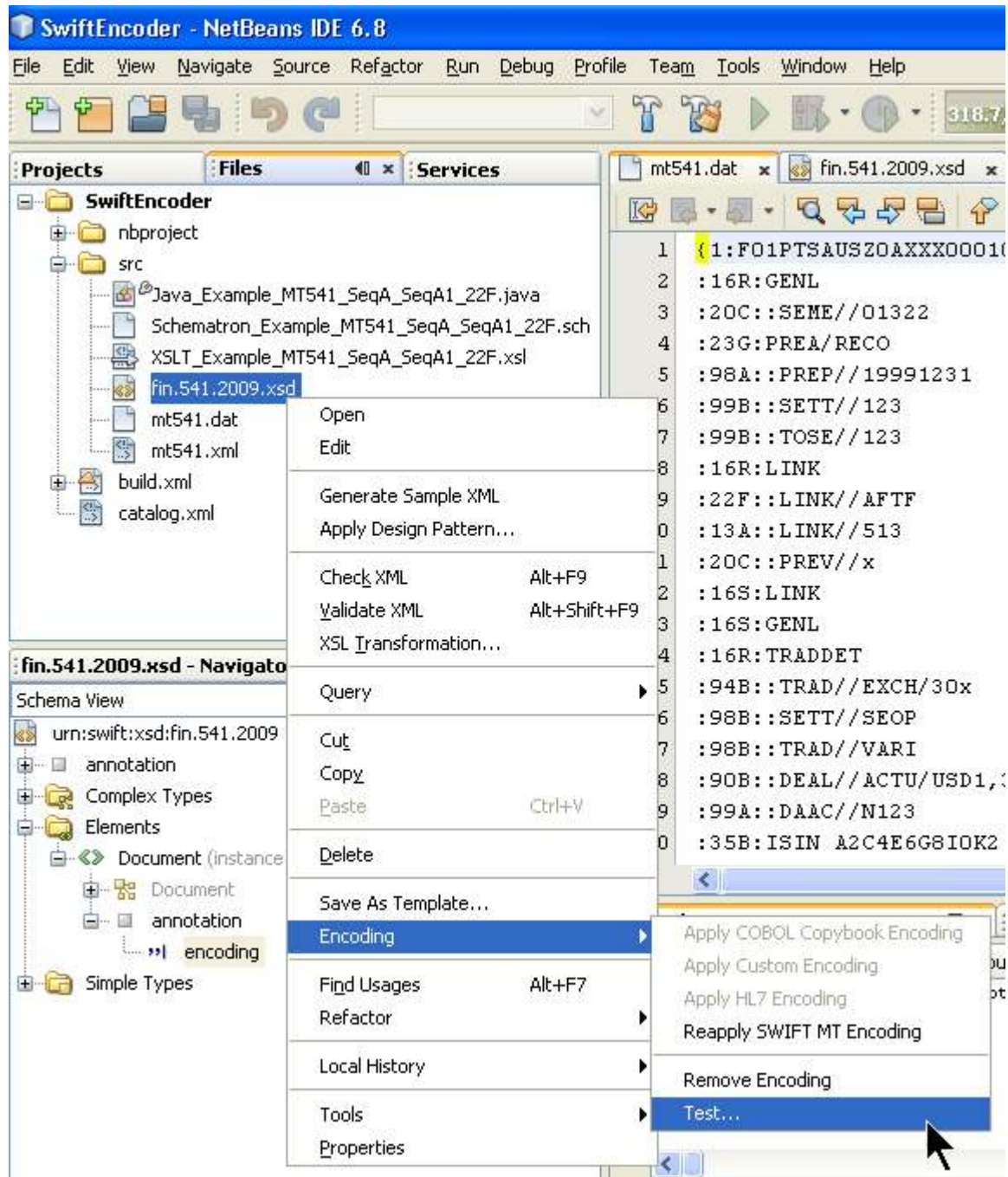
Select the sample Schematron file,  
“Schematron\_Example\_MT51\_SeqA\_SeqA1\_22F.sch”.



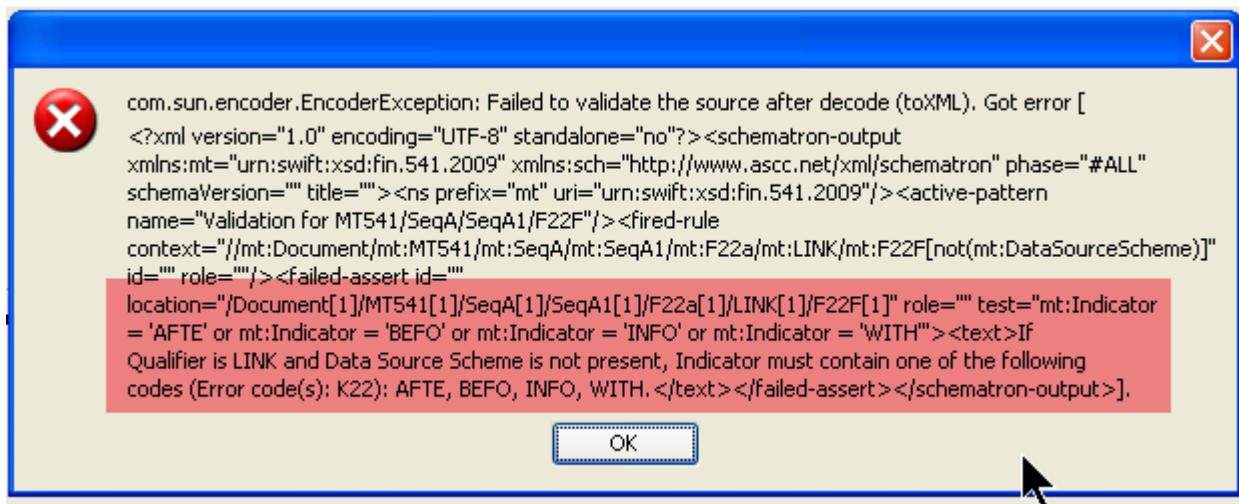
Finish the file selection.



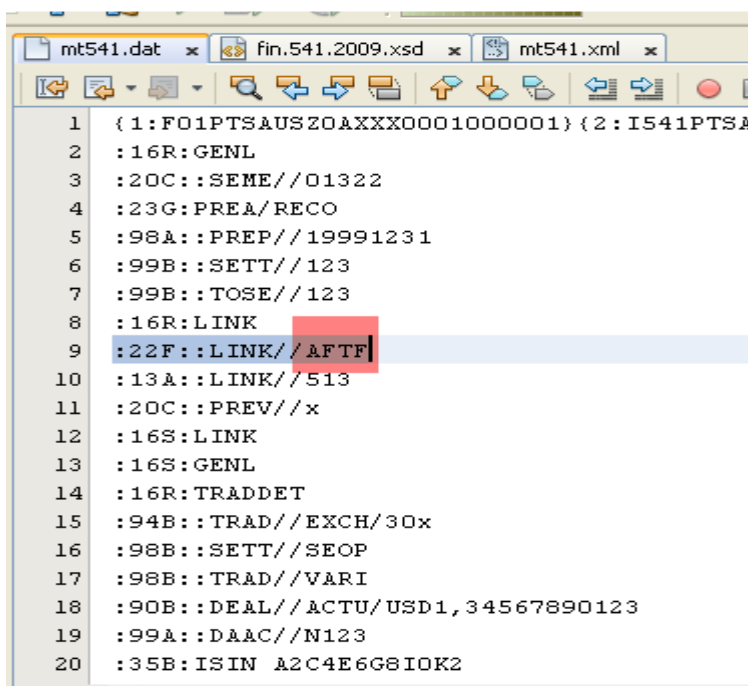
Save all and Test again ...



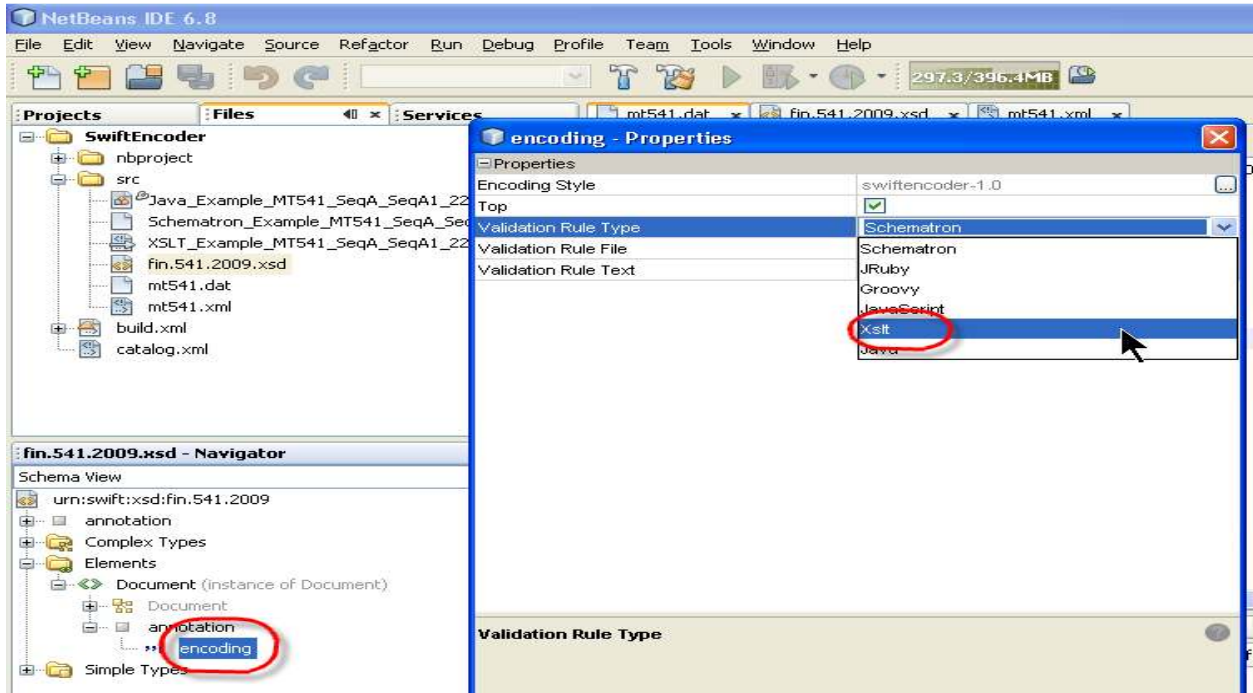
You see an error pop-up window. Schematron rules are taking effect. The data contains a semantic error, as described by the error message.



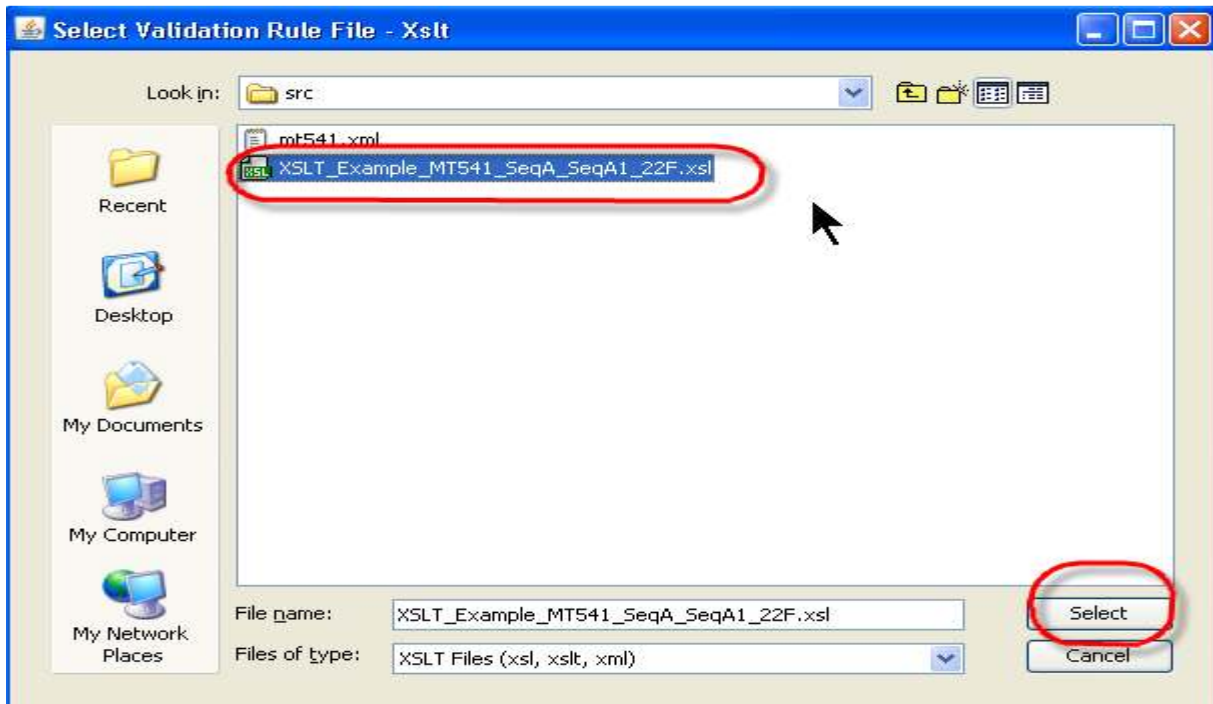
Checking back the MT data, locating the error point, "AFTF" is invalid.



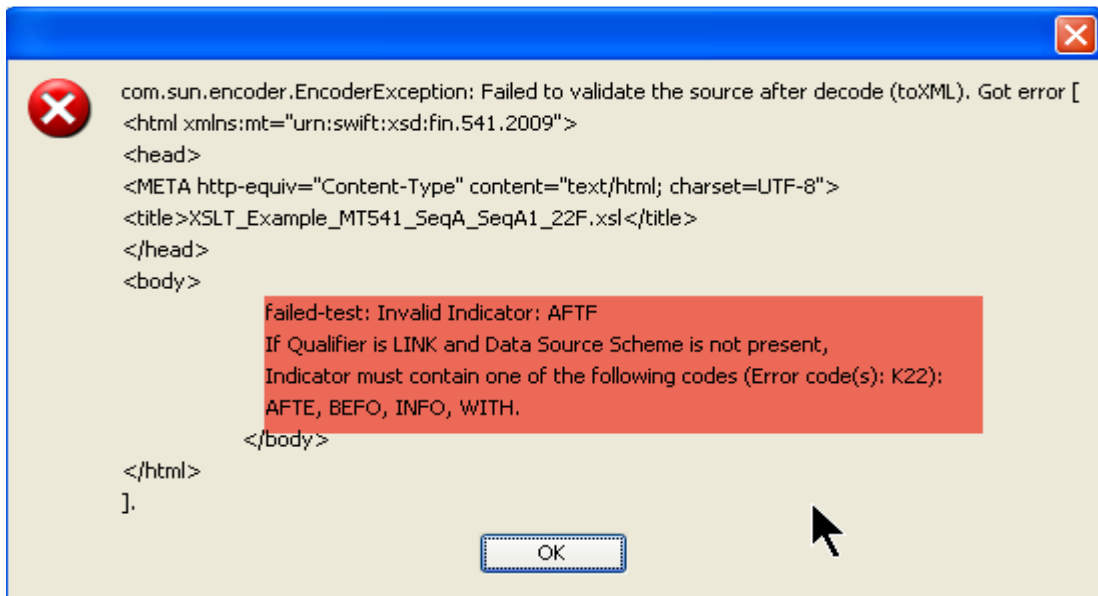
Try to use “XSLT” to validate it ...



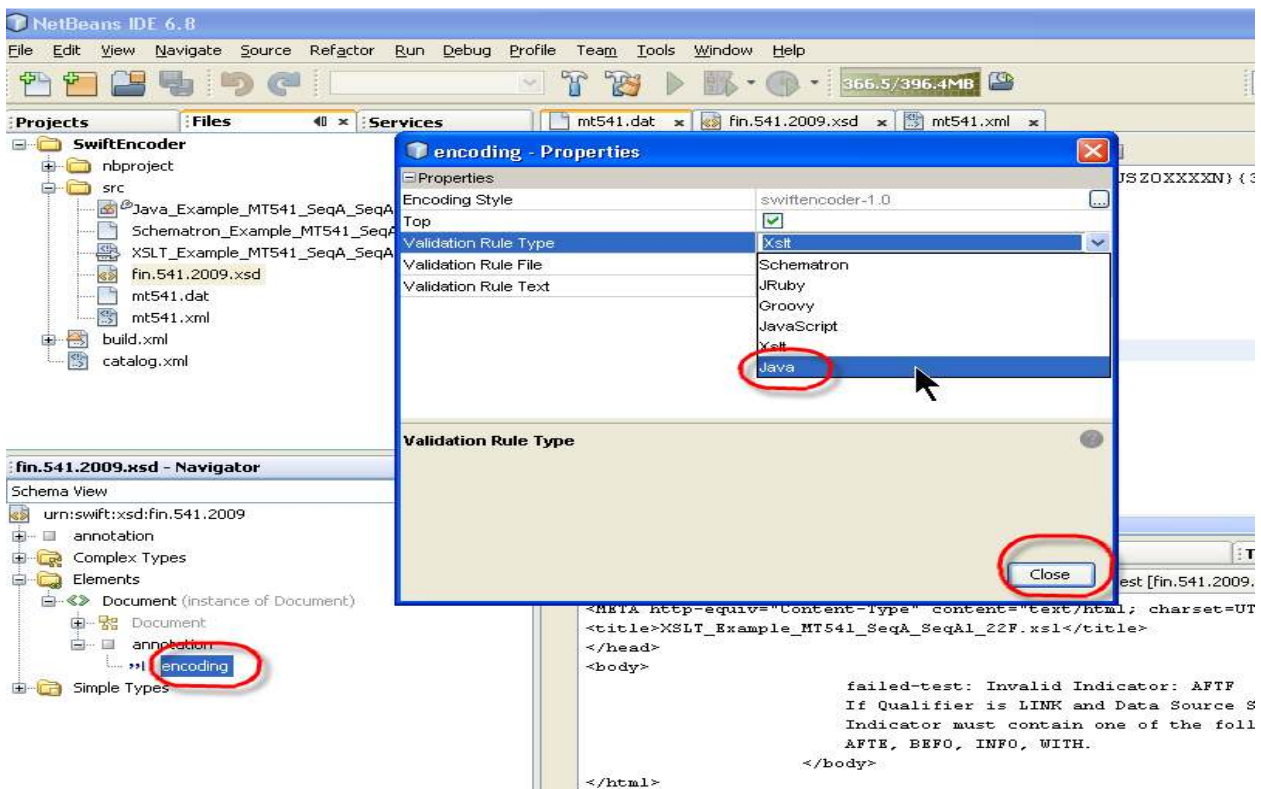
Select the provided XSLT sample code, ..., save, test.



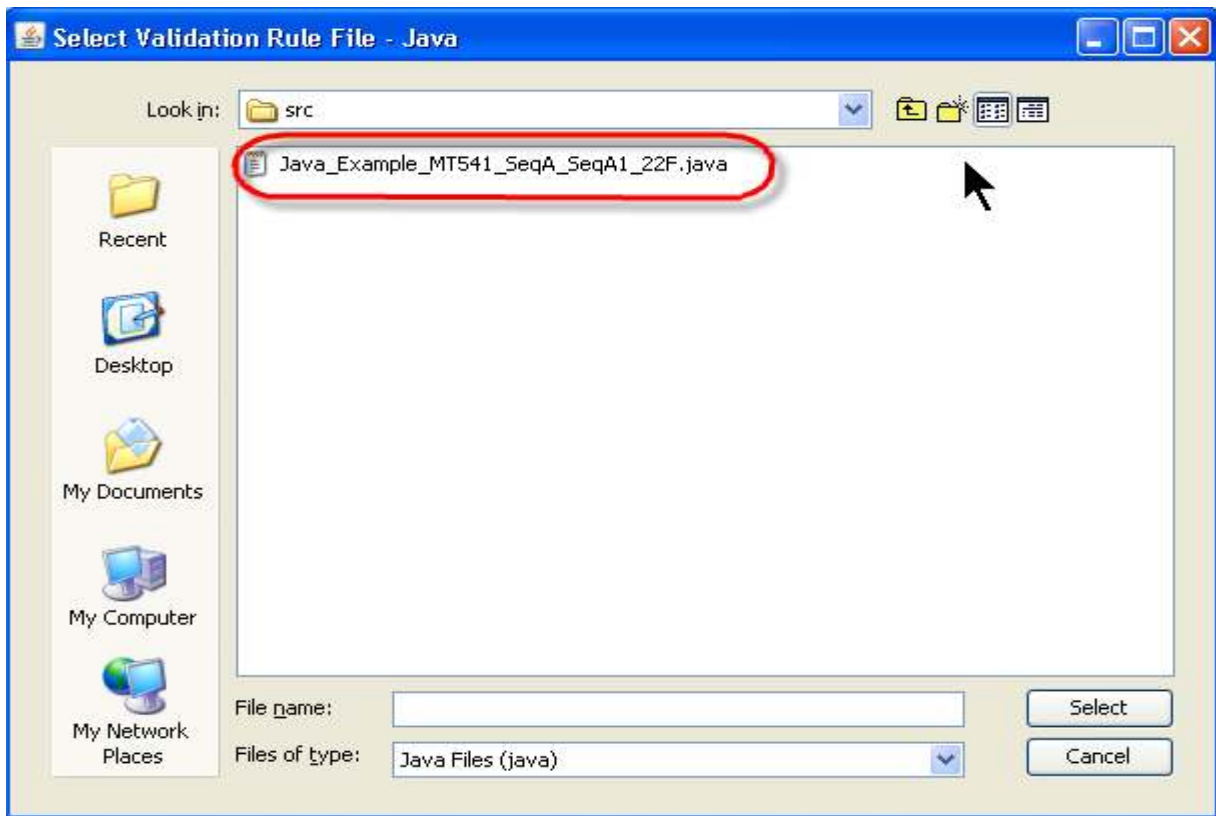
Good. You get an error pop-up window, XSLT rules are taking effect. You also can see the error description.



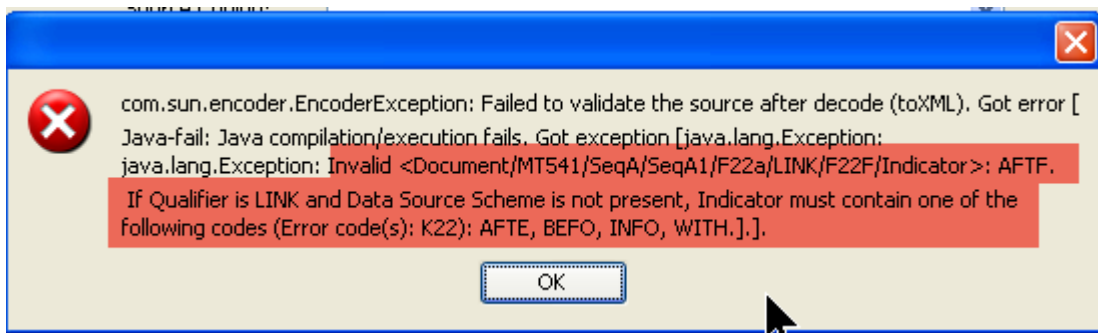
Let's try "Java" ...



Select the provided Java sample code, ..., save, test.



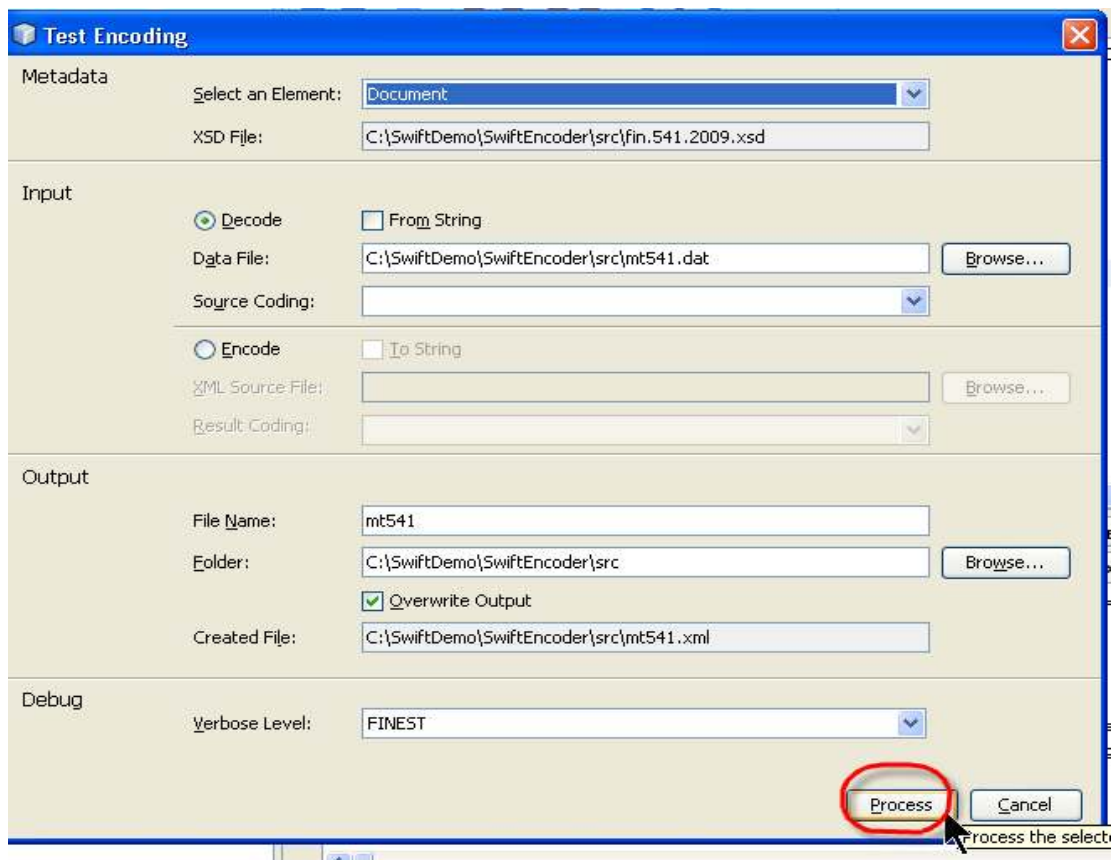
You'll see, Java also catches the data error.



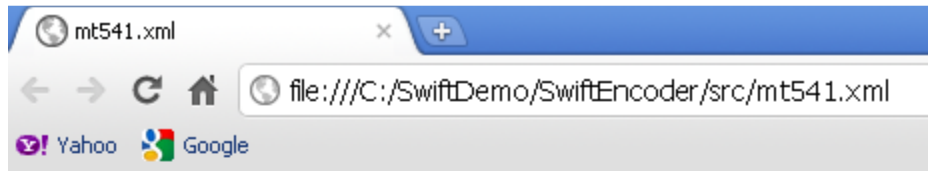
Let's correct the MT data error by changing "AFTF" to "AFTE".

```
mt541.dat x fin.541.2009.xsd x mt541.xml
1 {1:F01PTSAUSZ0AXXX0001000001}{2:I5-
2 :16R:GENL
3 :20C::SEME//01322
4 :23G:PREA/RECO
5 :98A::PREP//19991231
6 :99B::SETT//123
7 :99B::TOSE//123
8 :16R:LINK
9 :22F::LINK//AFTE
10 :13A::LINK//513
11 :20C::PREV//x
12 :16S:LINK
13 :16S:GENL
14 :16R:TRADDET
15 :94B::TRAD//EXCH/30x
16 :98B::SETT//SEOP
17 :98B::TRAD//VARI
18 :90B::DEAL//ACTU/USD1,34567890123
19 :99A::DAAC//N123
20 :35B:ISIN A2C4E6G8I0K2
```

Save the data. Test again ...



Ok. Now error goes away. XML output will be generated.



This XML file does not appear to have any style information associated with

```
▼ <Document xmlns="urn:swift:xsd:fin.541.2009">
  ▼ <MT541>
    ▼ <SeqA>
      ▼ <F16a_1>
        <!-- line=1 -->
        <F16R>GENL</F16R>
      </F16a_1>
      ▼ <F20a>
        ▼ <SEME>
          <!-- line=2 -->
          ▼ <F20C>
            <Reference>01322</Reference>
          </F20C>
        </SEME>
      </F20a>
      ▼ <F23a>
        <!-- line=3 -->
        ▼ <F23G>
          <Function>PREA</Function>
          <SubFunction>RECO</SubFunction>
        </F23G>
      </F23a>
```

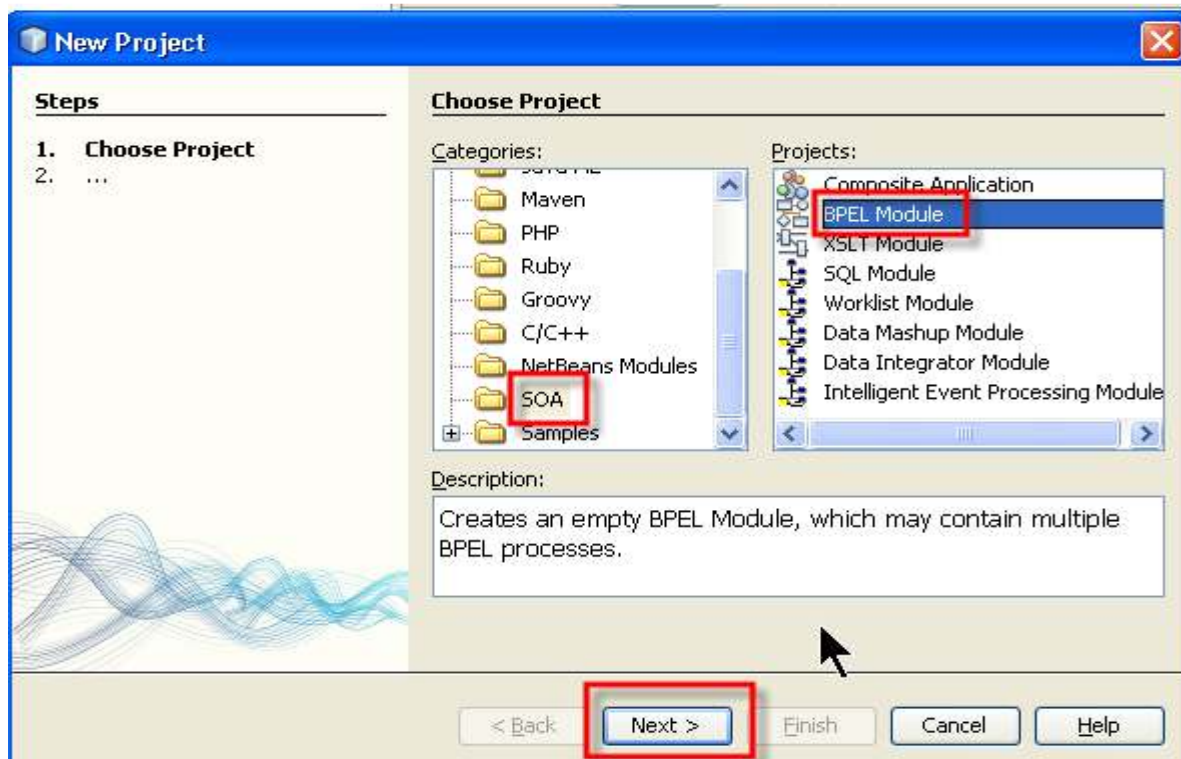
## 2. Using SWIFT Encoder in BPEL

Demonstrate how to use SWIFT Encoder in BPEL SE (Service Engine) by use of a simple example.

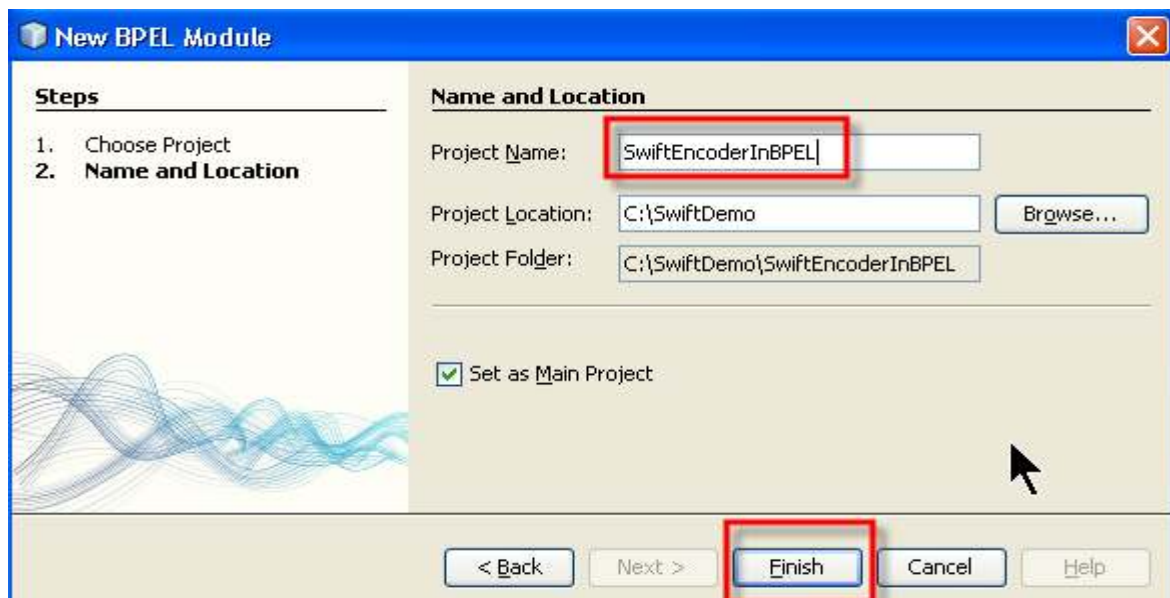
Here is a screen cast:

<http://swiftencoder.appspot.com/SwiftEncoderInBPEL.html>

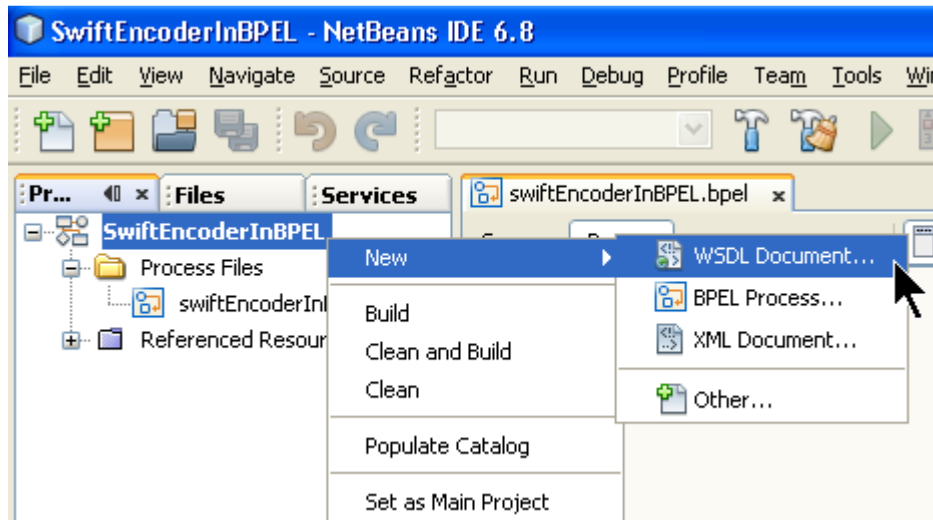
Create an BPEL project.



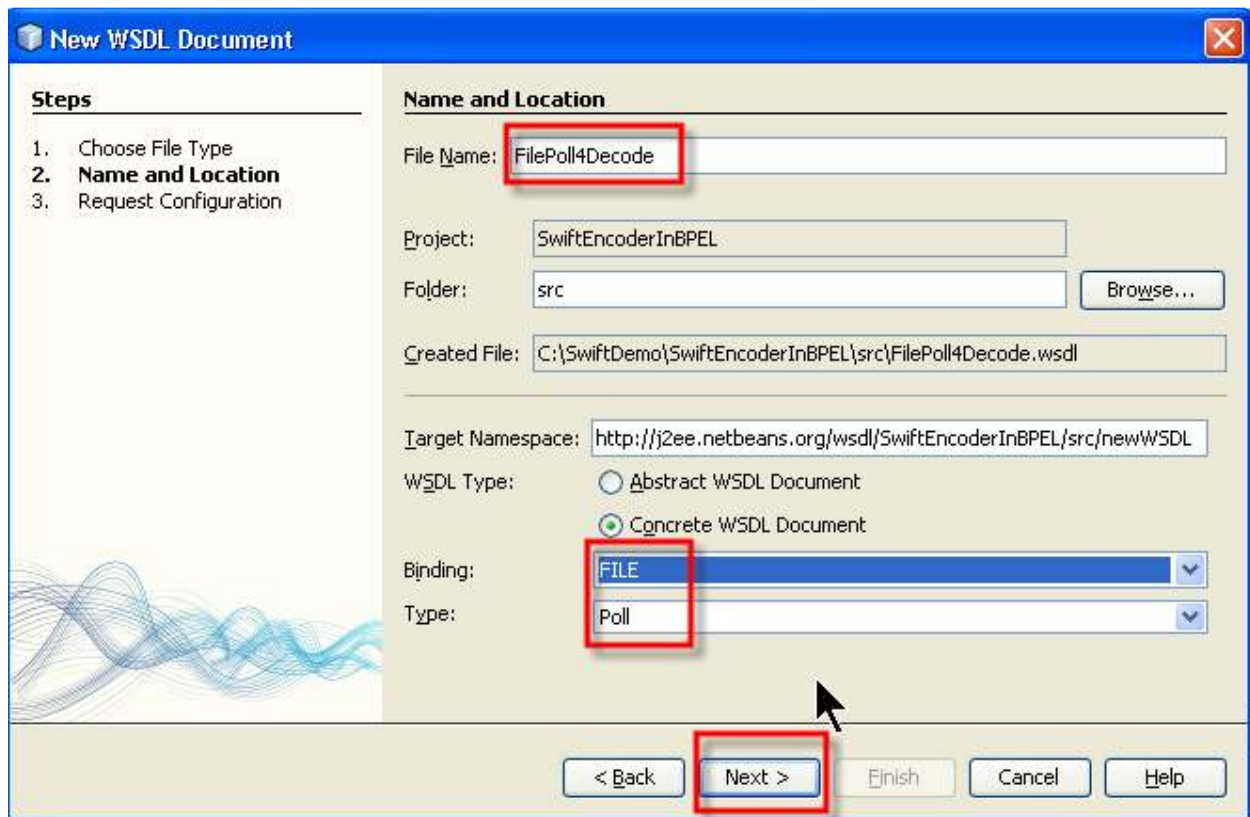
Name it as “SwiftEncoderInBPEL”



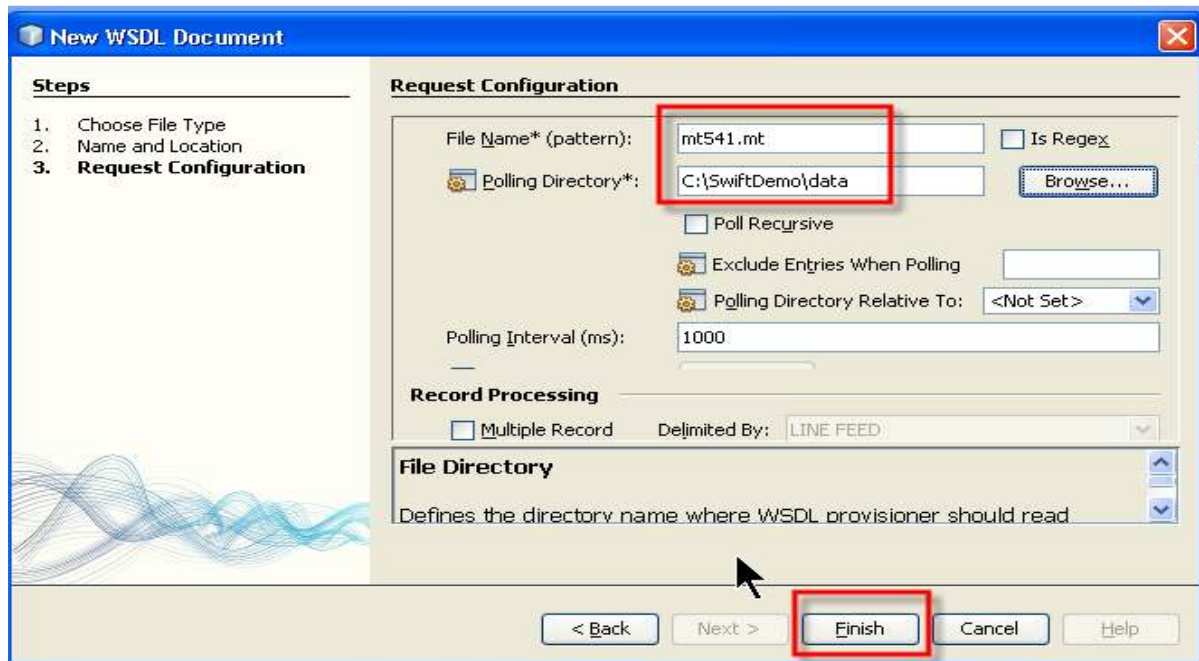
Create an WSDL ...



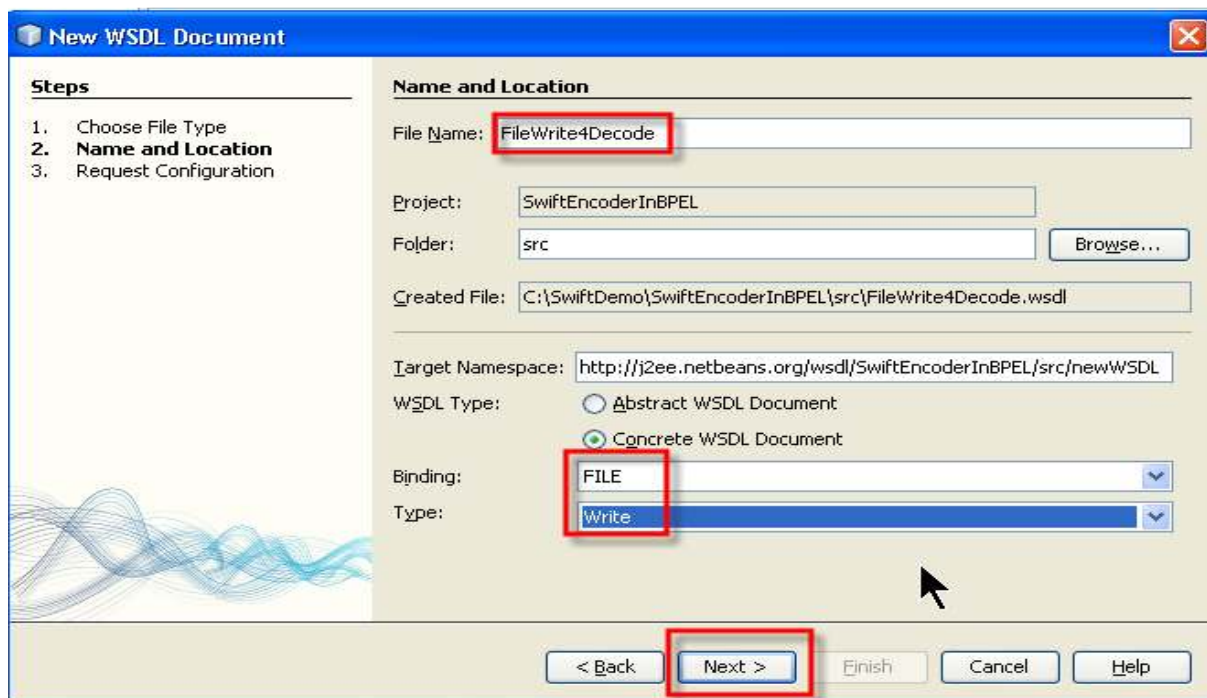
Name it as “FilePoll4Decode”, select “FILE”, “Poll” ...



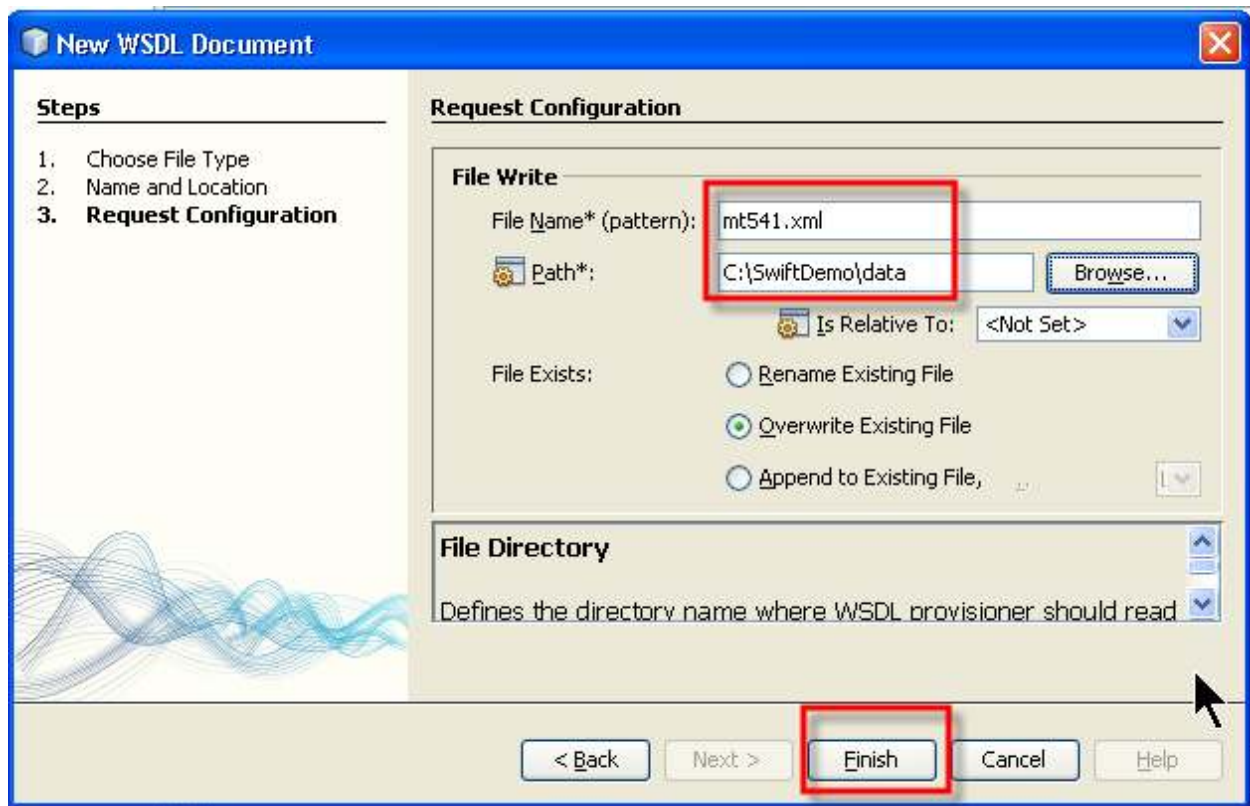
Specify MT file name ([mt541.mt](#)) and directory.



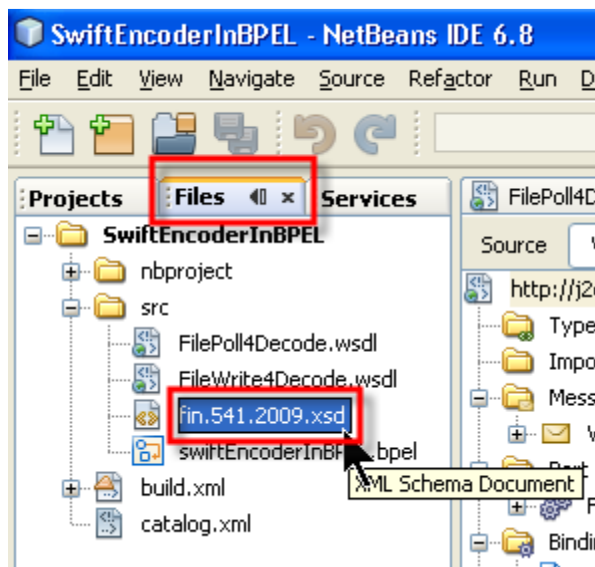
Create another WSDL for “FILE” ... “Write”, name it as “FileWrite4Decode”.



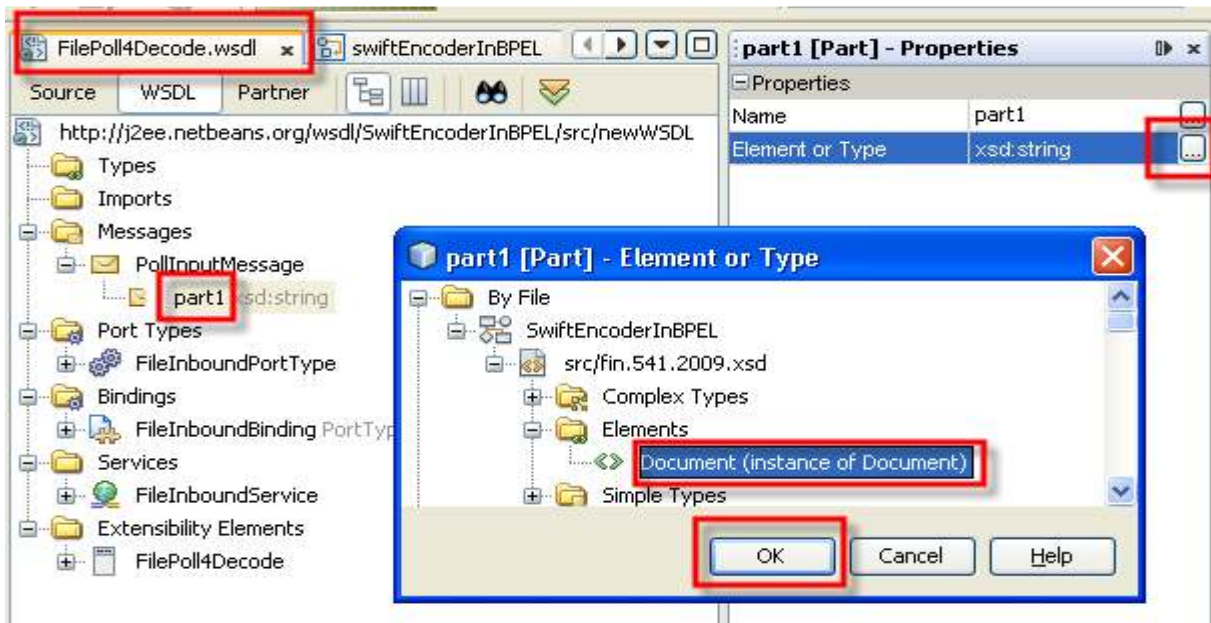
Specify XML file name ([mt541.xml](#)) and directory.



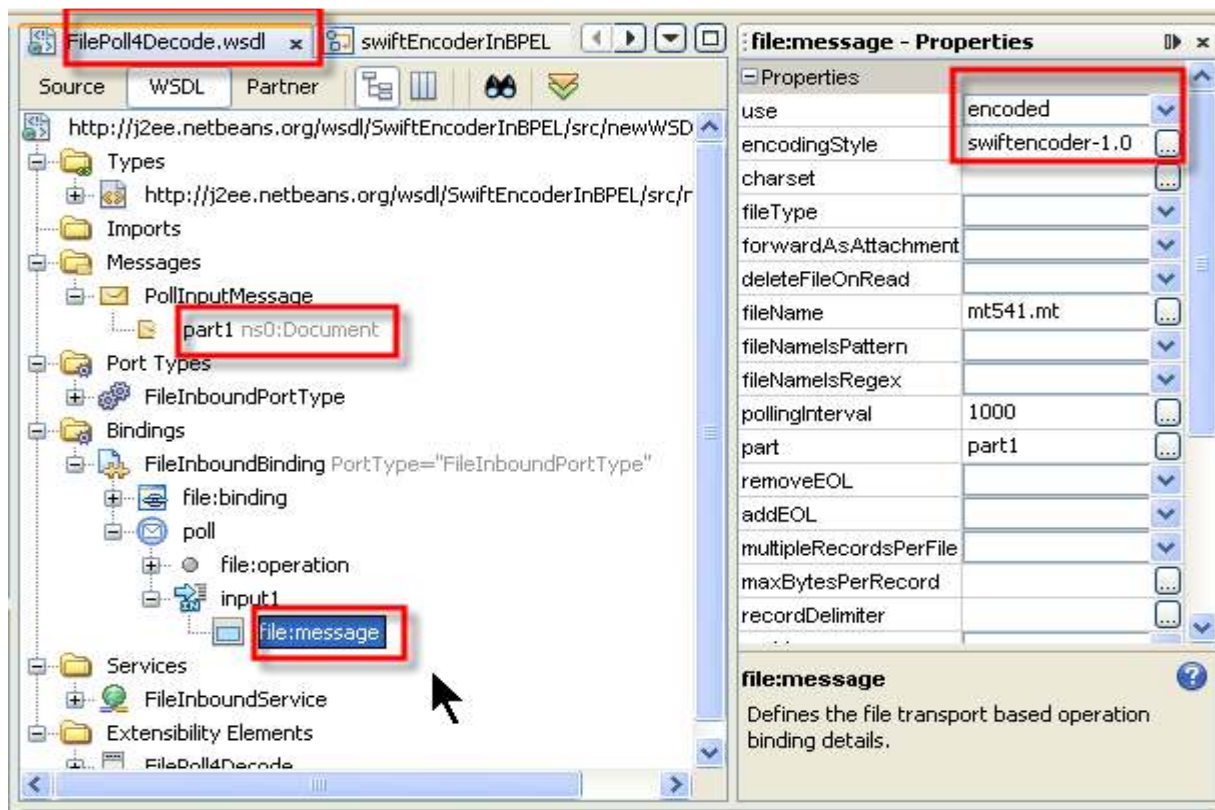
Copy the xsd schema “fin.541.2009.xsd” into project’s src folder.



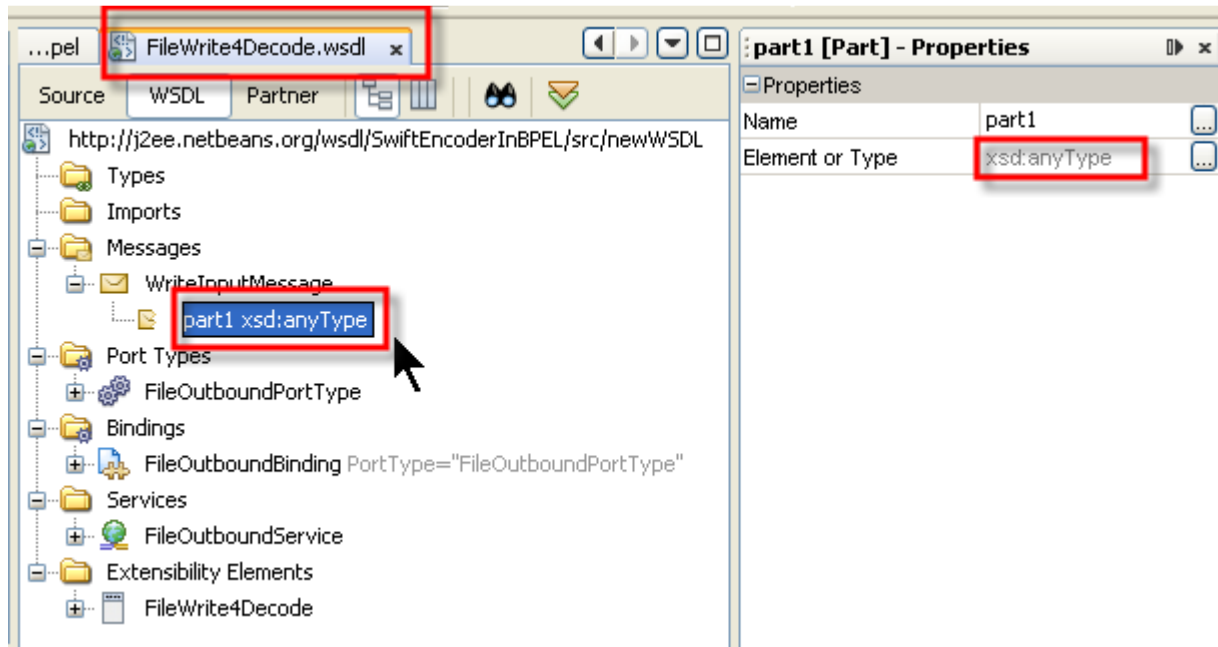
Configure WSDL “FilePoll4Decode” ... for part1, change “Element or type” to element “Document” in “fin.541.2009.xsd”



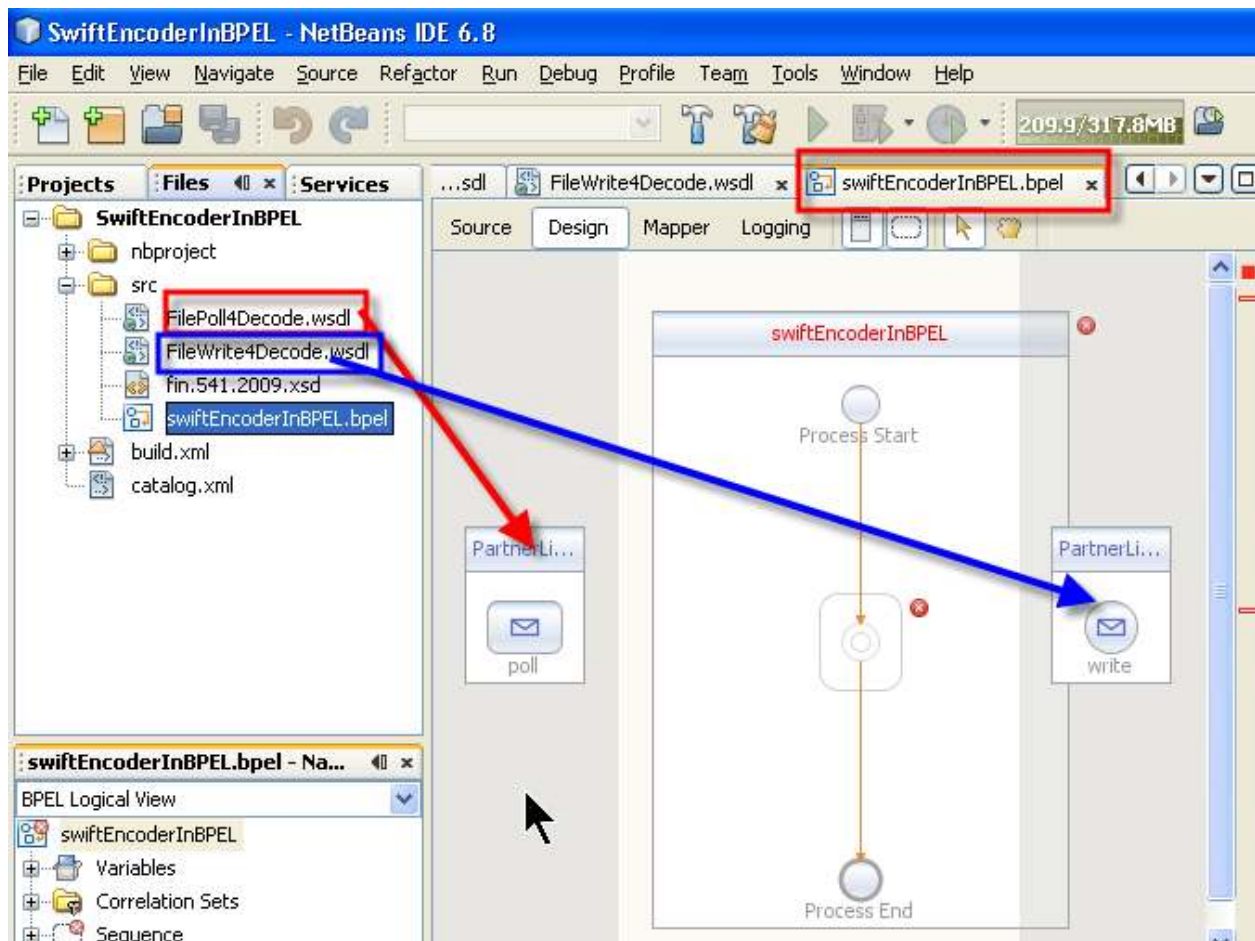
Configure use “Encoded” and encodingStyle “swiftencoder-1.0”.



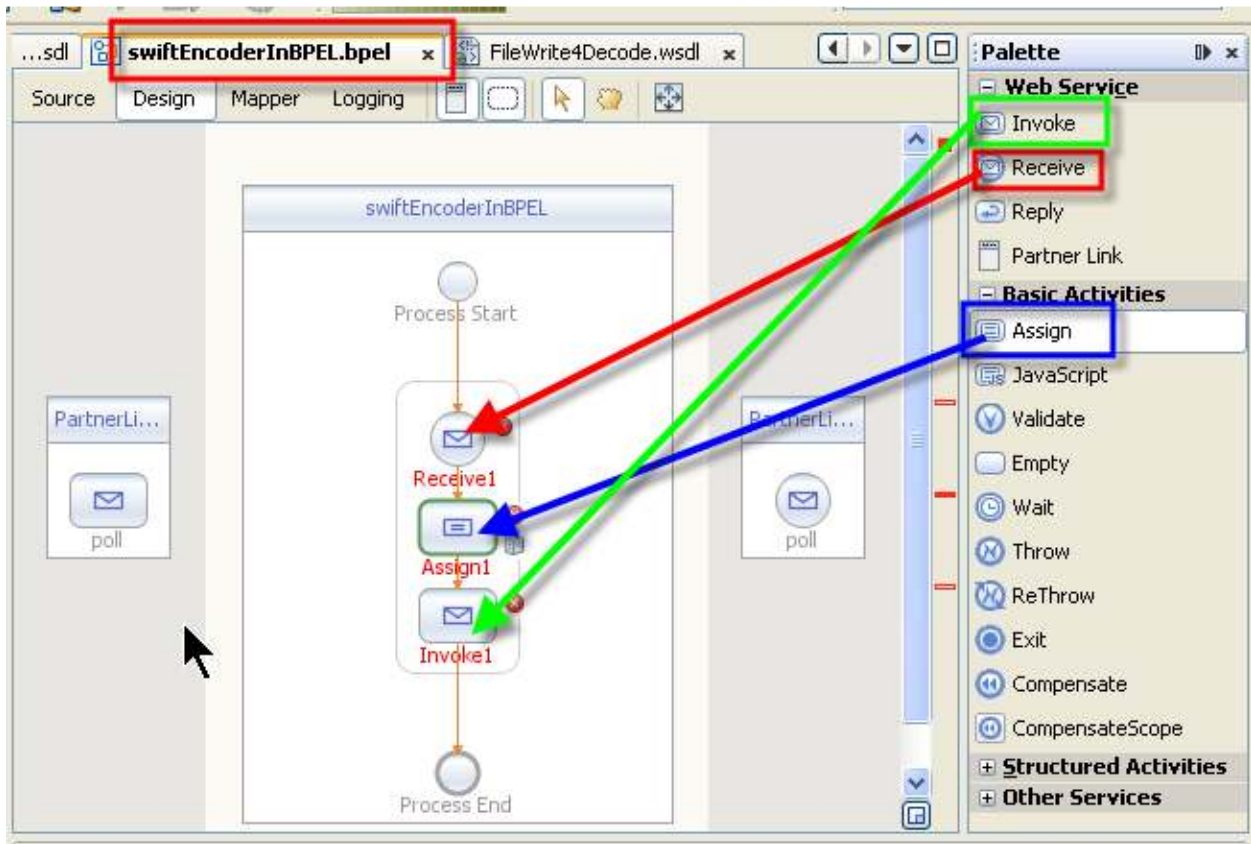
For WSDL “FileWrite4Decode”, change part1 type to “xsd:anyType”.



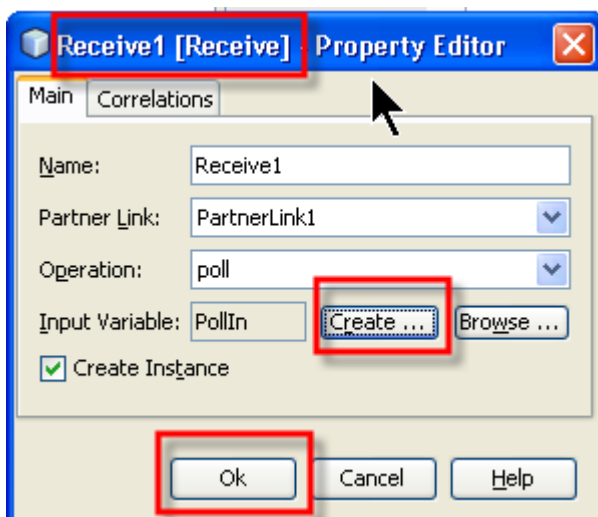
In BPEL editor, drag-n-drop two WSDL files into the panel as indicated.



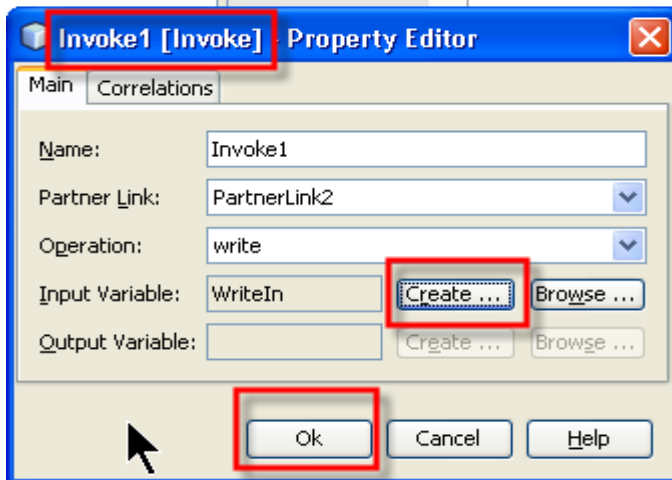
Orchestrate BPEL process: **Receive** → **Assign** → **Invoke** ... by drag-n-drop as arrows indicated.



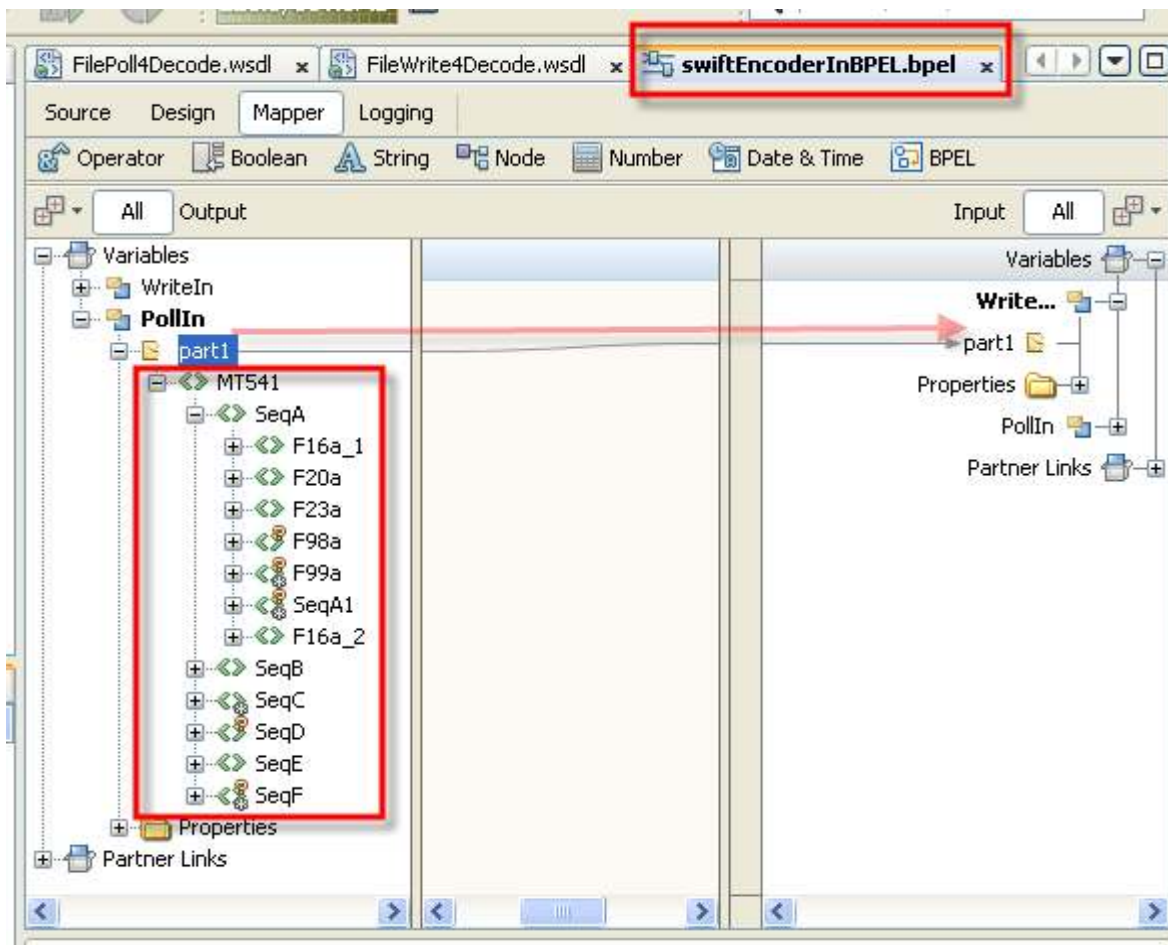
Configure “Receive1”



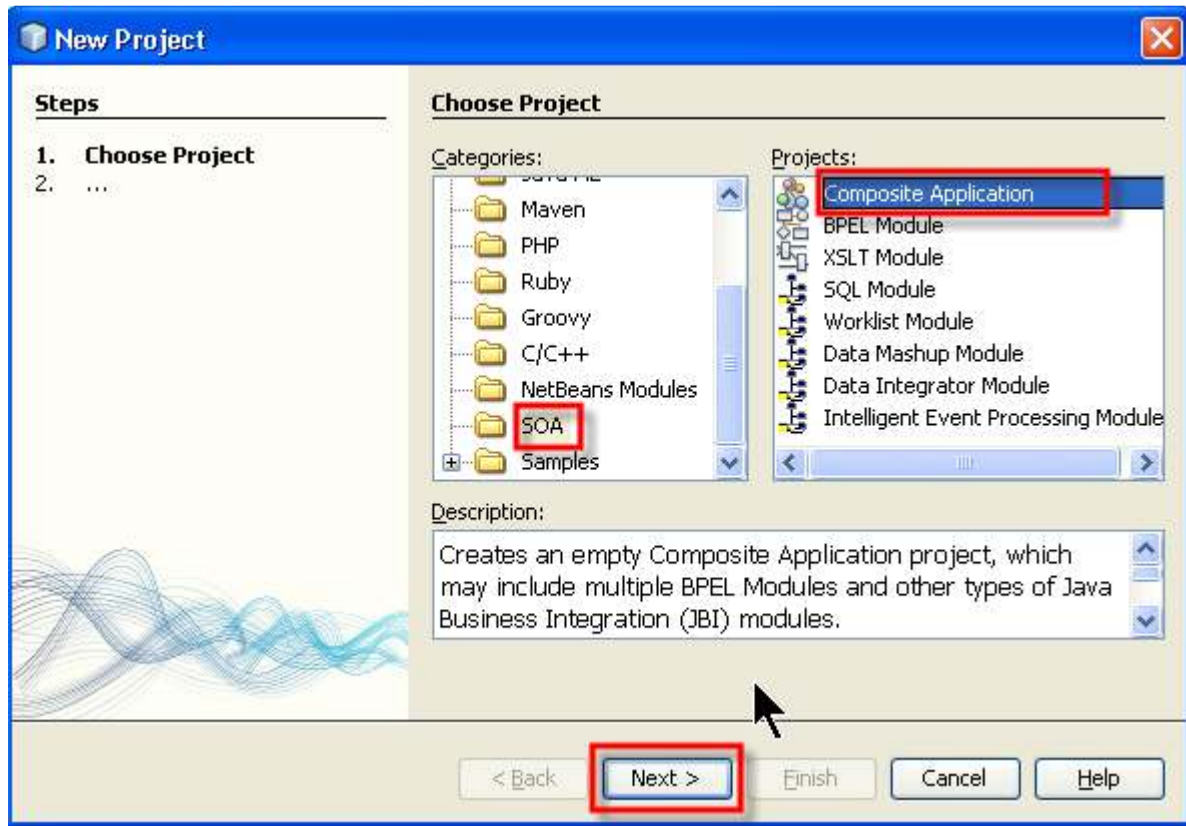
## Configure “Invoke1”



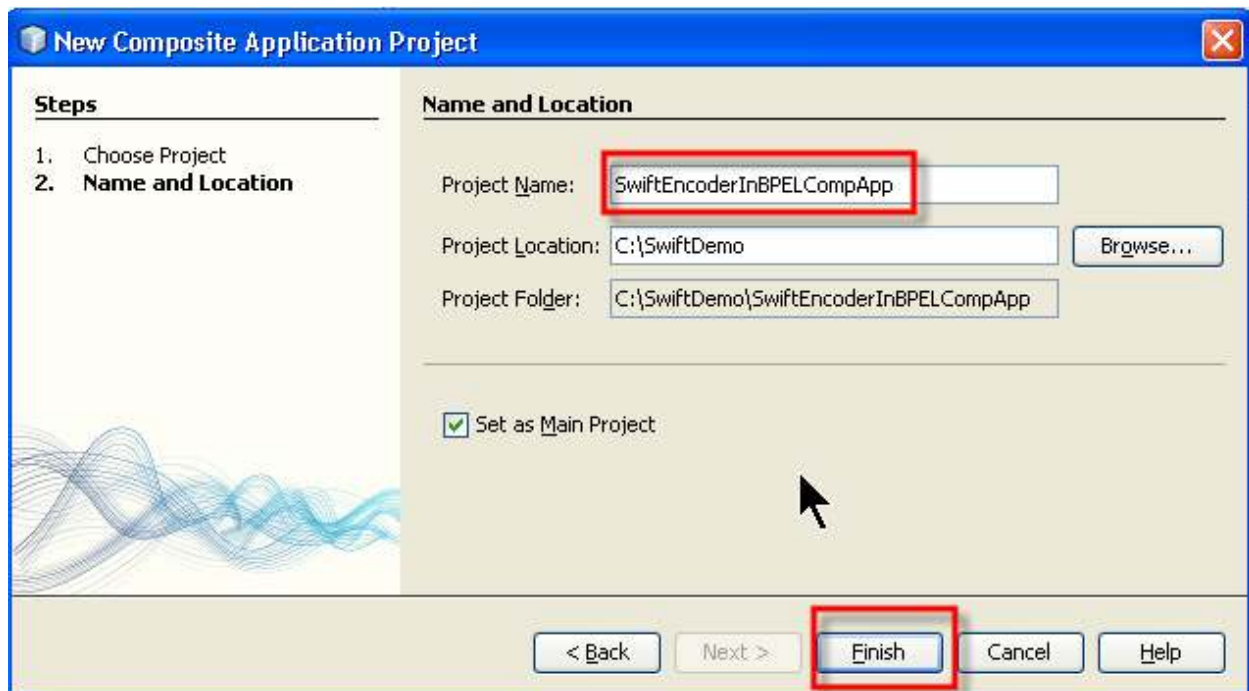
Open “Assign1” map editor. Map **PollIn/part1** → **Write/part1** as indicated. Note you see the MT541 message structure on left hand side.



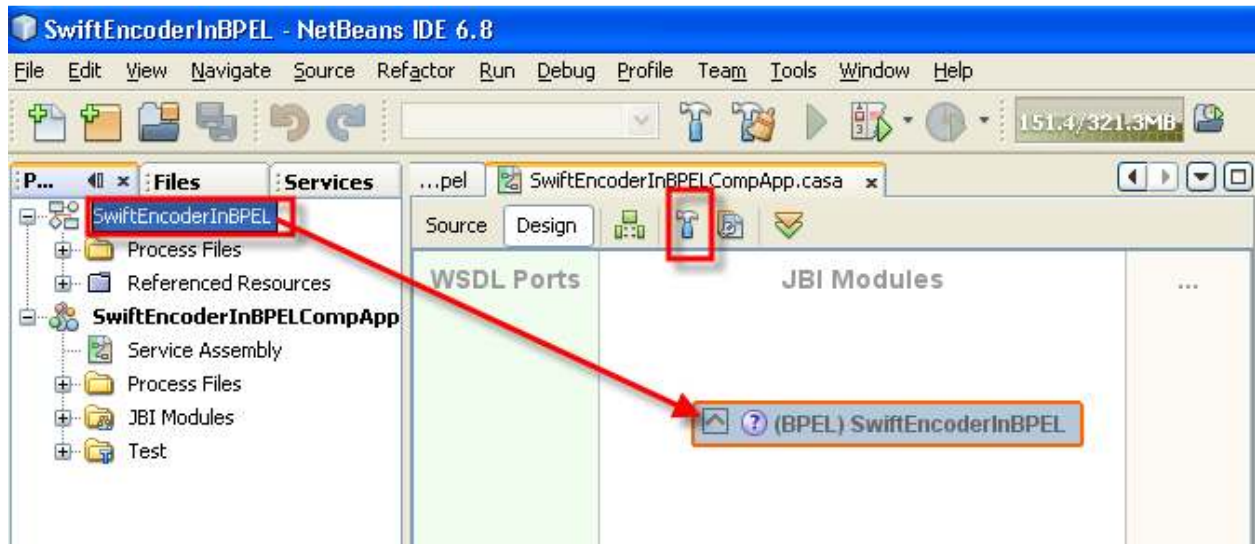
Create a project SOA → Composite Application



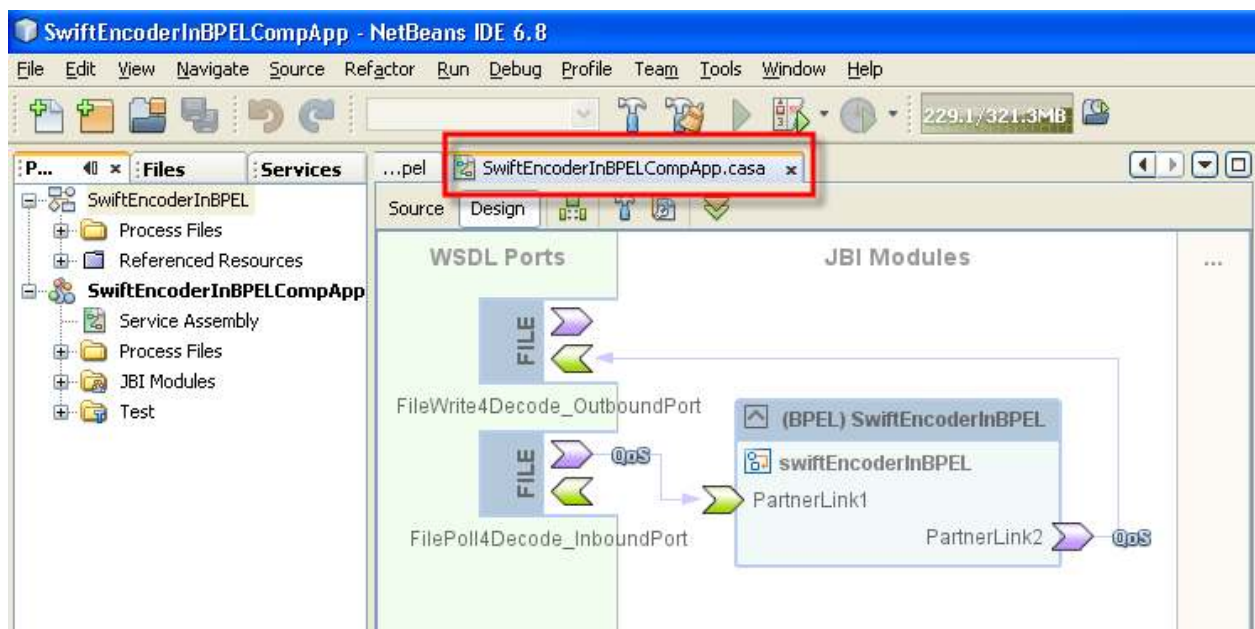
Give name "SwiftEncoderInBPELCompApp".



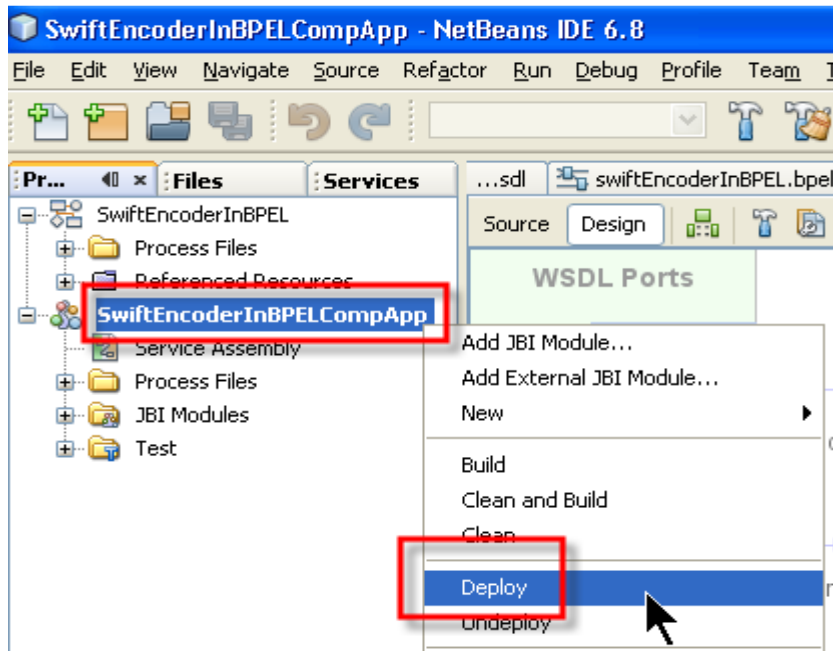
Drag-n-drop project “SwiftEncodeInBPEL” into casa panel. Click build icon.



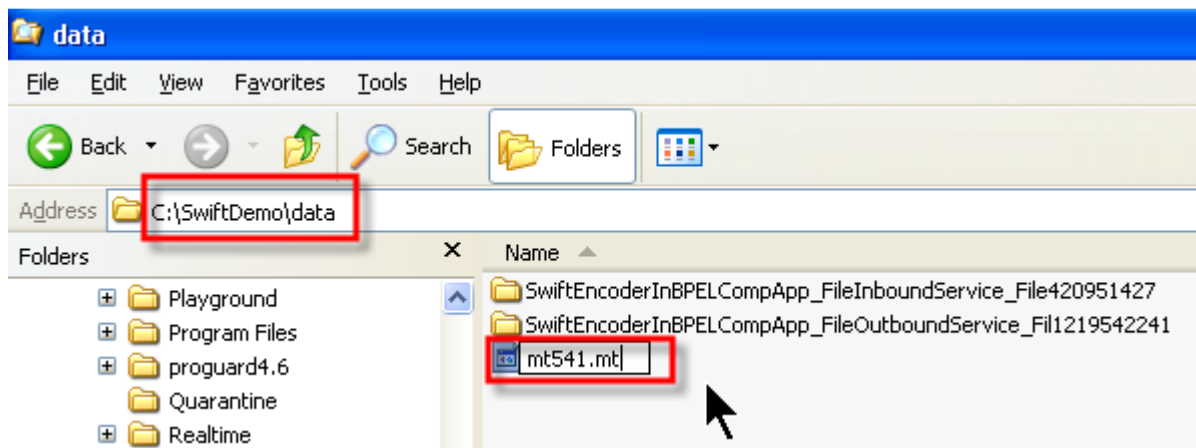
Panel will be refreshed as below.



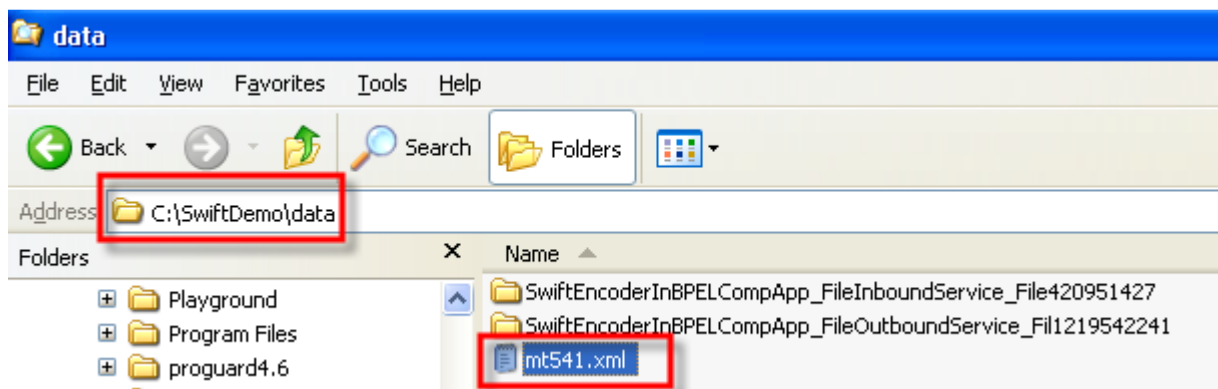
Deploy the composite app “SwiftEncoderInBPELCompApp”.



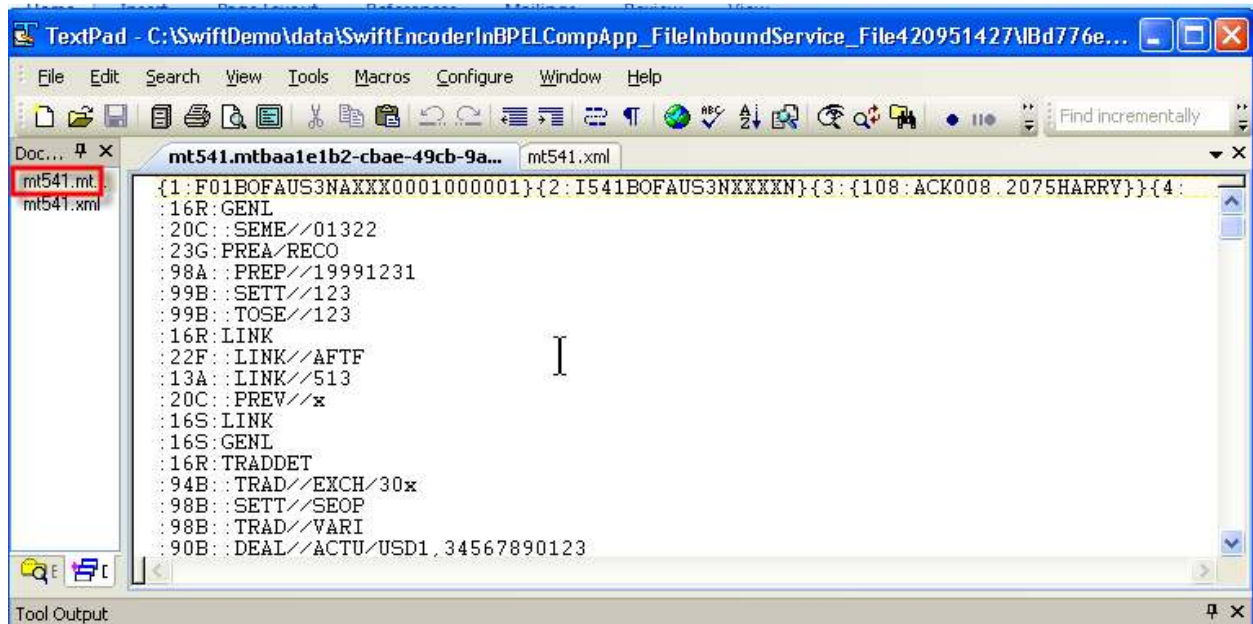
Feed in the input MT file as configured “C:\SwiftDemo\data\mt541.mt”.



The input file disappears and output XML “mt541.xml” appears.



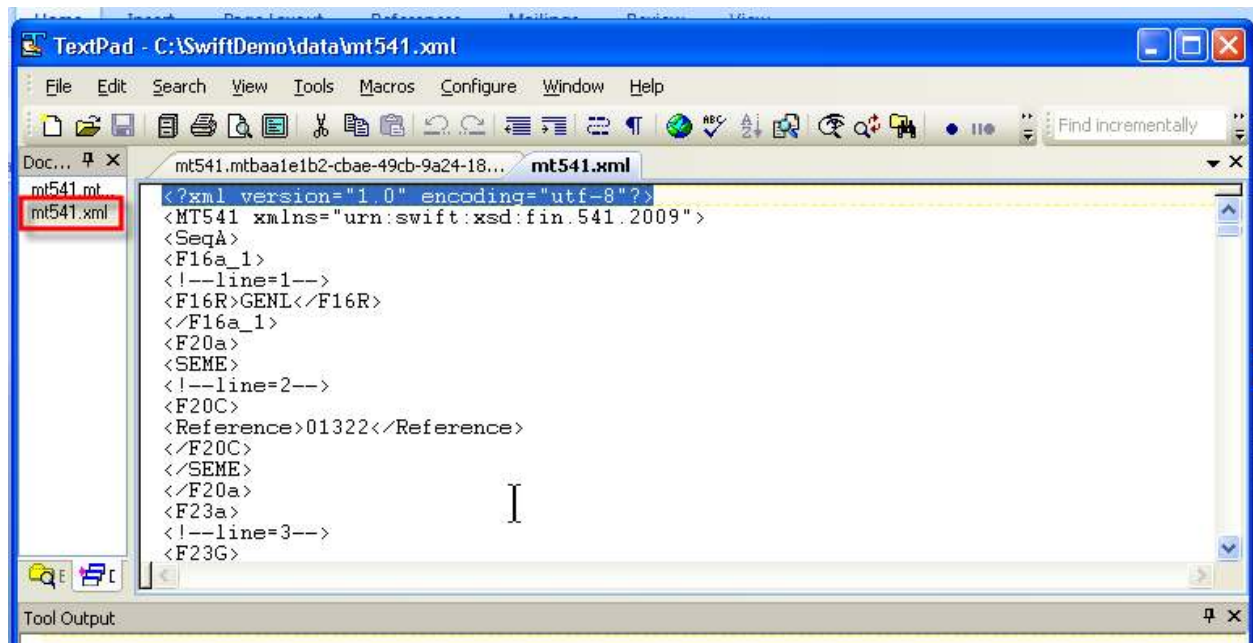
Take a look at the input MT file “mt541.mt”.



The screenshot shows a TextPad window with the file 'mt541.mt' open. The file content is a Swift MT541 message in plain text format. The file name 'mt541.mt' is highlighted in the left sidebar. The main text area contains the following message:

```
{1:F01BOFAUS3NAXXX0001000001}{2:I541BOFAUS3NXXXXN}{3:{108:ACK008.2075HARRY}}{4:  
:16R:GENL  
:20C::SEME//01322  
:23G:PREA/RECO  
:98A::PREP//19991231  
:99B::SETT//123  
:99B::TOSE//123  
:16R:LINK  
:22F::LINK//AFTF  
:13A::LINK//513  
:20C::PREV//x  
:16S:LINK  
:16S:GENL  
:16R:TRADDET  
:94B::TRAD//EXCH/30x  
:98B::SETT//SEOP  
:98B::TRAD//VARI  
:90B::DEAL//ACTU/USD1.34567890123
```

Also take a look the output XML file “mt541.xml”.



The screenshot shows a TextPad window with the file 'mt541.xml' open. The file content is the XML representation of the MT541 message. The file name 'mt541.xml' is highlighted in the left sidebar. The main text area contains the following XML structure:

```
<?xml version="1.0" encoding="utf-8"?>  
<MT541 xmlns="urn:swift:xsd:fin.541.2009">  
<SeqA>  
<F16a_1>  
<!--line=1-->  
<F16R>GENL</F16R>  
</F16a_1>  
<F20a>  
<SEME>  
<!--line=2-->  
<F20C>  
<Reference>01322</Reference>  
</F20C>  
</SEME>  
</F20a>  
<F23a>  
<!--line=3-->  
<F23G>
```

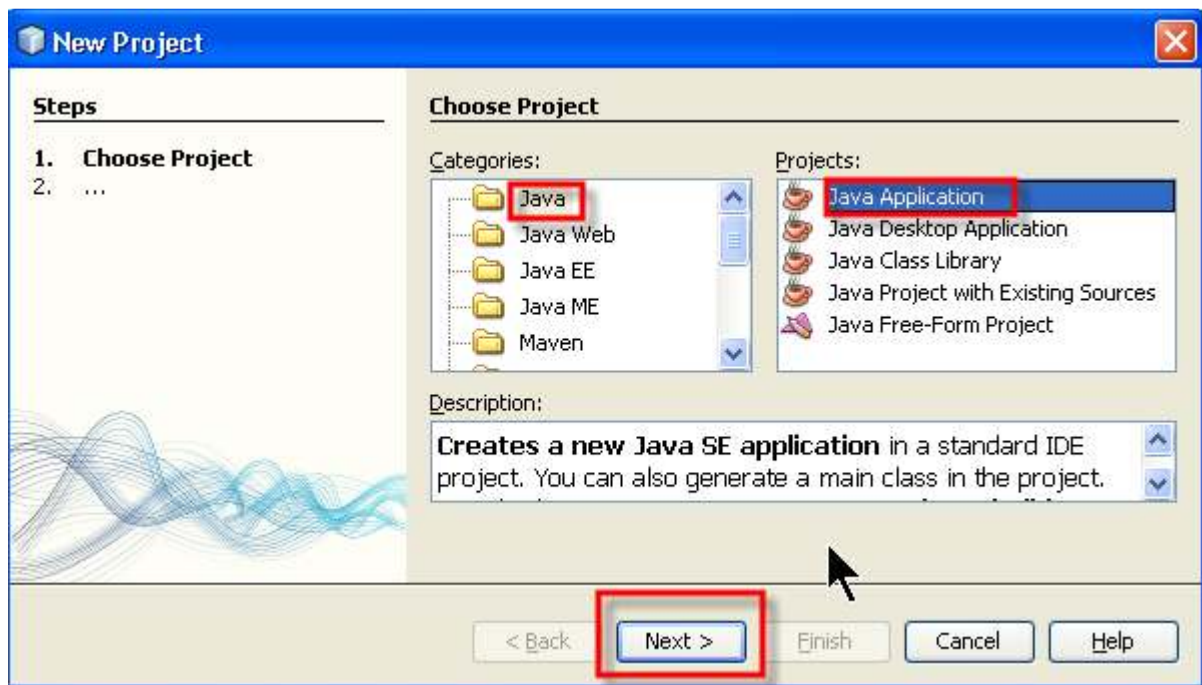
### 3. Using SWIFT Encoder in POJO

Demonstrate how to use SWIFT Encoder in POJO SE (Service Engine) by use of a simple example.

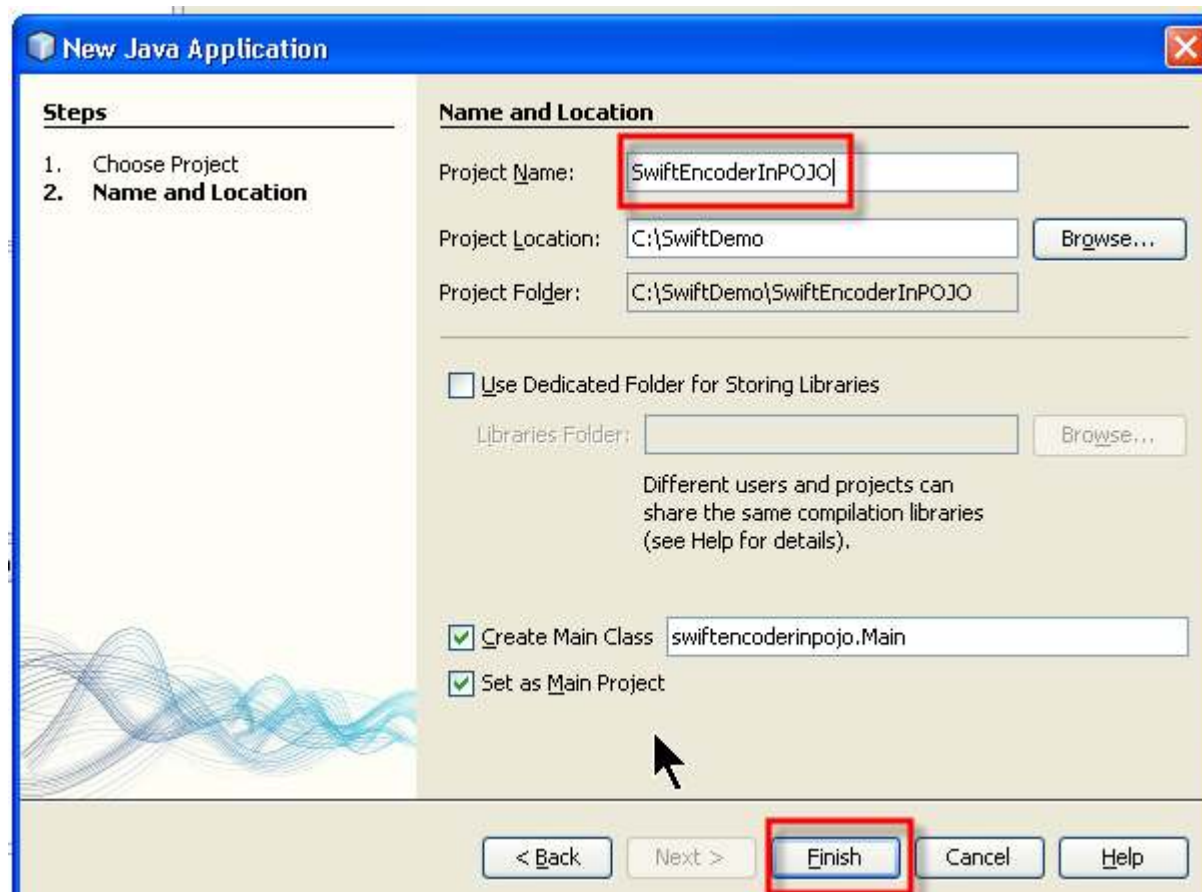
Here is a screen cast:

<http://swiftencoder.appspot.com/SwiftEncoderInPOJO.html>

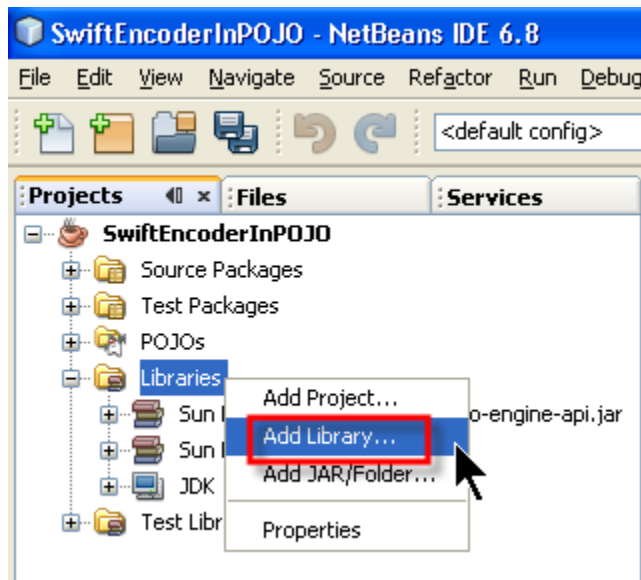
Create a project Java → Java Application



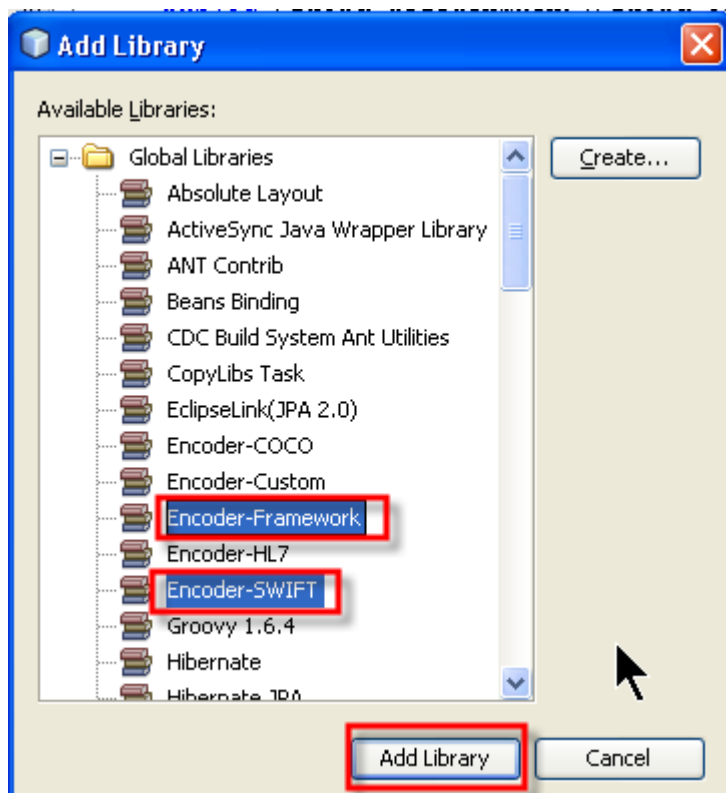
Give name "SwiftEncoderInPOJO"



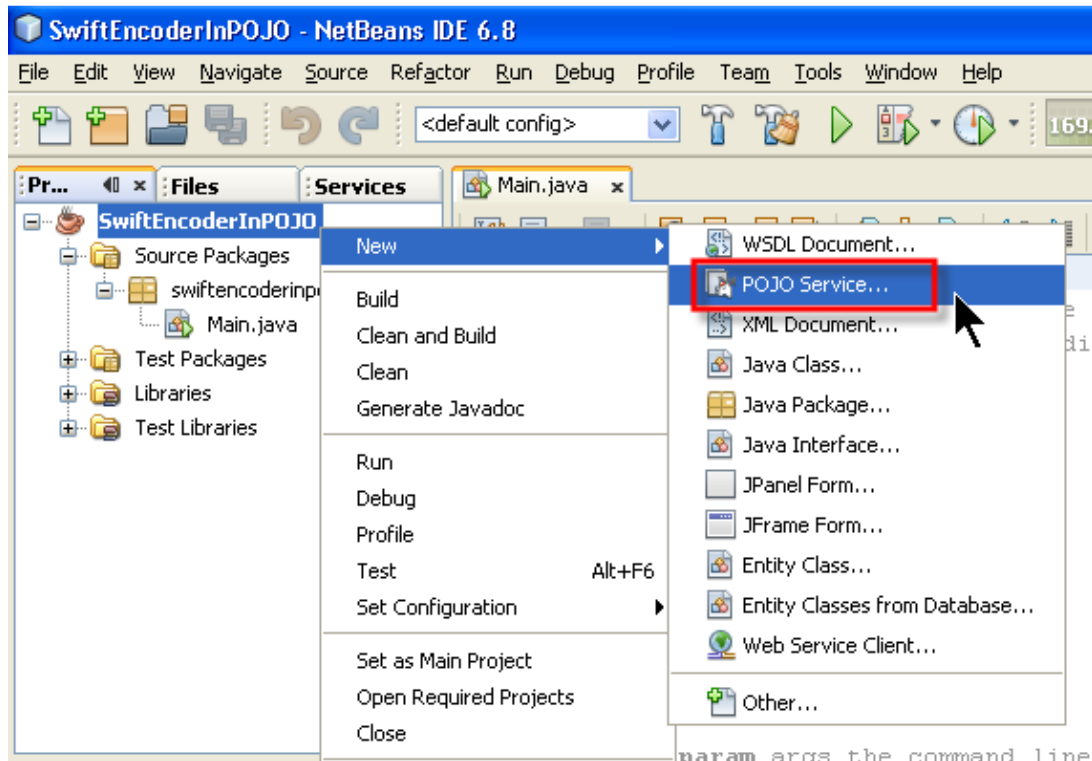
Add Library by right-click on project node “Libraries”.



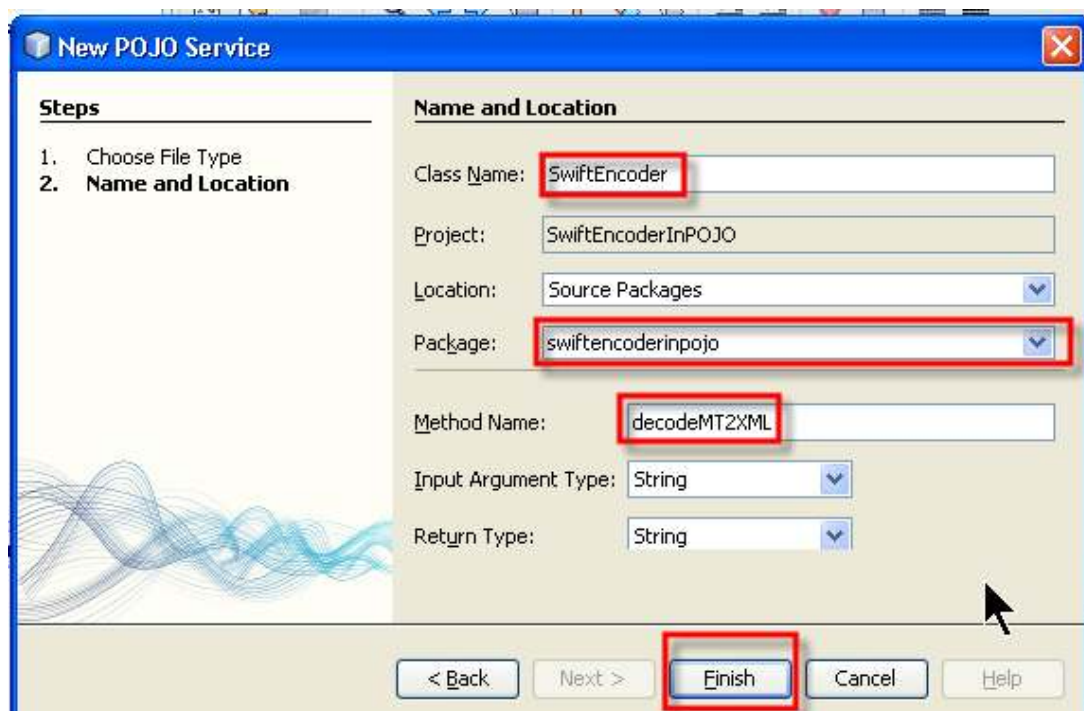
Choose libraries “Encoder-Framework” and “Encoder-SWIFT”.



Right-click “SwiftEncoderInPOJO”, choose “New”, “POJO Service ...”.



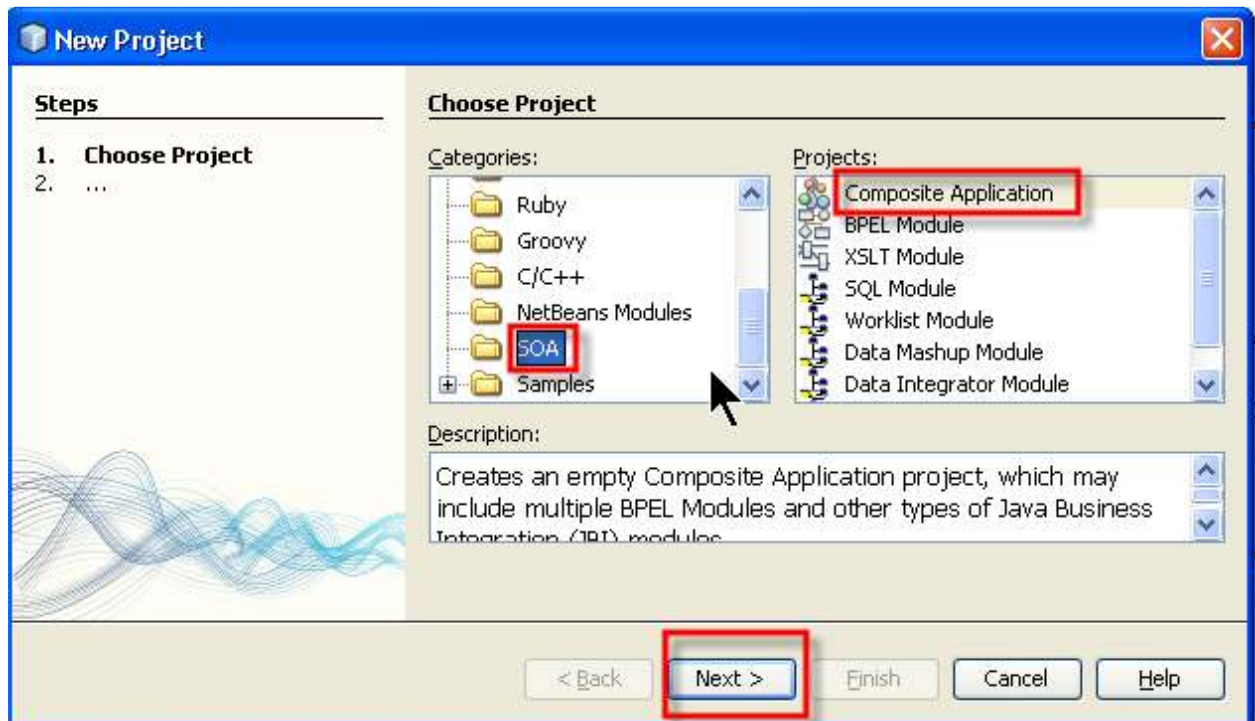
Give name “SwiftEncoder”, package “swiftencoderpojo”, method name “decodeMT2XML” ...



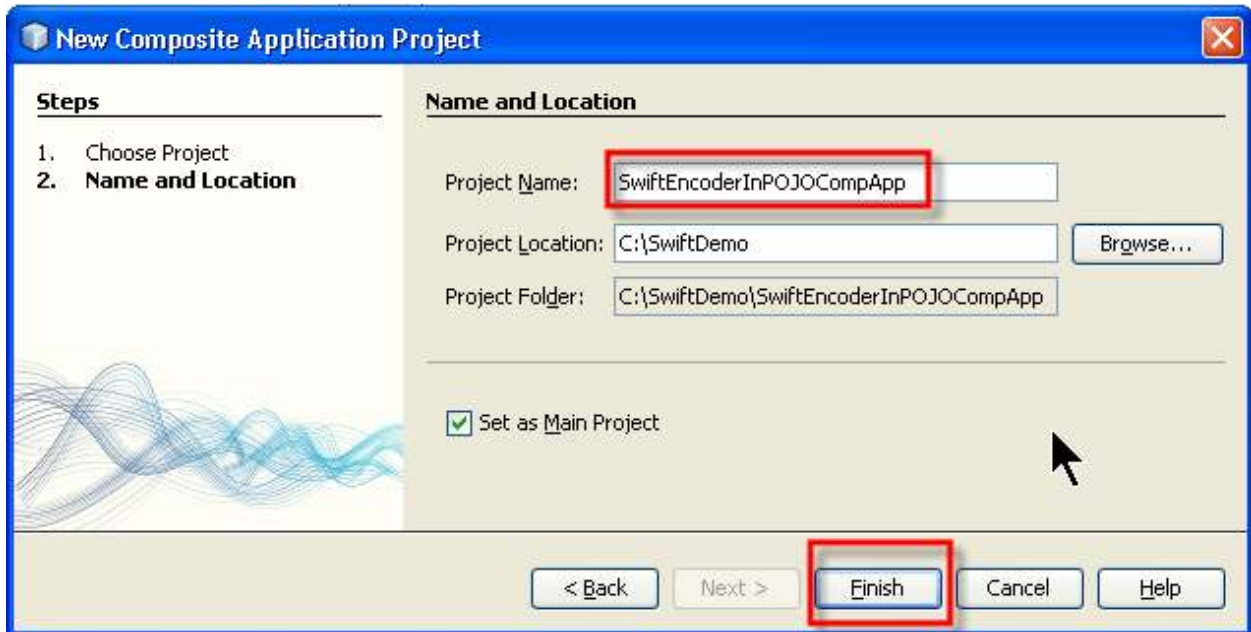
Write the code for method “decodeMT2XML” ...

```
35 * @param input input of type String input
36 * @return String
37 */
38 @Operation (outMessageTypeQN="{http://swiftencoderinpojo/SwiftEncoder/}Sw
39 public String decodeMT2XML(String input) {
40     String mt = input.replace("(CRLF)", "\\r\\n"); // work-around composite
41     String xml = "";
42     try {
43         StringWriter writer = new StringWriter();
44         Transformer transformer = TransformerFactory.newInstance().newTre
45         transformer.transform(
46             new SwiftEncoderProvider().newEncoder(null, null).decodeF
47             new StreamResult(writer));
48         xml = writer.toString();
49     } catch (Exception e) {
50         xml = "Input is [" + input + "] Got exception: " + e.toString();
51         logger.log(Level.SEVERE, xml, e);
52     }
53
54     return xml;
55 }
```

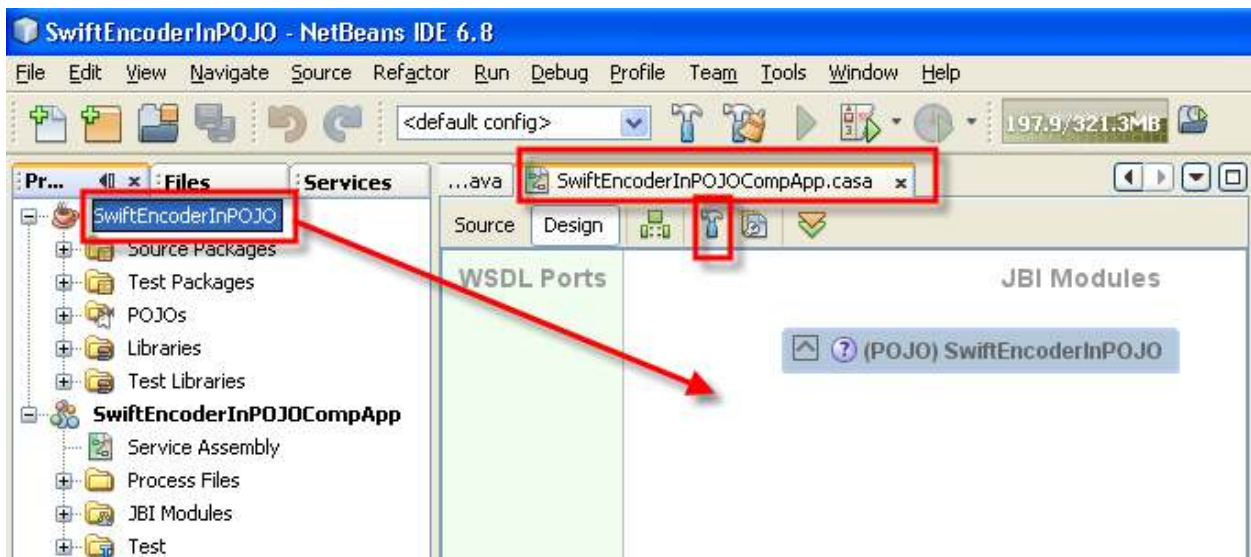
Create a project SOA → Composite Application



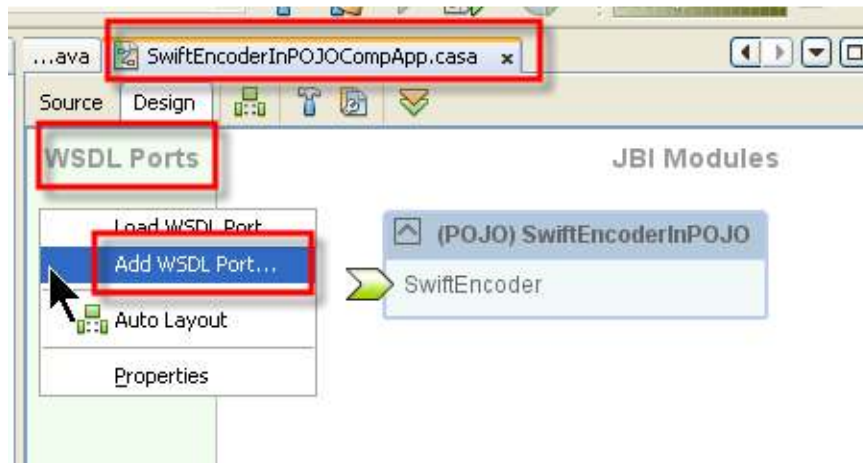
Give project name “SwiftEncoderInPOJOCmpApp”.



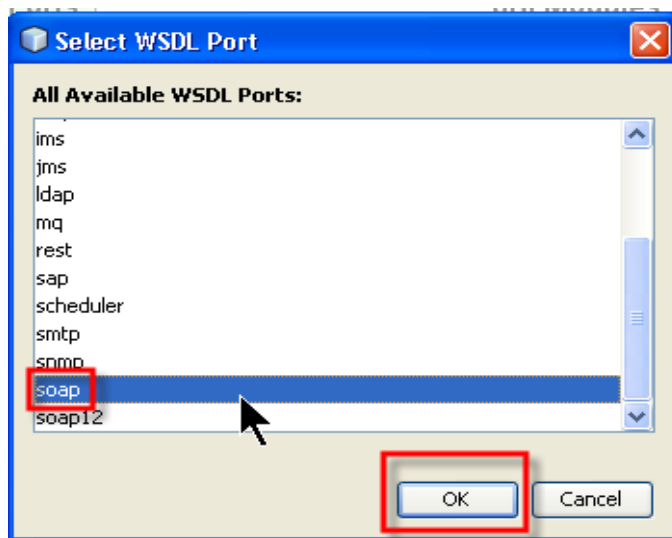
Drag-n-drop “SwiftEncoderPOJO” into casa panel. Then click “Build” button.



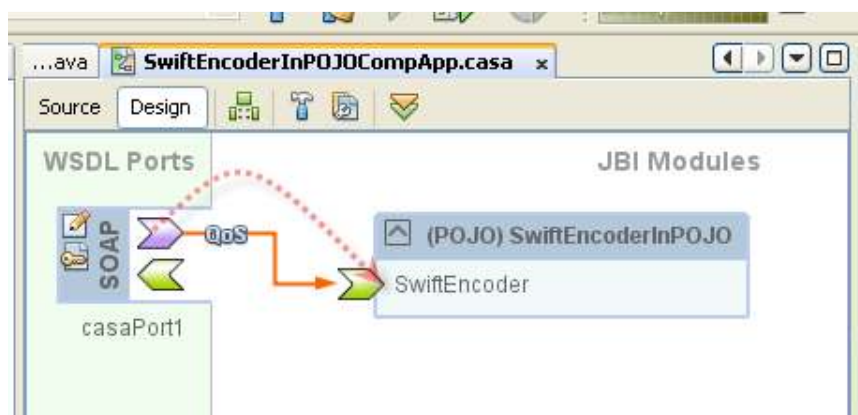
Right-click on the left hand side of casa panel, select “Add WSDL Port” from the pop-up menu.



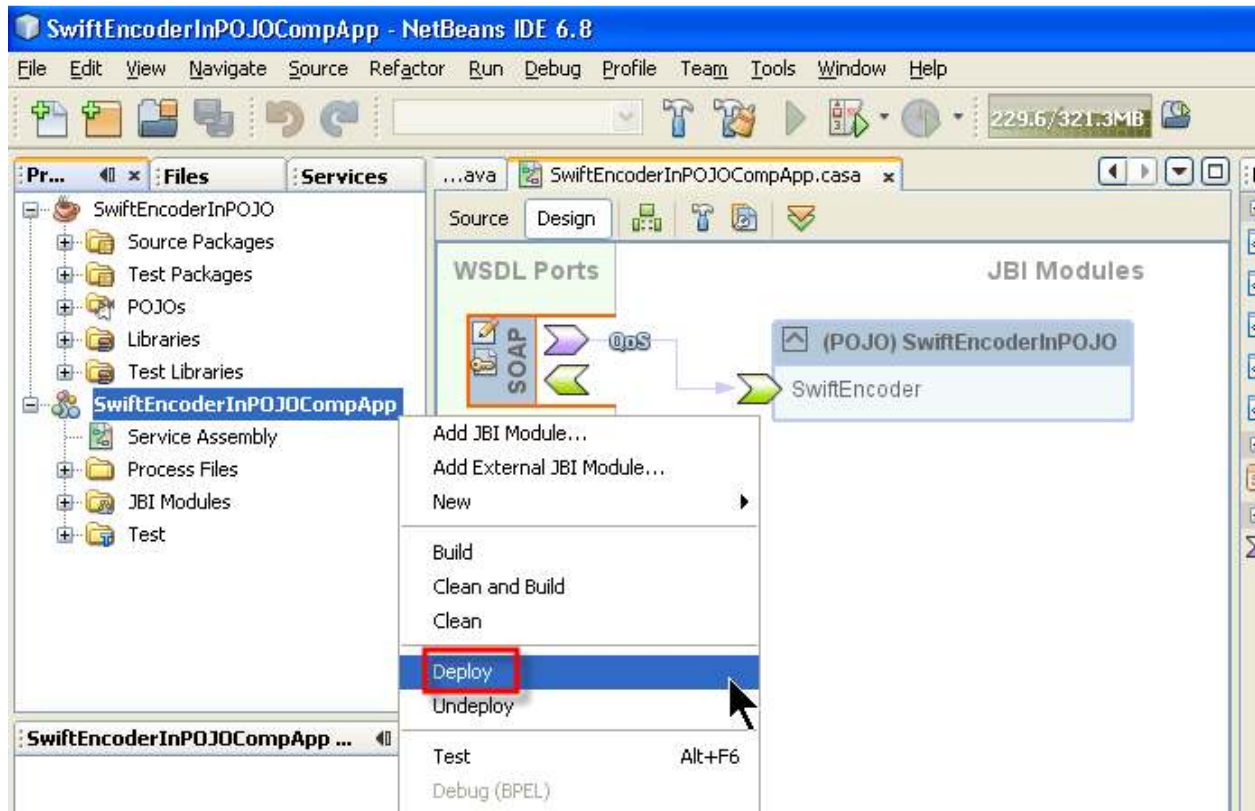
Choose soap WSDL port.



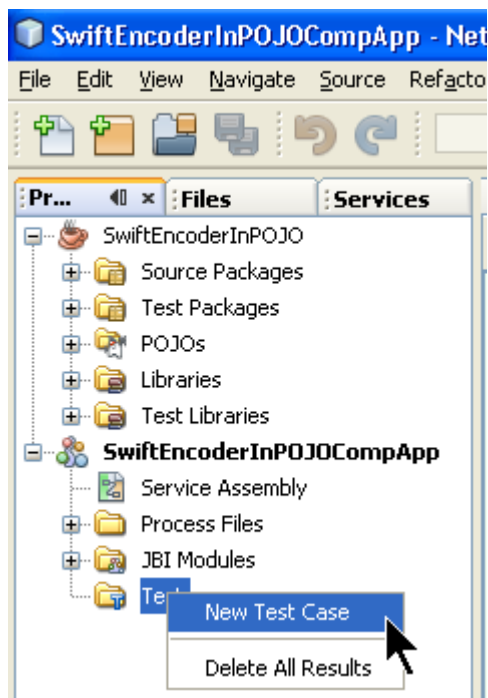
Connect soap port to POJO in casa panel.



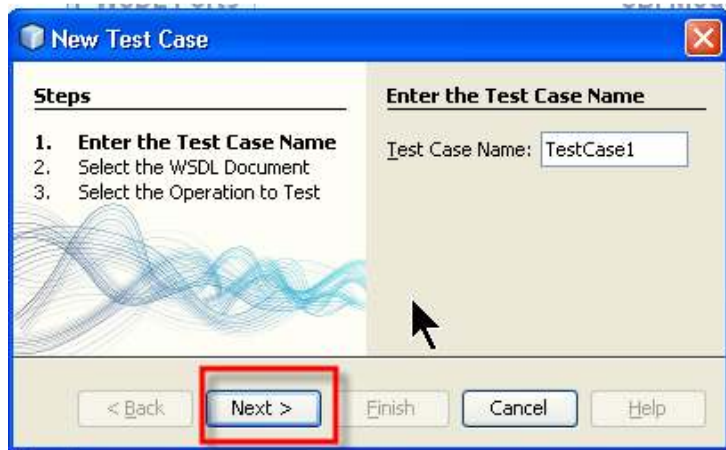
Save all. **Deploy** composite app “SwiftEncoderInPOJOCmpApp”.



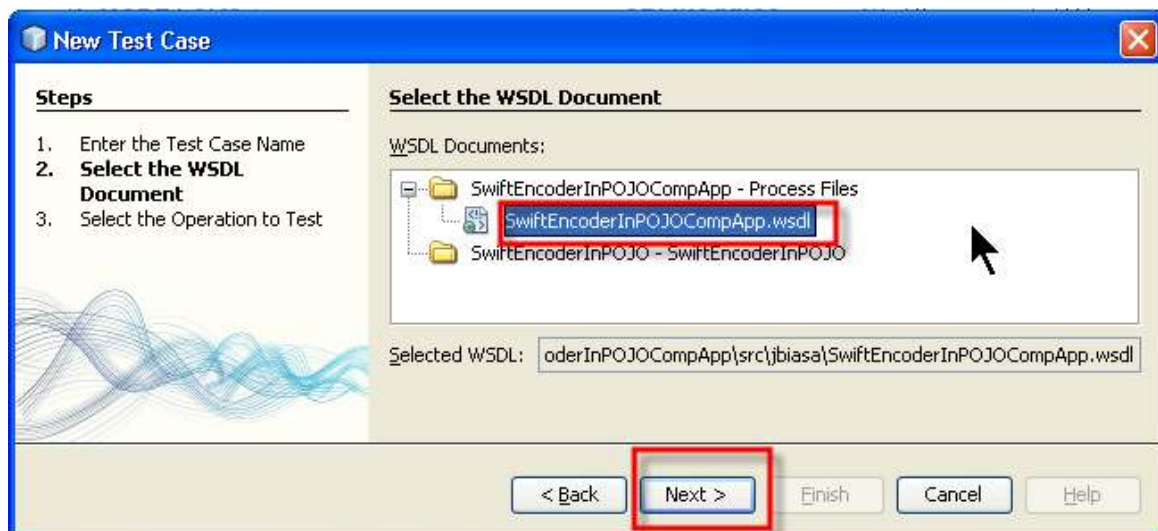
Create Test case in composite app “SwiftEncoderInPOJOCmpApp”



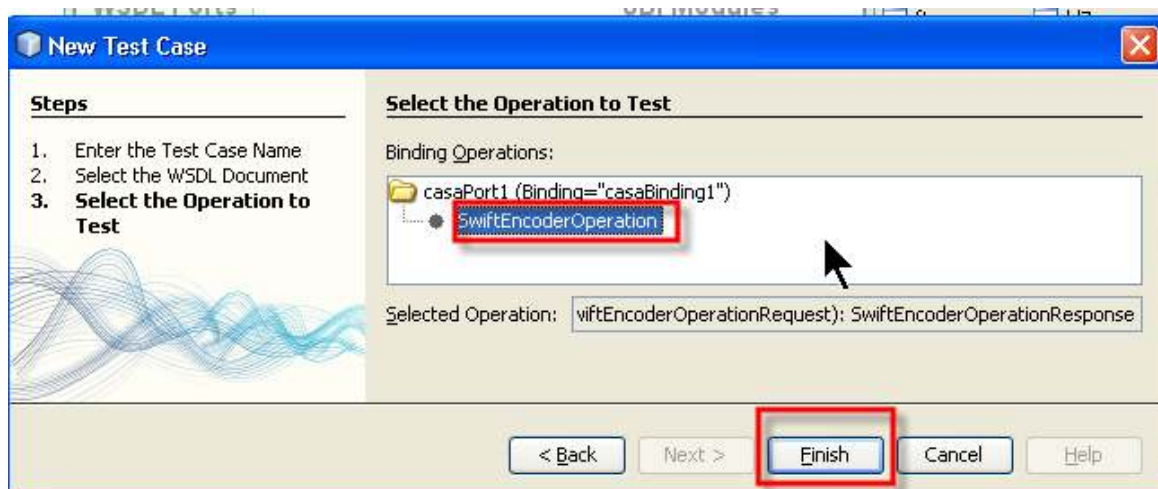
## TestCase1 ...



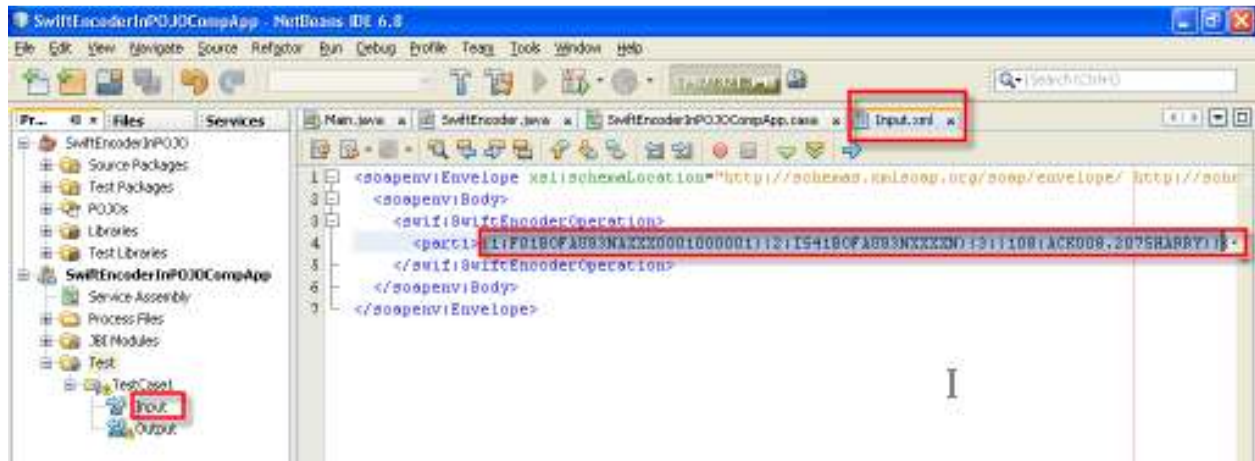
## Choose WSDL ...



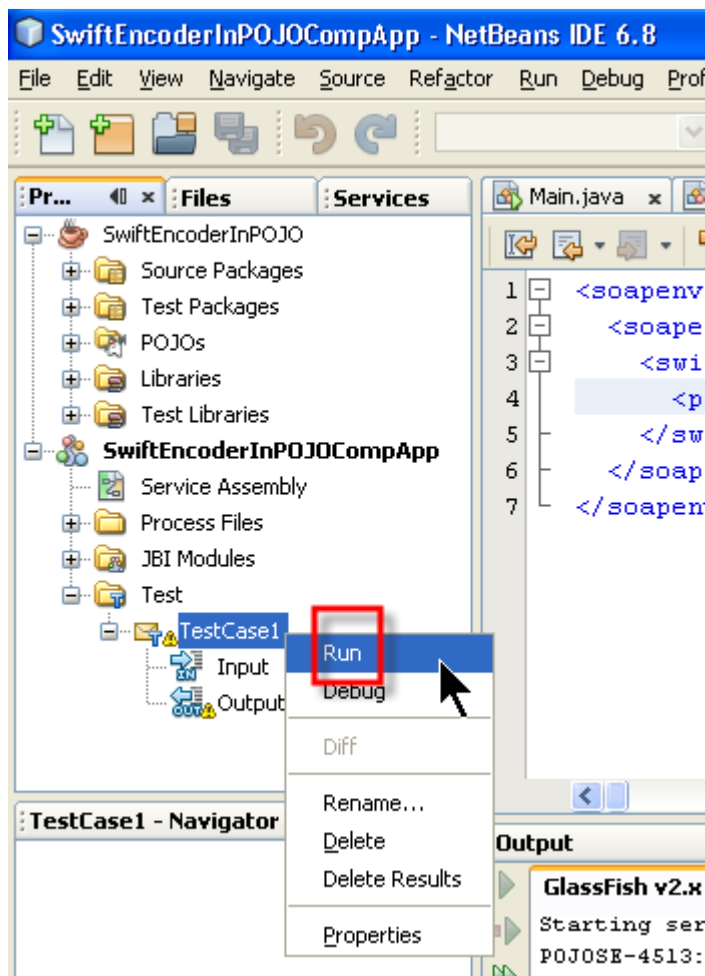
## Choose operation.



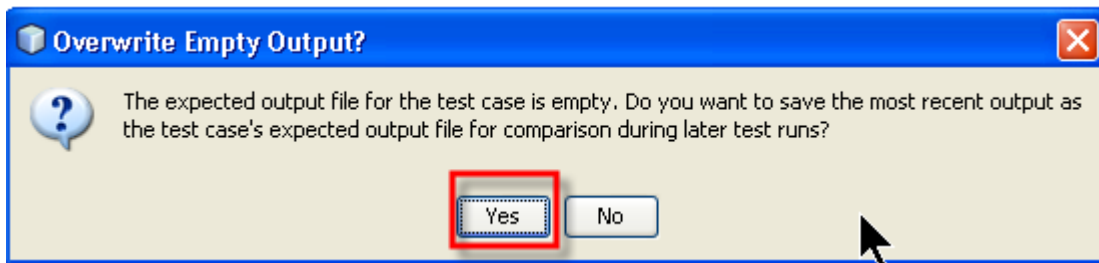
Copy-n-paste the input MT data.



Run the test.

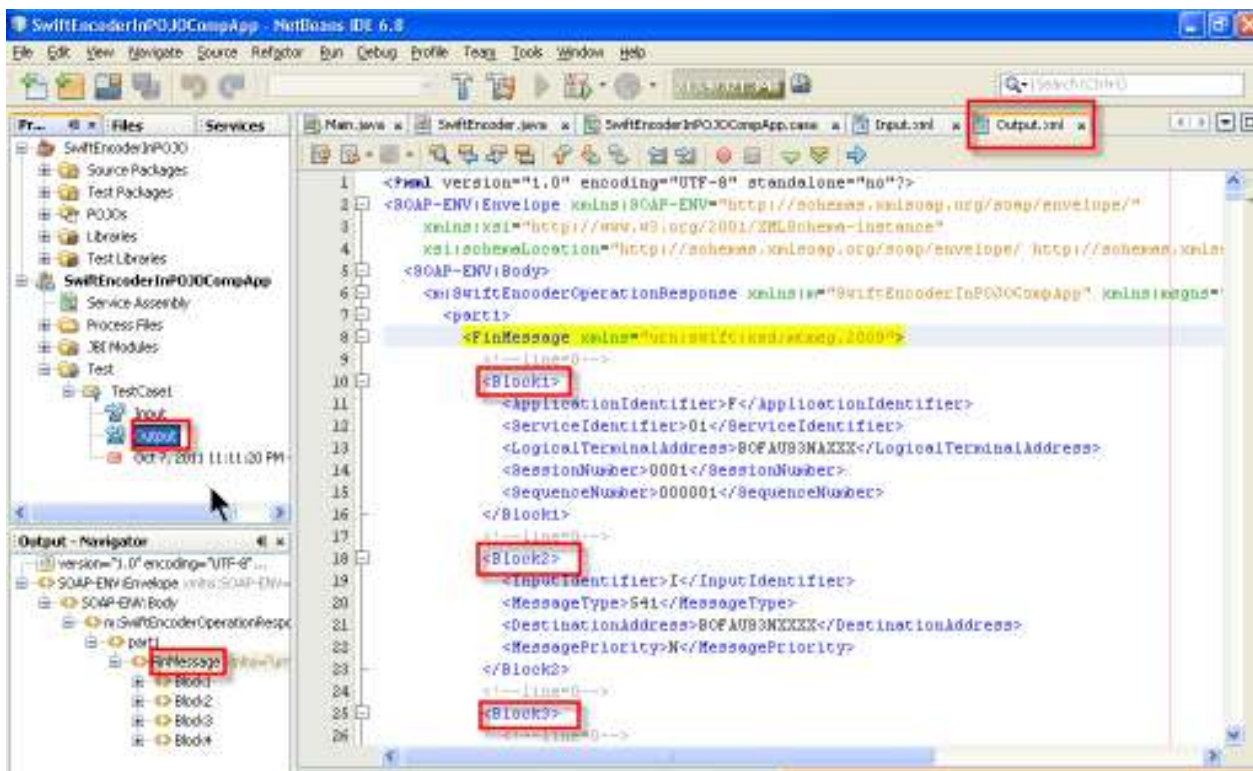


Answer **yes** because it's the first time run.



Open the output (the test result), you can see the XML.

Note you'll be seeing the entire XML document including block1-block5.



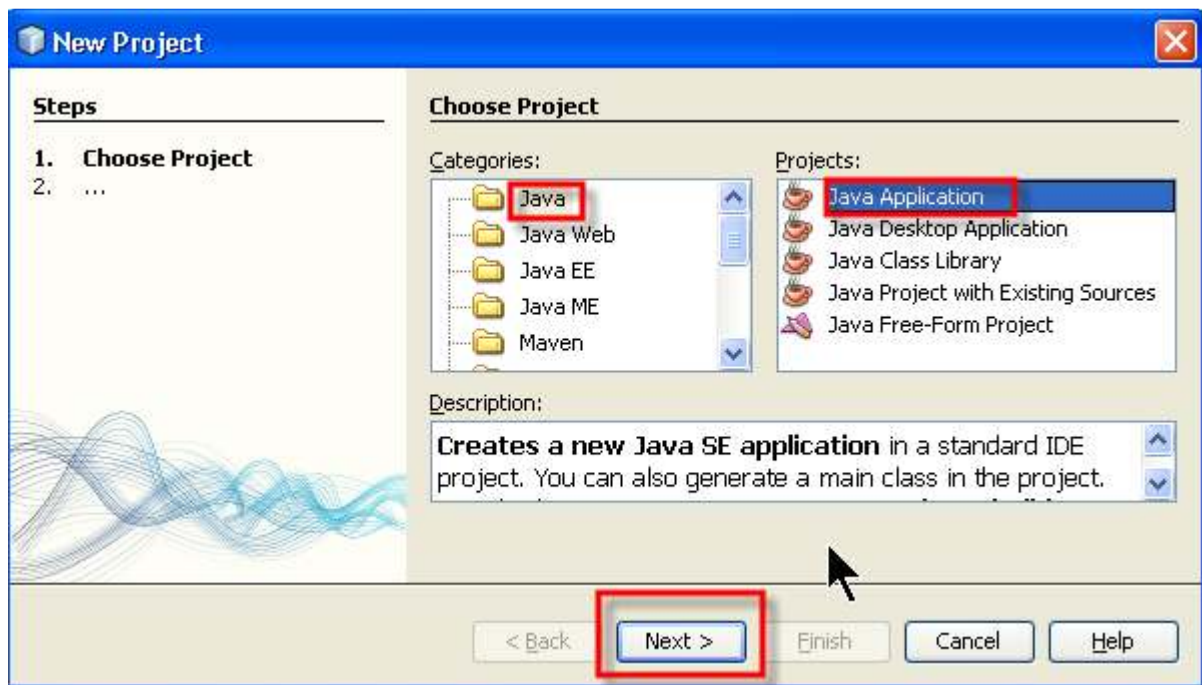
## 4. Using SWIFT Encoder in Java SE

Demonstrate how to use SWIFT Encoder in Java SE (Standard Edition) by use of a simple example.

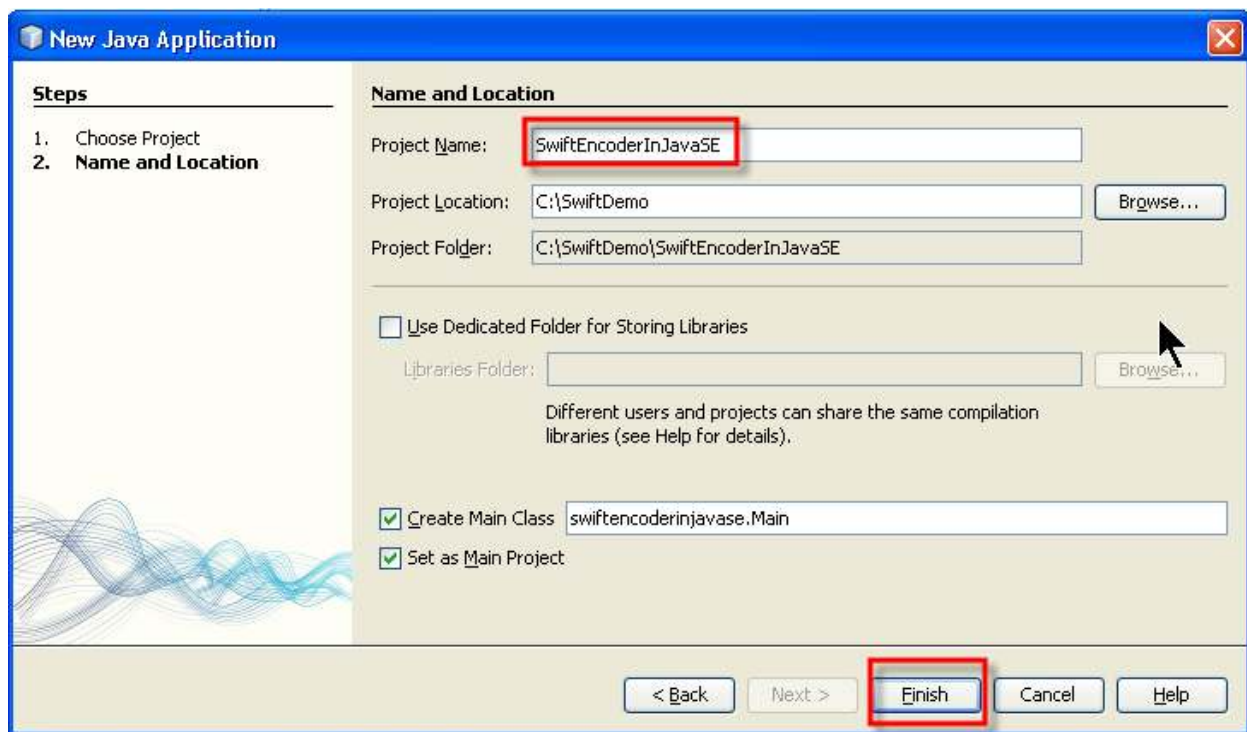
Here is a screen cast:

<http://swiftencoder.appspot.com/SwiftEncoderInJavaSE.html>

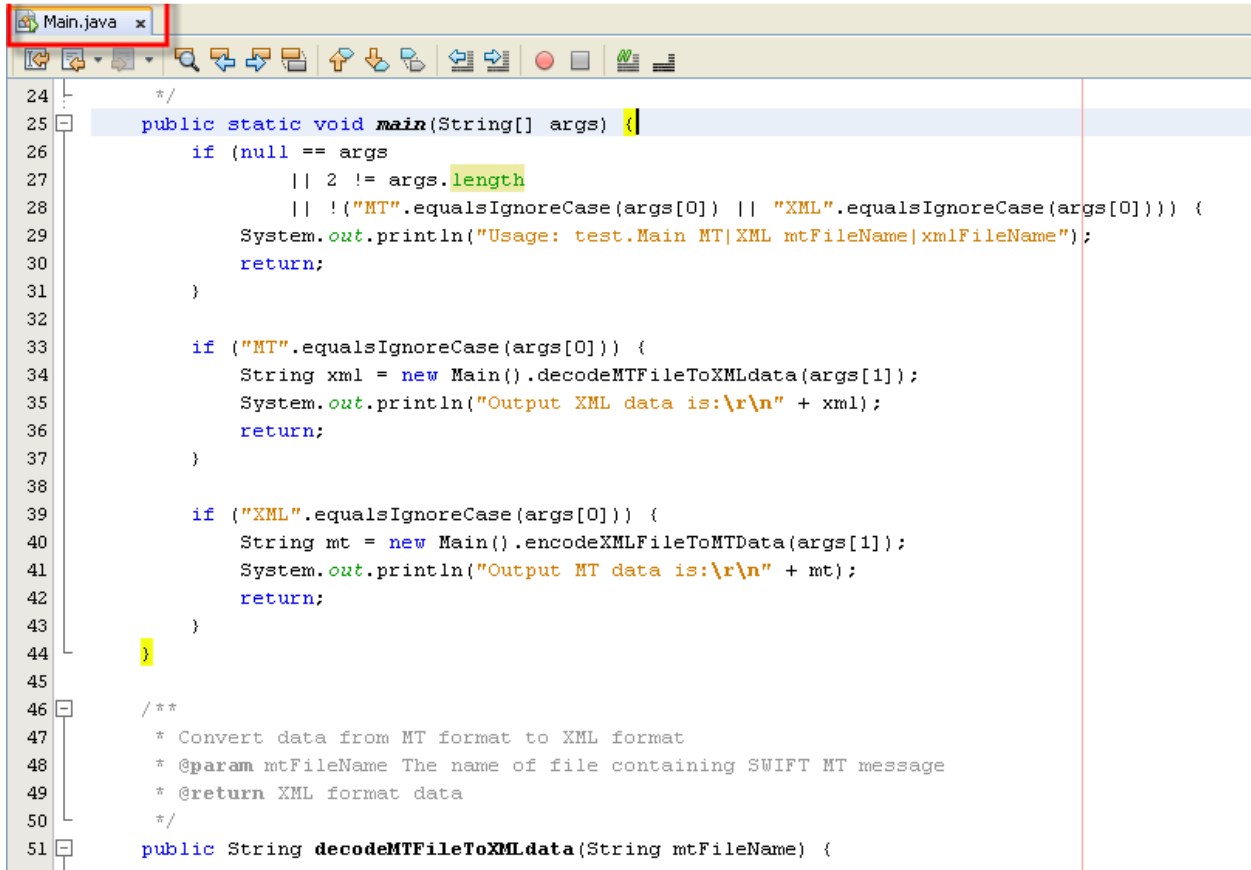
Create a project Java → Java Application



Name it as "SwiftEncoderInJavaSE"



Here is a screen shot.



```
24 |  
25 | public static void main(String[] args) {  
26 |     if (null == args  
27 |         || 2 != args.length  
28 |         || !"MT".equalsIgnoreCase(args[0]) || "XML".equalsIgnoreCase(args[0])) {  
29 |         System.out.println("Usage: test.Main MT|XML mtFileName|xmlFileName");  
30 |         return;  
31 |     }  
32 |  
33 |     if ("MT".equalsIgnoreCase(args[0])) {  
34 |         String xml = new Main().decodeMTFileToXMLdata(args[1]);  
35 |         System.out.println("Output XML data is:\r\n" + xml);  
36 |         return;  
37 |     }  
38 |  
39 |     if ("XML".equalsIgnoreCase(args[0])) {  
40 |         String mt = new Main().encodeXMLFileToMTData(args[1]);  
41 |         System.out.println("Output MT data is:\r\n" + mt);  
42 |         return;  
43 |     }  
44 | }  
45 |  
46 | /**  
47 |  * Convert data from MT format to XML format  
48 |  * @param mtFileName The name of file containing SWIFT MT message  
49 |  * @return XML format data  
50 |  */  
51 | public String decodeMTFileToXMLdata(String mtFileName) {
```

Code is similar to POJO sample (actually much simpler than POJO case), I won't describe the details any more.

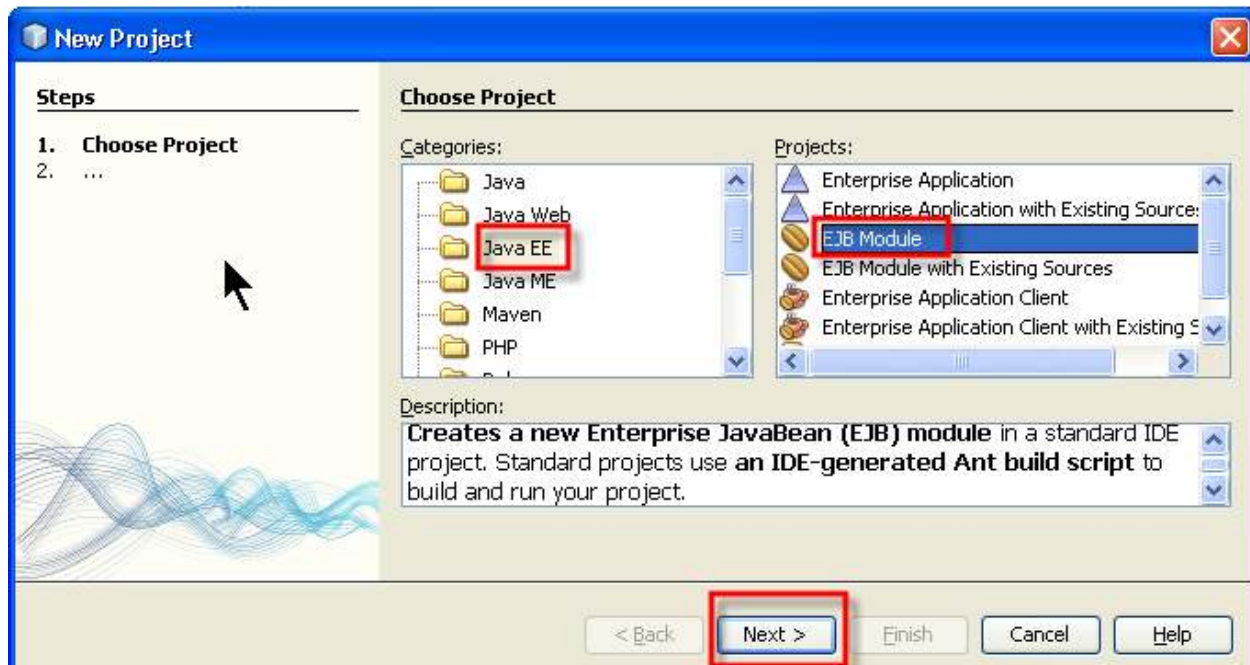
## 5. Using SWIFT Encoder in Java EE

Demonstrate how to use SWIFT Encoder in Java EE (Enterprise Edition) by use of a simple example.

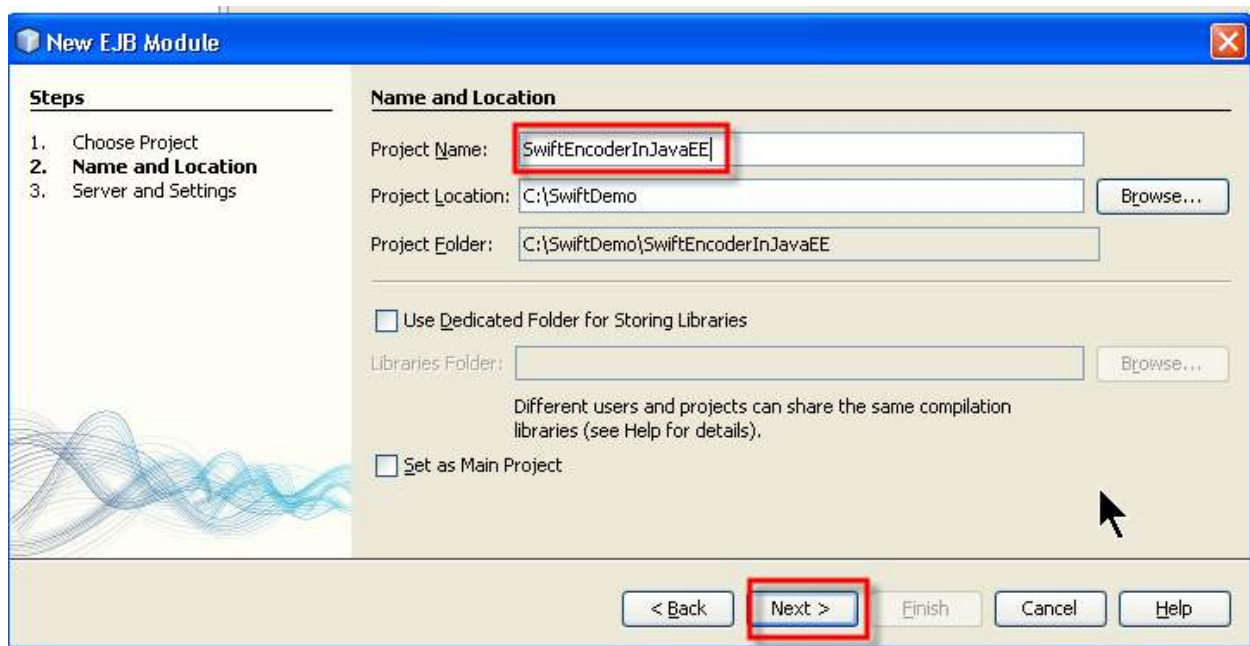
Here is a screen cast:

<http://swiftencoder.appspot.com/SwiftEncoderInJavaEE.html>

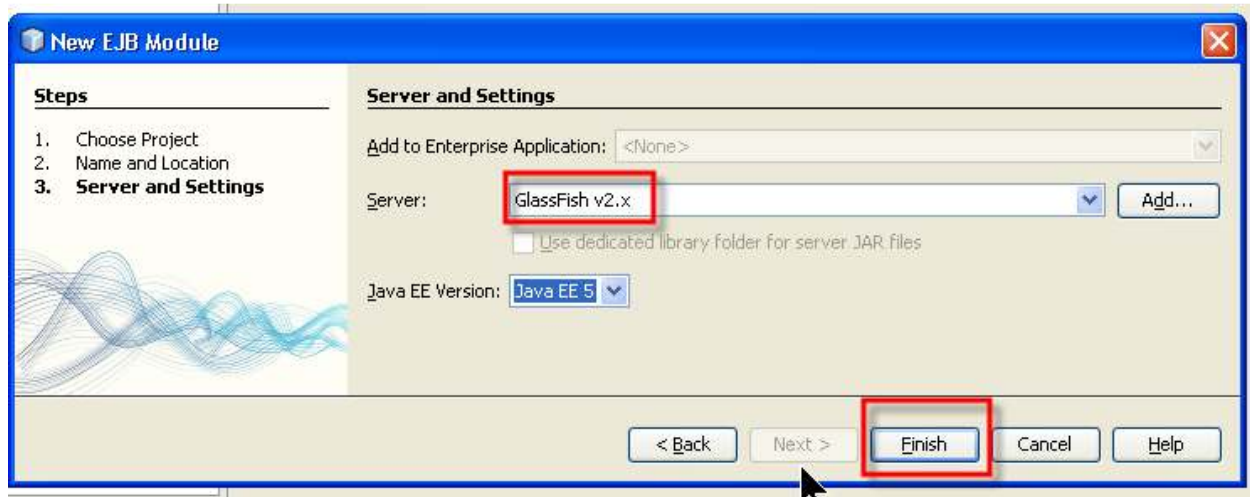
## Create a project Java EE → EJB Module



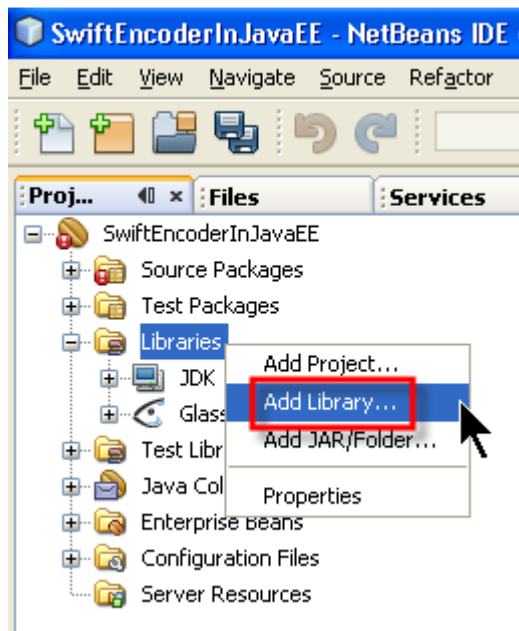
Give name “SwiftEncoderInJavaEE”



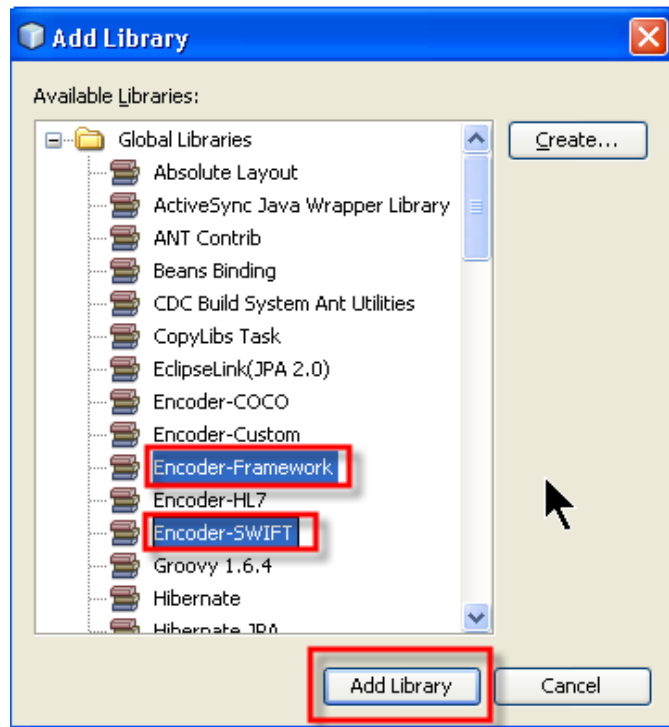
Choose the default server setting.



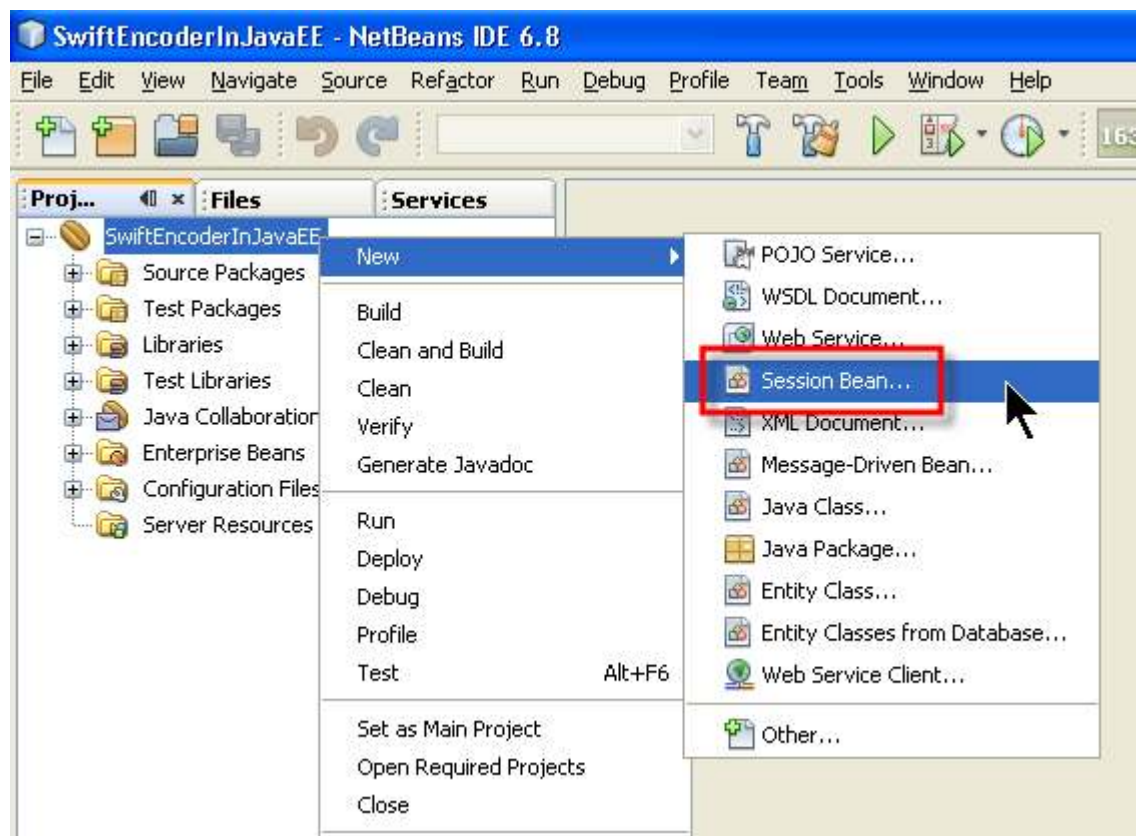
Add Library ...



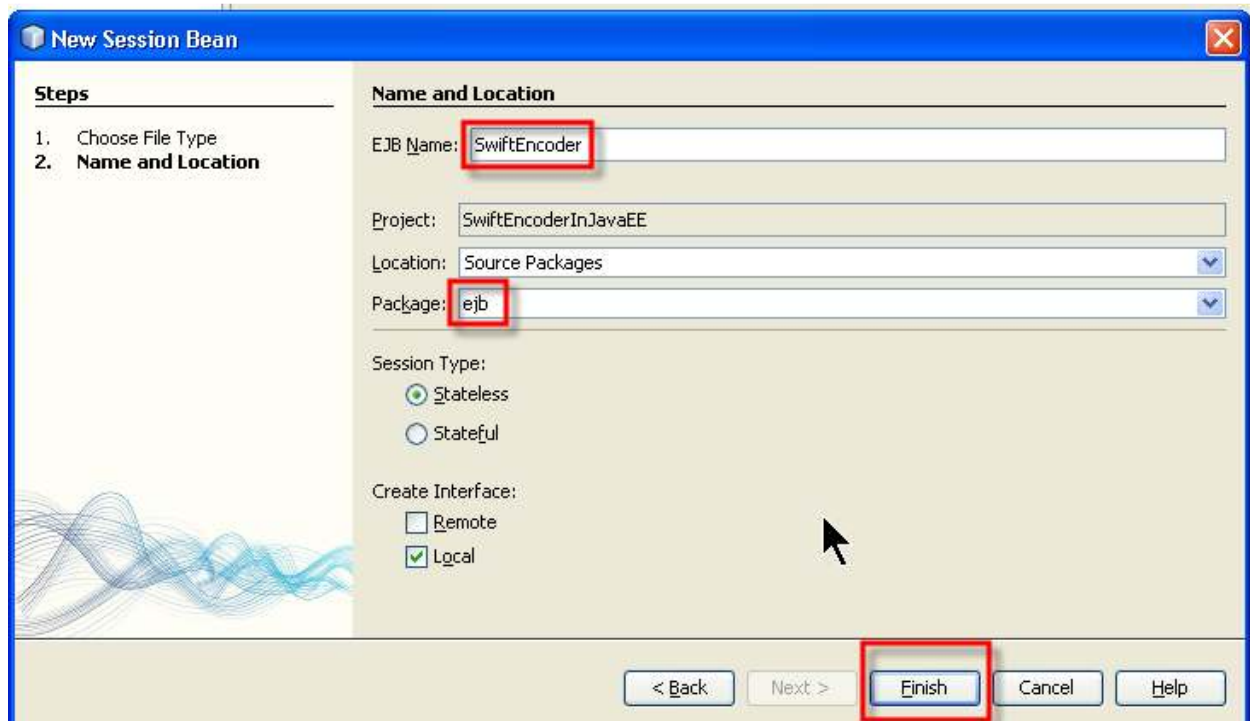
Choose libraries “Encoder-Framework” and “Encoder-SWIFT”.



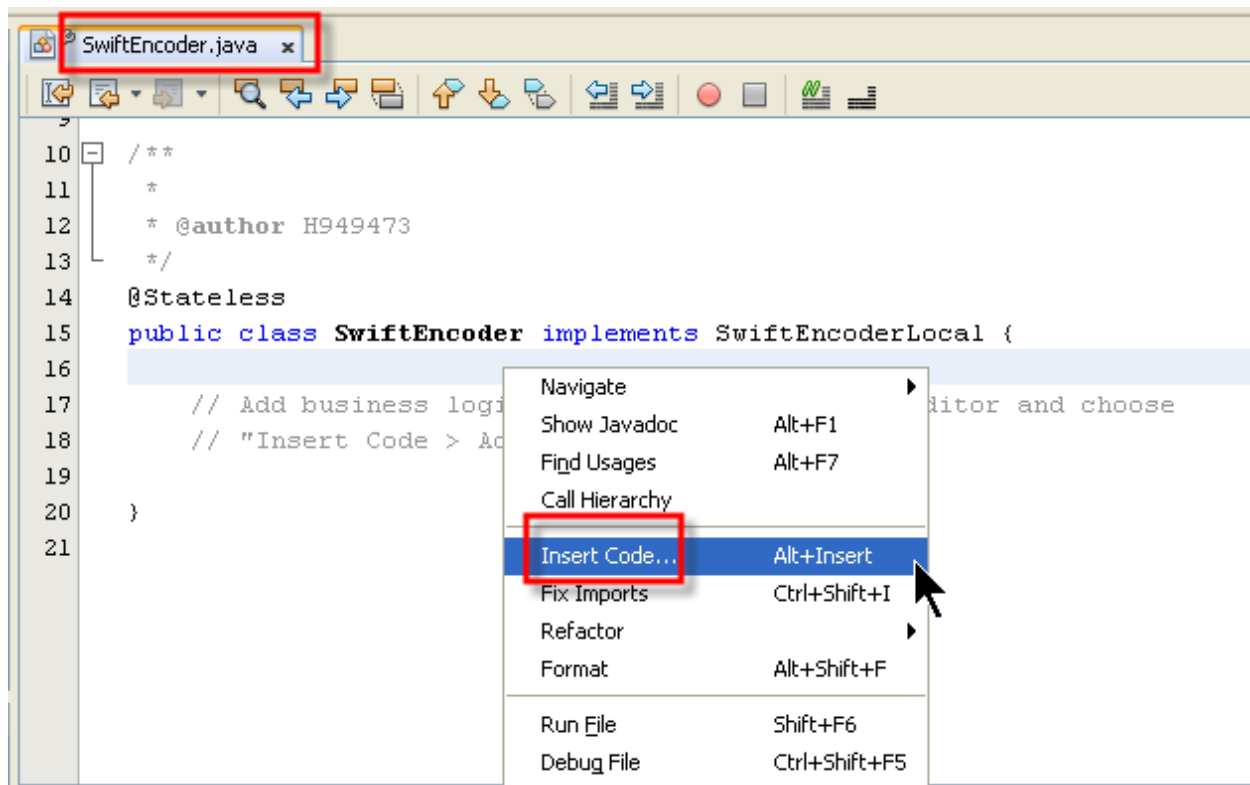
Create a [Session Bean ...](#)



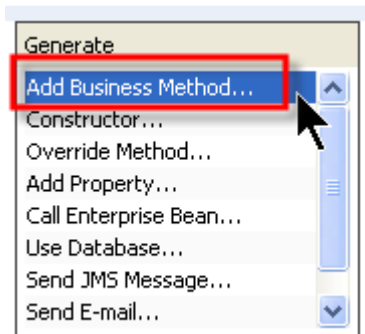
Give Bean name “SwiftEncoder”, package “ejb”.



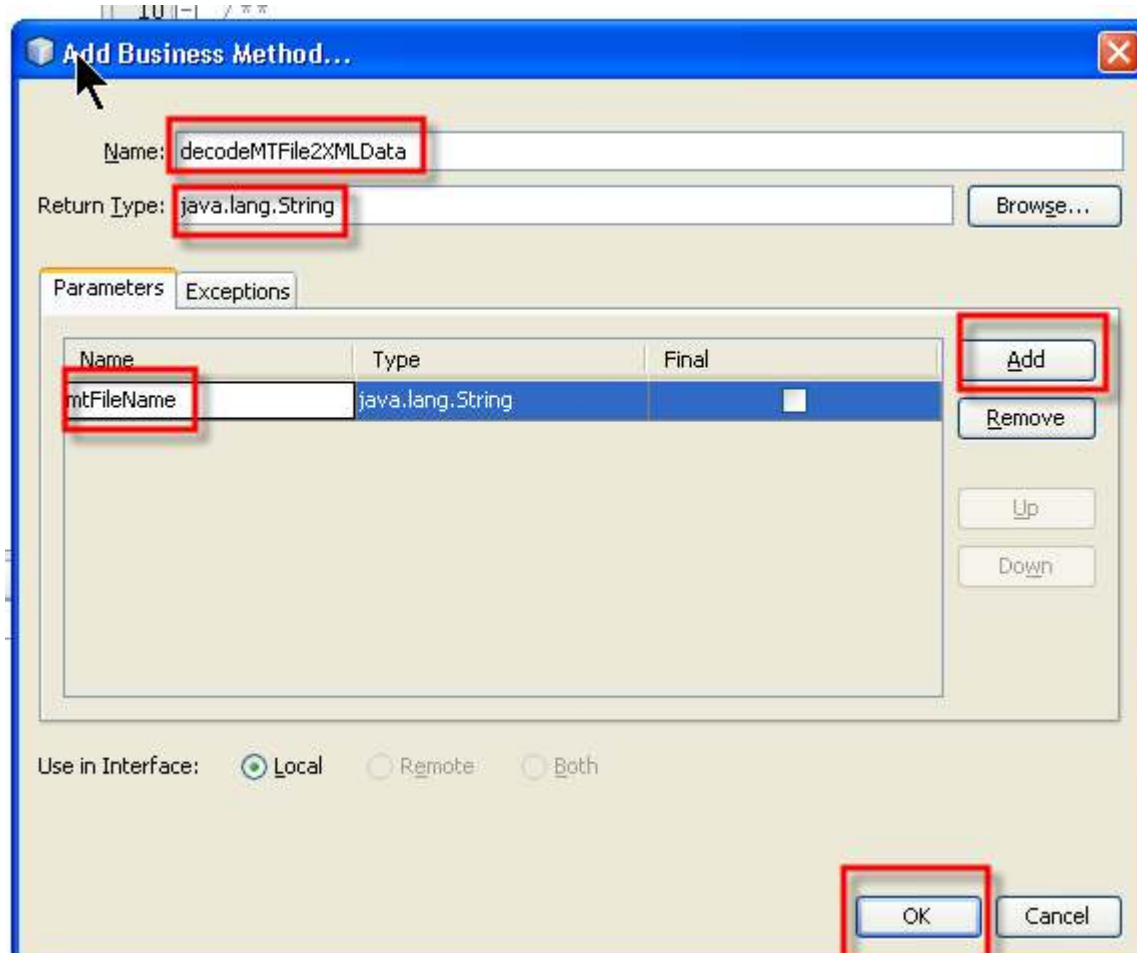
Right-click in source panel, select “Insert Code ...”



Choose “Add Business Method ...” from pop-up menu

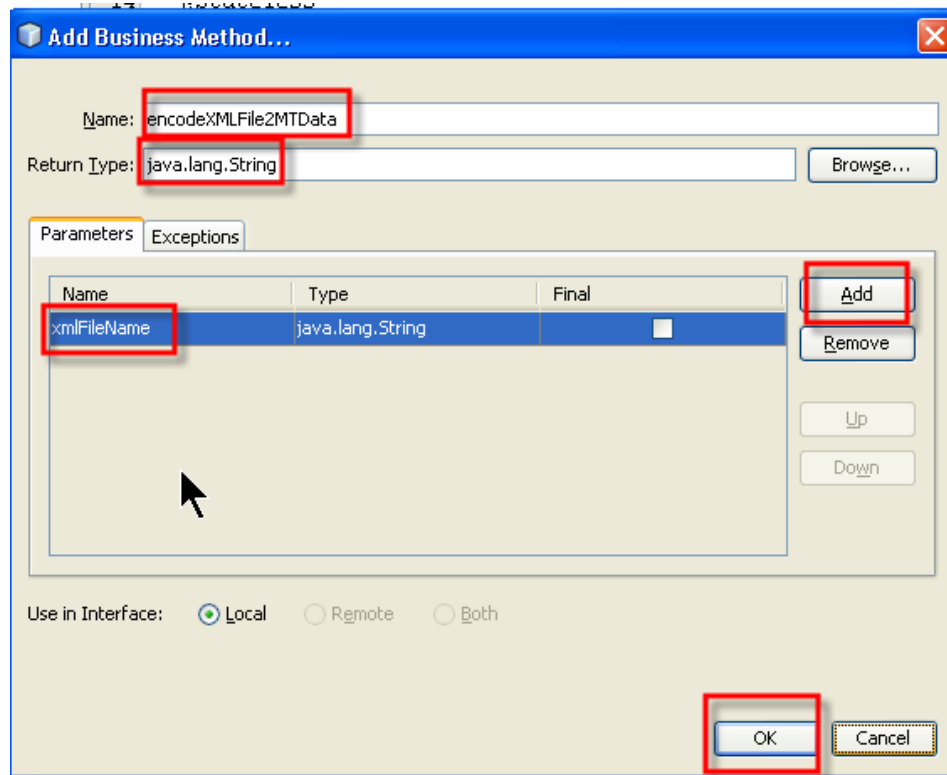


Give name “decodeMTFile2XMLData”, return type “java.lang.String”, add Parameter “mtFileName” with type “java.lang.String”.

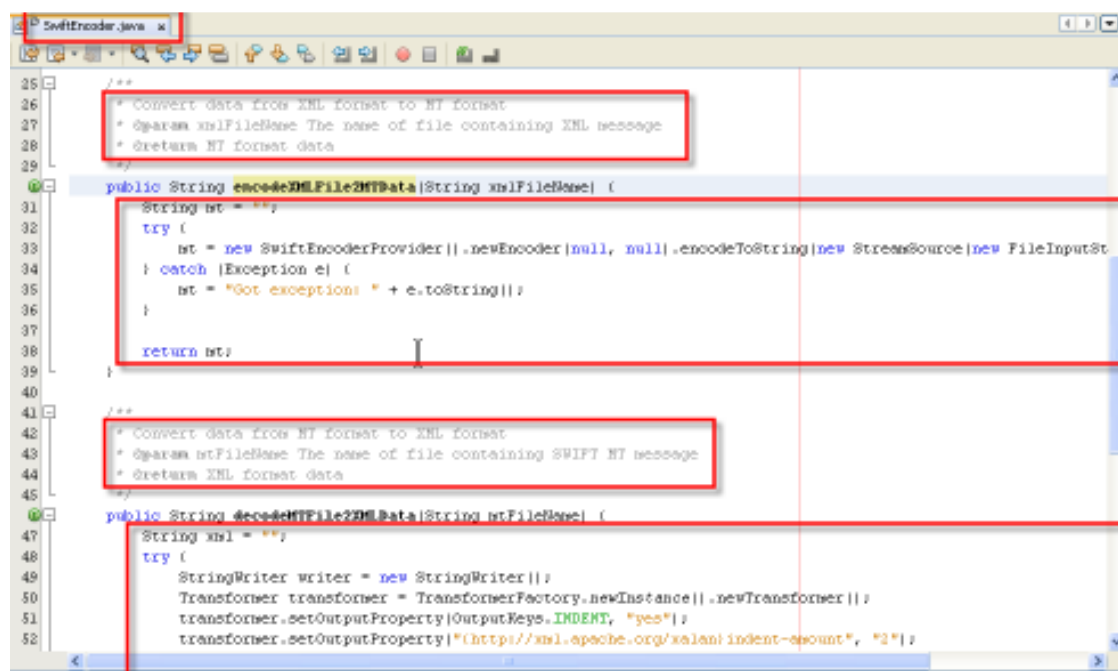


Add another method ...

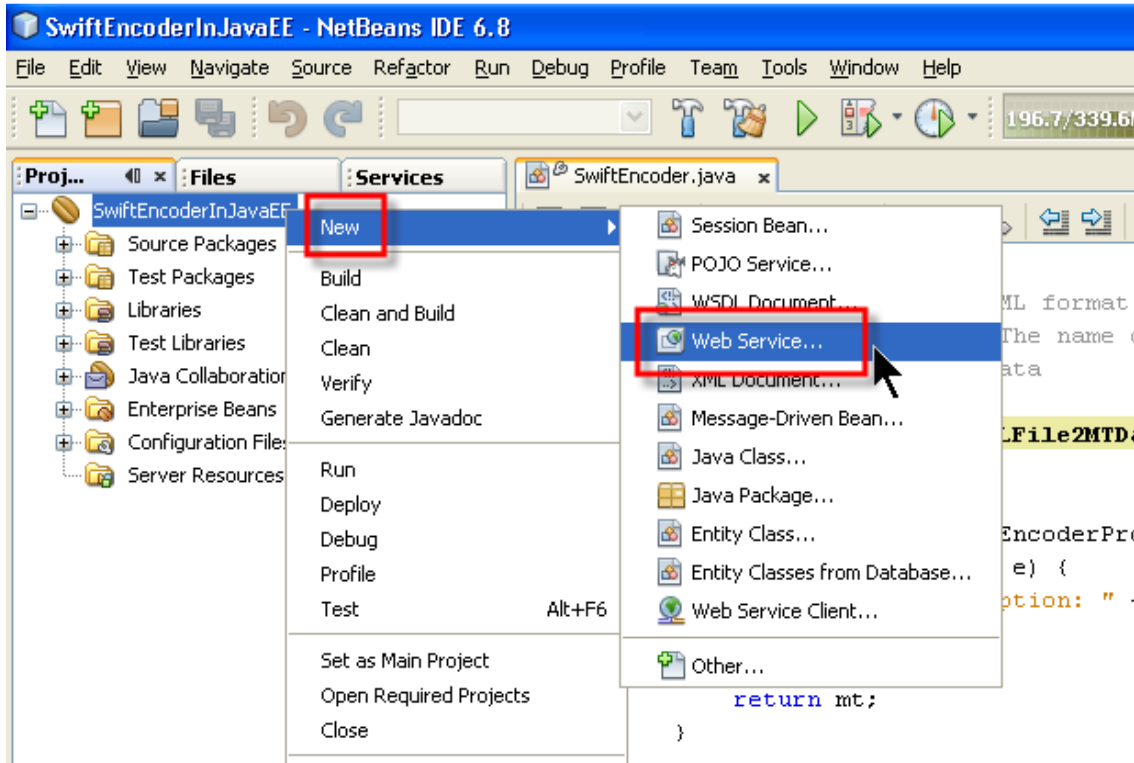
Name “`encodeXMLFile2Data`”, return type “`java.lang.String`”, add Parameter “`xmlFileName`” with type “`java.lang.String`”.



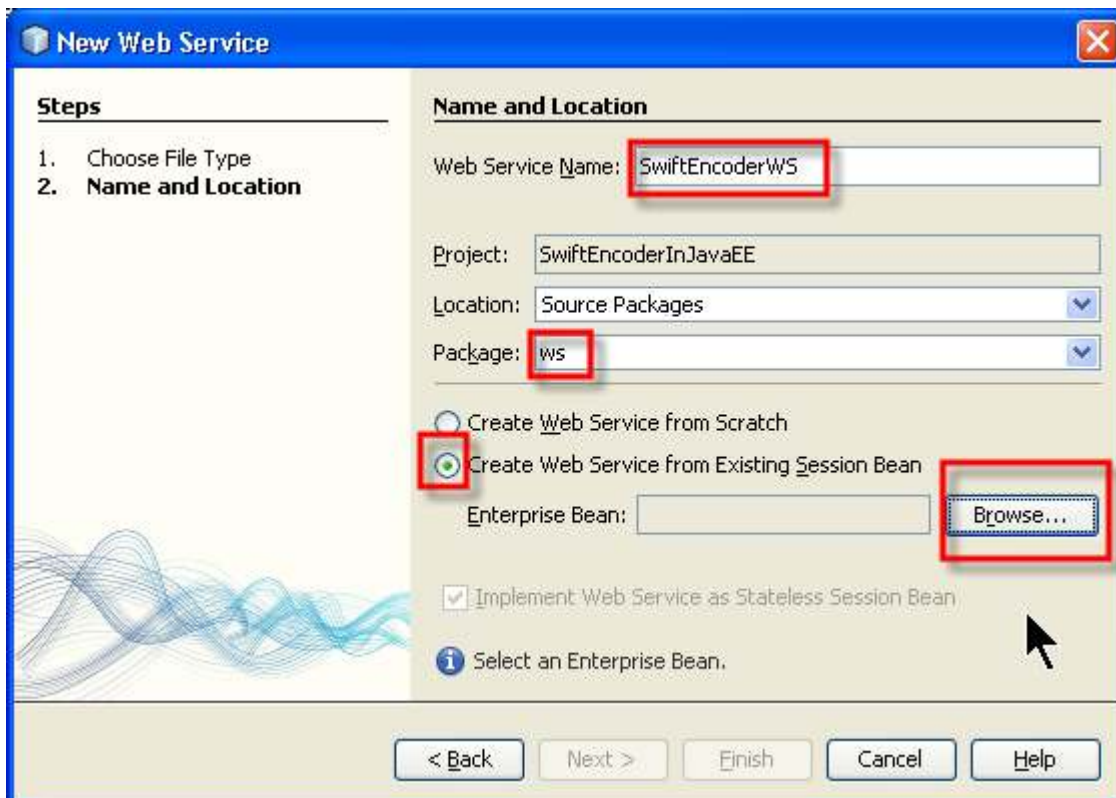
Write codes for two methods ...



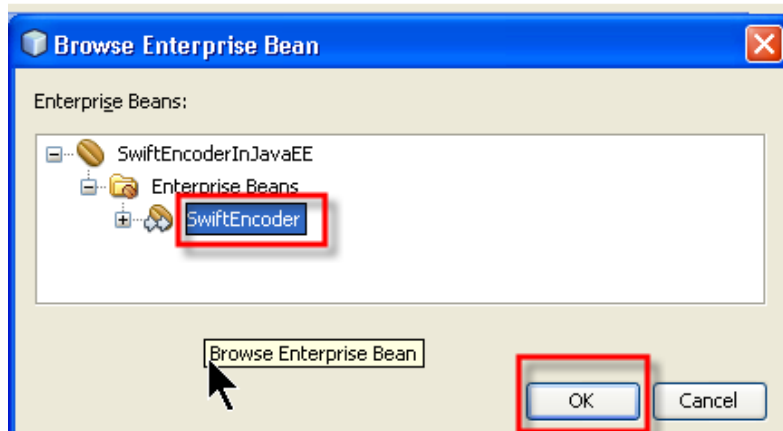
Create a [web-service](#) for Session Bean. [New ... Web Service ...](#)



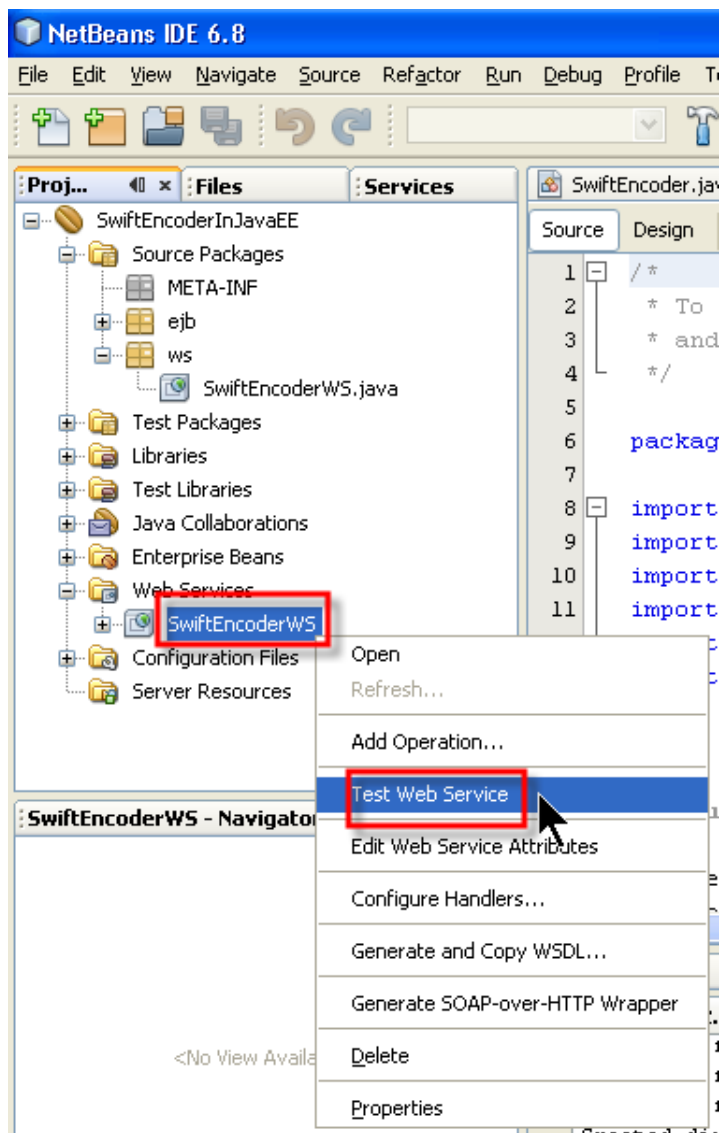
Name "SwiftEncoderWS", package "ws", click "Browse" button.



Select EJB “SwiftEncoder”.



Right-click web-service “SwiftEncoderWS”, select “Test Web Service”



A browser open the Tester Web Service URL, two operations appear as two buttons: “[decodeMTFile2XMLData](#)” and “[encodeXMFile2MTData](#)”.

Specify a test MT data file in the text box next to button “[decodeMTFile2XMLData](#)”, e.g. “[C:\SwiftDemo\mt541.dat](#)”, then click button “[decodeMTFile2XMLData](#)”.



SwiftEncoderWSService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

---

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

**Methods :**

```
public abstract java.lang.String test.ws.SwiftEncoderWS.decodeMTFile2XMLData(java.lang.String)
```

[decodeMTFile2XMLData](#) (  )

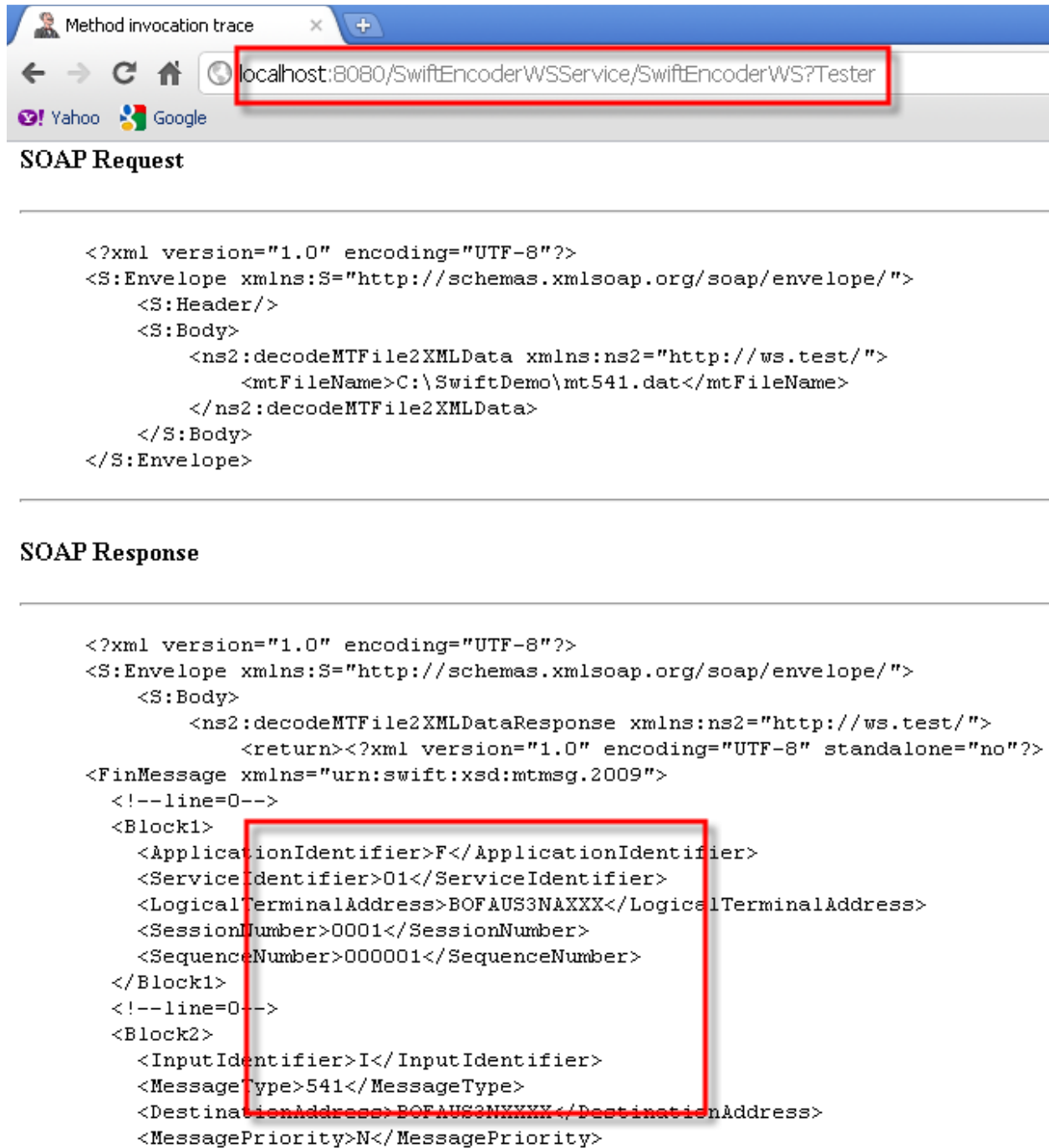
---

```
public abstract java.lang.String test.ws.SwiftEncoderWS.encodeXMLFile2MTData(java.lang.String)
```

[encodeXMLFile2MTData](#) (  )

---

The decoded XML is returned with the SOAP response.



Method invocation trace

localhost:8080/SwiftEncoderWSService/SwiftEncoderWS?Tester

Yahoo Google

### SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:decodeMTFile2XMLData xmlns:ns2="http://ws.test/">
      <mtFileName>C:\SwiftDemo\mt541.dat</mtFileName>
    </ns2:decodeMTFile2XMLData>
  </S:Body>
</S:Envelope>
```

### SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:decodeMTFile2XMLDataResponse xmlns:ns2="http://ws.test/">
      <return><?xml version="1.0" encoding="UTF-8" standalone="no"?>
<FinMessage xmlns="urn:swift:xsd:mtmsg.2009">
  <!--line=0-->
  <Block1>
    <ApplicationIdentifier>F</ApplicationIdentifier>
    <ServiceIdentifier>01</ServiceIdentifier>
    <LogicalTerminalAddress>BOFAUS3NAXXX</LogicalTerminalAddress>
    <SessionNumber>0001</SessionNumber>
    <SequenceNumber>000001</SequenceNumber>
  </Block1>
  <!--line=0-->
  <Block2>
    <InputIdentifier>I</InputIdentifier>
    <MessageType>541</MessageType>
    <DestinationAddress>BOFAUS3NXXXX</DestinationAddress>
    <MessagePriority>N</MessagePriority>
```

Now test `encode` function ...

Specify an XML file in the text box next to button “`encodeXMLFile2MTData`”, e.g. “`C:\SwiftDemo\mt541.xml`”, then click button “`encodeXMLFile2MTData`”.

SwiftEncoderWSService Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

---

To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

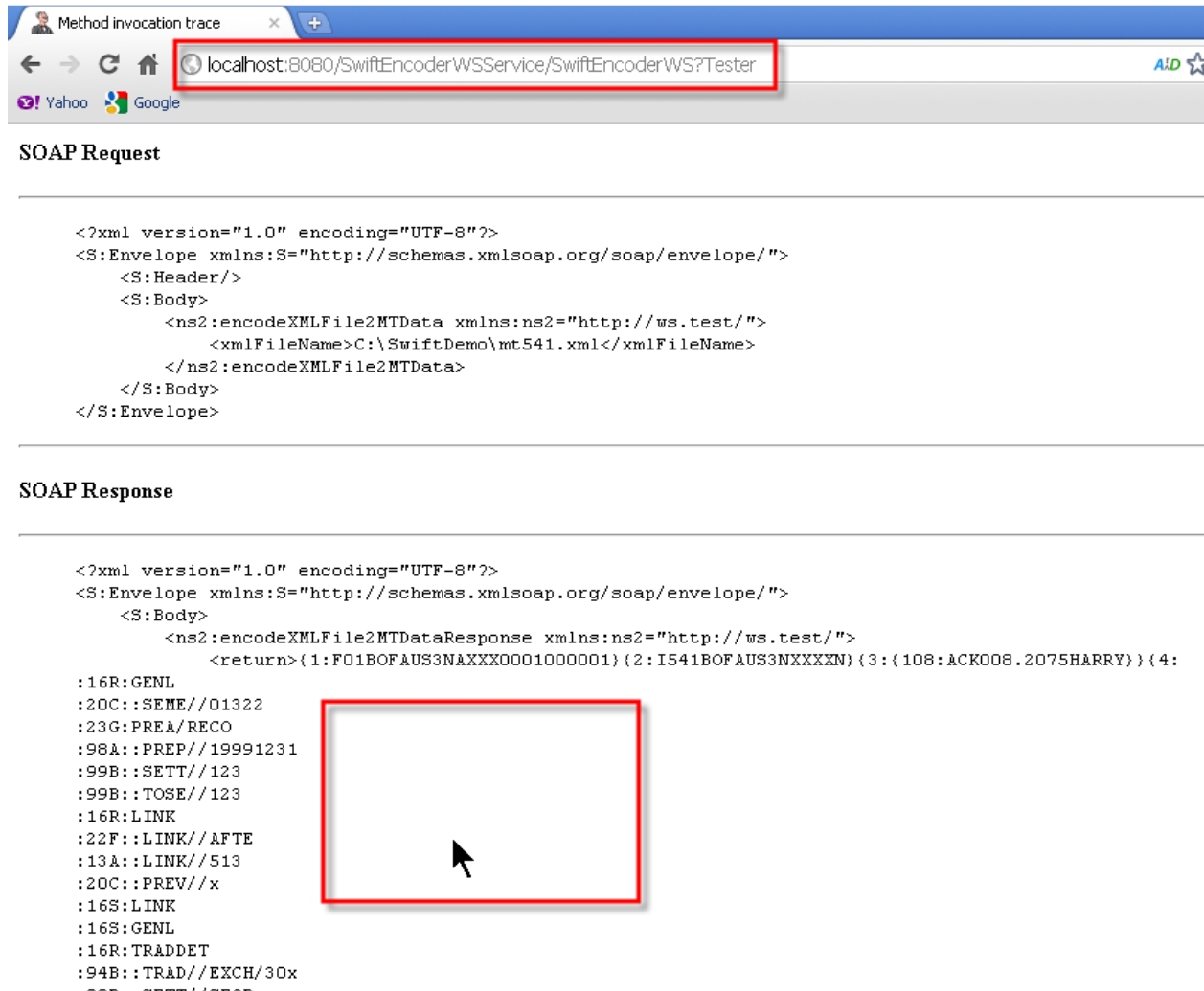
**Methods :**

public abstract java.lang.String test.ws.SwiftEncoderWS.decodeMTFile2XMLData(java.lang.String)  
decodeMTFile2XMLData (  )

---

public abstract java.lang.String test.ws.SwiftEncoderWS.encodeXMLFile2MTData(java.lang.String)  
encodeXMLFile2MTData (  )

You can see the MT data in SOAP response.



Method invocation trace

localhost:8080/SwiftEncoderWSService/SwiftEncoderWS?Tester

### SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Header/>
  <S:Body>
    <ns2:encodeXMLFile2MTData xmlns:ns2="http://ws.test/">
      <xmlFileName>C:\SwiftDemo\mt541.xml</xmlFileName>
    </ns2:encodeXMLFile2MTData>
  </S:Body>
</S:Envelope>
```

### SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:encodeXMLFile2MTDataResponse xmlns:ns2="http://ws.test/">
      <return>{1:F01BOFAUS3NAXXX0001000001}{2:I541BOFAUS3NXXXXXN}{3:{108:ACK008.2075HARRY}}{4:
:16R:GENL
:20C::SEME//01322
:23G:PREA/RECO
:98A::PREP//19991231
:99B::SETT//123
:99B::TOSE//123
:16R:LINK
:22F::LINK//AFTE
:13A::LINK//513
:20C::PREV//x
:16S:LINK
:16S:GENL
:16R:TRADDET
:94B::TRAD//EXCH/30x
:99B::SETT//16EOP
```