Using the new Eclipse 3.2 BPEL Designer with JBoss BPEL engine

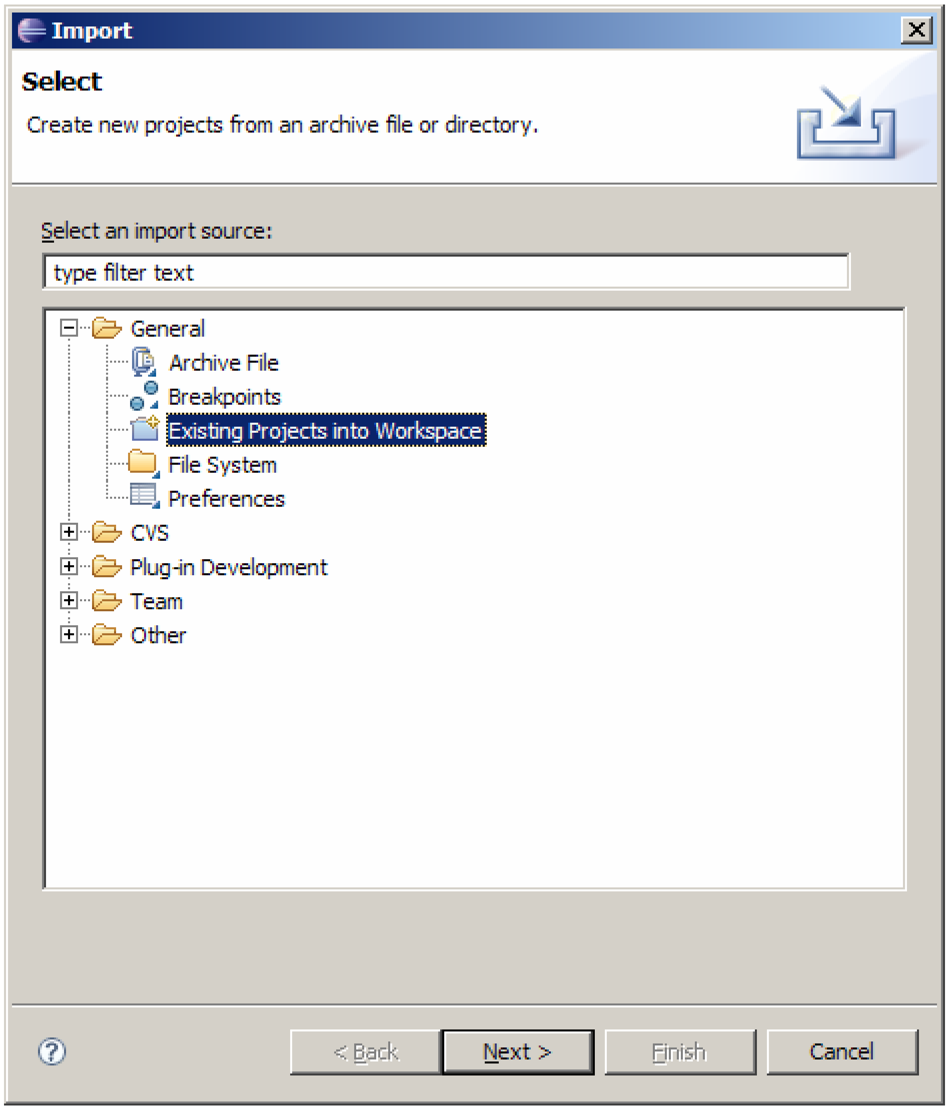
Eclipse 3.2
http://www.eclipse.org/downloads/

BPEL Designer
http://www.eclipse.org/bpel/

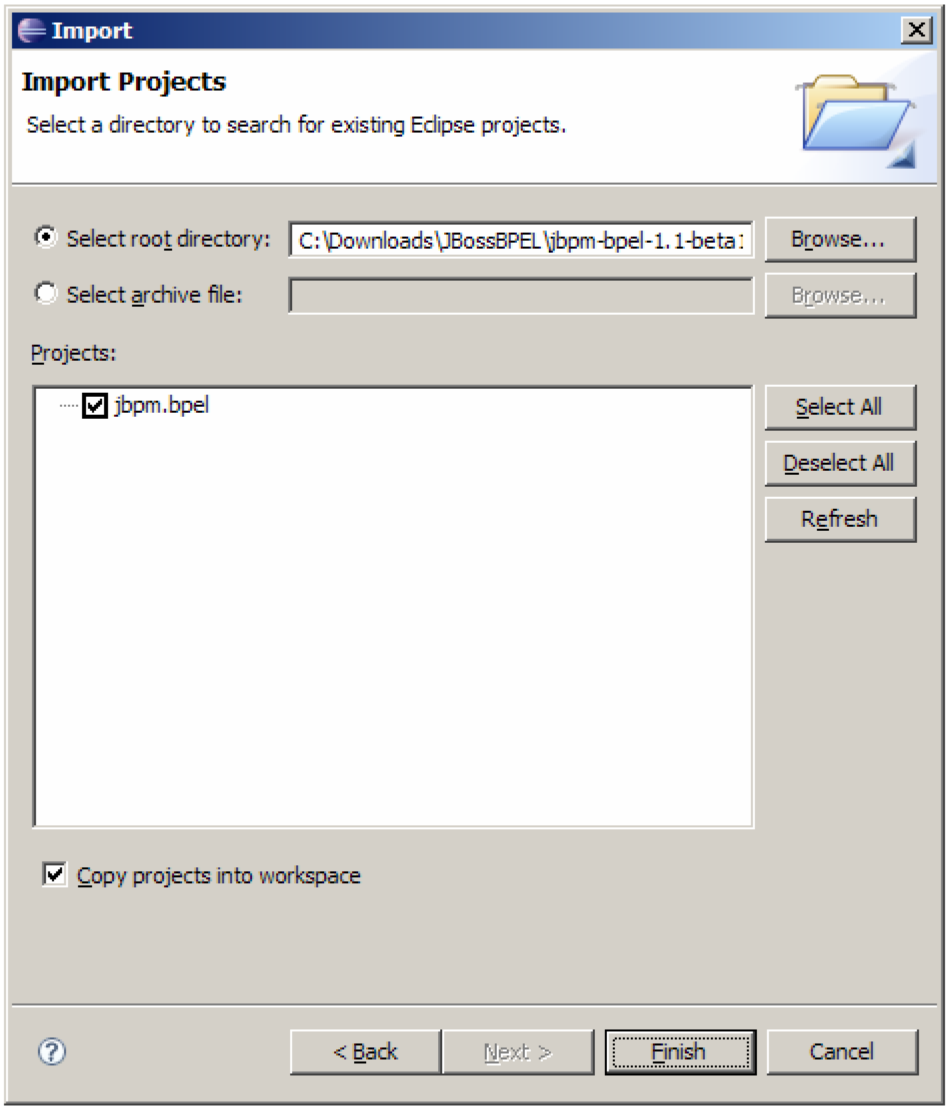
Note: A clean installation of these two items may be required. The process below also assumes that you've reviewed the BPEL User Guide in the "docs" directory and have correctly installed jBPM BPEL

1. Unzip the jBPM BPEL download into a location that you'll remember

2. Start Eclipse 3.2 and use the Import Existing Projects into Workspace feature from the File menu.

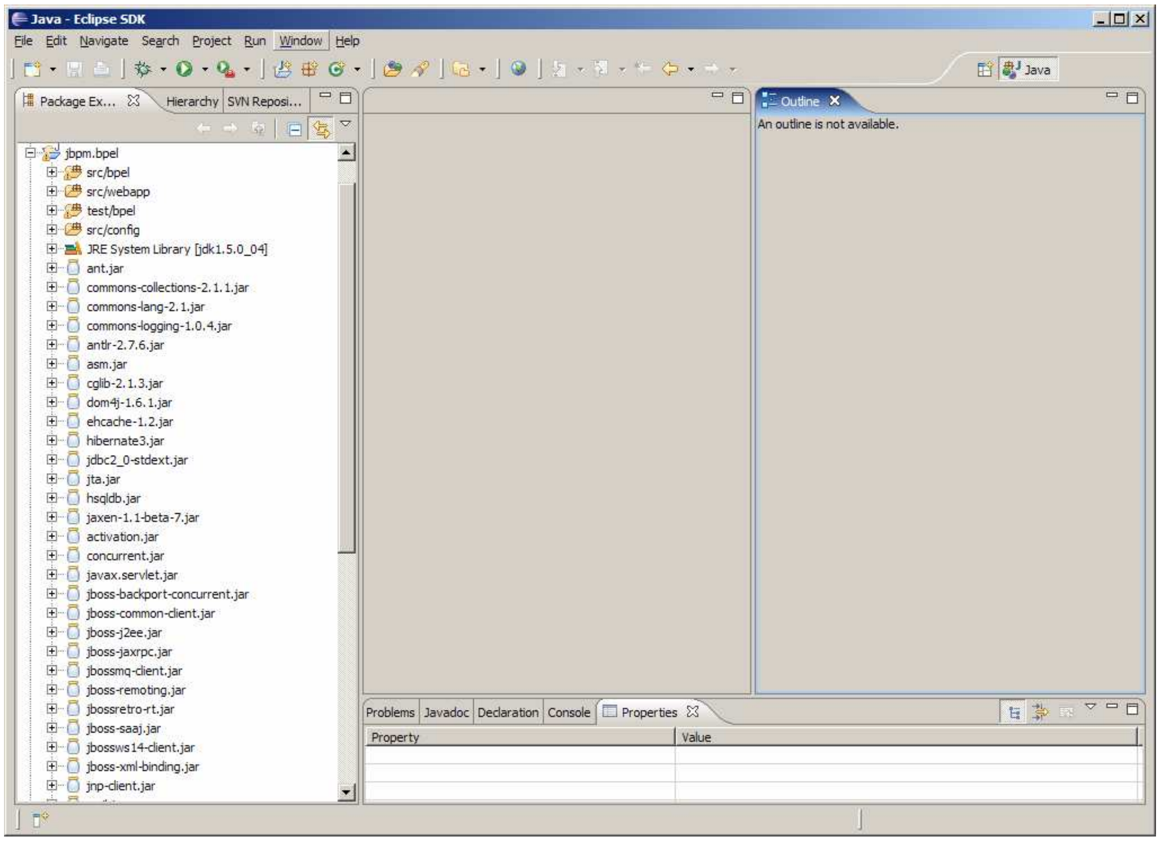


Use the Browse button to navigate to the directory where you unzipped the JBoss BPEL download.



Check "Copy projects into workspace" so you keep a clean copy in the original location.

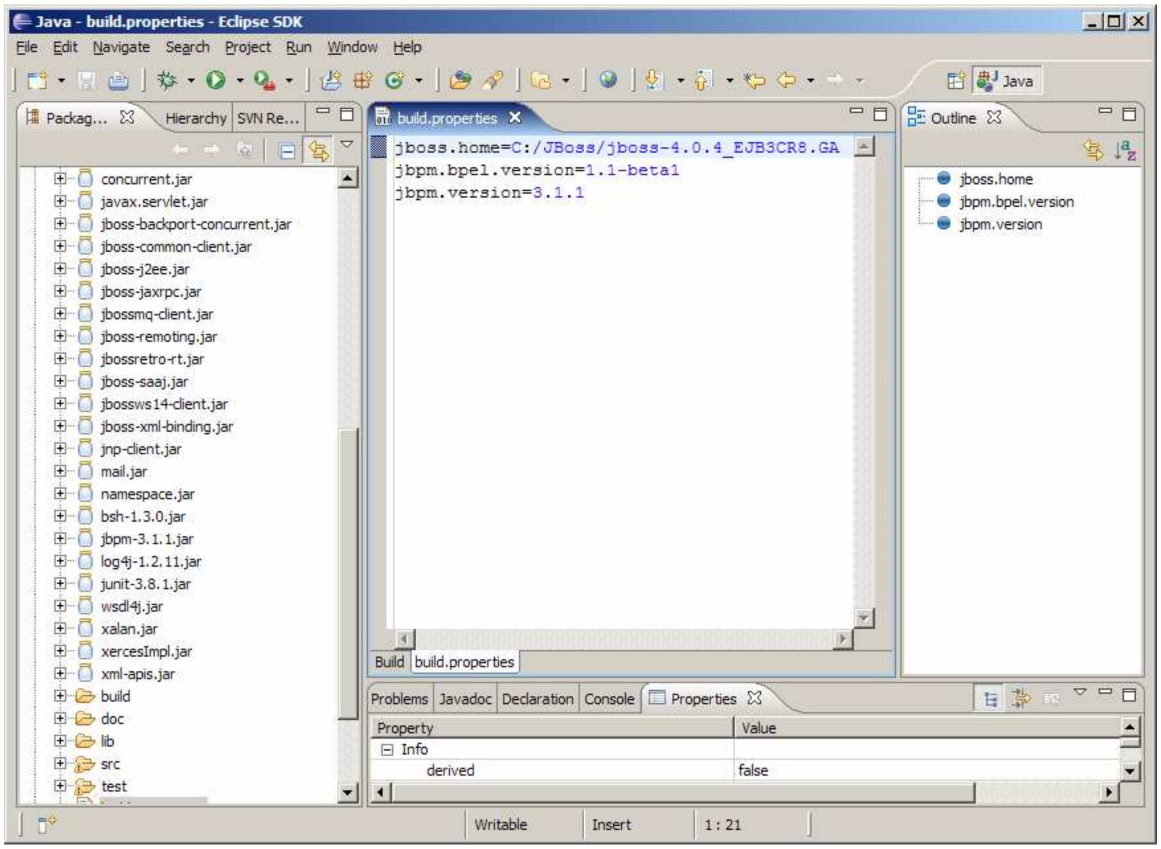
Once jBPM BPEL has been imported into Eclipse you should be able to navigate around its structure via the Package Explorer.


Java - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Java
Package Ex... Hierarchy SVN Reposi...
Outline
Problems Javadoc Declaration Console Properties
Property
Value

```
jbpm.bpel
  src/bpel
  src/webapp
  test/bpel
  src/config
  JRE System Library [jdk1.5.0_04]
  ant.jar
  commons-collections-2.1.1.jar
  commons-lang-2.1.jar
  commons-logging-1.0.4.jar
  antlr-2.7.6.jar
  asm.jar
  cglib-2.1.3.jar
  dom4j-1.6.1.jar
  ehcache-1.2.jar
  hibernate3.jar
  jdbc2_0-stdext.jar
  jta.jar
  hsqldb.jar
  jaxen-1.1-beta-7.jar
  activation.jar
  concurrent.jar
  javax.servlet.jar
  jboss-backport-concurrent.jar
  jboss-common-client.jar
  jboss-j2ee.jar
  jboss-jaxrpc.jar
  jbossmq-client.jar
  jboss-remoting.jar
  jbossretro-rt.jar
  jboss-saaj.jar
  jbossws14-client.jar
  jboss-xml-binding.jar
  jnp-client.jar
```

An outline is not available.

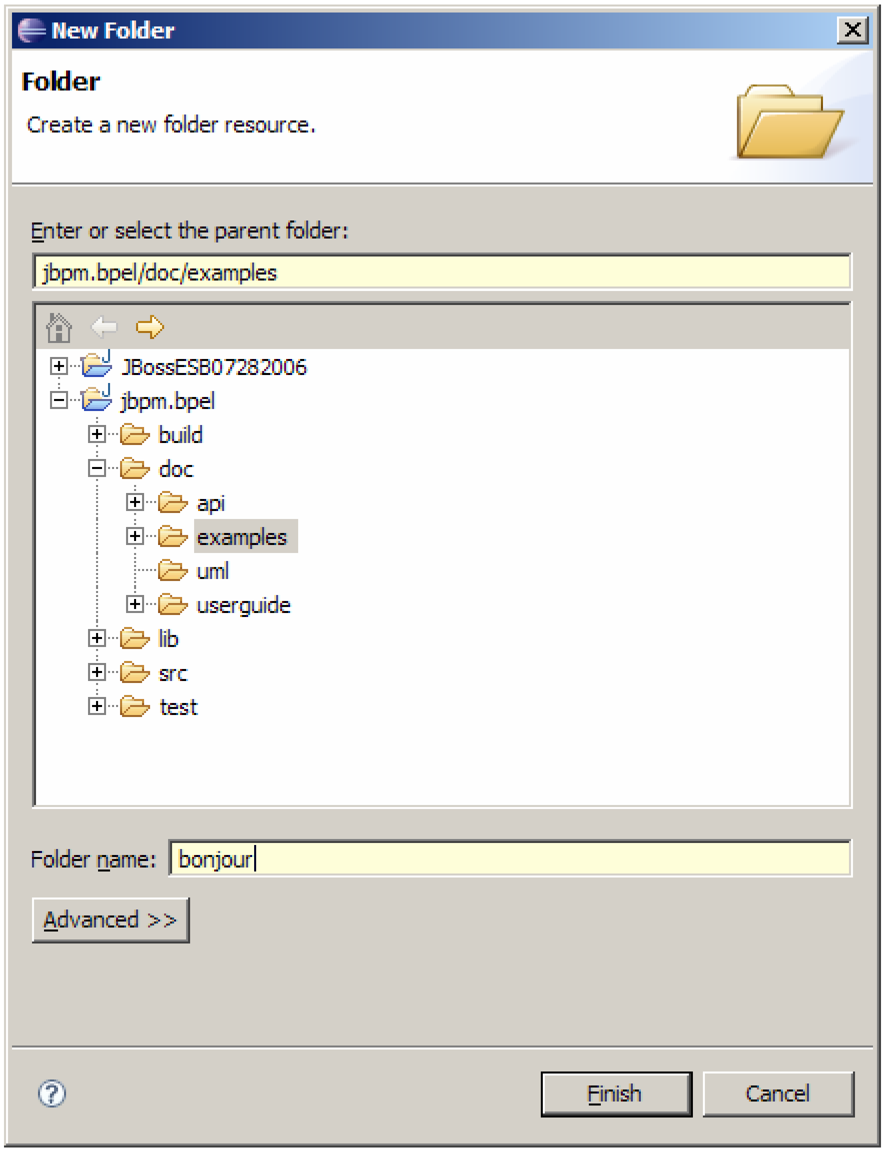
3. Modify build.properties to point jboss.home to your installation of JBoss 4.0.4.GA



```
jboss.home=C:/JBoss/jboss-4.0.4_EJB3CR8.GA
jbpm.bpel.version=1.1-beta1
jbpm.version=3.1.1
```

Property	Value
Info	
derived	false

4. Under “doc/examples” add a sub-folder called “bonjour” by right-clicking on “examples” and selecting New Folder from the pop-up context menu.


New Folder

Folder

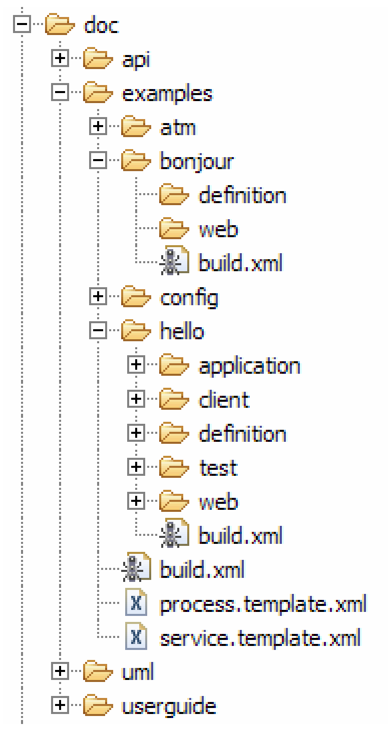
Create a new folder resource.

Enter or select the parent folder: jbpm.bpel/doc/examples

```
JBossESB07282006
jbpm.bpel
    build
    doc
        api
        examples
        uml
        userguide
    lib
    src
    test
```

Folder name: bonjour

5. Under the “bonjour” folder, add “definition” and “web” sub-folders.
6. Copy “build.xml” from the hello folder to the bonjour folder.
7. Modify build.xml so that the project name=”bonjour”

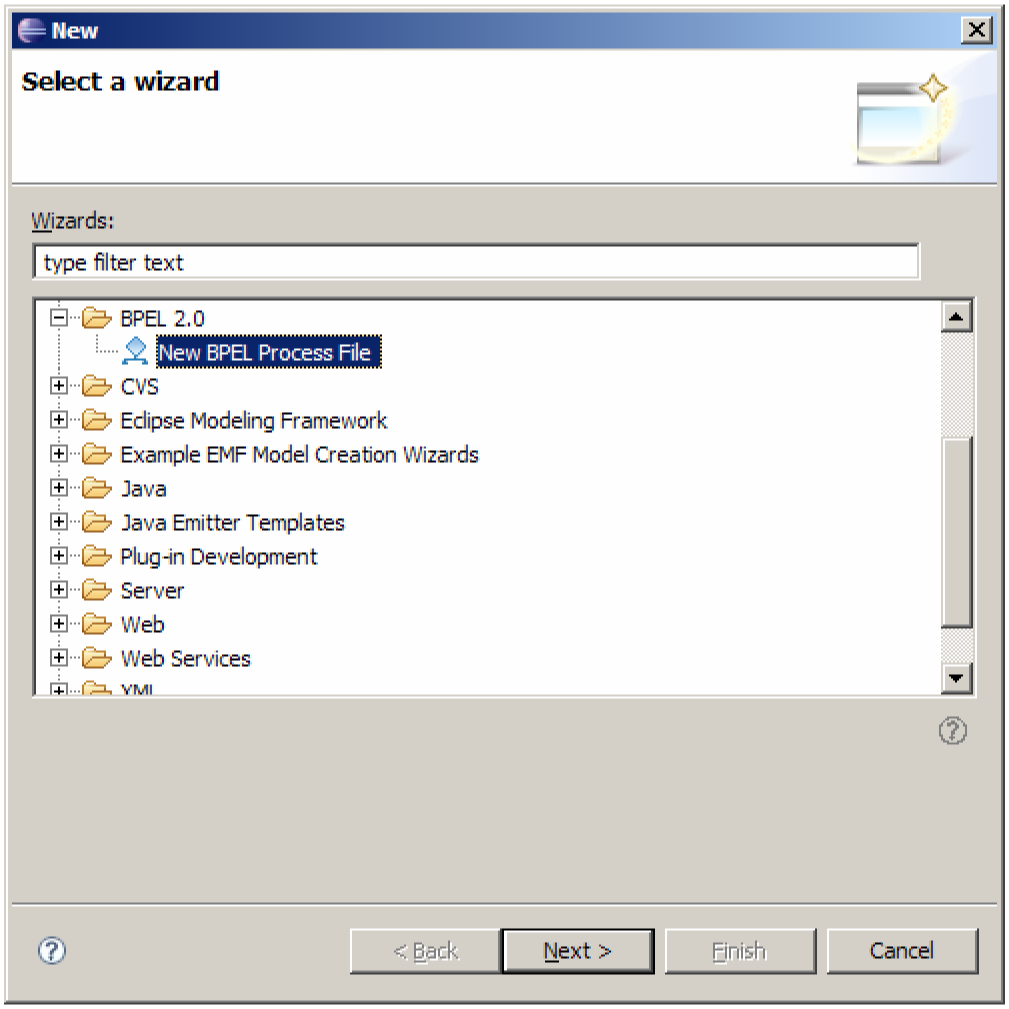


```
<?xml version="1.0"?>
<project name="bonjour" default="deploy-application" basedir=".">

  <import file="../process.template.xml"/>

</project>
```

8. Modify the build.properties file in doc\examples\config:
WSDP is the Web Services Developer Pack from Sun.
http://java.sun.com/webservices/downloads/webservicespack.html
It is used by the custom Ant tasks used in the BPEL “process.template.xml”
JBoss Home (jboss.home) is where you have installed 4.0.4.GA
jBPM BPEL Home (jbpm.bpel.home) is where this project has been imported to

```
doc\examples\config\build.properties
wsdp.home=C:/Tools/jwsdp-2.0
jboss.home=C:/JBoss/jboss-4.0.4_EJB3CR8.GA
jbpm.bpel.home=c:/EclipseLabs3.2/jbpm.bpel
jboss.server=default
jbpm.version=3.1.1
jbpm.bpel.version=1.1-beta1
```

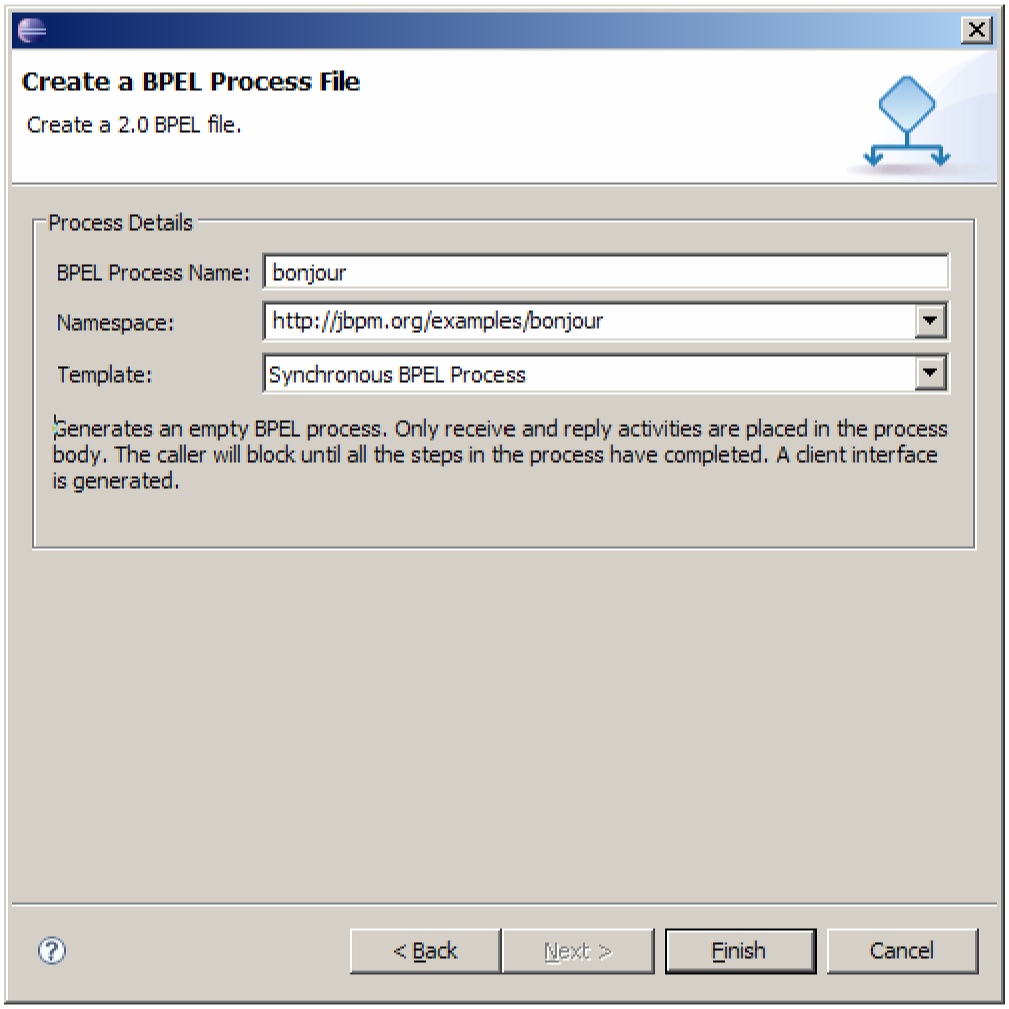
9. Right click on the “definition” folder and select New – Other… then select New BPEL Process File.



Select Next

9. BPEL Process Name: bonjour
   Namespace: http://jbpm.org/examples/bonjour
   Synchronous BPEL Process

Select Finish



Create a BPEL Process File

Create a 2.0 BPEL file.

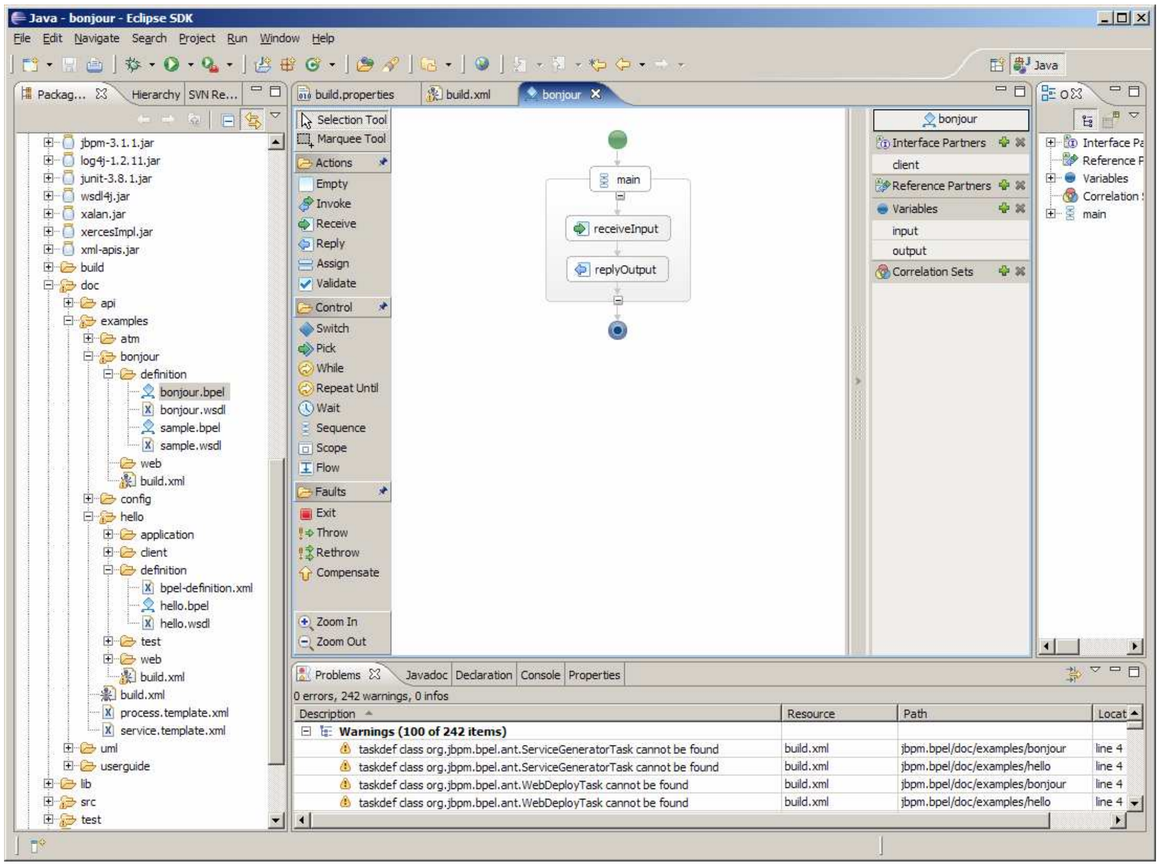
Process Details

BPEL Process Name: bonjour

Namespace: http://jbpm.org/examples/bonjour

Template: Synchronous BPEL Process

Generates an empty BPEL process. Only receive and reply activities are placed in the process body. The caller will block until all the steps in the process have completed. A client interface is generated.



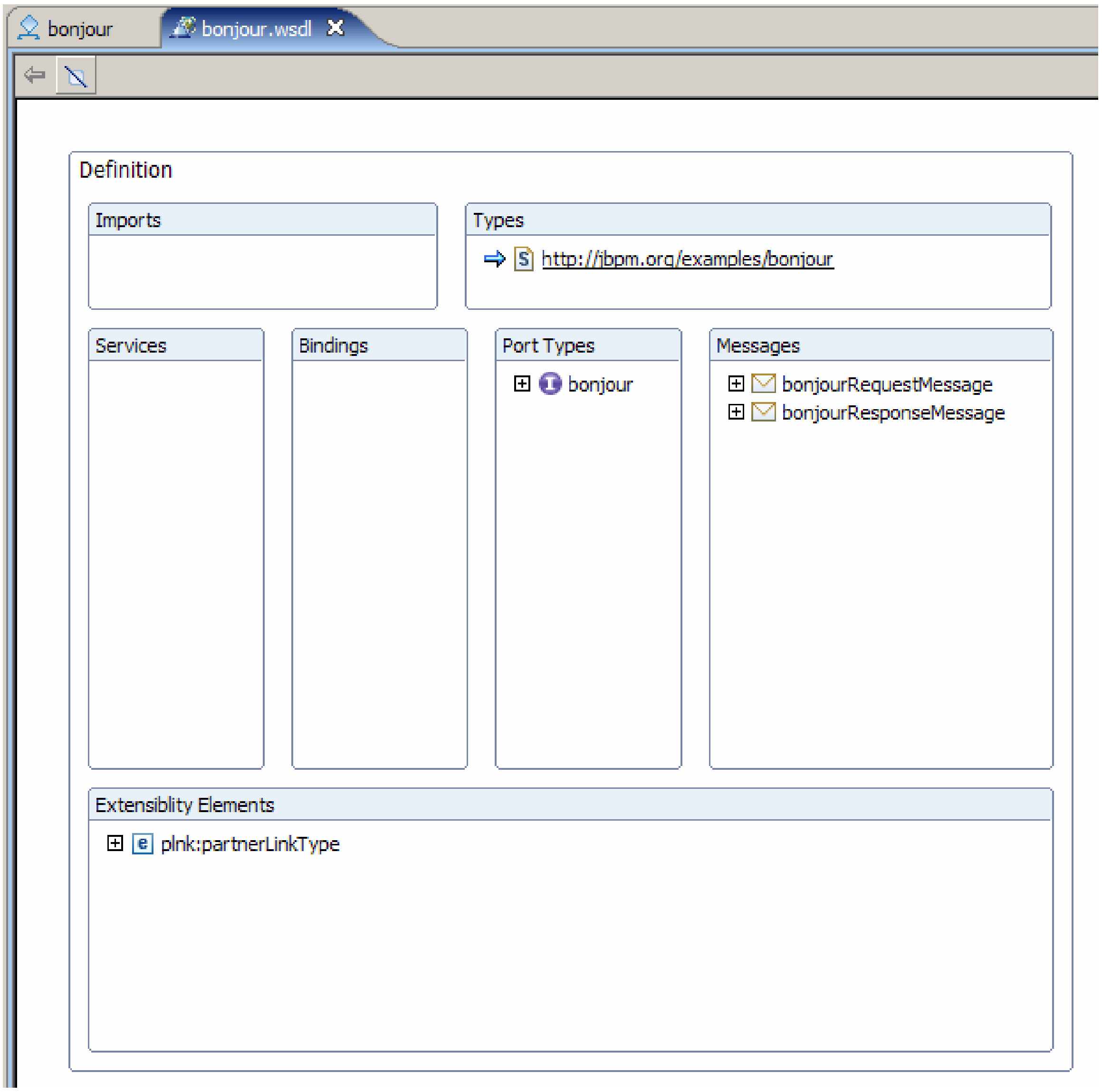
0 errors, 242 warnings, 0 infos

Description	Resource	Path	Locat
Warnings (100 of 242 items)			
taskdef class org.jbpm.bpel.ant.ServiceGeneratorTask cannot be found	build.xml	jbpm.bpel/doc/examples/bonjour	line 4
taskdef class org.jbpm.bpel.ant.ServiceGeneratorTask cannot be found	build.xml	jbpm.bpel/doc/examples/hello	line 4
taskdef class org.jbpm.bpel.ant.WebDeployTask cannot be found	build.xml	jbpm.bpel/doc/examples/bonjour	line 4
taskdef class org.jbpm.bpel.ant.WebDeployTask cannot be found	build.xml	jbpm.bpel/doc/examples/hello	line 4

Note: You can NOT currently open/edit a .bpel file that was created using this designer.

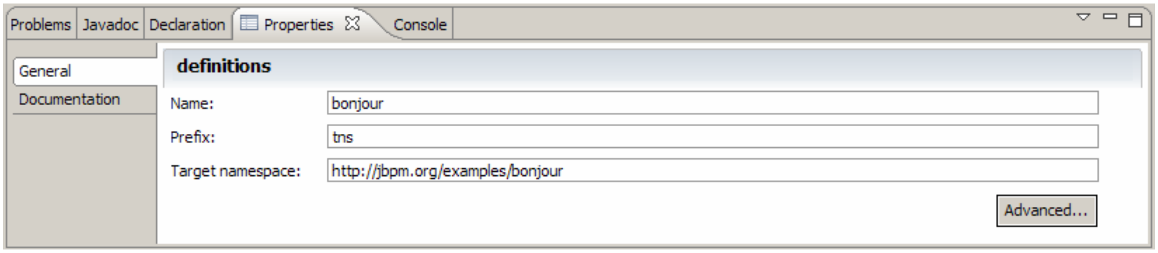
Your definition folder will now include a bonjour.bpel file, a bonjour.wsdl file and at the time of this writing a sample.bpel and sample.wsdl file which are not used in this tutorial

10. Open bonjour.wsdl



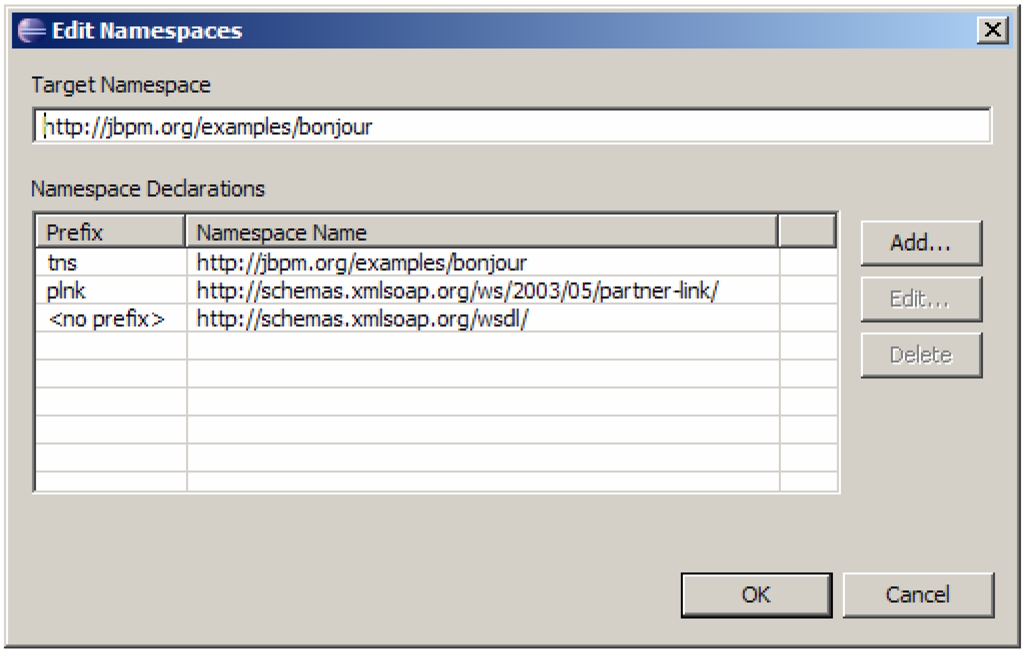
This is a screenshot of the WSDL visual editor that is included as part of the Eclipse BPEL Designer.

11. Click on the “Definition” area and find the Properties window (typically found at the bottom of Eclipse along side the Console, Problems, etc).



Name: bonjour
Prefix: tns
Target namespace: http://jbpm.org/examples/bonjour

12. Click on the Advanced… button

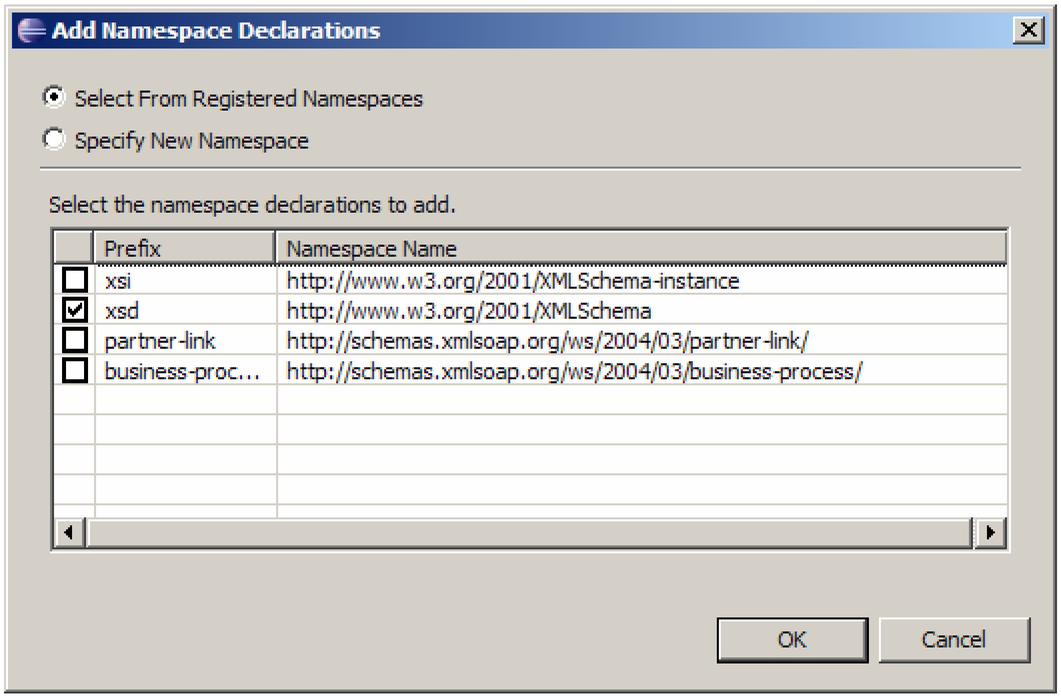


Target Namespace: http://jbpm.org/examples/bonjour

Namespace Declarations

Prefix	Namespace Name
tns	http://jbpm.org/examples/bonjour
plnk	http://schemas.xmlsoap.org/ws/2003/05/partner-link/
<no prefix>	http://schemas.xmlsoap.org/wsdl/

13. Click on Add.. and check “xsd”

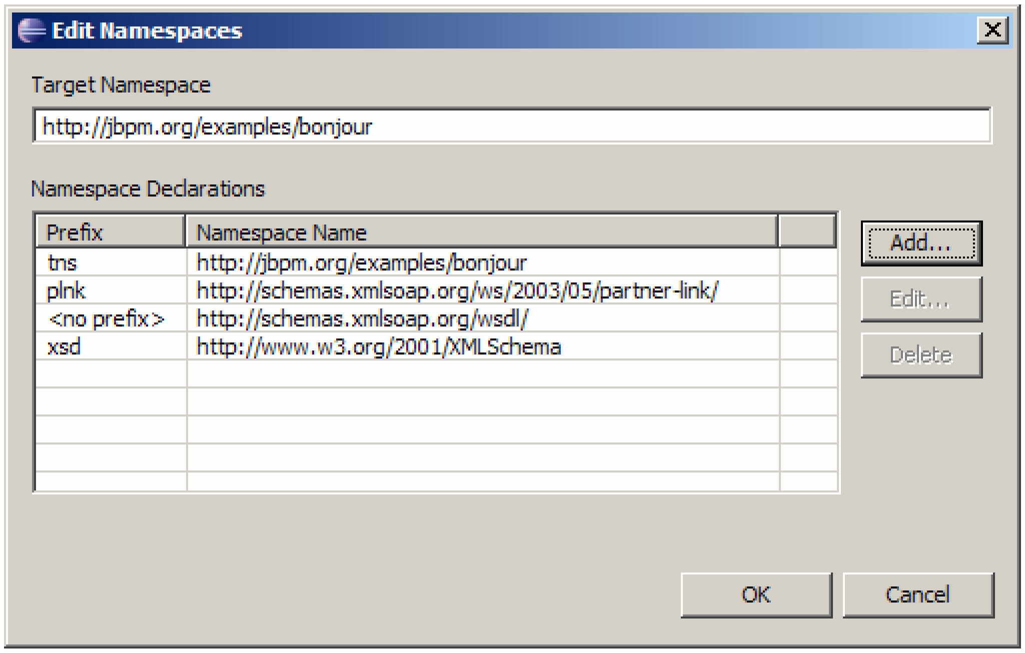


[✓ Select From Registered Namespaces] Specify New Namespace

Select the namespace declarations to add.

	Prefix	Namespace Name
☐	xsi	http://www.w3.org/2001/XMLSchema-instance
✓	xsd	http://www.w3.org/2001/XMLSchema
☐	partner-link	http://schemas.xmlsoap.org/ws/2004/03/partner-link/
☐	business-proc...	http://schemas.xmlsoap.org/ws/2004/03/business-process/

Click OK.



Target Namespace: http://jbpm.org/examples/bonjour

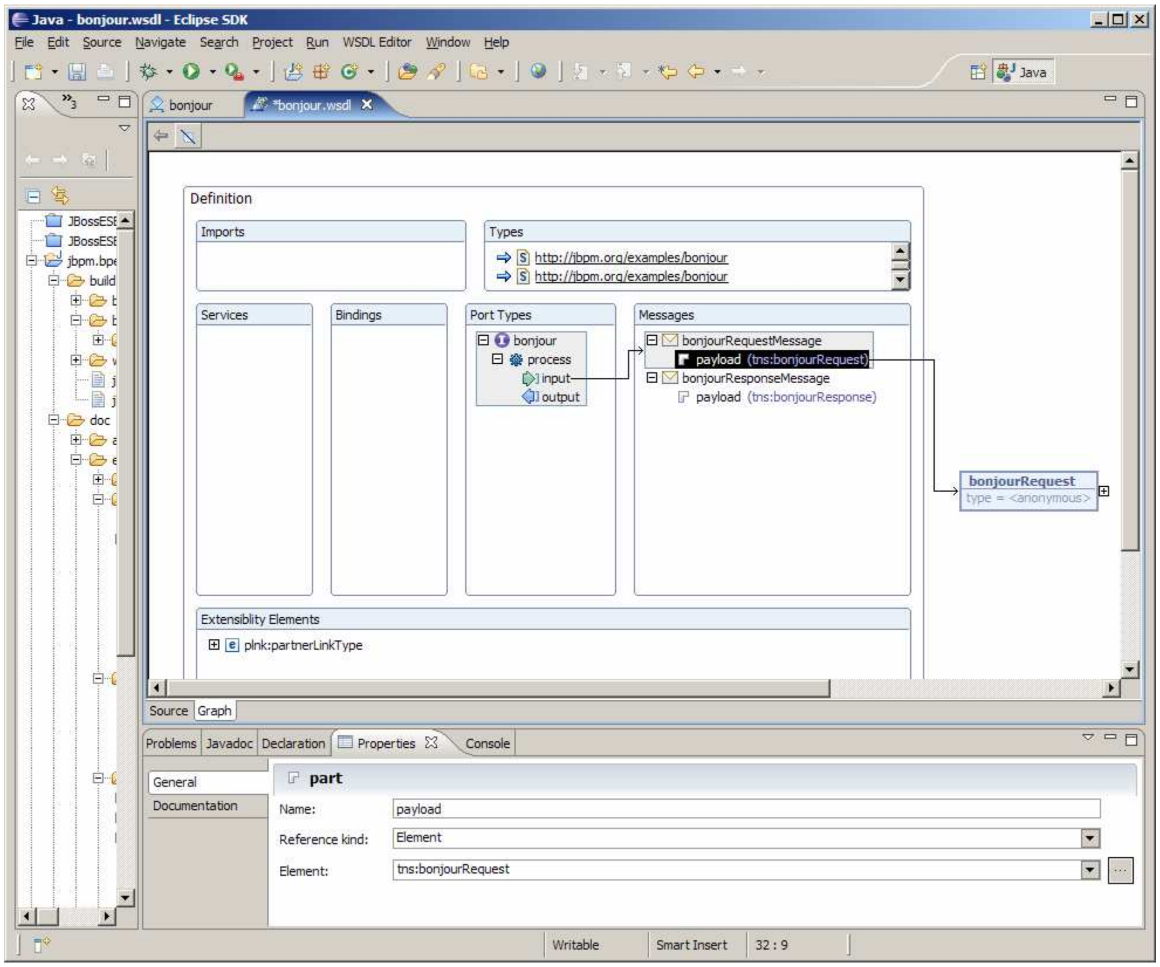
Namespace Declarations

Prefix	Namespace Name
tns	http://jbpm.org/examples/bonjour
plnk	http://schemas.xmlsoap.org/ws/2003/05/partner-link/
<no prefix>	http://schemas.xmlsoap.org/wsdl/
xsd	http://www.w3.org/2001/XMLSchema

Click OK again to return to the Visual WSDL Editor.

This series of steps now allows you to refer to standard XML Schema types. We specifically need "xsd:string" for this example.

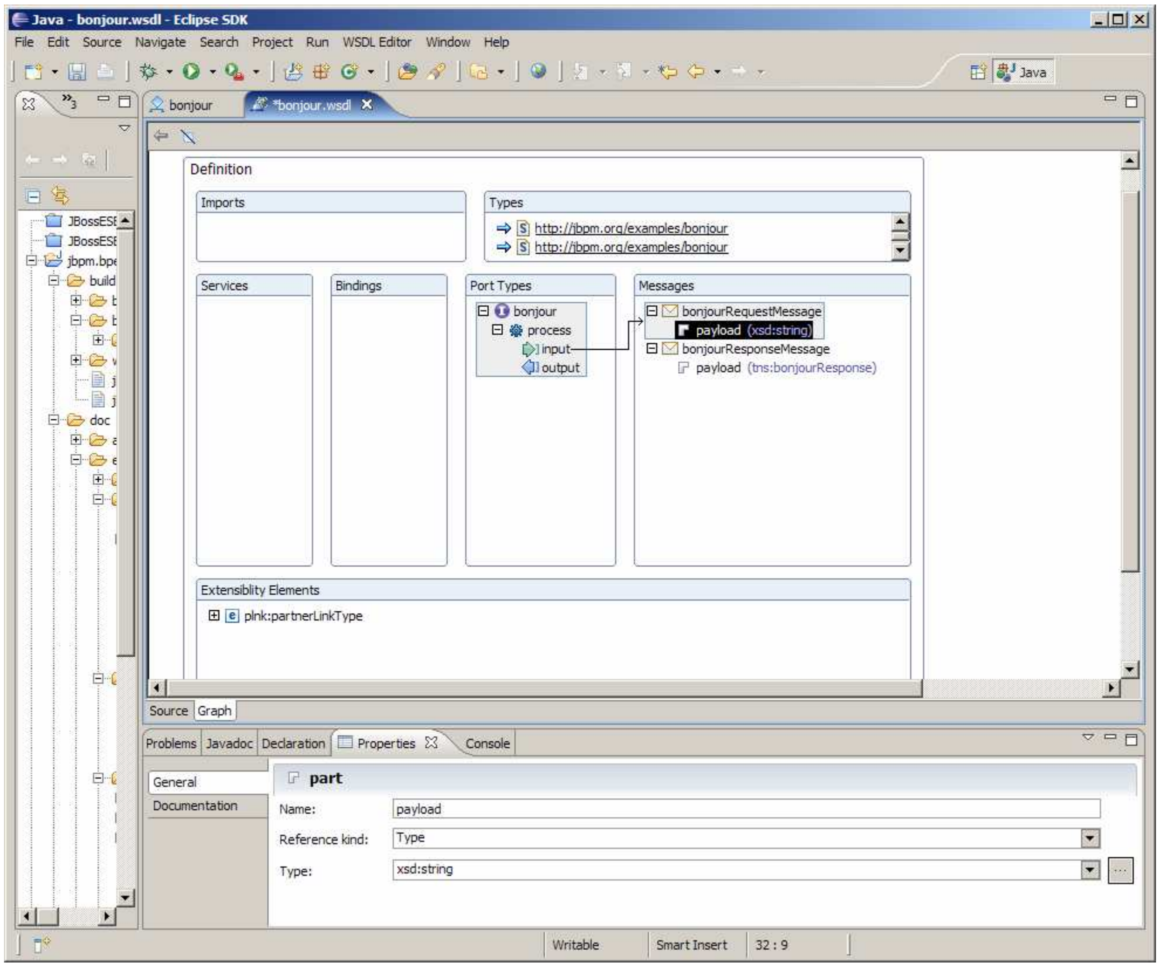
14. Open up and then highlight "payload (tns:bonjourRequest)" under the bonjourRequestMessage in the Messages section of the Visual WSDL Editor.


Java - bonjour.wsdl - Eclipse SDK
File Edit Source Navigate Search Project Run WSDL Editor Window Help
bonjour
*bonjour.wsdl
Definition
Imports
Types
http://jbpm.org/examples/bonjour
http://jbpm.org/examples/bonjour
Services
Bindings
Port Types
Messages
bonjour
process
input
output
bonjourRequestMessage
payload (tns:bonjourRequest)
bonjourResponseMessage
payload (tns:bonjourResponse)
bonjourRequest
type = <anonymous>
Extensiblity Elements
plnk:partnerLinkType
Source Graph
Problems Javadoc Declaration Properties Console
General
Documentation
part
Writable Smart Insert 32 : 9

Name: payload
Reference kind: Element
Element: tns:bonjourRequest

The Properties tab shows the mapping of input message payload part to "tns:bonjourRequest" Element.

15. Change the Reference kind to Type using the drop down list box. The Type will default to xsd:string.

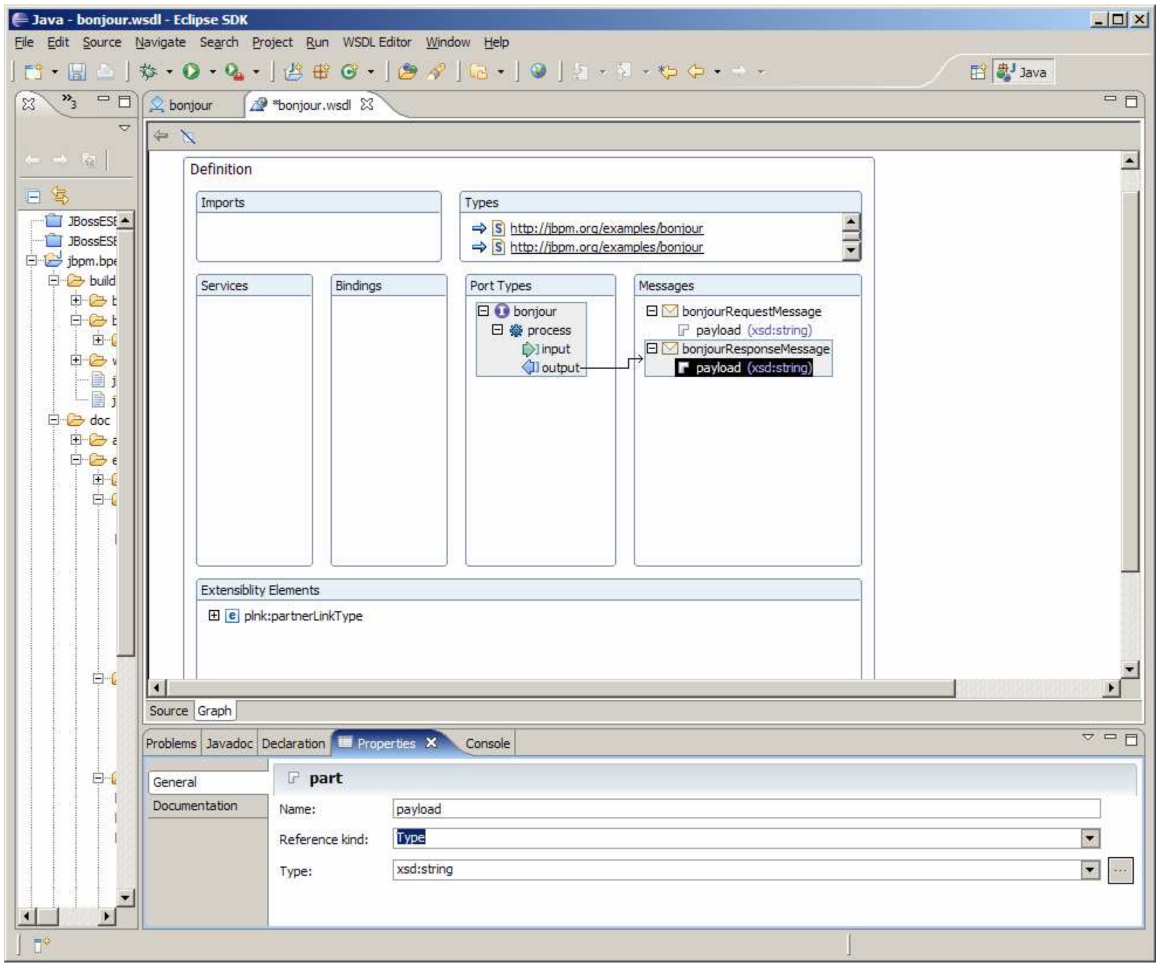

Java - bonjour.wsdl - Eclipse SDK
File Edit Source Navigate Search Project Run WSDL Editor Window Help
bonjour
*bonjour.wsdl
Definition
Imports
Types
http://jbpm.org/examples/bonjour
http://jbpm.org/examples/bonjour
Services
Bindings
Port Types
bonjour
process
input
output
Messages
bonjourRequestMessage
payload (xsd:string)
bonjourResponseMessage
payload (tns:bonjourResponse)
Extensiblity Elements
plnk:partnerLinkType
Source Graph
Problems Javadoc Declaration Properties Console
General
Documentation
part
Writable Smart Insert 32 : 9

Name: payload

Reference kind: Type

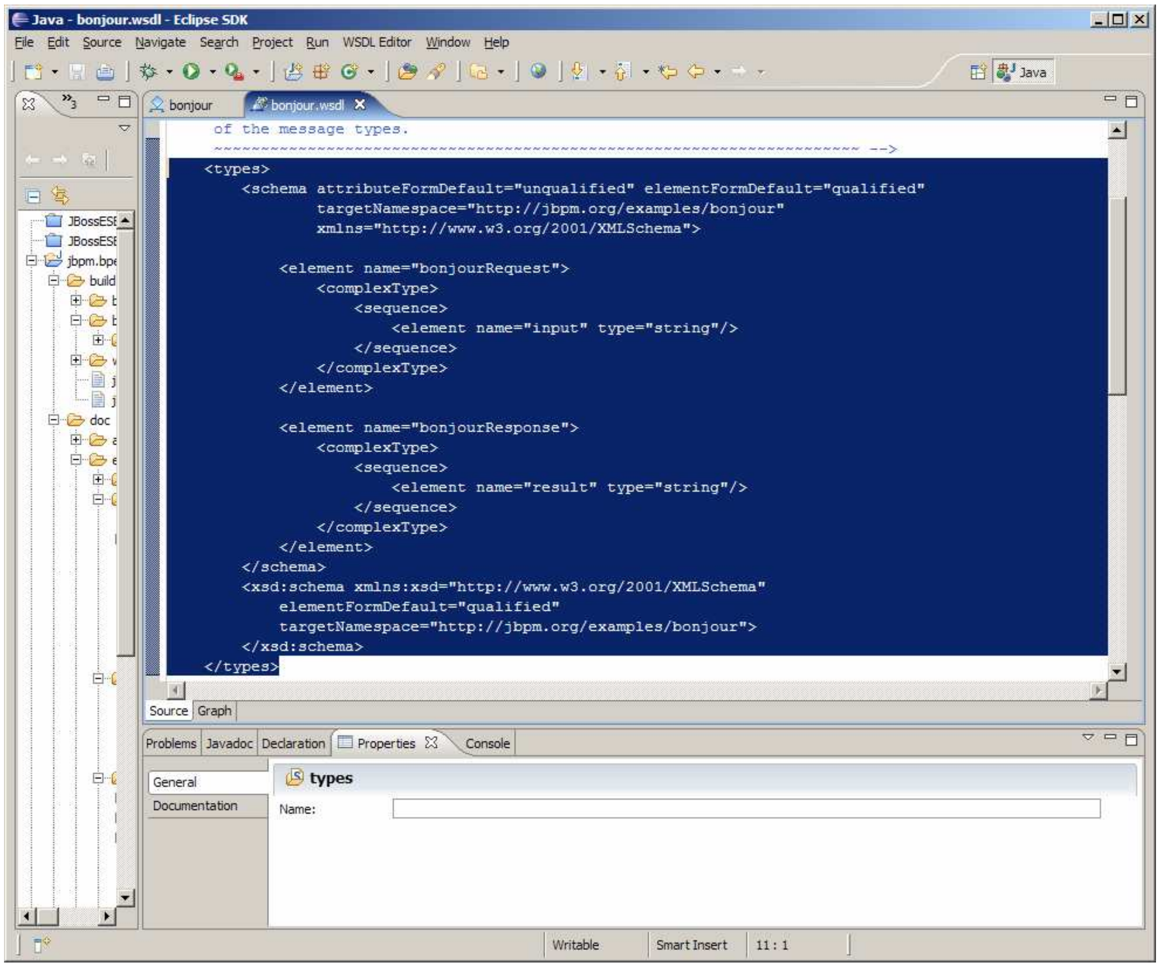
Type: xsd:string

16. Change the output message bonjourResponseMessage’s payload part to Type xsd:string.



Name: payload
Reference kind: Type
Type: xsd:string

17. Click on the Source tab for the WSDL editor. The <types> section is no longer necessary or valid so delete that section in the WSDL.


Java - bonjour.wsdl - Eclipse SDK
File Edit Source Navigate Search Project Run WSDL Editor Window Help
Java
bonjour
bonjour.wsdl
JBossESB
JBossESB
jbpm.bpe
build
doc
Source Graph
Problems Javadoc Declaration Properties Console
General
Documentation
types
Name:
Writable Smart Insert 11 : 1

```
    of the message types.
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
  <types>
    <schema attributeFormDefault="unqualified" elementFormDefault="qualified"
            targetNamespace="http://jbpm.org/examples/bonjour"
            xmlns="http://www.w3.org/2001/XMLSchema">

        <element name="bonjourRequest">
            <complexType>
                <sequence>
                    <element name="input" type="string"/>
                </sequence>
            </complexType>
        </element>

        <element name="bonjourResponse">
            <complexType>
                <sequence>
                    <element name="result" type="string"/>
                </sequence>
            </complexType>
        </element>
    </schema>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified"
        targetNamespace="http://jbpm.org/examples/bonjour">
    </xsd:schema>
  </types>
```

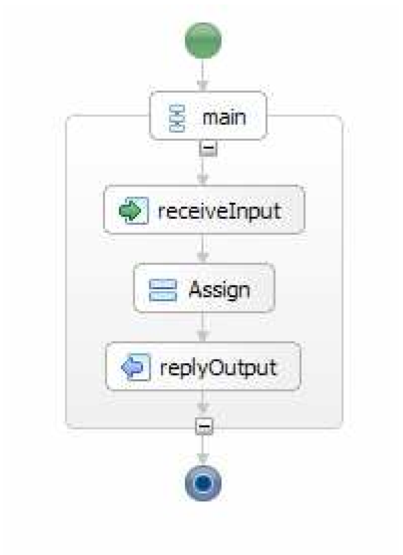
Delete the highlighted section in the screenshot above. Review the Message section which should look a lot like the following:

```
<message name="bonjourRequestMessage">
   <part name="payload" type="xsd:string"/>
</message>
<message name="bonjourResponseMessage">
   <part name="payload" type="xsd:string"/>
</message>
```

Save bonjour.wsdl

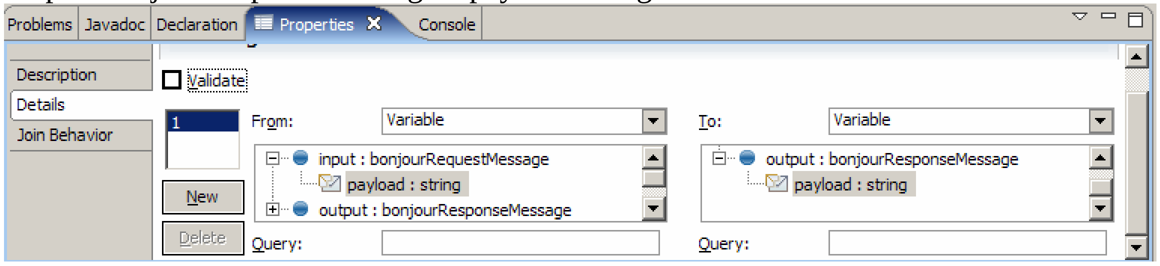
18. Open bonjour.bpel in a text editor. You can do this in Eclipse by right-clicking on bonjour.bpel and selecting Open With…Text Editor from the pop-up menu.
Add the following line between <bpws:process> and <bpws:partnerLinks>
<bpws:import importType="http://schemas.xmlsoap.org/wsdl/" location="bonjour.wsdl" namespace="http://jbpm.org/examples/bonjour"/>

18. Return to the bonjour.bpel visual editor and add an Assign Activity between receiveInput and replyOutput. You can simply click on Assign in the left-hand toolbar and then click in between the receiveInput and replyOutput activities.



19. Go to Properties for the Assign activity and select Details.
From: Variable
input: bonjourRequestMessage – payload:string

To: Variable
output: bonjourResponseMessage – payload:string



From: Variable
input : bonjourRequestMessage
payload : string
output : bonjourResponseMessage
Query:

To: Variable
output : bonjourResponseMessage
payload : string
Query:

Note: If the “payload:string” doesn’t show up make sure you have the following variable declarations in the BPEL source (use the Open With Text Editor trick).

```
<bpws:variable messageType="tns:bonjourRequestMessage" name="input"/>
<bpws:variable messageType="tns:bonjourResponseMessage" name="output"/>
```

20. Save bonjour.bpel
The Save action actually creates a bonjour.bpelex file which is needed to make the Designer function correctly. Absence of this may mean that the editor will not load the .bpel file correctly.

The default partnerLink is “client” and partnerLinkType is “tns:bonjour”

21. Copy the “bpel-definition.xml” file from the hello example to your bonjour/definition directory.

22. Open bonjour’s bpel-definition.xml and make the following changes:

```
<bpelDefinition location="bonjour.bpel"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jbpm.org/bpel
                    http://jbpm.org/bpel/bpel_definition_1_0.xsd"
  xmlns="http://jbpm.org/bpel" >

  <!-- makes WSDL interface elements available to the process -->
          <imports>
            <wsdl namespace="http://jbpm.org/examples/bonjour"
location="bonjour.wsdl"/>
  </imports>

</bpelDefinition>
```

23. Make a “classes” sub-folder under “web” then copy bpel-application.xml from “hello/web/classes” to “bonjour/web/classes”

24. Modify the bonjour bpel-application.xml and change “helloWorld” to “bonjour”. “bonjour” is the name of this particular process which you can double-check by looking at the <process name=”bonjour”…> of the bonjour.bpel file

The next section of this guide focuses on wrapping this BPEL process as a web service and modifying your build.xml to make it easier to change, build, deploy and test.

25. Copy the “web.xml” from the hello example to the “bonjour/web” folder. Make the following changes:

<servlet-class>org.jbpm.bpel.tutorial.**bonjour**.**Bonjour**_Impl</servlet-class>

```
<message-destination-ref>
   <!-- queue assigned to caller partner link -->
   <message-destination-ref-name>jms/client</message-destination-ref-name>
   <message-destination-type>javax.jms.Queue</message-destination-type>
   <message-destination-usage>ConsumesProduces</message-destination-usage>
</message-destination-ref>
```

26. Make a client folder below bonjour. This folder is targeted by the Ant build script that ships with the project. This guide doesn’t use the generated Java test client.

27. Copy the “application.xml” from “hello/application” into “bonjour/application” folder and make the following changes:

```
  <description>Bonjour Business Process</description>
  <display-name>BonjourProcess</display-name>
  <!-- bonjour service -->
  <module>
    <web>
```

```
      <web-uri>bonjour.war</web-uri>
      <context-root>/bonjour</context-root>
    </web>
  </module>
  <!-- bonjour client -->
  <module>
    <java>bonjour.jar</java>
  </module>
```

28. Copy “hello/web/jboss-web.xml” from the hello example to the bonjour web folder and make the following changes:

```
<jboss-web>

  <resource-ref>
    <!-- JMS connection factory reference (in web.xml) -->
    <res-ref-name>jms/ConnectionFactory</res-ref-name>
    <!-- actual resource in java JNDI context -->
    <jndi-name>java:ConnectionFactory</jndi-name>
  </resource-ref>

  <message-destination-ref>
    <!-- caller queue reference (in web.xml) -->
    <message-destination-ref-name>jms/client</message-destination-ref-name>
    <!-- actual resource in global JNDI context -->
    <jndi-name>queue/testQueue</jndi-name>
  </message-destination-ref>

</jboss-web>
```

29. Copy “hello/web/webservices.xml” from the hello example to the bonjour web folder and make the following changes:

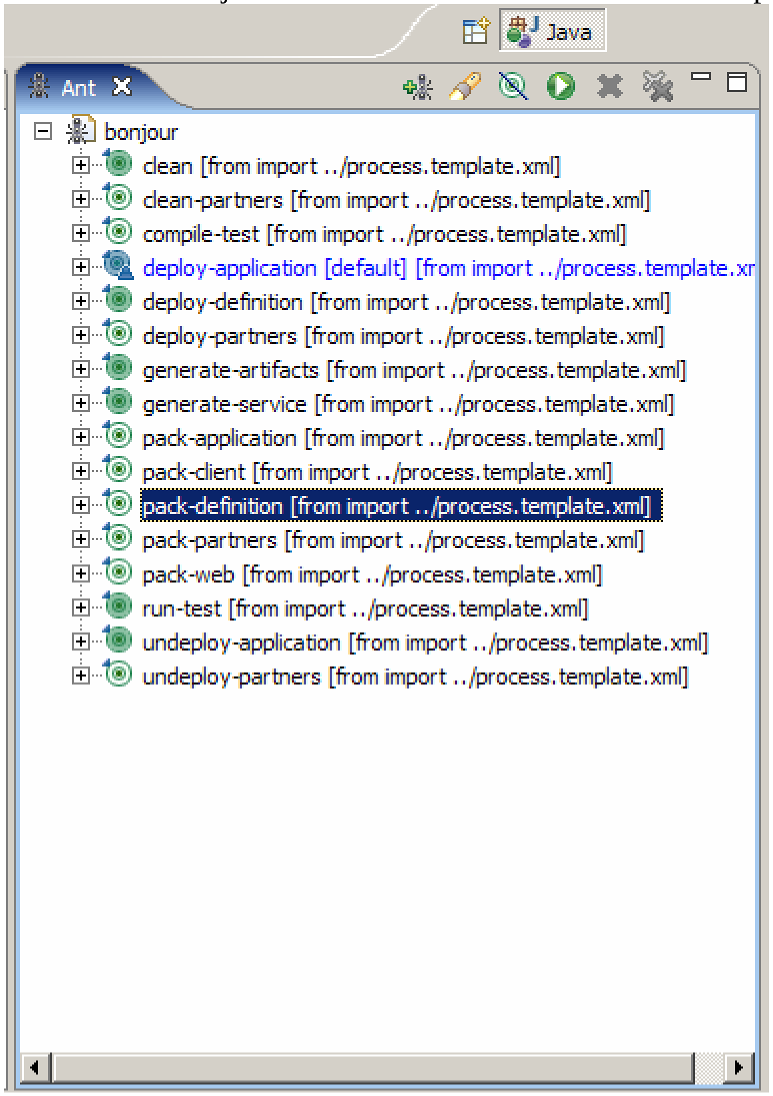
```
<webservice-description-name>Bonjour</webservice-description-name>

<port-component-name>clientPort</port-component-name>
<!-- WSDL port element (in WSDL implementation file) -->
<wsdl-port xmlns:portNS="http://jbpm.org/examples/bonjour"> portNS:clientPort
</wsdl-port>
<!-- service endpoint interface class -->
<service-endpoint-interface>
  org.jbpm.bpel.tutorial.bonjour.Bonjour
</service-endpoint-interface>

<init-param>
    <description>name of the associated partner link</description>
    <param-name>portName</param-name>
    <param-value>client</param-value>
```

</init-param>

Note : service.wsdl and jaxrpc-mapping.xml will be generated by the Ant script

30. Copy “hello/web/wscompile.xml” from the hello example to the bonjour web folder and make the following changes:
<wsdl location="wsdl/service.wsdl" packageName="org.jbpm.bpel.tutorial.**bonjour**">

31. Load the bonjour’s build.xml into the Ant view of Eclipse.

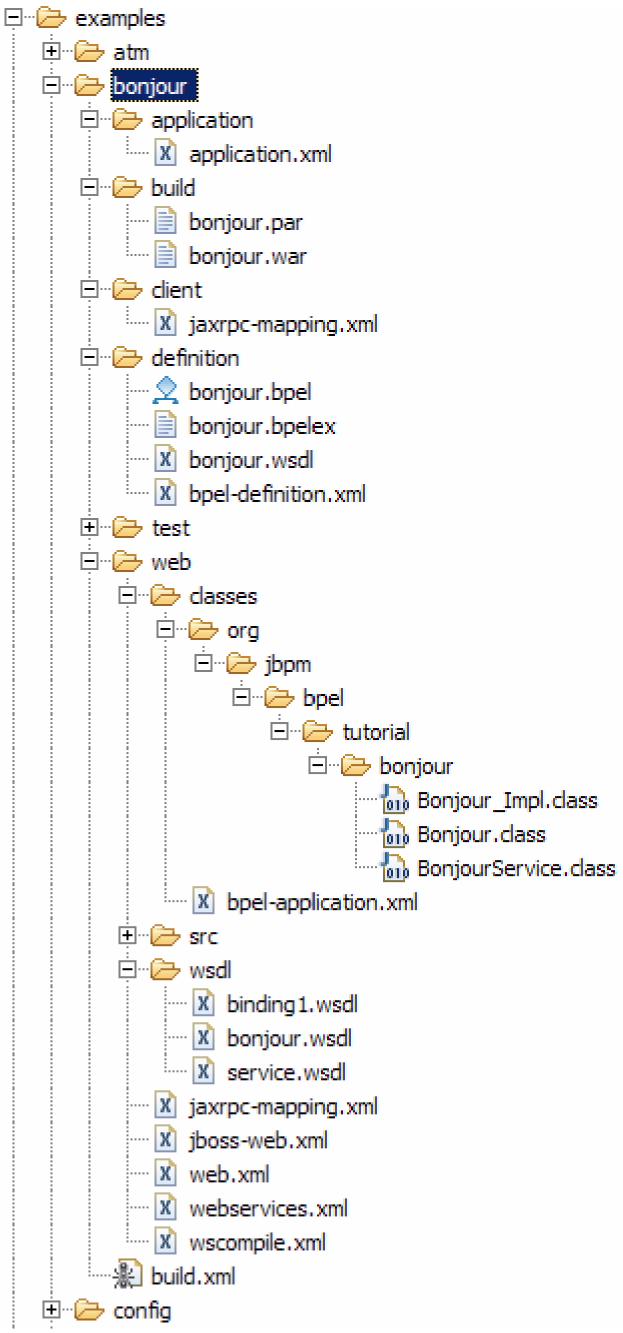


```
bonjour
  clean [from import ../process.template.xml]
  clean-partners [from import ../process.template.xml]
  compile-test [from import ../process.template.xml]
  deploy-application [default] [from import ../process.template.xr
  deploy-definition [from import ../process.template.xml]
  deploy-partners [from import ../process.template.xml]
  generate-artifacts [from import ../process.template.xml]
  generate-service [from import ../process.template.xml]
  pack-application [from import ../process.template.xml]
  pack-client [from import ../process.template.xml]
  pack-definition [from import ../process.template.xml]
  pack-partners [from import ../process.template.xml]
  pack-web [from import ../process.template.xml]
  run-test [from import ../process.template.xml]
  undeploy-application [from import ../process.template.xml]
  undeploy-partners [from import ../process.template.xml]
```

Run the following Ant Tasks:
1. pack-definition - generates the .par file into the build directory
2. deploy-definition - sends it to the properly configured and running 4.0.4.GA server
On your first deployment of the BPEL .par you may see a lot of activity on your server console as jBPM builds its underlying data structures.
3. generate-service - generates three wsdl files into a wsdl folder below web.

4. generate-artifacts - generates .java and .class files for the Java components mentioned in the config files above, these are basically the files needed to implement a Web Service endpoint in the JSR 109 fashion. This task also creates the jaxrpc-mapping.xml file
5. pack-web – generates the .war file into the build directory

32. Right click on the examples folder in the Navigator or Package Explorer view and select Refresh from the context-menu to see all of these new files in your project.



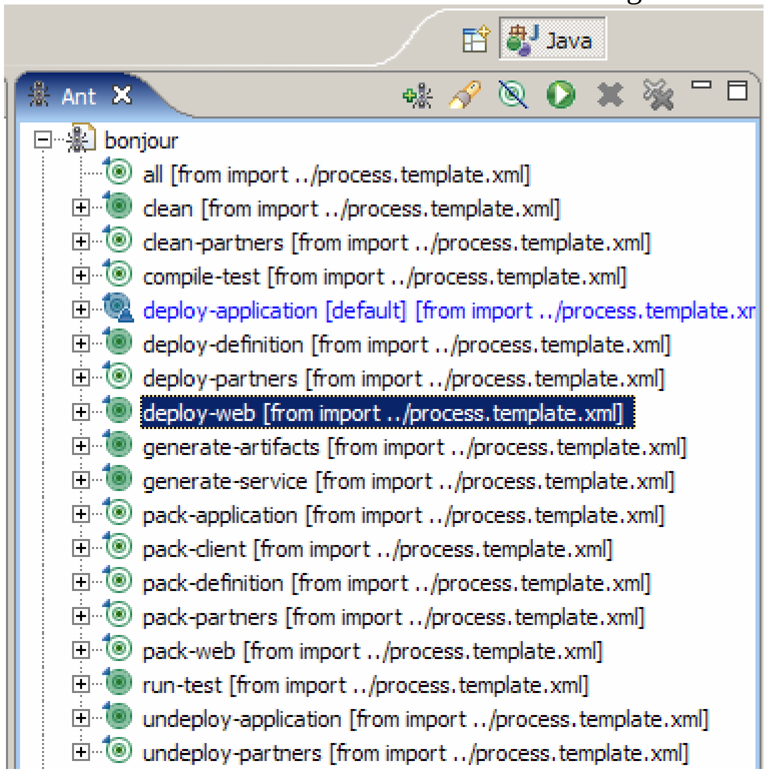
```
examples
  atm
  bonjour
    application
      application.xml
    build
      bonjour.par
      bonjour.war
    client
      jaxrpc-mapping.xml
    definition
      bonjour.bpel
      bonjour.bpelex
      bonjour.wsdl
      bpel-definition.xml
    test
    web
      classes
        org
          jbpm
            bpel
              tutorial
                bonjour
                  Bonjour_Impl.class
                  Bonjour.class
                  BonjourService.class
        bpel-application.xml
      src
      wsdl
        binding1.wsdl
        bonjour.wsdl
        service.wsdl
      jaxrpc-mapping.xml
      jboss-web.xml
      web.xml
      webservices.xml
      wscompile.xml
    build.xml
  config
```

The result of running the Ant tasks listed above.

33. Modify the process.template.xml file to include the additional tasks:

```
<!-- My Deploy Web -->
<target name="deploy-web" description="deploy the bpel web application">
<copy file="${build.dir}/${app.name}.war" todir="${jboss.server.dir}/deploy" />
</target>

<!--  My All -->
<target name="all" depends="pack-definition,deploy-definition,generate-
service,generate-artifacts,pack-web,deploy-web" />
```

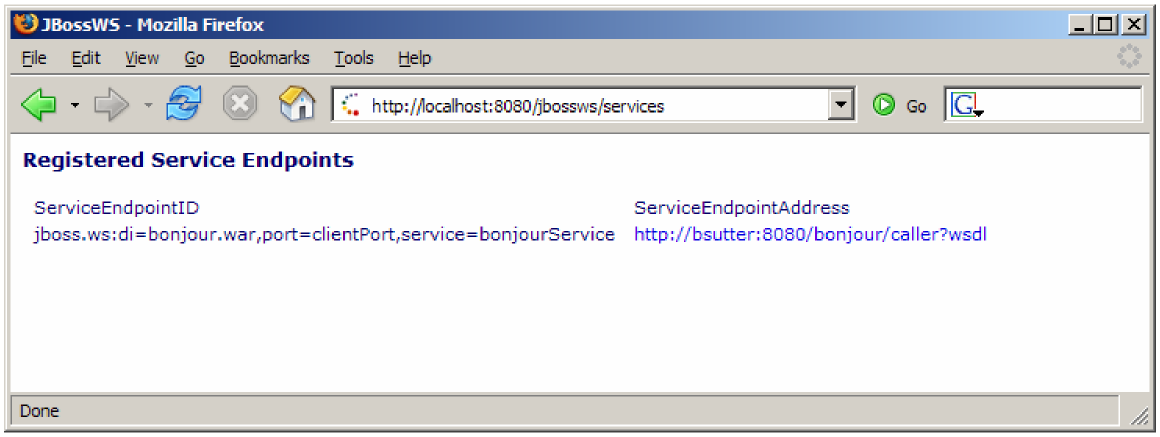
Save and Refresh the Ant view to see these changes.



```
bonjour
  all [from import ../process.template.xml]
  clean [from import ../process.template.xml]
  clean-partners [from import ../process.template.xml]
  compile-test [from import ../process.template.xml]
  deploy-application [default] [from import ../process.template.xr
  deploy-definition [from import ../process.template.xml]
  deploy-partners [from import ../process.template.xml]
  deploy-web [from import ../process.template.xml]
  generate-artifacts [from import ../process.template.xml]
  generate-service [from import ../process.template.xml]
  pack-application [from import ../process.template.xml]
  pack-client [from import ../process.template.xml]
  pack-definition [from import ../process.template.xml]
  pack-partners [from import ../process.template.xml]
  pack-web [from import ../process.template.xml]
  run-test [from import ../process.template.xml]
  undeploy-application [from import ../process.template.xml]
  undeploy-partners [from import ../process.template.xml]
```

34. Execute the “deploy-web” Ant task to copy the bonjour.war file to JBoss’ hot deploy directory. Take a quick look at the App Server console (typically the command prompt window in which you started the App Server) to see if there is an exception stack trace. Errors associated with incorrect BPEL syntax or mapping issues show up in that stack trace. Since we are not using the J2EE application client code we are simply going to deploy a .war file.

35. Use your browser to hit the following URL:
http://localhost:8080/jbossws/services


JBossWS - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080/jbossws/services
Go
Done

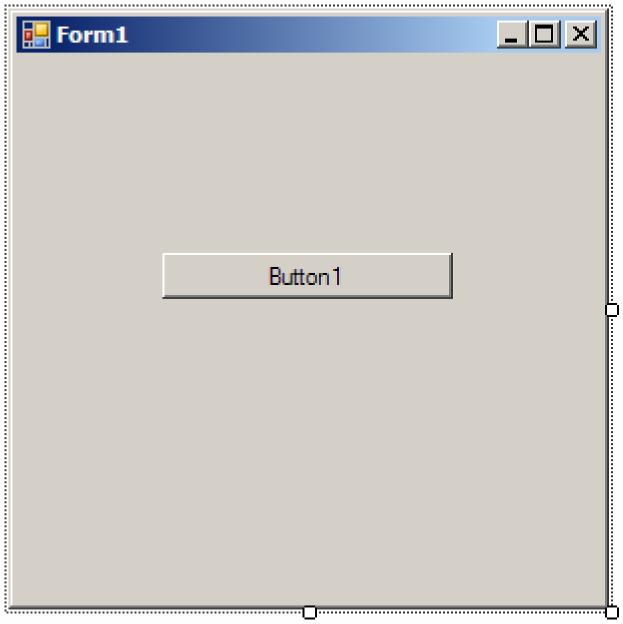
Registered Service Endpoints

ServiceEndpointID	ServiceEndpointAddress
jboss.ws:di=bonjour.war,port=clientPort,service=bonjourService	http://bsutter:8080/bonjour/caller?wsdl

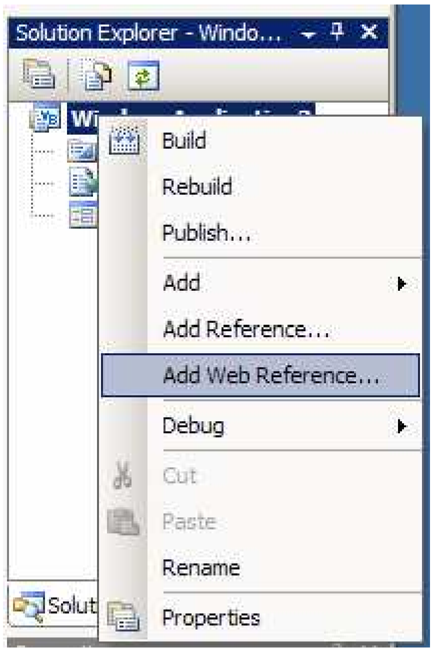
This will show you that the bonjour BPEL process was deployed as a web service on the JBoss Application Server. Drill down in to the WSDL link (http://yourhost:8080/caller?wsdl) for more details.

You are now ready to build clients/consumers that can interact with this new web service. For this tutorial, we are using Visual Basic.NET and the freely available Visual Basic Express Edition downloadable from Microsoft.com.

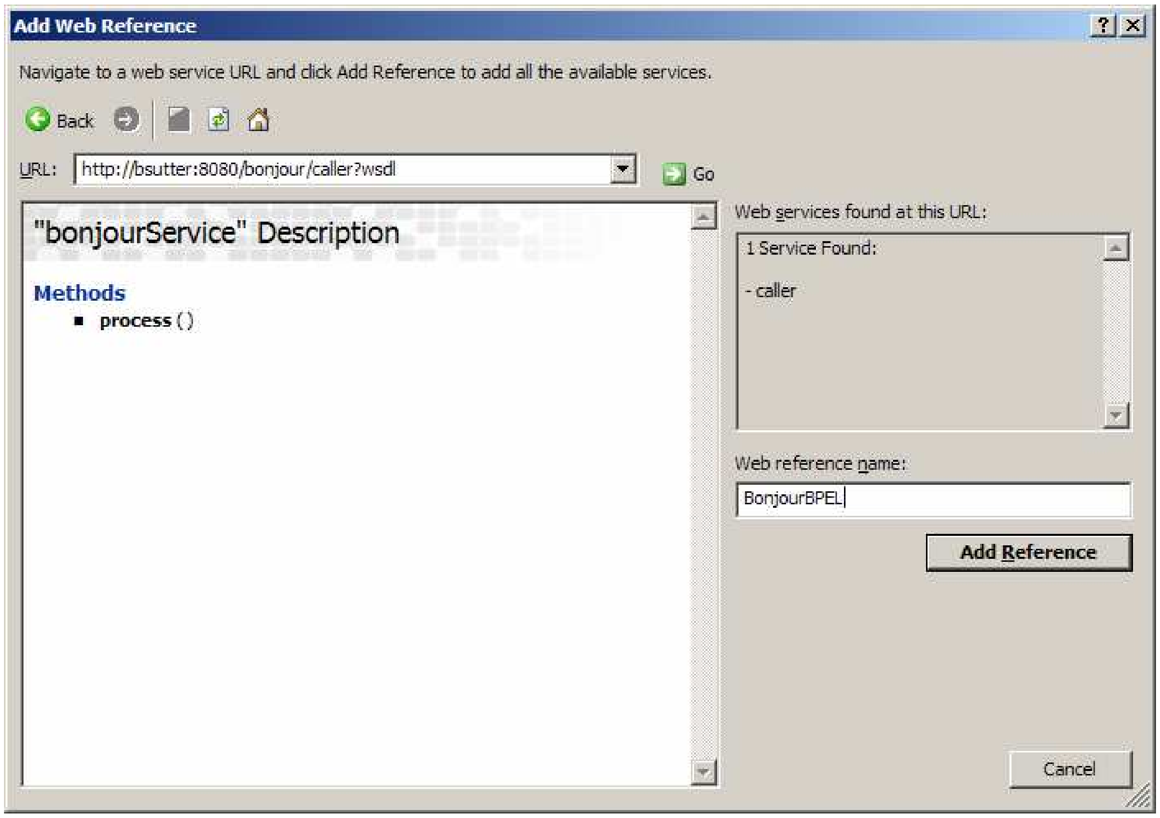
36. Start with a new Windows Application Project
Add a Button from the Toolbox to Form1



37. Right click in the Solution Explorer (positioned on the right-side of the screen) and select Add Web Reference from the context menu.



38. Type (or copy/paste) the URL to the bonjour WSDL into the URL field and select the Go button.
Type “BonjourBPEL” into the Web reference name textbox.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://bsutter:8080/bonjour/caller?wsdl

"bonjourService" Description

Methods
- process ()

Web services found at this URL:
1 Service Found:
- caller

Web reference name: BonjourBPEL

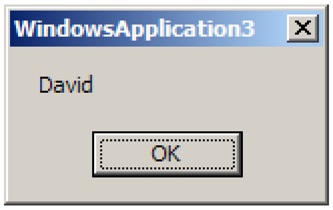
Select the Add Reference button. This action will generate a client-side proxy for the server-side BPEL process hosted on the JBoss Application Server.

39. Double click on Button1 and add the following code for its click event:

```
Dim proxy As New BonjourBPEL.bonjourService
Dim myString As String
myString = "David"
proxy.process(myString)
MsgBox(myString)
```

Note: the proxy.process() method is declared as a “sub” in VB.NET which allows it to accept a pass-by-reference argument “myString” containing the value of “David”.

40. Run the WinForm and click on Button1

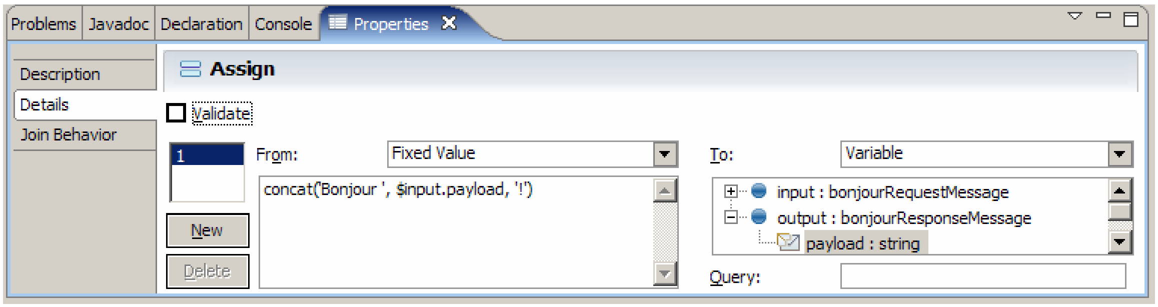


```
David
```

If there are no errors then you can assume that this properly executed the BPEL process on the server-side, however, let’s tweak the BPEL process to modify the inbound string and return a different result.

41. Return to the Eclipse BPEL Designer and load bonjour.bpel via the Business Process Editor.   Select the Assign activity’s Properties-Details. Change the From drop-down list to “Fixed Value” and add the following expression:

    concat('Bonjour ', $input.payload, '!')



From: Fixed Value

```
concat('Bonjour ', $input.payload, '!')
```

To: Variable

input : bonjourRequestMessage
output : bonjourResponseMessage
payload : string

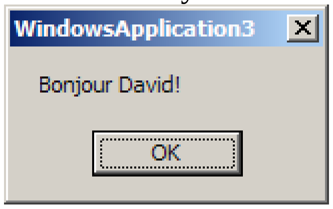
Query:

Save bonjour.bpel

42. Execute the pack-definition and deploy-definition Ant tasks.
Note: In the current beta1.1 version of JBoss BPEL you might need to restart the server to deploy a new/changed .par (Process Archive) file.

You might also make changes that require you to run the Ant all task that will execute all of the steps necessary to rebuild the .par and .war files and deploy them to the local application server.

43. Return to your Visual Basic.NET WinForm and Run



Bonjour David!