EJERCICIOS DE ORQUESTACIÓN DE SERVICIOS BPEL

Fecha límite: Jueves 16 de Julio 2015 23:59

PARTE 1

Vamos a crear desarrollar, desplegar y probar un sencillo servicio BPEL utilizando Netbeans, basándonos en los pasos que acabamos de ver. Empezaremos con un proceso BPEL síncrono que recibe un mensaje de entrada y en función de su contenido devuelve una respuesta al cliente. Este mensaje de entrada consiste en una cadena de caracteres.

1. Pasos previos con Open-ESB (Netbeans)

OpenESB es un IDE basado en Netbeans que incluye funcionalidades para trabajar con proyectos SOA. Para trabajar con este IDE seguimos los siguientes pasos:

a) Descargar OpenESB v2.3.1 desde http://open-esb.net/index.php?option=com_dtracker&filename=v231/openesb-v231-installer-windows.exe&filetitle=OpenESB%20V2.3.1%20for%20Windows
b) Instalar OpenESB
c) verificar si tenemos instalado el plugin JUnit para poder ejecutar los casos de prueba. Si no está instalado ir a Tools->Plugins->Available Pluguins.
d) En la pestaña Services-> Servers vemos que tenemos instalado el servidor Glassfish 2.x, está versión nos permite trabajar con BPEL. Este sevidor tiene instalado el **runtime JBI** (service engines, binding components, service assemblies).
e) Podemos acceder a la consola de administración:

```
http://localhost:4848
Usuario: admin
Contraseña: adminadmin
```

2. Crear proyecto BPEL

Vamos a utilizar un proyecto ejemplo que viene con OpenESB:

a) Vamos a New Project -> Samples -> SOA -> Synchronous BPEL process. Nombre: *EjemploBpelSincrono*.
b) Obtenemos un proyecto con los wsdl y xsd que podemos modificar si lo deseamos.
c) Deben notar que automáticamente se generó un proyecto Composite Application y que el módulo BPEL se ha añadido automáticamente como un JBI module.
d) Podemos ver las propiedades del proyecto BPEL haciendo clic derecho en el proyecto -> propiedades:
    a. **General**: muestra la ruta de directorios de los ficheros del proyecto, entre otros elementos.
    b. **Referencias del proyecto**: muestra otros proyectos BPEL referenciados por nuestro proyecto BPEL
    c. **Catálogo XML**: muestra las entradas del catálogo XML usado en el proyecto BPEL. Los catálogos XML proporcionan información de mapeo entre una entidad externa en un documento XML y la localización real del documento que está siendo referenciado.

3. WSDL y esquema de nombres del proyecto BPEL

Se han creado plantillas para el WSDL y el XSD. Los WSDL podemos visualizarlos como: código fuente, como árbol o gráficamente.

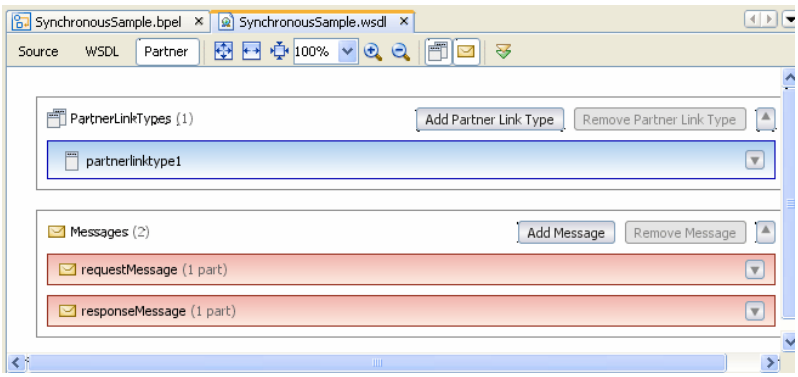

Fig. 1 Vista gráfica del WSDL

Este WSDL contiene la definición *partnerLinkType* que define el rol *partnerlinktyperole1*, asociado al *portType portType1*. También tenemos las definiciones de dos mensajes: uno de entrada (*requestMessage*), y otro de salida (*responseMessage*), que contienen una cadena de caracteres.

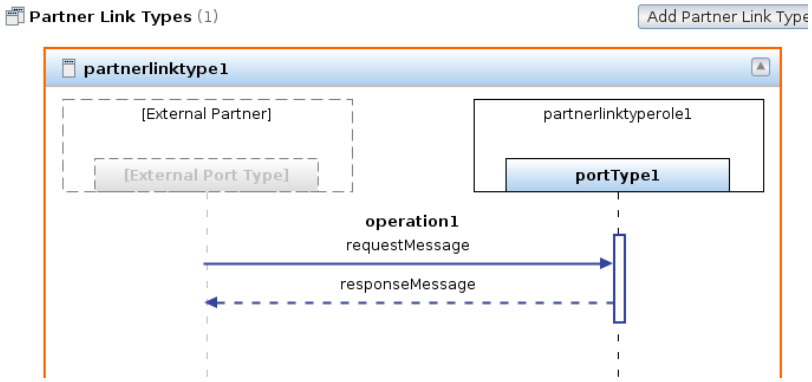

Fig. 2 Definición de partnerLinktype en WSDL

La definicón del *xsd* es opcional en en proyecto BPEL. En este caso ha sido generado automáticamente y podemos verlo de manera similar que el WSDL (fuente, árbol, gráfico).

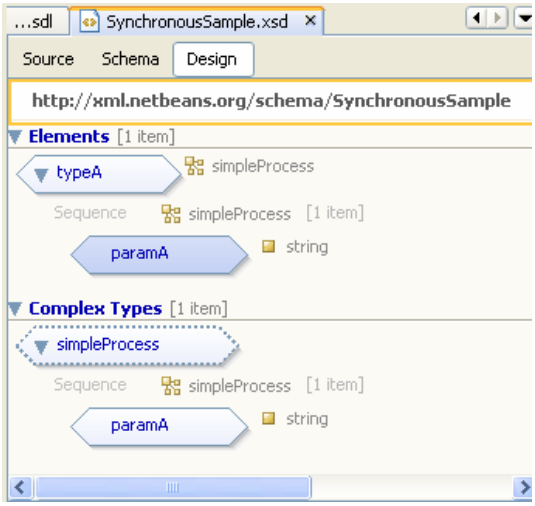

Fig. 3 Vista gráfica XSD

Podemos ver que se ha definido el tipo *typeA*, que es un elemento del tipo complejo *simpleProcess*, que a su vez es una cadena de caracteres.

En caso de que tuviésemos que crear nosotros los ficheros WSDL y de esquema, el proceso que se sigue es similar en ambos casos, tal y como ya hemos visto en sesiones anteriores:

a) Crear un nuevo documento: Clic con el botón derecho del ratón sobre el nodo Process Files y elegimos New->WSDL Document, o New->Other/XML/XML Schema.

b) Desde la vista gráfica, añadimos componentes arrastrándolos desde la ventana Palette a la vista gráfica.

c) Desde la vista de árbol, añadimos componentes y clic con el botón derecho sobre nodo del tipo de componente que queremos añadir, y elegimos el opción Add ... correspondiente del menú emergente

d) Desde la vista de fuente, escribimos el código en el editor de texto

4. Lógica del proceso BPEL

Para crear un proceso BPEL desde cero tendríamos que:

a) New Project -> SOA -> BPEL module

O

b) Clic derecho sobre el nodo Process Files -> New-> BPEL Process en el menú emergente.

El proceso BPEL creado en el ejemplo es muy simple y tenemos tres vistas: Source, Design y Mapper. Si vamos a la vista gráfica vemos:

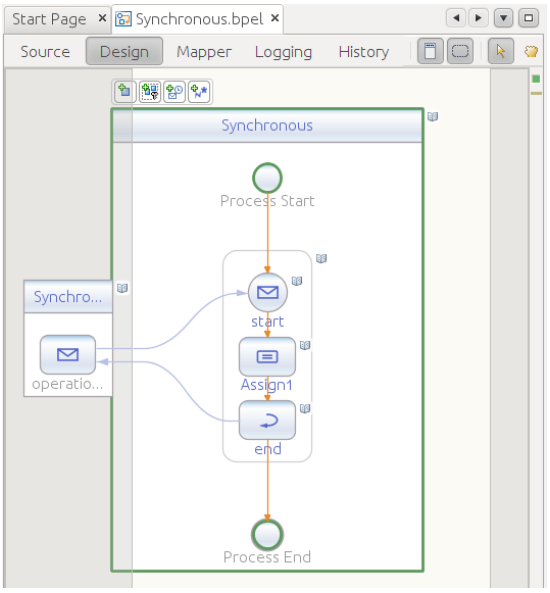

Fig. 4 Vista de diseño del proceso BPEL síncrono

Para añadir elementos a nuestro proceso BPEL los arrastramos desde la ventana Palette hasta la vista de diseño del proceso BPEL.

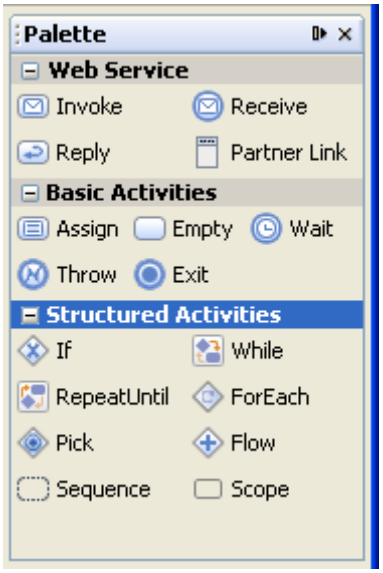

Fig. 5 Paleta de componentes BPEL

En la paleta de componentes tenemos los elementos que podemos insertar en el proceso BPEL, clasificados en tres categorías:

a) **Servicios Web**: en donde encontramos las actividades Invoque, Receive, Reply y el componente PartnerLink.
b) **Actividades básicas**: Assign, Empty, Wait, Throw y Exit
c) **Actividades estructuradas**: If, While, Sequence,...

Si vamos al código fuente del proceso BPEL podemos ver que la lógica del proceso de negocio consiste en esperar a recibir una llamada del *partnerLink* cliente sobre la operación *operation1*, con el mensaje de entrada *inputVar*. Una vez que el proceso BPEL ha sido invocado por un cliente, asignamos el valor del mensaje de entrada en el mensaje de salida, denominado *outputVar* y se lo enviamos al cliente utilizando la actividad *reply*.

Añadir una actividad *if*

Vamos a modificar este proceso inicial añadiendo una actividad *if* en la vista de diseño.

Para ello:

a) Elegimos la actividad estructurada *if* de la paleta de componentes y la arrastramos a la vista de diseño del proceso BPEL, situándolo entre las actividades Start y Assign. Esto nos creará una actividad *if* denominada *if1* en la vista de diseño. (El IDE nos indica dónde podemos situar los elementos que arrastramos mediante dos círculos concéntricos, coloreando el más pequeño).
b) Damos clic dos veces sobre la actividad *if1* y aparecerá la ventana de mapeado (BPEL Mapper). Otra forma de acceder a la ventana de mapeado es, teniendo el componente *if1* seleccionado (podemos ver que un componente está seleccionado porque aparece enmarcado en un rectángulo verde con bordes redondeados), simplemente cambiamos a la vista Mapper. Usaremos la ventana de mapeado para definir una condición booleana.
c) En el panel de la izquierda vemos las variables de nuestro proceso BPEL. En el panel de la derecha nos aparece una condición booleana asociada a la actividad *if*. Seleccionamos la condición booleana con el botón izquierdo del ratón, así como el elemento *paramA* de la variable *inputVar.*

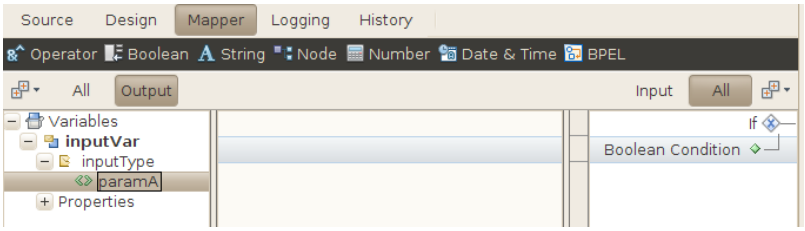

Fig. 6 Ventana de mapeado de BPEL

d) En la barra de menús, damos clic en *Operator* y seleccionamos la función *EQUAL* en el menú emergente. La función *equal* aparecerá en la parte central de la ventana de mapeado. Dicha función tiene asociados dos conectores de entrada y uno de salida (representados con forma de puntas de flecha).

e) En la barra de menús, damos clic sobre la función *String* y seleccionamos *String Literal* en la lista desplegable. Aparecerá una caja con la etiqueta *String literal* en el panel central.

f) Tecleamos `'Hola mundo'` en la función *string literal* y pulsamos enter.

g) Clic sobre *paramA* y lo arrastramos hasta el segundo conector de entrada de la función *equal*.

h) Clic sobre el conector de salida de *String Literal* y lo arrastramos hasta el otro conector de entrada de *Equal*.

i) Desde la función *equal*, arrastramos el conector de salida hasta *Boolean Condition* en el panel de la derecha. Si no se muestra el nodo *Boolean Condition*, simplemente haremos doble clic sobre la condición *if1*.

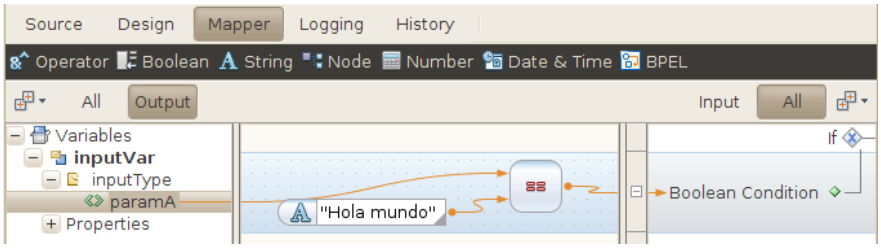

Fig. 7 Ventana de mapeado de BPEL a la que hemos añadido una condición

Añadimos una actividad Assign

En la vista de diseño, arrastramos la actividad *Assign1* existente hasta la actividad *if*. Colocamos esta actividad entre rombos con el icono X en el área de la actividad *if1.* A continuación, elegimos la actividad *Assign* de la sección de actividades básicas de la paleta, y la arrastramos a la derecha de la actividad *Assign1* existente.

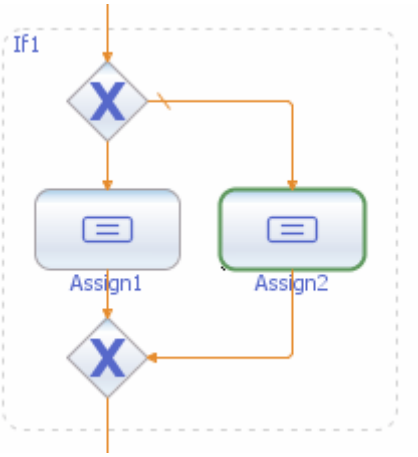

Fig. 8 Resultado de añadir una segunda asignación

Seleccionamos la nueva actividad *Assign2* y volvemos a utilizar la ventana de mapeado de la siguiente forma:

a) Expandimos los nodos *Variables -> inputVar -> inputType* y *Variables -> outputVar -> resultType* (en el panel de la izquierda). Para poder ver los sub-nodos hacemos doble clic sobre el nodo *variables*, y nos aparecerá el sub-nodo *outputVar*, si de nuevo hacemos doble clic sobre *outputVar*, nos aparecerá el sub-nodo *resultType*, y así sucesivamente hasta que aparezca el nodo *paramA.*

b) Seleccionamos las variables *paramA* tanto del panel de la derecha como el de la izquierda. Seguidamente seleccionamos el grupo de funciones de *String* y elegimos sobre la función *concat*. Aparecerá la función *Concat* en el panel central.

c) Hacemos doble clic sobre el primer campo de la función *Concat*, tecleamos `'Hola '` y pulsamos enter.

d) Arrastramos *paramA* del panel de la izquierda hasta el segundo campo de la función *concat.*

e) Arrastramos el ratón desde el conector de salida de la función *concat* hasta *paramA* (en el panel de la derecha). Esto concatena la cadena `'Hola '` con la entrada y copia el resultado en la salida.

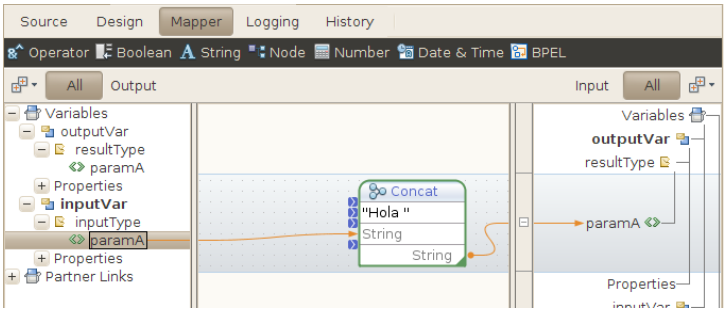

Fig. 9 Concatenamos el resultado

5. Desplegamos el proyecto en el servidor de aplicaciones

Como se mencionó anteriormente, nuestro proyecto BPEL no es directamente desplegable en el servidor de aplicaciones. Para poder hacer el despliegue tenemos que incluir el "jar" de nuestro proyecto como un módulo JBI en una Composite Application (en nuestro caso será el proyecto *EjemploBpelSincrono*Application) la composite application contendrá un ensamblado de servicios que sí podremos desplegar en el servidor de aplicaciones.

Para desplegar el proyecto realizaremos los siguientes pasos:

a) Compilamos nuestro proyecto Bpel (*EjemploBpelSincrono*) haciendo clic con botón el derecho y seleccionando *Build.*

b) Clic con el botón derecho sobre el proyecto *EjemploBpelSincronoApplication* y seleccionamos *Add JBI module....* Elegimos el proyecto *EjemploBpelSincrono* y clic sobre *"Add Project JAR files"*

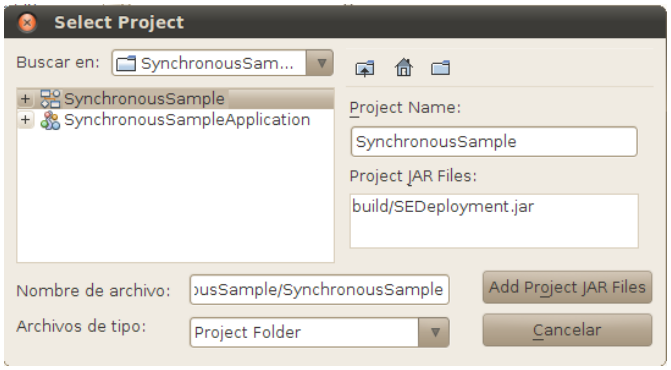

Fig. 10 Añadimos el módulo JBI a la Composite Application

c) Clic con el botón derecho sobre el proyecto *EjemploBpelSincronoApplication* y elegimos *Deploy*, seleccionando *Glassfish v2.x*. Aparecerá el mensaje *BUILD SUCCESSFUL* en la ventana *Output*.

d) Podemos ver el resultado del despliegue en la pestaña *Services*, en el servidor de aplicaciones *Glassfish v2.x*. El componente *EjemploBpelSincronoApplication* se ha desplegado como un ensamblado de servicios (Service Assembly) dentro del meta-contenedor JBI.

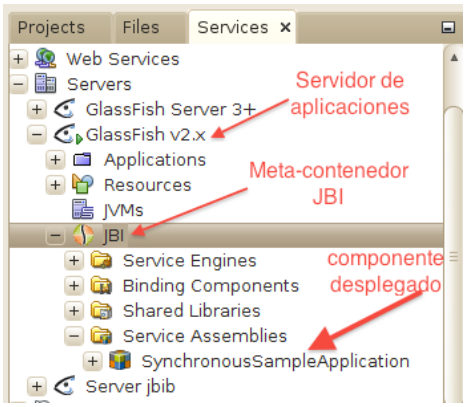

Fig. 11 Componente desplegado como un módulo JBI

6. Creamos un conductor de pruebas

Vamos a añadir casos de prueba para comprobar el funcionamiento del proceso BPEL que hemos creado. Para ello seguiremos los siguientes pasos:

a) En la ventana *Projects*, expandimos el nodo del proyecto *EjemploBpelSincronoApplication*, clic con el botón derecho sobre el nodo *Test*, y elegimos *New Test Case* del menú emergente.

b) Vamos a darle el nombre CasoPrueba1 y clic en *Next*.

c) Expandimos *EjemploBpelSincrono -> Source Packages* y seleccionamos *EjemploBpelSincrono*.wsdl. Clic en Next.

d) Seleccionamos *operation1* y clic en *Finish*. Ahora, en el árbol del proyecto, bajo el nodo *Test* se ha creado la carpeta *CasoPrueba1*, que contiene dos ficheros: *Input* y *Output.*

e) Clic dos veces sobre *Input* y modificamos su contenido de la siguiente forma:
   a. En la siguiente línea en los contenidos del cuerpo:
      `<syn:paramA>?string?<syn:paramA>`
   b. Reemplaza `?string?` con `Amigos`
   c. Grabamos los cambios desde *File->Save*

f) Clic dos veces sobre *Output.xml* para examinar los contenidos. Antes de ejecutar las pruebas, este fichero está vacío. Cada vez que ejecutemos las pruebas, la salida actual se compara con los contenidos de *Output*. La entrada *(input)* se copia sobre *Output* cuando *Output* está vacío.

7. Ejecutamos las pruebas sobre *EjemploBpelSincronoApplication*

En la ventana *Projects*, expandimos *EjemploBpelSincronoApplication -> Test -> CasoPrueba1*. El nodo *CasoPrueba1* contiene dos ficheros XML: *Input* para la entrada, y *Output* para la salida esperada. Como ya hemos indicado, cada vez que se ejecuta el caso de prueba, la salida actual se compara con el contenido de *Output*. Si coinciden, entonces superamos la prueba y veremos en la ventana Output el mensaje: *Nombre_Caso_Prueba passed*. Si no coinciden, aparecerá el mensaje *Nombre_Caso_Prueba FAILED.*

Clic con el botón derecho sobre el nodo *CasoPrueba1*, y elegimos *Run* en el menú emergente. Nos fijamos en la ventana *Output* y veremos que el test se ha completado y que no pasamos el test, ya que aparece el mensaje *CasoPrueba1 FAILED*. En este caso, como el fichero *Output* está vacío, se nos pregunta si queremos sobreescribirlo. Contestamos que sí. Después de la primera ejecución, el fichero *Output.xml* ya no está vacío, por lo que su contenido será preservado y no será sobreescrito por el nuevo resultado. Si ejecutamos de nuevo el test, podremos ver el mensaje: *CasoPrueba1 passed*.

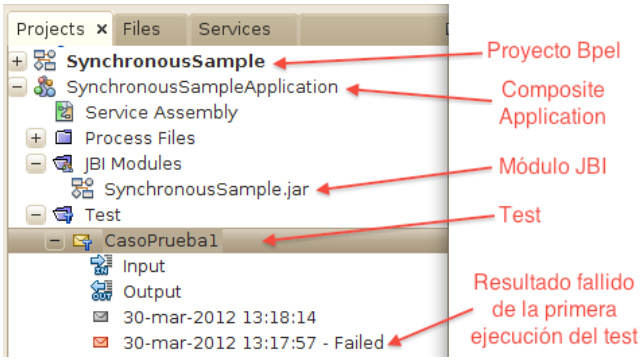

Fig. 12 Casos de prueba para nuestro proyecto Bpel

PARTE 2: Cambios en la lógica de negocio

Modificar la lógica del proceso BPEL de forma que si la cadena de entrada es `null`, devuelva el mensaje de error "*ERROR: Valor null*". Si la cadena de entrada no es `null` y es `"Hola mundo"`, entonces debe devolver la misma cadena como salida, y si no es igual a `"Hola Mundo"` debe devolver *"Hola cadena"*, siendo *"cadena"* el valor de la cadena de entrada.

Añadir dos casos de prueba nuevos, con nombres *TestHolaMundo* y *TestNull*. Para el primer test, el valor de la cadena de entrada debe ser *"Hola mundo"*, para el segundo test, el valor de entrada debe ser *"null"*.

Notas:

a) Cuando añadamos el nuevo *"if"* en la vista de diseño, y pasemos a la vista de mapeado, en el panel de la derecha vemos que únicamente nos aparece el nodo *"if"*. Si hacemos doble clic sobre él, nos generará el nuevo sub-elemento *"boolean condition".*

b) Una forma de comprobar si un elemento es *"null"* es comprobando si el número de elementos del nodo correspondiente es cero (menú *"Node"*, función *"Count"*). La siguiente figura ilustra la comprobación de que la cadena de caracteres de entrada no sea *"null"*.

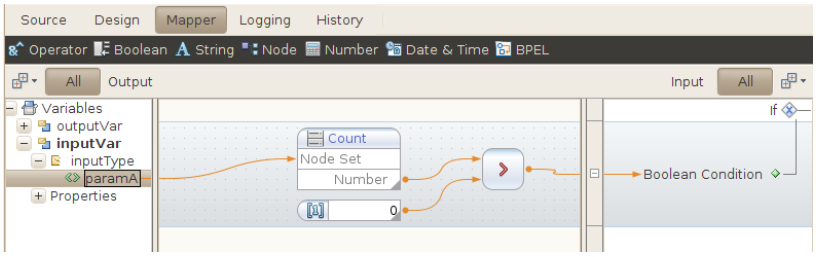

Fig. 13 Comprobación de entrada con valor NULL

c) Después de hacer las modificaciones en la lógica del proceso BPEL, hay que compilar de nuevo el proyecto BPEL, borrar el componente JBI de la *Composite Application* (*EjemploBpelSincrono.jar*) con botón derecho seleccionando "Delete", a continuación añadimos de nuevo el componente JBI, y volvemos a desplegar *EjemploBpelSincronoApplication.*

d) En el mensaje SOAP de entrada para el caso de prueba *TestNull*, simplemente *"borra"* la línea `<syn:paramA>?string?</syn:paramA>` del fichero *input.xml.*