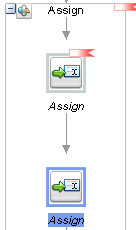
Probando transformaciones entre **Schemas**

\* En este tutorial vamos a probar directo en el pipeline la transformación creada.

\* Para este tutorial se añadió un “**XQuery”** sencillo que mapea el id de entrada hacia otro id. De esta forma podremos ver el operador de asignación y el uso de variables.

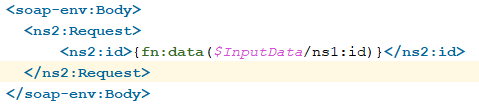
\* Debe tener en cuenta que todos los “**body”** provenientes y salientes en un “**request**” pertenecen al “**namespace**” de “**soap envelope**”. Por lo que una asignación de una conversión al “**body**” del response directamente dará error. Para resolver este problema, en el código del “**XQuery”** debemos poner lo siguiente.

\* Tenga en cuenta que la dirección de las flechas marca la dirección que serán ejecutados los nodos.

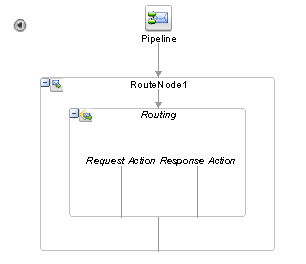


Esto quiere decir que el “**asign**” de arriba se ejecuta primero que el de abajo.

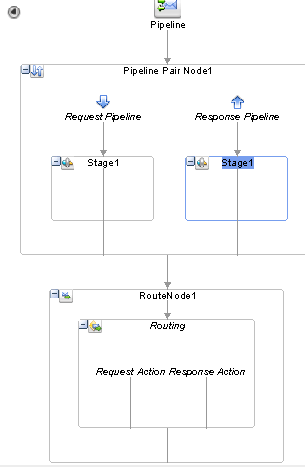




1. Debemos ir al pipeline para modificar el proceso de transformación de los datos.
2. Comenzamos con el “**Pipeline**” generado por defecto por el adaptador de base de datos.

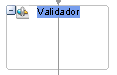


1. Para no modificar internamente el comportamiento del “**RouteNode1**” vamos a agregar un [] para poder trabajar con las peticiones de entrada y salida.
   1. Después de agregado quedaría de la siguiente forma.

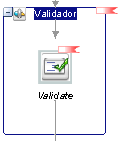


Validador

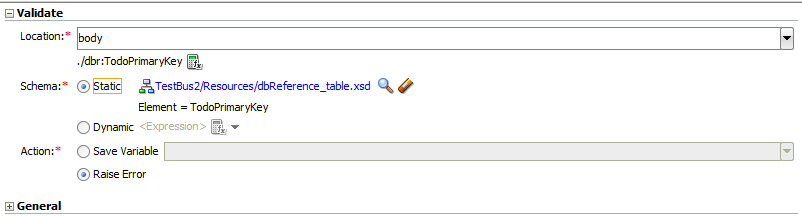
1. Vamos a cambiar el nombre del “**Stage**” del “**Request Pipeline**” por “**Validador**”. Además, le vamos a agregar un validador de datos de entrada. En este caso queremos que valide que el dato que entra sea el “**TodoPrimaryKey”**
   1. **Cambiar el nombre:** Clic sobre **“Stage1”** y le ponemos **“Validador”**



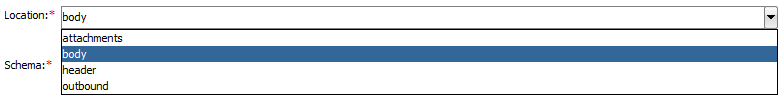
* 1. Agregamos un nodo de validación [] dentro del stage “**Validador**”.



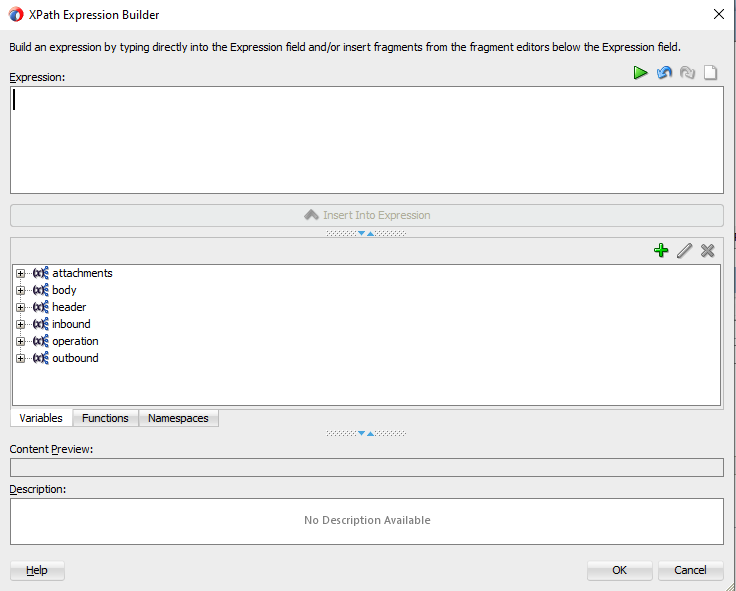
* + 1. El icono () significa que hay un error dentro.
  1. Presionamos sobre el nodo “**Validate**” y vamos a sus “**properites**”.



* + 1. Vamos por parte:
       1. **Location**: Lugar de la petición entrante que vamos a validar.



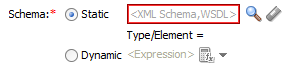
* + - 1. Debajo de “**location**” []: Buscar la variable que vamos a validar.



* + - 1. De ahí nos interesa el árbol de abajo. Desplegamos el “**body**” y seguimos la siguiente línea: ObtenerId > $body – request > TodoPrimaryKey.



* + - 1. Con “**TodoPrimaryKey**” marcado presionamos el botón [] y damos [**OK**].
    1. Ahora tenemos que decir que “**schema**” vamos a utilizar para la validación:

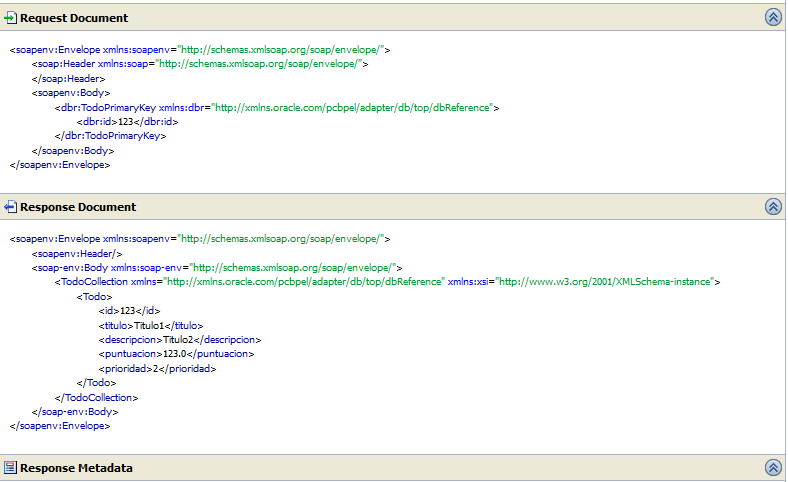


* + 1. Lo mantenemos en “**Static**” y damos clic sobre la [] para buscar el “**Schema**” a utilizar.

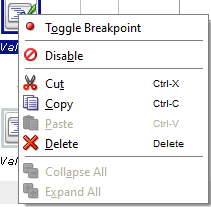


* + 1. En este caso queremos si lo que está entrando es un “**TodoPrimaryKey**”
    2. Presionamos [**OK**] y ya tenemos un validador creado.

1. Quedaría de la siguiente forma:
2. 
   1. Como podemos observar el icono () ya no está; por lo que no tenemos errores en la validación.
3. Ahora vamos a probar. Realizamos los mismos pasos que en **tutorial 2 – 4.2.**



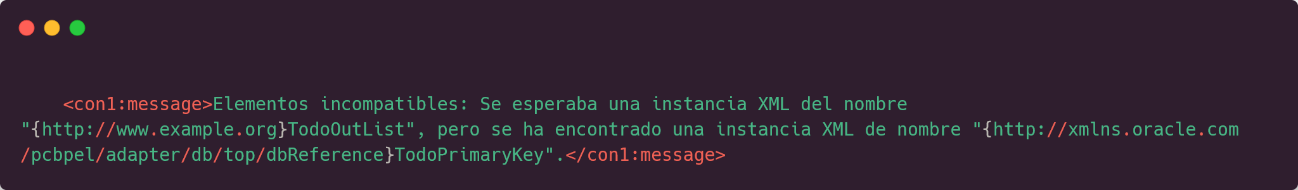
* 1. Aquí podemos ver que pasó sin problemas el validador. Ya que al buscar el valor 123, el sistema nos devuelve el Todo relacionado con él.
  2. Vamos a hacer un cambio en el validador y vamos a ponerle que espera recibir una colección de todo.
     1. En este caso queremos que solo se ejecute el segundo validador y quitar del medio al primer validador. Sería un problema si lo borrásemos, ya que sabemos que funciona correctamente.
     2. Para ello damos clic derecho sobre el nodo y seleccionamos “**disable**”. De esta manera no se tiene en cuenta en el flujo y no tenemos que borrarlo.



* 1. Desplegamos y probamos el pipeline.



* 1. Aquí podemos observar que el validador funcionó. Estábamos esperando un “**TodoCollection**” como entrada, pero lo que recibimos fue un “**TodoPrimaryKey**”. En el siguiente fragmento de código de respuesta se puede ver mejor.



1. Ya de esta forma conocemos como trabajan los validadores, por lo que si necesitamos comprobar que alguna parte del “**request**” o “**response**” está bien formada para evitar errores; podemos usar los validadorse.

Asignar

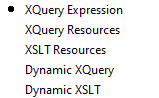
1. Mantendremos los cambios anteriormente realizados, excepto que deshabilitaremos el error provocado.
2. Ahora, el “**request**” hacia el servicio de base de datos espera un id. En este caso realizaremos la siguiente operación.
   1. Usaremos el mapeo simple para asignarle al “**body**” del “**request**” el valor mapeado.
   2. Para ello, primero creamos un “stage” [] debajo de las validaciones y lo llamaremos asignación.
   3. Dentro, pondremos un nodo de asignación [].
   4. En sus “**properties**” podremos ver dos campos:



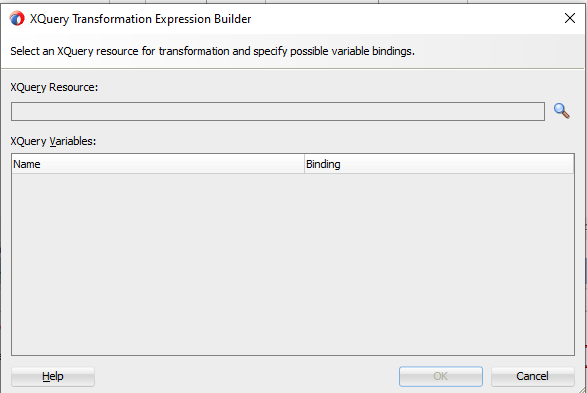
* + 1. **Value**: Aquí definiremos el valor a mapear mediante [].
    2. **Variable**: La variable donde vamos a guardar el resultado.
       1. Podemos crear una o utilizar las globales creadas por defecto.



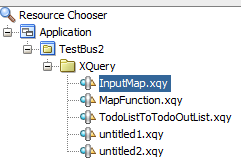
* 1. Vamos a definir la función de transformación [] a utilizar:
     1. Cuando desplegamos el icono de [] nos salen las siguientes opciones.



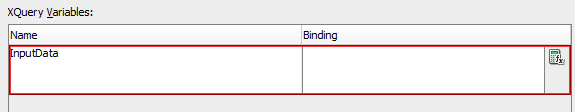
* + - 1. **XQuery Expression**: Crear una expresión de tipo “**XQuery”** en ese mismo momento.
      2. **XQuery Resources y XSLT Resources**: Nos permitirán buscar los archivos “**XQuery”** o “**XSL”** creados para utilizar.
      3. Los dos últimos, es selección dinámica del tipo de mapeo a elegir según un criterio.
    1. En este caso vamos a trabajar con (**XQuery Resource**).



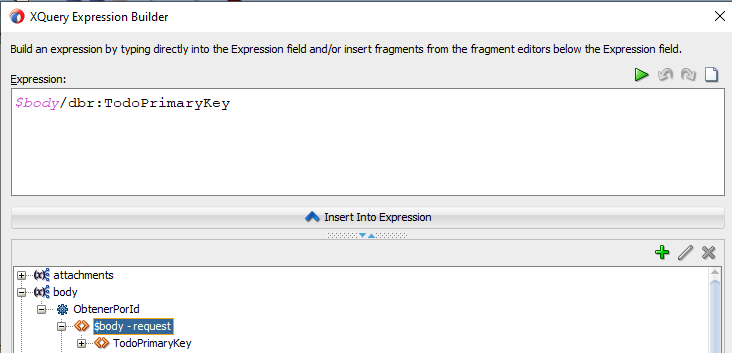
* + 1. Ahora nos pide seleccionar el recurso “**XQuery**” creado.
    2. Damos en la lupa [] para buscar dentro de nuestra jerarquía de carpetas el archivo a utilizar.



* + 1. Seleccionamos el que utilizaremos para el mapeo y damos [**ok**].
    2. Ahora nos toca realizar la vinculación de las variables.



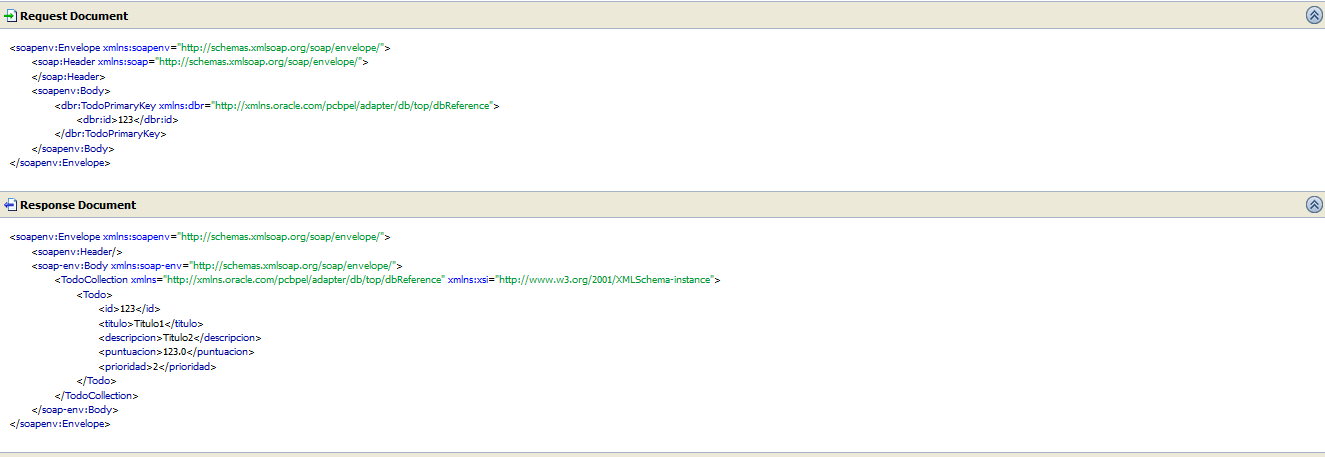
* + 1. Presionamos el ícono de [] y buscamos dentro del creador de expresiones que datos vamos a utilizar como parámetro de entrada.
    2. En este caso vamos a utilizar el TodoPrimeryKey del body > request ya que es el valor que estamos intentando mapear.



* + 1. Presionamos [**OK**] y luego [**OK**]
  1. Ahora solo nos falta decir a que variable le vamos a asignar el valor transformado. En caso que no lo hagamos, se mantendrá el error.

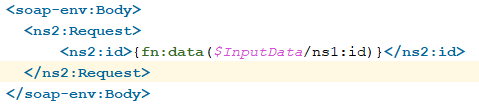


* + 1. Vamos a seleccionar la variable “**body**”, ya que nuestro interés es guardar la transformación en el mismo cuerpo de la petición.
  1. Realizamos “**Deploy**” y probamos que la petición al “**Pipeline**” devuelve un resultado valido.

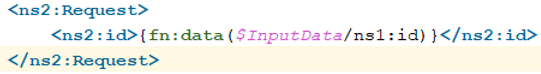


* 1. Como podemos observar, los cambios realizados funcionaron correctamente.

1. Ahora vamos a añadir dos variaciones:
   1. Asignar un valor al “**body**” de forma incorrecta, para verificar el valor del error.
   2. Quitar el “**soap-envelope**” de la transformación y hacerlo mediante dos pasos de asignación.
      1. Asignar transformación
      2. Cubrir variable de transformación con “**soap-envelope”**
2. Vamos para el primer caso. Provocar un error en el body. Para ello vamos a nuestro “**XQuery**” y quitamos la creación del “**namespace**” (**soap-env**)
   1. Antes de modificar:



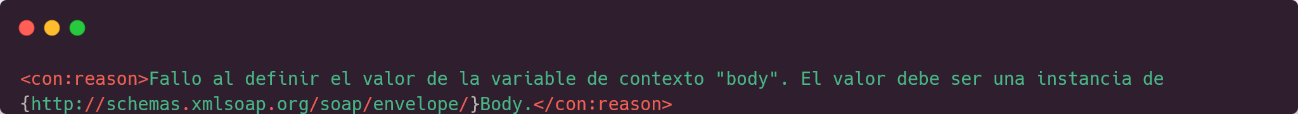
* 1. Después de modificar



* 1. Esto debería desatar un error, ya que el “**body**” debe pertenecer al “**namespace**” de (**soap-envelope**). Desplegamos y probamos.
     1. El resultado fue el esperado:



* + 1. En este fragmento podremos leer mejor el error:



1. Ahora toca el turno de dividir la asignación en 2.
   1. Vamos para el primer paso. Crear una asignación a una variable propia.
      1. Añadimos el asignar al pipeline y al igual que en el flujo normal de asignación visto anteriormente, le asignamos el valor mediante el “**XQuery**”
      2. La diferencia está en que en la (**variable**) vamos a crear una propia:



* + - 1. Para lograr esto, solo tenemos que poner el nombre que nos guste.
  1. Teniendo esto, creamos un asignar debajo para utilizar la variable (**requestIdFromOrigin**) cubierta por un “**soap-envelope**".
     1. Para este caso el valor seleccionado en [] tiene que ser “**XQeury Expression**”
     2. En este caso, la expresión a utilizar no la vamos a tomar del “**request**” como hicimos anteriormente, en este caso la crearemos manualmente. Por lo que solo trabajaremos en:



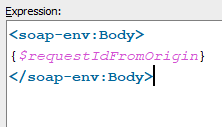
* + 1. Para llamar una variable dentro de expresión, tenemos que usar la siguiente notación:



* + 1. Entonces, tendremos que encerrar esa variable dentro de:



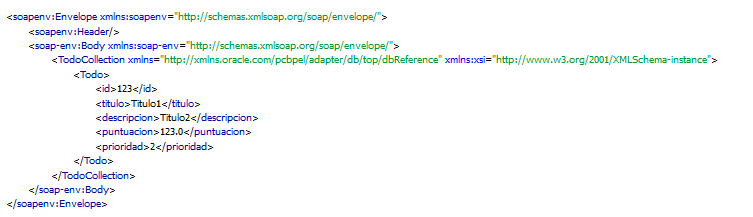
* + 1. Uniendo las dos piezas anteriores y poniéndolas dentro del área de expresión, quedaría:



* + 1. Ahora solo nos queda asignar la variable a “**body”**



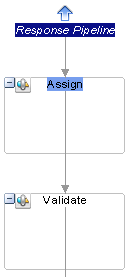
* 1. Desplegamos y probamos. Si todo va bien, deberíamos tener una respuesta similar a la del flujo normal.



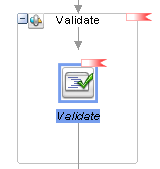
* 1. Como podemos ver, la respuesta fue la deseada, por lo que todo el proceso funcionó correctamente.

Asignar en entrada y salida.

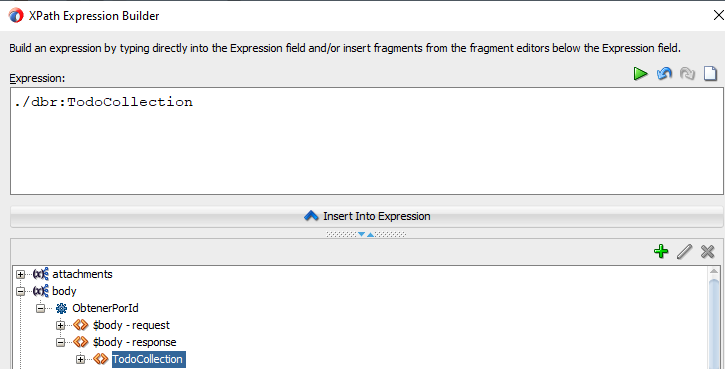
1. En este caso, vamos a mantener todas las validaciones y asignaciones realizadas en los pasos anteriores (flujo correcto).
2. En este caso vamos a crear:
   1. Validador del response original.
   2. Asignación a variable de la conversión del “**XQuery”**
   3. Envolver la variable transformada con “**soap-envelope**”. Esto lo haremos debido a que no queremos añadir datos extras a la transformación.
3. Comenzando por el validador:
   1. Añadimos un “**stage**” en el “**response**”:

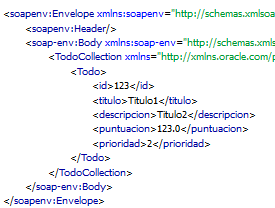


* 1. Dentro vamos a agregarle un nodo de “**Validate**”:

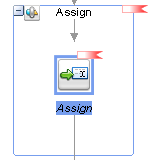


* + 1. En los “**properties**” pondremos los siguientes datos:
       1. 
       2. : Agregaremos el “**TodoCollection**” ya que esos son los valores que vienen desde el servicio de base de datos.

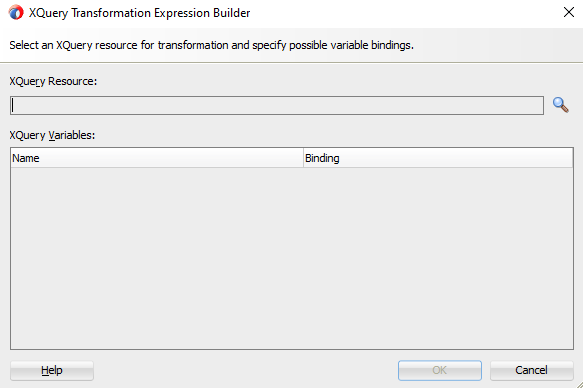


* + - 1. 
         1. Por último, definimos el “**Schema**” que deberían seguir esos datos. En este caso decidimos que deben seguir la estructura de “**TodoCollection**”.
  1. Desplegamos y probamos para ver si todo está bien.
  2. 
  3. Los resultados devueltos son los esperados y no existió ningún error. Por lo que podemos decir que el validador está funcionando.

1. Ahora viene el turno de la asignación doble. Vamos a hacer igual que en el ejemplo anterior.
   1. Primero hacemos la asignación en variable:



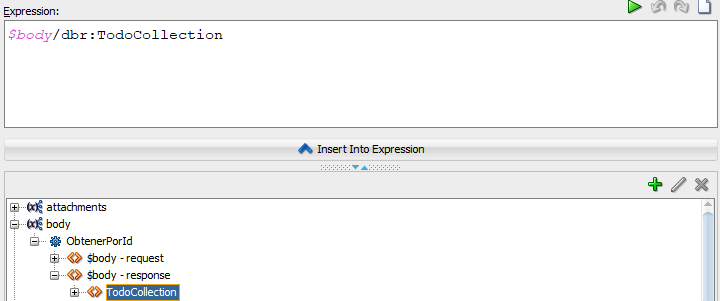
* 1. En los “**properites”**:
     1. : Seleccionamos el “**XQuery Resource**”.



* + 1. Damos en la lupa [] para seleccionar el archive “**XQuery**” a utilizar.



* + 1. Le decimos que dato utilizar del “**response**” mediante []:



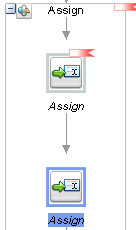
* + 1. Cuanto tengamos definidos todos los parámetros deseados, volvemos a los “**properties**” para crear la variable:



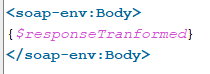
* 1. Ya con esto tenemos el primer paso. Ahora toca envolver esa variable entre:



* 1. Vamos al pipeline y añadimos un nodo asignar [] sobre el ya creado.



* 1. Para quitar la bandera de error, vamos a los “**properties**” a completar sus valores obligatorios.
     1. : Seleccionamos la opción de “**XQuery Expression**”.
        1. Añadimos la expresión de envoltorio con la variable:



* + - 1. Volvemos a los “**properites**”.
    1. Ahora nos toca decir que la variable a utilizar para guardar es “**body**”.

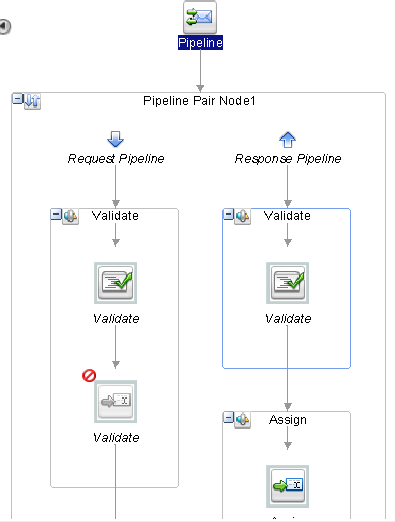


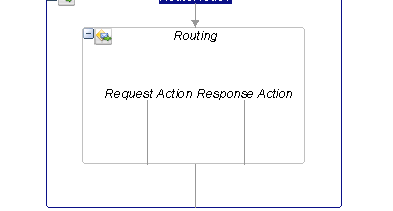
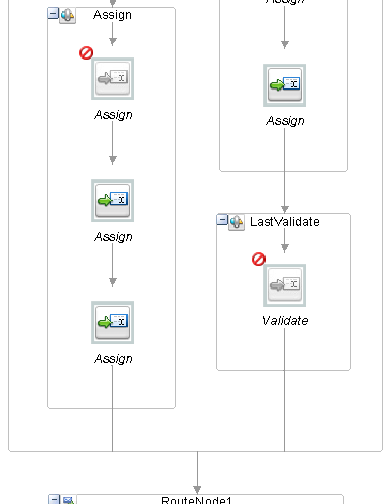
* 1. Ya con esto tenemos la transformación del response completada.

1. Ahora solo queda desplegar y probar.



1. El resultado final es el siguiente:





Tip:

Si queremos ver los pasos seguidos por nuestra petición. Podemos ir al apartado de “**Invocation Trace**”. Y en formato árbol nos va a enseñar el trayecto que sigue nuestra aplicación.

