# Arquitectura Orientada a Servicios

## Orquestación de Servicios: BPEL

Edwin Salvador

10 de julio de 2015

Sesión 14

# Contenido I

# Introducción

- SOA es la respuesta a la integración de aplicaciones.

# Introducción

- SOA es la respuesta a la integración de aplicaciones.
- Existen diferentes enfoques para integrar los servicios:

# Introducción

- SOA es la respuesta a la integración de aplicaciones.
- Existen diferentes enfoques para integrar los servicios:
  - **Bottom-up** considera que las diferentes aplicaciones de la empresa exponen sus funcionalidades a través de WS. Así se puede acceder a diferentes funcionalidades de diferentes aplicaciones (antiguas o nuevas) de una forma común a través de WS.

# Introducción

- SOA es la respuesta a la integración de aplicaciones.
- Existen diferentes enfoques para integrar los servicios:
  - **Bottom-up** considera que las diferentes aplicaciones de la empresa exponen sus funcionalidades a través de WS. Así se puede acceder a diferentes funcionalidades de diferentes aplicaciones (antiguas o nuevas) de una forma común a través de WS.
  - **Top-down** realiza un análisis que requiere que los procesos de negocio existentes se vuelvan orientados a servicios y promueve el re-alineamiento del modelo de negocios de una organización.

# Introducción

- SOA es la respuesta a la integración de aplicaciones.
- Existen diferentes enfoques para integrar los servicios:
  - **Bottom-up** considera que las diferentes aplicaciones de la empresa exponen sus funcionalidades a través de WS. Así se puede acceder a diferentes funcionalidades de diferentes aplicaciones (antiguas o nuevas) de una forma común a través de WS.
  - **Top-down** realiza un análisis que requiere que los procesos de negocio existentes se vuelvan orientados a servicios y promueve el re-alineamiento del modelo de negocios de una organización.
- Simplemente desarrollar servicios web y exponer sus funcionalidades no es suficiente.

# Introducción

- SOA es la respuesta a la integración de aplicaciones.
- Existen diferentes enfoques para integrar los servicios:
  - **Bottom-up** considera que las diferentes aplicaciones de la empresa exponen sus funcionalidades a través de WS. Así se puede acceder a diferentes funcionalidades de diferentes aplicaciones (antiguas o nuevas) de una forma común a través de WS.
  - **Top-down** realiza un análisis que requiere que los procesos de negocio existentes se vuelvan orientados a servicios y promueve el re-alineamiento del modelo de negocios de una organización.
- Simplemente desarrollar servicios web y exponer sus funcionalidades no es suficiente.
- Se necesita componer estos servicios en un orden correcto, para ello podemos usar dos técnicas: **orquestación y coreografía**.

# Contenido I

# Orquestación

- Un proceso central (puede ser otro WS) controla todos los WS implicados en la realización de una tarea.

# Orquestación

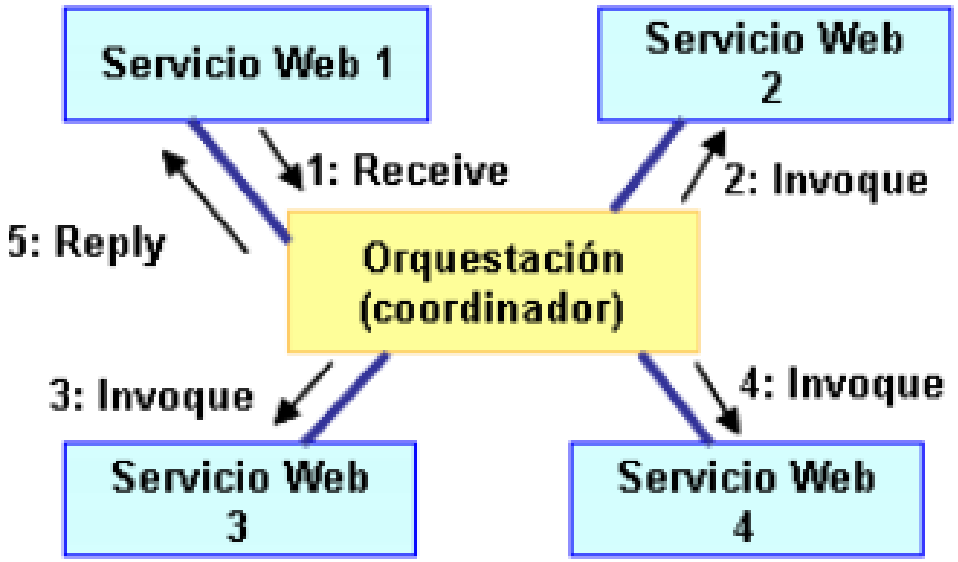
- Un proceso central (puede ser otro WS) controla todos los WS implicados en la realización de una tarea.
- Los WS no conocen que forman parte de una composición, solo el WS central es “consciente” de la meta a conseguir.

# Orquestación

- Un proceso central (puede ser otro WS) controla todos los WS implicados en la realización de una tarea.
- Los WS no conocen que forman parte de una composición, solo el WS central es "consciente" de la meta a conseguir.
- Se centraliza mediante definiciones explícitas de las operaciones y del orden en que se invocan.

# Orquestación

- Un proceso central (puede ser otro WS) controla todos los WS implicados en la realización de una tarea.
- Los WS no conocen que forman parte de una composición, solo el WS central es "consciente" de la meta a conseguir.
- Se centraliza mediante definiciones explícitas de las operaciones y del orden en que se invocan.
- Se utiliza en procesos de negocio privados.



# Coreografía

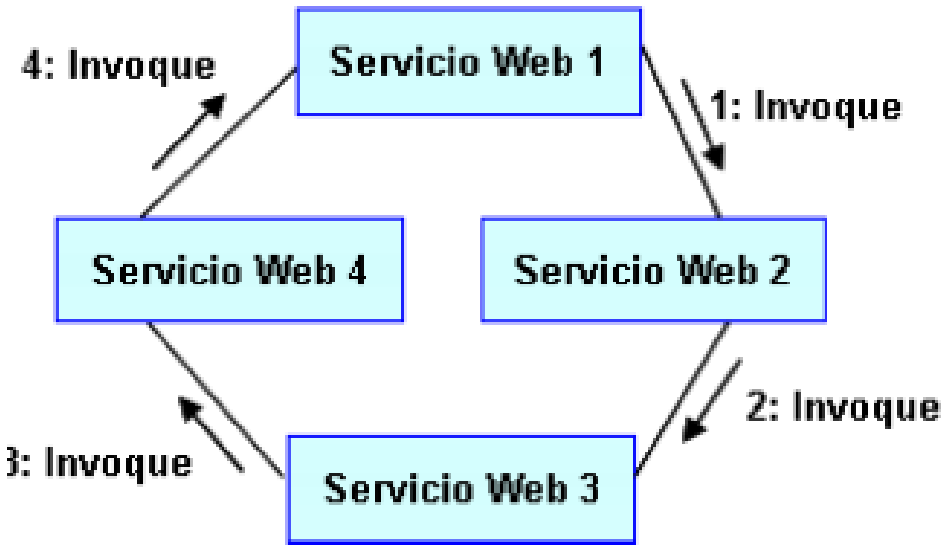
- No hay coordinador central.

# Coreografía

- No hay coordinador central.
- Es un esfuerzo colaborativo centrado en el intercambio de mensajes en procesos de negocio públicos.

# Coreografía

- No hay coordinador central.
- Es un esfuerzo colaborativo centrado en el intercambio de mensajes en procesos de negocio públicos.
- Todos los participantes deben estar informados del proceso de negocio, operaciones a ejecutar, mensajes a intercambiar y tiempo a intervenir.



# Contenido I

# ¿Por qué Orquestar WS?

- Los WS proporcionan una interfaz común que unifica el modelo de integración para todos los servicios independientemente de su origen.

# ¿Por qué Orquestar WS?

- Los WS proporcionan una interfaz común que unifica el modelo de integración para todos los servicios independientemente de su origen.
- Amazon y Google exponen sus interfaces en forma de servicios web par que sus aplicaciones sean integrables.

# ¿Por qué Orquestar WS?

- Los WS proporcionan una interfaz común que unifica el modelo de integración para todos los servicios independientemente de su origen.
- Amazon y Google exponen sus interfaces en forma de servicios web par que sus aplicaciones sean integrables.
- El siguiente paso de brindar WS es la orquestación de los mismos en procesos de negocio de más alto nivel.

# ¿Por qué Orquestar WS?

- Los WS proporcionan una interfaz común que unifica el modelo de integración para todos los servicios independientemente de su origen.
- Amazon y Google exponen sus interfaces en forma de servicios web par que sus aplicaciones sean integrables.
- El siguiente paso de brindar WS es la orquestación de los mismos en procesos de negocio de más alto nivel.
- La facilidad de descubrimiento y el bajo acoplamiento de los WS ayudan a la orquestación sin utilizar herramientas de programación tradicionales como C o Java.

# ¿Por qué Orquestar WS?

- Los WS proporcionan una interfaz común que unifica el modelo de integración para todos los servicios independientemente de su origen.
- Amazon y Google exponen sus interfaces en forma de servicios web par que sus aplicaciones sean integrables.
- El siguiente paso de brindar WS es la orquestación de los mismos en procesos de negocio de más alto nivel.
- La facilidad de descubrimiento y el bajo acoplamiento de los WS ayudan a la orquestación sin utilizar herramientas de programación tradicionales como C o Java.
- **La orquestación de servicios Web pretende proporcionar una aproximación abierta, basada en estándares, para conectar servicios Web de forma conjunta con el objetivo de crear procesos de negocio de más alto nivel.**

# Contenido I

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.
- Facilita la ejecución en varias plataformas.

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.
- Facilita la ejecución en varias plataformas.
- Brinda mayor libertad a clientes.

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.
- Facilita la ejecución en varias plataformas.
- Brinda mayor libertad a clientes.
- Menores costos de mantenimiento

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.
- Facilita la ejecución en varias plataformas.
- Brinda mayor libertad a clientes.
- Menores costos de mantenimiento
- Menores costes de soporte

# ¿Por qué Orquestar con BPEL?

- BPEL es un lenguaje estándar para la integración y automatización de procesos.
- Facilita la ejecución en varias plataformas.
- Brinda mayor libertad a clientes.
- Menores costos de mantenimiento
- Menores costes de soporte
- Amplía el grupo de dearrolladores (tareas altamente especializadas)

# ¿Por qué Orquestar con BPEL?

También BPEL brinda soporte para:

- **Elevados tiempos de ejecución**: procesos de negocio toman bastante tiempo. Al hacerlo con Java necesitaríamos monitorear constantemente los procesos terminados y en espera por una respuesta.

# ¿Por qué Orquestar con BPEL?

También BPEL brinda soporte para:

- **Elevados tiempos de ejecución**: procesos de negocio toman bastante tiempo. Al hacerlo con Java necesitaríamos monitorear constantemente los procesos terminados y en espera por una respuesta.
- **Compensación** dar marcha atrás a efectos de actividades que ya se han llevado a cabo como parte de un proceso de negocio que está siendo abandonado.

# ¿Por qué Orquestar con BPEL?

También BPEL brinda soporte para:

- **Elevados tiempos de ejecución**: procesos de negocio toman bastante tiempo. Al hacerlo con Java necesitaríamos monitorear constantemente los procesos terminados y en espera por una respuesta.
- **Compensación** dar marcha atrás a efectos de actividades que ya se han llevado a cabo como parte de un proceso de negocio que está siendo abandonado.
- **Reacción ante eventos** de tipo mensaje generados por mensajes de entrada a través de la invocación de operaciones sobre los `port types`. Los eventos de tipo alarma están relacionados con el tiempo y se generan después de un tiempo especificado.

# ¿Por qué orquestar con BPEL?

- **Modelado de actividades concurrentes** se modela mediante la actividad <flow>. Podemos especificar qué actividades pueden comenzar y cuando.

# ¿Por qué orquestar con BPEL?

- **Modelado de actividades concurrentes** se modela mediante la actividad <flow>. Podemos especificar qué actividades pueden comenzar y cuando.
- **Modelos con estado** Los procesos de negocio requieren el uso de un modelo con estado. Cuando un cliente comienza un proceso de negocio, se crea una nueva instancia que "vive" mientras no termine el proceso de negocio. Los mensajes enviados al proceso necesitan ser entregados a las instancias correctas del proceso de negocio.

# ¿Por qué orquestar con BPEL?

- **Modelado de actividades concurrentes** se modela mediante la actividad <flow>. Podemos especificar qué actividades pueden comenzar y cuando.
- **Modelos con estado** Los procesos de negocio requieren el uso de un modelo con estado. Cuando un cliente comienza un proceso de negocio, se crea una nueva instancia que "vive" mientras no termine el proceso de negocio. Los mensajes enviados al proceso necesitan ser entregados a las instancias correctas del proceso de negocio.

BPEL es la opción más adecuada para orquestar procesos frente a otros lenguajes como Java.

# Contenido I

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.
- Soporta tecnologías de WS (SOAP, WSDL, UDDI, WS-Reliable Messaging. WS-Addressing, WS-Coordination y WS-Transaction).

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.
- Soporta tecnologías de WS (SOAP, WSDL, UDDI, WS-Reliable Messaging. WS-Addressing, WS-Coordination y WS-Transaction).
- Diseñado para definir procesos de negocio.

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.
- Soporta tecnologías de WS (SOAP, WSDL, UDDI, WS-Reliable Messaging. WS-Addressing, WS-Coordination y WS-Transaction).
- Diseñado para definir procesos de negocio.
- Puede utilizarse:

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.
- Soporta tecnologías de WS (SOAP, WSDL, UDDI, WS-Reliable Messaging. WS-Addressing, WS-Coordination y WS-Transaction).
- Diseñado para definir procesos de negocio.
- Puede utilizarse:
  - **dentro de una empresa** estandariza la integración de aplicaciones y extiende la integración de sistemas previamente aislados.

# Lenguaje BPEL

- BPEL (*Business Process Execution Language*)
- Lenguaje de orquestación de servicios.
- Basado en XML.
- Soporta tecnologías de WS (SOAP, WSDL, UDDI, WS-Reliable Messaging. WS-Addressing, WS-Coordination y WS-Transaction).
- Diseñado para definir procesos de negocio.
- Puede utilizarse:
  - **dentro de una empresa** estandariza la integración de aplicaciones y extiende la integración de sistemas previamente aislados.
  - **entre varias empresas** Permite una integración más fácil y efectiva con *partners* del negocio.

# BPEL

Un **proceso de negocio** (Business Process) es un flujo de trabajo, formado por una serie de pasos o actividades, que son requeridas para completar una transacción de negocio.
BPEL soporta dos tipos de procesos de negocio:

- **Procesos de negocio ejecutables**: especifican los detalles exactos de los procesos del negocio y pueden ser ejecutados en una máquina de orquestación (orchestration engine). **En la mayoría de los casos BPEL se utiliza para procesos ejecutables.**

# BPEL

Un **proceso de negocio** (Business Process) es un flujo de trabajo, formado por una serie de pasos o actividades, que son requeridas para completar una transacción de negocio.
BPEL soporta dos tipos de procesos de negocio:

- **Procesos de negocio ejecutables**: especifican los detalles exactos de los procesos del negocio y pueden ser ejecutados en una máquina de orquestación (orchestration engine). **En la mayoría de los casos BPEL se utiliza para procesos ejecutables.**
- **Procesos abstractos de negocio:** especifican solamente el intercambio de mensajes públicos entre las partes implicadas, sin incluir detalles específicos de los flujos de los procesos. No son ejecutables y raramente se utilizan.

# Contenido I

# Estructura de procesos BPEL

- Un **proceso BPEL** especifica el orden exacto en el que se deben invocar los servicios Web participantes (secuencial o paralelo).

# Estructura de procesos BPEL

- Un **proceso BPEL** especifica el orden exacto en el que se deben invocar los servicios Web participantes (secuencial o paralelo).
- Podemos expresar un comportamiento condicional, construir bucles, declarar variables, copiar, asignar valores, definir manejadores de fallos, etc.

# Estructura de procesos BPEL

- Un **proceso BPEL** especifica el orden exacto en el que se deben invocar los servicios Web participantes (secuencial o paralelo).
- Podemos expresar un comportamiento condicional, construir bucles, declarar variables, copiar, asignar valores, definir manejadores de fallos, etc.
- Esto nos permite construir procesos de negocio complejos de una manera algorítmica.

# Estructura de procesos BPEL

- Un **proceso BPEL** especifica el orden exacto en el que se deben invocar los servicios Web participantes (secuencial o paralelo).
- Podemos expresar un comportamiento condicional, construir bucles, declarar variables, copiar, asignar valores, definir manejadores de fallos, etc.
- Esto nos permite construir procesos de negocio complejos de una manera algorítmica.
- El proceso de negocio BPEL recibe una petición de un cliente (aplicación cliente u otro WS), el proceso invoca a diversos WS (teniendo en cuenta el WSDL) para atender a esta petición y finalmente responde al cliente que realizó la llamada.

# Estructura de procesos BPEL

- Un **proceso BPEL** especifica el orden exacto en el que se deben invocar los servicios Web participantes (secuencial o paralelo).
- Podemos expresar un comportamiento condicional, construir bucles, declarar variables, copiar, asignar valores, definir manejadores de fallos, etc.
- Esto nos permite construir procesos de negocio complejos de una manera algorítmica.
- El proceso de negocio BPEL recibe una petición de un cliente (aplicación cliente u otro WS), el proceso invoca a diversos WS (teniendo en cuenta el WSDL) para atender a esta petición y finalmente responde al cliente que realizó la llamada.
- Para los clientes de un proceso BPEL, este es como otro WS.

# Definición de un procesos BPEL

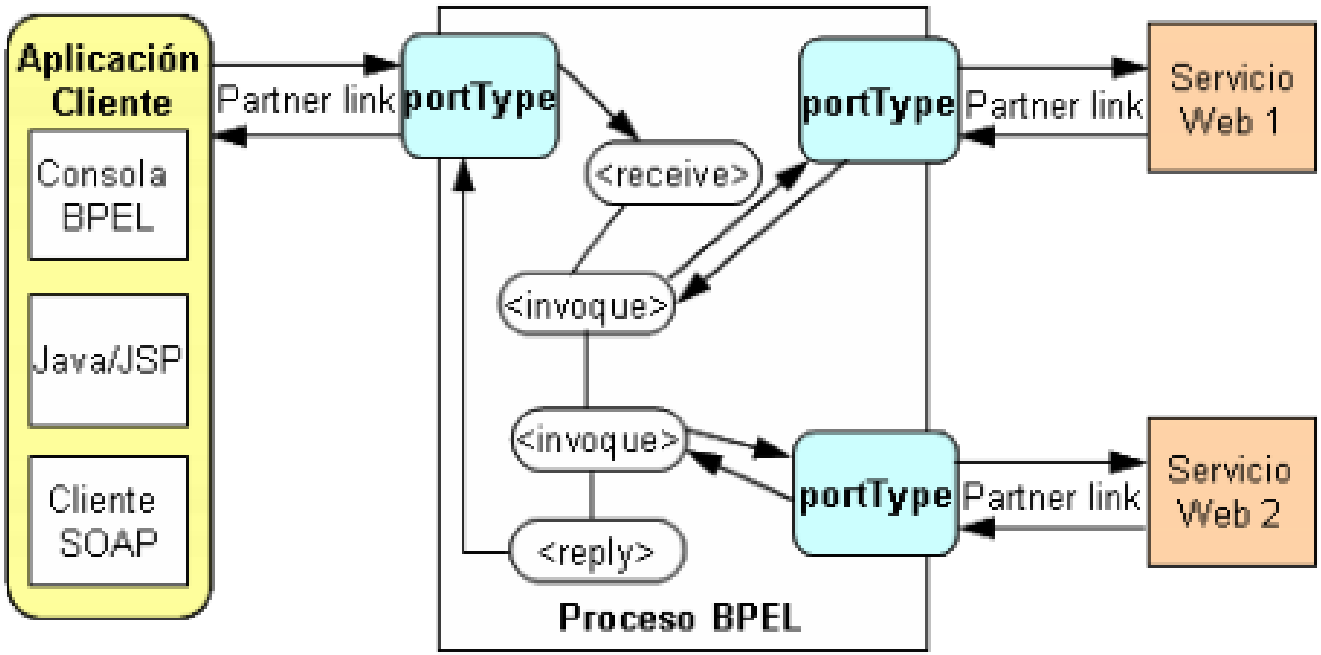
- La interfaz de un WS BPEL utiliza un conjunto de `portTypes` a través de cuales ofrece operaciones como cualquier otro WS.

# Definición de un procesos BPEL

- La interfaz de un WS BPEL utiliza un conjunto de `portTypes` a través de cuales ofrece operaciones como cualquier otro WS.
- Un portType puede contener varias operaciones.

# Definición de un procesos BPEL

- La interfaz de un WS BPEL utiliza un conjunto de `portTypes` a través de cuales ofrece operaciones como cualquier otro WS.
- Un portType puede contener varias operaciones.
- Para invocar al proceso de negocio descrito con BPEL tenemos que invocar al WS resultante de la composición.



# Tipos de procesos BPEL

- Dos tipos de procesos:

# Tipos de procesos BPEL

- Dos tipos de procesos:
  - **Síncronos** Bloquea al cliente que usa el proceso BPEL hasta que finaliza y devuelve el resultado.

# Tipos de procesos BPEL

- Dos tipos de procesos:
  - **Síncronos** Bloquea al cliente que usa el proceso BPEL hasta que finaliza y devuelve el resultado.
  - **Asíncronos** no bloquea al cliente. Utiliza una llamada *callback* que devuelve un resultado.

# Tipos de procesos BPEL

- Dos tipos de procesos:
  - **Síncronos** Bloquea al cliente que usa el proceso BPEL hasta que finaliza y devuelve el resultado.
  - **Asíncronos** no bloquea al cliente. Utiliza una llamada *callback* que devuelve un resultado.
- Generalmente se utilizan procesos asíncronos para proceso que toman mucho tiempo y procesos síncronos para los que toman poco tiempo.

# Tipos de procesos BPEL

- Dos tipos de procesos:
  - **Síncronos** Bloquea al cliente que usa el proceso BPEL hasta que finaliza y devuelve el resultado.
  - **Asíncronos** no bloquea al cliente. Utiliza una llamada *callback* que devuelve un resultado.
- Generalmente se utilizan procesos asíncronos para proceso que toman mucho tiempo y procesos síncronos para los que toman poco tiempo.
- Si el proceso BPEL utiliza WS asíncronos, entonces el proceso BPEL también lo es.

# Estructura básica de un documento BPEL

```
<process name="nameProcess">

  <partnerLinks>
    <!-- Declaracion de partner links -->
  </partnerLinks>

  <variables>
    <!-- Declaracion de variables -->
  </variables>

  <sequence>
    <!-- Cuerpo principal de la definicion del proceso BPEL -->
  </sequence>
</process>
```

# La etiqueta `process`

## Estructura básica de un documento BPEL

- Aquí se añaden:

# La etiqueta `process`

## Estructura básica de un documento BPEL

- Aquí se añaden:
  - El `targetNamespace`

# La etiqueta `process`

## Estructura básica de un documento BPEL

- Aquí se añaden:
  - El `targetNamespace`
  - Los espacios de nombres para los WSDL de los WS que invoca

# La etiqueta `process`

## Estructura básica de un documento BPEL

- Aquí se añaden:
  - El `targetNamespace`
  - Los espacios de nombres para los WSDL de los WS que invoca
  - El WSDL del proceso BPEL

# La etiqueta process

## Estructura básica de un documento BPEL

- Aquí se añaden:
  - El `targetNamespace`
  - Los espacios de nombres para los WSDL de los WS que invoca
  - El WSDL del proceso BPEL
  - El namespace de para todas la etiquetas y actividades BPEL (el espacio de nombres por defecto):
    `http://schemas.xmlsoap.org/ws/2003/03/business-process/`

```
<process name="nameProcess"
      targetNamespace="http://..."
      xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-
     process/"
      xmlns:sw1="http://namespace del servicio Web sw1"
      xmlns:sw2="http://namespace del servicio Web sw2"
     >

...

</process>
```

# Etiqueta <sequence>

- Contiene el conjunto de actividades que conforman el servicio que proporciona el proceso BPEL.

# Etiqueta <sequence>

- Contiene el conjunto de actividades que conforman el servicio que proporciona el proceso BPEL.
- La etiqueta <process> está siempre presente en un proceso BPEL. Es el elemento raíz.

# Contenido I

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.
- El proceso BPEL interactúa con los WS de dos formas distintas:

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.
- El proceso BPEL interactúa con los WS de dos formas distintas:
  - Invoca operaciones sobre otros WS.

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.
- El proceso BPEL interactúa con los WS de dos formas distintas:
  - Invoca operaciones sobre otros WS.
  - Recibe invocaciones de clientes. El usuario inicial y otros WS que han sido invocados por el proceso BPEL y que está devolviendo respuestas (callbacks).

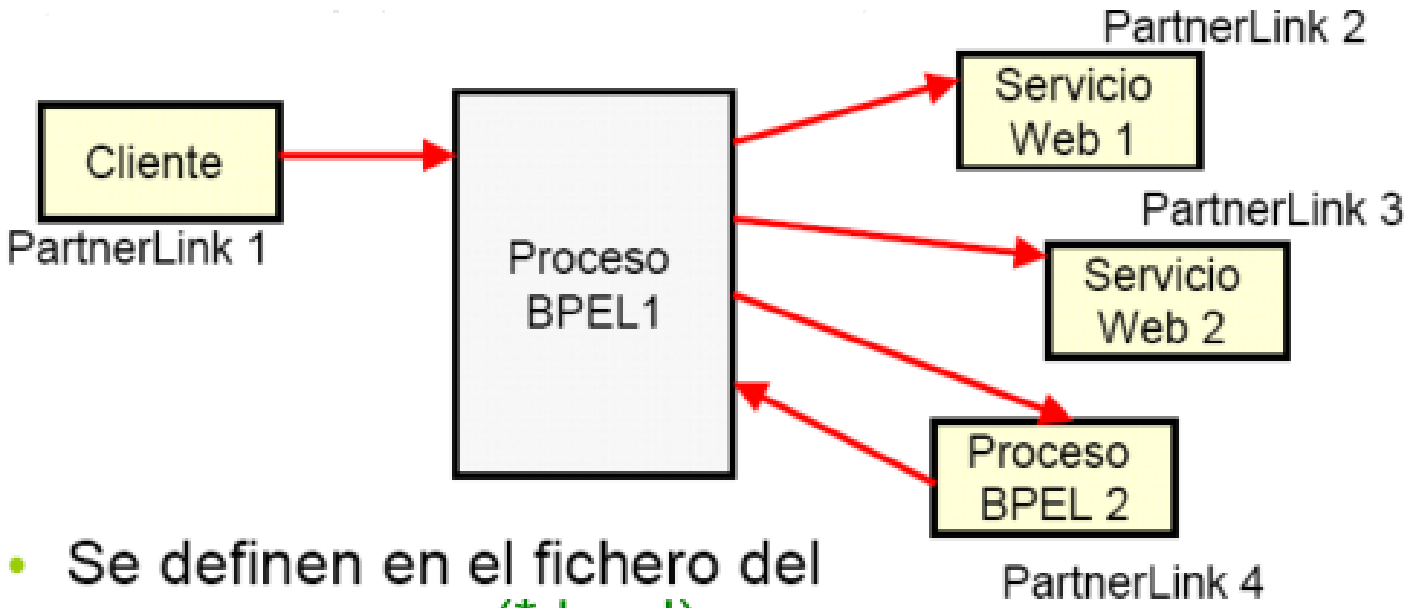
# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.
- El proceso BPEL interactúa con los WS de dos formas distintas:
  - Invoca operaciones sobre otros WS.
  - Recibe invocaciones de clientes. El usuario inicial y otros WS que han sido invocados por el proceso BPEL y que está devolviendo respuestas (callbacks).
- Un proceso BPEL tiene al menos un `partnerLink` (debe ser invocado por un cliente), y tendrá normalmente al menos un partnerLink al que invoque.

# Partner Links

- Es la definición de los enlaces con las diferentes partes que interactúan con el proceso BPEL.
- Cada partner link tiene asociado un `partnerLinkType` que se definirá en el WSDL correspondiente.
- El proceso BPEL interactúa con los WS de dos formas distintas:
  - Invoca operaciones sobre otros WS.
  - Recibe invocaciones de clientes. El usuario inicial y otros WS que han sido invocados por el proceso BPEL y que está devolviendo respuestas (callbacks).
- Un proceso BPEL tiene al menos un `partnerLink` (debe ser invocado por un cliente), y tendrá normalmente al menos un partnerLink al que invoque.
- Los clientes son tratados como partnerLinks para proporcionar interacciones asíncronas y para distinguir entre diferentes clientes y ofrecerles únicamente la funcionalidad que estos están autorizados.

# PartnerLinks




- Se definen en el fichero del proceso BPEL (*.bpel)

# Sintaxis PartnerLink

```
<partnerLinks>
  <partnerLink name="ncname" partnerLinkType="qname"
               myrole="ncname" partnerRole="ncname">
  </partnerLink>
</partnerLinks>
```

- **ncname** y **qname** expresan el formato que deben seguir ciertos nombres dados a los atributos de la etiquetas BPEL.
- **ncname** es una cadena de caracteres que empieza con una letra o un subguión.
- **qname** es un `ncname` que también debe mostrar de que espacio de nombres XML proviene. Ej: `tns:ncname`

# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:

# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:
  - **`myRole`** indica el rol del proceso BPEL. Si solo definimos este atributo entonces cualquier partner puede interaccionar con el proceso BPEL sin ningún requerimiento adcional.

# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:
  - **`myRole`** indica el rol del proceso BPEL. Si solo definimos este atributo entonces cualquier partner puede interaccionar con el proceso BPEL sin ningún requerimiento adcional.
  - **`partnerRole`** indica el rol del partner. Si solo definimos este atributo estamos permitiendo la interacción con un partner que no imponga requerimientos sobre el proceso que haga la llamada.

# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:
  - **`myRole`** indica el rol del proceso BPEL. Si solo definimos este atributo entonces cualquier partner puede interaccionar con el proceso BPEL sin ningún requerimiento adcional.
  - **`partnerRole`** indica el rol del partner. Si solo definimos este atributo estamos permitiendo la interacción con un partner que no imponga requerimientos sobre el proceso que haga la llamada.
- Para una operación síncrona utilizaremos un solo rol.

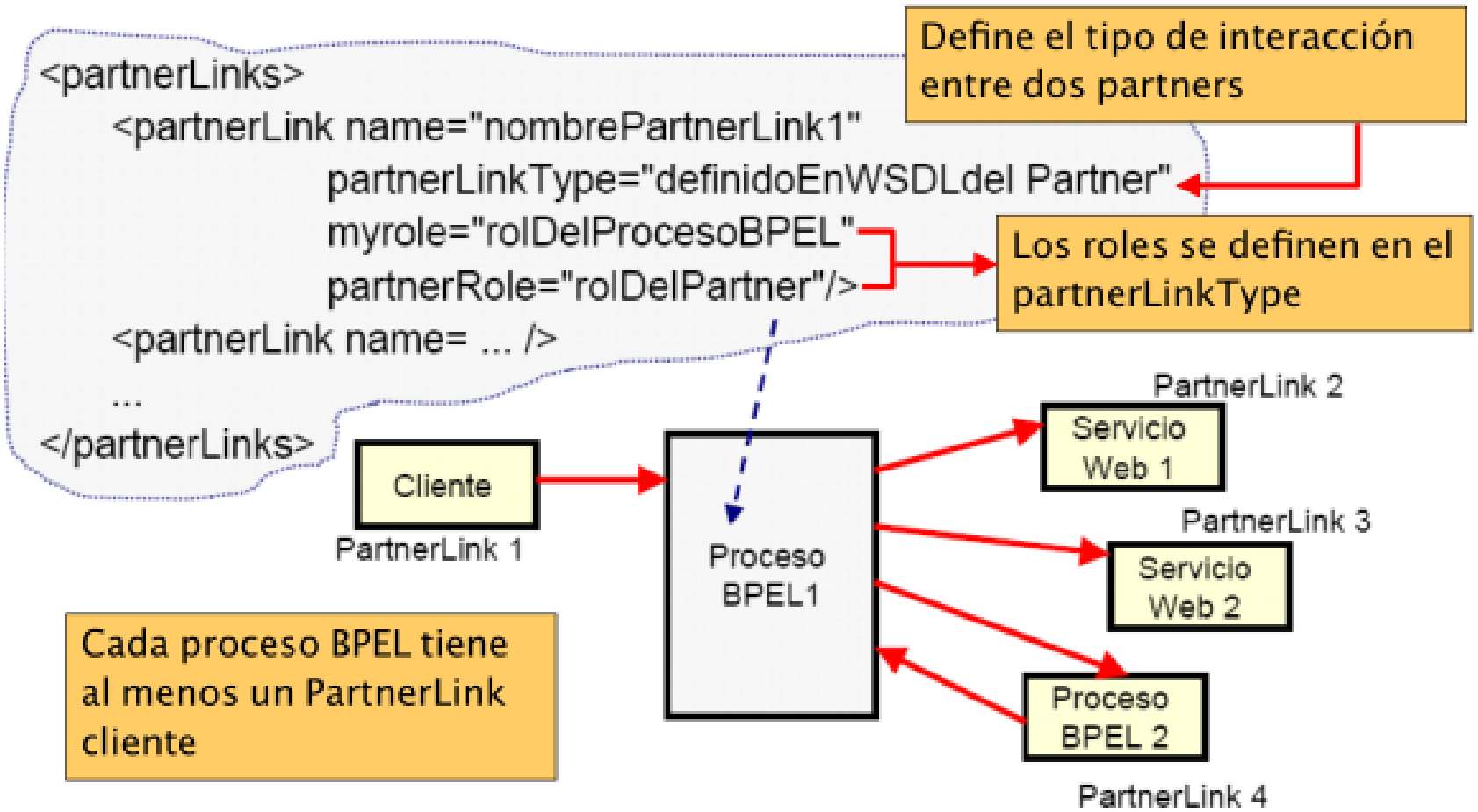
# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:
  - **`myRole`** indica el rol del proceso BPEL. Si solo definimos este atributo entonces cualquier partner puede interaccionar con el proceso BPEL sin ningún requerimiento adcional.
  - **`partnerRole`** indica el rol del partner. Si solo definimos este atributo estamos permitiendo la interacción con un partner que no imponga requerimientos sobre el proceso que haga la llamada.
- Para una operación síncrona utilizaremos un solo rol.
- Para operaciones asíncronas utilizaremos los dos roles. El rol del partener cambia cuando se realiza un `callback`.

# Sintaxis PartnerLink

- Cada `partnerLink` es definido por un `partnerLinkType` y un nombre de rol o dos:
  - **`myRole`** indica el rol del proceso BPEL. Si solo definimos este atributo entonces cualquier partner puede interaccionar con el proceso BPEL sin ningún requerimiento adcional.
  - **`partnerRole`** indica el rol del partner. Si solo definimos este atributo estamos permitiendo la interacción con un partner que no imponga requerimientos sobre el proceso que haga la llamada.
- Para una operación síncrona utilizaremos un solo rol.
- Para operaciones asíncronas utilizaremos los dos roles. El rol del partener cambia cuando se realiza un `callback`.
- Estos roles se definen en el WSDL de cada partner cuando se especifican los `partnerLinkTypes`.

# Sintaxis de PartnerLinks



# PartnerLinkTypes

- Especifican la relación entre dos servicios definiendo el rol que cada servicio implementa (cómo interaccionan y que ofrecen).

# PartnerLinkTypes

- Especifican la relación entre dos servicios definiendo el rol que cada servicio implementa (cómo interaccionan y que ofrecen).
- Los nombres de los roles son cadenas arbitrarias.

# PartnerLinkTypes

- Especifican la relación entre dos servicios definiendo el rol que cada servicio implementa (cómo interaccionan y que ofrecen).
- Los nombres de los roles son cadenas arbitrarias.
- Cada rol especifica exactamente un `porType` WSDL que debe ser implementado por el servicio correspondiente.

# PartnerLinkTypes

- Especifican la relación entre dos servicios definiendo el rol que cada servicio implementa (cómo interaccionan y que ofrecen).
- Los nombres de los roles son cadenas arbitrarias.
- Cada rol especifica exactamente un `porType` WSDL que debe ser implementado por el servicio correspondiente.
- Los **`partnerLinks`** se definen en el proceso BPEL mientras que los **`partnerLinkTypes`** y los **roles** se definen en los ficheros WSDL.

# partnerLink y partnerLinkType

- El tipo `MyPartnerLinkType` **definido en el WSDL** tiene el rol `ProveedorServicioSaludo`, el servicio que implemente este rol deberá implementar `SaludoPortType`. Describe la relación entre el cliente del proceso BPEL y el propio proceso BPEL.

```
<!-- Extracto de Saludo.wsdl -->
<partnerLinkType name="MyPartnerLinkType">
  <role name="ProveedorServicioSaludo"
      portType="SaludoPortType"/>
  </role>
</partnerLinkType>
```

# partnerLink y partnerLinkType

- El tipo `MyPartnerLinkType` **definido en el WSDL** tiene el rol `ProveedorServicioSaludo`, el servicio que implemente este rol deberá implementar `SaludoPortType`. Describe la relación entre el cliente del proceso BPEL y el propio proceso BPEL.

```
<!-- Extracto de Saludo.wsdl -->
<partnerLinkType name="MyPartnerLinkType">
  <role name="ProveedorServicioSaludo"
      portType="SaludoPortType"/>
  </role>
</partnerLinkType>
```

- El `partnerLink` **definido en el proceso BPEL**:

```
<!-- Extracto de Saludo.bpel -->
<partnerLinks>
  <partnerLink name="cliente"
    partnerLinkType="MyPartnerLinkType"
    myRole="ProveedorServicioSaludo"/>
</partnerLinks>
```

# Contenido I

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.
- También podemos almacenar datos que representen el estado del proceso y no se envíen a los partners.

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.
- También podemos almacenar datos que representen el estado del proceso y no se envíen a los partners.
- Para cada variable tenemos que especificar su tipo que puede ser:

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.
- También podemos almacenar datos que representen el estado del proceso y no se envíen a los partners.
- Para cada variable tenemos que especificar su tipo que puede ser:
  - tipo mensaje WSDL: indicado mediante el atributo `messageType`

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.
- También podemos almacenar datos que representen el estado del proceso y no se envíen a los partners.
- Para cada variable tenemos que especificar su tipo que puede ser:
  - tipo mensaje WSDL: indicado mediante el atributo `messageType`
  - tipo simple de un esquema XML: indicado mediante el atributo `type`.

# Variables

- Se utilizan para almacenar, reformatear y transformar mensajes que tienen el estado del proceso.
- Necesitamos una variable por cada mensaje que enviemos o recibamos de los partners.
- También podemos almacenar datos que representen el estado del proceso y no se envíen a los partners.
- Para cada variable tenemos que especificar su tipo que puede ser:
  - tipo mensaje WSDL: indicado mediante el atributo `messageType`
  - tipo simple de un esquema XML: indicado mediante el atributo `type`.
  - tipo elemento de esquema XML: indicado mediante el atributo `element`.

# Sintaxis declaración variables

- Los atributos `messageType`, `type`, `element`, son excluyentes solo puede utilizar **uno** de los tres.

```
<variables>
  <variable name="nombreVar"
    messageType="qname"
    type="qname"
    element="qname"/>
</variables>
```

# Sintaxis declaración variables

- Los atributos `messageType`, `type`, `element`, son excluyentes solo puede utilizar **uno** de los tres.

```
<variables>
  <variable name="nombreVar"
    messageType="qname"
    type="qname"
    element="qname"/>
</variables>
```

- Ejemplos: `messageType` y `element` se declaran en el WSDL

```
<variables>
  <variable name="PeticionArticulo"
    messageType="art:MensajePeticionArticulo"/>
  <variable name="DescripcionParcialArticulo"
    element="art:DescripcionArticulo"/>
  <variable name="Telefono"
    type="xs:string"/>
</variables>
```

# Sintaxis de asignación de variables

- Copiar los datos de una variable a otra.

```
<assign>
  <copy>
    <from variable="ncname" part="ncname"/>
    <to variable="ncname" part="ncname"/>
  </copy>
</assign>
```

# Sintaxis de asignación de variables

- Copiar los datos de una variable a otra.

```
<assign>
  <copy>
    <from variable="ncname" part="ncname"/>
    <to variable="ncname" part="ncname"/>
  </copy>
</assign>
```

- Se le asigna la parte del mensaje de una variable de una a otra.

# Contenido I

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones
  - **Estructuradas** permiten combinar actividades primitivas.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones
  - **Estructuradas** permiten combinar actividades primitivas.
    - `<sequence>` define el orden de invocación de un conjunto de actividades.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones
  - **Estructuradas** permiten combinar actividades primitivas.
    - `<sequence>` define el orden de invocación de un conjunto de actividades.
    - `<flow>` conjunto de actividades que se invocarán en paralelo.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones
  - **Estructuradas** permiten combinar actividades primitivas.
    - `<sequence>` define el orden de invocación de un conjunto de actividades.
    - `<flow>` conjunto de actividades que se invocarán en paralelo.
    - `<while>` define bucles.

# Actividades

- Un proceso BPEL está formado por una serie de pasos (actividades).
- Dos tipos de actividades:
  - **Primitivas** representan construcciones básicas.
    - `<receive>` bloque el proceso hasta recibir un mensaje.
    - `<reply>` envía un mensaje como respuesta al `receive`
    - `<invoque>` invoca un WS (síncrona o asíncrona)
    - `<assign>` asigna un valor a una variable
    - `<wait>` suspende el proceso por un tiempo
    - `<throw>` indica fallos y excepciones
  - **Estructuradas** permiten combinar actividades primitivas.
    - `<sequence>` define el orden de invocación de un conjunto de actividades.
    - `<flow>` conjunto de actividades que se invocarán en paralelo.
    - `<while>` define bucles.
    - `<pick>` el proceso espera la llegada de un mensaje (`<onMessage>`) o evento (`<onAlarm>`) y realiza un conjunto de actividades dependiendo del evento.

# <receive>

- Especifica un `partnerLink`, un `portType` y una operación a ser invocada.

# <receive>

- Especifica un `partnerLink`, un `portType` y una operación a ser invocada.
- `createInstance="yes"` para iniciar el proceso.

```
<receive partnerLink="ncname"
  portType="qname"
  operation="ncname"
  variable="ncname"
  createInstance="yes|no">
```

# <reply>

- La combinación de una <receive> y una <reply> crea una interacción síncrona (**request-response**). Deben especificar el mismo partnerLink, portType y operación.

# <reply>

- La combinación de una <receive> y una <reply> crea una interacción síncrona (**request-response**). Deben especificar el mismo partnerLink, portType y operación.
- Puede transmitir datos por medio de una variable.

```
<reply partnerLink="ncname"
  portType="qname"
  operation="ncname"
  variable="ncname">
</reply>
```

# <invoque>

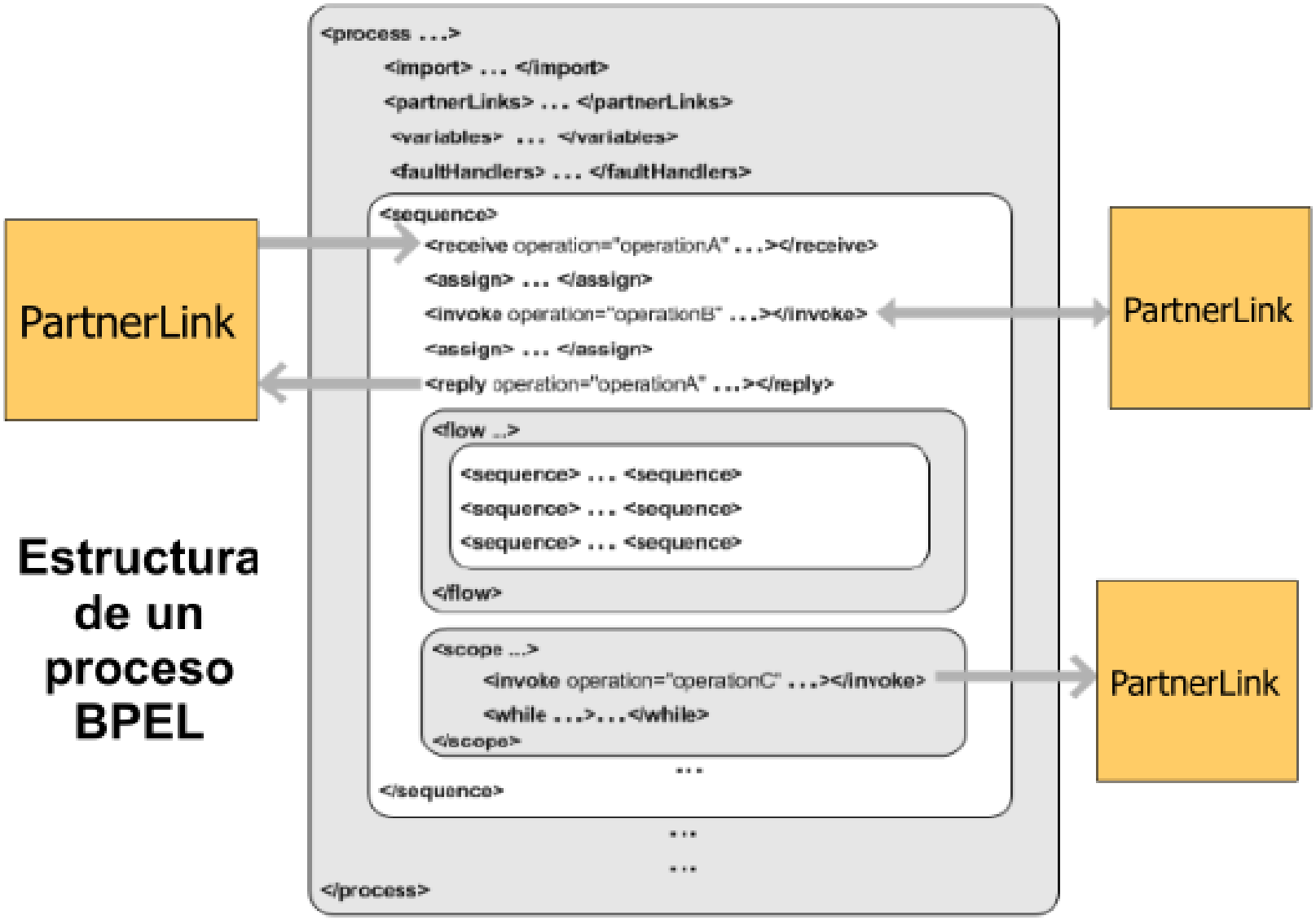
- Permite invocar a un WS que haya sido definido como partner.

# <invoque>

- Permite invocar a un WS que haya sido definido como partner.
- Puede ser asíncrona (especifica solo una variable de entrada) o asíncrona (especifica variable de entrada y salida).

```
<invoque partnerLink="ncname"
  portType="qname"
  operation="ncname"
  inputVariable="ncname"
  outputVariable="ncname">
</invoque>
```

# Estructura de un proceso BPEL



# Contenido I

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL
   - Definir los `partnerLinkTypes` del cliente.

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL
   - Definir los `partnerLinkTypes` del cliente.
3. Desarrollar el proceso BPEL

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL
   - Definir los `partnerLinkTypes` del cliente.
3. Desarrollar el proceso BPEL
   - Definir los `partnerLinks`

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL
   - Definir los `partnerLinkTypes` del cliente.
3. Desarrollar el proceso BPEL
   - Definir los `partnerLinks`
   - Declarar las variables

# Pasos para desarrollar un proceso de negocio con BPEL

1. Conocer los WS implicados
   - Familiarizarnos con los `portTypes` de los WS partners
2. Definir el WSDL del proceso BPEL
   - Definir los `partnerLinkTypes` del cliente.
3. Desarrollar el proceso BPEL
   - Definir los `partnerLinks`
   - Declarar las variables
   - Escribir la definición de la lógica del proceso

# Contenido I

# Contenido I

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.
- Utilizaremos Open-ESB (basado en Netbeans) para desarrollar y desplegar procesos BPEL.

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.
- Utilizaremos Open-ESB (basado en Netbeans) para desarrollar y desplegar procesos BPEL.
- Podemos descargar Open-ESB desde `http://www.open-esb.net/`

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.
- Utilizaremos Open-ESB (basado en Netbeans) para desarrollar y desplegar procesos BPEL.
- Podemos descargar Open-ESB desde `http://www.open-esb.net/`
- Open-ESB utilizar el entorno de ejecución (JBI: Java Business Integration) que incluye varios componentes que interactúan utilizando un modelos de servicios.

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.
- Utilizaremos Open-ESB (basado en Netbeans) para desarrollar y desplegar procesos BPEL.
- Podemos descargar Open-ESB desde `http://www.open-esb.net/`
- Open-ESB utilizar el entorno de ejecución (JBI: Java Business Integration) que incluye varios componentes que interactúan utilizando un modelos de servicios.
- Uno de estos componentes es la máquina de servicios BPEL (BPEL Service Engine) que proporciona servicios para ejecutar los procesos de negocio.

# Despliegue y pruebas del proceso BPEL

- Una vez desarrollado el proceso BPEL debemos desplegarlo en un **BPEL engine.**
- Y finalmente realizar pruebas.
- Utilizaremos Open-ESB (basado en Netbeans) para desarrollar y desplegar procesos BPEL.
- Podemos descargar Open-ESB desde `http://www.open-esb.net/`
- Open-ESB utilizar el entorno de ejecución (JBI: Java Business Integration) que incluye varios componentes que interactúan utilizando un modelos de servicios.
- Uno de estos componentes es la máquina de servicios BPEL (BPEL Service Engine) que proporciona servicios para ejecutar los procesos de negocio.
- Los Binding components son los componenetes que proporcionan acceso a servicios externos al entorno JBI.

# Componentes JBI

- Se instalan formando parte del servidor de aplicaciones Glassfish (se debe contar con la versión de Glassfish que incluye JBI).

# Componentes JBI

- Se instalan formando parte del servidor de aplicaciones Glassfish (se debe contar con la versión de Glassfish que incluye JBI).
- Open-ESB está formado por componente que dan soporte al diseño y que se ejecutan dentro de Netbeans y componentes que dan soporte a la ejecución que se ejecutan en el servidor de aplicaciones Glassfish.

# Componentes JBI

- Se instalan formando parte del servidor de aplicaciones Glassfish (se debe contar con la versión de Glassfish que incluye JBI).
- Open-ESB está formado por componente que dan soporte al diseño y que se ejecutan dentro de Netbeans y componentes que dan soporte a la ejecución que se ejecutan en el servidor de aplicaciones Glassfish.
- Para ver los componentes JBI instalados vamos a la pestaña Services -> Servers -> Glassfish v2.x -> JBI. El servidor debe estar corriendo.

# Contenido I

# BPEL Service Engine

- Componente JBI que proporciona servicios para ejecutar procesos de negocio desarrollados con BPEL.

# BPEL Service Engine

- Componente JBI que proporciona servicios para ejecutar procesos de negocio desarrollados con BPEL.
- Para desplegar un proceso BPEL debemos añadirlos como un módulo JBI a un proyecto de composición de aplicaciones (composite application).

# BPEL Service Engine

- Componente JBI que proporciona servicios para ejecutar procesos de negocio desarrollados con BPEL.
- Para desplegar un proceso BPEL debemos añadirlos como un módulo JBI a un proyecto de composición de aplicaciones (composite application).
- Arranca junto con el servidor de aplicaciones (Glassfish). `sun-bpel-engine`.

# Contenido I

# Proyecto de aplicaciones compuestas

- `Composite Application` son aplicaciones que en lugar de ser desarrolladas desde cero se ensamblan a partir de servicios disponibles en una arquitectura SOA.

# Proyecto de aplicaciones compuestas

- `Composite Application` son aplicaciones que en lugar de ser desarrolladas desde cero se ensamblan a partir de servicios disponibles en una arquitectura SOA.
- Un proyecto de aplicaciones compuestas en (Open-ESB Netbeans) se utiliza para crear un ensamblado de servicio (`Service Assembly`) que puede desplegarse en el servidor de aplicaciones como un componente JBI.

# Proyecto de aplicaciones compuestas

- `Composite Application` son aplicaciones que en lugar de ser desarrolladas desde cero se ensamblan a partir de servicios disponibles en una arquitectura SOA.
- Un proyecto de aplicaciones compuestas en (Open-ESB Netbeans) se utiliza para crear un ensamblado de servicio (`Service Assembly`) que puede desplegarse en el servidor de aplicaciones como un componente JBI.
- No podemos desplegar directamente un proyecto BPEL. Primero debemos añadir dicho proyecto BPEL como módulo JBI a un proyecto `Composite Application`.

# Proyecto de aplicaciones compuestas

- `Composite Application` son aplicaciones que en lugar de ser desarrolladas desde cero se ensamblan a partir de servicios disponibles en una arquitectura SOA.
- Un proyecto de aplicaciones compuestas en (Open-ESB Netbeans) se utiliza para crear un ensamblado de servicio (`Service Assembly`) que puede desplegarse en el servidor de aplicaciones como un componente JBI.
- No podemos desplegar directamente un proyecto BPEL. Primero debemos añadir dicho proyecto BPEL como módulo JBI a un proyecto `Composite Application`.
- Al hacer clic derecho sobre el proyecto `Composite application` y clic en `Deploy`, se compilan los ficheros del proyecto, empaqueta el proceso BPEL compilado y los artefactos de los WS relacionados (incluyendo WSDL Y XSD) y se despliega en el servidor de aplicaciones.

# Contenido I

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.
- Para crear un caso de prueba:

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.
- Para crear un caso de prueba:
  - Añadir un caso de prueba y enlazarlo con una operación BPEL.

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.
- Para crear un caso de prueba:
  - Añadir un caso de prueba y enlazarlo con una operación BPEL.
  - Determinar las propiedades de la prueba (como por ejemplo el número de segundos que debe durar como máximo la ejecución de la prueba).

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.
- Para crear un caso de prueba:
  - Añadir un caso de prueba y enlazarlo con una operación BPEL.
  - Determinar las propiedades de la prueba (como por ejemplo el número de segundos que debe durar como máximo la ejecución de la prueba).
  - Modificar convenientemente las entradas de las pruebas, para ello editaremos el fichero Input.xml según nos interese.

# Creación y ejecución de casos de prueba

- Podemos crear casos de prueba para el proceso BPEL en proyectos composite application (previamente desplegado).
- Los casos de prueba actúan como servicios partner remotos que envían mensajes SOAP al runtime de la máquina de servicios BPEL.
- Para crear un caso de prueba:
  - Añadir un caso de prueba y enlazarlo con una operación BPEL.
  - Determinar las propiedades de la prueba (como por ejemplo el número de segundos que debe durar como máximo la ejecución de la prueba).
  - Modificar convenientemente las entradas de las pruebas, para ello editaremos el fichero Input.xml según nos interese.
  - Ejecutar las pruebas. El resultado de la ejecución se comparará con el resultado esperado almacenado en el fichero Output.xml. Si ambos resultados son diferentes, se habrá detectado un error.

# Contenido I

# Deber

- Realizar el ejercicio que está en el repositorio Ejercicio.pdf de la sesion14. Fecha limite 16 de julio 2015

# Deber

- Realizar el ejercicio que está en el repositorio Ejercicio.pdf de la sesion14. Fecha limite 16 de julio 2015
- Leer y realizar un resumen sobre el artículo “La Arquitectura Orientada a Servicios (SOA) de Microsoft”. Máximo 10 hojas. Fecha: 16 julio 2015. Entregar vía Turnitin.

# Deber

- Realizar el ejercicio que está en el repositorio Ejercicio.pdf de la sesion14. Fecha limite 16 de julio 2015
- Leer y realizar un resumen sobre el artículo "La Arquitectura Orientada a Servicios (SOA) de Microsoft". Máximo 10 hojas. Fecha: 16 julio 2015. Entregar vía Turnitin.
- Consultar y realizar un informe sobre el Gobierno de SOA (SOA Governance) y los modelos de madurez de SOA. Máximo 4 hojas. Fecha: 16 julio 2015. Entregar vía Turnitin.

# A tener en cuenta para el ejercicio

- La primera vez que ejecutemos las pruebas obtendremos como resultado la indicación de que la prueba ha fallado (se ha detectado un error). La salida producida no coincidirá con el resultado almacenado en el fichero Output.xml (que inicialmente estará vacío). Al ejecutar la prueba la primera vez, el contenido de Output.xml será reemplazado por la salida generada en esta primera ejecución.

# A tener en cuenta para el ejercicio

- La primera vez que ejecutemos las pruebas obtendremos como resultado la indicación de que la prueba ha fallado (se ha detectado un error). La salida producida no coincidirá con el resultado almacenado en el fichero Output.xml (que inicialmente estará vacío). Al ejecutar la prueba la primera vez, el contenido de Output.xml será reemplazado por la salida generada en esta primera ejecución.
- Si ejecutamos la prueba de nuevo sin cambiar la entrada, las ejecuciones siguientes informarán del éxito de la prueba, ya que la salida será idéntica al contenido de Output.xml.

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:
  - Si la propiedad feature-status tiene como valor asignado progress, entonces la prueba indica éxito incluso aunque no coincida el resultado con Output.xml.

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:
  - Si la propiedad feature-status tiene como valor asignado progress, entonces la prueba indica éxito incluso aunque no coincida el resultado con Output.xml.
  - Si la propiedad feature-status tiene como valor asignado done, entonces la prueba indica fallo si no coincide el resultado y guarda el resultado obtenido.

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:
  - Si la propiedad feature-status tiene como valor asignado progress, entonces la prueba indica éxito incluso aunque no coincida el resultado con Output.xml.
  - Si la propiedad feature-status tiene como valor asignado done, entonces la prueba indica fallo si no coincide el resultado y guarda el resultado obtenido.
  - Si pinchamos con el botón derecho del ratón sobre el caso de prueba y seleccionamos Diff sobre el menú emergente, se muestra la diferencia entre la última salida y los contenidos de Output.xml.

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:
  - Si la propiedad feature-status tiene como valor asignado progress, entonces la prueba indica éxito incluso aunque no coincida el resultado con Output.xml.
  - Si la propiedad feature-status tiene como valor asignado done, entonces la prueba indica fallo si no coincide el resultado y guarda el resultado obtenido.
  - Si pinchamos con el botón derecho del ratón sobre el caso de prueba y seleccionamos Diff sobre el menú emergente, se muestra la diferencia entre la última salida y los contenidos de Output.xml.
  - En cada ejecución posterior a la primera, y asumiendo que Output.xml ya no está nunca vacío, se preserva su contenido. Es decir, una salida previa, nunca se sobreescribe con nuevos resultados (con la excepción ya comentada de la primera ejecución).

# A tener en cuenta

- Si cambiamos el valor de la entrada en Input.xml y volvemos a ejecutar la prueba, entonces:
  - Si la propiedad feature-status tiene como valor asignado progress, entonces la prueba indica éxito incluso aunque no coincida el resultado con Output.xml.
  - Si la propiedad feature-status tiene como valor asignado done, entonces la prueba indica fallo si no coincide el resultado y guarda el resultado obtenido.
  - Si pinchamos con el botón derecho del ratón sobre el caso de prueba y seleccionamos Diff sobre el menú emergente, se muestra la diferencia entre la última salida y los contenidos de Output.xml.
  - En cada ejecución posterior a la primera, y asumiendo que Output.xml ya no está nunca vacío, se preserva su contenido. Es decir, una salida previa, nunca se sobreescribe con nuevos resultados (con la excepción ya comentada de la primera ejecución).
  - Para ver los resultados de pruebas anteriores, podemos elegir de la lista desplegable el fichero Actual_yymmddhhmmss.xml correspondiente, y pulsar en el botón Refresh.