

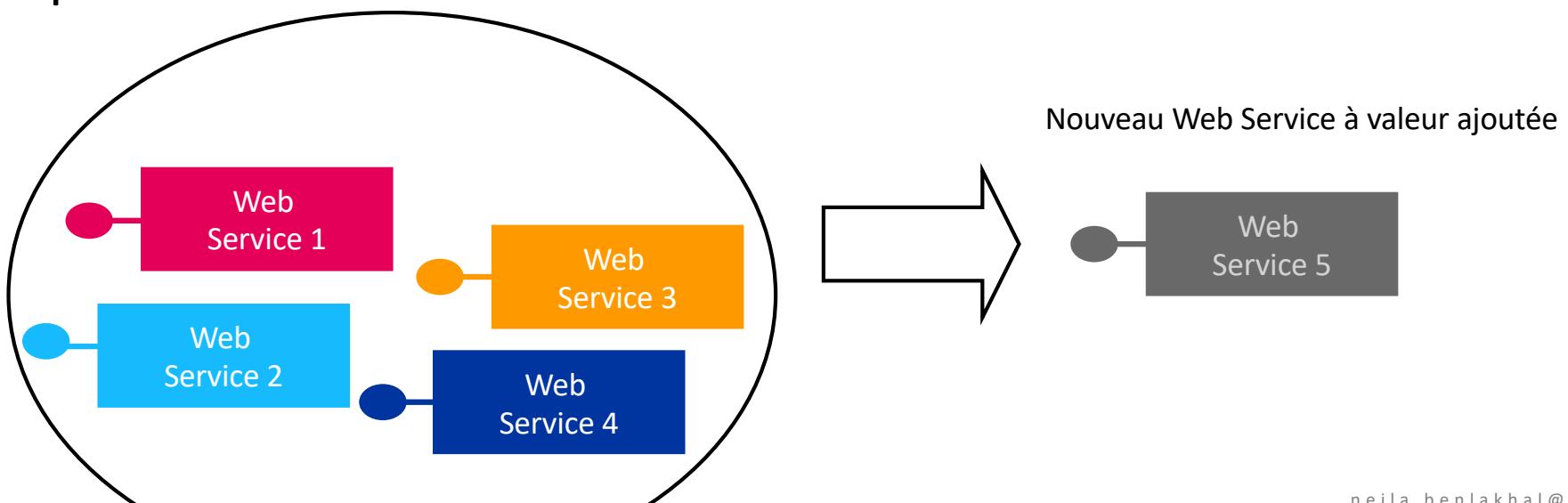
Composition de Web services avec BPEL

Préparé par:

**Dr.Neila Ben Lakhal (@Titech Japan)
PhD from Tokyo Institute of Technology
Assistant professor @ Enicartha**

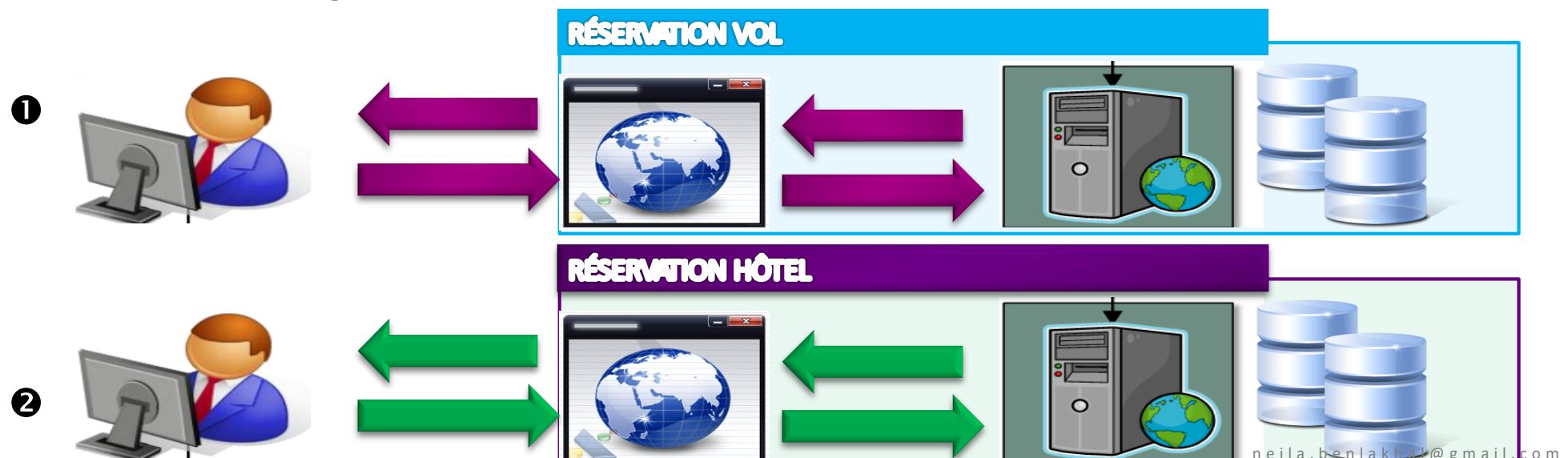
Tout d'abord, Composition de web services, c'est quoi ?

- ▶ Composer des Web Services élémentaires pour aboutir à un nouveau Web Service à valeur ajoutée selon la logique d'un processus métier



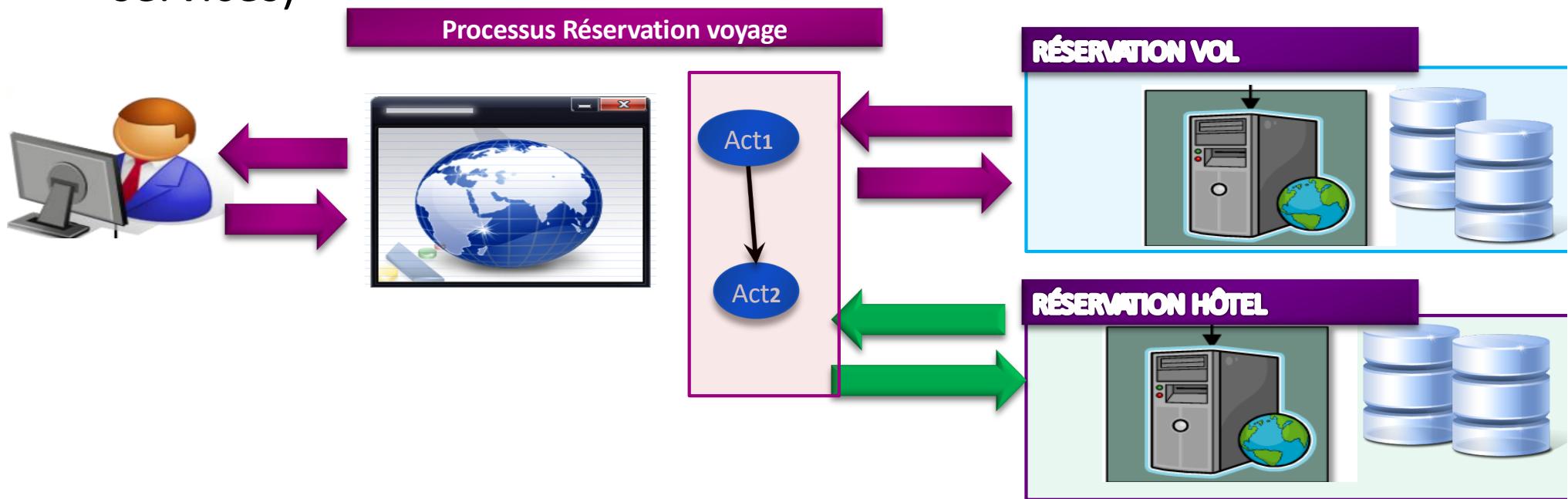
Composition de Web services, pourquoi faire ? 1/2

- ▶ Prenons un scénario : pour réserver tout un package pour un client doit se rendre sur des sites différents et fournir les mêmes informations 😞



Composition de Web services, pourquoi faire ? 2/2

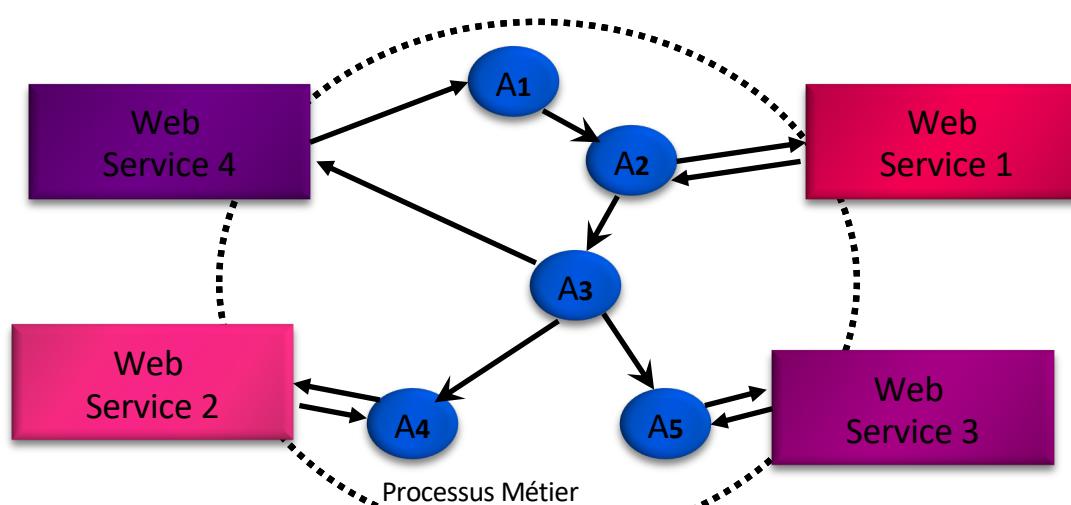
- ▶ Une meilleure solution : le client accède à un seul processus de réservation qui orchestre les différentes activités (des Web services)



neila.benlakhal@gmail.com

Composition de web services

- ▶ Faire intervenir plusieurs web services et les mapper aux différentes activités du processus métier :



Web services composition ?

“Service composition means the creation of new services from combining existing services. Services need to be composable so that smaller services can be combined into larger services that provide a specific value”

[Extrait de SOA made simple]

Web services composition :

- ▶ Ma définition :

...More recently , one issue that is gaining notable momentum by the research community , is WS composition (WSC):

to create what is called value-added services or composed Web services (CWS) by taking a set of preexistent elementary WS, typically owned and managed by diverse entities, and weaving them together, to build more powerful and feature-rich business processes. An example of CWS is one that books a flight, rents a car, and makes a hotel reservation to provide a complete trip reservation process.

Deux types de compositions : Aggregation versus Orchestration

- ▶ Sometimes the terms service composition, service aggregation, and service orchestration are used interchangeably. While service composition denotes the fact that services are combined, it does not elaborate on the way in which services are combined.
- ▶ Aggregation and orchestration are specific ways to combine services.

[Extrait de SOA made simple]

Deux types de compositions :

Aggregation versus Orchestration

- ▶ **Aggregation (also called choreography)** : combining two or more services into larger ones is fairly straightforward (for example invoke the service operation A, then service operation B, then service operation C).
 - ▶ Service aggregation involves short-running services.
- ▶ **Orchestration**: Combining several services into a flow or process that is more complex in nature and contains decision logic and different execution paths. Orchestration involves a central component such as a BPM platform that manages the orchestration of services.
 - ▶ Service orchestration involves long(er)-running services.

Où effectuer la composition ?

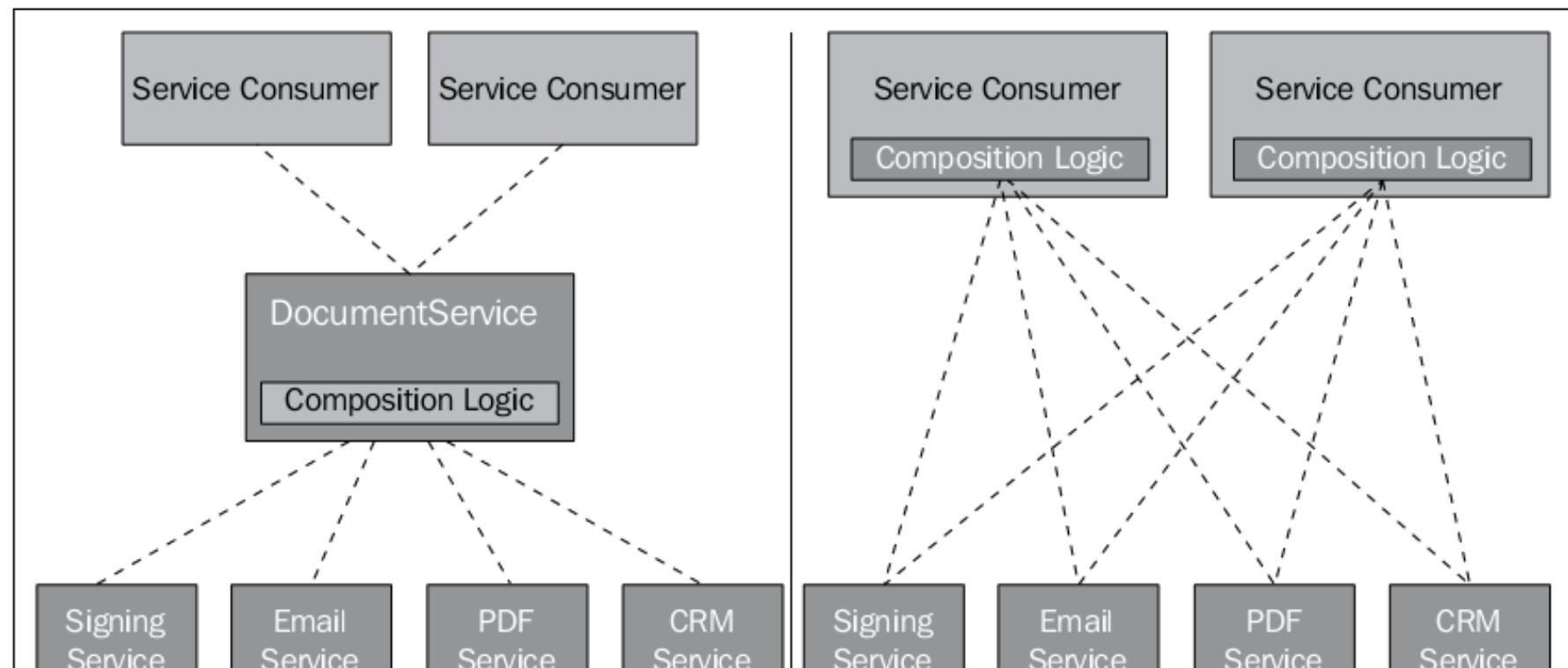
- ▶ Le composition peut se faire de deux manières :
 - ▶ **Au niveau du service consumer** : le consommateur de service invoque plusieurs opérations provenant de différents web services une à une pour aboutir à un but final.
 - ▶ **Au niveau des services** : un nouveau service composite est créé; il se charge d'invoquer les différentes opérations des différents web services de façon transparente, et expose cette fonctionnalité en tant que nouvelle opération.

[Extrait de SOA made simple]

Où effectuer la composition ?

Exemple :

[Extrait de SOA made simple]



Composition Frameworks

*“Up to now, a myriad of specifications for composing WS has been made available exemplified by the emerging standards such as **BPEL** (Business Process Execution) [4] and the industrial solutions such as IBM’s Emerging Technologies Toolkit **ETTK** [5] and Microsoft’s .Net [6]. In addition, a substantial amount of research efforts is underway by academic researchers who have been working on a whole panoply of WSC strategies including mainly, **dynamic composition** (e.g., eFlow [7,8]), **declarative composition** (e.g., SELF-SER V [9, 10]), and **semantic composition** (e.g., SHOP2 [11]). A careful investigation of major part of the available solutions for WSC reveals that only very few cases were geared toward a distributed environment like the SELF-SER V framework. Yet all the other approaches like BPEL4WS and eFlow, they only support the integration of WS in a centralized model consisting of dedicated centralized engine(s). They have totally ignored the nature of the WS environment where interaction follows a peer-to-peer model and where each peer WS owner provides a set of services that can be composed to CWS .” – Neila @VLDB*

Langages de modélisation de processus métier

- ▶ À ce jours, différents langages permettent de modéliser les processus métiers
- ▶ On distingue :
 - ▶ Les langages visuels/graphiques:
 - ▶ UML, Jopera, WebML, etc.
 - ▶ Les langages textuels : (XML) de composition de WS
 - ▶ 2001 : Xlang, WSFL, ebXML, BPML, etc.
 - ▶ 2002 : WSCL, WSCI, BPEL 1.0 etc.
 - ▶ **2007 : BPEL 2.0**

Qu'est-ce que BPEL?

- ▶ WS-BPEL (ou BPEL en abrégé) est l'acronyme de **Business Process Execution Language**
- ▶ C'est un langage standard **d'orchestration de services Web** intégrés aux processus métiers modélisés.
 - ▶ Repose sur des standards comme WSDL, Schéma XML, XPath, etc.
 - ▶ Se base également sur la notation BPMN et fourni les opérateurs classiques d'un système de workflow (séquence, parallèle, boucle, ...)
- ▶ Syntaxe fortement basée sur XML.

Ancêtres de BPEL

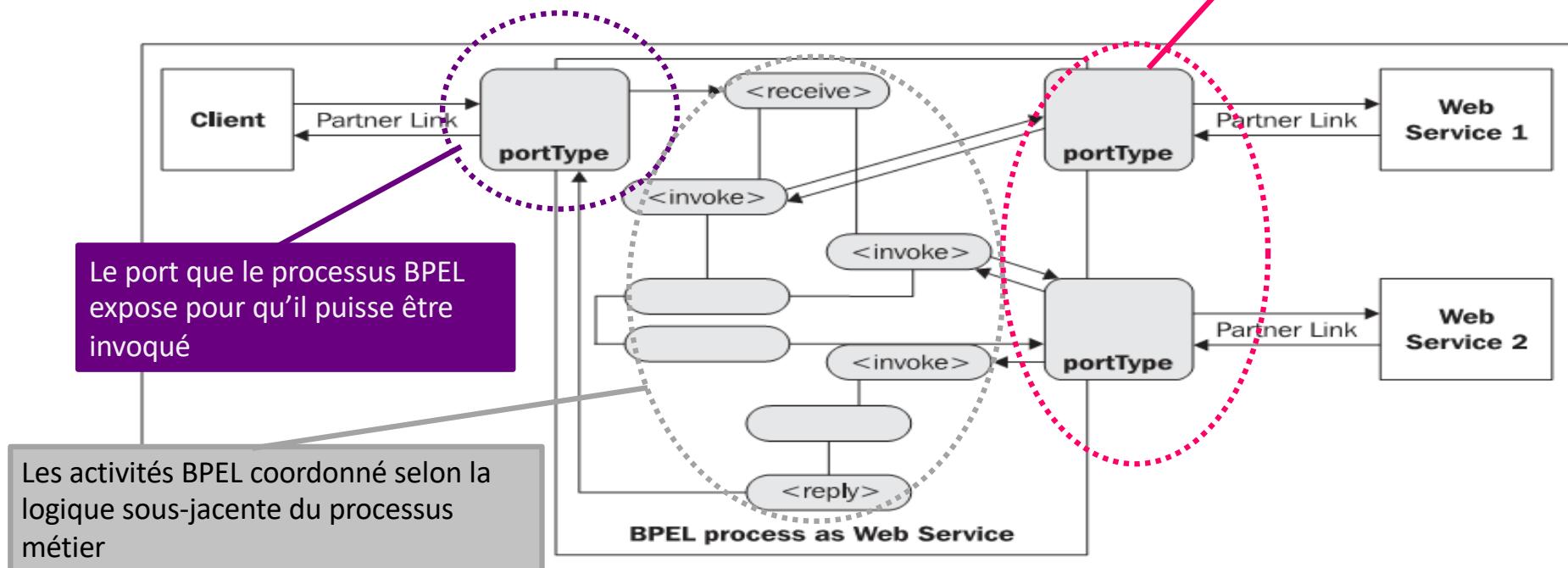
- ▶ WSFL
 - ▶ (Web Services Flow Language), May 2001 (IBM)
- ▶ XLANG
 - ▶ May 2001 (Microsoft)
- ▶ BPEL 1.0
 - ▶ July 2002 (BEA, IBM, Microsoft), ↗ A merger of WSFL and XLANG.
- ▶ BPEL4WS 1.1(Business Process Execution Language for Web Services)
 - ▶ March 2003 (BEA, IBM, Microsoft, SAP, Siebel) :↗ The specification submitted to OASIS
- ▶ WS-BPEL 2.0
 - ▶ March 2007 (OASIS, 39 companies as members of the technical committee)
 - ▶ Un standard !

Qu'est-ce qu'un processus BPEL?

- ▶ Le langage BPEL est un langage de définition de processus métier sous deux formes :
- ▶ Un processus BPEL exécutable (.bpel) qui compose un ensemble de Web services existants déployé pour exécution sur un « **moteur d'exécution BPEL** ».
- ▶ Un processus BPEL abstrait (.wsdl) décrit les activités, les interactions et les échanges de messages entre les services Web définies comme « **partenaires** » du processus BPEL.
- ▶ **Un processus BPEL est lui-même considéré comme un nouveau Web service qui compose d'autres WS : décrit par un (.WSDL).**

Composition de Web services

► Un processus métier selon BPEL :



Pour faire du BPEL : Outils

- ▶ Pour faire du BPEL il nous faut:
 - ▶ Un outil pour créer le processus **exécutable** (.BPEL)
 - ▶ Un outil pour générer la définition **abstraite** (*.WSDL)+(*.xsd)
 - ▶ Un moteur d'exécution pour générer et exécuter les instances.
- ▶ Divers solutions BPEL existantes comportent :
 - ▶ Des éditeurs graphiques de processus BPEL intégrables dans des IDE (Eclipse, NetBeans, JDevelopers, etc.)
 - ▶ Des moteurs BPEL intégrables dans des serveurs d'applications (TOMCAT, Apache ODE, Glassfish, JBOSS, etc.)

Quelques Solutions existantes pour faire du BPEL

- ▶ **OpenESB (via Glassfish 2.1) et Netbeans 6.7.1**
 - ▶ http://www.logicoy.com/download_glass
- ▶ Apache ODE et Eclipse BPEL Editor
 - ▶ <http://ode.apache.org/> et <http://www.eclipse.org/bpel/>
- ▶ Oracle BPEL Process Manager (Weblogic + JDeveloper)
 - ▶ <http://www.oracle.com/technology/bpel>
- ▶ Intalio | BPMS (Designed around the open source Eclipse BPMN Modeler, Apache ODE BPEL engine)
 - ▶ <http://www.intalio.com/downloads>
- ▶ Petals ESB <http://doc.petalslink.com/display/petalsesb/Petals+ESB>
- ▶ Orchestra <http://orchestra.ow2.org/xwiki/bin/view/Main/>
- ▶ IBM (Websphere server)
- ▶ Microsoft BPEL for Windows Workflow Foundation (Visual Studio et Biztalk server)
- ▶ Etc..

Étudier BPEL par l'exemple

- ▶ Pour introduire la présentation du langage BPEL, nous utiliserons l'exemple d'un processus développé sous openESB (open source) :

- ▶ OpenESB v2.2

+

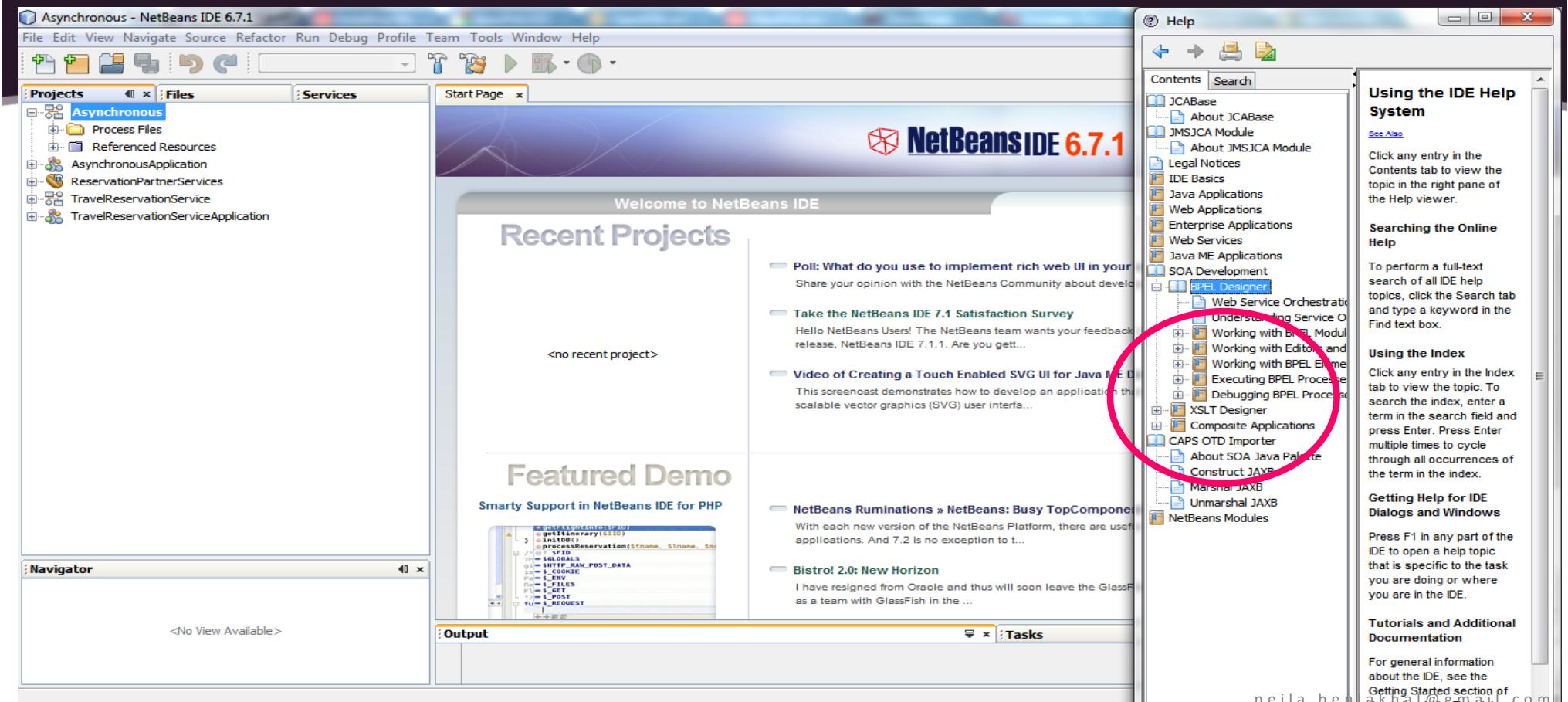
- ▶ Glassfish 2.1

+

- ▶ Netbeans 6.7.1

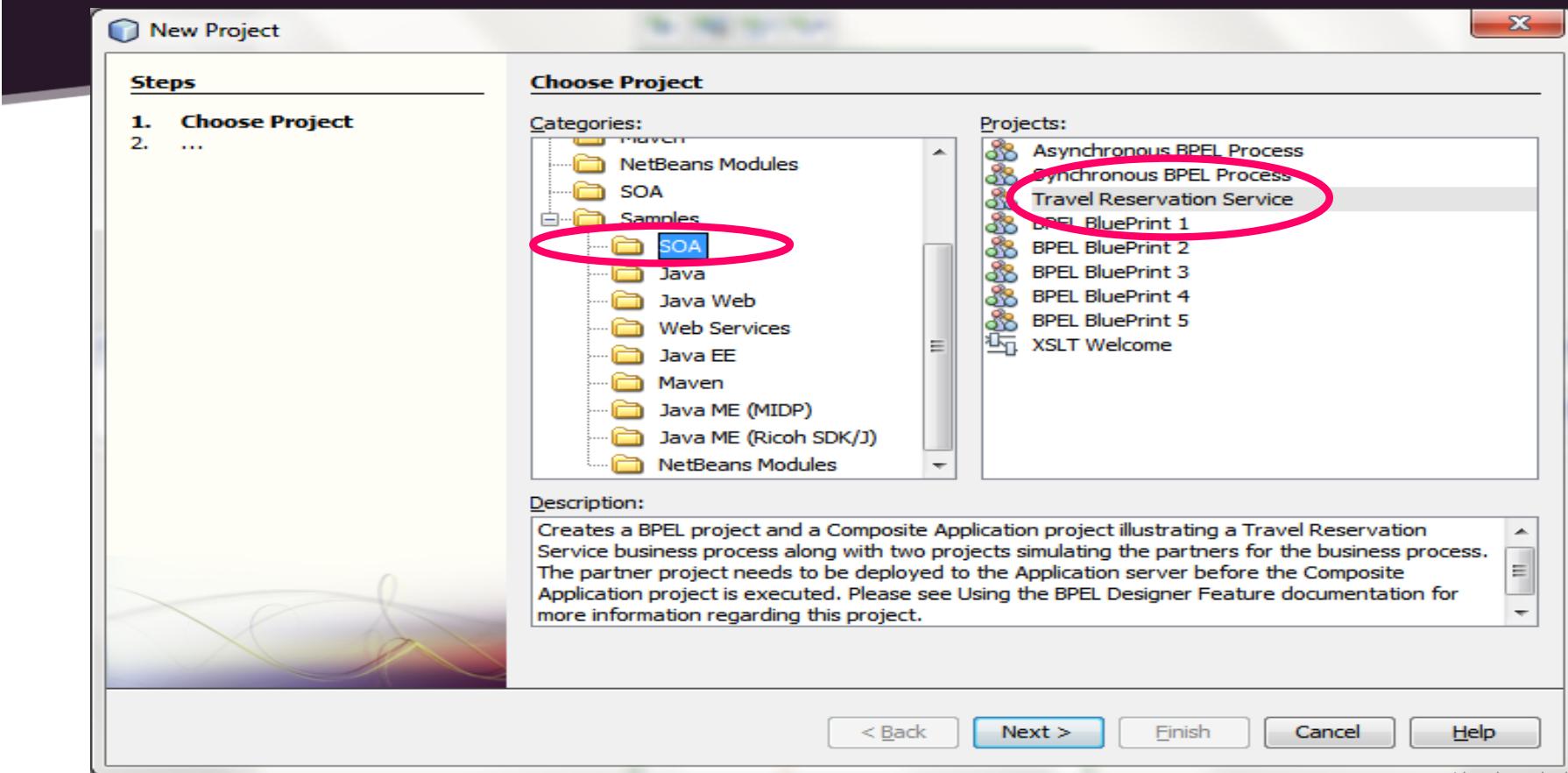


Pour commencer : Ouvrir le volet HELP

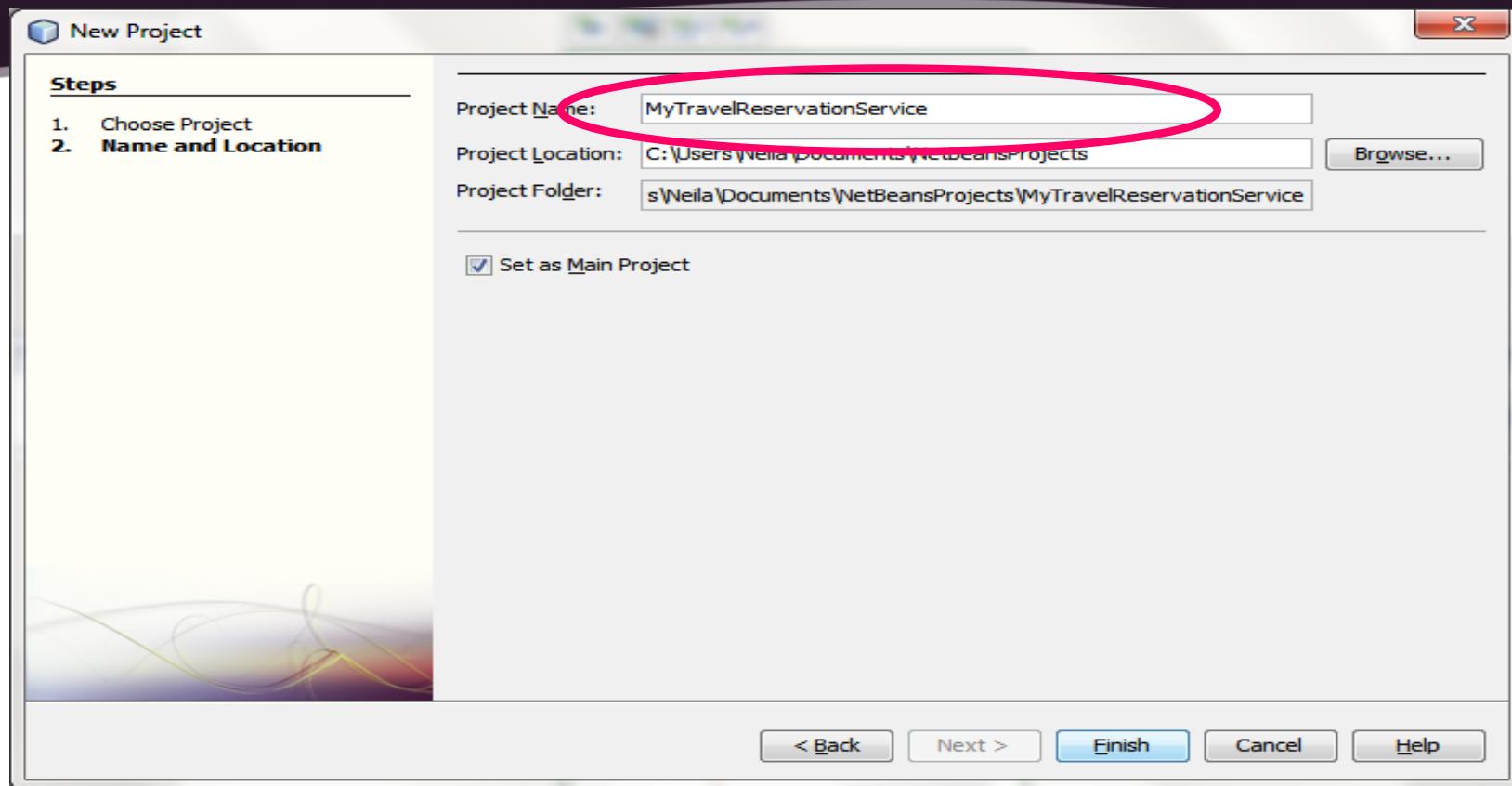


Exemple de processus de réservation d'un voyage

Création d'un nouveau projet BPEL (File -> New Project) à partir des exemples (Samples)

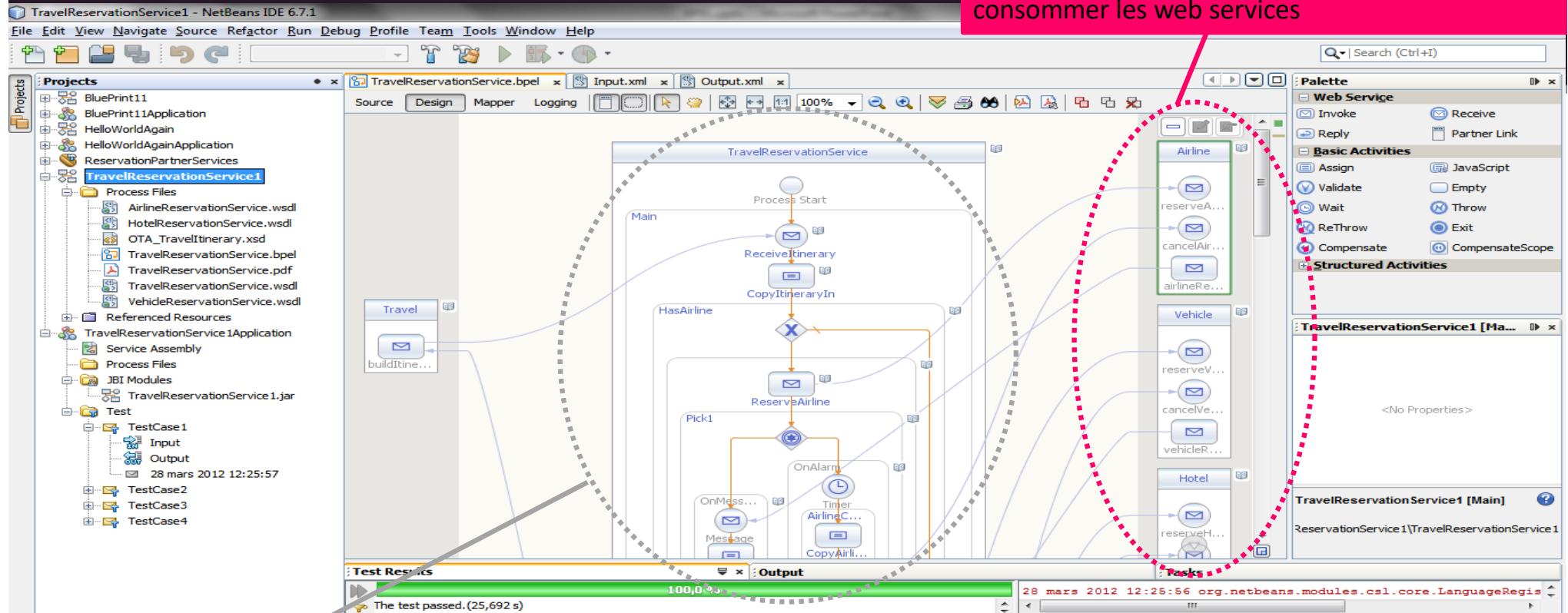


Attribuer un nom à votre projet, et le tour est joué



neila.benlakhal@gmail.com

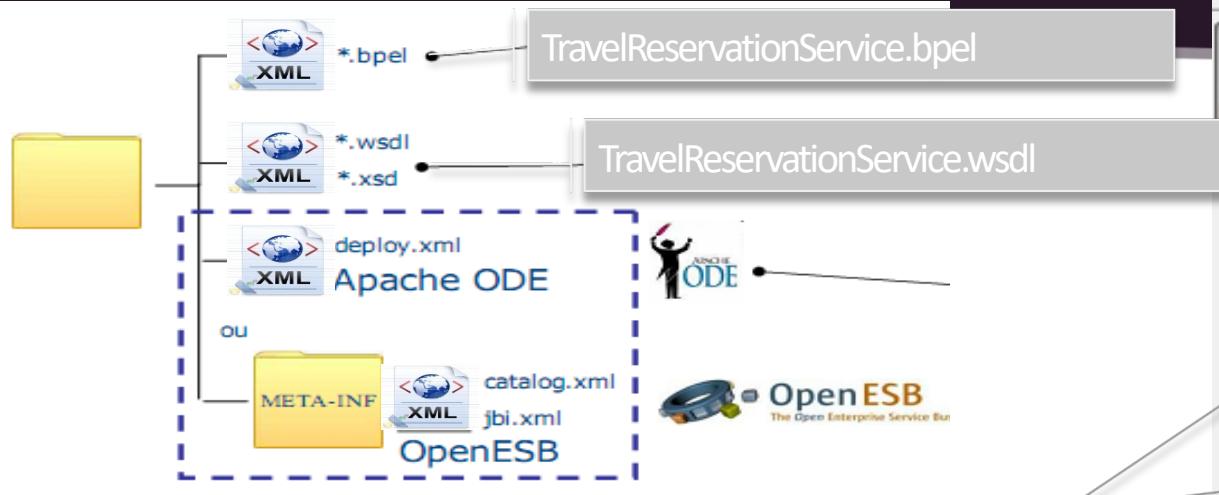
Les ports que le processus BPEL définit pour consommer les web services



Les activités BPEL coordonné selon la logique sous-jacente du processus métier

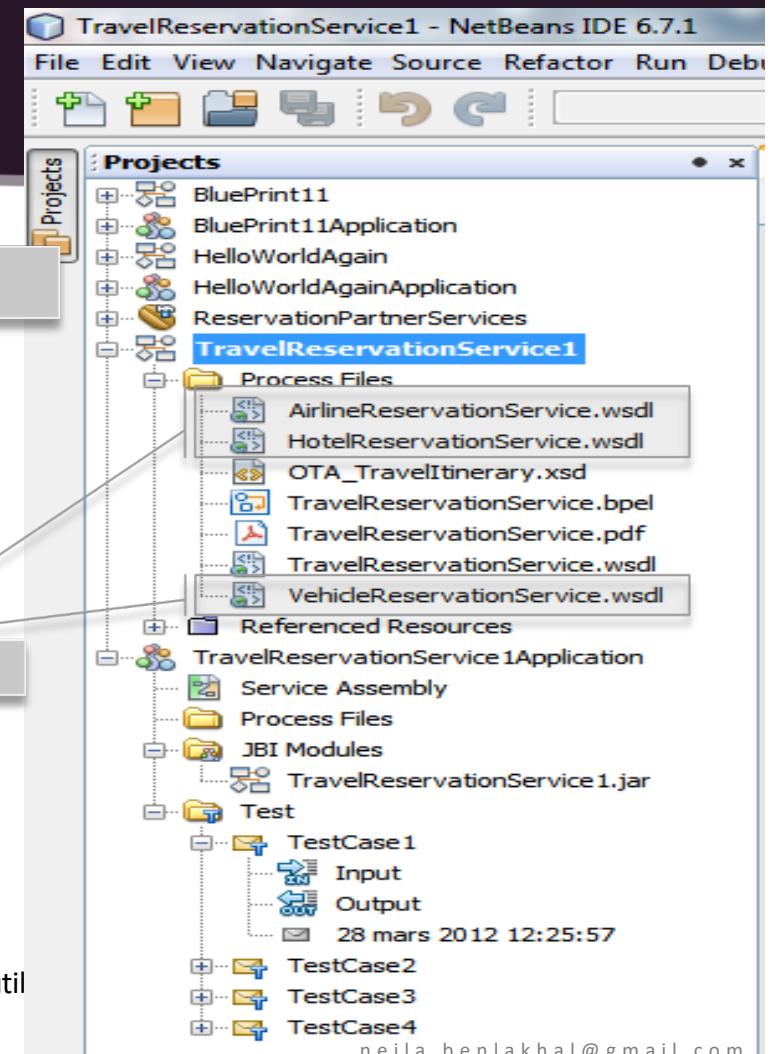
neila.benlakhal@gmail.com

Structure d'un processus BPEL



La description des Web services qui composent de processus

- ▶ Dans le volet Project, votre nouveau projet est généré.
- ▶ Un projet BPEL se compose de 3 catégories de fichiers :
 - ▶ Les fichiers de description des processus exécutable(*.bpel)+abstrait(*.wsdl et *.xsd)
 - ▶ Les fichiers de description des Services Web (*.wsdl et *.xsd)
 - ▶ Les fichiers de déploiement (*.xml) dont la structure dépend fortement du moteur BPEL utilisé



Processus BPEL en mode « design/source »

The screenshot displays the NetBeans IDE interface for editing BPEL processes. The title bar shows the file name "Synchronous.bpel". The menu bar includes "Team", "Tools", "Window", and "Help". The toolbar contains various icons for file operations like Open, Save, and Print. The main window has tabs for "Source", "Design", "Mapper", and "Logging", with "Source" currently selected. The Source tab displays the XML code for the BPEL process:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2007, Sun Microsystems, Inc. All rights reserved.-->
<process name="Synchronous"
    targetNamespace="http://enterprise.netbeans.org/bpel/Synchronous/Synchronous"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:sxt="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Trace"
    xmlns:sxed="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/Editor"
    xmlns:sxeh="http://www.sun.com/wsbpel/2.0/process/executable/SUNExtension/ErrorHandle"
    xmlns:wSDLNS="http://enterprise.netbeans.org/bpel/Synchronous/Synchronous"
    xmlns:ns1="http://localhost/Synchronous/Synchronous"
    xmlns:ns2="http://xml.netbeans.org/schema/Synchronous">

    <documentation>
        <import namespace="http://localhost/Synchronous/Synchronous"
            location="Synchronous.wsdl"
            importType="http://schemas.xmlsoap.org/wsdl/" />
    </documentation>

    <partnerLinks>
    </partnerLinks>

    <variables>
    </variables>

    <sequence>
    </sequence>
</process>
```

The Design tab shows the visual representation of the BPEL process. It consists of a green oval labeled "Process Start", a blue rounded rectangle labeled "start" with a mail icon, a blue rounded rectangle labeled "Assign1" with a curved arrow, another blue rounded rectangle labeled "end" with a double-headed arrow, and a green oval labeled "Process End". A blue rounded rectangle labeled "operation1" is shown in the palette on the left. The Mapper and Logging tabs are also present but not active.

Processus BPEL: ça ressemble à quoi ?

```

<?xml version="1.0" encoding="UTF-8"?>
<process name="Synchronous" ...>
  <partnerLinks>
    <partnerLink
      name="Synchronous" partnerLinkType="ns1:partnerlinktype1"
      myRole="partnerlinktyperole1">
    </partnerLink>
  </partnerLinks>
  <variables>
    <variable name="outputVar" messageType="ns1:responseMessage"></variable>
    <variable name="inputVar" messageType="ns1:requestMessage"></variable>
  </variables>
  <sequence>
    <receive name="start" partnerLink="Synchronous" operation="operation1"
      portType="ns1:portType1" variable="inputVar" createInstance="yes">
    </receive>
    <assign name="Assign1">
      <copy>
        <from>$inputVar.inputType/ns2:paramA</from>
        <to>$outputVar.resultType/ns2:paramA</to>
      </copy>
    </assign>
    <reply name="end" partnerLink="Synchronous" operation="operation1"
      portType="ns1:portType1" variable="outputVar">
    </reply>
  </sequence>
</process>

```

Processus BPEL

Partenaires

Variables globales

Activités

Les éléments fondamentaux d'un processus BPEL

- ▶ Un processus BPEL est constitué d'un ensemble d'étapes, chaque étape est appelée « **activité** »
- ▶ BPEL supporte les **activités simples** et les **activités complexes** :
 - ▶ Activité qui permet d'invoquer un service web : `<invoke>`
 - ▶ Activité d'attente d'un message du client qui invoque le processus métier `<receive>`
 - ▶ Activité qui génère une réponse pour une opération synchrone `<reply>`
 - ▶ Activité de manipulation de variable (ex: affectation valeur) `<assign>`
 - ▶ Activité de détection d'erreur/d'exception `<throw>`
 - ▶ Activité d'attente pour un laps de temps `<wait>`
 - ▶ Activité pour terminer un processus `<terminate>`
 - ▶ Etc.

Les éléments fondamentaux d'un processus BPEL

- ▶ Ces **activités simples** sont combinées selon la logique sous-jacente du processus métier comme dans les langages de programmation
- ▶ BPEL défini un ensemble d'**activités complexes** qui permettent de combiner des activités simples comme:
 - ▶ <sequence> pour appeler un ensemble d'activité dans un ordre prédéfini
 - ▶ <flow> pour une exécution en parallèle d'un ensemble d'activité
 - ▶ <if>, <switch>, <while>, <pick> , etc.

Les éléments fondamentaux d'un processus BPEL

- ▶ Un processus BPEL défini aussi:
 - ▶ Des partenaires <partnerlinks> : les web services qu'il compose
 - ▶ Des variables <variables> : pour le stockage et l'échange des données, des états intermédiaires, des messages.

Processus BPEL

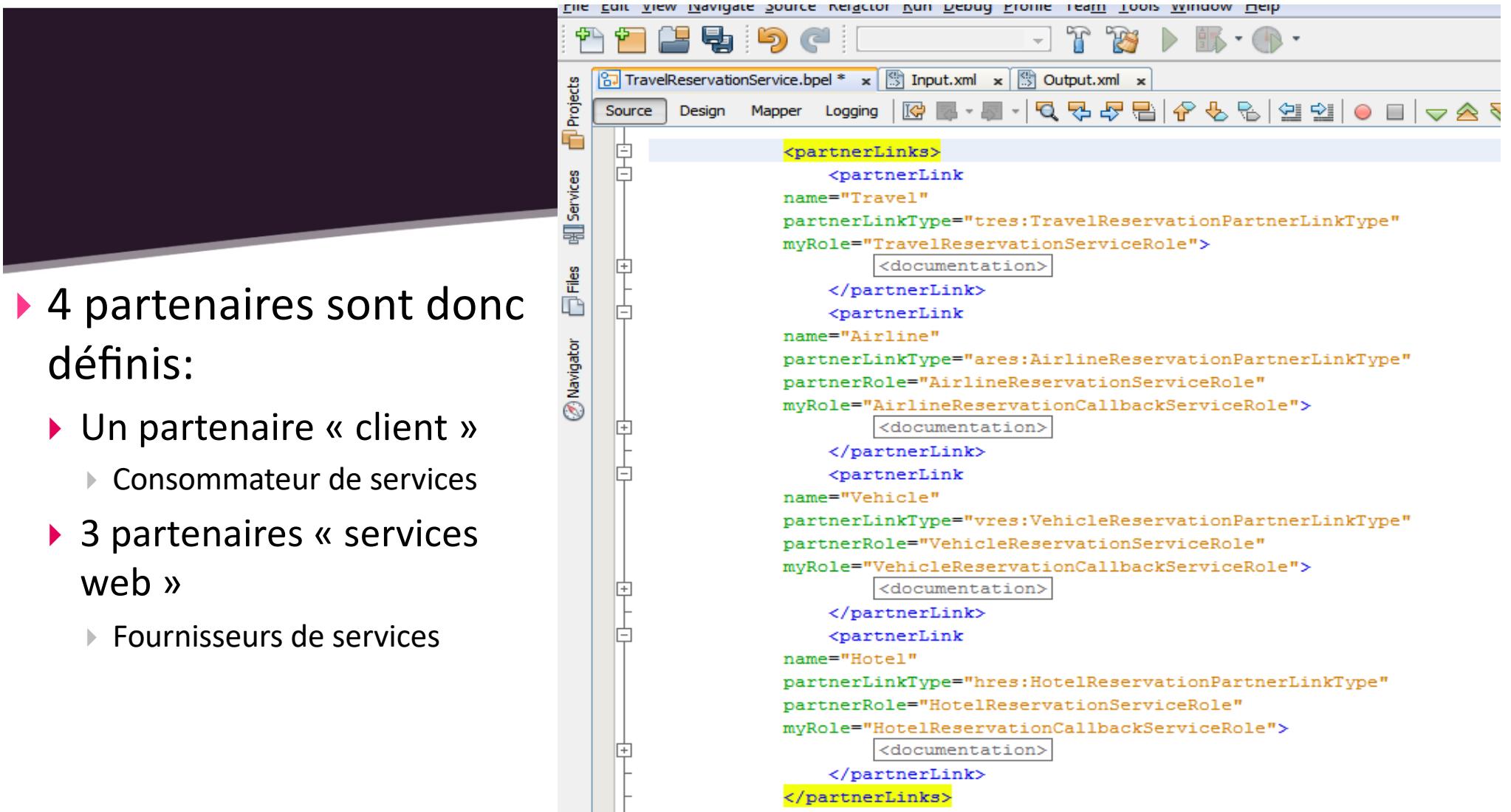
Partenaires

Variables globales

Activités

Exemple de modélisation d'un processus métier en BPEL

- ▶ On considère le processus métier suivant :
- ▶ À la réception d'un itinéraire d'un client, le processus fait une réservation d'un vol, puis d'une voiture de location puis d'un hôtel
- ▶ Voyons de plus près à quoi ressemble un processus BPEL d'un tel processus :
- ▶ Basculer du mode « design/source » pour le processus `TravelReservationService`
- ▶ Étape 1: on commence par déclarer le partenaires de notre processus `<partnerlinks>` :
 - ▶ Un partenaire client « consommateur » du processus BPEL: on l'appellera `client`
 - ▶ Et 3 partenaires Web services : `airline`, `vehicle` et `hotel`



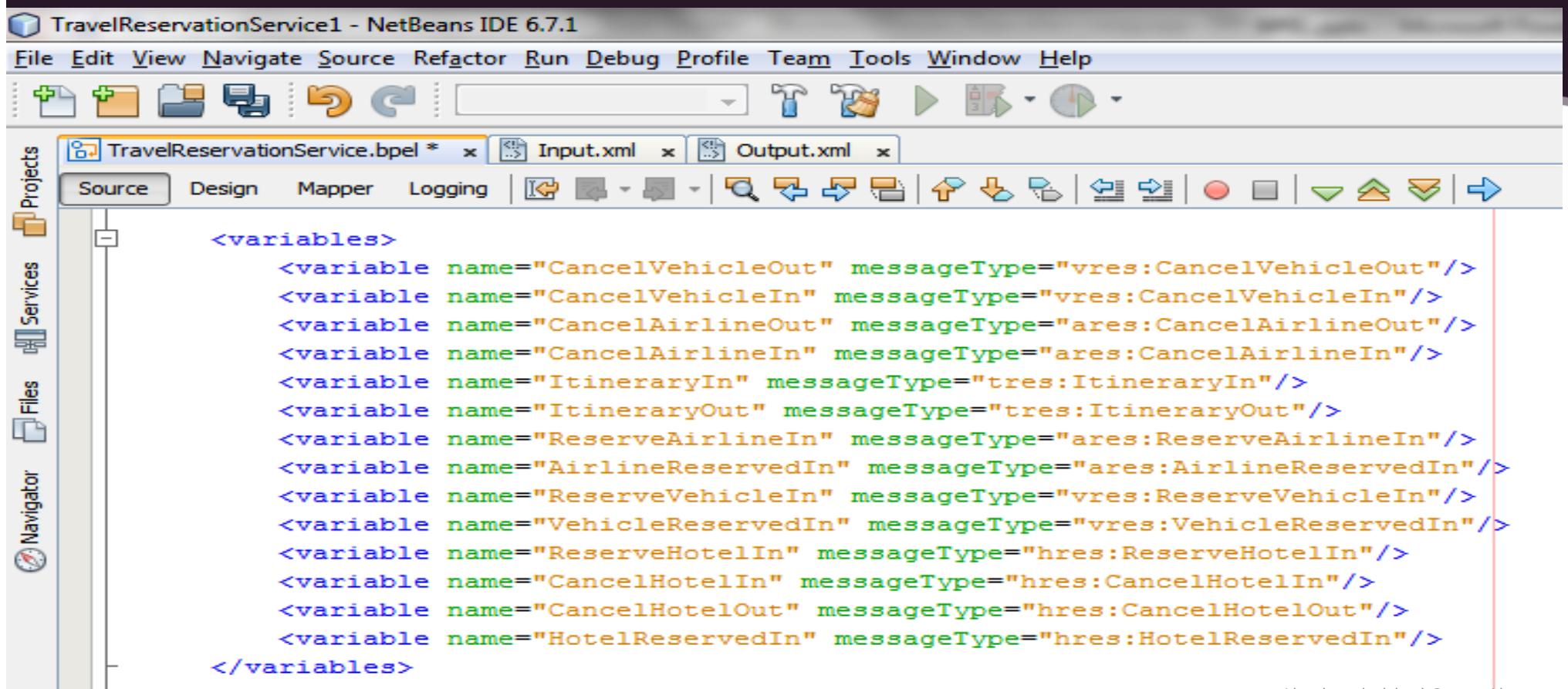
The screenshot shows a BPEL editor interface with the following details:

- Title Bar:** File Edit View Navigate Source Navigator Run Debug Profile Team Tools Window Help
- Toolbar:** Standard icons for Open, Save, Print, etc.
- Project Explorer:** Shows a single project named "TravelReservationService.bpel".
- Source View:** Displays the XML code for the partnerLinks section of the BPEL file.
- Input.xml and Output.xml tabs:** Associated with the current source file.
- Toolbars:** Mapper, Logging, and various BPEL specific icons.

```
<partnerLinks>
    <partnerLink
        name="Travel"
        partnerLinkType="tres:TravelReservationPartnerLinkType"
        myRole="TravelReservationServiceRole">
        <documentation>
    </partnerLink>
    <partnerLink
        name="Airline"
        partnerLinkType="ares:AirlineReservationPartnerLinkType"
        partnerRole="AirlineReservationServiceRole"
        myRole="AirlineReservationCallbackServiceRole">
        <documentation>
    </partnerLink>
    <partnerLink
        name="Vehicle"
        partnerLinkType="vres:VehicleReservationPartnerLinkType"
        partnerRole="VehicleReservationServiceRole"
        myRole="VehicleReservationCallbackServiceRole">
        <documentation>
    </partnerLink>
    <partnerLink
        name="Hotel"
        partnerLinkType="hres:HotelReservationPartnerLinkType"
        partnerRole="HotelReservationServiceRole"
        myRole="HotelReservationCallbackServiceRole">
        <documentation>
    </partnerLink>
</partnerLinks>
```

- ▶ 4 partenaires sont donc définis:
 - ▶ Un partenaire « client »
 - ▶ Consommateur de services
 - ▶ 3 partenaires « services web »
 - ▶ Fournisseurs de services

Ensuite, c'est la déclaration des variables :



```
<variables>
    <variable name="CancelVehicleOut" messageType="vres:CancelVehicleOut"/>
    <variable name="CancelVehicleIn" messageType="vres:CancelVehicleIn"/>
    <variable name="CancelAirlineOut" messageType="ares:CancelAirlineOut"/>
    <variable name="CancelAirlineIn" messageType="ares:CancelAirlineIn"/>
    <variable name="ItineraryIn" messageType="tres:ItineraryIn"/>
    <variable name="ItineraryOut" messageType="tres:ItineraryOut"/>
    <variable name="ReserveAirlineIn" messageType="ares:ReserveAirlineIn"/>
    <variable name="AirlineReservedIn" messageType="ares:AirlineReservedIn"/>
    <variable name="ReserveVehicleIn" messageType="vres:ReserveVehicleIn"/>
    <variable name="VehicleReservedIn" messageType="vres:VehicleReservedIn"/>
    <variable name="ReserveHotelIn" messageType="hres:ReserveHotelIn"/>
    <variable name="CancelHotelIn" messageType="hres:CancelHotelIn"/>
    <variable name="CancelHotelOut" messageType="hres:CancelHotelOut"/>
    <variable name="HotelReservedIn" messageType="hres:HotelReservedIn"/>
</variables>
```

neila.benlakhal@gmail.com

Définition des activités

- ▶ Ensuite, on défini l'ensemble des étapes du processus :
- ▶ D'abord, on attend une requête du client → l'activité **<receive>**
- ▶ On vérifie bien que la requête contient un itinéraire,
- ▶ On effectue une copie de l'itinéraire dans différentes variables qui vont être transmises aux différents services web : **<assign>** (équivalent à l'affectation) et **<copy>** (copier la variable dans une autre)
- ▶ S'il y a un vol à réserver (activité de test **<if>**), alors invoquer le partenaire **Airline** **<invoke>**
- ▶ On attend la réponse du partenaire **Airline**, ou bien d'un time-out se produit (activité **<onAlarm>**) : activité **<pick>** pour réunir deux choix exclusifs.
- ▶ Si la réservation a abouti, le résultat est copié dans la variable de sortie (activité **<assign>** et **<copy>**)
- ▶ De même pour la réservation du véhicule et de l'hôtel.
- ▶ Enfin, la réponse contenu dans une variable est retourné au client avec l'activité **<reply>**

Syntaxe d'un processus BPEL

- ▶ Là on va voir la syntaxe d'un processus BPEL et de ses différents éléments :
- ▶ L'élément racine d'un processus est l'élément <processus>

Les éléments de définition des processus BPEL: L'élément <process>

- ▶ C'est l'élément racine du processus, cet élément reprend l'ensemble de la définition d'un processus BPEL.
 - ▶ L'attribut **name** identifie le processus BPEL en lui affectant un nom.
 - ▶ L'élément **<process>** comporte également les espaces de noms associés aux types de données utilisés dans la définition du processus BPEL.

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="Synchronous"
    targetNamespace="http://enterprise.netbeans.org/bpel/Synchronous/Synchronous"
    xmlns="http://docs.oasis-open.org/wsbpel/2.0/process/executable"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable"...
    xmlns:wsdIINS="http://enterprise.netbeans.org/bpel/Synchronous/Synchronous"
    xmlns:ns1="http://localhost/Synchronous/Synchronous"
    xmlns:ns2="http://xml.netbeans.org/schema/Synchronous">...</process>
```

Syntaxe de composition d'un processus

- ▶ Un processus BPEL est un document XML ayant pour racine l'élément obligatoire **<processus>**
- ▶ Les activités sont généralement dans un élément **<sequence>** pour une **exécution sérialisée**
- ▶ Le processus, va attendre un message d'invocation pour commencer son exécution: c'est l'élément **<receive>**
- ▶ Ensuite le processus va invoquer les web services concernés en utilisant l'élément **<invoke>**

```
<process ...>
...
<sequence>
<!-- Wait for an incoming request to start the process -->
<receive ... />

<!-- Invoke a set of related web services, one by one -->
<invoke ... />
<invoke ... />
<invoke ... />
...
</sequence>
</process>
```

- ▶ Pour une **exécution simultanée** de services Web :
- ▶ Utiliser l'élément **<flow>**

```
<process ...>
...
<sequence>
    <!-- Wait for the incoming request to start the process -->
    <receive ... />

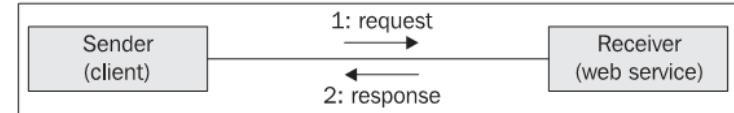
    <!-- Invoke a set of related web services, concurrently -->
    <flow>
        <invoke ... />
        <invoke ... />
        <invoke ... />
    </flow>
    ...
</sequence>
</process>
```

Processus synchrone/asynchrone

- ▶ On distingue deux types de processus BPEL :

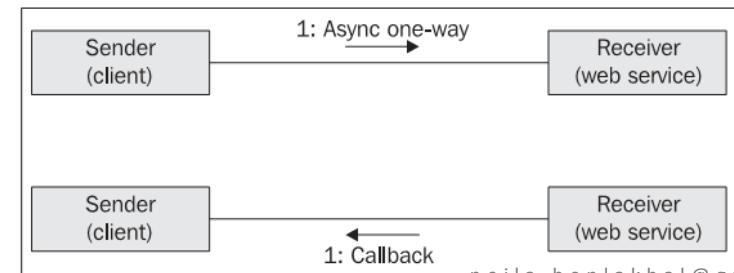
- ▶ **Processus synchrone**

- ▶ attente active (appel bloquant) de la réponse
 - ▶ traitements courts (quelques secondes)
 - ▶ sortie = message ou exception



- ▶ **Processus asynchrone**

- ▶ pas d'attente active (appel non bloquant) d'une éventuelle réponse
 - ▶ traitements longs
 - ▶ sortie = message



neila.benlakhal@gmail.com

A Synchronous Sample Process , c'est quoi ?

- ▶ Un processus synchrone réfère à un style de conversation où le client envoie une requête à un processus, et doit attendre la réponse pour pouvoir poursuivre l'exécution (comme les applications WEB en PHP ou encore le protocole HTTP)
- ▶ Pur le cas d'un processus **asynchrone**, le client n'a pas à attendre la réponse et poursuit son exécution
- ▶ Dans les deux cas, l'élément **<invoke>** peut être utilisé pour appeler le service Web de façon synchrone/Asynchrone

- ▶ Pour un processus synchrone,
 - ▶ L'élément **<invoke>** permet d'envoyer une requête et de recevoir la réponse de l'opération appelée
- ▶ Pour un processus Asynchrone,
 - ▶ On a besoin aussi de l'élément **<receive>** pour recevoir la réponse. Entre l'envoi et la réception de la réponse, le processus pourrait faire d'autres traitements

```
<process ...>
<sequence>

    <!-- Wait for the incoming request to start the process -->
    <receive ... />

    <!-- Invoke an asynchronous operation -->
    <invoke ... />

    <!-- Do something else... -->

    <!-- Wait for the callback -->
    <receive ... />

    ...

</sequence>
</process>
```

Synchronous/Asynchronous Business Processes

- ▶ Un processus BPEL est déployé en tant que service Web,
 - ▶ De ce fait, un processus peu être synchrone/Asynchrone
- ▶ **Un processus BPEL synchrone:** retourne immédiatement une réponse à un client une fois son exécution s'est achevée : le client est bloqué en attendant la réponse du processus
- ▶ D'un autre coté, un **processus BPEL Asynchrone** ne bloque pas un client.
- ▶ Les processus métiers ont tendance à durer dans le temps, c'est ce qui privilégie le choix de les modéliser en tant que processus asynchrones.
- ▶ **Comment alors la spécification BPEL modélise les processus Asynchrones?**
!!!

Processus Synchrone

```
<process ...>
  <sequence>
    <!-- Wait for the incoming request to start the process -->
    <receive ... />

    <!-- Invoke a set of related web services -->
    ...
    <!-- Return a synchronous reply to the caller (client) -->
    <reply ... />

  </sequence>
</process>
```

Processus Asynchrone

```
<process ...>
  <sequence>
    <!-- Wait for the incoming request to start the process -->
    <receive ... />

    <!-- Invoke a set of related web services -->
    ...
    <!-- Invoke a callback on the client (if needed) -->
    <invoke ... />

  </sequence>
</process>
```

<partnerLinks>

- ▶ Un processus BPEL interagit avec des services Web de deux façons:
 - ▶ Le processus BPEL peut invoquer des opérations fourni par d'autres web services
 - ▶ Le processus BPEL peut être lui même invoqué par des clients soit pour une requête soit pour un callback d'un client qui a fini de s'exécuter
- ▶ Ces liens sont modélisés avec l'élément <**partnerlinks**>
- ▶ Un processus BPEL doit impérativement définir au moins un seul élément <**partnerlink**> : le client qui invoque le processus.
- ▶ Un <**partnerlink**> est une sorte d'intermédiaire entre les activités du processus BPEL et les opérations des services Web mis à disposition des partenaires.

Types de partenaire

- ▶ Pour résumer, un partenaire d'un processus Web, c'est :
 - ▶ Un service web que le processus BPEL invoque
 - ▶ Un service web qui invoque le processus BPEL
 - ▶ Un service web qui est en même temps invoqué par le processus et invoque le processus.
- ▶ L'élément `<partnerlinks>` regroupe l'ensemble des partenaires: un élément `<partenairlink>` pour chaque partenaire

```
...<partnerLinks>
    <partnerLink
        name="Travel"
        partnerLinkType="tres:TravelReservationPartnerLinkType"
        myRole="TravelReservationServiceRole">
    </partnerLink>
    <partnerLink
        name="Airline"
        partnerLinkType="ares:AirlineReservationPartnerLinkType"
        partnerRole="AirlineReservationServiceRole"
        myRole="AirlineReservationCallbackServiceRole">
    </partnerLink>
    <partnerLink
        name="Vehicle"
        partnerLinkType="vres:VehicleReservationPartnerLinkType"
        partnerRole="VehicleReservationServiceRole"
        myRole="VehicleReservationCallbackServiceRole">
    </partnerLink>
    <partnerLink
        name="Hotel"
        partnerLinkType="hres:HotelReservationPartnerLinkType"
        partnerRole="HotelReservationServiceRole"
        myRole="HotelReservationCallbackServiceRole">
    </partnerLink>
</partnerLinks>
```

Partenaire client

Partenaires services Web

Les éléments de définition des processus BPEL: L'élément < partnerlinks > :

- ▶ L'élément `partnerlink` comporte un nom(`name`), un type de lien partenaire(`partnerLinkType`) et un rôle(`myRole`) ainsi que le rôle du partenaire (`partnerRole`) comme attributs.
- ▶ Ex: Ce partenaire représente le client qui envoi un input au processus BPEL et reçoit la réponse :

```
<partnerLinks>...  
<partnerLink  
    name="Travel"  
    partnerLinkType="tres:TravelReservationPartnerLinkType"  
    myRole="TravelReservationServiceRole">  
</partnerLink>...  
</partnerLinks>
```



Où sont définis les types ??

Les types de ports des partenaires

- ▶ Les types des ports que fourni les <partnerlink> sont défini dans le WSDL du processus BPEL ou des services web:
- ▶ Un port d'un partenaire défini un rôle qui est la fonction/l'opération que rempli le processus
- ▶ Ouvrez TravelReservationService.wsdl

TravelReservationService1 - NetBeans IDE 6.7.1 49

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Search (C)

Projects

Files Services

...age TravelReservationService.bpel Input.xml Output.xml Asynchronous.bpel Synchronous.bpel TravelReservationService.wsdl *

Source WSDL Partner

```
<bpws:propertyAlias  
    propertyName="tres:ItineraryRefId"  
    messageType="vres:ReserveVehicleIn"  
    part="itinerary">  
    <bpws:query>/ota:TravelItinerary/ota:ItineraryRef/ota:UniqueID</bpws:query>  
</bpws:propertyAlias>  
  
<bpws:propertyAlias  
    propertyName="tres:ItineraryRefId"  
    messageType="hres:HotelReservedIn"  
    part="itinerary">  
    <bpws:query>/ota:TravelItinerary/ota:ItineraryRef/ota:UniqueID</bpws:query>  
</bpws:propertyAlias>  
  
<bpws:propertyAlias  
    propertyName="tres:ItineraryRefId"  
    messageType="hres:ReserveHotelIn"  
    part="itinerary">  
    <bpws:query>/ota:TravelItinerary/ota:ItineraryRef/ota:UniqueID</bpws:query>  
</bpws:propertyAlias>  
  
<plnk:partnerLinkType name="TravelReservationPartnerLinkType">  
    <plnk:role name="TravelReservationServiceRole"  
        portType="tres:TravelReservationPortType">  
    </plnk:role>  
</plnk:partnerLinkType>  
</definitions>
```

http://www.sun.com/javaone/05/Tr...
www.sun.com/javaone/05/TravelReservationService
pes
http://www.sun.com/javaone/05/TravelReservat
ports
HotelReservationService.wsdl
AirlineReservationService.wsdl
VehicleReservationService.wsdl
ssages
ItineraryIn

Test Results Output Tasks

<no tasks>

<no tasks> in all opened projects

neila.benlakhal@gmail.com

```
<!--définition d'un partenaire ds .bpel-->
<partnerLinks>...
<partnerLink
    name="Travel"
    partnerLinkType="tres:TravelReservationPartnerLinkType"
    myRole="TravelReservationServiceRole">
</partnerLink>...
</partnerLinks>

<!-- definition du type du partenaire dans .wsdl-->
<plnk:partnerLinkType
    name="TravelReservationPartnerLinkType">
    <plnk:role name="TravelReservationServiceRole"
        portType="tres:TravelReservationPortType">
        </plnk:role>
</plnk:partnerLinkType>

<!-- definition du role du partenaire dans .wsdl-->
<portType name="TravelReservationPortType">
    <operation name="buildItinerary">
        <input message="tns:ItineraryIn"/>
        <output message="tns:ItineraryOut"/>
        <fault name="itineraryProblem" message="tns:ItineraryFault"/>
    </operation>
</portType>
```

Les éléments de définition des processus BPEL: L'élément <variables> :

- ▶ Lors de l'exécution d'un processus BPEL,
- ▶ Un ensemble d'opérations des services partenaires sont invoqués par un échange de messages
- ▶ On a besoin alors de stocker le résultat de l'invocation quelque part pour l'utiliser
- ▶ C'est le rôle essentiel de l'élément <**variables**>

Les éléments de définition des processus BPEL: L'élément <variables> :

- ▶ L'élément <variables> regroupe les variables utilisées dans le processus BPEL afin de sauvegarder les échanges de messages avec les partenaires ainsi que les états intermédiaires du processus (utile pour le recouvrement d'erreurs)
- ▶ Les variables en BPEL fonctionnent de la même manière que dans la programmation classique :
 - ▶ On y stocke des valeurs temporaires
 - ▶ Elles font partie d'une expression
 - ▶ Elles sont passées à des Partner (externe).

<variables>

```
  <variable name="outputVar" messageType="ns1:responseMessage">
    <documentation>Output variable.</documentation>
  </variable>...</variables>
```

L'élément <variable>

- ▶ Plusieurs déclarations de variables sont logiquement regroupées dans l'élément <variables>
- ▶ Portée des variables :
 - ▶ Les variables défini dans la racine sont dites globales
 - ▶ Toutes autres variables est dite locale
- ▶ Une variable a comme attributs: **name + type** :

```
<variables>
    <variable name="outputVar" messageType="ns1:responseMessage">
        <documentation>Output variable.</documentation>
    </variable>      ...
</variables>
```

Les types de variables

- ▶ Il y a 3 types de variables :
 - ▶ Variables décrites dans le fichier BPEL
 - ▶ Variables contenant des messages décrits dans le fichier WSDL du processus
 - ▶ Variables contenant des messages décrits dans le fichier WSDL du Service Web
- ▶ Selon le type de la variable, l'élément `<variable>` a un contenu différent,
 - ▶ L'attribut **messageType**
 - ▶ La variable contient un message défini dans la description WSDL du service Web.
 - ▶ L'attribut **element**
 - ▶ La variable contient un élément défini dans un schéma XML.
 - ▶ L'attribut **type**
 - ▶ La variable contient un type simple de donnée défini dans un schéma XML comme par exemple le type chaîne de caractères.

Exemples de variables

```
<variables>
<variable name="CancelVehicleOut"
          messageType="vres:CancelVehicleOut"/>
<variable name="PartialInsuranceDescription"
          element="ins:InsuranceDescription"/>
<variable name="LastName"
          type="xs:string"/>
</variables>
```

Ces déclarations
réfèrent à des types
se trouvant dans les
fichier s .wsdl

Les activités d'un processus BPEL

- ▶ Un processus BPEL est décrit par un ensemble d'étapes /activités (**activity**)
- ▶ BPEL supporte 2 types d'activités :
 - ▶ **Les activités simples** comparables aux tâches décrites dans les processus métiers
 - ▶ **Les activités structurées** d'ordonnancement de l'exécution d'un ensemble d'activités simples/structurées
 - ▶ Les structures de contrôle,
 - ▶ les flux de données,
 - ▶ la gestion des exceptions,
 - ▶ la gestion des événements externes et la coordination de l'échange de messages entre les instances de processus BPEL
 - ▶ etc.

Les activités simples

- ▶ BPEL défini 8 éléments de description d'activités simples :
 - ▶ L'élément `receive` de réception d'un message.
 - ▶ L'élément `reply` de renvoi d'une réponse au partenaire.
 - ▶ L'élément `invoke` d'invocation d'une opération d'un partenaire.
 - ▶ L'élément `assign` d'affectation des variables.
 - ▶ L'élément `throw` de déclenchement d'une exception.

L'élément <receive>, <invoke> et <reply>

- ▶ Rappelons que :
- ▶ Avec <invoke>, le processus BPEL invoque les opérations des services web
- ▶ Avec <receive>, il attend la réponse d'invocation d'un service Web ou encore il attend d'être invoquer par un client : c'est le message initial qui lancera le processus
- ▶ Autre usage typique de <receive> est les callbacks : pour les processus asynchrones
- ▶ Quant à <reply>, un processus BPEL peut envoyer une réponse en mode synchrone.

Exemple d'activité <invoke>

- ▶ Cette activité <invoke> a pour nom(name) ReserveAirline, elle appelle une opération sur le service Web partenaire (partnerLink) Airline, à travers le port(portType) ares:AirlineReservationPortType la variable envoyée est l'itinéraire du vol demandé dans la variable inputVariable est ReserveAirlineln

<invoke

```
  name="ReserveAirline"  
  partnerLink="Airline"  
  portType="ares:AirlineReservationPortType"  
  operation="reserveAirline"  
  inputVariable="ReserveAirlineln"> ...
```

</invoke>

Exemple d'activité <receive>

- ▶ Cette activité <receive> a pour nom(name) `Receiveltinerary`, elle attend l'appel de l'opération `buildItinerary` par le client partenaire (partnerLink) `Travel`, à travers le port de ce processus (portType) `tres:TravelReservationPortType`. L'itinéraire demandé par le client est stocké dans la variable `tineraryIn`
- ▶ À chaque nouvelle exécution du processus une nouvelle instance est créée par le BPEL engine: `createInstance="yes"`)

```
<receive
    name="Receiveltinerary"
    partnerLink="Travel"
    portType="tres:TravelReservationPortType"
    operation="buildItinerary"
    createInstance="yes"
    variable="ItineraryIn">...</receive>
```

Ce que permet BPEL

- ▶ Les activités complexes: switch, loop, scopes, correlation sets, etc
- ▶ Traitement transactionnel
- ▶ Gestion des erreurs et compensation : Fault Handlers
- ▶ Compensation Handlers, Event Handlers
- ▶ Etc.

Labs about BPEL process design and execution

Atelier BPEL

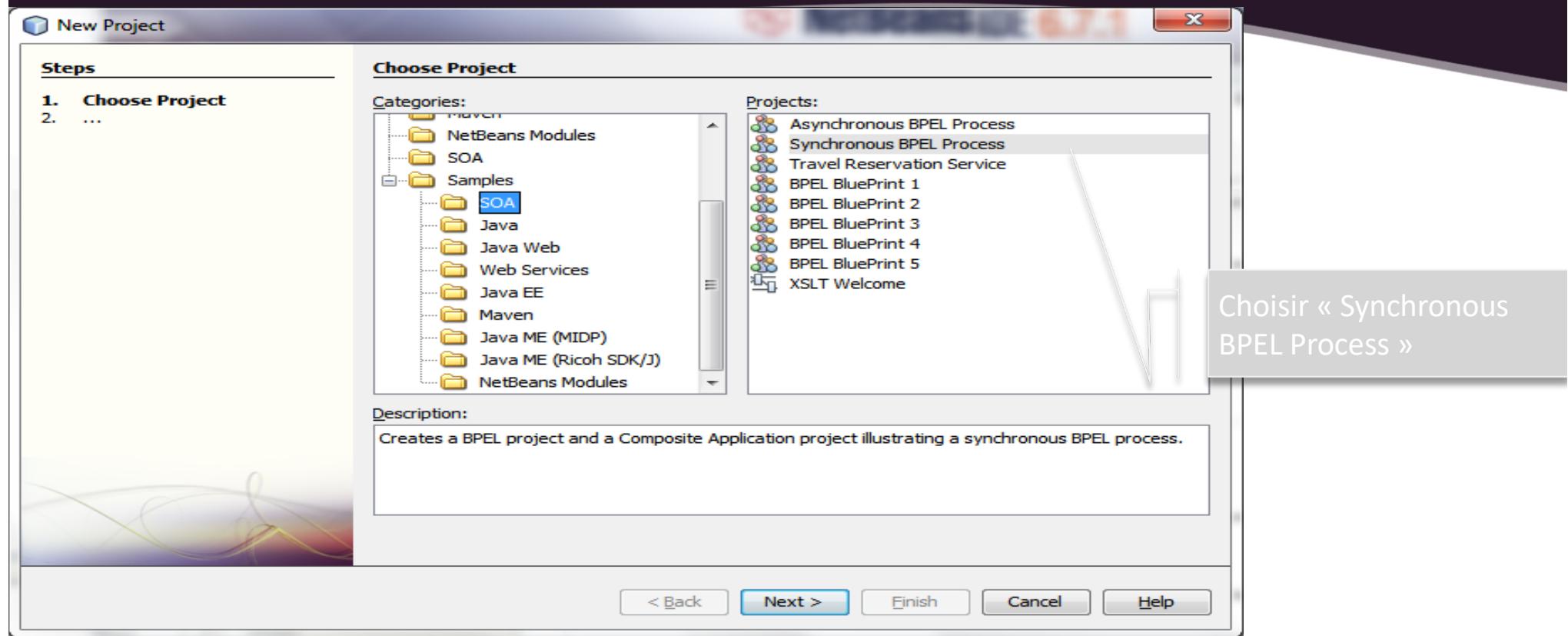
- ▶ On utilisera en partie la documentation officiel de OpenESB community pour l'atelier BPEL:
- ▶ <http://www.open-esb.net/>
- ▶ <http://docs.oracle.com/cd/E19182-01/821-0539/821-0539.pdf>

Atelier BPEL

- ▶ Je vous demande de préparer les prérequis pour ce tutoriel en s'inscrivant et installant OpenESB 2.3 :
 - ▶ <http://www.open-esb.net/>

Quelques captures d'écran du premier tutoriel

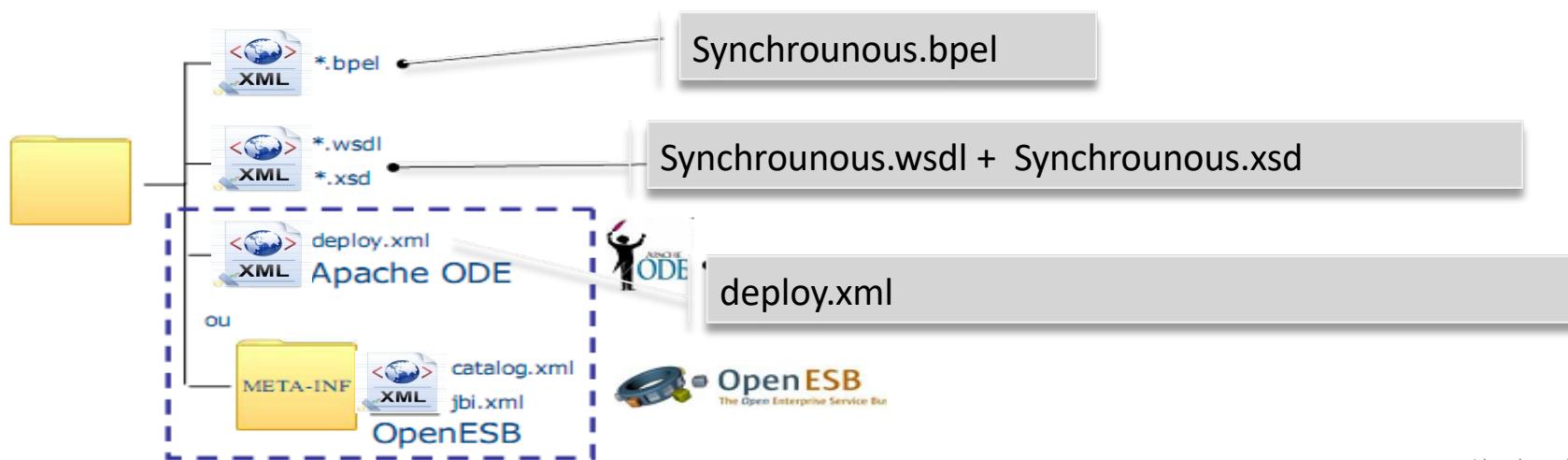
Création d'un nouveau projet BPEL (File -> New Project) à partir des exemples (Samples)



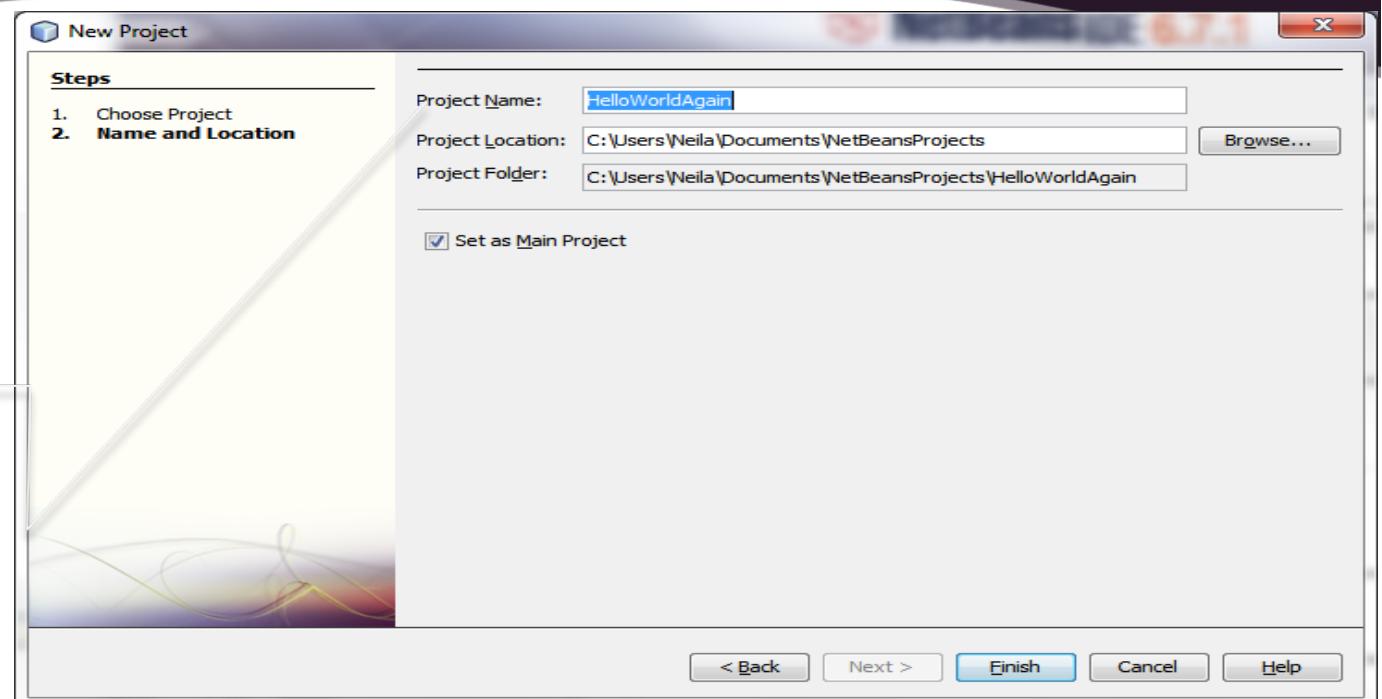
Structure d'un projet BPEL

Apache ODE (Orchestration Director Engine) executes business processes written following the BPEL standard.

- ▶ Dans le volet **Project**, votre nouveau projet est généré.
- ▶ Votre projet BPEL synchrone se compose des 3 fichiers suivants :

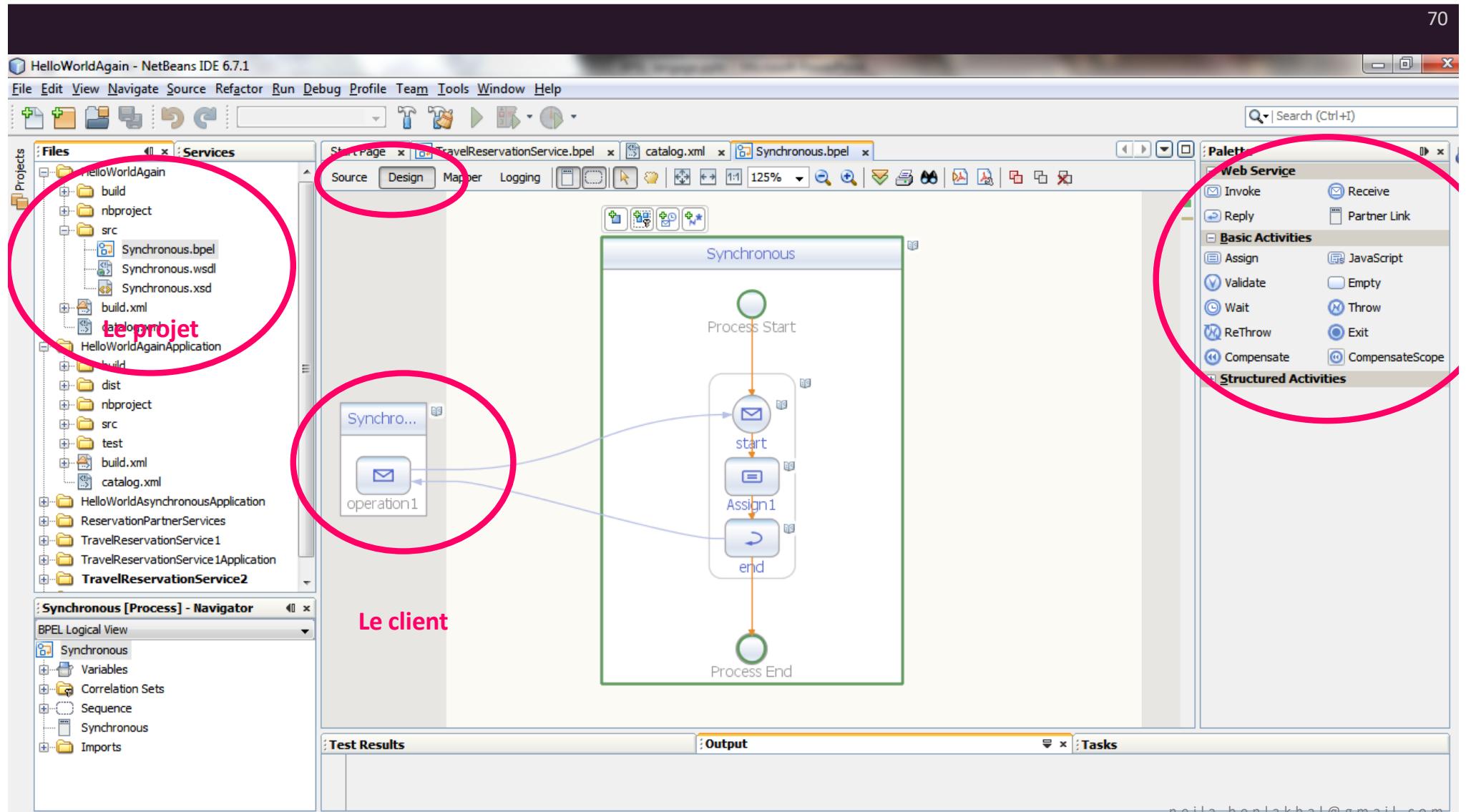


Préciser le nom « HelloWorldAgain » et l'emplacement du projet BPEL;
Clique sur Finish et le tour est joué !



La définition abstraite du processus

- ▶ La partie statique d'un processus est définie par un document WSDL : Synchronous.wsdl
- ▶ Rappel : le document WSDL permet de décrire
 - ▶ Les points d'entrées et de sorties du processus
 - ▶ Définir les types des données (XML Schema) et les messages utilisés pour décrire l'état du processus
 - ▶ Les opérations qui sont autorisées et qui permettent d'invoquer le processus



Passer en mode <source> pour éditer le processus BPEL en mode « texte »⁷¹

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Copyright (c) 2007, Sun Microsystems, Inc. All rights reserved.-->
<process name="Synchronous" ...>

<import namespace="http://localhost/Synchronous/Synchronous" location="Synchronous.wsdl"
    importType="http://schemas.xmlsoap.org/wsdl/" />

<partnerLinks>
    <partnerLink name="Synchronous" partnerLinkType="ns1:partnerlinktype1"
        myRole="partnerlinktyperole1">
    </partnerLink>
</partnerLinks>
<variables>
    <variable name="outputVar" messageType="ns1:responseMessage"></variable>
    <variable name="inputVar" messageType="ns1:requestMessage"></variable>
</variables>
<sequence>
<receive ...></receive>
<assign name="Assign1">
    <copy> <from>$inputVar.inputType/ns2:paramA</from>
        <to>$outputVar.resultType/ns2:paramA</to>
</assign>
<reply ...></reply>
</sequence>
</process>
```

Un partenaire est défini

Deux variables: input et output sont définies

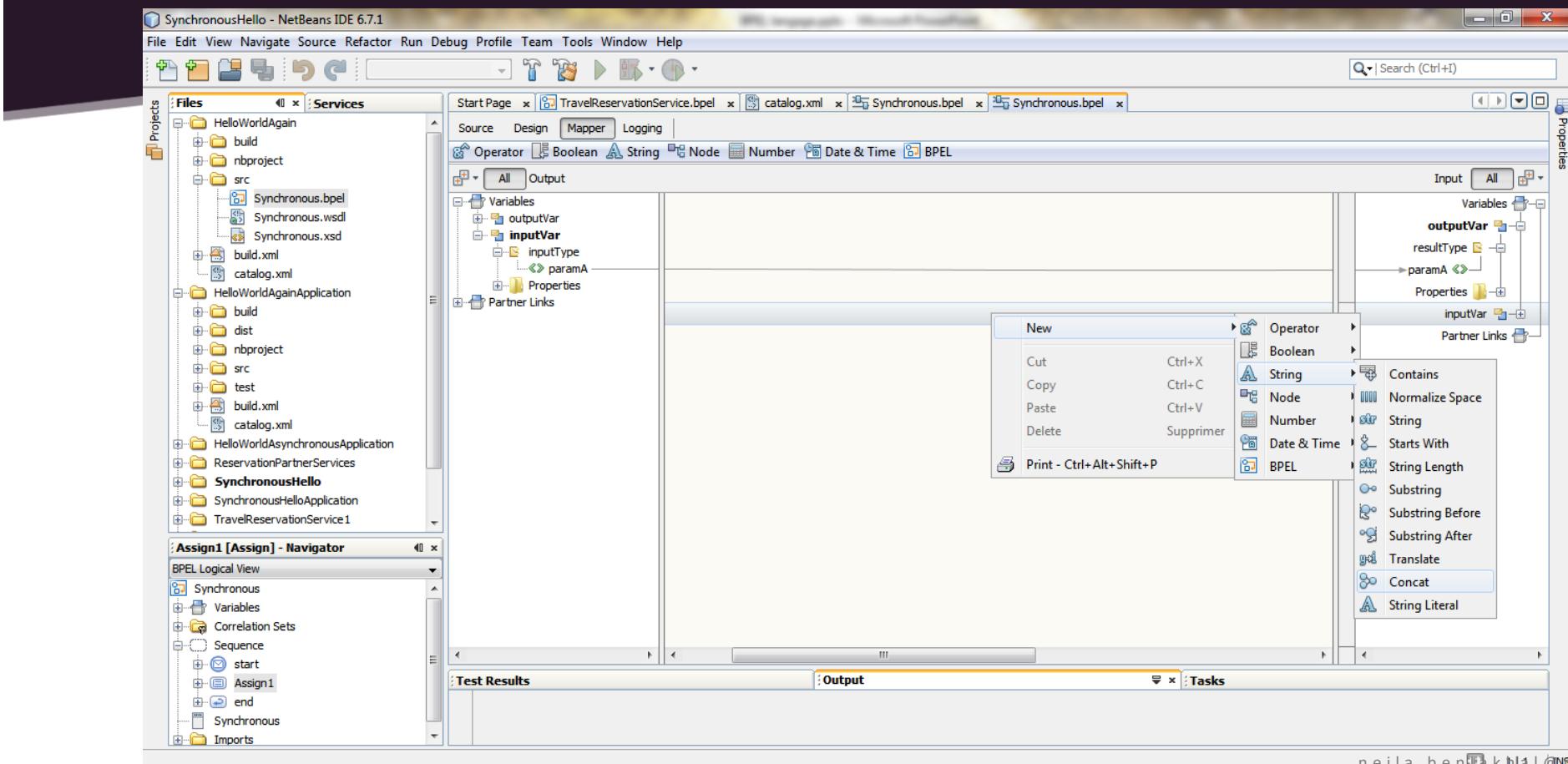
3 activités déclenché de façon sérialisées (sequence)

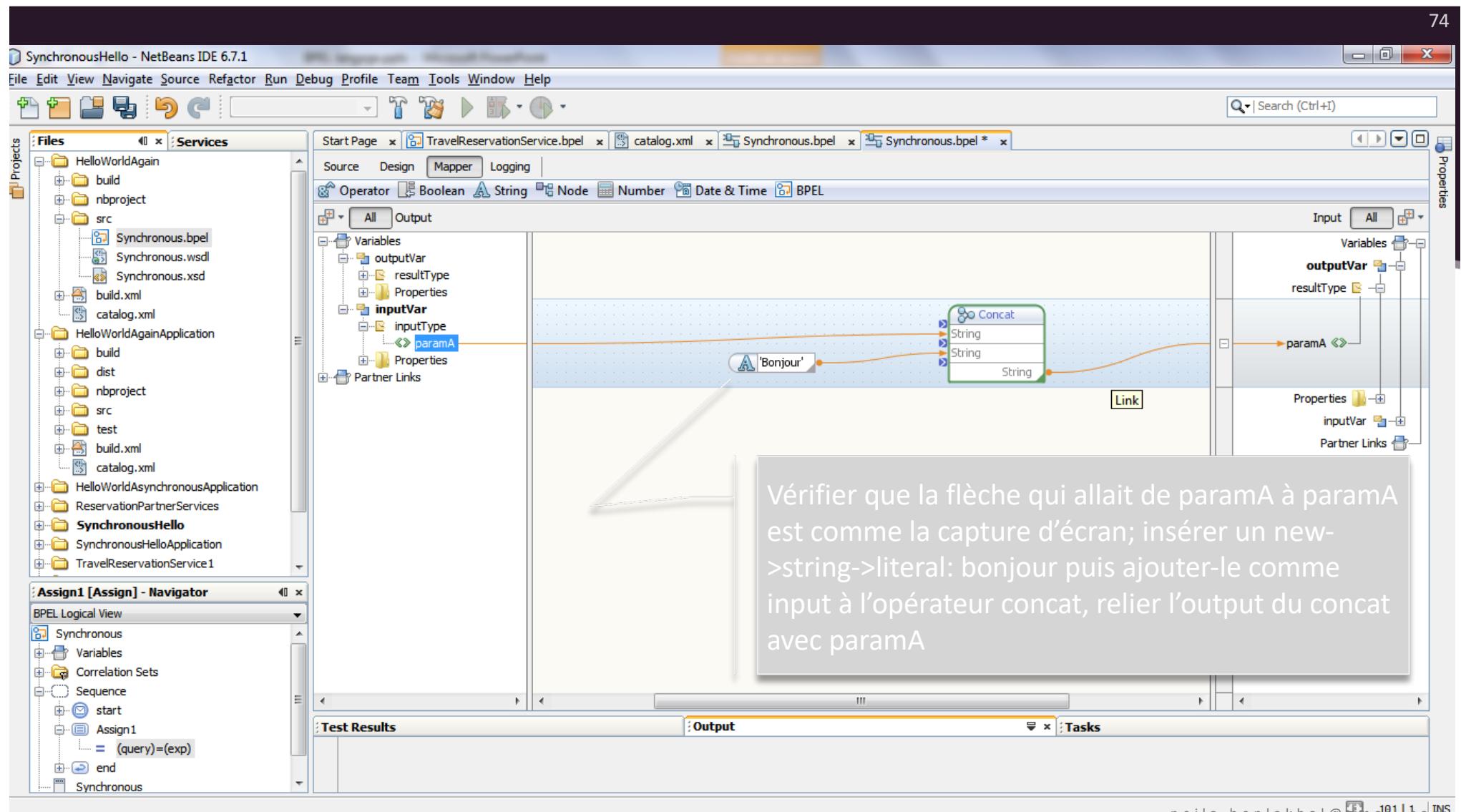
- ▶ Actuellement l'output est identique à l'output dans ce processus BPEL

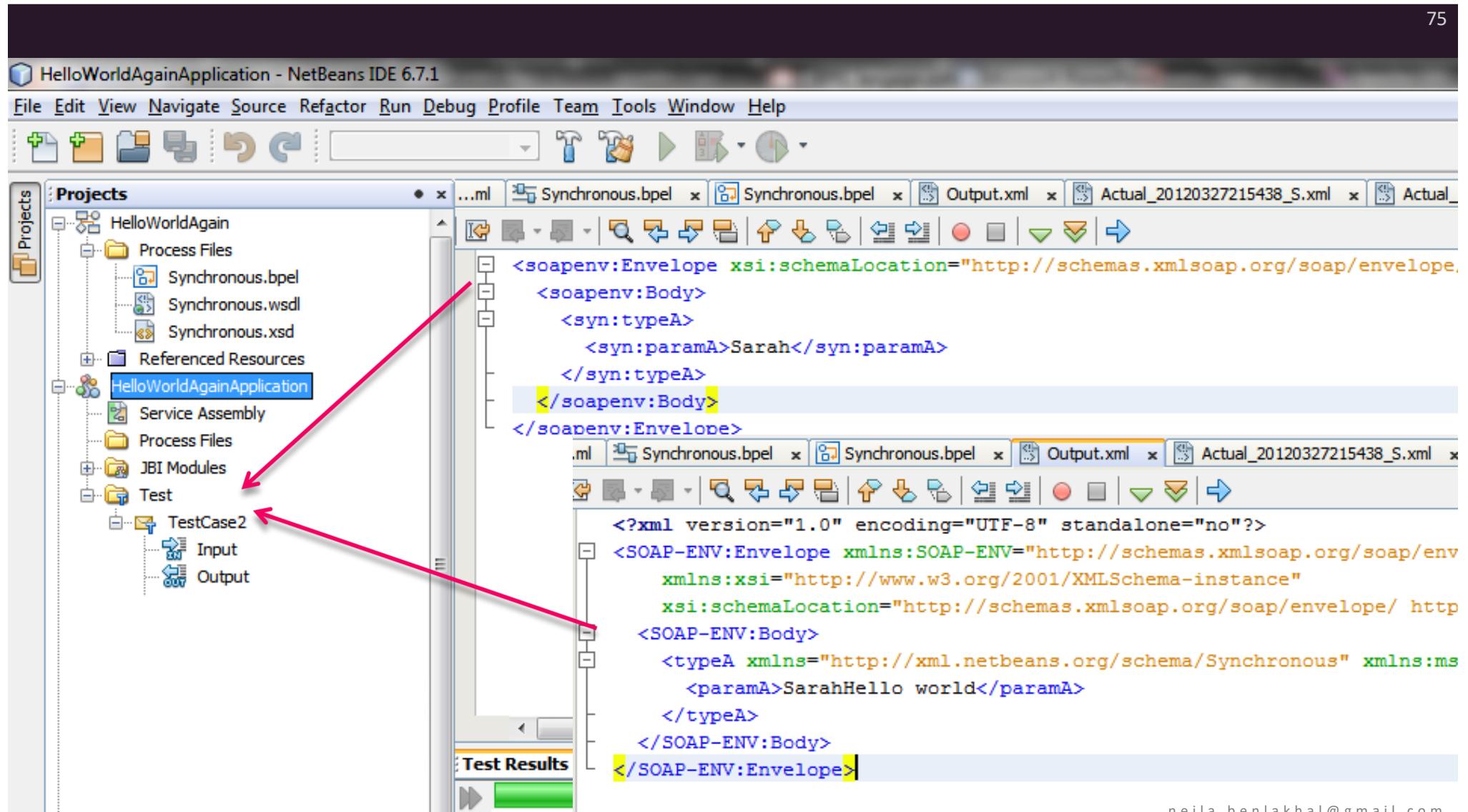
```
<assign name="Assign1">
  <copy> <from>$inputVar.inputType/ns2:paramA</from>
    <to>$outputVar.resultType/ns2:paramA</to>
```

- ▶ On veut changer le processus synchrone BPEL pour faire la concaténation de la chaîne Hello à l'input et la retourner :

Faire un double clique sur l'activité <assign> pour la modifier







Ressources

- ▶ Lien de téléchargement de GlassFish ESB v2.2 (including GlassFish 2.1.1 and NetBeans 6.7.1)

- ▶ <http://www.logicoy.com/>
- ▶ <https://www.open-esb.net/>