02267: Software Development of Web Services

Week 5

Hubert Baumeister

huba@dtu.dk

Department of Applied Mathematics and Computer Science
Technical University of Denmark

Fall 2015

Recap

- XML Schema
  - Complex Data in SOAP Messages
  - JAXB: XML Schema definitions and Java
  - WSDL: User defined Fault Messages
  - WSDL: Document Style Binding

→ Done with single SOAP based Web service

→ Proceed to Composite Web services representing Business Processes

Contents

SOAP Encoding

Business Processes

BPEL

BPEL and NetBeans

A BPEL Process as a Web service

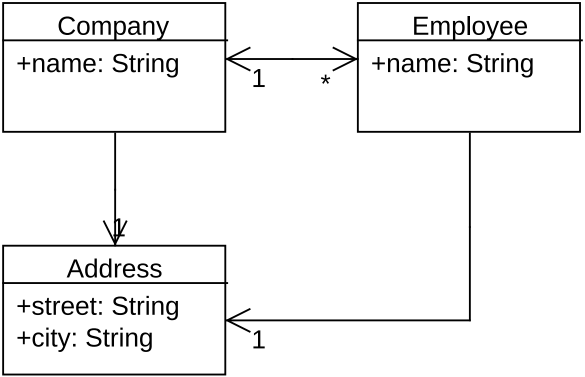
Calling other Web services from BPEL

Choices in the binding section:

- style = RPC | document
- encoding = literal | encoded
  - → Gives: RPC/literal, document/literal, RPC/encoded, (document/encoded does not make sense. Why?)
- literal
  - → The XML in the SOAP message for **data types** is exactly as described in the types section
  - Manual encoding of cycles and aliasing
- encoding
  - → The XML described in the WSDL is **encoded** in a **graph** structure represented in XML
    - Originated in pre-WSDL times to encode programming language datatypes in SOAP messages
    - Can deal with cycles and aliasing

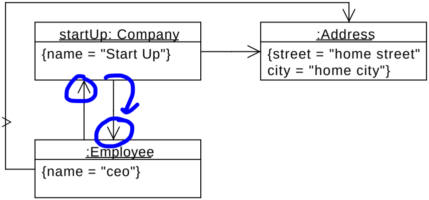
SOAP Encoding: Cycles and Aliasing

- Each employee has a pointer back to its company → possibility for object diagrams with cycles



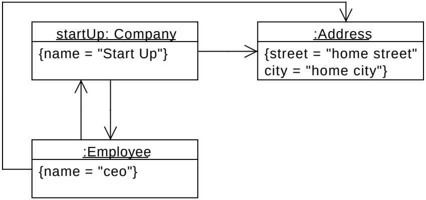
Encoding = literal

```
<ns2:someOp xmlns:n2="..."><input>
<company>
 <name>Start Up</name>
 <employee>
  <name>ceo</name>
  <address>
   <street>home street</street>
   <city>home city</city>
  </address>
  <company>
    <name>Start up</name>
    <employee>
     <name>ceo</name>
     <address>
      <street>home street</street>
      <city>home city</city>
     </address>
     <company>...</company>
    </employee>
    <address>...</address>
 </employee>
 <address>
  <street>home street</street>
  <city>home city</city>
 </address>
</company>
</input></ns2:someOp>
```



Encoding = encoded

```
<ns2:someOp xmlns:n2="..."><input>
<company href="id02"/>
<multiref id="id02">
    <name>Start Up</name>
    <address href="id01"/>
    <employee>
        <company href="id02"/>
        <name>ceo</name>
        <address href="id01"/>
    </employee>
</multiref>
<multiref id="id01">
    <street>Home Street</street>
    <city>Home City</city>
</multiref>
</input></ns2:someOp>
```



Literal vs Encoded

- literal
  - + The XML in the SOAP message for **data types** is exactly as described in the types section
  - – Manual encoding of cycles and aliasing
- encoded
  - → The XML described in the WSDL is **encoded** in a **graph** structure represented in XML
    - + Can deal with cycles and aliasing
    - – Java → XML Schema (via JAXB) → XML encoding (via SOAP encoding)
    - ! SOAP encoding predates WSDL
  - → WS-I Basic profile discourages the use of SOAP encoding

    `http://www.ws-i.org/deliverables/workinggroup.aspx?wg=basicprofile`
  - → JAX-WS does not support anymore SOAP encoding

Contents

SOAP Encoding

Business Processes

BPEL

BPEL and NetBeans

A BPEL Process as a Web service

Calling other Web services from BPEL

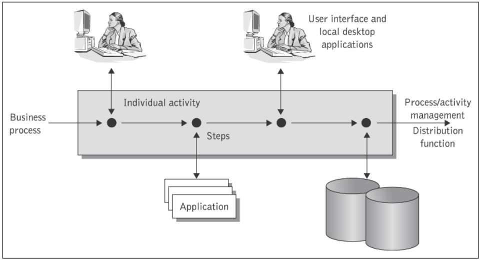
Example: Purchase Order

The customer wants to purchase some goods (via the Internet)

1. Customer contacts the supplier and orders the goods
2. Sales department check with credit card department if credit is okay
3. Sales department check with inventory department if the goods are on stock
4. Sales department informs the billing department to bill the customer
5. Sales department informs the shipment department to send out the goods
6. Shipment department sends the goods to customer
7. Shipment department informs the billing department to send the invoice
8. Billing department sends the invoice to the customer

Business Process

- set of logically related tasks
- to achieve a well-defined business outcome
- horizontal view on an organisation
  - by focusing on the activities needed to design / produce / deliver a specific product for the customer


User interface and local desktop applications
Individual activity
Business process
Steps
Process/activity management
Distribution function
Application

adapted from Papazoglou, Web services, 2008

Business Process Characteristics

A process

- may be well-defined, but also ad-hoc
- is widely distributed and possibly customized
    - e.g. can involve serveral departments of a company
        - e.g. billing process
        - e.g. employment process
- based on business rules
    - E.g. ”all money spent requires approval by a superior”
- has a duration that may vary widely
- may contain a set of automated activities besides manual activities
- possibly of transactional nature

Business Processes and Web services

a Web services offer ”services” to be used in business processes

b Business processes offer ”services”

→ Composing Web Services

- A service-oriented architecture produces a lot of (Web) services

→ abstraction similar to procedures/methods

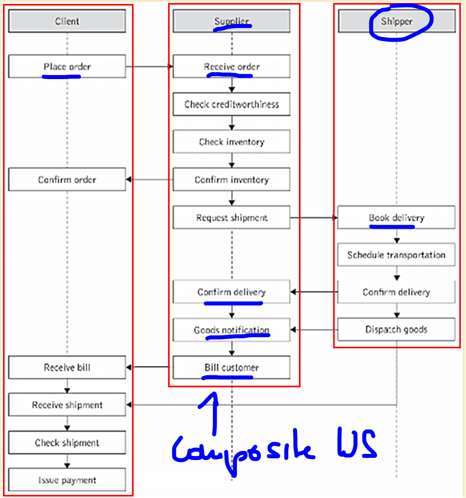
- services are constructed by calling other services (including services constructed out of other services)
- Composition hierarchy

→ Service composition / orchestration

→ Composition can be done in, e.g., Java or special languages for business processes, like BPEL

Example Workflow Results into 3 Composite Web Services

- Supplier composite Web service
  - make order, confirm delivery, goods dispatched
- Client composite Web service
  - confirm order, bill, confirm shipment received
- Shipper composite Web service
  - book delivery



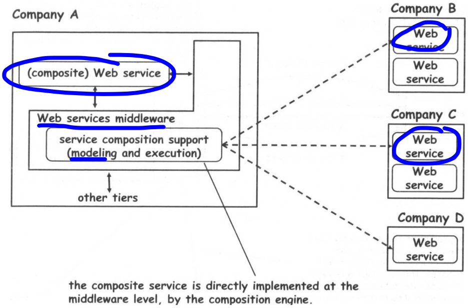
Why special languages for service composition?

- Business process modelling
  - More abstract modelling than programming languages
  - Understandable by managers
  - Sometimes graphical development environments
- Services represent business processes
  - potential long running
    - → several days, weeks
  - transactional
    - → compensation
  - dialog nature
    - → support for sessions
  - complex data / business oriented data
    - → XML
  - Not necessarily sequential
    - → event driven, flow oriented, concurrent activities

→ A programming language with services as procedures

  - BPEL4WS (WS-BPEL), Orc, . . .

Web Service Composition Middleware



Alonso et al. Web Services, 2004

In our case:

- OpenESB standalone server
- OpenESB
  - (ESB = Enterprise Service Bus)
- BPEL (Business Process Execution Language)

Contents

SOAP Encoding

Business Processes

BPEL

BPEL and NetBeans

A BPEL Process as a Web service

Calling other Web services from BPEL

BPEL

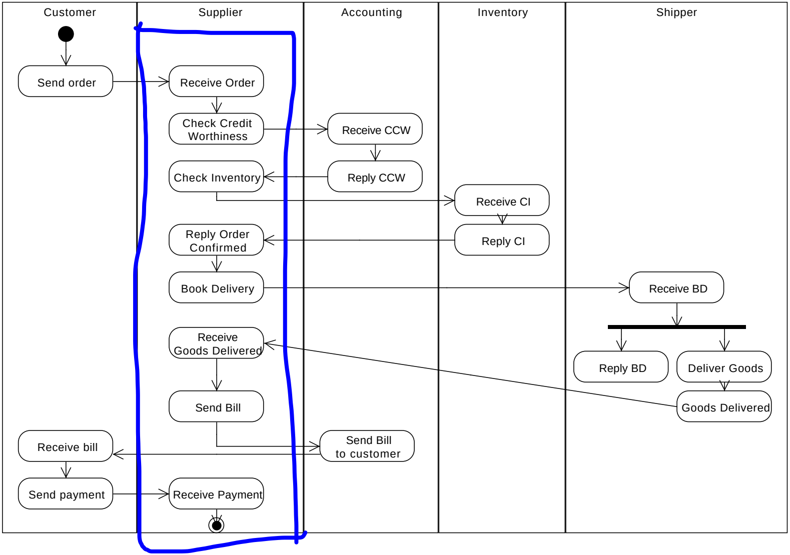
- Business Process Execution Language
- BPEL is a service composition language
- Developed by BEA, IBM and Microsoft in 2002
- Based on WSFL (Web Service Flow Language) and XLANG
- Built on XML and Web services (but has usually a graphical representation)
- BPEL4WS 1.1 `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`
- WS-BPEL 2.0 `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`
  - → Primer is again recommended

Why BPEL for business processes

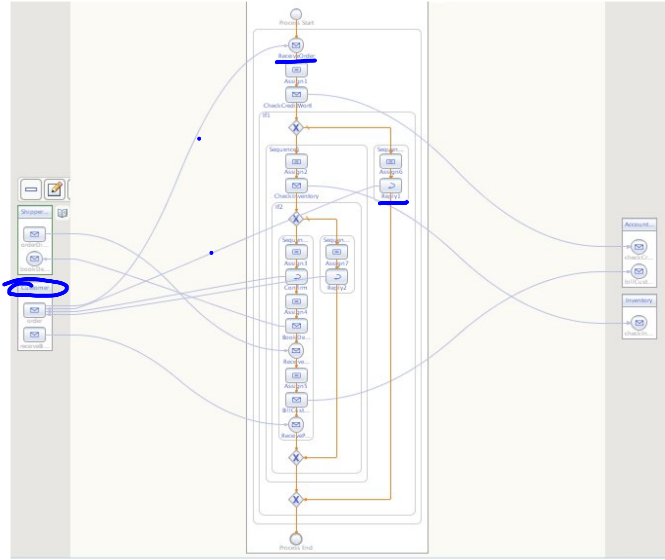
BPEL has support for

- Dialogs (Correlation sets and process states)
- Long running processes
- Exception Handling
- Flows, Concurrency, Event-driven
- Transactions and compensation

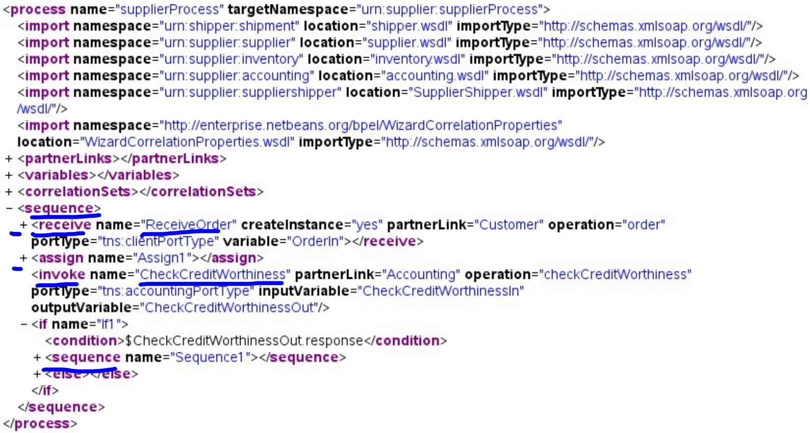
Order business process as activity diagram



Supplier process: Graphical representation



Supplier process: BPEL file



```
<process name="supplierProcess" targetNamespace="urn:supplier:supplierProcess">
  <import namespace="urn:shipper:shipment" location="shipper.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:supplier:supplier" location="supplier.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:supplier:inventory" location="inventory.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:supplier:accounting" location="accounting.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
  <import namespace="urn:supplier:suppliershipper" location="SupplierShipper.wsdl" importType="http://schemas.xmlsoap.org
  /wsdl/"/>
  <import namespace="http://enterprise.netbeans.org/bpel/WizardCorrelationProperties"
  location="WizardCorrelationProperties.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
+ <partnerLinks></partnerLinks>
+ <variables></variables>
+ <correlationSets></correlationSets>
- <sequence>
  + <receive name="ReceiveOrder" createInstance="yes" partnerLink="Customer" operation="order"
    portType="tns:clientPortType" variable="OrderIn"></receive>
  + <assign name="Assign1"></assign>
    <invoke name="CheckCreditWorthiness" partnerLink="Accounting" operation="checkCreditWorthiness"
    portType="tns:accountingPortType" inputVariable="CheckCreditWorthinessIn"
    outputVariable="CheckCreditWorthinessOut"/>
  - <if name="If1">
      <condition>$CheckCreditWorthinessOut.response</condition>
    + <sequence name="Sequence1"></sequence>
    + <else></else>
    </if>
  </sequence>
</process>
```

Contents

SOAP Encoding

Business Processes

BPEL

BPEL and NetBeans

A BPEL Process as a Web service

Calling other Web services from BPEL

Creating and Running a BPEL Process

- We use NetBeans to create BPEL processes and OpenESB standalone server to execute BPEL procsses
  - Alternatives are Glassfish, Tomcat and Active BPEL, Oracle BPEL Designer, . . .
- Instructions on ”How to work with BPEL processes in NetBeans” are on the Course Web page

Basic process for creating and running a BPEL process using NetBeans

1 Create a BPEL Module project → SOA
   - 1.a) Create XSD files (if necessary)
   - 1.b) Create WSDL files
   - 1.c) Create BPEL file

2 Create a Composite Application project
   - 2.a) Drag and drop BPEL Module project on Service Assembly
   - 2.b) Deploy the composite application
     - Possible error message: Glassfish does not support BPEL: Choose the OpenESB standalone server in project properties of the composite application

3 Test the Composite Application
   - 3.a) Use a JUnit test (i.e. write a Web service client against the WSDL file)
   - 3.b) Or use the test feature of a Composite Application Project

Contents

SOAP Encoding

Business Processes

BPEL

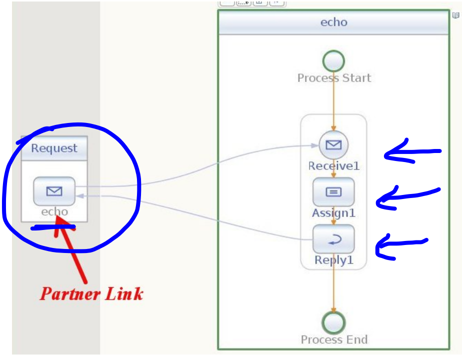
BPEL and NetBeans

A BPEL Process as a Web service

Calling other Web services from BPEL

BPEL process: Receiving requests and replying

- Example: Echo BPEL process
  - Create a simple process that takes a string as an argument and returns the same string

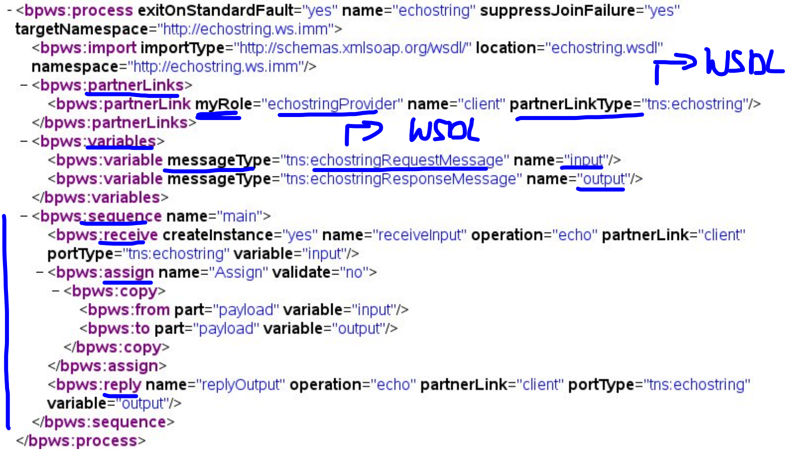


Process shown using the BPEL Designer from NetBeans

A Simple Example: Echo String

- A process consists of several files
  - BPEL file containing the process .bpel
    - The behaviour of the Web service
  - WSDL file containing the interface to the BPEL process .wsdl
  - Possible other WSDL files for services the BPEL process uses

EchoString process: BPEL file

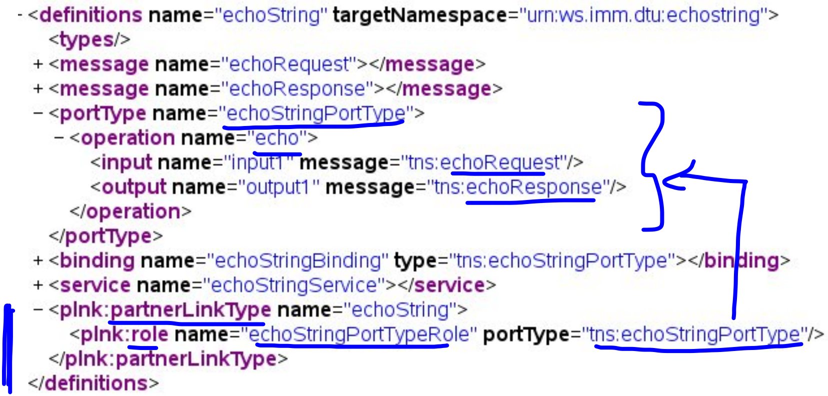


```
- <bpws:process exitOnStandardFault="yes" name="echostring" suppressJoinFailure="yes"
  targetNamespace="http://echostring.ws.imm">
    <bpws:import importType="http://schemas.xmlsoap.org/wsdl/" location="echostring.wsdl"
    namespace="http://echostring.ws.imm"/>
  - <bpws:partnerLinks>
      <bpws:partnerLink myRole="echostringProvider" name="client" partnerLinkType="tns:echostring"/>
    </bpws:partnerLinks>
  - <bpws:variables>
      <bpws:variable messageType="tns:echostringRequestMessage" name="input"/>
      <bpws:variable messageType="tns:echostringResponseMessage" name="output"/>
    </bpws:variables>
  - <bpws:sequence name="main">
      <bpws:receive createInstance="yes" name="receiveInput" operation="echo" partnerLink="client"
      portType="tns:echostring" variable="input"/>
    - <bpws:assign name="Assign" validate="no">
      - <bpws:copy>
          <bpws:from part="payload" variable="input"/>
          <bpws:to part="payload" variable="output"/>
        </bpws:copy>
      </bpws:assign>
      <bpws:reply name="replyOutput" operation="echo" partnerLink="client" portType="tns:echostring"
      variable="output"/>
    </bpws:sequence>
</bpws:process>
```

→ WSDL

→ WSDL

EchoString process: WSDL file



```
- <definitions name="echoString" targetNamespace="urn:ws.imm.dtu:echostring">
    <types/>
  + <message name="echoRequest"></message>
  + <message name="echoResponse"></message>
  - <portType name="echoStringPortType">
    - <operation name="echo">
        <input name="input1" message="tns:echoRequest"/>
        <output name="output1" message="tns:echoResponse"/>
      </operation>
    </portType>
  + <binding name="echoStringBinding" type="tns:echoStringPortType"></binding>
  + <service name="echoStringService"></service>
  - <plnk:partnerLinkType name="echoString">
      <plnk:role name="echoStringPortTypeRole" portType="tns:echoStringPortType"/>
    </plnk:partnerLinkType>
  </definitions>
```

A Simple Example: Echo String (II)

- The elements of a WSDL file
  - Interface
  - The usual WSDL elements (types, messages, port types, . . .)
  - plus partner link types
- The elements of the simple example BPEL file
  - Process / behaviour
  - partner links maintain the connection to the caller and are instances of the partner link types defined in the WSDL
  - variables for the request message and the response message
  - A sequence of three actions:
    - receive a request
    - Using assign to copy the input to the output
    - reply a response

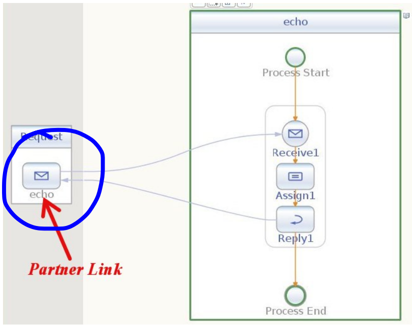
Activity: Receive

- Takes a request from a client via a partner link stores it in a variable
- The request comes via the operation of the indicated port type (has to coincide with the port type of the partner link type of which the partner link is an instance)

```
<receive name="Receive"
   createInstance="yes"
   partnerLink="Request"
   operation="echo" xmlns:tns="urn:echo"
   portType="tns:Echo"
   variable="EchoIn"/>
```

- Each process has to start with a receive activity (where `createInstance="yes"`)
- If the request has the request-response type, then the process has to have a corresponding reply activity

Partner Link and Partner Link Type for the Echo BPEL process



Partner Link BPEL

```
<partnerLink name="Request"
  xmlns:tns="urn:echo"
  partnerLinkType="tns:echo"
  myRole="EchoRole"/>
```

Partner Link Type WSDL

```
<plnk:partnerLinkType name="echo">
    <plnk:role name="EchoRole"
       portType="tns:Echo"/>
</plnk:partnerLinkType>
```

Activity: Reply

- The reply replies to a previous request (coming in via a receive activity)
- The reply needs to use the same partner link, port type, and operation as the receive activity

```
<reply name="Reply1"
   partnerLink="Request"
   operation="echo" xmlns:tns="urn:echo"
   portType="tns:Echo"
   variable="EchoOut"/>
```

Activity: Assign

- Assignment in the echo.bpel file

```
<assign name="Assign1">
   <copy>
      <from variable="EchoIn" part="input"/>
      <to variable="EchoOut" part="output"/>
   </copy>
</assign>
```

- Variable declaration in the echo.bpel file

```
<variable name="EchoIn" xmlns:tns="urn:echo"
   messageType="tns:echoRequest"/>
<variable name="EchoOut" xmlns:tns="urn:echo"
   messageType="tns:echoResponse"/>
```

- Message declaration in the echo.wsdl file

```
<message name="echoRequest">
    <part name="input" type="xsd:string"/>
</message>
<message name="echoResponse">
    <part name="output" type="xsd:string"/>
</message>
```

from-spec

Assignment

$$Var_{\underline{to}-spec} := \underline{Exp}_{\underline{from}-spec}$$

- Most important from-spec types:
  - Variable (containing messages / XML / literals)
  - Expression (using XPath 1.0)
  - Literal values
  - ...

from-spec

```
<from variable="ncname" part="ncname"? query="expr"? />
<from expression="general-expr"/>
<from><literal> ... literal value ... </literal></from>
...
```

to-spec

```
<to variable="ncname" part="ncname"? query="expr"?/>
...
```

Expression Language

- Can be defined in the process element; default is XPath 1.0
- XPath 1.0
  - Access to elements
  - Access to attributes
  - Expressions
    - Datatypes
    - Operations on datatypes
  - Resources
    - `http://www.w3.org/TR/1999/REC-xpath-19991116`
    - XPath Tutorial: `http://www.zvon.org/xxl/XPathTutorial/Output/`
    - Oracle Tutorial on assignments: `http://www.oracle.com/technology/products/ias/bpel/pdf/orabpel-Tutorial3-DataManipulationTutorial.pdf`

Contents

SOAP Encoding

Business Processes

BPEL

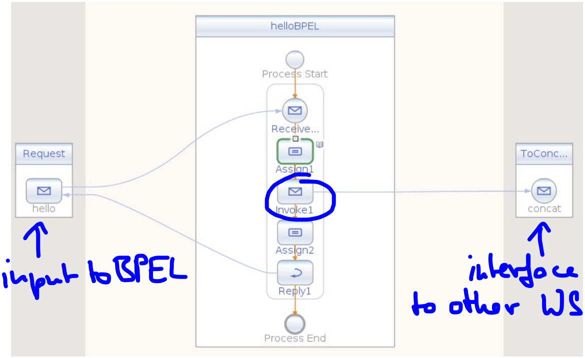
BPEL and NetBeans

A BPEL Process as a Web service

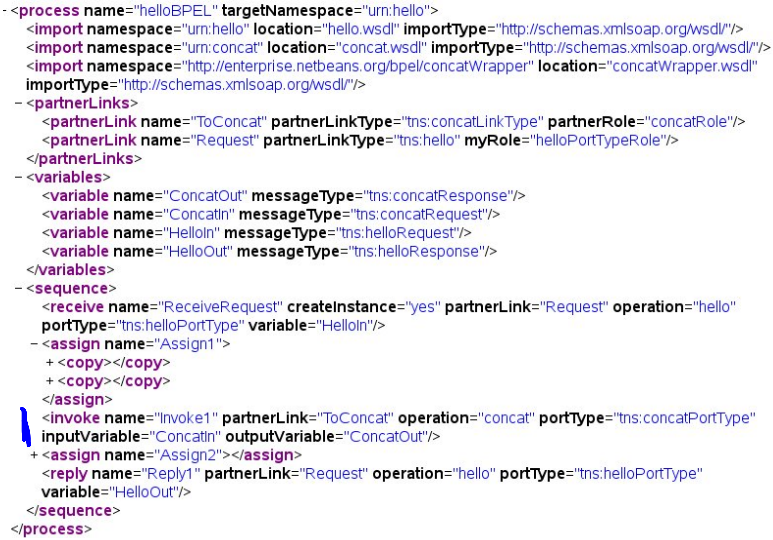
Calling other Web services from BPEL

Calling Web services from a BPEL process

- Example: Hello Service
  - Create a process that takes a string as an arguments and prepends ”Hello ” to it by using a generic concatenation service.
  - Principle sequence: receive request; invoke concat service; reply to request

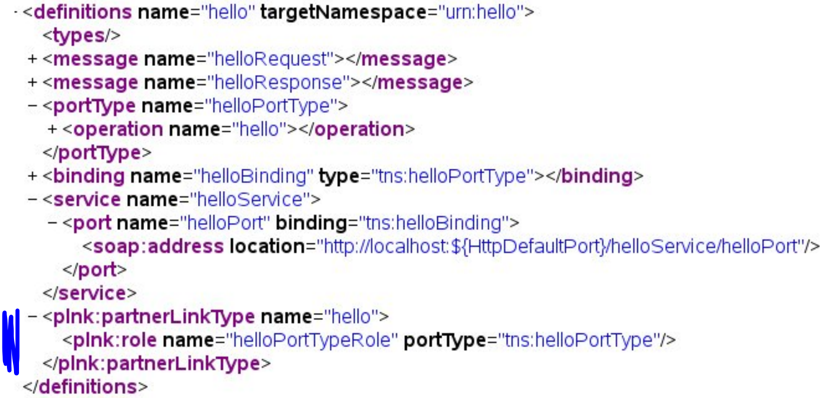


Hello Service BPEL process



```
- <process name="helloBPEL" targetNamespace="urn:hello">
    <import namespace="urn:hello" location="hello.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="urn:concat" location="concat.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/concatWrapper" location="concatWrapper.wsdl"
    importType="http://schemas.xmlsoap.org/wsdl/"/>
  - <partnerLinks>
      <partnerLink name="ToConcat" partnerLinkType="tns:concatLinkType" partnerRole="concatRole"/>
      <partnerLink name="Request" partnerLinkType="tns:hello" myRole="helloPortTypeRole"/>
    </partnerLinks>
  - <variables>
      <variable name="ConcatOut" messageType="tns:concatResponse"/>
      <variable name="ConcatIn" messageType="tns:concatRequest"/>
      <variable name="HelloIn" messageType="tns:helloRequest"/>
      <variable name="HelloOut" messageType="tns:helloResponse"/>
    </variables>
  - <sequence>
      <receive name="ReceiveRequest" createInstance="yes" partnerLink="Request" operation="hello"
      portType="tns:helloPortType" variable="HelloIn"/>
    - <assign name="Assign1">
       + <copy></copy>
       + <copy></copy>
      </assign>
      <invoke name="Invoke1" partnerLink="ToConcat" operation="concat" portType="tns:concatPortType"
      inputVariable="ConcatIn" outputVariable="ConcatOut"/>
    + <assign name="Assign2"></assign>
      <reply name="Reply1" partnerLink="Request" operation="hello" portType="tns:helloPortType"
      variable="HelloOut"/>
    </sequence>
  </process>
```

Hello Service: Hello Service WSDL



```
- <definitions name="hello" targetNamespace="urn:hello">
    <types/>
  + <message name="helloRequest"></message>
  + <message name="helloResponse"></message>
  - <portType name="helloPortType">
    + <operation name="hello"></operation>
    </portType>
  + <binding name="helloBinding" type="tns:helloPortType"></binding>
  - <service name="helloService">
    - <port name="helloPort" binding="tns:helloBinding">
        <soap:address location="http://localhost:${HttpDefaultPort}/helloService/helloPort"/>
      </port>
    </service>
  - <plnk:partnerLinkType name="hello">
      <plnk:role name="helloPortTypeRole" portType="tns:helloPortType"/>
    </plnk:partnerLinkType>
</definitions>
```

Finding the value of ${HttpDefaultPort}

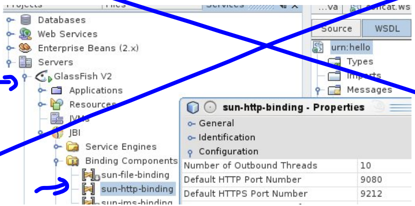
```
<service name="helloService">
 <port name="helloPort" binding="tns:helloBinding">
  <soap:address
   location="http://localhost:${HttpDefaultPort}/helloService/helloPo
 </port>
</service>
```

2

9080

? wsdl

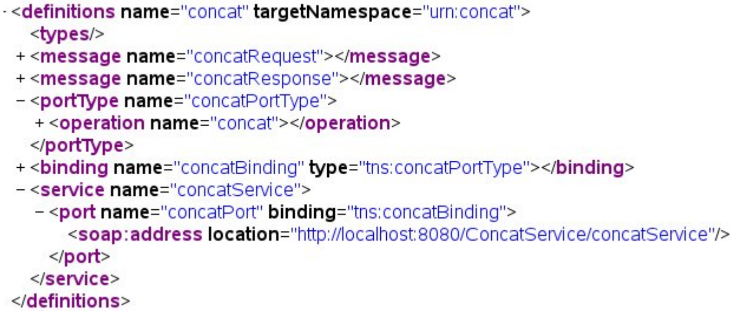
- ~~Can contain `$HttpDefaultPort` in soap:address, but can also be replaced by a *fixed port* (≠ 8080!)~~



Number of Outbound Threads	10
Default HTTP Port Number	9080
Default HTTPS Port Number	9212

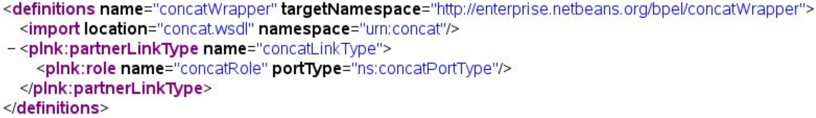
Hello Service: Concat and ConcatWrapper WSDL

- Concat WSDL



```
- <definitions name="concat" targetNamespace="urn:concat">
    <types/>
  + <message name="concatRequest"></message>
  + <message name="concatResponse"></message>
  - <portType name="concatPortType">
    + <operation name="concat"></operation>
    </portType>
  + <binding name="concatBinding" type="tns:concatPortType"></binding>
  - <service name="concatService">
    - <port name="concatPort" binding="tns:concatBinding">
        <soap:address location="http://localhost:8080/ConcatService/concatService"/>
      </port>
    </service>
  </definitions>
```

- Concat Wrapper WSDL



```
<definitions name="concatWrapper" targetNamespace="http://enterprise.netbeans.org/bpel/concatWrapper">
   <import location="concat.wsdl" namespace="urn:concat"/>
 - <plnk:partnerLinkType name="concatLinkType">
      <plnk:role name="concatRole" portType="ns:concatPortType"/>
   </plnk:partnerLinkType>
</definitions>
```

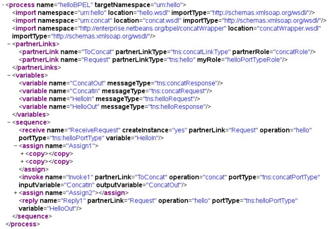
Invoke Activity

Send a message via a partner link to another Web service

- Wait for an answer if output variable is present (synchronous call)
- If output variable is not present, don't wait for an answer (asynchronous call)

```
<invoke name="Invoke1"
  partnerLink="ToConcat"
  operation="concat"
  xmlns:tns="urn:concat"
  portType="tns:concatPortType"
  inputVariable="ConcatIn"
  outputVariable="ConcatOut"/>
```

Partner Link and Partner Link Type for the Hello BPEL process



```
- <process name="helloBPEL" targetNamespace="urn:hello">
    <import namespace="urn:hello" location="hello.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="urn:concat" location="concat.wsdl" importType="http://schemas.xmlsoap.org/wsdl/"/>
    <import namespace="http://enterprise.netbeans.org/bpel/concatWrapper" location="concatWrapper.wsdl"
    importType="http://schemas.xmlsoap.org/wsdl/"/>
  - <partnerLinks>
      <partnerLink name="ToConcat" partnerLinkType="tns:concatLinkType" partnerRole="concatRole"/>
      <partnerLink name="Request" partnerLinkType="tns:hello" myRole="helloPortTypeRole"/>
    </partnerLinks>
  - <variables>
      <variable name="ConcatOut" messageType="tns:concatResponse"/>
      <variable name="ConcatIn" messageType="tns:concatRequest"/>
      <variable name="HelloIn" messageType="tns:helloRequest"/>
      <variable name="HelloOut" messageType="tns:helloResponse"/>
    </variables>
  - <sequence>
      <receive name="ReceiveRequest" createInstance="yes" partnerLink="Request" operation="hello"
      portType="tns:helloPortType" variable="HelloIn"/>
    - <assign name="Assign1">
       + <copy></copy>
       + <copy></copy>
      </assign>
      <invoke name="Invoke1" partnerLink="ToConcat" operation="concat" portType="tns:concatPortType"
      inputVariable="ConcatIn" outputVariable="ConcatOut"/>
    + <assign name="Assign2"></assign>
      <reply name="Reply1" partnerLink="Request" operation="hello" portType="tns:helloPortType"
      variable="HelloOut"/>
    </sequence>
  </process>
```

Partner Link

```
<partnerLink name="ToConcat"
  partnerLinkType="tns:concatLinkType"
  partnerRole="concatRole"/>
```

Partner Link Type

```
<plnk:partnerLinkType name="concatLinkType"
    <plnk:role name="concatRole"
       portType="ns:concatPortType"/>
</plnk:partnerLinkType>
```