

Developing BPEL processes

Second Part: BPEL Basic Concepts and Examples



Table of contents

- Learning BPEL by examples
- Oracle BPM and BPEL 2.0
- BPEL:PartnerLink
- BPEL:Receive
- BPEL:Reply
- BPEL:Invoke
- A word about BPEL variable
- A word about BPEL variable (2)
- BPEL:Assign
 - Basic variable manipulation: PlusOneBasic
 - Exercise: PlusOne
 - Basic web service invocation: GetBibleBooksTitles
 - Exercise: CurrencyConvertor
- BPEL:Switch
- BPEL:While
 - A more complicated example: Factorial
 - Exercise: ImprovedFactorial
 - Homework: IterativeFactorial



Learning BPEL by examples

- A list of examples explaining BPEL core concepts
 - Web Services related activities
 - Communication to / from BPEL process
 - Variable declaration & editing
 - Flow control activities
 - Most examples built “on the fly”
- Then, some exercises
 - Solutions available online (after the end of the lesson)

Oracle BPM and BPEL 2.0

- A word of caution:
 - Oracle BPM doesn't *fully* support BPEL 2.0 specs
 - It supports BPEL4WS 1.1 specs only
- There could be small differences
 - Some activities have different names (Switch → If, Termination → Exit)
 - Some are missing (RepeatUntil, CompensateScope, TerminationHandlers, ForEach, Rethrow)
 - However, same features could be achieved through proprietary Oracle extensions (example: FlowN instead of ForEach)
 - Others can be easily reimplemented (example: RepeatUntil via While)
- Only feature actually not available: TerminationHandler



BPEL:PartnerLink

- PartnerLink elements identify business process interaction partners
 - A process must have at least one Plink
 - Each Plink has one or two PlinkType and Role
- PartnerLinkTypes and Roles are defined in the service WSDL file
 - But you don't have to modify original WSDL
 - Just use a “wrapper” WSDL instead
- PartnerLinks defines process' “access points”



BPEL:Receive

- The Receive element blocks the business process execution until a certain message has arrived
- Need to specify:
 - PLink
 - PortType
 - Operation
 - An input variable
- Each process starts with a Receive
 - With createinstance="yes"



BPEL:Reply

- This activity's goal is to return a message from the process to the partner that started the conversation
- Receive + Reply activities = request-response operation
 - Synchronous / Asynchronous
- Plink, PortType and Operation MUST match those of corresponding “Reply”
- May have an output variable
- A Reply activity closes a process by answering the requester



BPEL:Invoke

- The Invoke activity starts a one-way or request-response operation
- It allows business processes to send messages to partners
 - Synchronously / Asynchronously
- Need to specify:
 - PLink
 - PortType
 - Operation
 - Associated WSDL's namespace
- Optionally:
 - Input / Output variable for message exchanging

A word about BPEL variable

- They don't differ much from Java variable
- Their main tasks:
 - Holding temporary values
 - Composing expressions
 - Passing input/output parameters to external partners
- They are **typed**:
 - WSDL message types
 - XML Schema - defined types
 - XML built-in types (string, int, ...)

A word about BPEL variable (2)

- **But:**
 - BPEL variable needs to be initialized before use
 - (If involved) are automatically updated by Web Service related activities (Receive, Reply, Invoke)
 - Store incoming / outgoing messages
 - Content accessible only through Assign activity

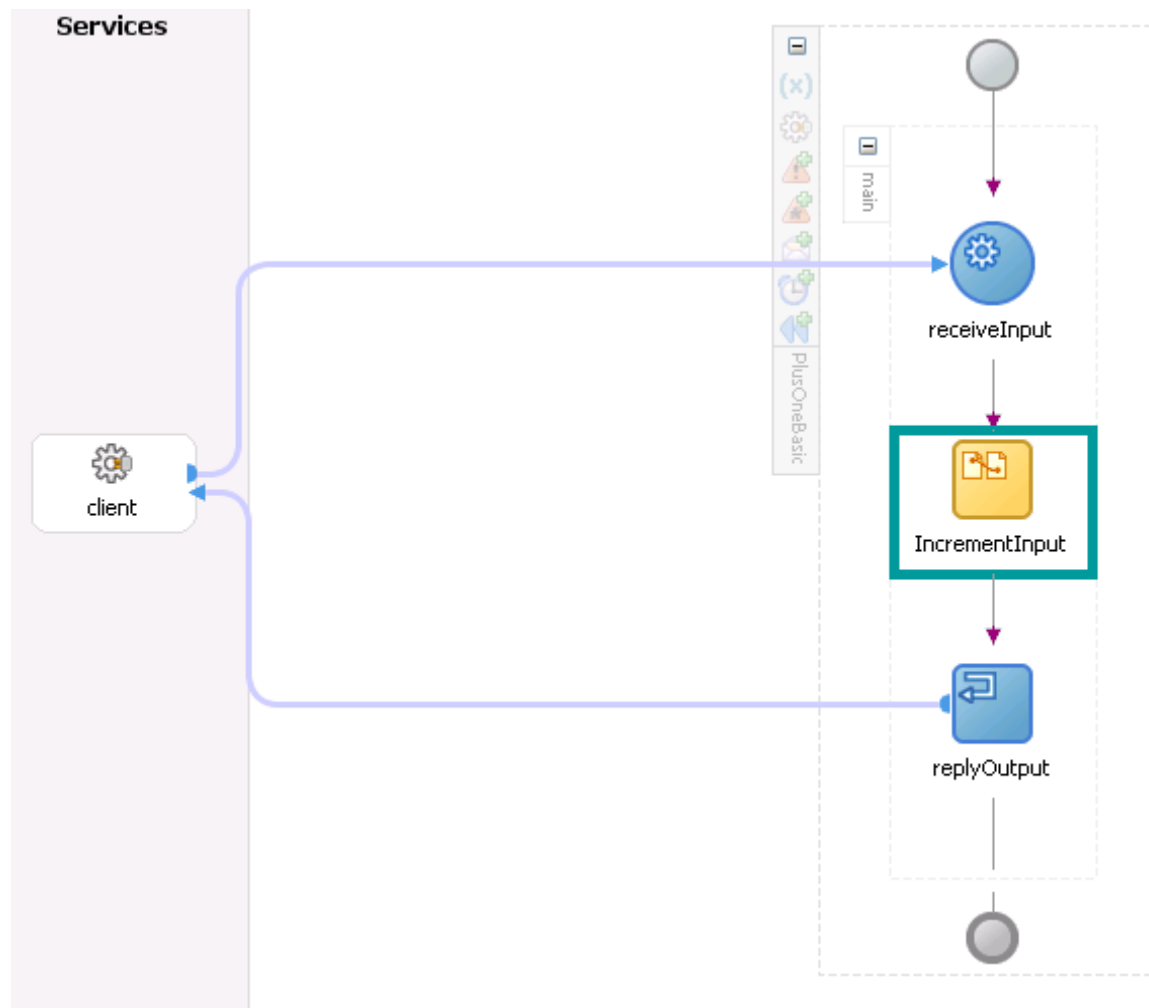


initializeInputVariable

BPEL:Assign

- To read / modify variable content, you have to use the Assign activity
- Assign copies data from
 - Variables
 - Expressions
 - Constants
 - Messages parts
- To variables
- Each Assign can contain more than one “copy” part

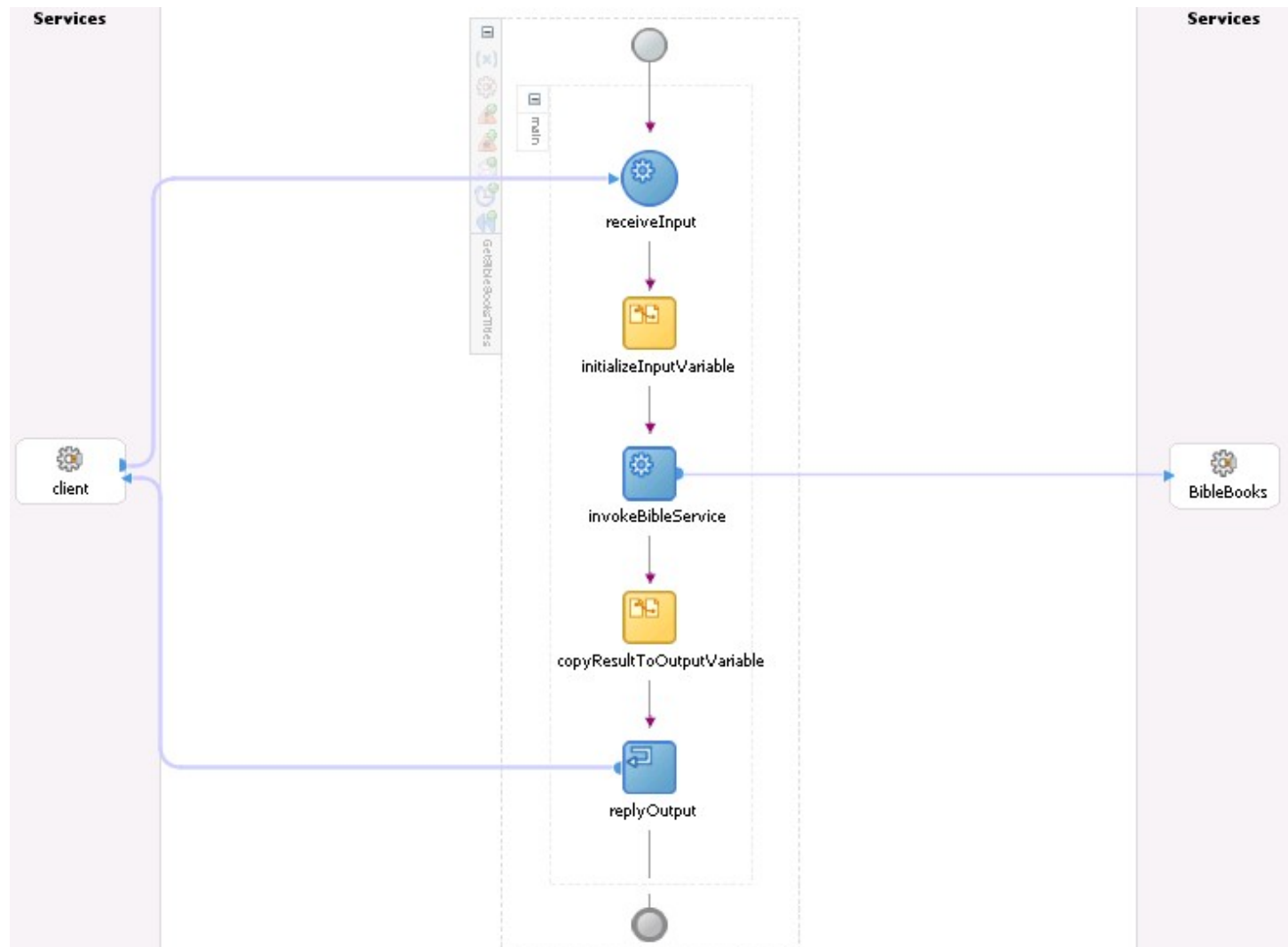
Basic variable manipulation: PlusOneBasic



Exercise: PlusOne

- Works just like PlusOneBasic
 - But takes an integer as input and returns a string
 - Example: if user enters 5, it returns “5 + 1 = 6”
 - You need to change input / output messages type definition in the schema
 - You can do everything in a single Assign activity
 - Or split it into two different activity
 - Or a single Assign with two copy parts
 - In the last two cases, you may want to define a temporary variable to help you doing the assignment

Basic web service invocation: GetBibleBooksTitles



Exercise: CurrencyConvertor

- We would like to convert from Euro to US Dollars a certain amount of money
 - CurrencyConvertorBPEL services will tell the current currency rate from Euro to US Dollars
 - Not very different from the previous case
 - CurrencyConvertor service WSDL address is:
<http://www.webservices.net/CurrencyConvertor.asmx?wsdl>
 - Requires “FromCurrency” and “ToCurrency”
 - Set them to “EUR” and “USD” respectively
 - Process input and output must be of type double



BPEL:Switch

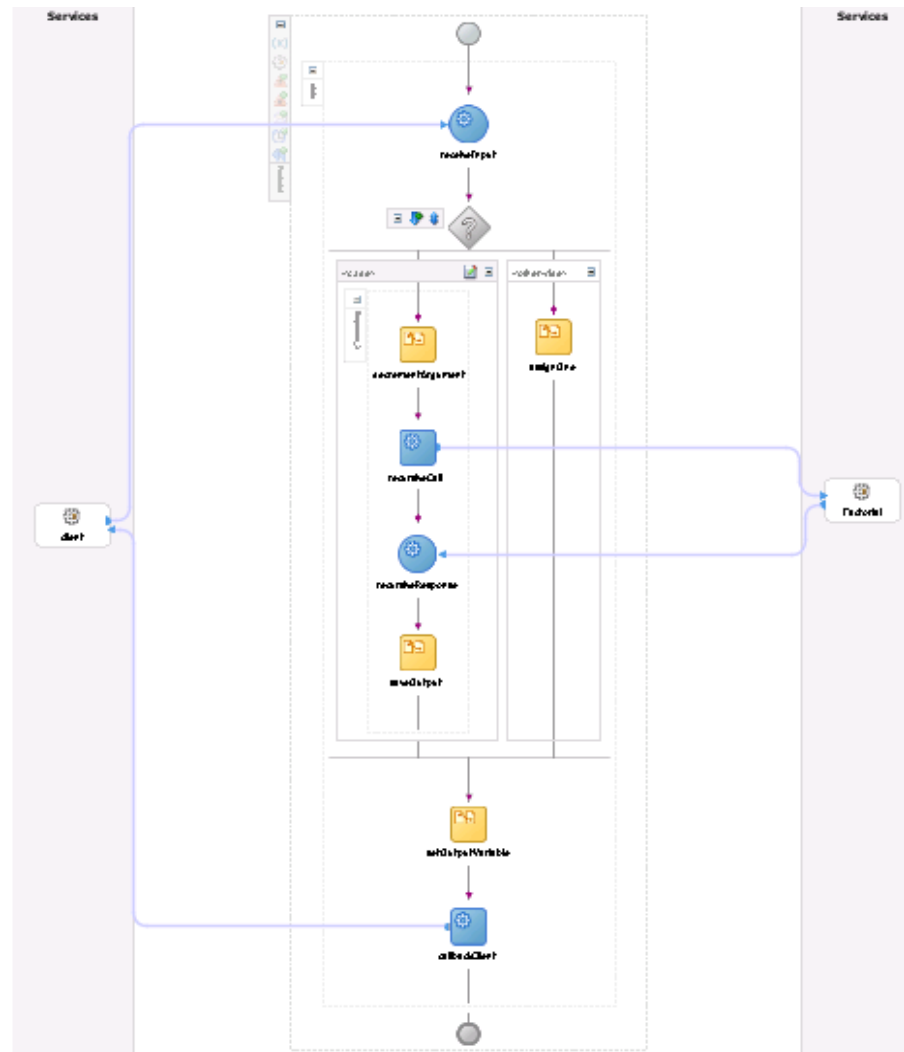
- Define conditional behavior
- Work like Java counterparts
- Requires to define a condition
 - If condition is true, corresponding branch is executed
 - Otherwise, the other branch is executed
- Can be nested



BPEL:While

- The While element allows you to repeat a one or more activities as long as a specified condition is true
- Two mandatory parts:
 - Condition
 - Loop's body (any activity)
- Behavior is just like corresponding Java instruction
 - But this one doesn't automatically define a scope

A more complicated example: Factorial



Exercise: ImprovedFactorial

- Factorial process doesn't complain when inserting negative numbers
 - Create a new process, which receives strings as input
 - Also outputs strings
 - Receives the number (as a string) and returns an error message if it's negative
 - Otherwise, just invoke the Factorial process to get the result and then return it to the user
 - Remember: Factorial is a BPEL process, and therefore also a WebService!
 - When defining the PartnerLink, click on the torch light and then under BPEL services to find the instance of Factorial deployed on your machine

Homework: Iterative Factorial

- Reimplement the Factorial exercise, this time using iteration instead of recursion
 - Use While activity
 - A suggestion: define a temporary variable for holding intermediate results
 - Be careful when specifying condition, to avoid infinite loops