SOAの中核技術としてのBPEL入門（2）

BPELでの変数の定義と代入

丸山不二夫 | （稚内北星学園大学）

今回は複数の Web サービスを統合してビジネスプロセスを記述するための Web サービス標準，WS-BPEL（Web Services Business Process Execution Language. 以下「BPEL」）の解説の 2 回目です（**図 -1**）．前回は，BPEL でどのようにサービスを結合するのかを，サービス間の関係を抽象的に表現する PartnerLink という概念を中心に説明してきました．今回は，BPEL での変数の宣言と値の代入方法を中心に，実行可能なプログラム言語としての BPEL の特徴を解説したいと思います．

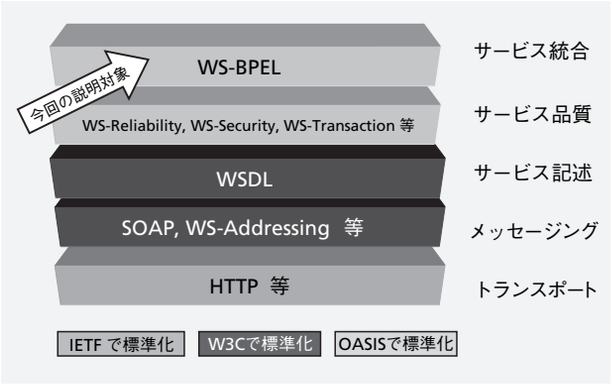


図 -1　WS-BPEL の位置付け

BPEL言語の構造

前回，その一端を見たのですが，BPEL は XML を用いてプログラムを記述します．最初に，リスト 1 に，BPEL プログラムの基本構造を示します．

リスト 1　BPEL プログラムの基本構造

```
<process>
   <!-- ①パートナーリンク定義：
        Definition and roles of process participants -->
   <partnerlinks> ... </partnerlinks>
   <!-- ②プロセス変数定義：
        Data/state used within the process-->
   <variables> ... </variables>
   <!-- ③メッセージ相関定義：
        Properties that enable conversations -->
   <correlationSets> ... </correlationSets>

   <!-- ④ハンドラ定義：Exception handling -->
   <faultHandlers> ... </faultHandlers>
   <!-- Error recovery ? undoing actions -->
   <compensationHandlers> ... </compensationHandlers>
   <!-- Concurrent events with process itself -->
   <eventHandlers> ... </eventHandlers>
   <!-- ⑤アクティビティ定義 :Business process flow  -->
      (activities)*
</process>
```

BPEL プログラムでは process 要素でプログラム全体を束ねます．リスト 1 ①の partnerlinks 要素には，前回説明した PartnerLink の定義により外部パートナー Web サービスとの関係を記述します．②の variables 要素は，BPEL プロセス内部のデータや状態を表す変数です．BPEL では，変数は Web サービスによりネットワーク上でやりとりされるメッセージと関係付ける必要があります．これについては，後で少し詳しく説明します．③ correlationSets 要素は，WS-Addressing の技術を用いて，非同期のプロセス間の通信を定義しているのですが，今回は説明を割愛します．④のハンドラ定義部分には faultHandlers（例外ハンドラ），compensationHandlers（補償ハンドラ），eventHandlers（イベントハンドラ）という 3 種類の Handler 要素を定義することができます．これらの Handler の働きについては，次回に説明をします．

BPEL プログラムの本体部分は，リスト⑤のアクティビティ定義の部分以降に記述します．BPEL では，具体的な動作を持つプログラムの単位をアクティビティ（activity）と呼ぶのですが，前回見た，receive，invoke は，Web サービスを通じてメッセージをやりとりする，BPEL の代表的な（基本）アクティビティです．そのほかにも，BPEL プログラムの流れを制御する，while や switch といったいくつかの（構造化）アクティビティがあります．代表的なアクティビティを**表 -1** に整理します．BPEL の制御構造（構造化アクティビティ）についての説明は次回に行います．

要素名	説明
基本アクティビティ（プリミティブアクティビティ）一覧	
<invoke>	パートナー Web サービスの呼び出し
<receive>	メッセージの受信待ち
<reply>	<receive> アクティビティで受信したメッセージに対する応答
<assign>	プロセス変数への値の代入，XPath 式を使ったプロセス変数の更新 <copy>, <from>, <to> の子要素を用いて値を代入する
<throw>	例外発生の通知用
<terminate>	プロセス終了用
<wait>	指定期間または指定日時までの待機を記述する
<empty>	並行して実行するプロセスの同期化等に利用する
<compensate>	すでに正常に終了したプロセスを元に戻す補償処理用
構造化されたアクティビティー覧	
<flow>	並行性および同期を提供するアクティビティ
<scope>	複数のアクティビティをまとめて有効範囲を指定するためのブロック定義用アクティビティ
<while>	条件ループ処理
<sequence>	順次処理
<switch>	選択処理
<pick>	メッセージの受信待ちと指定期間・日時までの待機を同時に実行するブロック定義．<receive>, <wait> と組み合わせることができる

表 -1　BPEL の主要なアクティビティー覧

BPELの変数の特徴

ビジネスプロセスは，当然，状態を持ったリソースを対象にします．ビジネスプロセスの「状態」を表現するには，「状態変数」が必要になります．BPEL の variable（プロセス変数定義）は，そのためのものです．状態を表す「変数」だけでなく，プロセスをコントロールする上で状態の値を操作することを考えると，他のプログラム言語と同じように，変数からなる「式」や変数への「代入（assign）」という概念が自然と生まれてきます．

BPEL のデータの扱いで，特徴的なのは，「変数」が基本的にはプロセス間で Web サービスを通じて交換されるメッセージに対応していることです．Web サービスのメッセージのかたちは，WSDL で規定されていますので，BPEL では，変数の型の宣言で WSDL でのメッセージの型の定義を参照することがあります．

BPEL 変数の値は，XML で表現されています．ですから，変数から値を取り出すときに，XPath 等の XML でよく知られている Query 言語を用いると，さまざまなデータ操作が可能になります．

ある BPEL プログラムでの変数定義のサンプルを見てみましょう（リスト 2）．この BPEL プログラムは，ユーザが車のローン loanApplication を申し込むと，まず，ユーザがローンを組めるかどうかの信用チェック CreditRating を行い，複数の金融機関に問合せを行って，最も条件のいいローンを組んでくれるというものです．リスト 2 では，4 つの変数が定義されています．変数名は，variable 要素内の name 属性で指定されています．以下では，「○○という name 属性を持つ要素」を「○○という名前を持つ要素」と，簡略化して言うことにしましょう．たとえば，「input という名前の variable」のようになります．

この変数定義で注意してほしいのは，ここではすべての variable の定義に登場している messageType 属性です．この属性の値は，実は，サービスを記述する WSDL の message 節の名前を参照しています．次の章で，もう少し詳しく，この messageType と WSDL との関係を説明しましょう．

変数の定義には，messageType のほかに，type や element を使うこともできるのですが，今回は説明を省略します．

リスト 2　BPEL での変数定義のサンプル

```
<variables>
   <variable name="input"
    messageType="tns:LoanServiceRequestMessage"/>
   <variable name="crInput"
    messageType=
     "services:CreditRatingServiceRequestMessage"/>
   <variable name="crOutput"
    messageType=
     "services:CreditRatingServiceResponseMessage"/>
   <variable name="output"
    messageType="tns:LoanServiceResponseMessage"/>
</variables>
```

BPELの変数とWSDLの関係

リスト 2 の variable 定義で，input という名前の変数が参照している WSDL の一部をリスト 3 に示します．リスト 3 で次の 1 から 5 までを確認し，BPEL の変数が，対応する Web サービスのメッセージやインタフェースが定義されている WSDL と関係していることを理解してください．

1. まず，portType 節を見てください．LoanService という名前の portType 要素中の，initiate という operation 要素中の input 要素の messeage 属性が，LoanServiceRequestMessage という名前であることを確認してください．

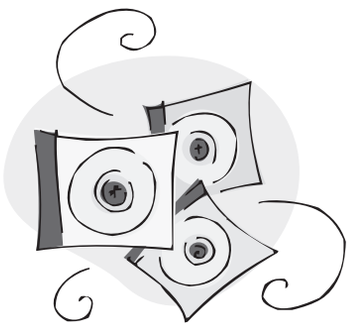
2. 次いで，message 節を見ます．先に見た LoanService RequestMessage という名前を持つ message 節がありますね．リスト 2 の variable 定義で参照されたのは，この名前です．また，この message 要素中の part 要素内の element 属性が，loanApplication であることを確認してください．

3. 今度は，types 節です．この中の schema 定義中に，loanApplication という名前を持つ element 要素があることと，この要素の type 属性が LoanApplication Type であることを確認してください．

4. 同じく types 節の schema 定義中に，LoanApplication Type という名前を持つ complexType 要素があることを確認してください．この中の sequence 要素が，具体的なメッセージのかたちを規定しています．

5. 最後に，BPEL に対応した WSDL の拡張である，LoanService という名前を持つ partnerLinkType 節を見てください．ここの LoanServiceProvider という名前の role 要素中に LoanService という名前の portType 要素があるのを確認してください．この portType は，1 で見た portType にほかなりません．

少し，面倒だったかもしれません．ただ，WSDL を「読む」ためには，こうした名前間の連関が存在していることを知ることが，重要な予備知識になります．



リスト 3　LoanService.wsdl（部分）

```
<definitions name="LoanService" .....>

    <types>
        <schema>
          ......
          ......
          <element name="loanApplication"
                   type="s1:LoanApplicationType"/>
          <complexType name="LoanApplicationType">
              <sequence>
                <element name="SSN" type="string"/>
                <element name="email" type="string"/>
                <element name="customerName"
                         type="string"/>
                <element name="loanAmount"
                         type="double"/>
                <element name="carModel"
                         type="string"/>
                <element name="carYear"
                         type="string"/>
                <element name="creditRating"
                         type="int"/>
              </sequence>
          </complexType>
        </schema>
    </types>

    <message name="LoanServiceRequestMessage">
        <part name="payload"
              element="s1:loanApplication"/>
    </message>
    ......

    <portType name="LoanService">
        <operation name="initiate">
            <input message=
              "tns:LoanServiceRequestMessage"/>
        </operation>
    </portType>
    .....

    <plnk:partnerLinkType name="LoanService">
        <plnk:role name="LoanServiceProvider">
            <plnk:portType name="tns:LoanService"/>
        </plnk:role>
        <plnk:role name="LoanServiceRequester">
            <plnk:portType name=
              "tns:LoanServiceCallback"/>
        </plnk:role>
    </plnk:partnerLinkType>

</definitions>
```

ネットワークを流れるメッセージのかたち

こうしたWSDLの分析から，クライアントが先のWSDLで定義された，LoanServiceというportTypeのinitiateというoperationを呼び出したとき，リスト4のようなSOAPメッセージがネットワーク上を流れるのが分かります（SOAP Headerの部分は，省略しています）．

3，4で見た，WSDL（リスト3）のtypes節のLoanApplicationTypeのschema定義と，そのドキュメント表現の1つである，リスト4中のloanApplication要素を比べてみてください．

この呼び出しは，operation要素中にinput要素1つしかありませんので，クライアントからサーバへの非同期One-wayメッセージの送り出しになります．input要素は，サーバから見たメッセージの向きを表していますので，メッセージの流れはクライアントからサーバということになります．

リスト4　ネットワーク上のメッセージのかたち

```
<soapenv:Body>
  <initiate>
    <auto:loanApplication>
         <auto:SSN>123456789</auto:SSN>
         <auto:email>
          maruyama@wakhok.ac.jp
         </auto:email>
         <auto:customerName>
          maruyama
         </auto:customerName>
         <auto:loanAmount>9999</auto:loanAmount>
         <auto:carModel>CIMA</auto:carModel>
         <auto:carYear>1999</auto:carYear>
         <auto:creditRating>88</auto:creditRating>
    </auto:loanApplication>
  </initiate>
</soapenv:Body>
```

WebサービスがBPELを呼び出す場合のデータの受け渡し

クライアントが非同期にBPELを呼び出す場合に，どのようにデータが受け渡されるのかを見てみましょう．基本的には，前回の後半部分で見てきたことと重なっています．ただ，前回は，PartnerLinkに注目し，今回はデータの受け渡しに注目している点が異なります．

ここでは，先に見たように，クライアントがBPELのLoanServiceというportTypeのinitiateというoperationを呼び出したとしましょう．すなわち，リスト4のようなSOAPメッセージがクライアントからBPELサーバに送り出されたとしましょう．

このとき，BPELサーバは，次のreceive命令で，変数inputにSOAPメッセージに含まれている，クライアントから渡されたデータであるloanApplication要素を受け取ることができます．

```
<receive name="receiveInput"
         partnerLink="client"
         portType="tns:LoanService"
         operation="initiate"
         variable="input"
         createInstance="yes"/>
```

ここでは，データを受け取る入れ物であるvariableの指定と同時に，partnerLinkとportTypeとoperationの三つ組みが指定されていることに注意してください．

逆に，BPELサーバがクライアントに，非同期でデータを返すときには，たとえば，次のようなかたちのinvoke命令を使います．

```
<invoke name="replyOutput"
        partnerLink="client"
        portType="tns:LoanFlowCallback"
        operation="onResult"
        inputVariable="selectedLoanOffer"/>
```

ここでは，クライアントに送るデータの入れ物であるinputVariableの指定とともに，先のreceiveと同じように，partnerLinkとportTypeとoperationの三つ組みが指定されています．紙幅の関係で，このinvoke命令に対応するWSDLの部分は，リスト3では省略しています．

もしも，operation initiateが非同期型ではなく同期型の場合には，たとえば，次のかたちのreplyがデータをクライアントに返すことになります．

```
<reply   name="replyOutput"
         partnerLink="client"
         portType="tns:LoanService"
         operation="initiate"
         variable="output"/>
```

ここでは，クライアントに返すデータの入れ物はvariableで指定され，あと，先と同様の三つ組みが利用されています．

BPELがWebサービスを呼び出す場合のデータの受け渡し

BPELが，非同期的に外部のWebサービスを呼び出す場合には，invokeとreceiveを，この順番で使います．その例を以下に示します．

```
<!--  initiate the remote service -->
<invoke name="invokeUnitedLoan"
        partnerLink="UnitedLoanService"
        portType="services:LoanService"
        operation="initiate"
        inputVariable="loanApplication"/>

<!--  receive the result of the remote service -->
<receive name="receive_invokeUnitedLoan"
         partnerLink="UnitedLoanService"
         portType="services:LoanServiceCallback"
         operation="onResult"
         variable="loanOffer1"/>
```

inputVariable属性，variable属性の意味は，先に説明したのと同じです．前者はサービスに送り出すデータを，後者はサービスから受け取るデータを，変数名で指定しています．

BPELが，同期的に外部のWebサービスを呼び出す場合には，次のような同期型のinvoke命令を使います．

```
<invoke name="invokeCR"
        partnerLink="creditRatingService"
        portType="services:CreditRatingService"
        operation="process"
        inputVariable="crInput"
        outputVariable="crOutput"/>
```

ここでのinputVariableは同期型サービスのRequestを，outputVariableはサービスのResponseに対応しています．

同期型，非同期型，いずれの場合でも，サービスの指定には，partnerLinkとportTypeとoperationの三つ組みが指定されていますね．

BPELでの変数の代入操作

ここまではBPELと外部のサービス間の変数を使ったデータのやりとりを見てきました．今度は，BPEL内部での変数の操作を見ていきましょう．BPELでの変数への値の代入は，assignアクティビティを利用します．assignには，いくつかの基本的な利用パターンがあるので，それを見ていきましょう．

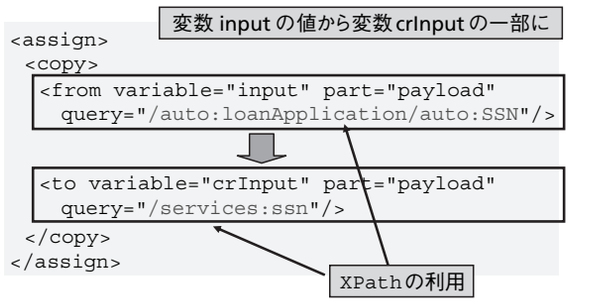


図-2　assignの例：変数の代入操作

まず，変数から変数への代入です．**図-2**を見てください．assignの中に，copy要素があって，その中にfrom要素とto要素があります．もちろん，fromからtoへコピーするという意味です．

fromとtoの中には，いずれにも，variableとpartとqueryという3つの属性があります．variableには，変数名を指定します．partは，先にも見たようにBPELの変数はWSDLのmessage節に対応するのですが，messege節ではpart要素が複数ある場合がありますので，partの指定が残っています．でも，document-literalのスタイルでしたらpartは1つしかありませんので，あまり気にする必要はありません．

大事なのは，queryです．ここには，XPathの指定をする文字列が入ります．XPathは，XMLの要素の中にサブ要素があるという入れ子の状態を，ファイル・システムのディレクトリの中にサブ・ディレクトリがあるというのと同じだと考えて，スラッシュ"/"を使ったパスの考え方でXMLの各要素にアクセスしようとするものです．**図-3**に，XPathのサンプルを示します．XPathを使えば，変数の一部の値を，他の変数の一部に入れることができます．

これが基本パターンですが，変数全体をコピーするときには，query属性を省略できますし，copy要素は，繰り返し出現することも可能です．

BPELの式から変数への代入操作も，次のようないくつかのパターンがあります．リスト5を見てください．まず，直接，リテラルを変数に入れることができます．また，expression式を用いて，文字列や，整数，bool値を，変数に代入することができます．

変数inputの値

```
<auto:loanApplication>
   <auto:SSN>123456789</auto:SSN>
   <auto:email>maruyama@wakhok.ac.jp</auto:email>
   <auto:customerName>maruyama</auto:customerName>
   <auto:loanAmount>9999</auto:loanAmount>
   <auto:carModel>CIMA</auto:carModel>
   <auto:carYear>1999</auto:carYear>
   <auto:creditRating>88</auto:creditRating>
</auto:loanApplication>
```

XPathの利用

```
/auto:loanApplication/auto:SSN     ➡ 123456789
/auto:loanApplication/auto:email   ➡ maruyama@wakhok.ac.jp
/auto:loanApplication/auto:carModel➡ CIMA
/auto:loanApplication/auto:carYear➡ 1999
```

図 -3　XPath での変数値へのアクセス例

リスト 5　BPEL の変数への代入

```
<!-- リテラルから変数への代入 -->
<assign>
 <copy>
    <from>
      <result xmlns="http://samples.otn.com/assign">
        <name/>
        <symbol/>
        <price>12.3</price>
        <quantity>0</quantity>
        <approved/>
        <message/>
      </result>
    </from>
    <to variable="output" part="payload"/>
  </copy>
</assign>

<!-- 文字式から変数へのコピー -->
  <copy>
    <from expression="GE" />
    <to variable="output" part="payload"
       query="/tns:result/tns:symbol"/>
  </copy>

<!-- 整数式から変数へのコピー -->
  <copy>
    <from expression="100" />
    <to variable="output" part="payload"
       query="/tns:result/tns:quantity"/>
  </copy>

<!-- boolean 値関数から変数へのコピー -->
  <copy>
    <from expression="true()" />
    <to variable="output" part="payload"
     query="/tns:result/tns:approved"/>
  </copy>
```

変数定義と代入の例

変数定義と変数代入の実際の例を，まとめとして，あらためて見ておきましょう．まず，リスト 6 のように，"tAddress" あるいは "address" の Schema 定義が与えられているとします（電話番号付きの住所情報と考えてください）．

リスト 6

```
  <complexType name="tAddress">
    <sequence>
      <element name="number" type="xsd:int"/>
      <element name="street" type="xsd:string"/>
      <element name="city" type="xsd:string"/>
      <element name="phone">
        <complexType>
          <sequence>
            <element name="areacode" type="xsd:int"/>
            <element name="exchange" type="xsd:int"/>
            <element name="number" type="xsd:int"/>
          </sequence>
        </complexType>
      </element>
    </sequence>
  </complexType>

  <element name = "address" type = "tAddress"/>
```

このとき，次のような message の型が WSDL で定義されているとしましょう．メッセージ person は，full-name パートに文字列を抱えていて，address パートに先に定義された型の address 情報を抱えていることが分かります．

```
<message name="person"
         xmlns:x="http://tempuri.org/bpws/example">
  <part name="full-name" type="xsd:string"/>
  <part name="address" element="x:address"/>
</message>
```

BPEL では，次のような variable の定義が可能です．

```
<variable name="c1" messageType="x:person"/>
<variable name="c2" messageType="x:person"/>
<variable name="c3" element="x:address"/>
```

このとき，次のような assign が，何を意味するかを考えてください．

```
<assign>
  <copy>
    <from variable="c1"/>
    <to variable="c2"/>
  </copy>
  <copy>
    <from variable="c1" part = "address"/>
    <to variable="c3"/>
  </copy>
</assign>
```

前半の copy は，ある人 (person)c1 の情報を c2 の情報にコピーしますし，後半の copy は，c1 の address 部分を，c3 のアドレス部分にコピーします．

前回は，BPEL の PartnerLink という概念を主に説明し，今回は，BPEL の変数と代入操作を見てきました．次回は，BPEL の紹介の最後として，BPEL の制御構造を見ていきたいと思います．

（平成 19 年 2 月 7 日受付）

丸山不二夫（正会員）
maruyama@wakhok.ac.jp

東大教育学部卒業．一橋大学大学院社会学研究科博士課程修了．「最北端・最先端」をモットーに，稚内で新しいスタイルとコンテンツの情報教育を展開．「新しい時代の新しい大学」を目指して，社会人 IT 技術者をターゲットとしたサテライト校を秋葉原に設置．アジアでの IT 教育も熱心に展開している．現在，稚内北星学園大学学長．