BPEL 介绍与举例

B20160310 王真

1 背景

随着 Web Service 技术日益成熟和流行，许多企业的很多部门相应地创建了 Web Service 服务。但是，单个 Web 服务通常难以满足实际中灵活多变的需求。如何在不改变这些 Web Service 正常运行的情况下，将这些 Web Service 集成起来创造新的业务模型、业务流程成为一个比较突出的业务需求。

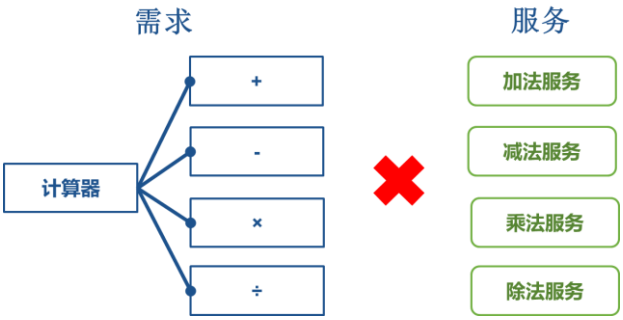

图 1 单一的 Web 服务难以满足实际需求

上图左边表示一个计算器的功能需求，右边表示已存在的服务。该计算器需要加减乘除四个功能，但是在已经存在的 Web 服务中，没有能够同时满足这四种需求的 Web 服务。在这种情况下，单一的 Web 服务不能满足实际的需求，那么如何在不改变这些 Web 服务的情况下，将这些 Web 服务集成起来满足新的需求便成为了一个具有重要现实意义的研究问题。

2 Web 服务的组合方式

Web 服务的组合方式有两种：编制和编排（如图所示）。在编制中，一个中央流程(可以是另一个 Web 服务)控制相关的 Web 服务并协调所涉及 Web 服务的不同操作的执行。相关的 Web 服务并不“知道”(也无需知道)它们参与了组合流程并在参与更高级别的业务流程。只有编制的中央协调员知道此目标，因此编制主要集中于操作的显式定义以及 Web 服务的调用顺序。相反，编排所涉及的每个 Web 服务完全知道执行其操作的时间以及交互对象。编排是一种强调在公共业务流程中交换消息的协作方式。编排的所有参与者都需要知道业务流程、要执行的操作、要交换的消息以及消息交换的时间。

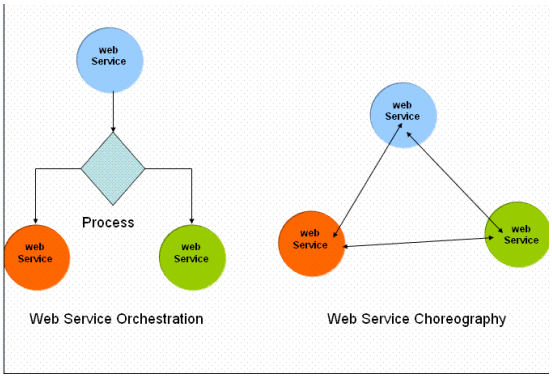

图 2　Web 服务的两种组合方式

3 BPEL 介绍

BPEL（Business Process Execution Language）是一种基于 XML 的可执行服务组装语言，可以将多个 Web Service 组合成复杂的业务流程，完成更大的业务需求。BPEL 利用了若干 XML 规范：WSDL 1.1，XML Schema 1.0，Xpath 1.0 和 XSLT 1.0。BPEL 流程由 WSDL 消息和 XML Schema 类型定义提供数据模型，由 Xpath 和 XSLT 提供数据操作支持，并由 WSDL 描述 Web 服务的对外接口。

```
<?xml version="1.0" encoding="UTF-8"?>
<process name="SupplyChainProcess"
 targetNamespace="http://ustb.vxbpel.org"
 xmlns:suc="http://supplychain.org/wsdl/supply-chain/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema" ···>
 <import namespace="http://supplychain.org/wsdl/supply-chain/"
  location="SupplyChainArtifacts.wsdl"
  importType="http://schemas.xmlsoap.org/wsdl/" />
```

name space

```
 <partnerLinks>
   <partnerLink name="ordergoodsPL" partnerLinkType="suc:OrderGoodsLT"
    myRole="consumer" />
   ···
 </partnerLinks>
```

partner links

```
 <variables>
     <variable name="input" messageType="order:OrderGoodsRequest" />
     <variable  name="output" messageType="order:OrderGoodsResponse"/>
  </variables>
```

variable declarations

```
<sequence name="main">
     <receive createInstance="yes" name="receiveInput"
      operation="OrderGoods" partnerLink="ordergoodsPL"
      portType="order:OrderGoodsPT" variable="input"/>

     <invoke name="Warehouse"  outputVariable="warehouseAmessagen"
      inputVariable="input"  operation="InquireGoods"
      partnerLink="warehouseAPL" portType="waa:WarehouseAPT">
     </invoke>
     <if name="warehouse">
          <condition>($warehouseAmessage.WarehouseAResponse &gt;=
$input.amount)
         </condition>
          <sequence>
            <invoke inputVariable="input" name="Shipper"
             operation="ShipGoods" outputVariable="shipperAmessage"
              partnerLink="shipperAPL" portType="sha:ShipperAPT">
             </invoke>
            ···
         </sequence>
     <reply name="reply" operation="OrderGoods" partnerLink="ordergoodsPL"
      portType="order:OrderGoodsPT" variable="output">
     </reply>
</sequence>
</process>
```

activity

interactions

图 3　BPEL 程序示例

BPEL 使用一个中心流程来协调不同的 Web 服务操作，流程本身可以看作一个新的 Web 服务，可以执行，只是执行的过程需要调用别的 Web 服务。

3.1 BPEL 流程的结构

BPEL 流程主要由变量声明(variable declaration)、伙伴链接声明（partner link）、处理器声明（handler）及业务流程描述四个部分组成。下图展示的是一个 BPEL 程序，描述的是一个供销链管理的业务流程。

（1）变量声明部分使用<variable>定义 BPEL 流程中使用的数据变量，用于存储消息以及对这些消息进行重新格式化和转换。通常需要为发送到合作伙伴以及从合作伙伴收到的每个消息定义一个变量。BPEL 可以使用三种类型的变量声明：WSDL 消息类型，XML Schema 类型（简单或复杂的）以及 XML Schema 元素。如例 1 所示。

例 1：使用 WSDL 文档中声明的消息类型，目标命名空间为 http://example.com/orders

```
<variable xmlns:ORD="http://example.com/orders"
          name="orderDetails"
```

（2）伙伴链接定义一系列参与交互的 Web 服务，主要分为两种：一种是 BPEL 流程所要调用的外部服务，一种是指 BPEL 自己所要提供的服务。使用<partnerLink>定义合作伙伴链接。定义合作伙伴链接包括如下两部分定义：

- 在 BPEL 中定义合作伙伴的<partnerLink>；
- 在该 BPEL 流程对应的 WSDL 中通过<partnerLinkType>定义合作伙伴对应的 WSDL 的 portType。

```
<partnerLinks>
    <partnerLink name="NCName"
                 partnerLinkType="QName"
                 myRole="NCName"?
                 partnerRole="NCName"?
```

外部服务使用 partnerRole 定义伙伴角色；业务程序自身的角色通过 myRole 表示，表示这个服务的接口是 BPEL 自己提供服务的。

（3）处理器声明部分在<faultHandlers>中定义故障处理程序。BPEL 程序在调用合作伙伴的过程中，或者合作伙伴的服务可能会抛出异常，或者 BPEL 流程内部调用中也会抛出异常。该元素通过<catch>和<catchAll>构造器定义。每个<catch>构造器被定义为截取某种具体的故障类型，通过故障 QName 定义。<catchall>可以被添加给任意未被具体故障处理器捕获的故障。

```
<faultHandlers>
        <catch faultName="QName"? … >
                activity
         </catch>
        <catchAll>?
```

（4）业务流程描述是实现特定工作流的主体，指定调用合作伙伴 Web 服务的顺序，由一组活动（activity）及其之间的联系构成，通常以 <sequence>（用于定义多个将按顺序执行的操作）开始。

3.2 BPEL 活动

BPEL 活动分为两类：基本活动和结构性活动。基本活动描述程序行为的基本步骤，包括 assign、invoke、receive、reply、throw、wait、empty、exit、rethrow 等。结构性活动描述流程的控制流逻辑，可包含其他基本活动或结构性活动，包括 scope、sequence、flow、switch、if、while、repeatuntil、foreach、pick 等。每个活动有两个可选择的标准属性：活动的 name 以及指出当连接故障发生时是否抑制的 suppressJoinFailure，同时有两个可选择的标准元素<source>和<target>，分别包括在容器<sources>和<targets>中。

BPEL 常用的活动如下：

- <receive>/<reply>（接收/回复）
- <receive>是整个 BPEL 的起点，一旦 BPEL 引擎从客户端接收到请求消息，将会启动一个 BPEL 流程。<reply>是整个 BPEL 的终点，BPEL 流程会把响应结果返回给服务请求者。
- <assign>/<invoke>（赋值/调用）
- <sequence>/<flow>（顺序/并行）
- <empty>/<exit>（空活动/立即结束）
- <if> /<while>（分支/循环）

BPEL 语言与传统的程序设计语言一样有顺序、分支、循环等标准的控制结构，但也有其与众不同的特点，主要体现在如下四个方面：

(1) BPEL 提供一种显式的集成机制组装 Web 服务，而这样的集成在传统程序中是隐式；

(2) BPEL 参与组装的服务可以采用不同语言实现，而传统的程序设计语言模块通常需要由同一种语言实现；

(3) BPEL 程序表示为 XML 文件，不同于传统应用程序；

(4) BPEL 通过 flow 活动支持并发机制、带有 link 的 flow 活动支持同步机

制。

BPEL 流程和 Web Service 的交互应用如图所示：

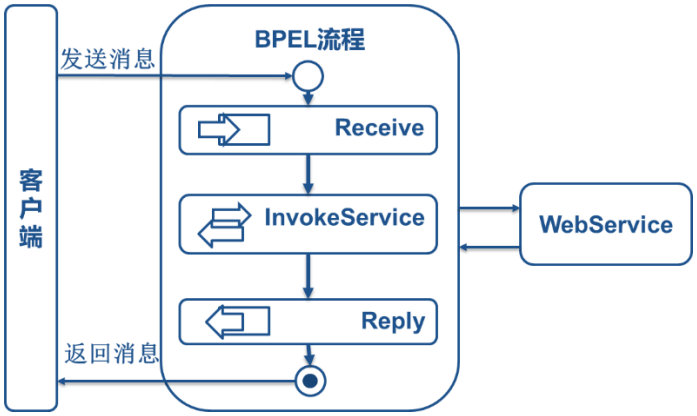

图 4　BPEL 流程与 Web 服务的交互过程

4 BPEL 服务组装流程

BPEL 程序开发的手段多种多样，目前比较流行的是使用 BPEL Designer 来完成 BPEL 流程的设计和 BPEL 程序的开发。BPEL Designer 是一个支持 BPEL 流程设计和程序开发的 Eclipse 插件。该插件能够以可视化的方式实现流程的设计可程序的开发，非常便于程序开发人员的使用。本文将通过组装一个包含加减法功能的计算器服务来示例 BPEL 程序的开发过程。

4.1 环境配置

开发 BPEL 程序之前，首先准备相关开发与运行环境。本文采用的环境包括 JDK1.6、Tomcat6.0、Eclipse4.6.0 和 Apache ODE 1.3.6。JDK 是 Java 语言的软件开发工具包，主要用于桌面应用、移动设备、嵌入式设备上的 Java 应用程序开发。JDK 是整个 Java 开发的核心，包含了 Java 运行环境，Java 工具和 Java 基础类库。Apache ODE 是一个能够解析并执行 BPEL 程序的引擎，能够根据 BPEL 流程与 Web 服务进行交互、发送和接收消息、处理数据操作并恢复错误。本文使用 Apache ODE 执行 BPEL 程序。Tomcat 是一个用于运行 Web 应用的容器，本文使用 Tomcat 运行用于执行 BPEL 程序的 Apache ODE 引擎。

（1） Apache ODE 引擎的安装：方式比较简单，只需要将 ODE 引擎的 war 文件拷贝至 Tomcat 安装路径中的 webapps 文件夹，然后启动 Tomcat 服务器。

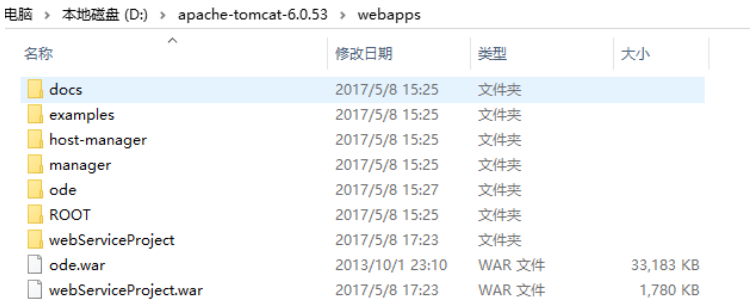

图 5　webapps 文件夹

待启动完毕后，会发现 Tomcat 安装路径中的 webapps 文件夹下会出现 ODE 文件夹。此时，启动浏览器，输入 http://localhost:8080/ode，就会看到 ODE 的界面，至此，ODE 安装完毕。

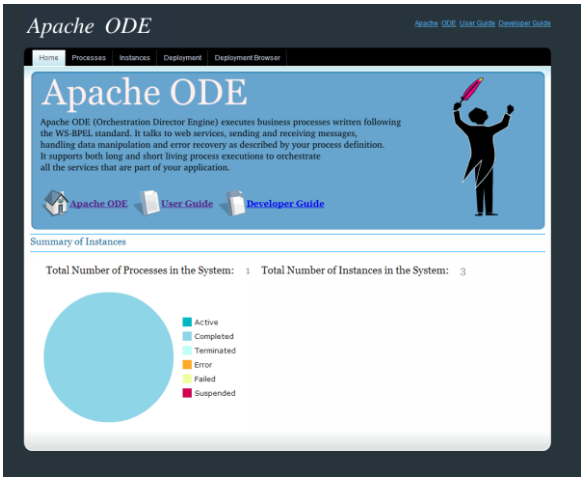

图 6　ApacheODE 界面

（2） 建立 ODE Server：在 Eclipse 中选择 File->New->Other 选项，选择 Server，如下图：

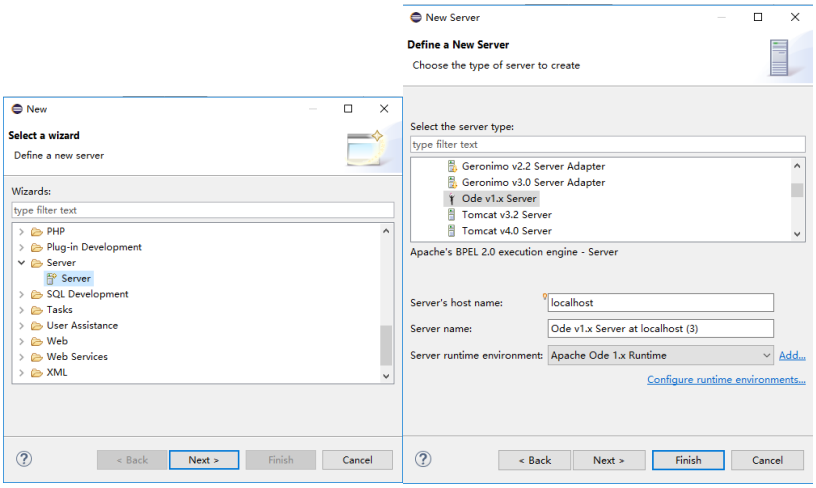

图 7 建立 ODE Server

（3） 安装 BPEL Designer：在 Eclipse 中选择 Help->Install New Software，在弹

出的窗口中电机 Add，添加 URL 如下。之后选择 BPEL Designer1.0.5 安装即可。

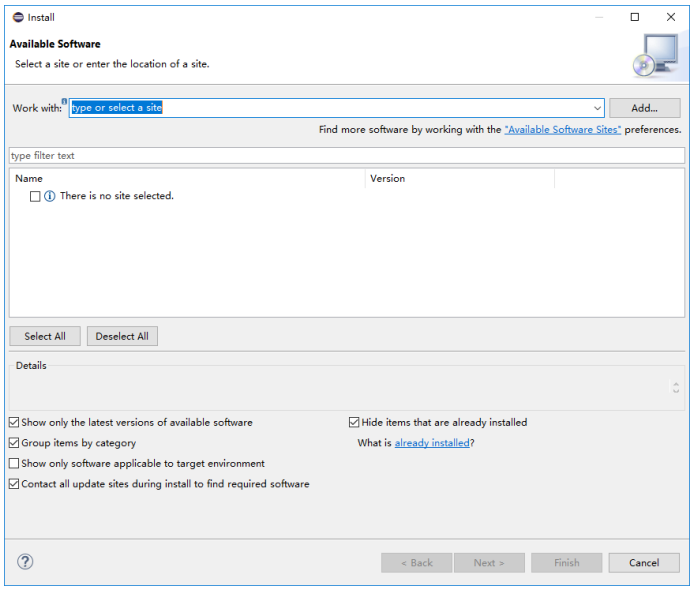

图 8 安装 BPEL Designer 1

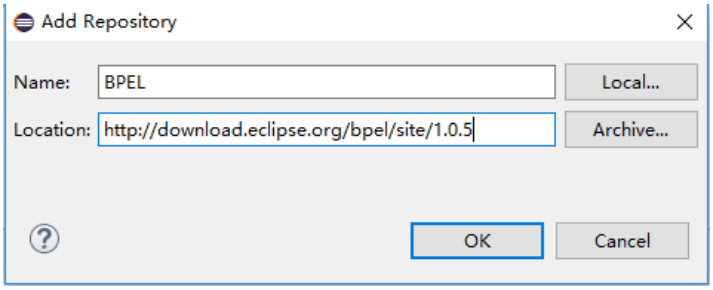

图 9 安装 BPEL Designer 2

4.2 构建待组装服务

环境配置完成后，构建用于服务组装的加法和减法 Web 服务。两个 Web 服务的开发和部署分为以下几个部分：

（1） 启动 Eclipse，建立一个动态 Web Project，工程名字为 webServiceProj，工程下面建立两个包 ws.example.add 和 ws.example.sub：

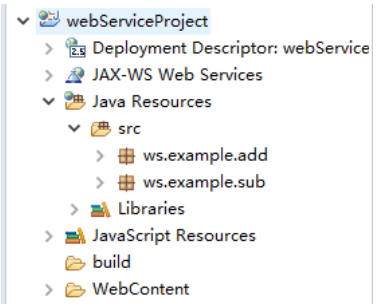

图 10 动态 Web Project 结构

1) 在此以加法为例，代码如下所示：

AddService.java | SubService.java | Ode v1.x Server at localhost (2)

```
1  package ws.example.add;
2
3  public class AddService {
4
5      public double add(double a1, double a2){
6          System.out.println(a1 + " + " + a2 + " = " + (a1 + a2));
7          return a1+a2;
8      }
9
10 }
11
```

图 11 加法服务代码

2) 加减法 WebService 部署：选择 AddService.java，右键 WebService 选项，选择 Create WebService，如图所示：

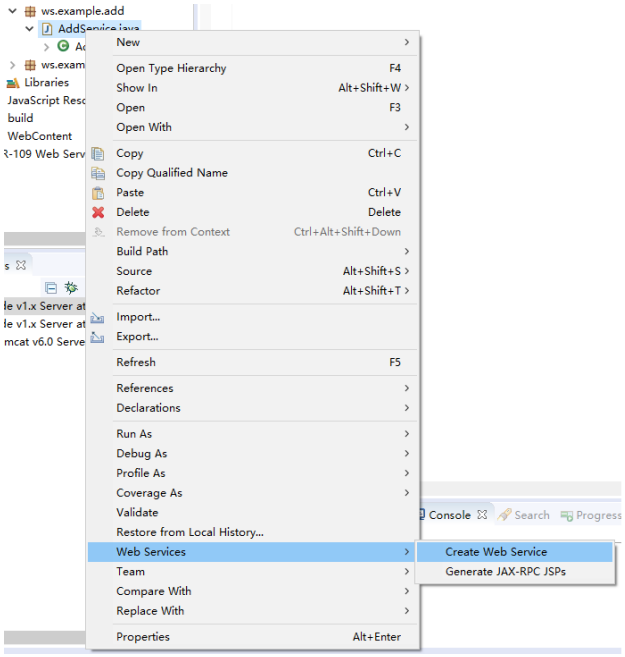

图 12 将 AddService 转换为 Web 服务

按照图示方式配置：

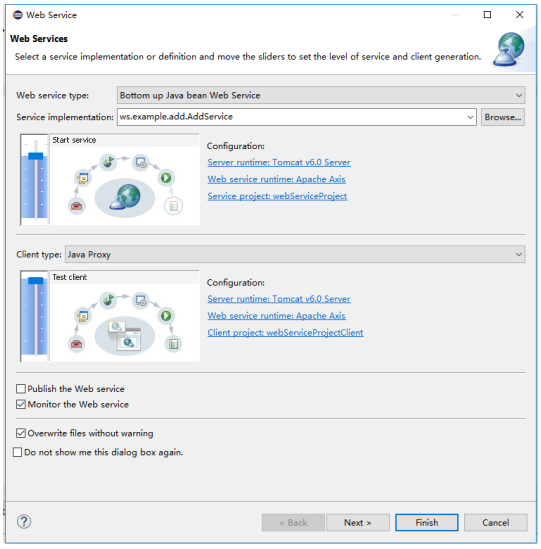

图 13 转换配置

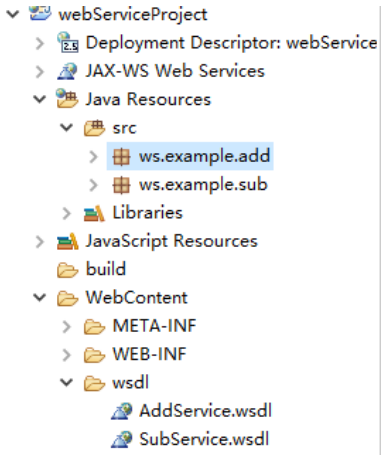

图 14 转换结果

点击 finish，就会生成 Java 类对应的 wsdl 和配置文件，并发布到 tomcat 中，同时会生成调用服务的客户端。将工程 Export 为一个 war 包，并将该包放在 Tomcat 服务器根目录中的 webapps 目录下，重启 Tomcat，则完成服务的发布。

4.3 BPEL 实例开发和部署

用于服务组装的两个 Web 服务开发部署完毕后，接下来进行 BPEL 实例的开发和部署，分为以下几个步骤：

（1） 创建 BPEL 工程：点击 File->New->Other，选择 BPEL2.0->BPEL Project。在弹出的 BPEL 工程对话框中，将工程命名：

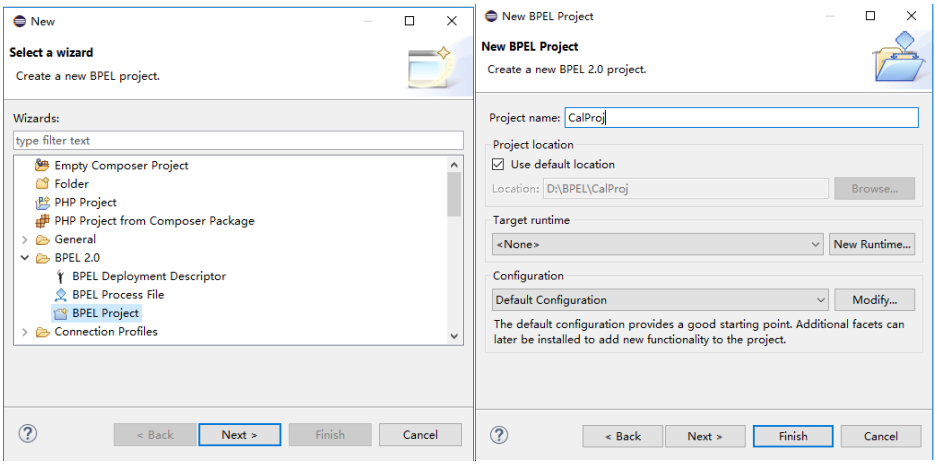

图 15 创建 BPEL 工程

点击 Finish，完成 BPEL 工程的创建

（2） 创建流程：选择刚刚建立的 BPEL 工程，点击 File->New->Other，然后选择 BPEL2.0 ->BPEL Process File，打开流程创建对话框，流程名字为 CalculatorProcess，命名空间为 http://www.ustb.edu.cn/bpel/sample：

（3） 选择同步流程：

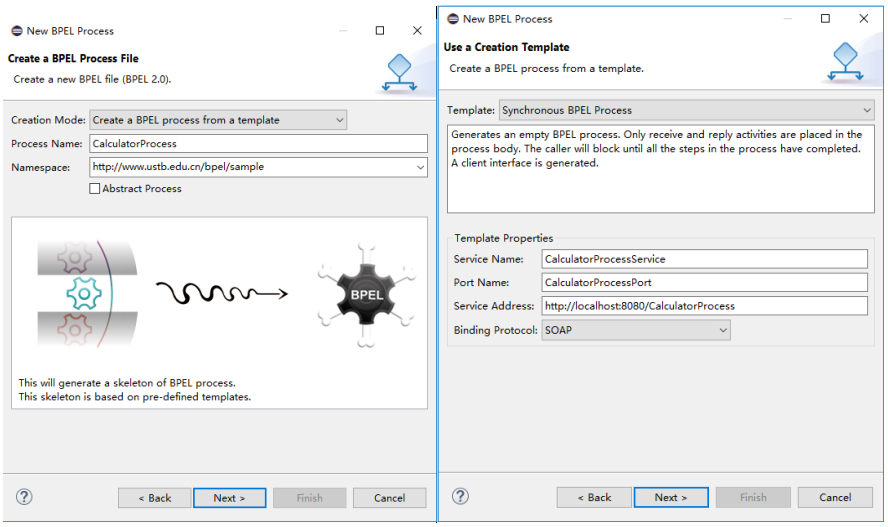

图 16 创建 BPEL 流程

最后点击 Finish 即可。

（4） 导入加法服务和减法服务的 wsdl 文件：将 AddService.wsdl 和 SubService.wsdl 复制到工程目录里面：

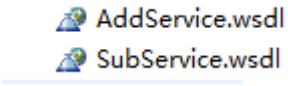

图 17 导入 WSDL 文件

（5） 设置伙伴链接：BPEL 把流程中涉及的 wsdl 称为伙伴链接。BPEL 对应的 wsdl 本身也是一个伙伴链接。对于流程中用到的加法服务和减法服务，生成对应的伙伴链接：

- 在右边的 Partner Links 中，增加两个链接：addPL 和 subPL，分别对应加法服务和减法服务：

图 18 添加伙伴链接

- 点选 addPL，在属性视图中 Details 页面中选 Browser 按钮，打开一个对话框；点 add WSDL，将加法服务的 WSDL 和减法服务的 WSDL 加进来，对话框会把这两个 WSDL 文件中的接口都显示出来：

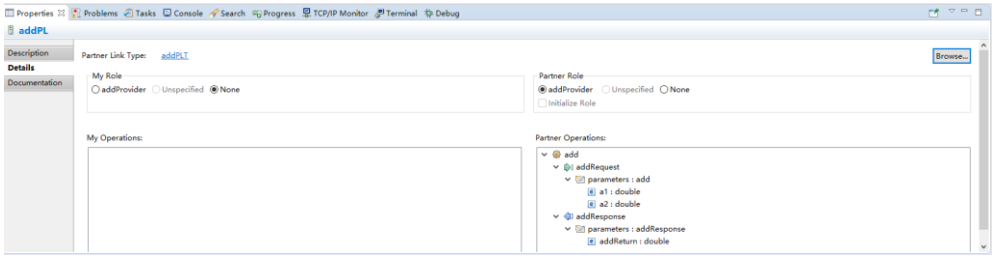

图 19 配置伙伴链接

- 选上 AddService 接口，点击 OK，之后会弹出伙伴链接类型定义，把名字取为 addPLT，点击 Next，并对 Role Name 输入 addProvider，点击 Finish 完成，并把属性视图中的 PartnerRole 选为 addProvider；

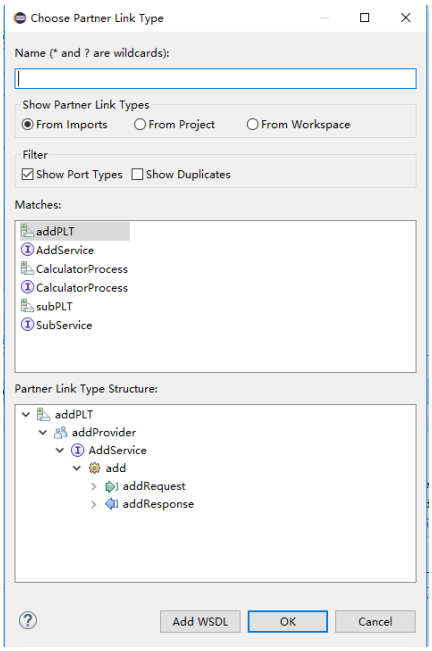

图 20 选择对应的 WSDL

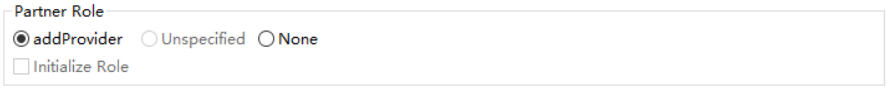

图 21 修改 Partner Role

（6） 修改流程的输入输出：打开 CalculatorProcessArtifacts.wsdl 文件，增加输入输出参数，并定义相对应的类型：

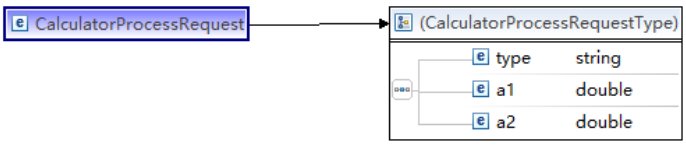

图 22 修改输入参数

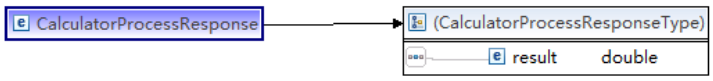

图 23 修改输出参数

（7） 创建调用服务时需要的变量：每次调用一个服务时都需要有对应的变量，同时需要进行赋值。在这里调用加法服务和减法服务时，需要额外加入 addRequest、addResponse、subRequest 和 subResponse，添加完成后如下：

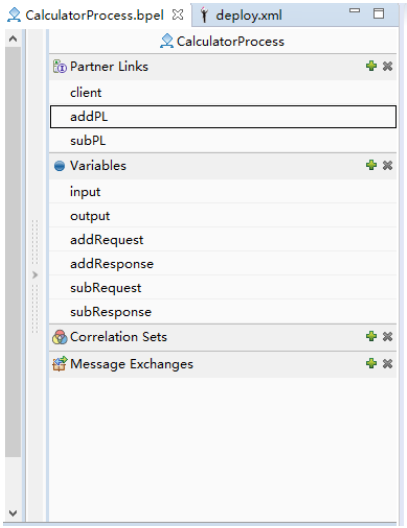

图 24 添加变量

（8） 配置变量：点击 addRequest 变量，在属性视图中切换到 Details 页面，点击右上角的 Browse 按钮，打开类型选择对话框，addRequest 是加法服务的输入，因此把它的类型设置为加法服务里面的输入定义就可以了。

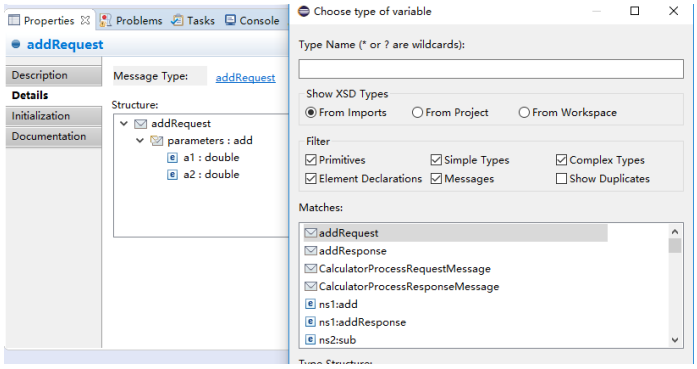

图 25 配置变量

以同样的方式定义其他三个变量。

（9） 设计 BPEL 流程：首先按照业务逻辑将 BPEL 流程画图，如下所示：

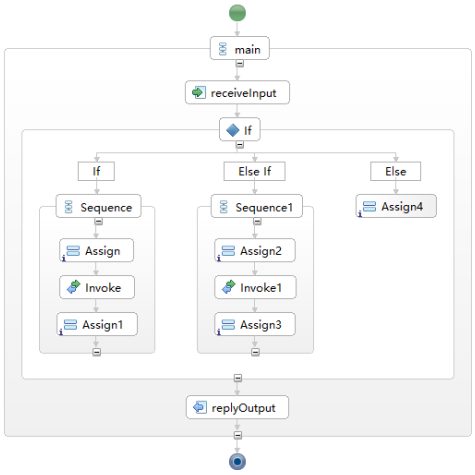

图 26 BPEL 流程图

- 首先配置 if 语句，如果 type = ‘add’，则调用加法服务，如果 type = ‘sub’，则调用减法服务，if 分支的判断条件如下所示

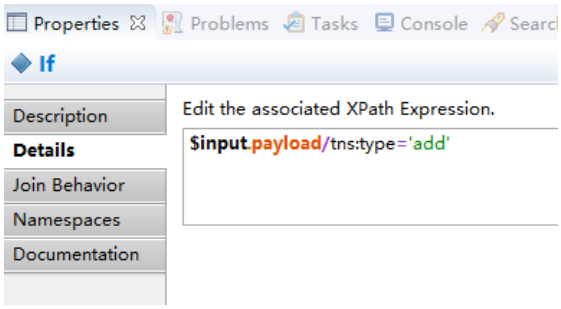

图 27 IF 分支配置

- 下面配置 Assign 语句，四个 Assign 语句的配置过程都一样，这里只讲解一个 Assign 的配置。第一个 Assign 的作用是把输入变量 input 中的前两个参数传给 addRequest 变量。

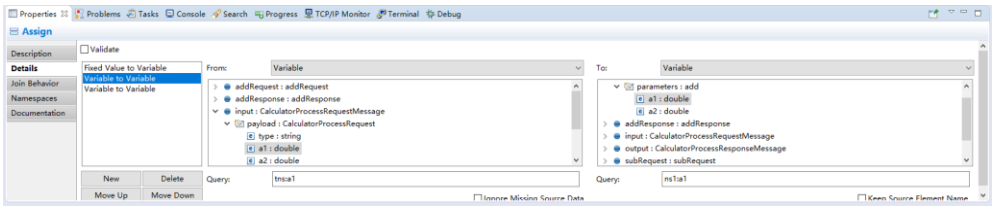

图 28 Assign 节点配置

- 把 input 的 a1 赋给 addRequest 下的 a1，然后将 a2 赋给 addRequest 下的 a2。中间会提示是否初始化 addRequest，点击 Yes 即可。然后配置 Invoke 语句。Invoke 语句的作用是调用伙伴链接接口下的对应操作，还需要指明输入输出变量。在 Details 页面进行如下配置：

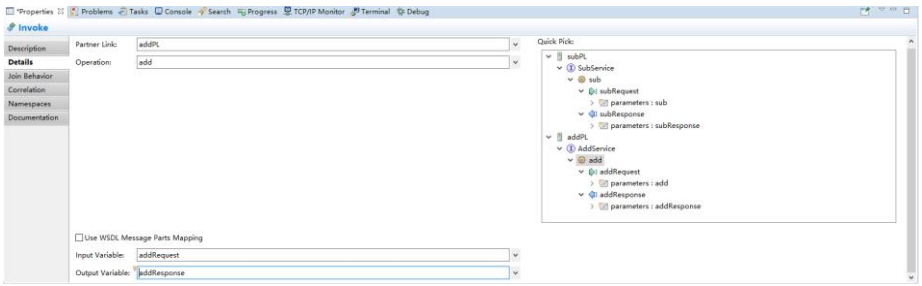

图 29 Invoke 节点配置

（10）流程发布：在项目名上选择 File->New->Other->BPEL2.0，选择 BPEL Deployment Descriptor，直接点击 Finish 即可生成一个 deploy.xml 文件。打开并配置伙伴链接的端口：

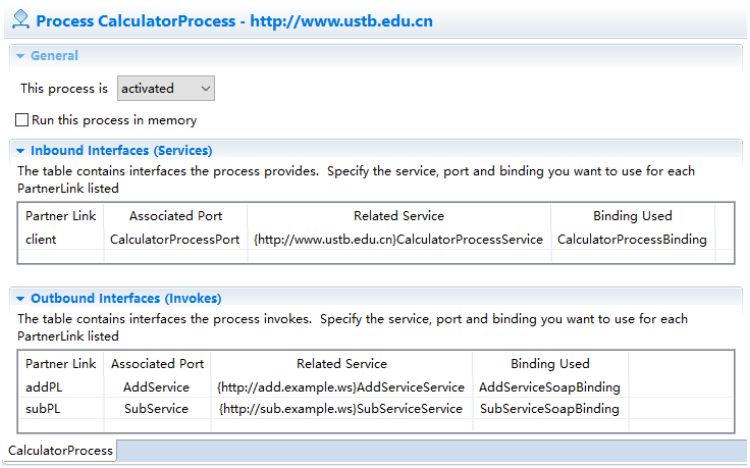

图 30 Deployment Descriptor 配置

之后将其部署到 ODE 中，将发布地址改为 http://localhost:8080/ode/processes/CalculatorProcess；

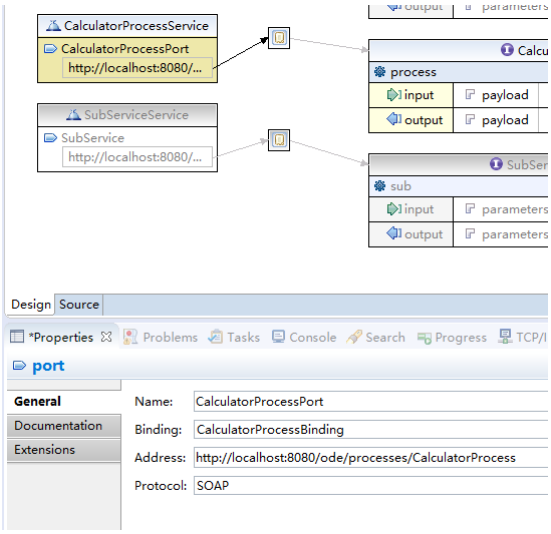

图 31 修改发布路径

BPEL 流程开发与部署完毕后，进行 BPEL 的运行和测试，分为以下步骤：

（1） 启动 ODE 服务器：将 BPEL 工程部署在 ODE 服务器上，Console 信息如下图：

```
六月 11, 2017 1:49:50 下午 org.apache.catalina.startup.HostConfig deployWAR
信息: Deploying web application archive webServiceProject.war
六月 11, 2017 1:49:50 下午 org.apache.axis.utils.JavaUtils isAttachmentSupported
警告: Unable to find required classes (javax.activation.DataHandler and javax.mail.internet.MimeMultipart). Attachment support is disabled.
六月 11, 2017 1:49:50 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory docs
六月 11, 2017 1:49:50 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory examples
六月 11, 2017 1:49:51 下午 org.apache.catalina.core.ApplicationContext log
信息: ContextListener: contextInitialized()
六月 11, 2017 1:49:51 下午 org.apache.catalina.core.ApplicationContext log
信息: SessionListener: contextInitialized()
六月 11, 2017 1:49:51 下午 org.apache.catalina.startup.HostConfig deployDirectory
信息: Deploying web application directory ROOT
六月 11, 2017 1:49:51 下午 org.apache.coyote.http11.Http11AprProtocol start
信息: Starting Coyote HTTP/1.1 on http-8080
六月 11, 2017 1:49:51 下午 org.apache.coyote.ajp.AjpAprProtocol start
信息: Starting Coyote AJP/1.3 on ajp-8009
六月 11, 2017 1:49:51 下午 org.apache.catalina.startup.Catalina start
信息: Server startup in 11439 ms
13:49:53,627 INFO  [BpelServerImpl] Unregistered process {http://www.ustb.edu.cn}CalculatorProcess-16.
13:49:54,694 INFO  [BpelServerImpl] Deleted only the process {http://www.ustb.edu.cn}CalculatorProcess-16.
13:49:55,083 INFO  [BpelServerImpl] Registered process {http://www.ustb.edu.cn}CalculatorProcess-17.
13:49:55,083 INFO  [DeploymentPoller] Deployment of artifact CalculatorProgress successful: [{http://www.ustb.edu.cn}CalculatorProcess-17]
```

图 32 服务部署信息

（2） 测试 BPEL：右击 CalculatorProcessArtifacts.wsdl，选择 Web Service->Test with Web Service Explorer，之后输入参数，运行流程，观察返回结果是否和预期相符。

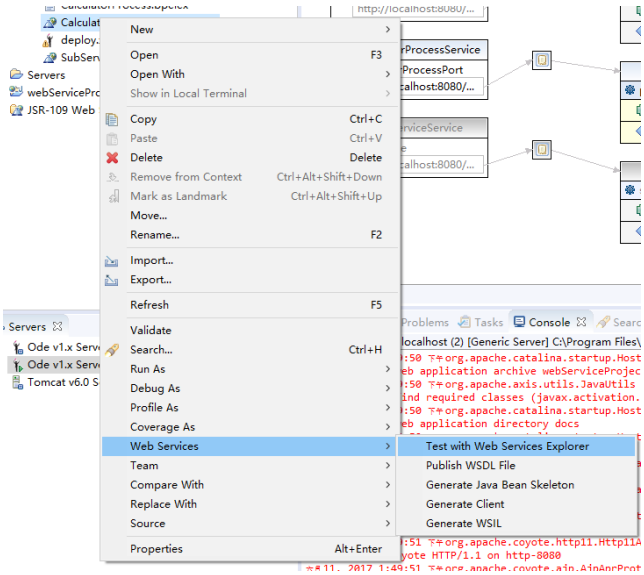

图 33 测试组装的服务

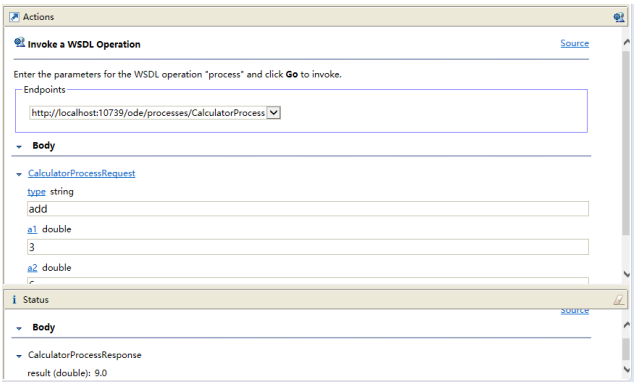

图 34 测试结果

参考文献：

[1] BPEL2.0 规范

[2] BPEL 实例教程

[3] BPEL 思想

[4] 一起学 BPEL 实例教程

[5] http://www.docin.com/p-1399666881.html.

[6] https://www.ibm.com/developerworks/cn/education/webservices/ws-understand-web-services7/.