**Bpel简介及实例总结**

# Bpel简介

[面向 Web 服务的业务流程执行语言](http://www-128.ibm.com/developerworks/library/ws-bpel/)(BPEL 或 BPEL4WS)是一种使用 Web 服务定义和执行业务流程的语言。BPEL 使您可以通过组合、编排和协调 Web 服务自上而下地实现面向服务的体系结构 (SOA)。BPEL 提供了一种相对简单易懂的方法，可将多个 Web 服务组合到一个新的复合服务（称作业务流程）中。

## BPEL规范

Web 服务的业务流程执行语言（Business Process Execution Language for Web Services，BPEL4WS）规范，其定位是成为整合方面的 Web 服务标准。您可以创建能够完成 Web 服务调用、抛出故障或终止一个流程等工作的不同活动，然后将它们连接起来，从而创建出复杂的流程。这些活动可以嵌套到结构化活动中，结构化活动定义了其中的活动的运行方式，如串列或并行还是取决于某些条件。它从本质上来说是IBM的WSFL和Microsoft的XLANG的结合物，目前已经成为业界标准。WSFL 支持图形化的流程，而XLANG在结构化构造方面有独到的方法，而BPEL4WS正是吸取了两者的优点，同时摒弃了一些复杂繁琐的部分，形成了一种较为自然的描述商业活动的抽象高级语言。

## BPEL语言特点

   如果对BPEL4WS语言的语法做较深入的研究，你就会发现它其实只是对原有编程语言思想的继承和发展。  
关于继承，BPEL4WS语言拥有传统编程语言的一些基本特性，如：  
赋值操作（由对〈assign〉的操作完成）；  
循环操作（由〈while〉操作完成）；  
选择操作（由〈switch〉和〈case〉操作完成）；  
远程调用操作（由〈invoke〉操作完成）；  
错误捕捉操作（由〈catchfault〉和〈catchall〉操作完成）；  
错误抛出操作（由〈throw〉操作完成）

除此之外bpel还有其他一些操作

<receive>：一般作为业务流程的起点出现。允许业务流程阻塞等待匹配消息的到达，该活动当消息到达时完成。

<reply>：一般作为业务流程的终点出现，把业务流程的结果返回给服务请求

<wait>：等待一段给定的时间或等到某一时刻，需指定到期时间。

<empty>：在业务流程中插入“no-op”指令。例如，这个构造可被用于并行活动的同步。

<extensionActivity>：该元素用于通过说明新活动类型来扩展ws-bpel。该元素的内容必须是ws-bpel标准属性（standard-attributes）和标准元素（standard-elements）可使用的独立的元素。这类新的活动称为扩展活动。该元素的内容必须是一个独立的元素，并且其命名空间必须与ws-bpel的命名空间不同。

<exit>：该活动用于立刻终止业务流程实例。所有当前运行的活动必须被立刻终止。不用引用任何终点处理、错误处理或者补偿行为。

上述的这些操作，在画流程设计流程的时候，名称都是可以自定义的，但是这个活动的性质不会变，还是实现相关的操作，平时常用的一些操作会在下面的例子项目中进行实践。

# 搭建bpel流程环境

对于接触流程这块的内容不是很熟悉的话，我想应该会有这样的疑问，因为我就是这样的一个新手。

流程从哪来的？

流程设计器的工作。流程设计器也是工作流系统不可缺少的一部分，通过它，熟悉本公司业务流程的员工可以设计出各种日常事务，或是生产过程的流程。它一般是可视化的图形界面，有很多的框框、线条之类的，通过拖拽来构建一个流程（——!当然了，比在word里面画图要难的多，因为还需要设置很多参数，或是指定活动等。这些参数就在日后流程运行时，控制了流程的流向啊）。把流程创建好之后，一般会生成一个基于XML的文本文件，这个文件其实就是你刚才设计的流程。然后，你把文件交给流程引擎，设计器的工作就算完成了。嗯？？引擎是什么

2）引擎是什么？

你光有了上面的流程文件也没啥用啊，这时就需要一个东西来处理这个文件了，它读取这个文件的内容，根据文件设定的流程走向来控制这个流程。比如，文件审批的流程启动之后，引擎发现首先要项目经理通过，所以它就给项目经理这个用户发送一个消息，说XXX要审批X文件，然后，项目经理进入系统，pass这个文件，这时候引擎知道了项目经理已经通过了这个文件，根据流程定义文件，下一个应该是处长了，所以它又要通知处长了。。。类似的。引擎就好像流程的指南针一样，指导着流程走向不同的地方。它是整个系统的核心。

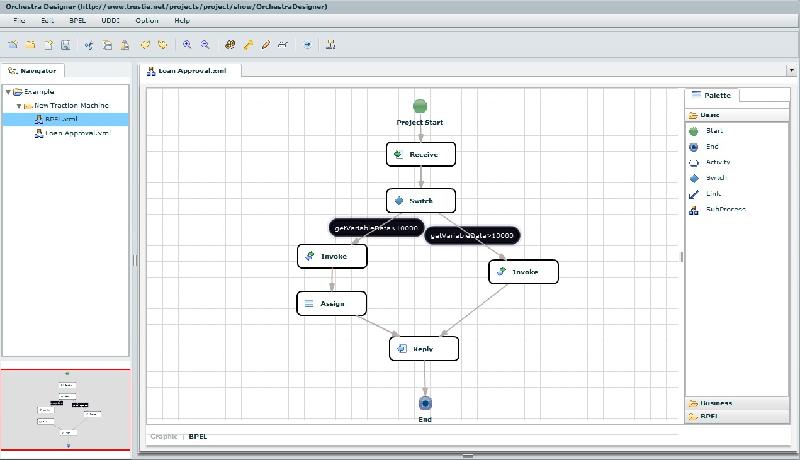
3）个人的理解

也许有的流程比较复杂，毕竟bpel流程主要是针对web service的，对于其他的接口，比如说就像集成平台的BPD，里面提供好多借口，都可以加载到流程里面去，包括DLL接口，DB接口等等，这些我感觉也就是本来也是单一的，后来为了方便集成到一起的，刚接触这些，有时候也会向别人请教一些他们感觉很幼稚的问题，但是越是简单的问题就越难理解的很清楚，虽然没接触过但是看了学了就可以理解，以后通过实践我感觉可以透彻的理解。

## bpel流程设计器

要搭建bpel流程首先需要选择一个合适的流程设计器，bpel的流程设计器有好几个。

1.首先是eclipse可以加载相应的bpel开发插件，这个可以从官方网站下载，但是这个下载方式，一般是从软件的工具栏的help—>install new software 在弹出的对话框中输入需要更新的插件所在的更新站点，类似于镜像站的这种方式，在国内访问出奇的慢。也可以下载压缩包进行本地安装，但是这样容易出错误，如果安装bpel插件成功的话，打开eclipse，我安装的时eclipse-jee-neon-3版本的，这个版本的安装插件比较方便，因为刚开始我安装的是eclipse版的，安装插件完成后即可。

2.另一种bpel流程设计器是Orchestra Designer该项目来源于2009年OW2 开源比赛题目，目标是为OW2上的开源BPEL引擎Orchestra提供一个基于Flex技术的在线工作流编辑工具，并与Orchestra的Web 2.0管理控制台集成。该建模工具针对非技术人员，采用一种比BPEL更面向业务、更直观的图元作为建模基础，生成的模型可以在后台转换成BPEL输出， 并部署在BPEL引擎上运行，这个流程设计器功能比较单一，又是在线的，灵活性也不太好，所以并没有用。

1. 第三种流程设计器是Oracle公司的OracleBPELProcessManager，这个比较强大，也比较复杂，安装包就有好几个G，也充分说明了起容纳性，同时他也贯彻了Oracle一贯的作风软件的复杂度略高，一不小心就会出错，出错就会很纠结，同时关于此国内资料也很少，去官网下载个东西，速度比蜗牛还快。。。

除此之外还有其它的开源的流程设计器，并没那么多时间去看，，，

1. 第四种是ActiveBpel Designer这个貌似下载不到了，折腾了半天也找不到。这个设计器是紧密配合ActiveBPEL引擎的，其它的三种设计器都是本身自带。

## Bpel引擎+tomacat + ode

ActiveBPEL 是BPEL引擎的代表，也是一款可执行BPEL4WS规范的开源流程引擎，其结构和实现方式具有很高的参考价值。目前国内很多正在开发基于BPEL产品的中小型软件厂商，其实现的很多基础性内容和思想都参考自ActiveBPEL。受目前国内中小型客户对流程需求的限制，基于BPEL的开源引擎或小型产品被市场接受度还很低。但BPEL所围绕的业务流程及流程整合应用是一个发展趋势。

ActiveBpel支持标准bpel语言，而且开源，有很大的研究价值。对于工作流工作原理、技术方法的研究都有很大帮助，但是不好的是，它没有自己的设计器，它的公司已经将其设计器ActiveVOS商业化。

下面介绍如何搭建配置ActiveBPEL引擎，开源的东西果然不是白白免费的，光[**版本控制**](http://lib.csdn.net/base/git)的问题就足够让人崩溃，用了好长时间才配好。好在终于守得云开见月明，引擎终于跑起来了，下面来看一下配置过程，这里我引用写过的一篇博客，理解比较浅，只是简单的完成了配置，使引擎跑起来，其它的比如人家说的持久化并没有做，我也不理解持久化是做什么的，这也是个人的一个毛病，不理解的东西，没一点兴趣去做下去，我个人感觉

不理解再去应付的做下去，并没什么意思，下面介绍一下配置过程：

1.版本问题

     ActiveBPEL引擎选择的是5.0.2，需要Servlet容器支持，这里选择的是Tomcat。只有Tomcat5.5.27及以下的版本才可以，高一点版本与ActiveBPEL不兼容，我的java环境是1.80的，其实最好安装1.5的，不会出现问题，我是懒得装了，并且我做测试是在eclipse下作的如果要用ActiveBPEL部署流程则需要安装java1.5版。

     ActiveBPEL的持久化需要JNDI数据源。这个不做研究了。。。。暂时还理解不了

2.安装ActiveBPEL+tomacat

    最好的参考文档自然是安装包自带的文档，位于根目录/doc下的install\_engine.txt，E文的，不过很容易读懂，就是读不懂可以去翻译。。。

    第一步，安装Tomcat并配置CATALINA\_HOME环境变量。

    第二步，命令行下执行install.bat(windows)。 执行后在Tomcat根目录下生成bpr目录和webapps下四个部署文件，bpr下是ActiveBPEL的配置文件。

    随后，启动Tomcat和关闭Tomcat会自动启动和关闭ActiveBPEL引擎。

Tomcat启动后，浏览器中键入<http://localhost:8080/BpelAdmin/>即看到管理界面，表示安装成功，引擎的状态为Running。这时HOME页中的Description表示采用的是In\_Memory模式。

此时说明tomacat已经安装成功了，这个网址还是记一下，有时候挺容易忘的。

Tomcat在这里同时安装了5.5和6.0版，下面的两种方法会用到。

1. 安装ode，如果想直接使用eclipse里面的插件bpel2.0开发bpel流程并部署的话，ode是必须要安装的，ode同tomcat一样也是Apche的产品，可以从这里下载ODE

<http://ode.apache.org/getting-ode.html> 下载速度你懂得，关于ODE的配置与安装，这里作一下简单介绍，将下载下来的ODE（我下载的事1.3）解压，将里面的war格式的文件复制到你安装的tomacat下的webapps目录下，这个文件其实就是一个网页，这个文件其实可以通过eclipse的导出功能生成，然后重启Tomcat即可，可以打开<http://localhost:8080/ode/> 进行验证，可开成功即安装成功，此时在webapps目录下也会生成ode文件夹，用途下边具体说。

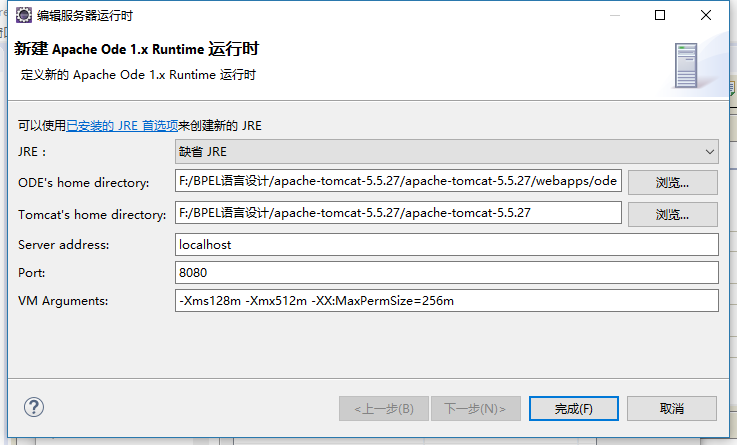
# 使用eclipse bpel插件开发并执行 bpel 流程

## 相关服务的开发

（1）在 Eclipse 上配置 ODE server

首先是显示出 Servers 视图。通过 **Windows->Show View->Servers**。

在 Servers 视图中点右键,选择 **New->Server**，开始配置，主要是下面这个对话框。



主要是配置 ODE 和 Tomcat 的目录，要注意的是 **ODE** 的目录是你安装在 **Tomcat** 下的目录，而不是你把 **ODE** 的压缩包解压的地方。

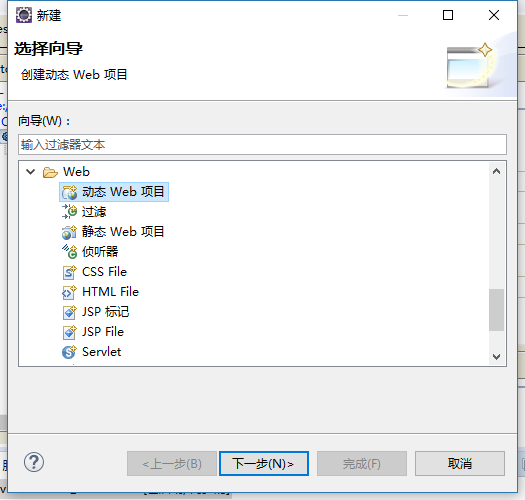
配置完后就可以在 eclipse 的 servers 视图中启动，停止 ODE 了。（在启动 ODE 时，

Tomcat 也会启动，不需要单独去启动 Tomcat）.

 为了进行服务组合，首先必须有服务。因此，这一部分会向大家介绍Eclipse环境下

的服务的简单开发过程。

1. 1.新建一个动态 web project- **webServiceProj**

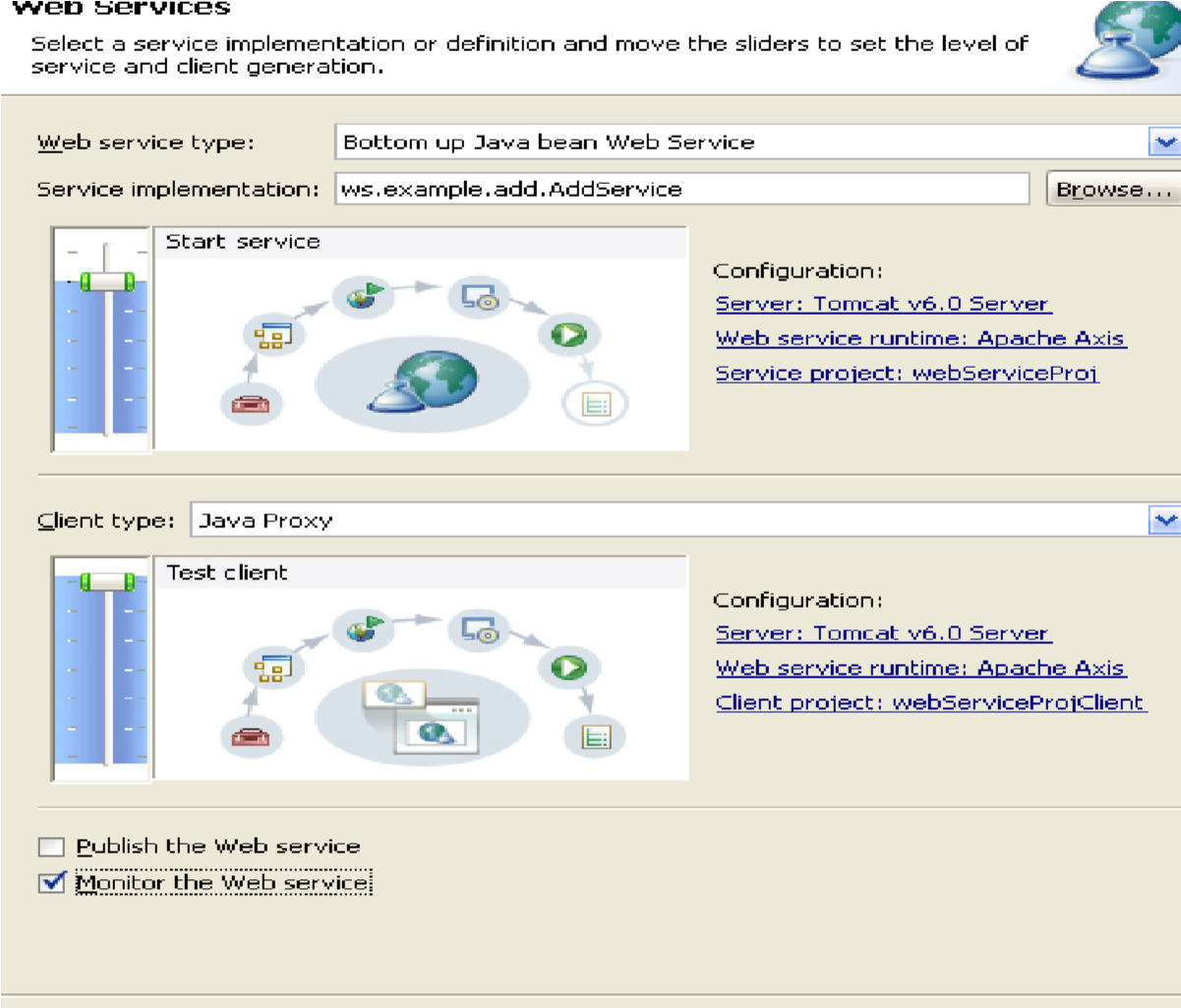


2.在工程的 src 目录下建两个包：ws.example.add 和 ws.example.sub 分别放加法和减法的实现类。因为加法服务和减法服务的实现过程都一样，下面只讲解加法服务的实现。

3.在包 ws.example.add 下建一个 java 类 **AddService.java**,具体实现如下：

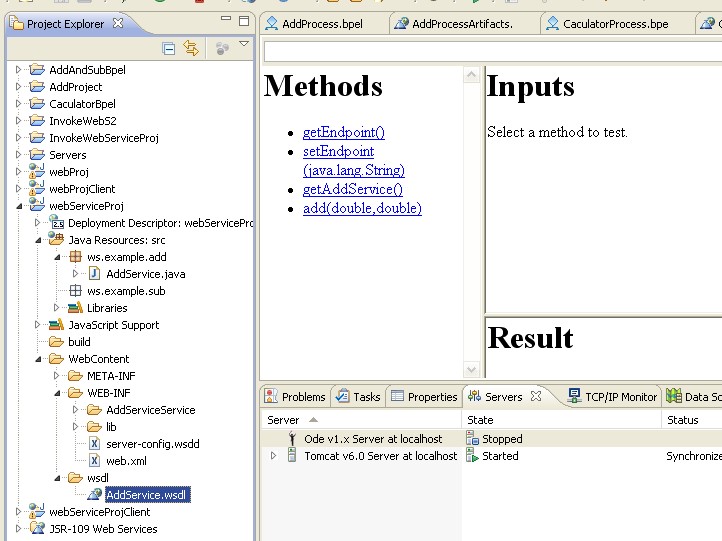
|  |
| --- |
| **package** ws.example.add; **public** **class** AddService {  **public** **double** add(**double** a1,**double** a2){ System.*out*.println(a1+"+"+a2+"="+(a1+a2)); **return** a1+a2;  }  } |

选上 **AddService.java**，点 **File -> New -> Other... -> Web Services -> Web Service**.

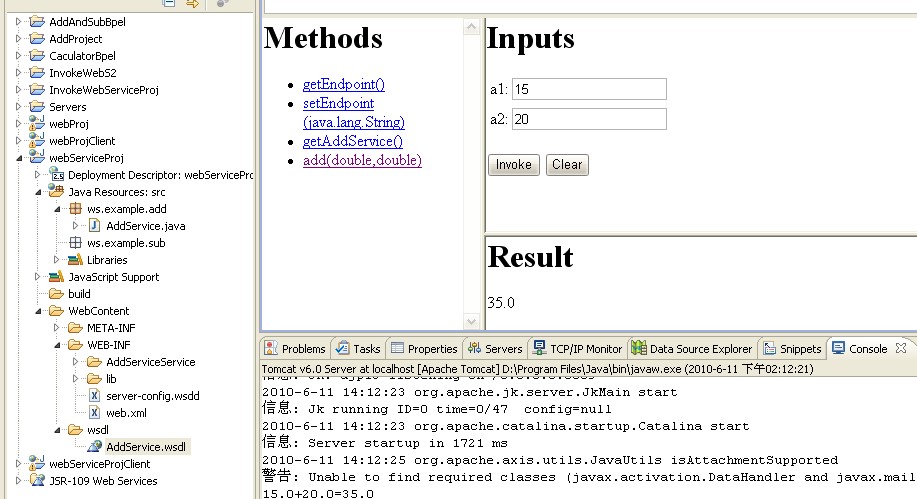


点 finish 就会生成 java 类对应的 wsdl 和配置文件，并发送到

tomcat 中，同时会生成调用服务的客户端，其实此时需要安装Tomcat6.0环境配置服务器，不然会出错。



1. 在方法中点 add(double,double),输入两个数，点 invoke 则返回对应的结果。说明服务生成成功。



1. 如果你在生成 web service 的过程中选择了“**Monitor Web service**”，则可以在 **TCP/IP Montor** 中看到每次调用过程中的 SoAP 消息。这里就不展示了。
2. 同样的方式生成减法服务。
3. 要想把生成的服务放到 Tomcat 目录下，则很简单，把这个工程 Export 成一个 war 包即可。

**File****Export****Web****WAR Fil**e,选择保存位置，则会生成 **webServiceProj.war**，把生成的 war 包复制到 Tomcat\_Home/webapps 目录下，重启 Tomcat，则完成服务的发布。

在浏览器里输入[**http://localhost:8080/webServiceProj/services/AddService?wsdl**](http://localhost:8080/webServiceProj/services/AddService?wsdl)则能打开加法服务对应的wsdl.这也意味着服务发布成功了。

## Bpel流程的开发

接下来的内容就是开发 BPEL 了。这一部分内容大家可以参考[这里](http://www.codeproject.com/KB/webservices/BPELInvokeWebService.aspx)。

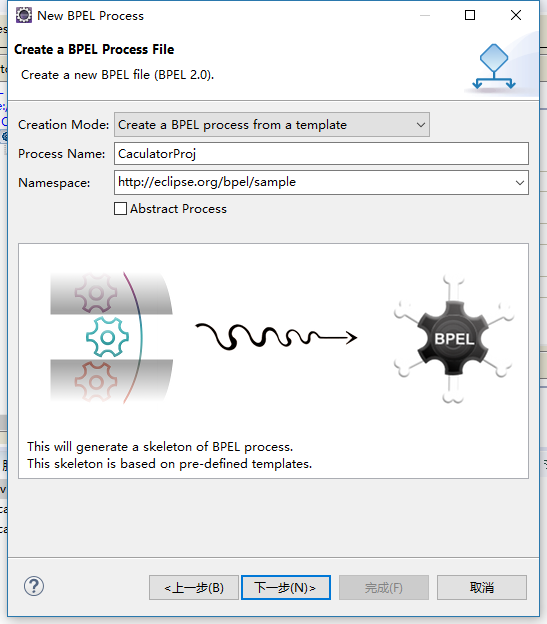
### 1.创建bpel工程。

点 File > New > Other，然后点 BPEL 2.0 -> BPEL Project 打开 BPEL 项目对话框，这里只需要输入项目名称，其它保持默认，点 Finish 就行了。这里项目名取为

**CaculatorProj**

### 2.创建流程

选上刚建立的项目，点File > New > Other，然后BPEL 2.0 > New BPEL Process File，打开流程创建对话框。按如下设置，点finish即可。流程名是**CaculatorProcess**，名字空间设为：[**http://www.tju.edu.cn/bpel/sample**，](http://www.tju.edu.cn/bpel/sample)选择同步流程。



这样，自动生成了两个文件：CaculatorProcess.bpel 用来保存 BPEL 流程代码， CaculatorProcessArtifacts.wsdl 用来保存该流程对应的 WSDL 文件，通过该文件，就知道该流程对外提供的接口。

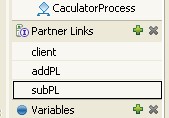
### 3.导入加法服务和减法服务的wsdl文件。

很简单，就是把 AddService.wsdl 和 SubService.wsdl 复制到工程目录里。

### 4.设置伙伴链接

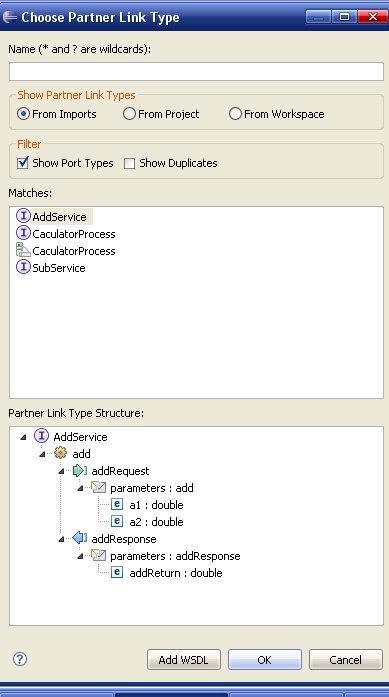
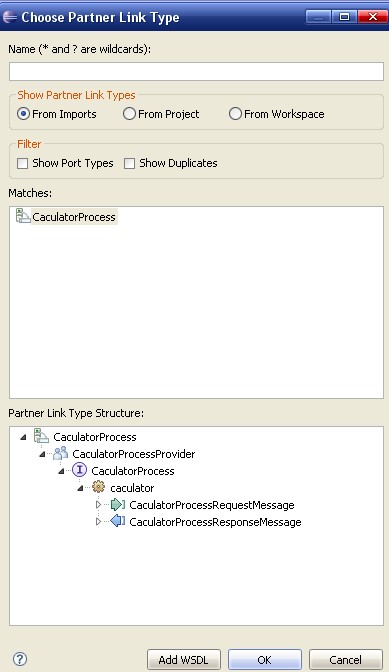
Bpel 把涉及其中的所有服务都称之为伙伴链接。Bpel 对应的 wsdl 本身也是一个伙伴链接。系统已经自动生成了。对于流程中要用到的加法服务，减法服务，应该生成对应的伙伴链接。

在最右边的 Partner Links 中，增加两个链接：addPL，subPL，分别对应加法服务和减法服务。下面具体讲下 addPL 的配置。



点选上 addPL，在属性视图中 Details 页面中点 Browser..按钮，打开一个对话框。

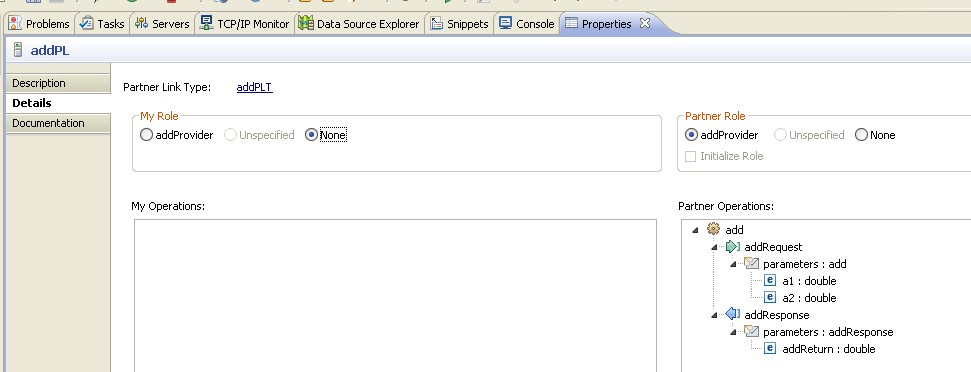
点 Add WSDL 把用到的两个 WSDL 即 AddService.wsdl 和 SubService 加进来，这样会在 CaculatorProcessArtifacts.wsdl 文件中生成两条对应的<import>语句。加完后这个对话框也会把这两个 wsdl 文件中的接口都显示出来。如上面左图所示。



一定要记住，所谓的伙伴链接一定是针对某个 **wsdl** 文件的某个接口而言的。

选上 AddService 接口，点 Ok,弹出伙伴链接类型定义，把名字取为 addPLT，点 Next。 要求输入 Role Name，这里输入 “addProvider”,点 Finish 完成。同时在属性视图中把

ParterRole 选为 addProvider.



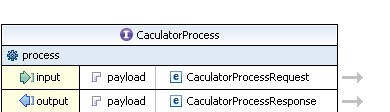
同样的方式配置 subPL，只不过把名字分别换成 subPLT，subProvider.，同时别忘记

Partner Role 要选上 subProvider.

### 5.修改CaculatorProcessArtifacts.wsdl文件.

主要是改输入输出，使之符合我们的要求。

1.1 用 open with WSDL Editor 打开该文件，转到设计视图里。

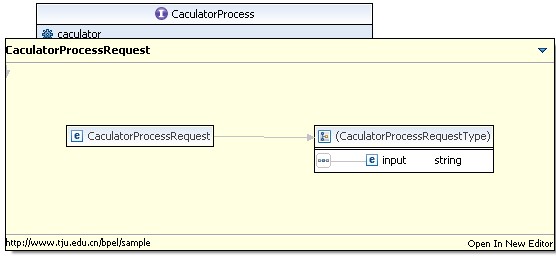


可以看到，默认生成的 WSDL 很简单，只有一个操作 process，该操作有输入 input，输出 output.

1.2 为了让操作名更有意义，首先把操作名 process 改为 caculator,可以在图上改也可以在属性视图里改。

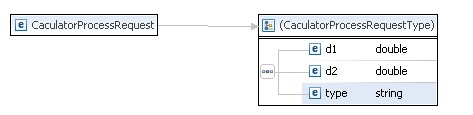
1.3 当把鼠标移到输入输出所在行右边的箭头时，可以查看输入输出消息的详细定义。

下图是输入消息：

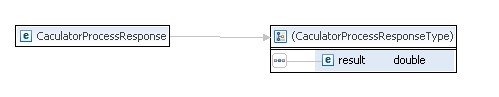


可以看到该输入消息只有一个元素 input，和我们的要求是不符合的，点击右下角的

Open In New Editor,打开一个新的 Editor 对输入消息进行配置。进去后选中类型，右键，就可以增加元素，并设置元素的类型，具体过程就不多说了，很简单的，只给出一个结果。

 咱们要做的流程是接受三个参数，前两个是 double 型数据，第三个表示是调用加法服务还是减法服务，当 type=’add’，时，调用加法服务，当 type=’sub’时调用减法服务。

同样的方法，改一下输出，把类型改为 double。



这里需要注明下。我一开始的顺序其实是：**5****3****4** 的，本想先定义好接口再导入相关的 **wsdl**，但做完后发现 **CaculatorProcessArtifacts.wsdl** 文件中的类型定义出错了。大家不妨试试，印象会深刻些。最后只好改成现在的 **3****4****5**。（这应该算是 **Eclipse BPEL** 的 **bug** 吧，现在我只好每次都先导入需要的 **wsdl**，再进行其它的操作，如果谁有好的解决方案，麻烦告诉我，我会补充进来。**zhongkem@163.com**）

### 6.创建调用服务时需要的变量。

每次 invoke 一个服务时都要有对应的变量，同时要进行赋值。我们这里要调用加法服务和减法服务。因此需要额外加入 addRequest,addResponse,subRequest,subResponse。

加完后的效果如下：

|  |  |  |
| --- | --- | --- |
|  | |  | | --- | | Input,output 分别对应流程的输入输出 | |
| |  | | --- | | addRequest,addResponse 分别对应加法服务的输入输出 | |
| |  | | --- | | subRequest,subResponse 分别  对应减法服务的输入输出 | |

### 7.配置变量

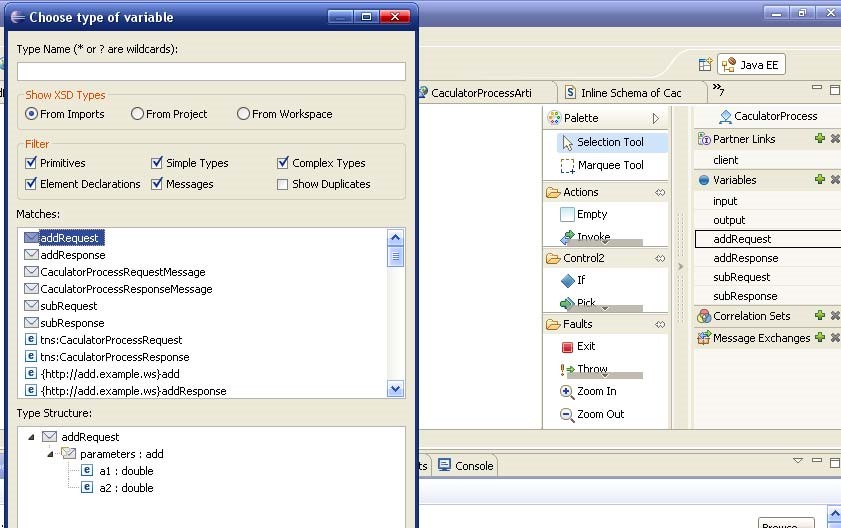
Input,output 变量都自动配置好了，对应 CaculatorProcessArtifacts.wsdl 定义的消息。

下面以 addRequest 为例讲解配置过程。

点上 addRequest 变量，属性视图中切换到 Details 页，点右上角的 Browse..按钮，打开类型选择对话框。addRequest 是加法服务的输入，因此把它的类型设为加法服务里的输入定义就可以了。

一个变量的类型可以是简单类型，也可以是复杂类型，还可以是 wsdl 中的 Message 定义。简单起见，我们这里就把 addRequest 的类型设为 AddService 服务中的 addRequest

Message。如下图所示：（**Message** 前面的复选框一定要选上）

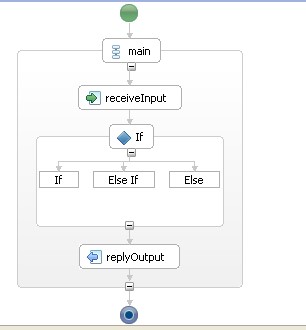


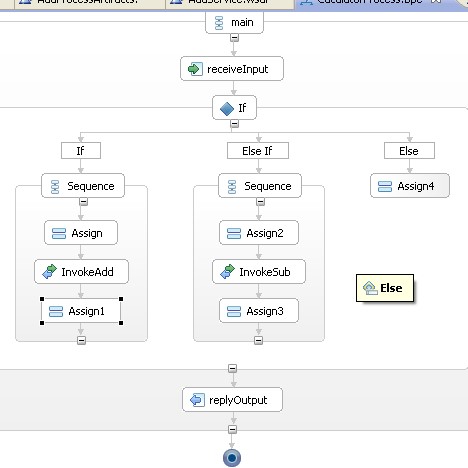
点 ok 即完成变量的定义，同样的方式定义其它三个变量。至此，准备工作就算是做好了。下一步开始真正的 bpel 流程设计。

### 8.BPEL流程设计

8.1 根据需要，我们首先是根据输入的第三个变量的值来决定调用加法服务还是减法服务，所以，显然需要一个条件 IF 语句，拖到 receiveInput 后面。

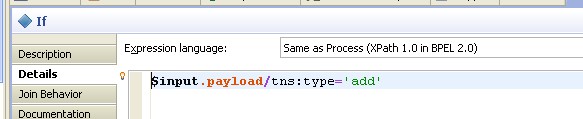
8.2 选中刚建好的 if 模块，点右键，依次点 Add ElseIf，Add Else。基本结构如下：

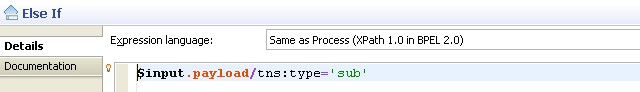


思路应该比较清晰，如果 type=’add’，则调用加法服务，如果 type=’sub’则调用减法服务，其它情况就返回错误信息。进一步完善后，如下：  接下来就是对各个结点进行具体的配置了。

8.3 If 语句的配置

选择 If,在属性视图中点开 Details 页，做如下配置：输入条件语句：**$input.payload/tns:type='add'**

 同样，选择 Else If



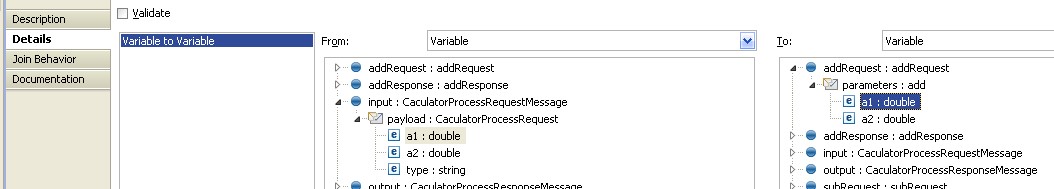
意思应该很明了，就不多解释了。

8.4 Assign 语句的配置

五个 Assign 语句的配置过程都一样，这里只讲解下第一个 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。需要注意的是，**BEPL** 中的变量赋值只能是一个一个的赋，不能直接把 **input** 赋给 **addRequest**（当然这里也不匹配，即使匹配也不行），而只能把 **input** 下的参数挨个赋给 **addRequest..**

选择 Assign，同样在属性视图中点开 Details 页，点 New 按钮，作如下的选择：



意思应该很明了，把 input 下的 a1 赋给 addRequest 下的 a1.

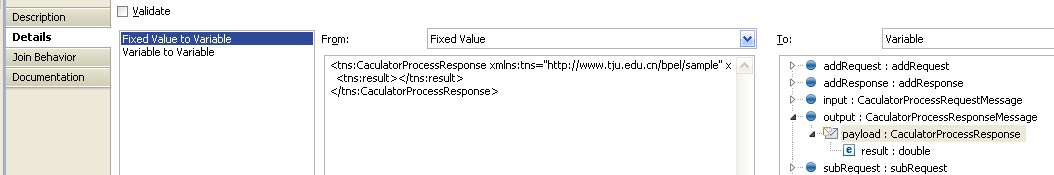
然后会弹出一个对话框，问是否需要初始化 addRequest,你点 Yes 就是了。

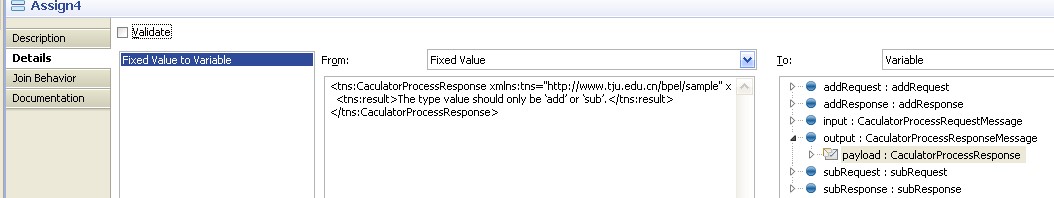
再次点 New，把 input 下的 a2 赋给 addRequest 下的 a2.

对于 Assign1，它的作用是把 addResponse 的值传给流程的输出变量 output.如果如下：

 对于 Assign2,Assign3，做出相应的配置就行了，别忘了这是针对减法服务。

对于 Assign4,直接返回一个错误消息就行了，别忘了这些变量的类型都是消息，因此直接赋值也需要返回一条相应的消息。为了不会书写错误，我首先还是点 New..，但此时 From 里需不选任何东西，直接在to里选择outputpayloadresult，然后再在别处点一下鼠标，Eclipse 会提醒你要给 output 变量赋初值。点”Yes”，则会生成如下的赋值：

 把 Variable to Variable 赋值语句删掉。然后把上面的<tns:result></tns:result>间加上一句话：The type value should only be ‘add’ or ‘sub’.

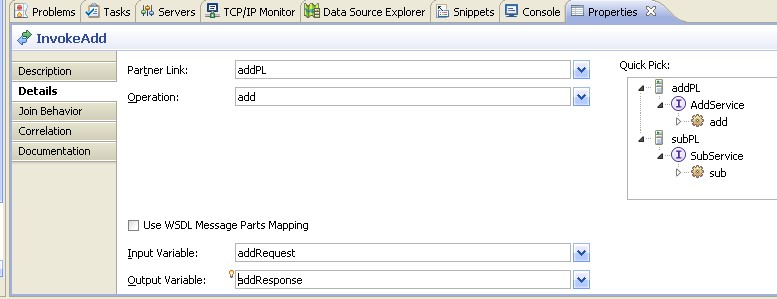


需要注意的是，To 里面选择的应该是 outputpayload，而不是下面的 result.

8..5 Invoke 的配置。

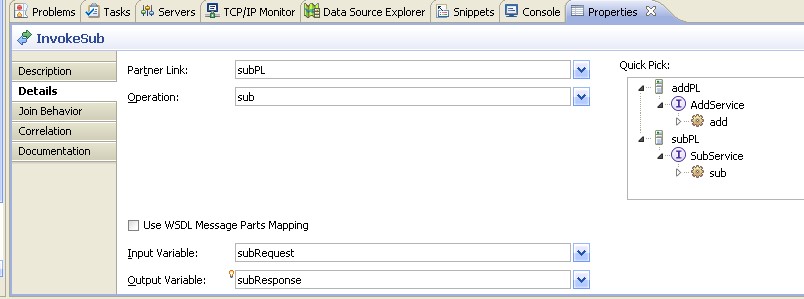
Invoke 的作用是调用伙伴链接对应接口下的对应操作，还需要指定输入输出变量。

选上 InvokeAdd,同样在属性视图的 Details 页面中进行如下配置：



如上所示，很简单，就涉及三方面内容，选择伙伴链接，选择对应操作，选择对应的输入输出变量。

同理配置 InvokeSub:

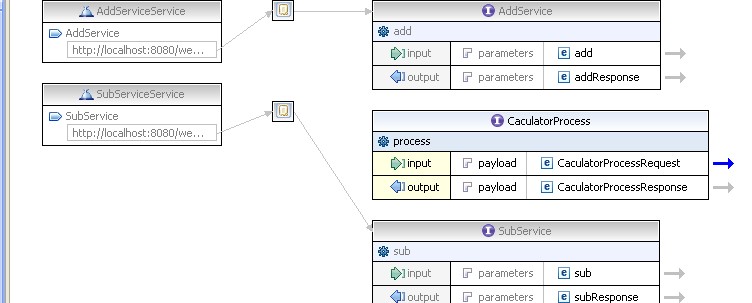


到此，流程部分已经做完了。

### 9.流程的发布

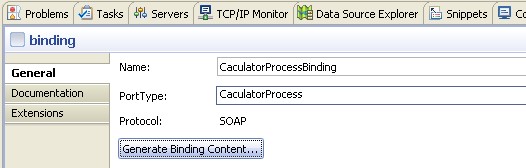
BPEL 流程最终的目的也是发布成一个 web service，只不过这个 web 服务自己基本上不实现功能，主要是通过调用别人的功能来完成任务。因此，最后也要把流程对应的 WSDL 发布出去。默认生成的 WSDL 只有类型定义，消息定义，和操作定义，缺少绑定信息和服务定义。

a) 打开 CaculatorProcessArtifacts.wsdl，此时图形如下：

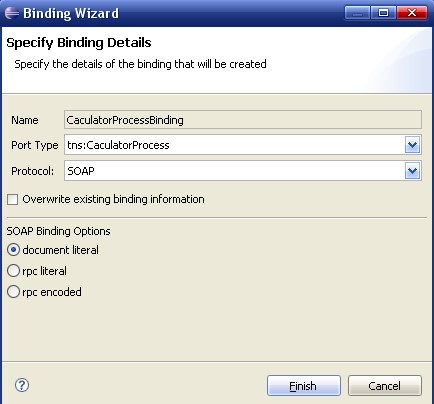


从图中可以看到，import 进来的 wsdl 也会显示在图形视图中。 b) 加入 Binding 信息

在空白处点右键，选择 Add Binding，选上创建的 Binding，进行如下的配置：



首先设置绑定名为：CaculatorProcessBinding 然后接口类型选择：CaculatorProcess 最后点 Generate Binding Content…弹出一个对话框，选择协议为 SOAP 即可。



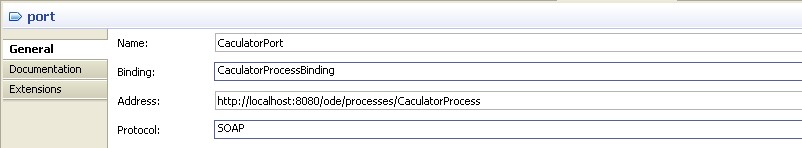
这一步所做的，对应该源代码即是下面这些代码：

|  |
| --- |
| <binding name=*"CaculatorProcessBinding"* type=*"tns:CaculatorProcess"*>  <soap:binding style=*"document"*  transport=*"http://schemas.xmlsoap.org/soap/http"* />  <operation name=*"process"*>  <soap:operation  soapAction=*"http://www.tju.edu.cn/bpel/sample/process"* /> <input>  <soap:body use=*"literal"* />  </input>    <output>  <soap:body use=*"literal"* />  </output>  </operation>  </binding> |

1. 加入服务定义

点右键，选择 Add Service,,服务名改为 CaculatorService，选择服务下的 NewPort，

进行如下定义：



源码中对应的代码即是：

<service  name=*"CaculatorService"*>

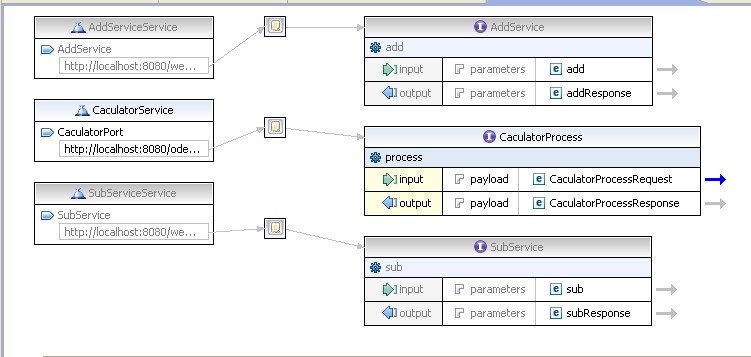
<port name=*"CaculatorPort"* binding=*"tns:CaculatorProcessBinding"*>

<soap:address location=*"http://localhost:8080/ode/processes/CaculatorProcess"* />

</port>

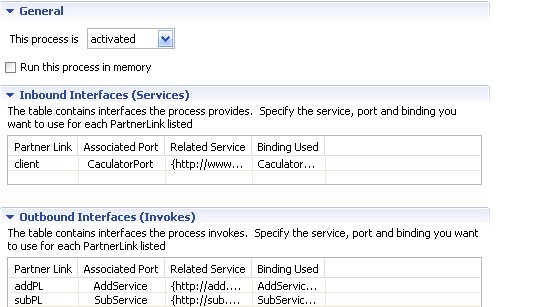
</service>

这样，一个服务就定义完全了：



1. 创建发布 ODE 发布文件

FileNewOtherBPEL 2.0 Apache ODE Deployment Descriptor，直接点 Finish 即会生成一个 deploy.xml 文件，



主要把伙伴链接对应的端口对应选上即可。

1. 启动 ODE

从 Servers 视图中启动 ODE.

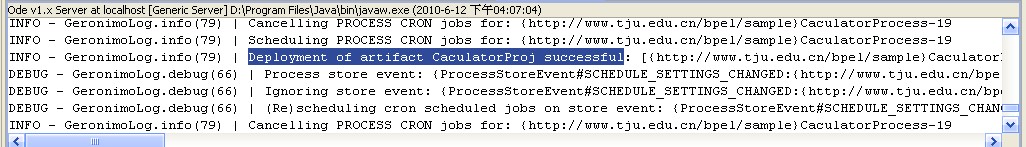
先启动 **ODE** 再把 **BPEL** 工程复制进去是为让排错，**ODE** 热发布时，如果有错，会在控制台提示错误信息，这样好排错。如果先进行 **9.6.**。就不好找错了。

1. **Bpel** 工程复制到 **ODE** 目录中

即把工作空间中的整个目录复制到 D:\JavaTools\Tomcat

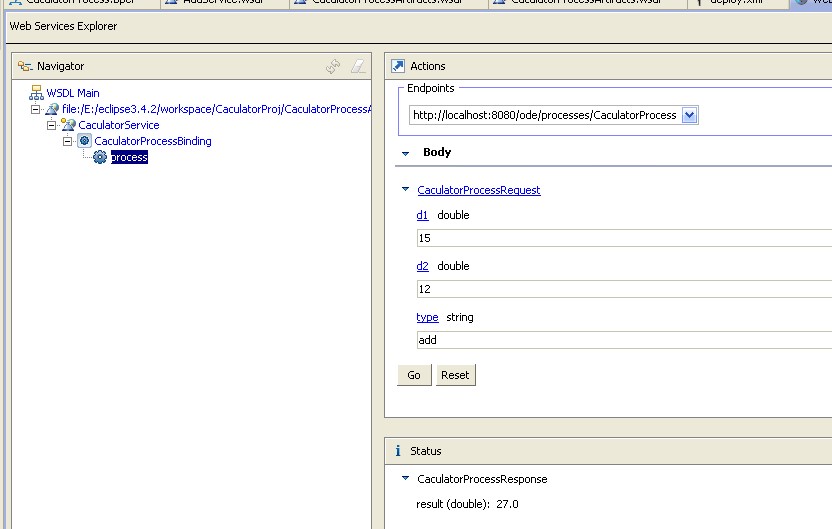
6.0\webapps\ode\WEB-INF\processes目录下，ODE会热发布，在控制台中应该

能够看到发布成功的提示：



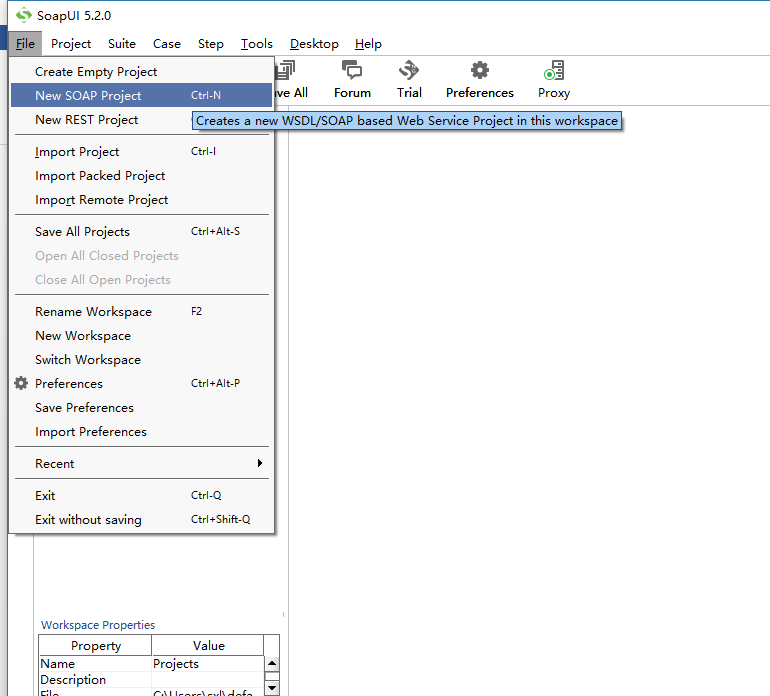
1. 运行BPEL

右击 CaculatorProcessArtifacts.wsdlWeb ServiceTest with Web Service Explore:

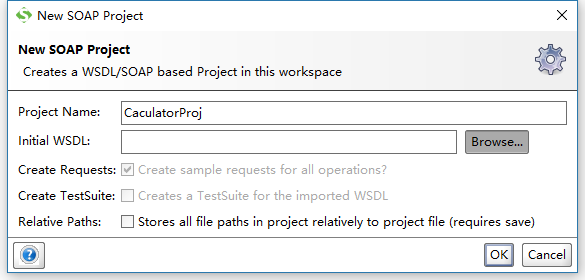


### 使用SoapUI 5.2.0 测试bpel流程

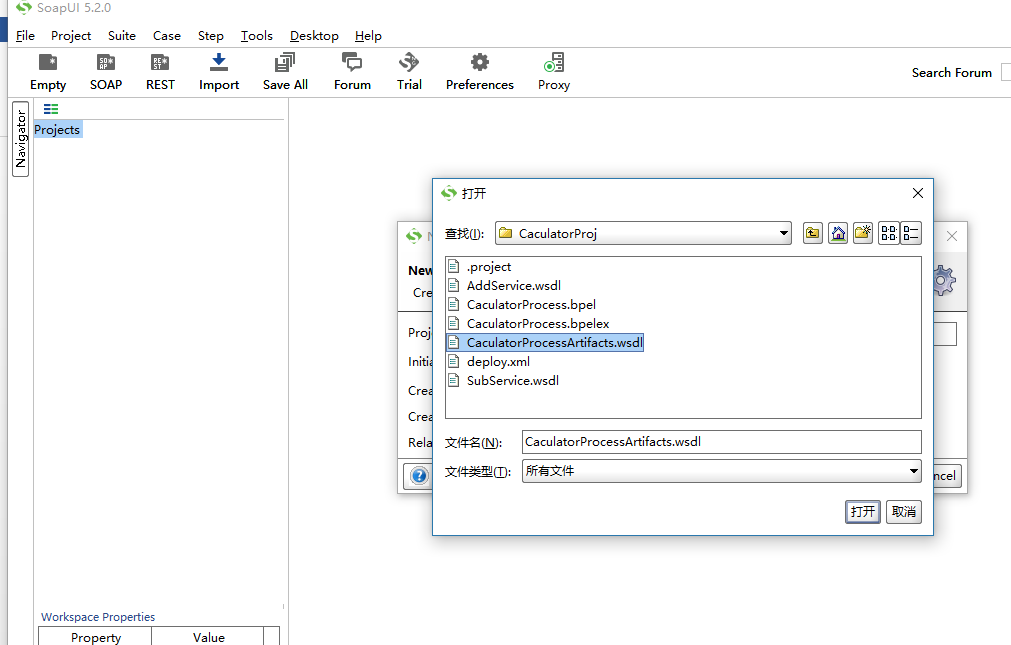
1.首先下载安装SoupUI工具，安装好之后，新建工程，如下图：

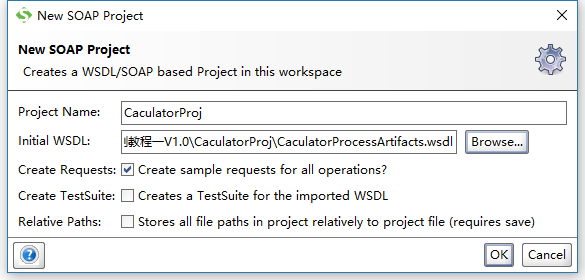


然后弹出新建工程的属性窗口，如下图所示：

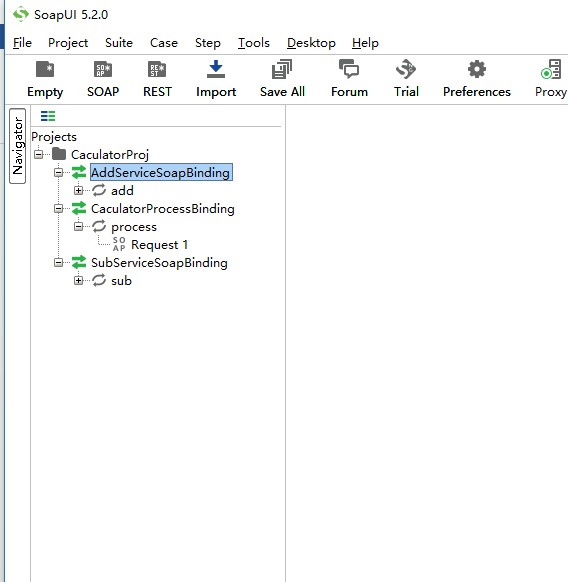


工程名称命名和bpel流程名字一样，然后点击Browse按钮，选择bpel工程目录下的WSDL文件，这个是合并后的服务生成的WSDL文件，如下图所示：

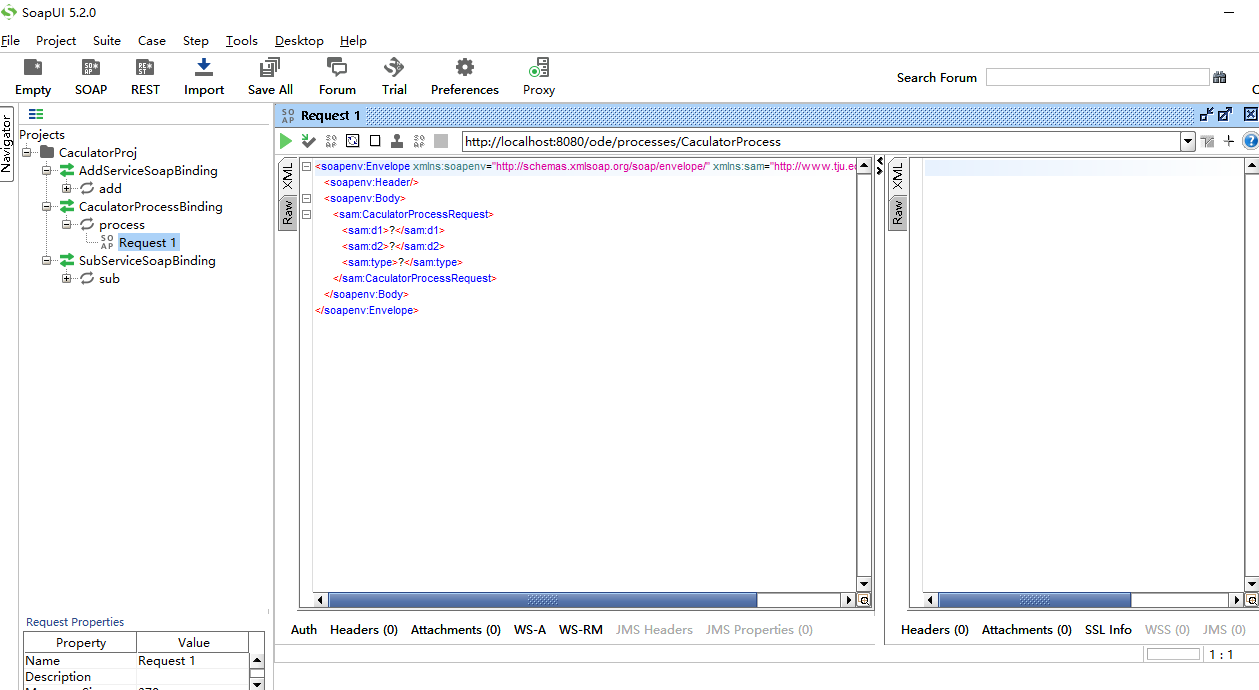




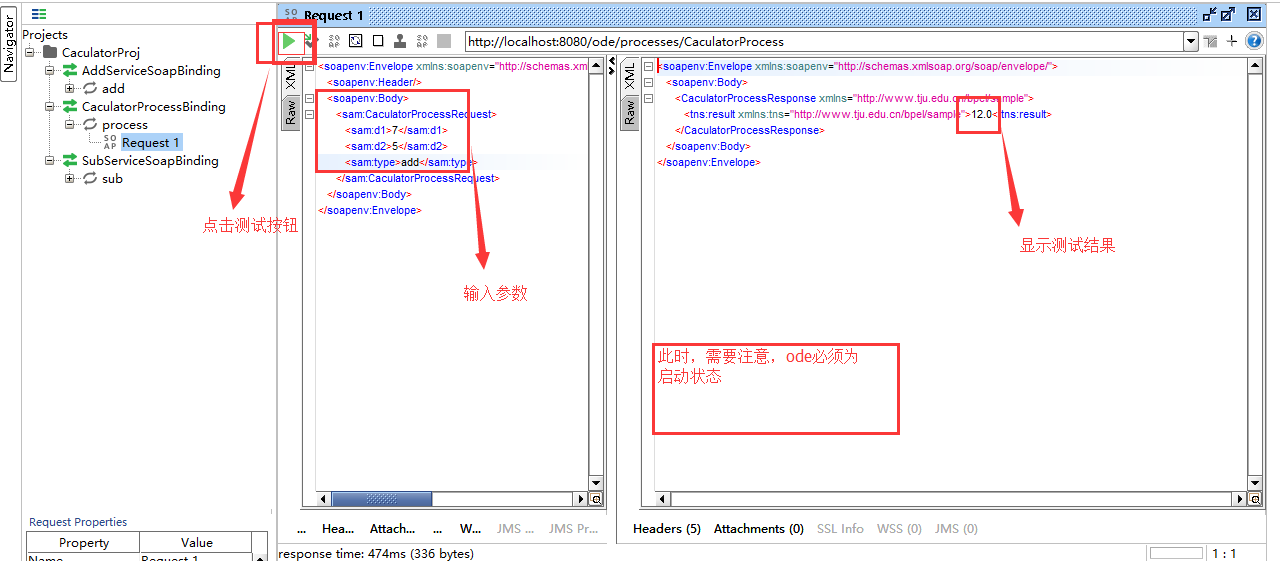
然后点击OK 出现如下界面：



然后点击process下面的的 Request1 ：



然后需要启动ode并输入参数，点击测试按钮即可，如下图所示：



# 实践总结

## 不足之处

很少写文档，搞得现在写文档的口气就像是在写博客一样，还有就是这次写的有点乱，本来搭建环境的时候是想用两种方式实现bpel流程的开发与部署的，但是文中只用了一种，虽然没写，我在这里简单的介绍一下：

第一种：使用Oracle提供的流程设计器ActiveBpel Designer设计bpel流程然后使用ActiveBPEL引擎部署运行，这个貌似挺官方的，但是流程设计器下载不到，SO。。。

第二种:就是在上文中使用的方法，使用eclipse的bpel 2.0插件，设计器 引擎一气呵成，只需要配合Tomcat+ode即可，同时eclipse的其他插件也可以同时开发相关服务（本例中使用的Java开发的服务，刚接触只是实现了一个简单的加减法，比C++服务简单的多）

## 对比集成平台的流程开发与部署

集成平台我只是简单的了解，甚至连皮毛都不算，我只发表一下当前简单的理解

共同性：都是设计相关的商业流程，通过流程设计器，引擎，以及中间件and tomcat 和自己开发的页面程序实现流程的部署与发布，通过相关的接口扩大数据的交互性。

不同点：BPD提供webservice，dll，DB，python，JS，socket等接口（用词估计不恰当，但就是这个意思），实现流程的部署还需要通信中间件（我感觉是数据量较大，需要稳定性。。。。）

而bpel定位于web方向，主要是面向服务的的整合，可以将多个服务整合到一起，比如例子中实现的一个加法服务和减法服务最后都合成到了一个主服务process上面去，而且是开源的，有相关的灵活性，但是开源的比较复杂，比较乱，各有所长。

以上观点和方法肯定有不足之处，对于bpel的理解也是非常浅显，如有错误，望大神们指出，同时也感谢大神们对bpel的研究总结。