BPMX3的WebService客户端调用

# 环境准备

Apache CXF，BPMX3是基于CXF发布WebService的，其他业务系统可以传统的WebService的调用方式来进行生成客户端，也可以用CXF提供的打包服务端的接口及相关类，作为客户端来直接调用服务端。

CXF下载地址：

<http://www.apache.org/dyn/closer.cgi?path=/cxf/2.7.0/apache-cxf-2.7.0.zip>

BPMX3提供了工作流的WebService调用方式，访问地址如：

<http://localhost:8080/bpmx3/service/ProcessService?wsdl>

# 访问WebService的客户端

## 方式一

通过以下Ant脚本可以快速生成不依赖于任何第三方包的JAVA客户端代码，生成其他语言的客户端请调用其他命令参数。

<?xml version=*"1.0"*?>

<project name=*"cxf wsdl2java"* basedir=*"."*>

<property name=*"cxf.home"* location=*"D:\tools\apache cxf\apache-cxf-2.7.0"* />

<path id=*"cxf.classpath"*>

<fileset dir=*"${cxf.home}/lib"*>

<include name=*"\*.jar"* />

</fileset>

</path>

<target name=*"cxfWSDLToJava"*>

<java classname=*"org.apache.cxf.tools.wsdlto.WSDLToJava"* fork=*"true"*>

<arg value=*"-client"* />

<arg value=*"-d"* />

<arg value=*"src"* />

<arg value=*"http://localhost:8080/bpmx3/service/ProcessService?wsdl"* />

<classpath>

<path refid=*"cxf.classpath"* />

</classpath>

</java>

</target>

</project>

生成后，调用如下所示：

**package** com.hotent.client;

**import** java.net.MalformedURLException;

**import** java.net.URL;

**import** org.junit.Test;

**import** com.hotent.platform.webservice.api.\*;

**import** com.hotent.platform.webservice.api.ProcessCmd.Variables;

**import** com.hotent.platform.webservice.api.ProcessCmd.Variables.Entry;

**import** com.hotent.platform.webservice.impl.ProcessServiceImplService;

**public** **class** ProcessRunServiceTestCase {

**private** **static** ProcessService *processService*;

**static** {

*processService*=**new** ProcessServiceImplService().getProcessServiceImplPort();

//or construct as below

// String url="http://localhost:8080/bpmx3/service/ProcessService?wsdl";

// try {

// processService=new ProcessServiceImplService(new URL(url)).getProcessServiceImplPort();

// } catch (MalformedURLException e) {

// e.printStackTrace();

// }

}

@Test

**public** **void** testStartProcess() **throws** Exception\_Exception{

ProcessCmd processCmd=**new** ProcessCmd();

//工作流定义Key

processCmd.setFlowKey("myflowKey");

//启动人员账号

processCmd.setUserAccount("csx");

//构建加入工作流的变量

Variables vars=**new** Variables();

Entry en1=**new** Entry();

en1.setKey("orderId");

en1.setValue(1l);

Entry en2=**new** Entry();

en2.setKey("orderNo");

en2.setValue("XT00001");

vars.getEntry().add(en1);

vars.getEntry().add(en2);

//设置流程变量

processCmd.setVariables(vars);

//设置业务主键

processCmd.setBusinessKey(**new** Long(1).toString());

//启动工作流，并且返回流程实例

ProcessRun processRun=*processService*.start(processCmd);

System.*out*.println("runId:" + processRun.getRunId());

}

}

## 方式二

直接利用BPMX3提供的build.xml中的jar-webservice-client的任务，完成后，

<target name="jar-webservice-client">

<echo message="将发布的Web Service 接口打包.">

</echo>

<delete dir="jar-web-client" />

<jar encoding="UTF-8" destfile="metadata/webservice/bpm-web-client.jar" basedir="src/main/webapp/WEB-INF/classes">

<include name="com/hotent/core/model/BaseModel.class" />

<include name="com/hotent/core/bpm/model/ProcessCmd.class"/>

<include name="com/hotent/core/bpm/model/ProcessTask.class"/>

<include name="com/hotent/core/page/PageBean.class"/>

<include name="com/hotent/platform/model/bpm/BpmNode.class"/>

<include name="com/hotent/platform/model/bpm/BpmVar.class"/>

<include name="com/hotent/platform/model/bpm/ProcessRun.class"/>

<include name="com/hotent/platform/model/bpm/ForkUser.class"/>

<include name="com/hotent/platform/model/bpm/TaskOpinion.class"/>

<include name="com/hotent/platform/webservice/model/BpmUser.class"/>

<include name="com/hotent/platform/webservice/model/BpmTaskOpinion.class"/>

<include name="com/hotent/platform/model/system/ResourcesUrl.class" />

<include name="com/hotent/platform/model/system/ResourcesUrlExt.class" />

<include name="com/hotent/platform/model/system/SysRole.class" />

<include name="com/hotent/platform/model/system/SubSystem.class" />

<include name="com/hotent/platform/webservice/api/\*\*" />

</jar>

</target>

在metadata/webservice/目录下会生成bpm-web-client.jar，把该jar发至客户端工程的lib目录下，并且把以下CXF依赖的包加入即可。

commons-logging-1.1.jar

geronimo-activation\_1.1\_spec-1.0-M1.jar (or Sun's Activation jar)

geronimo-annotation\_1.0\_spec-1.1.jar (JSR 250)

geronimo-javamail\_1.4\_spec-1.0-M1.jar (or Sun's JavaMail jar)

geronimo-servlet\_2.5\_spec-1.1-M1.jar (or Sun's Servlet jar)

geronimo-ws-metadata\_2.0\_spec-1.1.1.jar (JSR 181)

jaxb-api-2.1.jar

jaxb-impl-2.1.6.jar

jaxws-api-2.1.jar

jetty-6.1.5.jar

jetty-util-6.1.5.jar

neethi-2.0.jar

saaj-api-1.3.jar

saaj-impl-1.3.jar

stax-api-1.0.1.jar

wsdl4j-1.6.1.jar

wstx-asl-3.2.1.jar

XmlSchema-1.2.jar

xml-resolver-1.2.jar

调用方式如下：

**package** com.hotent.shop;

**import** java.util.Properties;

@Component

**public** **class** ProcessServiceProxy {

**private** **static** ProcessService *processService*=**null**;

@Resource

Properties configProperty;

**public** ProcessService getProcessService(){

**if**(*processService*!=**null**) **return** *processService*;

**if**(*processService*==**null**){

JaxWsProxyFactoryBean factoryBean=**new** JaxWsProxyFactoryBean();

**if**(configProperty.containsKey("service.url")){

String url=configProperty.getProperty("service.url");

factoryBean.setAddress(url);

}**else**{

factoryBean.setAddress("http://localhost:8080/bpmx3/service/ProcessService");

}

factoryBean.setServiceClass(ProcessService.**class**);

*processService*=(ProcessService)factoryBean.create();

}

**return** *processService*;

}

}

**package** com.hotent.shop;

**import** java.util.Date;

@Controller

@RequestMapping("/shipOrder/")

**public** **class** ShipOrderFormController {

@Resource

ProcessServiceProxy processServiceProxy;

@Resource

ShipOrderDao shipOrderDao;

@RequestMapping("save")

**public** ModelAndView save(HttpServletRequest request, HttpServletResponse response, ShipOrder shipOrder, BindingResult bindResult) **throws** Exception {

**boolean** isNew=shipOrder.getOrderId()==**null**?**true**:**false**;

shipOrderDao.save(shipOrder);

ProcessCmd processCmd=**new** ProcessCmd();

**try**{

ProcessService processService=processServiceProxy.getProcessService();

//启动流程

**if**(isNew){

processCmd.setFlowKey("orderCheckFlow");

//设置流程变量，流程变量由流程执行过程中使用

Map<String,Object> variables=**new** HashMap<String,Object>();

variables.put("orderId", shipOrder.getOrderId());

variables.put("total", shipOrder.getTotal());

variables.put("orderNo", shipOrder.getOrderNo());

//设置业务主键

processCmd.setBusinessKey(shipOrder.getOrderId().toString());

processCmd.setVariables(variables);

//设置启动流程的用户账号

processCmd.setUserAccount("limin");

//启动流程，并且返回流程实例，调用该功能可以放在Dao或Service层，与业务数据一起，以实现对事务的有效控制

ProcessRun processRun= processService.start(processCmd);

shipOrder.setRunId(processRun.getRunId());

shipOrderDao.save(shipOrder);

}**else**{

//流程执行下一步

String taskId=request.getParameter("taskId");

String destTask=request.getParameter("destTask");

**if**(taskId!=**null** && !"".equals(taskId)){

processCmd.setTaskId(taskId);

processCmd.setDestTask(destTask);

//设置执行流程的用户账号

processCmd.setUserAccount("limin");

//0=弃权,1=同意,2=反对,3=驳回,4=追回

processCmd.setVoteAgree((**short**)1);

//当前意见

processCmd.setVoteContent("我同意该订单");

//完成当前任务并驱动流程跳下一步，调用该功能可以放在Dao或Service层，与业务数据一起，以实现对事务的有效控制

processService.doNext(processCmd);

}

}

}**catch**(Exception ex){

ex.printStackTrace();

}

**return** **new** ModelAndView("redirect:list.htm");

}

@ModelAttribute

**protected** ShipOrder getFormObject(HttpServletRequest request, Model model) **throws** Exception{

String orderId=request.getParameter("orderId");

ShipOrder shipOrder=**null**;

**if**(orderId!=**null** && !"".equals(orderId)){

shipOrder=shipOrderDao.getByOrderId(**new** Long(orderId));

}**else**{

shipOrder=**new** ShipOrder();

shipOrder.setCreatetime(**new** Date());

}

**return** shipOrder;

}

}