**一、总结前一天**

前一天中我们讲述了如何生成一个Axis2的WebService, 如何布署以及4种不同的客户端, 它们是: 传统式, 非阻塞式, 双工模式, 双工非阻塞。

并且我们看到了一个Axis2的Web Service的布署描述：

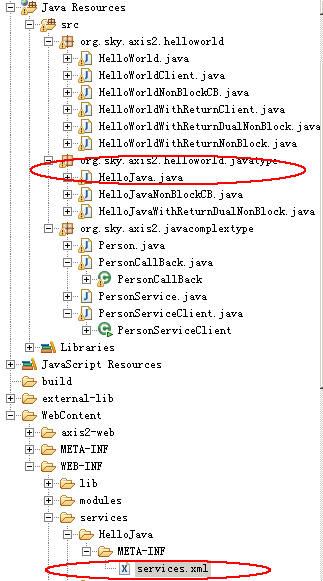
|  |
| --- |
| <service name="HelloWorld">           <parameter name="ServiceClass">org.sky.axis2.helloworld.HelloWorld</parameter>      <operation name="sayHello">          <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>          <actionMapping>urn:sayHello</actionMapping>      </operation>  </service> |

这个描述代表我们的这个Web Service的方法有一进一出两个参数，且是Axis2特有的” OMElement”类型。

那么，我们想要一个public String sayHello(String name)这样的一种简单的java类型来书写我们的WebService可以吗？

当然，只不过我们的布署描述和我们的客户端调用返回值上稍稍有一些不一样。

**二、使用简单Java类型书写我们的WebService**



**HelloJava类：**

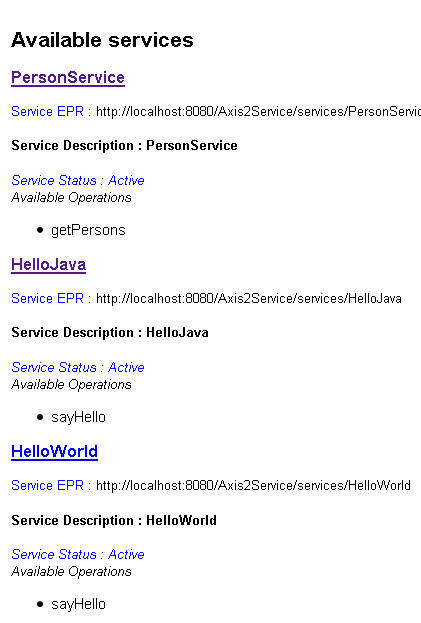
|  |
| --- |
| package org.sky.axis2.helloworld.javatype;  import javax.xml.stream.XMLStreamException;  import org.apache.axiom.om.OMElement;  public class HelloJava {           public String sayHello(String name) throws XMLStreamException {                     StringBuffer hello = new StringBuffer();                     hello.append("hello: ");                     hello.append(name);                     return hello.toString();           }  } |

**Service描述文件：**

此时我们相应的布署文件就是**service.xml**文件内容稍稍不一样，来看

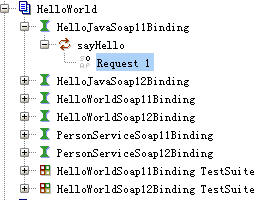
|  |
| --- |
| <service name="HelloJava">           <parameter name="ServiceClass" locked="false">                     org.sky.axis2.helloworld.javatype.HelloJava           </parameter>           <messageReceivers>                     <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"                              class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />           </messageReceivers>           <actionMapping>urn:sayHello</actionMapping>  </service> |

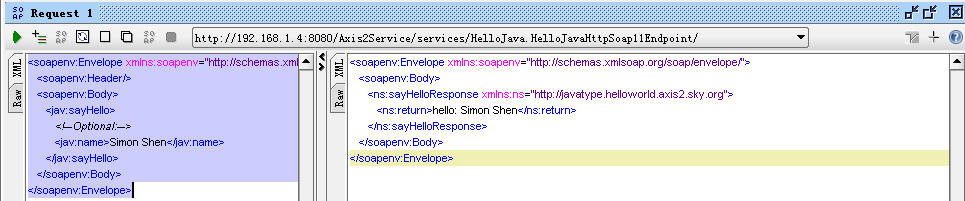
我们把这个WebService布署入Tomcat后启动起来看



是不是多了一个Service叫”HelloJava”。

我们把这个Service的wsdl地址导入SOAP UI工具并生成模拟客户端，然后再来看看它的调用与返回值。





注意这个返回值：

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">     <soapenv:Body>        <ns:sayHelloResponse xmlns:ns="http://javatype.helloworld.axis2.sky.org">  **<ns:return>hello: Simon Shen</ns:return>**        </ns:sayHelloResponse>     </soapenv:Body>  </soapenv:Envelope> |

标有**红色加粗**的部分，反正一粗，就有用就是好东西（**我喜欢粗，YEAH**）。

再来比较一下昨天我们用“org.apache.axis2.receivers.RawXMLINOutMessageReceiver”这个Service描述生成的返回值

**org.apache.axis2.receivers.RawXMLINOutMessageReceiver**

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">     <soapenv:Body>        <hel:sayHello xmlns:hel="http://helloworld.axis2.sky.org">           <!--Optional:-->  **<hel:sayHello>Monica</hel:sayHello>**        </hel:sayHello>     </soapenv:Body>  </soapenv:Envelope> |

 看到区别没有？于是，更改我们的客户端调用代码如下，在此我们使用异步双工模式：

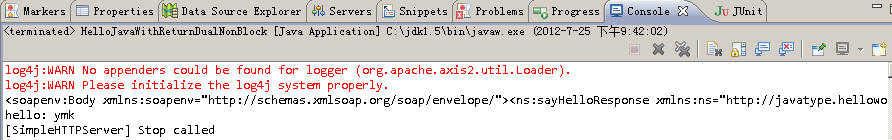
**org.sky.axis2.helloworld.javatype.HelloJavaWithReturnDualNonBlock**

|  |
| --- |
| package org.sky.axis2.helloworld.javatype;    import org.apache.axiom.om.OMAbstractFactory;  import org.apache.axiom.om.OMElement;  import org.apache.axiom.om.OMFactory;  import org.apache.axiom.om.OMNamespace;  import org.apache.axis2.AxisFault;  import org.apache.axis2.Constants;  import org.apache.axis2.addressing.EndpointReference;  import org.apache.axis2.client.Options;  import org.apache.axis2.client.ServiceClient;  import org.apache.axis2.context.ConfigurationContext;  import org.apache.axis2.context.ConfigurationContextFactory;    public class HelloJavaWithReturnDualNonBlock {           private static EndpointReference targetEPR = new EndpointReference(                              "**http://localhost:8080/Axis2Service/services/HelloJava**");           public static boolean finish = false;             public void sayHello() {                       OMFactory fac = OMAbstractFactory.getOMFactory();                     OMNamespace omNs = fac.createOMNamespace(                                       "http://javatype.helloworld.axis2.sky.org", "");                     OMElement method = fac.createOMElement("sayHello", omNs);                     OMElement name = fac.createOMElement("name", omNs);                     name.setText("ymk");                     method.addChild(name);                     method.build();                     Options options = new Options();                     options.setTo(targetEPR);                     options.setTransportInProtocol(Constants.TRANSPORT\_HTTP);                     options.setUseSeparateListener(true);                     options.setAction("urn:sayHello");                     ServiceClient sender = null;    **HelloJavaNonBlockCB callback = new HelloJavaNonBlockCB();**                     try {                              sender = new ServiceClient();                              sender.engageModule(Constants.MODULE\_ADDRESSING);                              sender.setOptions(options);                              sender.sendReceiveNonBlocking(method, callback);                              synchronized (callback) {                                       try {                                                callback.wait();                                       } catch (InterruptedException e) {                                                 e.printStackTrace();                                       }                              }                     } catch (Exception e) {                              e.printStackTrace();                     } finally {                              try {                                       sender.cleanup();                              } catch (Exception e) {                              }                     }             }             public static void main(String[] args) {                     HelloJavaWithReturnDualNonBlock testClient = new HelloJavaWithReturnDualNonBlock();                     testClient.sayHello();           }  } |

相关的CallBack类，**org.sky.axis2.helloworld.javatype.HelloJavaNonBlockCB**：

|  |
| --- |
| package org.sky.axis2.helloworld.javatype;    import java.util.Iterator;    import javax.xml.namespace.QName;    import org.apache.axiom.om.OMElement;  import org.apache.axiom.om.OMNode;  import org.apache.axis2.client.async.AxisCallback;  import org.apache.axis2.context.MessageContext;  import org.apache.axis2.databinding.utils.BeanUtil;  import org.apache.axis2.engine.DefaultObjectSupplier;    public class HelloJavaNonBlockCB implements AxisCallback {           private boolean complete = false;             public void onMessage(MessageContext msgContext) {                     System.out.println(msgContext.getEnvelope().getBody());                     OMElement element = msgContext.getEnvelope().getBody()                                       .getFirstElement();  **OMElement result = element.getFirstChildWithName(new QName(**  **"http://javatype.helloworld.axis2.sky.org", "return"));**                     System.out.println(result.getText());                     synchronized (this) {                              this.notify();                     }           }             public boolean isComplete() {                     return complete;           }             public void onFault(MessageContext msgContext) {                     System.out.println(msgContext.getEnvelope().getBody().getFault()                                       .toString());                     synchronized (this) {                              this.notify();                     }           }             public void onError(Exception e) {                     e.printStackTrace();                     synchronized (this) {                              this.notify();                     }           }             public void onComplete() {                     this.complete = true;                     synchronized (this) {                              this.notify();                     }           }  } |

下面是客户端运行后的输出：



注意在CallBack类中的这一句：

**OMElement result = element.getFirstChildWithName(new QName(**

**"http://javatype.helloworld.axis2.sky.org","return"));**

和下面这个SOAP UI的返回值来作个比较“**注意这个Return**”

“**<ns:return>**hello: Simon Shen**</ns:return>**”

**三、深入理解Axis2的API的写法**

我们一边看着SOAP UI给我们生成的调用端和结果来看这些个API。

Axis2中的一切都是一种OMElement，它其实就是一个“**XML**”结构的Java对象。

**3.1 先来看调用端的生成**

下面是Soap UI帮我们生成的客户端

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:jav="http://javatype.helloworld.axis2.sky.org">     <soapenv:Header/>     <soapenv:Body>        <jav:sayHello>           <!--Optional:-->           <jav:name>Simon Shen</jav:name>        </jav:sayHello>     </soapenv:Body>  </soapenv:Envelope> |

通过上述的内容我们可以知道，调用一个以wsdl方式暴露的Web Service的客户端需要下面这些元素：

1)      调用地址 end point

|  |
| --- |
| **private static EndpointReference targetEPR = new EndpointReference(**  **"http://localhost:8080/Axis2Service/services/HelloJava");** |

2)      Web Service方法，注意service.xml描述中的这一行

|  |
| --- |
| <actionMapping>urn:sayHello</actionMapping> |

这个被称为我们的Web Method，它指向了我们在类中的：

|  |
| --- |
| public String sayHello(String name) throws XMLStreamException { |

该方法有一个返回，一个进参，返回已经我们的service.xml描述中用：

|  |
| --- |
| <messageReceivers>              <messageReceiver mep="http://www.w3.org/2004/08/wsdl/in-out"                       class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />  </messageReceivers> |

它的进参我们假设是一个String类型的。

这样一个Web Method的原始调用（其实我们把它称为SOAP调用）应该是如SOAP UI工具中给我们生成的语句那样：

|  |
| --- |
| <jav:sayHello>           <!--Optional:-->           <jav:name>Simon Shen</jav:name>      </jav:sayHello> |

因此，为了生成上述的SOAP语句，我们采Axis2的API书写如下，下面的代码的效果将会在程序运行时产生等同于上面的SOAP调用语句，一起来看看：

|  |
| --- |
| OMFactory fac = OMAbstractFactory.getOMFactory();             OMNamespace omNs = fac.createOMNamespace(                               "http://javatype.helloworld.axis2.sky.org", "");             OMElement method = fac.createOMElement("sayHello", omNs);             OMElement name = fac.createOMElement("name", omNs);             name.setText("ymk");             method.addChild(name);  **method.build();** |

当”method.build()”命令发出后，上述这些API就生成SOAP调用语句，去调用”**sayHello**”这个方法，然后往里面传入了一个String类型的值。

可以看到，这个API非常像dom4j的语句，只是这边不是一个document类型而是一个类XML的OMElement结构体，都是addChild, setText之类的，对吧？

再继续往下看我们的API

|  |
| --- |
| Options options = new Options();             options.setTo(targetEPR);             options.setTransportInProtocol(Constants.TRANSPORT\_HTTP);             options.setUseSeparateListener(true);                     options.setAction("urn:sayHello"); |

1)      将该web method的调用指向指定的end point

2)      开启双工模式

3)      因为是双工模式，需要在client端开启一个监听器

4)      设置调用的web method为”urn:sayHello”

|  |
| --- |
| sender = new ServiceClient();                              sender.engageModule(Constants.MODULE\_ADDRESSING);                              sender.setOptions(options);                              sender.sendReceiveNonBlocking(method, callback); |

1)      生成调用句柄

2)      由于我们采用的是双工模式，因此需要寻址:engageModule

这个engageModule就是需要访问你的工程的WEB-INF\modules\目录下的一个叫addressing-1.4.mar的文件。

**因此在调用engageModule语句之间有两种方式**来调用你的WEB-INF\modules目录下的addressing-1.4.mar文件。

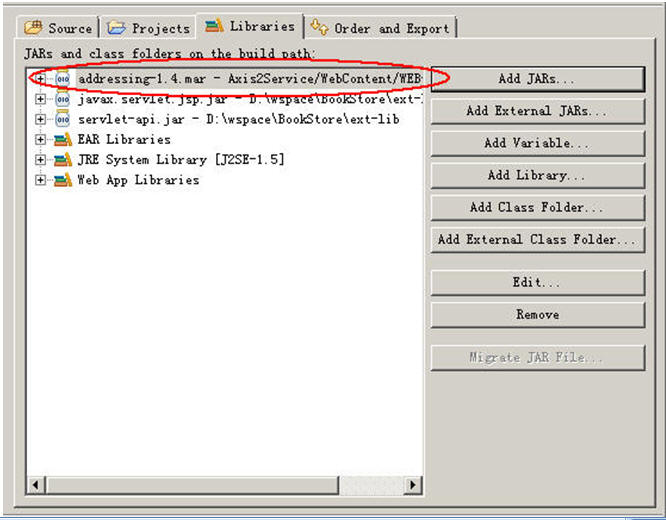
**第一种方式：**

|  |
| --- |
| ConfigurationContext sysContext = ConfigurationContextFactory                                         .createConfigurationContextFromFileSystem(                                                           "D:\\wspace\\Axis2Service\\WebContent\\WEB-INF",                                                           null);  **sender = new ServiceClient(sysContext, null);**                              sender.engageModule(Constants.MODULE\_ADDRESSING); |

**第二种方式：**

|  |
| --- |
| sender = new ServiceClient(sysContext, null);           sender.engageModule(Constants.MODULE\_ADDRESSING); |

在第二种方式中，不需要为new ServiceClient()指定第一个sysContext参数，但是，你必须把WEB-INF\modules\addressing-1.4.mar指定到你的工程的classpath中去，如下图



要不然运行时会抛出下面这个exception:

*org.apache.axis2.AxisFault:Unable to engage module : addressing*

1)      使用非阻塞方式调用:sendReceiveNonBlocking(方法, callback类)

最后我们在CallBack具体的实类中我们巧妙使用了线程的wait()和notify()来实现“阻塞”。

**3.2 再来看如何解析返回的值**

由于返回的值也是一个OMElement结构体，来看SOAP UI工具中给我们生成的返回结果

|  |
| --- |
| <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">     <soapenv:Body>        <ns:sayHelloResponse xmlns:ns="http://javatype.helloworld.axis2.sky.org">           <ns:return>hello: Simon Shen</ns:return>        </ns:sayHelloResponse>     </soapenv:Body>  </soapenv:Envelope> |

其实，我们需要做的就是解析这个SOAP的返回包（soap response）。

|  |
| --- |
| public void onMessage(MessageContext msgContext) {                     System.out.println(msgContext.getEnvelope().getBody());                     OMElement element = msgContext.getEnvelope().getBody()                                       .getFirstElement();                     OMElement result = element.getFirstChildWithName(new QName(                                       "http://javatype.helloworld.axis2.sky.org", "return"));                     System.out.println(result.getText());                     synchronized (this) {                              this.notify();                     }  } |

在SOAP的世界中，这个namespace很重要，相当于一个头标记，如果我们在调用下面这句话时

|  |
| --- |
| OMElement result = element.getFirstChildWithName(new QName(                                       "**http://javatype.helloworld.axis2.sky.org**", "return")); |

没有加[http://javatype.helloworld.axis2.sky.org](http://javatype.helloworld.axis2.sky.org/)，那么这段soapresponse将无法被正确定位到

|  |
| --- |
| <ns:return>hello: Simon Shen</ns:return> |

那么你将得到NullPointerException。

相信，经过这样的讲解，因该能够使你看得懂一个Axis2的API客户端了吧，从此处我们可以发觉，很多语句都是“**固化**”的，和EJB的调用一样，一堆**固化**的语句，写多了，理解起来就并不难了，呵呵。

**四、复杂类型的Webservice**

更多的时候，我们的Service需要返回类似于List<Person>, List<String>这样的数据结构，因为有时我们先要在server端通过数据库取值，然后再把值返回给客户端，我们就一起来看如何生成一个具有复杂类型的Web Service以及它相应的客户端调用的写法吧。

下面是Service端：

**org.sky.axis2.javacomplextype.PersonService**

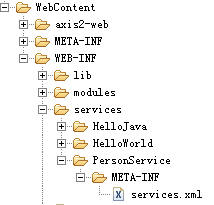
|  |
| --- |
| package org.sky.axis2.javacomplextype;    import org.apache.axiom.om.OMElement;  import org.apache.axis2.databinding.utils.BeanUtil;  import java.util.\*;  import javax.xml.namespace.QName;    public class PersonService {           public OMElement getPersons(OMElement person) {                     Person man = new Person();                     List<Person> list = new ArrayList<Person>();                     man.setName("Wallance Shao");                     man.setAge("25");                     list.add((Person) man.clone());                       man.setAge("11");                     man.setName("Hong Wayne");                     list.add((Person) man.clone());                     OMElement omElement = BeanUtil.getOMElement(new QName("root"),                                       list.toArray(), new QName("person"), false, null);                     return omElement;           }  } |

它生成了一个List<Person>的复杂类型，它的返回类型必须为OMElement结构，下面再来看Person.java类：

**org.sky.axis2.javacomplextype.Person**

|  |
| --- |
| package org.sky.axis2.javacomplextype;  import java.io.\*;  public class Person implements Serializable,Cloneable{             private String name="";           private String age="";           public String getName() {                     return name;           }           public void setName(String name) {                     this.name = name;           }           public String getAge() {                     return age;           }           public void setAge(String age) {                     this.age = age;           }           public Object clone(){                     Object o = null;                     try{                              o=super.clone();                     }catch(Exception e){                              e.printStackTrace();                              o=null;                     }                     return o;           }  } |

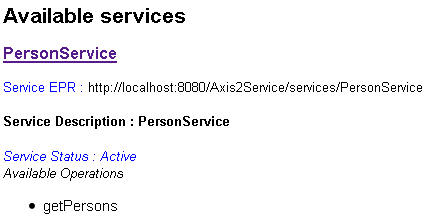
然后我们编写我们的service.xml文件



它的内容如下：

|  |
| --- |
| <service name="PersonService">           <parameter name="ServiceClass">org.sky.axis2.javacomplextype.PersonService</parameter>      <operation name="getPersons">          <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>          <actionMapping>urn:getPersons</actionMapping>      </operation>  </service> |

把它布署进我们的tomcat，启动后可以得到一个新的service



书写我们的客户端，此处我们使用非阻塞式写法。

**注意：**

把Server端的Person.java拷贝一份到客户端的package中去，一定记得，要不然造型时会出错

**org.sky.axis2.javacomplextype.PersonServiceClient**

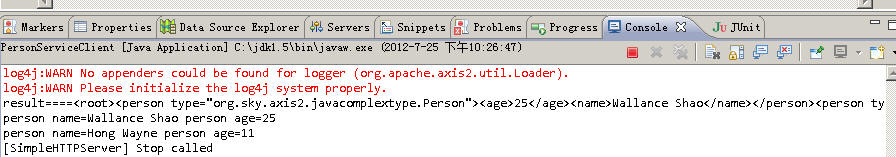
|  |
| --- |
| package org.sky.axis2.javacomplextype;    import java.util.Iterator;    import javax.xml.namespace.QName;    import org.apache.axis2.AxisFault;  import org.apache.axis2.Constants;  import org.apache.axis2.addressing.EndpointReference;  import org.apache.axis2.client.Options;  import org.apache.axis2.client.ServiceClient;  import org.apache.axis2.context.ConfigurationContext;  import org.apache.axis2.context.ConfigurationContextFactory;  import org.apache.axiom.om.OMAbstractFactory;  import org.apache.axiom.om.OMElement;  import org.apache.axiom.om.OMFactory;  import org.apache.axiom.om.OMNamespace;  import org.apache.axis2.client.async.AxisCallback;  import org.apache.axis2.context.MessageContext;    public class PersonServiceClient {           private static EndpointReference targetEPR = new EndpointReference(                              "http://localhost:8080/Axis2Service/services/PersonService");             public void getPersons() {                     Options options = new Options();                     options.setAction("urn:getPersons");                     options.setTo(targetEPR);                     options.setTransportInProtocol(Constants.TRANSPORT\_HTTP);                     options.setUseSeparateListener(true);                     ServiceClient sender = null;                     try {                              PersonCallBack callback = new PersonCallBack();                              sender = new ServiceClient();                              sender.setOptions(options);                              sender.engageModule(Constants.MODULE\_ADDRESSING);                              OMFactory fac = OMAbstractFactory.getOMFactory();                              OMNamespace omNs = fac.createOMNamespace(                                                 "http://javacomplextype.axis2.sky.org", "");                              OMElement method = fac.createOMElement("getPersons", omNs);                              OMElement person = fac.createOMElement("person", omNs);                              person.setText("");                              method.addChild(method);                              sender.sendReceiveNonBlocking(method, callback);                              synchronized (callback) {                                       try {                                                 callback.wait();                                       } catch (InterruptedException e) {                                                 e.printStackTrace();                                       }                              }                     } catch (Exception e) {                              System.out.println("-------Error Occured-------");                              e.printStackTrace();                     } finally {                              try {                                       sender.cleanup();                              } catch (Exception e) {                              }                     }           }             public static void main(String[] args) {                     PersonServiceClient testClient = new PersonServiceClient();                     testClient.getPersons();             }  } |

由于是非阻塞式，因此需要写CallBack具体实现类，下面是我们的CallBack具体实现类，注意红色加粗部分为我们的核心代码。

**org.sky.axis2.javacomplextype.PersonCallBack**

|  |
| --- |
| package org.sky.axis2.javacomplextype;    import java.util.Iterator;    import javax.xml.namespace.QName;    import org.apache.axiom.om.OMElement;  import org.apache.axiom.om.OMNode;  import org.apache.axis2.client.async.AxisCallback;  import org.apache.axis2.context.MessageContext;  import org.apache.axis2.databinding.utils.BeanUtil;  import org.apache.axis2.engine.DefaultObjectSupplier;    public class PersonCallBack implements AxisCallback {             public void onMessage(MessageContext msgContext) {  **OMElement result = msgContext.getEnvelope().getBody().getFirstElement();**  **System.out.println("result====" + result);**  **if (result == null) {**  **return;**  **}**  **Iterator iterator = result.getChildElements();**  **while (iterator.hasNext()) {**  **OMNode omNode = (OMNode) iterator.next();**  **if (omNode.getType() == OMNode.ELEMENT\_NODE)**{  **OMElement omElement = (OMElement) omNode;**    **if (omElement.getLocalName().toLowerCase().equals("person")) {**  **try {**  **Person person = (Person) BeanUtil.processObject(**  **omElement, Person.class, null, true,**  **new DefaultObjectSupplier());**  **System.out.println("person name=" + person.getName()**  **+ " person age=" + person.getAge());**  **} catch (Exception e) {**  **System.out.println("-------Error Occured-------");**  **e.printStackTrace();**  **}**                                       }                              }                     }                     synchronized (this) {                              try {                                       this.notify();                              } catch (Exception e) {                              }                     }           }             public void onFault(MessageContext msgContext) {                     System.out.println(msgContext.getEnvelope().getBody().getFault()                                       .toString());                     synchronized (this) {                              try {                                       this.notify();                              } catch (Exception e) {                              }                     }           }             public void onError(Exception e) {                     synchronized (this) {                              try {                                       this.notify();                              } catch (Exception ex) {                              }                     }                     e.printStackTrace();           }             public void onComplete() {                     synchronized (this) {                              try {                                       this.notify();                              } catch (Exception e) {                              }                     }           }  } |

运行我们的PersonServiceClient类，得到输出如下：



完成我们Axis2的第二天的课程！！！