# 使用SAAJ 和JAXM的 SOAP客户端及服务

2019-01-30 11:39:02 [小师兄.](https://me.csdn.net/u014252478" \t "https://blog.csdn.net/u014252478/article/details/_blank) 阅读数 168更多

分类专栏： [10. Java之WebService](https://blog.csdn.net/u014252478/article/category/8036109" \t "https://blog.csdn.net/u014252478/article/details/_blank)

版权声明：本文为博主原创文章，遵循[CC 4.0 BY-SA](http://creativecommons.org/licenses/by-sa/4.0/" \t "https://blog.csdn.net/u014252478/article/details/_blank)版权协议，转载请附上原文出处链接和本声明。

本文链接：<https://blog.csdn.net/u014252478/article/details/86700996>

# **sun公司的一个很好的学习平台：**

[https://docs.huihoo.com/one\_and\_net/app7/12.1.htm#02](https://docs.huihoo.com/one_and_net/app7/12.1.htm" \l "02)

本模块讲述了如何使用用于JavaTM 的带附件的SOAP（SAAJ）和用于XML消息的Java TM API（JAXM）创建能够发送和接收消息的客户端，以及如何部署它们到Sun ONE Application Server上。本模块包括以下几节：

* [SOAP客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "01" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [SOAP服务](https://blog.csdn.net/u014252478/article/details/86700996" \l "02" \t "https://blog.csdn.net/u014252478/article/details/_self)

## ****SOAP客户端****

本节讲述了SOAP客户端可以采用的两种消息模型以及开发和部署这类客户端的过程。本节包括以下主题：

* [SOAP客户端消息模型](https://blog.csdn.net/u014252478/article/details/86700996" \l "03" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [开发SOAP客户端](https://docs.huihoo.com/one_and_net/app7/12.1.htm" \l "04)
* [组装并部署SOAP客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "05" \t "https://blog.csdn.net/u014252478/article/details/_self)

### ****SOAP客户端消息模型****

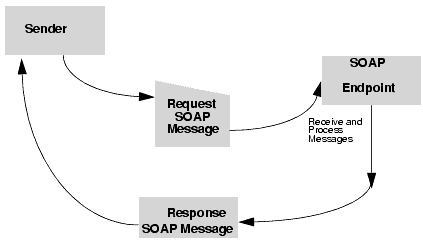
可以使用以下两种消息模型建立SOAP客户端

* [没有消息提供者的客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "06" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [使用消息提供者的客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "04" \t "https://blog.csdn.net/u014252478/article/details/_self)

****没有消息提供者的客户端****

不使用消息提供者的应用程序只能交换同步消息。也就是说，扮演客户端角色的应用程序只能发送请求-响应消息。这种类型的客户端采用SAAJ API的SOAPConnection方法。下图演示了在没有消息提供者的情况下，同步消息如何在发送者和接收者之间交换。

****不使用消息提供者的SOAP消息****



不使用消息提供者的客户端具有以下优点：

* 可以采用J2SE平台编写应用程序。
* 不需要在servlet或J2EE容器中部署应用程序。
* 不需要配置消息提供者。

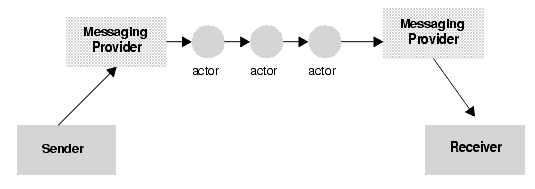
不使用消息提供者的客户端具有以下局限性：

* 客户端只能发送请求-响应消息
* 客户端只能扮演客户端角色

****使用消息提供者的客户端****

如果想要获得并且保存在任何时间发送给你的请求，你必须使用消息提供者。JAXM API提供了使用消息提供者发送和接收消息的框架。你需要在容器中运行客户端，容器提供了消息基础结构让提供者使用。下图演示了在使用消息提供者的情况下，异步消息如何在发送者和接收者之间交换。

****使用消息提供者的SOAP消息****



使用消息提供者的客户端具有以下优点：

* 客户端能够扮演客户端或者服务角色
* 客户端能够切换消息传递给提供者
* 在客户端传递消息到最终接收者之前，它能够发送消息到一个或多个目的地。这些中间的消息接收者被称为*actor*，它们在消息的SOAPHeader对象中被指定。
* 客户端能够利用任何提供者支持的SOAP消息协议和影响消息类型与可靠性的‘服务质量’，以及消息传递服务的质量。

****注意****

Sun ONE Application Server包含了一个示例JAXM提供者，它演示了如何使用提供者为发送客户端激活“发后不理（fire and forget）”消息。请察看示例文档获取关于如何激活、部署和使用它的全面信息。示例应用程序可以从以下位置得到:

*install\_dir*/samples/webservices/jaxm/jaxm-provider/

Sun ONE Application Server的未来版本将会包括支持可靠SOAP消息和ebXML消息的JAXM提供者。

### **SOAP消息**

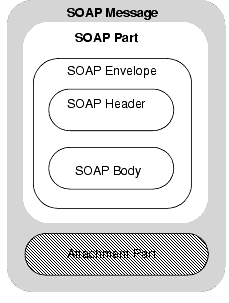
本节向你介绍SOAP消息的结构和组成部分、如何访问它们、以及如何处理SOAP消息。本节讲述以下主题：

* [SOAP消息的组成部分](https://blog.csdn.net/u014252478/article/details/86700996" \l "08" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [访问消息的元素](https://blog.csdn.net/u014252478/article/details/86700996" \l "09" \t "https://blog.csdn.net/u014252478/article/details/_self)

****SOAP消息的组成部分****

SOAP消息是一个包括SOAPEnvelope、一个可选的SOAPHeader、以及SOAPBody的XML文件。SOAP消息头包含了使消息在到达最终目的地之前，能够被路由到一个或多个中间节点的信息。

SOAP消息的组成部分



这个图展示了SOAP消息的结构和组成部分。不同对象代表了SOAP消息的每一个部分。

SOAPMessage对象包括：

* 一个SOAPPart对象，其中包括
  + 一个SOAPEnvelope对象，其中包括
    - 一个空的SOAPHeader对象 – 可选，包括它是为了方便，因为大多数消息都要用到它。
    - 一个空的SOAPBody对象 -可以容纳消息的内容，还能容纳包含了状态信息或者消息故障明细的错误消息。
* AttachmentPart可能容纳普通文本或者图像文件。

SOAPEnvelope是代表消息的XML文件的根元素。它为消息如何处理、由谁处理定义了框架。XML内容从SOAPEnvelope开始。

SOAPHeader是添加特性到SOAP消息的基本机制。它可以容纳任意数目的扩展了基础协议的子元素。例如，header子元素可能会定义认证信息、事务信息、本地信息、等等。处理消息的软件可以在没有事先约定的情况下，使用这个机制定义谁应该处理某个特性，以及该特性是强制的还是可选的。

body是发给消息最终接收者的强制信息的容器。SOAP消息还可以容纳一个附件，它不一定非得是XML文件。

****访问消息的元素****

当创建消息体或附件的时候，以及处理消息的时候，你需要访问消息的组成部分。

SOAPMessage对象含有一个SOAPPart对象。通过message对象可以得到它。

SOAPPart soapPart = message.getSOAPPart();

然后你可以利用SOAPPart获得它所含有的SOAPEnvelope对象。

SOAPEnvelope envelope = soapPart.getEnvelope();

现在你可以利用envelope获得它的空SOAPHeader和SOAPBody对象。

SOAPHeader header = envelope.getHeader();

SOAPBody body = envelope.getBody();

SOAPBody对象最初是空的。

****命名空间****

XML命名空间是一种限定元素和属性名称的方法，目的是消除它们与同一文件中其他名称之间产生的歧义。一个名确的XML命名空间声明采用如下格式：

<prefix:myElement

xmlns:prefix ="URI">

声明定义prefix作为一个特定URI的别名。在myElement元素内，你可以对任何元素或属性使用前缀，从而指定元素或属性属于URI指定的命名空间。下面的代码是命名空间声明的一个例子。

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">

该定义定义了SOAP\_ENV作为命名空间http://schemas.xmlsoap.org/soap/envelope/

的别名。

定义了别名之后，可以使用它作为Envelope元素中任何属性或元素的前缀。

****预定义的SOAP命名空间****

* SOAPEnvelope，SOAP消息的根元素，拥有下面的命名空间标识http://schemas.xmlsoap.org/soap/envelope。
* SOAP serialization，定义SOAP序列化规则的URI，拥有下面的命名空间标识"http://schemas.xmlsoap.org/soap/encoding".。

当使用SAAJ或JAXM构造或者消费消息的时候，你有责任保证正确设定或处理命名空间，并且丢弃含有错误命名空间的消息。

****创建SOAP消息的时候使用命名空间****

当创建SOAP消息的body或header元素的时候，你必须使用命名空间为元素指定一个完整的名称。通过调用方法SOAPEnvelope.createName可以获得一个名称。

调用该方法时，可以传递一个本地名称作为参数，或者可以指定本地名称、前缀、以及URI。例如，下面的命令行定义了一个名为bodyName的Name对象。

Name bodyName = MyEnvelope.createName("TradePrice", "GetLTP", "http://foo.eztrade.com");

这和下面的命名空间声明作用相同。

<GetLTP:TradePrice xmlns:GetLTP= "http://foo.eztrade.com">

下面的代码演示了如何创建一个名称并把它和SOAPBody元素绑定。注意createName方法的使用和位置。

SoapBody body = envelope.getBody();//get body from envelope

Name bodyName = envelope.createName("TradePrice", "GetLTP", "http://foo.eztrade.com");

SOAPBodyElement gltp = body.addBodyElement(bodyName);

### ****开发SOAP客户端****

利用SAAJ，客户端能够以点对点的方式创建并且发送SOAP消息。JAXM为使用消息提供者的XML消息定义了API。JAXM依赖用于 Java 的带有附件的 SOAP API（SAAJ），它定义了在Java中操作带有附件的SOAP消息模型的API。Sun ONE Application Server不包括它所支持的JAXM消息提供者。然而它的确包含了示例应用程序，一个简单的JAXM提供者，演示了一个消息提供者如何处理来自客户端的异步SOAP消息。本节讲述以下主题：

* [SOAP消息是如何出现的？](https://blog.csdn.net/u014252478/article/details/86700996" \l "101" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [创建SOAP客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "102" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [组装并部署SOAP客户端](https://blog.csdn.net/u014252478/article/details/86700996" \l "103" \t "https://blog.csdn.net/u014252478/article/details/_self)

****SOAP消息是如何出现的？****

当消息工厂生产的SOAP消息通过连接被发送到一个端点的时候，SOAP消息就出现了。本节讲述以下主题：

* [端点](https://blog.csdn.net/u014252478/article/details/86700996" \l "104" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [连接](https://blog.csdn.net/u014252478/article/details/86700996" \l "105" \t "https://blog.csdn.net/u014252478/article/details/_self)

****端点****

端点标识消息的最终目的地。端点是由URLEndpoint类来定义的。如果不使用提供者，你可以构造或者查找到一个消息端点。

****构造端点****

你可以通过调用构造函数或者在名称服务中查找来初始化一个端点。

以下代码使用构造函数创建一个URLEndpoint：

myEndpoint = new URLEndpoint("http://*host*/myServlet")

****使用端点发送消息****

指定端点作为SOAPConnection.call方法的一个参数，该方法是用来发送SOAP消息的。

****向多个端点发送消息****

管理对象是封装了针对提供者的配置和命名信息的对象。如果使用管理对象定义端点，需要注意的是：可以把该管理对象绑定到多个URL——每个URL都能够处理到达的SOAP消息。

以下代码示例把lookup名称为myEndpoint的端点绑定到两个URL：http://www.myServlet1/

和 http://www.myServlet2/。这个语法允许你使用SOAP连接发布SOAP消息到多个端点。

imqobjmgr add   
   -t e   
   -l "cn=myEndpoint"   
   -o "imqSOAPEndpointList=http://www.myServlet1/   
      http://www.myServlet2/"

****连接****

想要利用SAAJ或JAXM发送SOAP消息，必须分别获得一个SOAPConnection 或ProviderConnection。你还可以使用消息队列传输消息；更多信息请参见*Sun ONE消息队列开发人员指南。*

SOAP 连接

SOAPConnection允许你直接向远程伙伴发送消息。通过调用静态方法SOAPConnectionFactory.newInstance()，可以容易地得到SOAPConnection对象。这种类型的连接既不保证可靠性也不保证安全性。

****提供者连接****

从ProviderConnectionFactory 获得的ProviderConnection可以创建到特定消息提供者的连接。当使用提供者发送SOAP消息的时候，消息被传递给提供者，然后提供者负责传递消息到它的最终目的地，这样保证了可靠并且安全的消息传输。

****创建SOAP客户端****

创建SOAP客户端之前，确定你已经设置了客户端环境。关于设置客户端环境的更多信息，请参见["配置客户端环境 "](https://docs.huihoo.com/one_and_net/app7/10.1.htm" \l "14)。

如果创建一个点对点客户端，你必须导入SAAJ API的javax.xml.soap程序包。如果创建一个pub/sub客户端，需要导入import JAXM API的javax.xml.messaging程序包。除此之外，你还必须导入以下程序包：

import java.net.\*;  
import java.io.\*;  
import java.util.\*;

import javax.servlet.http.\*;  
import javax.servlet.\*;

import javax.activation.\*;  
import javax.naming.\*;

创建SOAP客户端并访问消息包括以下步骤：

* [获得连接](https://blog.csdn.net/u014252478/article/details/86700996" \l "111" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [创建消息](https://blog.csdn.net/u014252478/article/details/86700996" \l "222" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [向消息头添加内容](https://blog.csdn.net/u014252478/article/details/86700996" \l "qqq" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [向消息添加内容](https://blog.csdn.net/u014252478/article/details/86700996" \l "www" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [向消息添加一个附件](https://blog.csdn.net/u014252478/article/details/86700996" \l "eee" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [发送消息](https://blog.csdn.net/u014252478/article/details/86700996" \l "999" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [从响应消息中检索内容](https://blog.csdn.net/u014252478/article/details/86700996" \l "789" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [访问消息的附件部分](https://blog.csdn.net/u014252478/article/details/86700996" \l "369" \t "https://blog.csdn.net/u014252478/article/details/_self)

****获得连接****

单独客户端

不使用消息提供者的客户端使用SOAPConnection对象创建连接。使用SoapConnection对象发送的消息直接从发送者到发送者指定的URL。

必须获得一个SOAPConnectionFactory对象来创建连接。SAAJ API通过提供SOAPConnectionFactory类的一个默认实现简化了这一过程。以下代码演示了如何获得该实现的一个实例：

SOAPConnectionFactory scf = SOAPConnectionFactory. newInstance();

现在，你可以使用scFactory创建SOAPConnection对象

SOAPConnection con = scf.createConnection();

****创建消息****

使用MessageFactory对象来创建消息。如果想要创建一个单独客户端，也就是既不使用消息提供者也不在容器中运行的客户端，你可以使用SAAJ API提供的MessageFactory类的实现。以下代码演示了如何获得默认消息工厂的一个实例，然后用它创建消息：

MessageFactory mf = MessageFactory.newInstance();

SOAPMessage msg = mf.createMessage();

消息创建过程会完成SOAPPart的创建，它是SOAP 1.1规范中每个消息的必要部分。

SOAPPart sp = msg.getSOAPPart();

****注意：****

通常，使用默认的消息工厂创建消息。但是，你也可以编写自己的消息工厂实现，并且把它插入到系统属性中，如下所示：

* 通过扩展javax.xml.soap.MessageFactory实现消息工厂类。 javax.xml.soap是在SAAJ API中定义的程序包。
* 通过设定系统属性javax.xml.soap.MessageFactory为消息工厂实现类mypackage.MySOAPMessageFactoryImpl的完全路径名，指定希望被实例化的消息工厂类。

关于SOAP消息结构和组成部分的信息，请参见["SOAP消息的组成部分"](https://docs.huihoo.com/one_and_net/app7/12.1.htm" \l "08)。

****向Header添加内容****

创建一个SOAPHeaderElement对象，从而可以添加内容到header中。以下代码演示了如何使用SOAPEnvelope对象创建一个SOAPHeaderElement。

SOAPHeader hdr = envelope.getHeader();

Name headerName = envelope.createName("Purchase Order","PO", "http://www.sonata.com/order");

SOAPHeaderElement headerElement = hdr.addHeaderElement(headerName);

HeaderElement通过Name对象headerName进行标识。AddHeaderElement方法是用来添加或者创建header元素的。

为了添加内容到headerElement，请象以下代码示例那样使用addTextNode方法。

headerElement.addTextNode("order");

SOAPHeader对象包含一个内容为"order"的SOAPHeaderElement对象。

****向消息添加内容****

你可以向SOAPPart对象，或者一或多个AttachmentPart对象添加内容，或者可以向消息的这两个部分都添加内容。

为了向消息体添加内容，请创建一个SOAPBodyElement对象并添加一个使用SOAPElement.addTextNode方法建立的XML元素。下面的代码演示了如何添加内容到消息：

SOAPEnvelope envelope = sp.getSOAPEnvelope();

SOAPBody bdy = envelope.getSOAPBody();

SOAPBodyElement gltp = bdy.addBodyElement(envelope.createName("GetLastTradePrice", "ztrade", "http://wombat.ztrade.com"));

gltp.addChildElement(envelope.createName("symbol","ztrade", "http://wombat.ztrade.com")).addTextNode("SUNW");;

代码的前三行访问了SOAPBody对象body，目的是创建一个新SOAPBodyElement对象并把它添加到body上。CreateName方法需要Name对象作为参数，用来指定被添加的SOAPBodyElement。最后一行添加传递给addTextNode方法的XML字符串。

****向消息添加一个附件****

向消息添加附件的过程对于使用或不使用消息提供者的客户端来说都是一样的。使用AttachmentPart对象向消息中添加附件部分。

使用SOAPMessage对象创建AttachmentPart对象。SOAPMessage类含有三个创建附件的方法。第一个方法允许你创建一个没有内容的AttachmentPart。就是说，以后再使用AttachmentPart的方法setContent来添加内容到附件。

URL url = new URL(data);

AttachmentPart ap = msg.createAttachmentPart(new DataHandler(url));

SetContent有两个参数，对应内容的Java对象以及标识内容类型的String对象。内容是消息的SOAPBody部分，消息有一个值为"text/xml"的内容-类型（Content-Type）header，这是因为内容只能是XML格式。在AttachmentPart中，内容的类型只能被指定为该对象能够容纳的任意类型。

每个AttachmentPart都有一个或多个与之相关的header。setContent方法中用到的类型就是Content-Type这个header类型。这是唯一必须具备的header。你还可以设置其他可选的header，例如Content-Id和Content-Location。为了方便，JAXM和SAAJ API为Content-type、 Content-Id、和Content-Location这几个header提供了get和set方法。这些header在消息拥有多个附件的时候，可以帮助对附件的访问。

以下代码演示了如何使用setContent方法：

String stringContent = "Update address for Sunny Skies " + "Inc., to 10 Upbeat Street, Pleasant Grove, CA 95439";

ap.setContent(stringContent, "text/html");  
ap.setContentId("update\_address");  
msg.addAttachmentPart(ap);

如果你还想附加一个jpeg图像，setContent方法的第二个参数必须被设为"image/jpeg"。下面的代码演示了如何使用setContent方法附加图片：

AttachmentPart ap2 = msg.createAttachmentPart();

byte[] jpegData = . . .;

ByteArrayInputStream stream = new ByteArrayInputStream(jpegData);

ap2.setContent(stream, "image/jpeg");

msg.addAttachmentPart(ap2);

AttachmentPart的另外两个方法允许你创建带内容的AttachmentPart对象。其中一个与AttachmentPart.setContent方法很象。它需要一个包含内容的Java对象和一个指定内容类型的String作为参数。对象可以是String、stream、 javax.xml.transform.Source对象、或  javax.activation.DataHandler对象。

另一个创建带内容的AttachmentPart对象的方法需要一个DataHandler对象作为输入参数，它是JavaBeans激活框架(JAF)的一部分

以下代码演示了如何在内容中使用DataHandler。首先为想要作为内容添加的文件创建一个java.net.URL对象。创建DataHandler对象，即javax.activation.DataHandler 对象dh，使用URL对象对其进行初始化，然后把dh传递给createAttachmentPart方法。

URL url = new URL("http://greatproducts.com/gizmos/img.jpg");  
DataHandler dh = new DataHandler(url);   
AttachmentPart ap = msg.createAttachmentPart(dh);  
ap.setContentId("gyro\_image")  
msg.addAttachmentPart(ap);

****发送消息****

单独客户端

为了发送消息，单独客户端使用SOAPConnection的call方法。该方法需要两个输入参数，要发送的消息和消息的目的地，目的地是一个包含接收者URL的Endpoint对象。

利用SoapConnection的情况下，使用javax.xml.soap.SOAPConnection.call()发送消息。

例如：

URL urlEndpoint = new URL(to);

SOAPMessage reply = con.call(msg, urlEndpoint);

****从响应消息中检索内容****

客户端使用onMessage方法检索消息内容。客户端通过消息得到envelope，再通过envelope得到body，从而访问SOAPBody对象。访问SOAPBody对象是因为内容存放在该元素中。为了检索由Node.addTextNode方法添加的内容，请调用Node.getValue方法。GetValue返回调用元素的直接子元素的值。为了访问bodyElement，需要在body上调用getChildElement方法。以下代码演示了如何从响应消息中检索内容。

public SOAPMessage onMessage(SOAPMessage message)

{

SOAPEnvelop env = msg.getSOAPPart().getEnvelope();

env getBody()  
   .addChildElement(env.createName("Response"))  
   .addTextNode("This is a Response");

return msg;

}

想要从含有附件的消息中检索内容，你需要访问附件。没有参数的情况下，SOAPMessage.getAttachments方法返回指向所有AttachmentPart对象的java.util.Iterator对象。以下代码打印出了SOAPMessage对象消息中每一个AttachmentPart对象的内容。

java.util.Iterator it = message.getAttachments();

while (it.hasNext()) {

AttachmentPart attachment = (AttachmentPart)it.next();  
Object content = attachment.getContent();  
String id = attachment.getContentId();  
System.out.print("Attachment " + id + " contains: " + content);  
System.out.println("");

}

****访问消息的附件部分****

当收到带有附件的消息或是希望改变消息附件的时候，都需要访问消息的附件。没有附加信息的SOAPMesssage.getAttachments方法返回指向消息中所有AttachmentPart对象的java.util.Iterator对象。以下代码演示了如何访问附件从而获得SOAPMessage对象消息中每个AttachmentPart对象的内容。

java.util.Iterator it = msg.getAttachments();

while (it.hasNext()) {

AttachmentPart ap = it.next();   
Object content = ap.getContent();  
String id = ap.getContentId();   
System.out.print("Attachment " + id + " contains: " + content);  
System.out.println("");

}

****组装并部署SOAP客户端****

利用JAXM API和SAAJ API创建的应用程序被组装为web应用（WAR）或者基于J2EE平台的应用（EAR）。关于组装和部署web应用，请参见*Sun ONE Application Server开发人员web应用指南。*

## **SOAP 服务**

本节讲述如何编写SOAP服务以及如何在SOAP消息中处理异常和错误。本节讲述以下主题：

* [创建SOAP服务](https://blog.csdn.net/u014252478/article/details/86700996" \l "357" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [异常和错误处理](https://blog.csdn.net/u014252478/article/details/86700996" \l "159" \t "https://blog.csdn.net/u014252478/article/details/_self)

### ****创建SOAP服务****

SOAP服务是SOAP消息的最终接收者，并且被实现为servlet。要么创建你自己的servlet，要么扩展javax.xml.messaging包中的JAXMServlet类。本节讲述了基于JAXMServlet类创建SOAP服务的过程。

为了创建SOAP服务，你的servlet必须实现ReqRespListener或者OneWayListener接口。ReqRespListener需要返回一个回应。

public class MyServlet extends JAXMServlet implements ReqRespListener{

...

...

}

public SOAPMessage onMessage(SOAP Message msg)

使用任意借口，实现一个叫做onMessage(SOAPMsg)的方法。

以下代码是使用JAXMServlet 的SOAP消费者的完整清单：

public class MyServlet extends JAXMServlet implements ReqRespListener {

   public SOAPMessage onMessage(SOAP Message msg) {

   //Process message here

   }  
}

JAXMServlet在使用HTTP POST方法接受消息之后会调用onMessage。你就省去了实现自己的doPost()方法把收到的消息转换成为SOAP消息的麻烦。

OnMessage方法需要分解由servlet传入的SOAP消息并处理它的内容。处理消息包括访问SOAP消息的组成部分。如果在处理消息过程中出现故障，服务需要创建SOAP fault对象并把它发回到客户端。关于处理错误的更多信息，请参见["错误处理"](https://docs.huihoo.com/one_and_net/app7/12.1.htm" \l "189)。

以下代码演示了SOAP消息的处理：

{http://xml.coverpages.org/dom.html

SOAPEnvelope env = reply.getSOAPPart().getEnvelope();

SOAPBody sb = env.getBody();

// create Name object for XElement that we are searching for Name ElName = env.createName("XElement");

//Get child elements with the name XElement   
Iterator it = sb.getChildElements( ElName );

//Get the first matched child element.   
//We know there is only one.   
SOAPBodyElement sbe = (SOAPBodyElement) it.next();

//Get the value for XElement   
MyValue = sbe.getValue(); }

### ****异常和错误处理****

On the client's side, JAXM and SAAJ uses a SOAP exception to handle errors that occur during the generation of the SOAP request or unmarshalling of the response. This section describes the following topics:

在客户端，JAXM和SAAJ使用SOAP异常来处理SOAP 请求生成过程中，或者响应反编组过程中发生的错误。本节讲述以下主题：

* [错误处理](https://blog.csdn.net/u014252478/article/details/86700996" \l "189" \t "https://blog.csdn.net/u014252478/article/details/_self)
* [定义SOAP错误](https://docs.huihoo.com/one_and_net/app7/12.1.htm" \l "333)

****错误处理****

服务器端代码必须使用SOAPFault对象处理在反编组请求、处理消息、或是编组响应过程中发生的错误。SOAPFault接口扩展了SOAPBodyElement接口。

在服务器端，SOAP消息拥有用来形成错误报告的特定元素和格式：SOAP消息的body可以包含一个SOAPFault元素来报告发生在请求处理过程中的错误。在服务器端创建并且从服务器发回到客户端的SOAP消息包含了SOAPFault对象，它报告了消息生成者没有预期到的任何行为。

SOAPFault元素定义了以下四个子元素

****faultcode****

标识错误的代码。软件使用该代码为鉴别错误提供一个算法机制。此元素是必需的。

****faultstring****

一个描述错误代码所标识的错误的字符串。该元素提供了为人们无法理解的错误提供了一个解释。此元素是必需的。

****faultactor****

指定错误来源的URI：谁引起了错误。如果消息没有通过任何中间站直接发送给它的最终目的地，该元素不是必需的。如果在中间站发生了错误，这个错误必须包含一个faultactor元素。

****detail****

该元素持有与body元素相关的特定信息。如果body元素的内容不能够被成功处理，它就必须被给出。因此，如果这个元素丢失了，客户端应该推断出body元素已经被处理了。除了畸形负载，这个元素对任何错误都不是必须的，你可以在其他情况下用它来为客户端提供附加信息。

****预定义的错误代码****

SOAP规范列出了以下四个预定义的faultcode值：

VersionMismatch

处理方发现SOAP envelope元素有一个不合法的命名空间；也就是说，SOAPEnvelope元素的命名空间不是http://schemas.xmlsoap.org/soap/envelope/.

MustUnderstand

SOAPHeader元素的一个直接子元素要么无法理解，要么接收者不能正确处理。这个元素的MustUnderstand属性设被为1(true)。

Client

消息格式不正确或者没包含合适的信息。例如，消息没有包含正确的认证或付款信息。客户端应该翻译这个代码从而指定这个消息在重新发送之间必须被修改。如果这是一个返回代码，SOAPFault对象应该包括一个detailEntry对象，它为格式错误的消息提供附加信息。

Server

该消息由于一些与它的内容没有联系的原因而不能被处理。例如，消息的一个处理者无法与另一个上游的、没有相应的消息处理者进行通信。或者，服务器需要访问的数据库崩溃了。客户端应该解释这个错误，以指出事务可以在以后的某个及时时间点成功。

****定义SOAP错误****

可以使用SOAPFault对象的方法为faultcode、 faultstring、和faultctor指定值。以下代码演示了SOAPFault对象的创建和faultcode、 faultstring、faultctor属性的设定。

SOAPFault fault;

reply = factory.createMessage();

envp = reply.getSOAPPart().getEnvelope(true);

someBody = envp.getBody();

fault = someBody.addFault():

fault.setFaultCode("Server");

fault.setFaultString("Some Server Error");

fault.setFaultActor("http://xxx.me.com/list/endpoint.esp/");

reply.saveChanges();

一旦发生服务器错误，服务器能够在对收到的SOAP消息做出的回应中，返回这个对象。

以下代码演示了如何定义一个detail和detail entry对象。注意，必须为detail entry对象创建一个名称。

SOAPFault fault = someBody.addFault();

fault.setFaultCode("Server");

fault.setFaultActor("http://foo.com/uri");

fault.setFaultString ("Unkown error");

DetailEntry entry = detail.addEntry(envelope.createName("125detail", "m", "Someuri");

entry.addTextNode("the message cannot contain the string //");

reply.saveChanges();

****组装并部署SOAP服务****

使用JAXM API和SAAJ API创建的应用被组装为web应用（WAR）或者基于J2EE平台的应用（EAR）。关于组装和部署web应用，请参见*Sun ONE Application Server开发人员web应用指南。*

****客户端和服务示例****

客户端和服务应用程序示例被捆绑在Sun ONE Application Server中。这些例子示范如何创建发送和接收XML消息的服务和客户端。你可以在下面的位置找到这些例子：