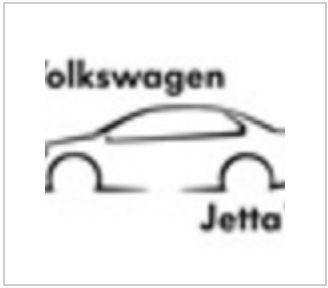
hideto

永久域名 http://hideto.javaeye.com



hideto

1231 1231 345 345 565431 565431 351 351

2009-03-13 通过*Firefox*插件

>>更多闲聊

浏览: 1103713 次

性别:

来自: 北京

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

ynstudio

悠游小草

在UDDI注册中心里使用WSDL | 学习D语言之helloworld

2007-03-01

Axis2快速上手指南

关键字: soa axis2 webservice

原文链接:http://ws.apache.org/axis2/1_1_1/quickstartguide.html
本指南的目的是让你尽可能快的创建使用Axis2的服务和客户端，我们将使用一个简单的StockQuote服务并显示给你一些创建和部署它的不同的方式，以及快速的看看Axis2自带的一些工具，然后我们将看看创建访问这些服务的客户端。

内容

Java代码

```
1.  介绍
2.  做好准备
3.  Axis2服务
4.  创建服务
5.      部署POJOs
6.      使用AXIOM构建服务
7.      使用ADB生成服务
8.      使用XMLBeans生成服务
9.      使用JiBX生成服务
10. 生成客户端
11.     使用AXIOM创建一个客户端
12.     使用ADB生成一个客户端
13.     使用XMLBeans生成一个客户端
14.     使用JiBX生成一个客户端
15. 总结
16. 进一步学习
```

快速安装笔记:
文档的代码可以在解压的标准二进制发布[/url]找到，更明确的位于其中的Axis2_HOME/samples/目录－quickstart，quickstartdb，quickstartaxiom，quickstartjibx和quickstartxmlbeans，如果你继续下去它将帮你掌握它。它包含一个Ant构建文件(build.xml)贯穿所有的例子我们将提到它来使得编译更容易。

kajima lnaigg

博客分类

我的相册

screenshot

共 1 张

我的留言簿 >>更多留言

- 请问classloder可以用于手

介绍

让我们以服务本身开始。我们将使它简单，所以你可以看到当我们构建并部署服务时会发生什么，一个StockQuoteService例子看起来像这个，所以让我们使用下面的(参看Code Listing 1)。

Code Listing 1: StockQuoteService类

Java代码

```java
package samples.quickstart.service.pojo;

import java.util.HashMap;

public class StockQuoteService {
    private HashMap map = new HashMap();

    public double getPrice(String symbol) {
        Double price = (Double) map.get(symbol);
        if(price != null){
            return price.doubleValue();
        }
        return 42.00;
    }

    public void update(String symbol, double price) {
        map.put(symbol, new Double(price));
    }
}
```

它将为一个具有两个可能的调用的简单服务，其中一个是一个in/out消息，另一个则为一个只能in的服务，最终我们将打包服务并用四种不同的方式部署部署它。

首先，让我们看看这个简单的Java类怎样响应一个服务。

做好准备

在我们使用Axis2构建任何东西之前，我们将需要关注一些家务事。现在你将需要准备好你使用Axis2的环境，幸运的是，它只包括一些简单的步骤:

1，下载并安装Java(版本至少为JDK1.4)

2，下载Axis2并解压到一个目标目录

3，复制axis2.war文件到你的servlet引擎的webapps目录

4，设置AXIS2_HOME环境变量来指出目标目录，注意Axis2生成的所有脚本和构建文件依赖于这个值，所以不要遗漏了这个步骤。

大多数情况下，我们的服务也将需要一个WSDL文件，Axis2的Java2WSDL可以用来生成一个WSDL。执行以下步骤来从一个Java类生成一

个WSDL文件:

1，创建并编译Java类

机j2me源代码的加密吗？ 我的联系办法是chr ...

-- by chr1118888

- 想请教一个问题 Mule有没有一种方式单独控制每一个配置文件，我的意思是单独开启/ ...

  -- by mtvwang

- 抱抱

  -- by Hooopo

其他分类

- 我的收藏 (6)
- 我的书籍 (2)
- 我的新闻 (1)
- 我的论坛主题贴 (634)
- 我的所有论坛贴 (364)
- 我的精华良好贴 (8)

最近加入圈子

- 火星常驻JE办事处
- ror-party
- flex
- D语言
- Ubuntu For Fun

存档

- 2010-02 (1)
- 2009-08 (3)
- 2009-07 (8)
- 更多存档...

最新评论

2，使用该命令生成WSDL:

%AXIS2_HOME%/bin/java2wsdl -cp . -cn samples.quickstart.service.pojo.StockQuoteService -of StockQuoteService.wsdl

移动你生成WSDL文件，你可以做你需要的任何更改。例如，你可能添加自定义的过错或者改变生成的元素名。例如，该StockQuoteService.wsdl位于%AXIS2_HOME%/samples/quickstartadb/resources/META-INF文件夹，我们将在本指南的其他部分使用它，代替生成过程创建的一般参数。

Axis2服务

在我们构建任何东西之前，理解最终产品看起来像什么是有用的。Axis2的服务器端可以被部署在任何Servlet引擎上，并且有如下的Code Listing 2显示的结构。

Code Listing 2: axis2.war的目录结构

Java代码

```
axis2-web
META-INF
WEB-INF
    classes
    conf
        axis2.xml
    lib
        activation.jar
        ...
        xmlSchema.jar
    modules
        modules.list
        addressing.mar
        ...
        soapmonitor.mar
    services
        services.list
        aservice.aar
        ...
        version.aar
    web.xml
```

从最上面开始，axis2-web是组成Axis2管理程序的一些JSPs，你可以通过它来执行任何需要的动作，例如添加服务，使用模块和禁止模块。WEB-INF包含了运行部署到服务目录的任何服务的实际上的java类和其他支持文件。

这里主要的文件是axis2.xml，它控制程序怎样与接收的消息打交道，决定Axis2是否需要使用模块目录里定义的任何模块。

这里你可以看到，服务可以被部署为*.aar文件，但是它们的内容必须以一个特殊的方式安排。例如，服务的结构将为如下:

Java代码

```
- StockQuoteService
   - META-INF
```

- PHP、CakePHP哪凉快哪呆 ...

  “这位兄弟心态很好呵呵ruby不用你说我在工作中用了2年了我倒是建议只用php的那 ...

  -- by ycmhn
- 每天一剂Rails良药之Live ...

  rails recipes -- live preview

  -- by orcl_zhang
- attachment_fu的一个bug

  貌似没有效果!

  -- by tangyuanjian
- Erlang里的make

  学习了。 Erlang自带的Emakefile与Makefile区别挺大的，Joe ...

  -- by xuexixht
- PHP、CakePHP哪凉快哪呆 ...

  肤浅

  -- by applestar

评论排行榜

- G1入门
- HPM Note1,Book Organization
- 轻量级开源BI需求设计
- HPM Note3, Benchmarking and Profiling
- HPM Note2, MySQL Architecture


```
      - services.xml
    - samples
      - quickstart
        - service
          - pojo
            - StockQuoteService.class
```

这不是太复杂，服务的名字为StockQuoteService，它在services.xml中指定，并且根据包名把任何Java类放到合适的位置。META-INF目录包含关于Axis2需要来正确执行服务的额外信息。services.xml文件定义了服务本身并把Java类链接到它(参看Code Listing 3)。

Code Listing 3: 服务定义文件

Java代码

```
<service name="StockQuoteService" scope="application">
    <description>
        Stock Quote Sample Service
    </description>
    <messageReceivers>
        <messageReceiver
            mep="http://www.w3.org/2004/08/wsdl/in-only"
    class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
        <messageReceiver
            mep="http://www.w3.org/2004/08/wsdl/in-out"
    class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>
    <parameter name="ServiceClass">
        samples.quickstart.service.pojo.StockQuoteService
    </parameter>
</service>
```

这里你看到服务本身被定义了，以及不同的消息交换模式相关的messageReceiver类型。
META-INF目录也是你打算包含在该程序中的自定义WSDL文件的位置。
你可以通过简单的采用该文件目录结构并复制到你的servlet引擎的webapps目录来部署一个服务，这是著名的"爆发"形式，但是你也可以压缩你的文档到一个*.aar文件，类似于一个*.jar文件，并直接将*.aar文件放到servlet引擎的webapps目录。
既然你理解了我们在尝试完成什么，我们几乎准备好开始构建了。
首先，下载并解压合适版本的Axis2标准二进制发布。确认你设置了AXIS2_HOME变量的值来匹配你解压该版本内容的位置。
让我们看看一些创建客户端和服务的不同方式。

创建服务
在这个部分，我们将看看根据StockQuoteService类创建服务的五种方式:部署Plain Old Java Objects(POJO)，使用AXIOM的OMElement
构建服务，使用Axis2 Databinding Framework(ADB)生成服务，使用XMLBeans生成服务和使用JiBX生成服务。

[什么是RSS?]

部署**POJOs**

使用POJOs(Plain Old Java Objects)来部署服务，执行下面的步骤。

注意包含在<AXIS2_HOME>/samples/quickstart的目录结构(services.xml文件来自于该指南的第一个部分):

Java代码

```
- quickstart
   - README.txt
   - build.xml
   - resources
     - META-INF
       - services.xml
   - src
     - samples
       - quickstart
         - service
           - pojo
             - StockQuoteService.java
```

注意你可以通过在quickstart目录键入**ant generate.wsdl**来生成WSDL。

尽管如此，创建StockQuoteService.wsdl是可选的。它可以是直接从Java类生成的版本，或者该文件的一个自定义版本，并且services.xml是本文档前面提到的同一文件。

现在通过在quickstart目录键入**ant generate.service**来构建工程，该目录创建了以下目录结构:

Java代码

```
- quickstart/build/classes
   - META-INF
     - services.xml
   - samples
     - quickstart
       - service
         - pojo
           - StockQuoteService.class
```

如果你想以爆发形式部署服务，重命名classes目录为StockQuoteService，并复制它到你的servlet引擎的webapps/axis2/WEB-INF/services目录。否则，复制build/StockQuoteService.aar文件到你的servlet引擎的webapps/axis2/WEB-INF/services目录。然后通过视察以下服务列表来确认服务被正确部署:

http://localhost:8080/axis2/services/listServices

你也可以检验以下WSDL:

http://localhost:8080/axis2/services/StockQuoteService?wsdl

以及以下结构:

http://localhost:8080/axis2/services/StockQuoteService?xsd

一旦这些urls工作，让我们快速测试一下服务。让你的浏览器访问以下URL试试:

http://localhost:8080/axis2/rest/StockQuoteService/getPrice?symbol=IBM

你将得到如下应答:

Java代码

```
<ns:getPriceResponse xmlns:ns="http://pojo.service.quickstart.samples/xsd"><ns:return>42</ns:return></ns:getPriceResponse>
```

如果你像这样调用update方法:
http://localhost:8080/axis2/rest/StockQuoteService/update?symbol=IBM&price=100
然后调用第一个getPrice url。你可以看到price被更新了。

使用**AXIOM**构建服务
为了使用AXIOM"从零开始"构建一个服务，执行以下步骤。
注意包含在/samples/quickstartaxiom的目录结构:

Java代码

```
- quickstartaxiom
   - README.txt
   - build.xml
   - resources
     - META-INF
       - services.xml
       - StockQuoteService.wsdl
   - src
     - samples
       - quickstart
         - service
           - axiom
             - StockQuoteService.java
         - clients
           - AXIOMClient.java
```

由于AXIOM有一点不同，你将需要一个与POJO所用不同的services.xml文件，在Code Listing 4中显示了它的定义。
Code Listing 4: 服务定义文件

Java代码

```
<service name="StockQuoteService" scope="application">
    <description>
        Stock Quote Service
    </description>
    <operation name="getPrice">
        <messageReceiver class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>
    </operation>
    <operation name="update">
```

```
        <messageReceiver class="org.apache.axis2.receivers.RawXMLINOnlyMessageReceiver"/>
    </operation>
    <parameter name="ServiceClass">samples.quickstart.service.axiom.StockQuoteService</parameter>

</service>
```

注意，几乎是一样的，除了操作是显示地定义在service.xml文件中，而且MessageReceiver现在是RawXML。
现在，上面提到的简单的使用Axis2库的类的StockQuoteService.java类，在Code Listing 5中定义。
Code Listing 5:使用AXIOM的StockQuoteService类

Java代码

```java
package samples.quickstart.service.axiom;

import javax.xml.stream.XMLStreamException;
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 java.util.HashMap;
public class StockQuoteService {
    private HashMap map = new HashMap();

    public OMElement getPrice(OMElement element) throws XMLStreamException {
        element.build();
        element.detach();

        OMElement symbolElement = element.getFirstElement();
        String symbol = symbolElement.getText();

        String returnText = "42";
        Double price = (Double) map.get(symbol);
        if(price != null){
            returnText  = "" + price.doubleValue();
        }
        OMFactory fac = OMAbstractFactory.getOMFactory();
        OMNamespace omNs =
            fac.createOMNamespace("http://axiom.service.quickstart.samples/xsd", "tns");
        OMElement method = fac.createOMElement("getPriceResponse", omNs);
        OMElement value = fac.createOMElement("price", omNs);
        value.addChild(fac.createOMText(value, returnText));
        method.addChild(value);
        return method;
```

```
33.     }
34.
35.     public void update(OMElement element) throws XMLStreamException {
36.         element.build();
37.         element.detach();
38.
39.         OMElement symbolElement = element.getFirstElement();
40.         String symbol = symbolElement.getText();
41.
42.         OMElement priceElement = (OMElement)symbolElement.getNextOMSibling();
43.         String price = priceElement.getText();
44.
45.         map.put(symbol, new Double(price));
46.     }
47. }
```

Axis2使用AXIOM，或者AXIs Object Model，一个基于StAX API(Streaming API for XML)的DOM(Document Object Model)类似的结构作为服务的方法必须使用OMElement作为它们的参数，OMElement表示一个XML元素，在这里它则为进来的SOAP消息的有效载荷。这里，
你提取有效载荷元素的第一个孩子，添加文本给它，并使用它作为返回的OMElement的内容。除非这是一个"只有in"的服务，这些方法必须返回一个OMElement，因为它成为返回的SOAP消息的有效载荷。
现在通过在Axis2_HOME/samples/quickstartaxiom目录键入**ant generate.service**来构建工程。
把StockQuoteService.aar文件放在servlet引擎的webapps/axis2/WEB-INF/services目录，通过视察下面的服务列表来确认服务正确部署:
http://localhost:8080/axis2/services/listServices
你也可以检验以下WSDL:
http://localhost:8080/axis2/services/StockQuoteService?wsdl
以及以下结构:
http://localhost:8080/axis2/services/StockQuoteService?xsd

使用**ADB**生成服务
执行以下步骤来使用Axis2 Databinding Framework(ADB)生成和部署服务。
通过在Axis2_HOME/samples/quickstartadb目录键入以下内容来使用WSDL2Java工具生成骨架:
%AXIS2_HOME%/bin/WSDL2Java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.adb -d adb -s -ss -sd -ssi -o build/service
或者在Axis2_HOME/samples/quickstartadb目录简单的键入**ant generate.service**
选项-d adb指定了Axis Data Binding(ADB)。-s转换指定同步或者只模块化调用。-ss转换创建服务器端代码(骨架和相关文件)。-sd转换创建一个服务描述符(services.xml文件)。-ssi转换为服务骨架创建一个接口。服务文件现在应该定位于build/service。
如果你通过使用WSDL2Java直接生成代码，下一步你需要修改生成的骨架来实现服务(如果你使用"ant generate.service"，一个完全的骨架会自动复制并覆盖生成的那个)。
打开build/service/src/samples/quickstart/adb/service/StockQuoteServiceSkeleton.java文件并修改它来添加你的服务的功能性到生成的方法，下面的Code Listing 6显示了。

Code Listing 6:定义服务骨架文件

Java代码

```java
package samples.quickstart.service.adb;

import samples.quickstart.service.adb.xsd.GetPriceResponse;
import samples.quickstart.service.adb.xsd.Update;
import samples.quickstart.service.adb.xsd.GetPrice;

import java.util.HashMap;

public class StockQuoteServiceSkeleton {

    private static HashMap map;

    static{ map = new HashMap(); }

    public void update(Update param0) {
        map.put(param0.getSymbol(), new Double(param0.getPrice()));
    }

    public GetPriceResponse getPrice(GetPrice param1) {
        Double price = (Double) map.get(param1.getSymbol());
        double ret = 42;
        if(price != null){
            ret = price.doubleValue();
        }
        GetPriceResponse res =
                new GetPriceResponse();
        res.set_return(ret);
        return res;
    }
}
```

现在你可以通过在build/service目录键入以下命令构建工程:**ant jar.server**
如果一切进展顺利，你应该在你的窗口看到BUILD SUCCESSFUL消息，而且StockQuoteService.aar文件在build/service/build/lib目录中。复制该文件到servlet引擎的webapps/axis2/WEB-INF/services目录。
你可以通过视察下面的服务列表来确认服务正确部署:
http://localhost:8080/axis2/services/listServices
你也可以检验以下WSDL:
http://localhost:8080/axis2/services/StockQuoteService?wsdl
以及以下结构:
http://localhost:8080/axis2/services/StockQuoteService?xsd

使用**XMLBeans**生成服务
执行以下步骤来使用XMLBeans生成服务。
通过在Axis2_HOME/samples/quickstartxmlbeans目录键入以下命令来使用WSDL2Java工具生成骨架:
%AXIS2_HOME%/bin/WSDL2Java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.service.xmlbeans -d xmlbeans -s -ss -sd -ssi -o build/service
或者在Axis2_HOME/sampels/quickstartxmlbeans目录简单的键入**ant generate.service**
选项-d xmlbeans指定XMLBeans数据绑定。-s转换指定同步或者只是模块化调用。-ss转换创建服务器端代码(骨架和相关文件)。-sd转换创建一个服务描述符(services.xml文件)。-ssi转换创建一个服务骨架的接口。现在服务文件应该位于build/service。
如果你通过使用WSDL2Java直接生成代码，下一步你需要修改生成的骨架来实现服务(如果你使用"ant generate.service"，一个完全的骨架将被自动复制并覆盖生成的那个)。
下一步打开build/service/src/samples/quickstart/service/xmlbeans/StockQuoteServiceSkeleton.java文件并修改它来添加你的服务的功能性到生成的方法(参看Code Listing 7)。
Code Listing 7:定义服务骨架

Java代码

```java
package samples.quickstart.service.xmlbeans;

import samples.quickstart.service.xmlbeans.xsd.GetPriceDocument;
import samples.quickstart.service.xmlbeans.xsd.GetPriceResponseDocument;
import samples.quickstart.service.xmlbeans.xsd.UpdateDocument;

import java.util.HashMap;

public class StockQuoteServiceSkeleton implements StockQuoteServiceSkeletonInterface {

    private static HashMap map;

    static{ map = new HashMap(); }

    public void update(UpdateDocument param0) {
    }

    public GetPriceResponseDocument getPrice(GetPriceDocument param1) {
    }
}
```

通过在build/service目录键入命令**ant jar.server**构建项目，该目录包含build.xml文件。
如果一切进展顺利，你应该在你的窗口看到BUILD SUCCESSFUL消息，而且StockQuoteService.aar文件位于新创建的build/service/build/lib目录。复制该文件到servlet引擎的webapps/axis2/WEB-INF/services目录。
你可以通过视察下面的服务列表来确认服务正确部署:
http://localhost:8080/axis2/services/listServices
你也可以检验以下WSDL:

http://localhost:8080/axis2/services/StockQuoteService?wsdl
以及以下结构:
http://localhost:8080/axis2/services/StockQuoteService?xsd

使用**JiBX**生成服务
执行以下步骤来使用JiBX数据绑定生成和部署服务。
通过在Axis2_HOME/samples/quickstartjibx目录的控制台键入以下内容来使用WSDL2Java工具生成骨架:
%AXIS2_HOME%/bin/wsdl2java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.service.jibx -d jibx -s -ss -sd -ssi -uw -o build/service
或者在Axis2_HOME/samples/quickstartjibx目录简单的键入"**ant generate.service**"
选项-d jibx指定了JiBX数据绑定。-s转换指定同步或者只是模块化调用。-ss转换创建服务器端代码(骨架和相关文件)。-sd转换创建一个服务描述符(services.xml文件)。-ssi转换创建服务骨架的接口。-uw转换解开传递给服务操作和从服务操作传递出去的参数，来创建一个更自然的编程接口。
在运行WSDL2Java后，服务文件应该位于build/service。如果你通过使用WSDL2Java直接生成代码，下一步你需要修改生成的骨架来实现服务(如果你使用"ant generate.service"则一个完全的骨架将自动被复制并覆盖生成的那个)。打开build/service/src/samples/quickstart/service/jibx/StockQuoteServiceSkeleton.java文件并修改它来添加你的服务的功能性到生成的方法，在Code Listing 8中显示了。
Code Listing 8:定义服务骨架文件

Java代码

```java
package samples.quickstart.service.jibx;

import java.util.HashMap;

public class StockQuoteServiceSkeleton implements StockQuoteServiceSkeletonInterface {
    private HashMap map = new HashMap();

    public void update(String symbol, Double price) {
        map.put(symbol, price);
    }

    public Double getPrice(String symbol) {
        Double ret = (Double) map.get(symbol);
        if (ret == null) {
            ret = new Double(42.0);
        }
        return ret;
    }
}
```

现在你可以通过在build/service目录键入命令**ant jar.server**构建工程
如果一切进展顺利，你应该在你的窗口看到BUILD SUCCESSFUL消息，并且StockQuoteService.aar文件位于build/service/build/lib目录。复制该文件到servlet引擎的webapps/axis2/WEB-INF/services目录。

你可以通过视察下面的服务列表来确认服务正确部署:
http://localhost:8080/axis2/services/listServices
你也可以检验以下WSDL:
http://localhost:8080/axis2/services/StockQuoteService?wsdl
以及以下结构:
http://localhost:8080/axis2/services/StockQuoteService?xsd
对于和Axis2使用JiBX的更多信息，参考[urlhttp://ws.apache.org/axis2/1_1_1/jibx/jibx-quotegen-integration.html]JiBX代码生成集成[/url]。你也可以检查JiBX Axis2 Wiki页面得到更多同Axis2使用
JiBX的更新信息。

创建客户端
在这个部分，我们将看看基于StockQuoteService类创建客户端的四种方式:构建基于AXIOM的客户端，使用Axis2 Databinding Framework(ADB)生成客户端，使用XMLBeans生成客户端，使用JiBX生成客户端。

使用**AXIOM**创建一个客户端
执行以下步骤来使用AXIOM构建一个客户端。
也注意在使用AXIOM创建服务部分显示的目录结构，为了完整性下面的重复了。

Java代码

```
- quickstartaxiom
   - README.txt
   - build.xml
   - resources
     - META-INF
       - services.xml
       - StockQuoteService.wsdl
   - src
     - samples
       - quickstart
         - service
           - axiom
             - StockQuoteService.java
         - clients
           - AXIOMClient.java
```

上面提到的AXIOMClient.java类的定义显示在下面的Code Listing 9。
Code Listing 9:使用AXIOM的AXIOMClient类

Java代码

```
package samples.quickstart.clients;

import org.apache.axiom.om.OMAbstractFactory;
import org.apache.axiom.om.OMElement;
```

```
5.  import org.apache.axiom.om.OMFactory;
6.  import org.apache.axiom.om.OMNamespace;
7.  import org.apache.axis2.Constants;
8.  import org.apache.axis2.addressing.EndpointReference;
9.  import org.apache.axis2.client.Options;
10. import org.apache.axis2.client.ServiceClient;
11.
12. public class AXIOMClient {
13.
14.     private static EndpointReference targetEPR =
15.         new EndpointReference("http://localhost:8080/axis2/services/StockQuoteService");
16.
17.     public static OMElement getPricePayload(String symbol) {
18.         OMFactory fac = OMAbstractFactory.getOMFactory();
19.         OMNamespace omNs = fac.createOMNamespace("http://axiom.service.quickstart.samples/xsd",
    "tns");
20.
21.         OMElement method = fac.createOMElement("getPrice", omNs);
22.         OMElement value = fac.createOMElement("symbol", omNs);
23.         value.addChild(fac.createOMText(value, symbol));
24.         method.addChild(value);
25.         return method;
26.     }
27.
28.     public static OMElement updatePayload(String symbol, double price) {
29.         OMFactory fac = OMAbstractFactory.getOMFactory();
30.         OMNamespace omNs = fac.createOMNamespace("http://axiom.service.quickstart.samples/xsd",
    "tns");
31.
32.         OMElement method = fac.createOMElement("update", omNs);
33.
34.         OMElement value1 = fac.createOMElement("symbol", omNs);
35.         value1.addChild(fac.createOMText(value1, symbol));
36.         method.addChild(value1);
37.
38.         OMElement value2 = fac.createOMElement("price", omNs);
39.         value2.addChild(fac.createOMText(value2,
40.                                          Double.toString(price)));
41.         method.addChild(value2);
42.         return method;
43.     }
44.
45.     public static void main(String[] args) {
46.         try {
```

```
47.             OMElement getPricePayload = getPricePayload("WSO");
48.             OMElement updatePayload = updatePayload("WSO", 123.42);
49.             Options options = new Options();
50.             options.setTo(targetEPR);
51.             options.setTransportInProtocol(Constants.TRANSPORT_HTTP);
52.
53.             ServiceClient sender = new ServiceClient();
54.             sender.setOptions(options);
55.
56.             sender.fireAndForget(updatePayload);
57.             System.err.println("done");
58.             OMElement result = sender.sendReceive(getPricePayload);
59.
60.             String response = result.getFirstElement().getText();
61.             System.err.println("Current price of WSO: " + response);
62.
63.         } catch (Exception e) {
64.             e.printStackTrace();
65.         }
66.     }
67.
68. }
```

Axis2使用AXIOM，或者AXIs Object Model，一个基于StAX API(Streaming API for XML)的DOM(Document Object Model)类似的结构这里你为服务的update和getPrice方法建立有效载荷。有效载荷的创建类似于你为AXIOM服务创建getPriceResponse有效载荷。然后你创建Options类，并创建用来与服务交流的ServiceClient。首先你调用update方法，一个什么也不返回的fireAndForget方法。最后你调用getPrice方法，并从服务得到当前价格并显示它。
现在你可以通过在Axis2_HOME/samples/quickstartaxiom目录键入**ant run.client**构建并运行AXIOM客户端。
你应该得到以下输出:

Java代码

```
1. done
2. Current price of WSO: 123.42
```

使用**ADB**生成一个客户端
执行以下步骤来使用Axis Data Binding(ADB)构建一个客户端。
通过在Axis2_HOME/samples/quickstartadb目录键入以下命令来生成客户端数据绑定:
%AXIS2_HOME%/bin/WSDL2Java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.clients -d adb -s -o build/client
或者在Axis2_HOME/samples/quickstartadb目录简单的键入**ant generate.client**。
下一步看看quickstartadb/src/samples/quickstart/clients/ADBClient.java，并看看它在Code Listing 10中是怎样定义的。

Code Listing 10:ADBClient类

Java代码

```java
package samples.quickstart.clients;

import samples.quickstart.service.adb.StockQuoteServiceStub;

public class ADBClient{
    public static void main(java.lang.String args[]){
        try{
            StockQuoteServiceStub stub =
                new StockQuoteServiceStub
                ("http://localhost:8080/axis2/services/StockQuoteService");

            getPrice(stub);
            update(stub);

        } catch(Exception e){
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }

    /* fire and forget */
    public static void update(StockQuoteServiceStub stub){
        try{
            StockQuoteServiceStub.Update req = new StockQuoteServiceStub.Update();
            req.setSymbol ("ABC");
            req.setPrice (42.35);

            stub.update(req);
            System.err.println("done");
        } catch(Exception e){
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }

    /* two way call/receive */
    public static void getPrice(StockQuoteServiceStub stub){
        try{
            StockQuoteServiceStub.GetPrice req = new StockQuoteServiceStub.GetPrice();

            req.setSymbol("ABC");
```

```
42.
43.             StockQuoteServiceStub.GetPriceResponse res =
44.                 stub.getPrice(req);
45.
46.             System.err.println(res.get_return());
47.         } catch(Exception e){
48.             e.printStackTrace();
49.             System.err.println("\n\n\n");
50.         }
51.     }
52.
53. }
```

该类使用你创建的Axis Data Bindings创建一个客户端存根。然后它在Web服务上调用getPrice和update操作。getPrice方法操作创建GetPrice有效载荷并设置symbol为ABC。然后它发送请求并显示当前价格。update方法创建一个Update有效载荷，设置symbol为ABC及
price为42.35。
现在通过在Axis2_HOME/samples/quickstartadb目录键入**ant run.client**来构建并运行客户端。
你应该得到以下输出:

Java代码

```
1. 42
2. done
```

使用**XMLBeans**生成一个客户端
执行以下步骤来使用XMLBeans数据绑定来构建一个客户端。
通过在xmlbeansClient目录键入以下命令来生成数据绑定:
%AXIS2_HOME%/bin/WSDL2Java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.service.xmlbeans -d
xmlbeans -s -o build/client
或者简单的在Axis2_HOME/samples/quickstartxmlbeans目录下键入**ant generate.client**。
注意这只会创建客户端存根代码而不会创建服务器端代码。
下一步看看quickstartxmlbeans/src/samples/quickstart/clients/XMLBEANSClient.java，并在Code Listing 11看看它怎样定义的
Code Listing 11:XMLBEANSClient类

Java代码

```
1. package samples.quickstart.clients;
2.
3. import samples.quickstart.service.xmlbeans.StockQuoteServiceStub;
4. import samples.quickstart.service.xmlbeans.xsd.GetPriceDocument;
5. import samples.quickstart.service.xmlbeans.xsd.GetPriceResponseDocument;
6. import samples.quickstart.service.xmlbeans.xsd.UpdateDocument;
```

```
7.
8.  public class XMLBEANSClient{
9.
10.     public static void main(java.lang.String args[]){
11.         try{
12.             StockQuoteServiceStub stub =
13.                 new StockQuoteServiceStub
14.                 ("http://localhost:8080/axis2/services/StockQuoteService");
15.
16.             getPrice(stub);
17.             update(stub);
18.
19.         } catch(Exception e){
20.             e.printStackTrace();
21.             System.err.println("\n\n\n");
22.         }
23.     }
24.
25.     /* fire and forget */
26.     public static void update(StockQuoteServiceStub stub){
27.         try{
28.             UpdateDocument reqDoc = UpdateDocument.Factory.newInstance();
29.             UpdateDocument.Update req = reqDoc.addNewUpdate();
30.             req.setSymbol ("ABC");
31.             req.setPrice (42.32);
32.
33.             stub.update(reqDoc);
34.             System.err.println("done");
35.         } catch(Exception e){
36.             e.printStackTrace();
37.             System.err.println("\n\n\n");
38.         }
39.     }
40.
41.     /* two way call/receive */
42.     public static void getPrice(StockQuoteServiceStub stub){
43.         try{
44.             GetPriceDocument reqDoc = GetPriceDocument.Factory.newInstance();
45.             GetPriceDocument.GetPrice req = reqDoc.addNewGetPrice();
46.             req.setSymbol("ABC");
47.
48.             GetPriceResponseDocument res =
49.                 stub.getPrice(reqDoc);
50.
```

```
            System.err.println(res.getGetPriceResponse().getReturn());
        } catch(Exception e){
            e.printStackTrace();
            System.err.println("\n\n\n");
        }
    }
}
```

该类使用你创建的XMLBeans数据绑定创建一个客户端存根，然后它在Web服务上调用getPrice和update操作。getPrice方法操作创建GetPriceDocument，它内部的GetPrice类，并设置symbol为ABC。然后它发送请求并得到GetPriceResponseDocument并显示当前price
update方法创建一个UpdateDocument和Update并设置symbol为ABC及price为42.32，当完成时显示done。
现在通过在Axis2_HOME/samples/quickstartxmlbeans目录键入**ant run.client**构建并运行工程。
你应该得到下列输出:

Java代码

```
42
done
```

使用**JiBX**生成一个客户端
执行以下步骤来使用JiBX构建一个客户端。
通过在Axis2_HOME/samples/quickstartjibx目录的控制台键入以下命令来生成客户端存根:

Java代码

```
%AXIS2_HOME%/bin/wsdl2java -uri resources/META-INF/StockQuoteService.wsdl -p samples.quickstart.clients -d jibx -s
-uw -o build/client
```

或者简单的键入"**ant generate.client**"。
下一步看看quickstartjibx/src/samples/quickstart/clients/JiBXClient.java，在Code Listing 12显示了。
Code Listing 12:JiBXClient类

Java代码

```
package samples.quickstart.clients;

import samples.quickstart.service.jibx.StockQuoteServiceStub;

public class JiBXClient{
    public static void main(java.lang.String args[]){
        try{
            StockQuoteServiceStub stub =
                new StockQuoteServiceStub
```

```
                    ("http://localhost:8080/axis2/services/StockQuoteService");

                getPrice(stub);
                update(stub);

            } catch(Exception e){
                e.printStackTrace();
                System.err.println("\n\n\n");
            }
        }

        /* fire and forget */
        public static void update(StockQuoteServiceStub stub){
            try{
                stub.update("ABC", new Double(42.35));
                System.err.println("done");
            } catch(Exception e){
                e.printStackTrace();
                System.err.println("\n\n\n");
            }
        }

        /* two way call/receive */
        public static void getPrice(StockQuoteServiceStub stub){
            try{
                System.err.println(stub.getPrice("ABC"));
            } catch(Exception e){
                e.printStackTrace();
                System.err.println("\n\n\n");
            }
        }

    }
```

该类使用创建的JiBX客户端存根在Web服务上访问getPrice和update操作。getPrice方法为股票"ABC"发送请求并显示当前price。update方法为股票"ABC"设置价格为42.35。
现在通过在Axis2_HOME/samples/quickstartjibx目录的控制台键入"[b]ant run.client[b]"来构建并运行客户端。
你应该得到以下输出:

Java代码

```
42
done
```

参考JiBX代码生成集成来得到更多关于同Axis2使用JiBX的信息细节。

总结

Axis2是一个立刻让web服务运行起来的灵活和健壮的方式。本指南呈现了创建一个可以在Axis2上部署的服务的五种方法。现在你拥有了使用多种不同技术来创建Web服务的灵活性。

进一步学习

Apache Axis2-http://ws.apache.org/axis2/
Axis2 Architecture-http://ws.apache.org/axis2/1_0/Axis2ArchitectureGuide.html
Introduction to Apache Axis2-http://www.redhat.com/magazine/021jul06/features/apache_axis2/
Working With Apache Axis2-http://www.wso2.net/articles/axis2/java/2006/09/13/working-with-axis2

在UDDI注册中心里使用WSDL | 学习D语言之helloworld

23:22 | 浏览 (35905) | 评论 (14) | 分类: Java | 相关推荐

评论

14 楼 minjiaren 2009-06-23 引用

我自己写了个DEMO,用ADB方式部署的,为什么
我用wsdl2java代码生成的代码没有STUB的代码呢?
好郁闷,
http://topic.csdn.net/u/20090623/16/a4282d10-56d4-4851-a50f-03a113c772af.html?2178
到这个帖子帮小弟指点一二..

13 楼 hideto 2009-01-05 引用

不用刷屏了，楼主现在已经抛弃WS-*规范，回归Rails和REST阵营了
Axis这么古老的东西怎么还有人用啊？

12 楼 whoyoulove 2009-01-05 引用

1. fasdfa
1. adsfasd
1. adsf
1. asdf
1. asdf
1. adsf
1. adsf

11 楼 whoyoulove 2009-01-05 引用

fasdf

10 楼 whoyoulove 2009-01-05 引用

[flash=200,200]
有点不是很清楚， 希望有详细的步骤： [/flash]

9 楼 whoyoulove 2009-01-05 引用

> 引用
>
> sdfasdfasdf

8 楼 whoyoulove 2009-01-05 引用

> **Java**代码
>
> ```
> 1. zdfasdfasdf
> 2. fasdfasdf
> 3. asdf
> 4. asdfasdfasdf
> ```

7 楼 whoyoulove 2009-01-05 引用

> **Java**代码
>
> ```
> 1. java
> ```

6 楼 whoyoulove 2009-01-05 引用

> 1.

5 楼 kevindurant 2009-01-02 引用

StockQuoteServiceStub
每个客户端都有这个类...我看例子似乎没有发现它呀???

高人们指点下?

4 楼 sanshiz 2008-10-10 引用

有点不是很清楚， 希望有详细的步骤：

毕竟对于读者来说会更顺利些， 然后逐步明白， 这样岂不更好：？

行100， 而今至90；

3 楼 zznj1123 2008-06-26 引用

看得很累,希望楼主写个HelloWorld.谢谢

2 楼 wasteland 2008-05-16 引用

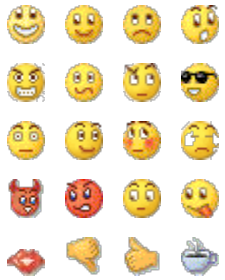
看的很累...

1 楼 为你而来 2007-06-11 引用

不错啊，正在学习中

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交