环境变量配置：

把bin路径添加到path中

WSDL文档

1. web service接口

types元素（该元素就是schema文档）

2n个message元素

portType元素（包含n个operation子元素，每个operation定义一个ws操作）

1. web service实现

binding元素（n个operation子元素，详细定义operation操作）

service元素（part元素。指定了webservice绑定地址）

soap协议：

1. 根元素是Envelope;
2. header包括

默认情况，header元素不强制出现，

header元素由程序员控制添加，主要用于携带一些额外的信息，比如用户名，密码

1. body元素

如果调用正确。body元素的内容应该遵守wsdl所要求的格式。

如果调用错误，body元素的内容就是fault子元素

wsdl文档描述了webservice三个方面

what：该webservice包含了什么操作。

how：该webservice的操作应该怎么调用。

where：该webservice的服务地址。

使用cxf开发web service 服务端

1. 开发一个Web Service业务接口。该接口要用@WebService修饰
2. 开发一个接口实现类。实现类也需要用@WebService修饰
3. 发布WebService

|  |
| --- |
| **package** com.lee.server;  **import** javax.xml.ws.Endpoint;  **import** org.fkjava.cxf.ws.HelloWorld;  **import** org.fkjava.cxf.ws.HelloWorldImpl;  **public** **class** Server {  **public** **static** **void** main(String[] args){  HelloWorld hello = **new** HelloWorldImpl();  //调用Endpoint的publish方法发布web server服务  Endpoint.*publish*("http://192.168.0.101/ccc", hello);  System.***out***.println("暴露成功");  }  } |

浏览器访问的地址为：<http://192.168.0.101/ccc?wsdl>

使用cxf开发web service 客户端

1. 调用CXF提供的wsdl2java工具，根据wsdl文档生成相应java代码（wsdl：web service definition language）

任何语言实现了web service，都需要提供，并暴露wsdl文档。

1. 找到wsdl2java中集成了service的类，该类的实例可当成工厂来使用。
2. 调用Service子类的实例的getxxxport方法，返回远程webservice代理。

|  |
| --- |
| **public** **class** Server {  **public** **static** **void** main(String[] args){  HelloWorldImpl factory = **new** HelloWorldImpl ();  HelloWorld hw = factory.getHelloWordPort();  System.out.println(hw.sayHi(“xxx“));  }  } |

形参和返回值

1. 当形参，返回值是String，基本类型的时候，CXF肯定可以轻松处理。
2. 当形参、返回值是JavaBean式的复合类，List集合，数组等，CXF也可以处理。
3. 还有一些像Map或者非javaBean式的复合类，CXF不能处理。

|  |
| --- |
| **package** org.fkjava.cxf.ws;  **public** **class** Cat {  **private** Integer id;  **private** String name;  **private** String color;  **public** Cat() {  **super**();  }  **public** Cat(Integer id, String name, String color) {  **super**();  **this**.id = id;  **this**.name = name;  **this**.color = color;  }    } |

|  |
| --- |
| **package** org.fkjava.cxf.ws;  **import** java.util.ArrayList;  **import** java.util.HashMap;  **import** java.util.List;  **import** java.util.Map;  **public** **class** UserService {  **static** Map<User,List<Cat>> *db* = **new** HashMap<>();    //初始化块  **static**{  List<Cat> cat1 = **new** ArrayList<Cat>();  cat1.add(**new** Cat(1,"name1","red1"));  cat1.add(**new** Cat(2,"name2","red2"));  *db*.put(**new** User("user1","age1","address1"), cat1);    List<Cat> cat2 = **new** ArrayList<Cat>();  cat2.add(**new** Cat(3,"name3","red3"));  cat2.add(**new** Cat(4,"name4","red4"));  *db*.put(**new** User("user2","age2","address2"), cat2);  }  **public** List<Cat> getCatByUser(User user) {  **return** *db*.get(user);  }  } |

|  |
| --- |
| **package** org.fkjava.cxf.ws;  **public** **class** User {  **private** String userName;  **private** String age;  **private** String address;      //无参数构造器  **public** User() {  }      //构造器  **public** User(String userName, String age, String address) {  **super**();  **this**.userName = userName;  **this**.age = age;  **this**.address = address;  }  **public** String getUserName() {  **return** userName;  }  **public** **void** setUserName(String userName) {  **this**.userName = userName;  }  **public** String getAge() {  **return** age;  }  **public** **void** setAge(String age) {  **this**.age = age;  }  **public** String getAddress() {  **return** address;  }  **public** **void** setAddress(String address) {  **this**.address = address;  }  @Override  **public** **int** hashCode() {  **final** **int** prime = 31;  **int** result = 1;  result = prime \* result + ((address == **null**) ? 0 : address.hashCode());  result = prime \* result + ((userName == **null**) ? 0 : userName.hashCode());  **return** result;  }  @Override  **public** **boolean** equals(Object obj) {  **if** (**this** == obj)  **return** **true**;  **if** (obj == **null**)  **return** **false**;  **if** (getClass() != obj.getClass())  **return** **false**;  User other = (User) obj;  **if** (address == **null**) {  **if** (other.address != **null**)  **return** **false**;  } **else** **if** (!address.equals(other.address))  **return** **false**;  **if** (userName == **null**) {  **if** (other.userName != **null**)  **return** **false**;  } **else** **if** (!userName.equals(other.userName))  **return** **false**;  **return** **true**;  }    } |

|  |
| --- |
| **package** org.fkjava.cxf.ws;  **import** java.util.List;  **import** javax.jws.WebService;  @WebService(endpointInterface="org.fkjava.cxf.ws.HelloWorld",serviceName="HelloWorld")  **public** **class** HelloWorldImpl **implements** HelloWorld {  @Override  **public** String sayHi(String name) {  **return** name+"你好";  }  @Override  **public** List<Cat> getCatByUser(User user) {  UserService us = **new** UserService();  **return** us.getCatByUser(user);  }  } |

|  |
| --- |
| **package** org.fkjava.cxf.ws;  **import** java.util.List;  **import** javax.jws.WebService;  @WebService  **public** **interface** HelloWorld {  String sayHi(String name);    List<Cat> getCatByUser(User user);  } |

客户端这样写

|  |
| --- |
| **public** **class** Main {  **public** **static** **void** main(String[] args){  HelloWorldImpl factory = **new** HelloWorldImpl ();  HelloWorld hw = factory.getHelloWordPort();  System.***out***.println(hw.sayHi("xxx"));    User user = **new** User();  user.setUserName("xx");  user.setAge(12);  user.setAddress("xxxx");  List<Cat> cat1 = hw.getCatByUser(user);  **for**(Cat cat:cat1){  Sysetm.out.println(cat.getName());  }  }  } |

|  |
| --- |
| <?xml version="1.0"?>  <wsdl:definitions targetNamespace="http://ws.cxf.fkjava.org/" name="HelloWorld" xmlns:ns1="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws.cxf.fkjava.org/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <wsdl:types>  <xs:schema targetNamespace="http://ws.cxf.fkjava.org/" xmlns:tns="http://ws.cxf.fkjava.org/" version="1.0" elementFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="getCatByUser" type="tns:getCatByUser"/>  <xs:element name="getCatByUserResponse" type="tns:getCatByUserResponse"/>  <xs:element name="sayHi" type="tns:sayHi"/>  <xs:element name="sayHiResponse" type="tns:sayHiResponse"/>  <xs:complexType name="getCatByUser">  <xs:sequence>  <xs:element name="arg0" type="tns:user" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="user">  <xs:sequence>  <xs:element name="address" type="xs:string" minOccurs="0"/>  <xs:element name="age" type="xs:string" minOccurs="0"/>  <xs:element name="userName" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="getCatByUserResponse">  <xs:sequence>  <xs:element name="return" type="tns:cat" minOccurs="0" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="cat">  <xs:sequence/>  </xs:complexType>  <xs:complexType name="sayHi">  <xs:sequence>  <xs:element name="arg0" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="sayHiResponse">  <xs:sequence>  <xs:element name="return" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  </xs:schema>  </wsdl:types>  <wsdl:message name="sayHiResponse">  <wsdl:part name="parameters" element="tns:sayHiResponse"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCatByUser">  <wsdl:part name="parameters" element="tns:getCatByUser"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCatByUserResponse">  <wsdl:part name="parameters" element="tns:getCatByUserResponse"> </wsdl:part>  </wsdl:message>  <wsdl:message name="sayHi">  <wsdl:part name="parameters" element="tns:sayHi"> </wsdl:part>  </wsdl:message>  <wsdl:portType name="HelloWorld">  <wsdl:operation name="getCatByUser">  <wsdl:input name="getCatByUser" message="tns:getCatByUser"> </wsdl:input>  <wsdl:output name="getCatByUserResponse" message="tns:getCatByUserResponse"> </wsdl:output>  </wsdl:operation>  <wsdl:operation name="sayHi">  <wsdl:input name="sayHi" message="tns:sayHi"> </wsdl:input>  <wsdl:output name="sayHiResponse" message="tns:sayHiResponse"> </wsdl:output>  </wsdl:operation>  </wsdl:portType>  <wsdl:binding name="HelloWorldSoapBinding" type="tns:HelloWorld">  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>  <wsdl:operation name="getCatByUser">  <soap:operation style="document" soapAction=""/>  <wsdl:input name="getCatByUser">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="getCatByUserResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  <wsdl:operation name="sayHi">  <soap:operation style="document" soapAction=""/>  <wsdl:input name="sayHi">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="sayHiResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  </wsdl:binding>  <wsdl:service name="HelloWorld">  <wsdl:port name="HelloWorldImplPort" binding="tns:HelloWorldSoapBinding">  <soap:address location="http://192.168.0.101/ccc"/>  </wsdl:port>  </wsdl:service>  </wsdl:definitions> |

Web Service三个技术基础

1. WSDL Web Service Definition Language - Web Service定义语言

WSDL文档

Web Service接口

type元素（该元素就是scheme文档）

2n个message元素

portType元素（包含n个operation子元素，每个operation定义一个WS操作：实现）

Web Service实现

1. SOAP
2. UDDI

一次webservice调用，其实并不是方法调用，而是发送soap消息（即xml文档片段）、

对于sayHi操作来说，

|  |
| --- |
| <xs:complexType name="sayHi">  <xs:sequence>  <xs:element name="arg0" type="xs:string" minOccurs="0"/></xs:sequence>  </xs:complexType>  <xs:complexType name="sayHiResponse">  <xs:sequence>  <xs:element name="return" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType> |

传入的消息是:

<sayHi>

<arg0>字符串</arg0>

</ sayHi >

传出的消息是：

<sayHiResponse>

<return>字符串</return>

</sayHiResponse>

对于getCatByUser操作来说，

|  |
| --- |
| <xs:complexType name="getCatByUser">  <xs:sequence>  <xs:element name="arg0" type="tns:user" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="user">  <xs:sequence>  <xs:element name="address" type="xs:string" minOccurs="0"/>  <xs:element name="age" type="xs:string" minOccurs="0"/>  <xs:element name="userName" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="getCatByUserResponse">  <xs:sequence>  <xs:element name="return" type="tns:cat" minOccurs="0" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="cat">  <xs:sequence/>  </xs:complexType> |

传入的消息是：

<getCatByUser>

<arg0>

<address>字符串</address>

< age >字符串</ age >

< userName >字符串</ userName >

<arg0>

</getCatByUser>

传出消息是：

<getCatByUserResponse>

<return>//可出现无数次

//无

</return>

</getCatByUserResponse>

调用一次Web Service的本质：

1. 客户端把调用方法参数，转换为xml文档片段（soap消息，input消息）。该文档片段必须符合wsdl定义的格式。
2. 客户端通过网络，把xml文档片段传给服务器。
3. 服务器接收到xml文档片段。
4. 服务器端解析xml文档片段，提取其中数据。并把数据转换成调用web service所需要的参数值。
5. 服务器执行方法。
6. 把执行方法得到的返回值，再次转换生成为xml文档片段（soap消息，output消息）。该文档片段必须符合wsdl定义的格式。
7. 通过网络，把xml文档片段传给客户端。
8. 客户端接收到xml文档片段
9. 客户端解析xml文档片段，提取其中数据。并把数据转换成调用web service所需要的返回值。

从上面的调用本质来看，要一个语言支持web service。唯一的要求是：该语言支持xml文档解析、生成、支持网络传输。

只要得到web service的wsdl文档就可以完成调用。

对于在CXF开发中，如果遇到系统无法处理的类型，就需要程序员自行处理。

处理思路：提供一个转换器。该转换器负责把CXF不能处理的类型，转换成CXF能处理的类型

1. 使用：@XmlJavaTypeAdapter修饰CXF无法自动处理的类型（使用该Annotation时，通过value属性指定一个转换器）
2. 实现自己的转换器。实现转换器的时候，需要开发一个CXF能处理的类型。

对于getCat操作来说：

传入消息是

<getCat></ getCat >

传出消息是

< getCatResponse >

<return>//出现一次

<entries>//出现0到n次

<key>字符串</key>

<value>

<color></color>

<id></id>

<name></name>

</value>

</entries>

</return>

</ getCatResponse >

为了让程序员能访问、修改CXF框架所生成的SOAP消息，CXF提供了拦截器

服务器端添加拦截器

1. 获取Endpoint.*publish*的返回值。
2. 调用该方法返回值的getInInterceptors()/getOutInterceptors()方法来获取in、out拦截器列表，接下来就可以添加拦截器了

客户端添加拦截器

1. 调用ClientProxy.getClient方法，可以远程web service的代理参数

2.调用client对象的getInInterceptors()/getOutInterceptors()方法来获取in、out拦截器列表，接下来就可以添加拦截器了

对于sayHi操作

<?xml version=”1.0” ?>

<S:Envelope xmlns:S=”http://schemas.xmlsoap.org/soap/envelope/”>

<S:Body>

<ns2:sayHi xmlns:ns2=”http://ws.cxf.fkjava.org/”>

<arg0>孙悟空</arg0>

</ns2:sayHi>

</S:Body>

</S:envelope>

传出消息

<soap:Envelope xmlns:soap=”http://schemas.xmlsoap.org/soap/envelope”>

<soap:Body>

<ns2:sayHiResponse xmlns:ns2=”http://ws.cxf.fkjava.org/”>

<return>xxxx</return>

</ns2:sayHiResponse>

</soap：Body>

</soap:Envelope>

soap协议：

1. 根元素是Envelope;
2. header包括

默认情况，header元素不强制出现，

header元素由程序员控制添加，主要用于携带一些额外的信息，比如用户名，密码

1. body元素

如果调用正确。body元素的内容应该遵守wsdl所要求的格式。

如果调用错误，body元素的内容就是fault子元素

自定义拦截器，需要实现InterCepter接口。实际上我们一般继承AbstractPhaseIntercepter.

服务器端拦截器：

server中添加

|  |
| --- |
| //添加In拦截器  ep.getInInterceptors().add(**new** AuthInterceptor()); |

拦截器

|  |
| --- |
| **package** com.lee.util;  **import** java.util.List;  **import** org.apache.cxf.binding.soap.SoapMessage;  **import** org.apache.cxf.headers.Header;  **import** org.apache.cxf.interceptor.Fault;  **import** org.apache.cxf.phase.AbstractPhaseInterceptor;  **import** org.apache.cxf.phase.Phase;  **import** org.w3c.dom.Element;  **import** org.w3c.dom.NodeList;  //通过AuthInterceptor可以指定拦截器在哪个阶段起作用  **public** **class** AuthInterceptor **extends** AbstractPhaseInterceptor<SoapMessage> {  **public** AuthInterceptor(){  **super**(Phase.***PRE\_INVOKE***);//该拦截器会在调用之前拦截soap消息  }    **public** **void** handleMessage(SoapMessage msg) **throws** Fault{  System.***out***.println("-------"+msg);  List<Header> headers = msg.getHeaders();  //如果没有header信息  **if**(headers == **null**||headers.size()<1){  **throw** **new** Fault(**new** IllegalArgumentException("没有header"));  }    //若第一个header中包含的是userid和userpswd，则取出来  Header firstHeader = headers.get(0);  Element ele = (Element) firstHeader.getObject();  NodeList userIds = ele.getElementsByTagName("userId");  NodeList userPds = ele.getElementsByTagName("userPd");  //userid和userPd里的第一个元素的文本内容作为用户名和密码  String userId = userIds.item(0).getTextContent();  String userPd = userPds.item(0).getTextContent();  }  } |

客户端拦截器:

客户端中这样写

|  |
| --- |
| client。getOutInterceptors().add(new AuthInterceptor("userId","userPd")); |

|  |
| --- |
| **package** com.lee.util;  **import** org.apache.cxf.binding.soap.SoapMessage;  **import** org.apache.cxf.interceptor.Fault;  **import** org.apache.cxf.phase.AbstractPhaseInterceptor;  **import** org.apache.cxf.phase.Phase;  **import** org.w3c.dom.Element;  **public** **class** AuthInterceptor **extends** AbstractPhaseInterceptor<SoapMessage>{  **private** String userId;  **private** String passWd;  **public** AuthInterceptor(String userId,String userPd){  **super**(Phase.PREPARE\_SEND);  **this**.userId = userId;  **this**.passWd = userPd;  }    **public** **void** handleMessage(SoapMessage msg) **throws** Fault{  List<Header> headers = msg.getHeaders();  //此处创建的元素应该是按照服务器的要求创建  Document doc = DOMUtils.createDocument();//创建document对象  Element ele = doc.createElement("authHeader");  Element id = doc.createElement("userId");  Element ps = doc.createElement("userPd");    id.setTextContent(userId);  ps.setTextContent(userPd);  ele.appendChild(id);  ele.appendChild(ps);    /\*\*  \* 上面代码生成了如下的xml文档片段  \* <authHeader>  \* <userId>userId</userId>  \* <userPd>userPd</userPd>  \* </authHeader>  \*\*/  //把ele元素包装成header，并添加到soap消息中  headers.add(**new** Header(**new** QName(),ele));    }  } |

CXF与spring的整合

在传统java EE的基础上添加一层web service就可以对外暴露web service 这样就允许任何平台，任何语言编写的程序来调用java EE应用。

添加转换器之后的wsdl文档

|  |
| --- |
| <?xml version="1.0"?>  <wsdl:definitions targetNamespace="http://ws.cxf.fkjava.org/" name="HelloWorld" xmlns:ns1="http://schemas.xmlsoap.org/soap/http" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws.cxf.fkjava.org/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">  <wsdl:types>  <xs:schema targetNamespace="http://ws.cxf.fkjava.org/" xmlns:tns="http://ws.cxf.fkjava.org/" version="1.0" elementFormDefault="unqualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">  <xs:element name="getCat" type="tns:getCat"/>  <xs:element name="getCatByUser" type="tns:getCatByUser"/>  <xs:element name="getCatByUserResponse" type="tns:getCatByUserResponse"/>  <xs:element name="getCatResponse" type="tns:getCatResponse"/>  <xs:element name="sayHi" type="tns:sayHi"/>  <xs:element name="sayHiResponse" type="tns:sayHiResponse"/>  <xs:complexType name="getCat">  <xs:sequence/>  </xs:complexType>  <xs:complexType name="getCatResponse">  <xs:sequence>  <xs:element name="return" type="tns:stringCat" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="stringCat">  <xs:sequence>  <xs:element name="entries" type="tns:entry" minOccurs="0" nillable="true" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="entry">  <xs:sequence>  <xs:element name="key" type="xs:string" minOccurs="0"/>  <xs:element name="value" type="tns:cat" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="cat">  <xs:sequence/>  </xs:complexType>  <xs:complexType name="getCatByUser">  <xs:sequence>  <xs:element name="arg0" type="tns:user" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="user">  <xs:sequence>  <xs:element name="address" type="xs:string" minOccurs="0"/>  <xs:element name="age" type="xs:string" minOccurs="0"/>  <xs:element name="userName" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="getCatByUserResponse">  <xs:sequence>  <xs:element name="return" type="tns:cat" minOccurs="0" maxOccurs="unbounded"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="sayHi">  <xs:sequence>  <xs:element name="arg0" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  <xs:complexType name="sayHiResponse">  <xs:sequence>  <xs:element name="return" type="xs:string" minOccurs="0"/>  </xs:sequence>  </xs:complexType>  </xs:schema>  </wsdl:types>  <wsdl:message name="sayHiResponse">  <wsdl:part name="parameters" element="tns:sayHiResponse"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCat">  <wsdl:part name="parameters" element="tns:getCat"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCatByUser">  <wsdl:part name="parameters" element="tns:getCatByUser"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCatByUserResponse">  <wsdl:part name="parameters" element="tns:getCatByUserResponse"> </wsdl:part>  </wsdl:message>  <wsdl:message name="getCatResponse">  <wsdl:part name="parameters" element="tns:getCatResponse"> </wsdl:part>  </wsdl:message>  <wsdl:message name="sayHi">  <wsdl:part name="parameters" element="tns:sayHi"> </wsdl:part>  </wsdl:message>  <wsdl:portType name="HelloWorld">  <wsdl:operation name="getCat">  <wsdl:input name="getCat" message="tns:getCat"> </wsdl:input>  <wsdl:output name="getCatResponse" message="tns:getCatResponse"> </wsdl:output>  </wsdl:operation>  <wsdl:operation name="getCatByUser">  <wsdl:input name="getCatByUser" message="tns:getCatByUser"> </wsdl:input>  <wsdl:output name="getCatByUserResponse" message="tns:getCatByUserResponse"> </wsdl:output>  </wsdl:operation>  <wsdl:operation name="sayHi">  <wsdl:input name="sayHi" message="tns:sayHi"> </wsdl:input>  <wsdl:output name="sayHiResponse" message="tns:sayHiResponse"> </wsdl:output>  </wsdl:operation>  </wsdl:portType>  <wsdl:binding name="HelloWorldSoapBinding" type="tns:HelloWorld">  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>  <wsdl:operation name="getCat">  <soap:operation style="document" soapAction=""/>  <wsdl:input name="getCat">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="getCatResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  <wsdl:operation name="getCatByUser">  <soap:operation style="document" soapAction=""/>  <wsdl:input name="getCatByUser">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="getCatByUserResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  <wsdl:operation name="sayHi">  <soap:operation style="document" soapAction=""/>  <wsdl:input name="sayHi">  <soap:body use="literal"/>  </wsdl:input>  <wsdl:output name="sayHiResponse">  <soap:body use="literal"/>  </wsdl:output>  </wsdl:operation>  </wsdl:binding>  <wsdl:service name="HelloWorld">  <wsdl:port name="HelloWorldImplPort" binding="tns:HelloWorldSoapBinding">  <soap:address location="http://192.168.0.101/ccc/hello"/>  </wsdl:port>  </wsdl:service>  </wsdl:definitions> |