用户操作

[留言] [发消息] [加为好友]

订阅我的博客

1 位读者 POWERED BY FEEDSKY 订阅

订阅到 鲜果

订阅到 Google

订阅到 抓虾

thinkGhoster的公告

文章分类

存档

2008年11月(1)
2008年07月(3)
2008年04月(3)
2007年11月(3)
2007年10月(7)

转 一个简单的Spring Web Service示例 收藏

刚接触web service，好不容易找到一篇spring-ws的例子，还琢磨了好长一段时间，很多概念性的问题都没弄清楚。只能依葫芦画瓢，照搬过来，稍微修改了一下，使结构更加清晰，原文出自http://fuxueliang.javaeye.com/blog/175184#。

基本环境：

JDK6、Tomcat 6.0、MyEclipse 6.6、spring 2.0、spring-ws-1.5.5

1、spring-ws-servlet.xml

这个地方出现了一段插曲，hello.wsdl放在WEB-INF下老是报错，说hello.wsdl找不到，后来放到classpath下才OK。

创建一个Web项目, 由于Spring Web Service是基于Spring MVC的, 在web.xml中添加如下servlet, 并在WEB-INF下建立SpringMVC的默认配置文件spring-ws-servlet.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <bean id="payloadMapping" class="org.springframework.ws.server.endpoint.mapping.PayloadRootQNameEndpointMapping">
        <property name="endpointMap">
        <map>
            <entry key="{http://www.ispring.com/ws/hello}eRequest">
                <ref bean="helloEndpoint"/>
            </entry>
        </map>
    </property>
</bean>

<bean id="hello" class="org.springframework.ws.wsdl.wsdl11.SimpleWsdl11Definition"
>
    <!-- --><property name="wsdl" value="classpath://hello.wsdl"></property>
    <!-- <property name="wsdl" value="/WEB_INF/hello.wsdl"></property> -->
    <!-- <constructor-arg value="/WEB_INF/hello.wsdl"/> -->
</bean>

<bean id="helloEndpoint" class="com.sws.HelloEndPoint">
    <property name="helloService" ref="helloService"></property>
</bean>

<bean id="helloService" class="com.sws.HelloServiceImpl"></bean>

</beans>
```

其中最主要的bean就是payloadMapping, 它定义了接收到的message与endpoint之间的mapping关系:将SOAP Body中包含的xml的根节点的QName为{http://www.fuxueliang.com/ws/hello}HelloRequest交给helloEndpoint处理.

SimpleWsdl11Definition这个bean则是定义了这个服务的wsdl, 访问地址是:

http://localhost:8080/springws/hello.wsdl.

2、web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4"
    xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <!-- begin Spring配置
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/spring-ws-servlet.xml,
        </param-value>
    </context-param>
    <listener>
        <listener-class>
            org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>
    <listener>
        <listener-class>
            org.springframework.web.util.IntrospectorCleanupListener
        </listener-class>
    </listener>-->
    <!-- end Spring配置 -->

    <servlet>
        <servlet-name>spring-ws</servlet-name>
        <servlet-class>org.springframework.ws.transport.http.MessageDispatcherServlet</s
ervlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>spring-ws</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

</web-app>

```

3、hello.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:schema="http://www.ispring.com/ws/hello"
    xmlns:tns="http://www.ispring.com/ws/hello/definitions"
    targetNamespace="http://www.ispring.com/ws/hello/definitions">

    <wsdl:types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.ispring.com/ws/hello">
            <element name="dRequest" type="string" />
            <element name="dResponse" type="string" />
        </schema>
    </wsdl:types>

    <wsdl:message name="bRequest">
        <wsdl:part element="schema:dRequest" name="cRequest" />
    </wsdl:message>
    <wsdl:message name="bResponse">
        <wsdl:part element="schema:dResponse" name="cResponse" />
    </wsdl:message>

    <wsdl:portType name="HelloPortType">
        <wsdl:operation name="sayHello">
            <wsdl:input message="tns:bRequest" name="aRequest" />
            <wsdl:output message="tns:bResponse" name="aResponse" />
        </wsdl:operation>
    </wsdl:portType>
```

```xml

    <wsdl:binding name="HelloBinding" type="tns:HelloPortType">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="sayHello">
            <soap:operation soapAction="" />
            <wsdl:input name="aRequest">
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output name="aResponse">
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="HelloService">
        <wsdl:port binding="tns:HelloBinding" name="HelloPort">
            <soap:address location="http://localhost:8088/springws/webservice" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

4、HelloService.java

```java
package com.sws;

public interface HelloService {

    String sayHello(String name);

}

```

5、HelloServiceImpl.java

```java
package com.sws;

```

```java
public class HelloServiceImpl implements HelloService {

    @Override
    public String sayHello(String name) {
        return "Hello, " + name + "!";
    }

}

```

6、HelloEndPoint.java

实现一个EndPoint来处理接收到的xml及返回xml.当然, Spring Web Service提供了很多抽象的实现, 包括Dom4j, JDom等等.这里我们使用JDK自带的, 需要继承org.springframework.ws.server.endpoint.AbstractDomPayloadEndpoint.

```java
package com.sws;

import org.springframework.util.Assert;
import org.springframework.ws.server.endpoint.AbstractDomPayloadEndpoint;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
import org.w3c.dom.Text;

public class HelloEndPoint extends AbstractDomPayloadEndpoint{

    /**
     * Namespace of both request and response
     */
    public static final String NAMESPACE_URI = "http://www.ispring.com/ws/hello";

    /**
     * The local name of the expected request
     */
    public static final String HELLO_REQUEST_LOCAL_NAME = "eRequest";

    /**
     * The local name of the created response
     */
    public static final String HELLO_RESPONSE_LOCAL_NAME = "fResponse";

    private HelloService helloService;


    @Override
    protected Element invokeInternal(Element requestElement, Document document) throws Exception {
        Assert.isTrue(NAMESPACE_URI.equals(requestElement.getNamespaceURI()), "Invalid namespace");
        Assert.isTrue(HELLO_REQUEST_LOCAL_NAME.equals(requestElement.getLocalName()), "Invalid local name");

        NodeList children = requestElement.getChildNodes();
        Text requestText = null;
        for(int i=0; i<children.getLength(); i++){
            if(children.item(i).getNodeType() == Node.TEXT_NODE){
                requestText = (Text) children.item(i);
            }
        }

        if(requestText == null){
            throw new IllegalArgumentException("Could not find request text node");
        }

        String response = helloService.sayHello(requestText.getNodeValue());
        Element responseElement = document.createElementNS(NAMESPACE_URI, HELLO_RESPONSE_LOCAL_NAME);
        Text responseText = document.createTextNode(response);
        responseElement.appendChild(responseText);
        return responseElement;
    }


    public HelloService getHelloService() {
        return helloService;
    }


    public void setHelloService(HelloService helloService) {
        this.helloService = helloService;
    }

}

```

7、HelloWebServiceClient.java(saaj实现)

```java
package com.sws;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.Iterator;

import javax.xml.soap.MessageFactory;
import javax.xml.soap.Name;
import javax.xml.soap.SOAPBodyElement;
import javax.xml.soap.SOAPConnection;
import javax.xml.soap.SOAPConnectionFactory;
import javax.xml.soap.SOAPEnvelope;
import javax.xml.soap.SOAPException;
import javax.xml.soap.SOAPFault;
import javax.xml.soap.SOAPMessage;

public class HelloWebServiceClient {

    public static final String NAMESPACE_URI = "http://www.ispring.com/ws/hello";

    public static final String PREFIX = "tns";

    private SOAPConnectionFactory connectionFactory;

    private MessageFactory messageFactory;

    private URL url;

    public HelloWebServiceClient(String url) throws UnsupportedOperationException, SOAPException, MalformedURLException{
        connectionFactory = SOAPConnectionFactory.newInstance();
        messageFactory = MessageFactory.newInstance();
        this.url = new URL(url);
    }

    private SOAPMessage createHelloRequest() throws SOAPException{
        SOAPMessage message = messageFactory.createMessage();
        SOAPEnvelope envelope = message.getSOAPPart().getEnvelope();
        Name helloRequestName = envelope.createName("eRequest",PREFIX,NAMESPACE_URI);
        SOAPBodyElement helloRequestElement = message.getSOAPBody().addBodyElement(helloRequestName);
        helloRequestElement.setValue("ispring");
        return message;
    }

    public void callWebService() throws SOAPException{
        SOAPMessage request = createHelloRequest();
        SOAPConnection connection = connectionFactory.createConnection();
        SOAPMessage response = connection.call(request, url);
        if(!response.getSOAPBody().hasFault()){
            writeHelloResponse(response);
        }else{
            SOAPFault fault = response.getSOAPBody().getFault();
            System.err.println("Received SOAP Fault");
            System.err.println("SOAP Fault Code : " + fault.getFaultCode());
            System.err.println("SOAP Fault String : " + fault.getFaultString());
        }
    }

    private void writeHelloResponse(SOAPMessage message) throws SOAPException{
        SOAPEnvelope envelope = message.getSOAPPart().getEnvelope();
        Name helloResponseName = envelope.createName("fResponse",PREFIX,NAMESPACE_URI);
        Iterator childElements = message.getSOAPBody().getChildElements(helloResponseName);
        SOAPBodyElement helloResponseElement = (SOAPBodyElement)childElements.next();
        String value = helloResponseElement.getTextContent();
        System.out.println("Hello Response [" + value + "]");
    }

    public static void main(String[] args) throws UnsupportedOperationException, MalformedURLException, SOAPException {
        String url = "http://localhost:8088/SpringWS";
        HelloWebServiceClient helloClient = new HelloWebServiceClient(url);
        helloClient.callWebService();
    }

}

```

发表于 @ 2008年11月30日 16:53:00 | 评论(2) | 举报 | 收藏

旧一篇:JavaScript的Static Variable和Instance Property

查看最新精华文章 请访问博客首页相关文章

xiangdian007 □□□Thursday, July 02, 2009 17:22:29 □□ □□

□□□□□□□spring-ws-1.5.5 □□□□□□□□□□□□□jdk1.6□□□□□□□□□□spring-ws-1.5.5□□□□
□□□xiangdian007@163.com□□□

thinkGhoster □□□Friday, July 24, 2009 19:16:42 □□ □□

□□□□□□

发表评论

表 情:

评论内容:

用 户 名: huapuyu6 ☐匿名评论

发表评论