[**真正的轻量级WebService框架——使用JAX-WS(JWS)发布WebService**](https://www.cnblogs.com/coprince/p/5732822.html)

WebService历来都很受重视，特别是Java阵营，WebService框架和技术层出不穷。知名的XFile(新的如CXF)、Axis1、Axis2等。

　　而Sun公司也不甘落后，从早期的JAX-RPC到现在成熟的、支持RPC调用与消息传递的JAX-WS都经过了市场的考验，十分成熟，而且使用JAX-WS开发WebService的收益是很大的，它是轻量级的。

　　我们使用JAX-WS开发WebService只需要很简单的几个步骤：写接口和实现=>发布=>生成客户端(测试或使用)。

　　而在开发阶段我们也不需要导入外部jar包，因为这些api都是现成的。首先是接口的编写(接口中只需要把类注明为@WebService，把 要暴露给客户端的方法注明为@WebMethod即可，其余如@WebResult、@WebParam等都不是必要的，而客户端和服务端的通信用RPC 和Message-Oriented两种，区别和配置以后再说)：

[复制代码](javascript:void(0);)

package service;

import java.util.Date;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import javax.jws.WebResult;

import javax.jws.WebService;

import javax.jws.soap.SOAPBinding;

/\*\*

\* 作为测试的WebService接口

\*

\* @author Johness

\*

\*/

@WebService

@SOAPBinding(style = SOAPBinding.Style.RPC)

public interface SayHiService {

/\*\*

\* 执行测试的WebService方法

\*/

@WebMethod

void SayHiDefault();

/\*\*

\* 执行测试的WebService方法(有参)

\*

\* @param name

\*/

@WebMethod

void SayHi(@WebParam(name = "name") String name);

/\*\*

\* 执行测试的WebService方法(用于时间校验)

\*

\* @param clentTime 客户端时间

\* @return 0表示时间校验失败 1表示校验成功

\*/

@WebMethod

@WebResult(name = "valid")

int CheckTime(@WebParam(name = "clientTime") Date clientTime);

}

[复制代码](javascript:void(0);)

然后是实现类(注解@WebService及其endpointInterface属性是必要的)：

[复制代码](javascript:void(0);)

package service.imp;

import java.text.SimpleDateFormat;

import java.util.Date;

import javax.jws.WebService;

import javax.jws.soap.SOAPBinding;

import service.SayHiService;

/\*\*

\* 作为测试的WebService实现类

\*

\* @author Johness

\*

\*/

@WebService(endpointInterface = "service.SayHiService")

@SOAPBinding(style = SOAPBinding.Style.RPC)

public class SayHiServiceImp implements SayHiService {

@Override

public void SayHiDefault() {

System.out.println("Hi, Johness!");

}

@Override

public void SayHi(String name) {

System.out.println("Hi, " + name + "!");

}

@Override

public int CheckTime(Date clientTime) {

// 精确到秒

String dateServer = new java.sql.Date(System.currentTimeMillis())

.toString()

+ " "

+ new java.sql.Time(System.currentTimeMillis());

String dateClient = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(clientTime);

return dateServer.equals(dateClient) ? 1 : 0;

}

}

[复制代码](javascript:void(0);)

然后是发布(一般有两种方式)：

　　方式一(此方式只能作为调试，有以下bug：

　　　　jdk1.6u17?以下编译器不支持以Endpoint.publish方式发布document方式的soap，必须在service接口和实现类添加“@SOAPBinding(style = SOAPBinding.Style.RPC)”注解；

　　　　访问受限，似乎只能本机访问(应该会绑定到publish的URL上，如下使用localhost的话就只能本机访问)……)：

[复制代码](javascript:void(0);)

package mian;

import javax.xml.ws.Endpoint;

import service.imp.SayHiServiceImp;

public class Main {

/\*\*

\* 发布WebService

\* 简单

\*/

public static void main(String[] args) {

Endpoint.publish("http://localhost:8080/testjws/service/sayHi", new SayHiServiceImp());

}

}

[复制代码](javascript:void(0);)

方式二(基于web服务器Servlet方式)：

　　以Tomcat为例，首先编写sun-jaxws.xml文件并放到WEB-INF下：

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime"

version="2.0">

<endpoint name="SayHiService"

implementation="service.imp.SayHiServiceImpl"

url-pattern="/service/sayHi" />

</endpoints>

[复制代码](javascript:void(0);)

然后改动web.xml，添加listener和servlet(url-pattern要相同哦)：

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" version="2.5">

<listener>

<listener-class>

com.sun.xml.ws.transport.http.servlet.WSServletContextListener

</listener-class>

</listener>

<servlet>

<servlet-name>SayHiService</servlet-name>

<servlet-class>

com.sun.xml.ws.transport.http.servlet.WSServlet

</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>SayHiService</servlet-name>

<url-pattern>/service/sayHi</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

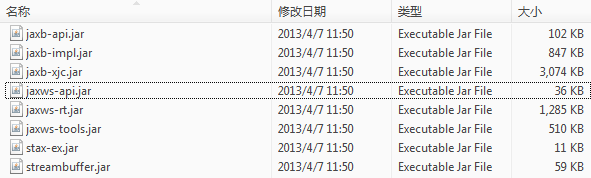
<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

[复制代码](javascript:void(0);)

最后部署到Tomcat里，值得一提的是您可能需要添加以下jar包(因为Tomcat没有)：



　　启动Tomcat。

　　服务端工作就完成了，注意两个事情。

　　注意：项目需要使用UTF-8编码(至少sun-jaxws.xml必须是UTF-8格式的)；

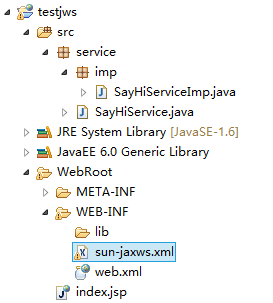
　　　　对于MyEclipse的内置Tomcat，可能会出现不需要手动添加上述jar包，但独立部署时应该添加，因为它们使用的class-path不一样；

　　　　多个不同路径的接口也要使用同一个WSServlet；

　　　　最好加上@SOAPBinding(style = SOAPBinding.Style.RPC)注解。

　　部署好了之后打开浏览器输入网址：[http://localhost:8080/testjws/service/sayHi?wsdl](http://localhost/sayhi/service/sayHi?wsdl)。可以看到东西就证明发布成功了。

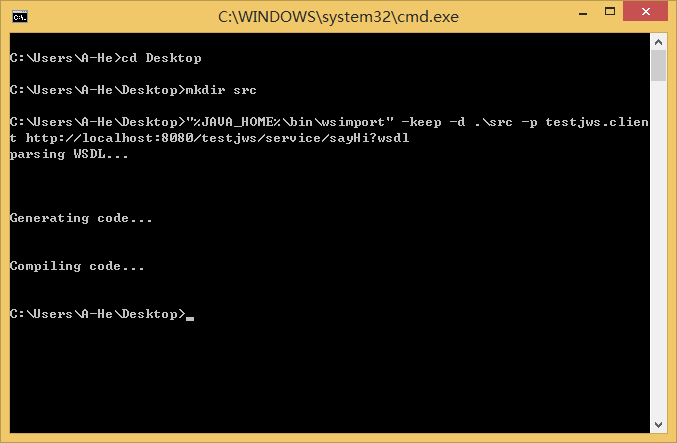
　　附上项目树状图：



　　最后是客户端使用，由于WebService是平台和语言无关的基于xml的，所以我们完全可以使用不同语言来编写或生成客户端。

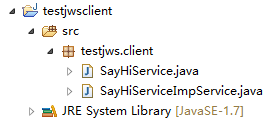
　　一般有三种方式来使用(对于Java语言而言)：

　　　　一，使用jdk自带工具wsimport生成客户端：



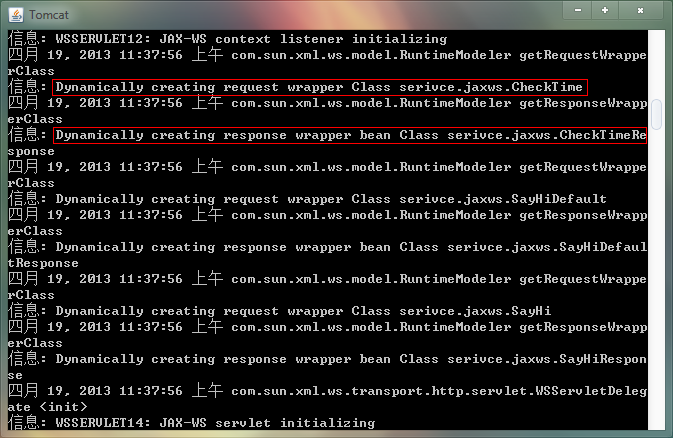
　　jdk自带的wsimport工具生成，上图我是把客户端文件生成到了桌面src文件中(-d)，并保留了源文件(-keep)，指定了包名(-p)。

　　然后我们就可以使用生成的文件来调用服务器暴露的方法了：



　　　　值得一提的是你生成使用的jdk和你客户端的jre需要配套！

　　从上面的目录结构我们可以发现：服务端的每个webmethod都被单独解析成为了一个类(如果使用了实体，实体也会被解析到客户端，并且是源码，所以建议使用实体时慎重)。



　　（上面的图是旧图，只是为了表示一下jaxws是为每个webmethod生成类的情况）

　　而我们的service则被生成了一个代理类来调用服务，接下来我们看看使用情况：

[复制代码](javascript:void(0);)

package test;

import java.util.Date;

import java.util.GregorianCalendar;

import javax.xml.datatype.DatatypeConfigurationException;

import javax.xml.datatype.DatatypeFactory;

import javax.xml.datatype.XMLGregorianCalendar;

import testjws.client.SayHiService;

import testjws.client.SayHiServiceImpService;

public class Main {

public static void main(String[] args) throws DatatypeConfigurationException {

// 获取service

SayHiService service = new SayHiServiceImpService().getSayHiServiceImpPort();

// sayhi

service.sayHiDefault();

service.sayHi("Ahe");

// checktime

// 这里主要说一下时间日期的xml传递，方法还略显复杂

GregorianCalendar calender = new GregorianCalendar();

calender.setTime(new Date(System.currentTimeMillis()));

XMLGregorianCalendar xmldate = DatatypeFactory.newInstance().newXMLGregorianCalendar(calender);

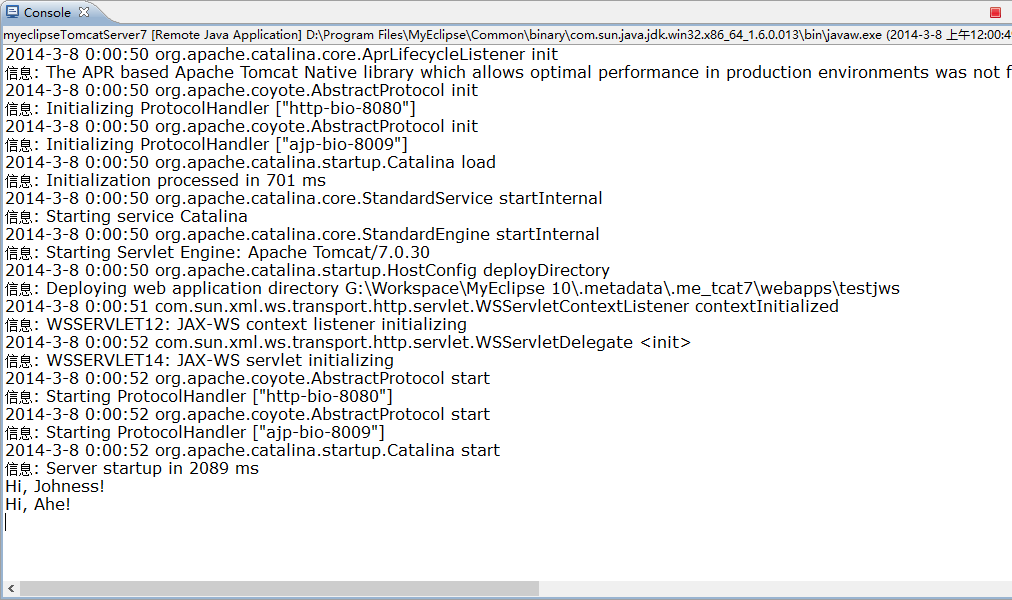
System.out.println(service.checkTime(xmldate));

}

}

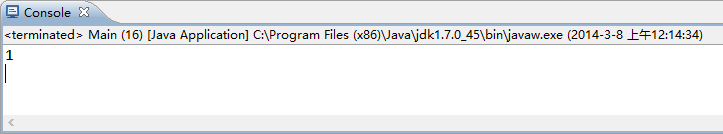
[复制代码](javascript:void(0);)

看看服务器的输出，我们是否调用成功：

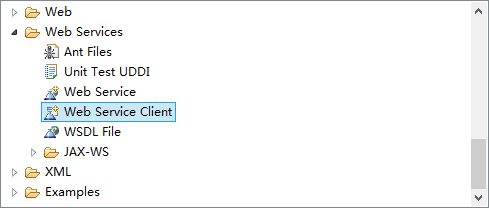


　　　　成功了！

　　　　对于校验时间的方法客户端也收到反馈了：



　　　　二，使用诸如MyEclipse（Eclipse for Jave EE也可以）创建一个Web Service Client的项目



　　　　然后填入wsdl地址即可，后续步骤我就不贴出了。

　　　　三，自己写代码-\_-，其实这个方法不得不说是最好的。