# WebService环境搭建

## 一、ecplise 中把项目转成web service 和web service client方式

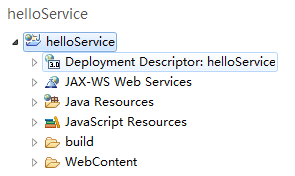
Web service是一个平台独立的，低耦合的，自包含的、基于可编程的web的应用程序，可使用开放的XML（标准通用标记语言下的一个子集）标准来描述、发布、发现、协调和配置这些应用程序，用于开发分布式的互操作的应用程序。

Web Service技术， 能使得运行在不同机器上的不同应用无须借助附加的、专门的第三方软件或硬件， 就可相互交换数据或集成。依据Web Service规范实施的应用之间， 无论它们所使用的语言、 平台或内部协议是什么， 都可以相互交换数据。Web Service是自描述、 自包含的可用网络模块， 可以执行具体的业务功能。Web Service也很容易部署， 因为它们基于一些常规的产业标准以及已有的一些技术，诸如标准通用标记语言下的子集XML、HTTP。Web Service减少了应用接口的花费。Web Service为整个企业甚至多个组织之间的业务流程的集成提供了一个通用机制。

也就是说，通过webservice我们就可以衔接C#、java等之间的通信。

下面，我通过eclipse来建立一个简单的Webservice服务：

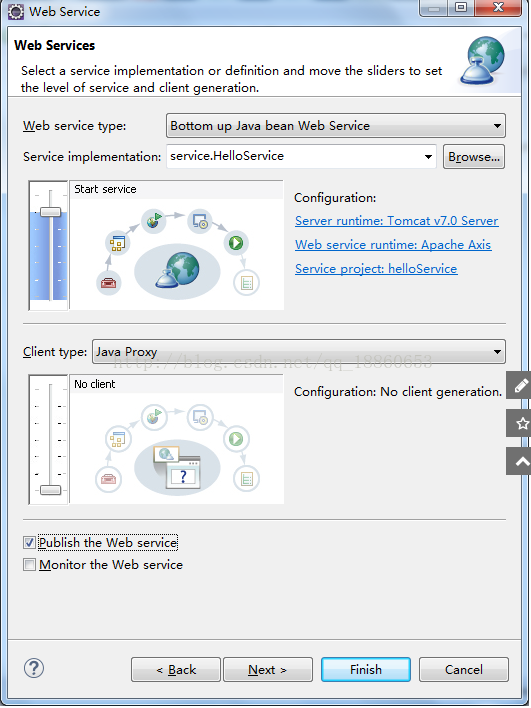
首先，我们在eclipse中像建立一个web项目一样，new->Dynamic Web Project，这里我取名为



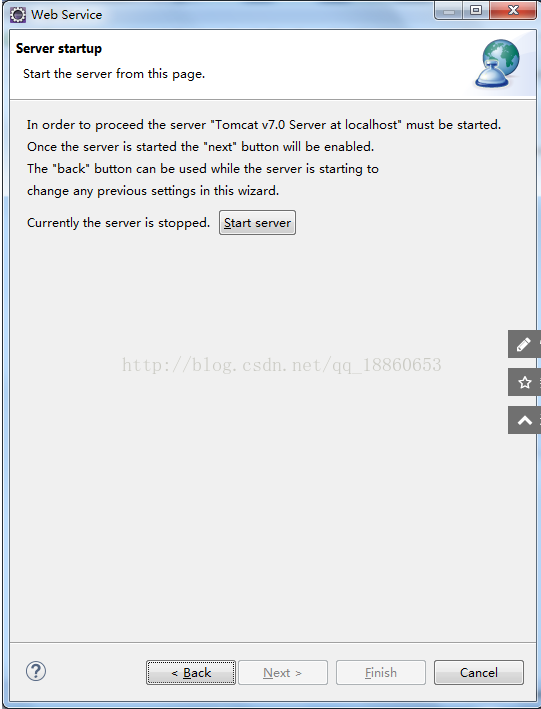
接着，我们先建一个需要暴露给外部的方法，

1. **package** service;
3. **public** **class** HelloService {
4. **public** String say(String name) **throws** InterruptedException{
5. **return** "hello "+name;
6. }
7. }

然后呢，右击这个项目，new -> other->web services->webservice



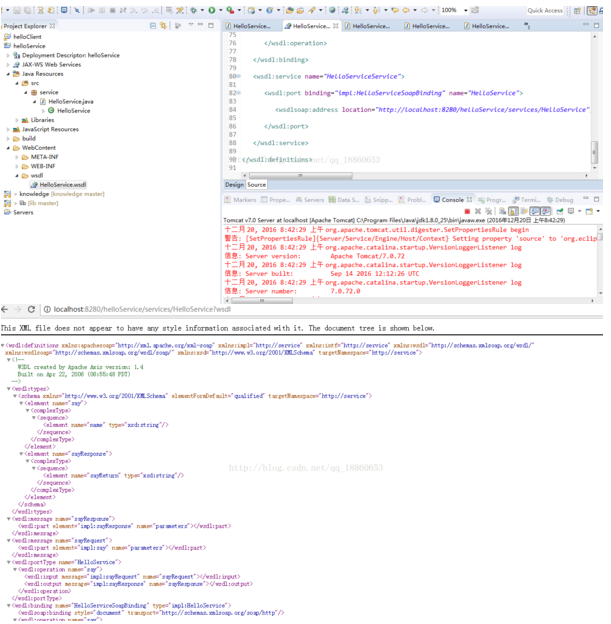
选择需要暴露的实现，service.HelloService，然后选择发布



我们是通过tomcat发布的，直接start即可。常用的框架有，cxf、 axis、 axis2等，这里选择了axis

发布之后，我们打开网页输入地址即可打开它的wsdl：http://localhost:8280/helloService/services/HelloService?wsdl

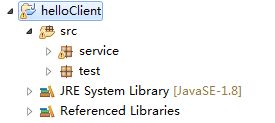
前面的地址，在helloService\WebContent\wsdl\HelloService.wsdl下可以看到，



现在服务端已经建立。

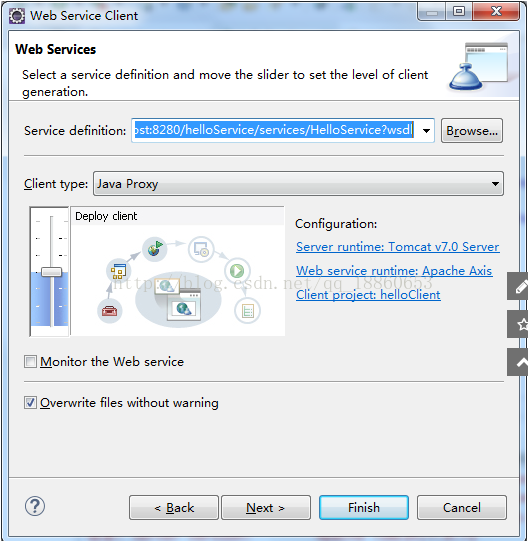
接下来，我们就需要使用client去连这个Webservice服务了，

新建一个java工程（都可以）

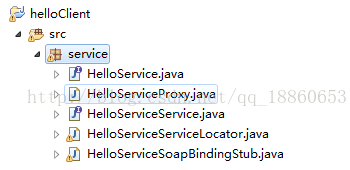


然后新建一个Webservice client就可以，

输入wsdl地址，finish即可

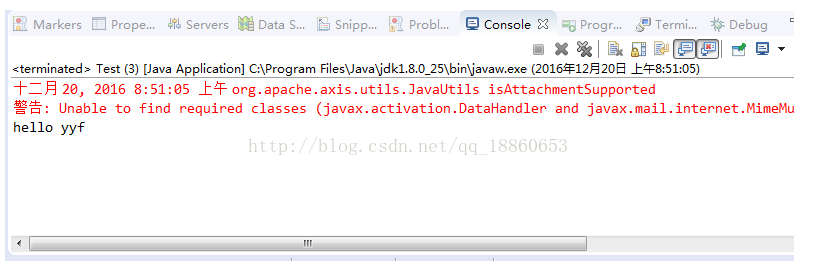


然后可以看到目录下的Webservice java类，



我们新建一个test，去测试以下

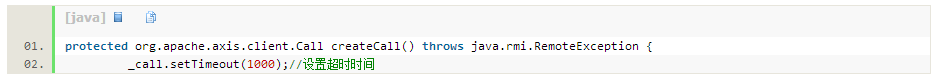
1. **package** test;
3. **import** java.rmi.RemoteException;
5. **import** service.HelloService;
6. **import** service.HelloServiceProxy;
8. **public** **class** Test {
10. **public** **static** **void** main(String[] args) **throws** RemoteException {
11. HelloServiceProxy helloPxy = **new** HelloServiceProxy();
12. HelloService service = helloPxy.getHelloService();
13. String res = service.say("yyf");
14. System.out.println(res);
15. }
17. }



Webservice的入门结束。

客户端超时断开连接设置：

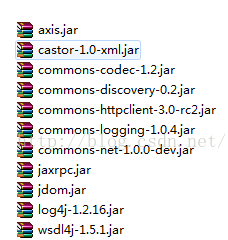
HelloServiceSoapBindingStub->



## 二、 WebService核心文件【server-config.wsdd】详解及调用示例

### 一、准备工作

导入需要的jar包：



### 二、配置web.xml

在web工程的web.xml中添加如下配置：

1. <servlet>
2. <servlet-name>AxisServlet</servlet-name>
3. <servlet-class>org.apache.axis.transport.http.AxisServlet</servlet-class>
4. <load-on-startup>15</load-on-startup>
5. </servlet>
6. <servlet-mapping>
7. <servlet-name>AxisServlet</servlet-name>
8. <url-pattern>/services/\*</url-pattern>
9. </servlet-mapping>

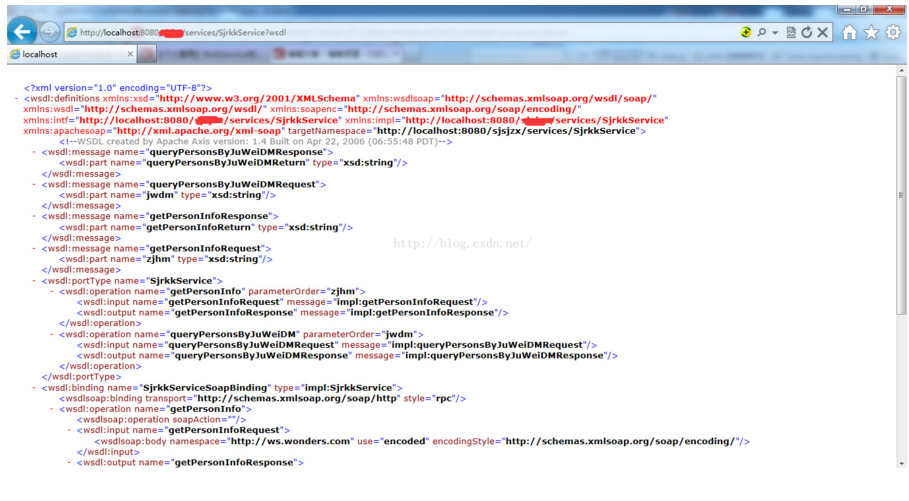
### 三、配置server-config.wsdd文件

在WEB-INF下添加WebService核心文件server-config.wsdd：

1. <?xml version="1.0" encoding="UTF-8"?>
2. <deployment xmlns="http://xml.apache.org/axis/wsdd/"
3. xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
4. <globalConfiguration>
5. <!-- 变量设定 -->
6. <parameter name="sendMultiRefs" value="true" />
7. <parameter name="disablePrettyXML" value="true" />
8. <parameter name="adminPassword" value="admin" />
9. <parameter name="attachments.Directory"
10. value="D:\myproject\calileo\xxx\WebContent\WEB-INF\attachments" />
11. <parameter name="dotNetSoapEncFix" value="true" />
12. <parameter name="enableNamespacePrefixOptimization" value="true" />
13. <parameter name="sendXMLDeclaration" value="true" />
14. <parameter name="attachments.implementation"
15. value="org.apache.axis.attachments.AttachmentsImpl" />
16. <parameter name="sendXsiTypes" value="true" />
17. <requestFlow>
18. <handler type="java:org.apache.axis.handlers.JWSHandler">
19. <parameter name="scope" value="session" />
20. </handler>
21. <handler type="java:org.apache.axis.handlers.JWSHandler">
22. <parameter name="scope" value="request" />
23. <parameter name="extension" value=".jwr" />
24. </handler>
25. </requestFlow>
26. </globalConfiguration>
27. <handler name="LocalResponder"
28. type="java:org.apache.axis.transport.local.LocalResponder" />
29. <handler name="URLMapper" type="java:org.apache.axis.handlers.http.URLMapper" />
30. <handler name="Authenticate"
31. type="java:org.apache.axis.handlers.SimpleAuthenticationHandler" />
33. <!-- 核心重点：服务发布，name：表示发布的service名 -->
34. <service name="SjrkkService" provider="java:RPC">
35. <!-- 发布的service接口类 -->
36. <parameter name="className" value="com.xxx.ws.SjrkkService" />
37. </service>
39. <transport name="http">
40. <requestFlow>
41. <handler type="URLMapper" />
42. <handler type="java:org.apache.axis.handlers.http.HTTPAuthHandler" />
43. </requestFlow>
44. <parameter name="qs:list"
45. value="org.apache.axis.transport.http.QSListHandler" />
46. <parameter name="qs:wsdl"
47. value="org.apache.axis.transport.http.QSWSDLHandler" />
48. <parameter name="qs.list"
49. value="org.apache.axis.transport.http.QSListHandler" />
50. <parameter name="qs.method"
51. value="org.apache.axis.transport.http.QSMethodHandler" />
52. <parameter name="qs:method"
53. value="org.apache.axis.transport.http.QSMethodHandler" />
54. <parameter name="qs.wsdl"
55. value="org.apache.axis.transport.http.QSWSDLHandler" />
56. </transport>
57. <transport name="local">
58. <responseFlow>
59. <handler type="LocalResponder" />
60. </responseFlow>
61. </transport>
62. </deployment>

### 四、测试WSDL

在浏览器输入：[http://localhost:8080/xxx/services/SjrkkService?wsdl](http://localhost:8080/sjsjzx/services/SjrkkService?wsdl)      //xxx和server-config.wsdd文件中的xxx一致，访问结果如下：



### 五、调WebService

**package test;**

**import java.rmi.RemoteException;**

**import org.apache.axis.client.Call;**

**import org.apache.axis.client.Service;**

**import service.HelloService;**

**import service.HelloServiceProxy;**

**public class Test {**

**public static void main(String[] args) {**

**Test test = new Test();**

**String url = "http://localhost:8080/Axis2Service/services/AxisService?wsdl";**

**String method = "say";**

**String idCard = "chow";**

**try {**

**String word = test.getWebServiceResult(url, method, idCard);**

**System.out.println(word);**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**}**

**/\*\***

**\* 调用webservice的方法.**

**\* @param url wsdl链接**

**\* @param method 调用的接口**

**\* @param idCard 传入的参数**

**\* @return**

**\* @throws Exception**

**\*/**

**private String getWebServiceResult( String url, String method, String idCard) throws Exception{**

**String rtnXml = null;**

**try {**

**String endpoint = url;**

**Service service = new Service();**

**Call call = (Call) service.createCall();**

**call.setTargetEndpointAddress(new java.net.URL(endpoint));**

**call.setOperationName(method);**

**rtnXml = (String) call.invoke(new Object[]{idCard});**

**} catch (Exception e) {**

**e.printStackTrace();**

**}**

**return rtnXml;**

**}**

**}**