WebService

# webService的实现思路：

两个角色：

1. 作为服务端
2. 作为客户端

作为客户端：（思路）

1.和普通的Service层的区别就是：需要的数据是从远处（别的）url中获取的。

2.然后处理：处理成自己的Bean，然后像普通的Service一样处理实现（因为一般从远处url里面的都是接口，这里的实现需要根据不同的需要自行实现）。

作为服务端：（思路）

1.提供一系列的接口，其中的接口必须含有注解@WebService（name=”xxxx”）的声明，因为这个声明是表示这个Service可以作为服务接口提供给别人的标志。FYI：其中还有需要用注解参数@WebParam(name=”xxx”)

2.然后生成wsdl：方法：通过weblogic发布部署，然后在web里面看到通过注解声明的Service都在里面——>点击测试，会生成想xml文件，然后通过Apache cxf的一个指令——>生成Client

3.处理，然后可以当做本地的Bean、Service处理

注意：无论作为客户端还是服务端都必须要一个url参数

# WebService部署全步骤

【作为服务端】

1. 编辑需要提供的服务接口
2. 部署发布，获取client代码
3. 在业务层Biz（buslogic中获取端口PROT,写测试）

【作为客户端】

1. 获取端口PROT
2. 调用对方的方法/传入参数测试

# 作为服务端

【编辑提供的服务】

1. 写一个接口

public interface xxxxInterface{

//写一个需要发布的方法（共外界使用的方法）

public abstract String xxxx(String xxxx);

}

1. 写一个普通的java类

自动导入的两个包：

Import java.javax.ejb.Stateless;

Impoirt javax.jws.WebService;

【规律：1.头部】

@WebService

@Stateless(mappedName=”当前的类名，大小写相同”)

【2.继承刚才的接口】

继承接口之后就在里面重写方法，然后在这里面实现

【部署，获取client的代码】

图解：

图1：

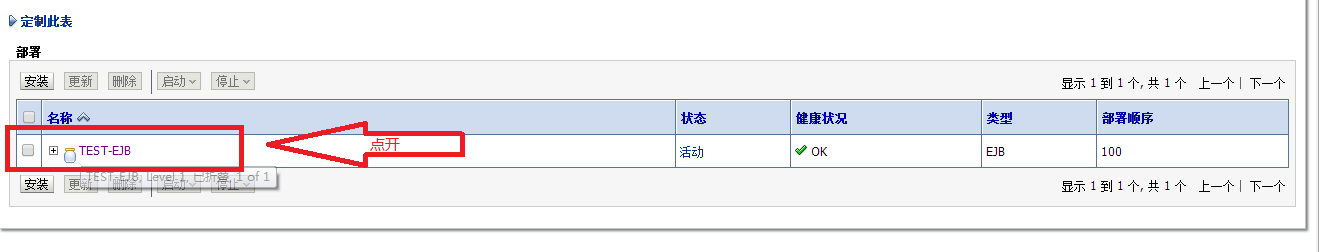
图2：

图3：对应图2的项目内的@WebService注解

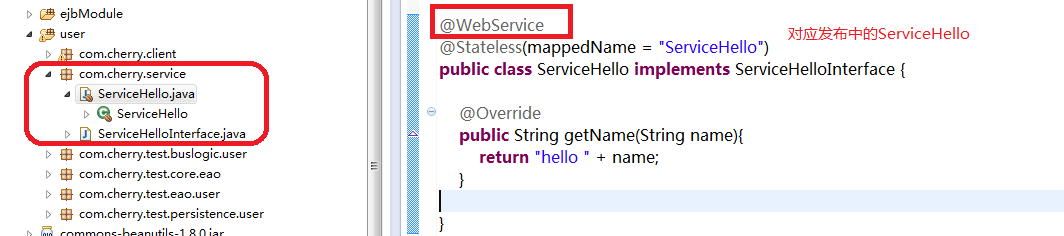


图4：点击图2指向的部分，点击进去之后，进入测试页面



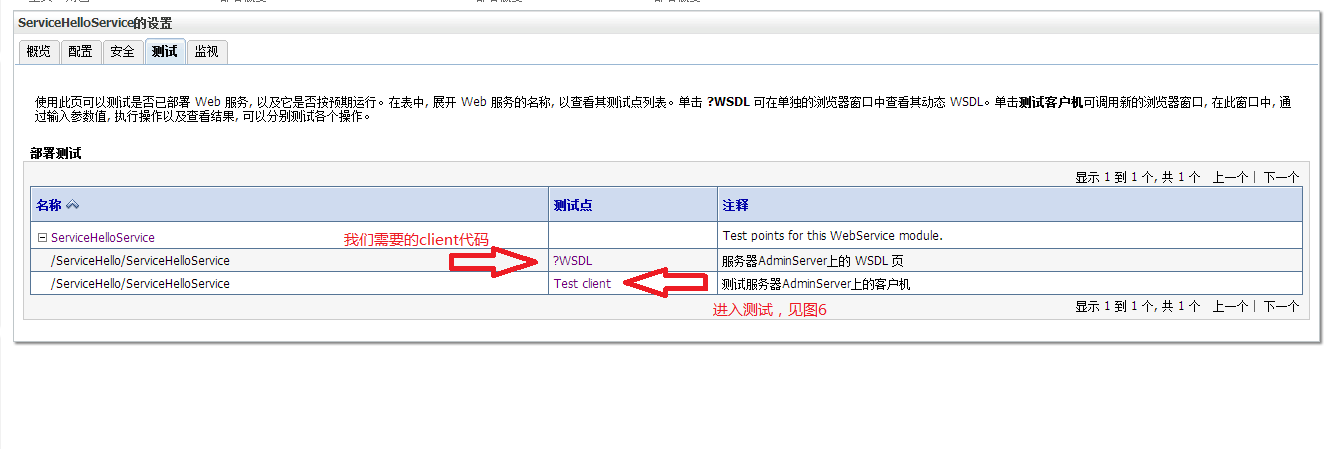
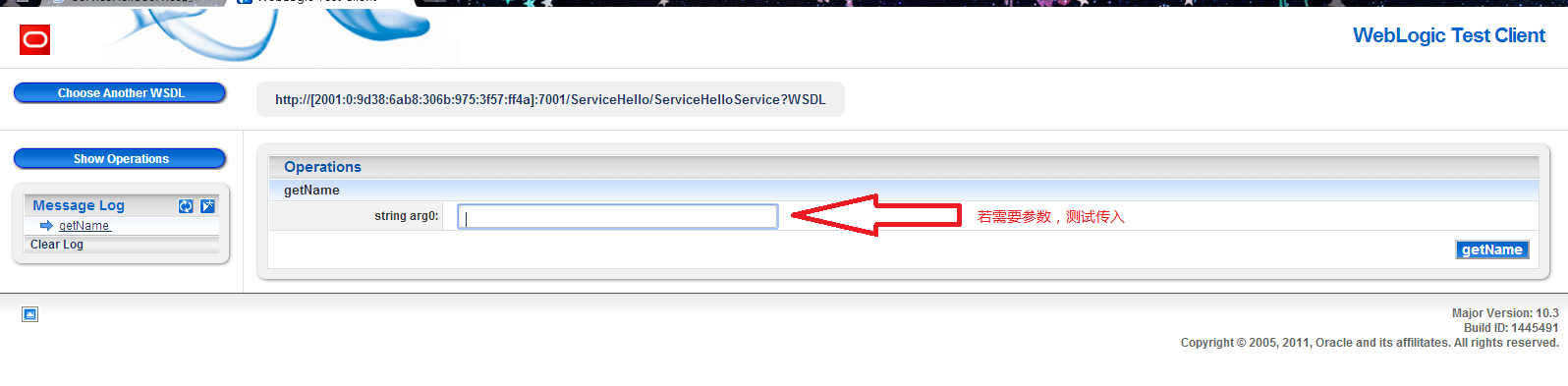
图5：

图6：分支（进入测试部分）

若需要参数，传入：



执行结果：

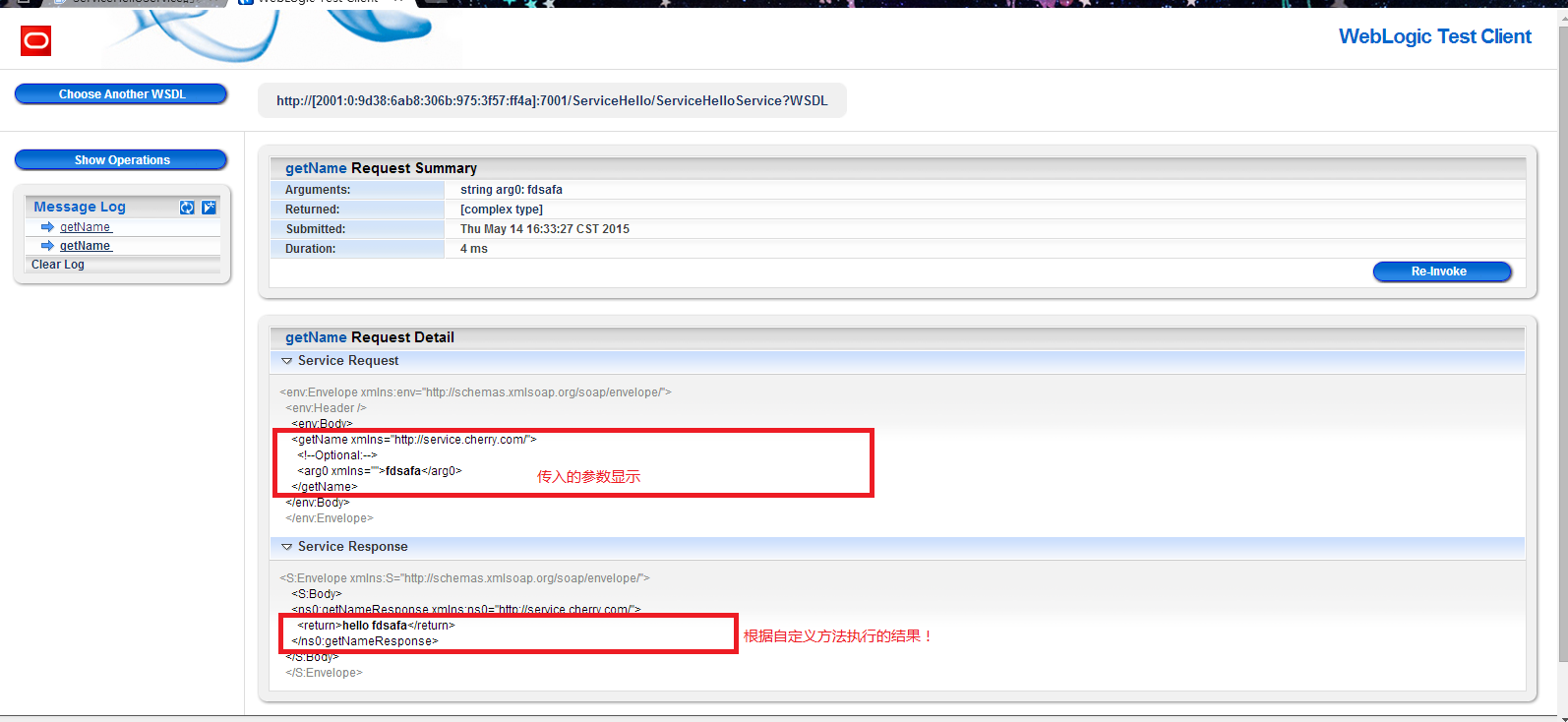
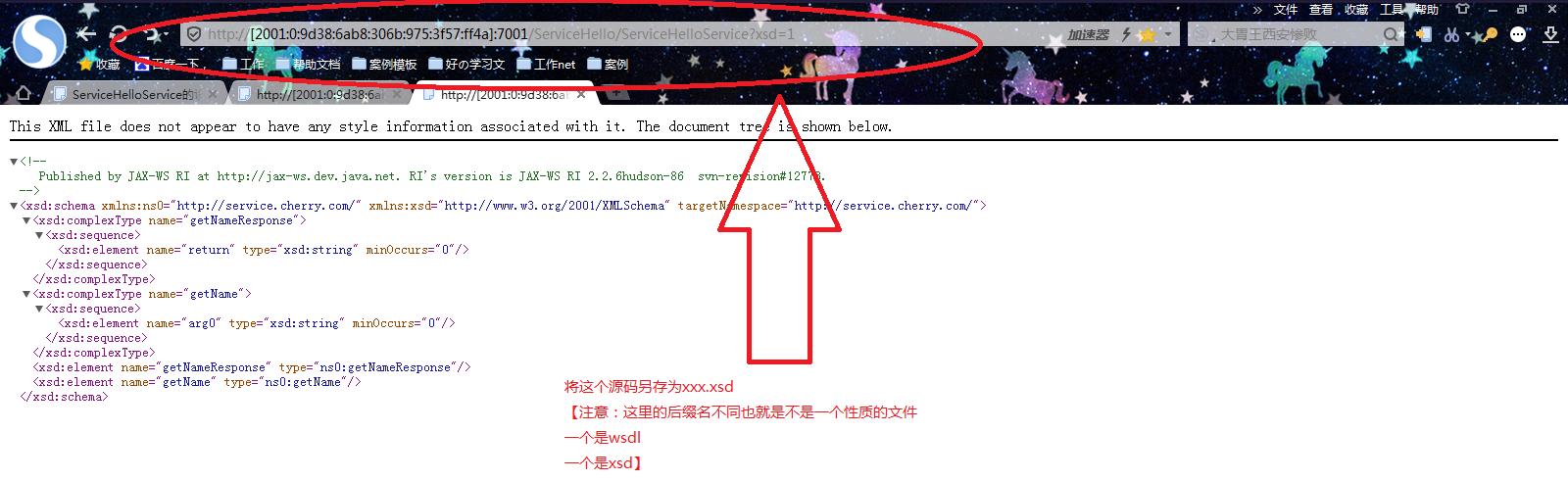


图7：分支（需要的client代码）

图8：代码获取：



获取另一份wsdl



【获取client文件需要注意的】

步骤：

1. 一个工具
2. 一个指令
3. 两个固定的位置存放：工具和指令

获取client文件是为了生成在项目中com.xxx.client的代码

这里获取的两个文件

1. 一个是xxx.wsdl,一个是xxx.xsd文件
2. 获取之后需要生成对应的项目源码，这里需要一个工具（一个技术）：【apache-cxf-2.7.11】

建议：

在一个指定的位置安装apache-cxf-2.7.11这个文件

在一个指定的位置安装所有的生成client源码，建议的文件名称：【wsdl2java】

在里面需要用到一行dos命令，可以写成一个dos的文件方便以后执行，执行之后就可以让这个工具自动生成项目所需要的源码

【指令】

Eg:

wsdl2java -encoding utf-8 -d E:\Software\wsdl2java\ -p com.cherry.client -client a.wsdl

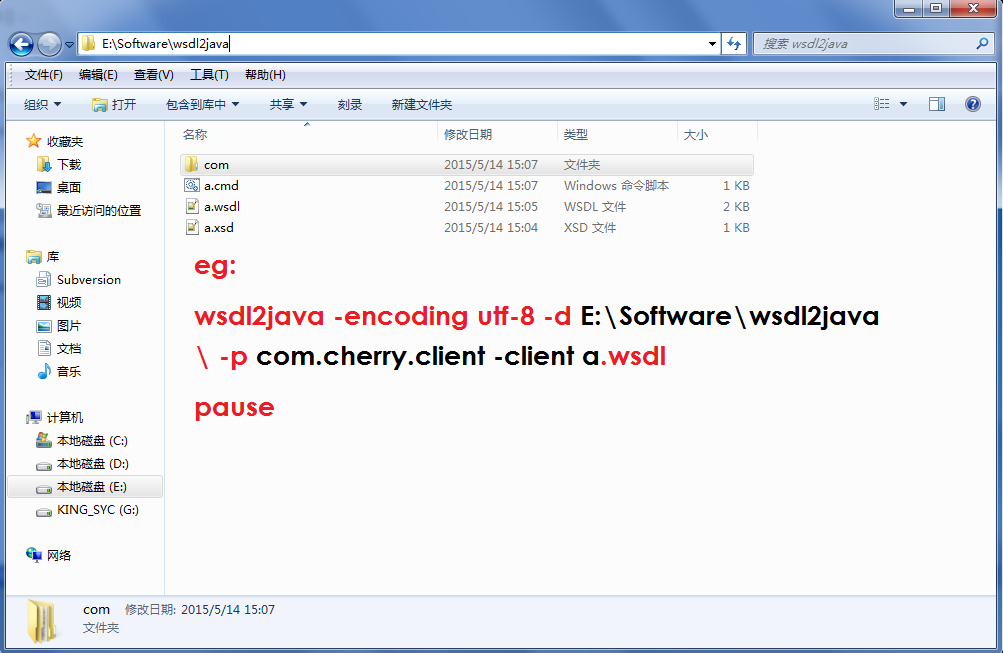
pause

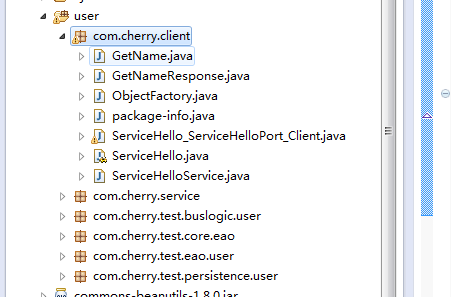
解析：

wsdl2java -encoding utf-8 -d 存放工具的路径 -p 最好是和项目的com.xx.client相同的名称，方便直接拷入项目 xxx（从浏览器另存为的文件名）.wsdl

pause

eg：



然后拷入项目

【在业务层Biz（buslogic中获取端口PROT,写测试）】

在buslogic中写（biz层写）一个实现类

基本固定代码：

其中黄色部分是@WebService注解的那个类(一开始的提供服务的那个类)

public static void main(String[] args) {

FACTORY.setAddress("http://localhost:7001/ServiceHello/ServiceHelloService");

FACTORY.setServiceClass(ServiceHello.class);//

FACTORY.getInInterceptors().add(new LoggingInInterceptor());

FACTORY.getOutInterceptors().add(new LoggingOutInterceptor());

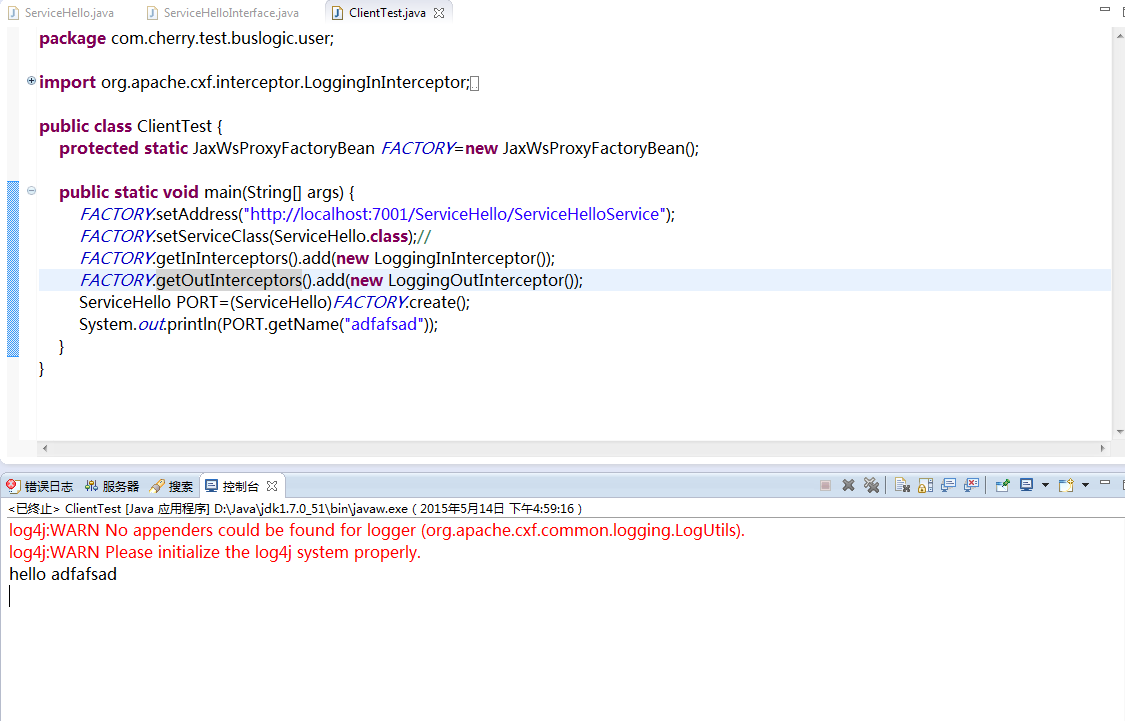
ServiceHello PORT=(ServiceHello)FACTORY.create();

//随便传入一个参数调用方法

System.out.println(PORT.getName("adfafsad"));

}

运行结果：



# 作为客户端

【作为客户端】

1. 获取端口PROT
2. 调用对方的方法/传入参数测试

【在业务层Biz（buslogic中获取端口PROT,写测试）】

在buslogic中写（biz层写）一个实现类

基本固定代码：

其中黄色部分是@WebService注解的那个类(一开始的提供服务的那个类)

public static void main(String[] args) {

FACTORY.setAddress("http://localhost:7001/ServiceHello/ServiceHelloService");

FACTORY.setServiceClass(ServiceHello.class);//

FACTORY.getInInterceptors().add(new LoggingInInterceptor());

FACTORY.getOutInterceptors().add(new LoggingOutInterceptor());

ServiceHello PORT=(ServiceHello)FACTORY.create();

//随便传入一个参数调用方法

System.out.println(PORT.getName("adfafsad"));

}

运行结果：

