# word

# 1 Webservice 概述&应用场景

## 1.1 简介

* Web service 即web服务，它是一种跨编程语言和跨操作系统平台的远程调用技术**即跨平台远程调用技术**。
* 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**，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**。

## 1.2 应用场景

* 访问天气预报WebService地址
* http://www.webxml.com.cn/WebServices/WeatherWebService.asmx

|  |
| --- |
|  |

* 演示getSupportCity中获取广东的所有城市

|  |
| --- |
|  |

|  |
| --- |
|  |

# 2 Webservice 的标准和实现方式

## 2.1 标准

* 采用标准**SOAP(Simple Object Access Protocol) 协议传输**，soap属于w3c标准。**Soap协议是基于http的应用层协议，soap协议传输是xml数据**。
* soap协议就是定义xml的标签格式

## 2.2 方式

* 采用**wsdl(Web Services Description Language)作为描述语言**【即webservice接口使用说明书】wsdl属w3c标准。
* **xml是webservice的跨平台的基础**，XML主要的优点在于它既与平台无关，又与厂商无关。
* XSD，W3C为webservice制定了一套**传输数据类型**，使用xml进行描述，即XSD(XML Schema Datatypes)，任何编程语言写的webservice接口在发送数据时都要转换成webservice标准的XSD发送。
* **WebService数据传输的方式**

1. SOAP:http+xml
2. Http rest :一套用来创建Web Service的方法,REST式的Web Service使用HTTP里的方法：GET，POST，DELETE， PUT
3. **http自定义数据协议:**比如http传输json数据，http传输xml数据等

## 2.3 WebService 三要素

### soap

* SOAP即简单对象访问协议(Simple Object Access Protocal) 是一种**简单的基于 XML 的协议，它使应用程序通过 HTTP 来交换信息**.
* 简单理解为**soap=http+xml**,就是通过HTTP发xml数据
* Soap协议版本主要使用**soap1.1、soap1.2。**
* SOAP不是webservice的专有协议，其他应用协议也使用soap传输数据。例如，SMTP。

### Wsdl

WSDL 是基于 XML 的用于描述Web Service及其函数、参数和返回值。**通俗理解Wsdl是webservice的使用说明书**

### Uddi【了解】

* UDDI 是一种目录服务，通过它，企业可注册并搜索 Web services。企业将自己提供的Web Service注册在UDDI，也可以使用别的企业在UDDI注册的web service服务，从而达到资源共享。
* UDDI旨在将**全球的webservcie资源进行共享，促进全球经济合作**。
* 目前大部分企业使用webservice并不是必须使用UDDI，因为用户通过WSDL知道了web service的地址，可以直接通过WSDL调用webservice

# 3 Webservice 开发规范

JAVA 中共有三种 实现WebService 开发规范**JAX-WS（JAX-RPC）、JAXM & SAAJ、JAX-RS**

## 3.1 JAX-WS 概述

**JAX-WS**  的全称为 **Java API for XML-Based Webservices** ，早期的基于SOAP 的JAVA 的Web 服务规范**JAX-RPC（Java API For XML-Remote Procedure Call）**目前已经被JAX-WS 规范取代。

* **从jdk1.5开始支持JAX-WS2.0版本，**
* **Jdk1.6.0\_13以后的版本支持2.1版本，**
* **jdk1.7支持2.2版本。**
* 也就是说java提供了一个jar包来实现webservice

## 3.2 JAXM&SAAJ概述

* **JAXM（JAVA API For XML Message）**主要定义了包含了发送和接收消息所需的API
* **SAAJ（SOAP With Attachment API For Java，JSR 67）**是与JAXM 搭配使用的API，为构建SOAP 包和解析SOAP 包提供了重要的支持，**支持附件传输等**
* **JAXM & SAAJ** 与**JAX-WS** 都是基于SOAP 的Web 服务
* 相比之下**JAXM&SAAJ** 暴漏了SOAP更多的底层细节，编码比较麻烦
* **而JAX-WS 更加抽象，隐藏了更多的细节，更加面向对象，实现起来你基本上不需要关心SOAP 的任何细节【开发者更喜欢这种方式开发webservice】**

## 3.3 JAX-RS 概述

JAX-RS 是JAVA 针对**REST(Representation State Transfer)风格制定的一套Web 服务规范**

由于推出的较晚，该规范（JSR 311，目前JAX-RS 的版本为1.0）并未随JDK1.6 一起发行。

# 4 JAX-WS

## 4.1 案例：Webservice第一个入门程序

需求:提供一个天气的WebService服务器接口，以查询天气为例(画图)

### 步骤

### 第一步：

创建一个01\_jax-ws-server java项目创建一个SEI(Service Endpoint Interface)接口，本质就是一个Weather接口

|  |
| --- |
|  |

### 第二步：

实现天气接口，在类上添加**@Webservice**注解（先不加，后面再加）

|  |
| --- |
|  |

### 第三步：

发布:使用Endpoint的publish发布一个WebService服务

|  |
| --- |
|  |

### 第四步：

测试Webservice:在浏览器中访问http://127.0.0.1:12345/weather?wsdl

|  |
| --- |
|  |

## 4.2 wsdl说明书结构

* <service>服务视图，webservice的服务结点，它包括了服务端点
* <binding>为每个服务端点定义消息格式和协议细节
* <portType>服务端点，描述 web service可被执行的操作方法，以及相关的消息，通过binding指向portType
* <message>定义一个操作（方法）的数据参数(可有多个参数)
* <types> 定义 web service 使用的全部数据类型

|  |
| --- |
|  |

## 4.3 wsdl 说明书阅读方式

**从下往上读**,先找到服务视图，通过binging找到protType，找到了protType就找到了我们要调用的webservice方法。

## 4.4 Wsimport命令

* **wsimport是jdk自带的webservice客户端工具**,可以根据wsdl文档生成客户端调用代码(java代码).当然,无论服务器端的WebService是用什么语言写的,都可以生成调用webservice的客户端代码，服务端通过客户端代码调用webservice。
* wsimport.exe**位于JAVA\_HOME\bin**目录下.
* 常用参数为:

-s<目录> - 将生成.java文件。（学习）

-d<目录> - 将生成.class文件。默认参数。

-p<生成的新包名> -将生成的类，**放于指定的包下**。

(wsdlurl) - http://server:port/service?wsdl，必须的参数。

* 示例：

C:/> wsimport –s . http://127.0.0.1:1234/weather?wsdl

**注意：-s不能分开，-s后面有个小点**

## 4.5案例:通过Wsimport生成客户端调用类

* 在Eclipse中创建一个空的java工程为**02.jax\_ws\_client**，此工程作为存放客户端代码。

|  |
| --- |
|  |

* cmd命令行进入此wsimport 工程的src目录，
* 输入以下命令:

**wsimport** –s . http://127.0.0.1:12345/weather?wsdl

|  |
| --- |
|  |

**注:**

**可以通过java –version检查你当前的版本号保证是jdk1.6以上。**

**Jdk1.6.0\_13以后的版本支持jaxws2.1。**

## 4.6案例:WebService 客户端编写

在上面的项目代码中

* 创建服务窗口
* 通过服务对象找到服务端点
* 调用webservice的方法

|  |
| --- |
|  |

## 4.7两个应用程序业务层原理分析

|  |
| --- |
|  |

## 4.8 TCP/IP Monitor 监控SOAP网络请求

* 使用**TCP/IP Monitor可以监视tcp/ip协议的报文内容**，由于http是基于Tcp的应用协议，而webservice是基于http实现，所以通过tcp/ip monitor可以监视webservice请求及响应的内容。
* 案例:使用Eclipse的TCP/IP Monitor监控Soap协议网络请求
* 讲解代理服务器
* 步骤

1. 点击Window -> show View -> other -> 搜索TCP/IP Monitor
2. 点击TCP/IP Monitor 右边小三角，点击Properties(左图)
3. 设置代理端口、监听IP、监听端口，点OK(中图)
4. 然后选中后，点击start开启监听(右图)
5. 可以在浏览器和客户端访问WebService(**使用客户端是要更改实现里面的端口为代理端口**)

|  |
| --- |
|  |
|  |
|  |

## Soap1.1协议网络请求数据格式分析

Soap是基本http请求，他的请求和响应头的响应类型都是text/xml

一般的请求，请求类型是text/html

|  |
| --- |
|  |

**注:JAX-WS 这里使用的2.2.9版本, jdk1.7支持2.2版本**

## 4.9 Soap1.1和soap1.2 的实现

* SOAP协议有两个版本，1.1和1.2
* 如何发布SOAP1.2服务端（掌握）

在实现类中加入如下注解@BindingType(SOAPBinding.SOAP12HTTP\_BINDING)

|  |
| --- |
|  |

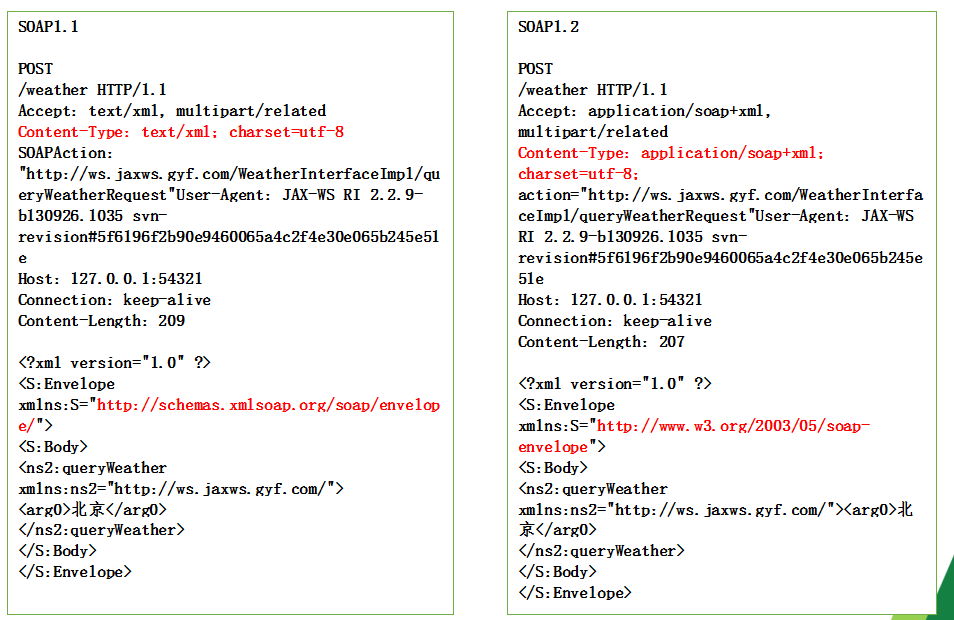
* **如果Jaxws不支持SOAP1.2服务端发布,说明JDK版本较低在1.7以下，可在服务端引入第三方JAR（jaxws-ri-2.2.8）(网上下载即可)**

|  |
| --- |
| 1.7会出现不支持1.2错误 |

* **但现在1.8jdk已经支持soap1.2,可以不用加第三方jar包**

## 4.10 Soap1.1和soap1.2 的请求头区别

请求头的contentType和S:Envelope的命令空间不一样



## 4.11 Soap1.1和soap1.2 的区别

|  |
| --- |
| * **相同点：**   请求发送方式相同：都是使用POST  协议内容相同：都有Envelope和Body标签   * **不同点：**   数据格式不同：content-type不同  SOAP1.1：text/xml;charset=utf-8  SOAP1.2：application/soap+xml;charset=utf-8   * **命名空间不同**：   SOAP1.1：http://schemas.xmlsoap.org/soap/envelope  SOAP1.2：http://www.w3.org/2003/05/soap-envelope |

## 4.12 webservice优点&缺点

**优点**

1、采用xml支持跨平台远程调用。

2、基于http的soap协议，可跨越防火墙。

3、支持面向对象开发。

4、有利于软件和数据重用，实现松耦合。

**缺点**

由于soap是基于**xml传输**，本身使用xml传输会传输一些**无关的东西从而效率不高**，随着soap协议的完善，soap协议增加了许多内容，这样就导致了使用soap协议进行数据传输的效率不高。

# WebService调用的几种方式

第一种:客户端方式wsimport 【掌握】

案例:手机号码归属地查询WEB服务

案例:天气查询WEB服务

第二种:Service方式 【了解】

案例:天气查询WEB服务

第三种:HTTPURLConnection方式【代码自己发送http请求-掌握】

案例:手机号码归属地查询WEB服务

第四种:Ajax调用方式【不要求掌握,原理与第三种一样,只不过是用js写】

....

注意：天气预报的webservice有两个，案例演示，自己选择

http://ws.webxml.com.cn/WebServices/WeatherWS.asmx

包含0230个以上中国城市和100个以上国外城市天气预报数据

http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx

**包括 340 多个中国主要城市和 60 多个国外主要城市三日内的天气预报数据。**

## 5.1 第一种:客户端方式【推荐使用】

### 案例：手机归属地查询

公网服务地址http://www.webxml.com.cn/zh\_cn/index.aspx

手机归属地Web服务 http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl

步骤

第一步：wsimport生成客户端代码

|  |
| --- |
| **wsimport -p com.gyf.mobile.ws -s . http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx?wsdl** |

第二步：阅读使用说明书，使用生成客户端代码调用服务端

第三步: 获取指定电话号码的归属地信息,代码如下

|  |
| --- |
|  |

### 案例：天气查询WEB服务

公网服务地址http://www.webxml.com.cn/zh\_cn/index.aspx

天气Web服务 http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl

步骤

第一步：wsimport生成客户端代码**【此命令放在你在工程src目录下执行】**

|  |
| --- |
| **wsimport -p com.gyf.weather.ws -s . http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl** |

**使用上面的命令时会出现失败**，出现『则很可能需要更改 ‘s:schema’ 的前缀』错误

|  |
| --- |
|  |

解决办法是，将wsdl另存为一个xml文件，然后打开文件，把s:schema内容删除

 是把整个s:element标签删除

在终端执行下面命令**wsimport -p com.gyf.weather.ws -s . +xml文件路径**

如果是window,路径写 file://c:/…

如果是linux/mac 直接写 根路径/…

|  |
| --- |
|  |

第二步：阅读使用说明书，使用生成客户端代码调用服务端

第三步: 根据城市获取天气信息

|  |
| --- |
|  |

## 5.2 第二种:Service方式【了解】

### 案例:天气查询WEB服务

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  //说明书地址  URL url = **new** URL("http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl");  //String namespaceURI 命名空间  //String localPart 服务名称  QName qName = **new** QName("http://WebXml.com.cn/","WeatherWebService");  //创建服务  Service service = Service.*create*(url,qName);  //获取服务端点  WeatherWebServiceSoap soap = service.getPort(WeatherWebServiceSoap.**class**);    **for**(String item : soap.getSupportProvince().getString()){  System.***out***.println(item);  }  } |

## 5.3 第三种:HTTPURLConnection方式

### 案例:手机号码归属地查询WEB服务

根据http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx?op=getMobileCodeInfo说明的soap1.1请求格式来实现即可

使用java.net.HttpURLConnection发送POST网络请求

注意事项

设置post请求方式时，post一定要大写

默认HttpURLConnection是没有权限写数据，需要设置setDoOutput(true)

|  |
| --- |
| **public** **class** Demo2 {  **public** **static** **void** main(String[] args) **throws** Exception {  //创建url  URL url = **new** URL("http://ws.webxml.com.cn/WebServices/MobileCodeWS.asmx");  HttpURLConnection conn = (HttpURLConnection) url.openConnection();  conn.setRequestMethod("POST");  //设置可以写请求数据  conn.setDoOutput(**true**);    //设置请求头  conn.setRequestProperty("Content-Type", "text/xml; charset=utf-8");      OutputStream os = conn.getOutputStream();  os.write(*getXML*("13633123359").getBytes());    **int** code = conn.getResponseCode();  System.***out***.println("响应状态码:" + code);    **if**(code == 200){  //读取响应数据  InputStream is = conn.getInputStream();  InputStreamReader reader = **new** InputStreamReader(is,"UTF-8");  BufferedReader br = **new** BufferedReader(reader);  String line = **null**;  **while**((line = br.readLine()) != **null**){  System.***out***.println(line);  }  br.close();  }  }    **public** **static** String getXML(String tel){  **return** "<?xml version=\"1.0\" encoding=\"utf-8\"?>"+  "<soap:Envelope xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\" xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\"" +  " xmlns:soap=\"http://schemas.xmlsoap.org/soap/envelope/\">" +  "<soap:Body>" +  "<getMobileCodeInfo xmlns=\"http://WebXml.com.cn/\">" +  "<mobileCode>" + tel + "</mobileCode>" +  "<userID></userID>" +  "</getMobileCodeInfo>" +  "</soap:Body>" +  "</soap:Envelope>";  }  } |

### 案例:天气查询WEB服务

# 注解修改WSDL内容

|  |
| --- |
|  |

## 案例：

|  |
| --- |
|  |

# 开发中WebService的选择

## 适用场景

发布一个服务（对内/对外），不考虑客户端类型，不考虑性能，建议使用webservice

服务端已经确定使用webservice，客户端不能选择，必须使用webservice

## 不适用场景

考虑性能时不建议使用webservice

同构程序下不建议使用webservice，比如java 用RMI，不需要翻译成XML的数据

## RMI概念

RMI（即Remote Method Invoke 远程方法调用）。在Java中，只要一个类extends了java.rmi.Remote接口，即可成为存在于服务器端的远程对象，供客户端访问并提供一定的服务。