讲师：安涛，手机号：18201597710（微信）；邮箱：

at_1000@126.com

Webservice

Webservice 就是一种远程调用技术，他的作用就是从远程系统中获取业务数据

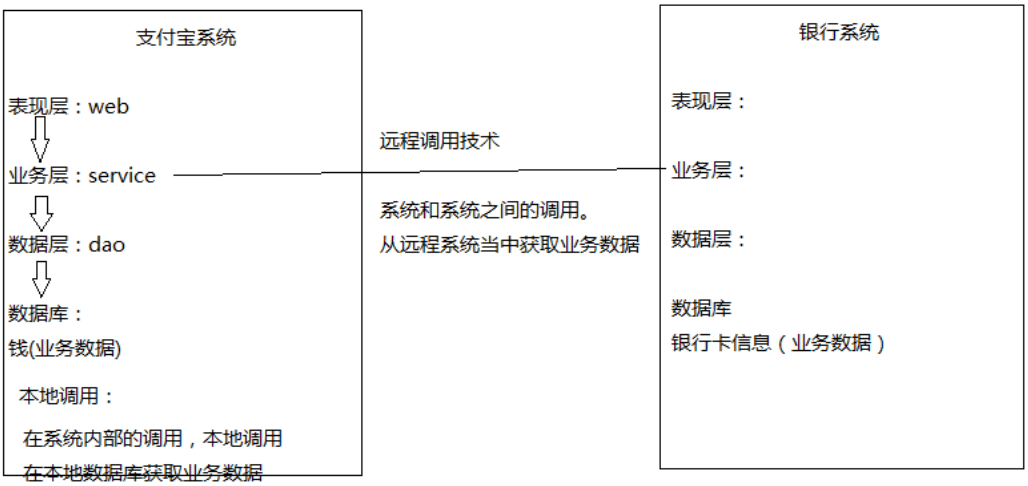
1 课程安排

- 什么是 webservice
- Webservice 入门程序
- Webservice 的应用场景
- Webservice 的三要素
  - WSDL：web 服务描述语言
  - SOAP：简单对象访问协议
  - UDDI：目录服务
- Webservice 的四种客户端调用方式
  - 生成客户端调用方式
  - 客户端编程调用方式
  - HttpURLConnecton 调用方式
  - Ajax 调用方式
- 深入开发：用注解修改 WSDL 内容

2 什么是 webservice

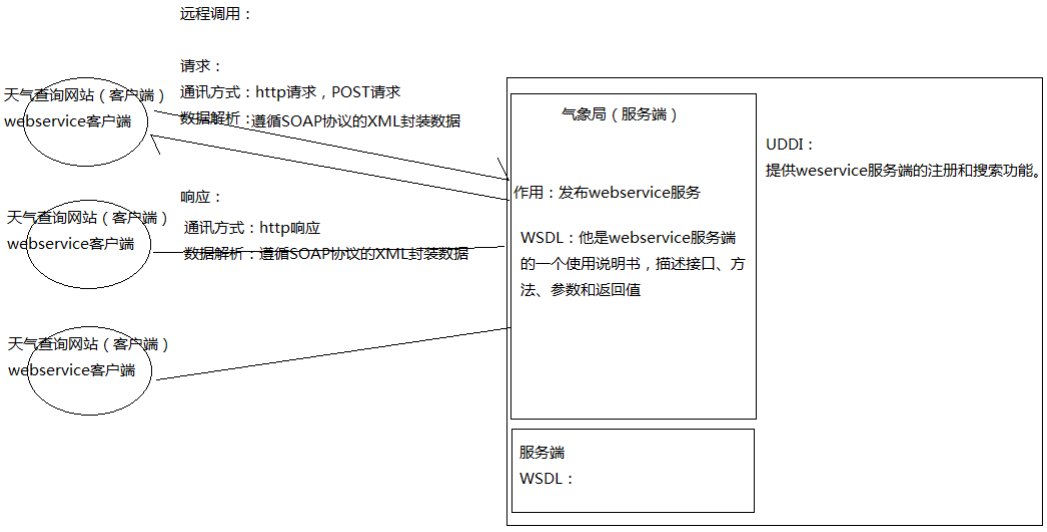
2.1 什么是远程调用技术

远程调用数据定义：是系统和系统之间的调用



2.2 Webservice 的原理图

- Webservice 是使用 Http 发送 SOAP 协议的数据的一种远程调用技术
- Webservice 要开发服务端
- Webservice 要开发客户端
- Webservice 客户端开发需要阅读服务端的使用说明书（WSDL）



3 Webservice 的入门程序

3.1 需求

- 服务端：发布一个天气查询服务，接收客户端城市名，返回天气数据给客户端
- 客户端：发送城市名称给服务端，接收服务端的返回天气数据，打印

3.2 环境

- JDK：1.7
- Eclipse：mars

3.3 实现

3.3.1 服务端：

开发步骤：

- 第一步：创建 SEI（Service Endpoint Interface）接口，本质上就是 Java 接口

```
package cn.itcast.ws.jaxws.ws;

/**
 *
 * <p>Title: WeatherInterface.java</p>
 * <p>Description:SEI接口</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日上午9:28:00
 * @version 1.0
 */
public interface WeatherInterface {

    public String queryWeather(String cityName);

}
```

- 第二步：创建 SEI 实现类，在实现类上加入@WebService

```
package cn.itcast.ws.jaxws.ws;
```

```java
import javax.jws.WebService;

/**
 *
 * <p>Title: WeatherInterfaceImpl.java</p>
 * <p>Description:SEI实现类</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日上午9:29:27
 * @version 1.0
 */
@WebService//@WebService表示该类是一个服务类，需要发布其中的public的方法
public class WeatherInterfaceImpl implements WeatherInterface {

    @Override
    public String queryWeather(String cityName) {
        System.out.println("from client..."+cityName);
        String weather = "晴";
        return weather;
    }

}
```

- 第三步：发布服务，Endpoint 发布服务，publish 方法，两个参数：1.服务地址；2.服务实现类

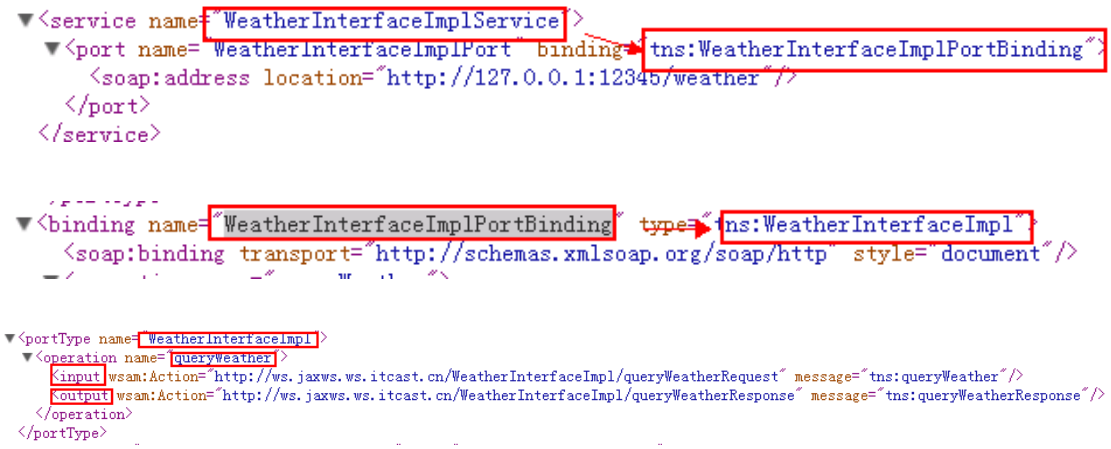
```java
package cn.itcast.ws.jaxws.ws;

import javax.xml.ws.Endpoint;

/**
 *
 * <p>Title: WeatherServer.java</p>
 * <p>Description:天气服务端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日上午9:41:20
 * @version 1.0
 */
public class WeatherServer {
```

```
    public static void main(String[] args) {
        //Endpoint发布服务
        //参数解释
        //1.address - 服务地址
        //2.implementor - 实现类
        Endpoint.publish("http://127.0.0.1:12345/weather", new
WeatherInterfaceImpl());
    }
}
```

- 第四步：测试服务是否发布成功，通过阅读使用说明书，确定客户端调用的接口、方法、参数和返回值存在，证明服务发布成功。
  - WSDL 地址：服务地址+"?wsdl"
  - WSDL 阅读方式：从下往上



3.3.2 客户端：

开发步骤

- 第一步：wsimport 命令生成客户端代码

wsimport -s . http://127.0.0.1:12345/weather?wsdl

- 第二步：根据使用说明书，使用客户端代码调用服务端
  - 第一步：创建服务视图，视图是从 service 标签的 name 属性获取
  - 第二步：获取服务实现类，实现类从 portType 的 name 属性获取
  - 第三步：获取查询方法，从 portType 的 operation 标签获取

```
package cn.itcast.ws.jaxws.ws.client;
```

```java
import cn.itcast.ws.jaxws.ws.WeatherInterfaceImpl;
import cn.itcast.ws.jaxws.ws.WeatherInterfaceImplService;

/**
 *
 * <p>Title: WeatherClient.java</p>
 * <p>Description:天气查询客户端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日上午9:57:40
 * @version 1.0
 */
public class WeatherClient {

    public static void main(String[] args) {
        //创建服务视图
        WeatherInterfaceImplService
weatherInterfaceImplService = new
WeatherInterfaceImplService();
        //获取服务实现类
        WeatherInterfaceImpl weatherInterfaceImpl =
weatherInterfaceImplService.getPort(WeatherInterfaceImpl.clas
s);
        //调用查询方法，打印
        String weather = weatherInterfaceImpl.queryWeather("北
京");

        System.out.println(weather);
    }
}
```

3.4 Webservice 的优缺点

优点：

- 发送方式采用 http 的 post 发送，http 的默认端口是 80，防火墙默认不拦截 80，所以跨防火墙
- 采用 XML 格式封装数据，XML 是跨平台的，所以 webservice 也可以跨平台。
- Webservice 支持面向对象

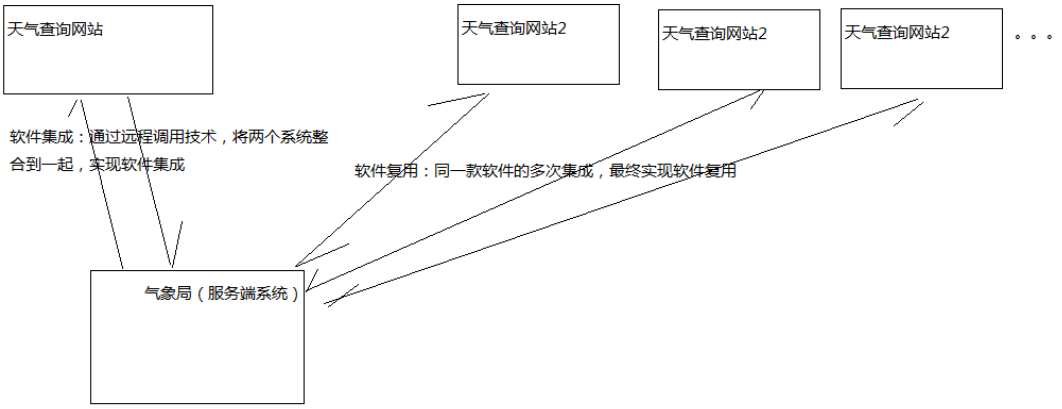
缺点：

- 采用 XML 格式封装数据，所以在传输过程中，要传输额外的标签，随着 SOAP 协议的不

断完善，标签越来越大，导致 webservice 性能下降

4 Webservice 应用场景

4.1 软件集成和复用



4.2 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

4.3 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

5 WSDL

5.1 定义

WSDL 及 web 服务描述语言，他是 webservice 服务端使用说明书，说明服务端接口、方法、参数和返回值，WSDL 是随服务发布成功，自动生成，无需编写

5.2 文档结构

- <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

5.3 阅读方式：从下往上

6 SOAP

6.1 定义：

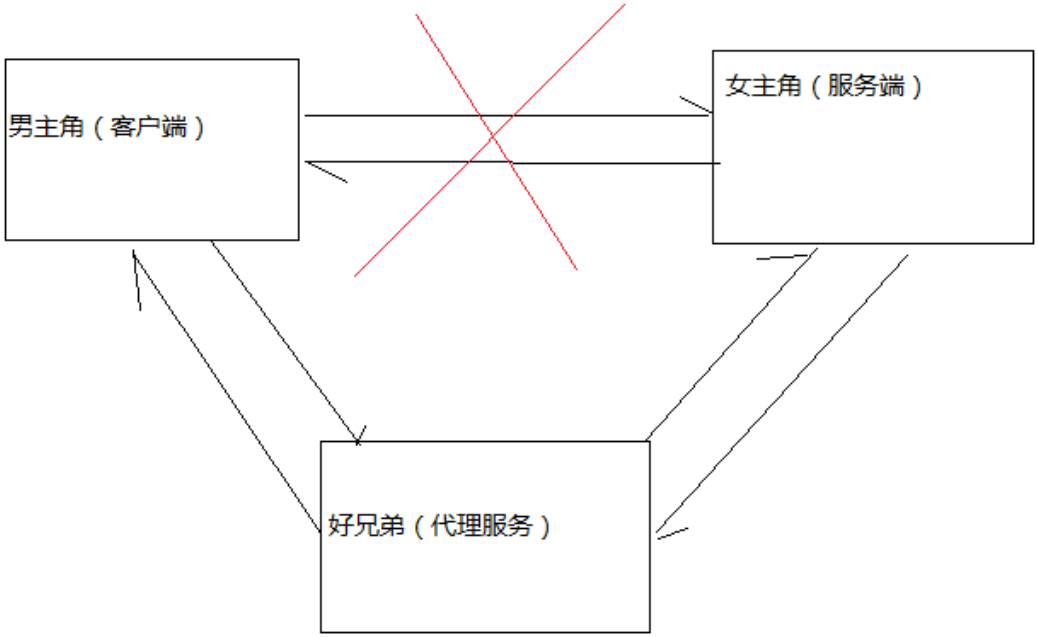
- SOAP 即简单对象访问协议，他是使用 http 发送的 XML 格式的数据，它可以跨平台，跨防火墙，SOAP 不是 webservice 的专有协议。
- SOAP=http+xml

6.2 协议格式

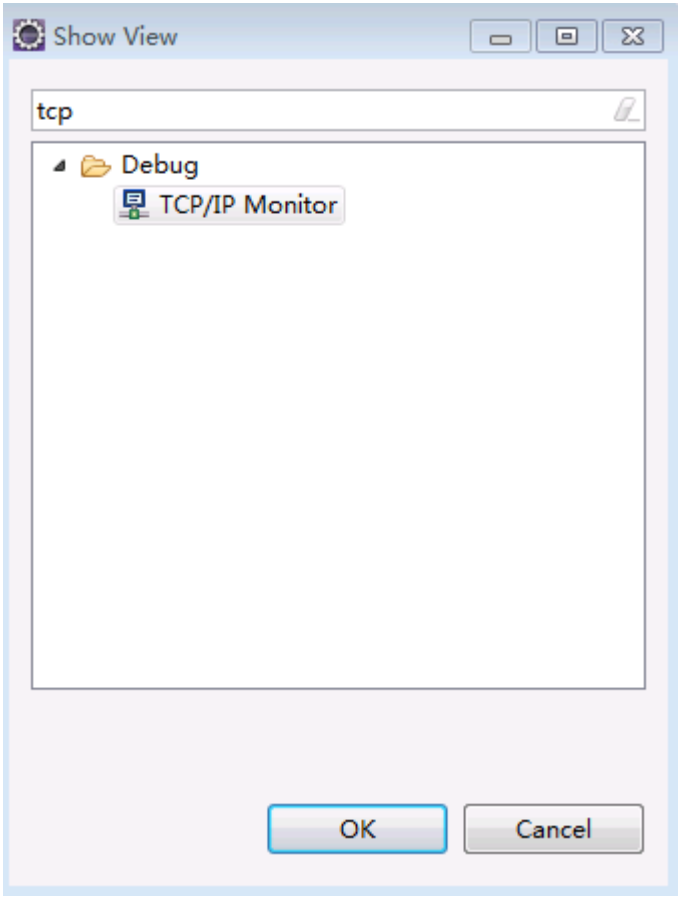
- 必需有 Envelope 元素，此元素将整个 XML 文档标识为一条 SOAP 消息
- 可选的 Header 元素，包含头部信息
- 必需有 Body 元素，包含所有的调用和响应信息
- 可选的 Fault 元素，提供有关在处理此消息所发生错误的信息

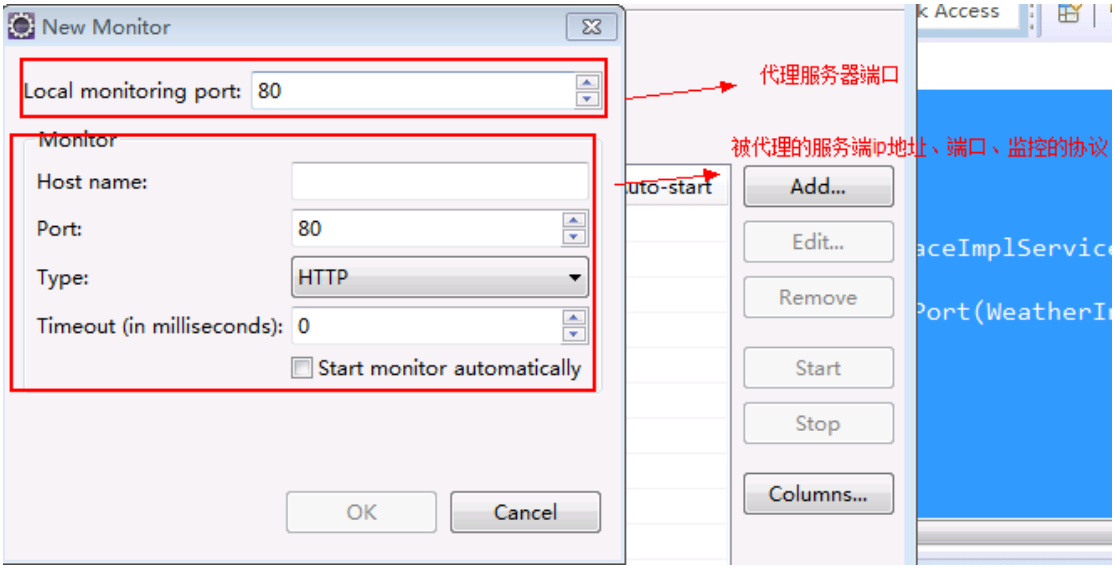
6.3 TCP/IP Monitor

6.3.1 代理原理


男主角（客户端）
女主角（服务端）
好兄弟（代理服务）

6.3.2 配置


Show View
tcp
Debug
TCP/IP Monitor
OK
Cancel



6.3.3 测试

在浏览器中输入代理服务地址，能正常访问，代表代理服务器设置成功

6.4 SOAP1.1

请求

```
POST /weather HTTP/1.1
Accept: text/xml, multipart/related
Content-Type: text/xml; charset=utf-8
SOAPAction: "http://ws.jaxws.ws.itcast.cn/WeatherInterfaceImpl/queryWeatherRequest"
User-Agent: JAX-WS RI 2.2.4-b01
Host: 127.0.0.1:54321
Connection: keep-alive
Content-Length: 214

<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body><ns2:queryWeather xmlns:ns2="http://ws.jaxws.ws.itcast.cn/"><arg0> 北 京
</arg0></ns2:queryWeather>
</S:Body>
</S:Envelope>
```

响应

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: text/xml; charset=utf-8
Date: Thu, 26 Nov 2015 03:14:29 GMT

<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
<ns2:queryWeatherResponse xmlns:ns2="http://ws.jaxws.ws.itcast.cn/"><return> 晴
</return></ns2:queryWeatherResponse>
</S:Body>
</S:Envelope>
```

6.5 SOAP1.2

- 如何发布 SOAP1.2 服务端
  - Jaxws 不支持 SOAP1.2 服务端发布，直接发布会报如下异常

```
in thread "main" com.sun.xml.internal.ws.server.ServerRtException: Cannot generate WSDL for binding "http://www.w3.org/2003/05/soap/binding
t com.sun.xml.internal.ws.server.EndpointFactory.generateWSDL(EndpointFactory.java:421)
t com.sun.xml.internal.ws.server.EndpointFactory.createEndpoint(EndpointFactory.java:198)
t com.sun.xml.internal.ws.api.server.WSEndpoint.create(WSEndpoint.java:498)
t com.sun.xml.internal.ws.transport.http.server.EndpointImpl.createEndpoint(EndpointImpl.java:246)
t com.sun.xml.internal.ws.transport.http.server.EndpointImpl.publish(EndpointImpl.java:170)
t com.sun.xml.internal.ws.spi.ProviderImpl.createAndPublishEndpoint(ProviderImpl.java:118)
t javax.xml.ws.Endpoint.publish(Endpoint.java:240)
t cn.itcast.ws.jaxws.ws.WeatherServer.main(WeatherServer.java:21)
```

  - 如果想发布 SOAP1.2 服务端，需要在服务端引入第三方 JAR（jaxws-ri-2.2.8）
  - 在实现类上加入如下注解

```
@BindingType(SOAPBinding.SOAP12HTTP_BINDING)
```

请求：

```
POST /weather HTTP/1.1
Accept: application/soap+xml, multipart/related
Content-Type: application/soap+xml; charset=utf-8;
action="http://ws.jaxws.ws.itcast.cn/WeatherInterfaceImpl/queryWeatherRequest"
User-Agent: JAX-WS RI 2.2.4-b01
Host: 127.0.0.1:54321
Connection: keep-alive
Content-Length: 212

<?xml version="1.0" ?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
<S:Body><ns2:queryWeather xmlns:ns2="http://ws.jaxws.ws.itcast.cn/"><arg0> 北 京
</arg0></ns2:queryWeather>
```

```
</S:Body>
</S:Envelope>
```

响应

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: application/soap+xml; charset=utf-8
Date: Thu, 26 Nov 2015 03:25:24 GMT

<?xml version='1.0' encoding='UTF-8'?>
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope">
<S:Body>
<ns2:queryWeatherResponse xmlns:ns2="http://ws.jaxws.ws.itcast.cn/"><return> 晴
</return></ns2:queryWeatherResponse>
</S:Body>
</S:Envelope>
```

6.6 SOAP1.1 和 SOAP1.2 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

7 UDDI

UDDI 是一种目录服务，企业可以使用它对 Web services 进行注册和搜索。UDDI，英文为 "Universal Description, Discovery and Integration"，可译为“通用描述、发现与集成服务”。

UDDI 并不像 WSDL 和 SOAP 一样深入人心，因为很多时候，使用者知道 Web 服务的位置（通常位于公司的企业内部网中）。

8 课程回顾

- 什么是 webservice
  - 什么是远程调用，系统和系统之间的调用，从远程系统当中获取业务数据。
  - Webservice 是 web 服务，他是用 http 传输 SOAP 协议数据的一种远程调用技术
- Webservice 入门程序
  - 服务端
    - 第一步：创建 SEI 接口
    - 第二步：创建 SEI 实现类，要在类上加入@WebService
    - 第三步：发布服务，Endpoint 的 publish 方法，2 两个参数：1.服务地址；2.实现类实例
    - 第四步：测试服务是否发布成功，通过阅读使用说明书，确定服务接口、方法、参数、返回值存在，说明服务发布成功。
      - WSDL 地址：服务地址+"?wsdl"
      - WSDL 阅读方式，从下往上，servvice->binding->portType->其中有接口、方法、参数和返回值
  - 客户端
    - 第一步：使用 wsimport 生成客户端代码
    - 第二步：根据使用说明书，使用客户端调用服务端
      - 创建服务视图，视图是从 service 的 name 属性获取
      - 获取服务实现类，从 portType 的 name 属性获取
      - 调用查询方法，从 portType 下的 operation 标签的 name 属性获取
  - 优缺点：
    - 发送方式采用 http 的 post，http 默认端口是 80，所以跨越防火墙
    - 数据封装使用 XML 格式，XML 是跨平台，所以 webservice 可以跨平台
    - Webservice 支持面向对象开发
- Webservice 应用场景
  - 软件集成和复用
  - 适用场景：
    - 发布服务（对内/对外），不考虑性能，不考虑客户端类型，建议使用 webservice
    - 服务端已确定使用 webservice，客户端只能使用 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，比如客户端服务端都是 java 开发，建议 Java RMI
- WSDL
  - 定义：WSDL 即 Web 服务描述语言，他是 webservice 服务端的使用说明书，他说明服务端接口、方法、参数和返回值，他是随服务发布成功，自动生成，无需编写
  - 文档结构：
    - Service

        - Binding
        - portType
        - message
        - types
    - 阅读方式：从下往上

- SOAP
    - 定义：SOAP 即简单对象访问协议，他是使用 http 发送的 XML 格式的数据，跨平台、跨防火墙，他不是 webservice 的专有协议
    - SOAP=http+xml
    - 协议的格式：
        - 必须有：envelope 和 body
        - 非必有：header 和 fault
    - SOAP1.1 和 1.2 区别：
        - 相同点：
            - 都使用 http 的 POST 发送请求
            - 协议的格式都相同：都有 envelope 标签和 body 标签
        - 不同点：
            - Content-type：
              SOAP1.1：text/xml;charset=utf-8;SOAP1.2:application/soap+xml;charset=utf-8
            - 命名空间不同：

- UDDI：就是一个目录服务，提供搜索和注册功能，因为不常用，所以了解下就可以了。

9 Webservice 的四种客户端调用方式

公网服务地址：
http://www.webxml.com.cn/zh_cn/index.aspx

9.1 第一种生成客户端调用方式

9.1.1 Wsimport 命令介绍

- Wsimport 就是 jdk 提供的的一个工具，他作用就是根据 WSDL 地址生成客户端代码
- Wsimport 位置 JAVA_HOME/bin
- Wsimport 常用的参数：
    - -s，生成 java 文件的

- -d，生成 class 文件的，默认的参数
- -p，指定包名的，如果不加该参数，默认包名就是 wsdl 文档中的命名空间的倒序

- Wsimport 仅支持 SOAP1.1 客户端的生成

9.1.2 调用公网手机号归属地查询服务

- 第一步：wsimport 生成客户端代码

```
wsimport -p cn.itcast.mobile -s . http://webservice.we
bxml.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

```
package cn.itcast.mobile.client;

import cn.itcast.mobile.MobileCodeWS;
import cn.itcast.mobile.MobileCodeWSSoap;

/**
 *
 * <p>Title: MobileClient.java</p>
 * <p>Description:公网手机号查询客户端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日下午3:16:05
 * @version 1.0
 */
public class MobileClient {

    public static void main(String[] args) {
        //创建服务视图
        MobileCodeWS mobileCodeWS = new MobileCodeWS();
        //获取服务实现类
        MobileCodeWSSoap mobileCodeWSSoap =
mobileCodeWS.getPort(MobileCodeWSSoap.class);
        //调用查询方法
        String reuslt =
mobileCodeWSSoap.getMobileCodeInfo("13888888", null);
        System.out.println(reuslt);
    }
```

```
}
```

9.1.3 公网天气服务端查询

```
package cn.itcast.mobile.client;

import java.util.List;

import cn.itcast.weather.ArrayOfString;
import cn.itcast.weather.WeatherWS;
import cn.itcast.weather.WeatherWSSoap;

/**
 *
 * <p>Title: WeatherClient.java</p>
 * <p>Description:公网天气查询客户端</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日下午3:24:12
 * @version 1.0
 */
public class WeatherClient {

   public static void main(String[] args) {
      WeatherWS weatherWS = new WeatherWS();
      WeatherWSSoap weatherWSSoap =
weatherWS.getPort(WeatherWSSoap.class);
      ArrayOfString  arrayOfString =
weatherWSSoap.getWeather("北京", "");
      List<String> list = arrayOfString.getString();

      for(String str : list){
         System.out.println(str);
      }
   }
}
```

9.1.4 特点

该种方式使用简单，但一些关键的元素在代码生成时写死到生成代码中，不方便维护，所以仅用于测试。

10 第二种：service 编程调用方式

```
package cn.itcast.mobile.client;

import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import cn.itcast.mobile.MobileCodeWSSoap;

/**
 *
 * <p>Title: ServiceClient.java</p>
 * <p>Description:Service编程实现服务端调用</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日下午3:43:55
 * @version 1.0
 */
public class ServiceClient {

    public static void main(String[] args) throws IOException
{
        //创建WSDL的URL，注意不是服务地址
        URL url = new
URL("http://webservice.webxml.com.cn/WebServices/MobileCodeWS
.asmx?wsdl");

        //创建服务名称
        //1.namespaceURI - 命名空间地址
        //2.localPart - 服务视图名
        QName qname = new QName("http://WebXml.com.cn/",
"MobileCodeWS");
```

```
        //创建服务视图
        //参数解释:
        //1.wsdlDocumentLocation - wsdl地址
        //2.serviceName - 服务名称
        Service service = Service.create(url, qname);
        //获取服务实现类
        MobileCodeWSSoap mobileCodeWSSoap =
service.getPort(MobileCodeWSSoap.class);
        //调用查询方法
        String result =
mobileCodeWSSoap.getMobileCodeInfo("1866666666", "");
        System.out.println(result);
    }
}
```

10.1特点

该种方式可以自定义关键元素，方便以后维护，是一种标准的开发方式

11 第三种：HttpURLConnection 调用方式

开发步骤：
第一步：创建服务地址

第二步：打开一个通向服务地址的连接

第三步：设置参数
设置 POST，POST 必须大写，如果不大写，报如下异常

```
<terminated> HttpClient (16) [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (2015年11月26日 下午4:16:47)
Exception in thread "main" java.net.ProtocolException: Invalid HTTP method: post
        at java.net.HttpURLConnection.setRequestMethod(HttpURLConnection.java:428)
        at cn.itcast.mobile.client.HttpClient.main(HttpClient.java:30)
```

如果不设置输入输出，会报如下异常

```
Exception in thread "main" java.net.ProtocolException: cannot write to a URLConnection if doOutput=false - call setDoOutput(true)
        at sun.net.www.protocol.http.HttpURLConnection.getOutputStream(HttpURLConnection.java:1073)
        at cn.itcast.mobile.client.HttpClient.main(HttpClient.java:39)
```

第四步：组织 SOAP 数据，发送请求

第五步：接收服务端响应，打印

```
package cn.itcast.mobile.client;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

/**
 *
 * <p>Title: HttpClient.java</p>
 * <p>Description:HttpURLConnection调用方式</p>
 * <p>Company: www.itcast.com</p>
 * @author  传智.at
 * @date    2015年11月26日下午3:58:57
 * @version 1.0
 */
public class HttpClient {

    public static void main(String[] args) throws IOException
{
        //第一步：创建服务地址，不是WSDL地址
        URL url = new
URL("http://webservice.webxml.com.cn/WebServices/MobileCodeWS
.asmx");
        //第二步：打开一个通向服务地址的连接
        HttpURLConnection connection = (HttpURLConnection)
url.openConnection();
        //第三步：设置参数
        //3.1发送方式设置：POST必须大写
        connection.setRequestMethod("POST");
        //3.2设置数据格式：content-type
        connection.setRequestProperty("content-type",
"text/xml;charset=utf-8");
        //3.3设置输入输出，因为默认新创建的connection没有读写权
```

```
限，
        connection.setDoInput(true);
        connection.setDoOutput(true);

        //第四步：组织SOAP数据，发送请求
        String soapXML = getXML("15226466316");
        OutputStream os = connection.getOutputStream();
        os.write(soapXML.getBytes());
        //第五步：接收服务端响应，打印
        int responseCode = connection.getResponseCode();
        if(200 == responseCode){//表示服务端响应成功
            InputStream is = connection.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);

            StringBuilder sb = new StringBuilder();
            String temp = null;
            while(null != (temp = br.readLine())){
                sb.append(temp);
            }
            System.out.println(sb.toString());

            is.close();
            isr.close();
            br.close();
        }

        os.close();
    }

    /**
     * <?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>string</mobileCode>
      <userID>string</userID>
    </getMobileCodeInfo>
  </soap:Body>
</soap:Envelope>
     * @param phoneNum
     * @return
```

```
     */
    public static String getXML(String phoneNum){
        String soapXML = "<?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>"+phoneNum+"</mobileCode>"
            +"<userID></userID>"
           +"</getMobileCodeInfo>"
         +"</soap:Body>"
        +"</soap:Envelope>";
        return soapXML;
    }
}
```

12 Ajax 调用方式

```
<!doctype html>
<html lang="en">
 <head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script type="text/javascript">
    function queryMobile(){
        //创建 XMLHttpRequest 对象
        var xhr = new XMLHttpRequest();
        //打开连接

    xhr.open("post","http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx",
true);
        //设置数据类型
        xhr.setRequestHeader("content-type","text/xml;charset=utf-8");
        //设置回调函数
        xhr.onreadystatechange=function(){
```

```
                //判断是否发送成功和判断服务端是否响应成功
                if(4 == xhr.readyState && 200 == xhr.status){
                    alert(xhr.responseText);
                }
            }
            //组织 SOAP 协议数据
            var soapXML = "<?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>"+document.getElementById("phoneNum").value+"</mobileCode>"
                   +"<userID></userID>"
                +"</getMobileCodeInfo>"
              +"</soap:Body>"
            +"</soap:Envelope>";
            alert(soapXML);
            //发送数据
            xhr.send(soapXML);
     }
   </script>
 </head>
 <body>
   手机号查询：<input type="text" id="phoneNum"/> <input type="button" value="查询"
onclick="javascript:queryMobile();"/>
 </body>
</html>
```

13　深入开发：用注解修改 WSDL 内容

WebService 的注解都位于 javax.jws 包下:

@WebService-定义服务，在 public class 上边

targetNamespace：指定命名空间

name：portType 的名称

portName：port 的名称

serviceName：服务名称

endpointInterface：SEI 接口地址，如果一个服务类实现了多个接口，只需要发布一个接口的方法，可通过此注解指定要发布服务的接口。

@WebMethod-定义方法，在公开方法上边

operationName：方法名

exclude：设置为 true 表示此方法不是 webservice 方法，反之则表示 webservice 方法，默认是 false

@WebResult-定义返回值，在方法返回值前边

name：返回结果值的名称

@WebParam-定义参数，在方法参数前边

name：指定参数的名称

作用：

通过注解，可以更加形像的描述 Web 服务。对自动生成的 wsdl 文档进行修改，为使用者提供一个更加清晰的 wsdl 文档。

当修改了 WebService 注解之后，会影响客户端生成的代码。调用的方法名和参数名也发生了变化