

#11 Web服务开发



目录

Web服务的核心技术及其规范

Web服务的调用原理

Web服务的应用

**Web服务的基本
概念**

Web服务 (web services)



- 基于网络的、分布式的模块化组件，它执行特定的任务，遵守具体的技术规范，这些规范使得**Web Service**能与其他兼容的组件进行互操作。
- **Web Services** 主要利用 **HTTP** 和 **SOAP** 协议使商业数据在 **Web** 上传输，**SOAP**通过**HTTP** 调用商业对象执行远程功能调用，**Web** 用户能够使用 **SOAP** 和 **HTTP**通过 **Web**调用的方法来调用远程对象。

Web服务的核心技术及其规范

- **Web**服务一般是由企业发布的，具有特定商业需求的在线应用服务。应用软件能够通过互联网来访问和使用这项服务。
- **Web**服务的主要目标是在不同平台下的可操作性。

Web服务的核心技术及其规范

Web服务主要用到以下几个核心技术和规范：

-  XML: 描述数据的标准方法
-  SOAP: 表示信息交换的协议
-  WSDL: Web服务描述语言
-  UDDI (Universal Description, Discovery and Integration): 通用描述、发现与集成协议，它是一种独立于平台的，基于XML语言的用于在互联网上描述商务的协议

Web服务的核心技术及其规范

➤ XML(Extensible Markup Language)可扩展标记语言

XML是当前处理结构化文档信息的有力工具，是网络环境中跨平台并依赖于内容的技术，是一种简单的数据存储语言。

➤ SOAP(Simple Object Access Protocol)简单对象访问协议

SOAP技术把基于HTTP的Web技术与XML的可扩展性相结合，实现异构程序和平台之间的互操作性，使应用能够被不同的用户所访问。

Web服务的核心技术及其规范

➤ WSDL(Web Services Description Language) Web服务描述语言

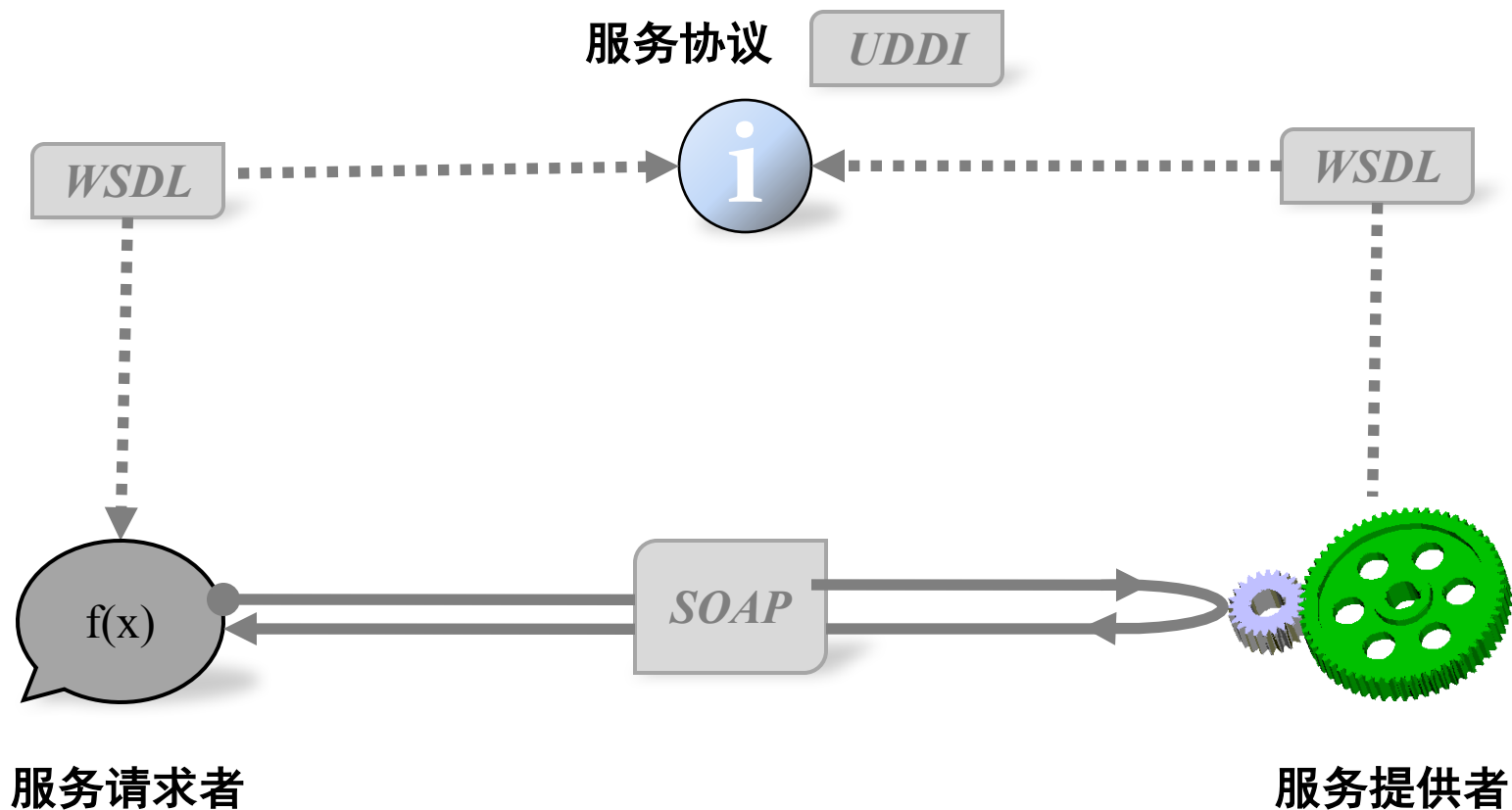
WSDL是一种用于描述Web服务的XML格式。WSDL提供服务的详细操作信息。

➤ UDDI(Universal Description, Discovery and Integration) 通用描述、发现与集成协议

UDDI是一个独立平台的，基于XML语言的注册表和机制。注册表记录了互联网上的商务应用。它也提供了等级和查找Web服务应用程序的机制。

Web服务的核心技术及其规范

Web服务的架构



Web服务调用原理

- 服务提供者首先建立Web服务，然后把服务发布给所有用户。
- 任何平台上的用户可以通过阅读其WSDL文档生成一个SOAP请求消息。
这个SOAP消息嵌入到一个HTTP POST请求中发送到Web服务所在的Web服务器。
- Web服务器把请求转发给Web服务请求处理器，请求处理器解析SOAP请求，然后调用Web服务生成相应的SOAP应答。
- Web服务器得到SOAP应答后通过HTTP送回客户端。

Web服务调用原理

高层接口

- 使用高层接口，不需要知道SOAP和XML的任何信息，就可以生成和使用一个Web服务。
- Soap Toolkit 2.0 通过提供SoapClient和SoapServer两个COM对象来完成这些功能。

Web服务调用原理

底层接口

使用底层接口必须对SOAP和XML有所了解。这种接口可以对SOAP的处理过程进行控制，特别是要做特殊处理的时候。

- 1.创建一个HttpConnector对象负责HTTP连接。
- 2.创建SoapSerializer对象，用于生成SOAP消息。
- 3.SOAP消息作为Payload通过HttpConnector被发送到服务端。
- 4.生成一个SoapReader对象，负责读取服务端返回的SOAP消息。

Web服务的应用

企业之间的应用

- **Web**服务用于电子商务应用的标准和开发工具。
- 企业间的电子商务（**B2B**）：**Web**服务应用于以企业采购、物流和分销内容的供应链。
- 企业与消费者之间的电子商务（**B2C**）：**Web**服务应用于涉及到零售以及中间业务的支付系统

Web服务的应用

企业内部的应用

- 采用中间件应用服务器软件作为工具将企业各项应用都进行改造和开发
- 企业内部应用软件的网络化，包括现在流行的ERP以及CRM等

Android平台Web服务实现

第三方类库（**KSOAP2**）简介

KSOAP2的使用

应用实例详解

第三方类库(KSOAP2)简介

- Ksoap2是一个SOAP Webservice客户端包。主要用于资源受限制的Java环境如Applets或J2ME应用程序（CLDC/CDC/MIDP）。
- 官方网站: <http://ksoap.objectweb.org/>
- KSOAP2改进了对Microsoft .Net的兼容

KSOAP2的常用接口

接口

```
org.kSOAP2.SoapEnvelope  
org.kSOAP2.SoapSerializationEnvelope  
org.kSOAP2.SoapObject  
org.KSOAP2.transport.HttpTransport
```

- SoapEnvelope与SOAP规范中的SOAP Envelope相对应，封装了head和body。
- SoapSerializationEnvelope对SoapEnvelope进行了扩展来支持SOAP序列化规范，能够把简单对象自动进行序列化。

KSOAP2的常用接口

接口

```
org.kSOAP2.SoapEnvelope  
org.kSOAP2.SoapSerializationEnvelope  
org.kSOAP2.SoapObject  
org.KSOAP2.transport.HttpTransport
```

- SoapObject能够构造SOAP调用。
- HttpTransport屏蔽了网络请求或访问以及获取服务器SOAP的具体细节。

KSOAP2和Web服务

- 利用Web服务传递String给MIDP(Mobile Information Device Profile, 移动信息设备配置文件)

服务器端的主服务类：

服务器端

```
Public class kSOAPWS {  
  
    public kSOAPWS() {}  
  
    public String WSMMethod (String user, String pwd) {  
        return "WSResponse";  
    }  
}
```

KSOAP2和Web服务

KSOAP调用服务器的Web服务有6步：

- 1.指定Web服务的命名空间和调用方法的名称；
- 2.设置调用方法的参数（可选）；
- 3.生成调用Web服务的SOAP请求信息；
- 4.指定Web服务的WSDL文档的URL；
- 5.利用call调用Web服务；
- 6.利用getResponse方法获得Web服务的返回结果。

KSOAP2和Web服务

1.指定Web服务的命名空间和调用方法的名称

①.利用SoapObject类完成调用。

```
SoapObject request =new SoapObject(ServiceNamespace, MethodName);
```

➤ **ServiceNamespace** – Web服务的命名空间， 可从WSDL文档中找到。

➤ **MethodName** – 所调用方法的名字。

KSOAP2和Web服务

例如：

```
String url = http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx  
String namespace = "http://WebXml.com.cn/";  
String methodName = "getSupportCity";
```

命名空间和调用方法可参考网站

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

KSOAP2和Web服务

从WSDL查看WebService的Namespace

进入 <http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx?WSDL>

```
▼ <wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  
  xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"  
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"  
  xmlns:tns="http://WebXml.com.cn/" xmlns:s="http://www.w3.org/2001/XMLSchema"  
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"  
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://WebXml.com.cn/">  
  ▼ <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
```



Namespace

KSOAP2和Web服务

进入 <http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx>

查看WebService提供的方法

- [getSupportCity](#)

查询本天气预报Web Services支持的国内外城市或地区信息

输入参数: byProvinceName = 指定的洲或国内的省份, 若为ALL或空则表示返回全部城市; 返回数据: -

- [getSupportDataSet](#)

获得本天气预报Web Services支持的洲、国内外省份和城市信息

输入参数: 无; 返回: DataSet。DataSet.Tables(0) 为支持的洲和国内省份数据, DataSet.Tables(1) 为国外省份数据。

Tables(0): ID = ID主键, Zone = 支持的洲、省份; Tables(1): ID 主键, ZoneID 对应Tables(0)ID

- [getSupportProvince](#)

获得本天气预报Web Services支持的洲、国内外省份和城市信息

输入参数: 无; 返回数据: 一个一维字符串数组 String(), 内容为洲或国内省份的名称。

- [getWeatherbyCityName](#)

MethodName

KSOAP2和Web服务

2.设置调用方法的参数（可选），如果方法没有参数，则这一步可以省略

```
request.addProperty("User", "Password");
```

- `addProperty`方法设置的参数需要与Web服务类中的方法参数顺序保持一致
- 参数对大小写敏感,要与服务端一致

KSOAP2和Web服务

进入

<http://ws.webxml.com.cn/WebServices/WeatherWebService.asmx?op=getWeatherbyCityName>

查看

getWeatherbyCityName

WebService读入参数格式定义
WebService输出参数格式定义

以下是 SOAP 1.2 请求和响应示例。所显示的占位符需替换为

```
POST /WebServices/WeatherWebService.asmx HTTP/
Host: ws.webxml.com.cn
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://WebXml.com.cn/getWeatherby
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/20
  <soap:Body>
    <getWeatherbyCityName xmlns="http://WebXml
      <theCityName>string</theCityName>
    </getWeatherbyCityName>
  </soap:Body>
</soap:Envelope>
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/20
  <soap:Body>
    <getWeatherbyCityNameResponse xmlns="http:
      <getWeatherbyCityNameResult>
        <string>string</string>
        <string>string</string>
      </getWeatherbyCityNameResult>
    </getWeatherbyCityNameResponse>
  </soap:Body>
</soap:Envelope>
```

KSOAP2和Web服务

3.利用SoapSerializationEnvelope对象生成调用Web服务的SOAP请求信息

```
SoapSerializationEnvelope en =  
    new SoapSerializationEnvelope (SoapEnvelope.VER11);  
  
en.bodyOut = request;  
  
en.dotNet = true;          /*访问.NET的WebService必须加上这行*/
```

- SoapEnvelope.VER11是SOAP协议的版本号，该版本号要与服务器端Web服务的版本号一致；
- 在创建SOAP序列化封装对象后，需要设置属性bodyOut为第一步的SoapObject对象。

KSOAP2和Web服务

4.创建HttpTransportSE对象，通过这个对象的构造方法指定Web服务的WSDL文档的URL

```
HttpTransportSE ht = new HttpTransportSE (url);
```

5.使用call方法调用Web服务

```
HT.call(ServiceNamespace + MethodName, en);
```

- 第1个参数是完整的方法名,前面加上命名空间
- 第2个参数就是在之前创建的SoapSerializationEnvelope对象。

KSOAP2和Web服务

6.使用getResponse方法获得Web服务的返回结果

- 返回值是对象时:利用第3步创建的SOAP序列化封装对象获得的Web服务的返回结果，并强制类型转换为SoapObject类。

```
SoapObject SO = (SoapObject) en.getResponse();
```

- 返回值是单值时:则是SoapPrimitive，不是SoapObject；利用toString()得到结果

```
SoapPrimitive SO = (SoapPrimitive) en.getResponse();  
String txt = SO.toString();  /*返回值转换成字符串*/
```

KSOAP2和Web服务

小结：

- KSOAP调用Web服务需要运用HttpTransport类，实际上是调用了HttpConnection作为网络连接。
- 在KSOAP调用Web服务的时候，如果由于某种原因，Web服务不能立即返回，Android界面上的组件仍然需要处于活动状态供用户使用，不能造成阻塞。
- 为了防止UI组件的阻塞，KSOAP调用Web服务的时候，必须另起一个线程。

KSOAP的类型映射

KSOAP能够把四种SOAP类型映射为Java类型：

SOAP Type	Java Type
xsd:int	java.lang.Integer
xsd:long	java.lang.Long
xsd:string	java.lang.String
xsd:boolean	java.lang.Boolean

- 其余类型需要进行类型映射，把成员变量序列化为byte[]，通过网络传送后再反序列化

KSOAP的类型映射

- 在KSOAP中，利用Base64把二进制流编码为ASCII字符串，使二进制数据能够通过XML/SOAP传输；
- Org.kSOAP2.serialization中的MarshalBase64的目的就是把SOAP XML中的xsd:base64Binary元素序列化为Java字节数组类型。
- KSOAP2提供Marshaldate和MarshalHashtable类来把相应的元素序列化为Java的Data和Hashtable类型。

KSOAP应用实例

通过Web服务生成验证码图片

- 使用Ksoap2，发送文字信息到指定的网络service并获得生成的验证码图片显示出来
- 验证码图片Web服务：

<http://ws.webxml.com.cn/WebServices/ValidateCodeWebService.asmx?wsdl>

KSOAP应用实例

- 使用的方法是其中的enValidateByte 方法，该方法支持中文、英文、数字、和符号。

```
▼ <wsdl:operation name="enValidateByte">
  <soap:operation soapAction="http://WebXml.com.cn/enValidateByte" style="document"/>
  ▼ <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  ▼ <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
</wsdl:operation>
```

KSOAP应用实例

- KSOAP2可以简单地通过SoapObject的bodyIn方法获取验证码图片的字节流

```
HttpTransportSE se = new HttpTransportSE(URL);  
  
se.call(SOAPACTION, envelope);  
  
SoapObject result = (SoapObject) envelope.bodyIn;  
  
SoapPrimitive detail = (SoapPrimitive)result.getProperty("enValidateByteResult")
```

KSOAP应用实例

- 因为webservice 返回的是验证码图片的字节流，还需要转化成位图才能显示出图片

```
byte[] data = Base64.decode((msg.obj.toString()).getBytes(), Base64.DEFAULT);  
Bitmap bitmap = BitmapFactory.decodeByteArray(data, 0, data.length);
```

KSOAP应用实例

➤ 具体程序

//第1步：创建SoapObject对象，并制定Web服务的命名空间

```
SoapObject request = new SoapObject(NAMESPACE, METHODNAME);
```

//第2步：设置Web服务方法的参数

```
request.addProperty("byString", Code);
```

//第3步：创建SoapSerializationEnvelope对象，并制定Web服务的版本

```
SoapSerializationEnvelope envelope = new SoapSerializationEnvelope(SoapEnvelope.VER10);
```

```
envelope.dotNet = true;
```

```
envelope.setOutputSoapObject(request);
```

KSOAP应用实例

➤ 具体程序

//第4步：创建HttpTransportSE对象，并指定WSDL文档的URL

```
HttpTransportSE transport = new HttpTransportSE(URL);
```

```
try {
```

//第5步：调用Web服务

```
transport.call(SOAPACTION, envelope);
```

```
} catch (IOException e) {
```

```
    e.printStackTrace();
```

```
} catch (XmlPullParserException e) {
```

```
    e.printStackTrace();
```

```
}
```

//第6步：使用bodyIn方法获得Web服务方法的返回结果

```
SoapObject result = (SoapObject) envelope.bodyIn;
```

小结

KSOAP2是第三方开发的专门用于在移动设备调用WebService的类库。

- 使用KSOAP2调用WebService可分为6步来完成
- SoapObject对象来指定了要调用的方法
- 通过HttpTransportSE对象的call方法来调用WebService的方法
- 通过getResponse方法返回结果

总结

Web服务

- 创建可互操作的分布式应用程序的新平台。
- 为了达到跨平台操作，**Web**服务是完全基于**XML**、**XSD**等独立于平台、独立于软件供应商的标准的。
- **Web**服务适用于应用程序集成、**B2B**集成、代码和数据重用，以及通过**Web**进行客户端和服务器的通信。

QUESTIONS?

