**WSDL与WADL介绍与比较**

# 1背景知识介绍

Web Services 拥有三种基本的元素：SOAP、WSDL 以及 UDDI。

**什么是 SOAP？**

* 基本的 Web services 平台是 XML + HTTP。
* SOAP 指简易对象访问协议
* SOAP 是一种通信协议
* SOAP 用于应用程序之间的通信
* SOAP 是一种用于发送消息的格式
* SOAP 被设计用来通过因特网进行通信
* SOAP 独立于平台
* SOAP 独立于语言
* SOAP 基于 XML
* SOAP 很简单并可扩展
* SOAP 允许您绕过防火墙
* SOAP 将作为 W3C 标准来发展

**什么是 WSDL?**

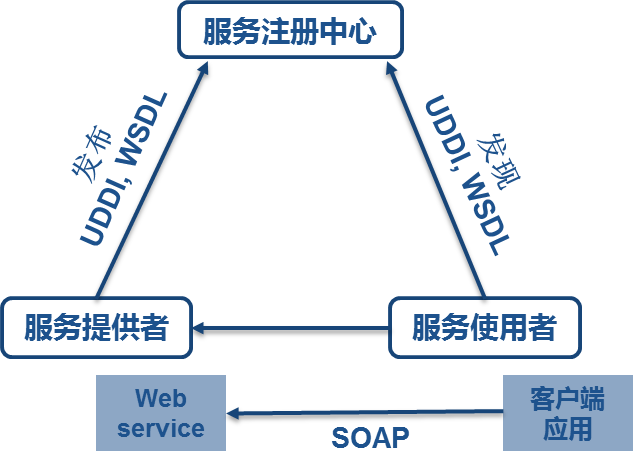
* WSDL 是基于 XML 的用于描述 Web Services 以及如何访问 Web Services 的语言
* WSDL 指网络服务描述语言
* WSDL 使用 XML 编写
* WSDL 是一种 XML 文档
* WSDL 用于描述网络服务
* WSDL 也可用于定位网络服务
* WSDL 还不是 W3C 标准

**什么是UDDI？**

* UDDI 是一种目录服务，通过它，企业可注册并搜索 Web services
* UDDI 指通用的描述、发现以及整合（Universal Description, Discovery and Integration）
* UDDI 是一种用于存储有关 web services 的信息的目录
* UDDI 是一种由 WSDL 描述的网络服务接口目录
* UDDI 经由 SOAP 进行通迅
* UDDI 被构建于 Microsoft .NET 平台之中

Web 服务的目的是通过使用Web 标准来完成应用程序之间的互操作性。下列技术规范定义了最初的Web 服务空间：SOAP [SOAP 1.1],Web 服务描述语言（WSDL）[WSDL 1.1],以及统一描述、发现，以及集成（UDDI）[UDDI]. SOAP 为基本服务的互操作性定义了 XML 消息传递协议。WSDL 介绍了描述服务的基本语法。UDDI为系统地发布和发现服务提供了所需的基础结构。

所有的Web 服务消息都通过SOAP(简单对象访问协议)标准的XML(可扩展标记语言)消息处理协议交换, 然后通过HTTP协议实现服务对象的远程调用。在调用的过程中, 用户根据自己的需求通过UDDI(统一描述、发现和集成协议)发现符合需求的服务, 最后利用WSDL(Web 服务描述语言)描述的接口规范编程实现接口。



# 2.WSDL

## 2.1 WSDL产生背景

为什么要使用WSDL描述服务？

* 开发工具可以自动处理通讯细节；
* 分布式系统的文档：其他系统知道如何与你的系统交流；系统复杂度越高越重要；
* 利于标准化：广为人知的服务。

WSDL（网络服务描述语言，Web Services Description Language）是基于 XML 的用于描述 Web Services 以及如何访问 Web Services 的语言。

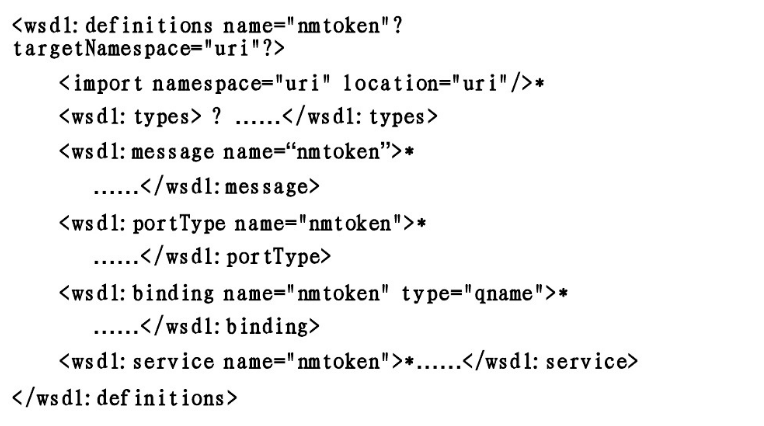
2001 年 3 月，WSDL 1.1 被 IBM、微软提交到W3C 作为一项建议标准，用于描述网络服务。2006年，W3C推出WSDL2.0版，对它做了一些修改，增加了一些新的特性。

WSDL描述Web服务的三个基本属性：

* What：服务做些什么，即服务提供的操作
* How：如何访问服务，即数据格式以及访问服务操作的必要协议
* Where：服务的地址，如URL

## 2.2 WSDL文档结构

WSDL文档的主要结构如下：



WSDL文档使用如下元素描述Web服务，这些元素嵌套在根元素definitions元素中：

* types - 数据类型定义的容器，一般使用XML Schema中的类型系统。
* message - 通信消息的数据结构的抽象类型化定义。使用types所定义的类型来定义整个消息的数据结构。
* operation - 描述服务中所支持的操作，一般单个Operation描述一个访问入口的请求/响应消息对。
* portType - 某个访问入口点类型所支持的操作的抽象集合，可以包含一个或多个operation 元素。
* binding - 特定端口类型的具体协议和数据格式规范的绑定。
* port - 定义单个服务访问点的访问入口部署细节。
* service - 相关服务访问点的集合，可以包含一个或多个port元素。

<types>元素用来定义数据结构。定义的数据类型在消息中会被引用。WSDL本身并没有提供定义数据类型的语言，而是利用已有的语言。为了实现最大程度的互操作性和平台中立性，WSDL选用XML Schema定义（XSD）作为标准类型系统。

**<definitions .... >**

**<types>**

**<xsd:schema .... />\***

**</types>**

**</definitions>**

<message>元素定义了Web服务之间交换的消息类型。每一条消息又可以分为多个部分，每一部分用一个<part>元素来标识。<part>元素可以通过type属性来引用在<types>元素中定义的数据类型，也可以通过<element>元素引用全局定义的元素。

例：使用type属性定义part元素

**<message name="FindItemsRequest">**

**<part name="MaxEntries" type="xsd:int"/>**

**<part name="SellerID" type="xsd:string"/>**

**</message >**

<portType>元素定义了Web服务支持的操作组。每一个操作由<operation>元素定义，其中又分别通过<input>和<output>元素定义了该操作的输入消息和输出消息。这两个元素都使用message属性引用在<message>元素中定义的消息。WSDL理论上支持4种操作，分别为：

* 单向请求：只有input消息，没有output消息。
* 请求/响应：先input消息，后output消息。
* 响应/请求：先output消息，后input消息。
* 单向响应：只有output消息，没有input消息。

其中前两种应用比较广泛。

<binding>元素定义了如何将端口类型绑定到具体的传输协议及如何格式化数据。WSDL支持不同的绑定，应用最广泛的是SOAP绑定。SOAP绑定规定如何抽取操作中的<message>元素来构造相应的SOAP消息。SOAP有两种编程模式，即文档模式和RPC模式。在SOAP绑定时也需要指定编程模式。使用文档模式时，在<message>元素中定义的<part>元素直接作为SOAP消息<body>元素的子元素插入到SOAP消息中，而使用RPC模式则需额外引入一个包装元素，用来标识该RPC调用。

**<binding…>**

**<soap:binding transport="uri"? style="rpc|document"?>**

**</binding>**

* style属性值是它所包含的操作子元素的style属性的默认值。
* transport属性是必需的，它的值表示绑定的传输方式与SOAP的何种传输方式一致。uri值[ http：//schemas.xmlsoap.org/soap/http ]表示传输方式和SOAP规范的HTTP绑定相一致。

例：文档模式

**<wsdl:binding name="JustServiceSoapBinding" type="impl:JustService">**

**<wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>**

**<wsdl:operation name="just">**

**<wsdlsoap:operation soapAction=""/>**

**<wsdl:input name="justRequest">**

**<wsdlsoap:body use="literal"/>**

**</wsdl:input>**

**<wsdl:output name="justResponse">**

**<wsdlsoap:body use="literal"/>**

**</wsdl:output>**

**</wsdl:operation>**

**</wsdl:binding>**

<service>元素用来指明服务的地址。对于每一个定义的绑定，<service>元素包含一个<port>元素来指明它的具体地址。一个<service>元素理论上可以包括多个<port>元素，分别指向不同的portType。

**<wsdl:service name="JustService">**

**<wsdl:port name="JustService" binding="impl:JustServiceSoapBinding">**

**<wsdlsoap:address location="http://localhost:8080/axis2/services/JustService"/>**

**</wsdl:port>**

**</wsdl:service>**

一个portType可以对应多个binding，一个binding可以对应多个port。

**binding**

**binding**

**binding**

**port**

**port**

**port**

**port**

相对于WSDL1.1，WSDL2.0主要在以下几方面有所改进：

* 根元素为<description>,替代了1.1中的<definitions>元素。
* 去掉了<message>元素，在定义操作时直接引用XML Schema定义的全局元素。
* 使用元素<interface>代替了1.1中的<portType>，更代表它的真实含义。
* 引入了接口的继承, 可以像面向对象一样，通过继承现有的接口来定义新的接口。
* 使用元素<endpoint>代替了<port>元素。
* 每个<service>只能实现一个接口，但可以包含不同的实现，即可以包含多个<endpoint>元素，分别对应不同的地址和实现。

# 3 SOAP协议

## 3.1 SOAP协议简介

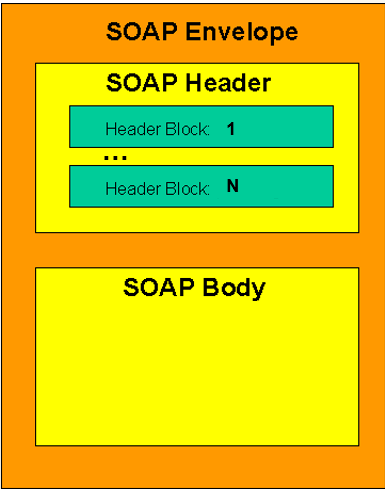
SOAP协议一开始代表“简单对象访问协议”，是由微软最早提出的，稍后IBM也加入了该协议的制定。双方在2000年推出了SOAP1.1版。2003年形成1.2版。

SOAP协议（Simple Object Access Protocol，简单对象访问协议）是在分散或分布式的环境中交换信息的简单协议，是一个基于XML的协议。SOAP协议主要包括四个部分：

* SOAP封装(envelop)，定义了一个消息框架，描述消息的内容是什么，是谁发送的，谁应当接受并处理它以及如何处理；
* SOAP编码规则（encoding rules），用于表示应用程序需要使用的数据类型的实例;
* SOAP RPC表示(RPC representation)，表示远程过程调用和应答的协定;
* SOAP绑定（binding），使用底层协议交换信息。

一条 SOAP 消息就是一个普通的 XML 文档，包含下列元素：

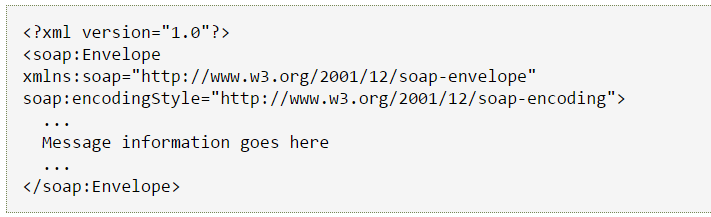
* 必需的 Envelope 元素，可把此 XML 文档标识为一条 SOAP 消息
* 可选的 Header 元素，包含头部信息
* 必需的 Body 元素，包含所有的调用和响应信息
* 可选的 Fault 元素，提供有关在处理此消息所发生错误的信息



SOAP消息的基本结构如下所示：



Envelope是SOAP消息的根元素，可把 XML 文档定义为 SOAP 消息。包括一个可选的Header元素和一个必需的Body元素。encodingStyle 属性：用于定义在文档中使用的数据类型。此属性可出现在任何 SOAP 元素中，并会被应用到元素的内容及元素的所有子元素上。SOAP 消息没有默认的编码方式。



可选的 SOAP Header 元素包含头部信息，可包含有关 SOAP 消息的应用程序专用信息（比如认证、支付等）。如果 Header 元素被提供，则它必须是 Envelope 元素的第一个子元素。Header元素可以存储一些元信息，提供了一种通用的实现对SOAP扩展的机制。Header又可以被分为许多小块，每一小块被称为一个Header Block。



上面的例子包含了一个带有一个 "Trans" 元素的头部，它的值是 234，此元素的 "mustUnderstand" 属性的值是 "1"。

对于Header元素，SOAP定义了三个属性：actor、mustUnderstand 以及 encodingStyle。这些被定义在 SOAP 头部的属性可定义容器如何对 SOAP 消息进行处理。

* actor 属性： SOAP 的 actor 属性可用于将 Header 元素寻址到一个特定的端点。省略SOAP actor属性，则表明接收者是SOAP消息的最终接收者。
* mustUnderstand 属性：mustUnderstand=“0|1”，可用于标识一个Header条目对于要对其进行处理的接收者来说是强制的还是可选的。"mustUnderstand="1"，则处理此头部的接收者必须认可此元素。假如此接收者无法认可此元素，则必须宣称处理消息失败。
* encodingStyle属性同上。

必需的 SOAP Body 元素可包含打算传送到消息最终端点的实际 SOAP 消息，由消息的接收者处理。下例表示请求苹果的价格。下面的 m:GetPrice 和 Item 元素是应用程序专用的元素。它们并不是 SOAP 标准的一部分。



## 3.2 SOAP的编程风格

SOAP消息需要由底层的传输层来传输，但又不局限于某个传输协议，而是可以使用多种不同的传输协议。在SOAP消息传递过程中，不同的SOAP结点之间也可以使用不同的传输协议，这样使SOAP消息的传输非常灵活。应用最广泛的为HTTP协议绑定。

SOAP1.1和SOAP1.2的区别：

* 两者的命名空间不同。

Soap1.1的命名空间：

xmlns:soap=“http://schemas.xmlsoap.org/soap/envelope/”

Soap1.2 命名空间：

xmlns:soap="http://www.w3.org/2003/05/soap-envelope”

* SOAP1.1版本与SOAP1.2版本在头信息上存在差异。

SOAP1.1的HTTP请求头：Content-Type: text/xml; charset=UTF-8; SOAPAction

SOAP1.2的请求头：Content-Type: application/soap+xml; charset=UTF-8; 无

* 基于SOAP1.1生成的WSDL和基于SOAP1.2生成的WSDL不同。

如：在定义Service部分差别如下：

Soap1.1是以：soap:address定义。

Soap1.2是以: soap12:address定义。

* 在CXF中两种协议请求的方式也不一样。

Soap1.1为content-Type: text/xml; charset=UTF-8

Soap1.2为content-Type: application/soap+xml; charset=UTF-8

# 4 WADL

## 4.1 WADL产生背景

WADL比较简单, 应用范围有限,仅限于描述HTTP应用程序, 并不涉及安全性等功能。 WSDL 2.0 HTTP绑定具有更加丰富功能的，但是代价是文档的复杂性增加。

首先介绍一下Web应用的概念：Web应用时基于HTTP的应用，其交互能够被计算机处理。Web应用由完成特定任务的各种Web组件构成并通过Web将服务展示给外界。Web应用常见的架构是Browser/Server架构，即B/S架构。其第一个优势是用户很容易访问，只需要有浏览器就可以访问Web应用了；第二个优势是能够很好地应用在Internet上，原因是采用Internet标准的通信协议。

但是Web应用仍然存在一些问题。第一个问题是在缺乏对Web应用程序的正式描述的情况下，基于非浏览器交互的Web应用程序开发变得困难。这个问题讲的是在纯粹基于浏览器的上下文中，没有必要提供这样的描述，因为Web应用程序以（X）HTML和Web表单的形式向用户提供其接口，在浏览器中呈现，由最终用户直接进行交互。缺乏对Web应用程序界面的正式描述，使得与Web应用程序开发非基于浏览器的交互变得困难（尽管当然不是不可能）。 想要鼓励基于非浏览器的访问的Web应用程序的作者通常提供了基于语言的自然语言描述，这些描述受到错误解释和版本偏差问题的影响。第二个问题是基于自然语言的Web应用程序描述存在诸多问题，比如错误解释和版本偏差，另外就是不能自动化或简化与web应用交互的软件的开发。错误解释指的是Web应用及其描述之间没有正式的链接，所以描述与实际的服务接口不一致是相当常见的事情；版本偏差问题指的是应用程序接口发生变化但自然语言描述没有相应的变化。

针对这两个问题，研究者提出描述Web应用时使用形式化的描述语言的解决方案。形式化语言使用精确的数学或机器可处理的公式定义的语言，这样做的好处有两个，一是可以简化与Web应用交互的软件的设计和开发，不再需要考虑如何去描述一个Web应用，而使用大家约定好的方法来描述Web应用，二是为我们可以依据此语言开发相关支持工具来帮助我们做事，可以在Web应用程序生成接口时以自动化的方式生成描述文档，确保描述与实现一致，从而减轻上述基于自然语言的描述的问题。

## 4.2 WADL文档结构

WADL是一种Web应用描述语言，它是以XML的方式对基于HTTP的Web应用进行描述，并且机器可读。WADL的特点是以资源为中心。以资源为中心说的是它主要描述的是Web应用提供的资源，并为Web应用提供的资源及其之间的关系进行建模。另一个特点是平台无关性和语言无关性。XML是可扩展标记语言，用来标记数据，定义数据类型，是一种通用的数据交换格式。这种语言在不同的平台和编程语言里面的解析方式都是一样的，一种是SAX，一种是DOM。所以说WADL是平台无关的和语言无关的。在对Web应用进行描述时，WADL使用一组资源元素来描述应用。其中，请求元素指定如何表示输入及需要哪些类型的输入，响应元素描述了资源的相应及故障信息的表示方式。

WADL文档主要由以下几个元素构成：

* application元素：WADL文档的根；
* grammars 元素：承载数据格式的定义；
* resources元素：Web应用所提供资源的容器；
* resource元素：描述一组资源；
* resource\_type元素：描述一组定义资源行为的方法；
* method元素：描述可以应用于资源的HTTP协议方法的输入和输出；
* param元素：定义其父元素的参数组成；
* representation元素：描述资源状态的表示方法。

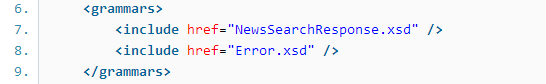
<application>元素形成了WADL描述文档的根，包括以下几个子元素：

* doc 元素------描述父元素
* grammars元素------承载数据格式的定义
* resources元素------ Web应用所提供资源的容器
* resource\_type元素------描述一组定义资源行为的方法
* method元素------描述应用于资源的HTTP协议方法的输入和输出
* representation元素------描述资源状态的表示方法
* param元素------定义其父元素的参数组成



上图是WADL文档的示例，可以发现application元素形成了整个文档的根，其他元素都是该元素的子元素。

<grammars>元素定义WADL文档描述的协议执行期间交换的数据的描述格式，通常使用include元素引用数据描述格式的定义。对于一个Web应用来说，它与外界交互时是需要遵循某些数据格式的，如果不知道一个Web应用对外交互时所遵循的数据格式，那么也没办法与其进行交互。Grammars元素用来定义WADL文档描述的协议执行期间交换的数据的描述格式，即这里告诉当前Web应用对外交互所使用的数据格式。



上图为<grammars>元素的示例，该元素引用了NewsSearchResponse和Error两个数据交换格式的描述。

<resources>元素充当Web应用提供的资源的容器。该元素的元素属性使base，该属性为每个resource标识符提供基础URI（base URI）。通常一个resources元素由数个resource元素组成，而resource元素是用来描述一组资源的。



上图是resources元素的示例。在该文档中下面的一个元素是resources元素，该元素充当Web应用提供的资源的容器，即所有的资源都放在这个元素中。该元素具有的元素属性是base，它表示的是为其resource子元素提供基础URI。该属性类似于提供一个eclipse中工作空间的绝对路径，资源都在这个路径里面。在该文档中，我们可以看到这个resources元素的基础URI如红线出base属性中的地址，元素的内部有一个resource元素，说明有一个资源。

<resource>元素用来描述一组资源，即对web应用提供的资源进行描述。该元素具有的元素属性如下所示：

* Id:标识resource元素，可选属性
* path:描述资源的相对URI，可选属性
* type:定义一组资源所支持的方法，可选属性

该元素的子元素包括doc、method和resource。doc用来描述当前元素；method是方法元素，即对资源的操作方法；resource元素可作为另一个resource元素的子元素，来描述一组资源的集合。



在该文档中根据base基础URI和resource中的相对URI我们可以确定resource元素的URI为图中的红字，即前面的加后面的。

<method>元素用来描述可以应用于资源的HTTP协议方法的输入和输出。该元素的属性包括两个，如下所示：

* name：指明使用的HTTP方法
* id：method元素的标识符

Method的子元素包括两部分，一部分是描述输入的，即request，另一部分是描述输出的即request。允许存在name属性值相同的多个method元素，这些method元素的输入不同。这些元素类似于java中的构造函数，可以定义多个，但是参数不同。



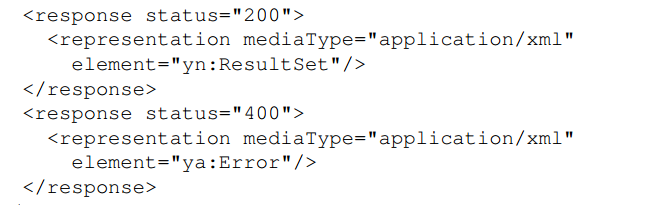
在这个method方法中，可以看到使用资源的方法是使用HTTP中的GET，下面的request和response分别描述了使用GET时应当包括的参数和硁硁返回的结果。

<request>元素描述HTTP方法应用于资源时所需要的输入。该元素没有元素属性。其子元素包括三个，分别是doc、representation和param。其中representation元素用来描述资源的状态的表示方法，param元素则用来描述请求所需要的参数。



在上述request元素中，可以看到的是使用该方法需要的参数很多，包括String类型的“appid”，String类型的“query”等。

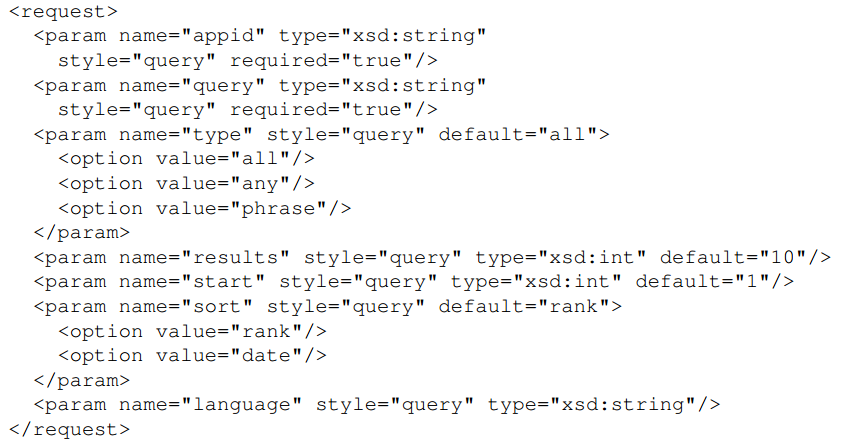
<response>元素用于描述在资源上执行HTTP方法所产生的输出。该元素包含一个元素属性status，用于提供与特定相应相关联的HTTP状态码的列表。该元素的子元素与require元素相同，分别是doc、representation和param。其中representation元素用来描述资源的状态的表示方法，param元素则用来描述输出的结果的参数类型。



上面的两个response元素分别表示两种响应状态。其中状态为“200”的在HTTP中表示请求成功，其返回的内容是representation元素，描述的是资源的状态，用xml来表示结果集。第二种状态为“400”的表示请求出错，并返回错误状态。

<param>元素描述其父元素的参数，可以描述的对象包括resource、application、request和response。该元素的属性有：

* id：标识当前参数，可选属性
* name: 指定参数的名称，必选属性
* style：指定参数的样式，可选属性
* type：指定参数的类型（XML中的类型），默认为XSD:string，可选属性
* default： 指定参数的默认值，可选属性
* required：指定参数是否需要存在，默认false，可选属性
* fixed：为参数提供固定值



# 5 WADL和WSDL的区别

WADL (Web Application Description Language)，用于描述基于HTTP的网络应用，如遵循REST架构风格的应用。

WSDL 2.0 (Web Services Description Language)，用于描述网络服务的接口，如基于SOAP的应用，也可以用来描述不使用SOAP的网络应用。

* Resources vs. Interfaces
* WADL：资源描述语言。WSDL：接口描述语言。
* HTTP only vs. transport protocol independent
* Message exchange pattern（Request-response vs. In-only,out-only,in-out&out-in）
* Stateless vs. Stateful
* Authentication（Not supported vs. Supported）
* URL encoded data（Not supported vs. Supported）
* Schema language for describing message content（XML Schema & RelaxNG vs. XML Schema

参考文献：

1. Erik Christensen, Francisco Curbera, Greg Meredith, et al. Web Services Description Language (WSDL) 1.1[EB/OL].(2001-03-15)[2017-04-24]. https://www.w3.org/TR/wsdl.
2. 柴晓路. WSDL : 描述你的Web服务[EB/OL]. [2017-04-24]. https://www.ibm.com/developerworks/cn/webservices/ws-wsdl/index.html.
3. SOA实践:构建基于Java Web服务和BPEL的企业级应用[M]. 电子工业出版社, 2009.
4. W3School. SOAP 教程[EB/OL].[2017-04-24]. http://www.w3school.com.cn/soap/ index.asp.
5. 段智华. 浅谈 SOAP[EB/OL]. [2017-04-24]. https://www.ibm.com/developerworks/ cn/xml/x-sisoap/.
6. Don Box, David Ehnebuske, Gopal Kakivaya, et al. Simple Object Access Protocol (SOAP)1.1[EB/OL]. (2000-05-08)[2017-04-24].

https://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

1. Toshiro Takase, Satoshi Makino, Shinya Kawanaka, et al. Definition Languages for RESTful Web Services: WADL vs. WSDL 2.0[EB/OL]. [2017-04-24]. https://www.ibm.com/developerworks/cn/webservices/specification/ws-wadlwsdl/.
2. Marc Hadley. Web Application Description Language[EB/OL]. (2009-08-31) [2017-04-24]. https://www.w3.org/Submission/wadl/.