

wsdl 文件结构分析

WSDL (Web Services Description Language, Web 服务描述语言) 是一种 XML Application, 他将 Web 服务描述定义为一组服务访问点, 客户端可以通过这些服务访问点对包含面向文档信息或面向过程调用的服务进行访问 (类似远程过程调用)。WSDL 首先对访问的操作和访问时使用的请求/响应消息进行抽象描述, 然后将其绑定到具体的传输协议和消息格式上以最终定义具体部署的服务访问点。相关的具体部署的服务访问点通过组合就成为抽象的 Web 服务。本文将详细讲解 WSDL 文档的结构, 并分析每个元素的作用。

一: WSDL 定义

WSDL 是一个用于精确描述 Web 服务的文档, WSDL 文档是一个遵循 WSDL XML 模式的 XML 文档。WSDL 文档将 Web 服务定义为服务访问点或端口的集合。在 WSDL 中, 由于服务访问点和消息的抽象定义已从具体的服务部署或数据格式绑定中分离出来, 因此可以对抽象定义进行再次使用: 消息, 指对交换数据的抽象描述; 而端口类型, 指操作的抽象集合。用于特定端口类型的具体协议和数据格式规范构成了可以再次使用的绑定。将 Web 访问地址与可再次使用的绑定相关联, 可以定义一个端口, 而端口的集合则定义为服务。

一个 WSDL 文档通常包含 7 个重要的元素, 即 types、import、message、portType、operation、binding、service 元素。这些元素嵌套在 definitions 元素中, definitions 是 WSDL 文档的根元素。文章的下一部分将会详细介绍 WSDL 的基本结构。

二: WSDL 的基本结构--概述

如第一部分最后描述的那样, 一个基本的 WSDL 文档包含 7 个重要的元素。下面将分别介绍这几个元素以及他们的作用。

WSDL 文档在 Web 服务的定义中使用下列元素:

- **Types** - 数据类型定义的容器, 它使用某种类型系统 (一般地使用 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** - 对服务中所支持的操作的抽象描述, 一般单个 Operation 描述了一个访问入口的请求/响应消息对。
- **PortType** - 对于某个访问入口点类型所支持的操作的抽象集合, 这些操作可以由一个或多个服务访问点来支持。


- Binding - 特定端口类型的具体协议和数据格式规范的绑定。
- Port - 定义为协议/数据格式绑定与具体 Web 访问地址组合的单个服务访问点。
- Service- 相关服务访问点的集合。

WSDL 的 xml schema 可以参照如下网址：<http://schemas.xmlsoap.org/wsdl/>

三：WSDL 的基本结构--详述

本节将通过一个例子详细描述 WSDL 文档每个元素的作用。下面一个例子是一个简单的 WSDL 文档的内容。

一个简单的 Web Service 的 WSDL 文档，该服务支持名为 sayHello 的唯一操作，该操作通过在 http 上运行 SOAP 协议来实现的。该请求接受一个字符串 name，经过处理后返回一个简单的字符串。文档如下：

Xml 代码 

```

1. <?xml version="1.0" encoding="UTF-8" ?>
2. <wsdl:definitions
3.     targetNamespace="http://com.liuxiang.xfireDemo/HelloService"
4.     xmlns:tns="http://com.liuxiang.xfireDemo/HelloService"
5.     xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
6.     xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
7.     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
8.     xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
9.     xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
10.    xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
11.    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
12.    <wsdl:types>
13.        <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
14.            attributeFormDefault="qualified" elementFormDefault="qualified"

```

```

15.         targetNamespace="http://com.liuxiang.xf
ireDemo/HelloService">
16.             <xsd:element name="sayHello">
17.                 <xsd:complexType>
18.                     <xsd:sequence>
19.                         <xsd:element m
axOccurs="1" minOccurs="1"
20.                             name="n
ame" nillable="true" type="xsd:string" />
21.                     </xsd:sequence>
22.                 </xsd:complexType>
23.             </xsd:element>
24.             <xsd:element name="sayHelloResponse">
25.                 <xsd:complexType>
26.                     <xsd:sequence>
27.                         <xsd:element m
axOccurs="1" minOccurs="1"
28.                             name="o
ut" nillable="true" type="xsd:string" />
29.                     </xsd:sequence>
30.                 </xsd:complexType>
31.             </xsd:element>
32.         </xsd:schema>
33.     </wsdl:types>
34.     <wsdl:message name="sayHelloResponse">
35.         <wsdl:part name="parameters" element="tns:say
HelloResponse" />
36.     </wsdl:message>
37.     <wsdl:message name="sayHelloRequest">
38.         <wsdl:part name="parameters" element="tns:say
Hello" />
39.     </wsdl:message>
40.     <wsdl:portType name="HelloServicePortType">
41.         <wsdl:operation name="sayHello">
42.             <wsdl:input name="sayHelloRequest"
43.                 message="tns:sayHelloRequest"
/>
44.             <wsdl:output name="sayHelloResponse"
45.                 message="tns:sayHelloResponse"
/>
46.         </wsdl:operation>

```

```

47.         </wsdl:portType>
48.         <wsdl:binding name="HelloServiceHttpBinding"
49.             type="tns:HelloServicePortType">
50.             <wsdlsoap:binding style="document"
51.                 transport="http://schemas.xmlsoap.org/s
oap/http" />
52.             <wsdl:operation name="sayHello">
53.                 <wsdlsoap:operation soapAction="" />
54.
55.                 <wsdl:input name="sayHelloRequest">
56.                     <wsdlsoap:body use="literal"
57. />
58.                     </wsdl:input>
59.                     <wsdl:output name="sayHelloResponse">
60.                         <wsdlsoap:body use="literal"
61. />
62.                         </wsdl:output>
63.                     </wsdl:operation>
64.                 </wsdl:binding>
65.                 <wsdl:service name="HelloService">
66.                     <wsdl:port name="HelloServiceHttpPort"
67.                         binding="tns:HelloServiceHttpBinding">
68.                             <wsdlsoap:address
69.                                 location="http://localhost:8080
/xfire/services/HelloService" />
70.                             </wsdl:port>
71.                         </wsdl:service>
72.                     </wsdl:definitions>

```

- ◆ types 元素使用 XML 模式语言声明在 WSDL 文档中的其他位置使用的复杂数据类型与元素；
- ◆ import 元素类似于 XML 模式文档中的 import 元素，用于从其他 WSDL 文档中导入 WSDL 定义；
- ◆ message 元素使用在 WSDL 文档的 type 元素中定义或在 import 元素引用的外部 WSDL 文档中定义的 XML 模式的内置类型、复杂类型或元素描述了消息的有效负载；
- ◆ portType 元素和 operation 元素描述了 Web 服务的接口并定义了他的方法。portType 元素和 operation 元素类似于 java 接口和接口中定义的方法声明。

operation 元素使用一个或者多个 message 类型来定义他的输入和输出的有效负载;


◆ Binding 元素将 portType 元素和 operation 元素赋给一个特殊的协议和编码样式;

◆ service 元素负责将 Internet 地址赋给一个具体的绑定;

1、definitions 元素

所有的 WSDL 文档的根元素均是 definitions 元素。该元素封装了整个文档, 同时通过其 name 提供了一个 WSDL 文档。除了提供一个命名空间外, 该元素没有其他作用, 故不作详细描述。


下面的代码是一个 definitions 元素的结构:

Xml 代码 

```
1. <wsdl:definitions
2.     targetNamespace="http://com.liuxiang.xfireDemo/HelloService"
3.     xmlns:tns="http://com.liuxiang.xfireDemo/HelloService"
4.     xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
5.     xmlns:soap12="http://www.w3.org/2003/05/soap-envelope"
6.     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
7.     xmlns:soapenc11="http://schemas.xmlsoap.org/soap/encoding/"
8.     xmlns:soapenc12="http://www.w3.org/2003/05/soap-encoding"
9.     xmlns:soap11="http://schemas.xmlsoap.org/soap/envelope/"
10.    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
11.</wsdl:definitions>
```

2、types 元素

WSDL 采用了 W3C XML 模式内置类型作为其基本类型系统。types 元素用作一个容器, 用于定义 XML 模式内置类型中没有描述的各种数据类型。当声明消息部分的有效负载时, 消息定义使用了在 types 元素中定义的数据类型和元素。在本文的 WSDL 文档中的 types 定义:

Xml 代码 

```

1. <wsdl:types>
2.           <xsd:schema xmlns:xsd="http://www.w3.org/2001/
   XMLSchema"
3.               attributeFormDefault="qualified" eleme
   ntFormDefault="qualified"
4.               targetNamespace="http://com.liuxiang.xf
   ireDemo/HelloService">
5.           <xsd:element name="sayHello">
6.               <xsd:complexType>
7.                   <xsd:sequence>
8.                       <xsd:element m
   axOccurs="1" minOccurs="1"
9.                           name="n
   ame" nillable="true" type="xsd:string" />
10.                   </xsd:sequence>
11.               </xsd:complexType>
12.           </xsd:element>
13.           <xsd:element name="sayHelloResponse">
14.               <xsd:complexType>
15.                   <xsd:sequence>
16.                       <xsd:element m
   axOccurs="1" minOccurs="1"
17.                           name="o
   ut" nillable="true" type="xsd:string" />
18.                   </xsd:sequence>
19.               </xsd:complexType>
20.           </xsd:element>
21.       </xsd:schema>
22. </wsdl:types>

```

上面是数据定义部分，该部分定义了两个元素，一个是 sayHello，一个是 sayHelloResponse：

sayHello：定义了一个复杂类型，仅仅包含一个简单的字符串，将来用来描述操作的参入传入部分；


sayHelloResponse：定义了一个复杂类型，仅仅包含一个简单的字符串，将来用来描述操作的返回值；

3、import 元素

import 元素使得可以在当前的 WSDL 文档中使用其他 WSDL 文档中指定的命名空

间中的定义元素。本例子中没有使用 import 元素。通常在用户希望模块化 WSDL 文档的时候，该功能是非常有效果的。

import 的格式如下：

Code 代码 

```
1. <wsdl:import namespace="http://xxx.xxx.xxx/xxx/xxx" location="http://xxx.xxx.xxx/xxx/xxx.wsdl"/>
```


必须有 namespace 属性和 location 属性：

namespace 属性：值必须与正导入的 WSDL 文档中声明的 targetNamespace 相匹配；

location 属性：必须指向一个实际的 WSDL 文档，并且该文档不能为空。

4、message 元素

message 元素描述了 Web 服务使用消息的有效负载。message 元素可以描述输出或者接受消息的有效负载；还可以描述 SOAP 文件头和错误 detail 元素的内容。定义 message 元素的方式取决于使用 RPC 样式还是文档样式的消息传递。在本文中的 message 元素的定义，本文档使用了采用文档样式的消息传递：

Xml 代码 

```
1. <wsdl:message name="sayHelloResponse">
2.     <wsdl:part name="parameters" element="tns:sayHelloResponse" />
3. </wsdl:message>
4. <wsdl:message name="sayHelloRequest">
5.     <wsdl:part name="parameters" element="tns:sayHello" />
6. </wsdl:message>
```

该部分是消息格式的抽象定义：定义了两个消息 sayHelloResponse 和 sayHelloRequest：

sayHelloRequest：sayHello 操作的请求消息格式，由一个消息片断组成，名字为 parameters，元素是我们前面定义的 types 中的元素；

sayHelloResponse：sayHello 操作的响应消息格式，由一个消息片断组成，名字为 parameters，元素是我们前面定义的 types 中的元素；


如果采用 RPC 样式的消息传递，只需要将文档中的 element 元素应修改为 type

即可。

5、portType 元素

portType 元素定义了 Web 服务的抽象接口。该接口有点类似 Java 的接口，都是定义了一个抽象类型和方法，没有定义实现。在 WSDL 中，portType 元素是由 binding 和 service 元素来实现的，这两个元素用来说明 Web 服务实现使用的 Internet 协议、编码方案以及 Internet 地址。

一个 portType 中可以定义多个 operation，一个 operation 可以看作是一个方法，本文中 WSDL 文档的定义：


Xml 代码 

```
1. <wsdl:portType name="HelloServicePortType">
2.     <wsdl:operation name="sayHello">
3.         <wsdl:input name="sayHelloRequest"
4.             message="tns:sayHelloRequest" />
5.         <wsdl:output name="sayHelloResponse"
6.             message="tns:sayHelloResponse" />
7.     </wsdl:operation>
8. </wsdl:portType>
```

portType 定义了服务的调用模式的类型，这里包含一个操作 sayHello 方法，同时包含 input 和 output 表明该操作是一个请求 / 响应模式，请求消息是前面定义的 sayHelloRequest，响应消息是前面定义的 sayHelloResponse。input 表示传递到 Web 服务的有效负载，output 消息表示传递给客户的有效负载。

6、binding

binding 元素将一个抽象 portType 映射到一组具体协议(SOAP 和 HTTP)、消息传递样式、编码样式。通常 binding 元素与协议专有的元素和在一起使用，本文中的例子：

Xml 代码 

```
1. <wsdl:binding name="HelloServiceHttpBinding"
2.     type="tns:HelloServicePortType">
3.     <wsdlsoap:binding style="document"
4.         transport="http://schemas.xmlsoap.org/soap/http"
5.     />
6.     <wsdl:operation name="sayHello">
7.         <wsdlsoap:operation soapAction="" />
8.         <wsdl:input name="sayHelloRequest">
```



```
8.          <wsdlsoap:body use="literal" />

9.          </wsdl:input>
10.         <wsdl:output name="sayHelloResponse">
11.             <wsdlsoap:body use="literal" />

12.         </wsdl:output>
13.     </wsdl:binding>
```