第33讲 | 基于XML的SOAP协议：不要说NBA，请说美国职业篮球联赛

笔记本: P.趣谈网络协议
创建时间: 2018/8/3 17:07
作者: hongfenghuoju
URL:

第33讲 | 基于XML的SOAP协议：不要说NBA，请说美国职业篮球联赛

2018-08-01 刘超

第33讲 | 基于XML的SOAP协议：不要说NBA，请说美国职业篮球联赛
刘超
- 00:00 / 09:10

上一节我们讲了RPC的经典模型和设计要点，并用最早期的ONC RPC为例子，详述了具体的实现。

ONC RPC存在哪些问题?

ONC RPC将客户端要发送的参数，以及服务端要发送的回复，都压缩为一个二进制串，这样固然能够解决双方的协议约定问题，但是存在一定的不方便。

首先，**需要双方的压缩格式完全一致**，一点都不能差。一旦有少许的差错，多一位，少一位或者错一位，都可能造成无法解压缩。当然，我们可以用传输层的可靠性以及加入校验值等方式，来减少传输过程中的差错。

其次，**协议修改不灵活**。如果不是传输过程中造成的差错，而是客户端因为业务逻辑的改变，添加或者删除了字段，或者服务端添加或者删除了字段，而双方没有及时通知，或者线上系统没有及时升级，就会造成解压缩不成功。

因而，当业务发生改变，需要多传输一些参数或者少传输一些参数的时候，都需要及时通知对方，并且根据约定好的协议文件重新生成双方的Stub程序。自然，这样灵活性比较差。

如果仅仅是沟通的问题也还好解决，其实更难弄的还有**版本的问题**。比如在服务端提供一个服务，参数的格式是版本一的，已经有50个客户端在线上调用了。现在有一个客户端有个需求，要加一个字段，怎么办呢？这可是一个大工程，所有的客户端都要适配这个，需要重新写程序，加上这个字段，但是传输值是0，不需要这个字段的客户端很“冤”，本来没我啥事儿，为啥让我也忙活?

最后，**ONC RPC的设计明显是面向函数的，而非面向对象**。而当前面向对象的业务逻辑设计与实现方式已经成为主流。

这一切的根源就在于压缩。这就像平时我们爱用缩略语。如果是篮球爱好者，你直接说NBA，他马上就知道什么意思，但是如果你给一个大妈说NBA，她可能就不知所云。

所以，这种RPC框架只能用于客户端和服务端全由一拨人开发的场景，或者至少客户端和服务端的开发人员要密切沟通，相互合作，有大量的共同语言，才能按照既定的协议顺畅地进行工作。

XML与SOAP

但是，一般情况下，我们做一个服务，都是要提供给陌生人用的，你和客户不会经常沟通，也没有什么共同语言。就像你给别人介绍NBA，你要说美国职业篮球赛，这样不管他是干啥的，都能听得懂。

放到我们的场景中，对应的就是用**文本类**的方式进行传输。无论哪个客户端获得这个文本，都能够知道它的意义。

一种常见的文本类格式是XML。我们这里举个例子来看。

```
<?xml version="1.0" encoding="UTF-8"?>
<geek:purchaseOrder xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:geek="http://www.example.com/geek">
<order>
<date>2018-07-01</date>
<className>趣谈网络协议</className>
<Author>刘超</Author>
<price>68</price>
</order>
</geek:purchaseOrder>
```

我这里不准备详细讲述XML的语法规则，但是你相信我，看完下面的内容，即便你没有学过XML，也能一看就懂，这段XML描述的是什么，不像全面的二进制，你看到的都是010101，不知所云。

有了这个，刚才我们说的那几个问题就都不是问题了。

首先，**格式没必要完全一致**。比如如果我们把price和author换个位置，并不影响客户端和服务端解析这个文本，也根本不会误会，说这个作者的名字叫68。

如果有的客户端想增加一个字段，例如添加一个推荐人字段，只需要在上面的文件中加一行：

```
<recommended> Gary </recommended>
```

对于不需要这个字段的客户端，只要不解析这一行就是了。只要用简单的处理，就不会出现错误。

另外，这种表述方式显然是描述一个订单对象的，是一种面向对象的、更加接近用户场景的表示方式。

既然XML这么好，接下来我们来看看怎么把它用在RPC中。

传输协议问题

我们先解决第一个，传输协议的问题。

基于XML的最著名的通信协议就是**SOAP**了，全称**简单对象访问协议**（Simple Object Access Protocol）。它使用XML编写简单的请求和回复消息，并用HTTP协议进行传输。

SOAP将请求和回复放在一个信封里面，就像传递一个邮件一样。信封里面的信分**抬头**和**正文**。

```
POST /purchaseOrder HTTP/1.1
Host: www.geektime.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn
```

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
<soap:Header>
<m:Trans xmlns:m="http://www.w3schools.com/transaction/"
soap:mustUnderstand="1">1234
</m:Trans>
</soap:Header>
<soap:Body xmlns:m="http://www.geektime.com/perchaseOrder">
<m:purchaseOrder">
<order>
<date>2018-07-01</date>
<className>趣谈网络协议</className>
<Author>刘超</Author>
<price>68</price>
</order>
</m:purchaseOrder>
</soap:Body>
</soap:Envelope>
```

HTTP协议我们学过，这个请求使用POST方法，发送一个格式为 application/soap + xml 的XML正文给 www.geektime.com，从而下一个单，这个订单封装在SOAP的信封里面，并且表明这是一笔交易（transaction），而且订单的详情都已经写明了。

协议约定问题

接下来我们解决第二个问题，就是双方的协议约定是什么样的?

因为服务开发出来是给陌生人用的，就像上面下单的那个XML文件，对于客户端来说，它如何知道应该拼装成上面的格式呢？这就需要对于服务进行描述，因为调用的人不认识你，所以没办法找到你，问你的服务应该如何调用。

当然你可以写文档，然后放在官方网站上，但是你的文档不一定更新得那么及时，而且你也写的文档也不一定那么严谨，所以常常会有调试不成功的情况。因而，我们需要一种相对比较严谨的**Web服务描述语言**，**WSDL**（Web Service Description Languages）。它也是一个XML文件。

在这个文件中，要定义一个类型order，与上面的XML对应起来。

```
<wsdl:types>
<xsd:schema targetNamespace="http://www.example.org/geektime">
<xsd:complexType name="order">
<xsd:element name="date" type="xsd:string"></xsd:element>
<xsd:element name="className" type="xsd:string"></xsd:element>
<xsd:element name="Author" type="xsd:string"></xsd:element>
<xsd:element name="price" type="xsd:int"></xsd:element>
</xsd:complexType>
</xsd:schema>
</wsdl:types>
```

接下来，需要定义一个message的结构。

```
<wsdl:message name="purchase">
<wsdl:part name="purchaseOrder" element="tns:order"></wsdl:part>
</wsdl:message>
```

接下来，应该暴露一个端口。

```
<wsdl:portType name="PurchaseOrderService">
<wsdl:operation name="purchase">
<wsdl:input message="tns:purchase"></wsdl:input>
<wsdl:output message="......"></wsdl:output>
</wsdl:operation>
</wsdl:portType>
```

然后，我们来编写一个binding，将上面定义的信息绑定到SOAP请求的body里面。

```
<wsdl:binding name="purchaseOrderServiceSOAP" type="tns:PurchaseOrderService">
<soap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="purchase">
<wsdl:input>
<soap:body use="literal" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal" />
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
```

最后，我们需要编写service。

```
<wsdl:service name="PurchaseOrderServiceImplService">
<wsdl:port binding="tns:purchaseOrderServiceSOAP" name="PurchaseOrderServiceImplPort">
<soap:address location="http://www.geektime.com:8080/purchaseOrder" />
</wsdl:port>
</wsdl:service>
```

WSDL还是有些复杂的，不过好在有工具可以生成。

对于某个服务，哪怕是一个陌生人，都可以通过在服务地址后面加上“?wsdl”来获取到这个文件，但是这个文件还是比较复杂，比较难以看懂。不过好在也有工具可以根据WSDL生成客户端Stub，让客户端通过Stub进行远程调用，就跟调用本地的方法一样。

服务发现问题

最后解决第三个问题，服务发现问题。

这里有一个**UDDI**（Universal Description, Discovery, and Integration），也即**统一描述、发现和集成协议**。它其实是一个注册中心，服务提供方可以将上面的WSDL描述文件，发布到这个注册中心，注册完毕后，服务使用方可以查找到服务的描述，封装为本地的客户端进行调用。

小结

好了，这一节就到这里了，我们来总结一下。

- 原来的二进制RPC有很多缺点，格式要求严格，修改过于复杂，不面向对象，于是产生了基于文本的调用方式——基于XML的SOAP。
- SOAP有三大要素：协议约定用WSDL、传输协议用HTTP、服务发现用UDDL。

最后，给你留两个思考题：

1. 对于HTTP协议来讲，有多种方法，但是SOAP只用了POST，这样会有什么问题吗？
2. 基于文本的RPC虽然解决了二进制的问题，但是SOAP还是有点复杂，还有一种更便捷的接口规则，你知道是什么吗？

我们的专栏更新到第33讲，不知你掌握得如何？每节课后我留的思考题，你都有没有认真思考，并在留言区写下答案呢？我会从**已发布的文章中选出一批认真留言的同学**，赠送学习奖励礼券和我整理的独家网络协议知识图谱。

欢迎你留言和我讨论。趣谈网络协议，我们下期见！

_CountingStars

1.没有充分利用http协议原有的体系 比如get表示获取资源 post表示创建资源 delete表示删除资源 patch表示更新资源
2.restful协议

空档滑行

1.只使用post需要把动作封装到传输内容里
2.其他的协议比如json

blackpiglet

1. POST 请求构造比较麻烦，需要专门的工具，所以调用和调试更费事。
2. 更简单的应该就是RESTful 了吧，SOAP 感觉不太好用，复杂度比较高，用起来没有http顺手。

vloz

面向函数和面向对象在信息交互上的特征是什么？为什么讲onc合适面向函数？

叹息无门

感觉这篇写的不是很严谨:
1，首先SOAP并非只能通过HTTP进行传输，关于SOAP binding应该提一下?
2，SOAP 的HTTP Binding 支持比较完整的Web Method，http GET/POST都是可以支持的，并且对应不同的模式。大多数情况下只使用POST是具体实现的问题。

作者回复

是的，这里说的是通常的使用情况

凡凡

1.虽然http协议有post，get，head，put，delete等多种方法，但是平常来说post，get基本足够用。所以soap只支持post方法的差别应该在缺少get方法，get方法可以浏览器直接跳转，post必须借助表单或者ajax等提交。也就限制了soap请求只能在页面内获取或者提交数据。
另外，soap协议规范上是支持get的，但是由于一般xml比较复杂，不适合放在get请求的查询参数里，所以soap协议的服务多采用post请求方法。
2.应该要讲restful协议了，一种使用json格式交互数据的，基于http协议的，轻量级网络数据交互规范。

---

andy

可以使用类似thrift的DSL来描述服务接口，然后生成服务端和客户端

---

spdia

soap的方言问题过于严重。其实简单场景可以用http rest或者json+http post,或者用比较新的graphql