WebService

> 布尔教育 http://www.itbool.com
> 燕十八 著
> 欢迎传播 不传必究

快速了解WebService

通俗的说:按一定的XML格式,调用远程服务器的方法,且服务器按一定的格式返回XML内容.

"一定的格式"----SOAP（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的协议,是一个基于XML的协议.

远程服务器 ---- 一般通过HTTP协议来传递消息

总结: WebServie == HTTP协议 + Soap格式的XML

快速尝试WebService

```
POST /WebServices/MobileCodeWS.asmx HTTP/1.1
Host: webservice.webxml.com.cn
Content-Type: text/xml; charset=utf-8
Content-Length: 354
SOAPAction: "http://WebXml.com.cn/getMobileCodeInfo"

<?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
```

PHP客户端请求WebService

修改PHP.ini
extension=php_soap.dll 前的";"去掉.
并重启apache

PHP SoapClient类可以用来请求WebService

```php
$soap = new soapClient('http://webservice.webxml.com.cn/WebServices/MobileCodeWS.asmx?WSDL');
print_r($soap->getMobileCodeInfo( array('mobileCode'=>'13426060134') ) );
```

```
Array
(
    [0] => getMobileCodeInfoResponse getMobileCodeInfo(getMobileCodeInfo $parameters)
    [1] => getDatabaseInfoResponse getDatabaseInfo(getDatabaseInfo $parameters)
)
Array
(
    [0] => struct getMobileCodeInfo {
 string mobileCode;
 string userID;
}
    [1] => struct getMobileCodeInfoResponse {
 string getMobileCodeInfoResult;
}
    [2] => struct getDatabaseInfo {
}
    [3] => struct getDatabaseInfoResponse {
 ArrayOfString getDatabaseInfoResult;
}
    [4] => struct ArrayOfString {
```

```
 string string;
}
```

```
// 调用方法
print_r($soap->getMobileCodeInfo( array('mobileCode'=>'13426060134') ) );
```

// 打印结果

```
stdClass Object ( [getMobileCodeInfoResult] => 13426060134：北京 北京 北京移动动感地带卡 )
```

搭建WebService服务器

wsdl是什么?

wsdl是WebService的规格说明书.
详细阐述了WebService提供的
服务[service]--下的-->
频道[port];

阐述了[porttype]---下的--->
操作[operation]---下的--->
消息[message]-----下的--->
类型[type]

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions name='自定义名称[可选]'
  targetNamespace='命名空间[一般为URL]'
  xmlns:tns='命名空间[值同targetNamespace]'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns='http://schemas.xmlsoap.org/wsdl/'>

<!--<types> 元素定义 web service 使用的数据类型,WSDL 使用 XML Schema 语法来定义数据类型,也可以自定义Schema不包含的类型-->
 <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="[值同上tns]">
    </xsd:schema>
 </types>

<!--
<message> 元素可定义每个消息的部件,以及相关联的数据类型.
-->
<message name='操作名Request'>
    <part name="term" type="xsd:string"/>
</message>
<message name='操作名Response'>
    <part name="value" type="xsd:string"/>
</message>

<!--
<portType> 元素是最重要的 WSDL 元素.它可描述一个 web service、可被执行的操作,以及相关的消息.
它告诉你去哪个WebService的连接点,扮演了一个控制者.
-->
<portType name='操作列表名'>
  <operation name='操作名'>
    <input message='tns:操作名Request'/>
    <output message='tns:操作名Response'/>
  </operation>
</portType>

<!--<binding> 元素为每个端口定义消息格式和协议细节-->
```

```
<binding name='WS下的频道名称' type='tns:频道下的操作列表'>
<!--style:属性可取值 "rpc" 或 "document",ransport:属性定义了要使用的 SOAP 协议.在这个例子中我们使用 HTTP-->
  <soap:binding style='rpc'
    transport='http://schemas.xmlsoap.org/soap/http'/>
    <!--operation 元素定义了每个端口提供的操作符,对于每个操作,相应的 SOAP 行为都需要被定义-->
  <operation name='test'>
    <soap:operation soapAction='http://www.cwtservice.cn/newOperation/'/>
    <input>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
</binding>

<!--<service>包含一个或者多个port元素,每个port元素表示一个不同的Web服务-->
<service name='WebService名称[如weatherWS,shopWS]'>
  <port name='WS下的频道名称[如cartSoap,购物车服务]' binding='tns:[频道名,同左]'>
    <soap:address location='http://[webservice地址]'/>
  </port>
</service>
</definitions>
```

wsdl实例

```
<?xml version ='1.0' encoding ='UTF-8' ?>
<definitions
  targetNamespace='http://localhost/00/'
  xmlns:tns='http://localhost/00/'
  xmlns:soap='http://schemas.xmlsoap.org/wsdl/soap/'
  xmlns:xsd='http://www.w3.org/2001/XMLSchema'
  xmlns:soapenc='http://schemas.xmlsoap.org/soap/encoding/'
  xmlns:wsdl='http://schemas.xmlsoap.org/wsdl/'
  xmlns='http://schemas.xmlsoap.org/wsdl/'>

<!--<types> 元素定义 web service 使用的数据类型,WSDL 使用 XML Schema 语法来定义数据类型,也可以自定义Schema不包含的类型-->
 <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://localhost/00/">
    </xsd:schema>
 </types>

<!--
<message> 元素可定义每个消息的部件,以及相关联的数据类型.
-->
<message name='testRequest'>
    <part name="term" type="xsd:string"/>
</message>
<message name='testResponse'>
    <part name="value" type="xsd:string"/>
</message>

<!--
<portType> 元素是最重要的 WSDL 元素.它可描述一个 web service、可被执行的操作,以及相关的消息.
它告诉你去哪个WebService的连接点,扮演了一个控制者.
-->
<portType name='oplist'>
  <operation name='test'>
    <input message='tns:testRequest'/>
    <output message='tns:testResponse'/>
  </operation>
</portType>
```

```
<!--<binding> 元素为每个端口定义消息格式和协议细节-->
<binding name='cartSoap' type='tns:oplist'>
<!--style:属性可取值 "rpc" 或 "document",ransport:属性定义了要使用的 SOAP 协议.在这个例子中我们使用 HTTP-->
  <soap:binding style='rpc'
    transport='http://schemas.xmlsoap.org/soap/http'/>
    <!--operation 元素定义了每个端口提供的操作符,对于每个操作,相应的 SOAP 行为都需要被定义-->
  <operation name='test'>
    <soap:operation soapAction='http://www.cwtservice.cn/newOperation/'/>
    <input>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </input>
    <output>
      <soap:body use='encoded' namespace='urn:xmethods-delayed-quotes'
        encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'/>
    </output>
  </operation>
</binding>

<!--<service>包含一个或者多个port元素,每个port元素表示一个不同的Web服务-->
<service name='shopWS'>
  <port name='cartSoap' binding='tns:cartSoap'>
    <soap:address location='http://localhost/00/wss.php'/>
  </port>
</service>
</definitions>
```

Server端示例:

```
function test($x) {
    return $x;
}

$ss = new SoapServer('http://localhost/00/wsdl.xml');
$ss->addFunction('test');
$ss->handle();
```

Client调用:

```
$soap = new soapClient('http://localhost/00/wsdl.xml',array('trace'=>true));
var_dump($soap->test('10086'));
```

传递和返回数组参数

如果传递或返回的参数为数组,可以在message标签中做说明.

```
<message name='testRequest'>
    <part name="term" type="xsd:ArrayOfString"/>
</message>
<message name='testResponse'>
    <part name="value" type="xsd:ArrayOfString"/>
</message>
```

注:也可以不指定type,这样soap客户端和服务器端会自行判断

XML-RPC调用

XML-RPC可以理解为简化版的soap,对数据的包装相对简洁.
php.ini中,要打开extension=php_xmlrpc.dll

```
/*
求和函数
注意,rpc服务器在调用函数时,传的参数是这样的:
```

```
array(0=>'函数名' , 1=>array(实参1,实参2,...实参N) , 2=>NULL)
*/

function hello() {
    return 'hello';
}

function sum($method , $args , $extra) {
    return array_sum($args);
}

// 创建RPC Server
$server = xmlrpc_server_create ();
xmlrpc_server_register_method ($server , 'hello' , 'hello');
xmlrpc_server_register_method ($server , 'sum' , 'sum');

// 收取请求
$request = $HTTP_RAW_POST_DATA;

//执行调用客户端的XML请求后获取执行结果
$xmlrpc_response = xmlrpc_server_call_method($server, $request , null);
//把函数处理后的结果XML进行输出
header('Content-Type: text/xml');
echo $xmlrpc_response;

//销毁XML-RPC服务器端资源
xmlrpc_server_destroy($server);
```

客户端:

```
class rpcclient {
    protected $url;

    public function __construct($url='' ) {
        $this->url = $url;
    }

    protected function query($request) {
        $context = stream_context_create(array('http' => array(
        'method' => "POST",
        'header' => "Content-Type: text/xml",
        'content' => $request
        )));
        $xml = file_get_contents($this->url, false, $context);
        return xmlrpc_decode($xml);
    }

    public function __call($method , $args) {
        $request = xmlrpc_encode_request($method , $args);
        return $this->query($request);
    }

}

$rpc = new rpcclient('http://localhost/00/rpcs.php');
var_dump($rpc->hello());
var_dump($rpc->sum(4,5,6));
```

WebService与json Api的区别

比较	WebService	json API
数据封装	XML	json

复杂度	高	低
底层协议	不限	HTTP
数据类型	可严格定义	不可严格定义
自说明性	自说明	需额外API文档

WebService诞生十几年了,最初是IBM、微软比较热心在推,一直也不温不火.

究其原因,还是WebService实在太笨重了,SOAP信封格式书写麻烦,开发速度慢.
尤其是2000年以后,伴随着Web和互联网的快速发展,技术的变迁,无不向着"快速开发"发展.

因此,RESTful风格逐步流行.(简言之即用HTTP协议的PUT DELETE GET POST方法与服务器交换JSON).

再后来,大伙儿干脆连PUT、DELETE都懒得用,直接用GET和POST,并用JSON交流数据,就是现在常用的API

JSON比起XML,可读性要好很多,解析规则也简单.
XML解析的时候规则太多,这对追求高开发速度和低开发门槛的企业来说,是个硬伤.

但WebService也有其优势,比如:
XML可以对输入输出的类型做严格的说明,如int,double
而JSON则不能精确说明数据类型,比如:
`{price:12580}`
在json里,你无法知道这个价格是int, float还是double.

因此,在某些业务复杂,稳定性和正确性要求高的领域(如ERP,数学运算,天文运算),WebService还有是用武之地的.