Ruby XML, XSLT 和 XPath 教程 Ruby 多线程

# Ruby Web Service 应用 - SOAP4R

## 什么是 SOAP ?

简单对象访问协议(SOAP,全写为Simple Object Access Protocol)是交换数据的一种协议规范。

SOAP 是一种简单的基于 XML 的协议，它使应用程序通过 HTTP 来交换信息。

简单对象访问协议是交换数据的一种协议规范，是一种轻量的、简单的、基于XML（标准通用标记语言下的一个子集）的协议，它被设计成在WEB上交换结构化的和固化的信息。

更多 SOAP 教程请查看：http://www.w3cschool.cc/soap/soap-tutorial.html。

## SOAP4R 安装

SOAP4R 由Hiroshi Nakamura开发实现，用于 Ruby 的 SOAP 应用。

SOAP4R 下载地址：http://raa.ruby-lang.org/project/soap4r/。

**注意：**你的ruby环境可能已经安装了该该组件。

Linux 环境下你也可以使用 gem 来安装该组件，命令如下：

```
gem install soap4r --include-dependencies
```

如果你是window环境下开发，你需要下载zip压缩文件，并通过执行 install.rb 来安装。

## SOAP4R 服务

SOAP4R 支持两种不同的服务类型：

- 基于 CGI/FastCGI 服务 (SOAP::RPC::CGIStub)
- 独立服务 (SOAP::RPC:StandaloneServer)

本教程将为大家介绍如何建立独立的 SOAP 服务。步骤如下：

### 第1步 - 继承SOAP::RPC::StandaloneServer

为了实现自己的独立的服务器，你需要编写一个新的类，该类为 SOAP::RPC::StandaloneServer 的子类：

```
class MyServer < SOAP::RPC::StandaloneServer
.............
end
```

**注意：**如果你要编写一个基于FastCGI的服务器，那么你需要继承 SOAP::RPC::CGIStub 类，程序的其余部分将保持不变。

### 第二步 - 定义处理方法

接下来我们定义Web Service的方法，如下我们定义两个方法，一个是两个数相加，一个是两个数相除：

```
class MyServer < SOAP::RPC::StandaloneServer
.............
```

```
# 处理方法
def add(a, b)
return a + b
end
def div(a, b)
return a / b
end
end
```

## 第三步 - 公布处理方法

接下来添加我们在服务器上定义的方法，initialize方法是公开的，用于外部的连接：

```
class MyServer < SOAP::RPC::StandaloneServer
def initialize(*args)
add_method(receiver, methodName, *paramArg)
end
end
```

以下是各参数的说明：

| 参数 | 描述 |
| --- | --- |
| **receiver** | 包含方法名的方法的对象。 如果你在同一个类中定义服务方法，该参数为 *self*。 |
| **methodName** | 调用 RPC 请求的方法名。 |
| **paramArg** | 参数名和参数模式 |

为了理解 *inout* 和 *out* 参数，考虑以下服务方法，需要输入两个参数:inParam 和 inoutParam，函数执行完成后返回三个值：ret Val、inoutParam 、outParam:

```
def aMeth(inParam, inoutParam)
retVal = inParam + inoutParam
outParam = inParam . inoutParam
inoutParam = inParam * inoutParam
return retVal, inoutParam, outParam
end
```

公开的调用方法如下：

```
add_method(self, 'aMeth', [
%w(in inParam),
%w(inout inoutParam),
%w(out outParam),
%w(retval return)
])
```

## 第四步 - 开启服务

最后我们通过实例化派生类，并调用 start 方法来启动服务：

```
myServer = MyServer.new('ServerName',
'urn:ruby:ServiceName', hostname, port)
```

```
myServer.start
```

以下是请求参数的说明：

| 参数 | 描述 |
| --- | --- |
| **ServerName** | 服务名，你可以取你喜欢的 |
| **urn:ruby:ServiceName** | Here *urn:ruby* 是固定的，但是你可以为你的服务取一个唯一的 *ServiceName* |
| **hostname** | 指定主机名 |
| **port** | web 服务端口 |

## 实例

接下来我们通过以上的步骤，创建一个独立的服务：

**实例**

```
require "soap/rpc/standaloneserver"
begin
class MyServer < SOAP::RPC::StandaloneServer
# Expose our service
def initialize(*args)
add_method(self, 'add', 'a', 'b')
add_method(self, 'div', 'a', 'b')
end
# Handler methods
def add(a, b)
return a + b
end
def div(a, b)
return a / b
end
end
server = MyServer.new("MyServer",
'urn:ruby:calculation', 'localhost', 8080)
trap('INT){
server.shutdown
}
server.start
rescue => err
puts err.message
end
```

执行以上程序后，就启动了一个监听 8080 端口的本地服务，并公开两个方法：add 和 div。

你可以再后台执行以上服务：

```
$ ruby MyServer.rb &
```

# SOAP4R 客户端

ruby 中使用 SOAP::RPC::Driver 类开发 SOAP 客户端。接下来我们来详细看下 SOAP::RPC::Driver 类的使用。

调用 SOAP 服务需要以下信息：

- SOAP 服务 URL 地址 (SOAP Endpoint URL)
- 服务方法的命名空间（Method Namespace URI）
- 服务方法名及参数信息

接下来我们就一步步来创建 SOAP 客户端来调用以上的 SOAP 方法：add 、 div:

## 第一步 - 创建 SOAP Driver 实例

我们可以通过实例化 SOAP::RPC::Driver 类来调用它的新方法，如下所示：

```
SOAP::RPC::Driver.new(endPoint, nameSpace, soapAction)
```

以下是参数的描述：

| 参数 | 描述 |
| --- | --- |
| **endPoint** | 连接 SOAP 服务的 URL 地址 |
| **nameSpace** | 命名空间用于 SOAP::RPC::Driver 对象的所有 RPC . |
| **soapAction** | 用于 HTTP 头部的 SOAPAction 字段值。如果是字符串是"" 则默认为 *nil* |

## 第二步 - 添加服务方法

为 SOAP::RPC::Driver 添加 SOAP 服务方法，我们可以通过实例 SOAP::RPC::Driver 来调用以下方法：

```
driver.add_method(name, *paramArg)
```

以下是参数的说明：

| 参数 | 描述 |
| --- | --- |
| **name** | 远程web服务的方法名 |
| **paramArg** | 指定远程程序的参数 |

## 第三步 - 调用SOAP服务

最后我们可以使用 SOAP::RPC::Driver 实例来调用 SOAP 服务：

```
result = driver.serviceMethod(paramArg...)
```

serviceMethod SOAP服务的实际方法名，paramArg为方法的参数列表。

## 实例

基于以上的步骤，我们可以编写以下的 SOAP 客户端：

**实例**

```
#!/usr/bin/ruby -w
require 'soap/rpc/driver'
NAMESPACE = 'urn:ruby:calculation'
URL = 'http://localhost:8080/'
begin
driver = SOAP::RPC::Driver.new(URL, NAMESPACE)
# Add remote sevice methods
driver.add_method('add', 'a', 'b')
# Call remote service methods
puts driver.add(20, 30)
rescue => err
puts err.message
end
```

以上我们只是简单介绍 Ruby 的 Web Service 。 如果你想了解更多可以查看官方文档：Ruby 的 Web Service

← Ruby XML, XSLT 和 XPath 教程 Ruby 多线程 →