# 使用 XML-RPC 为 C++ 应用程序启用 Web 服务

将 C++ 方法作为服务公开的详细指南

XML-RPC 是一个简单而强大的轻量级消息传递协议，可支持基于 XML 的跨异类平台的通信。在本文中，您将了解如何为 C++ 程序构建基于 XML-RPC 的服务。

0 [评论](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#icomments)

[Karthik Subbian](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#authorN1001B) ([ksubbian@in.ibm.com](mailto:ksubbian@in.ibm.com?subject=使用%20XML-RPC%20为%20C++%20应用程序启用%20Web%20服务)), 高级软件工程师, IBM India

[Ramakrishnan Kannan](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#authorN10030) ([rkrishnan@in.ibm.com](mailto:rkrishnan@in.ibm.com?subject=使用%20XML-RPC%20为%20C++%20应用程序启用%20Web%20服务)), 高级软件工程师, IBM India

2006 年 9 月 18 日

* expand内容



在 IBM Bluemix 云平台上开发并部署您的下一个应用。

[开始您的试用](https://developer.ibm.com/sso/bmregistration?lang=zh_CN&ca=dwchina-_-bluemix-_-ws-xml-rpc-_-sidebar)

## 引言

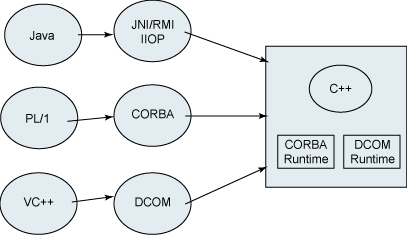
Internet 现在的受欢迎程度越来越高，由于这个原因及其固有的优势，促使开发人员和 IT 部门开始着手将复杂的 C/C++ 业务和科学应用程序向基于 Web 的环境迁移。简单对象访问协议（Simple Object Access Protocol，SOAP）、代表性状态传输（Representational State Transfer，REST）以及 XML 远程过程调用协议（XML Remote Procedure Call，XML-RPC）等 Web 服务协议可帮助将此类遗留应用程序集成到万维网，例如，可以使用 XML-RPC 作为将现有 C/C++ 程序与其他客户端技术集成的机制。这篇文章将帮助您确定何时选择 XML-RPC 而不使用 SOAP 与 REST。另外，本文还提供了详细步骤指南以及使用开放源代码 XML-RPC 库的 C++ 集成示例代码片段。

## 为什么选择 XML-RPC？

集 成 C/C++ 的挑战可以通过多种方式加以解决。C/C++ 代码集成的典型方法包括通过公共对象请求代理体系结构（Common Object Request Broker Architecture，CORBA）、分布式组件对象模型（Distributed Component Object Model，DCOM）、远程方法调用（Remote Method Invocation，RMI）Internet ORB互联协议（Internet Inter-ORB Protocol，IIOP）以及 Java™ 本机接口（Java™ Native Interface，JNI）等进行集成。

图 1 显示了利用现有 C++ 代码集成（使用上面提到的典型方法）的采用不同编程语言（Java、VC++、PL/1）开发的三个不同应用程序。

##### 图 1. 没有 XML-RPC 的当前方案



正如您看到的，C++ 代码应为每个 RMI/IIOP/JNI、CORBA 和 DCOM 客户端集成技术公开相应的接口。这就要求进行三次开发工作，显然会使部署和管理此类复杂接口的过程变得更为困难和麻烦。

在这些情况下，XML-RPC 是一个更好的选择，因为它可能帮助简化开发、部署和管理工作。

## 关于 XML-RPC 和 REST

XML-RPC 是 UserLand Software 的 Dave Winer 于 1998 年提出的。请访问 [Dave Winer 的博客](http://davenet.scripting.com/1998/02/27/rpcOverHttpViaXml)，以了解他关于 XML-RPC 的最初想法。

REST 最初是由 Roy Thomas Fielding 作为其在加州大学尔湾分校的[博士论文](http://www.ics.uci.edu/%7Efielding/pubs/dissertation/top.htm)的一部分提出的。

##### 图 2. 使用 XML-RPC

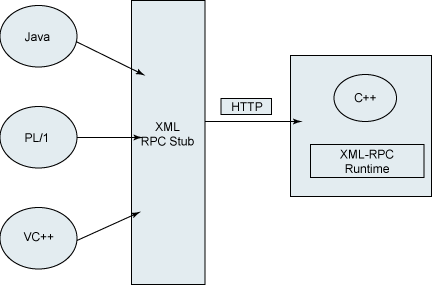


图 2 显示了如何使用 XML over HTTP 通过远程过程调用来调用 C++ 程序。SOAP 和 REST 等备选技术也能用于相同的目的。不过，您将在下一部分中了解到，这些技术之间存在一些关键区别。

## SOAP、XML-RPC 及 REST 间的对比

尽管这三个协议都支持 XML-RPC over HTTP，但就 C++ 而言，它们彼此之间是有区别的。表 1 提供了这些协议的各个元素间详细的比较。

##### 表 1. SOAP、XML-RPC 及 REST 间的对比

|  | SOAP | XML-RPC | REST |
| --- | --- | --- | --- |
| 定义 | SOAP 是用于在分散的分布式环境中进行信息交换的轻量级协议。该协议基于 XML，包括三个部分：信封、一组编码规则以及用于表示远程过程调用和响应的约定。 | 这是使用 HTTP 作为传输协议和使用 XML 作为编码方式的远程过程调用。XML-RPC 的设计力求简单，并同时允许传输、处理和返回复杂数据结构。 | 代表性状态传输旨在反映设计良好的 Web 应用程序的行为图像：由网页组成的网络，其中用户通过选择链接继续进行使用，选择链接将导致将下一页传输给用户并进行呈现，以供其使用。 |
| 目标 | SOAP 可实现用户定义的数据类型，提供指定接收者的功能、消息特定的处理控制以及其他功能，从而对 XML-RPC 进行了扩展。 | 非常简单且条理清楚的可扩展格式。HTML 编码人员应该能够查看包含 XML-RPC 过程调用的文件，理解其进行的工作，并能够对其进行修改，只需一两次尝试就能使其正常工作。此协议非常容易实现，可以快速对其进行调整，以在其他环境或其他操作系统上运行。 | 创建 REST 的目的是为了提供有关 Web 应该如何运行的设计模式，并作为 Web 标准和设计 Web 服务的指导框架。 |
| 支持的数据类型 | 整数、Boolean、ASCII 字符串、双精度带符号浮点数、日期时间、结构、数组、字节数组、枚举、用户定义数据类型、多态访问器 | 整数、Boolean、ASCII 字符串、双精度带符号浮点数、日期时间、结构、数组。 | 特定于实现。通常支持的类型有整数、Boolean、ASCII 字符串、双精度带符号浮点数、日期时间、集合、列表、属性。 |
| 简单性 | 比 XML-RPC 略微复杂 | 易于理解和进行开发 | 特定于实现 |
| 稳定性 | W3C 接受的标准 | 不是标准 | 体系结构参考。不需要标准 |
| 互操作性 | 无法与 REST/XML-RPC 互操作 | 无法与 REST/SOAP 互操作 | 无法与 SOAP/XML-RPC 互操作 |
| 工具 | 很多主要公司（包括 IBM 和 Microsoft）均已开始在其工具中支持 SOAP。 | 工具仍在开发中。 | 并没有很多工具支持。 |
| 自定义能力 | 高度可自定义，不受数据类型和协议限制 | 轻量级，只能在 HTTP 上工作，具有有限的数据类型支持 | 只能在 HTTP 上工作 |
| 库 | 有很多开放源代码库可用。 | 有很多开放源代码库可用。将在下面的部分进行说明 | 并没有很多实现库可用 |

##### 表 2. C++ 的各种 XML-RPC 实现

| 库和包名称 | 说明 |
| --- | --- |
| [PDEL](http://www.dellroad.org/pdel/index) | Packet Design Embedded Library 是一个 C 库，其中通过 http\_xml\_send\_xmlrpc 和 http\_servlet\_xmlrpc 方法包含了客户机和服务器的 XML-RPC 实现。这些函数将帮助通过 HTTP 传输协议发送自定义 XML 数据。这个包还包含很多 XML-RPC 之外的其他功能。 |
| [XMLRPC++](http://sourceforge.net/projects/xmlrpcpp) | 这是 XML-RPC 的 C++ 实现。其中提供了简单的服务器和客户机。通过使用面向对象的技术，我们可以集成这些服务器和客户机类，并实现我们自己的 XML-RPC 服务器，以将业务功能作为服务公开。在本文中，我们的示例实现及相关示例将使用这个库。 |
| [XMLRPC-C](http://sourceforge.net/project/showfiles.php?group_id=16847) | 这是一个 C 实现，可供 C 和 C++ 应用程序用于将这些方法作为服务公开。此包中包括一个 abyss Web 服务器。为了公开 C++ 方法，我们可以为所需的 C++ 方法编写 C 样式的包装，然后使用此库公开此方法。 |

## 安装 XML-RPC++ 库

为 C++ 程序启用 XML-RPC 的第一步是下载并安装 XML-RPC 库实现。为 C++ 程序提供了各种实现。可以在[参考资料](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#resources)部分获得一个指向更多实现的链接。

我们的示例程序将使用 XMPLRPC++ 实现。有关将此实现下载并安装到 Linux、AIX、32 位 Windows 平台以及其他类似平台的详细信息，请参阅[参考资料](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#resources)部分。我们的示例实现将基于 Red Hat 9 平台，使用的是 XML-RPC++ 0.7 库。

## 示例 C++ 应用程序

此处的示例应用程序是一个简单的两个整数相加的操作，将使用名为“operations”的用户定义类。清单 1 显示了 operations 类的代码片段。

## 您知道吗？

可以使用 Eclipse 作为开发 C/CPP 应用程序的 IDE。有关更多信息，请访问 [Eclipse CDT](http://www.eclipse.org/cdt/)。

##### 清单 1. Operations

class operations {

public:

int add();

operations(int i, int j);

private:

int op1;//Operand 1

int op2;//Operand 2

};

该类的构造函数接受两个整数参数，并将其分别设置为私有变量 op1 和 op2。该类的 add 方法如清单 2 中所示。这个方法就是要作为 XML-RPC 服务公开的方法。

##### 清单 2. Operations.cpp

int operations::add()

{

std::cout << "Sum of "<<op1<<" + "<<op2<<" = "<<op1+op2<<std::endl;

return(op1 + op2);

}

[回页首](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#ibm-pcon)

## XML-RPC 库的组件

在此部分，我们将使用一个类关系图来说明 XML-RPC 库的各个组件，并介绍其如何与我们服务器端的 operations 类通过接口连接。

##### 图 3. XML-RPC 库和示例应用程序的类关系图

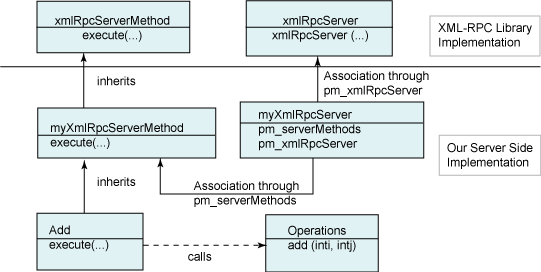


表 3 对每个类进行了详细说明。

##### 表 3. 类详细信息

| 类名称 | 用途 |
| --- | --- |
| Operations | 要公开的 add 方法在该类中实现 |
| Add | 调用 operations 的 add 方法的包装类。该类也从 myXmlRpcServerMethod 继承 |
| myXmlRpcServerMethod | 该类从 XML-RPC 库的 XmlRpcServerMethod 类继承。该类的 execute 将在 Add 类中通过继承覆盖。 |
| xmlRpcServermethod | 需 要向服务器注册的每个方法都必须通过 myXmlRpcServerMethod 类从该类进行继承，并实现自己的 execute 方法。这个 execute 方法将为公开的实际服务的包装。服务器收到 XML-RPC 调用时，将会直接触发此包装类的 execute 方法。在我们的示例中，Add 将为包装类，从客户端调用“Add”服务时，将调用其 execute 方法。 |
| myXmlRpcServer | 该类具有两个重要的私有变量   1. pm\_serverMethods：指向在服务器中注册的 myXmlRpcServerMethods 的指针列表。 2. pm\_xmlRpcServer：用于设置服务器 IP、端口和其他属性。   三个重要的方法   1. Class constructor：使用 IP/端口详细信息初始化服务器对象，并将其绑定。 2. pm\_registerMethods：创建指向 Add 类的指针对象，并将其加入列表 pm\_serverMethods 中。 3. run：xmlRpcServer 类中 work 方法的包装 |
| xmlRpcServer | 该类是创建服务器对象的 XML-RPC 服务器类。该类具有以下两个重要的方法   1. bindAndListen(port)：绑定并侦听指定的特定端口 2. work(...)：启动服务器 |

清单 3 显示了上表中每个类的每个 .cpp 文件的代码。由于 xmlRpcServer 和 xmlRpcServerMethod 是在 XML-RPC 库中实现的，因此我们将重点讨论剩下的四个类。

##### 清单 3. myXmlRpcServer.cpp

#include "myXmlRpcServer.h"

using namespace XmlRpc;

using namespace std;

myXmlRpcServer::myXmlRpcServer()

{

//call register methods

pm\_registerMethods();

//set port bind and listen

int port = 8085;

pm\_xmlRpcServer.bindAndListen(port);

std::cout<<"XmlRpcSever running in port "<<port<<std::endl;

}

void

myXmlRpcServer::pm\_registerMethods()

{

Add\* a=new Add(&pm\_xmlRpcServer);

myXmlRpcServerMethod \*p=a;

pm\_serverMethods.push\_back(p);

}

void

myXmlRpcServer::run()

{

pm\_xmlRpcServer.work(-1);

}

##### 清单 4. myXmlRpcServer.h

#include <iostream>

#include "myXmlRpcServerMethods.h"

#include "XmlRpc.h"

class myXmlRpcServer {

public:

myXmlRpcServer();

void run();

private:

void pm\_registerMethods();

XmlRpc::XmlRpcServer pm\_xmlRpcServer;

std::list< myXmlRpcServerMethod\* > pm\_serverMethods;

};

清单 5 和清单 6 显示了用于将方法注册到 XmlRpc Server（作为 XML-RPC 库的一部分提供）的类的代码。

##### 清单 5. myXmlRpcServerMethods.cpp

#include <iostream>

#include "myXmlRpcServer.h"

#include "operations.h"

using namespace std;

Add::Add(XmlRpcServer\* s) : myXmlRpcServerMethod("Add", s) {};

Void Add::execute(XmlRpcValue & params, XmlRpcValue& result)

{

operations a(10,12);

try

{

cout << "Inside Add::execute method\n";

result = a.add();

}

catch(std::exception & stde)

{

throw XmlRpcException(stde.what());

}

}

##### 清单 6. myXmlRpcServerMethods.h

class myXmlRpcServerMethod : public XmlRpcServerMethod

{

public:

myXmlRpcServerMethod

(const char \*name, XmlRpcServer \* server):XmlRpcServerMethod(name, server) {}

virtual void execute(XmlRpcValue & params, XmlRpcValue& result) {assert(0);}

};

class Add:public myXmlRpcServerMethod

{

public:

Add(XmlRpcServer\* s);

virtual void execute(XmlRpcValue & params, XmlRpcValue& result);

};

## 服务器驱动程序

服务器端的入口点将是服务器驱动程序。将从此处实例化 myXmlRpcServer 对象，并调用 run() 方法，而后者将最终启动服务器。

##### 清单 7. myServerDriver.cpp

#include <iostream>

#include "myXmlRpcServer.h"

int main(int argc, char\* argv[])

{

myXmlRpcServer GeeBoomBaa;

std::cout<<"About to run the server\n";

GeeBoomBaa.run();

return 0;

}

[回页首](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#ibm-pcon)

## 启动服务器

为了编译该代码，请记住包含 (XML\_RPC\_INSTALL\_DIR)/src 和 (XML\_RPC\_INSTALL\_DIR)/include 目录。为了进行链接，请包含 libXmlRpc.a 库。代码成功编译并链接后，将获得一个可执行文件，该文件就是 XML-RPC 服务器。在我们的示例实现中，服务器将运行于 localhost 上，并侦听端口 8085。这个设置硬编码在 myXmlRpcServer.cpp 文件中。也可以使程序读取配置文件，或者从命令行提示符将此作为参数传递给程序。运行成功进行了编译和链接后得到的 a.out 程序，以启动服务器。

[回页首](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#ibm-pcon)

## 示例客户机

示例客户机应该从 XML-RPC 库提供的 XmlRpcClient 类实例化一个对象。该类的“execute(...)”将实际接受三个参数：

1. 方法名称，const char\* 类型
2. 端口号，const int 类型
3. 可选的 URL 字符串，以作为 http get header 中的 URI 进行发送

清单 8 显示了一个示例客户机，此客户机将执行“add(...)”方法，并在客户端输出结果。

##### 清单 8. sampleClient.cpp

#include <iostream>

#include "XmlRpc.h"

using namespace XmlRpc;

int main(int argc, char\* argv[])

{

const char \*server = "localhost";

const int port = 8085;

const char \*uri = NULL;

XmlRpcValue args, res;

XmlRpcClient c( server, port, uri);

c.execute("Add", args, res);

std::cout<<"result is "<<res<<std::endl;

}

[回页首](http://www.ibm.com/developerworks/cn/webservices/ws-xml-rpc/#ibm-pcon)

## 结束语

XML-RPC 是一个简单而强大的轻量级消息传递协议，可支持基于 XML 的跨异类平台通信。此标准固有的简单性在将遗留应用程序与企业集成方面表现得非常强大而实用。由于各个 XML-RPC 实现都是开放源代码的，使得此技术在企业应用程序集成领域越来越受欢迎。随着越来越多的成熟 XML-RPC 工具的出现，我们可以预见，在不久的将来，此技术将成为企业内部集成的“事实”标准。