# c/c++的Soap应用

## 关于soap

在许多项目中团队中，我们常常会听到这样的话：我们这里是用webservice交互的。而说话的场景往往就是交互对象双方比较异构，所谓异构、即双方是不同的开发语言、不同的运行环境等。比如常见的c/c++后台程序与java的web程序间的通信，当然这里的通信是网络通信，如果是一体化单机系统内，可能第一反应是JNI方式了。

异构体系间的通信，就是webservice的基本应用场景。而soap（simple object access protocal）则是webservice在实际操作中需要遵守的协议，webservice的其它关键元素还有：WSDL, http, xml等。

## 适用情况

如前所述、在一个项目中不同开发体系间希望交互信息，同时不愿花大气力自己做通信框架，产品需求定位为：快捷开发、稳定有效，没有明显大并发的需求，那么webservice是一个很好的选择。

之所以排除大并发的应用场景，个人觉得大并发平台网络环境复杂，在接口伸缩性和问题定位方面，一般都是公司内部封装通信框架更合适。另外webservice通信中携带的冗余信息太多也是一大诟病，所以大并发的场景，想想也算了。

## C++ gsoap工具

Gsoap是c/c++在webservice开发中一个强大工具，c/c++er在做webservice开发一般都是用的这个利器。

从网上很容易找到gsoap的源码，以我下载的gsoap-2.8为例。如果不用特别的研究编译工具源码，那么下载后需要用的东西是两个地方：

1. 在\gsoap-2.8\gsoap\bin\win32目录下有两个编译工具：wsdl2h.exe和soapcpp2.exe。

**wsdl2h.exe**：用于将wsdl文件转换为c/c++使用的头文件。

**soapcpp2.exe**：用于将上述头文件转换为c/c++项目使用的基础代码。包括客户端代码、服务端代码、头文件的wsdl描述。

1. 在\gsoap-2.8\gsoap目录下有两个文件：stdsoap2.cpp和stdsoap2.h。这两个文件即为c++使用webservice通信的底层soap协议实现。

## C++应用

对于C++来说，webservice就是一种RPC（Remote Procedure Call Protocol远程过程调用协议）方式，既然是RPC，通俗的说、就是本地C++需要用到异地环境的方法，那么本地与异地双方就有一个基本的方法列表，之后达到就像在调用本地方法一样的调用异地方法。由此引出C++在webservice开发的基本过程：

1. **C++服务端开发：**
2. 列出能够提供的方法接口，写入头文件；
3. 用gsoap的soapcpp2.exe编译工具将1)中头文件编译生成服务端代码；
4. 将2）中生成的代码引入到自己的服务端项目中，进行服务端业务开发，需要注意的是服务端必须实现1)中头文件定义的方法。
5. **c++客户端开发：**
6. 如果有服务方提供的wsdl文件，则用gsoap提供的wsdl2h.exe工具生成头文件，然后同c++服务端开发的前两步一样，生成客户端代码；

如果C/S双方都是C++开发，那么可以不需要wsdl的“介绍”，直接在上述c++服务端开发2)中，同时生成客户端代码，拿到这里用即可。

1. 将上述gsoap框架下的客户端代码引入到自己的客户端项目中，就可以调用服务端方法了。
2. **归纳：**

C++的webservice开发，如果自己玩，可以不需要wsdl，但如果与其它体系一起协同通信，就需要wsdl（网络服务描述语言）来描述头文件的那些方法列表。Soap协议使得通信双方不需要关心具体通信实现，双发维护好提供业务的方法即可。

Gsoap是一个开源的soap封装，从stdsoap2.cpp中可以看出其跨平台的实现，比如“#ifdef WIN32”这样的痕迹。stdsoap2.h中代码风格有值得学习的地方，比如条件预处理格式等，一个头文件和一个cpp就实现了soap的协议封装，短小精悍。\gsoap-2.8\gsoap\doc中文档介绍朴素简洁，建议以html方式查看，一目了然。

## 5. 简单例子

本例子通过服务端提供一个字符串置反的方法，之后通过SoapUI测试客户端调用。

**C++服务端：**

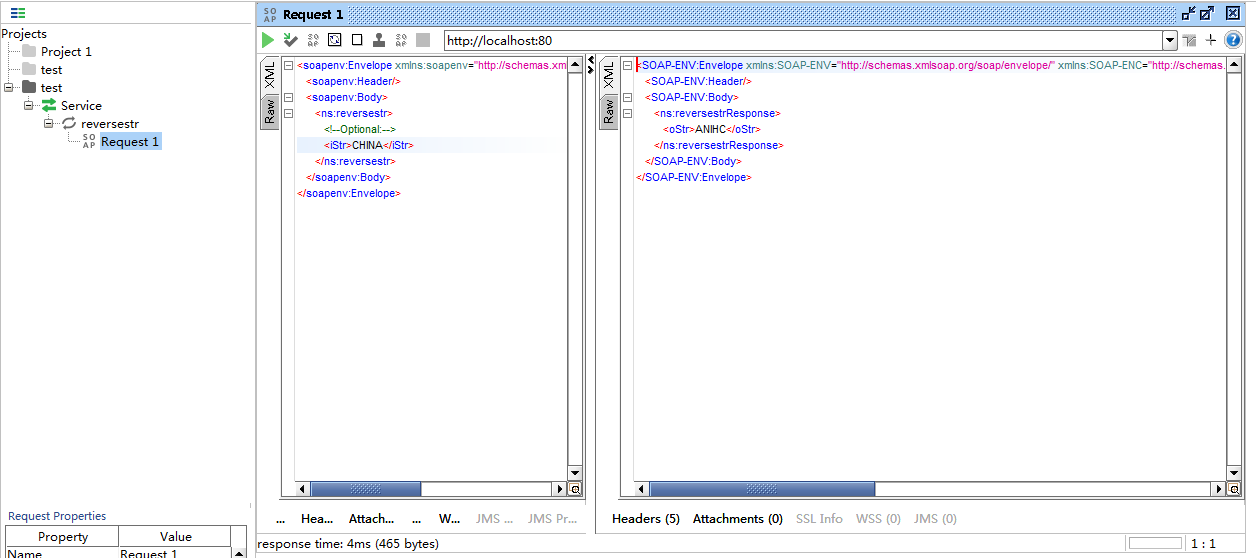
服务端工程，根据前面所述C++服务端开发步骤，首先给出头文件reverse.h

|  |
| --- |
| int ns\_\_reversestr(char \*iStr, char \*\*oStr); |

然后就只有一个main.cpp

|  |
| --- |
| #include "soapService.h"  //服务方法的实现  int Service::reversestr(char \*iStr, char \*\*oStr)  {  if (NULL == iStr)  {  return this->error;  }  int strLen = strlen(iStr);  \*oStr = (char\*)soap\_malloc(this, strLen + 1);  memset(\*oStr, 0, strLen + 1);  char \*pOutBuf = \*oStr;  while (strLen-- > 0)  {  \*(pOutBuf++) = \*(iStr + strLen);  }  return this->error;  }  //服务入口，这里是最基本的服务形式  int main()  {  Service serv;  serv.serve();  int port = 80; //服务端口，启用前先用netstat查看下该端口是否被占用  if (serv.run(port))  {  serv.soap\_stream\_fault(std::cerr);  exit(-1);  }  return 0;  } |

**SoapUI测试：**

****