用C++实现WebService(ZZ)

作者：魏琼
来自：linux宝库 (http://www.linuxmine.com)
联系：weiqiong#gmail.com

一．系统环境

linux操作系统kernel2.4.2，安装gsoap2.6到目录/usr/local/gsoap

二．gSOAP的简要使用例子

下面是一个简单的例子，实现一个加法运算的WebService，具体功能是cli端输入num1和num2，server端返回一个num1和num2相加的结果sum。

1． 首先，我们需要做的是写一个函数声明文件，来定义接口函数ns__add，文件名字为add.h，内容如下：

```
//gsoap ns service name: add
//gsoap ns service namespace: http://mail.263.net/add.wsdl
//gsoap ns service location: http://mail.263.net
//gsoap ns service executable: add.cgi
//gsoap ns service encoding: encoded
//gsoap ns schema namespace: urn:add

int ns__add( int num1, int num2, int* sum );
```

系分考试
札记

存档

2009年09月(1)
2006年12月(1)
2006年11月(1)
2006年09月(3)
2006年07月(1)
2006年05月(1)
2006年04月(30)
2006年03月(6)
2006年02月(6)
2005年12月(2)
2005年11月(9)
2005年10月(6)
2005年09月(5)
2005年03月(6)
2005年01月(55)
2004年09月(2)

2． 然后我们需要创建文件Makefile，从而利用gsoapcpp2工具由add.h生成一些.xml文件、.c文件和.h文件，这些文件均为自动生成，Makefile的内容如下：

```
GSOAP_ROOT=/usr/local/gsoap
WSNAME=add
CC=g++ -g -DWITH_NONAMESPACES
INCLUDE=-I $(GSOAP_ROOT)
SERVER_OBJS=$(WSNAME)C.o $(WSNAME)Server.o stdsoap2.o
CLIENT_OBJS=$(GSOAP_ROOT)/env/envC.o $(WSNAME)ClientLib.o stdsoap2.o
ALL_OBJS=${WSNAME}server.o $(WSNAME)C.o $(WSNAME)Server.o ${WSNAME}test.o ${WSNA
ME}client.o $(WSNAME)ClientLib.o

#总的目标
all:server

${WSNAME}.wsdl:${WSNAME}.h
$(GSOAP_ROOT)/soapcpp2 -p$(WSNAME) -i -n -c ${WSNAME}.h

stdsoap2.o:$(GSOAP_ROOT)/stdsoap2.c
$(CC) -c $?

#编译一样生成规则的.o文件
$(ALL_OBJS):%.o:%.c
$(CC) -c $? $(INCLUDE)

#编译服务器端
server:Makefile ${WSNAME}.wsdl ${WSNAME}server.o $(SERVER_OBJS)
$(CC) ${WSNAME}server.o $(SERVER_OBJS) -o ${WSNAME}server

#编译客户端
client:Makefile ${WSNAME}.wsdl ${WSNAME}client.c ${WSNAME}test.c $(ALL_OBJS) stdsoap2.o
$(CC) ${WSNAME}test.o ${WSNAME}client.o $(CLIENT_OBJS) -o ${WSNAME}test

cl:
rm -f *.o *.xml *.a *.wsdl *.nsmap $(WSNAME)H.h $(WSNAME)C.c $(WSNAME)Server.c $(WSNA
ME)Client.c $(WSNAME)Stub.* $(WSNAME)$(WSNAME)Proxy.* $(WSNAME)$(WSNAME)Object.* $
(WSNAME)ServerLib.c $(WSNAME)ClientLib.c $(WSNAME)server ns.xsd $(WSNAME)test
```

3．我们先来做一个server端，创建文件addserver.c文件，内容如下：

```
#include "addH.h"
#include "add.nsmap"

int main(int argc, char **argv)
{
int m, s; /* master and slave sockets */
struct soap add_soap;
soap_init(&add_soap);
soap_set_namespaces(&add_soap, add_namespaces);
if (argc < 2)
{
printf("usage: %s <server_port> \n", argv[0]);
exit(1);
}
else
{
m = soap_bind(&add_soap, NULL, atoi(argv[1]), 100);
if (m < 0)
{
soap_print_fault(&add_soap, stderr);
exit(-1);
}
fprintf(stderr, "Socket connection successful: master socket = %d\n", m);
for ( ; ; )
{
s = soap_accept(&add_soap);
if (s < 0)
{
soap_print_fault(&add_soap, stderr);
exit(-1);
}
fprintf(stderr, "Socket connection successful: slave socket = %d\n", s);
add_serve(&add_soap);//该句说明该server的服务
soap_end(&add_soap);
```

```
}
}
return 0;
}
//server端的实现函数与add.h中声明的函数相同，但是多了一个当前的soap连接的参数
int ns__add(struct soap *add_soap, int num1, int num2, int *sum)
{
*sum = num1 + num2;
return 0;
}
```

4．让我们的server跑起来吧：

```
shell>make
shell>./addserver 8888
```

如果终端打印出“Socket connection successful: master socket = 3”，那么你的server已经在前台run起来了，应该是值得高兴的 。

打开IE，键入http://本机IP:8888，显示XML，服务已经启动，终端打印出“Socket connection successful: slave socket = 4”，表示服务接收到了一次soap的连接。

5．让我们再来写个客户端（这个只是将soap的客户端函数封装一下，具体的调用参见下面的addtest.c），创建文件addclient.c，内容如下：

```
#include "addStub.h"
#include "add.nsmap"
/**
* 传入参数：server：server的地址
* num1,num2：需要相加的数
* 传出参数：sum：num1和num2相加的结果
* 返回值：0为成功，其他为失败
*/
int add( const char* server, int num1, int num2, int *sum )
{
struct soap add_soap;
int result = 0;
soap_init(&add_soap);
soap_set_namespaces(&add_soap, add_namespaces);
```

```
//该函数是客户端调用的主要函数，后面几个参数和add.h中声明的一样，前面多了3个参数，函数名是接口函数名ns__add前面加上soap_call_
soap_call_ns__add( &add_soap, server, "", num1, num2, sum );
if(add_soap.error)
{
printf("soap error:%d,%s,%s\n", add_soap.error, *soap_faultcode(&add_soap), *soap_faultstring(&add_soap) );
result = add_soap.error;
}
soap_end(&add_soap);
soap_done(&add_soap);
return result;
}
```

6．我们最终写一个可以运行的客户端调用程序，创建文件addtest.c，内容如下：

```
#include <stdio.h>
#include <stdlib.h>

int add(const char* server, int num1, int num2, int *sum);

int main(int argc, char **argv)
{
int result = -1;
char* server="http://localhost:8888";
int num1 = 0;
int num2 = 0;
int sum = 0;
if( argc < 3 )
{
printf("usage: %s num1 num2 \n", argv[0]);
exit(0);
}

num1 = atoi(argv[1]);
num2 = atoi(argv[2]);
```

```
result = add(server, num1, num2, &sum);
if (result != 0)
{
printf("soap err,errcode = %d\n", result);
}
else
{
printf("%d+%d=%d\n", num1, num2, sum );
}
return 0;
}
```

7．让我们的client端和server端通讯

```
shell>make client
shell>./addtest 7 8
```

当然，你的server应该还在run，这样得到输出结果7+8=15，好了，你成功完成了你的第一个C写的WebService，恭喜。

三．图示说明

四．要注意的问题

1． add.h文件前面的几句注释不能删除，为soapcpp2需要识别的标志

2． 接口函数的返回值只能是int，是soap调用的结果，一般通过soap.error来判断soap的连接情况，这个返回值没有用到。

3． 接口函数的最后一个参数为传出参数，如果需要传出多个参数，需要自己定义一个结构将返回项封装。

4． 在.h文件中不能include别的.h文件，可能不能生效，需要用到某些结构的时候需要在该文件中直接声明。

5． 如果客户端的调用不需要返回值，那么最后一个参数

五．参考文档

1．gsoap主页

http://gsoap2.sourceforge.net

2．跟我一起写Makefile

http://dev.csdn.net/develop/article/20/20025.shtm

3．Web Services： A Technical Introduction（机械工业出版社）

六．备注

192.168.18.233和192.168.18.234的/usr/local/gsoap目录下的3个需要的文件及一个env目录，不是编译安装的，是在别的地方编译好了直接copy过来的（实际编译结果中还有wsdl2h工具及其他一些文件，但是我们的实

际开发中只是用到了这3个文件及env目录）。因为时间仓促，本人还没有时间研究编译的问题，相关细节可以查看参考文档1。
在192.168.18.233的/home/weiqiong/soap/sample目录下及192.168.18.234的/tmp/soap/sample目录下有本文讲到的加法运算的例子。

发表于 @ 2005年11月29日 10:04:00 | 评论(1) | 举报 | 收藏

旧一篇:微软Live服务的一些测试签到地址及已经推出的测试网址汇总 | 新一篇:实现COM组件向Web Services的转变

lunareye 发表于Wednesday, November 30, 2005 11:41:00 举报 回复

一□□□
□较□

发表评论

表 情:

评论内容:

用 户 名: 匿名用户 登录 注册

验 证 码: CCSJ 重新获得验证码

发表评论