Practica 2: RPC Apache Thrift

Javier Béjar Méndez

Compilación

Para compilar el servidor en cpp usamos:

```
[zes@osbees gen-cpp]$ g++ calculadora_constants.cpp -std=c++11 Calculadora_serve
r.cpp calculadora_types.cpp Calculadora.cpp -lthrift -I/usr/include/thrift -o se
rver
```

Generación

calculadora.thrift

```
1  namespace cpp calculadora
2  namespace py calculadora
3
4  struct parametros{
5   1:double param_1;
6   2:double param_2;
7  }
8
9  service Calculadora{
10   double suma(1:parametros params);
11   double mul(1:parametros params);
12   double div(1:parametros params);
13   double res(1:parametros params);
14 }
```

Definimos los lenguajes que queremos generar con “namespace lenguaje nombre”, luego la estructura para los parámetros de entrada y finalmente las funciones que queremos implementar. Mediante el comando thrift -gen cpp calculadora.thrift y thrift -gen py calculadora.thrift generamos las versiones en cpp y python respectivamente. En el caso de cpp se genera un skeleton del servidor donde definiremos las operaciones, para python creamos el archivo de cliente.

Prueba

Cliente en python(izq) y servidor en cpp(derecha):

Explicación

Para el servidor en cpp lo único que hay que hacer es definir las funciones en Calculadora_server.cpp:

```cpp
double suma(const parametros& params) {
  // Your implementation goes here
  double result = params.param_1 + params.param_2;
  printf("%f + %f = %f\n", params.param_1, params.param_2, result);

  return result;
}

double mul(const parametros& params) {
  // Your implementation goes here
  double result = params.param_1 * params.param_2;
  printf("%f * %f = %f\n", params.param_1, params.param_2, result);

  return result;
}
```

En el cliente de python hay que crea el archivo cliente.py e importar lo necesario, estableciendo la conexión en el puerto especificado:

```python
from calculadora import Calculadora
from thrift import Thrift
from thrift.transport import TSocket
from thrift.transport import TTransport
from thrift.protocol import TBinaryProtocol
from calculadora.ttypes import *

transport = TSocket.TSocket ('localhost', 9090)
transport = TTransport.TBufferedTransport ( transport )
protocol = TBinaryProtocol.TBinaryProtocol ( transport )
# creamos el cliente
client = Calculadora.Client ( protocol )

transport.open ()
params = parametros(1, 1)
resultado = client.suma(params)
print (str(params.param_1) + " + " + str(params.param_2) + " = " + str(resultado))

params = parametros(2, 3)
resultado = client.mul(params)
print (str(params.param_1) + " * " + str(params.param_2) + " = " + str(resultado))

params = parametros(2, 3)
resultado = client.div(params)
print (str(params.param_1) + " / " + str(params.param_2) + " = " + str(resultado))

params = parametros(2, 3)
resultado = client.res(params)
print (str(params.param_1) + " - " + str(params.param_2) + " = " + str(resultado))

transport.close ()
```