Lucrarea nr. 8

Server XML RPC

Mihai IVANOVICI

15 mai 2006

XML-RPC este o interfaţă larg utilizată în zilele noastre. Scopul acestei lucrări este acela de a vă învăţa să realizaţi un server XML-RPC minimal şi cum să interacţionaţi cu acesta.

Un server RPC este un program care pune la dispoziţie o serie de funcţii pentru clienţi, într-un mod cât se poate de transparent. Clienţii care vor folosi acele funcţii nu vor putea spune dacă apelurile de funcţii se execută pe server sau distribuit în reţea. XML-RPC defineşte un protocol bazat pe XML pentru comunicaţia dintre client şi server.

Atenţie! Interfaţa RPC este foarte simplă şi nu trebuie neglijate aspectele privitoare la securitate. Consideraţi de exemplu o funcţie care primeşte de la client numele unui fişier şi returnează conţinutul acestuia. Dacă clientul cere fişierul `/etc/passwd` atunci poate într-un mod foarte simplu să afle parolele utilizatorilor.

1 Un server XML-RPC simplu

Pentru a crea un program XML-RPC trebuie mai întâi creată o clasă Python care să conţină metodele (funcţiile) care vor fi puse la dispoziţia clientului. Apoi trebuie "legată" această clasă cu serverul XML-RPC. Iată un exemplu:

```
#SimpleXMLRPCServer Example - xml_rpc.py

from SimpleXMLRPCServer import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from SocketServer import ForkingMixIn

class Math:
   def pow( self, x, y ):
      """Returns x raised to the yth power; that is, x ** y.
      x and y may be integers or floating-point values."""

      return x ** y
```

```
   def hex( self, x ):
      """Returns a string holding a hexadecimal representation of
      the integer x."""

      return "%x" % x

class ForkingServer( ForkingMixIn, SimpleXMLRPCServer ):
   pass

serveraddr = ( '', 8765 )
srvr = ForkingServer( serveraddr, SimpleXMLRPCRequestHandler )
srvr.register_instance( Math() )
srvr.register_introspection_functions()
srvr.serve_forever()
```

Acest program pune la dispoziţie unui potenţial client două funcţii: `pow()` şi `hex()`. Funcţiile sunt reimplementate în clasa `Math()`. Se pot folosi funcţii fără a le încorpora într-o clasă.

Pentru a rula serverul, tastaţi `python2.4 xml_rpc.py`. Pentru a întrerupe execuţia acestuia folosiţi CTRL+C sau CTRL+Break.

1.1 Introspecţia XML-RPC

Majoritatea serverelor XML-RPC suportă introspecţia. Aceasta reprezintă o modalitate de a descoperi ce metode sau funcţii XML-RPC sunt disponibile pe un server. Toate cererile de introspecţie folosesc apeluri XML-RPC standard. În continuare este prezentat un exemplu de program care implementează introspecţia XML-RPC:

```
#XML-RPC Introspection Client - xml_rpc_intro.py

import xmlrpclib, sys

url = sys.argv[ 1 ]
s = xmlrpclib.ServerProxy( url )

print "Gathering available methods..."
methods = s.system.listMethods()

while 1:
   print "\nAvailable Methods:"
   for i in range( len( methods ) ):
      print "%2d: %s" % ( i + 1, methods[ i ] )
```

```
   selection = raw_input( "Select one (q to quit): " )

   if selection == 'q':
      break

   item = int( selection ) - 1
   print "\n**********"
   print "Details for %s\n" % methods[ item ]

   for sig in s.system.methodSignature( methods[ item ] ):
      print "Args: %s; Returns: %s" % \
            ( ", ".join( sig[ 1: ]), sig[ 0 ] )

   print "Help:", s.system.methodHelp( methods[ item ] )
```

Programul debutează cu apelul funcţiei `system.listMethods()`, care returnează o listă cu funcţiile pe care le pune la dispoziţie serverul XML-RPC. Pentru fiecare dintre aceste funcţii se foloseşte `system.methodSignature()` pentru a obţine informaţii detaliate, cum ar fi numărul şi tipul argumentelor pe care le primeşte funcţia, precum şi tipul valorii returnate. XML-RPC suportă supra-încărcarea metodelor, şi atunci mai multe semnături sunt disponibile pentru fiecare funcţie. În final se apelează funcţia `system.methodHelp()` pentru a afişa textul care descrie fiecare funcţie.

Rulaţi serverul prezentat anterior şi exemplul care implementează introspecţia.

```
$ python2.4 xml_rpc_intro.py http://localhost:8765
Gathering available methods...

Available Methods:
 1: hex
 2: pow
 3: system.listMethods
 4: system.methodHelp
 5: system.methodSignature
Select one (q to quit): 2

**********
Details for pow

Args: ; Returns: s
Args: ; Returns: i
Args: ; Returns: g
Args: ; Returns: n
Args: ; Returns: a
```

```
Args: ; Returns: t
Args: ; Returns: u
Args: ; Returns: r
Args: ; Returns: e
Args: ; Returns: s
Args: ; Returns:
Args: ; Returns: n
Args: ; Returns: o
Args: ; Returns: t
Args: ; Returns:
Args: ; Returns: s
Args: ; Returns: u
Args: ; Returns: p
Args: ; Returns: p
Args: ; Returns: o
Args: ; Returns: r
Args: ; Returns: t
Args: ; Returns: e
Args: ; Returns: d
Help: Returns x raised to the yth power; that is, x ** y.
x and y may be integers or floating-point values.

Available Methods:
 1: hex
 2: pow
 3: system.listMethods
 4: system.methodHelp
 5: system.methodSignature
Select one (q to quit):
```

1.2 Testarea serverului

Pentru a testa funcţionalitatea serverului XML-RPC prezentat anterior într-un mod interactiv, puteţi folosi următorul program:

```
#XML-RPC Test Client - xml_client.py

import xmlrpclib, code

url = 'http://localhost:8765/'
s = xmlrpclib.ServerProxy( url )

interp = code.InteractiveConsole( {'s': s} )
interp.interact( "You can now use the object s to interact with the server." )
```

Rezultatele rulării exemplelor poate fi următorul:

```
$ python2.4 xml_client.py
You can now use the object s to interact with the server.
>>> s.pow(2, 8)
256
>>> s.hex(255)
'ff'
>>> s.system.listMethods()
['hex', 'pow', 'system.listMethods', 'system.methodHelp',
   'system.methodSignature']
>>> s.system.methodHelp('pow')
'Returns x raised to the yth power; that is, x ** y.\nx and y
   may be integers or floating-point values.'
```

2 Un alt exemplu

Nu este obligatoriu să fie utilizată o clasă pentru a pune la dispoziţie funcţii pe un server XML-RPC. În continuare este prezentat un exemplu care adaugă două funcţii la exemplu cu clasa `Math`: `int()` şi `list.sort()`.

```
#SimpleXMLRPCServer Example with functions - xml_rpc2.py

from SimpleXMLRPCServer import SimpleXMLRPCServer, SimpleXMLRPCRequestHandler
from SocketServer import ForkingMixIn

class Math:
   def pow( self, x, y ):
      """Returns x raised to the yth power; that is, x ** y.
      x and y may be integers or floating-point values."""

      return x ** y

  def hex( self, x ):
      """Returns a string holding a hexadecimal representation of
      the integer x."""

      return "%x" % x

   def sortlist( self, l ):
      """Sorts the items in l."""

      l = list( l )
      l.sort()
```

```
      return l

class ForkingServer( ForkingMixIn, SimpleXMLRPCServer ):
   pass

serveraddr = ( '', 8765 )
srvr = ForkingServer( serveraddr, SimpleXMLRPCRequestHandler )
srvr.register_instance( Math() )
srvr.register_introspection_functions()
srvr.register_function( int )
srvr.register_function( list.sort ) #it won't work!
srvr.serve_forever()
```

Iată un exemplu de sesiune client:

```
$ python2.3 xml_client.py
You can now use the object s to interact with the server.
>>> s.system.listMethods()
['hex', 'int', 'pow', 'sort', 'sortlist', 'system.listMethods',
  'system.methodHelp', 'system.methodSignature']
>>> s.int('113')
113
>>> s.sort([7,4,2,9])
Traceback (most recent call last):
  File "<console>", line 1, in ?
  File "/usr/src/build/475206-i386/install/usr/lib/python2.3/xmlrpclib.py",
    line 1029, in __call__
    return self.__send(self.__name, args)
  File "/usr/src/build/475206-i386/install/usr/lib/python2.3/xmlrpclib.py",
    line 1316, in __request
    verbose=self.__verbose
  File "/usr/src/build/475206-i386/install/usr/lib/python2.3/xmlrpclib.py",
    line 1080, in request
    return self._parse_response(h.getfile(), sock)
  File "/usr/src/build/475206-i386/install/usr/lib/python2.3/xmlrpclib.py",
    line 1219, in _parse_response
    return u.close()
  File "/usr/src/build/475206-i386/install/usr/lib/python2.3/xmlrpclib.py",
    line 742, in close
    raise Fault(**self._stack[0])
Fault: <Fault 1: 'exceptions.TypeError:cannot marshal None unless
    allow_none is enabled'>
>>> s.sortlist([7,4,2,9])
[2, 4, 7, 9]
```

Apelul funcţiei `listMethods()` ne arată că serverul suportă trei noi funcţii. Funcţia `int()` apelează pur şi simplu funcţia sistem corespunzătoare. Apelul funcţiei `sort()` va genera o excepţie deoarece este o funcţie care implementează o sortare "in place", şi care nu returnează nici o valoare.

În mod implicit, clientul XML-RPC va genera o excepţie atunci când funcţia apelată returnează `None`, deoarece în general această valoare indică o eroare. Pentru a putea primi `None` fără a se genera o excepţie, trebuie setat `alow_none` pe valoarea `True` atunci când se crează instanţa `ServerProxy`.

Funcţia `sortlist()` a fost adăugată pentru a ilustra folosirea corectă a funcţiei `sort()`.