Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

xml-rpc: fem-ho fàcil (7363 lectures)

Per **Antoni Aloy López**, ***aaloy*** (http://trespams.com)

Creado el 03/08/2003 22:20 modificado el 03/08/2003 22:20

Per què complicar-nos la vida amb arquitectures complexes? Sovint tenim maneres de fer les coses que no aprofitam i que són molt més senzilles. L'aprofitament de la potència de l'xml-rpc n'és una.

Introducció

Fa unes quantes setmanes vaig llegir un article, ho sento no record la font, on venia a dir que potser ens estavem complicam massa la vida amb això de soap, corba i tecnologies similars. Que per què no treiem el suc a tecnologies més senzilles. En poques paraules, **si podem per què no seguir la regla KISS**?

En aquest article intentaré explicar-vos com aplicar una d'aquestes tecnologies senzilles, l'xml-rpc per tal de crear una aplicació multicapa d'una manera molt senzilla. L'objectiu, a més és tenir una manera de poder desenvolupar aplicacions mixtes, on hi pugui haver una part d'interficie d'usuari via web i una altra part, pels usuaris dins la xarxa local, típicament, amb un tipus d'interfície gràfica o texte més amigable i/o ràpida.

Objectiu

Volem desenvolupar una aplicació que tendrà una interfície web i una interfície gràfica, de manera que no tenguem que duplicar el codi de la lògica del negoci i sense complicar-ne la programació de manera excessiva.

Per tal de complir aquest objectiu, montarem la nostra aplicació damunt un servidor Zope, i implementarem la lògica del negoci mitjançant scripts python dins el servidor, de manera que hi podrem accedir per xml-rpc tant des del navegador web com des de la interfície gràfica o texte de l'aplicatiu.

Requeriments

Per posar en marxa un aplicatiu com el que desenvoluparem necessitarem un servidor Zope[1], el llenguatge de programació Python[2] i les llibreries PyQt[3] encara que la seva instal·lació no és imprescindible per a entendre l'esperit de l'article.

La comunitat Zope posa al nostre abast un servidor gratuït a Internet, un tant limitat, però que en servirà per al nostre exeple. Ho podeu trobar aquí[4] i és el que jo faré servir en l'exemple.

Anem per feina!

Desenvoluparem una aplicació que el que farà serà sumar dos nombres i ens retornarà el resultat. Farem que això mateix ho poguem fer des de la web, des d'un aplicatiu de consola i des d'una típica aplicació Qt. L'exemple és força trivial, però és la base de la tecnologia que utilitzarem. Senzillesa i potència!

Creació de l'script a zope

Dins Zope, he creat una carpeta anomenada tests o hi posarem la part de Zope d'aquest mini-aplicatiu nostre. El primer que he creat és un script python anomenat *sumar* amb paràmetres *a* i *b*.

```
# Suma dos objectes Python si es pot fer, si no
```

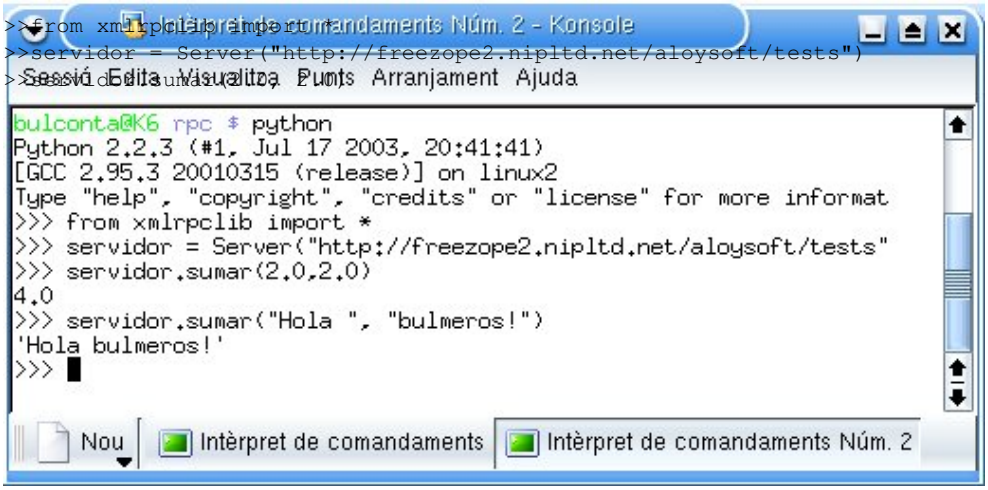
```
# retorna un missatge d'error
request = container.REQUEST
RESPONSE =  request.RESPONSE

try:
   rta = a + b
except:
   rta = "Error! Mala sort!"

return rta
```

Ja hem acabat!

Doncs gairebé. Amb aquest petit script que hem fet podem fer que el nostre servidor Zope ens faci de calculadora. Anau a la consola Python i teclejau:



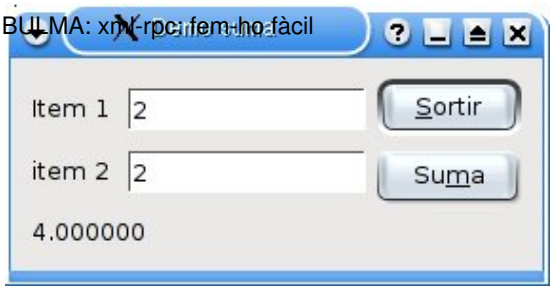
Uep! Que us pareix?, funciona i tot! Ja hem feta la nostra primera aplicació xml-rpc, fitxau-vos en la captura de pantalla. Com que no he fet cap suposició damunt el contingut de les variables a sumar, el codi Python ens serveix tant per sumar nombres, com cadenes, com llistes, etc.

La part Qt

Per fer la interfície d'usuari he fet servir PyQt, Eric el Qt-Designer. Com podeu comprovar no m'he complicat massa la vida, es tracta de fer el mateix que he fet per consola. El codi és força semblant, llevat de que en aquest cas imposam el que volem sumar dos nombres.

La feina principal ha esta crear la interfície amb el dissenyador, convertir l'arxiu .ui en codi Python i crear el mètode sumaBtn_clicked que és el que farà la feina.

```
def sumaBtn_clicked(self):
                servidor = Server("http://freezope2.nipltd.net/aloysoft/tests/")
                valor = servidor.sumar(float(self.item1Edit.text().latin1()),
                        float(self.item2Edit.text().latin1()))
                self.rta.setText("%f" % valor)
```

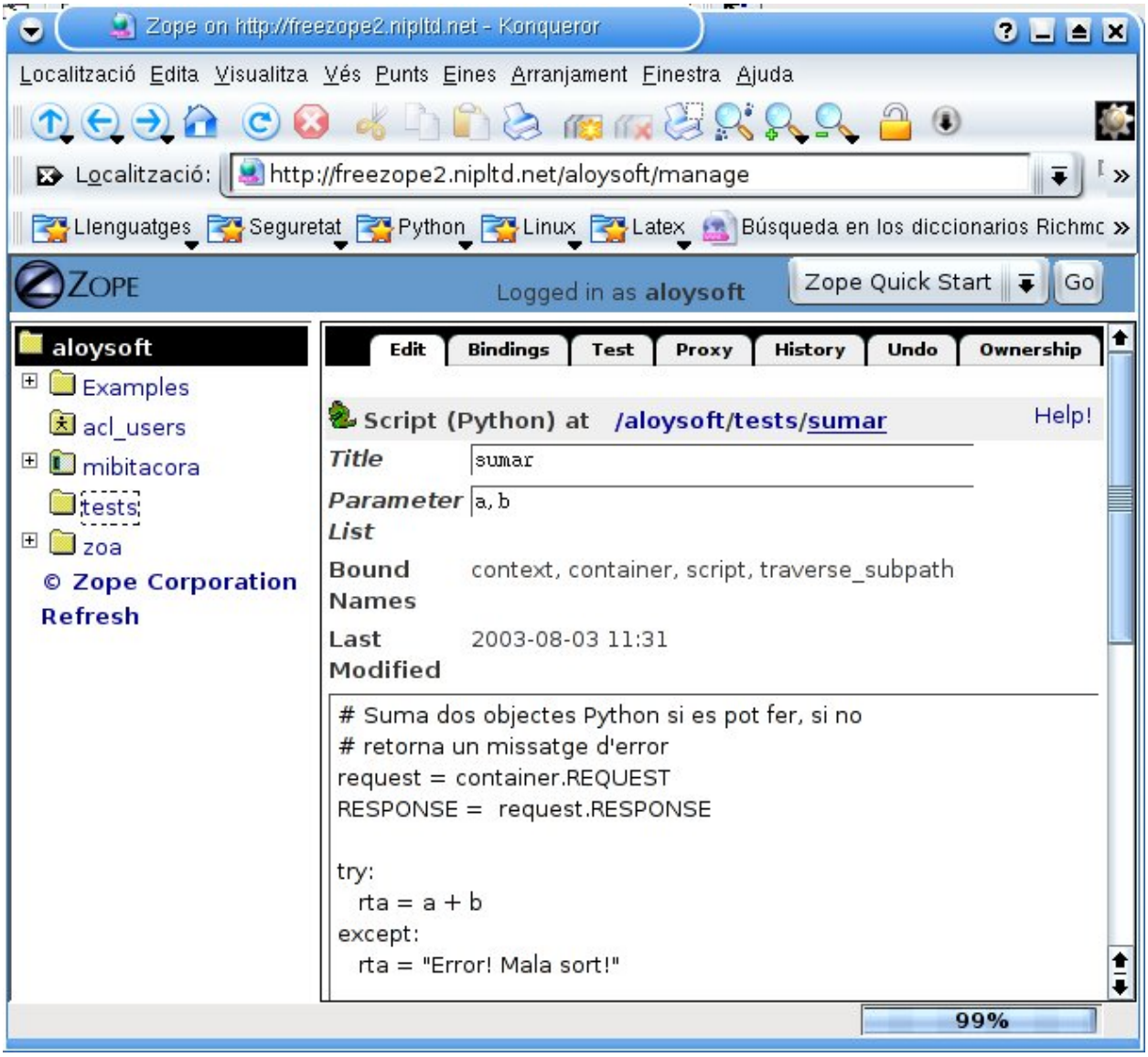


El codi font ho podeu trobar aquí[5]

La part html

Aquí hem d'entrar un poquet en el que és la programació en la part del servidor Zope. Bàsicament crearem un formulari i una pàgina per a presentar la resposta.

Tampoc és l'objectiu d'aquest aritcle ensenyar-vos a programar en un entorn Zope així que el que faré és donar-vos l'enllaç per a que proveu que efectivament funciona. Així que els incrèduls podeu pitjar aquí[6]. Per als qui els piqui la curiositat, de com és l'entorn de programació Zope



I per als que vulguin saber més, llegiu el llibre[7] damunt aquest potent framework de desenvolupament d'aplicacions web.

Un món per descobrir!

Les possibilitats que tenim de fer aplicatius accessibles tant per web com per altres tipus d'interfícies d'usuari sols estan limitades per la nostra imaginació. L'arquitectura que us proposo té una corba d'aprenentatge no massa grossa, encara que no la puc qualificar de suau. Haurem d'aprendre a fer servir Zope o un producte semblant, però els beneficis que aconseguirem són també importants: senzillesa en el codi, senzillesa en el manteniment i tenir un conjunt de regles ben definit i centralitzat per a la nostra aplicacio.

L'exemple que hem fet és prou simple, però les coses no es compliquen massa més. Malhauradament el servidor gratuït és massa limitat per a explorar algunes de les possibilitats que tenim al nostre abast, però intentaré donar-vos algunes idees:

Zope és especialment potent en l'accés a bases de dades. Per tal de fer el mateix que hem fet bastaria crear un accés a la base de dades i crear-ne els mètodes d'inserció, edició, esborrat i cerca. Amb això tendirem llesta la capa bàsica d'accés. Després crearíem els mètodes Python amb les regles del negoci que atacarien a la capa anterior.
Finalment ja sols ens quedaria fer servir la interfície que més en agradés, de la mateixa manera que hem fet amb l'exemple d'aquest article.

Un punt fonamental que no hi ha que oblidar és que el client pot ser qualsevol llenguatge que admeti crides xlm-rpc: perl, java, c++, etc.

I encara ho podem fer més divertit si feim servir una base de dades que admeti procediments amagatzemants i disparadors. Firebird per això és la meva preferida, encara que teniu en compte que llavors ens carregarem la independència de la BD que ens dóna Zope, però, quí vol portar aplicacions entre bases de dades :D

Referències

- Documentació xmlrpc [8]
- The Python Web services developer
- xmp-rpc how to[9]
- Zope xml-rpc[10]
- XML-RPC: It Works Both Ways[11]
- Zope[1]
- Python[12]
- xml.org[13]
- xmlrpc.org[14]

Lista de enlaces de este artículo:

1. http://www.zope.org
2. http://www.python.org
3. http://www.riverbankcomputing.co.uk/pyqt/index.php
4. http://freezope2.nipltd.net/
5. http://bulma.net/~aaloy/rpc/demosuma.py
6. http://freezope2.nipltd.net/aloysoft/tests/input
7. http://www.zope.org/Documentation/Books/ZopeBook/current/IntroducingZope.stx
8. http://www.pythonware.com/products/xmlrpc/
9. http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html
10. http://linux.userland.com/stories/storyReader$18
11. http://www.onlamp.com/pub/a/python/2001/01/17/xmlrpcserver.html
12. http://www.pyhon.org
13. http://www.xml.org
14. http://www.xmlrpc.org

E-mail del autor: aaloy _ARROBA_ bulma.net
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1835