Uma aplicação distribuída

César H. Kallas

O Centro de Ciências Exatas, Ambientais e de Tecnologias

Pontifícia Universidade Católica de Campinas – Campinas – Brasil

Faculdade de Engenharia de Computação

cesarkallas @ cesarkallas.net RA: 02099224

***Resumo**:*

O projeto consiste no desenvolvimento de uma aplicação cliente/servidor usando o sistema de chamdas de funções remotas (RPC).
O servidor RPC da aplicação terá 3 funções básicas.

- *Inverter uma frase;*
- *Calcular o maior divisor de um número e*
- *Calcular a raiz quadrada de um número.*

Essas funções serão executadas no servidor, a pedido do cliente.

O cliente conecta no servidor e passa o parâmetro e função que deseja que o servidor execute.

1. Introdução

O projeto foi desenvolvido utilizando a linguagem interpretada Python.

Ele possui dois arquivos:

servidor.py

cliente.py

O servidor.py tem uma função principal main() que recebe como parâmetro a porta no qual o servidor http RPC aceita conexões, essa função main() que cria uma nova instância da classe Funcoes, que possui as funções abaixo:

InversaoFrase(frase)

MaximoDivisor(numero)

RaizQuadrada(numero)

Quit

Com a instância dessa classe cria, o servidor cria um novo servidor de requisições RPC com XML e registra a instância dessa classe de funções nesse servidor, para que o cliente possa usá-las e o inicia.

Todo o pedido de acesso ao servidor pelo cliente será feito através de um servidor http embutido no servidor.py. As mensagens de troca entre o servidor e o cliente são feitas através de arquivos XML (Quadro 1), o qual possui os dados da função desejada pelo cliente.

O cliente.py tem uma função principal main() que recebe como parâmetros o endereço ip/host e a porta no qual vai conectar, essa função main() cria uma nova

conexão RPC com o servidor que retorna um objeto, que permite ao cliente chamar funções que estão no servidor e receber respostas.

```
<methodCall>
  <methodName>sample.sumAndDifference</methodName>
  <params>
    <param><value><int>5</int></value></param>
    <param><value><int>3</int></value></param>
  </params>
</methodCall>
```

Quadro 1: Exemplo de arquivo XMLRPC

Após conectar, o cliente pede as opções possiveis do servidor e a exibe no cliente. O cliente por sua vez, escolhe uma opção e a envia para o servidor RPC, esperando então uma resposta dele.

Se der erro, o cliente receberá uma mensagem de exceção.

2. Seções Específicas

2.1 Inversão de Frase

A função vai ler uma entrada do usuário (string) e vai inverte-la através de uma expressão regular.

A inversão ocorre no retorno da variável “string”, que recebe como parâmetro uma expressão regular de inversão: “[::-1]”, “return string[::-1]”.

2.2 Cálculo da Raiz Quadrada

A raiz quadrada é calculada através de uma função embutida no módulo math que já vem no python, sqrt(float).

A função sqrt recebe um float, calcula sua raiz e retorna um float com o resultado.

2.3 Máximo Divisor

A função MaximoDivisor(numero) recebe um número como parâmetro, e o máximo divisor desse número é calculado através da divisão sucessiva dele por 2.

2.4 Protocolo de Comunição

A comunicação do cliente com o servidor se dá através do protocolo TCP/IP, usando socket.

Pedidos possíveis pelo cliente:	Respostas do servidor:
'InversaoString(frase)' aonde frase é uma cadeia de caracteres.	Para frase = 'cesar kallas' Resposta = 'sallak rasec'
'MaximoDivisor(numero)' aonde numero é um inteiro de 32 bits	Para numero = 10 Resposta = 5

Pedidos possíveis pelo cliente:	Respostas do servidor:
'RaizQuadrada(numero)' aonde número é um inteiro de 32 bits	Para numero = 4 Resposta = 2
'Quit'	Desconecta o cliente

3. Resultado da Execução do Programa Desenvolvido

O resultado da execução do programa foi satisfatório.

Resultados do cliente.py

```
02099224@localhost /tmp $ python cliente.py http://172.16.12.163:10000
1. Inverter string
2. Calcular Raiz Quadrada
3. Calcular maximo divisor
4. Sair
Digite a opcao: 1
Digite a frase: cesar
rasec

1. Inverter string
2. Calcular Raiz Quadrada
3. Calcular maximo divisor
4. Sair
Digite a opcao: 2
Numero: 100000000
10000.0

1. Inverter string
2. Calcular Raiz Quadrada
3. Calcular maximo divisor
4. Sair
Digite a opcao: 2
Numero: 123123123123
350889.046741

1. Inverter string
2. Calcular Raiz Quadrada
3. Calcular maximo divisor
4. Sair
Digite a opcao: 3
Numero: 500
250

1. Inverter string
2. Calcular Raiz Quadrada
3. Calcular maximo divisor
4. Sair
Digite a opcao: 4
02099224@localhost /tmp $
```

4. Conclusão

O propósito do programa foi alcançado com sucesso.

5. Referências

PYTHON. Linguagem de programação orientada a objetos. Disponível na internet em: <http://www.python.org>. Acesso em 30/08/2006.

6. Apêndice

Quadro 2: servidor.py

```
import socket
import sys
import time
import math
import string
import thread
import SimpleXMLRPCServer

class Funcoes(object):
	def InversaoFrase(self, frase):
		"""
		Funcao que inverte uma frase
		Entradas:
			frase: string
		Saida:
			fraseInvertida: string com a frase de entrada invertida
			-1: para falha
		"""
		try:
			fraseInvertida = frase[::-1]
			return fraseInvertida
		except Exception:
			return "-1"


	def RaizQuadrada(self, numero):
		"""
		Funcao que calcula a raiz quadrada de um numero
		Entrada:
			numero: numero a ser calculado a raiz quadrada
		Saida:
			raiz: raiz quadrada do numero
			-1: para falha
		"""
		try:
			raiz = math.sqrt(float(numero))
			return raiz
		except Exception:
			return "-1"


	def MaximoDivisor(self, numero):
		"""
		Funcao que calcula o maximo divisor de um numero
		Entrada:
			numero: numero para ser calcula o seu maximo divisor
		Saida:
			maxDiv: maximo divisor do numero
			-1: para falha
		"""
		try:
			numero = int(numero)
			if numero <= 1:
					return numero
			maxDiv = numero

			for i in range(1, numero-1):
					if numero % (i) == 0:
							maxDiv = i

			if maxDiv == 1:
					if numero > 1:
							return numero
			return maxDiv
		except Exception:
			return "-1"


def main(porta):
```

Quadro 2: servidor.py

```
        funcoes = Funcoes()
        servidorRPC = SimpleXMLRPCServer.SimpleXMLRPCServer(('', porta))
        servidorRPC.register_instance(funcoes)
        servidorRPC.serve_forever()


if __name__ == "__main__":
        if len(sys.argv) == 2:
                porta = sys.argv[1]
                if porta.isdigit() == True:
                        main(int(porta))
                        sys.exit(0)
        print "Uso: %s porta" % sys.argv[0]
```

Quadro 3: cliente.py

```
from socket import *
from string import *
import sys
import string
import xmlrpclib

def main(servidor):
	opcoes = (
				"1. Inverter string",
				"2. Calcular Raiz Quadrada",
				"3. Calcular maximo divisor",
				"4. Sair"
				)

	servidor = xmlrpclib.ServerProxy(str(servidor))

	while(1):
		for opcao in opcoes:
			print opcao

		entrada = raw_input("Digite a opcao: ", )

		if entrada == "1":
			frase = raw_input("Digite a frase: ", )
			if len(frase) > 0:
				frase = servidor.InversaoFrase(frase)
				print frase
			else:
				print "frase invalida"

		elif entrada == "2":
			numero = raw_input("Numero: ", )
			print servidor.RaizQuadrada(numero)

		elif entrada == "3":
			numero = raw_input("Numero: ", )
			print servidor.MaximoDivisor(numero)

		elif entrada == "4":
			break

		print "\n"


if __name__ == "__main__":
	if len(sys.argv) == 2:
		main(sys.argv[1])
	else:
		print "Uso: %s host\nExemplo: %s http://localhost:10000" % (sys.argv[0], sys.argv[0])
```