Sistemas Distribuídos

Aula Prática - RED e Protocol Buffer

Prof. Rodrigo Campiolo

UTFPR - Universidade Tecnológica Federal do Paraná

21 de agosto de 2020

Introdução

Objetivos

- Desenvolver uma aplicação de agenda simples usando duas linguagens (Python e Java).
- Apresentar a instalação, configuração e uso de Protocol Buffer para RED.
- Avaliar o uso de **Protocol Buffer** para representação das estruturas trocadas entre cliente e servidor.

Prática: RED e Protocol Buffer

Materiais

- JDK e Python 3
- protoc
  `https://github.com/protocolbuffers/protobuf/releases/download/v3.13.0/protoc-3.13.0-linux-x86_64.zip`
- Python 3 APIs
  `pip3 install python3-protobuf protobuf`
- Java API
  `https://repo1.maven.org/maven2/com/google/protobuf/protobuf-java/3.13.0/protobuf-java-3.13.0.jar`

Prática: RED e Protocol Buffer

Instalação

- criar e acessar pasta de trabalho
  mkdir −p protoaula/protoc
  cd protoaula/protoc
- obter e descompactar o compilador protoc
  wget `https://github.com/protocolbuffers/protobuf/releases/download/v3.13.0/protoc-3.13.0-linux-x86_64.zip`
  unzip protoc−3.13.0−linux−x86_64.zip

Prática: RED e Protocol Buffer

Configuração de pastas

- acessar o diretório **protoaula**
  cd protoaula
- criar as pastas: **javacode** e **pythoncode**
  mkdir javacode
  mkdir pythoncode
- criar atalho para o **protoc**
  alias protoc=/home/user/protoaula/protoc/bin/protoc [1]

[1] pode-se adicionar no .bashrc

Especificando estrutura para troca de dados

- Em protoaula, criar o arquivo **addressbook.proto**

```
1 syntax = "proto3";
2
3 message Person {
4   string name = 1;
5   int32 id = 2;
6   string email = 3;
7 }
```

Prática: RED e Protocol Buffer

Gerando código da estrutura

- Gerar a estrutura e o código para Python:
  protoc --python_out=pythoncode/ addressbook.proto
- Gerar a estrutura e o código para Java:
  protoc --java_out=javacode/ addressbook.proto

Prática: RED e Protocol Buffer

Compilar a estrutura em Java

- Acessar o diretório e obter a API:
  cd javacode
  wget `https://repo1.maven.org/maven2/com/google/protobuf/protobuf-java/3.13.0/protobuf-java-3.13.0.jar`
- Compilar o código fonte:
  javac −−classpath .:protobuf−java−3.13.0.jar ∗.java

Cliente - Python

```
import socket
import addressbook_pb2

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(("localhost", 7000))

# instanciar e preencher a estrutura
person = addressbook_pb2.Person()
person.id = 234
person.name = "Rodrigo Campiolo"
person.email = "rcampiolo@ibest.com.br"

# marshalling
msg = person.SerializeToString()
size = len(msg)

client_socket.send((str(size) + "\n").encode())
client_socket.send(msg)

client_socket.close()
```

Servidor - Java

```
public class ServidorTcpAddressBook {
    public static void main(String args[]) {
        try {
            int serverPort = 7000;
            ServerSocket listenSocket = new ServerSocket(serverPort);
            while (true) {
                Socket clientSocket = listenSocket.accept();

                DataInputStream inClient =
                    new DataInputStream(clientSocket.getInputStream());

                String valueStr = inClient.readLine();
                int sizeBuffer = Integer.valueOf(valueStr);
                byte[] buffer = new byte[sizeBuffer];
                inClient.read(buffer);

                /* realiza o unmarshalling */
                Addressbook.Person p = Addressbook.Person.parseFrom(buffer);

                /* exibe na tela */
                System.out.println("--\n" + p + "--\n");
            } //while
        } catch (IOException e) {
            System.out.println("Listen socket:" + e.getMessage());
        } //catch
    } //main
} //class
```

Prática: RED e Protocol Buffer

Executando a aplicação

- Compilar o código fonte Java:

  javac −−classpath .:protobuf−java−3.13.0.jar ∗.java

- Iniciar o servidor em Java no diretório *javacode*:

  java −−classpath .:protobuf−java−3.13.0.jar ServidorTcpAddressBook

- Iniciar o cliente em Python no diretório *pythoncode*:

  python3 client.py

Prática: RED e Protocol Buffer

Atividades

1. Implementar o servidor em Python e o cliente em Java.
2. Implementar a comunicação usando UDP.
3. Adicionar novas funcionalidades: armazenar, consultar, listar e remover contatos.

Referências Bibliográficas

Referências

- Protocol Buffer Basics: Java. Disponível em: `https://developers.google.com/protocol-buffers/docs/javatutorial`.
- Protocol Buffer Basics: Python. Disponível em: `https://developers.google.com/protocol-buffers/docs/pythontutorial`.