APPRENDIMENTO sockets

#sockets

Sommario

Di

Capitolo 1: Iniziare con le prese

Osservazioni

Questa sezione fornisce una panoramica di cosa sono i socket e perché uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare eventuali soggetti di grandi dimensioni all'interno delle prese e collegarsi agli argomenti correlati. Poiché la Documentazione per i socket è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Examples

Come creare un'istanza di un oggetto classe socket

L'istanziazione di una presa può essere eseguita in vari modi.

1. con la dichiarazione e l'istanziazione di 2 righe:

   Per prima cosa dobbiamo definire una variabile che possa contenere un oggetto di classe Socket:

   ```
   Socket socket;
   ```

   allora possiamo creare un oggetto di classe Socket:

   ```
   socket = new Socket();
   ```

2. Possiamo anche creare una definizione e un'istanza di una riga:

   ```
   Socket  socket = new Socket();
   ```

entrambi i modi creeranno un socket non connesso.

Possiamo usare altri costruttori parametrizzati per istanziare oggetti socket class connessi o non connessi:

Per dettagli vedi le specifiche di classe doc:

https://docs.oracle.com/javase/7/docs/api/java/net/Socket.html

Crea socket non connesso, prova a connetterti e controlla se la connessione è stabilita

```
public class ConnectSocketExample {
```

```
    private int HTTP_PORT = 80;

    /**
     * example method to create unconnected socket
     * then connect to it
     * at end return connected socket
     *
     * @param httpHostName - endpoint host name fot socket connection
     * @throws IOException - if the socket is already connected or an error occurs while
connecting.
     */
    protected Socket connectSocket(String httpHostName) throws IOException {
        // define local variable for socket and create unconnected socket
        Socket socket = new Socket();
        //  create iNet address for socket
        InetSocketAddress inetSocketAddress = new InetSocketAddress(httpHostName, HTTP_PORT);
        // connect socket to inet address (end point )
        socket.connect(inetSocketAddress);
        // return connected socket for later use
        return socket;
    }

    /**
     * public method for try to create connected to goole.com http port socket
     * and with check and system out print if this try was successful
     **/
    public void createAndCheckIfConnected() {
        try {
            Socket connectedSocket = connectSocket("google.com");
            boolean connected = connectedSocket.isConnected();
            System.out.print("Socket is:" + (!connected ? " not" : "" +  " connected"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

Scrivi su socket una semplice richiesta get http e rispondi alla risposta

```
/**
 * we reuse a class written in example:
 * http://stackoverflow.com/documentation/sockets/2876/introduction-to-
sockets#t=201607262114505531351
 * pleas to familiar with it first to continue with this one
 **/
public class WriteToSocketExample extends ConnectSocketExample {

    private String CRLF = "\r\n"; // line termination (separator)

    /**
     * write a simple http get request to socket
     * @param host - host to establish a connection
     *              (http server - see ConnectSocketExample HTTP_PORT )
     * @param path - path to file ( in this case a url location - part used in browser after
host)
     * @return a  connected socket with filled in raw get request
     * @throws IOException - see ConnectSocketExample.connectSocket(host);
     */
    protected Socket writeGetToSocket(String host, String path) throws IOException {
        // create simple http raw get request for host/path
        String rawHttpGetRequest = "GET "+ path +" HTTP/1.1 " + CRLF  // request line
                + "Host: "+ host + CRLF
                + CRLF;
        // get bytes of this request using proper encodings
        byte[] bytesOfRequest = rawHttpGetRequest.getBytes(Charset.forName("UTF-8"));
        // create & connect to socket
        Socket socket = connectSocket(host);
        // get socket output stream
        OutputStream outputStream = socket.getOutputStream();
        // write to the stream a get request we created
        outputStream.write(bytesOfRequest);
        // return socket with written get request
        return  socket;
    }

    /**
     * create, connect and write to socket simple http get request
     * then dump response of this request
     * @throws IOException
     */
    public void writeToSocketAndDumpResponse() throws IOException {
        // send request to http server for / page content
        Socket socket = writeGetToSocket("google.com", "/");
        // now we will read response from server
        InputStream inputStream = socket.getInputStream();
        // create a byte array buffer to read respons in chunks
        byte[] buffer = new byte[1024];
        // define a var to hold count of read bytes from stream
        int weRead;
        // read bytes from sockets till exhausted or read time out will occurred ( as we
didn't add in raw get header Connection: close (default keep-alive)
        while ((weRead = inputStream.read(buffer)) != -1) {
            // print what we have read
            System.out.print(new String(buffer));
        }
    }
}
```

Leggi Iniziare con le prese online: https://riptutorial.com/it/sockets/topic/2876/iniziare-con-le-prese

Capitolo 2: Socket C ++

introduzione

Questo argomento riguarderà il moderno stile C ++ Berkeley Socket Programming (questo è un codice per Linux, ma facilmente trasportabile su altre piattaforme)

Examples

Esempio di codice server

```
constexpr const size_t addressSize = sizeof(sockaddr_in);
constexpr const uint16_t defaultPort = 80; // The port you want to use

int serverSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
sockaddr_in serverAddress, clientAddress;

memset(&serverAddress, 0, addressSize);
serverAddress.sin_family = AF_INET;
serverAddress.sin_addr.s_addr = htonl(INADDR_ANY);
serverAddress.sin_port = htons(defaultPort);

bind(serverSocket, (sockaddr*)&serverAddress, addressSize);
listen(serverSocket, SOMAXCONN);

while (true) { // Infinite running app
    std::thread{ // Create new thread for every client
        handleConnection, //Connection handler
        accept(serverSocket, (sockaddr*)&clientAddress, &addressSize) //Client socket
        // Any other parameters for the handler here
    }.detach(); // Detached thread to make resource management easier
}
return 0;
```

Leggi Socket C ++ online: https://riptutorial.com/it/sockets/topic/8265/socket-c-plusplus

Capitolo 3: Socket TCP Python; semplici esempi di server e client con annotazioni

Osservazioni

Questi sono due programmi di esempio che funzionano insieme. Uno è un semplice server, l'altro un semplice client. Avvia il server in una finestra:

```
python tserver.py
```

Modificare l'indirizzo del server nel file sorgente del client, se lo si desidera. Quindi corri

```
python tclient.py
```

Il client si connette al server, quindi chiede l'input dalla console, quindi lo invia al server. Per ogni buffer ricevuto, il server prepara alcune informazioni predefinite e lo invia al client.

Ho lavorato su alcune insidie che sorgono nel porting del codice tra python2 e python3 - in particolare le differenze tra byte e stringhe. Una spiegazione completa di ciò richiederebbe molto spazio e distolse dal focus del `socket` .

Avvertenze:

L'esempio del server, in particolare, è focalizzato sulle operazioni `socket` eseguite da un server, ma serializzate per chiarezza. Quindi, accetta solo una singola connessione alla volta. Un programma "reale" dovrebbe imporre un nuovo processo per gestire ciascuna connessione oppure utilizzare `select` per gestire più connessioni contemporaneamente.

I programmi reali gestiscono le eccezioni nelle varie chiamate socket e si ripristinano o escono con garbo.

I programmi reali dovrebbero preoccuparsi dei limiti dei messaggi (poiché TCP non li rispetta). Poiché questi programmi inviano singoli buffer alla volta attivati dall'input dell'utente, questo è stato ignorato.

Examples

Programma server di esempio (annotato)

```
#!/usr/bin/env python
"""
An annotated simple socket server example in python.

WARNING: This example doesn't show a very important aspect of
TCP - TCP doesn't preserve message boundaries. Please refer
```

```
to http://blog.stephencleary.com/2009/04/message-framing.html
before adapting this code to your application.

Runs in both python2 and python3.
"""
import socket

# Optionally set a specific address. This (the empty string) will listen on all
# the local machine's IPv4 addresses. It's a common way to code a general
# purpose server. If you specify an address here, the client will need to use
# the same address to connect.
SERVER_ADDRESS = ''

# Can change this to any port 1-65535 (on many machines, ports <= 1024 are
# restricted to privileged users)
SERVER_PORT = 22222

# Create the socket
s = socket.socket()

# Optional: this allows the program to be immediately restarted after exit.
# Otherwise, you may need to wait 2-4 minutes (depending on OS) to bind to the
# listening port again.
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

# Bind to the desired address(es) and port. Note the argument is a tuple: hence
# the extra set of parentheses.
s.bind((SERVER_ADDRESS, SERVER_PORT))

# How many "pending connections" may be queued. Exact interpretation of this
# value is complicated and operating system dependent. This value is usually
# fine for an experimental server.
s.listen(5)

print("Listening on address %s. Kill server with Ctrl-C" %
      str((SERVER_ADDRESS, SERVER_PORT)))

# Now we have a listening endpoint from which we can accept incoming
# connections. This loop will accept one connection at a time, then service
# that connection until the client disconnects. Lather, rinse, repeat.
while True:
    c, addr = s.accept()
    print("\nConnection received from %s" % str(addr))

    while True:
        data = c.recv(2048)
        if not data:
            print("End of file from client. Resetting")
            break

        # Decode the received bytes into a unicode string using the default
        # codec. (This isn't strictly necessary for python2, but, since we will
        # be encoding the data again before sending, it works fine there too.)
        data = data.decode()

        print("Received '%s' from client" % data)

        data = "Hello, " + str(addr) + ". I got this from you: '" + data + "'"

        # See above
        data = data.encode()
        # Send the modified data back to the client.
        c.send(data)

    c.close()
```

Esempio di programma client (annotato)

```
#!/usr/bin/env python
"""
An annotated simple socket client example in python.

WARNING: This example doesn't show a very important aspect of
TCP - TCP doesn't preserve message boundaries. Please refer
to http://blog.stephencleary.com/2009/04/message-framing.html
before adapting this code to your application.

Runs in both python2 and python3.
"""
import socket

# Note that the server may listen on a specific address or any address
# (signified by the empty string), but the client must specify an address to
# connect to. Here, we're connecting to the server on the same machine
# (127.0.0.1 is the "loopback" address).
SERVER_ADDRESS = '127.0.0.1'
SERVER_PORT = 22222

# Create the socket
c = socket.socket()

# Connect to the server. A port for the client is automatically allocated
# and bound by the operating system
c.connect((SERVER_ADDRESS, SERVER_PORT))

# Compatibility hack. In python3, input receives data from standard input. In
# python2, raw_input does exactly that, whereas input receives data, then
# "evaluates" the result; we don't want to do that. So on python2, overwrite
# the input symbol with a reference to raw_input. On python3, trap the
# exception and do nothing.
try:
    input = raw_input
except NameError:
    pass

print("Connected to " + str((SERVER_ADDRESS, SERVER_PORT)))
while True:
    try:
        data = input("Enter some data: ")
    except EOFError:
        print("\nOkay. Leaving. Bye")
        break

    if not data:
        print("Can't send empty string!")
        print("Ctrl-D [or Ctrl-Z on Windows] to exit")
        continue

    # Convert string to bytes. (No-op for python2)
    data = data.encode()

    # Send data to server
    c.send(data)

    # Receive response from server
    data = c.recv(2048)
    if not data:
        print("Server abended. Exiting")
        break

    # Convert back to string for python3
    data = data.decode()

    print("Got this string from server:")
    print(data + '\n')

c.close()
```

Leggi Socket TCP Python; semplici esempi di server e client con annotazioni online: https://riptutorial.com/it/sockets/topic/5668/socket-tcp-python--semplici-esempi-di-server-e-client-con-annotazioni

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con le prese	ceph3us, Community, Gil Hamilton
2	Socket C ++	Zhyano
3	Socket TCP Python; semplici esempi di server e client con annotazioni	Gil Hamilton, Vovanrock2002