eBook Gratuit

APPRENEZ sockets

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#sockets

Table des matières

À propos

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: sockets

It is an unofficial and free sockets ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official sockets.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapitre 1: Démarrer avec les sockets

Remarques

Cette section fournit une vue d'ensemble de ce que sont les sockets et pourquoi un développeur peut vouloir l'utiliser.

Il convient également de mentionner tous les grands sujets dans les sockets et de les relier aux sujets connexes. La documentation des sockets étant nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Examples

Comment instancier un objet de classe socket

L'instanciation d'un socket peut se faire de différentes manières.

1. par déclaration et instanciation sur 2 lignes:

   Nous devons d'abord définir une variable qui contiendra un objet de classe Socket:

   ```
   Socket socket;
   ```

   alors nous pouvons créer un objet de classe Socket:

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

2. Nous pouvons également faire une définition et une instanciation d'une seule ligne:

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

Dans les deux cas, un socket non connecté sera créé.

Nous pouvons utiliser d'autres constructeurs paramétrés pour instancier un objet de classe de socket connecté ou non:

Pour plus de détails, voir les spécifications de la classe doc:

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

Créez un socket non connecté, essayez de vous y connecter et vérifiez si la connexion est établie

```
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();
        }
    }

}
```

Écrivez à la socket une simple requête de requête http et une réponse de vidage

```
/**
 * we reuse a class written in example:
 * http://stackoverflow.com/documentation/sockets/2876/introduction-to-
sockets#t=20160726211450553135 1
 * 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));
        }
    }
}
```

Lire Démarrer avec les sockets en ligne: https://riptutorial.com/fr/sockets/topic/2876/demarrer-avec-les-sockets

Chapitre 2: Sockets C ++

Introduction

Ce sujet portera sur la programmation moderne de Socket Berkeley (c'est le code pour Linux, mais facilement transférable sur d'autres plates-formes)

Examples

Exemple de code serveur

```
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;
```

Lire Sockets C ++ en ligne: https://riptutorial.com/fr/sockets/topic/8265/sockets-c-plusplus

Chapitre 3: Sockets Python TCP; exemples simples de serveur et de client avec annotation

Remarques

Ce sont deux exemples de programmes qui fonctionnent ensemble. L'un est un simple serveur, l'autre un simple client. Démarrer le serveur dans une fenêtre:

```
python tserver.py
```

Modifiez l'adresse du serveur dans le fichier source du client si vous le souhaitez. Puis courir

```
python tclient.py
```

Le client se connecte au serveur, puis demande une entrée à partir de la console, puis l'envoie au serveur. Pour chaque tampon reçu, le serveur ajoute des informations prédéfinies et les renvoie au client.

J'ai travaillé sur certains écueils qui surviennent lors du portage du code entre python2 et python3 - en particulier les différences entre octets et chaînes. Une explication complète de ceci nécessiterait beaucoup d'espace et détournerait l'attention du `socket` .

Mises en garde:

L'exemple de serveur, en particulier, se concentre sur les opérations de `socket` qu'un serveur effectuera, mais sérialisé pour plus de clarté. Par conséquent, il n'accepte qu'une seule connexion à la fois. Un "vrai" programme décomposerait un nouveau processus pour gérer chaque connexion, ou utiliser `select` pour gérer plusieurs connexions à la fois.

Les vrais programmes gèrent les exceptions dans les différents appels de socket et récupèrent ou quittent avec élégance.

Les vrais programmes devraient s'inquiéter des limites des messages (puisque TCP ne les respecte pas). Comme ces programmes envoient des tampons uniques à la fois déclenchés par une entrée utilisateur, ils ont été ignorés.

Examples

Exemple de programme serveur (annoté)

```
#!/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()
```

Exemple de programme client (annoté)

```
#!/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()
```

Lire Sockets Python TCP; exemples simples de serveur et de client avec annotation en ligne: https://riptutorial.com/fr/sockets/topic/5668/sockets-python-tcp--exemples-simples-de-serveur-et-de-client-avec-annotation

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec les sockets	ceph3us, Community, Gil Hamilton
2	Sockets C ++	Zhyano
3	Sockets Python TCP; exemples simples de serveur et de client avec annotation	Gil Hamilton, Vovanrock2002