EBook Gratis

APRENDIZAJE sockets

Free unaffiliated eBook created from
Stack Overflow contributors.

#sockets

Tabla de contenido

Acerca de

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.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

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

Capítulo 1: Empezando con sockets

Observaciones

Esta sección proporciona una descripción general de qué son los sockets y por qué un desarrollador puede querer usarlos.

También debe mencionar cualquier tema grande dentro de sockets, y vincular a los temas relacionados. Dado que la Documentación para sockets es nueva, es posible que deba crear versiones iniciales de los temas relacionados.

Examples

Cómo instanciar un objeto de clase socket

La creación de una instancia de socket se puede hacer de varias maneras.

1. por 2 líneas de declaración y creación de instancias:

   Primero necesitamos definir una variable que contendrá un objeto de clase Socket:

   ```
   Socket socket;
   ```

   Entonces podemos crear un objeto de clase Socket:

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

2. También podemos hacer una definición de una línea y una instanciación:

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

Ambas formas crearán un socket desconectado.

Podemos usar otros constructores parametrizados para crear una instancia de objeto de clase de socket conectado o no conectado:

Para más detalles ver especificaciones de clase doc:

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

Cree un socket desconectado, intente conectarse a él y verifique si la conexión está establecida

```
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();
        }
    }

}
```

Escriba en el socket una solicitud de obtención de http simple y descargue la respuesta

```
/**
 * 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));
        }
    }
}
```

Lea Empezando con sockets en línea: https://riptutorial.com/es/sockets/topic/2876/empezando-con-sockets

Capítulo 2: C ++ Sockets

Introducción

Este tema tratará sobre la programación de zócalos Berkeley de estilo C ++ moderno (este es un código para Linux, pero fácilmente transportable a otras plataformas)

Examples

Código de servidor de muestra

```
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;
```

Lea C ++ Sockets en línea: https://riptutorial.com/es/sockets/topic/8265/c-plusplus-sockets

Capítulo 3: Conectores TCP de Python; Ejemplos simples de servidor y cliente con anotación.

Observaciones

Estos son dos programas de ejemplo que trabajan juntos. Uno es un servidor simple, el otro es un cliente simple. Inicie el servidor en una ventana:

```
python tserver.py
```

Edite la dirección del servidor en el archivo fuente del cliente si lo desea. Entonces corre

```
python tclient.py
```

El cliente se conecta al servidor, luego solicita información de la consola y luego la envía al servidor. Para cada búfer recibido, el servidor prepara un poco de información enlatada y la envía al cliente.

He trabajado alrededor de ciertos escollos que surgen al portar código entre python2 y python3, en particular las diferencias entre bytes y cadenas. Una explicación completa de eso requeriría mucho espacio y distraería del enfoque de `socket` .

Advertencias:

El ejemplo del servidor, en particular, se centra en las operaciones de `socket` realizará un servidor, pero se serializa para mayor claridad. Por lo tanto, solo acepta una conexión a la vez. Un programa "real" daría un nuevo proceso para manejar cada conexión, o utilizaría `select` para manejar múltiples conexiones a la vez.

Los programas reales manejarían las excepciones en las diversas llamadas de socket, y se recuperarán o cerrarán con gracia.

Los programas reales deberían preocuparse por los límites de los mensajes (ya que TCP no los respeta). Dado que estos programas envían buffers individuales a la vez activados por la entrada del usuario, eso se ha ignorado.

Examples

Programa de servidor de muestra (anotado)

```
#!/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()
```

Programa cliente de muestra (anotado)

```
#!/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()
```

Lea Conectores TCP de Python; Ejemplos simples de servidor y cliente con anotación. en línea: https://riptutorial.com/es/sockets/topic/5668/conectores-tcp-de-python--ejemplos-simples-de-servidor-y-cliente-con-anotacion-

Creditos

S. No	Capítulos	Contributors
1	Empezando con sockets	ceph3us, Community, Gil Hamilton
2	C ++ Sockets	Zhyano
3	Conectores TCP de Python; Ejemplos simples de servidor y cliente con anotación.	Gil Hamilton, Vovanrock2002