

Forum:[Hack Forums](#)

Thread: [Java Client-Server socket sending/receiving](#)

+Ervinas - July 27, 2017, 1:07 am

Hello, got a huge problem. Stuck here for a week now.. Need help with my code a bit.

I'm trying to send object from client to server.

Steps:

1. Run server
2. Run client
3. Connection - OK.
4. Server creates thread that waits for client input.
5. Client sends AUTH information to server , server RECEIVES and checks it.
6. Auth completed, client joined server.
7. Client sends String[] object to server
8. Server gets it and thows me error : Code java.io.StreamCorruptedException: invalid type code: AC

After all googling and reading, i know that 1 socket can have only 1 ObjectOutputStream and 1 ObjectInputStream on each sides (client/server).

I double checked my ServerComm.java and ServerThread.java if there are any more, but there arent!

Everything is fine. Don't know what to do.

ServerComm.java : It's class from Client side, to communicate with server.

Code package client;

```
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.net.UnknownHostException;

public class ServerComm{

    private Socket socket;
    private static ObjectOutputStream oos;

    public ServerComm(Socket socket) throws UnknownHostException, IOException {

        this.socket = socket;
        oos = new ObjectOutputStream(socket.getOutputStream());

    }
```

```
protected static void sendLine(String line, String type, Socket socket) throws IOException{
```

```
    String[] a = {line,type};
    sendToServerStringArray(a);
    System.out.println("Sending line: " + line);
    System.out.println("Sending type: " + type);
```

```
}
```

```
public Socket getSocket() {
    return this.socket;
}
```

```
protected static boolean sendAuth(Socket socket) throws IOException {
```

```
    String project_id = Config.getProject_id();
    String user_id = Config.getUser_id();
```

```
    String[] auths = {project_id, user_id};
    sendToServerStringArray(auths);
    System.out.println("Authentication process started");
```

```
    ObjectInputStream ois = new ObjectInputStream(socket.getInputStream());
```

```
    if(ois.readBoolean() == true) {
        System.out.println("Authentication successfull!");
        return true;
    }
    System.out.println("Authentication failed, dropping connection.");
    return false;
```

```
}
```

```
protected static void sendToServerStringArray(String[] array) throws IOException {
```

```
    oos.reset();
    oos.writeUnshared(array);
    oos.flush();
```

```
}
```

```
}
```

ServerThread.java : it's a thread that accept client connections and check auths and waiting for new String[] object from client to accept.

Code package server;

```
import java.io.BufferedReader;
import java.io.EOFException;
import java.io.IOException;
```

```
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.ArrayList;
```

```
class ServerThread extends Thread {
```

```
    public static String name;
```

```
    String line = null;
```

```
    BufferedReader is = null;
```

```
    PrintWriter os = null;
```

```
    public static Socket socket = null;
```

```
    static ObjectInputStream ois;
```

```
    @SuppressWarnings("static-access")
```

```
    public ServerThread(Socket socket) throws ClassNotFoundException, IOException{
```

```
        this.setName("Client");
```

```
        name = this.getName();
```

```
        this.socket = socket;
```

```
        ois = new ObjectInputStream(socket.getInputStream());
```

```
    }
```

```
    public void run() {
```

```
        try {
```

```
            while(true) {
```

```
                String[] variables = getInputFromClient();
```

```
                for(String variable : variables) {
```

```
                    System.out.println(variable);
```

```
                }
```

```
            }
```

```
        }
```

```
        catch(NullPointerException e)
```

```
        {
```

```
            line = this.getName();
```

```
            System.out.println("Client "+line+" Closed");
```

```
        } catch (ClassNotFoundException e) {
```

```
            e.printStackTrace();
```

```
        } catch (IOException e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
        finally
```

```

{
    try{
        System.out.println("Connection Closing..");
        if (is != null){
            is.close();
            System.out.println("Socket Input Stream Closed");
        }

        if(os!=null){
            os.close();
            System.out.println("Socket Out Closed");
        }
        if (socket!=null){
            socket.close();
            System.out.println("Socket with :"+ this.getName() + " were closed");
        }

    }
    catch(IOException ie){
        System.out.println("Socket Close Error");
    }
}
}

```

protected static boolean checkAuth() throws IOException, ClassNotFoundException {
 ArrayList<String> auths = new ArrayList<String>();

// read ois truksta

String[] authsFromClient = getInputFromClient();

```

for(String auth : authsFromClient) {
    auths.add(auth);
}

```

int temp = 0;

for(int i = 0 ; i < auths.size() ; i++) {

```

    if(Config.getProjectId().equals(auths.get(i)) || Config.getUserId().equals(auths.get(i)) ){
        temp += 1;
    }
    else {
        temp -= 1;
    }
}

```

```

}
if(temp == auths.size()) {
    temp = 0;
    ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
    oos.writeBoolean(true);
    oos.flush();
    return true;
}

```

```

    }
    else {

        temp = 0;
        ObjectOutputStream oos = new ObjectOutputStream(socket.getOutputStream());
        oos.writeBoolean(false);
        oos.flush();
        return false;

    }

}

private static String[] getInputFromClient() throws ClassNotFoundException, IOException {

    String[] temp = (String[]) ois.readObject();
    return temp;

}

}

```

P.S. This method gets me error.

```

Code    private static String[] getInputFromClient() throws ClassNotFoundException, IOException {

        String[] temp = (String[]) ois.readObject();
        return temp;

    }
Code String[] temp = (String[]) ois.readObject();

```

matt - July 27, 2017, 1:20 am

This error occurs because you're initializing a new ObjectOutputStream each time you want to send data, you want to use only 1 throughout the life of the socket. Also I noticed that when you're receiving data its good practice to not cast it right away as what you're receiving may not be what you want so I'd do something like

```

Code Object o = ois.readObject();
if(o instanceof String[].class) {
    String[] data = (String[]) o;
}

```

+Ervinas - July 27, 2017, 1:34 am

(Jul 27, 2017 01:20 AM)Midnight Blue™ Wrote: [\[CITE\]](#) This error occurs because you're initializing a new ObjectOutputStream each time you want to send data, you want to use only 1 throughout the life of the socket. Also I noticed that when you're receiving data its good practice to not cast it right away as what you're receiving may not be what you want so I'd do something like

```

Code Object o = ois.readObject();
if(o instanceof String[].class) {
    String[] data = (String[]) o;
}

```

I know what you said about socket and one ObjectOutputStream, i do use one only. Am i

understanding usage?

And is it possible to make them work faster? Like flushing socket and receiving?

matt - July 27, 2017, 7:34 am

(Jul 27, 2017 01:34 AM)+Ervinas Wrote: [\[CITE\]](#) I know what you said about socket and one ObjectOutputStream, i do use one only. Am i understanding usage?

And is it possible to make them work faster? Like flushing socket and receiving?
Every time you call sendAuth() you're initializing a new stream

sexyPudding - July 27, 2017, 1:00 pm

I would just use JSON and just use the http protocol for almost all communication

matt - July 27, 2017, 1:56 pm

(Jul 27, 2017 01:00 PM)sexyPudding Wrote: [\[CITE\]](#) I would just use JSON and just use the http protocol for almost all communication
that's lame with a lot of unnecessary overhead

sexyPudding - July 29, 2017, 1:26 pm

(Jul 27, 2017 01:56 PM)Midnight Blue™ Wrote: [\[CITE\]](#) that's lame with a lot of unnecessary overhead

If you are only sending data between Java JVMs, then either choice is possible.

A textual representation (XML, JSON, or custom) has several advantages:

- it's easier to make it interoperable between Java and other languages
- it's less brittle in the face of version changes or slightly different versions of your code at each end of the socket
- it's vastly easier to test and debug

Depending on the format, it may be a little slower, but this often not significant.

matt - July 29, 2017, 2:16 pm

(Jul 29, 2017 01:26 PM)sexyPudding Wrote: [\[CITE\]](#) If you are only sending data between Java JVMs, then either choice is possible.

A textual representation (XML, JSON, or custom) has several advantages:

- it's easier to make it interoperable between Java and other languages
- it's less brittle in the face of version changes or slightly different versions of your code at each end of the socket
- it's vastly easier to test and debug

Depending on the format, it may be a little slower, but this often not significant.

I like your point that it's easily usable between languages but I could argue that it's good practice to use proper protocols when applicable so you don't fall in to bad habits. :P