## Sockets

Trong Java các lớp URL và URLConnections cung cấp một cơ chế high-level để truy cập tài nguyên trên Internet. Tuy nhiên đôi khi các chương trình của bạn yêu cầu việc giao tiếp với nhau qua mạng nội bộ, ví dụ như khi bạn muốn viết một ứng dụng client-server.

Trong các ứng dụng client-server, server cung cấp một số dịch vụ, chẳng hạn như xử lý các truy vấn cơ sở dữ liệu hoặc gửi giá cổ phiếu hiện tại. Client sử dụng dịch vụ được cung cấp bởi server, hoặc hiển thị kết quả truy vấn cơ sở dữ liệu cho người dùng hoặc đưa ra đề xuất mua cổ phiếu cho nhà đầu tư. Giao tiếp xảy ra giữa client và server phải đáng tin cậy. Tức là, không có dữ liệu nào có thể bị biến mất và phải đến phía client theo thứ tự mà server đã gửi nó.

TCP(Transmission Control Protocol) cung cấp một kênh truyền thông đáng tin cậy cho phép ứng dụng client-server sử dụng để giao tiếp với nhau. Để có thể giao tiếp với nhau qua TCP, client và server được thiết lập để kết nối với nhau qua Socket.

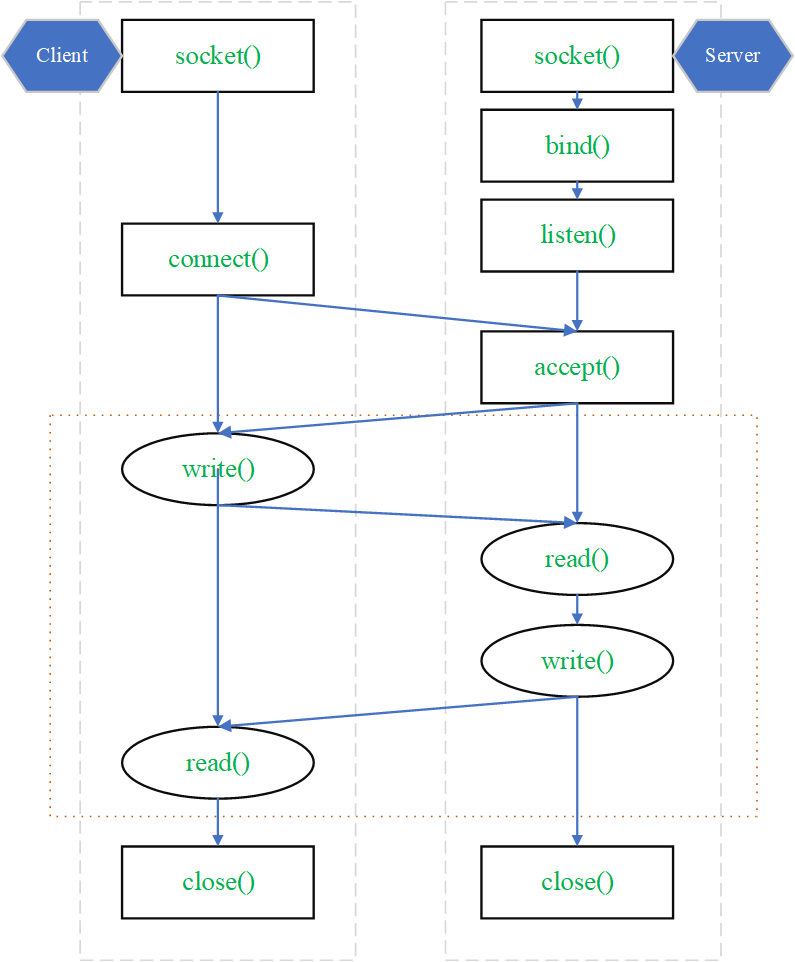
## Socket là gì?

Socket là một điểm cuối(end-point) của liên kết truyền thông hai chiều(two-way communication) giữa hai chương trình chạy trên mạng. Các lớp Socket được sử dụng để biểu diễn kết nối giữa client và server. Gói [java.net](http://java.net/) cung cấp hai lớp - Socket và ServerSocket - thể hiện kết nối giữa client và server.

Các Socket cung cấp kỹ thuật giao tiếp giữa hai máy tính sử dụng TCP. Một chương trình Client tạo một socket trên đầu cuối của giao tiếp và cố gắng để kết nối socket đó tới một Server.

Khi kết nối được tạo, Server tạo một đối tượng Socket trên đầu cuối của giao tiếp. Client và Server bây giờ có thể giao tiếp bằng việc đọc và ghi từ Socket.

Lớp java.net.Socket biểu diễn một Socket, và lớp java.net.ServerSocket cung cấp một kỹ thuật cho chương trình Server để nghe thông tin từ các Client và thành lập các kết nối với chúng.

Mô hình Client-Server sử dụng Socket ở chế độ có kếtnối kết (TCP)

Có thể phân thành 4 giai đoạn như sau:

* **Giai đoạn 1**: Server tạo Socket, gán số hiệu cổng và lắng nghe yêu cầu nối kết. Server sẵn sàng phục vụ Client.socket(): Server yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển.
  + bind(): Server yêu cầu gán số hiệu cổng (port) cho socket.
  + listen(): Server lắng nghe các yêu cầu nối kết từ các client trên cổng đã được gán.
* **Giai đoạn 2**: Client tạo Socket, yêu cầu thiết lập một nối kết với Server.
  + socket(): Client yêu cầu tạo một socket để có thể sử dụng các dịch vụ của tầng vận chuyển, thông thường hệ thống tự động gán một số hiệu cổng còn rảnh cho socket của Client.
  + connect(): Client gởi yêu cầu nối kết đến server có địa chỉ IP và Port xác định.
  + accept(): Server chấp nhận nối kết của client, khi đó một kênh giao tiếp ảo được hình thành, Client và server có thể trao đổi thông tin với nhau thông qua kênh ảo này.
* **Giai đoạn 3**: Trao đổi thông tin giữa Client và Server.
  + Sau khi chấp nhận yêu cầu nối kết, thông thường server thực hiện lệnh read() và nghẽn cho đến khi có thông điệp yêu cầu (Request Message) từ client gởi đến.
  + Server phân tích và thực thi yêu cầu. Kết quả sẽ được gởi về client bằng lệnh write().
  + Sau khi gởi yêu cầu bằng lệnh write(), client chờ nhận thông điệp kết quả (ReplyMessage) từ server bằng lệnh read().
* **Giai đoạn 4**: Kết thúc phiên làm việc.
  + Các câu lệnh read(), write() có thể được thưc hiện nhiều lần (ký hiệu bằng hình ellipse).
  + Kênh ảo sẽ bị xóa khi Server hoặc Client đóng socket bằng lệnh close().

## Ví dụ về Socket

package vuta.socket.example;

import java.net.\*;

import java.io.\*;

class MyServer {

public static void main(String args[]) throws Exception {

ServerSocket ss = new ServerSocket(3333);

Socket s = ss.accept();

DataInputStream din = new DataInputStream(s.getInputStream());

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = "", str2 = "";

while (!str.equals("stop")) {

str = din.readUTF();

System.out.println("client says: " + str);

str2 = br.readLine();

dout.writeUTF(str2);

dout.flush();

}

din.close();

s.close();

ss.close();

}

}

package vuta.socket.example;

import java.net.\*;

import java.io.\*;

class MyClient {

public static void main(String args[]) throws Exception {

Socket s = new Socket("localhost", 3333);

DataInputStream din = new DataInputStream(s.getInputStream());

DataOutputStream dout = new DataOutputStream(s.getOutputStream());

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

String str = "", str2 = "";

while (!str.equals("stop")) {

str = br.readLine();

dout.writeUTF(str);

dout.flush();

str2 = din.readUTF();

System.out.println("Server says: " + str2);

}

dout.close();

s.close();

}

}