今日目标及课程安排

- 网络编程中的Socket以及ServerSocket应用
- 网络通讯的基本过程及原理

Socket入门

何为Socket?

java中用于网络通讯的一个对象,用于实现客户端与服务端的通讯.

Socket 入门案例?

- 需求:客户端向服务端写一句话,服务端进行输出.
- 实现步骤:

- 编写服务器类Server (创建服务对象,启动服务,接收客户端的连接,处理连接请求)
- 编写客户端类Client (创建客户端对象,建立连接,向服务器发送请求)
- 运行Server和Client对象(要注意先后顺序,先启动Server,再启动Client)

- 代码实现:

- 服务端代码实现:SimpleServer

```
package net;
public class SimpleServer{
    public static void main(String[] args)throws Exception{
        //1.创建Server对象,并申请8088端口,然后在此端口上进行服务监听
        ServerSocket server=new ServerSocket(8088);
        System.out.println("服务启动成功");
        //2.接收客户端的连接,并为客户端分配socket对象
        Socket socket=server.accept();
        System.out.println("有新客户端连接到了服务器");
        //3.创建流对象,读取Client端发送的数据
        ObjectInputStream in=new ObjectInputStream(socket.getInputStream());
        String content=in.readUTF();
        System.out.println(content);
    }
}
```

- 客户端代码实现:SimpleClient

```
package net;

import java.io.IOException;
import java.io.ObjectOutputStream;

public class SimpleClient {
```

```
    public static void main(String[] args) throws IOException {
        //1.创建Socket对象并与服务器建立连接
        Socket socket = new Socket("127.0.0.1", 8088);
        //2.向服务端发送数据
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        out.writeUTF("hello 8088");
        out.close();
        //3.释放资源
        socket.close();
    }

}
```

Socket 案例增强

- 需求:
- 客户端从键盘输入数据,然后向服务端写数据,这个过程可以反复执行
- 服务端可以从客户端不断地接收数据,然后将数据输出.
- 代码实现:
- 服务端代码实现:SimpleServerPlus

```
  package net;
  public class SimpleServerPlus{
      public static void main(String[] args)throws Exception{
          //1.创建Server对象,并申请8088端口,然后在此端口上进行服务监听
          ServerSocket server=new ServerSocket(8088);
          System.out.println("服务启动成功");
          //2.接收客户端的连接,并为客户端分配socket对象
          Socket socket=server.accept();
          System.out.println("有新客户端连接到了服务器");
          //3.创建流对象,读取Client端发送的数据
          ObjectInputStream in=new ObjectInputStream(socket.getInputStream());
          while(true) {
              String content = in.readUTF();
              System.out.println(content);
              if("exit".equals(content))break;
          }
      }
  }
```

- 客户端代码实现:SimpleClientPlus

```
package net;

import java.io.IOException;
import java.io.ObjectOutputStream;
import java.util.Scanner;
```

```
public class SimpleClientPlus {

    public static void main(String[] args) throws IOException {
        //1.创建Socket对象并与服务器建立连接
        Socket socket = new Socket("127.0.0.1", 8088);
        //2.向服务端发送数据
        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
        Scanner scanner=new Scanner(System.in);
        while(true) {
            String content = scanner.nextLine();
            out.writeUTF(content);
            out.flush();
            if("exit".equals(content))break;
        }
        out.close();
        //3.释放资源
        socket.close();
    }

}
```

简易聊天程序实现

- 服务端设计及实现

```
package net;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * 在这个Server实现两个功能:
 * 1.读取客户端写入到服务端的数据并输出
 * 2.向客户端写数据
 * 技术上的实现:
 * 1.在SimpleChatServer的构造方法中构建ServerSocket对象
 * 2.在SimpleChatServer类中定义start方法,在方法内部接收客户端的连接,然后读,写客户端数据
 * 3.读写数据时要求使用BufferedReader,PrintWriter
 * 4.所有方法内部都自己处理异常
 */
public class SimpleChatServer {
    private ServerSocket server;
    public SimpleChatServer(){
        try {
            server = new ServerSocket(9999);
            System.out.println("Server Start OK!");
        }catch (IOException e){
            e.printStackTrace();//日志记录
            throw new RuntimeException("服务器启动失败");
```

```
        }
    }
    public void start(){
        //这里的循环表示可以接收多个客户端的连接请求
        while(true){
            try {
                //接收客户端连接
                Socket socket = server.accept();
                System.out.println("客户端来了");
                //构建流对象读写数据
                BufferedReader reader=
                        new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                String msg=null;
                //可以不断的读取客户端写入到服务端的数据
                while(true) {
                    msg=reader.readLine();
                    if("exit".equals(msg))break;
                    System.out.println("来自客户端的数据:" + msg);
                }
                PrintWriter writer=new PrintWriter(socket.getOutputStream());
                writer.println("bye bye");
                writer.flush();
            }catch (IOException e){
                e.printStackTrace();
            }
        }
    }
    public static void main(String[] args) {
        SimpleChatServer server=new SimpleChatServer();
        server.start();
    }
}
```

- 客户端设计及实现

```
package net;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

/**
 * 这个Socket客户端有个需求:
 * 1.向服务端写数据
 * 2.从服务端读数据
 * 技术上的实现:
 * 1.在SimpleChatClient的构造方法中构建Socket对象
 * 2.在SimpleChatClient定义start方法,在方法内部写,读服务端数据
 * 3.读写数据时要求使用BufferedReader,PrintWriter
 * 4.所有方法内部都有自己处理异常
```

```
 */
public class SimpleChatClient {

    private Socket socket;
    public SimpleChatClient(){
        try {
            socket=new Socket("127.0.0.1",9999);
            System.out.println("连接成功了");
        } catch (IOException e) {
            e.printStackTrace();
            throw new RuntimeException("连接失败了");
        }
    }
    public void start(){
        try {
            PrintWriter pw = new PrintWriter(socket.getOutputStream());
            Scanner scanner=new Scanner(System.in);
            //不断从键盘输入数据,然后写入到服务端
            while(true) {
                String msg=scanner.nextLine();
                pw.println(msg);
                pw.flush();
                if("exit".equals(msg))break;
            }
            BufferedReader reader=
                    new BufferedReader(
                            new InputStreamReader(socket.getInputStream()));
            String content=reader.readLine();
            System.out.println("来自服务端的数据:"+content);
            pw.close();
            reader.close();
            socket.close();
        }catch (IOException e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        SimpleChatClient client=new SimpleChatClient();
        client.start();
    }
}
```

总结(Summary)

重难点分析

- Socket是什么?(Java中的一组网络通讯API,用于实现网络通讯)
- Socket常用API有哪些?(Socket,ServerSocket)
- 基于Socket实现网络编程的基本步骤?(服务器,客户端,获取流对象,读写数据,释放资源)

FAQ分析

- Java中实现网络通讯的API有哪些?(Socket,ServerSocket)

- 网络中计算机的唯一标识是什么?(IP地址)
- 计算机中应用程序的唯一标识是什么?(端口号)
- 启动Server时是否要指定端口?(要,建议提供的端口要大于1024,小于65535)
- ServerSocket中的accept方法作用什么?(等待客户端的连接,返回值是一个socket对象)
- 网络通讯中如何读写网路中数据?(IO流)

Buf分析

- 端口被占用
- 连接已关闭