Sends the specified text or file as a datagram to the specified port of the specified host : DatagramPacket « Network Protocol « Java

Sends the specified text or file as a datagram to the specified port of the specified host

```
/*
 * Copyright (c) 2004 David Flanagan.  All rights reserved.
 * This code is from the book Java Examples in a Nutshell, 3nd Edition.
 * It is provided AS-IS, WITHOUT ANY WARRANTY either expressed or implied.
 * You may study, use, and modify it for any non-commercial purpose,
 * including teaching and use in open-source projects.
 * You may distribute it non-commercially as long as you retain this notice.
 * For a commercial use license, or to purchase the book,
 * please visit http://www.davidflanagan.com/javaexamples3.
 */
//package je3.net;
import java.io.File;
import java.io.FileInputStream;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;

/**
 * This class sends the specified text or file as a datagram to the specified
 * port of the specified host.
 */
public class UDPSend {
  public static final String usage = "Usage: java UDPSend <hostname> <port> <msg>...\n"
      + "   or: java UDPSend <hostname> <port> -f <file>";

  public static void main(String args[]) {
    try {
      // Check the number of arguments
      if (args.length < 3)
        throw new IllegalArgumentException("Wrong number of args");

      // Parse the arguments
      String host = args[0];
      int port = Integer.parseInt(args[1]);

      // Figure out the message to send.
      // If the third argument is -f, then send the contents of the file
      // specified as the fourth argument. Otherwise, concatenate the
      // third and all remaining arguments and send that.
      byte[] message;
      if (args[2].equals("-f")) {
        File f = new File(args[3]);
        int len = (int) f.length(); // figure out how big the file is
        message = new byte[len]; // create a buffer big enough
        FileInputStream in = new FileInputStream(f);
        int bytes_read = 0, n;
        do { // loop until we've read it all
          n = in.read(message, bytes_read, len - bytes_read);
          bytes_read += n;
        } while ((bytes_read < len) && (n != -1));
      } else { // Otherwise, just combine all the remaining arguments.
        String msg = args[2];
        for (int i = 3; i < args.length; i++)
          msg += " " + args[i];
        // Convert the message to bytes using UTF-8 encoding
        message = msg.getBytes("UTF-8");
      }

      // Get the internet address of the specified host
      InetAddress address = InetAddress.getByName(host);

      // Initialize a datagram packet with data and address
      DatagramPacket packet = new DatagramPacket(message, message.length, address, port);

      // Create a datagram socket, send the packet through it, close it.
      DatagramSocket dsocket = new DatagramSocket();
      dsocket.send(packet);
      dsocket.close();
    } catch (Exception e) {
      System.err.println(e);
      System.err.println(usage);
    }
  }
}
```

XML
(/Code/Java/XML/CatalogXML.htm
)

Related examples in the same category

1. Get data from DatagramPacket (/Code/Java/Network-Protocol/GetdatafromDatagramPacket.htm)
2. Get address and port from DatagramPacket (/Code/Java/Network-Protocol/GetaddressandportfromDatagramPacket.htm)
3. Create DatagramPacket from byte array (/Code/Java/Network-Protocol/CreateDatagramPacketfrombytearray.htm)
4. Waits to receive datagrams sent the specified port (/Code/Java/Network-Protocol/Waitstoreceivedatagramssentthespecifiedport.htm)

(/Code/Java/Netv
Protocol/Waitstoreceivedatagramssentthespecifiedport.