IT NetworkZ

LETS GET INTERNETWORKED

Contents

- Introduction of Java Networking
- Networking Terminology
- java.net package
- URL Class
- URLConnection Class
- HttpURLConnection class
- InetAddress class
- DatagramSocket and DatagramPacket
- Socket Programming

Networking

- Java Networking is a concept of connecting two or more computing devices together so that we can share resources.
- Java socket programming provides facility to share data between different computing devices.

Networking

- Advantage of Java Networking
- sharing resources
- centralize software management

Networking Terminology

- IP Address
- Protocol
- Port Number
- MAC Address
- Connection-oriented and connection-less protocol
- Socket

Networking Terminology

1) IP Address

- IP address is a unique number assigned to a node of a network e.g. 192.168.0.1 . It is composed of octets that range from 0 to 255.
- It is a logical address that can be changed.

2) Protocol

A protocol is a set of rules basically that is followed for communication.

For example:

- TCP
- FTP
- Telnet
- SMTP
- POP etc.

Networking Terminology

3) Port Number

- The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.
- The port number is associated with the IP address for communication between two applications.

4) MAC Address

- MAC (Media Access Control) Address is a unique identifier of NIC (Network Interface Controller). A network node can have multiple NIC but each with unique MAC.

Networking Terminology

5) Connection-oriented and connection-less protocol

- In connection-oriented protocol, acknowledgement is sent by the receiver. So it is reliable but slow. The example of connection-oriented protocol is TCP.
- But, in connection-less protocol, acknowledgement is not sent by the receiver. So it is not reliable but fast. The example of connection-less protocol is UDP.

6) Socket

- A socket is an endpoint between two way communication.

java.net package

The java.net package provides many classes to deal with networking applications in Java.

- Authenticator
- CacheRequest
- CacheResponse
- ContentHandler
- CookieHandler
- CookieManager
- DatagramPacket
- DatagramSocket
- DatagramSocketImpl
- InterfaceAddress
- JarURLConnection

java.net package

- MulticastSocket
- InetSocketAddress
- InetAddress
- Inet4Address
- Inet6Address
- IDN
- HttpURLConnection
- HttpCookie
- NetPermission
- NetworkInterface
- PasswordAuthentication
- Proxy
- ProxySelector
- ResponseCache

java.net package

- SecureCacheResponse
- ServerSocket
- Socket
- SocketAddress
- SocketImpl
- SocketPermission
- StandardSocketOptions
- URI
- URL
- URLClassLoader
- URLConnection
- URLDecoder
- URLEncoder
- URLStreamHandler

URL Class

The **Java URL** class represents an URL. URL is an acronym fo Uniform Resource Locator. It points to a resource on the World Wide Web.

For example:

https://www.google.com/index.jsp

URL Class

A URL contains many information:

- Protocol
- Server name or IP Address
- Port Number
- File Name or directory name

URL Class

A URL contains many information:

- Protocol
- Server name or IP Address
- Port Number
- File Name or directory name

URL Class

Constructors of Java URL class

URL(String protocol, String host, int port, String file)

Creates an instance of a URL from the given protocol, host, port number, and file.

URL(String protocol, String host, int port, String file, URLStreamHandler handler)

Creates an instance of a URL from the given protocol, host, port number, file, and handler.

URL(String protocol, String host, String file)

Creates an instance of a URL from the given protocol name, host name, and file name.

URL(URL context, String spec)

Creates an instance of a URL by parsing the given spec within a specified context.

URL(URL context, String spec, URLStreamHandler handler)

Creates an instance of a URL by parsing the given spec with the specified handler within a given context.

Methods of Java URL class

Method	Description
public String getProtocol()	it returns the protocol of the URL.
public String getHost()	it returns the host name of the URL.
public String getPort()	it returns the Port Number of the URL.
public String getFile()	it returns the file name of the URL.

Methods of Java URL class

Method	Description
public String getAuthority()	it returns the authority of the URL.
public String toString()	it returns the string representation of the URL.
public String getQuery()	it returns the query string of the URL.
public String getDefaultPort()	it returns the default port of the URL.

Methods of Java URL class

Method	Description
public URLConnection openConnection()	it returns the instance of URLConnection i.e. associated with this URL.
public boolean equals(Object obj)	it compares the URL with the given object.
public Object getContent()	it returns the content of the URL.
public String getRef()	it returns the anchor or reference of the URL.

Example of Java URL class

```
//URLDemo.java
import java.net.*;
 public class URLDemo{
 public static void main(String[] args){
 try{
   URL url=new URL("http://www.google.com/java-tutorial");

  System.out.println("Protocol: "+url.getProtocol());
  System.out.println("Host Name: "+url.getHost());
  System.out.println("Port Number: "+url.getPort());
  System.out.println("File Name: "+url.getFile());

}catch(Exception e){System.out.println(e);}
}
}
```

output:

Protocol: http

Host Name: www.google.com

Port Number: -1

File Name: /java-tutorial

output:

Protocol: http

Host Name: www.google.com

Port Number: -1

File Name: /java-tutorial

URLConnection class

The **Java URLConnection** class represents a communication link between the URL and the application. This class can be used to read and write data to the specified resource referred by the URL.

URLConnection class

The object of URLConnection class

The openConnection() method of URL class returns the object of URLConnection class.

Syntax:

public URLConnection openConnection()**throw** IOException{}

URLConnection class

The object of URLConnection class

The openConnection() method of URL class returns the object of URLConnection class.

Syntax:

public URLConnection openConnection()**throw** IOException{}

URLConnection class methods

The URLConnection class provides many methods, we can display all the data of a webpage by using the getInputStream() method. The getInputStream() method returns all the data of the specified URL in the stream that can be read and displayed.

Example of Java URLConnection class

```
import java.io.*;
import java.net.*;
public class URLConnectionExample {
   public static void main(String[] args){
    try{
         URL url=new URL("http://www.javatpoint.com/java-tutorial");
         URLConnection urlcon=url.openConnection();
         InputStream stream=urlcon.getInputStream();
         int i;
         while((i=stream.read())!=-1){
          System.out.print((char)i);
         }
      }
     catch(Exception e){System.out.println(e);}
    }
}
```

HttpURLConnection class

- The **Java HttpURLConnection** class is http specific URLConnection. It works for HTTP protocol only.
- By the help of HttpURLConnection class, you can information of any HTTP URL such as header information, status code, response code etc.
- The java.net.HttpURLConnection is subclass of URLConnection class.

HttpURLConnection class

- The **Java HttpURLConnection** class is http specific URLConnection. It works for HTTP protocol only.
- By the help of HttpURLConnection class, you can information of any HTTP URL such as header information, status code, response code etc.
- The java.net.HttpURLConnection is subclass of URLConnection class.

HttpURLConnection class methods

The openConnection() method of URL class returns the object of URLConnection class.

Syntax:

public URLConnection openConnection()**throws** IOException{}

HttpURLConnecton Example

```
import java.io.*;
import java.net.*;
public class HttpURLConnectionDemo{
public static void main(String[] args){
    try{
     URL url=new URL("http://www.javatpoint.com/java-tutorial");
     HttpURLConnection huc=(HttpURLConnection)url.openConnection();
     for(int i=1;i<=8;i++){
            System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getHeaderField(i));
      }
     huc.disconnect();
     }catch(Exception e){System.out.println(e);}
    }
}
```

HttpURLConnecton Example

```
import java.io.*;
import java.net.*;
public class HttpURLConnectionDemo{
public static void main(String[] args){
    try{
     URL url=new URL("http://www.javatpoint.com/java-tutorial");
     HttpURLConnection huc=(HttpURLConnection)url.openConnection();
     for(int i=1;i<=8;i++){
            System.out.println(huc.getHeaderFieldKey(i)+" = "+huc.getHeaderField(i));
      }
     huc.disconnect();
     }catch(Exception e){System.out.println(e);}
    }
}
```

output:

Date = Wed, 10 Dec 2019 19:31:14 GMT
Set-Cookie = JSESSIONID=D70B87DBB832820CACA5998C90939D48; Path=/
Content-Type = text/html
Cache-Control = max-age=2592000
Expires = Fri, 09 Jan 2015 19:31:14 GMT
Vary = Accept-Encoding,User-Agent
Connection = close
Transfer-Encoding = chunked

InetAddress class

Java InetAddress class represents an IP address.

The java.net.InetAddress class provides methods to get the IP of any host name *for example* www.google.com, www.facebook.com, etc.

InetAddress class

Java InetAddress class represents an IP address.

The java.net.InetAddress class provides methods to get the IP of any host name *for example* www.google.com, www.facebook.com, etc.

Methods of InetAddress class

Method	Description
public static InetAddress getByName(String host) throws UnknownHostException	it returns the instance of InetAddress containing LocalHost IP and name.
public static InetAddress getLocalHost() throws UnknownHostException	it returns the instance of InetAdddress containing local host name and address.

Methods of InetAddress class

Method	Description
public String getHostName()	it returns the host name of the IP address.
public String getHostAddress()	it returns the IP address in string format.

Example of Java InetAddress class

```
import java.io.*;
import java.net.*;
public class InetDemo{
public static void main(String[] args){
   try{
    InetAddress ip=InetAddress.getByName("www.google.com");
System.out.println("Host Name: "+ip.getHostName());
System.out.println("IP Address: "+ip.getHostAddress());
 }catch(Exception e){System.out.println(e);}
 }
}
```

output:

Host Name: www.google.com

IP Address: 206.51.231.148

DatagramSocket and DatagramPacket

DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

DatagramSocket class

Java DatagramSocket class represents a connection-less socket for sending and receiving datagram packets.

A datagram is basically an information but there is no guarantee of its content, arrival or arrival time.

Constructors of DatagramSocket class

- **DatagramSocket() throws SocketEeption:** it creates a datagram socket and binds it with the available Port Number on the localhost machine.
- **DatagramSocket(int port) throws SocketEeption:** it creates a datagram socket and binds it with the given Port Number.
- **DatagramSocket(int port, InetAddress address) throws SocketEeption:** it creates a datagram socket and binds it with the specified port number and host address.

DatagramPacket class

Java DatagramPacket is a message that can be sent or received. If you send multiple packet, it may arrive in any order. Additionally, packet delivery is not guaranteed.

Constructors of DatagramPacket class

- **DatagramPacket(byte[] barr, int length):** it creates a datagram packet. This constructor is used to receive the packets.
- **DatagramPacket(byte[] barr, int length, InetAddress address, int port):** it creates a datagram packet. This constructor is used to send the packets.

Example of Sending DatagramPacket by DatagramSocket

```
//DSender.java
import java.net.*;
public class DSender{
  public static void main(String[] args) throws Exception {
    DatagramSocket ds = new DatagramSocket();
    String str = "Welcome java";
    InetAddress ip = InetAddress.getByName("127.0.0.1");
DatagramPacket dp = new DatagramPacket(str.getBytes(), str.length(),
 ip, 3000);
    ds.send(dp);
    ds.close();
  }
}
```

Example of Receiving DatagramPacket by DatagramSocket

```
//DReceiver.java
import java.net.*;
public class DReceiver{
  public static void main(String[] args) throws Exception {
    DatagramSocket ds = new DatagramSocket(3000);
    byte[] buf = new byte[1024];
    DatagramPacket dp = new DatagramPacket(buf, 1024);
    ds.receive(dp);
    String str = new String(dp.getData(), 0, dp.getLength());
    System.out.println(str);
    ds.close();
  }
}
```

Socket Programming

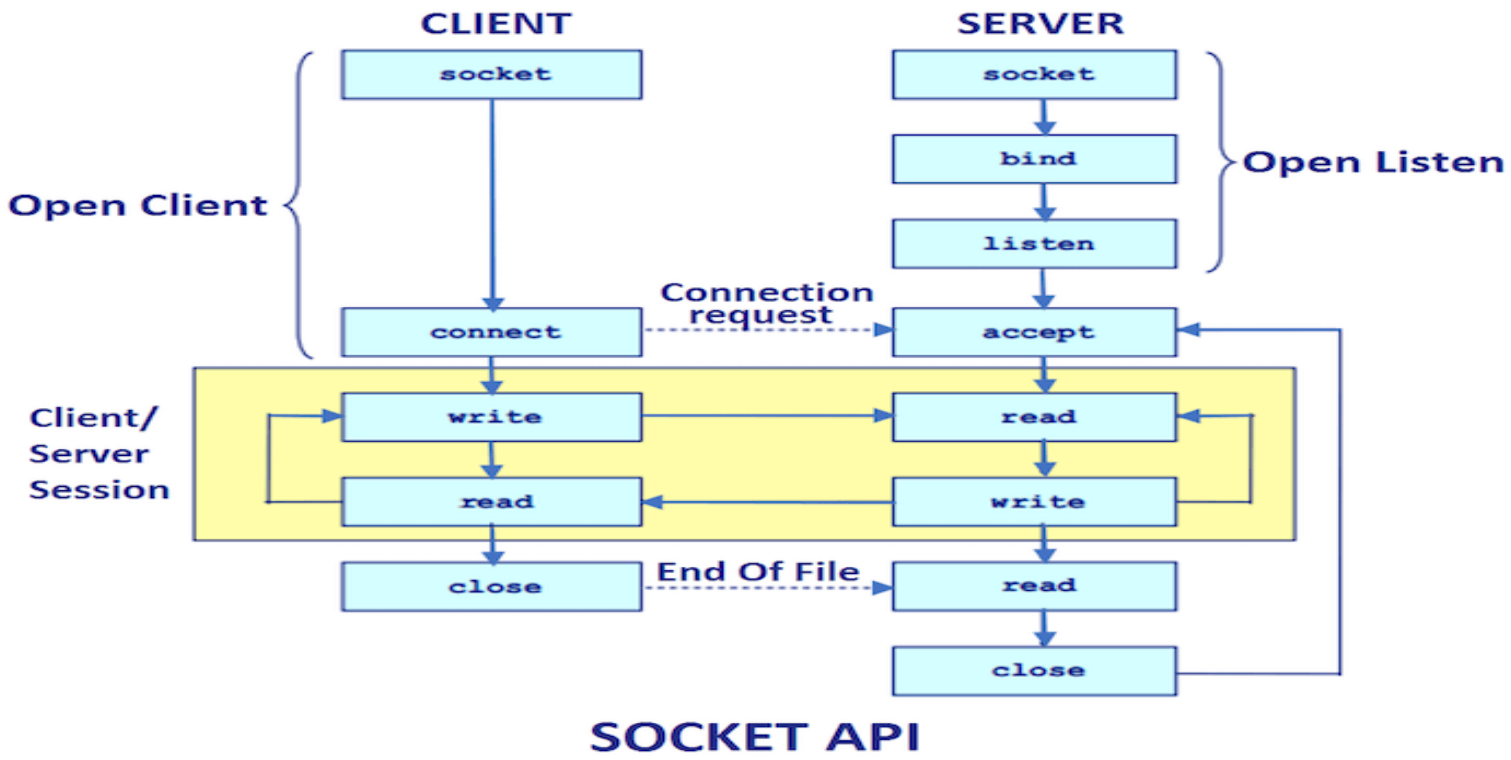
- Java Socket programming is used for communication between the applications running on different JRE.
- Java Socket programming can be connection-oriented or connection-less.
- Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

Socket Programming

The client in socket programming must know two information:

- IP Address of Server, and
- Port number.

Socket Programming



IT NetworkZ
LETS GET INTERNETWORKED

Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Socket class

Important methods

Method	Description
1) public InputStream getInputStream()	returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

ServerSocket class

Important methods

Method	Description
1) public Socket accept()	returns the socket and establish a connection between server and client.
2) public synchronized void close()	closes the server socket.

Example of Java Socket Programming

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept() method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket(6666);
Socket s=ss.accept();//establishes connection and waits for the client
```

Example of Java Socket Programming

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

Socket s=**new** Socket("localhost",6666);

Java socket programming

File: MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer {
public static void main(String[] args){
try{
    ServerSocket ss=new ServerSocket(6666);
    Socket s=ss.accept();//establishes connection
    DataInputStream dis=new DataInputStream(s.getInputStream());
    String  str=(String)dis.readUTF();
    System.out.println("message= "+str);
    ss.close();
    }catch(Exception e){System.out.println(e);}
  }
}
```

Java socket programming

File: MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient {
 public static void main(String[] args) {
    try{
    Socket s=new Socket("localhost",6666);
    DataOutputStream dout=new DataOutputStream(s.getOutputStream());
    dout.writeUTF("Hello Server");
    dout.flush();
    dout.close();
     s.close();
     }catch(Exception e){System.out.println(e);}
   }
}
```

output:

IT NetworkZ
LETS GET INTERNETWORKED

Thank You.....

IT NetworkZ

LETS GET INTERNETWORKED