JAVA NETWORKING

By Oswaldo Valles Azpeitia

El término networking se asocia con la escritura de programas que se pueden ejecutar en varios dispositivos, en los que todos los dispositivos están conectados entre sí para compartir recursos mediante una red.

Java Networking es una noción de combinar dos o más dispositivos informáticos para compartir recursos.

Protocolos Network

1. Transmission Control Protocol (TCP). Permite la comunicación segura entre diferentes aplicaciones. TCP es un protocolo orientado a la conexión, lo que significa que una vez que se establece una conexión, los datos se pueden transmitir en dos direcciones.

2. User Datagram Protocol (UDP). Es un protocolo sin conexión que permite que los paquetes de datos se transmitan entre diferentes aplicaciones. UDP es un protocolo de Internet más simple en el que no se requieren servicios de recuperación y verificación de errores.

Terminología

1. IP Adress. Una dirección IP es una dirección única que distingue a un dispositivo en Internet o en una red local. IP significa "Protocolo de Internet". Comprende un conjunto de reglas que rigen el formato de los datos enviados a través de Internet o una red local.

2. Port Number. Un número de puerto es un método para reconocer un proceso particular que conecta Internet u otra información de red cuando llega a un servidor.

3.Protocol. Un protocolo de red es un conjunto organizado de comandos que definen cómo se transmiten los datos entre diferentes dispositivos en la misma red.

4.MAC Address. es un número único que se utiliza para rastrear un dispositivo en una red.

5.Socket. Es un punto final de una conexión de comunicación bidireccional entre las dos aplicaciones que se ejecutan en la red.

Clases (java.net)

- CacheRequest. Se usa en Java siempre que sea necesario almacenar recursos en ResponseCache.

- CookieHandler. Se utiliza en Java para implementar un mecanismo de devolución de llamada para asegurar la implementación de una política de administración de estado HTTP dentro del controlador de protocolo HTTP.

- CookieManager. Se utiliza para proporcionar una implementación precisa de CookieHandler.

- DatagramPacket. Se utiliza para proporcionar una facilidad para la transferencia de mensajes sin conexión de un sistema a otro.

- InetAddress. Se utiliza para proporcionar métodos para obtener la dirección IP de cualquier nombre de host.

- Server Socket. Se utiliza para implementar la implementación independiente del sistema del lado del servidor de una conexión de socket cliente/servidor.

- Socket. Se utiliza para crear objetos de socket que ayudan a los usuarios a implementar todas las operaciones de socket fundamentales.

- DatagramSocket. Es un socket de red que proporciona un punto sin conexión para enviar y recibir paquetes.

- Proxy. Es un objeto inmutable y una especie de herramienta o método o programa o sistema, que sirve para preservar los datos de sus usuarios y computadoras.

- Socket. Se utiliza para crear objetos de socket que ayudan a los usuarios a implementar todas las operaciones de socket fundamentales.

- DatagramSocket. Es un socket de red que proporciona un punto sin conexión para enviar y recibir paquetes.

- Proxy. Es un objeto inmutable y una especie de herramienta o método o programa o sistema, que sirve para preservar los datos de sus usuarios y computadoras.

- URL. Es el punto de entrada a cualquier fuente disponible en Internet.

- URLConnection. Es una clase abstracta que describe una conexión de un recurso definido por una URL similar.

Interfaces

+ CookiePolicy: Proporciona las clases para implementar varias aplicaciones de red.

+ CookieStore: Interfaz que describe un espacio de almacenamiento para cookies.

+ FileNameMap: es una interfaz sencilla que implementa una herramienta para delinear un nombre de archivo y una cadena de tipo MIME.

+ SocketOption: Ayuda a los usuarios a controlar el comportamiento de los sockets.

+ SocketImplFactory: Define una fábrica para instancias de SocketImpl.

+ ProtocolFamily: Representa una familia de protocolos de comunicación.

SocketClass

MÉTODO	DESCRIPCIÓN
public void connect(SocketAddress host, int timeout)	Se utiliza para conectar el socket al host particularizado.
public int getPort()	Se utiliza para devolver el puerto al que está conectado el zócalo en la máquina remota.
public InetAddress getInetAddress()	Se utiliza para devolver la ubicación de la otra computadora a la que está conectado el zócalo.
public int getLocalPort()	Se utiliza para devolver el puerto al que se une el socket en la máquina local.
public SocketAddress getRemoteSocketAddress()	Devuelve la ubicación del socket remoto.

MÉTODO	DESCRIPCIÓN
public InputStream getInputStream()	Se utiliza para devolver el flujo de entrada del socket.
public OutputStream getOutputStream()	Se utiliza para devolver el flujo de salida del socket.
public void close()	Se utiliza para cerrar el enchufe

ServerSocketClass

MÉTODO	DESCRIPCIÓN
public int getLocalPort()	Se utiliza para devolver el puerto que el socket del servidor está monitoreando.
public void setSoTimeout(int timeout)	Se utiliza para establecer el valor de tiempo de espera para el tiempo en que el socket del servidor se detiene para un cliente durante el método accept().
public Socket accept()	Espera a un cliente entrante.(masculino)
public void bind(SocketAddress host, int backlog)	Se utiliza para vincular el socket al servidor y puerto particularizados en el objeto de SocketAddress.

InetAdress

MÉTODO	DESCRIPCIÓN
static InetAddress getByAddress(byte[] addr)	Se utiliza para devolver un objeto de la clase InetAddress siempre que la dirección IP sin formato.
static InetAddress getByAddress(String host, byte[] addr)	Se utiliza para crear una InetAddress basada en el nombre de host y la dirección IP dados.
static InetAddress getByName(String host)	Se utiliza para determinar la dirección IP de un host cuando se proporciona el nombre del host.
static InetAddress InetAddress getLocalHost()	Se utiliza para devolver el host local.

MÉTODO	**DESCRIPCIÓN**
String getHostName()	Se utiliza para obtener el nombre de la dirección IP.
String getHostAddress()	Devuelve la dirección IP en forma de cadena en una pantalla de texto.
String toString()	Se utiliza para convertir la dirección IP en una cadena.

URL

MÉTODO	DESCRIPCIÓN
public String getProtocol()	Devuelve el protocolo que utiliza la URL.
public String getHost()	Devuelve el nombre de host de la URL en la composición de IPv6.
public int getPort()	Devuelve el puerto asociado con el protocolo especificado por la URL.
public String getFile()	Devuelve el nombre del archivo.
public String getPath()	Devuelve la ruta de la URL, o nulo si está vacío.

MÉTODO	DESCRIPCIÓN
public String toString()	Se utiliza para devolver la representación de cadena del objeto URL proporcionado.
public int getDefaultPort()	Devuelve el puerto predeterminado utilizado.

Ejercicio

Con los métodos del tema URL

- Acceder a una URL
- Imprimir su protocolo
- Imprimir el nombre del Host
- Imprimir el nombre del archivo