

Tema 3: Java y los Sockets

ORACLE (Sign In/Register for Account | Help) United States Communities I am a... I want to... Secure Search

Products and Services Solutions Downloads Store Support Training Partners About Oracle Technology Network

Oracle Technology Network > Java

MOVING FORWARD

JAVA

FORWARD

RELEASE

JavaFX 2.0 Arrives and Will Be Open Sourced

Announced at JavaOne, JavaFX 2.0 has arrived with Java APIs for JavaFX, FXML, a new graphics pipeline for modern GPUs, and more than 50 easily customizable components. The source code will be released at OpenJDK as project OpenJFX.

Posted 10/3/11 // Tags: java, javafx, RIA // Headlines Archive

Software Downloads

View All Downloads

Top Downloads

- Java SE
- Java EE and GlassFish
- Java FX
- Java ME
- JDeveloper 11g and ADF
- Enterprise Pack for Eclipse
- NetBeans IDE
- Pre-Built VM for Java Developers

Get Java

New Downloads

- Java SE 7 Update 1
Released 10/18/11
- Java SE 6 Update 29
Released 10/18/11
- JavaFX 2.0.1
Released 10/18/11
- NetBeans 7.1 Beta
Released 10/4/11
- GlassFish Server 3.1.1
Released 7/28/11
- Oracle JDeveloper 11g R2
Released 6/6/11

Essential Links

- About Us/Become a Member
- Java APIs
- Technical Articles
- New to Java
- Java Certification & Training
- Java Bug Database
- "The Java Source" Blog
- Java on Twitter
- Java Developer Newsletter
- Java Magazine
- Delicious Feed
- Java.net
- Java User Groups
- Java Community Process
- Facebook | Forums
- Events | Developer Days

Developer Spotlight

- Using MWait in Spin Loops
- New LWUIT Tutorial
- Oracle Enterprise Pack for Eclipse with Support for ADF, Coherence and Public Cloud
- How to Build a Successful Java User Groups (JUG)
- Why Developers Should Not Write Programs That Call 'sun' Packages
- Java Reloaded
- JDK 7 Adoption Guide
- JCP Executive Committee Elections
- The Importance of Twitter's Participation in OpenJDK
- Javatuples 1.2 Released
- JavaOne Content on Parleys.com
- Introducing the Oracle Java Cloud Service: For Standard Java EE Apps
- JavaFX 2.0 and Scala, Like Milk and Cookies
- Java 7 Summit at EclipseConEurope (Germany, 02-04 Nov.)

Blogs

- Latest OEPE (11.1.1.8) - Eclipse 3.7.1-based
Posted: Nov 9
- Java Spotlight Episode 55: Georges Saab, Vice President of Development in the Java Platform Group
Posted: Nov 9
- Changing a Node from its PopUp Menu
Posted: Nov 8
- Devxxx for Java Developers
Posted: Nov 8
- The Last Migration - GlassFish Wiki
Posted: Nov 8
- Tip: "Unset as Main Project"
Posted: Nov 8
- Put Siri-like tech on your Java ME mobile phone with Zvpr
Posted: Nov 8
- GlassFish 3.1.2 themes and features
Posted: Nov 7
- On the methodvalue attribute in layer.xml file
Posted: Nov 6

Technologies

- Java SE
- Java SE Support
- Java SE Advanced & Suite
- Java Embedded
- Java EE
- Java ME
- JavaFX
- Java Card
- Java TV
- Java DB
- Developer Tools

Java Spotlight Podcast

- Java Spotlight Episode 54: Stuart Marks on the Coincification of JDK7
Posted: Oct 2
- Java Spotlight Episode 53: Mark Reinhold, Chief Architect of the Java Platform Group
Posted: Oct 25
- Java Spotlight Episode 52: Cameron Purdy, Vice President of Development at Oracle, on JavaEE
Posted: Oct 18

JavaOne

NEW!

Java magazine Get it now for FREE!

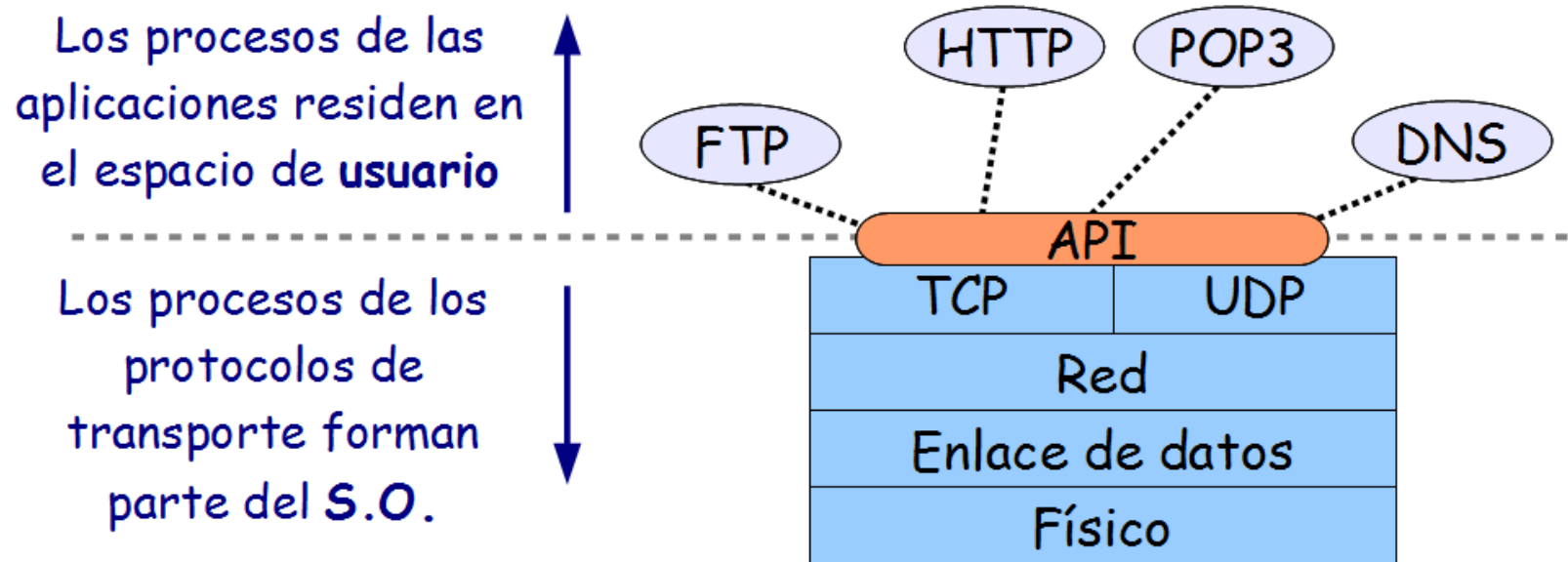
Subscribe Today



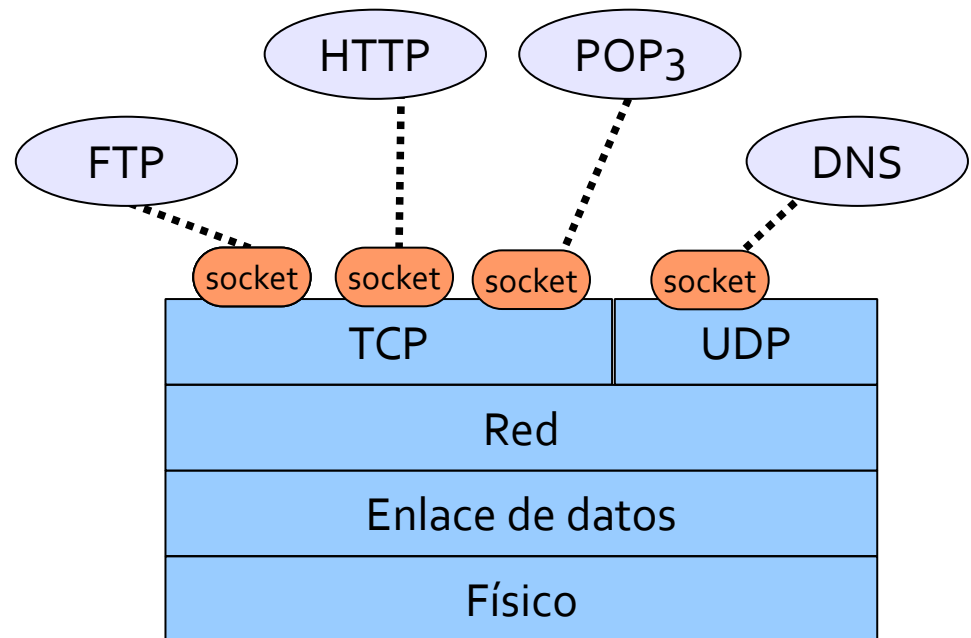
Bibliografía:

- ❑ [Kurose10] Apartados 2.1, 2.7, 2.8
- ❑ <http://www.oracle.com/technetwork/java/index.html>
- ❑ <http://java.com/es/about/>
- ❑ <http://download.oracle.com/javase/6/docs/api/index.html>

- Clientes y servidores utilizan protocolos de transporte
- Se necesita un mecanismo para ponerlos en contacto
 - API (*Application Programming Interface*)

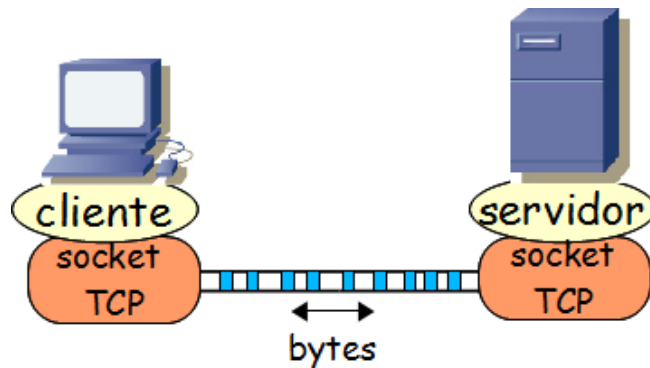


- Originalmente diseñado en BSD Unix
 - Permite a las aplicaciones utilizar los protocolos de la pila TCP/IP
- Define las operaciones permitidas y sus argumentos
 - Parecido a la forma de acceder a los ficheros en Unix
 - Operaciones: open, read, write, close
 - Cuando se abre un fichero obtenemos un descriptor
- Es un estándar de facto



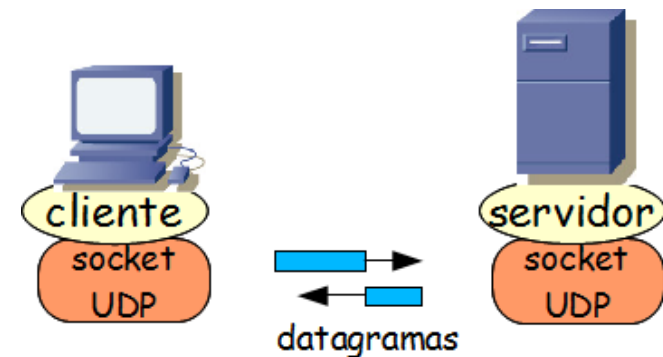
Sockets TCP

- Las aplicaciones piden al S.O. una comunicación controlada por TCP:
 - Orientada a la conexión
 - Comunicación fiable y ordenada
- También se denominan sockets de tipo **Stream**



Sockets UDP

- Las aplicaciones piden al S.O. una comunicación controlada por UDP:
 - Transferencia de bloques de datos
 - Sin conexión ni fiabilidad ni entrega ordenada
 - Permite difusiones
- También se denominan sockets de tipo **Datagram**



- Dentro del paquete `java.net` existen tres clases de sockets:
 - `Socket` Cliente TCP
 - `ServerSocket` Servidor TCP
 - `DatagramSocket` Cliente/Servidor UDP
- También hay otras clases auxiliares que facilitan la programación de aplicaciones en red
 - Ver referencias

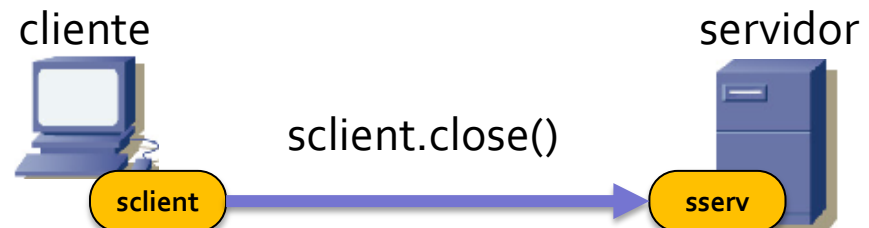
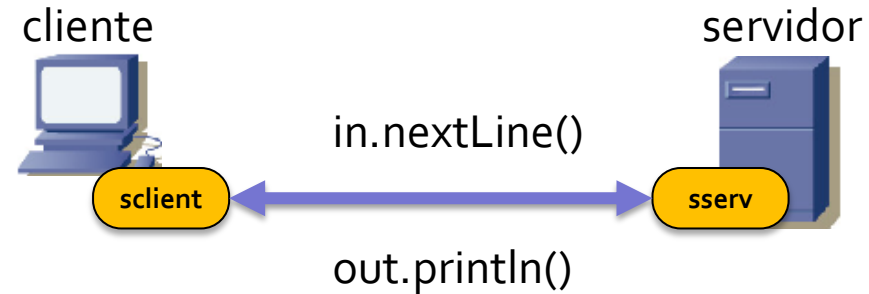


Servidor:

- Ha de estar en ejecución
- Debe haber creado un socket (**sserv**) donde recibir a los clientes que conectan con él

Cliente:

- Crea un socket (**sclient**) y lo conecta con el del servidor
- Transfiere información
- Cierra el socket y la conexión (a veces lo hace el servidor)



- Clase `Socket`
- Constructores más frecuentes
 - Crean un socket y lo conectan con el servidor y el puerto indicados
 - `Socket(String nombre, int puerto)`
 - `Socket(InetAddress dirIP, int puerto)`



- Clase **Socket**
- Algunos métodos importantes
 - **close()** : Cierra el socket
 - **InputStream getInputStream()**
 - Proporciona un descriptor para leer del socket
 - **InputStream** proporciona un flujo de bytes
 - Se puede leer un byte: **read()**
 - O un grupo de bytes: **read(byte[] b)**
 - **OutputStream getOutputStream()**
 - Proporciona un descriptor para escribir en el socket
 - **OutputStream** admite un flujo de bytes
 - Se puede escribir un byte: **write(int b)**
 - O un grupo de bytes: **write(byte[] b)**



- **Scanner** facilita la lectura de un flujo de bytes
 - Se puede leer la palabra siguiente (String): **next()**
 - el siguiente entero: **nextInt()**
 - el siguiente número en coma flotante: **nextFloat()**
 - o la siguiente línea: **nextLine()**
- Ejemplo:

```
import java.util.Scanner;  
...  
  
Scanner entrada=new Scanner(s.getInputStream());  
entrada.nextLine();
```



```
Scanner entrada = new Scanner(s.getInputStream());  
...  
while (entrada.hasNext()) {  
    System.out.println(entrada.nextLine());  
}
```



- A veces también es interesante leer de la entrada estándar (teclado)
- Ejemplo:

```
import java.util.Scanner;  
...  
  
Scanner leeTeclado=new Scanner(System.in);  
System.out.println("Nombre del servidor destino: ");  
String servidor = leeTeclado.nextLine();  
System.out.println("Introduce el puerto destino: ");  
int pto = leeTeclado.nextInt();
```



- PrintWriter permite enviar texto (caracteres)
 - Tiene métodos que permiten escribir texto:
`print(String s)`, `println(String s)`,
`printf(String s, Object ... args)`
- Ejemplo:

```
PrintWriter salida = new PrintWriter(s.getOutputStream());  
salida.printf("GET / HTTP/1.0" + "\r\n");  
salida.flush();
```

- Ejemplo:

```
PrintWriter salida = new PrintWriter(s.getOutputStream(), true);  
salida.printf("GET / HTTP/1.0" + "\r\n");  
// no hace falta salida.flush();
```





```
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class ClienteTCP0 {
    public static void main(String args[])
        throws UnknownHostException, IOException {
        Socket s=new Socket("zoltar.redes.upv.es", 7);
        System.out.println("Conectado");
        PrintWriter salida = new PrintWriter(s.getOutputStream());
        salida.printf("Hola Mundo!\r\n");
        salida.flush();
        Scanner entrada=new Scanner(s.getInputStream());
        System.out.println(entrada.nextLine());
        s.close();
        System.out.println("Desconectado");
    }
}
```

- ❑ Este cliente se conecta al servidor ECHO (puerto 7), envía y recibe una línea y después cierra la conexión



- Al intentar conectar un socket con un servidor hay dos problemas típicos, que generan excepciones en Java:
 - No se puede resolver el nombre del servidor
 - Se genera una excepción `UnknownHostException`
 - Se ha resuelto el nombre pero no se puede establecer la conexión
 - Se genera una excepción `IOException`
- Estas excepciones se pueden manejar mediante

```
try{...}  
catch(Exception e) {instrucciones al producirse la excepción}
```
- Hay que capturar primero la excepción más específica
 - (`UnknownHostException`)



➔ <http://download.oracle.com/javase/1.4.2/docs/api/java/io/IOException.html>

java.io

Class IOException

[java.lang.Object](#)

└ [java.lang.Throwable](#)

└ [java.lang.Exception](#)

└ [java.io.IOException](#)

All Implemented Interfaces:

[Serializable](#)

Direct Known Subclasses:

[ChangedCharSetException](#), [CharacterCodingException](#), [CharConversionException](#), [ClosedChannelException](#),
[EOFException](#), [FileLockInterruptedException](#), [FileNotFoundException](#), [IOException](#), [InterruptedIOException](#),
[MalformedURLException](#), [ObjectStreamException](#), [ProtocolException](#), [RemoteException](#), [SocketException](#), [SSLException](#),
[SyncFailedException](#), [UnknownHostException](#), [UnknownServiceException](#), [UnsupportedEncodingException](#),
[UTFDataFormatException](#), [ZipException](#)



```
public class ClienteTCP1 {  
    public static void main(String args[])  
        throws UnknownHostException, IOException {  
try{  
        Socket s=new Socket("noexiste.redes.upv.es", 7);  
        System.out.println("Conectado");  
        PrintWriter salida = new PrintWriter(s.getOutputStream());  
        salida.print("Hola Mundo!\r\n");  
        salida.flush();  
        Scanner entrada=new Scanner(s.getInputStream());  
        System.out.println(entrada.nextLine());  
        s.close();  
        System.out.println("Desconectado");  
    } catch (UnknownHostException e) {  
        System.out.println("Host desconocido");  
        System.out.println(e);  
    } catch (IOException e) {  
        System.out.println("No se puede conectar");  
        System.out.println(e);  
    }  
}  
}
```



- **InetAddress** es la clase que se utiliza para almacenar direcciones IP en Java
- Algunos métodos importantes
 - static InetAddress **getByName**(String nombre)
 - Obtiene la dirección IP asociada a un nombre
 - String **getHostAddress**()
 - Devuelve la dirección IP en formato "aa.bb.cc.dd"
 - Ejemplo:

```
InetAddress inet = InetAddress.getByName("www.mit.edu");
System.out.println("IP : " + inet.getHostAddress());
```
 - String **getHostName**()
 - Devuelve el nombre del host
 - Ejemplo:

```
InetAddress address = InetAddress.getLocalHost();
String sHostName = address.getHostName();
System.out.println(sHostName);
```



Obteniendo información de la conexión

```
public class EjemploInetAddress {
    public static void main(String args[])
        throws UnknownHostException, IOException {
    try{
        InetAddress zoltar = InetAddress.getByName("zoltar.redes.upv.es");
        Socket s=new Socket(zoltar, 7);
        System.out.println("Conectado");
        System.out.print("Host local:"); System.out.println(s.getLocalAddress().getHostName());
        System.out.print("IP local:"); System.out.println(s.getLocalAddress().getHostAddress());
        System.out.print("Puerto local:"); System.out.println (s.getLocalPort());
        System.out.print("Host remoto:"); System.out.println(s.getInetAddress().getHostName());
        System.out.print("IP remota:"); System.out.println(s.getInetAddress().getHostAddress());
        System.out.print("Puerto remoto:"); System.out.println(s.getPort());
        s.close();
        System.out.println("Desconectado");
    } catch (UnknownHostException e) {
        System.out.println("Host desconocido");
        System.out.println(e);
    } catch (IOException e) {
        System.out.println("No se puede conectar");
        System.out.println(e);
    }
}
```



- Argumentos de un programa desde la línea de órdenes:
 - `public static void main(String args[])`
 - `args[0]`, `args[1]`, ...
 - En ese caso, `args.length` debe ser mayor o igual a uno



Strings: obtener información

- Para obtener la longitud, número de caracteres que guarda un string se llama a la función miembro *length*.
 - `String str="El primer programa";`
 - `int longitud=str.length();`
- Podemos conocer si un string comienza con un determinado prefijo, llamando al método *startsWith*, que devuelve *true* o *false*, según que el string comience o no por dicho prefijo
 - `String str="El primer programa";`
 - `boolean resultado=str.startsWith("El");`
 - En este ejemplo la variable resultado tomará el valor *true*.
- De modo similar, podemos saber si un string finaliza con un conjunto dado de caracteres, mediante la función miembro *endsWith*.
 - `String str="El primer programa";`
 - `boolean resultado=str.endsWith("programa");`
- Si se quiere obtener la posición de la primera ocurrencia de la letra p, se usa la función *indexOf*.
 - `String str="El primer programa";`
 - `int pos=str.indexOf('p');`
- Para obtener las sucesivas posiciones de la letra p, se llama a otra versión de la misma función
 - `pos=str.indexOf('p', pos+1);`
 - El segundo argumento le dice a la función *indexOf* que empiece a buscar la primera ocurrencia de la letra p a partir de la posición pos+1.
- Otra versión de *indexOf* busca la primera ocurrencia de un substring dentro del string.
 - `String str="El primer programa";`
 - `int pos=str.indexOf("pro");`



■ Metodo equals():

```
- String str1="El lenguaje Java";  
- String str2=new String("El lenguaje Java");  
- if(str1==str2){  
-     System.out.println("Los mismos objetos");  
- }else{  
-     System.out.println("Distintos objetos");  
- }  
- if(str1.equals(str2)){  
-     System.out.println("El mismo contenido");  
- }else{  
-     System.out.println("Distinto contenido");  
- }
```



Compara objetos

Compara strings

■ Metodo compareTo()

- devuelve un entero menor que cero si el objeto string es menor (en orden alfabético) que el string dado, cero si son iguales, y mayor que cero si el objeto string es mayor que el string dado.

```
- String str="Tomás";  
- int resultado=str.compareTo("Alberto");
```

- La variable entera resultado tomará un valor mayor que cero, ya que Tomás está después de Alberto en orden alfabético.

