FIB Facultat d'Informàtica de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

CONCEPTES AVANÇATS DE SISTEMES OPERATIUS
Departament d'Arquitectura de Computadors

Sockets y otros objetos orientados a Internet en Delphi.

(Seminaris de CASO)

Autors
Patrick Mira Pedrol

Antes de nada. ¿Què és Delphi?

- Delphi es un entorno de Programación visual orientado a objetos para desarrollo rápido de aplicaciones (RAD) de propósito general, incluyendo aplicaciones cliente/servidor, desarrollo de bases de datos multinivel dimensionable, auténtica capacidad de reutilización orientada a objetos y compilador de código original de alto rendimiento.
- Delphi utiliza Object Pascal, un lenguaje de programación muy poderoso que está sin dudas a la altura del C++.
- Tiene un aspecto similar a Visual Basic, pero aunque el aspecto externo indica la misma facilidad de uso que Visual Basic, el corazón del sistema Delphi es mucho más potente y sobretodo mucho mas estable.
- Muchas fuentes lo colocan actualmente como el mejor entorno para desarrollo rápido de aplicaciones (RAD).

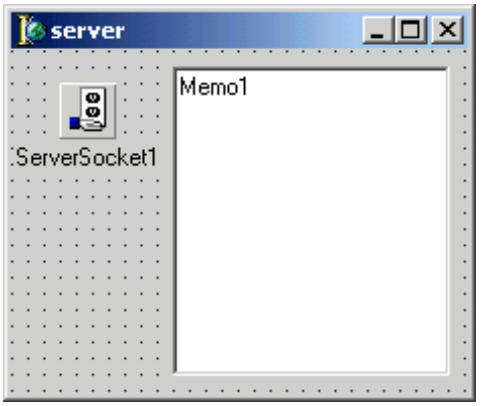
Objetos orientados a internet en Delphi

- Delphi incluye desde su primera versión varias clases para la creación de aplicaciones orientadas a Internet.
- Estas clases encapsulan muchos de los protocolos mas extendidos en Internet como FTP, SMTP - POP (Correo), HTTP (CGI's), NNTP (news), entre otros.
- Esto nos permite desarrollar aplicaciones rápidamente, usando estos protocolos sin preocuparnos en ningún momento de cual es su implementación interna.
- Además se incluyen dos clases de mas bajo nivel para Sockets, una para Cliente y otra para Servidor, que nos permitirán implementar cualquier servicio propio que necesitemos.

Sockets. TServerSocket y TClientSocket

- Estas dos clases encapsulan la mayoría de llamadas a sistema para Sockets vistas en la asignatura de CASO, haciendolas totalmente transparentes al programador.
- Esto nos permite desarrollar servicios propios y aplicaciones distribuidas de una manera rápida y sencilla sin preocuparnos de los Sockets que hay implementados por debajo.
- Sin embargo esta facilidad de uso no nos resta potencia en cuanto a los Sockets ya que se ha tenido en cuenta todos los posibles paradigmas que nos podamos encontrar. Por ejemplo la utilización de flujos (Threads) para cada cliente conectado al Servidor. Flujos que a la vez estan encapsulados en otra clase Tthread que no trataremos en aquí.

Ejemplo práctico. Servidor. (1)



Vamos ha hacer una pequeño servidor que nos muestre en el Objeto **Memo1** todo lo que reciba de los clientes que se conecten por el puerto 23.

Después veremos también como se haría la parte del cliente.

Cabe notar que es indiferente con que lenguaje o plataforma este implementado una y otra parte, es decir, que nuestro servidor puede usarse con cualquier cliente que use sockets, y viceversa.

Ejemplo práctico. Servidor. (2)

```
type
  TForm1 = class(TForm)
    ServerSocket1: TServerSocket;
    lbClients: TListBox;
    Label1: TLabel;
    Label2: TLabel;
    lbLog: TListBox;
    FileListBox1: TFileListBox;
    procedure ServerSocket1ClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocket1ClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocket1ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
  public
    procedure RefreshClients (var Msg: TMessage);
      message wm_RefreshClients;
  end;

var
  Form1: TForm1;
```

Ejemplo práctico. Servidor. (3)

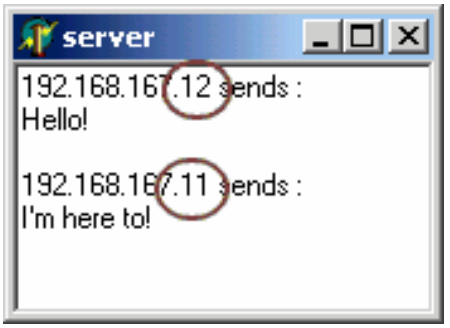
```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ServerSocket1.Port := 23;
  ServerSocket1.Active := True;
end;
```

```
procedure TForm1.FormClose
(Sender: TObject; var Action: TCloseAction);
begin
  ServerSocket1.Active := false;
end;
```


Ejemplo práctico. Servidor. (4)

```
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
  i:integer;
  sRec : string;
begin
  for i := 0 to ServerSocket1.Socket.ActiveConnections-1 do
  begin
    with ServerSocket1.Socket.Connections[i] do
    begin
      sRec := ReceiveText;
      if sRecr <> '' then
      begin
        Memo1.Lines.Add(RemoteAddress + ' sends :') ;
        Memo1.Lines.Add(sRecr);
      end;
    end;
  end;
end;
```

Ejemplo práctico. Servidor. (5)



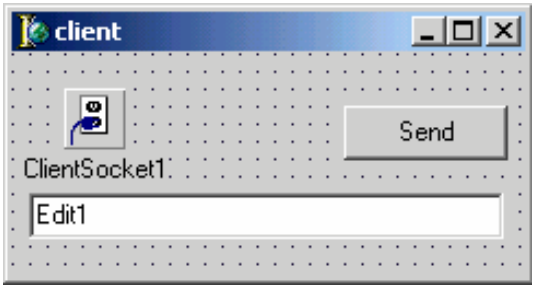
Y este sería el resultado final.

A partir de aquí es fácil empezar a imaginar aplicaciones distribuidas de todo tipo.

Cabe notar también que los ejecutables creados en Delphi son pequeños y totalmente independientes, es decir, que no precisan de ninguna DLL externa para su funcionamiento. Simplemente el ejecutable ha de funcionar en cualquier plataforma Windows.

Ejemplo práctico. Cliente. (1)

Ahora veamos rápidamente como haríamos la parte del Cliente en Delphi usando un objeto de la clase **TClientSocket**. Todo lo que ha de hacer es conectarse al servidor y enviar, cuando pulsemos el botón Send, lo que hayamos escrito en el objeto **Edit1**.



Ejemplo práctico. Cliente. (2)

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ClientSocket1.Port :=  23;
  //local TCP/IP address of the server
  ClientSocket1.Host :=  '192.168.167.12';
  ClientSocket1.Active :=  true;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction
begin
  ClientSocket1.Active := false;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if ClientSocket1.Active then
    ClientSocket1.Socket.SendText(Edit1.Text);
end;
```


Mas información sobre Delphi. Links.

- Tutoriales:
  - leo.worldonline.es/acanudas/delphi/tdelphi5.htm
  - www.hackerdude.com/courses/spanish/delphi/indice.html
- Borland Delphi 7 Studio .NET:
  - www.masternet.com.co/noti/noti35.htm
  - www.borland.com
- Otros links interesantes:
  - www6.uniovi.es/pub/delphi/index.htm
  - www.clubdelphi.com