EXERCISE SHEET 2

Search Engine

(Triatmoko 2900127)

EXERCISE 1

```
unit Webserver;


interface

uses

  SysUtils, Classes, HTTPApp;


type

  TWebModule2 = class(TWebModule)

    procedure WebModule2QueryAction(Sender: TObject; Request:
TWebRequest;

      Response: TWebResponse; var Handled: Boolean);

  private

    { Private declarations }

  public

    { Public declarations }

  end;


var

  WebModule2: TWebModule2;


implementation
```

```
uses WebReq;

{$R *.dfm}

procedure TWebModule2.WebModule2QueryAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);



var
  RequestQuery,str,strcopy:    string;
  i,x:integer;
begin
   //Validation query from client, Ex:query?repeat=87 will be 87
   RequestQuery:=request.Query;
   str:=trim(requestquery);
   strcopy:='';
   for i:=0 to length(str) do
      begin
        if str[i]='=' then
           begin
             x:=(length(str)-i) +1;
             strcopy:=copy(str,i+1,x);
           end
      end;
// if request is empty
// response with an error message as text/plain.
```

```
    if (strcopy='') or (str='') then
       begin
          response.contenttype := 'text/plain';
          response.content:='Error: on Request, check your request';
          response.sendresponse;
       end
    else
       begin
          //server will responds query from client, and send back the
            answer
          Response.ContentType:='TEXT/HTML';
          Response.SendRedirect('index.php?Loop=$'+strcopy+'');
          Response.SendResponse;
       end;
handled:=true; //its mean that query had been handled by server
end;


initialization
  if WebRequestHandler <> nil then
    WebRequestHandler.WebModuleClass := TWebModule2;


end.
```

EXERCISE 2

```
unit Webserver;


interface


uses

  SysUtils, Classes, HTTPApp;


type

  TWebModule2 = class(TWebModule)

    procedure WebModule2QueryAction(Sender: TObject; Request:
TWebRequest;

      Response: TWebResponse; var Handled: Boolean);

  private

    { Private declarations }

  public

    { Public declarations }

  end;


var

  WebModule2: TWebModule2;


implementation


uses WebReq;


{$R *.dfm}
```

```
Procedure TWebModule2.WebModule2QueryAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  RequestQuery,str,strcopy:    string;
  i,x:integer;
begin
//split request from client to right query
RequestQuery:=request.Query;
str:=trim(requestquery);
   for i:=0 to length(str) do
   begin
     if str[i]='=' then
        begin
          x:=(length(str)-i) +1;
          strcopy:=copy(str,i+1,x);
        end
   end;
// if request is empty
// response with an error message as text/plain.
   if (RequestQuery='')or (strcopy='') then
      begin
        response.contenttype := 'text/plain';
        response.content:='Error: on Request, check your request';
        response.sendresponse;
      end
   else
```

```
    begin
    //answer to client as HTML text
      Response.ContentType:='TEXT/HTML';
      Response.Content:='<HTML> <Body>
      <b>welcome'+strcopy+'<b></br></Body> </HTML> ';
      Response.SendResponse;
    end;
handled:=true;
end;


end;
end;

initialization
  if WebRequestHandler <> nil then
    WebRequestHandler.WebModuleClass := TWebModule2;


end.
```

EXERCISE 3

Simple HTML page

```
<html >
<head>
<title>Search Triatmoko</title>
</head>
<body>
<table width="351" height="165" border="1" align="center">
  <tr>
    <td width="310" height="45"> </td>
  </tr>
  <tr>
 <td>

    <form id="form1" name="form1" method="post"
action="http://localhost:8080/SearchEngine.SearchEngineServer/query? "
target="_self">
      <label>
        <input type="text" name="Query" id="Query" />
      </label>
      <label>
        <input type="submit" name="button" id="button" value="Submit"
/>
      </label>
    </form></td>
  </tr>
  <tr> <td>RESULT <br />
```

```
//after server send variable loop, page will show a sentence loop
times
<?php
$i=1;
while($i<=$loop)
{
echo " will repeat" . $loop. "<br />";
$i++;
}
?> </td></tr>
</table>


</body>
</html>
```

EXERCISE 4

Communication with Server using TCP

```
unit tcplient;


interface


uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, IdBaseComponent, IdComponent, IdTCPConnection,
  IdTCPClient, Buttons;


type
  TForm1 = class(TForm)
    IdTCPClient1: TIdTCPClient;
    edthost: TEdit;
    edtport: TEdit;
    edtmessage: TEdit;
    memRcvMessage: TMemo;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure edtmessageKeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

  TRecvThread = class(TThread)
  private
    Msg : string;
    procedure ReceivedLine;
  protected
    procedure Execute; override;
  end;

var
  Form1: TForm1;
  RecvThread : TRecvThread;
implementation

{$R *.DFM}

{TRecvThread }
procedure TRecvThread.Execute;
begin
  while not Terminated do begin
    try
      //  Read message from server
      Msg := Form1.IdTCPClient1.ReadLn;
      Synchronize(ReceivedLine);
    except
      Terminate;
    end;
  end;
```

```
end;


procedure TRecvThread.ReceivedLine;
begin
// Cek message from server, if empty, error message will be showing
  if Copy(Msg,1,4) = '-Err' then
    MessageDlg(Copy(Msg,5,Length(Msg)), mtError, [mbOk],0)
  Else
     // write to screen
     form1.memRcvMessage.Lines.Add(Msg);
end;


procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  { Assign values to the Host and Port Properties }
  with IdTCPClient1 do
  begin
    Host := '132.230.152.135';
    Port := StrToInt('9999');
    Connect;
  end;
  { Create the Thread }
  RecvThread := TRecvThread.Create(False);
  BitBtn1.Enabled := False;
  BitBtn2.Enabled := True;
  if  IdTCPClient1.Connected then
end;
```

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  { Client to terminate and wait until its  }
  RecvThread.Terminate;
  repeat
    Application.ProcessMessages;
  until RecvThread.Terminated;

  IdTCPClient1.Disconnect;
  BitBtn1.Enabled := True;
  BitBtn2.Enabled := False;
end;

procedure TForm1.edtmessageKeyPress(Sender: TObject; var Key: Char);
var
data: byte;
begin
  //Write to Server
  if Key = #13 then
  begin
    IdTCPClient1.WriteLn(edtMessage.Text);
    edtMessage.Text := '';

  end;
end;
end.
```

Communication with Server using UDP

```
unit udp;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, IdBaseComponent, IdComponent, IdTCPConnection,
  IdTCPClient, Buttons, IdUDPBase, IdUDPClient;

type
  TForm1 = class(TForm)
    IdTCPClient1: TIdTCPClient;
    edthost: TEdit;
    edtport: TEdit;
    edtmessage: TEdit;
    memRcvMessage: TMemo;

    IdUDPClient1: TIdUDPClient;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;

    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
  private
```

```
    { Private declarations }
  public
    { Public declarations }
  end;

  TRecvThread = class(TThread)
  private
    Msg : string;
    procedure ReceivedLine;
  protected
    procedure Execute; override;
  end;

var
  Form1: TForm1;
  RecvThread : TRecvThread;
implementation

{$R *.DFM}

{TRecvThread }
procedure TRecvThread.Execute;
var
msg:array[1..10000] of byte;
begin
  while not Terminated do begin
    try
//  Read message from server
      Msg := Form1.IdUDPClient1.ReceiveBuffer(0,10000);
```

```
      Synchronize(ReceivedLine);
    except
      Terminate;
    end;
  end;
end;

procedure TRecvThread.ReceivedLine;
begin
  if Copy(Msg,1,4) = '-Err' then
    MessageDlg(Copy(Msg,5,Length(Msg)), mtError, [mbOk],0)
  Else
     //write on the screen
     form1.memRcvMessage.Lines.Insert(0,msg);
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
  { Assign values to the Host and Port Properties }
  with IdUDPClient1 do
  begin
    Host := 132.230.152.135';
    Port := StrToInt('8888');
    Active:=true;
  end;
  { Create the Thread }
  RecvThread := TRecvThread.Create(False);
  Button1.Enabled := False;
  Button2.Enabled := True;
end;
```

```
procedure TForm1.Button2Click(Sender: TObject);
begin
  { Get the Thread to terminate and wait till it is terminated }
  RecvThread.Terminate;
  repeat
    Application.ProcessMessages;
  until RecvThread.Terminated;
//disconnecting from server
  IdUDPClient1.active:=false;
  Button1.Enabled := True;
  Button2.Enabled := False;
end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
Var
buff:array[1..10000] of byte;
begin
  { If the ENTER KEY is pressed, send the Text in Edit1 to the Server
}
  if Key = #13 then
  begin
//write message to server
    buff:=Edit1.SetSelTextBuf;
    IdUDPClient1.SendBuffer(Buff,sizeof(buf));
    Edit1.SetSelTextBuf.Text := '';
  end;
end;
end.
```

Communication to server by HTTP

```
unit HTTP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, UrlMon;

type
  TForm1 = class(TForm)
    WebDispatcher1: TWebDispatcher;
    Connect: TButton;
    procedure ConnectClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}
```

```
function Getfile(Source, Dest: string): Boolean;

begin

  try

   //Get valid URL

    Result :=  UrlDownloadToFile(nil, PChar(source), PChar(Dest), 0,
nil) = 0;

  except

    Result := False;

  end;

end;


procedure TForm1.ConnectClick(Sender: TObject);

Begin

 // Check file, and downloading file

  if Getfile  ('http://vulcano.informatik.uni-freiburg.de/file_100M,
'c:\') then

   ShowMessage('GetFile successful')

  else

   ShowMessage('GetFile unsuccessful')

end;
```