**Работа с HTTP протоколом в Delphi**

В связи с все большим вниманием, которое привлекает к себе Интернет, все больше людей становятся заинтересованы в сетевых технологиях. Данная статья посвящена программированию на Borland Delphi с использованием одного из самых популярных Интернет-протоколов - HTTP.

А именно, здесь мы рассмотрим компонент TNMHTTP (NetMasters HTTP), который можно обнаружить на вкладке FastNet палитры компонентов Дельфи.

Начнем с теории. Если Вы уже знаете, что такое HTTP и зачем он нужен, то пропустите следующий раздел.

**Зачем нужен HTTP**

Итак, где же используется HTTP? Если Вы хотя бы чуть-чуть заглядывали на Интернет-странички и встречались с термином Web, то наверняка обратили внимание на то, что адреса страничек, как правило, начинаются с http://. Протокол HTTP (HyperText Transfer Protocol) позволяет принимать и посылать не только гипертекстовые документы (типа html), но и любые другие (тексты (txt), изображения (gif, jpg), и т.д.). Ниже приведены типовые задачи, для выполнения которых необходимо использовать HTTP:

Браузеры - программы, позволяющие просматривать Интернет-странички;

Скачивальщики - программы, позволяющие скачивать из Интернета странички, рисунки и другие документы;

Чаты - программы, позволяющие общаться по сети. Часто документы HTTP используются для хранения сообщений (как, например, в конференциях).

- Это лишь список некоторых из стандартных направлений программирования с использованием HTTP. Вы можете применять этот протокол для любых своих целей. Например, автоматические системы обновления данных, посылка запросов в Интернетовские базы, и еще множество всяческих других возможностей!

**Краткое описание свойств, методов и событий**

Ниже приведена таблица, содержащая наиболее краткое описание основных свойств, методов и событий компонента TNMHTTP:

Свойства

Body - строка, содержащая либо путь к файлу, в который будет записано тело http-документа (если св-во InputFileMode равно True), либо непосредственно само тело (если св-во InputFileMode равно False). Тип: string;

Header - строка, содержащая либо путь к файлу, в который будет записан заголовок http-документа (если св-во InputFileMode равно True), либо непосредственно сам заголовок (если св-во InputFileMode равно False). Тип: string;

HeaderInfo - структура, содержащая различную информацию о http-документе (подробней см. в help-файле). Тип: THeaderInfo;

InputFileMode - тип записи результата. Значение True - запись в файлы, указанные в свойствах Body и Header, False - запись в сами эти свойства. Тип: Boolean;

OutputFileMode - тип отсылаемых данных (методами Put, Post и Trace). Значение True - данные для отправки содержатся в файлах, указанных при вызове этих методов, а False - в самих аргументах этих методов. Тип: Boolean;

Далее некоторые свойства, унаследованные от TPowerSock:

BytesRecvd, BytesSent, BytesTotal - количество отправленных, принятых и общее количество байтов соотвественно. Тип: LongInt;

Connected - показывает, установленно ли в данный момент соединение. Тип: Boolean;

BeenCanceled - показывает, было ли прервано соединение с сервером. Тип: Boolean;

Host - строка, содержащая хост-имя удаленного компьютера. Заполнять не надо, так как это свойство устанавливается автоматически при вызове методов Get, Put, Post и т.д. Тип: string. Port - Integer, содержащий порт удаленного компьютера (заполняется тоже автоматически);

TimeOut - таймаут в миллисекундах. Тип: Integer;

Еще есть множество свойств, но я пока остановлюсь на уже перечисленных. За дополнительной информацией обращайтесь к help-у по Дельфи.

Методы

Get(URL: string) - посылает запрос на указанный URL. Данные после выполнения этого запроса записываются в файлы или в сами свойства Body и Header (в зависимости от значения свойства InputFileMode);

Head(URL: string) - посылает запрос на указанный URL. Данные после выполнения этого запроса записываются в файл или в само свойство Header (в зависимости от значения свойства InputFileMode). В отличие от метода Get, при вызове Head запрос отсылается только на заголовок http-документа;

Post(URL, PostData: string) - посылает запрос на изменение http-документа (с адресом URL) на данные, содержащиеся в параметре PostData. Если OutputFileMode равен True, то в PostData должен содержаться путь к файлу, содержащему нужные данные.

Put(URL, PutData: string) - посылает запрос на создание http-документа (с адресом URL), содержащего данные, переданные в параметре PutData. Если OutputFileMode равен True, то в PostData должен содержаться путь к файлу, содержащему нужные данные.

Trace(URL, TraceData: string) - посылает запрос на получение отладочных данных (для отладки соединения с HTTP-сервером). Данные для запроса нужно указать в параметре TraceData. Если OutputFileMode равен True, то в TraceData должен содержаться путь к файлу, содержащему нужные данные.

Delete(URL: string) - посылает запрос на удаление http-документа (с адресом URL).

Далее некоторые методы, унаследованные от TPowerSock:

Abort и Cancel - прерывают соединение и обмен данными;

Disconnect - отсоединение от HTTP-сервера;

**События**

OnAuthenticationNeeded - возникает, когда сервер требует указания имени пользователя и пароля. В обработчике этого события (если оно возникнет) Вы должны ответить серверу, запонив нужными значениями соответствующие переменные. Примечание: Перед установлением соединения можно сразу заполнить поля UserID и Password в свойстве HeaderInfo;

OnAboutToSend - возникает, когда компонент TNMHTTP собирается отправлять данные (запрос). В обработчике этого события можно заполнить дополнительной информацией свойство SendHeader;

OnFailure - возникает, когда текущая операция завершилась неудачно, т.е. произошла ошибка;

OnRedirect - возникает, сервер переадресовал ссылку с указанной URL на другую ссылку. Установив параметр handled в значение True можно запретить переадресацию и остановиться на запрошенной URL. Значение по умолчанию - False;

OnSuccess - возникает, когда текущая операция завершилась успешно, т.е. запрос был выполнен без ошибок;

Далее некоторые методы, унаследованные от TPowerSock:

OnConnect - возникает, когда соединение с сервером успешно установлено;

OnDisconnect - возникает, когда соединение с сервером завершено;

OnConnectionFailed - возникает, когда соединение с сервером установить не удалось;

OnError - возникает, когда последняя операция была завершена с ошибкой;

OnHostResolved - возникает, когда от DNS получен IP-адрес указанного хоста;

OnInvalidHost - возникает, когда DNS вернул ошибку при попытке определить IP-адрес указанного хоста;

OnPacketRecvd - возникает, когда значения свойств BytesRecvd и BytesTotal изменены, т.е. была принята новая порция данных от сервера;

OnPacketSent - возникает, когда значения свойств BytesSent и BytesTotal изменены, т.е. была отправлена новая порция данных на сервер;

OnStatus - возникает, когда статус компонента был изменен (для обновления визуального оповещения пользователя);

**Практика и примеры**

Ну а теперь приступим к самому главному методу изучения - на примерах.

И самый первый пример - программа, позволяющая определить, существует ли заданный URL:

Пример 1. Проверка существования указанной URL

{... Здесь идет заголовок файла и определение формы TForm1 и ее экземпляра Form1}

{В форму нужно поместить кнопку TButton и одно поле TEdit. При нажатии на

кнопку вызывается обработчик события OnClick - Button1Click. Перед этим в

TEdit нужно ввести адрес URL. НЕ ЗАБУДЬТЕ ПОМЕСТИТЬ В ФОРМУ КОМПОНЕНТ TNMHTTP!}

procedure Button1Click(Sender: TObject);

begin

{Пытаемя получить заголовок}

NMHTTP1.Head(Edit1.Text);

{Если URL неверный, то здесь выскочит ошибка}

end;

Следующий пример - скачивание сразу нескольких URL одновременно. Надо заметить, что многие программисты пренебрегают многозадачностью Windows (неважно, как она реализована, речь сейчас не об этом). В Дельфи очень легко создавать отдельные, подчиненные Вашей программе процессы (а точнее - потоки) с помощью базового класса TThread. Но об этом мы поговорим в другой раз (в другой статье).

Пример 3. Одновременное скачивание указанных URL в заданный каталог

// Здесь идет заголовок файла и определение формы TForm1 и ее экземпляра Form1

// Описание класса отдельного процесса

type

THTTPThread = class(TThread)

private

{Для каждого процесса - создаем свой компонент TNMHTTP}

FHTTP: TNMHTTP;

protected

// Execute вызывается при запуске процесса; override - заменяем

// существующую процедуру базового класса TThread

procedure Execute; override;

// DoWork - созданная нами функция, выполнение которой синхронизируется в Execute

procedure DoWork;

public

// URL - созданная нами строка, указывающая процессу, какой URL ему нужно скачать

URL: string;

end;

// В форму нужно поместить три кнопки TButton, одно поле TEdit и один список

// TListBox. При нажатии на кнопку Button1 вызывается обработчик события

// OnClick - Button1Click. Перед этим в TEdit нужно ввести путь к каталогу, в

// котором будут храниться скачанные файлы, а ListBox1 нужно заполнить списком

// URL-ов для скачивания (с помощью кнопок Add (Button2) и Delete (Button3)).

procedure TForm1.Button3Click(Sender: TObject);

begin

{Удаление выделенного URL из списка}

if ListBox1.ItemIndex >= 0 then

ListBox1.Items.Delete(ListBox1.ItemIndex);

end;

procedure TForm1.Button2Click(Sender: TObject);

var s: string;

begin

{Добавление URL в список}

s := InputBox('Добавить','Введите URL:','');

if s <> '' then

ListBox1.Items.Add(s);

end;

procedure TForm1.Button1Click(Sender: TObject);

var i: Integer;

begin

{Проверка на существование каталога}

if Length(Edit1.Text) > 0 then

if not DirectoryExists(Edit1.Text) then

MkDir(Edit1.Text);

{Далее идет создание для каждого URL в списке своего процесса}

for i := 0 to ListBox1.Items.Count-1 do begin

with THTTPThread.Create(True) do begin

{Создаем приостановленную задачу, указываем ей ее URL и запускаем ее}

URL := ListBox1.Items[i];

Resume;

end;

end;

end;

// Операторы процесса THTTPThread

procedure THTTPThread.Execute;

begin

// Делаем так, чтобы каждый процесс выполнялся одновременно с другими (синхронизация)}

Synchronize(DoWork);

end;

procedure THTTPThread.DoWork;

var i: Integer;

begin

{Создаем компонент TNMHTTP}

FHTTP := TNMHTTP.Create(Form1);

{Результат надо записывать в файлы}

FHTTP.InputFileMode := True;

{Подбираем имена для файлов}

i := 1;

while FileExists(Form1.Edit1.Text+'\page'+IntToStr(i)+'.htm') do

Inc(i);

{Указываем, в какие именно файлы класть результат}

FHTTP.Body := Form1.Edit1.Text+'\body'+IntToStr(i)+'.htm';

FHTTP.Header := Form1.Edit1.Text+'\header'+IntToStr(i)+'.txt';

{Пытаемся послать запрос}

FHTTP.Get(URL);

{Перед завершением процесса не забываем освободить память из-под компонента}

FHTTP.Free;

end;

ПРИМЕЧАНИЕ: Чтобы завершить некоторый процесс (Thread), нужно вызвать метод Terminate класса этого процесса. Приостановить процесс можно оператором Suspend, а продолжить выполнение - Resume. Также можно настроить приоритет каждого отдельного процесса через свойство Priority.

Неплохой пример работы с процессами можно найти в подпапке Demos\Threads папки, куда Вы установили Delphi.

Замечания по алгоритмам типовых задач

Если Вы собираетесь создать скачивалку сайтов, то Вам необходимо учитывать следующее (решить следующие проблемы):

Нужно скачивать не только саму страничку в формате HTML, но и все входящие в нее рисунки (gif, jpg, и т.д.);

в некоторых случаях удобно скачивать не одну страничку, а несколько страниц, ссылки на которые находятся на первой из скачиваемых страничек. При этом нужно учитывать, что на страничке могут находиться и ссылки на другие сайты, поэтому необходимо анализировать скачиваемые ссылки (чтобы случайно не скачать весь Интернет). Для решения задачи со скачиванием нескольких страничек нужно использовать рекурсию;

необходимо качественно информировать пользователя о ходе закачки. Т.е. показывать общее и скачанное количество информации;

после скачивания нужно заменить Интернетовские ссылки на локальные, чтобы можно было просматривать странички в режиме offline.