Norbert Kolb

GetOrPost - das ist hier die Frage!

Immer wieder wurde in letzter Zeit die Frage gestellt, wie man HTML-Seiten auch direkt aus einer VO-Applikation heraus abfragen kann? Dazu gibt es ja die cHttp-Klasse aus der Internet-Library. Doch wer ein wenig mehr will, steht schnell an.

Einführung

Eine bestimmte Seite, von der die URL bekannt ist, soll aufgerufen und als Datei gespeichert werden. Wir rufen dazu die Klasse cHttp auf. Die Parameter sind: cCaption, n und lStat. Dabei gibt cCaption den User-Agent an. Dies kann ein beliebiger Name sein oder zB. für den IE: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1). So kann der Web-Applikation auch jeder beliebige Browser vorgetäuscht werden. n steht für den Port und ist mit 80 als default vorgegeben. lStat legt den Status fest mit dem Vorgabewert TRUE . Die Methode GetDocumentByURL() ruft nun die angegebene Seite auf und hat als Rückgabewert den HTML-Code der gewünschten Seite.

```
LOCAL oHttp AS cHttp
LOCAL cPage AS STRING

oHttp := cHttp{ "TestGET"}
cPage :=
oHttp:GetDocumentByURL("http://www.myweb.net/
test.htm")
MemoWrit( "Seite1.htm", cPage)
oHttp:CloseRemote()
oHttp:Axit()
```

Schauen wir uns einmal die Befehlszeile an, wenn wir mit Google nach "CAVO" suchen.

http://www.google.at/search?hl=de&q=CAVO&btnG=Google-Suche&meta=

Wir erkennen, dass an die eigentlichen URL zusätzliche Informationen angefügt werden. Diese werden zunächst mit einem ? von der URL getrennt. Dabei handelt es sich um Variablen und deren Inhalt. Die einzelnen Teile werden durch ein kaufmännisches und (&) getrennt. Ein so abgesetzter Befehl wird mit der Methode GET an den Web-Server übermittelt. Die Web-Applikation wiederum kann nun überprüfen ob und welche Variablen und Inhalte übergeben wurden und entsprechend darauf reagieren. Auch ein Aufruf einer Seite mit der Angabe der Variablen erfolgt mit der oben beschriebenen Methode.

Häufig aber genügt ein solcher Aufruf nicht. Stellen wir uns vor, wir sollten uns mit Benutzername und Passwort bei einer Seite anmelden, eine Bestellung aufgeben und uns wieder abmelden. Wenn wir diesen Ablauf mit dem Browser durchführen, so erhalten wir ein Formular mit der Aufforderung Benutzername und Passwort einzugeben. Drücken wir anschließend auf den Button "Anmelden" sehen wir in der Befehlszeile keine mitgesendeten Variablen wie zB. bei Google. Dennoch erhalten wir, korrekte Eingabedaten vorausgesetzt, Zugang zu unserer Seite. Die Informationen müssen also im Verborgenen gesendet werden. Sehen wir uns einmal den Code einer solchen Eingabemaske an.

```
<form method="POST" action="anmeldung.htm">
  <p>User: <input type="text" name="user"
size="20"></p>
  <p>Password: <input type="text" name="pass"
size="20"></p>
  <p><input type="submit" value="send"
name="okbutton"></p>
</form>
```

In der ersten Zeile erkennen wir, dass die Daten aus einer Form im Allgemeinen mit der Methode POST abgeschickt werden. Die Variablen werden hier im Header angefügt. Diese Methode wird von der oben beschriebenen Methode GetDocumentByURL() der cHttp-Klasse nicht unterstützt.

Die Erweiterung

Nachdem ich selbst beruflich häufig über das Internet oben beschriebenes Szenario durchführen muß, habe ich mich nach einer programmunterstützten Variante gesehnt. Nach langem Suchen und recherchieren im Internet habe ich dann die cHttp-Klasse um folgende Methode erweitert.

```
METHOD GetDocumentByGetOrPost(cIP, cDocument,
cData, cHeader, cMethod, nP, nFlags) CLASS
cHttp
  ************************************
  // GET or POST - that's the question
  // Norbert Kolb, 11/17/03, 12/02/04,
09/02/04 02/25/05
  // UG Bodensee (A, CH, D, FL)
  ************************************
  // First the InternetOpen function is cal-
led to initialize the
  // remaining functions. Then the
InternetConnect function is called
  // to identify the type of service being
requested. This is necessary
  // as the same set of functions can be used
to interact with a number
  // of different servers.
  ************************************
```

```
  LOCAL lResult  AS LOGIC
  LOCAL nDataLen AS DWORD
  LOCAL nBuffer AS DWORD
  LOCAL nBufferLen AS DWORD
  LOCAL cRet  AS STRING
  Default(@cDocument,"")
  Default(@cData, "")
  Default(@cMethod, "GET")
  Default(@nP,SELF:nPort)
  Default(@nFlags, 0)
  IF _and(DWORD(nFlags), INTERNET_FLAG_SECU-
RE) = INTERNET_FLAG_SECURE
     nP := INTERNET_DEFAULT_HTTPS_PORT
     SELF:nPort := nP
  ENDIF
  IF SELF:hSession == NULL_PTR
    lResult := SELF:Open(NIL, SELF:cProxy)
  ELSE
     lResult := .T.
  ENDIF
  IF lResult
     // Identify the type of service that is
being accessed
     IF SELF:hConnect == NULL_PTR .or.
SELF:cHostAddress <> cIP
       SELF:cHostAddress := cIP
       SELF:hConnect :=
InternetConnect(SELF:hSession, ;
       String2Psz(SELF:cHostAddress), ;
       nP, ;
       String2Psz(SELF:cUserName), ;
       String2Psz(SELF:cPassword), ;
       INTERNET_SERVICE_HTTP, ;
       nFlags, ;
       0)
     ENDIF
     IF SELF:hConnect <> NULL_PTR
       SELF:__SetStatusObject()
       //Once we've identified the service,
we need to indicate the
       //page to which the data will be
posted. In this example, it will
       //be "/getorpost.php"
       SELF:hRequest :=
HttpOpenRequest(SELF:hConnect, ;    //
hConnect
         String2Psz(cMethod), ;    //
lpszVerb
         String2Psz(cDocument), ;     //
pszObjectName
         String2Psz("HTTP/1.1"), ;    //
pszVersion
         NULL_PSZ, ;                   //
pszReferer
         NULL_PSZ, ;                   //
pszAcceptTypes
         nFlags, ;              // dwFlags
         SELF:__GetStatusContext())  //
dwContext
       IF SELF:hRequest <> NULL_PTR
         SELF:__SetStatus(SELF:hRequest)
         //We need to add a header to notify
the web server that the
         //incoming data is form encoded and
then send the request.
         IF cMethod == "POST"
           IF Empty(cHeader)
             cHeader := "Content-Type:
application/x-www-form-urlencoded" + CrLf +
HEADER_ACCEPT
           ENDIF
         ELSE
           IF Empty(cHeader)
             cHeader := HEADER_ACCEPT
           ENDIF
         ENDIF
         IF
HttpAddRequestHeaders(SELF:hRequest, ;
             String2Psz(cHeader), ;
             SLen(cHeader), ;
             HTTP_ADDREQ_FLAG_ADD)
/*
           // Sometimes you have to set
SECURITY_FLAGS
           InternetQueryOption(SELF:hRequest,
INTERNET_OPTION_SECURITY_FLAGS, @nBuffer,
@nBufferLen)
           nBuffer := _or(nBuffer,
SECURITY_FLAG_IGNORE_UNKNOWN_CA)
           InternetSetOption(SELF:hRequest,
INTERNET_OPTION_SECURITY_FLAGS, @nBuffer,
nBufferLen)
*/
           nDataLen := SLen(cData)
           IF HttpSendRequest(SELF:hRequest,
;
                 NULL, ;
                 0, ;
                 PTR(_CAST, cData), ;
                 nDataLen)
             //Finally, we need to collect
our reward for all this effort
             SELF:GetResponseHeader()
             cRet := SELF:GetResponse()
           ENDIF  // SendRequest
           SELF:CloseRequest()
         END  // AddRequestHeaders
       ENDIF
       SELF:__DelStatusObject()
     ENDIF
  ENDIF
  RETURN cRet
```


Damit ist es nun möglich, flexibel auf die verschiedensten Aufgaben einzugehen. Damit wir uns bei der Seite mit Benutzername und Passwort anmelden können sollten wir folgendermaßen vorgehen:

```
LOCAL oHttp AS cHttp
LOCAL cPage AS STRING
LOCAL cServer AS STRING
LOCAL cFolder AS STRING
LOCAL cData AS STRING
LOCAL cHeader AS STRING

cServer := "www.myweb.net"
cFolder := "test.php"
cData :=
"user=MyName&pass=secret&okbutton=send"
cHeader := "Accept: *" + "/" + "*" + CrLf +
;
  "Content-Type: application/x-www-form-
urlencoded" + CrLf +;
  "Accept-Encoding: gzip, deflate" + CrLf +
CrLf

oHttp := cHttp{"TestPOST"}
cPage :=
oHttp:GetDocumentByGetOrPost(cServer,
cFolder, cData, cHeader, "POST",;
 /*INTERNET_DEFAULT_HTTPS_PORT*/, /*INTER-
NET_FLAG_SECURE*/)
MemoWrit( "Seite2.htm", cPage)
oHttp:CloseRemote()
oHttp:Axit()
```

Dabei ist zu beachten, dass die URL hier ohne das vorangestellte http:// angegeben werden muß und nur den Namen der Domaine ent-

hält. Der Pfad, der die Seite angibt, wird mit cFolder übergeben. In der Variablen cData sind die Variablen nach dem gleichen Verfahren wie bei GET anzugeben. Zu beachten ist weiters, dass der Submit-Button selbst einen Namen und einen Wert besitzen kann. In unserem Fall okbutton=send. Dies wird oft übersehen.

Zum Header ist zu bemerken, dass dieser beim Senden von Daten aus einer Form unbedingt den "Content-Type: application/x-www-form-urlencoded" enthalten muß. Die einzelnen Informationen des Headers sind durch ein CrLf getrennt. Abgeschlossen wird der Header durch zwei CrLf.

Manchmal sind es aber https-Seiten deren Zugang wir unserer VO-Applikation ermöglichen wollen. Dazu sind lediglich der Port und das Flag mit den im Aufruf bereits angedeuteten Werten zu belegen und schon sollte auch hier Tür und Tor geöffnet sein. Nun sollte es uns gelingen, die gestellte Aufgabe zu lösen.

Versenden von Dateien

Stellen wir uns eine weitere Aufgabe vor. Aus unserer VO-Applikation soll über das http-Protokoll eine Datei an unsere Web-Applikation versendet werden. Dies kann zB. eine Bilddatei oder eine Textdatei mit den Lizenzdaten Ihres Programmes sein, damit Sie auch erfahren, wer sich wann welches Update von Ihrem Webserver geholt hat.

Auch das ist mit meiner Methode GetDocumentByGetOrPost() möglich. Allerdings ist hier ganz genau auf den Aufbau der Daten zu achten. Schauen wir uns zunächst ein Beispiel an:

```
oHttp := cHttp{"TestPOST"}
// to identify the boundary something like
that
cBoundary := "--UPLOAD--BOUNDARY--huklfehnv--
UPLOAD-BOUNDARY--"

// include to the header
cHeader := "Accept: text/plain, *" + "/" +
"*" + CrLf +;
  "Content-Type: multipart/form-data; bounda-
ry=" + cBoundary + CrLf

// send data
// The dashes before and after the boundary
value,
// as well as the positioning of the CrLf
variables
// is critical to the successful functioning
of this process.
// Also, if you plan on posting binary data
to the form,
// run the data through the Base64 algorithm.
// EncodeBase64(<hSource>, <hDestination>) --
-> nBytesEncoded
cPostData := "--" + cBoundary + CrLf +;
  [Content-Disposition: multipart/form-data;
name="Datei"; filename="test.txt"] + CrLf +;
  "Content-Type: text/plain" + CrLf + CrLf +;
  MemoRead("D:\WRK\Daten\test.txt") + CrLf +;
  "--" + cBoundary + CrLf +;
  [Content-Disposition: form-data;
name="Bestand"] + CrLf + CrLf +;
  [200 Teile zu bestellen] + CrLf +;
  "--" + cBoundary + "--"

cPage := oHttp:GetDocumentByForm( "www.myser-
ver.com",;
  "/temp/savefile.php?text=Ok",;
  cPostData,;
  cHeader)

oHttp:CloseRemote()
oHttp:Axit()
```

Als erstes müssen wir einen "Begrenzer" definieren. Dies ist eine beliebige Zeichenkette, die nur eindeutig sein muß und nirgends in den zu übermittelnden Daten vorkommen darf. Im Header muß dieser "Begrenzer" angegeben werden und teilt der Web-Applikation mit, wo die Daten beginnen und wo sie wieder zu Ende sind.

Der Aufbau der Daten muß nun genau nach dem angegebenen Beispiel erfolgen. Die Daten beginnen mit zwei Bindestrichen (--). Ein Begrenzer vom Begrenzer, wenn Sie so wollen. Dann kommt der eigentliche Begrenzer, gefolgt von einem CrLf. Im Header der Daten erfolgt die Beschreibung der zu sendenden Daten die vom Header durch zwei CrLf wieder getrennt sind. Die Daten werden von zwei Bindestrichen und dem Begrenzer abgeschlossen. Wie das Beispiel zeigt, können auf diese Weise auch mehrere Dateien gleichzeitig versendet werden. Wichtig ist dabei, dass auch am Schluß aller gesendeten Dateien wiederum zwei Bindstriche gesetzt werden. Ich habe in diesem Beispiel als Textbegrenzer die eckigen Klammern verwendet. Wenn Sie das Beispiel im Aufbau exakt übernehmen, sollten sich die meisten Aufgaben in diesem Zusammenhang lösen lassen.

Nachtrag

Ja, wenn Sie mich nun fragen, wie kommt man denn da darauf? Dann muß ich zuerst meinem Sohn danken, der sich gut in der Welt von PHP und Linux auskennt, leider aber nicht bei VO. Zum anderen benötigt man natürlich einen Netzwerk-Protokoll-Analyser. Die einen nehmen dazu Telnet. Ich verwende jenen von Ethereal. Dieser ist zu bekommen bei www.ethereal.com. Ausgiebiges Studium der MSDN und diverse Internet-Recherchen waren ebenfalls notwendig.

Mag. Norbert Kolb