Inserting RTF code into a rich edit control

We all know how to load RTF code into a rich edit control. For example, if *RE* is a *TRichEdit* control, to load from a stream we'd use:

```
RE.Lines.LoadFromStream(Stream);
```

Or, if we're really careful:

```
RE.PlainText := False;
RE.MaxLength := Stream.Size; // ensures long docs can display
RE.Lines.LoadFromStream(Stream);
```

That code overwrites the existing content. But what if we want to insert RTF code into some pre-existing code? Well, we need to hit the Windows API.

We have to fill in a structure and set up some flags then pass them to the rich edit control via a message. We also need a call back function that Windows calls when it wants to read a chunk of data.

The structure has three fields. The first takes a user-defined "cookie" that is passed to the callback function. The next field is an error code that the callback function passes back to the routine that sends the message. If this is non-zero the insertion should terminate. The final field is a reference to the callback function.

I always think it's more flexible to read from streams rather than files, so the following example code shows how to insert RTF code from a stream:

```
procedure RTFInsertStream(const RE: TRichEdit; const Stream: TStream);
var
  EditStream: TEditStream;  // callback used to read inserted RTF
begin
  RE.Lines.BeginUpdate;
  try
    // Make sure rich edit is large enough to take inserted code
    RE.MaxLength := RE.MaxLength + Stream.Size;
    // Stream in the RTF via EM_STREAMIN message
    EditStream.dwCookie := DWORD(Stream);
    EditStream.dwError := $0000;
    EditStream.pfnCallback := @EditStreamReader;
    RE.Perform(
      EM_STREAMIN,
      SFF_SELECTION or SF_RTF or SFF_PLAINRTF, LPARAM(@EditStream)
    );
    // Report any errors as a bug
    if EditStream.dwError <> $0000 then
      raise Exception.Create('RTFInsertStream: Error inserting stream');
  finally
    RE.Lines.EndUpdate;
  end;
end;
```

First we freeze the rich edit control then make sure it has sufficient capacity to receive the inserted code. Next we set up the structure:

- The *dwCookie* field receives a reference to the stream so that we can access it in the callback function.
- We initialise *dwError* to zero to indicate no error.
- *pfnCallback* is set to reference the function that Windows calls back to.

Then we send the message. The flags mean that the insertion will replace any current selection (*SFF_SELECTION*), we're inserting RTF code rather than text (*SF_RTF*) and that only keywords common to all languages are used (*SFF_PLAINRTF*). After the message call returns we check that the *dwError* field is still 0 and raise an exception if not. If you don't want to overwrite selections leave out the *SFF_SELECTION* flag.

All that remains is to define the callback function. Here it is:

```
function EditStreamReader(dwCookie: DWORD; pBuff: Pointer;
  cb: LongInt; pcb: PLongInt): DWORD; stdcall;
begin
```

```
  Result := $0000;  // assume no error
  try
    pcb^ := TStream(dwCookie).Read(pBuff^, cb); // read data from stream
  except
    Result := $FFFF;  // indicates error to calling routine
  end;
end;
```

The *dwCookie* parameter contains the stream reference we prevously stored in the *TEditStream* structure. We cast this parameter to *TStream* and use it to read the stream. The *cb* parameter contains the number of bytes Windows wants us to provide so we attempt to read *cb* bytes from the stream into the buffer pointed to by the *pBuff* parameter. *pcb^* is set to the number of bytes actually read. We return zero on success or non-zero if there is an error. Windows passes this return value back to the caller via the *TEditStream.dwError* field.

To make this code work you'll need the following **uses** clause:

```
uses
  Windows, Classes, ComCtrls, RichEdit, SysUtils;
```

Adapting the code

If you prefer to read directly from a file, you could adapt *RTFInsertStream* to take a file name as a parameter, open the required file and store its handle in *TEditStream.dwCookie*. You would then read the file handle in the call back function. You would also want to change the name of *RTFInsertStream*!

Author:	Peter Johnson
Contributor:	Peter Johnson
Added:	2007-10-29
Last updated:	2007-10-29