masutils unit

Routines

String handling
CutOutStrs
Like function
HexToInt
CheckIfHex
IsDigit
IsLower
IsUpper
Proper
ToLower
ToUpper

Files
GetAssociation
FindFile

Mats Asplund/MAs Prod.

HexToInt function

Unit
masutils

Declaration
```
function HexToInt(HexStr: string): Integer;
```

Description
Converts the hexadecimal value (0-9,A-F) in HexStr to an integer.
Returns -1 if conversion didn't succeed.

CutOutStrs procedure

Unit
masutils

Declaration
```
procedure CutOutStrs(Str, DelCh: string; var SStrs: TStringList);
```

Description
Use CutOutStrs to extract substrings from a string using delimiters.
Str is the string itself. DelCh is the delimiter-character.
The result is returned in SStrs.

FindFile function

Unit
masutils

Declaration
```
function FindFile(const filespec: TFileName): TStringList;
```

Description
Looks for files specified in filespec. Scans subdirectorys as well.

Example
Find all .exe-files and write them to Memo1.

```
procedure TForm1.Button1Click(Sender: TObject);
var List: TStringList;
begin
    List:= TStringList.Create;
    List:= FindFile('*.exe');
    Memo1.Text:= List.Text;
    List.Free;
end;
```

GetAssociation function

Unit
masutils

Declaration
```
function GetAssociation(const FileName: string): string;
```

Description
This function retrieves the full pathname for the application associated with FileName.

Like function

Unit
masutils

Declaration
```
function Like(AString, Pattern: string; CaseSensitive: boolean): boolean;
```

Description
This function compares a string with a pattern, and returns true if there is a match.
Use '?' for one "don't care" - character, and '*' for one or more "don't care" - characters.

Example
Like('Delphi', '?e?p*', false) will return true.

IsDigit function

Unit
masutils

Declaration
```
function IsDigit(ch: char): boolean;
```

Description
Returns true if ch is a digit ('0'-'9').

IsLower function

Unit
masutils

Declaration
```
function IsLower(ch: char): boolean;
```

Description
Returns true if ch is a lowercase character.

IsUpper function

Unit
masutils

Declaration
```
function IsUpper(ch: char): boolean;
```

Description
Returns true if ch is a uppercase character.

Proper function

Unit
masutils

Declaration
```
function Proper(const s: string): string;
```

Description
Capitalizes first letter of every word in s.

ToLower function

Unit
masutils

Declaration
```
function ToLower(ch: char): char;
```

Description
Changes a character to a lowercase letter.

ToUpper function

Unit
masutils

Declaration
```
function ToUpper(ch: char): char;
```

Description
Changes a character to an uppercase letter.

CheckIfHex function

Unit
masutils

Declaration
```
function CheckIfHex(Str: string; ShowMsg: boolean): boolean;
```

Description
The function returns true if all characters in Str is hexadecimal ('0'..'9', 'A'..'F', 'a'..'f'). If ShowMsg is true a message will show, indicating which character-positions that is wrong.

HelpScribble

HelpScribble is a help authoring tool written by Jan Goyvaerts. This help file was created with the unregistered version of HelpScribble, which is why you can read this ad. Once the author of this help file is so honest to register the shareware he uses, you will not see this ad again in his help files.
Recompiling the help project with the registered version is all it takes to get rid of this ad.

HelpScribble is a stand-alone help authoring tool. It does *not* require an expensive word processor. (Only a help compiler as Microsoft likes keeping the .hlp format secret. Not my fault.)

Here are some of HelpScribble's features:

- The Setup program will *properly* install and uninstall HelpScribble and all of its components, including registry keys.
- Create, edit and navigate through topics right in the main window. No need to mess with heaps of dialog boxes.
- All topics are listed in a grid in the main window so you won't lose track in big help projects. You can even set bookmarks.
- Use the built-in Browse Sequence Editor to easily create browse sequences.
- Use the built-in Window Editor to change the look of your help window and create secondary windows.
- Use the built-in Contents Editor to create Windows 95-style contents files. Works *a lot* better than Microsoft's HCW.
- No need to mess with Microsoft's SHED: use the built-in SHG Editor to create hotspot bitmaps. Draw your hotspots on the bitmap and pick the topic to link to from the list.
- With the built-in Macro Editor you can easily compose WinHelp macros whenever needed. It will tell you what the correct parameters are and provide information on them.
- If you have a problem, just consult the online help. The help file was completely created with HelpScribble, of course.
- HelpScribble is shareware. However, the unregistered version is *not* crippled in any way. It will only add a small note to your help topics to encourage you to be honest and to register the shareware you use.

These options are very interesting for Delphi and C++Builder developers:

- If you are a component writer, use the Delphi Parser to build an outline help file for your component. Just fill in the spaces and you are done. HelpScribble can also extract the comments from your source file and use them as the default descriptions.
- If you are an application writer, HelpScribble provides you with a property editor for the HelpContext property. You can select the topic you need from a list of topic titles or simply instruct to create a new topic. No need to remember obscure numbers.
- The property editor also provides a tree view of all the components on your form and their HelpContext properties. This works very intuitively. (Much nicer than those help tools that simply mess with your .dfm files.)
- HelpScribble can perform syntax highlighting on any Delphi source code in your help file.

HelpScribble is shareware, so feel free to grab your copy today from my web site at http://www.ping.be/jg/