DocMaker

Manual

By Marcus ‚MAC' Röckendorf

Version 1.0

Content

General

DocMaker is a tool to automatically create html documentations for PureBasic source codes. It is comparable to Javadoc but with some nice additional features. This tool is not designed to have doclets in realtime, but to compile a html file with all Procedures, Enumerations and Structures inside.

All you have to do in your source code is nothing more than adding some comments like this:

```
Procedure DocMakerTest(SomeText.s, Value.i=0)
  ;@ This is a procedure to test DocMaker's functionality.
  ;@param SomeText Some text to give to the procedure.
  ;@param Value A value to give the procedure as a parameter. <i>(optional)</i>
  ;@return This procedure returns nothing.
  Debug "The text: "+SomeText
  Debug "Given Value: "+Value
EndProcedure
```

And this way you have immediately created the documentation for this Procedure:

Procedures

DocMakerTest()

This is a procedure to test DocMaker's functionality.

Syntax:
DocMakerTest(SomeText.s, Value.i=0)

Parameters:

Parameter	Description
SomeText	Some text to give to the procedure.
Value	A value to give the procedure as a parameter. *(optional)*

Returns:
This procedure returns nothing.

DocMaker readable lines start with ;@ and a tag or a comment. If no tag is used the following text will be considered a description. Descriptions are available anytime using the current context (main program, procedure, enumeration, …). Descriptions don't have to be written together. DocMaker will collect all descriptions and export them as a block, no matter where they are in the source code.

For example to make a simple comment to be added to the documentation as plain description text you would use:

```
;@ This is some description.
```

An example for a tag would be:

```
;@param Value This is the description for the parameter.
```

to define a parameter of a procedure with the name "Value" and its description.

The easiest way to see how it works is to take a look at the file "DocMakerDemo.pb" and the resulting html documentation "DocMakerDemo.html".

Features

- Automatically collects general, procedure, enumeration and structure comments
- Sorting of procedures, enumerations and structures to chapters
- Configurable html style (colors, sizes, …)
- Configurable export options (export a table of contents yes/no, process enumerations, structures, procedures, …)
- Automatic creation of internal links
- Support for example code, images, links, …

Tags

General Tags

General tags are all tags that are not inside a Procedure, Enumeration or Structure block but in the main code.

;@title

Lets you define the title of the documentation. You would basically use the name of your project here. If this tag is not used in the source code, the first filename in your list will be used in the html documentation.

Syntax:

```
;@title[space]Some text
```

Example:

```
;@title DocMaker Demo
```

;@author

The author tag is used to define an author for the document. It will be written along with the version tag after the document title. Author and version tags can also be used inside of Procedures to have own authors or versions there that will be written inside the Procedure's documentation.

Syntax:

```
;@author[space]The author's name
```

Example:

```
;@author Marcus 'MAC' Röckendorf
```

;@version

The version tag lets you define a version for the whole project. This is not limited to numbers but can be any text. Author and version tags can also be used inside of Procedures to have own authors or versions there that will be written inside the Procedure's documentation.

Syntax:

```
;@version[space]The version text
```

Example:

```
;@version 1.3b
```

;@chapter

The chapter tag can be used to structure your documentation and give it a natural flow. The chapter tag can be used more than once in your source code. The procedures that are defined in a chapter will be sorted when the html documentation is exported. If you leave the chapter name blank, it will be switched to the "General" chapter.

Syntax:

```
;@chapter[space]The name of the chapter
```

Example:

```
;@chapter General

… some procedure declarations here …
```

```
;@chapter Basic functions

… some procedure declarations here …

;@chapter Misc functions

… some procedure declarations here …

;@chapter Basic functions

… some procedure declarations here that will be processes as if they were after the
first chapter “Basic functions” tag
```

Procedure Tags

Procedure tags are all tags that can be used inside a Procedure (between "Procedure …" and "EndProcedure").

;@author

The author tag is used to define an author for the procedure if he differs from the author defined for the whole project.

Syntax:

```
;@author[space]The author's name
```

Example:

```
;@author Marcus 'MAC' Röckendorf
```

;@version

The version tag lets you define a version for the procedure . This way it is possible to use other versions for the single procedures than for the whole project.

Syntax:

```
;@version[space]The version text
```

Example:

```
;@version 1.3b
```

;@param

The param tag is used to define the procedure's parameters. It is to be used with a parameter name and a description. The description can include html tags (e.g. to write in italics) and "\n" to insert a line feed. Of course you can also use
 for this.

Syntax:

```
;@param[space]Parameter name[space]The parameter description
```

Example:

```
;@param Text This is the <i>text</i> to use.\nAnd this is a <b>second</b> line.
```

;@return

The return tag is used to describe what a procedure returns.

Syntax:

```
;@return[space]Some describing text
```

Example:

```
;@return This procedure returns an integer with the computed value.
```

;@code

Code tags can be used to add an example code to your documentation. It will be exported in a <pre></pre> block to make it look like code. You can use one or more line. All lines with this tag will be collected and exported as a block.

Syntax:

```
;@code[space]Some code line
```

Example:

```
;@code Delay(500) ; Waits for 500 miliseconds

;@code End ; Ends the program
```

;@link

The link tag is used to insert links to other procedures, enumerations or external sources. These links will be written to a “See also” section to the end of the procedure’s description.

Syntax:

```
;@link[space]type[space]Link
```

Type defines the type of the link. Possible values for this are “proc”, “enum” and “ext”.

“proc” defines an internal link to a procedure (see the demo project to see how it works). If you use “proc” the “link” value will be the name of a procedure without its parameters (e.g. “Addition()”).

“enum” will create an internal link to a enumeration block or list of constants. If you use this type, the “link” value will be the name of the enumeration block.

“ext” will create a link to an external source or website.

Example:

```
;@link proc Addition()

;@link ext http://www.purebasic.com
```

;@event

If you have a procedure that processes events (e.g. #PB_Event_Gadget) you can define those events with the event tag. All events will be collected and written to the procedure’s documentation under “Supported events”.

Syntax:

```
;@event[space]Name of the event
```

Example:

```
;@event #PB_Event_LeftClick

;@event #PB_Event_LeftDoubleClick
```

;@image

This tag lets you include an image into your documentation. Images are always drawn above the procedures description. The filename has to be entered with a relative or absolute path.

Syntax:

```
;@image[space]The file name
```

Example:

```
;@image Logo.png

;@image images\nicepic.jpg
```

;@attribute

The attribute tag was especially added for me to create the documentation for “sqUId”, my own UI library, which uses specific attributes for each UI element available. If you use the attribute tag it will create a table like the parameter table to list up all attributes with their description.

Syntax:

```
;@attribute[space]Name[space]Rights[space]Description
```

“Name” is the name of the attribute.

“Rights” is used to include a column that shows something like this: “R” or “R/W”.

“Description” is the description of the attribute. Supports html tags and “\n”.

Example:

```
;@attribute #UI_Attribute_Width R/W This will read/write the width of the gadget.
```

Enumeration Tags

Enumeration tags are all tags that can be used inside an enumeration block (between "Enumeration …" and "EndEnumeration"). DocMaker will collect all constants declared inside the block and list them up in the exported documentation. Comments that are written behind the constant name will be read and used as the constants description:

```
Enumeration
  ;@name Mathematical functions
  #Math_Addition                    ; Defines an addition
  #Math_Substraction                ; Defines a subtraction
  #Math_Multiplication              ; Defines a multiplication
  #Math_Division                    ; Defines a division
EndEnumeration
```

Will be used this way:

Enumerations

Mathematical functions

Constant	Description
#Math_Addition	Defines an addition
#Math_Substraction	Defines a subtraction
#Math_Multiplication	Defines a multiplication
#Math_Division	Defines a division

;@name

The name tag is used to define a name for an enumeration block.

Syntax:

```
;@name[space]The name of the enumeration
```

Example:

```
;@name Mathematical functions
```

Structure Tags

Structure tags are all tags that can be used inside a structure definition (between "Struture …" and "EndStructure"). The only supported tag is the "name" tag and the general descriptions (";@ …").

;@name

The name tag is used to define a name for a structure.

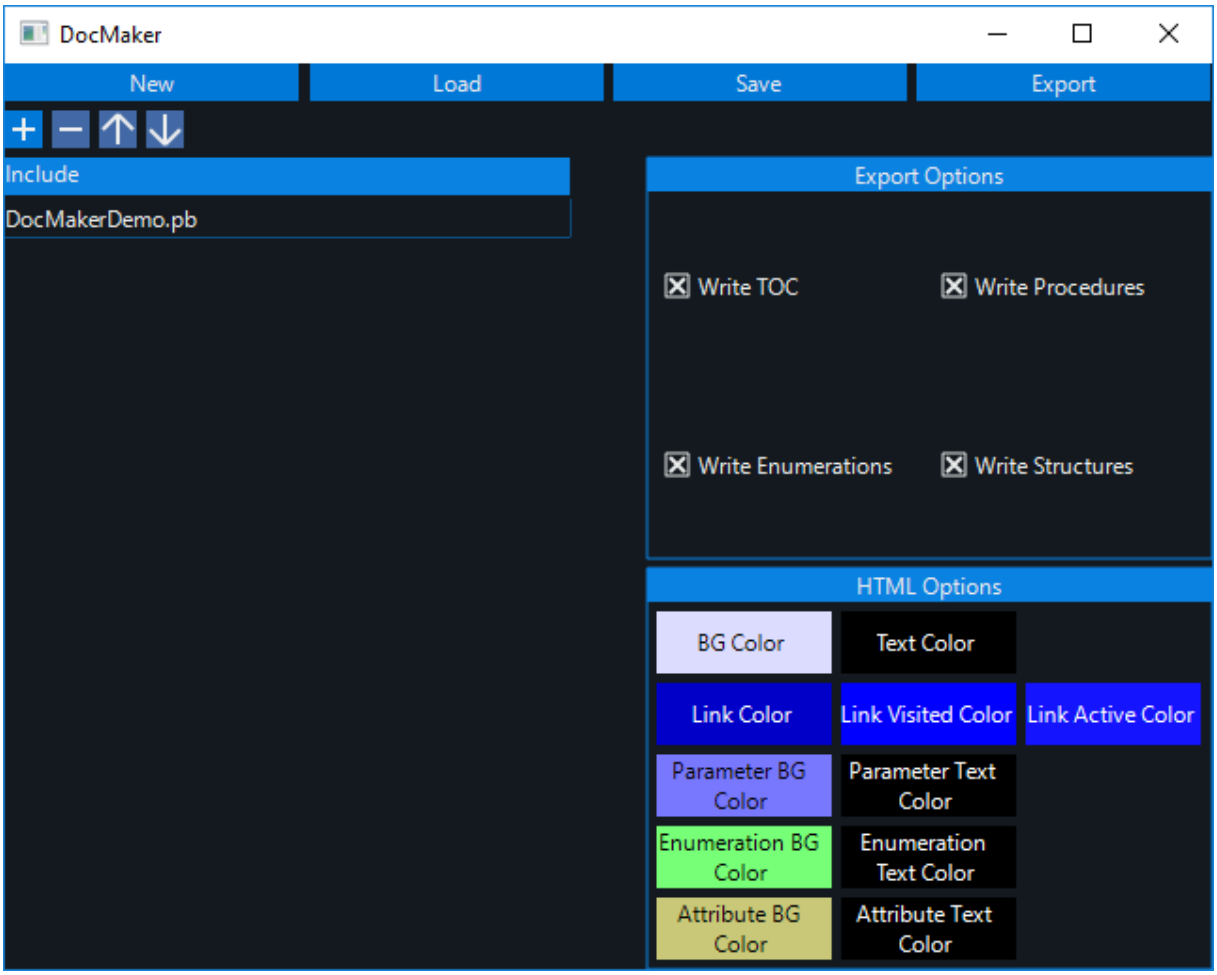
Syntax:

```
;@name[space]The name of the structure
```

Example:

```
;@name Entity structure
```

The Tool



Working with the tool shouldn't be a problem, since it is quite self explanatory.

Project buttons

The button toolbar on the top is used to create a new or load, save and export your project.

File list

The list on the left side contains all the files to process when creating the html documentation. You can add or remove files with the "+" and "-" buttons. To change the order of the files in the list use the arrow up/down buttons.

Export options

On the right side are options for exporting:

Write TOC – Should a table of contents (index) be created?

Write Procedures – If procedure documentations should be exported.

Write Enumerations – If enumeration documentations should be exported.

Write Structures – If structures should be exported.

These will be saved with your project.

HTML Options

On the lower right side are buttons to configure the colors to use in the documentation. These will be saved with your project.

Source code

The source code for the tool is included but won't be of much help since it uses "sqUId" as the UI system which is not included. So you won't be able to compile the source code, but maybe it will help to see how the processing works.