ROBODoc

안낙호

nako_kr1 (at) yahoo.co.kr

ROBODoc은 다양한 언어의 주석을 이용하여 문서를 제작해주는 매우 편리한 Documentation Tool입니다. 이 글은 ROBODoc에 대한 간단한 소개와 사용법을 담고 있습니다.

고친 과정

고침 1.02003-04-17고친이 nako an

최초 작성

1. 머리글

1.1. ROBODoc 소개

ROBODoc은 다양한 언어에서 사용할 수 있는 Documentation Tool이다. http://sourceforge.net/projects/robodoc에서 프로젝트가 진행중이며 이곳에서 소스를 받을 수 있다. 프로젝트의 홈페이지는 http://www.xs4all.nl/~rfsber/Robo (http://www.xs4all.nl/~rfsber/Robo) 이다.글을 쓰고 있는 현재 버전은 4.0.0이다.

ROBODoc은 매우 많은 언어를 지원한다. ROBODoc은 고유한 주석형태를 가지고 있기 때문에 사용하는 언어가 무엇인가에 크게 영향받지 않는다. ROBODoc이 지원한다고 메뉴얼에서 밝힌 언어는 다음과 같다.

C, C++, Java, Assembler, Basic, Fortran, LaTeX, Postscript, Tcl/Tk, LISP, Forth, Perl, Shell Scripts, Occam, COBOL, HTML와 기타 언어들.

ROBODoc은 문서화 작업을 위해 소스내의 주석을 사용하기 때문에 개발과 문서화 작업을 통합할 수 있고 문서의 업데이트를 용이하게 할 수 있다. 뿐만 아니라 HTML,LaTex,XML DocBook 을 비롯한 다양한 형식의 출력포맷을 지원하고 다중 파일형식으로 출력할 것인지 단일 파일로 출력할 것인지 등을 지정할 수 있게 한다.

ROBODoc은 설치와 사용법이 매우 간단하다. 배포되는 소스코드는 ROBODoc을 사용하여 주석처리 되어 있으며, 사용자들이 간단히 copy & paste로 사용할 수 있는 유용한 template도 함께 제공한다. 주석처리 문법은 계층구조를 지원하며 아주 직관적으로 구성되어 있어, 처음 보는 사람도 쉽게 익힐 수 있게 되어 있다.

이 문서에서는 다양한 언어와 출력포맷의 지원, 손쉬운 설치와 사용방법, 개발작업과 문서작업의 통합과 같은 특징을 지닌 ROBODoc을 이용해 문서화작업을 진행하는 방법을 살펴본다.

1.2. 저작권 정보

1.3. 책임의 한계

본 저자는 문서의 내용이 야기할 수 있는 어떠한 결과에 대해서도 책임을 지지 않습니다. 본 문서에서 내포하고 있는 정보들 및 예제들은 여러분이 알아서 활용하십시오. 비록 최선을 다 했으나 이 문서는 틀린 점이나 오류가 있을 수도 있습니다. 만약 여러분이 틀린 점을 발견했다면 꼭 저에게 알려 주시기 바랍니다.

1.4. 피드백

이 문서에 대한 발전적인 제안이나 수정사항, 문제점 등에 대한 피드백은 언제든지 환영합니다. `<nako_kr1 (at) yahoo.co.kr>`로 메일을 보내 주십시오.

2. ROBODoc의 설치

ROBODoc의 설치방법은 다음과 같다.

적당한 디렉토리에 `robodoc-4.0.0.tar.gz`파일을 갖다 놓는다.

#tar zxvf robodoc-4.0.0.tar.gz

#cd robodoc-4.0.0

#./configure

#make

#make install

이것으로 설치는 모두 끝났다. 이제 아무곳에서나 robodoc을 이용할 수 있다.

3. Code 내에 header 작성하기

소스 코드 내에 header를 작성하는 것으로 문서화를 위한 준비는 모두 끝난다.ROBODoc의 header는 매우 간단하면서도 체계적으로 구성되어 있다.

3.1. 형식

Documentation을 위해 소스코드 내에 삽입되는 요소는 Header라고 한다. Header는 begin marker와 end marker, item으로 구성되어 문서화 작업의 한 블록을 이룬다. 다음의 예제를 보자.

```
/****f* module-1/killhim
 * NAME
 *   killhim
 * SYNOPSYS
 *   int killhim(char *name, char *method, int count)
 * FUNCTION
 *   대상과 수단, 횟수를 골라 방법하는 함수다.
 * INPUT
 *   name -- 대상
 *   method -- 수단
 *   count -- 횟수
 * RESULT
 *   0 for success, else an error code return
 ******
 * 여기에 쓰는 주석은 문서화 되지 않는다.
 */

 int killhim(char *name,char *method, int count){
     if((strcmp(name,"bush"))==0)
         count *= 100;
     while(count-- > 0)
         hithim(name,method);
     return 0;
 }
```

위의 예제는 하나의 header를 이루고 있다. 이 header는 아래의 killhim에 대한 주석이자 문서가 되는 것이다.

```
****f* module-1/killhim
```

이 부분이 begin marker이다. 헤더가 시작됨을 표시한다. 플래그와 뒤의 계층에 대해서는 뒤에서 설명하겠다.

```
 * NAME
 *   killhim
 * SYNOPSYS
 *   int killhim(char *name, char *method, int count)
```

item 부분이다. item들은 정의되어 있으며 사용자 정의를 추가할 수도 있다.

```
******
```

end marker다. 이 표시를 기준으로 header는 끝난다. 이후에 이어지는 주석은 robodoc에 영향을 주지 못한다.

각각의 마크 앞에는 사용하는 언어의 주석이 붙게 된다. 다음은 각 언어별로 mark하는 방식이다.

```
/****       C, C++
 *
 ***

//****      C++
//
//***

(****       Pascal, Modula-2
 *
 ***
*)

{****       Pascal
 *
 ***
*}

;****       M68K assembler
;
;***

****        M68K assembler
*
***

C     ****  Fortran
C
C     ***


REM ****    BASIC
REM *
REM ***

%****       LaTeX, TeX, Postscript
%
%***
```

```
#****       Tcl/Tk
#
#***

****  COBOL
*
***

--****      Occam
--
--***

|****       GNU Assembler
|
|***

!!****      FORTRAN 90
!!
!!***
```

3.2. Header Type

header type은 현재 작성중인 헤더가 무엇에 대한 헤더인지를 알려준다. 즉, 함수에 대한 헤더인지 모듈에 대한 헤더인지에 대한 표시인 것이다. 헤더를 표시함으로써 인덱싱과 정렬을 효과적으로 할수 있게 된다.

앞의 begin marker ****f에서 "f"가 바로 헤더 타입이다. 지원하는 헤더 타입은 아래와 같다.

```
h -- 모듈(module in project)
f -- 함수(function)
s -- 구조체(structure)
c -- 클래스(class)
m -- 메소드(method)
v -- 변수(variable)
d -- 상수(constant)
* -- 일반(generic header)
```

3.3. item

ROBODoc에 정의된 item은 다음과 같으며, robodoc.rc파일을 이용해 추가할 수 있다.

- NAME -- Item name plus a short description.

- SYNOPSIS, USAGE -- How to use it.
- FUNCTION, DESCRIPTION, PURPOSE -- What does it do.
- AUTHOR -- Who wrote it.
- CREATION DATE -- When did the work start.
- MODIFICATION HISTORY, HISTORY -- Who has done which changes and when.
- INPUTS, ARGUMENTS, OPTIONS, PARAMETERS, SWITCHES -- What can we feed into it.
- OUTPUT, SIDE EFFECTS -- What output is made.
- RESULT, RETURN VALUE -- What do we get returned.
- EXAMPLE -- A clear example of the items use.
- NOTES -- Any annotations
- DIAGNOSTICS --Diagnostic output
- WARNINGS, ERRORS -- Warning and error-messages.
- BUGS -- Known bugs.
- TODO, IDEAS -- What to implement next and ideas.
- PORTABILITY -- Where does it come from, where will it work.
- SEE ALSO -- References to other functions, man pages, other documentation.
- METHODS, NEW METHODS -- OOP methods.
- ATTRIBUTES, NEW ATTRIBUTES -- OOP attributes
- TAGS -- Tag-item description.
- COMMANDS -- Command description.
- DERIVED FROM -- OOP super class.
- DERIVED BY -- OOP sub class.
- USES, CHILDREN -- What modules are used by this one.
- USED BY, PARENTS -- Which modules do use this one.
- SOURCE -- Source code inclusion.

3.4. section

/****f* module-1/killhim 에서 module-1/killhim에 해당하는 부분이 section이다. section은 문서의 계층화를 위해서 존재한다. 프로젝트/모듈/(함수 또는 변수나 class 등)으로 계층을 나누어 section을 활용하면 문서로 변환할 때 ROBODoc은 이를 인식하고 인덱싱해준다. 예를 들어 다음과 같은 구조의 프로젝트가 있다고 하자.

```
-ProjectA -- module1 -- function1
```

```
                   -- variable1
  -- module2 -- function2
             -- function3
```

이 경우 각각의 헤더는 다음과 같이 section으로 나눌 수 있다.

```
/****h* ProjectA/module1
...
/****h* ProjectA/module2
...
/****f* module1/function1
...
/****v* module1/variable1
...
/****f* module2/function2
...
/****f* module2/function3
...
```

이렇게 section을 지정한 후 ROBODoc을 이용해 HTML변환을 하면 다음과 같이 indexing 된다.

```
1 ProjectA/module1
1.1 module1/function1
1.2 module1/variable1
2 ProjectA/module2
2.1 module2/function2
2.2 moudle2/function3
..
```

4. ROBODoc으로 문서 생성하기

ROBODoc은 문서를 직접 생성한다. 일부 Documentation Tool처럼 중간과정에서 파일을 생성하고 다시 그 파일을 다른 문서형식으로 변환하는 방식이 아니라 HTML이든 LaTex든 상관 없이 소스코드에서 바로 문서를 생성해준다.

4.1. 세가지 모드

ROBODoc 문서화에는 세가지 모드가 있다.

singledoc 모드는 출력되는 문서를 하나의 파일로 저장한다. 반면 multidoc은 문서를 여러개의 작은 조각으로 나누어 준다. 이 모드는 browser로 보아야 하는 HTML형식의 문서를 출력할

때 매우 좋다. singlefile 모드는 디버깅 목적으로 존재한다고 하는데 정확히 무슨 뜻인지는 잘 모르겠다.

4.2. 출력 포맷

다음 4가지의 출력 포맷을 지원한다. 옆에는 각각의 출력포맷에 따른 옵션을 적어놓았다.

```
RTF, --rtf
HTML, --html
LaTeX, --latex
XML DocBook, --dbxml
```

4.3. option

ROBODoc에 사용되는 몇가지 중요한 옵션들이다.

```
--doc
  문서가 출력될 path를 지정한다.

--index
  master index 파일을 생성해준다. 실제로 해보면 multidoc 모드와 함께 쓰여질 때에만
  생성된다. html 파일을 생성할 경우 masterindex.html파일이 생성된다.

--multidoc
  하나의 소스당 하나의 문서를 생성한다. 디렉토리 구조를 계층구조로 사용한다.

--nodesc
  subdirectory 를 제외하고 top directory에 대해서만 문서화를 한다.

--src
source 파일과 디렉토리를 지정한다.
```

앞서 서술한 모드,출력포맷은 이름 그대로 옵션으로 사용한다. 예를 들어 multidoc모드는 --multidoc 옵션으로 넣어준다.

4.4. robodoc.rc파일

`robodoc.rc`파일은 robodoc사용을 최적화 하기 위한 설정 파일이다. `robodoc.rc`파일에는 item의 추가, 변환 옵션의 지정등이 가능하다. 다음은 manual에 나온 `robodoc.rc`파일의 예이다.

`robodoc.rc`

```
items:
    NAME
    SYNOPSIS
    INPUTS
    OUTPUTS
    HISTORY
ignore items:
    HISTORY
    BUGS
options:
    --src
    ./source
    --doc
    ./doc
    --html
    --multidoc
    --index
extensions:
    .bak
    ~
```

이 파일을 수정해 현재 작업 디렉토리에 넣어두고 robodoc을 실행시키면 된다.

4.5. 사용예

이제 만들어진 header를 이용하여 실제 문서를 생성하는 예를 보도록 하겠다. 아래의 예제에서는 source 디렉토리의 상위디렉토리에서 실행한 것이다. source의 top directory를 %source%라 하겠다.

HTML로 출력시

#robodoc --src ./%source% --doc ./docs --multidoc --index --html

%source% 에 있는 소스를 이용하여 ./docs 디렉토리에 문서를 생성하게 된다. html형식으로 출력하고 multidoc 모드이므로 파일을 나누게 된다. --index 옵션에 의해 masterindex.html파일을 생성한다. 주의할 것은 --src나 --doc 옵션 다음에 디렉토리가 올 경우 반드시 ./ 나 / 로 시작해야 한다는 것이다. 이를 지키지 않으면 에러 메세지를 보게 된다.

RTF로 출력시

#robodoc --src ./%source% --doc docu --singledoc --rtf --sections

rtf는 browse 할 수 있는 파일이 아니므로 --singledoc 옵션으로 변환한다. 위 명령의 결과로 docu.rtf 파일이 생성된다.

LaTex로 출력시

#robodoc --src ./%source% --doc docu --singledoc --latex --sections

결과로 docu.tex파일이 생성된다. 생성된 tex파일을 ps,dvi등으로 변환하는 것은 다른 문서를 참고하라. 아직 한글로는 실험해보지 않았기 때문에 한글이 잘 출력되는지는 장담할 수가 없다.

5. 기타

ROBODoc의 사용법에 대해서 살펴보았다. 이 문서는 소스와 함게 제공하는 manual페이지를 대부분 참고하여 작성하였다. 이 문서에서 다루지 않은 내용이나 불충분한 내용은 `Docs/manual.html`에서 찾을 수 있다.

풍부한 예제를 원한다면 ROBODoc 소스 자체를 살펴보면 된다. 배포되는 ROBODoc소스는 ROBODoc을 이용할 수 있도록 헤더가 삽입되어 있다.

소스의 Headers/ 디렉토리에는 바로 가져다 쓸수 있게 헤더들을 모아놓았다. 자신이 사용하는 언어와 관련된 파일을 열고 가져다 쓰면 편리하다.