

작성: 한동훈( traxacun at gmail.com )

작성일: 2005-05-23

제목: **Advanced ASP.NET 5. NDoc을 통한 코드 문서화**

프로젝트 진행시에 겪게되는 어려움 중에 하나는 코드 문서화에 대한 것이다. 거의 모든 회사들은 일정이 빠듯하다고 여겨지며 코딩이 끝난 구현물을 토대로 문서화를 만들게 된다.

<코딩후 문서화>는 여러가지 문제점을 갖고 있다. 개발자가 구현시에 고려했던 내용들을 문서화하는 시점에서는 잊혀지기 때문에 문서화에서 누락되는 경우가 발생한다. 나중에 문제가 발생한 경우 문서만으로 문제를 해결하지 못하고, 원 코드 제작자에게 문의하여 해결하게 되는 것도 이런 탓이다.

코딩후 문서화가 갖는 다른 문제는 개발 문서, 사용자 설명서, 사용자 시나리오 문서를 구분하지 못한채 문서화되는 경우가 많다는 점이다. 대체로 개발 문서에서 코드만 제거하고 사용자 설명서가 되는 경우도 많다.

여기서는 코딩을 하면서 바로바로 문서화할 수 있는 것을 도와주는 NDoc에 대해 살펴볼 것이다. Visual Studio .NET에는 [도구] -> [주석 웹페이지 빌드]를 통해 XML 문서화된 내용을 도움말로 빌드해주는 기능을 갖고 있지만 부족한 부분이 많다. 웹 페이지 뿐만 아니라 윈도우 도움말(.CHM, .HLP)까지 함께 빌드해 줄 수 있는 NDoc 사용법을 살펴볼 것이다.

오픈소스 문서화 도구중에 DoxyGen은 C#의 XML 주석을 이용하지 않고 자체 태그를 사용하고 있기 때문에 .NET 코드 문서화에는 사용하지 않기 때문에 다루지 않는다. 관심있는 분은 DoxyGen 사이트를 참고하기 바란다.

### XML 주석 작성하기

.NET에서는 XML 문서화를 지원하고 있으며 사용할 수 있는 태그목록은 다음과 같다. 보다 자세한 사항은

Recommended Tags for Documentation

Comments(<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vclrfTagsForDocumentationComments.asp>)를 참고하기 바란다.

<c>	<para>	<see>
<code>	<param>	<seealso>
<example>	<paramref>	<summary>
<exception>	<permission>	<value>
<include>	<remarks>	
<list>	<returns>	

표1. XML 문서화 태그 목록

XML 태그 이름에서 알 수 있는 것처럼 대충의 사용 용도는 알 수 있으며 Visual Studio.NET에서는 해당 태그를 입력하면 자동으로 태그의 형식을 입력해준다. Visual Studio .NET이 완성해주는 XML 문서화 태그 입력이 부족하다고 여겨진다면 GhostDoc(<http://www.roland-weigelt.de/ghostdoc/>)을 사용하자. 영어로 문서화를 해야하는 경우엔 유용하게 사용할 수 있다. 다음은 BasePage에 필자가 BR 태그를 개행문자로 변환하기 위해 만든 메서드와 그에 대한 주석이다.

```

#region Br2NL - BR 태그 -> 개행문자
/// <summary>
/// HTML BR 태그 -> 개행문자로 변경한다
/// </summary>
/// <remarks>
/// 개행문자는 윈도우 시스템과 비윈도우 시스템이 다르다.
/// 윈도우 시스템은 \r\n을 개행문자로, 비윈도우 시스템(UNIX/LINUX)은
/// \n을 개행문자로 취급한다.
/// 각 시스템 호환성을 위해 개행문자는 Environment.NewLine을 사용한다.
/// </remarks>
/// <example> BR 태그 -> 개행문자로 변환
/// <code>
/// string result = Br2NL( UserInput.Text );
/// </code>
/// </example>
/// <param name="str">개행문자로 변환할 HTML BR 형식 문자열</param>
/// <returns>개행문자 형식으로 변환된 문자열</returns>
public static string Br2NL( string str )
{
    if( str.Length > 0 )
    {
        str = Replace( str, BRTagPattern, Environment.NewLine );
    }
    return str;
}
#endregion

```

그림1. XML 문서화 사용 예

SUMMARY 태그는 해당 클래스나 메서드의 내용을 요약해서 보여주기 위해 사용한다. REMARKS 태그는 그에 대한 간단한 설명을 덧붙이기 위해 사용하며 EXAMPLE 태그는 예제설명이 시작이라는 것을 나타내며, 예제 코드에 대한 제목 역할을 한다. CODE 태그는 다른 사람이 참고할 수 있는 예제코드를 작성하는 부분이다. PARAM 태그는 각 메서드의 매개변수에 대한 설명이며, RETURNS는 반환값에 대한 주석을 작성할 수 있다. 이렇게 작성된 코드는 컴파일시에 XML 부분만 별도로 작성되는데 이를 위해서는 프로젝트 설정을 해야한다. 명령줄에서 `csc.exe`를 사용하여 직접 컴파일시에는 `/doc` 옵션으로 지정할 수 있다.

### XML 문서화 옵션 지정

Visual Studio.NET에서 [프로젝트] - [BasePage 속성(P)]를 클릭한다.

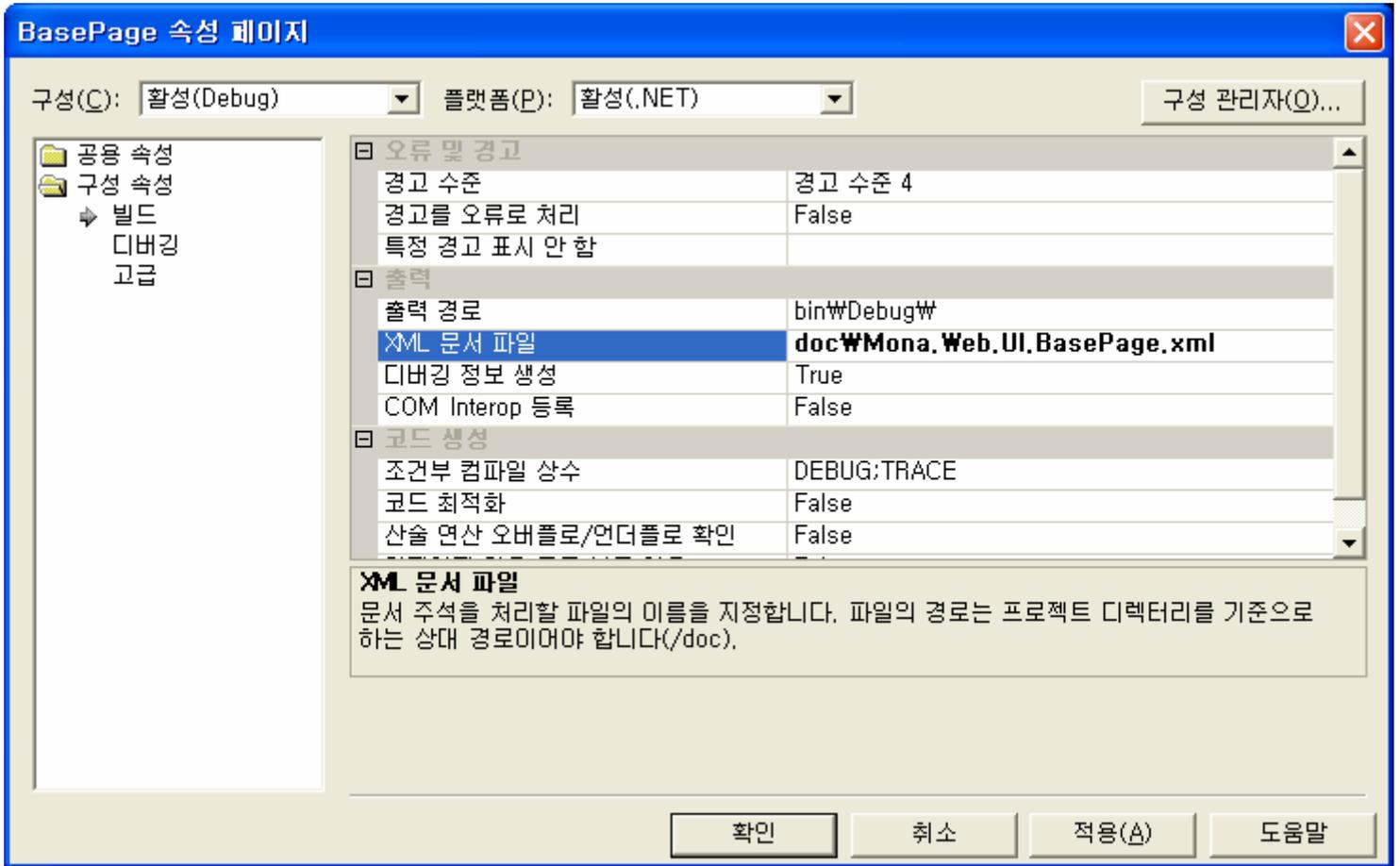


그림2. XML 문서화 설정하기

[XML 문서 파일]에 그림2와 같이 입력한다. 이제 [빌드] - [솔루션 다시 빌드]를 선택하면 전체 프로젝트를 다시 빌드하면서 XML 문서를 생성한다. 개발환경에서 빌드시에는 소스 코드 전체를 빌드하는 대신 수정한 부분만 다시 빌드한다. 이러한 부분 빌드시에는 XML 문서 파일이 갱신되지 않기 때문에 위와 같이 다시 빌드를 해야한다.

이렇게 생성한 XML 문서 파일과 NDoc을 이용해서 문서파일을 자동으로 생성해보자.

### NDoc 문서화

NDoc(<http://ndoc.sourceforge.net/>)을 받아서 설치했다고 가정하고 NDoc 사용법을 바로 살펴보자.

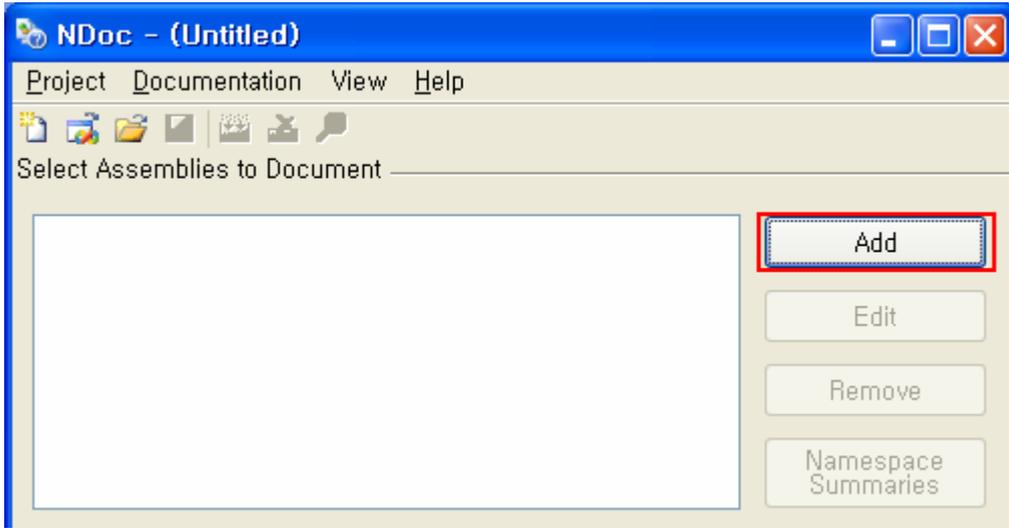


그림 3. NDoc 시작화면

[Add] 버튼을 클릭한다.

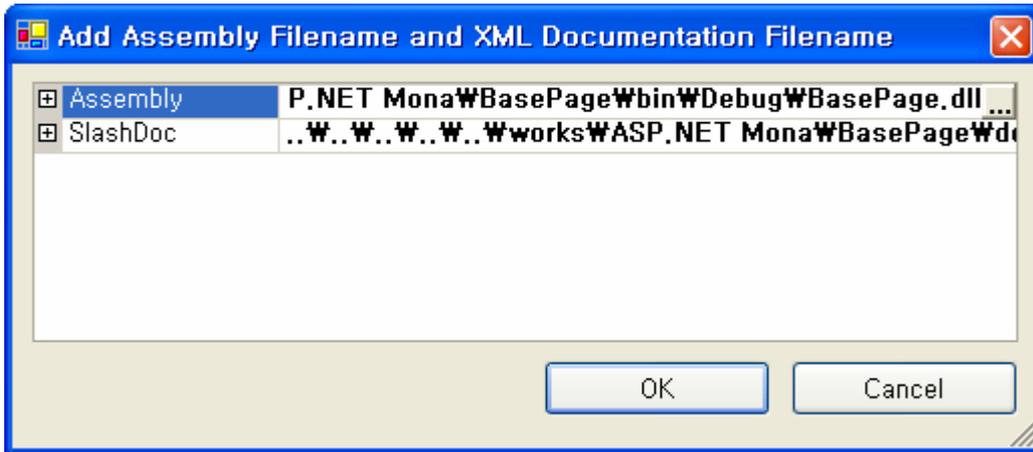


그림 4. 어셈블리와 XML 문서 추가

BasePage.dll과 앞에서 생성한 XML 문서, Mona.Web.UI.BasePage.xml을 추가하고 [OK]를 클릭한다.

다시 NDoc 메인화면으로 돌아오면 화면 중간에 다양한 속성을 설정할 수 있는 부분이 있다. 이 설정을 제대로 해야만 한글이 깨지지 않고 문서화가 된다. NDoc에서 설정할 필요가 있는 속성들만 다음에 나열하였다.

속성	값
HtmlHelpName	Mona.Web.UI.BasePage
SdkDocLanguage	Korean
Title	Mona 도움말
LangID	1042
DocumentInheritedAttribute	False
DocumentInheritedFrameworkMember	False
DocumentInheritedMembers	None
DocumentInternals	True

DocumentPrivates	True
------------------	------

## 표2. NDoc 속성 설정

언어 설정을 위해서는 SdkDocLanguage와 LangID를 변경하면 되며, DocumentInherited로 되어 있는 속성들은 기본값이 True인데 False로 변경하였다. 그렇지 않으면 .NET Framework의 기본 멤버들이 모두 문서화되어 여러분이 작성한 메서드보다 프레임워크에서 상속한 메서드에 대한 설명이 더 많아지게 된다. 실제로, .NET Framework 멤버에 대한 설명은 MSDN을 참고해야 할 부분이지 여러분이 작성할 문서에서 참고할 것은 아니다.

Internals와 Privates는 클래스내의 내부 데이터나 메서드에 대한 참조용으로 문서화를 하기 위해 True로 하였다. 만약, 다른 업체를 위한 외부용 문서라면 이 부분은 False로 설정하기 바란다.

NDoc 프로젝트도 BasePage 프로젝트 디렉터리와 같은 곳에 저장하고, VS.NET에서도 BasePage.ndoc 파일을 추가한다. ndoc 확장자를 NDoc 프로그램과 연결해두면 언제든지 VS.NET에서 NDoc을 불러서 설정할 수 있다. 여기까지 설정을 모두 마쳤으면 NDoc에서 [Documentation] - [Build]를 선택해서 문서를 빌드한다. 최종 컴파일이 끝나면 doc 디렉터리에 생성된 .CHM 도움말을 볼 수 있으며, NDoc에서는 [Documentation] - [View]를 선택해서 볼 수 있다.

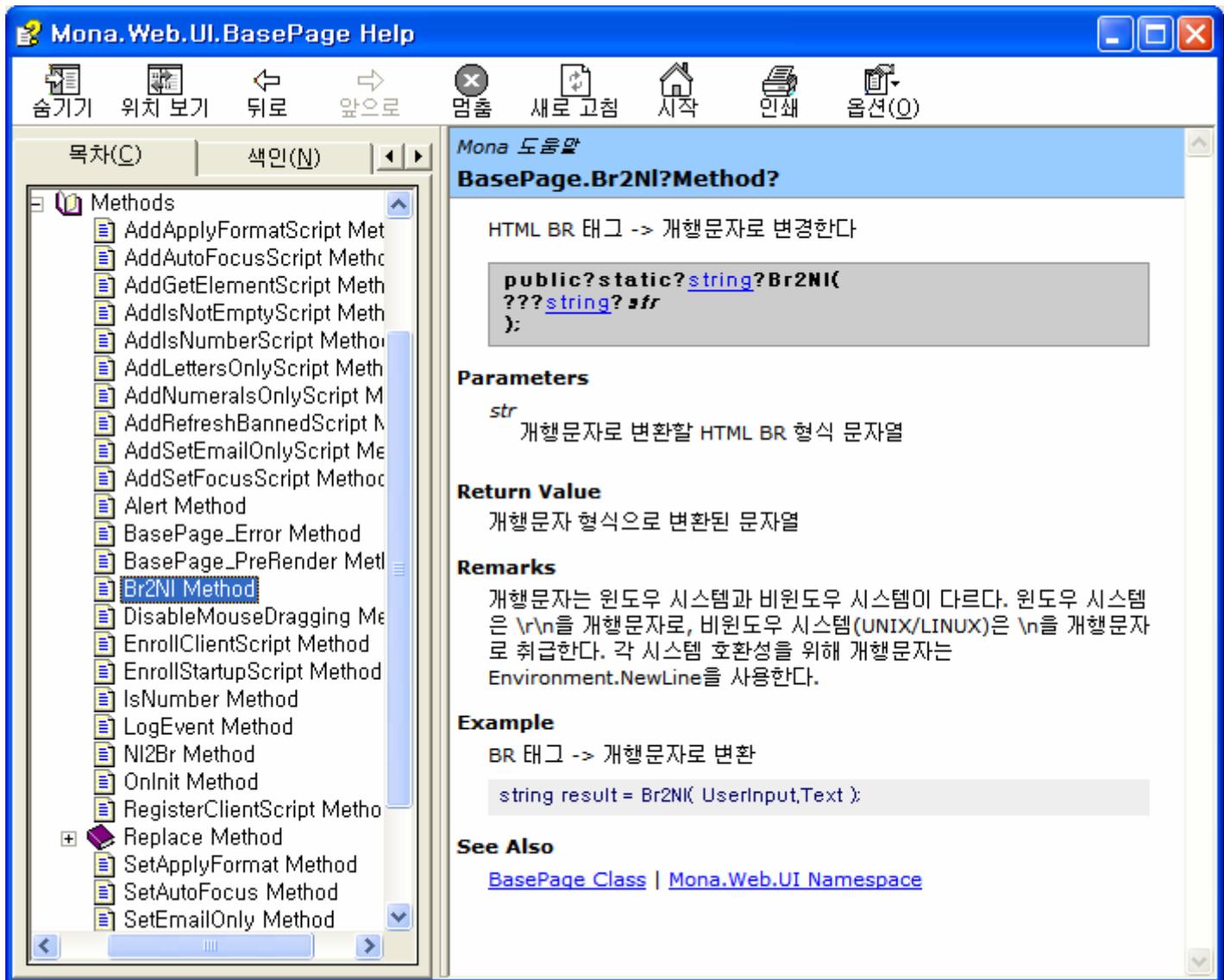


그림5. BasePage 도움말

그림5에서 볼 수 있는 것처럼 Br2NI 메서드에 대한 설명이 잘 정리되어 나타나는 것을 볼 수 있다. 그림1과 그림5를 비교하면 XML 문서태그의 어떤 부분이 어떤식으로 보여지는지 알 수 있다.

NDoc에서 문서화 옵션을 HTML & CHM으로 선택했기 때문에 웹 사이트에 바로 올려놓고 참조할 수 있는 HTML 페이지도 doc 디렉터리에 작성되어 있다. doc\index.html 파일을 열어보면 HTML 형식의 도움말이 어떻게 작성되었는지 살펴볼 수 있다.

NDoc을 이용한 문서화에서 한가지 걸리는 부분이 있는데, 클래스나 메서드 설명에서 공백문자가 ?로 표시되는 부분이다.

이를 위해서는 처음 NDoc을 실행해서 CHM 파일의 원본이 되는 HTML 파일을 만들고, 모든 HTML 파일에서 ?를 공백문자로 변환하고, 다시 NDoc에서 CHM 파일을 작성하는 편법을 이용한다.

## ? 제거하기

윈도우 환경에서는 mreplace와 같은 유틸리티를 이용하여 해결한다. Visual Studio .NET 2003에도 텍스트 파일의 내용을 한 번에 변경하는 기능을 제공하지만 프로젝트에 포함된 파일에 대해서만 사용할 수 있다. 파일안의 텍스트를 변경하기 위해 엄청나게 많은 수의 HTML 파일을 프로젝트에 등록하는 것은 번거롭다. 따라서 윈도우 환경에서는 mreplace와 같은 유틸리티를 쓰거나 필자와 같이 UNIX 계열 운영체제 명령어에 대한 배경지식이 있는 사람은 Cygwin을 이용할 수 있다. 또는 Mono를 사용하는 분들도 sed를 이용해서 간단하게 변경할 수 있다.

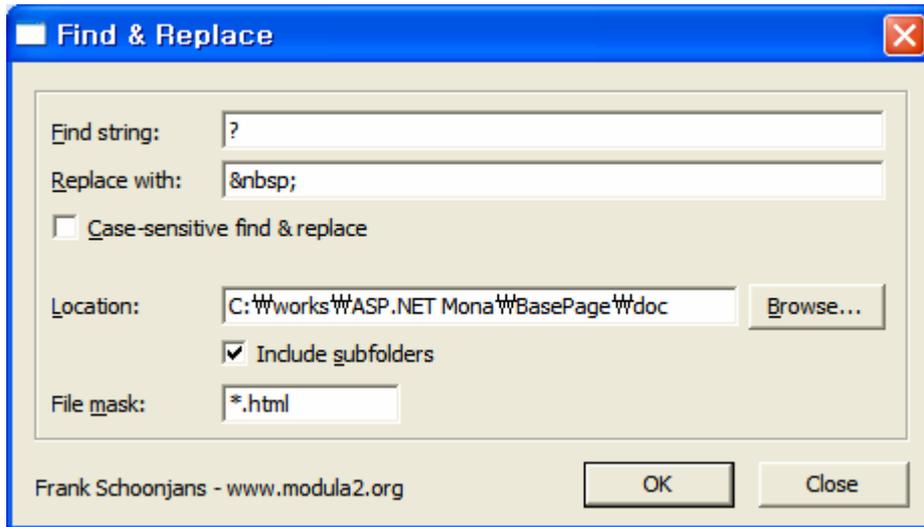


그림 6. mreplace 유틸리티

?를 공백문자로 변경해야 하는데 HTML 형식으로 변경하기 때문에 변경할 문자열에 **&nbsp;**를 입력한다. HTML 파일에 대해서만 적용할 것이므로 File mask에는 **\*.html**을 입력한다. [OK]를 클릭하면 모든 파일의 내용이 변경된다. Cygwin(<http://www.cygwin.com>)이나 UNIX 계열에서 이와 같은 작업을 하려면 다음과 같이 하면 된다.

```
root@ns2: ~/doc# find *.html -print |
> while read file
> do sed -e 's/\?/\&nbsp;/g' "$file" > "$file.new"
> mv "$file.new" "$file"
> done
```

HTML 파일을 정리했으면 doc 디렉터리로 이동한다. 여기에 보면 Mona.Web.UI.BasePage.hhp 파일이 있다. HTML Help Workshop의 프로젝트 파일이며, NDoc에서 자동으로 생성한 것이다. HTML Help Workshop을 설치했다면 이 파일을 클릭해서 열 수 있다.

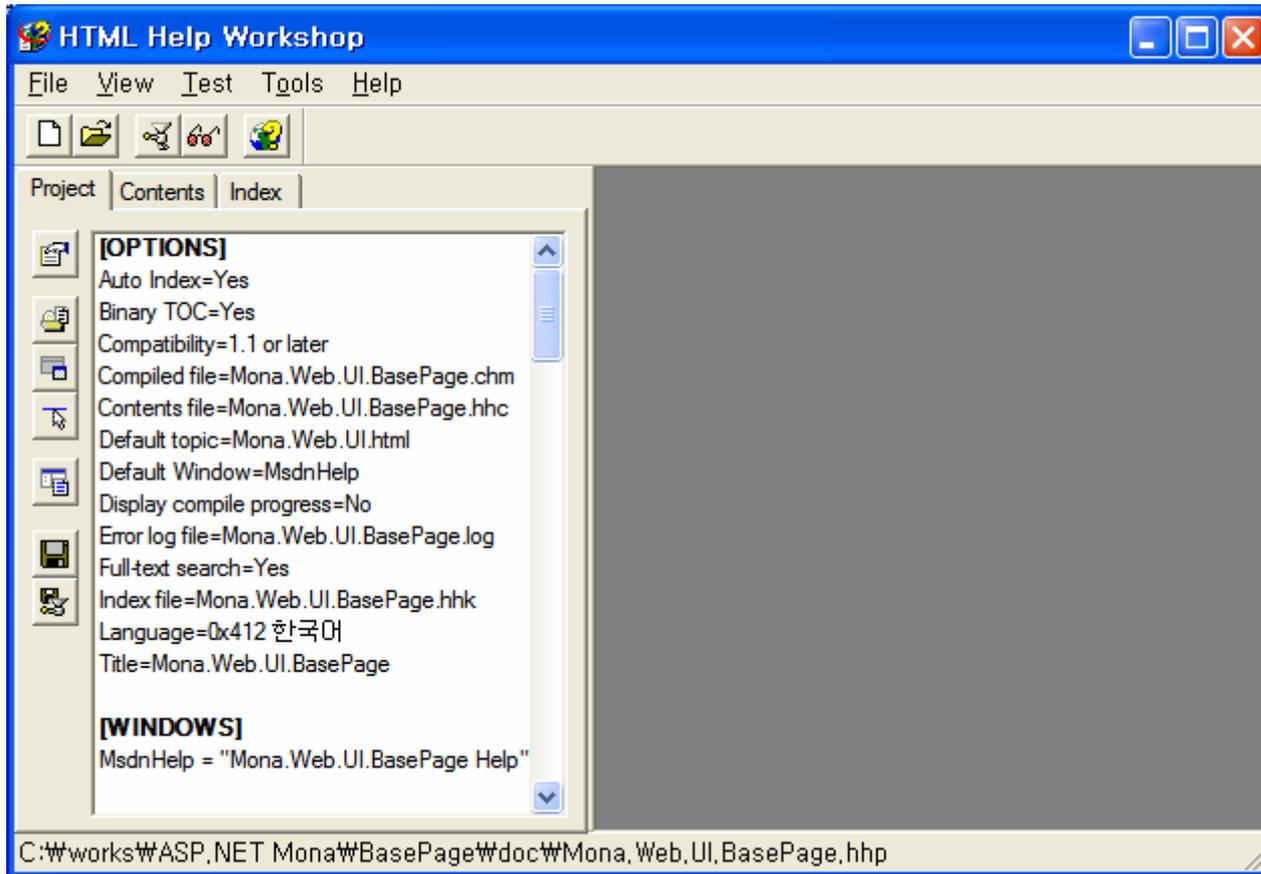


그림 7. HTML Help Workshop 실행화면

[File] - [Compile]을 선택해서 CHM 도움말을 빌드하면 ?등이 사라진 깨끗한 도움말을 볼 수 있다.

### Br2NI과 NI2Br 함수

사용자가 웹에서 여러 줄을 입력할 때 각 줄의 끝에는 개행문자가 붙지만, HTML 웹 페이지를 볼 때 줄 바꿈을 하려면 개행문자가 아니라 BR 태그를 사용해야 한다. BR 태그는 각 줄을 잘라내는 BReak를 뜻한다.

반대로 BR 태그가 들어간 문서를 수정할 때는 BR 태그가 아닌 개행문자로 보여주어야만 화면에 여러줄로 제대로 표시된다.

```
String.Replace( "\r\n", "<BR>" );
```

```
String.Replace( "<BR>", "\r\n" );
```

위와 같은 코드를 사용하면 Br2NI이나 NI2Br 함수를 만들 수 있다고 생각하기 쉽지만 실제로 웹 페이지는 <BR>, <br>, <BR/>, <BR />, <Br>, <br />, <br />과 같이 얼마든지 다양한 형태로 쓰일 수 있다. 따라서 문자열 형태의 치환이 아닌 정규식을 사용하여 제대로 치환하여야 한다.

마찬가지로 윈도우 환경에서는 개행문자를 `\r\n`으로 사용하지만 비윈도우 환경에서는 개행문자를 `\n`으로 사용한다. Linux 환경의 Apache에서 `mod_mono`를 이용하거나 `xsp`를 이용하는 경우와 윈도우 IIS 환경에서 동작하는 경우에도 문제없이 변환하려면 `\r\n`이나 `\n` 어느 한쪽으로만 변환해서는 안된다. 이런 경우에는 `Environment.NewLine` 상수를 사용해야 한다.

### **Replace 함수**

Br2NI과 NI2Br 함수에서는 `Replace`를 사용했는데 이는 `System.Text.RegularExpressions.RegEx` 클래스를 이용해서 만든 함수다.

```
#region Replace Overloading
public static string Replace(string source, char matchPattern,
    string replaceStr)
{
    return (Replace(source, matchPattern.ToString( ), replaceStr, -1, 0));
}

public static string Replace(string source, char matchPattern,
    string replaceStr, int count)
{
    return (Replace(source.ToString( ), matchPattern.ToString( ), replaceStr,
        count, 0));
}

public static string Replace(string source, char matchPattern,
    string replaceStr, int count, int startPos)
{
    return (Replace(source.ToString( ), matchPattern.ToString( ), replaceStr,
        count, startPos));
}

public static string Replace(string source, string matchPattern,
    string replaceStr)
{
    return (Replace(source, matchPattern, replaceStr, -1, 0));
}

public static string Replace(string source, string matchPattern,
    string replaceStr, int count)
{
    return (Replace(source, matchPattern, replaceStr, count, 0));
}

public static string Replace(string source, string matchPattern,
    string replaceStr, int count, int startPos)
{
    Regex RE = new Regex(matchPattern);
    string newString = RE.Replace(source, replaceStr, count, startPos);

    return (newString);
}
#endregion
}
```

그림8. Replace 함수

Replace 함수는 위 코드처럼 모두 5개의 매개변수로 되어 있으며, 마지막 2개는 몇 개나 일치시킬 것인가를 나타내는

count와 정규식 패턴 매치를 시작할 위치를 나타내는 startPos로 되어 있다.

5회까지 걸쳐 작성된 전체 소스는 이곳에서 다운받기 바란다.

## 참고자료

- HTML Help Workshop

(<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/hwMicrosoftHTMLHelpDownloads.asp>)

MS에서 제공하는 무료 도움말 제작도구이며 CHM 형식의 도움말을 제작하거나 해제할 수 있다.

- 유닉스 파워 툴(UXIX POWER TOOLS), 9.9 찾아낸 파일에 대해 원하는 명령을 실행하는 방법  
UNIX 환경에서 파일의 내용을 치환하는 스크립트 작성에 사용했다.

- C# Cookbook, O'Reilly, Recipe 8.5 - Replacing Characters or Words in a String

정규식을 이용한 손쉬운 문자열 치환을 수행하는 Replace 함수는 Recipe 8.5에서 인용했다.

- Ndoc(<http://ndoc.sourceforge.net>)

.NET의 XML 문서화를 자동화하기 위한 도구로 Java의 XML 문서화를 자동화하는 JDoc과 같은 이름을 따르는 문서화 도구다.

- DoxyGen(<http://www.stack.nl/~dimitri/doxygen/>)

C/C++을 위한 문서화 도구로 개발되었고, 현재는 Java, PHP, C#과 같은 다양한 언어들을 지원합니다. C/C++ 문서화 도구가 필요하다면 DoxyGen을 권합니다.

- GhostDoc(<http://www.roland-weigelt.de/ghostdoc/>)

VS.NET 환경에서 개발언어에 관계없이 XML 문서화를 자동화해주는 도구입니다. 특히, 메서드 이름을 잘 지은 경우에는 그것을 토대로 요약까지 자동으로 작성합니다. 메서드 이름 위에서 오른쪽 클릭해서 선택하면 자동으로 필요한 XML 태그를 작성해줍니다.

- mreplace(<http://www.modula2.org/projects/mreplace.php>)

Modula-2용으로 만든 유틸리티로 여기서는 파일 텍스트의 내용을 일괄적으로 변경하기 위해 사용했습니다. 윈도우 사용자가 사용하기에 가장 간단하고 편리합니다. 마지막에 사용한 설정을 기억하므로 프로젝트 작업시에 반복하는 일은 많지 않습니다.