3. Working in the editor

본 문서는 볼랜드사에서 제공하는 JBuilder 9 버전의 문서중
"Introducing JBuilder" 의 Chapter 5. Working in the editor를
번역한 것입니다.

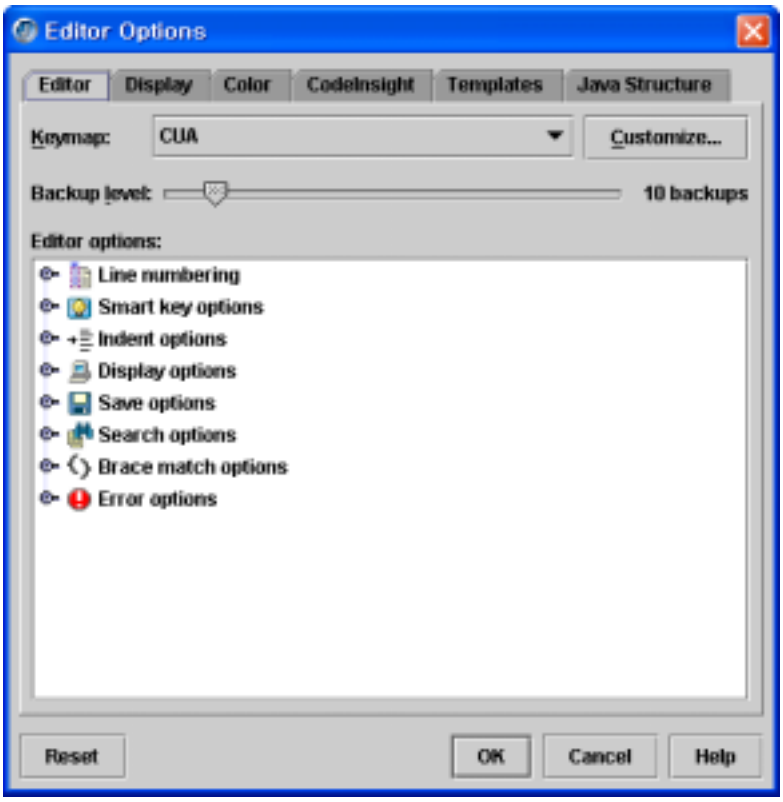
이 문서는 번역시 오류, 오타 등이 있을 수 있으며, 이 문서를 사용함으로서
발생되는 문제에 대해서 번역자는 책임이 없으며, JAVA를 배우고 개발 툴로
JBuilder를 선택한 초보자를 위한 것이며, 상업적인 목적이 아닌 한 자유롭게
활용및 배포할 수 있습니다.

본 문서에 대한 문의 사항은 아래로 연락바랍니다.

- Http://wowstone.lin4u.com
- wowstone@empal.com

3.1 Editor 개요

JBuilder는 code작성을 빠르고 쉽게 작업할 수 있는 많은 기능을 가진 훌륭한 code editor 이다. editor에서 파일을 열기 위해서는 project pane에서 텍스트 기반 파일을 더블 클릭 하거나, 파일을 선택하고 Enter키를 누른다. editor의 하단에 status bar에는 텍스트기반 파일의 파일이름, 커서위치, 삽입모드 등을 표시해준다. editor는 brace matching, keyboard shortcuts, syntax highlighting, customizable editor keymappings, coding shortcuts (Javadoc coding shortcuts포함), searching, printing 뿐만 아니라 완전한 사용자 정의 editor 와 같은 생산성을 위한 기능을 제공해 준다. 이와같은 많은 기능들은 메뉴에서 [Tools | Editor Options]를 선택하여 설정할 수 있다.



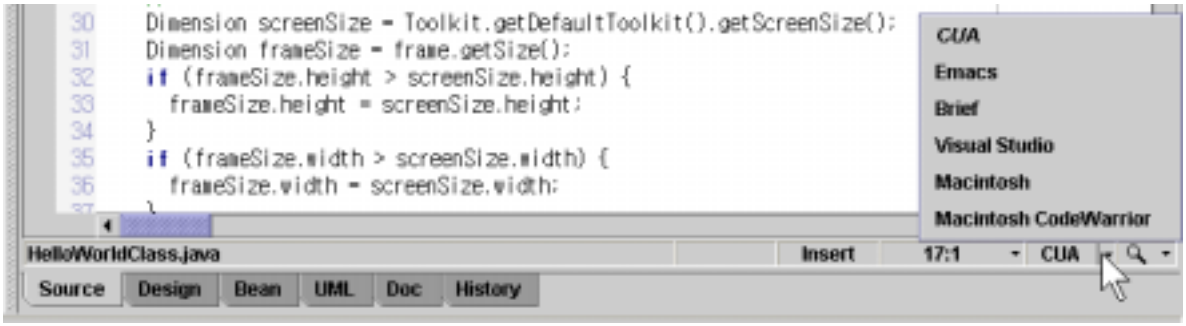
이와같은 많은 특징들은 editor의 context menu에서 사용가능 하다. 이 기능들은 사용자가 현재 사용하고 있는 JBuilder edition, project 설정값, menu가 표시될 당시 editor안의 커서 위치에 따라 다양하다. context menu를 표시하기 위해서는 editor안 커서가 있는 곳에서 마우스 오른쪽 버튼을 클릭하거나, editor를 클릭하고 [Shift+F10]키를 누른다.

3.2 Selecting a keymapping for the editor

키보드 단축키는 특정 작업을 빠르고 쉽게 작업할 수 있도록 해준다. 다른 editor map 은 같은 동작이라고 다른 keystroke 를 가진다. 이것은 아무리 잘 사용해도 불편하고, 최악의 경우에는 위험을 초래한다. 왜냐하면 서로다른 프로그래머는 각기 다른 습관과 선호도를 가지며, JBuilder는 몇개의 editor 에뮬레이션(emulation)을 제공한다. 사용자는 BRIEF, CUA, Emacs, Macintosh, Macintosh CodeWarrior, Visual Studio 와 호환되는 것들 중에 가장 편

한 것을 선택할 수 있다. JBuilder가 Windows, Linux, 또는 Solaris 등에 처음 설치되면 CUA keymapping 이 적용될 것이다. editor에서 다른 keymapping scheme를 변경시키기 위해서는 다음 단계를 따른다.

1. status bar 오른쪽 끝에 있는 돋보기 모양(magnifying glass)왼쪽 바로 옆에 있는 drop-down 화살표키를 클릭한다.
2. 선택하고자 하는 keymapping scheme 를 선택한다.



keymapping scheme를 변경하기위해 메뉴에서 [IDE Options] 나 [Editor Options]를 사용할 수도 있다.

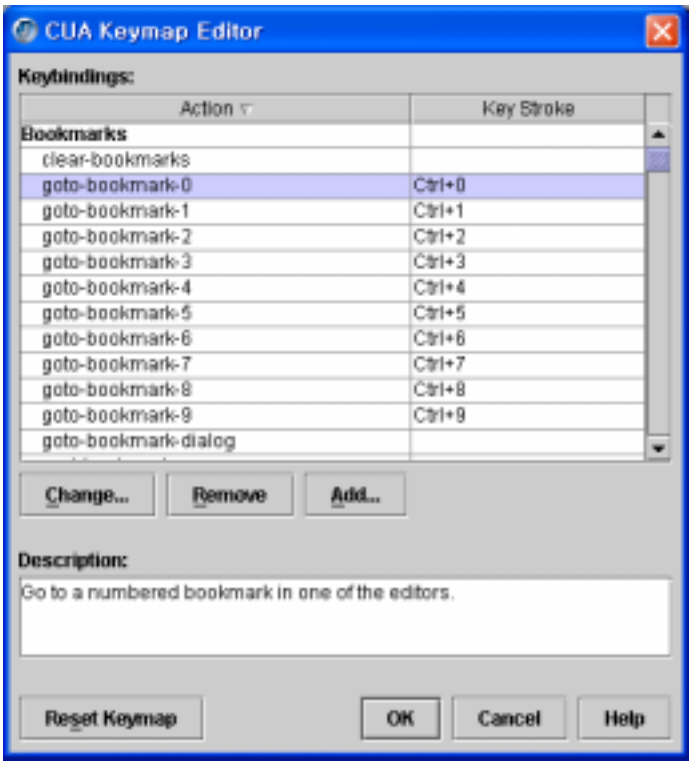
1. 메뉴에서 [Tools|IDE Options]를 선택하고 [Browse] 페이지를 선택하거나, [Tools|Editor Options]를 선택하고 [Editor] 페이지를 선택한다.
2. [keymap]항목을 클릭하고, drop-down 메뉴에서 원하는 editor 에뮬레이션을 선택한다.

이렇게 하면 새로운 keymap 에뮬레이션이 즉시 활성화된다.

3.2.1 Customizing keymaps

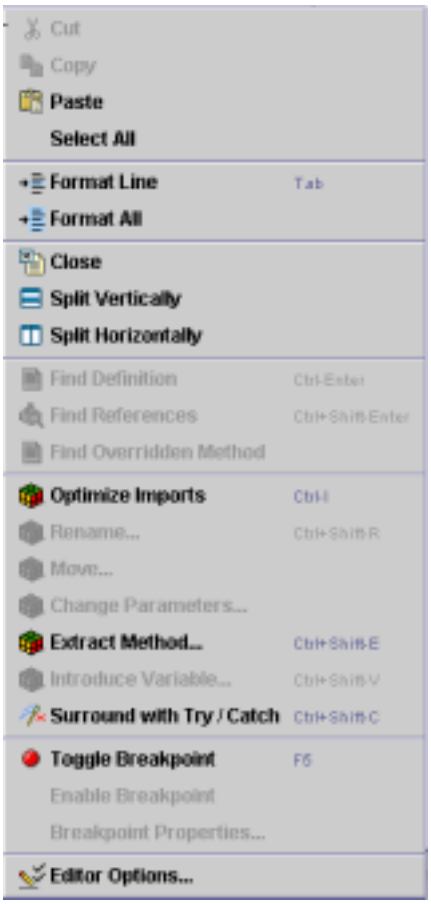
사용자는 Keymap Editor를 사용하여 기 선택되어진 keymapping를 재 정의할 수 있다. keymap Editor를 액세스하기 위해 다음 단계를 따른다.

1. 메뉴에서 [Tools|IDE Options]를 선택하고 [Browser] 페이지를 선택하거나, [Tools|Editor Options]를 선택하고 [Editor] 페이지를 선택한다.
2. [Keymap] 항목 옆에 있는 [Customize] 버튼을 클릭한다.
3. 사용자가 keymapping에 변경, 삭제, 추가하고자 하는 명령어를 클릭한다.
4. 원하는 작업을 수행하기 위하여 Change, Remove, Add 버튼을 클릭한다.
5. 정렬순서를 변경하기 위해 table header을 클릭한다.



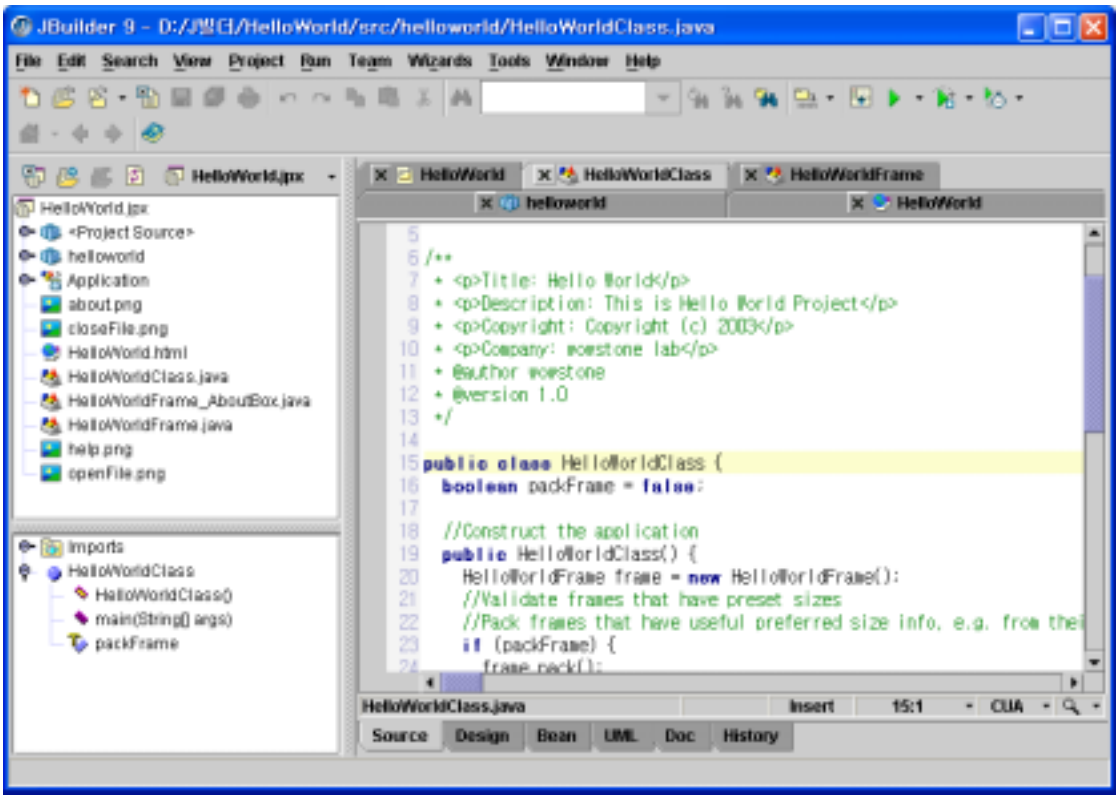
3.3 Splitting the sourece view

editor는 파일 소스 뷰를 2개 이상의 수평 또는 수직으로 분할 할 수 있도록 한다. 이와같은 설정은 개개의 파일에 적용되며, content pane안에 열려있는 모든 파일에는 적용되지 않는다. 사용자는 모든 열려있는 파일에 서로다른 설정을 할 수 있다. 뷰모드를 분할하기위해서 Source pane에서 마우스 오른쪽 버튼을 클릭하고 [Split Vertically]나 [Split Horizontally] 메뉴를 클릭한다. 하나의 pane에서 분리된 뷰모드를 원래대로 복귀하기 위해서는 각 pane에서 마우스 오른쪽 버튼을 클릭하고 [Close View] 메뉴를 선택한다.

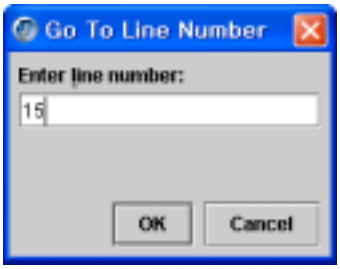


3.4 Line numbers

editor는 왼쪽여백에 line number를 표시할 수 있는 옵션을 가지고 있다. 이기능은 [Tools | Editor Options]메뉴를 선택했을때 나타나는 Editor Options 대화상자의 [Editor] 페이지의 [Line Numbering] 옵션을 사용하여 표시여부를 결정할 수 있다.

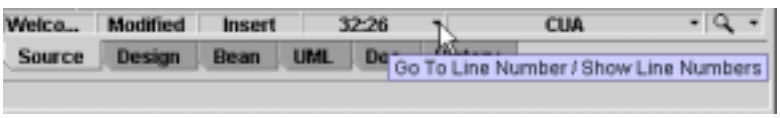


메뉴에서 [Search | Go to Line] 을 선택한 후 line number를 입력하여 즉시 커서를 지정한 line number로 이동시키는 것도 가능하다. 이와같은 기능은 status bar위에서 단축키를 사용해도 가능하다.



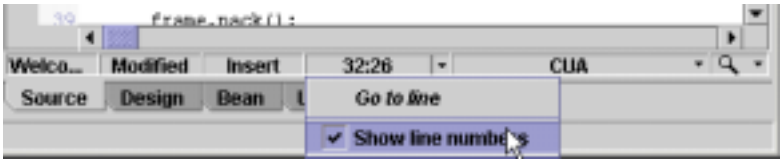
3.4.1 Displaying line numbers

사용자는 editor 아래 상태 표시 영역으로부터 line numbering를 빠르게 조정할 수 있다. 커서 위치를 나타내 주는 line number와 column number는 항상 status bar에 표시된다. 이 표시 바로 오른쪽에 drop-down 화살표가 있다.



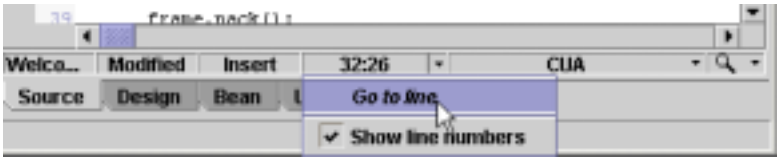
line number를 보이기 위해 drop-down 화살표를 클릭하고, [Show Line Numbers] 메뉴를

클릭한다. 이것을 해제하기 위해 다시 drop-down 화살표를 클릭하고 [Show Line Numbers]에 있는 체크 표시를 삭제한다.



3.4.2 Going to a specific line

특정 line number로 이동하기 위해서는 자체적으로 표시되는 line이나 column number를 클릭하거나, 또는 그것 바로 옆에있는 drop-down 화살표를 클릭하여, [Go To Line] 메뉴를 선택한 후 [Go To Line Number] 대화상자에 line number를 입력한다.



[Go To Line] 대화상자를 표시하기 위해 [CTRL+G] 키를 대신 누를수도 있다.

3.5 Selecting text

editor 안에서 text를 선택하기 위한 여러가지 방법이 있다. 마우스를 사용하여 text의 일부분을 선택할 수 있으며, 마우스 커서를 text 시작위치에 놓은 다음 Shift 키를 누른채로 선택하고자 하는 text의 위치를 클릭하여 text 일부분을 선택할 수도 있다. 또한 Ctrl+Shift 키와 화살표키를 사용하여 빠르게 단어와 행을 강조(highlight) 시킬 수 있다. 이외에도 왼쪽 여백에 표시되는 line number를 사용하여 빠르게 전체 행을 선택하기 위한 단축키가 있다.

먼저 전체 행을 선택하기 위해서는 line number를 클릭한다. 스크롤 할 필요 없이 블럭 단위로 행을 선택하기 위해서는 먼저 왼쪽 여백의 첫번째 line number를 클릭하고, Shift 키를 누른 상태로 마지막 line number를 클릭한다. 전체 파일에서 모든 행을 빠르게 선택하기 위해서는 line number 가 있는 왼쪽 여백 아무곳에다 커서를 위치시키고, 마우스를 사용하여 line number위에서 CTRL키를 누른상태에서 클릭한다.

3.6 Dragging and dropping text

editor 안에서 선택된 text 를 drag and drop 할 수 있다. 원하는 text를 강조시키고 마우스를 사용해서 새로운 위치로 drag 한다. 만일 여러분이 text의 전체행을 이동하고, [Tools|Editor Option] 에서 [Smart Paste] 옵션이 선택되어 있다면, 해당 text를 새로운 행에 drop 하면 자동적으로 indent 가 적용된다.

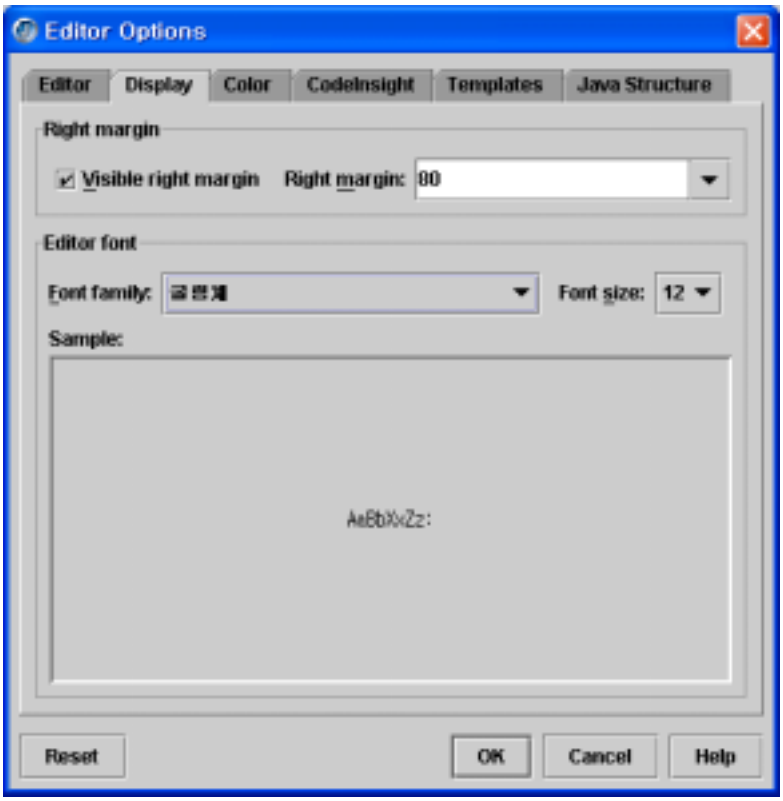
3.7 Resizing the font used in the editor

사용자는 빠르게 code font를 수정할 수 있다. 코드의 폰트크기를 확대하기 위해서 editor의

status bar에 위치하고 있는 돋보기 아이콘을 클릭한다. 사용자가 이것을 클릭할때마다 계속 text가 커진다.

돋보기 아이콘 옆에 있는 pull-down 화살표를 클릭하거나 메뉴에서 [Zoom in] 를 선택하여 폰트 크기를 증가시킬 수 있다. 또한 [Zoom out]를 이용하면 폰트 크기가 작아지며, 기본 크기로 설정하기 위해서는 [Normal] 메뉴를 선택한다.

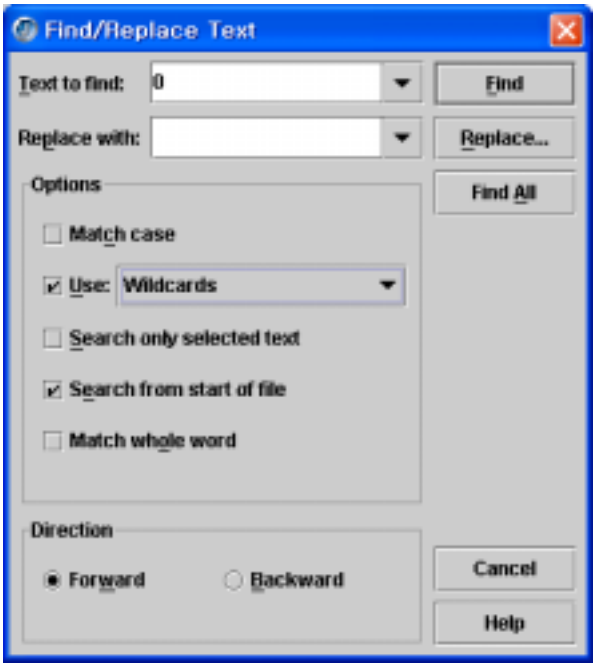
사용자는 메뉴에서 [Tools|Editor Options]를 선택하여 폰트와 크기를 함께 변경할 수 있다. 이를 위하여 [Display] 탭을 선택하고, 변경사항을 수정하고, [Editor font] 옵션을 사용한다. 더 많은 정보를 얻기위해서는 아래 대화상자에서 [Help] 버튼을 클릭한다.



editor를 포함하여 전체 IDE에서 사용된 폰트 크기변경에 대한 정보를 위하여 [Tools|IDE Options] 메뉴를 선택하고, 이 대화상자의 [Browser]페이지에서 폰트크기 증가 옵션을 보기 위해 [Help] 버튼을 클릭한다.

3.8 Finding text in the editor

editor는 지정된 text를 검색하고 바꾸기 위한 많은 방법을 제공해 주고 있다. 이러한 명령어는 main toolbar의 아이콘 뿐만아니라 [Search] 메뉴에 위치하고 있다. 검색옵션을 수정하기 위해서는 메뉴에서 [Tools|Editor]의 Editor 페이지를 수정한다.



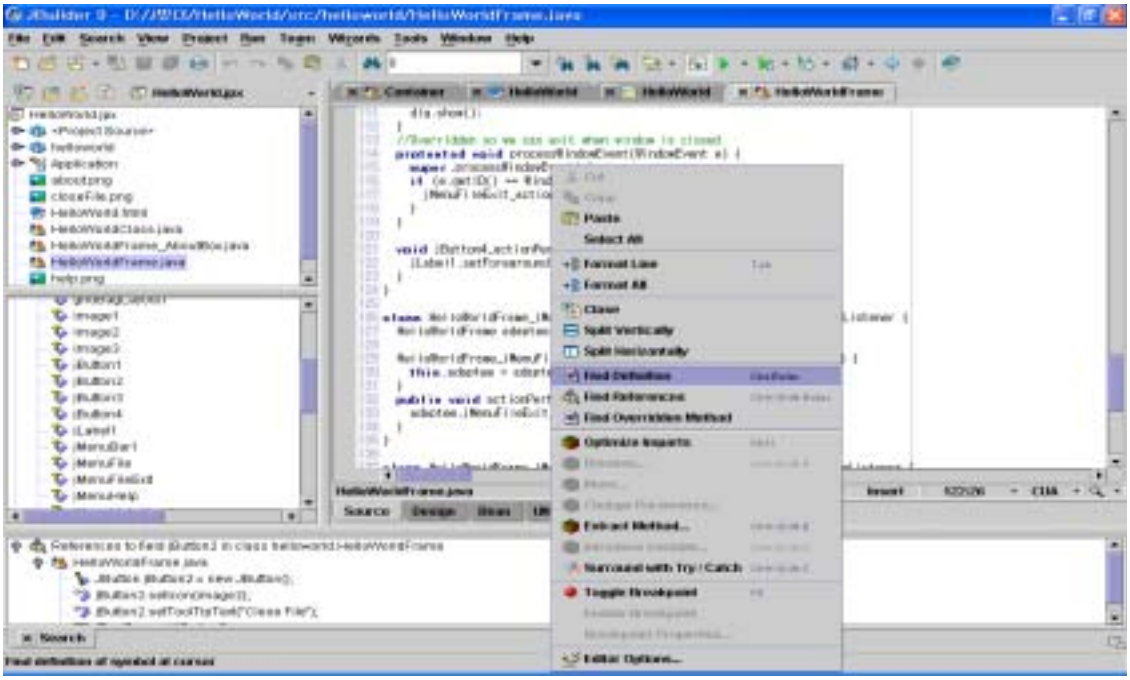
Find/Replace 대화상자는 text 교체를 위한 옵션을 제공해 준다.

명령어	의미
Search|Find	text 검색
Search|Find in Path	지정된 경로에서 모든 파일을 대상으로 text를 검색
Search|Replace	text를 검색하고 새로운 문자열로 치환
Search|Search Again	text 검색을 다시 실행
Search|Incremental Search	검색할 text를 입력할때마다 검색을 시행
Search|Go To Line	지정 행 으로 이동
Search|Find Classes	class, interface, or package 등을 검색
Search|Find Definition	variable, method, class, or interface의 정의을 검색
Search|Find References	variable, method, class, or interface의 인스턴스(instances)나 사용법을 검색
View|History	history list 에서 지정된 item를 검색
View|Back	history list 에서 이전 item를 검색
View|Forward	history list 에서 다음 item를 검색

이 대화상자는 대·소문자 구분(case-sensitive), 와일드카드 사용(wildcards). 정규표현식(regular expressions), 확장된 search/replace 작업 정의 등과 옵션을 포함하고 있다.

3.9 Finding a symbol's definition

[Search] 메뉴나 editor의 context menu에서 사용할 수 있는 [Find Definition] 명령어는 사용자가 심볼(symbol)사용에서 정의(definition)부분을 검색할 수 있도록 해준다. 심볼정의 부분을 검색하기 전에 사용자는 project를 컴파일 해야 하며, 정의 부분을 포함하고 있는 클래스는 import로 경로가 지정되어 있어야 한다.



어떻게 심볼이 정의 되어 있는지를 알아보기 위해 보고자 하는 심볼에 커서를 위치시키고 마우스 오른쪽 버튼을 클릭하고 [Find Definition]을 클릭한다. 그러면 심볼이 정의된 소스파일이 열리고, 커서가 심볼정의 부분에 위치한다.

3.10 Finding references to a symbol

[Search] 메뉴나 editor와 structure pane의 context menu에서 사용할 수 있는 [Find References] 명령어는 주어진 심볼(symbol)을 사용하는 모든 소스파일을 보여준다. 심볼에 대한 모든 참조를 검색하기 위해 심볼위에 마우스 커서를 위치시키고 마우스 오른쪽 버튼을 클릭한 후 [Find Reference] 메뉴를 선택한다. 참조가 존재하는 곳의 소스파일을 발견하기 위해서는 항목 node를 펼치고 message pane 안에 있는 체계트리(hierarchy tree)를 순회(travers)한다. 소스파일을 열고 그 파일에 커서가 해당 파일의 참조 위에 직접 위치하도록 참조를 더블클릭한다.

3.11 Finding the overridden method

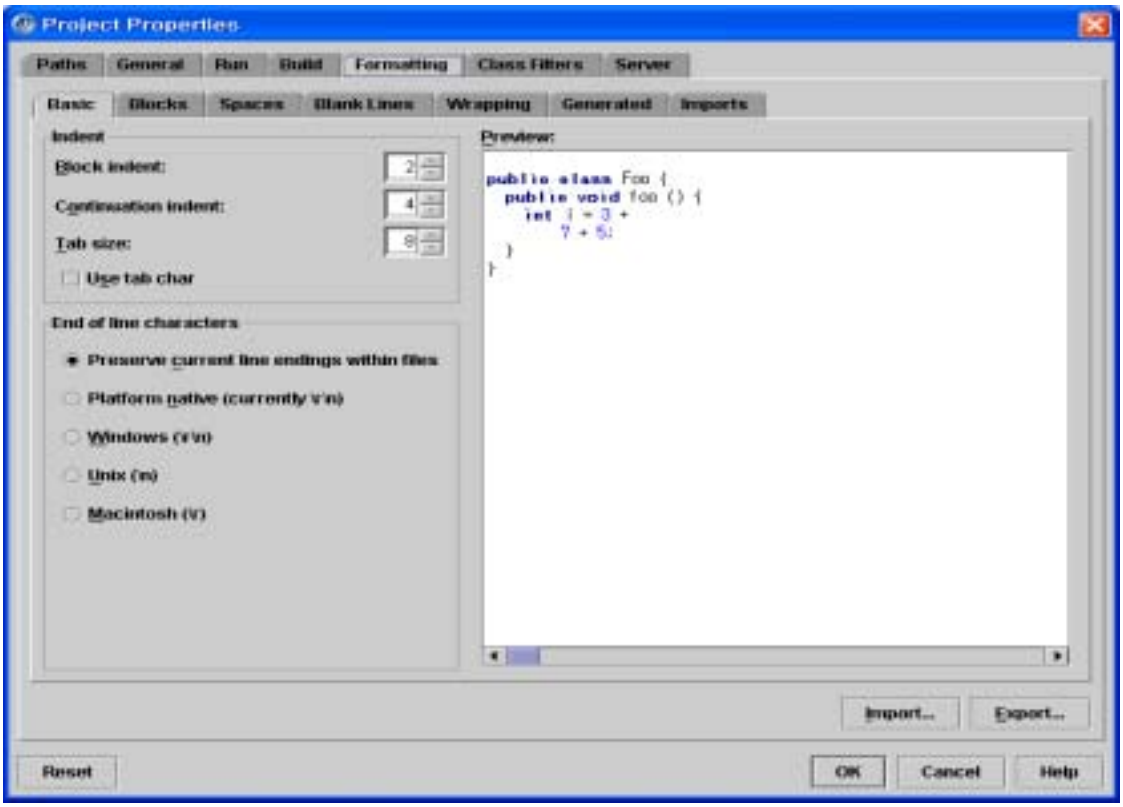
editor 또는 structure pane의 context menu에서 사용할 수 있는 [Find Overridden Method] 명령어를 이용하여 메소드(method)의 상위 클래스(superclass)을 검색할 수 있다. 이를 위하여 먼저 project 를 컴파일 하고, editor나 structure pane에서 해당 메소드에 커서를 위치시키고 마우스 오른쪽 버튼을 클릭한후 [Find Overridden Method] 메뉴를 선택한다. [Find Overridden Method] 명령어는 정확하게 상위 클래스로 이동시켜주며, 오버라이딩

(overridden)된 메소드를 강조(highlight)시켜준다. 사용자는 오버라이딩 된 모든 상위클래스의 메소드를 찾기 위한 상위클래스 연결(chain)을 검색할 수 있다. [Find Overridden Method] 명령어는 오버라이드된 상위 클래스 메소드가 없으면 비 활성화된다.

주의>
여기에서 사요된 Finding a symbol's definition, Finding references to a symbol, Finding the overridden method 기능은 JBuilder Developer와 Enterprise 버전에서만 제공되는 기능이다.

3.12 Formatting your code

사용자는 사용자가 좋아하는 형식으로 지정하거나 소스코드를 자동으로 형식(format)화 할 수 있다. 이 설정값을 이용하기 위해서 메뉴에서 [Project |Project Properties]를 선택하고 [Formatting] tab를 클릭한다.

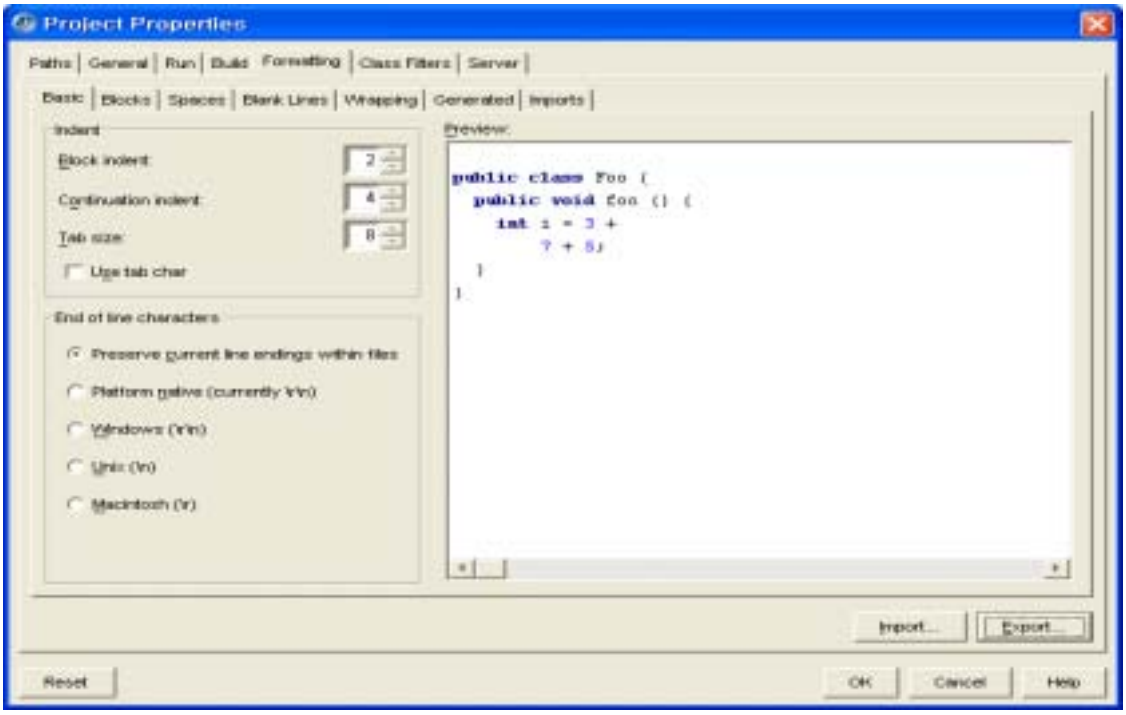


이 페이지에 있는 tab은 여백(Indenting), 탭 크기(Tab size), 행끝문자(End of line characters), 다중행 인자(Multi-line parameters), 연속여백(Continuation indent), 중괄호(Braces), 스페이스(Spaces), 공배행(Blank lines), 텍스트 자동정렬(Text wrapping), 이벤트핸들링(Event handling), 문장삽입과 순서(Import statements and their order) 등을 지정함으로써 사용자 코드 형식을 재정의 할 수 있다. 각각의 페이지는 사용자가 선택한 것과 똑같은 결과를 표시해주는 미로보기 윈도우가 있다.

3.12.1 Applying formatting to your code

사용자 코드에 형식을 적용시키기 위해 editor 안에 파일을 열고, 파일에서 Edit|Format를

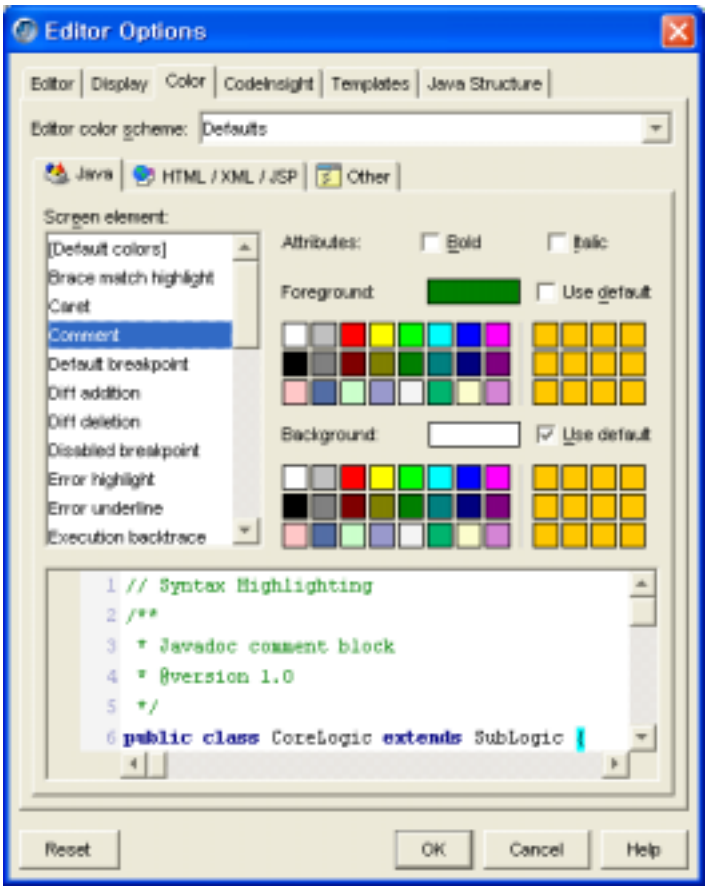
선택하거나 [TAB]키를 누른다. 이때 중요한 것은 형식속성(Formatting properties)은 project 안의 모든 파일에 적용된다.



사용자는 Project 대화상자에서 [Formatting]아래에 있는 [Import]와 [Export] 버튼을 이용하여 사용자가 설정한 형식은 별도 파일로 저장하거나 다시 불러올 수 있다.

3.12.2 Applying formatting to your code

editor에서 사용자 코드의 화면 요소(screen elements of your code)를 재정의하기 위해 메뉴에서 [Tools|Editor Options]를 선택한 후 표시되는 대화상자에서 [Color]페이지를 선택한다. 사용자는 여기에서 Java, HTML/XML/JSP, Clickable links and link highlights, Preprocessor, Standard error, Stand input and output과 같은 특별한 화면 요소를 위하여 구문강조(syntax highlighting)를 지정할 수 있다. 화면요소를 재 정의하기 위해 [Tools|Editor Options|Color]를 선택한다. 사용자는 화면요소의 항목을 선택하거나, sample box안에 있는 요소를 클릭하여, 화면요소를 재 정의 할 수 있다.

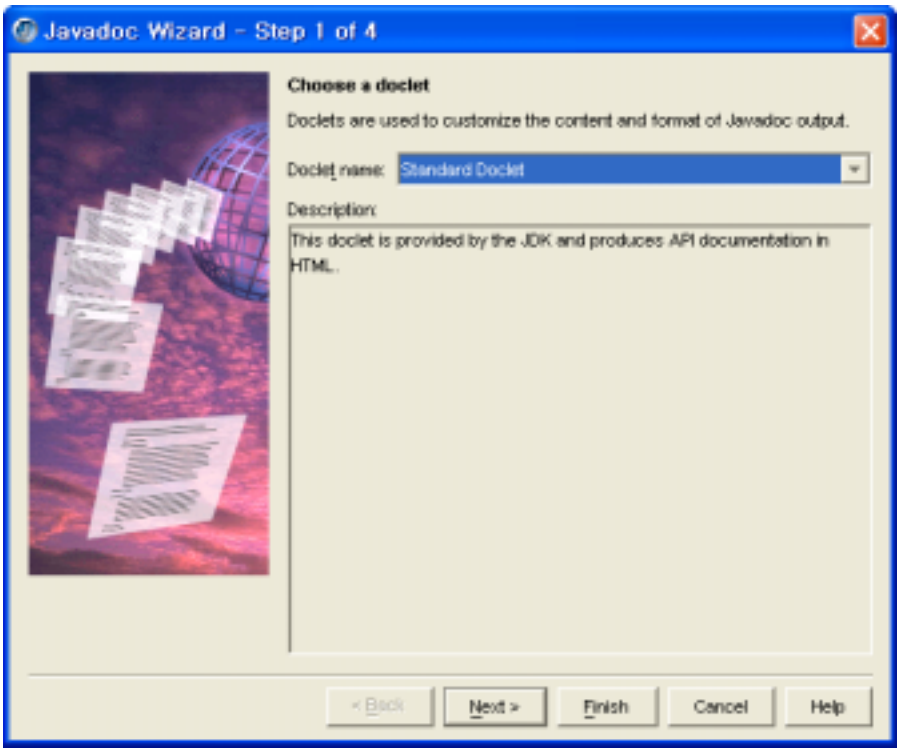


3.12.3 Wrapping curly braces around code blocks

JBuilder는 "{}" 에 대한 자동완성 기능을 제공한다. "{}" 안에서 코드 블록을 빠르게 완성하기 위해서, 시작위치에 "{" (Open curly brace)를 놓고, 그 블록 끝으로 이동한 다음 [Enter] 키를 누른다.

3.13 Javadoc shortcuts in the editor

Javadoc 은 API 스스파일 안에 사용자가 입력한 주석(comment)으로부터 HTML 문서(Documentation)을 생성해 내는 Sun Microsystem사의 유틸리티이다. 이 주석은 Javadoc표준에 의하여 형식(Formatted)화 되어야만 한다. JBuilder는 Javadoc 생성을 위한 속성을 설정하기 위하여 메뉴에서 [Wizards|Javadoc wizard]를 선택한다. 그러면 사용자가 project를 생성할 때 Javadoc 이 자동적으로 생성된다.

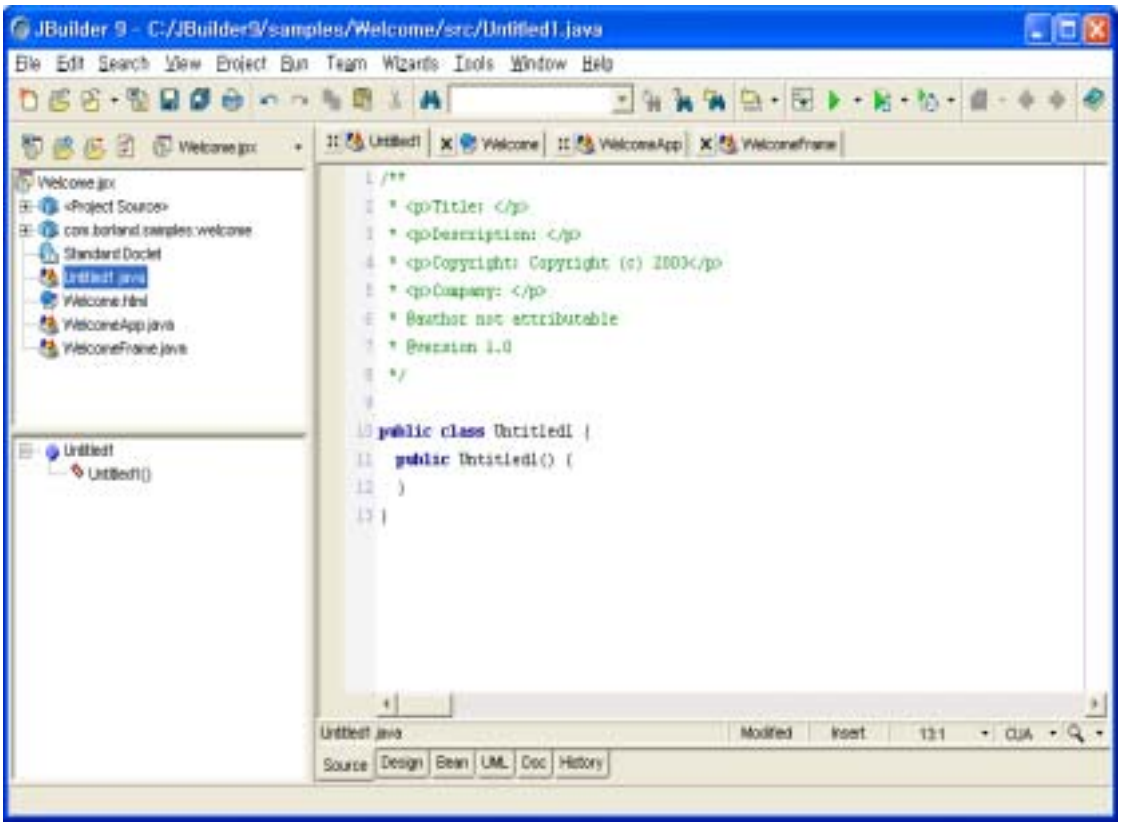


Javadoc 주석을 쉽게 코딩하기 위해서 JBuilder Editor 는 사용자가 /**을 입력하고 Enter 키를 눌렀을 때 활성화되는 Javadoc 주석 템플릿(Template)를 포함하고 있다. 템플릿은 자동적으로 주석 심볼인 */를 Javadoc 끝에 추가한다. 만일 커서가 class, interface, method 표시 전에 즉시 위치하면 템플릿은 가장 근접한 Javadoc tag를 포함하기 위해 확장된다.

Javadoc 주석 블록에 빠르게 추가하기 위해서는 class, interface, method 표시 전에 원하는 들여쓰기(Indentation)수준에 커서를 위치시킨후 /**를 입력한후 Enter key를 친다. editor 는 자동적으로 Javadoc 끝에 주석심볼을 추가시키고 커서위치는 주석의 2번째 라인에 위치한다. 필요하다면 적절한 태그를 추가한다.

예를 들어 class 소스파일 안에서 import 문장 바로앞에 **/ 입력하면, 아래와 같은 주석 블록을 자동으로 추가해 준다.

```
/**
 * <p>Title: </p>
 * <p>Description: </p>
 * <p>Copyright: Copyright (c) 2003</p>
 * <p>Company: </p>
 * @author not attributable
 * @version 1.0
 */
```

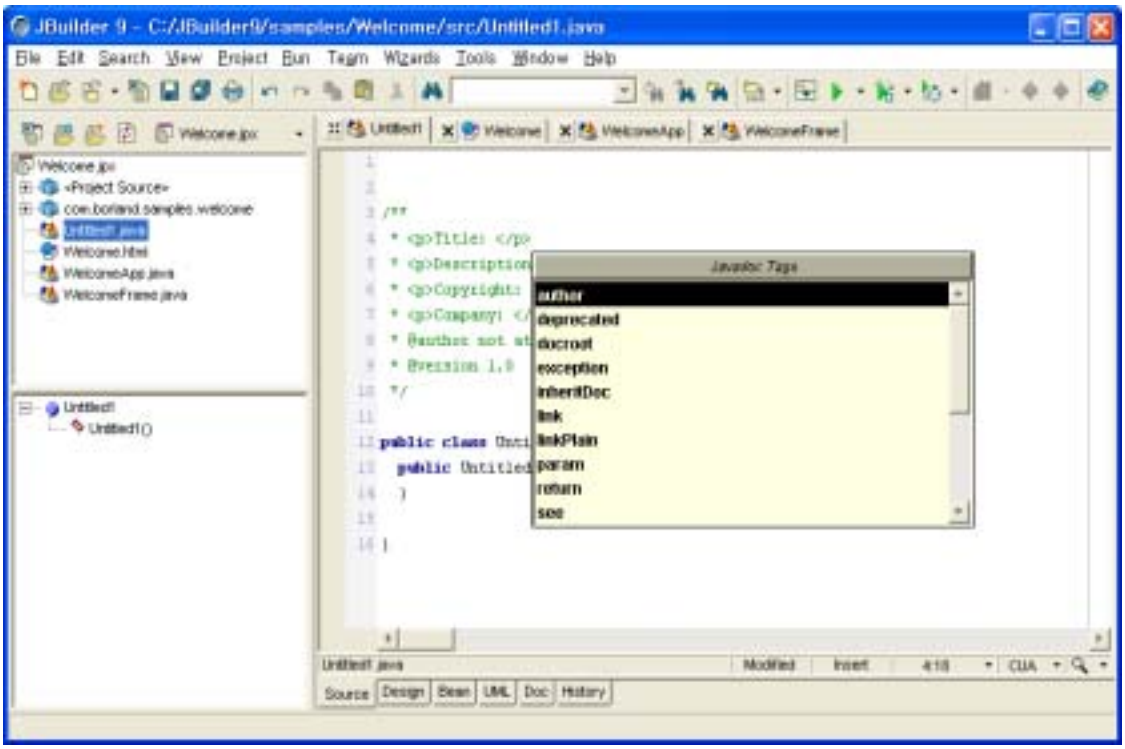


다음과 같은 메소드 형식 앞에서 /**을 입력하면 Javadoc은 다음과 같은 Javadoc commnet를 생성해 준다. 이것은 processWindowEvent()함수가 하나의 파라메터를 가지므로 한 개의 파라메터 값을 표시해 준다.

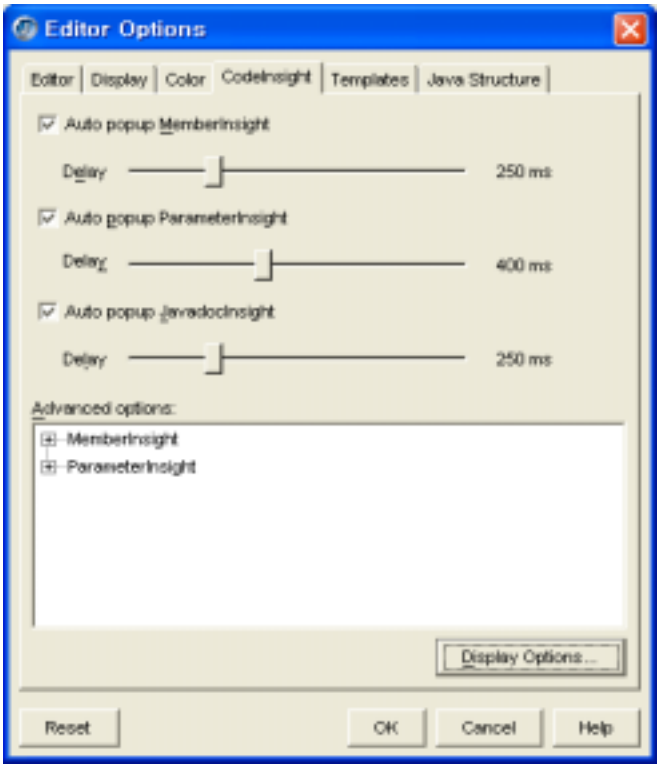
```
/**
 *
 * @param e
 */
protected void processWindowEvent(WindowEvent e) {
  super.processWindowEvent(e);
  if (e.getID() == WindowEvent.WINDOW_CLOSING) {
    System.exit(0);
  }
}
```

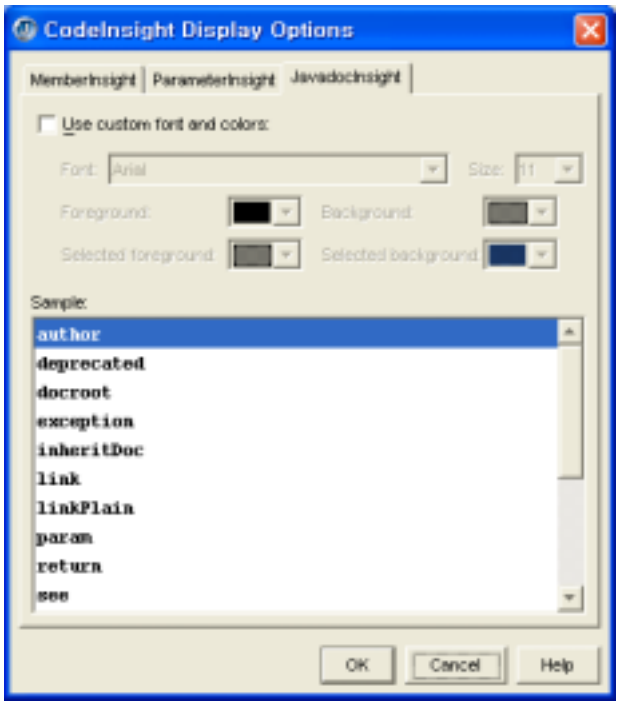
3.13.1 Using Javadocinsight in the editor

사용자는 사용자 Javadoc tag를 Javadoc에 추가할 수 있고, Javadocinsight를 사용하여 Javadoc tag를 Javadoc comment block에 넣을 수 있다. 또한 normal line 과 block comments 와 다른 color-code Javadoc를 사용할 수 있다. JavadocInsight 창을 액세스하고 Javadoc tab 항목을 보기 위해 커서를 Javadoc comment에 위치시킨후 [CTRL+Space]키나 "@"버튼을 클릭한다.

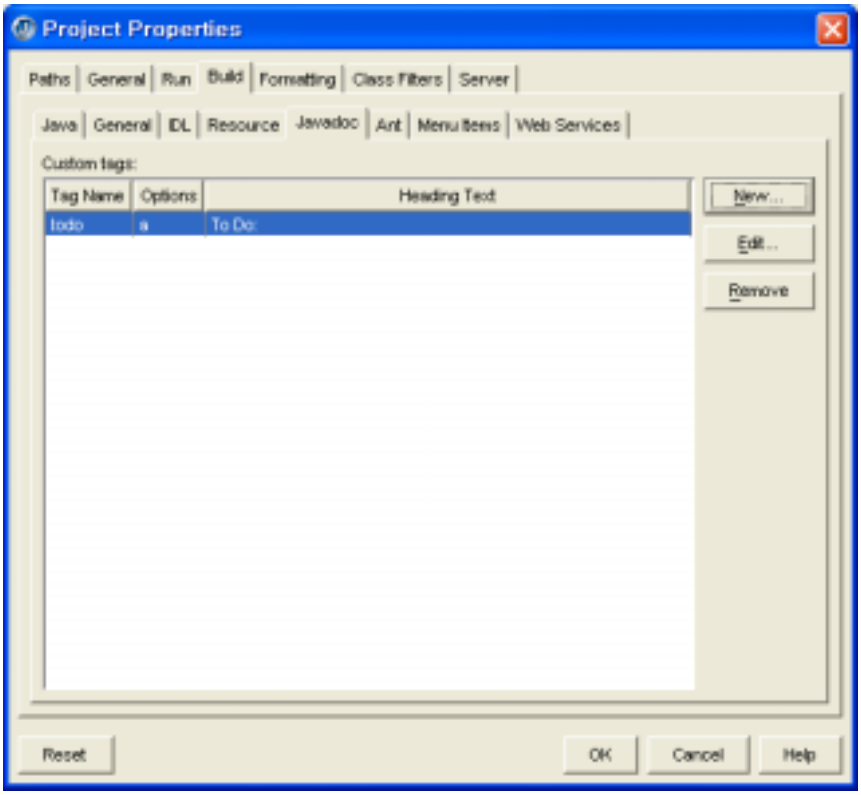


font와 color 옵션을 보기 위한 JavadocInsight page을 액세스하기 위하여 메뉴에서 [Tools | Editor Options | CodeInsight | Display Option | JavadocInsight]를 선택한다.

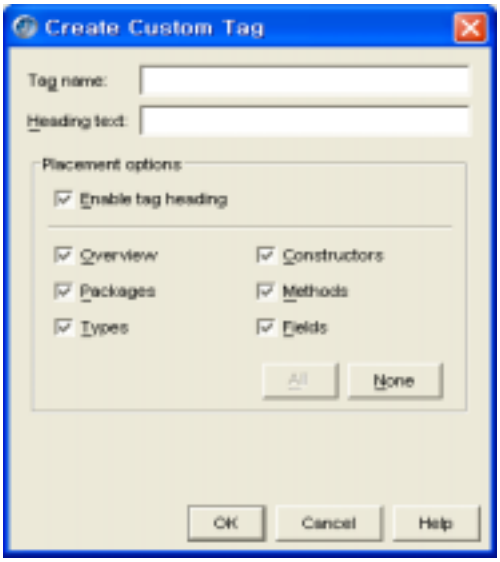




사용자 Javadoc tag를 생성,삭제,수정을 위한 Project 속성 editor의 Javadoc 페이지를 표시하기 위하여 메뉴에서 [Project | Project Properties | Build | Javadoc]를 클릭한다.



일단 Javadoc page가 표시되면 다음과 같은 작업중에 하나를 수행한다. 먼저 새로운 Javadoc tag를 생성하기 위하여 [New] 버튼을 클릭한다. 그러면 [Create Custom Tag] 대화상자가 표시되는데, 여기에서 tab name, heading text와 placement 옵션를 적절하게 입력하고 선택해 준다. 사용자 Javadoc tag를 수정하기 위해서 tag 항목안에서 수정하고자 하는 tag를 선택한 후 [Edit] 버튼을 누르고 수정한다. 사용자 Javadoc tag를 삭제하기 위해서 tag 항목안에서 삭제하고자 하는 tag를 선택한 후 [Remove] 버튼을 누르고 삭제한다.



사용자는 header 로서 Javadoc 출력를 표시해 주는 사용자 tag를 위한 텍스트를 추가하기 위해서 Javadoc page에 있는 옵션을 사용할 수 있다.

3.13.2 Using @todo tags in the editor

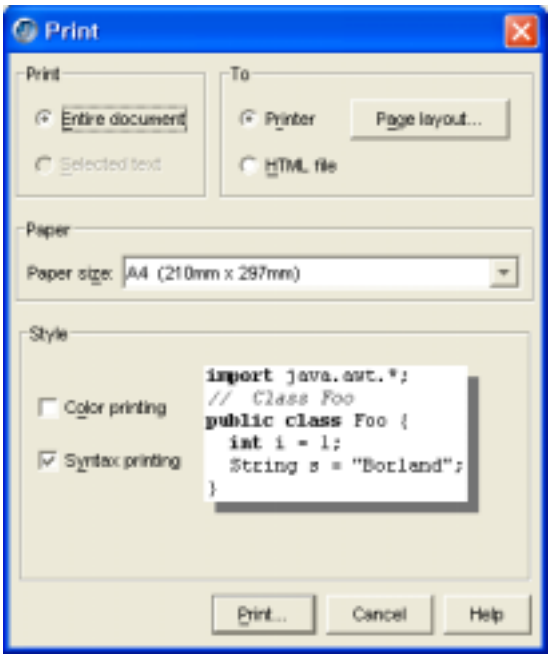
Javadoc @todo tag는 코드영역에서 작업을 필요로 하는 것에 대한 메모(Reminder)를 추가하는데 유용하다. Javadoc @todo tag를 Javadoc comment안에 놓는다. 그러면"@todo" tag는 "To Do" 폴더 안의 JBuilder structure pane에 표시된다. JBuilder code template는 사용자 코드에 매우 쉽게 "@todo" tag를 추가할 수 있도록 한다. 이를 위하여 먼저 editor 안의 적적한 여백에 "todo"을 입력한후, 사용자 코드에 template를 확장하기 위하여 [CTRL+J]를 클릭한다.

```
/** @todo <cursor  placed here>  */
```

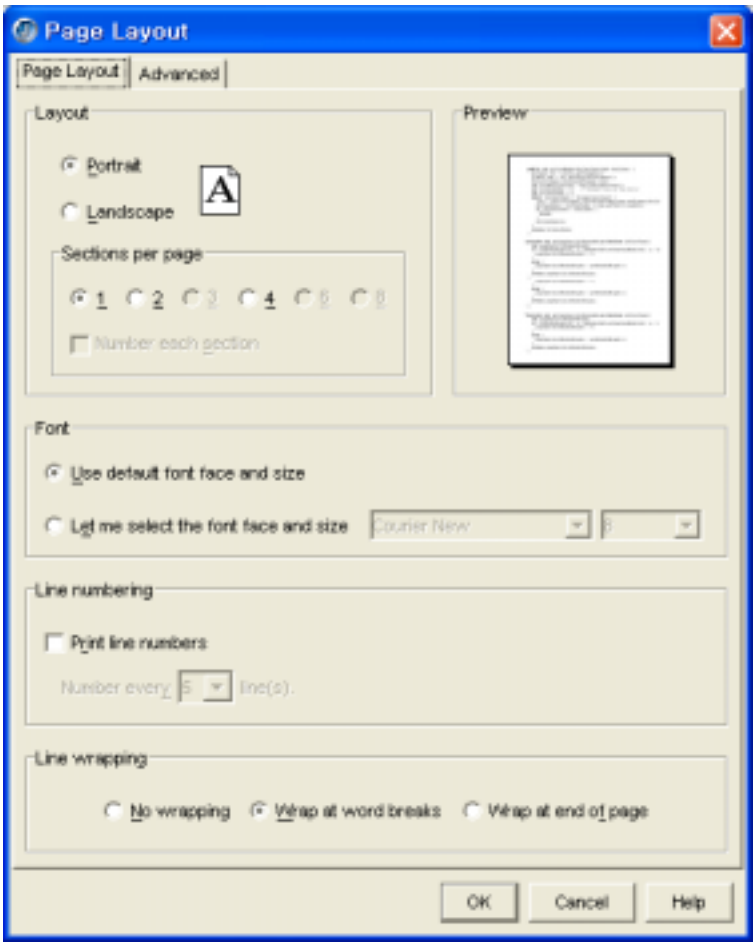
JBuilder 마법사에 따라서 코드를 마법사가 생성한 stub code에 추가하기 위해 "@Todo" tag만을 생성한다.

3.14 Printing support int the editor

사용자는 editor로부터 직접 소스코드를 출력하기 위해서 [File|Print]메뉴를 사용한다.



[File|Page Layout] 메뉴 명령어는 사용자가 layout옵션을 설정한 [Page Layout] 대화상자를 표시해 준다.



[PageLayout] tab 은 page orientation, layout, font, line numbering과 warapping 등을 선택할 수 있다. [Advanced] tab 은 margin과 print page header 를 설정하는 곳으로 page header에서 출력되는 제어변수는 아래와 같다.

변수	의미
%f	파일이름과 경로
%g	파일이름
%p	페이지 숫자
%n	전체 페이지 수
%t	시간
%d	날짜(긴표현)
%s	날짜(짧은표현)
%u	사용자 이름

3.15 Dragging a file into the editor

JBuilder 외부에 있는 파일을 editor 안으로 가져오기 위해 drag&drop를 사용한다. 이파일은 반드시 JBuilder 가 인식할 수 있는 형식이어야 한다. 외부소스로부터 해당 파일을 드래그 한 후에 project pane에 drop 한다. 그러면 그 파일이 editor 안에 열리고, project에 추가된다. project pane 꼭대기에 있는 [Add files/Packages/Classe] 버튼을 사용하거나, project pane안에 있는 project file을 선택하고 마우스 오른쪽 버튼을 클릭하고 [Add files/Packages/Classe] 버튼을 선택한다. drag&drop기능은 Editor 옵션 대화상자에서 사용 여부를 선택할 수 있는 옵션이다.

3.16 Coding shortcuts

JBuilder Editor 안에서 CodeInsight, ErrorInsight, Sync Edit, and code templates 같은 수많은 코딩 단축키를 제공한다. CodeInsight 코드 완성을 도와주며, ErrorInsight는 컴파일시 발생한 error에 대하여 빠른 교정을 제공하며, code template는 editor안에 빠르게 삽입할 수 있는 일반적인 코드 요소를 제공한다.

3.18 Customizing the editor

사용자는 editor 가 표시되고 동작하기 위한 여러 가지 방법을 지정할 수 있다. 예를들면, editor에서 사용하는 font와 font size를 변경할 수 있으며, line numbering를 활성화하거나 비 활성화 할 수 있으며, editor 요소를 위하여 사용자가 좋아하는 색상으로 지정할 수 있으며, 요소를 structure pane에 어떻게 표시될지를 정의할 수 있으며, 또한 일반적인 keystroke 행위를 지정할 수 있으며, 이외에도 많은 것을 할 수 가 있다. 사용자가 원하는 다양한 CodeInsight 기능을 어떻게 동작시키는 지에 대하여 지정할 수 있고, 사용자 자신의 code template를 생성하거나 수정할 수도 있다.

editor를 재 정의하기 위해서 Editor 옵션 대화상자를 이용하는데, 이를위하여 메뉴에서 [Tools|Editor Options] 를 선택한다. 여기에서 사용할 수 있는 모든 옵션에 대한 완전한 정보를 얻기 위해서는 Editor의 다양한 페이지에 있는 [Help] 버튼을 이용한다. 사용자는 메뉴에서 [Help|JBuilder Environment]를 클릭하여 "Customizing the editor" 토픽을 선택하여 editor의 사용자 정의에 대하여 사항을 읽을 수 있다.