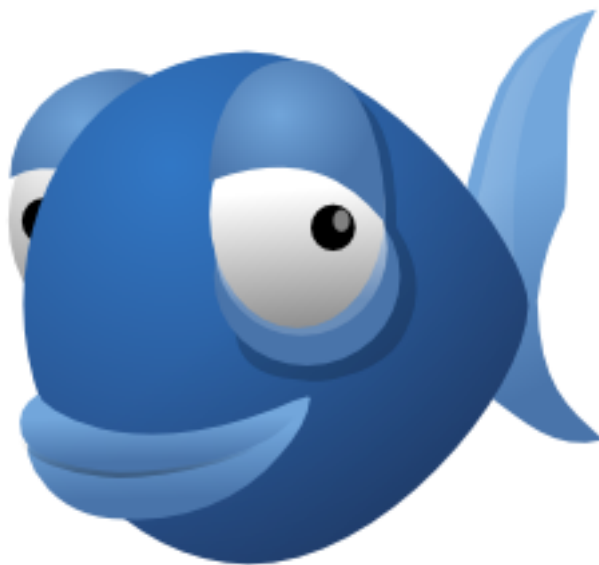


Daniel Blair
Michèle Garoche
Anita Lewis
Alastair Porter
Denny Reeh
Olivier Sessink
Scott White



Bluefish: The Definitive Guide

User's, Developer's, and Documentarian's Guide for Bluefish version 1.0.6

Bluefish: The Definitive Guide: User's, Developer's, and Documentarian's Guide for Bluefish version 1.0.6

by Daniel Blair, Michèle Garoche, Anita Lewis, Alastair Porter, Denny Reeh, Olivier Sessink, and Scott White

Logo art: Dave Lyon

Revision History

Revision 1.6	26/09/2006
Changed to DocBook XSL style sheets 1.71.0. Adding parts and splitting chapters. Improved stylesheets. Adding contents to reference. Adding shortcuts. Update for bluefish version 1.0.6.	
Revision 1.5	26/03/2006
Update for bluefish version 1.0.5.	
Revision 1.4	15/12/2005
Changed to DocBook XSL style sheets 1.69.1. Improved pdf stylesheets.	
Revision 1.3	30/05/2005
Update for bluefish version 1.0.1.	
Revision 1.2	13/04/2005
Changed to DocBook 4.4 and DocBook XSL style sheets 1.68.1. Improved css stylesheets. Added contents to Customizing Bluefish.	
Revision 1.1	03/02/2005
Added sections to Working with files and folders. Added examples to Custom Menu, External programs and filters. Added sections to Customizing Bluefish.	
Revision 1.0	10/01/2005
Initial release for bluefish version 1.0	

Table of Contents

Preface	xvii
1. About this Manual	xvii
2. What is Bluefish?	xvii
2.1. History of Bluefish	xvii
2.2. Main Features of Bluefish	xviii
2.3. How Stable is Bluefish?	xix
2.4. Contact Us	xix
1. Getting and Installing Bluefish	1
I. Getting Bluefish	3
1. Choosing a Version	3
1.1. How and When Updates are Released	3
1.2. Operating Systems Supported by Bluefish	3
2. Latest Stable Version	4
3. Latest Snapshot Version	4
4. Latest Developmental Version	4
II. Installing Bluefish	5
1. Requirements	5
2. Quick Standard Installation	5
3. System Specific Installation Issues	5
4. Installing a Bluefish Source Distribution	7
4.1. Quick Installation Overview	7
4.2. Installing from Development Source Tree	7
4.3. Problems Compiling?	7
5. Configure Options	8
5.1. Standard configuration flags	8
5.2. Flags personal to bluefish	9
6. Installing a Binary Distribution	12
7. Post-installation Setup	12
2. Using Bluefish	13
III. Starting Bluefish	15
1. Launching Bluefish	15
2. Command line options	15
IV. The user interface	17
1. Editing Area	17
2. Menus and Tool Bars	17
3. Side Panel	18
4. Status Bar	20
5. Help Menu	21
V. Working with files and folders	23
1. Creating files	24
2. Managing directories	25
3. Opening files	26
4. Saving files	28
5. Renaming files	29
6. Deleting files	29
7. Closing files	30
8. Inserting files	31
9. Editing	31
9.1. Undo and Redo	31
9.2. Cut, Copy, and Paste	32
9.3. Input methods	32
10. Basic Find and Replace	33
10.1. Searching for a word within a whole document	34
10.2. Setting limits to the search scope	35
10.3. Case sensitive search	37
10.4. Overlapping searches	37
10.5. Retrieving previous search strings	38
10.6. More on find	39
10.7. Replacing features	39
10.8. Retrieving previous replace strings	40

10.9. Changing letter case when replacing	40
10.10. Choosing strings to replace	41
10.11. More on replace	42
11. File types	42
11.1. Syntax highlighting	42
12. More on files	42
12.1. Remote files	42
12.2. Character encoding	43
12.3. Open advanced	43
VI. Navigation and Managing documents	45
1. Navigating through a document	45
2. Navigating through many documents	45
3. Projects	46
4. Bookmarks	49
4.1. Generating several bookmarks at once	50
5. Find and Replace	53
5.1. Find Again	53
5.2. Find from Selection	53
5.3. Find and Replace Using Regular Expressions	54
VII. More than a Text Editor	59
1. Indenting	59
2. Auto tag closing	59
3. Spell checker	60
4. Function reference	60
5. HTML	62
5.1. Special find and replace features	63
5.2. Thumbnail generation	64
6. Customizing the Quick bar	67
7. Custom menu	68
7.1. Adding a custom menu dialog	71
7.2. Adding a custom replace dialog	72
8. External programs, filters	74
8.1. Customizing browsers	75
8.2. Customizing Commands menu	76
8.3. Customizing Ouputbox menu	77
VIII. Customising Bluefish	81
1. Modifying shortcut keys	81
2. Showing hidden files and folders	81
3. Showing backup files	82
4. Editor appearance	82
5. Customizing the bookmarks path	84
6. Customizing the html tags style	84
7. Changing the author meta tag on the fly	84
8. Customizing files handling and browsing	85
8.1. Setting the encoding meta tag on save	85
8.2. Setting the default base directory	86
8.3. Merging file browser views	86
8.4. Backup files	86
8.5. Using multiple instances of a file	86
9. Customizing the user interface	86
10. Modifying file types	87
11. Modifying the files filters	88
12. Modifying the highlighting patterns	89
IX. Shortcuts available within Bluefish	93
1. General Shortcuts	93
2. HTML Shortcuts	94
3. Developing Bluefish	99
X. Debugging Bluefish	101
1. Using the Debugger	101
2. Filling a bug report	101
XI. Reference	103
1. Options in the Editor Tab	103
2. Options in the HTML Tab	103
3. Options in the Files Tab	104

4. Options in the User interface Tab	104
5. Options in the Images Tab	105
XII. Development guidelines	107
1. Indenting and formatting style	107
2. Naming	107
3. Declaring procedures	107
4. Header files	107
5. New files	107
6. File reference	108
7. Patches	108
8. Translations	108
8.1. Introduction	108
8.2. PO files basics	108
8.3. Shortcut keys	109
8.4. How to contribute	109
9. Some tips	109
10. Making releases	109
11. Useful stuff	109
A. Credits	111
1. Bluefish developers	111
2. Bluefish package maintainers	111
3. Bluefish translators	111
4. Supporters to bluefish	112
B. Bluefish change history	113
C. Guidelines for Writing this Manual	117
1. Introduction to DocBook	117
2. Manual building requirements	117
3. Make HTML/PDF/PostScript Versions of the Bluefish manual	118
4. Conventions for Writing this Manual	118
4.1. The id Attribute	119
4.2. Using Screen shots	119
4.3. Referencing Bluefish interface elements	119
4.4. Using procedures	120
4.5. Using notes, tips, warnings	120
4.6. Using links	120
4.7. Others tags	121
4.8. Recommendation	122
4.9. Contact us	122
D. GNU GENERAL PUBLIC LICENSE	123
1. Preamble	123
2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION	123
3. How to Apply These Terms to Your New Programs	125

List of Figures

IV.1. Bluefish Editor Array	17
IV.2. List of open Documents	17
IV.3. Bluefish Main Menu	17
IV.4. Bluefish Main Tool Bar	18
IV.5. Bluefish HTML Tool Bar	18
IV.6. Bluefish Custom Tool Bar	18
IV.7. Bluefish Sidebar Notebook Tab	18
IV.8. Bluefish File Browser	18
IV.9. Bluefish Function Reference Browser	19
IV.10. Bluefish Bookmark Browser	19
IV.11. Bluefish Status Bar	20
IV.12. Bluefish View Menu	20
IV.13. Bluefish About Window	21
V.1. Bluefish File Menu	23
V.2. The File Browser Contextual Menu	23
V.3. Untitled Document Window	24
V.4. The File Name Dialog	25
V.5. The Directory Name Dialog	25
V.6. Bluefish Open File Dialog	26
V.7. Filtering Files with the Bluefish File Browser	27
V.8. Info on open File	28
V.9. Saving an untitled File	28
V.10. A modified Document in the Documents' List	28
V.11. Saving a File under a new Name	29
V.12. Moving a File to another Location	29
V.13. Closing a File with the Document Tab Icon	30
V.14. Closing a modified File	30
V.15. Closing all Files	30
V.16. Inserting a File	31
V.17. The Input Methods Contextual Menu	32
V.18. Writing in Japanese with Bluefish	33
V.19. Finding a Word in a Document, from Start to End	34
V.20. Unsuccessful Search Window	34
V.21. Highlighted Search Result in the Document Window	35
V.22. Setting the Cursor Location	35
V.23. Choosing a limited Search Method	36
V.24. Limited Search Result	36
V.25. Making the Search Case Sensitive	37
V.26. Case Sensitive Search Result	37
V.27. Finding overlapping Strings	38
V.28. An overlapping String retrieved with the Find Dialog	38
V.29. Retrieving recent Searches	39
V.30. The Replace Dialog	40
V.31. Changing letter Case when Replacing	41
V.32. The Replace Confirm Dialog	41
V.33. Opening an URL from the Web	42
V.34. A Style Sheet opened via the Open URL Menu	43
V.35. Using the Open Advanced Dialog	44
VI.1. Bluefish Go Menu	45
VI.2. Using the Goto Line Dialog	46
VI.3. The Bluefish Project Menu	46
VI.4. The Create Project Dialog	46
VI.5. Creating a New Project	47
VI.6. Entering Bluefish Project Filename	47
VI.7. Selecting a Bluefish Project	48
VI.8. Opening a Bluefish Project	48
VI.9. How Bookmarks are marked	49
VI.10. Bookmarks in the Side Panel	49
VI.11. Contextual Menu on Bookmark in the Side Panel	50
VI.12. Editing a Bookmark	50
VI.13. A named Bookmark	50
VI.14. The Contextual Menu on a Document in the Bookmark Tab	50
VI.15. Bookmarking with Posix Regular Expression	51
VI.16. Bookmarks with Posix Regular Expression	51

VI.17. Bookmarking Objective C Functions via the Find Menu	52
VI.18. Bookmarking PHP Functions via the Find Menu	52
VI.19. Nth Occurrence with Find Again	53
VI.20. Nth+1 Occurrence with Find Again	53
VI.21. Selecting a String for subsequent Search	54
VI.22. Finding a String from Selection	54
VI.23. The table before Transformation	55
VI.24. The table after Transformation	56
VII.1. Indenting Part of a Text	59
VII.2. Bluefish Spell Checker	60
VII.3. The Reference Browser Contextual Menu	61
VII.4. The Reference Browser Options Menu	61
VII.5. A Function Reference Dialog Window	61
VII.6. Info available for a Function	62
VII.7. The HTML Tags Menu	62
VII.8. The HTML Dialogs Menu	63
VII.9. An HTML Button with a three-dotted Tool Tip	63
VII.10. A simple HTML Tool Tip Button	63
VII.11. The Replace Special Menu	63
VII.12. The Insert Thumbnail Icon	64
VII.13. The Multi Thumbnail Icon	64
VII.14. The Insert Thumbnail Dialog	64
VII.15. The Multi Thumbnail Dialog	66
VII.16. The Table Icon in the HTML Tool Bar	67
VII.17. Adding an Element to the Quick Bar	67
VII.18. The added Element in the Quick Bar	67
VII.19. Adding a Pop Up Menu Element to the Quick Bar	68
VII.20. Removing an Element from the Quick Bar	68
VII.21. Moving an Element within the Quick Bar	68
VII.22. Accessing the Custom Menu	68
VII.23. The Custom Menu Editor	69
VII.24. Extract of the default Custom Menu Path	70
VII.25. The Custom Replace Dialog	70
VII.26. A new Custom Entry in the Menu Path List	71
VII.27. A new Menu in the Custom Menu Tool Bar	71
VII.28. A Block of selected Text before Activating the Menu	71
VII.29. A Block of Text after Activating the Menu	72
VII.30. The new DIV with Class Dialog	72
VII.31. The Block of Text after Entering the Value	72
VII.32. The HTML Page before the Transformation	73
VII.33. The HTML Page after the Transformation	73
VII.34. The Custom Menu Replace Dialog filled in	74
VII.35. The Add Title Dialog	74
VII.36. Bluefish External Menu	75
VII.37. The Browsers Panel in Preferences	75
VII.38. Selecting the Browser's Line to be Moved	75
VII.39. Dragging the Browser's Line	76
VII.40. Dragging the Browser's Line to the Bottom	76
VII.41. Utilities and Filters Panel in Preferences	77
VII.42. The tidy Output Box	78
VII.43. The Output Parsers Tab in Preferences Panel	78
VIII.1. Adding a Shortcut to a Menu Item	81
VIII.2. Turning Files and Folders Visibility ON	81
VIII.3. Bluefish with a customized Gtk Theme	83
VIII.4. The Editor Tab in Preferences	83
VIII.5. The Bookmarks Path Pop up Menu in Preferences	84
VIII.6. The HTML Tab in Preferences	84
VIII.7. The Author Meta Tag filled in on Save	84
VIII.8. Update of the Author Meta Tag on Save	85
VIII.9. The Files Preference Panel	85
VIII.10. Choosing an Action on Backup Failure	86
VIII.11. The User Interface Preference Panel	87
VIII.12. The HTML Pattern	89
VIII.13. The <html> Tags Pattern	90
VIII.14. The HTML Attributes Pattern	90
VIII.15. The HTML Attribute Contents Pattern	91
VIII.16. The PHP Block Pattern	91

VIII.17. The Comment (C++/single line) Pattern	92
VIII.18. Syntax Highlighting Example	92

List of Tables

IX.1. Shortcuts for all Document Types	93
IX.2. Shortcuts for HTML Markup	94
XI.1. Editor Tab Reference	103
XI.2. HTML Tab Reference	103
XI.3. HTML Tab Reference	104
XI.4. User interface Tab Reference	104
XI.5. Images Tab Reference	105

List of Examples

VI.1. Retrieving all sections in an xml book	54
VI.2. Retrieving all functions in an Objective C file	55
VI.3. Retrieving all functions in a PHP file	55
VI.4. Transforming a table into a definition list	55
VIII.1. Adding a file type	87
VIII.2. Adding a file filter	88

List of Procedures

I.1. Getting the source	4
II.1. Getting the new defaults after upgrading - First method	12
II.2. Getting the new defaults after upgrading - Second method	12
V.1. Writing in Japanese with Bluefish on a non-Japanese system	33
V.2. Searching from selection	35
VI.1. Creating a New Project	46
VII.1. Generating a photos album with multi thumbnails	66
VII.2. Adding a custom menu based on custom dialog	71
VII.3. Adding a custom menu based on replace dialog	73
VII.4. Changing the order of browsers items	75
VII.5. Customizing an existent browser	76
VII.6. Adding a new browser	76
VII.7. Adding a Commands menu item	77
VII.8. Adding an Outputbox menu item	79
X.1. Running bluefish under gdb	101
C.1. Getting the Bluefish manual source files	117
C.2. Installing DocBook and DocBook XSL	117
C.3. Installing the xslt processors and parsers	117

Preface

1. About this Manual

Bluefish has a large feature set, allowing the user to customize the editing experience in numerous ways. This manual targets both novice and advanced users, providing a full resource for everyone.

[Part 1, “Getting and Installing Bluefish” \[1\]](#) and [Part 2, “Using Bluefish” \[13\]](#) are highly recommended for anyone new to Bluefish. They present general information, installation instructions, an introduction to the main features of Bluefish, and available shortcuts.

[Part 3, “Developing Bluefish” \[99\]](#) is oriented towards developers and testers. It explains how to debug Bluefish, contains the list of features, includes code formatting styles and a reference for all source files.

The manual targets the *end user*. To that end, we have tried to use a simple, well-explained approach whenever possible. Some typographic conventions are denoted below:

- Any URLs are denoted like this: <http://bluefish.openoffice.nl>
- Shortcuts look like this: **Ctrl-S**
- Menu options are displayed like this: **File**. However, many of Bluefish's menus are quite complex. When referring to submenus, options are separated by an arrow, like: **File → Open (Ctrl-O)**. The default keyboard shortcut is shown in parenthesis.
- When referring to user input, like issuing commands to the command prompt, a monotype font is used:

```
$ foo -bar | bang -l
```



Do not write the `$` character - it simply identifies the command prompt. For commands requiring root access, the prompt is shown as a `#`.

Finally, if you find errors or omissions in this manual, fill in a [documentation bug report \[xix\]](#)

2. What is Bluefish?

Bluefish is a powerful editor for experienced web designers and programmers based on the GTK2 GUI interface. Bluefish supports many programming and markup languages, but focuses on editing dynamic and interactive websites.

Bluefish is not a WYSIWYG¹ text editor. This is deliberate, allowing the programmer to stay in full control. To facilitate the editing process, a large number of features are at your disposal. For inserting markup and code, there are tool bars, dialogs, and predefined/user-customized menus. Syntax highlighting, advanced search/replace functionality, scalability and language function references make Bluefish a powerful tool for development.

2.1. History of Bluefish

Bluefish development started under a different name. A good and free text editor targeted towards web development was not available. Olivier Sessink started the project ProSite. Chris Mazuc also started an HTML editor. On a GTK development mailing list, Olivier Sessink and Chris Mazuc saw each others postings, and decided to team up. Olivier had a basic editor, Chris had many HTML dialogs ready. After merging the code this was for a while known as the Thtml editor.

After a while Neil Millar joined the project to add weblint integration and a color dialog. Because the project became larger and more mature, a logo was wanted. After many discussions about boring logos, Neil Millar came up with a cute blue fish. Because this logo was appreciated by all, the name changed into the final name Bluefish.

After this initial stage, many developers, translators, testers and users joined the project.

Several years have passed since the first Bluefish release. Since that time, the fish has gained a reputation as an excellent editor, with qualities like stability, usability and numerous features. Also, Bluefish is small, fast and efficient, making it usable even on slow machines.

¹What You See Is What You Get

2.2. Main Features of Bluefish

This list will give you an overview of the most important or outstanding features found in Bluefish:

- A What You Write Is What You Get interface
- Multiple document interface, will easily open 500+ documents (tested 3500 with documents simultaneously).
- Customizable syntax highlighting based on Perl compatible regular expressions, with subpattern support. Default patterns are included for:
 - C
 - cfml
 - ChangeLog
 - CSS stylesheet
 - Gettext po
 - HTML
 - Java
 - JScript
 - JavaScript
 - Octave
 - Pascal
 - Perl
 - PHP
 - Python
 - R
 - Shell
 - SQL
 - Tcl
 - Ruby
 - XML
- Anti-aliased text window
- Multiple encodings support, can convert between different character sets, supports multibyte characters, Unicode, UTF8, etc.
- Nice wizards for startup, tables, frames, and others
- Dialogs for many HTML tags, with all their attributes
- HTML tool bar and tear-off menus
- User-customizable tool bar for quick access to often used functions
- Open files based on filename patterns and/or content, from selection or URIs
- Fully featured image insert dialog
- Thumbnail creation and automatically linking of the thumbnail with the original image
- Multi-thumbnail generation for easy creation of photo albums or screen shot pages
- Line numbers along the document
- Bookmarks for lines across multiple documents, with bookmark browser
- A custom menu, specify your own tags or sets of code, and define your own dialogs
- Custom search and replace pattern support for the Custom menu
- Very powerful search and replace, allowing POSIX and Perl Compatible regular expressions and sub-pattern replacing
- Excellent undo/redo functionality
- Configurable recent documents and recent directories functionality
- Spell checking
- Translations in *twenty languages*
- User customizable integration with many programs, including weblint, tidy, make, javac, etc.
- XML based function reference. Currently, references are included for Apache, DHTML, DocBook, HTML, PHP, and SQL. A GTK reference is available, and support for Perl and Python will be added. You may also create your own function reference. The XML format is described later in the manual.
- XML based reference library for CSS2, HTML, PHP, and Python.
- Projects management.

As Bluefish is a part of a larger desktop environment, we have focused on making the GUI consistent with the Gnome HIG².

²GNOME Human Interface Guidelines, accessible at <http://developer.gnome.org/projects/gup/hig/>

However, we prefer not following it in every detail, as some parts are intended *for the end user*, while Bluefish is *for the programmer*.

2.3. How Stable is Bluefish?

Quite stable! The Bluefish developers aim to produce code that neither crashes nor leaks memory. Of course, that is not always easy to do. Leaks and crashes are often fixed in CVS as soon as they are discovered and hunted down. In addition to Bluefish's large user base, the developers use Bluefish for their daily work. So, fixing bugs and preventing crashes is always a major priority. However, some nags still exist. One example being the issue of slightly sluggish copy/paste functions.

For an updated list of open bugs, please go to the <http://bugzilla.gnome.org/browse.cgi?product=bluefish> page on bugzilla gnome.

We appreciate any and all contributions! Please tell us if Bluefish crashes on you :-).

2.4. Contact Us

We, the Bluefish development team, welcome all comments, user requests, constructive criticisms, and contributions. Are you curious or seeking information regarding Bluefish? Would you like to contribute by translating Bluefish or its manual? Here are your options:

- <http://bluefish.openoffice.nl/> - The main website where you will find news, updates, manual and more information.
- <http://bfwiki.tellefsen.net/> - The Bluefish WiKi is the notebook for the developers, containing a lot of information. This includes, but is not limited to: updated project road maps, status of translations.
- For bug reports and features requests, related to bluefish, its web site, and its manual, please fill the appropriate bug type at <http://bugzilla.gnome.org/browse.cgi?product=bluefish>, the bluefish bugzilla page on bugzilla gnome - for instructions, see [Section 2, "Filling a bug report" \[101\]](#)
- You can subscribe to the users' Bluefish mailing list by sending an email containing "subscribe bluefish-users" to <bluefish-users-request@lists.ems.ru>.
- You can subscribe to the developers' Bluefish mailing list by sending an email containing "subscribe bluefish-dev" to <bluefish-dev-request@lists.ems.ru>.
- Do you want to help translate Bluefish? Please let us know by dropping an email to Walter Echarri <wecharri(at)arnet.com.ar>, our friendly translation maintainer.
- If you want to help improve the manual or write new sections, please send an email to the Bluefish manual maintainer, Michèle Garoche <michele.garoche(at)easyconnect.fr>. Guidelines for this manual can be found in [Appendix C, Guidelines for Writing this Manual \[117\]](#)
- If you have a general question, drop an email to <bluefish(at)bluefish.openoffice.nl>.

Part 1. Getting and Installing Bluefish

In this part are described the various versions of Bluefish, their installation and the existing configure options.

Chapter I, *Getting Bluefish* [3]

Chapter II, *Installing Bluefish* [5]

Chapter I. Getting Bluefish

1. Choosing a Version

Currently, four versions of bluefish are available:

- The GTK1-version (v0.7) is deprecated and no longer updated, but is the choice for those of you still running GTK1.
- The latest GTK2-version (v1.0.5) is the version of choice for most users, and is regarded as stable enough for daily use.
- The latest development snapshot is always one step further than the latest stable release. In it, you will find some new features, bug fixes, and a prettier GUI. The catch is that it may have unfinished or buggy features. Try this if you want to see new features or are bothered by a bug in the latest stable release.
- CVS is the bleeding edge of Bluefish development. You may find that the CVS version has several bug fixes and enhancements, however, it may also contain new, inadvertent bugs. You will also need the CVS version if you want to contribute a patch. Although the CVS version may be unusable for short periods of time, it is often stable enough for daily use.



The CVS version comes with a compilation flag `--enable-unstable`, which allows you to install it concurrently with another version of bluefish, either the GTK2 one, or the snapshot one, or the 'normal' CVS one. It introduces supplementary functionalities. When this flag is used, bluefish is installed under the name of `bluefish-unstable`, vs. `bluefish` for any other version. This flag is enabled by default.

If you want the latest and greatest, read [Section 4, “Latest Developmental Version” \[4\]](#) below or [Section 3, “Latest Snapshot Version” \[4\]](#) if you do not want to install CVS. If you simply want to use Bluefish, read [Section 2, “Latest Stable Version” \[?\]](#) for how to get the latest stable package for your system.

1.1. How and When Updates are Released

Due to the small number of volunteer developers, the progression of Bluefish's development often fluctuates. For this reason, a long time may pass between each release. After all, the developers volunteer their time and effort because they actually want to *use* Bluefish :-)

Consequently the current CVS or CVS snapshots may be what you want to use. We do try to keep the CVS version usable at any time (actually, the CVS version is used by most of the development team on a daily basis).

1.2. Operating Systems Supported by Bluefish

Bluefish has been reported to work on a number of systems. The Bluefish team mainly support these platforms:

- Mandrake Linux
- Red Hat Linux
- Fedora Core
- Debian Linux (and derivatives)
- FreeBSD

Actually, any GNU/Linux distribution with GTK2 is fine and many distributions include Bluefish. In fact, Bluefish will likely work quite well on any POSIX compatible OS where GTK2 is available. Bluefish has been reported to work on the following:

- NetBSD - distributed in pkgsrc
- OpenBSD - available through their ports system
- SGI IRIX - see <http://freeware.sgi.com/>
- Mac OS X via [Fink](http://fink.sourceforge.net/) [<http://fink.sourceforge.net/>] or [Darwin Ports](http://darwinports.opendarwin.org/DarwinPorts) [<http://darwinports.opendarwin.org/DarwinPorts>].
- Sun Solaris
- Tru64
- AIX
- HP-UX
- Win32-cygwin - with a few nags.

2. Latest Stable Version

Many Linux distributions ship a version of Bluefish or make it available through their package systems. For example, Bluefish is available through the Debian apt-system and FreeBSD's ports. You may check if Bluefish is available through your favorite software installer.

However, the main source is the Bluefish website, where the software and a few contributions are available. The download page is reachable at <http://bluefish.openoffice.nl/download.html>. Here, you may download the source code and binary packages for Debian, Red Hat/Fedora, and Mandrake.

3. Latest Snapshot Version

Snapshots are made regularly to provide users with the latest functionalities without having to deal with CVS. Though often not tested, they may be convenient for evaluation. They are available at <http://bluefish.openoffice.nl/download.html>

4. Latest Developmental Version

To get the latest version of Bluefish you will need to download the source files from our CVS repository.

CVS¹, a version control system, is a widely used software development tool. It keeps track of changes to the source code, and allows for reversion to previous states. If you want to read more about CVS, have a look at the CVS-book by Karl Fogel, available at <http://cvsbook.red-bean.com/cvsbook.html>.

The Bluefish project's CVS repository is generously hosted by SourceForge.net². For more information about them, see their site. The project homepage is <http://sourceforge.net/projects/bluefish>. Our CVS repository contains the current Bluefish source code, including this manual. The repository is accessible by anyone, and is updated almost daily by the developers.

To access the repository, you need a few small utilities. They are likely to be available through your favorite source of software (ports, apt, etc). The above-mentioned CVS book is a great source for information.

Here's how you get the source.

Procedure I.1. Getting the source

1. Go to the directory in which you want to put the sources:

```
$ cd TheChooosenDirectory
```

2. Log in using the command:

```
$ cvs -d :pserver:anonymous@bluefish.cvs.sourceforge.net:/cvsroot/bluefish \
login
```



Hit Enter at the password prompt.

3. *Check out* the CVS module containing the source code files:

```
$ cvs -z3 -d :pserver:anonymous@bluefish.cvs.sourceforge.net:/cvsroot/bluefish \
co bluefish-gtk2
```

A lot of files will be downloaded, and listed one by one. If you're on dial-up, this might take a bit of time. When the downloads have completed, you will find the bluefish sources in the subdirectory `bluefish-gtk2`.

4. Enter that directory and install bluefish using the instructions detailed in [Section 4.2, "Installing from Development Source Tree" \[7\]](#).

¹Concurrent Versions System

²<http://www.sourceforge.net>

Chapter II. Installing Bluefish

1. Requirements

Bluefish aims to be portable; that is, wherever GTK is ported. A comparatively small set of external libraries are necessary for it to work. Any recent GNU/Linux distribution or other *NIX with GTK2 installed should be sufficient. In addition to the list of requirements below, you may also want to look at [Section 3, “System Specific Installation Issues” \[5\]](#). Note that these requirements fit the GTK2 version. If you only have GTK1, you want the last GTK1-version, v0.7.

The main requirements:

- gtk v2.0
- libpcre

Optional requirements:

- gnome_vfs - *for remote file support*
- libaspell - *spell checker*
- grep & find - used by the **File** → **Open advanced...** (**Shift-Ctrl-O**) dialog.

Compiling Bluefish requires a few additional packages. However, binary packages exist for many platforms, so it is likely you will not need to compile. Now, let us assume you want to compile, perhaps to get the latest and greatest from CVS. The requirements are as follows:

- Development files (header files, etc) for the packages above. *These are often distributed as separate packages. There is also a high probability you have these installed already.*
- gcc - *Bluefish has been tested to compile on the 2.95, 3.x, and 4.0.x branches.*
- gmake or BSD make
- autoconf - *only if you are going to compile from CVS*
- gtk v2.4 at least - *only if you are going to compile from CVS*
- gnome_vfs 2.6 - *only if you are going to compile from CVS*
- libxml 2.0 - *only if you are going to compile from CVS*

2. Quick Standard Installation

There are two main methods for installing Bluefish: Compile from source or install a binary package. Binary installation is easiest, so we will cover that first. There are a few different approaches, caused by the differences between systems. We will start off by summarizing a few really quick and simple approaches before dealing with this problem more extensively.

- Debian: run `su - && apt-get update && apt-get install bluefish`
- Red Hat, Mandrake (and other Linux distributions that support rpm): Download the latest .rpm from the [Bluefish website](http://bluefish.openoffice.nl/) [http://bluefish.openoffice.nl/]
- FreeBSD, NetBSD, OpenBSD, Fink, and DarwinPorts distribute Bluefish through their packaging systems.
- To compile, or install on another platform, see [Section 4, “Installing a Bluefish Source Distribution” \[7\]](#).

3. System Specific Installation Issues

Different systems have different approaches to solutions and packaging. You might find the information below interesting.

Cygwin:

- You need to install the following packages:
 - From the Admin section: **cygrunsrv** (installs a necessary service to Windows)
 - From the Base section: all packages in this section should be installed by default
 - From the Devel section: **ORBit2-devel**, **atk-devel**, **autoconf2.5**, **automake1.x** (here we suggest automake1.7), **binutils**, **bison**, **catgets**, **cvs**, **gcc**, **gettext**, **gettext-devel**, **glib2-devel**, **gtk2-x11-devel**, **libbonobo2-devel**, **libfontconfig-devel**, **libfreetype2-devel**, **libxml2-devel**, **make**, **minires-devel**, **openssl-devel**, **pango-devel**, **pcre**, **pcre-devel**, **pkgconfig**
 - From the Gnome section: **gnome-vfs2**, **libgnome2**
 - From the Libs section: **popt**
 - From the X11 section: **xorg-x11-devel**, **xorg-x11-base**, **xorg-x11-bin**, **xorg-x11-bin-dlls**, **xorg-x11-fenc**,

xorg-x11-fnts, xorg-x11-fsrv, xorg-x11-f100, xorg-x11-fcyr, xorg-x11-fscl

- You may want to install the following optional packages:
 - From the Devel section: **libxml2** (for the **xmllint** tool)
 - From the Gnome section: **libgnomeui2**
 - From the Interpreters section: **python** (only for Bluefish 1.1 and above)
 - From the Text section: **aspell-dev** (for spell-checker), **aspell-LANG** (dictionary for your language)
 - From the Utils section: **bzip2** (to decompress bzip2-compressed archives), **desktop-file-utils** (freedesktop.org menu support), **gnome-mime-data** (old GNOME <= 2.4 MIME support), **shared-mime-info** (freedesktop.org shared MIME-info database)
 - From the Web section: **tidy**, **wget** (to download Bluefish archives)
 - From the X11 section: **hicolor-icon-theme**
- Fetch the source and compile it as explained in [Section 4, “Installing a Bluefish Source Distribution” \[7\]](#) .
- To run Bluefish, you need to start the **cygrunsrv** Service. First log in to a Cygwin-Shell and run **/usr/bin/cygserver-config**. Answer Yes to install cygserver as service. Then open a Windows shell with the START button of your Windows box and choose **Run**. Type **cmd.exe** or **command.exe**, hit enter and type **net start cygserver**. To automatically start the service with Windows, set Starttype for **cygrunsrv** to *Automatic* (see Start > Control Panel > Computer Administration > Services and Applications > Services : CYGWIN cygserver : Properties). To stop the service, type **net stop cygserver**.
- To allow Bluefish to use the Cygserver facilities (to use the XSI IPC function calls like msgget successfully) you need to export the CYGWIN environment variable. Add the following line to your `~/ .bash_profile`:

```
$ export CYGWIN=server
```

- To run Bluefish, first start the X server with **startx**, then launch bluefish with **/usr/bin/bluefish.exe** (if you have configured bluefish with **./configure --prefix=/usr**).



For further details, see the [Compile Bluefish for Cygwin](http://bfwiki.tellefsen.net/index.php?pagename=CompilingUnderCygwin) [<http://bfwiki.tellefsen.net/index.php?pagename=CompilingUnderCygwin>] wiki page.

Debian:

- Debian Woody (the current Stable) has an old GTK 2.0.2 version, that contains several known bugs, but they are not serious.
- Debian Sarge (currently in Testing) has Bluefish 1.0 and GTK 2.6.4 version.
- Debian Sid (Unstable) will always have the latest Bluefish version.

Mandrake:

- libpcre: Breaks pcre into 3 different pieces, make sure pcre-devel is installed if compiling from source. Try this command:

```
$ rpm -ql pcre-devel
```

- ... *more nags with Mandrake?*

Fink:

- Fink binary distribution for Mac OS X 10.1 has Bluefish 0.7 (unmaintained).
- Fink for Mac OS X 10.2 has Bluefish 0.12 in binary or stable source distribution, and Bluefish 0.13 in unstable source branch.
- Fink for Mac OS X 10.3 and 10.4-transitional has Bluefish 1.0 in binary distribution, Bluefish 1.0.4 in stable source branch, and Bluefish 1.0.5 in unstable source branch. The source branches have three variants: bluefish, bluefish-gnomevfs2, and bluefish-gnome2.
- Fink's latest cvs version is always available from the [Fink's maintainer experimental cvs tree](http://fink.cvs.sourceforge.net/fink/experimental/michga/) [<http://fink.cvs.sourceforge.net/fink/experimental/michga/>]. It is meant to run on Mac OS X 10.4.



If you have trouble starting bluefish with fink, first read the [Fink's Faq page](http://fink.sourceforge.net/faq/index.php) [<http://fink.sourceforge.net/faq/index.php>]. If this does not solve your problem, send an email to the maintainer (<michele.garoche(at)easyconnet.fr>), finally, as last resort, ask for help on the [beginners or users Fink mailing lists](http://fink.sourceforge.net/lists/index.php) [<http://fink.sourceforge.net/lists/index.php>] or on the “#fink” channel on [freenode.net](http://irc.freenode.net/#fink) [[irc://irc.freenode.net/#fink](http://irc.freenode.net/#fink)].

DarwinPorts:

- Version 1.0.5 is available. Go to <http://bluefish.darwinports.com/> to find detailed informations about its installation.

4. Installing a Bluefish Source Distribution

By installing Bluefish from source, you may be able to get a newer version (from CVS) than those distributed as binaries. You may also need to compile from source if no binary is available for your system.

4.1. Quick Installation Overview

This is the short installation description. Consult the other chapters if you are in doubt.

Bluefish is installed using the standard 'configure, make, make install' steps. Assuming you have downloaded a bluefish source package, for instance `bluefish-ver.tar.gz` (naturally, change the filename to what's appropriate), you complete the installation with the following steps:

1. `tar -zxvf bluefish-ver.tar.gz`
2. `cd bluefish-ver`
3. `./configure`
4. `make`
5. `su -c 'make install'`
6. Now, type `bluefish` to run. You may delete the `bluefish-ver` directory.

The **configure** script is used to automatically find the appropriate settings for your system. Because of differences between systems, this compile-time configuration is necessary, and configure solves this challenge easily -- with an added bonus of telling whether you have everything needed to compile.

The configure-script can be, um, configured. This is something you most likely will not need to do, but it is easy to do if necessary. For a complete list of configure options, see [Section 5, "Configure Options" \[8\]](#)

4.2. Installing from Development Source Tree

You can get the latest Bluefish version via CVS using the instructions in [Section 4, "Latest Developmental Version" \[4\]](#). Next, install it with the following steps:

1. Enter the directory containing the bluefish source files: `cd bluefish-gtk2`
2. Next, generate the configure script by running `autoconf`
3. Then, you run **configure** with whatever options you might want.

This example will cause **make install** to install Bluefish with the specified directory as prefix (i.e. the binary is installed in `/usr/local/bf-cvs/bin/bluefish`). This is most likely not what you want -- just run configure without parameters instead.

```
$ ./configure --prefix=/usr/local/bf-cvs
```

If configure fails, it will probably give a hint telling you what is missing or wrong.

4. Assuming it completed successfully, your next step is to compile Bluefish. To do this, run **make**.
5. When **make** has completed, you can install Bluefish: (**su** to root first, unless you specified a user writable prefix to configure), then issue: `# make install`.

To update the sources at a later time, you run the command `cvs -z3 -q update` from within the `bluefish-gtk2` directory.

4.3. Problems Compiling?

If compilation fails, first make sure you have the necessary utilities and libraries. See [Section 1, "Requirements" \[5\]](#).

Next, see if your system is mentioned in [Section 3, "System Specific Installation Issues" \[5\]](#).

Below is a list of well known problems that have been mentioned on the bluefish-dev list:

- make: *** No targets specified and no makefile found. Stop.
This will happen if *configure* fails and you try to run **make**. It also happens if you're running **make** from the wrong directory.
- ... more trouble to come ;-)

If you're unable to find a solution (or if you think you have a solution others might want), feel free to contact us on the *bluefish-dev* list (See [Section 2.4, “Contact Us” \[xix\]](#)). You may want to contact the bluefish package maintainer for your distribution first when appropriated.

5. Configure Options

This section describes all the configure options available for bluefish.

5.1. Standard configuration flags

Configuration:

```
-h
--help
    display this help and exit
--help=short
    display options specific to this package
--help=recursive
    display the short help of all the included packages
-V
--version
    display version information and exit
-q
--quiet
--silent
    do not print "checking..." messages
--cache-file=FILE
    cache test results in FILE [disabled by default]
-C
--config-cache
    alias for --cache-file=config.cache
-n
--no-create
    do not create output files
--srcdir=DIR
    find the sources in DIR [configure dir or . . by default]
```

Installation directories:



By default, **make install** will install all the files in `/usr/local/bin`, `/usr/local/lib`, etc. You can specify an installation prefix other than `/usr/local` using `--prefix`, for instance `--prefix=$HOME`.

```
--prefix=PREFIX
    install architecture-independent files in PREFIX [/usr/local by default]
--exec-prefix=EPREFIX
    install architecture-dependent files in EPREFIX [PREFIX by default]
```

Fine tuning of the installation directories:



For better control, use the options below. Defaults are shown within brackets.

```
--bindir=DIR
    user executables [EPREFIX/bin]
--sbindir=DIR
    system admin executables [EPREFIX/sbin]
--libexecdir=DIR
    program executables [EPREFIX/libexec]
--datadir=DIR
    read-only architecture-independent data [PREFIX/share]
--sysconfdir=DIR
    read-only single-machine data [PREFIX/etc]
```

```
--sharedstatedir=DIR
    modifiable architecture-independent data [PREFIX/com]
--localstatedir=DIR
    modifiable single-machine data [PREFIX/var]
--libdir=DIR
    object code libraries [EPREFIX/lib]
--includedir=DIR
    C header files [PREFIX/include]
--oldincludedir=DIR
    C header files for non-gcc [/usr/include]
--infodir=DIR
    info documentation [PREFIX/info]
--mandir=DIR
    man documentation [PREFIX/man]
```

Program names:

```
--program-prefix=PREFIX
    prepend PREFIX to installed program names
--program-suffix=SUFFIX
    append SUFFIX to installed program names
--program-transform-name=PROGRAM
    run sed PROGRAM on installed program names
```

System types:

```
--build=BUILD
    configure for building on BUILD [guessed]
--host=HOST
    cross-compile to build programs to run on HOST [BUILD]
```

Some influential environment variables:



Use these variables to override the choices made by **configure** or to help it to find libraries and programs with nonstandard names/locations.

```
CC
    C compiler command
CFLAGS
    C compiler flags
LDFLAGS
    linker flags, e.g. -L<lib dir> if you have libraries in a nonstandard directory <lib dir>
CPPFLAGS
    C/C++ preprocessor flags, e.g. -I<include dir> if you have headers in a nonstandard directory <include dir>
CPP
    C preprocessor
```

5.2. Flags personal to bluefish

Optional Features:



These work like this: `--enable-feature` enables the feature, `--disable-feature` or `--enable-feature=no` disables the feature.

By default, the `--enable-feature` option is not enabled, you should pass it if you want to get it, the `--disable-xxx` option is not disabled, you should pass it if you want to disable it.

```
--enable-auto-optimization
    Optimizes the build process for a given architecture if possible. It works only on a selected set of x86 platforms.
```

How: rely on the result of:

1. `uname -p` or `grep "model name" /proc/cpuinfo | cut -d: -f2` to detect the architecture
2. the version of gcc to pass the arguments

Tested gcc versions: 3.2.*, 3.0.*, 2.95.*

Machines: Intel(R) Pentium(R) 4CPU, Pentium III, AMD-K6 (tm) 3D, Pentium 75 - 200, Pentium II, AMD Athlon(TM) XP

Other machines are ignored

`--enable-gcc3-optimization=type`

optimizes the build process for a given architecture if possible

Machines: i386, i486, pentium, pentium-mmx, pentiumpro, pentium2, pentium3, pentium4, k6, k6-2, k6-3, athlon, athlon-tbird, athlon-4, athlon-xp, athlon-mp, winchip-c6, winchip2, c3

Other machines are ignored

`--enable-gcc2-optimization=type`

optimizes the build process for a given architecture if possible

Machines: i386, i486, pentium, pentiumpro, k6

Other machines are ignored

`--enable-debugging-output`

turns debugging output on (this option impacts performance)

`--disable-splash-screen`

suppresses the display of the splash screen at launch time (Bluefish launches faster)

`--enable-highlight-profiling`

outputs statistics on where the program spends most of its time when highlighting patterns

Usage: for debugging highlight patterns or trying to optimize the program

`--enable-development`

enables development checks (slows down the program)

`--enable-gprof-profiling`

outputs statistics on where the program spends most of its time by generating extra code to write profile information suitable for the analysis (slows down the program)

`--enable-gcov-coverage`

Purpose: to be able to collect statistics on how many times each branch is executed and how long it has lasted. Creates data files for the gcov code-coverage utility (slows down the program)

`--disable-update-databases`

do not run the update-desktop-database or update-mime-database utilities after installation (mostly useful for package maintainers)

`--disable-nls`

disables the Native Language Support (might speed up the program)

Optional Packages:



These work like this: `--with-xxx=foo` enables the flag, `--without-xxx` disables it. When not enabled, the default is used.

`--without-libgnomeui`

disable libgnomeui detection

`--with-gnome1-menu`

customized path for the gnome1 menu (disabled by default)

Usage: `--with-gnome1-menu=customizedpath` or `--without-gnome1-menu`

disabled by default

`--with-freedesktop_org-menu`

customized path for the freedesktop.org (gnome and kde) menu

Usage: `--with-freedesktop_org-menu=customizedpath` or `--without-freedesktop_org-menu`
defaults to auto-detection, which tries:

- /usr/share/applications
- PREFIX/share/applications
- /usr/X11R6/share/gnome/applications
- PREFIX/share/gnome/applications

`--with-freedesktop_org-mime`

customized path for the freedesktop.org (gnome and kde) mime

Usage: `--with-freedesktop_org-mime=customizedpath` or `--without-freedesktop_org-mime`
defaults to auto-detection, which tries:

- `/usr/share/mime`
- `PREFIX/share/mime`
- `/usr/X11R6/share/gnome/mime`
- `PREFIX/share/gnome/mime`

`--with-gnome2_4-mime`

customized path for the gnome 2.4 mime

Usage: `--with-gnome2_4-mime=customizedpath` or `--without-gnome2_4-mime`
defaults to auto-detection, which tries:

- `/usr/share/mime-info`
- `PREFIX/share/mime-info`
- `/usr/X11R6/share/gnome/mime-info`
- `PREFIX/share/gnome/mime-info`

`--with-gnome2_4-appreg`

customized path for the gnome 2.4 application registry

Usage: `--with-gnome2_4-appreg=customizedpath` or `--without-gnome2_4-appreg`
defaults to auto-detection, which tries:

- `/usr/share/application-registry`
- `PREFIX/share/application-registry`
- `/usr/X11R6/share/gnome/application-registry`
- `PREFIX/share/gnome/application-registry`

`--with-icon-path`

customized path for the icon.

Usage: `--with-icon-path=customizedpath` or `--without-icon-path`
defaults to auto-detection, which tries:

- `/usr/share/pixmaps`
- `PREFIX/share/pixmaps`
- `/usr/X11R6/share/gnome/pixmaps`
- `PREFIX/share/gnome/pixmaps`

`--with-libiconv-prefix`

customized path for libiconv top level installation.

Usage: `--with-libiconv-prefix=customizeddir`

Effect: searches for libiconv in `customizeddir/include` and `customizeddir/lib`

`--with-included-gettext`

use the GNU gettext library included in the package

Features specific to the CVS version:

`--enable-unstable-install`

enable the installation of a bluefish development version with independent directories and files. Use this for CVS snapshots

`--enable-scanner`

use the BfTextView scanner for editor widget

`--disable-python`

do not build the python plugin

`--with-theme-path`

customized path for the theme icons.

Usage: `--with-theme-path=customizedpath` or `--without-theme-path`
defaults to auto-detection, which tries:

- `/usr/share/icons/hicolor`
- `PREFIX/share/icons/hicolor`

- `/usr/X11R6/share/gnome/icons/hicolor`
- `PREFIX/share/gnome/icons/hicolor`

6. Installing a Binary Distribution

Different packages -- different installation.

We will cover only a few approaches here¹, since the installation is very system-specific ;-). Let us have a look at some different systems:

- For Debian users this is very simple. To download, install, and configure bluefish in “One Swift Move”, run:

```
$ su - && apt-get update && apt-get install bluefish
```

You can check if the version available through apt is the latest -- see the Bluefish homepage, and compare the version there with what `apt-cache show bluefish` tells you. If there is a newer version on the Bluefish site, download it and install the package like this: `dpkg -i bluefish-ver.deb`

- For rpm based distributions, first check if your distribution has a recent Bluefish version. If it does not, download the rpm for your distribution from any of the Bluefish mirrors. Installing a downloaded rpm is as simple as pointing and clicking in your favorite GUI package manager, or issuing the following command from the command line (as root):

```
# rpm -Uvh bluefish-ver.rpm
```

- If you're using FreeBSD, NetBSD or OpenBSD, we probably do not need to tell *you* how to use your favorite package system ;-)
- For Mac users, just install it via Fink. Depending of you operating system's version you have the choice between:
 - bluefish v. 0.7 for Mac OS X 10.1
 - bluefish v. 0.12 for Mac OS X 10.2
 - bluefish v. 1.0 for Mac OS X 10.3 and above

7. Post-installation Setup

The first time you run Bluefish it will create a directory `~/ .bluefish` where all Bluefish's configuration options are stored. This includes all preferences, customized menus, highlighting-patterns, file history, etc.

Bluefish will work right out of the box, but you can and should take advantage of the many customizations available. Change the font in the main text view if you do not like it, remove unused tool bars, add shortcuts to the customizable menu, and edit the list of browsers and external programs.

If you are upgrading from a previous version, perhaps CVS, you should note that the syntax highlighting may have changed. To make sure you have the latest highlighting patterns, follow the following procedure:

Procedure II.1. Getting the new defaults after upgrading - First method

1. Exit Bluefish
2. Delete the `highlighting` file in your `~/ .bluefish` directory.

Next time Bluefish is started, the new defaults will be loaded.

Note that this will also annihilate all your changes to the highlighting. Here's a more gentle approach:

Procedure II.2. Getting the new defaults after upgrading - Second method

1. Exit Bluefish
2. Move your current `highlighting` file to `highlighting.old`
3. Start Bluefish to get the new patterns
4. Exit Bluefish
5. Run `diff -c highlighting.old highlighting` to find the differences.

If your settings become corrupted, unusable, or you simply want to revert to the defaults, you may safely delete the `~/ .bluefish` directory.

¹If you want to contribute a description on how to install Bluefish on your system, just drop us a note. :-)

Part 2. Using Bluefish

In this part, most of the functionalities of Bluefish are described. What you can do, how you do it, and how you can customize the default behavior.

[Chapter III, *Starting Bluefish* \[15\]](#)

[Chapter IV, *The user interface* \[17\]](#)

[Chapter V, *Working with files and folders* \[23\]](#)

[Chapter VI, *Navigation and Managing documents* \[45\]](#)

[Chapter VII, *More than a Text Editor* \[59\]](#)

[Chapter VIII, *Customising Bluefish* \[81\]](#)

[Chapter IX, *Shortcuts available within Bluefish* \[93\]](#)

Chapter III. Starting Bluefish

Here are the various ways of starting Bluefish.

1. Launching Bluefish

In GNOME, Bluefish can be started from the Applications/Programming menu. From a terminal, simply launch bluefish using the command **bluefish**.



In GNOME, bluefish is launched conforming to the system language. If you want to launch it with another language, first rename the directory `$HOME/.bluefish` to `$HOME/.bluefish-xx`, where `xx` is the previous language. This way you will retrieve your previous settings easily.

Then use this command in a terminal: **export LANGUAGE=fr_FR; export LANG=fr_FR; export LC_ALL=fr_FR; bluefish &** if you use a bash-like shell.

For French users, if you want to get the custom menu in French language, launch it with the same command as above. If you run bluefish in GNOME, launch it with this command the first time after installation, thereafter you may launch it via the Applications/Programming menu.

2. Command line options

There are several useful command line options:

- s
skip root check
- v
display the current version
- n
open a new window
- p *filename*
open a project
- h
display this help screen

Many programs like browsers, email clients and file managers can be configured to open files in Bluefish. For example, **bluefish '%s'** will open a file in the current window, **bluefish -n '%s'** will open a file in a new window, and **bluefish -p '%s'** will open a project file.

Chapter IV. The user interface

Here are the main elements of the user interface.

1. Editing Area

The biggest part of the user interface is the editor area. Because Bluefish has a so-called “Multiple Document Interface”, there are actually many editor areas in Bluefish, accessible via the tabs. By default the tabs are on the bottom.

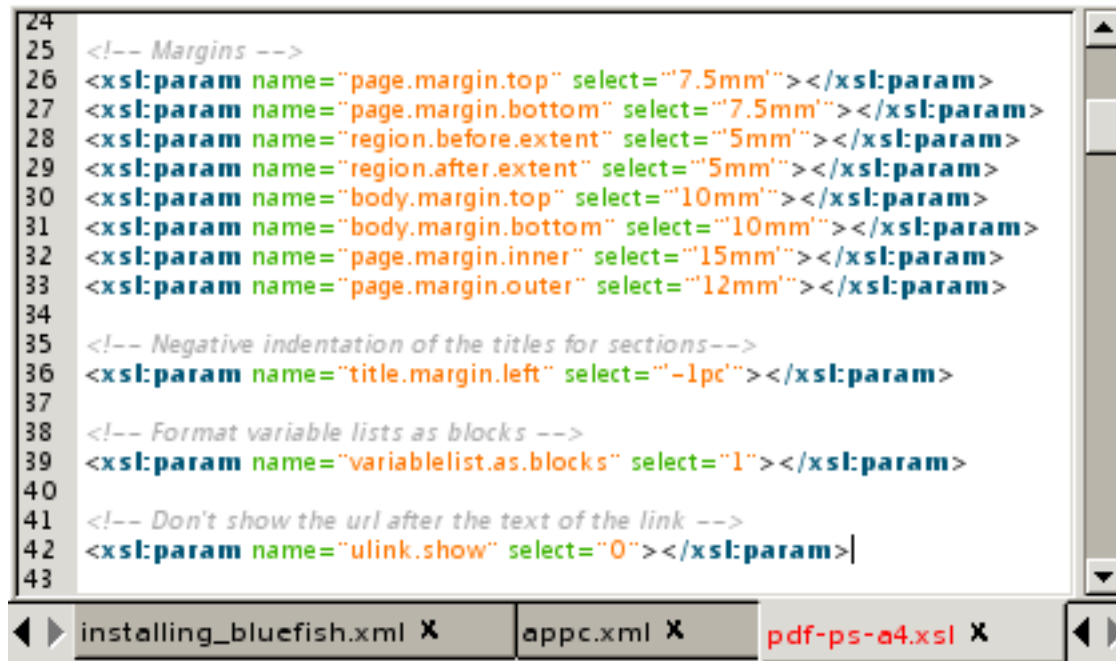


Figure IV.1. Bluefish Editor Array

Notice that the current document's tab may be raised by the Gtk theme, and if the document has been modified, its name on the tab is coloured in red. The changes are also noted with red on the open document list, accessible by right-clicking on the tab.

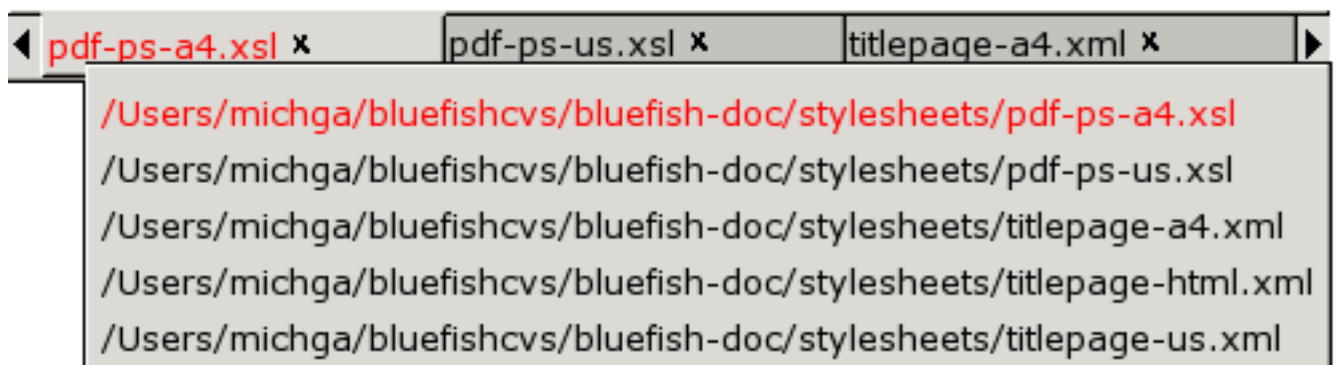


Figure IV.2. List of open Documents

2. Menus and Tool Bars

The top of the Bluefish interface consists of a menu, a main tool bar, an HTML tool bar, and a Custom menu.



Figure IV.3. Bluefish Main Menu

The main tool bar gives you quick access to the basic functionalities of a text editor.



Figure IV.4. Bluefish Main Tool Bar

The HTML tool bar provides access to the most commonly used HTML functionalities.



Figure IV.5. Bluefish HTML Tool Bar

The custom tool bar provides access to languages and replacement functions. It is fully customizable through the preferences panel.



Figure IV.6. Bluefish Custom Tool Bar

3. Side Panel

To the left of the editor area is the side panel. If you would prefer that the side bar be on the right side, simply change the setting in the **User Interface** tab found in the **Edit** → **Preferences** menu option. The side panel consists of a file browser, a function reference browser, and a bookmark browser, accessible by clicking on the corresponding sidebar notebook tab.



Figure IV.7. Bluefish Sidebar Notebook Tab

You may also customize the location of the sidebar notebook tab via **User Interface** tab found in the **Edit** → **Preferences** menu option.

The file browser provides quick access to files and directories.

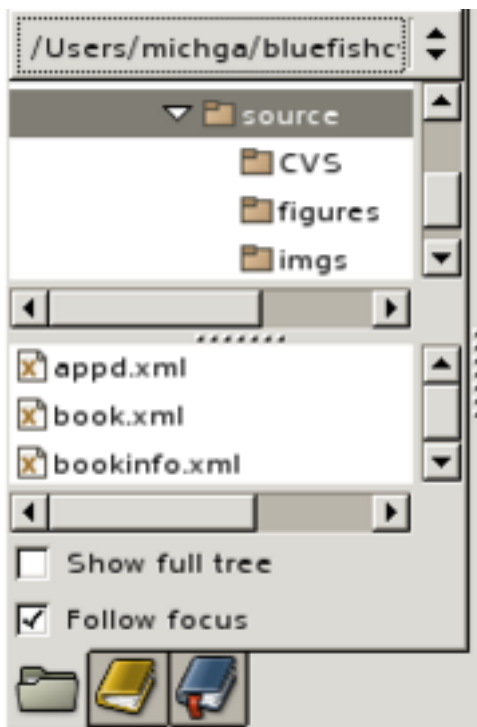


Figure IV.8. Bluefish File Browser

The function reference browser references CSS2, HTML, PHP, and Python functions with their syntax. Some of them provide dialogs to help you inserting them.

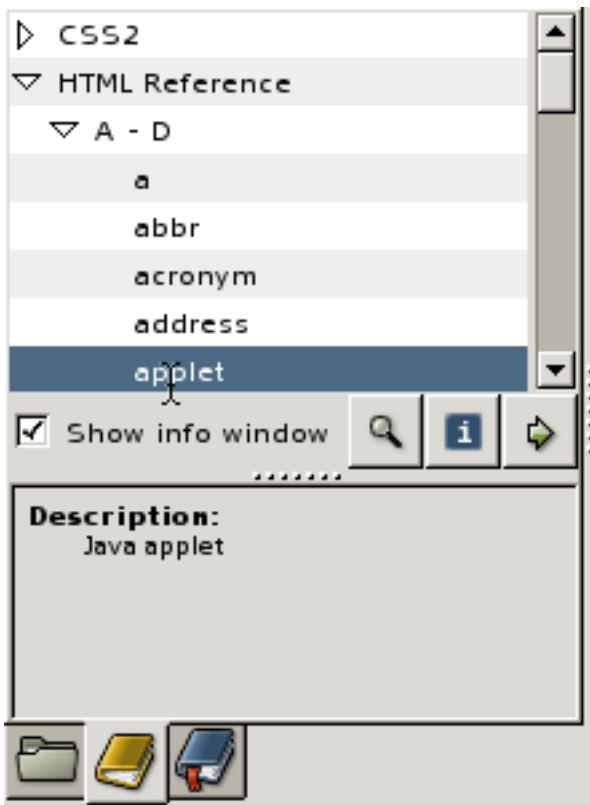


Figure IV.9. Bluefish Function Reference Browser

The bookmark browser provides access to previously marked positions in a file.

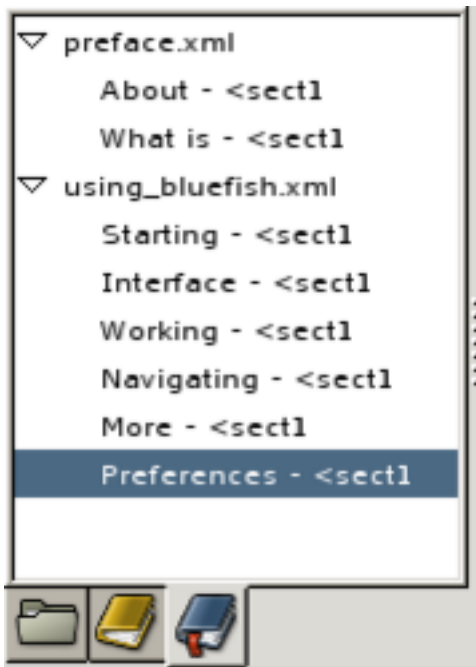


Figure IV.10. Bluefish Bookmark Browser

4. Status Bar

On the bottom of the Bluefish window is the status bar. Shown here are messages, the current line & column number, the insert (INS) or overwrite (OVR) mode for the cursor, and the file type & character encoding.

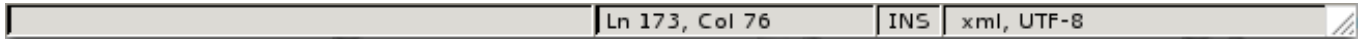


Figure IV.11. Bluefish Status Bar

The visibility of these items can be toggled via the **View** menu.

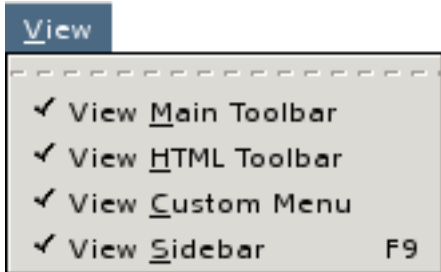


Figure IV.12. Bluefish View Menu

If you want to disable any of these items by default, you can set these options in the preferences under **User interface**.

5. Help Menu

The **Help** menu contains the typical About box. As usual, you will find in it developers', maintainers', and translators' details. Plus the configure flags used to compile Bluefish on your system and the packages detected automatically.

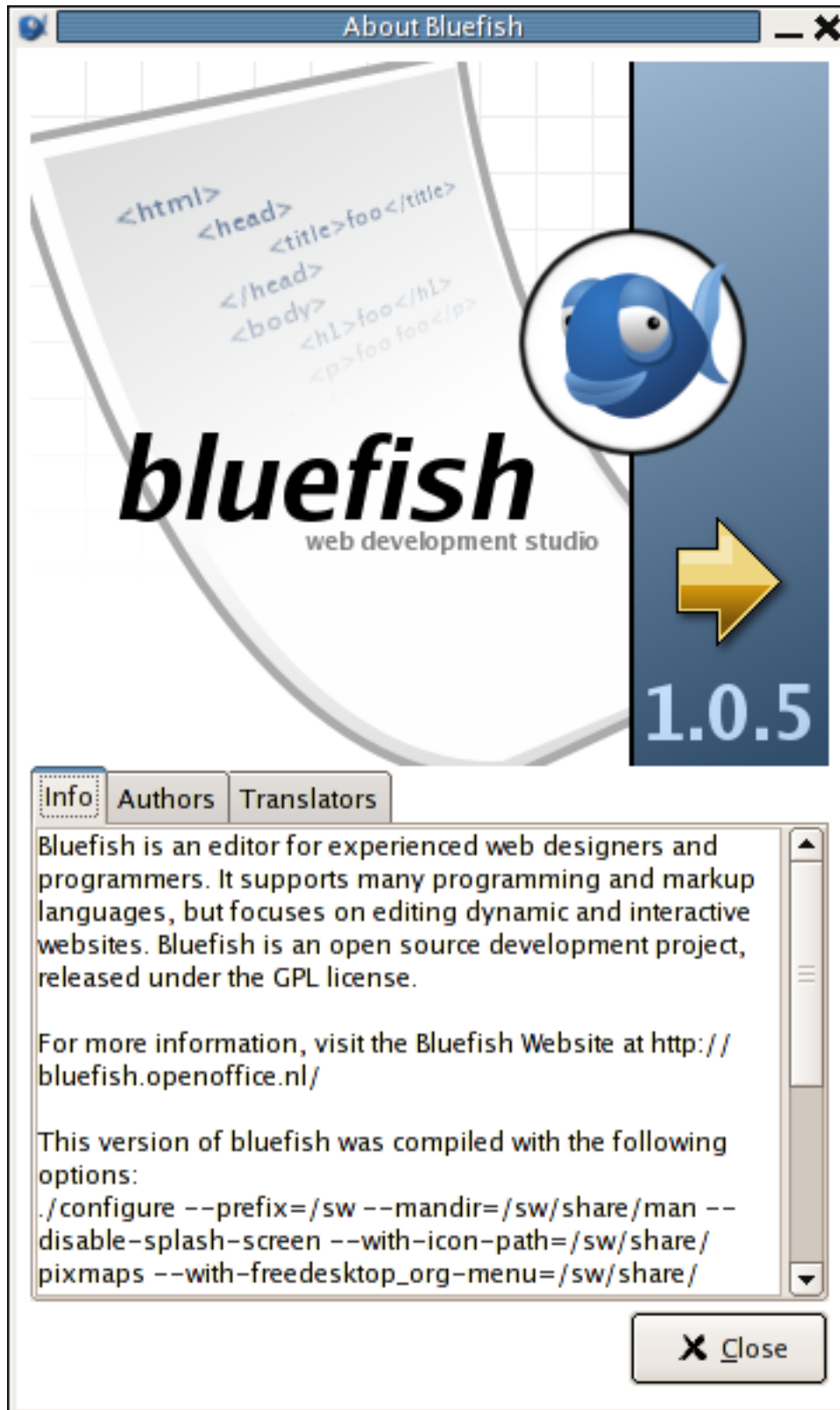


Figure IV.13. Bluefish About Window

The other menus are described in the following sections:

- the **Tags** and **Dialogs** menu in [Section 5, “HTML”](#) [62]
- the **Go** menu in [Section 2, “Navigating through many documents”](#) [45]

- the **Project** menu in [Section 3, “Projects”](#) [46]
- the **External** menu in [Section 8, “External programs, filters”](#) [74]

Chapter V. Working with files and folders

Most of the file operations are accessible from the **File** menu. Using this menu, a new file can be created, existing files opened, and opened files saved or renamed.

You may also insert a file into another one, and revert a modified file to its previously saved state.



Figure V.1. Bluefish File Menu



In case you compiled bluefish from source, the **File** → **Open URL...** menu item is shown only if you installed gnome-vfs2 beforehand.

For Mac OS X users installing via Fink, choose bluefish-gnomevfs2 or bluefish-gnome2 variant.

You may also add directories, delete files, and refresh the file browser in the side panel using its contextual menu.

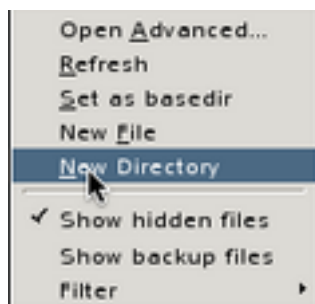


Figure V.2. The File Browser Contextual Menu

You may also open a file using drag and drop from the desktop or from Nautilus to any part of the Bluefish window but the editor part when you are under Gnome2 environment.



If you drag and drop the file to the editor window, the full path of the file is inserted.

1. Creating files

Apart from using **File** → **New** (**Ctrl-N**) or the **New** icon to create a new file, you may also use **File** → **New Window** (**Shift-Ctrl-N**).

Those methods create an untitled file of type text (except when you are working on a project with defined template file, see [Creating a New Project](#) [46] for more info) with default permissions and the default character encoding defined in the **Files** tab in the **Edit** → **Preferences** menu option. You will further have to [save](#) [28] it under the desired name.

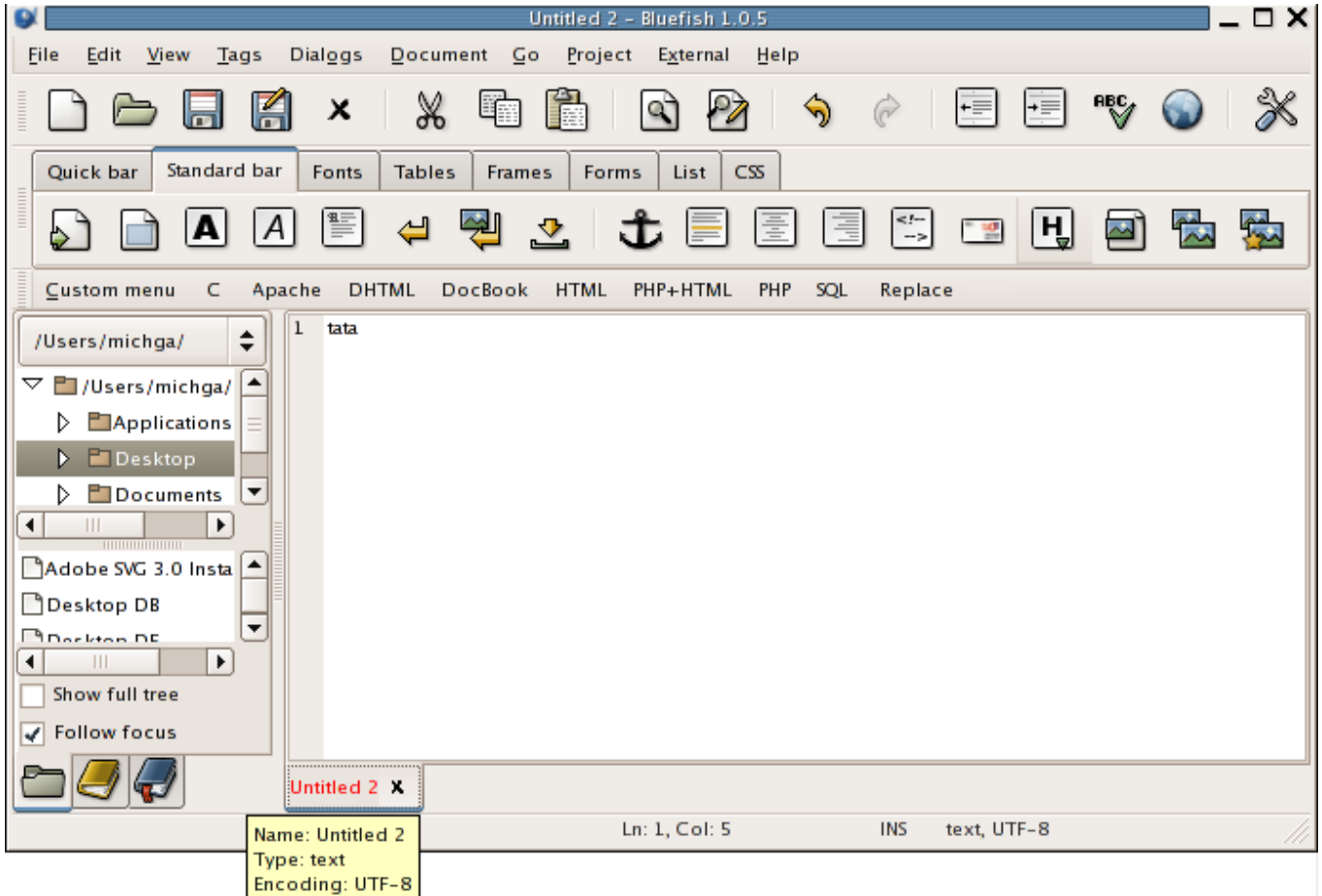


Figure V.3. Untitled Document Window



Note that the document window is titled “Untitled n - Bluefish versionnumber”, which indicates precisely that the file is not saved on the disk. The same applies to the document's tab which shows “Untitled n”. Moreover, if you move the cursor onto the document's tab, you will not see any size or modification date.



When creating new files, you may want to create them in a new window. In this case, use **File** → **New Window** to first open the new window and then create the desired files as usual.

To spare yourself the bother of saving, right click on the desired directory in the directory list of the file browser in the side panel and select **New File**. You will be presented with a **File name** dialog, where you will enter the desired name:

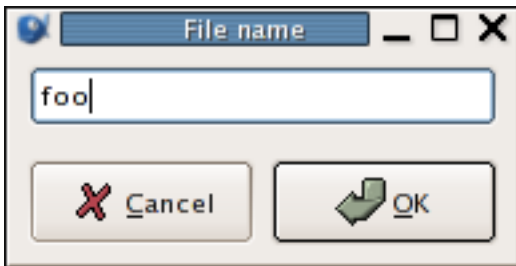


Figure V.4. The File Name Dialog



You may type a path relative to the clicked directory, provided that the target directory be under the clicked directory and that it exists.

2. Managing directories

To create a new directory, right click on the desired parent directory in the directory list of the file browser in the side panel and select **New Directory**. You may also right click on any file in the files part of the file browser.

Then enter the desired name in the **Directory name** dialog. Bluefish will create the named directory with the default permissions under the currently selected directory.

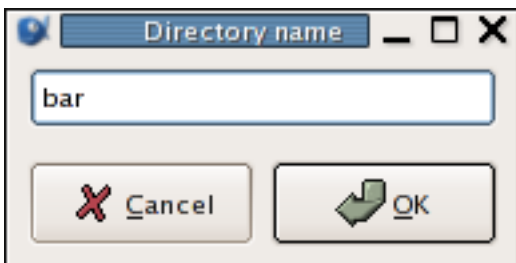


Figure V.5. The Directory Name Dialog

Note that you cannot delete directories within Bluefish, but you can refresh the view with the **Refresh** contextual menu item of the directory part of the file browser if you delete them externally. This is especially useful when you add files and directories or delete files.

If you right click a directory in the file browser within the side panel, you can make this directory the base directory for the file browser using the **Set as basedir** option. Then you can access it directly from the pop up menu in the upper part of the file browser.

By default the file browser follows the document focus. If you change to a different document, the file browser will show the contents of the directory where this document is located. This behaviour can be changed by unchecking **Follow focus** on the bottom of the file browser.

3. Opening files

Through the **File** → **Open...** (**Ctrl-O**) menu item or the **Open...** icon in the main tool bar, one or more files can be opened. If you want to open them in a new window, use **File** → **New Window** to first open the new window and then open the desired files as usual.

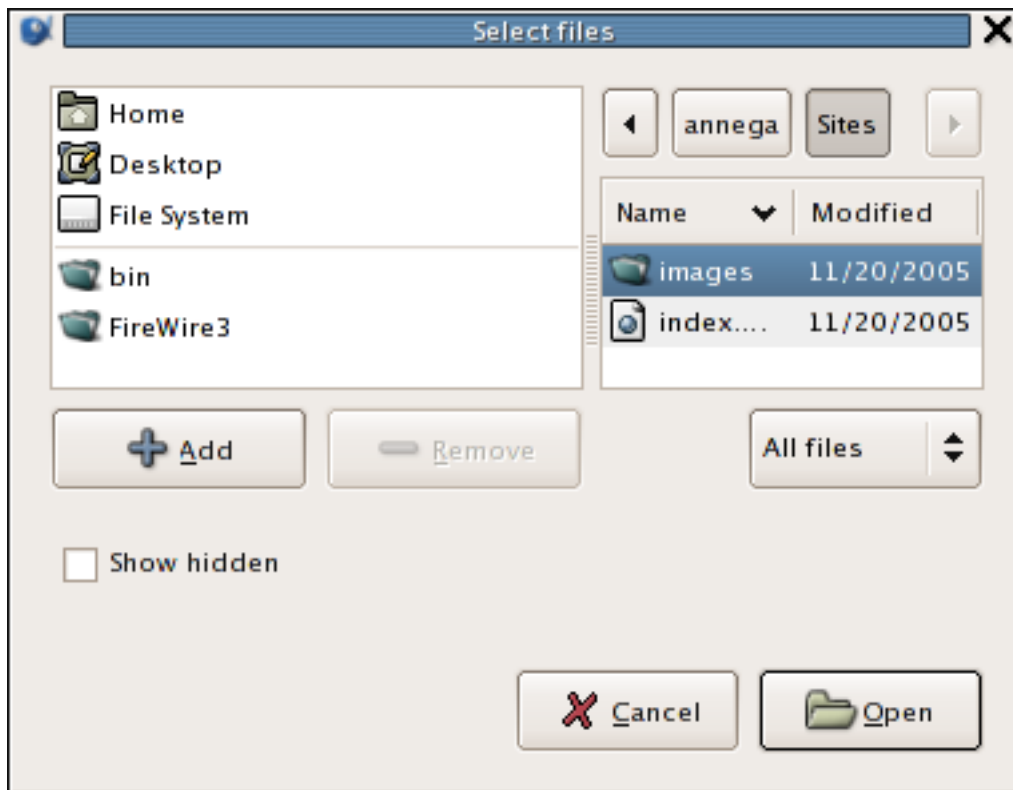


Figure V.6. Bluefish Open File Dialog



The most recently opened directories appear in the upper part of the side panel, while the lower part contains user-defined locations. To add a new directory to the list, click on **Add**. You can also filter the file list by file type using the pop menu located on the right side. The list of file types in the filter menu is provided through the **Filetypes** tab found in Bluefish's **Edit** → **Preferences** menu option.

Recently opened files can be opened by selecting them from the list within **File** → **Open recent**. The number of files in this menu can be set in the preferences under **Files**.

The contextual menus of both parts of the file browser in the side panel can also be used to open files. The file part supports opening a given file by right clicking it, and filtering files via the **Filter** submenu of its contextual menu. The directory part supports filtering files via the **Filter** submenu of its contextual menu, and advanced opening explained in the [Section 12.3, “Open advanced”](#) [43]

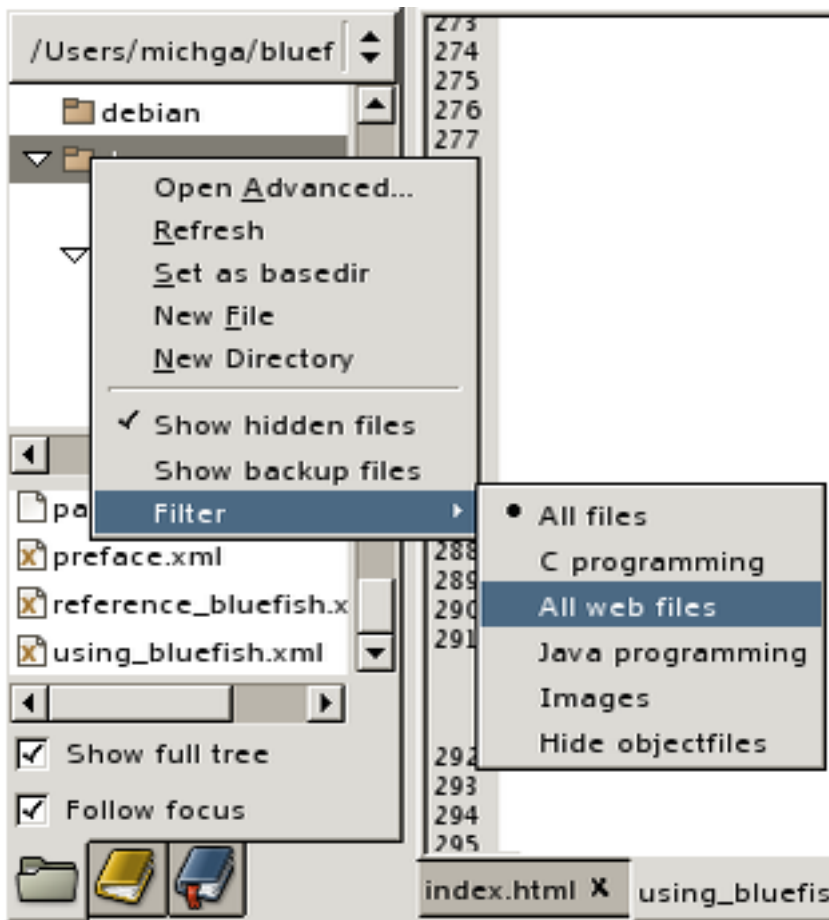


Figure V.7. Filtering Files with the Bluefish File Browser

The available filters may be modified in Preferences. For more information, see [Section 11, “Modifying the files filters” \[88\]](#)

An interesting feature of Bluefish is the ability to open files by selecting the text of a currently opened file in another application. For example, if a filename is shown in say a terminal application, you can select the filename, and use **File** → **Open from Selection** to open that file. The file, if it exists, will be opened in another tab within Bluefish.



The selected text should match the full path of an existing file.

Files can also be opened via the command line by feeding filenames to Bluefish as arguments. For example:

```
$bluefish ~/foo1.txt ~/foo2.txt
```

This can even be done while Bluefish is running and the resulting files will then show up in their own tabs.



If you have installed `gnome-vfs` or `gnome-vfs2` before installing Bluefish, you will be able to open files on remote desktop via the **File** → **Open URL...** menu item. See more on this in the [Section 12.1, “Remote files” \[42\]](#)

Finally, if the `gnome2` environment is installed on your system, you may open files by drag and drop from the desktop or the Nautilus file manager onto any part of the Bluefish interface, but the document's editor.

Be aware that if the file is huge it may take a very long time to get the rendering if syntax highlighting is enabled in the **Editor** option of Preferences, in particular if the PCRE UTF-8 support is enabled. The GTK editing widget used in Bluefish, furthermore, is not very good at handling files with very long lines, and that could also slow down Bluefish considerably.

Information about currently opened files can be seen if you move the mouse over the document tab (by default on the bottom of the screen). A so called tool tip will be shown with information about the full path, size, permissions, file type and encoding of the file.

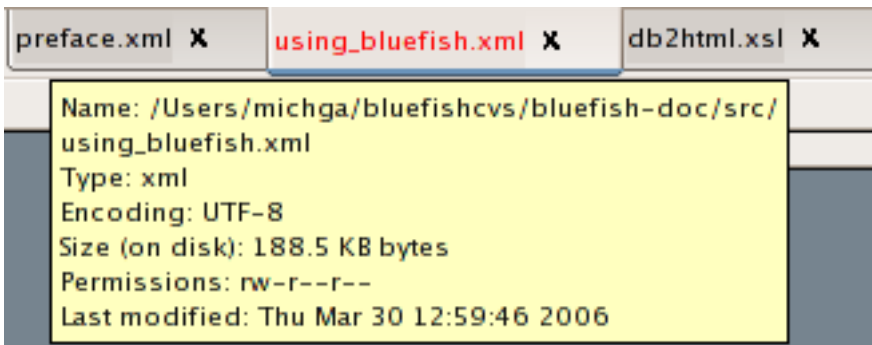


Figure V.8. Info on open File



Note that the document's tab is highlighted in red, because the document has been modified since the last save. Info in the tool tip match the state of the file as it was saved on disk.

4. Saving files

To save a document, you can use the **File** menu, the **Save** icon in the tool bar, or press the shortcut key combination **Ctrl-S**.

If the file does not exist already on the disk, you will be presented with the following dialog window:

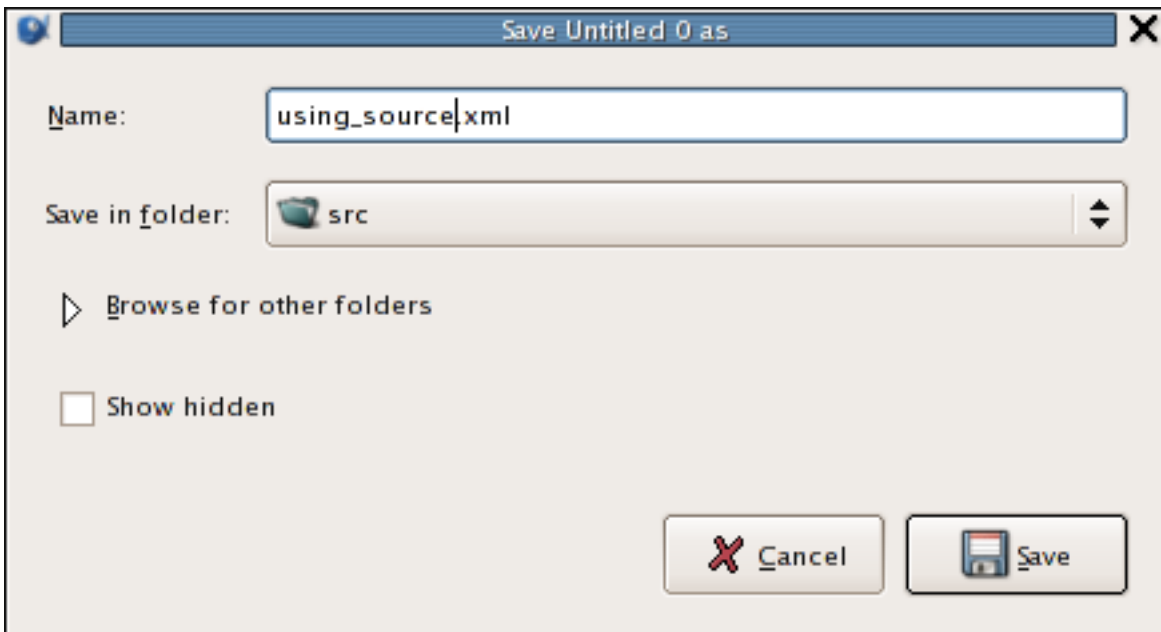


Figure V.9. Saving an untitled File

If the file already exists on the disk, no dialog window pops up. It is assumed that the file name does not change.

When a document has been modified, the filename is shown in red in the document tabs; moreover if you right click on the tabs, the full path is shown in red in the list that will pop up.



Figure V.10. A modified Document in the Documents' List

By default a backup is made during save. The original file is copied to the same filename with a tilde ~ appended. This suffix and the backup behaviour can be changed in the preferences under **Files**.

Before saving the file, Bluefish will check if the original file was changed on disk, using the last modified time and the file size. On some file systems the last modified time is sometimes not very precise (most notably on samba mounts). This makes Bluefish think the file is modified when it is not. This check can be changed in the preferences under **Files**.

You can also save a document under a different name, using the **Save As...** (**Shift-Ctrl-S**) menu entry, or the **Save As...** icon in the main tool bar. The original file will still exist.

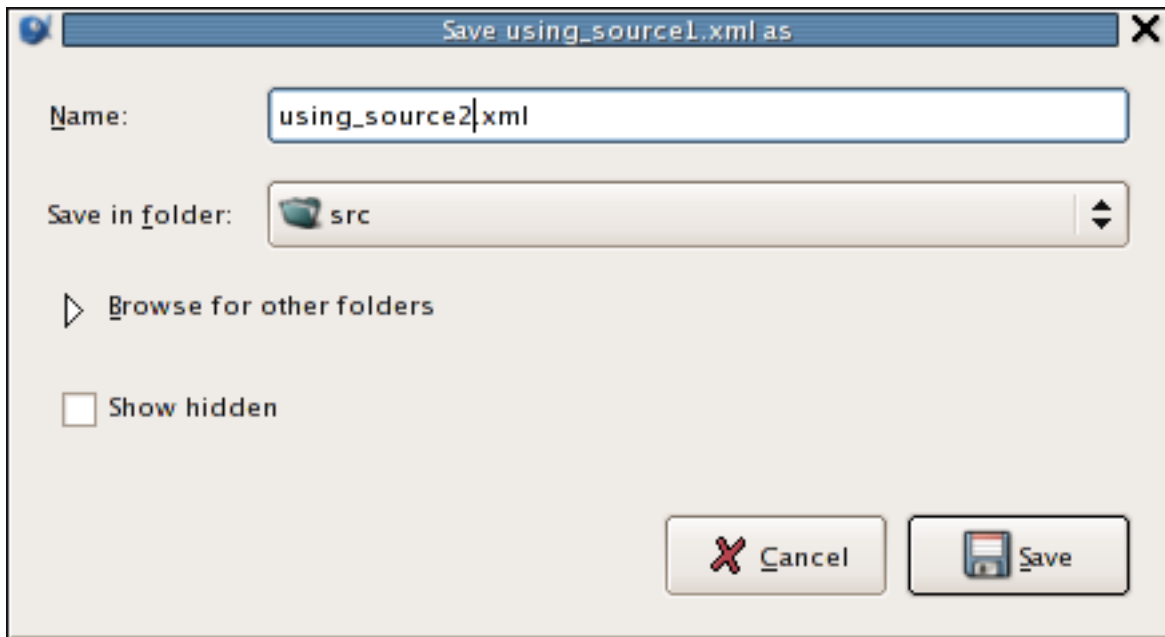


Figure V.11. Saving a File under a new Name

To save all modified files, you can use the **File** → **Save All** menu entry. This will save all documents that have been modified and present you with a save dialog if some files are new files.

5. Renaming files

It is also possible to move or rename a document, using the **File** → **Rename...** (**F2**) menu item, or right-clicking the file name in the side panel and choosing the **Rename** item.

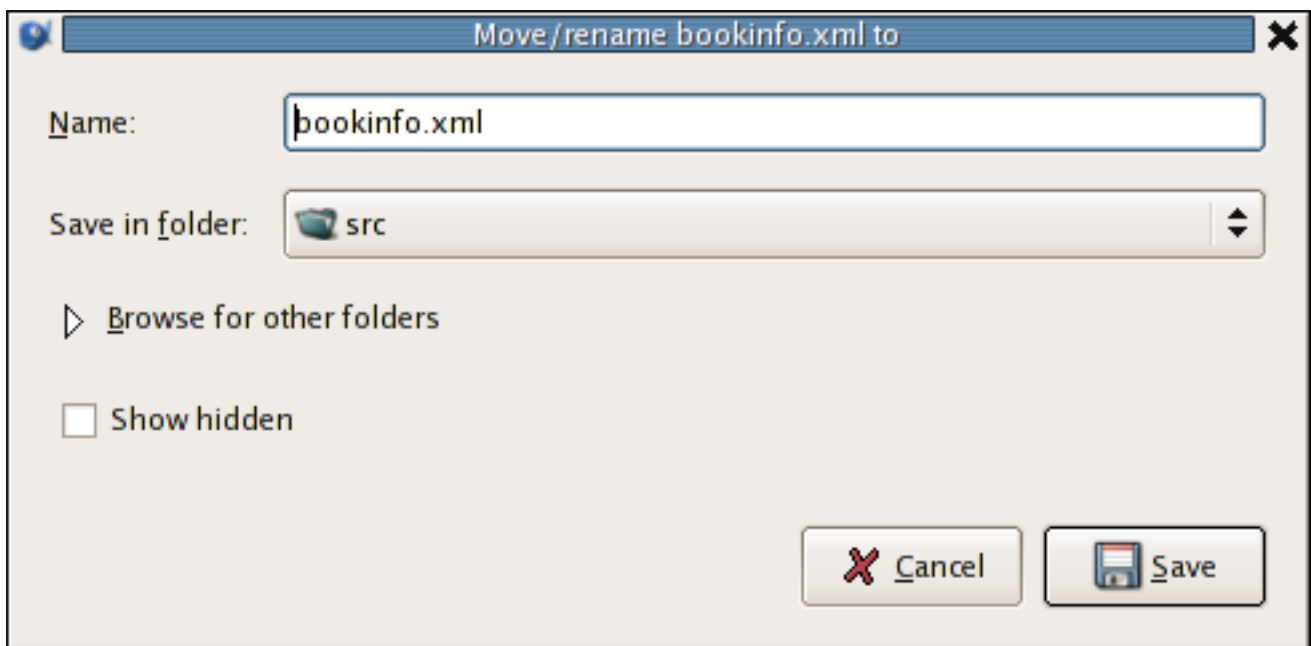


Figure V.12. Moving a File to another Location

6. Deleting files

To delete a file, right click on it in the file browser within the side panel. You will be asked either to confirm the deletion or to cancel the process, if you have the right permissions for it.



Be cautious with this feature, there is no easy way to recover the deleted file.

Mac OS X users: this command is not equivalent to putting the file in the trash, but to using the **rm** command, so that, apart using a specialized commercial tool, you cannot recover the file.

Combining this feature with the **Show hidden files** feature accessible via the contextual menu of the Filebrowser side panel allows you to delete hidden files more easily than via the **rm** tool on some systems.

Combining it with the **Rename** feature accessible via the contextual menu of the Filebrowser side panel allows you to delete an original file once you have edited a new file based on the former one, and to give the new file the same name as the original one.



The first safe usage I can find for this feature is, for example, to remove the backup `.#filename.version` files created by CVS during conflicts after resolving them, particularly on Mac OS X where those files are tricky to access.

A second usage would be, for example, to save a po file again under its original name after updating it via a shell script under a new name, and editing it manually.

7. Closing files

When you want to close a file quickly, click on the **close** icon in the document tab. You may also use the **Close** icon in the main tool bar, or the **File** → **Close** (**Ctrl-W**) menu item.



Figure V.13. Closing a File with the Document Tab Icon

If the file is unchanged, it is merely closed. If the file has been modified, you will be presented with a save dialog.

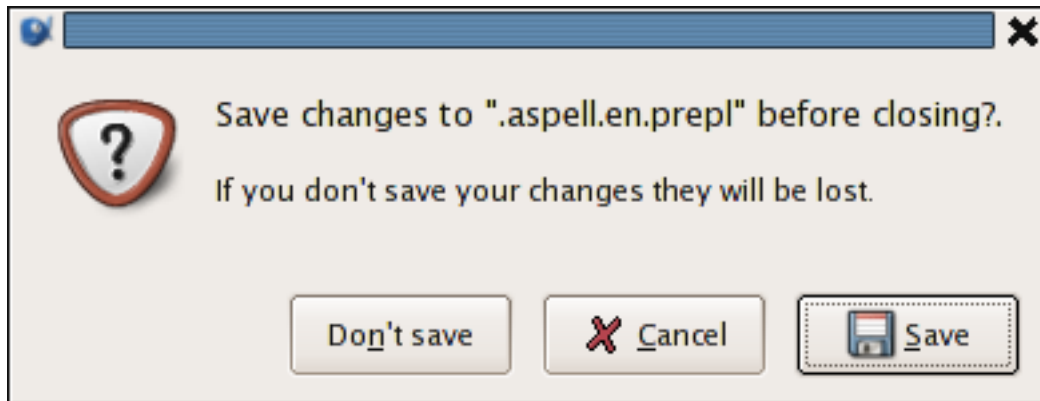


Figure V.14. Closing a modified File



Use it to save and close a file in one step.

When dealing with multiple files, you may want to use the **File** → **Close All** (**Shift-Ctrl-W**) menu item.

For each modified file, you will be presented with a save dialog, where you can choose to save the changes, close the file (i.e. discarding any change), or cancel the operation.

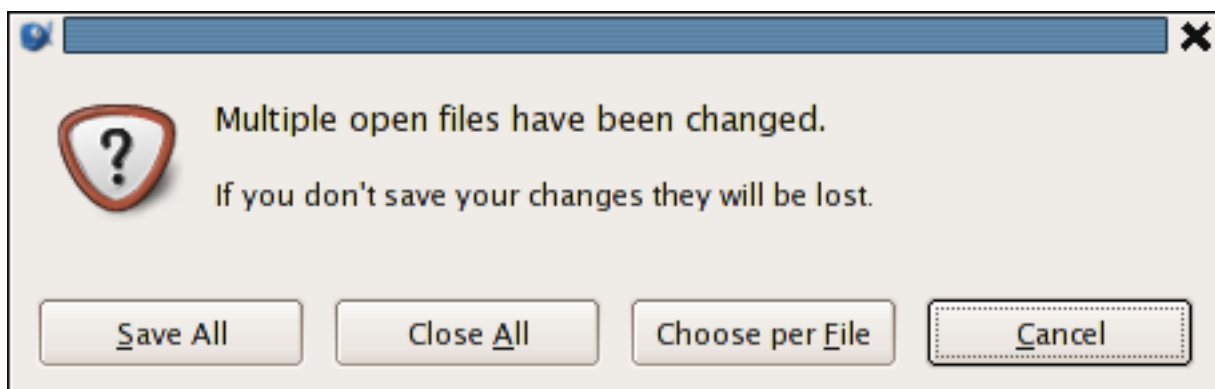


Figure V.15. Closing all Files



Say you have a number of open files, and only a few of them have been changed. To quickly close the unchanged files, and remain with the modified ones, use it answering cancel for the latter ones.

Note that the **File** → **Close Window** menu item offers the same behaviour.

8. Inserting files

You can insert any file into the current document with the **File** → **Insert...** menu item.

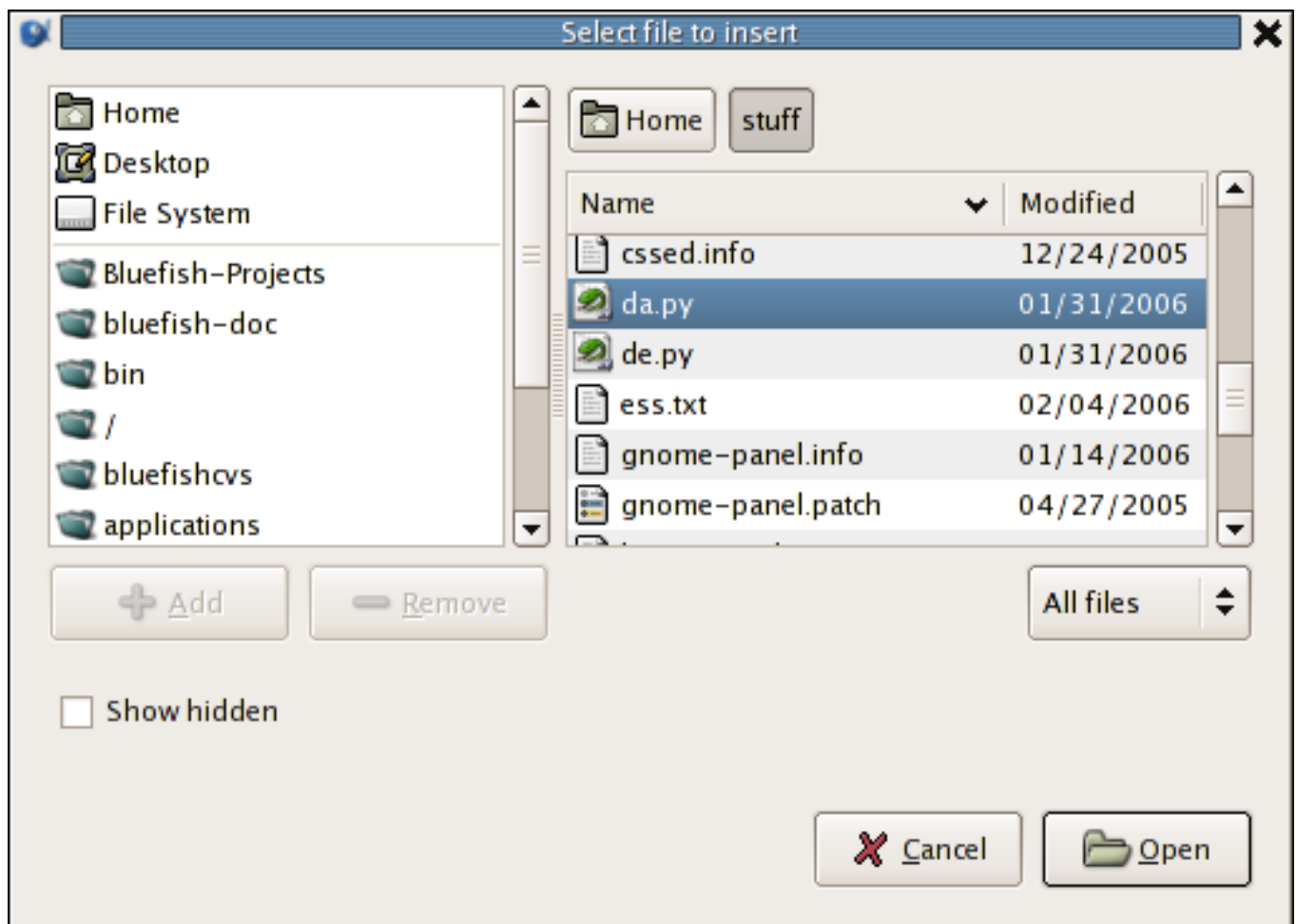


Figure V.16. Inserting a File

The file will be inserted at the cursor location.



Always save the file before inserting another one, this will prevent any potential loss of data.

For more in-depth information about dealing with files, see [Section 12, “More on files”](#) [42]

9. Editing

9.1. Undo and Redo

The undo and redo functionalities are available from the **Edit** menu, the main tool bar, and the keyboard shortcuts.

- **Undo** (**Ctrl-Z**)
- **Redo** (**Shift-Ctrl-Z**)

The functions **Undo All** and **Redo All** in the **Edit** menu will undo or redo all of the stored changes. The maximum number of changes can be configured in the preferences; by default Bluefish will remember the last 100 changes per document. It is possible to clear the changes after the document is saved by unchecking the **Clear undo history on save** check box in the **Editor** tab of Preferences.



Always perform a save on file before making important modifications. This will prevent any loss of data during undoing.

9.2. Cut, Copy, and Paste

The functions **Cut**, **Copy**, and **Paste** are available from the **Edit** menu, the main tool bar, and the keyboard shortcuts.

- **Cut (Ctrl-X)**
- **Copy (Ctrl-C)**
- **Paste (Ctrl-V)**

On X Windows Systems, you can also paste the current selected text using the middle mouse button. First select some text (in Bluefish or in any other X application), then press the middle mouse button where you want to paste the selected text.

Cut or copy and then paste can also be done by selecting some text and dragging it to the destination. If the text is dragged to another document (or another application), it is copied. If the text is dragged within one document it is moved. Dragging highlighted text from one application to another may or may not work. However, most GNOME and GTK programs support this feature.

9.3. Input methods

Bluefish handles a number of input methods, available from the contextual menu within a given document.

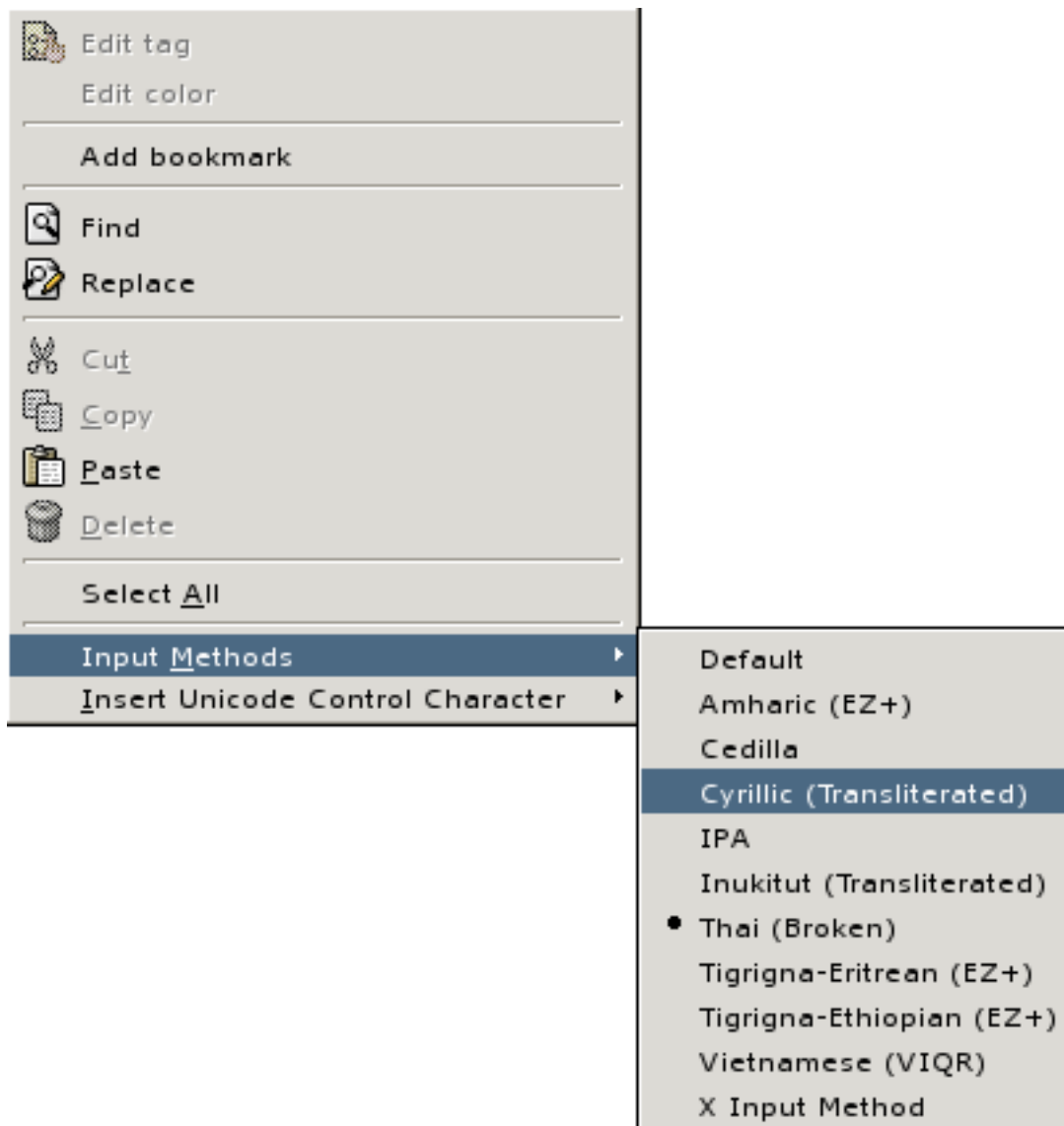


Figure V.17. The Input Methods Contextual Menu

- The default mode switches all input methods off.
- The Amharic mode is used for the most popular Ethiopian language.
- The Cedilla mode is used for languages such as French, which uses the cedilla.
- The Cyrillic mode is used to enter Russian with Roman letters. The transliteration occurs immediately.
- The Inukitut mode works the same as Cyrillic mode.
- The IPA mode is used for International phonetic alphabet.

- Other modes are used for Erythrean, Ethiopian, Thai and Vietnamese languages.

The X Input method relies on a client-server input system, and an input server.

For Japanese, Chinese, and Korean documents, you may have to install and launch the correct input system, such as canna, and the appropriate input server, such as kinput2.

Here is how to write a Japanese document on a non-Japanese system.

Procedure V.1. Writing in Japanese with Bluefish on a non-Japanese system

1. Launch the canna server if it is not running already
2. Set the encoding to Japanese, for example: `export LANG=ja_JP.UTF-8`
3. Set the Xinput method with `export XMODIFIERS="@im=kinput2"`
4. Launch kinput2 as a background process with `kinput2 &`
5. Launch bluefish as a background process with `bluefish &`
6. To activate the Xinput method within bluefish, use **Shift-Space**. A small window with a Japanese glyph will appear at one of the corner of the Bluefish window. Once the desired glyph has been composed, press **Space**, and hit **enter** to validate it.

Here, you can see the small Xinput method window, at the bottom left corner of the window and the first Japanese word not already validated in the Bluefish window launched on a French system.

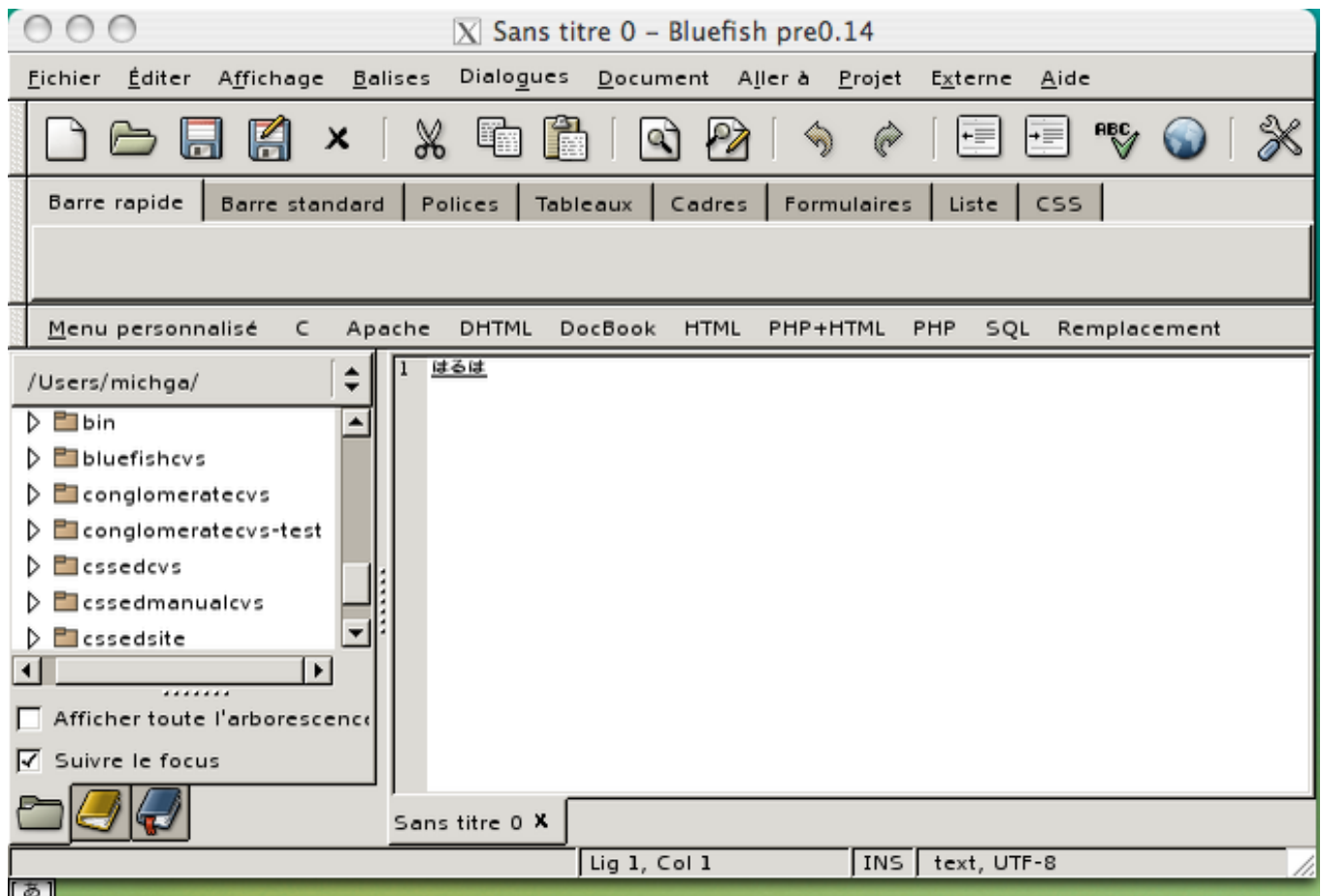


Figure V.18. Writing in Japanese with Bluefish

For an in-depth discussion on that subject, see [Inputting from the keyboard](http://www.jw-stumpel.nl/stestu.html#T6) [http://www.jw-stumpel.nl/stestu.html#T6].

10. Basic Find and Replace

Bluefish offers a wide range of find and replace methods in the **Edit** menu, also available through the contextual menu within a document. Here we will explore the most basic ones. For advanced find and replace methods, see [Section 5, “Find and Replace”](#) [53]

10.1. Searching for a word within a whole document

Choose the **Edit** → **Find...** (**Ctrl-F**) menu item. A **Find** dialog will be displayed. Enter the word to search for in the **Search for:** field. Then click **OK**.

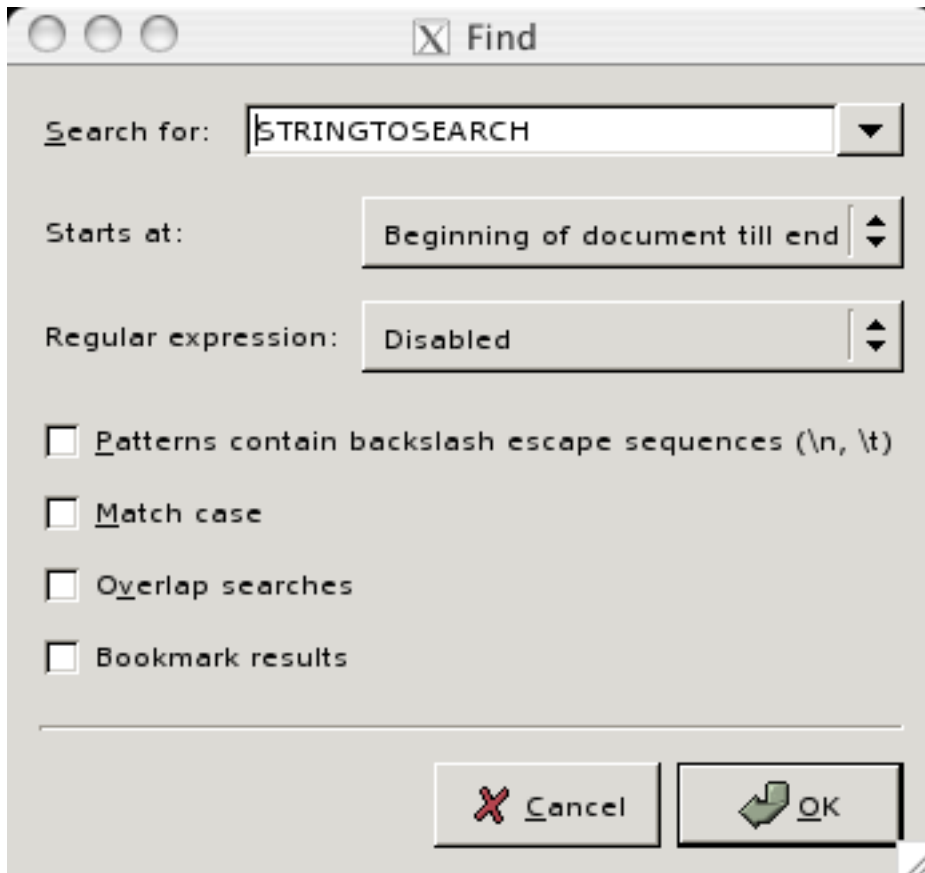


Figure V.19. Finding a Word in a Document, from Start to End

If the word does not exist in the document, a small window pops up.



Figure V.20. Unsuccessful Search Window

If the search is successful, the document window scrolls up to the first occurrence of the string in the document and highlights it.

Below is an example of a search applied to a shell script.

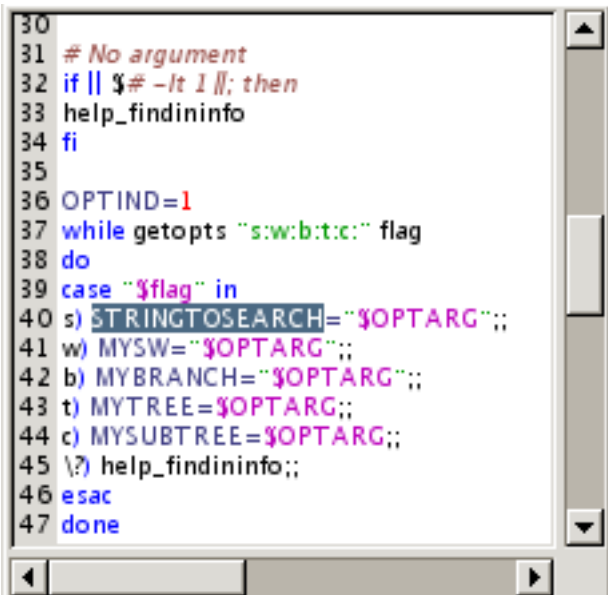


Figure V.21. Highlighted Search Result in the Document Window

To find a subsequent occurrence of the string, use the **Edit** → **Find again (Ctrl-G)** menu item. If no further occurrence is found, a dialog will be displayed notifying you that no match was found.

10.2. Setting limits to the search scope

You may want to search for a string from the cursor location till the end of the document. Here is an example to search all `name ==` occurrences within a python script from a given location.

Procedure V.2. Searching from selection

1. Put the cursor where you want to start the search from in the document window

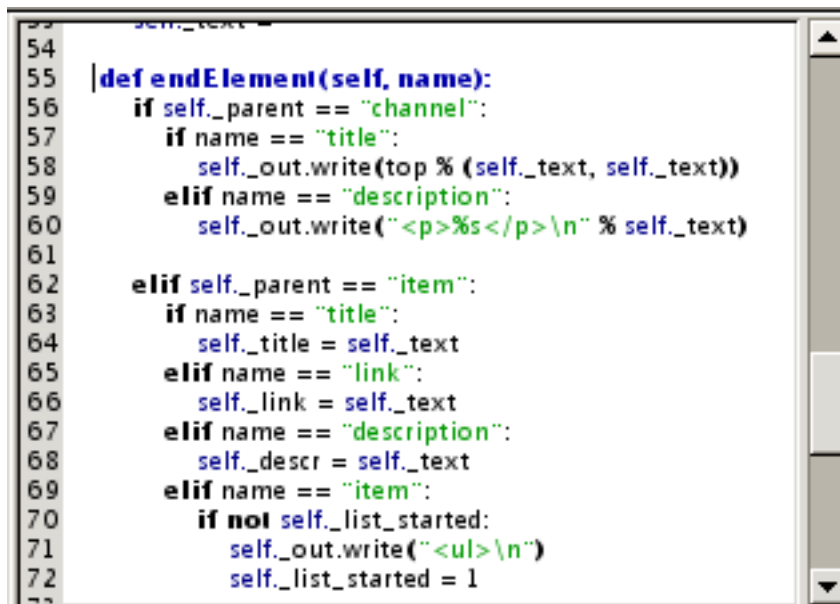


Figure V.22. Setting the Cursor Location

2. Open the **Find...** dialog
3. Enter your search string in the **Search for:** field
4. Choose **Current position till end** from the **Starts at:** pop up menu

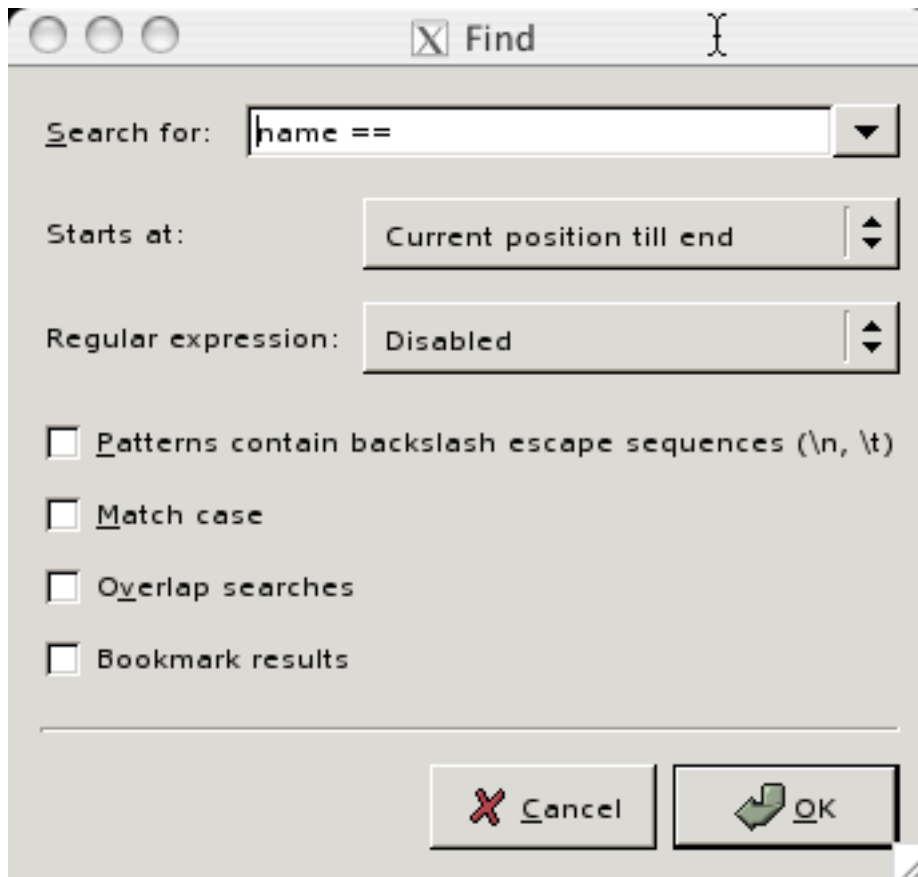


Figure V.23. Choosing a limited Search Method

5. Click OK.

Here is the result:

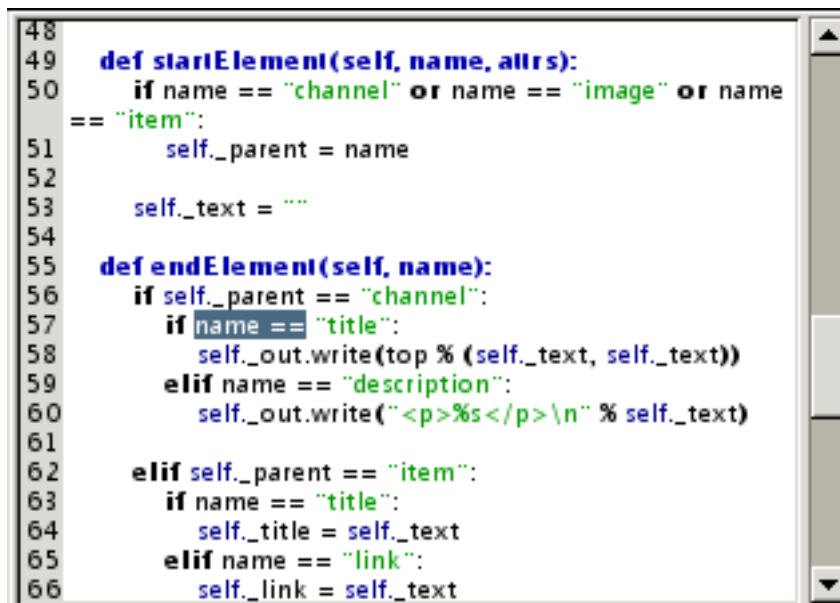


Figure V.24. Limited Search Result

Notice that the search does not take into account the occurrence of the same string at line 50, since it is outside the search scope.

You can also limit the search scope to a selection range. In that case, highlight the selection before the search, and choose **Beginning of selection till end of selection** from the **Starts at:** pop up menu in the **Find** dialog.

10.3. Case sensitive search

By default, the search process is case insensitive. If you want to make it case sensitive, just check the **Match case** box in the Find dialog.

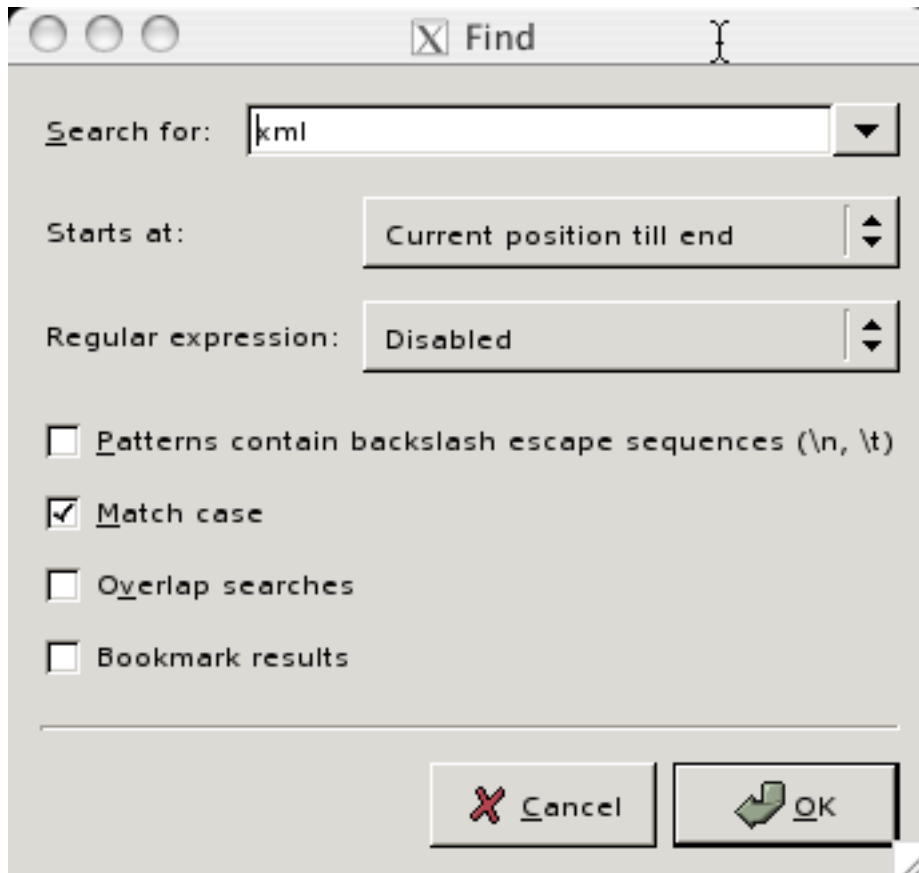


Figure V.25. Making the Search Case Sensitive

Here is the result applied to a ruby script:

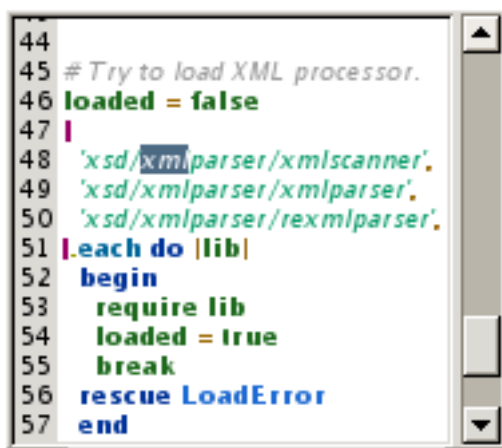


Figure V.26. Case Sensitive Search Result

Notice again that the result does not catch the XML string at line 45, since the search string was xml and case sensitive search was requested.

10.4. Overlapping searches

It may occur that the document contains some kind of palindrome you want to search for. The "normal" find process does not retrieve all occurrences of that kind of string.

In this case, you have to check the **Overlap searches** box in the Find dialog to retrieve all occurrences of the string.

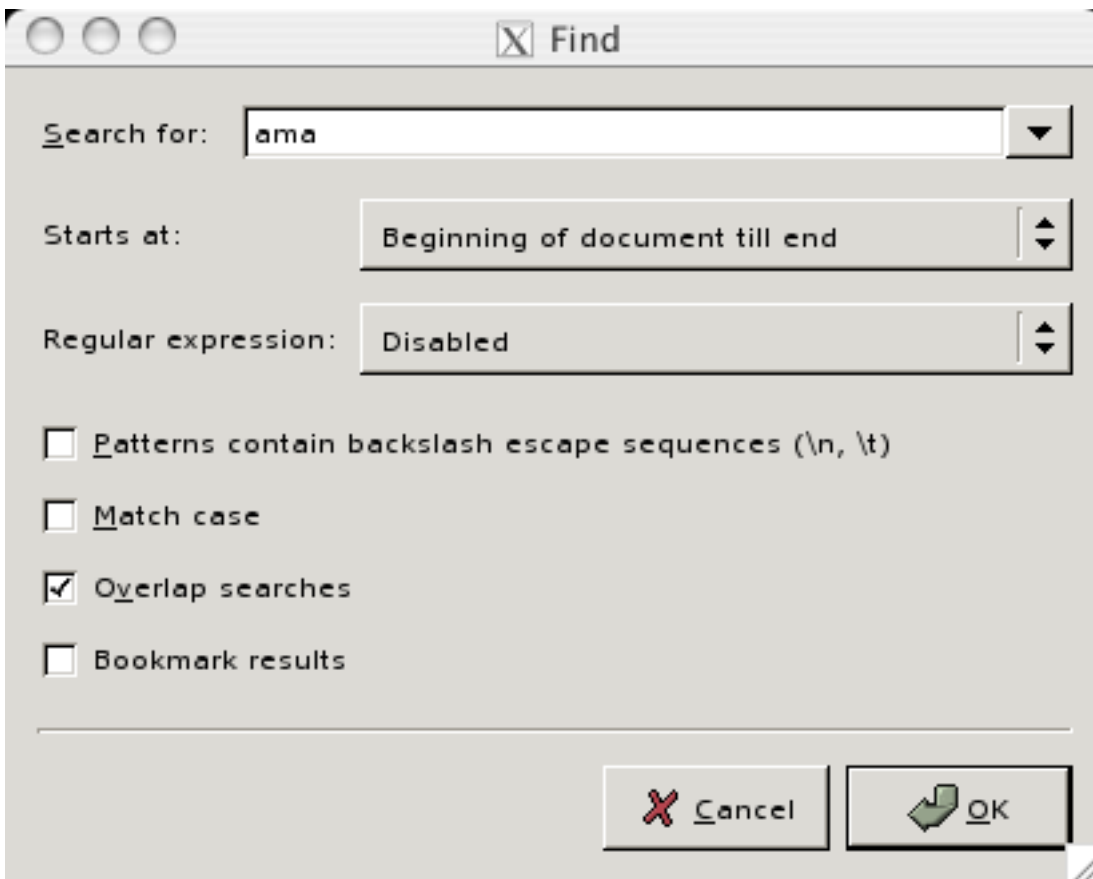


Figure V.27. Finding overlapping Strings

Applied to a shell script, the second search (with **Ctrl-F**, then **Ctrl-G**) will give the following result:

```

1  #!/bin/sh
2  #
3  # Script to install manpages in corresponding
4  # LANG directories via Fink on Mac
5  # Modified to give an example of palindrome
6  #
7  MANLANGS=.:de:fr
8  export MANLANGS
9  export amamadirman=/sw/share/man
10 for d in $(MANLANGS);
11 do amamaldirman=$(amamadirman)/$d/man1;
12 echo $(amamaldirman);
13 done;
14

```

Figure V.28. An overlapping String retrieved with the Find Dialog

10.5. Retrieving previous search strings

Notice that the pop up menu to the right of the **Search for** field in the **Find** dialog allows you to retrieve previous search strings. They are listed in reverse order by search history, providing quicker access to the most recent searches.

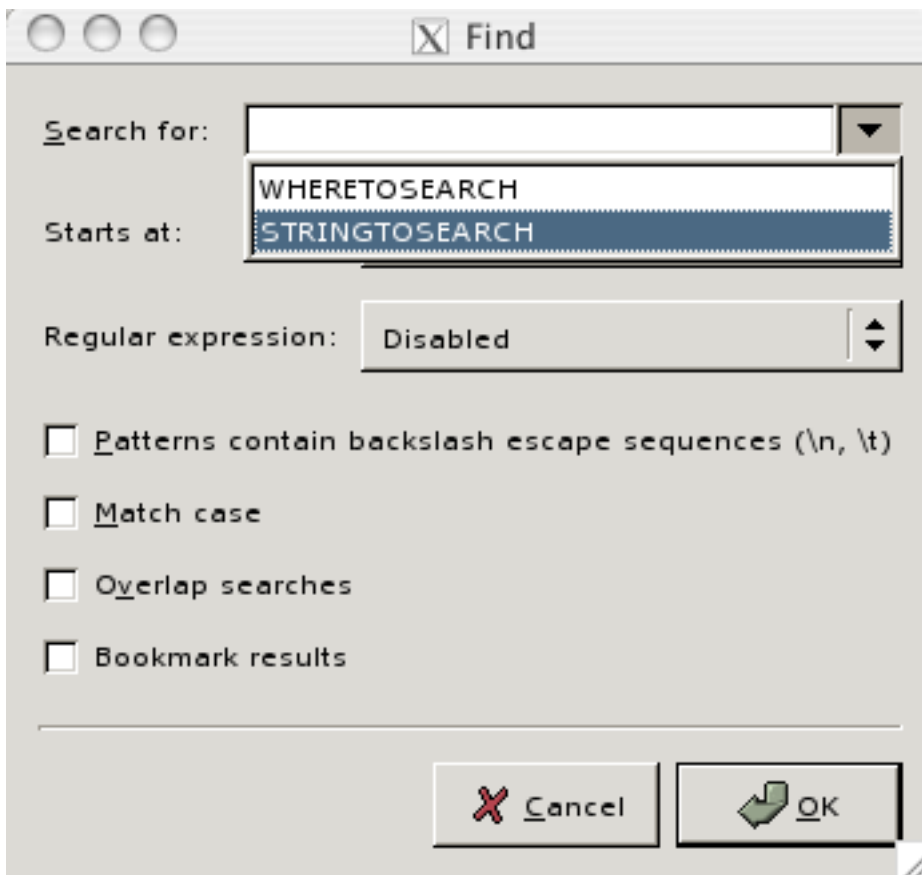


Figure V.29. Retrieving recent Searches

10.6. More on find

For an explanation of the **Bookmark results** box of the **Find** dialog, see [Section 4.1, “Generating several bookmarks at once”](#) [?].

You will find details on **Find Again** and **Find from Selection** in [Section 5, “Find and Replace”](#) [53]

For a quick way of switching from HTML entities to other types of encoding and changing letter cases, see [Section 5.1, “Special find and replace features”](#) [63]

10.7. Replacing features

The **Edit → Replace...** (**Ctrl-H**) menu item works the same way and has all the features, the **Edit → Find...** (**Ctrl-F**) menu item offers.

The **Replace** dialog is also accessible through the contextual menu within a document.

For the features common to the **Find** dialog, see [Section 10.1, “Searching for a word within a whole document”](#) [34].

Here we will explain the features unique to the **Replace** dialog.

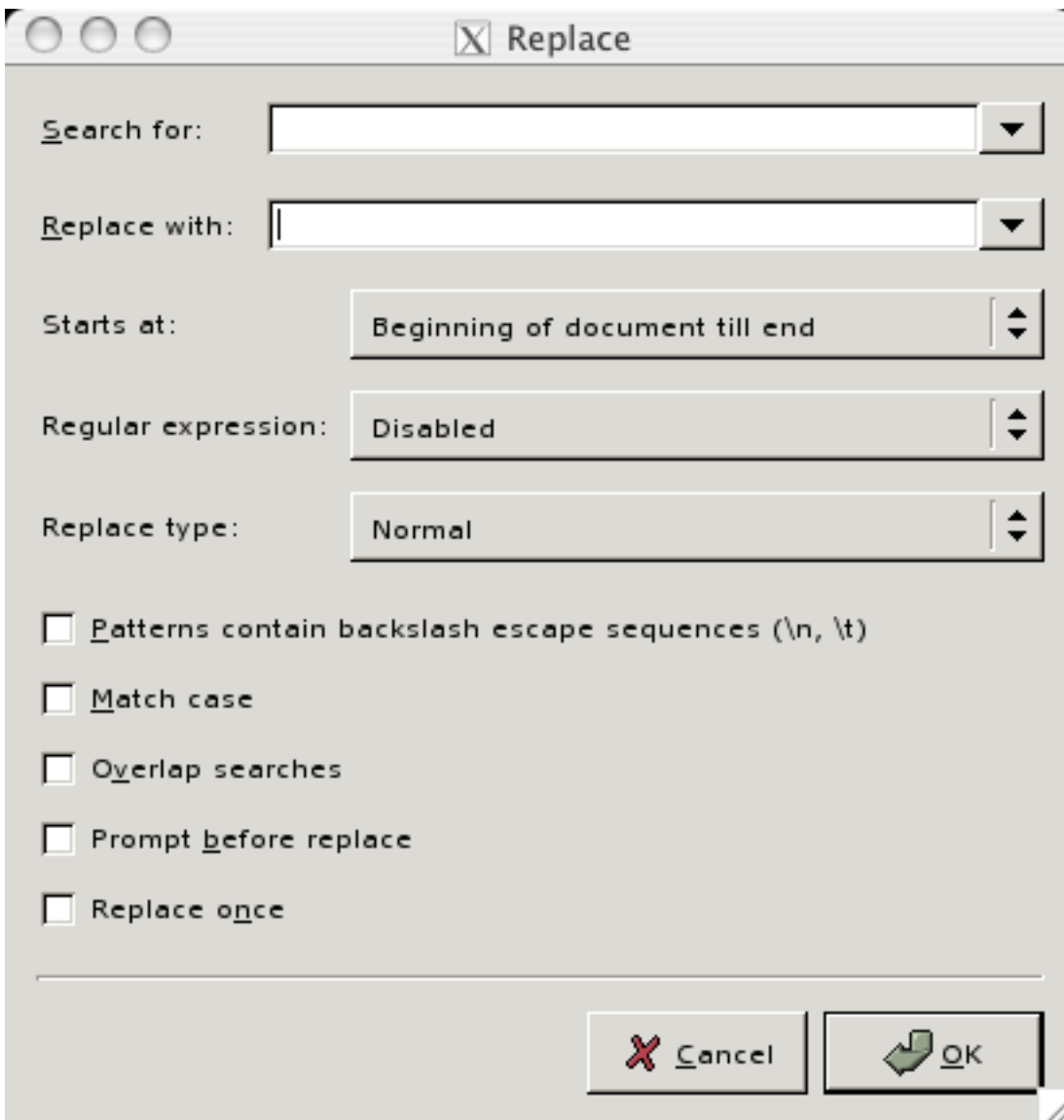


Figure V.30. The Replace Dialog

10.8. Retrieving previous replace strings

As for the **Search for** field's pop up menu, the **Replace with** field's pop up menu allows you to retrieve previous strings used for replace, the most recent ones being at the top of the list.

10.9. Changing letter case when replacing

If you want to change letter case when replacing, use the **Replace type** pop up menu.

The default choice is **Normal**, that is the case is not changed.

With the **Uppercase** replace type, the search string will be replaced with its uppercase translation.

Likewise, with the **Lowercase** replace type, the search string will be replaced with its lowercase translation.

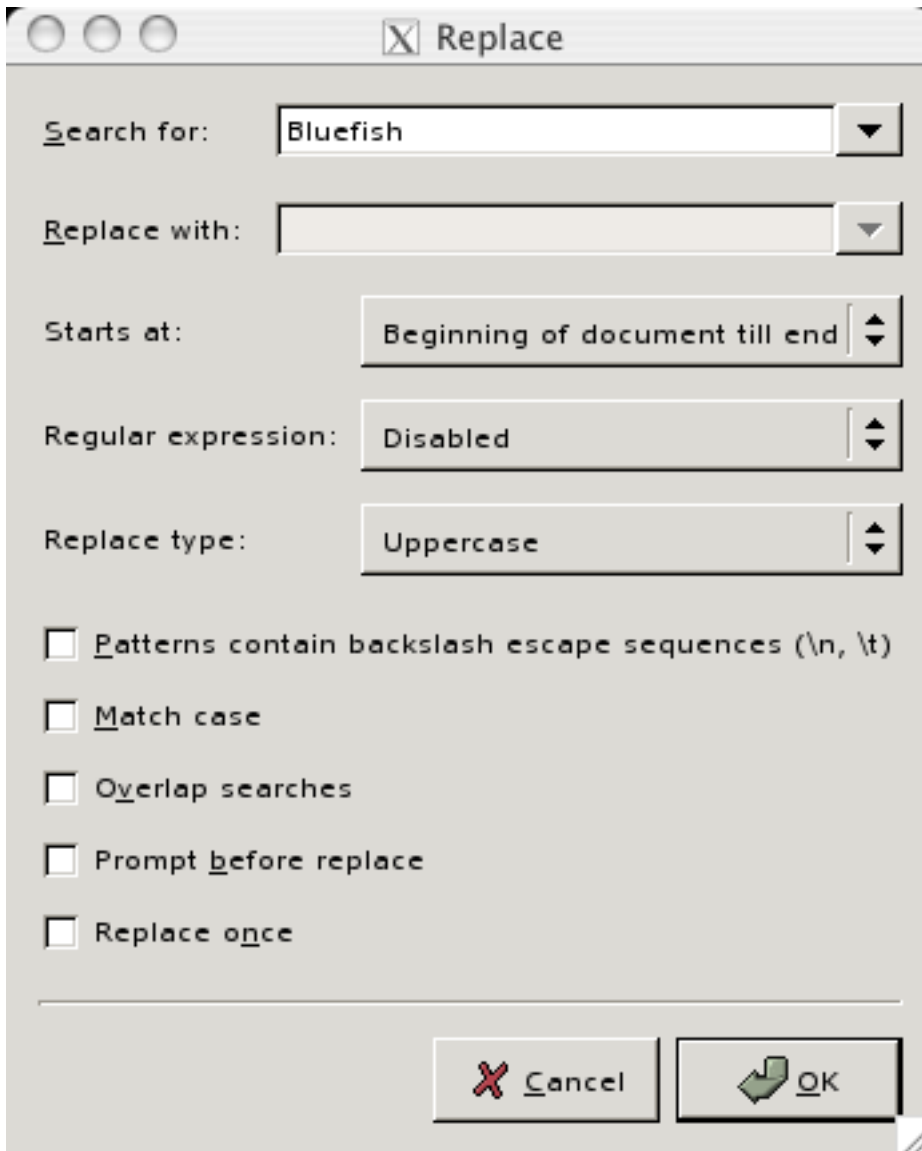


Figure V.31. Changing letter Case when Replacing

Notice that in this case, the **Replace with** field is deactivated, thus not taken into account even if you have entered some string in it.

10.10. Choosing strings to replace

It may occur that you do not want to replace all search strings retrieved by the search process, but only some of them. In this case, check the **Prompt before replace** box. A **Confirm replace** dialog will appear for each retrieved string where you can choose to **Skip** this string, i.e. leave it as it is, **Replace** it, **Replace all** strings within the search scope, or **Close** the dialog, i.e. cancel the process.

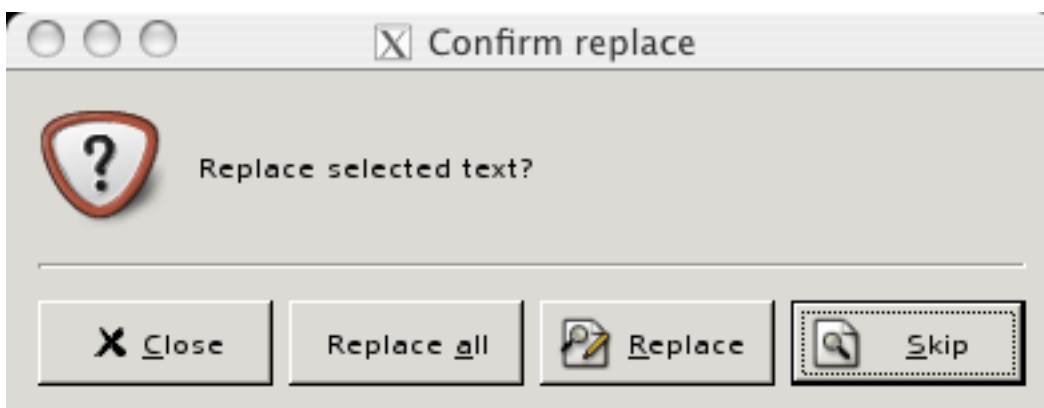


Figure V.32. The Replace Confirm Dialog

If you want to replace only the first occurrence of a search string, check the **Replace once** box instead.

10.11. More on replace

For further explanation on replace features within Bluefish, see [Section 5, “Find and Replace” \[53\]](#)

11. File types

Different file types can change the behaviour of Bluefish. File types are recognized by their extension, or by the beginning of the file's contents. The current document type is shown in the far right of the status bar. If the type of a file is not properly detected, you can change the type using the **Document** → **Document Type** menu. See [Chapter VIII, Customising Bluefish \[81\]](#) to change these extensions.

11.1. Syntax highlighting

Syntax highlighting is the coloring of words that have special meaning for a language. The patterns can vary: for example, "<title>" means "start of title" in HTML, "function" means "start of function" in PHP.

While editing, Bluefish will only update the highlighting patterns in the block of text around the cursor. The number of lines (the size) of this block can be adjusted in the preferences under **Editor**. The syntax highlighting for the total document can be refreshed using the **Document** → **Update Highlighting (F5)** menu. It can be disabled in the preferences under **Editor**. For more information about adding or modifying syntax highlighting for existent or new languages, see [here \[89\]](#)

12. More on files

12.1. Remote files

Assuming a working Internet connection, files can also be opened from the web using **File** → **Open URL**. This feature depends on your `gnome_vfs` setup. If it is installed and working, `http://`, `sftp://`, `smb://` and possibly more types of remote services should be supported by Bluefish. Depending on your `gnome_vfs` version, some of these protocols are not yet fully stable, which can crash Bluefish!

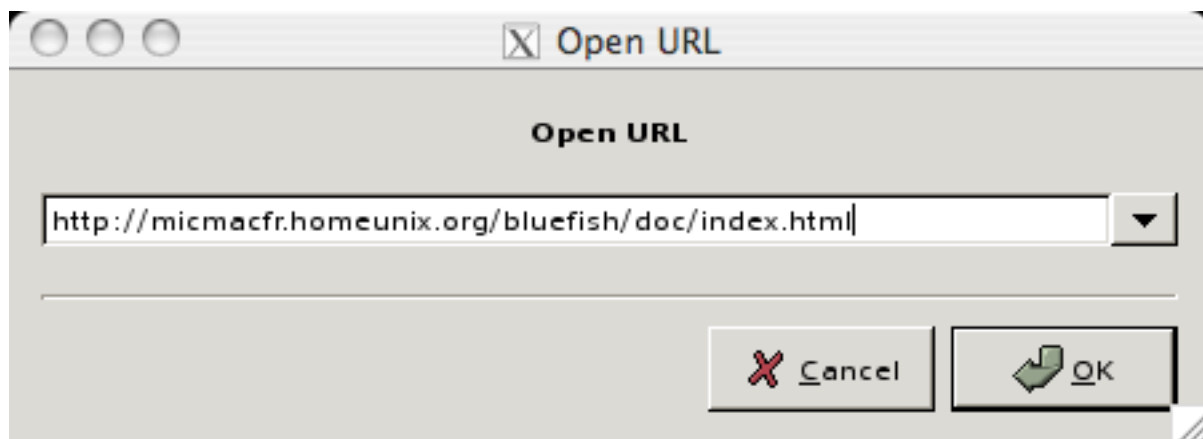


Figure V.33. Opening an URL from the Web

Here you can see the style sheet of an Apache web site, nicely highlighted after its opening via the Bluefish **File** → **Open URL** menu.

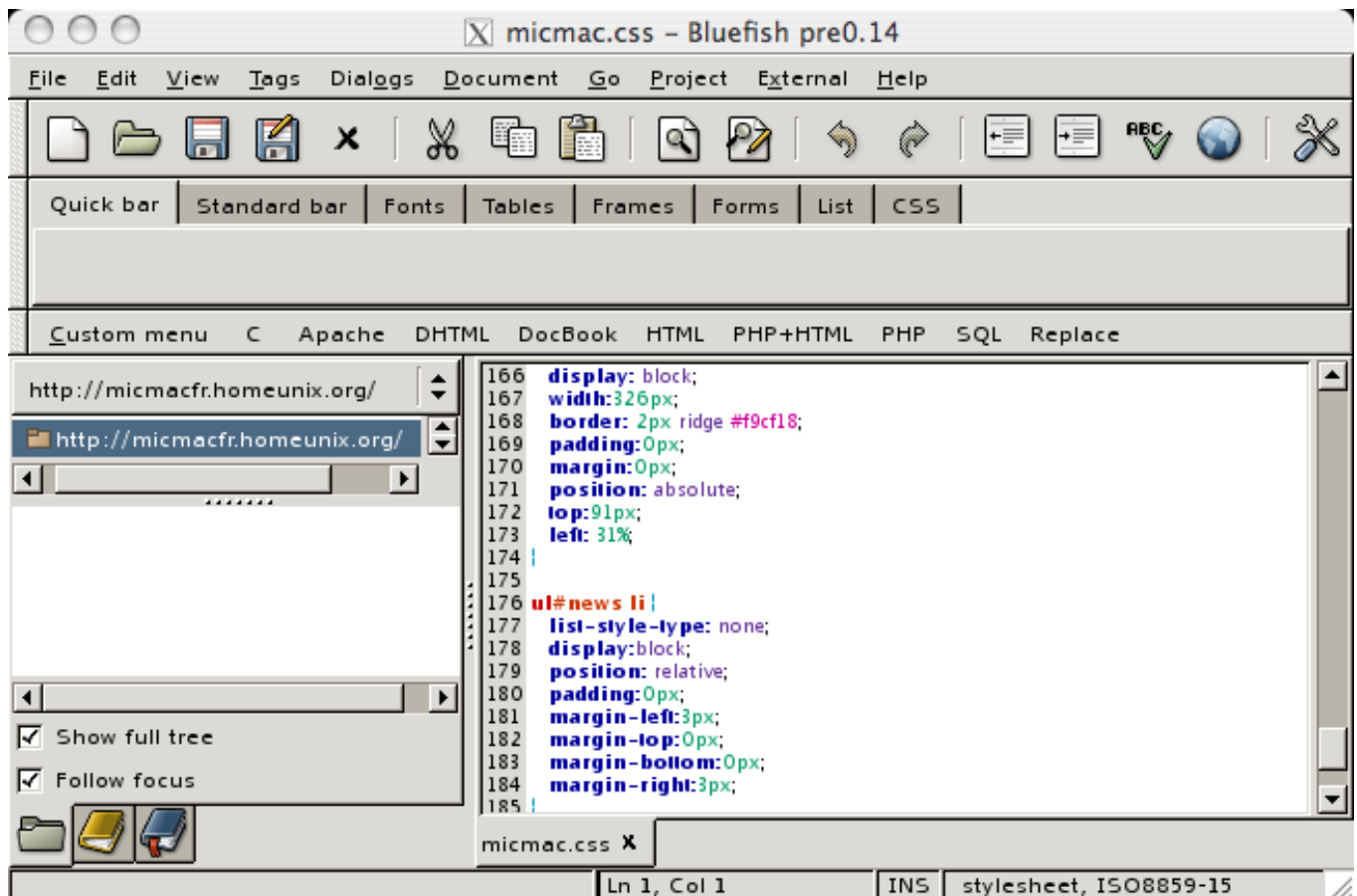


Figure V.34. A Style Sheet opened via the Open URL Menu

12.2. Character encoding

There are many different standards for character encoding of text files. Most well known is the ASCII standard, which describes only 127 characters, and is supported by every text editor in the world. The most common standard nowadays is UTF-8, which describes thousands of characters, and is backwards compatible with ASCII.

Internally, Bluefish will always work with UTF-8. When opening a file, Bluefish has to detect the correct encoding for the file. For HTML files, the encoding should be present in a `<meta name="encoding">` tag. Bluefish will always use this tag if it is available in the file. If this tag has an encoding that is not present in the Bluefish config file, this encoding is automatically added to the Bluefish config file.

The locale also defines a default encoding. If you are using a locale (a local setting, defining language, time format, currency format, number formatting etc.), Bluefish will try to load the file using the encoding defined in the locale.

Bluefish itself also has a setting for a default encoding. This is the next encoding Bluefish will try. This is also the encoding Bluefish will use for files created by Bluefish (UTF-8 by default).

If these steps fail, Bluefish will simply try every encoding defined in the Bluefish config file.

Filenames on disk can also contain non ASCII characters. All GNOME and GTK programs (including Bluefish) assume that filenames are in UTF-8 encoding. If you have filenames in the encoding of your locale on your disk, you have to set `G_BROKEN_FILENAMES=1` in the environment to make GNOME and GTK programs detect this encoding.

For information about writing documents in 16-bits encoded languages, such as Japanese, see [Section 9.3, “Input methods” \[32\]](#).

12.3. Open advanced

You can open multiple files at once with the **File → Open Advanced...** (**Shift-Ctrl-O**) menu item from a directory based on their extension or their contents. The same functionality is available from the file browser in the side panel by right-clicking a directory. This feature is available only when the find and grep utilities are installed on your system.

To open all files by extension, enter the extension in the dialog, and leave the search pattern empty. Check the **recursive** option if you want to include all subdirectories in the search. To open files by content, leave the extension at `*`, and enter a search pattern in the dialog. You can use regular expression patterns if you check the **Is regex** option.

You may also combine both methods. Here we open recursively all Chinese XML files in a given tree, whose contents contain

the word packaging.

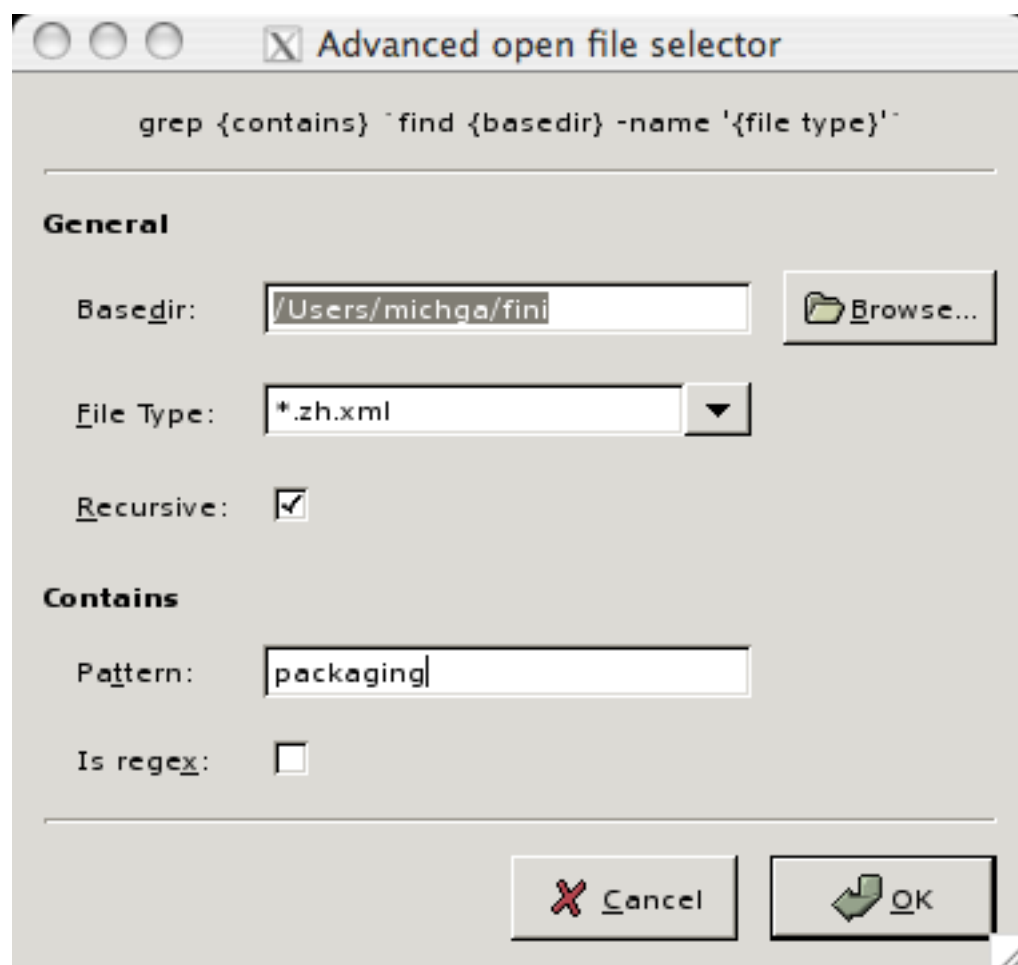


Figure V.35. Using the Open Advanced Dialog

Chapter VI. Navigation and Managing documents

1. Navigating through a document

The editing area is a standard GTK editing area. This means there are many keyboard shortcuts available to navigate through the text.

- **Ctrl-Right-Arrow** will jump to the next word boundary
- **Ctrl-Left-Arrow** will jump to the previous word boundary
- **End** will jump to the end of line
- **Home** will jump to the beginning of the line
- **Page-Up** will jump one page up
- **Page-Down** will jump one page down
- **Ctrl-Home** will jump to the top of the document
- **Ctrl-End** will jump to the end of the document

These shortcuts are also available when selecting text. Some examples:

- To select the current line, press **Home**, hold **Shift** and press **End**.
- To select the current word, press **Ctrl-Left-Arrow**, hold **Shift** and press **Ctrl-Right-Arrow**.

2. Navigating through many documents

Navigating through a large list of documents can be difficult. But if you right-click the document notebook tabs, you get a list of all opened documents.

Navigation between documents can also be done using the **Go** menu, or its shortcuts.

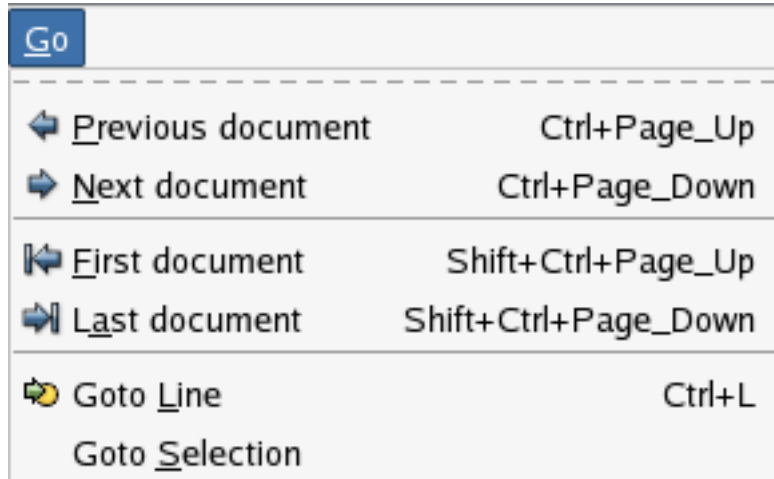


Figure VI.1. Bluefish Go Menu

The shortcuts are the following:

- **Ctrl-Page-Up** will change to the previous document
- **Ctrl-Page-Down** will change to the next document
- **Shift-Ctrl-Page-Up** will change to the first document
- **Shift-Ctrl-Page-Down** will change to the last document

The **Go → Goto Line (Ctrl-L)** offers an interesting feature.

If there is some number in the document, you may select it, then click the **From selection** label in the Goto line dialog. Bluefish will fill in the **Line number** field with that number and go directly to it. The same feature is available from the **Go → Goto Selection**.

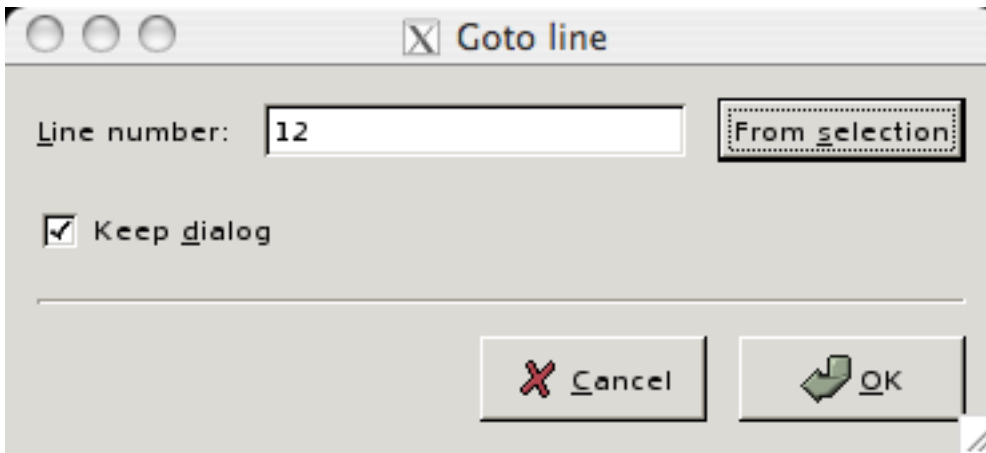


Figure VI.2. Using the Goto Line Dialog



Check the **Keep dialog** box to keep the dialog open, when you plan to access several parts of the document by line numbers.

3. Projects

The projects are a sort of *saved state* of Bluefish. Thus, they are a very convenient way to work with files scattered all over your disks or to pick up only the files you are interested in within a huge tree. Projects features are accessible through the **Project** menu.

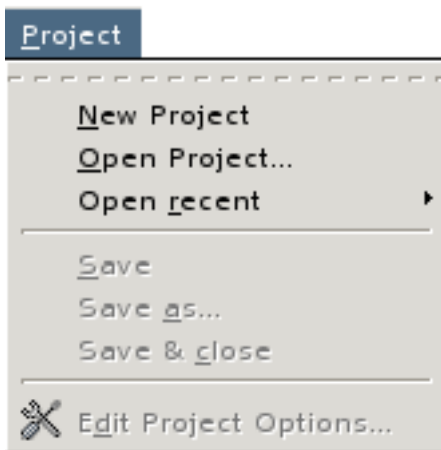


Figure VI.3. The Bluefish Project Menu

Procedure VI.1. Creating a New Project

1. Click on the **Project** → **New Project**

If some documents are already opened, check the appropriated box in the **Create project** dialog.

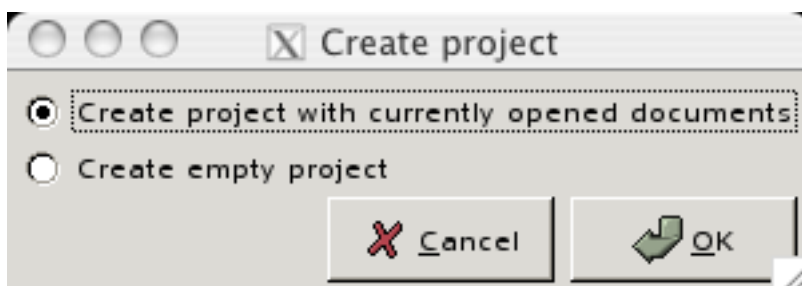


Figure VI.4. The Create Project Dialog

2. Fill in the fields in the **Create New Project** dialog

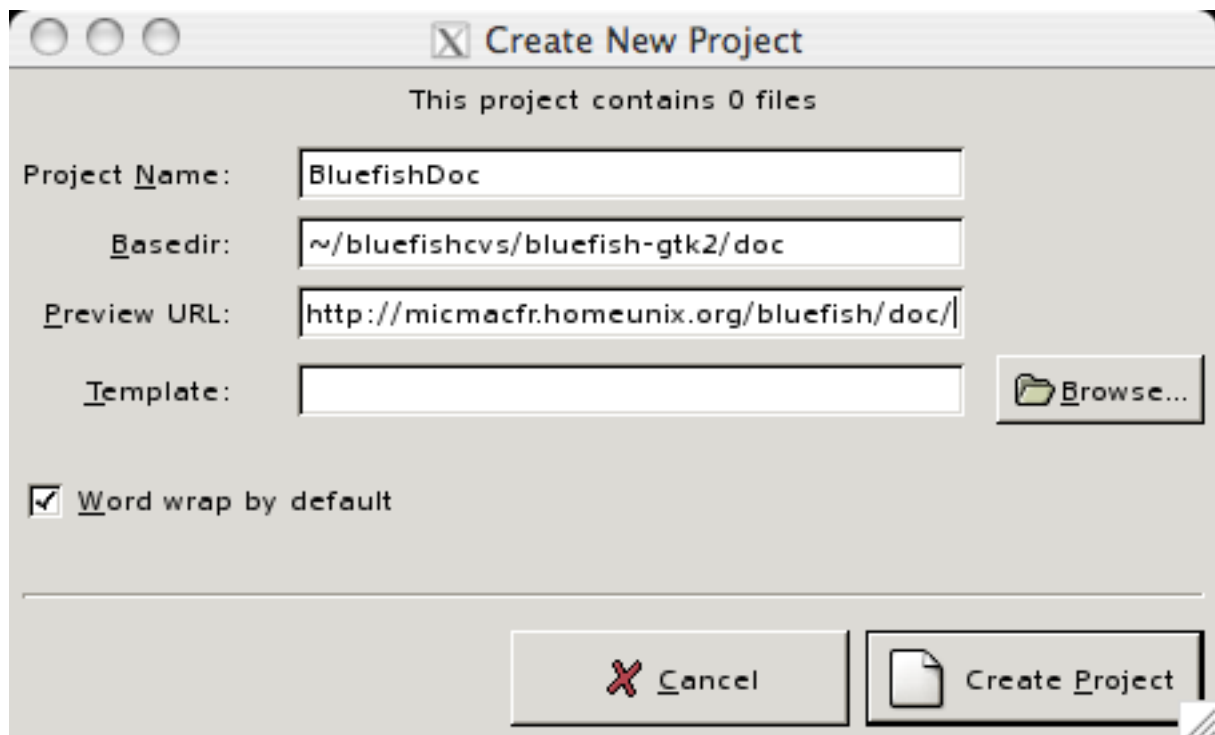


Figure VI.5. Creating a New Project

With a **Basedir** the file browser in the side panel shows only the files within its hierarchy. With **gnome-vfs** support, the **Basedir** can be remote, as `smb://user:pass@server/someshare/` or `sftp://someserver/somedir`.

The **Preview URL** allows Bluefish to launch the browser to the appropriate URL, for example `http://localhost/Bluefish`. This can be very convenient for testing server side scripting languages like PHP, JSP, etc.

If the **Template** field is used, Bluefish will use the template file's contents for new files, which can be requested either via the **New** button on the main tool bar or **File → New (CtrlN)**. Otherwise an empty document will be created.

3. Once the project is created, you need to tell Bluefish where you want to save it. An **Enter Bluefish project filename** dialog will be displayed. Notice that you can save the project in a location different from the files to which the project points.

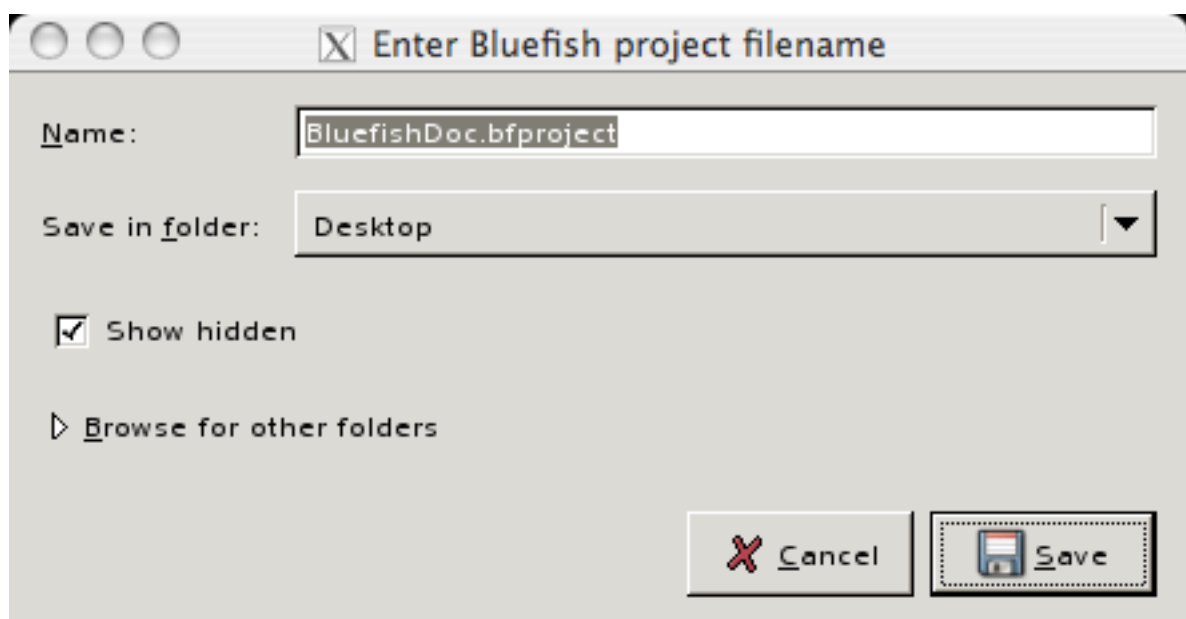


Figure VI.6. Entering Bluefish Project Filename

To open a project, you have the choice between **Project → Open Project...** or **Project → Open recent**. When you choose the former, a **Selecting a Bluefish Project** dialog is presented to you.

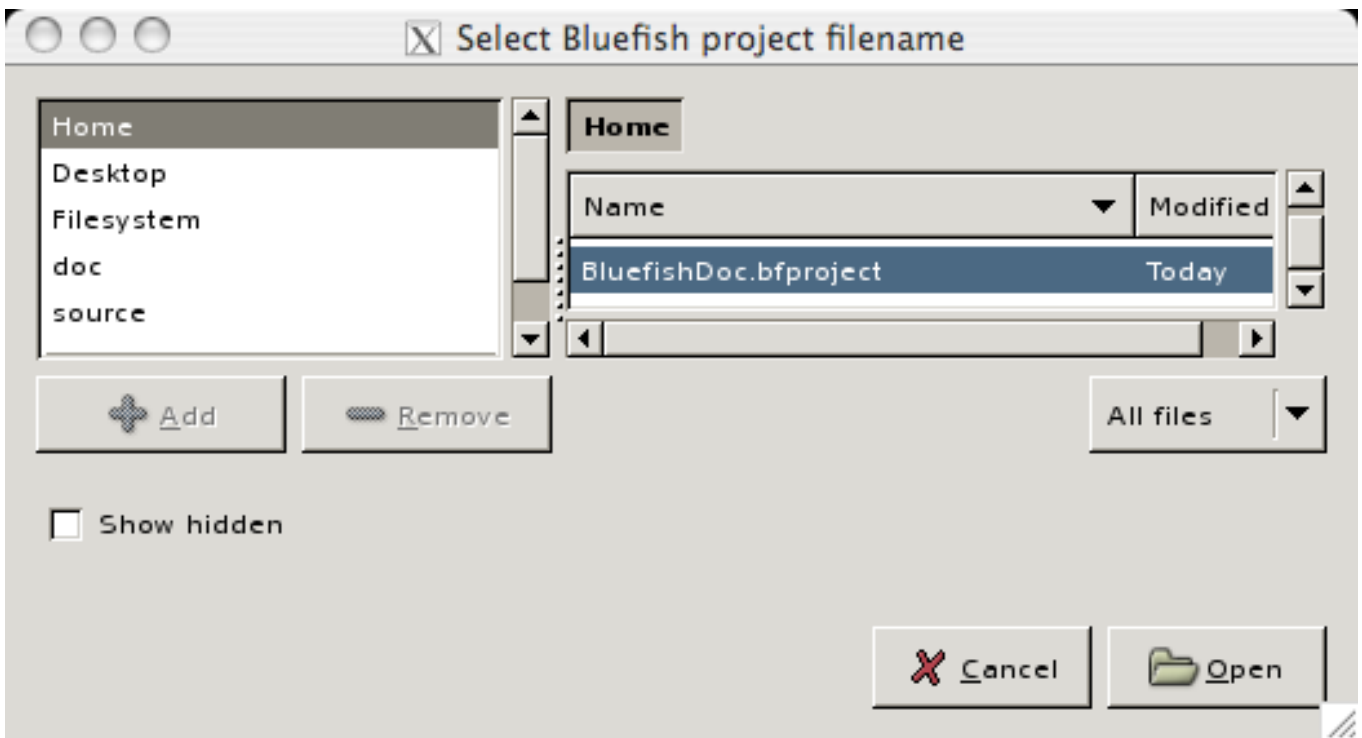


Figure VI.7. Selecting a Bluefish Project

To save the project under its current name/location, use **Project** → **Save** or **Project** → **Save & close**; to save it under a new name/location, use **Project** → **Save as....** If any file in the project has changed, a dialog will allow you to save the file, discard the changes, or cancel. All files open when the project is saved are automatically opened the next time you open the project.

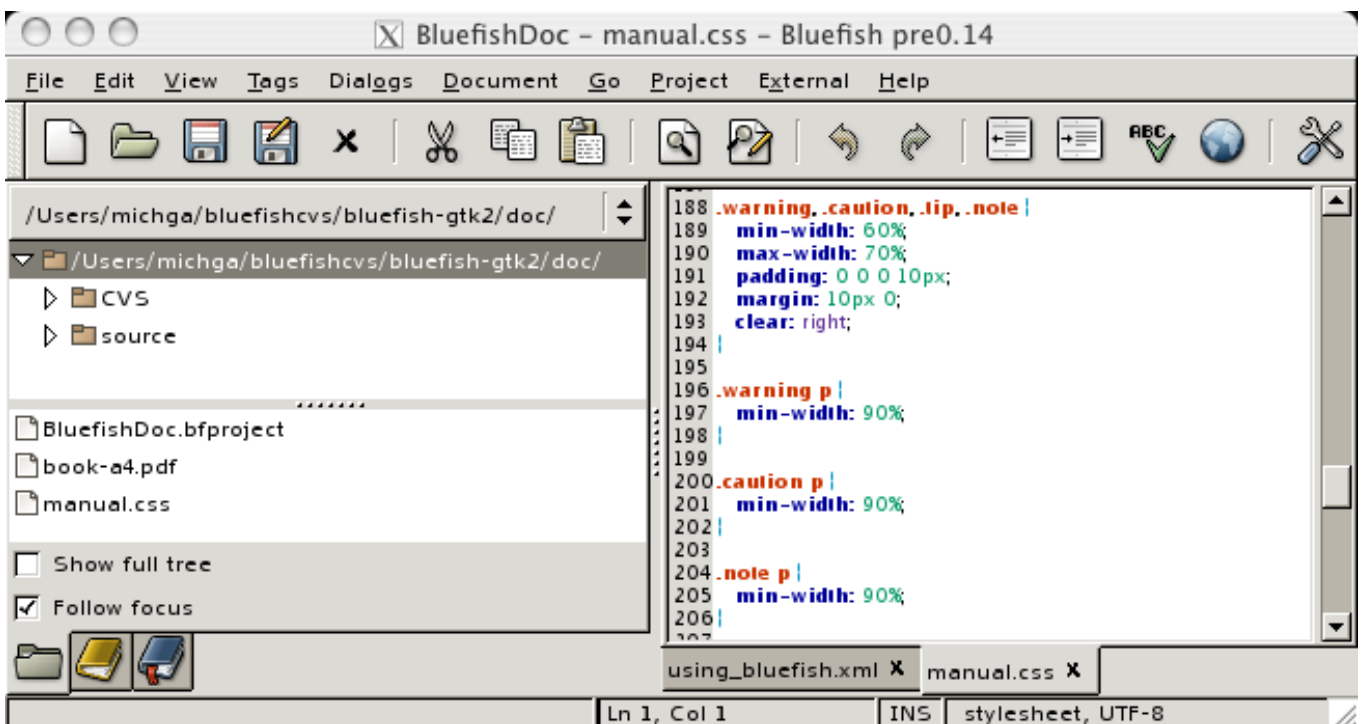


Figure VI.8. Opening a Bluefish Project

Notice that the side panel only shows the tree related to the project.

Also, the recently used files in that project are shown in the **File** → **Open recent** menu item.

A project also saves some basic Bluefish settings, giving the project its own customized Bluefish setup. Currently, the word wrap preference and the state of various tool and menu bars are saved in a project file. The project file itself is simply a text file in the standard Bluefish format (same format as the config file). This format is key: value. Here is an example:

```
name: BluefishDoc
basedir: ~/bluefishcvcs/bluefish-gtk2/doc/
webdir: http://micmacfr.homeunix.org/bluefish/doc
```

```
template:
view_main_toolbar: 1
view_left_panel: 1
view_custom_menu: 1
view_html_toolbar: 1
word_wrap: 1
```

4. Bookmarks

In Bluefish you can add bookmarks to a line in the text, and you can later use the bookmark to quickly jump to this location, or even to open the document referred to by the bookmark at that line.

Bookmarks can be added to the current cursor location by using the **Edit** → **Add Bookmark** (**Ctrl-D**) menu item; or by right-clicking in the text, and selecting **Add bookmark**. You can delete a bookmark using the **Delete bookmark** item in the document contextual menu.

Each bookmark in a given document is marked by a blue background in the line number margin.

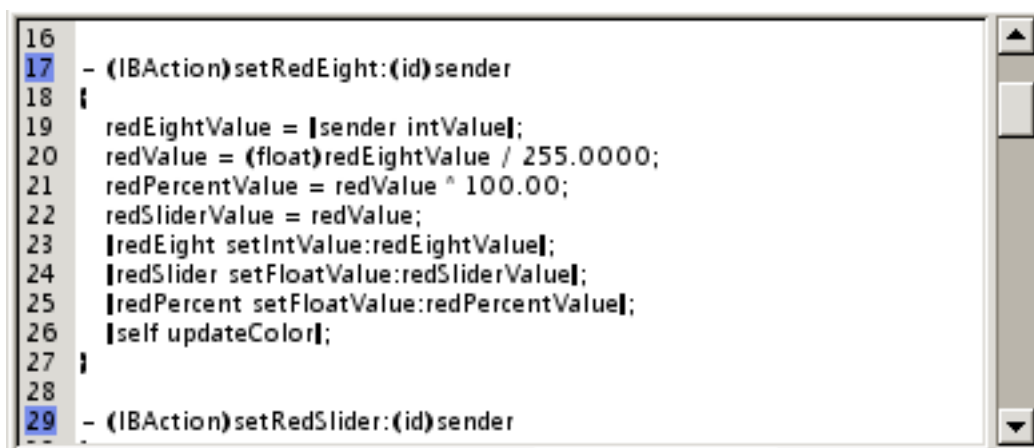


Figure VI.9. How Bookmarks are marked

Bookmarks can be temporary or permanent. Permanent bookmarks are stored, and temporary bookmarks are gone after Bluefish is closed. The default is set in the preferences under **Editor**.

Bookmarks can be found in the third tab of the side panel, sorted by document and line number.

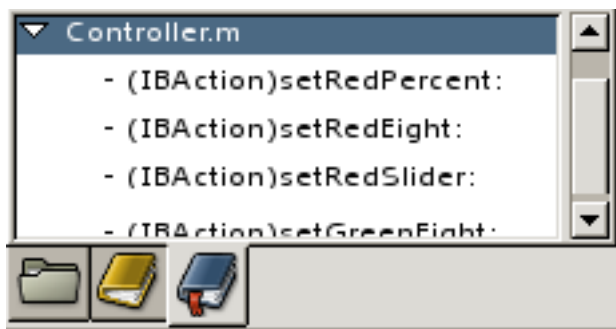


Figure VI.10. Bookmarks in the Side Panel

If you right click a bookmark in the bookmark tab of the side panel, you get a pop up menu with several options.

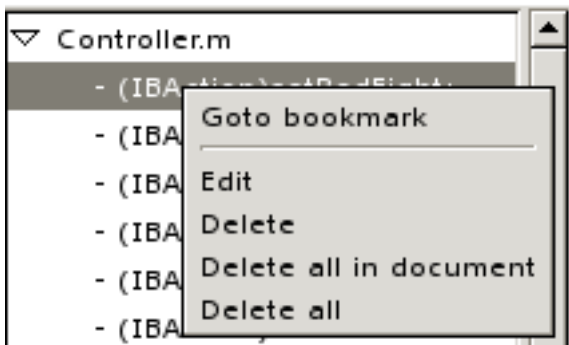


Figure VI.11. Contextual Menu on Bookmark in the Side Panel

The **Goto bookmark** item allows you to go to the bookmark location in the document, opening it if needed.

The **Edit** item allows you to change a bookmark from temporary to permanent or the other way around, to name it, and to give it a short description.

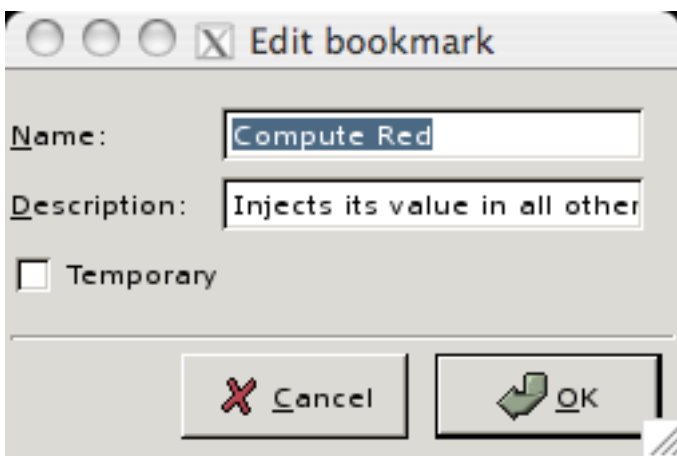


Figure VI.12. Editing a Bookmark

Note that after naming a bookmark, the default name - first characters of the bookmarked line - is displayed after the new name.



Figure VI.13. A named Bookmark

Via this contextual menu, you may also delete a bookmark, delete all bookmarks in the active document, or delete all bookmarks stored in the bookmark tab of the side panel. The latter ones are also available when you right click the name of a document in this tab.



Figure VI.14. The Contextual Menu on a Document in the Bookmark Tab

4.1. Generating several bookmarks at once

To add many bookmarks at once, use the **Edit → Find (Ctrl-F)** dialog. Check the **Bookmark result** option, and all search results will be added to your bookmarks.

For example, the XML files for this manual have sections, each identified by a header like `<sect1 id="nameofthesection">`. A way to automatically get a bookmark to every section is to search for the following posix regular expression pattern: `<sect[0-9]+ id="[^"]+ ">` and bookmark all results.

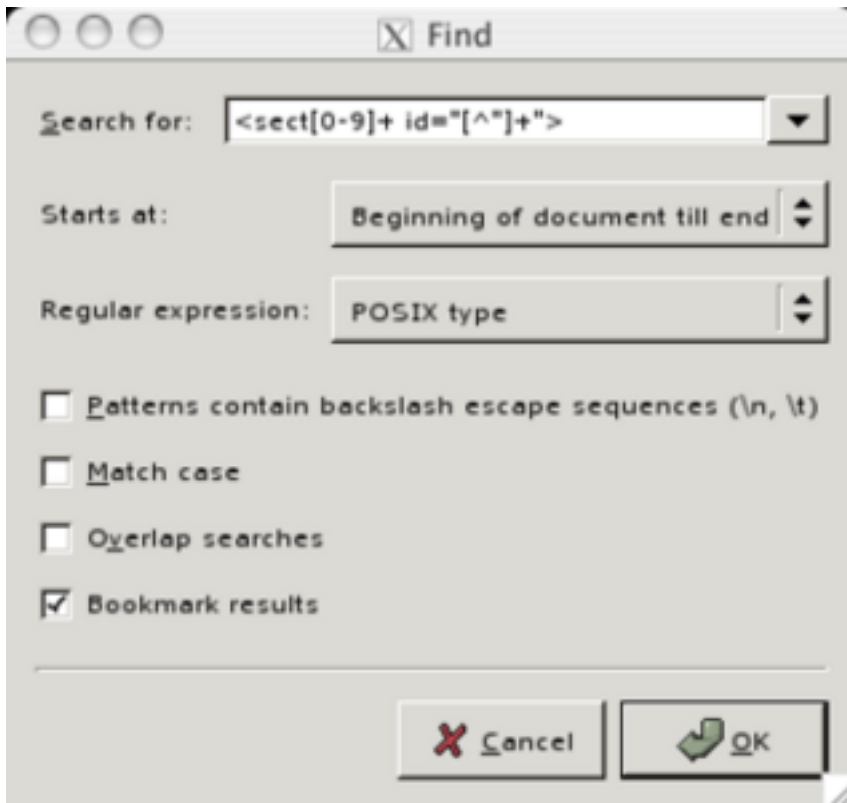


Figure VI.15. Bookmarking with Posix Regular Expression

Here is the result:

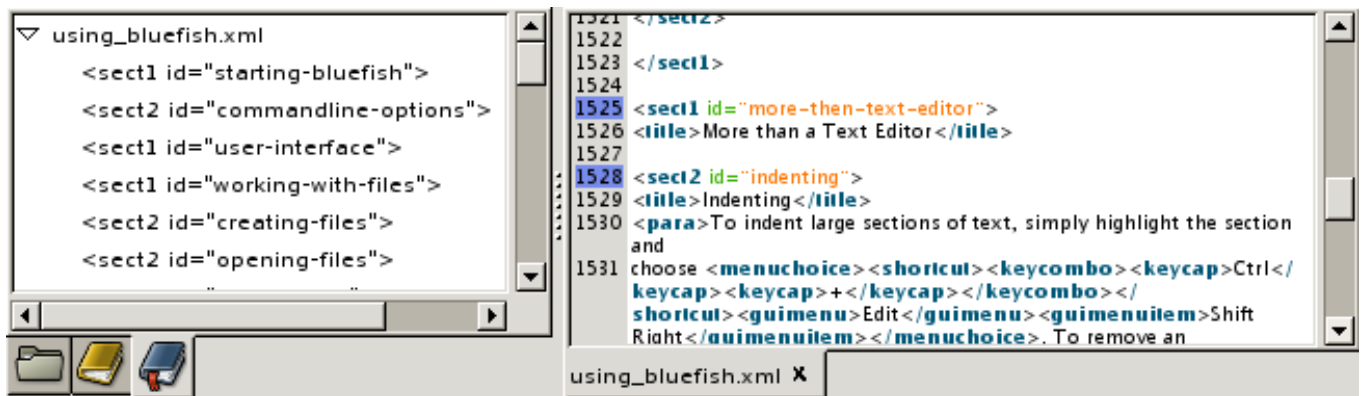


Figure VI.16. Bookmarks with Posix Regular Expression

Here are two examples which bookmarks all functions in Objective-C and PHP files with POSIX or PERL regular expressions:

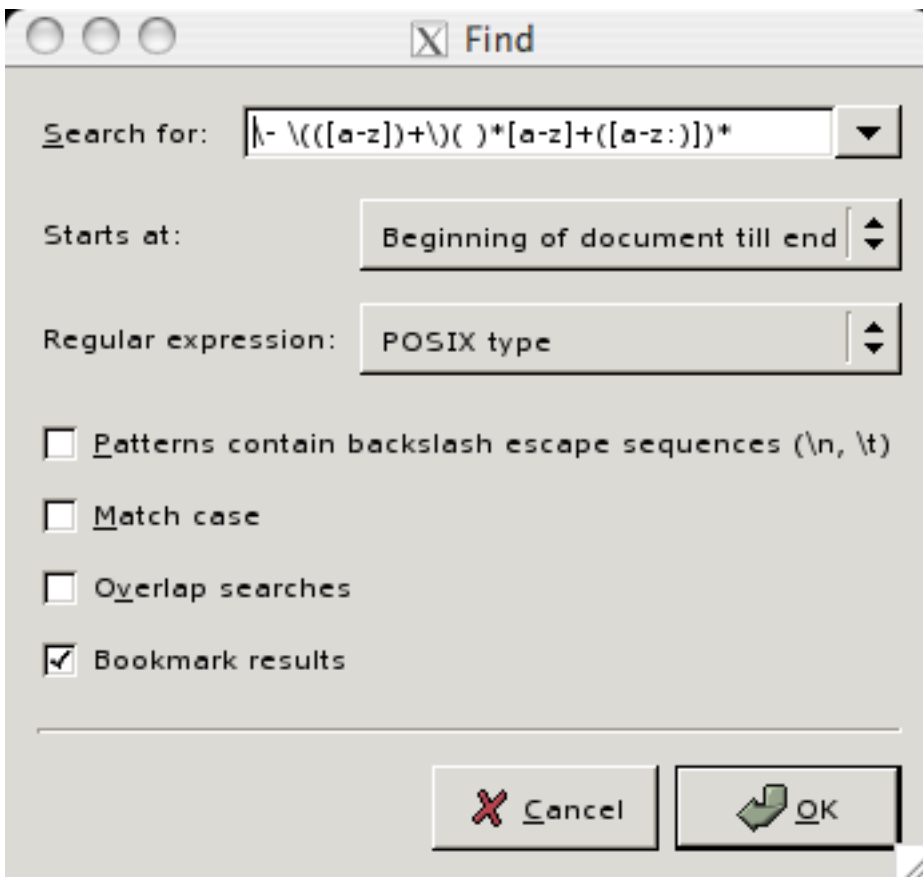


Figure VI.17. Bookmarking Objective C Functions via the Find Menu

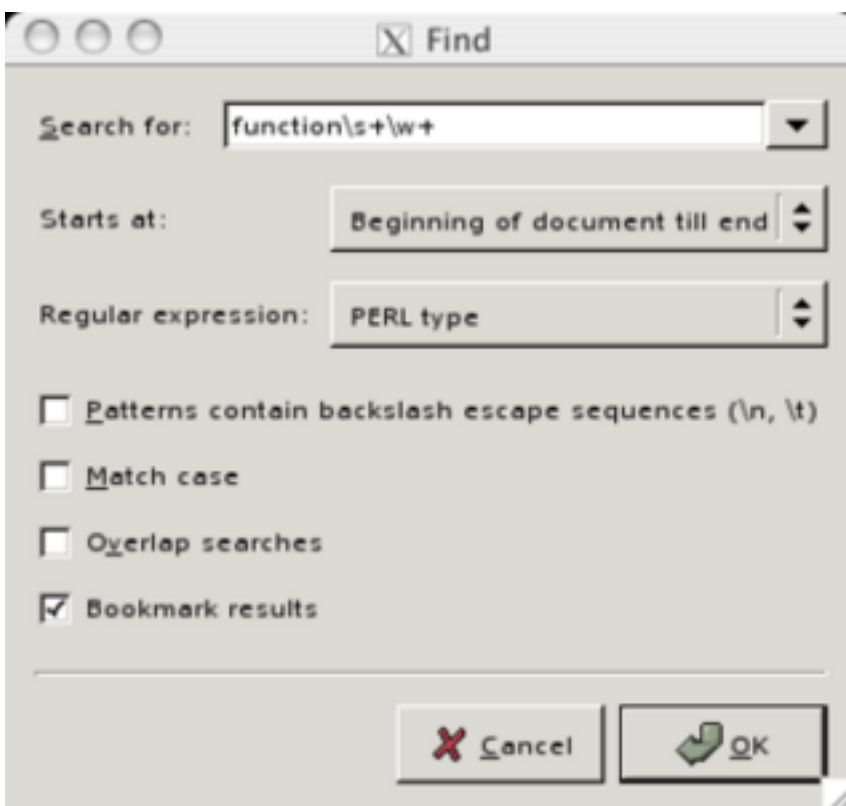


Figure VI.18. Bookmarking PHP Functions via the Find Menu

Check [Section 5.3, “Find and Replace Using Regular Expressions” \[54\]](#) for more information on finding and replacing with regular expression in Bluefish.

5. Find and Replace

The **Edit** menu features several options for Find and Replace. The **Edit → Find (Ctrl-F)** and **Edit → Replace... (Ctrl-H)** menu items will simply start their corresponding dialogs described in [Section 10, “Basic Find and Replace”](#) [33].

5.1. Find Again

The **Edit → Find again (Ctrl-G)** menu item will repeat the last used search. It will continue the search after the position where the previous search was stopped. If the end of file is reached, it will signaled it if the search operates on a unique file, nevertheless you can continue the search from the top of file with **Ctrl-G** after dismissing the **Search: no match found** dialog. If the search operates on multiple files, it will continue with the next file.

Here you can see how the **Find again** process operates on two successive searches for the word "url" in an xml file:

```

996 <imageobject>
997 <imagedata fileref="figures/open_url.png" format="PNG"/>
998 </imageobject>
999 <textobject>
1000 <phrase>A screen shot showing how to open an URL from the web</phrase>
1001 </textobject>
1002 </mediaobject></screenshot>
1003 </figure></para>
1004 <para>Here you can see the style sheet of an Apache web site, nicely highlighted after its
opening via the Bluefish <menuchoice><guimenu>File</guimenu><guimenuitem>Open URL</guimenuitem></menuchoice> menu.
1005 <figure id="figure-open-url-result">
1006 <title id="figure-open-url-result-title">A style sheet opened via the Open URL menu</
title>
1007 </screenshot>
1008 <mediaobject><imageobject>
1009 <imagedata fileref="figures/open_url_result.png" format="PNG"/>
1010 </imageobject>
1011 <textobject>
1012 <phrase>A screen shot of a style sheet opened via the Open URL menu</phrase>

```

Figure VI.19. Nth Occurrence with Find Again

```

996 <imageobject>
997 <imagedata fileref="figures/open_url.png" format="PNG"/>
998 </imageobject>
999 <textobject>
1000 <phrase>A screen shot showing how to open an URL from the web</phrase>
1001 </textobject>
1002 </mediaobject></screenshot>
1003 </figure></para>
1004 <para>Here you can see the style sheet of an Apache web site, nicely highlighted after its
opening via the Bluefish <menuchoice><guimenu>File</guimenu><guimenuitem>Open URL</guimenuitem></menuchoice> menu.
1005 <figure id="figure-open-url-result">
1006 <title id="figure-open-url-result-title">A style sheet opened via the Open URL menu</
title>
1007 </screenshot>
1008 <mediaobject><imageobject>
1009 <imagedata fileref="figures/open_url_result.png" format="PNG"/>
1010 </imageobject>
1011 <textobject>
1012 <phrase>A screen shot of a style sheet opened via the Open URL menu</phrase>

```

Figure VI.20. Nth+1 Occurrence with Find Again

5.2. Find from Selection

The **Edit → Find from selection** menu item will search for the currently selected text. If you select for example the name of a function, in bluefish, or in any other program, and you choose find from selection Bluefish will start a new search for this selected string.

Here we have selected a function in a C file:

```

7
8 #include <string.h>           // necessary on Mac OS X
9 #include <gdk/gdkkeysyms.h>   // to retrieve key mappings
10 #include <gtk/gtk.h>          // the necessary gtk
11
12 // Function prototypes
13 GtkWidget* create_dialog (void); // main gui
14 GtkWidget* create_message_dialog (gchar* message); // response
15 void button_enter_text_clicked( GtkWidget* button, gpointer user_data );
16 void button_clear_text_clicked( GtkWidget* button, gpointer user_data );
17
18 int main (int argc, char *argv[])
19 {
20     GtkWidget *dialog;
21
22     // init gtk before create any Gtk widget or use gtk signals
23     gtk_init (&argc, &argv);
24
25     // create the ui
26     dialog = create_dialog ();

```

Figure VI.21. Selecting a String for subsequent Search

Clicking on **Edit** → **Find from selection** gives the following result:

```

32     gtk_widget_show (dialog);
33
34     // Enter the gtk main loop
35     gtk_main ();
36
37     // End of the loop
38     return 0;
39 }
40
41 GtkWidget* create_dialog (void)
42 {
43     GtkWidget *dialog;
44     GtkWidget *dialog_vbox;
45     GtkWidget *hbox;
46     GtkWidget *hbox2;
47     GtkWidget *vbox;
48     GtkWidget *label_hi;
49     GtkWidget *vbox2;
50     GtkWidget *entry_text;
51     GtkWidget *answer_text;

```

Figure VI.22. Finding a String from Selection

Next occurrences of the string can be found with **Ctrl-G** as usual.

5.3. Find and Replace Using Regular Expressions

With find and replace you can do incredibly powerful searches. We have already seen some of them in [Section 4.1](#), “Generating several bookmarks at once” [50], which deserve some explanation here.

Example VI.1. Retrieving all sections in an xml book

The regular POSIX expression `<sect[0-9]+ id="[^"]+ ">` can be split into:

- `<sect`: a string beginning by `<sect`
- `[0-9]+` : followed by one or more (the `+` part) characters in the range of 0 to 9 (the `[0-9]` part), i.e. digits, followed by a space
- `id=`: followed by the string `id`, followed by an equal sign, followed by a double-quote
- `[^"]+`: followed by one or more (the `+` part) not double-quote characters (the `[^"]` part - `^` is a not)
- `">`: followed by a double-quote, and ending with a `>` sign

Therefore, it matches any string of type `<sectn id="nameofthesection">`, where `n` is a positive integer.

Example VI.2. Retrieving all functions in an Objective C file

In a simplified example, an objective C function may have two forms:

1. - `(IBAction)nameofthefunction:(id)parameter`
2. - `(void) nameofthefunction`

We will try to make a pattern from those forms:

- Hyphens and parentheses have special meanings in regular expressions, hence we need to escape them, i.e. to put before each of them a backslash, so that they will be interpreted as normal characters.

Thus, `- (` is matched by: `\- \((`

- `IBAction` or `void` are a non empty sequence of alphabetical characters. We have already seen something similar in the previous example.

They are matched by: `[a-z]+`, that is one or more characters in the range of `a` to `z`.

- Another parenthesis matched by: `\)`.
- A space or no space at all, it is matched by: `*`, that is a space followed by an asterisk, which means 0 or more times the preceding character.
- A non empty sequence of characters, matched by `[a-z]+` as already seen.
- A colon or no colon at all, which is matched by: `[:]*`.

Thus the whole POSIX regular expression is: `\- \([a-z]+\)\ * [a-z]+[:]*`. In the example, we have grouped the parts with parentheses, you may prefer this simplified form, though it is not recommended.

Example VI.3. Retrieving all functions in a PHP file

A php function has the form `function nameofthefunction(listofparameters)`, where the list of parameters can be empty. To match it with a PERL regular expression, you have to know that `\s` matches any white space and `\w` matches any alphanumerical character as well as white spaces.

Thus, the matching regular expression is: `function\s+\w+`.

Now, if you want to capture also the function's parameters, you have to add:

- An opening parenthesis: `\(`. Remember parentheses should be escaped with a backslash.
- Zero or more characters, none of them being a closing parenthesis: `[^\)] *`
- A closing parenthesis: `\)`

The PERL regular expression becomes: `function\s+\w+\([^\)]* \)`.

Here is a new example which transforms a table into a definition list inside an html file.

Example VI.4. Transforming a table into a definition list

Say you have the following table:

Software	Use	Requirements	Author	Date	Download
BackupSeek 1.8	To catalog your backup from all media. Prints labels too.	PPC	Ken Ng	17 November 1999	English version (452 Ko)
Biblioteca v. 1.0	An HyperCard 2.2 database with a cool interface for managing your library	System 7.x and 2 Mg RAM	Agustín Cortés	March 1996	English version (532 Ko)

Figure VI.23. The table before Transformation

You want to transform it in the following definition list:

BackupSeek 1.8**Use:**

To catalog your backup from all media. Prints labels too.

System requirements:

PPC

Author:

Ken Ng

Date:

17 November 1999

Download:

English version (452 Ko)

Figure VI.24. The table after Transformation

For the rendering, you will use the following css style sheet:

```
.st2 {  
    font-weight: 900;  
    color: #e38922;  
    margin-left: 30px;  
}  
dl {  
    font-weight: 900;  
    margin-left: 55px;  
}  
dt {  
    margin-top: 6px;  
}  
.dd1 {  
    font-style: normal;  
    font-weight: 400;  
}
```

The table's source code is the following:

```
<table border="1">  
<tr>  
<th>Software</th>  
<th>Use</th>  
<th>Requirements</th>  
<th>Author</th>  
<th>Date</th>  
<th>Download</th>  
</tr>  
<tr>  
<td>BackupSeek 1.8</td>  
<td>To catalog your backup from all media. Prints labels too.</td>  
<td>PPC</td>  
<td>Ken Ng</td>  
<td>17 November 1999</td>  
<td>English version (452 Ko)</td>  
</tr>  
<tr>  
<td>Biblioteca v. 1.0</td>  
...  
</tr>  
</table>
```

The definition list's source code will be the following:

```
<p class="st2">BackupSeek 1.8</p>  
<dl>
```

```

<dt>Use:</dt><dd><span class="ddl">To catalog your backup from all media. \
Prints labels too.</span></dd>
<dt>System requirements:</dt><dd><span class="ddl">PPC</span></dd>
<dt>Author:</dt><dd><span class="ddl">Ken Ng</span></dd>
<dt>Date:</dt><dd><span class="ddl">17 November 1999</span></dd>
<dt>Download:</dt><dd><span class="ddl">English version (452 Ko)</span></dd>
</dl>
<p class="st2">Biblioteca v. 1.0</p>
...
</dl>

```

Comparing both chunks of code, we see that the variable sequence of characters to capture is the one between one `<td>` tag and its closing `</td>` tag. That sequence can be interpreted as one or more characters which are not a `<`. We have already seen that. This is expressed as: `[^<]+`

To be able to retrieve it later, we need to embed it into parentheses. Thus, the string becomes: `([^<]+)`

Next, this sequence is embedded into `<td>` and `</td>`, which is expressed simply concatenating the strings: `<td>([^<]+)</td>`

We should also add the end of line character, which is expressed as: `\n`. The regular expression now describes a whole line: `<td>([^<]+)</td>\n`

As we cannot use variables to retrieve the headers of the table, we will merely repeat that string five times, so that the regular expression matches exactly the six lines of importance to us.



Do not type it six times in the search field. Select the string, use the shortcuts **Ctrl-C** to copy it, move to the end of the string with the right arrow, and use **Ctrl-V** five times to paste it at the end of the string.

The regular expression becomes (backslashes are inserted at end of line just for the purpose of not to have too long lines):

```

<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \

```

Those lines are at their turn embedded into `<tr>` and `</tr>` tags each of them on their own line, which can be expressed as: `<tr>\n` for the first one, and `</tr>\n` for the second one. We add those strings respectively at the beginning and at the end of our regular expression, which becomes:

```

<tr>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
<td>([^<]+)</td>\n \
</tr>\n

```

Now that we have described the search pattern, we will build the replace pattern. Each expression embedded into parentheses in the search string can be retrieved with `\x`, where `x` is an integer starting at 0 for the first expression, 1 for the second, etc. All others parts in the final string are fixed strings which we will express as they are.

The first line becomes (note the `\n` at the end to match the end of line character):

```
<p class="st2">\0</p>\n
```

The second line (again, note the `\n` to match the end of line characters):

```
<dl>\n<dt>Use:</dt><dd><span class="ddl">\1</dd>\n
```

And finally the whole replace pattern is:

```

<p class="st2">\0</p>\n \
<dl>\n<dt>Use:</dt><dd><span class="ddl">\1</dd>\n \
<dt>System requirements:</dt><dd><span class="ddl">\2</span></dd>\n \
<dt>Author:</dt><dd><span class="ddl">\3</span></dd>\n \
<dt>Date:</dt><dd><span class="ddl">\4</dd>\n \

```

```
<dt>Download:</dt><dd><span class="dd1">\5</span></dd>\n</dl>\n
```

After entering both patterns, choose **PERL** type in the **Regular expression** drop down list, check the **Patterns contain backslash escape sequences (\n, \t)** and click OK.

After replacement occurred, you have to remove the table headers and the last `</table>` tag and to insert the link to the style sheet.

Note that if some lines contain a `<` sign, the table row will not be translated, but others will.

In the **Find** and **Replace** dialogs it is not possible to insert the keys **Enter** or **Tab**. A simple way to do it is to copy two lines in a row from the current document into the **Find** or **Replace** dialog, this way you retrieve the end of line character. The same applies for **Tab**. A more elaborated way to do it is to use escaped characters to represent these characters. A new line character, produced by pressing the **Enter** key, is represented as `\n`. Use `\t` for a tab. To get an actual backslash, just escape the backslash, `\\`. There are many other escape characters used in regular expressions.



To enable the escaped characters in your searches check the **Patterns contain backslash sequences (\n, \t)** option.

If you have any search and replace patterns you use often, you can also add them to the Custom Menu. Check [Section 7, “Custom menu” \[68\]](#) for more information.

For more information about regular expressions you might want to read `man 1 perlre`, `man 3 pcrepattern`, `man 3 regex` or `man 7 regex`, or read any of the great Internet sites about regular expressions. As you become more familiar with regular expressions, you will realize that they make Bluefish a very powerful editor.

Chapter VII. More than a Text Editor

1. Indenting

To indent large sections of text, simply highlight the section and choose **Edit** → **Shift Right (Ctrl-.)**. To remove an indentation, choose **Edit** → **Shift Left (Ctrl-,)**. There are corresponding buttons in the tool bar for these menu options (see later in this text).

By default, Bluefish will use tabs for indenting, but can be configured to use spaces if you have **Use spaces to indent, not tabs** selected in the Editor preferences panel. The number of spaces used is the same as the **Tab width** option in the same preferences panel.

Here's an extract of Dante's work indented with the **Shift Right** button in the main tool bar:

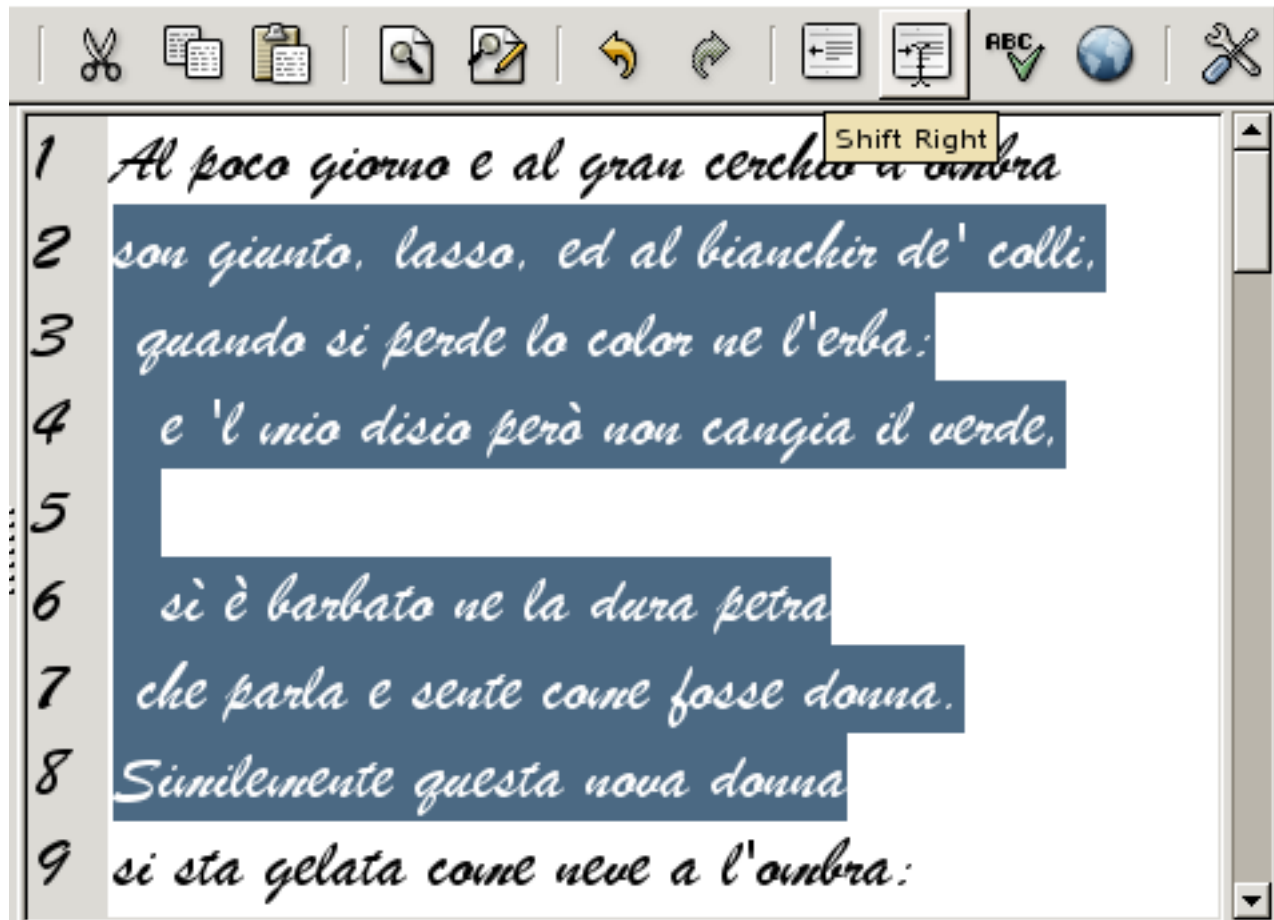


Figure VII.1. Indenting Part of a Text

2. Auto tag closing

By default, Bluefish will automatically produce closing tags for HTML and XML documents. For example, if you type `<p>` within an HTML document, bluefish will produce `</p>`. So, as soon as you finish typing a non-empty HTML tag, meaning the tag is supposed to have a closing tag, Bluefish will help you out and close the tag automatically. This feature can be turned off by unchecking the **Document** → **Auto Close HTML Tags (Ctrl-T)** menu option.

Bluefish has two modes for tag closing, an XML mode and an HTML mode. In XML mode, Bluefish will add a closing tag to any tag that is not closed itself with `/>`. In HTML mode, Bluefish excludes all known tags that do not need a closing tag, such as `
` and ``.

Bluefish will choose the mode based on the file type of the document. In the filetype preferences panel, the default mode for each file type can be set. See [Section 10, “Modifying file types” \[87\]](#) for more info.

3. Spell checker

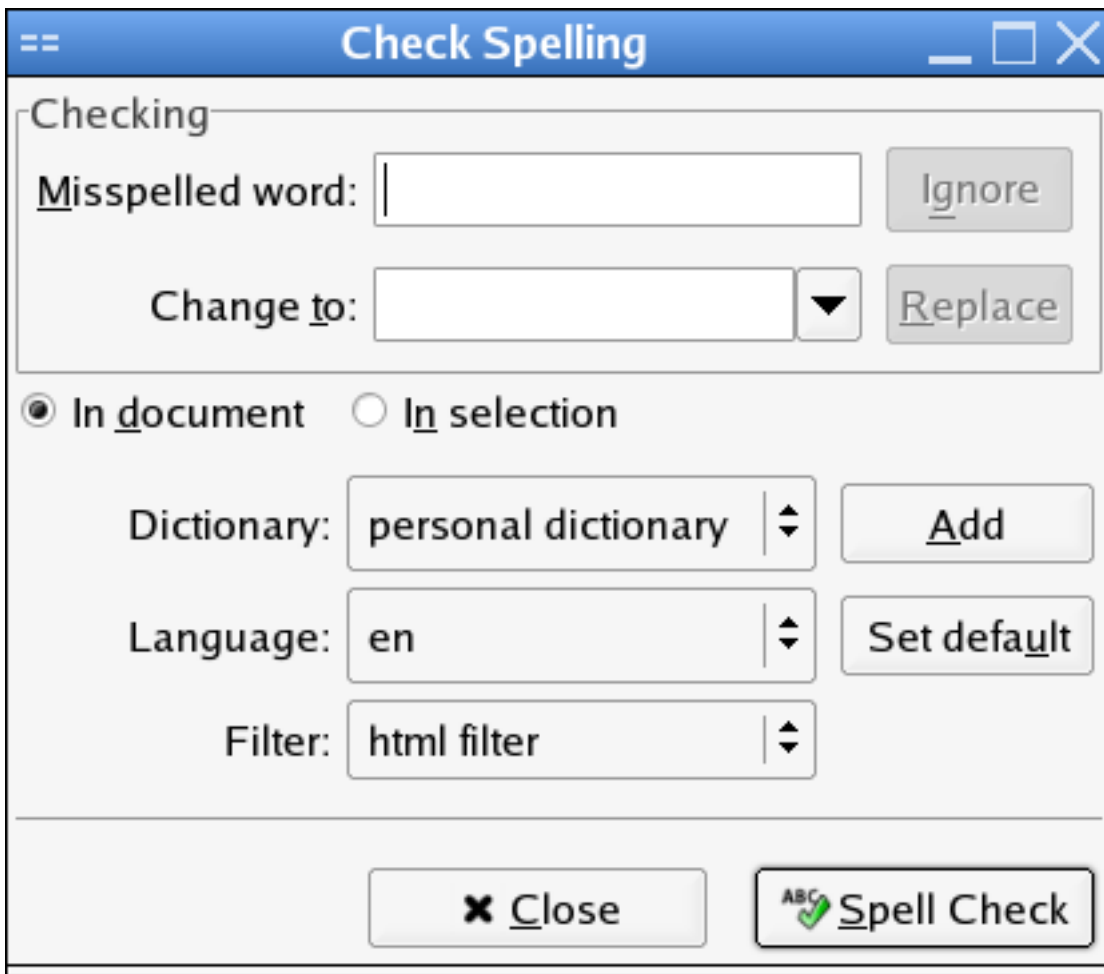


Figure VII.2. Bluefish Spell Checker

Bluefish uses [aspell](http://aspell.sourceforge.net/) [http://aspell.sourceforge.net/] for spell checking. If the aspell libraries are not installed on your system, then the spell checking feature will not be available. At the aspell web site you can also download dictionaries for many different languages.

To launch the spell checker, select **Document** → **Check Spelling...** or click on the **ABC** button on the main tool bar. The spell checker will launch in a separate window, which you can keep open as you edit files.

You have the option to check a whole document or just a selection, to use a personal or a session dictionary, and to choose the language depending on the installed dictionaries.

Click on **Spell Check** to start spell checking the current document.

You may want to set a default dictionary by first choosing the language in the **Language** pop up menu, then clicking on **Set default**.

Key words for different languages can be ignored using filters. Currently, the only filter is for HTML. If you want to help write more filters, join the [mailing list \[xix\]](#).

4. Function reference

The function reference browser contains reference information for different programming and markup languages. Currently, Bluefish comes with a PHP reference, a CSS 2.0 reference, an HTML reference, and a Python reference. The functions are grouped, depending on the language, by type, module, object, etc.

The function reference browser will display an info window on the bottom by checking the **Show info window** check box. In this window, information about the currently selected item is shown. The type of information shown can be configured in the right-click context menu (see Info Type later in this text).

In the reference browser's contextual menu, you can simply insert the text for the selected item by choosing **Insert**. Or, you can get a little help using **Dialog**, which launches a dialog window containing fields for the currently selected item's attributes

or parameters. For a summary of an item's usage, choose **Info**. The contextual menu is also accessible on a group of functions and at the top level of a reference.

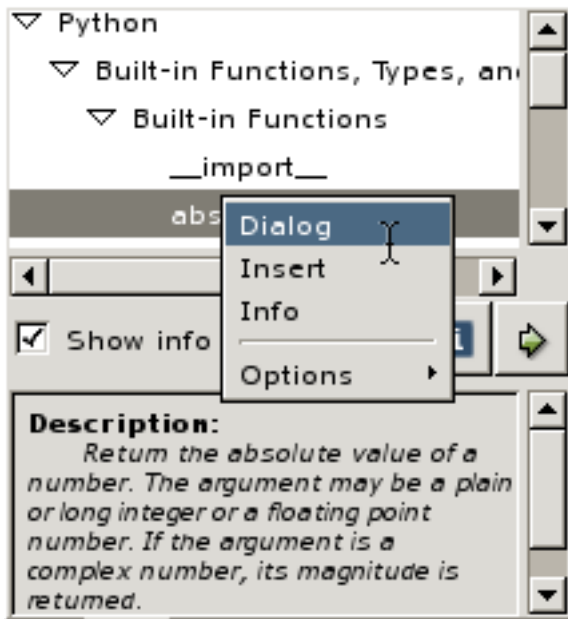


Figure VII.3. The Reference Browser Contextual Menu

The **Options** menu accessible via the contextual menu offers three actions:

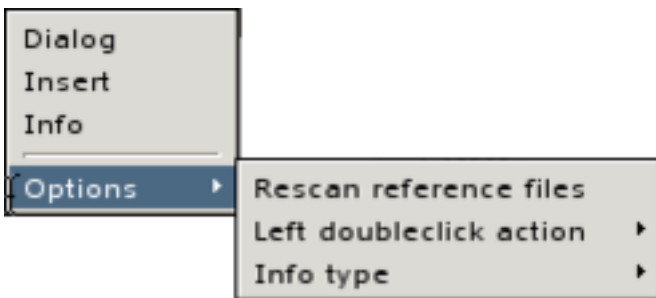


Figure VII.4. The Reference Browser Options Menu

- **Rescan reference files** in case you have customized one of them, so that the new items be available.
- **Left doubleclick action**, which can be:
 - **Insert** to insert the function in the document for latter parametrizing if needed
 - **Dialog** to insert the function in the document while filling in the parameters in a dialog window:

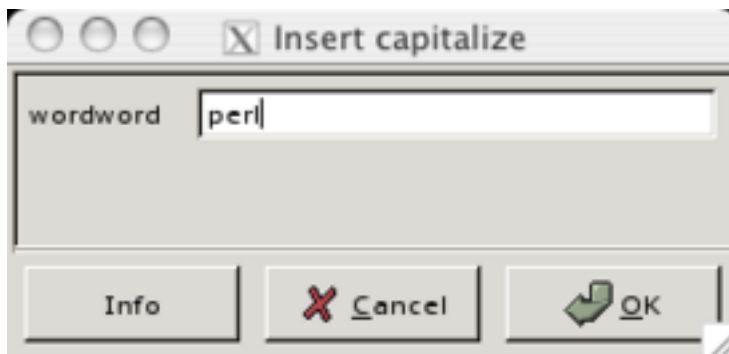


Figure VII.5. A Function Reference Dialog Window

- **Info** to display a window with all available info about the function:

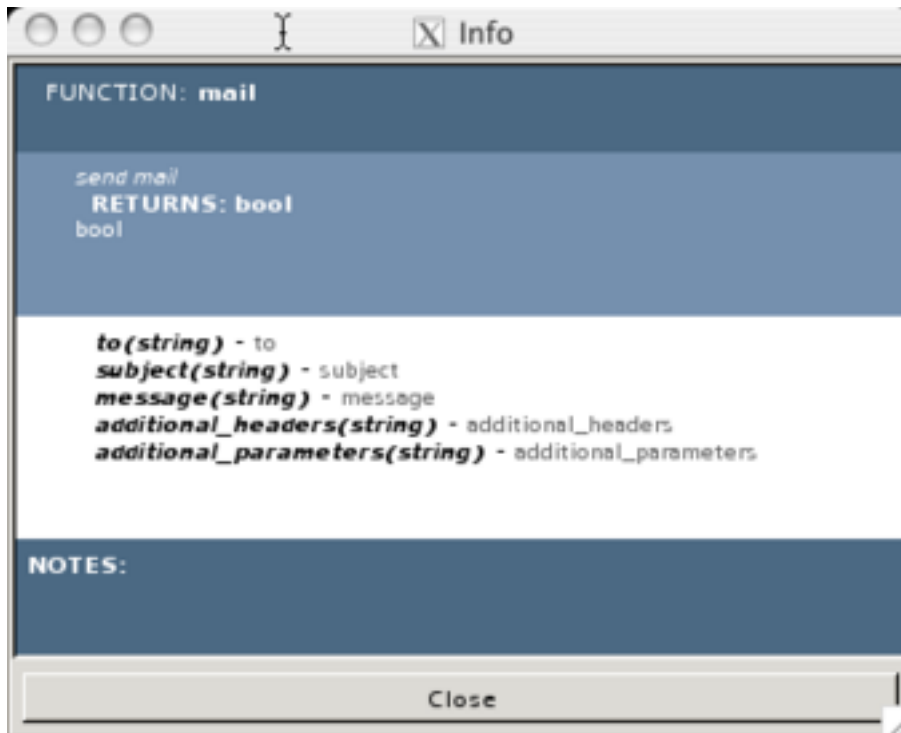


Figure VII.6. Info available for a Function

- **Info Type:** this is where you can customize what appears in the info window. It can be:
 - the function **Description** (this is the default)
 - the **Attributes/Parameters** of the function
 - some **Notes** about the function

5. HTML

HTML is obviously the most supported language in Bluefish. There is a special HTML tool bar with many dialogs, and two menus to work with tags:

- the **Tags** menu:

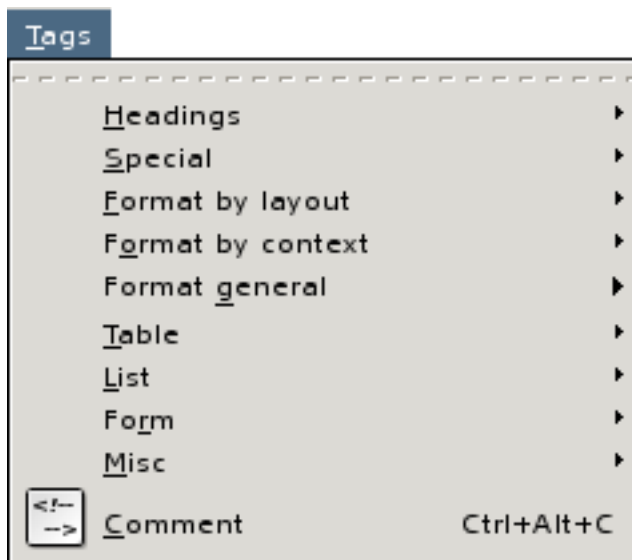


Figure VII.7. The HTML Tags Menu

- the **Dialogs** menu:



Figure VII.8. The HTML Dialogs Menu

The preferences have several settings on HTML style under **HTML**.

The HTML tool bar has two types of buttons. You can recognise each type by the tool tip if you move the mouse over the button:

- First there are buttons that will open a dialog for some HTML tag. These buttons have a tool tip that ends with three dots.



Figure VII.9. An HTML Button with a three-dotted Tool Tip

- Second, there are buttons that will directly insert text, these buttons do not have the dots in the tool tip.



Figure VII.10. A simple HTML Tool Tip Button

If you want to add an HTML tag around some block of text, select the block of text, use the HTML tool bar or the Tags or Dialogs menu to insert the tag. The opening tag will be inserted before the selected block, the closing tag after the selected block.

An existing tag can be edited by right-clicking the tag, and select **Edit tag** in the context menu. You can also place the cursor in the tag and use **Dialogs** → **Edit tag under cursor...** (**F3**). Not all tags, however, have a dialog, so this is not always possible. Colors in the style #RRGGBB can also be edited from the right-click context menu.

In the reference browser on the left panel there is an HTML reference available. All possible attributes and valid values can be found in this reference. See [Section 4, “Function reference”](#) [60] for more info.

5.1. Special find and replace features

There are several special search and replace actions in the menu **Edit** → **Replace special**. These can be used to convert special characters (like < and &), or ISO characters to their HTML entities, as well as to change the letters case.

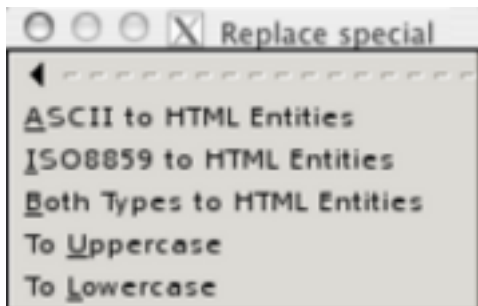


Figure VII.11. The Replace Special Menu

In all cases, when you want to replace some part of the text, you should first select the part to replace, then use the appropriated menu item.

5.2. Thumbnail generation

Bluefish can automatically generate thumbnails for images. A thumbnail is a small image, with a link to the larger image. Bluefish will create the small image based on your settings, and insert a `` tag in the file, and a `<a>` tag linking the original. The thumbnails are created in the same directory as the original sources.

The formats used for thumbnails may be png or jpg format. By default, the format used for thumbnails is png. You can change it in the **Images** panel of Preferences. For jpg images, the thumbnail extension is jpeg.

There are actually two thumbnail dialogs in Bluefish:

- an Insert thumbnail dialog, accessible from the **Dialogs** → **General** → **Insert Thumbnail...** (**Shift-Alt-N**) or from the **Standard bar** of the HTML toolbar.



Figure VII.12. The Insert Thumbnail Icon

- a Multi thumbnail dialog, only accessible from the **Standard bar** of the HTML toolbar.



Figure VII.13. The Multi Thumbnail Icon

The Insert thumbnail dialog is very straightforward. You select the image file, provide some `` tag attributes, choose the scaling, and press OK. The scaling factor is chosen by moving the slider directly under the image. The resulting image is previewed in the preview frame. Bluefish will create the thumbnail with extension `_thumbnail.png` or `_thumbnail.jpeg` (depending of the settings for images in the preferences).

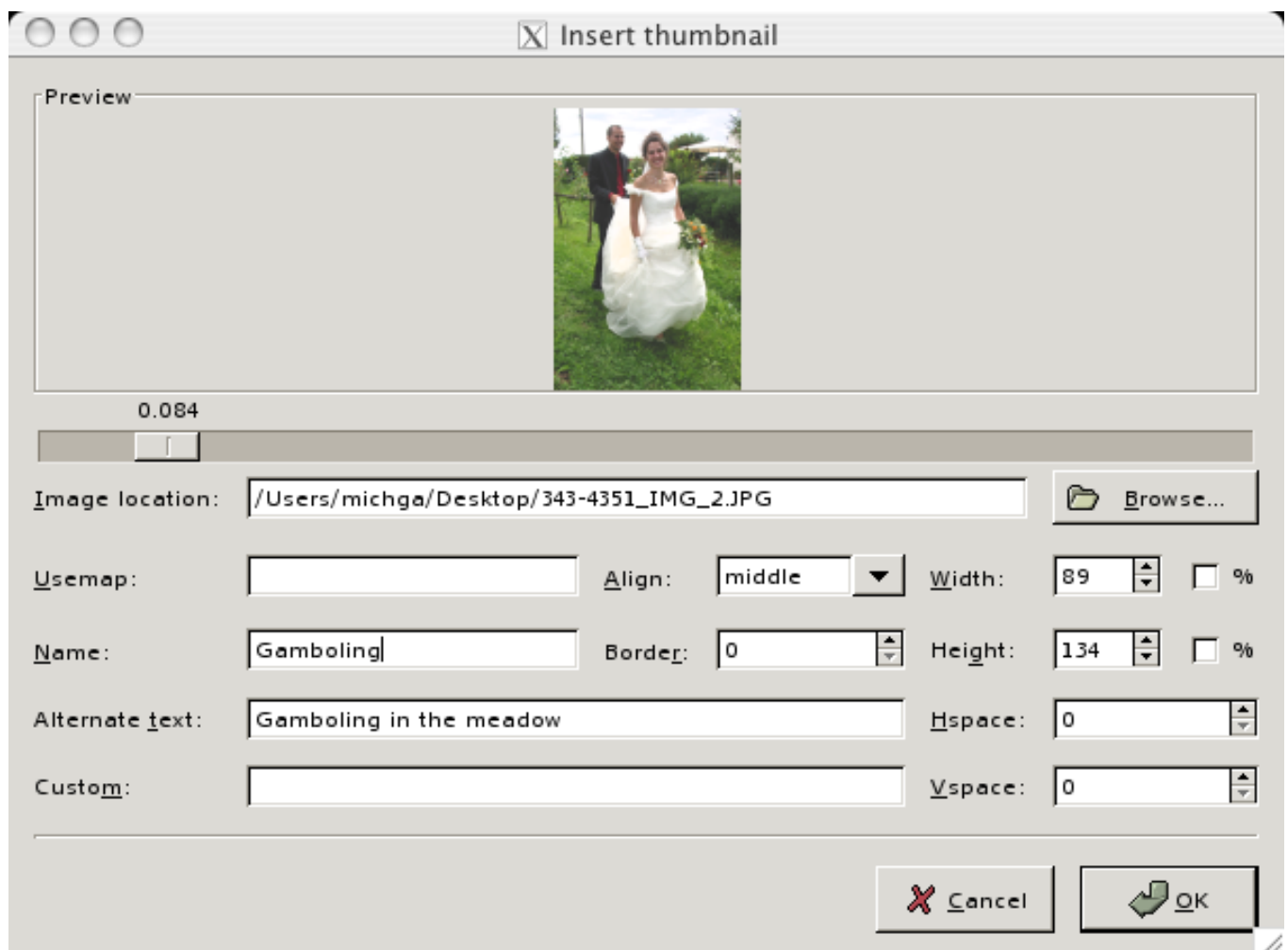


Figure VII.14. The Insert Thumbnail Dialog

If the source image is not accessible, change `webimage` to `image` in the **Select File** window loaded after clicking on **browse** for choosing an image. This way you can choose whichever format you want for the original sources. Another way to do it is to change the definition of `webimage` (see [Section 10, “Modifying file types” \[87\]](#)).

If that does not solve the problem, it is likely that the type of images you want to load is not defined yet in preferences. In this case, change the definition of `image` as explained in [Section 10, “Modifying file types” \[87\]](#)

As last resource, if you don't want to change the generic file types, you may choose **All files** in the pop up menu at the bottom of the **Select File** window.

The code generated for a png image and a png thumbnail looks like this:

```
<a href="/Users/michga/Desktop/343-4351_IMG_2.png">
</a>
```

and for a jpg image and jpg thumbnail:

```
<a href="/Users/michga/Desktop/343-4351_IMG_2.JPG">
</a>
```



You can perfectly mix jpg images with png thumbnails or the other way around.

If the html file exists beforehand, the paths to image and thumbnail are inserted relative to the location of the html file. On the contrary, if the html file does not exist beforehand, the full paths to the image and thumbnail are inserted in the code.

In the multi thumbnail dialog, you first choose the scaling method, then you set the corresponding width and/or height parameters. Finally, you may want to adjust the HTML code to be inserted for each image.

Scaling can be based on a fixed ratio, a fixed width, a fixed height, or a fixed width and fixed height (this last option does not keep the original aspect ratio!).

In the HTML code for each image, you can use several placeholders, such as:

- `%r` for the original filename
- `%t` for the thumbnail filename
- `%w` for the original width
- `%h` for the original height
- `%x` for the thumbnail width
- `%y` for the thumbnail height
- `%b` for the original file size (in bytes)

The default string is:

```
<a href="%r"></a>
```

After you have set up the scaling method and parameters, as well as the HTML code, you can select multiple images. Bluefish will create the thumbnails and insert the code.

Here is an example of two thumbnails created with a non nul border width and middle-aligned, with a fixed height and width, disregarding the aspect ratio.

The **Multi thumbnail** window is the following:

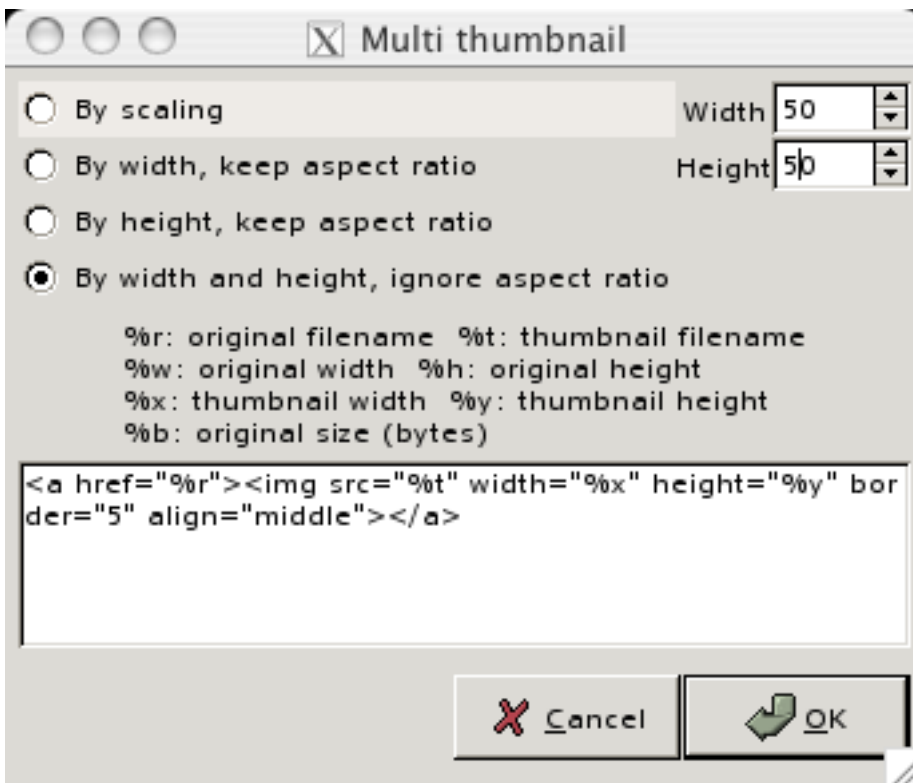


Figure VII.15. The Multi Thumbnail Dialog

And the generated code is:

```
<a href="/Users/michga/Desktop/tot/343-4351_IMG_2.png">
</a>
<a href="/Users/michga/Desktop/tot/343-4352_IMG_2.png">
</a>
```



Full pathnames are always used to reference original image sources. The paths to thumbnails are relative to the html file path if the html file already exists, while they are inserted as full paths when the html file does not exist.

Below is a full procedure to quickly generate thumbnails from a directory of image files. This example is purposely made with deprecated tags, so that you have an idea of what can be made with the variables. Feel free to adjust it when using CSS style sheets.

Procedure VII.1. Generating a photos album with multi thumbnails

1. Put the image files in a folder of their own
2. Open a new file in bluefish, click on the **Multi thumbnail...** icon in the **Standard** tab of the html tool bar.
3. Enter the scaling percentage in the **Scaling (%)** field
4. Change the html code as follows:

```
<tr><td><a href=\"%r\">
<img src=\"%t\"
width=\"%x\" height=\"%y\" border=\"0\"></a>
</td>
</tr>
<tr><td>Original: %w x %h</td></tr>
```

and click **OK**.

5. Choose the folder containing the images from the **Select files for thumbnail creation** window, click **Ctrl-A** to select all files, then click **OK**. The code generated by Bluefish will look like the following:

```
<tr><td><a href="/Users/michga/Desktop/photos/343-4344_IMG.JPG">
</a>
</td>
```

```
</tr>
<tr><td>Original: 1600 x 1065</td></tr>
<tr><td><a href="/Users/michga/Desktop/photos/343-4347_IMG.JPG">
</a>
</td>
</tr>
<tr><td>Original: 1600 x 1065</td></tr>
```

6. Use **Ctrl-A** to select the file's contents and click on the **Table** icon located in the **Tables** tab of the HTML tool bar to embed the code into table tags.



Figure VII.16. The Table Icon in the HTML Tool Bar

7. Save the file wherever you want.

If you want to add the file name and the file size in bytes, use this code:

```
<tr><td><a href="%r">
</a>
</td>
</tr>
<tr><td>%r: %w x %h (%b bytes)</td></tr>
```

6. Customizing the Quick bar

The **Quick bar** is a user defined tool bar. All HTML tool bar buttons can be added to the Quick bar by simply right-clicking the button and selecting **Add to quickbar**.

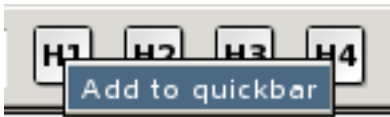


Figure VII.17. Adding an Element to the Quick Bar

And automatically you will see the element in the Quick bar:



Figure VII.18. The added Element in the Quick Bar

Note that you cannot add a pop up menu. Thus, if the item you want to add is inside a pop up menu (as is the code tag located in the **Context formatting** pop up menu of the **Fonts** tool bar), you have to first click on the pop up menu to display its contents, then to right click on the desired element to insert it in the **Quick bar**.



Figure VII.19. Adding a Pop Up Menu Element to the Quick Bar

If you want to remove items from the **Quick bar**, right-click them and select **Remove from Quick bar**.

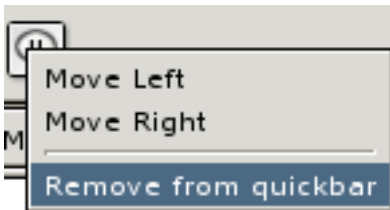


Figure VII.20. Removing an Element from the Quick Bar

You can also change the location of an element in the Quick bar. To do so, right-click the element and select **Shift left** or **Shift Right** as desired. The element will be moved to the left or to the right of its neighbor. Notice that this is not a drag and drop action; you may have to repeat the process if you want to move the element farther.

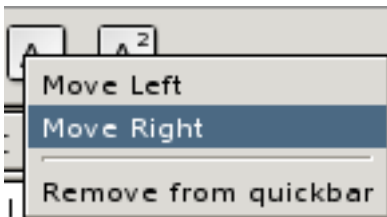


Figure VII.21. Moving an Element within the Quick Bar

7. Custom menu

To customize items in the **Custom menu** tool bar, you will use the **Custom menu** element:



Figure VII.22. Accessing the Custom Menu

The **Custom menu** → **Edit custom menu...** leads to the **Custom Menu Editor**. The **Load new** item allows you to load a new

menu in case you have directly changed the `custom_menu` file located in the `.bluefish` directory within your HOME directory, while **Reset** item allows you to return to the default custom menu under the same circumstances.

The `custom_menu` file is created upon install Bluefish and corresponds to some default entries, the ones you can see in the **Custom menu** tool bar. These will give you an idea what can be done with the custom menu.

The custom menu operates *only* on elements of the **Custom menu** tool bar, and allows you to:

- add "often used" items to an existing menu
- search and replace patterns to the **Replace** menu
- create new menus

The **Custom Menu Editor** is the place where you make all changes to the custom menu. The location for entries in the custom menu is defined by their menu path in the **Custom Menu Editor**:

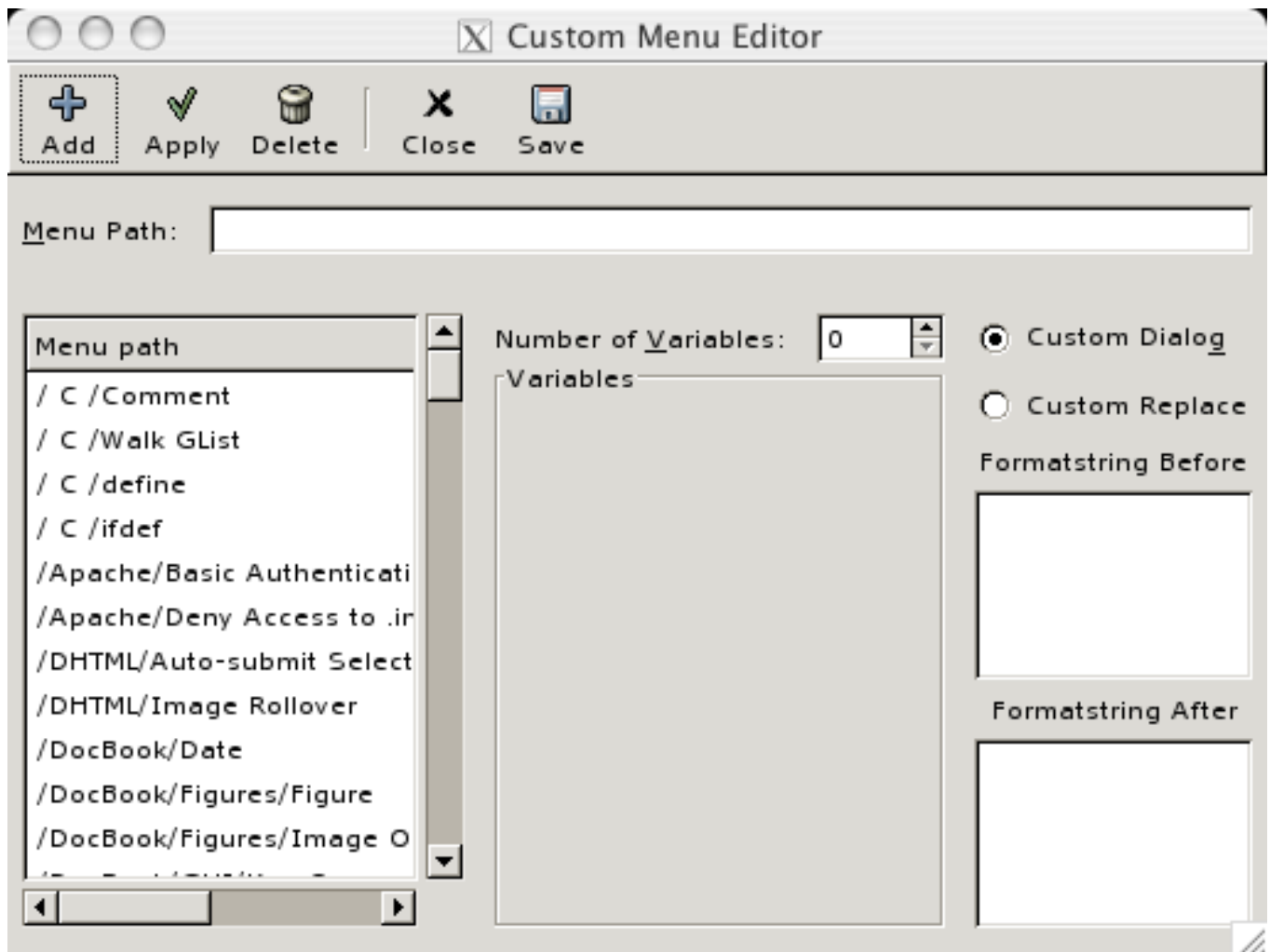


Figure VII.23. The Custom Menu Editor

It has four parts:

- The top one with all action buttons:
 - **Add** which adds new menu entries, once all necessary fields have been filled in
 - **Apply** which applies changes to an existing menu entry, once it has been edited
 - **Delete** which deletes the menu entry currently selected in the **Menu path** list
 - **Close** which discards changes
 - **Save** which saves the changes and exit the editor
- The **Menu Path** field below the buttons, to enter either an existing or a new menu path
- The **Menu path** list on the left side, which lists existing menu paths. A menu path looks like `/Main menu/submenu/item` or `/Main menu/item`. Here's an extract of the default custom menu paths:

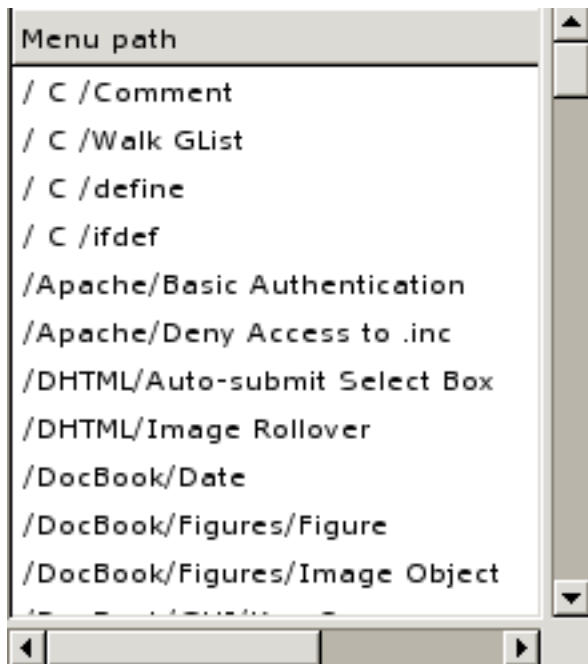


Figure VII.24. Extract of the default Custom Menu Path

- A custom part on the right side, whose contents changes depending of the type of menu. There are two types of items in the [Custom Menu Editor](#):
 - the Custom dialog, which will insert a string, optionally based on values asked in a dialog
 - the Custom Find and Replace, which will run a replace, also optionally based on values asked in a dialog. Here's how the [Custom Replace](#) dialog looks like:

Number of Variables:

Variables

☐ Custom Dialog

☒ Custom Replace

Replace:

Matching:

☐ Case Sensitive

Search Pattern

Replace String

Figure VII.25. The Custom Replace Dialog

7.1. Adding a custom menu dialog

The most simple custom dialog item has a `menupath`, for example `/MySite/author`, and a `Formatstring before`, for example *written by Olivier*. If you add this item, you can add this string by selecting the menu item.

Procedure VII.2. Adding a custom menu based on custom dialog

1. Choose **Custom menu** → **Edit custom menu...** in the custom menu tool bar.
2. Enter `/MySite/author` in the **Menu Path** field of the **Custom Menu editor**.
3. Enter *written by Olivier* in the **Formatstring Before** field located on the right.
4. Click on the **Add** button at the top.

Notice that upon adding the new entry, it is listed at the bottom of the **Menu path** list:

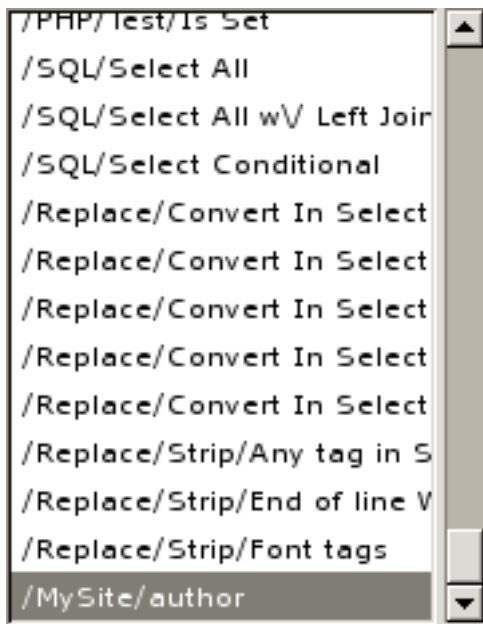


Figure VII.26. A new Custom Entry in the Menu Path List

5. Click on the **Save** button. This will add the menu to the Custom menu tool bar:



Figure VII.27. A new Menu in the Custom Menu Tool Bar

Note that the new menu is placed at the right end of the custom menu tool bar. When closing Bluefish and relaunching it, it will be placed in alphabetical order, except that the **Replace** menu will always be at the far right side.

In another example, you have a string you often need to set before and after some block of text, for example `<div class="MyClass">YourBlockOfText</div>`. To do it:

1. Open the **Custom Menu Editor**
2. Enter `/MySite/div with class` in the **Menu Path** field
3. Enter `<div class="MyClass">` in the **Formatstring Before** field
4. Enter `</div>` in the **Formatstring After** field
5. Click on **Add**, then on **Save**. The item will appear in the menu.

If you select some text:

```
La luna vino a la fragua
con su polisón de nardos.
El niño la mira, mira.
El niño la está mirando.
```

Figure VII.28. A Block of selected Text before Activating the Menu

And activate this menu item, the first bit of text is now added before the selection, and the second bit after the selection:

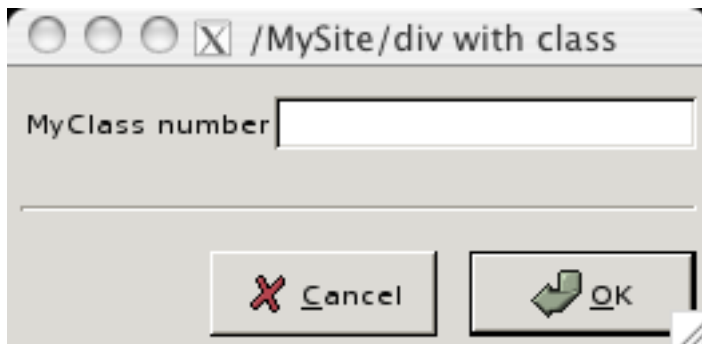
```
<div class="MyClass">La luna vino a la fragua
con su polisón de nardos.
El niño la mira, mira.
El niño la está mirando.</div>
```

Figure VII.29. A Block of Text after Activating the Menu

Suppose you want to improve this last example. You have both `MyClass1` and `MyClass2` and want to be able to choose the desired class when activating the menu. Here's how to do it:

1. Open the **Custom Menu Editor**
2. Browse the **Menu path** list to retrieve the `/MySite/class` with `div` entry and click on it to make appear its components in the **Menu Path** and **Custom Dialog** fields
3. Click on the top arrow of the **Number of Variables** pop up menu to get 1 in the field. As you see a **Variables** entry appears where you can enter the name for variable `%0`. As name we enter *MyClass number*
4. Now change the **FormatString Before** field to take this new variable into account, as following: `<div class="MyClass%0">`
5. Click on **Apply** so that your changes will be taken in account, and click on **Save** to update the menu.

If you now activate this menu after having selected a block of text, you will be presented with a new dialog asking you for the value of *MyClass number*:

**Figure VII.30. The new DIV with Class Dialog**

After entering the desired value, the same process as before will occur, using the value you provided. Here we have entered 1 as value:

```
<div class="MyClass1">La luna vino a la fragua
con su polisón de nardos.
El niño la mira, mira.
El niño la está mirando.</div>
```

Figure VII.31. The Block of Text after Entering the Value

You can use the **Return** and **Tab** keys to format the output.
Any variable can be used any times you want in the dialog.

7.2. Adding a custom replace dialog

Find and replace items are no different. The dialog has some more options, each of these options corresponds to the regular **Replace** dialog. Again you can use variables like `%0`, `%1` etc. to make a certain menu item more flexible.

Say you want to add title tags to a selection in a HTML page so that the user agent could render it either as a tool tip or as spoken words. To ease the discussion we will work on the following snippet of code:

```
<ul>
<li><a href="progsys01.html">Process scheduling</a> - 26/10/2002</li>
<li><a href="progsys02.html">Fork and Wait</a> - 02/11/2002</li>
</ul>
```

We will transform it into the following one:

```
<ul>
<li><a href="progsys01.html" title="blah Process scheduling">Process scheduling</a> \
- 26/10/2002</li>
<li><a href="progsys02.html" title="blah Fork and Wait">Fork and Wait</a> \
- 02/11/2002</li>
</ul>
```

where blah is any text you want.

The initial rendering:

- [Process scheduling](#) - 26/10/2002
- [Fork and Wait](#) - 02/11/2002

Figure VII.32. The HTML Page before the Transformation

will be transformed as is:

- [Process scheduling](#) - 26/10/2002
 - [Fork and Wait](#) - 02/11/2002
- This link leads to the page titled Process scheduling

Figure VII.33. The HTML Page after the Transformation

To do this, we need to express the `yourstring` part of the initial snippet of code as a Perl regular expression (see [Section 5.3, "Find and Replace Using Regular Expressions" \[54\]](#) for full details):

- a href=" will be expressed as is and embedded into parentheses to retrieve it as \0 variable.
- yoururl will be expressed as ([^"]+) to match one or more non double quote characters and retrieve it as \1 variable.
- The second double quote will be expressed as is and embedded into parentheses to retrieve it as \2 variable.
- The second > sign will be expressed as is and embedded into parentheses to retrieve it as \3 variable.
- yourstring will be expressed as ([^>]+) to match one or more non > characters and retrieve it as \4 variable.
- will be expressed as is and embedded into parentheses to retrieve it as \5 variable.

Thus, the search string will be: `()([^\>]+)()`

The replace string should be of the form: `yourstring`

Expressed as a regular Perl replacement expression, it will be as simple as: `\0\1\2 title=\2%0 \4\2\3\4\5` where %0 will match yourvariablestring, that is the value entered in the **Title** field of the **Replace** dialog at activating time.

Procedure VII.3. Adding a custom menu based on replace dialog

1. Choose **Custom menu** → **Edit custom menu...** in the custom menu tool bar.
2. Browse the Menu path list to retrieve the **/Replace/Convert in Selection/<td> to <th>** and click on it to make appear its components in the Menu Path and Custom Replace fields.
3. Change the Menu Path to **/Replace/Anchor/Add Title**.
4. Click on the top arrow of the Number of Variables pop up menu to get 1 in the field. Enter **Title** in the **%0 Variables field**.
5. Change the **Matching** pop up menu to **perl regular expressions**.
6. Change the **Search Pattern** field like this:

```
(<a href=")([^\"]+)(")(>)([^\>]+)(</a>)
```

7. Change the **Replace String** field like this:

```
\0\1\2 title=\2%0 \4\2\3\4\5
```

8. Click on the **Add** button.

The **Custom replace** dialog should have the following appearance:

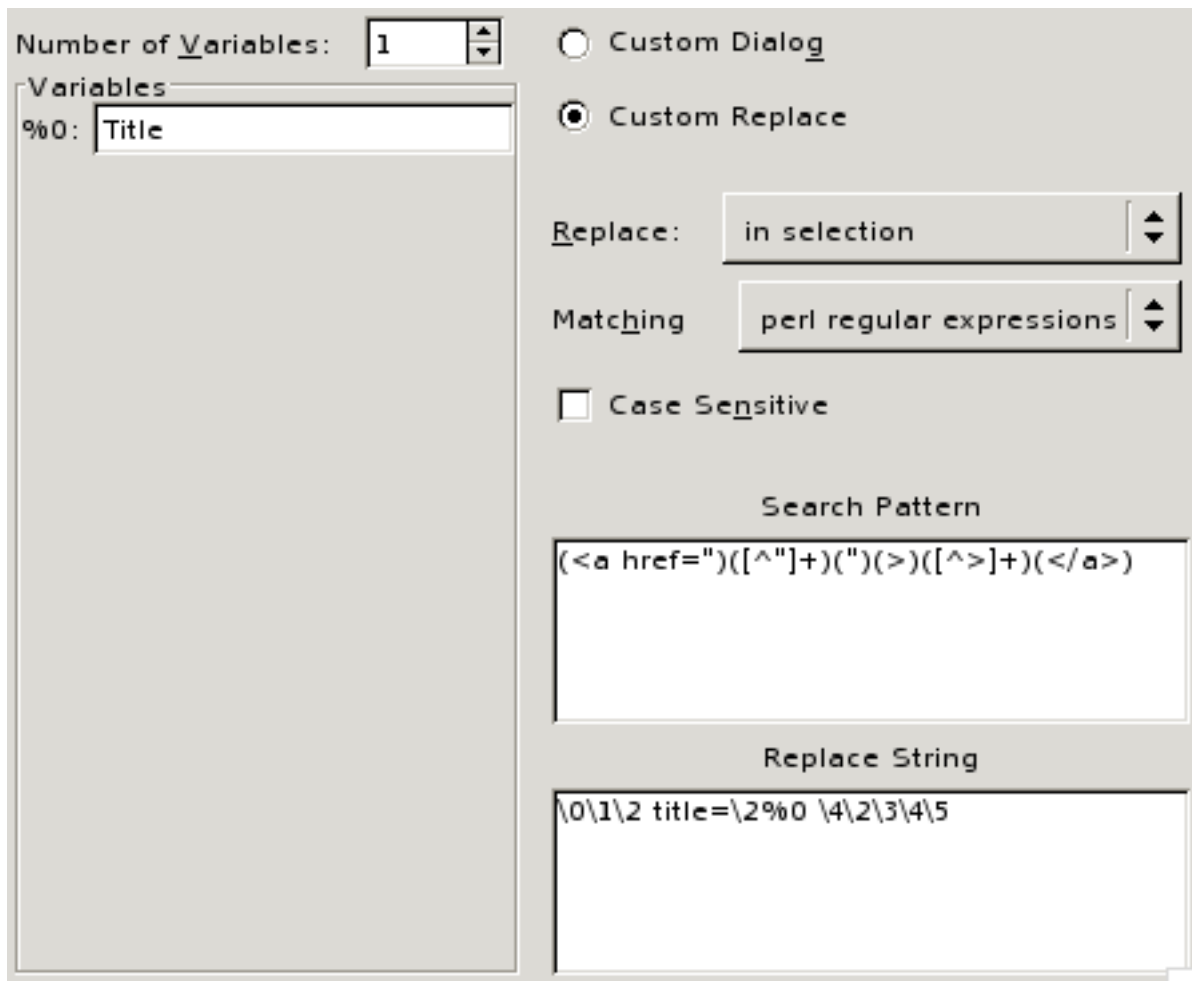


Figure VII.34. The Custom Menu Replace Dialog filled in

9. Click on the **Save** button.

To use the new menu item, select the lines to be changed in the HTML file and activate **Replace/Anchor/Add Title** in the custom menu bar. Fill in the dialog as follows:

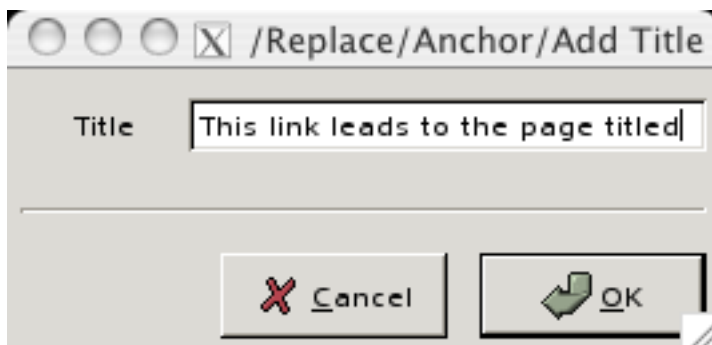


Figure VII.35. The Add Title Dialog

Click OK to proceed.

8. External programs, filters

The **External** menu provides a quick access to external default or user-added programs and text filters. It is divided into three parts by default (see [Chapter VIII, Customising Bluefish \[81\]](#) for layout change):

- The **Outputbox** submenu for text filters. Its name is derived from the output box shown at the bottom of the document window, where you can see the output of the process, when activating it.
- The **Commands** submenu for external programs.
- All other items are browsers. They are launched as subprocesses, hence you need to detach them to avoid freezing bluefish until the external program quits.



Typically all programs and filters apply to the current document. Nevertheless it is possible to invoke a program without applying it to the current document. On the contrary, it is not possible to apply text filters to anything but the current document.

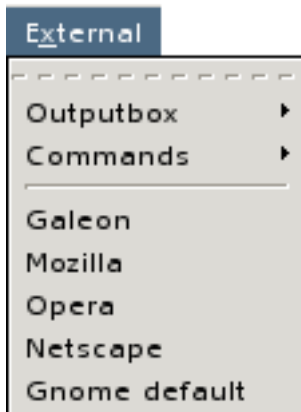


Figure VII.36. Bluefish External Menu

Customization of the **External** menu is performed in different parts of the **Edit Preferences** dialog:

- Items in the **Outputbox** submenu in the **Output parsers** tab.
- Items in the **Commands** submenu in the **Utilities and filters** part at the bottom of the **External programs** tab.
- Top level items in the **Browsers** part at the top of the **External programs** tab.

8.1. Customizing browsers

The **Browsers** panel in Preferences shows the items in the same order as in the **External** menu:

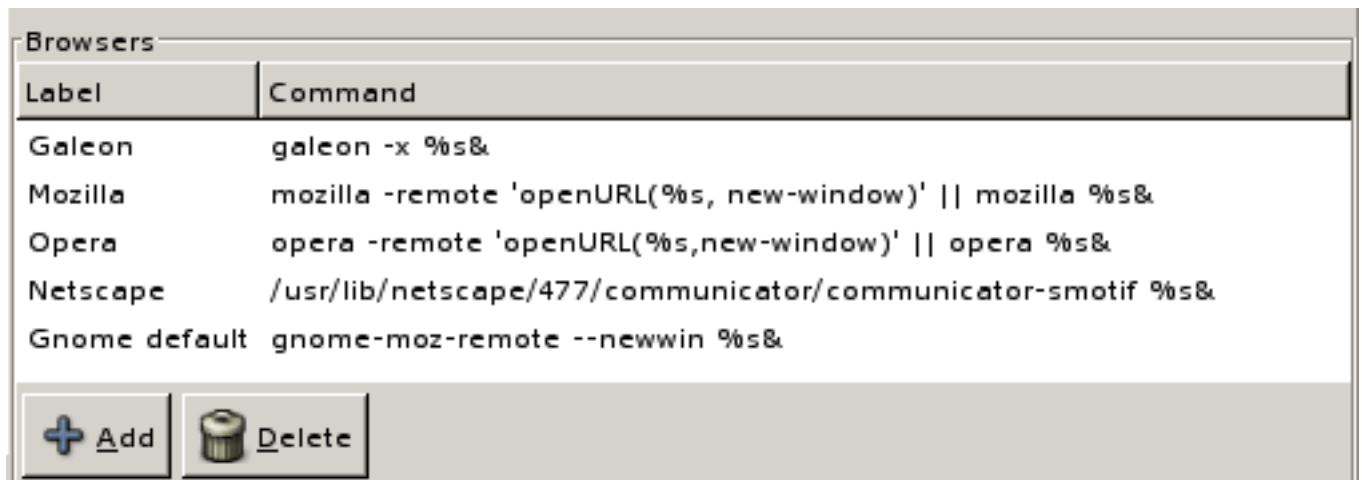


Figure VII.37. The Browsers Panel in Preferences



The first line in the panel will be the browser selected when clicking on the **View in browser** button in the main tool bar.

If you want to change the order of the browsers, apply the following steps:

Procedure VII.4. Changing the order of browsers items

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **External programs** tab to display the **Browsers** panel.
3. Click near the left border of the browser's line you want to move. The whole browser's line will be highlighted:



Figure VII.38. Selecting the Browser's Line to be Moved

4. While maintaining the click, drag the selected line over another line, until you reach the place you want, so that the

selected line covers entirely the latter one. The cursor will change its appearance and the dragged line will be shown as a framed line:

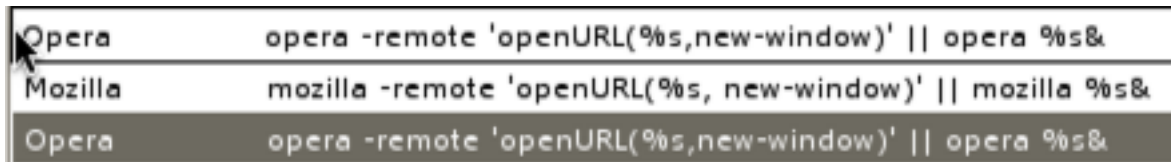


Figure VII.39. Dragging the Browser's Line



To drag a line to the end of the list, drag it until a thin line appears below the last item:



Figure VII.40. Dragging the Browser's Line to the Bottom



If you change your mind, drag the line over its original place and release the mouse button. There will be no change.

5. Release the mouse button to drop the line at the desired place.
6. Click on the Apply button to save the change if you plan to make further changes in the panel, otherwise click on the OK button to save the change and close the **Edit preferences** panel.

If you want to customize one of the browsers supplied by default, use the following procedure:

Procedure VII.5. Customizing an existent browser

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **External programs** tab to display the **Browsers** panel.
3. Click on the **Command** region of the browser's line you want to change. The line will be highlighted.
4. Double-click on the same location to allow editing. The line will be framed.
5. Make the desired change
6. Click on the OK button to save and close the panel.

To add a new browser, proceed as follows:

Procedure VII.6. Adding a new browser

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **External programs** tab to display the **Browsers** panel.
3. Click on the Add button. A new line will be shown, with an **Untitled** label.
4. Double-click on the label to allow editing, and enter the string you want to appear in the **External** menu.
5. Double-click in the **Command** zone and enter the command followed by the & sign to detach it from the main bluefish process, for example:
amaya %s &
6. Click on the OK button to save and close the panel.

To delete a browser, just click on the Delete button.

Though nothing impedes you to put any command (not necessary a browser) in the panel for quick access, you may want to avoid to put it at the top range, since it will be somewhat strange to click on **View in browser** to launch **abs** for example.

8.2. Customizing Commands menu

To add items to the **External** → **Commands** submenu, you use the **External programs** tab of the **Edit Preferences** panel:

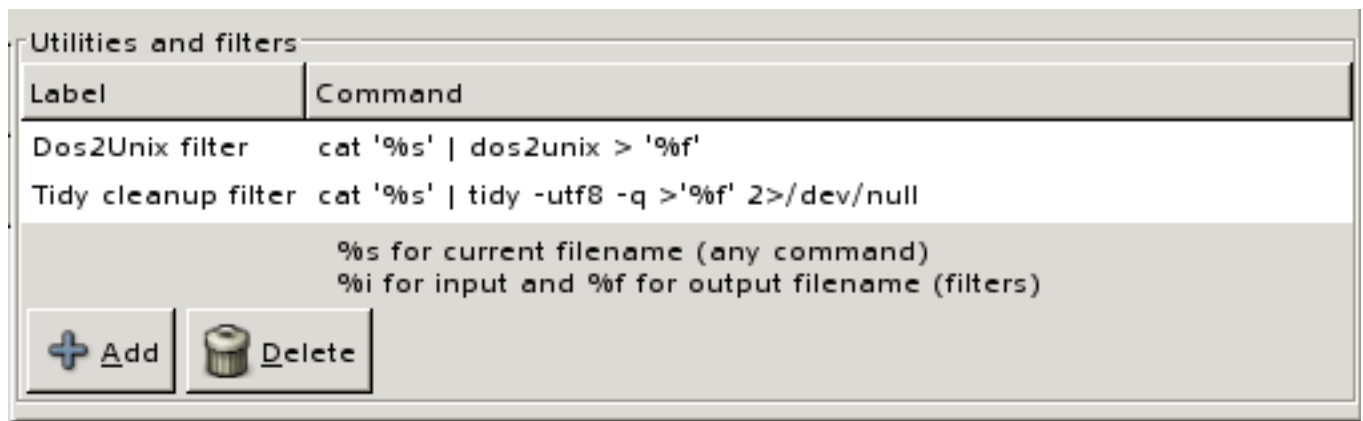


Figure VII.41. Utilities and Filters Panel in Preferences

You add, modify, delete, move commands or text filters the same way as described in [Section 8.1, “Customizing browsers”](#) [75].

Bluefish will apply the supplied command on the current document, while representing the document as it is before the command is applied by %s and the document after the command has been applied by %f. Usage of the %i parameter is not implemented yet. You should embed those parameters into simple quotes to prevent special characters to be interpreted by the shell.

Usage of the parameters depends on the command:

- If the command does not operate on the file, as **xterm**, you just supply it as you would in an xterm, detaching it to avoid bluefish freezing with:

```
xterm &
```

- If the command does operate on the file, but not on the file's contents, as **chmod**, you supply it as you would in an xterm, using %s as a reference to the current document:

```
chmod +x '%s'
```

- If the command operates on the standard input device by default, as **tidy**, you will have to redirect the document's contents, i.e. %s, with **cat** for example, to the standard output device, pipe the result so that it will be used as standard input device for the command, then redirect the result of the command to the document, i.e. %f, as in:

```
cat '%s' | tidy 'someoptions' > '%f'
```

- If the command operates on file's contents, as **sed**, you should use input, i.e. %s and output, i.e. %f redirection to feed the command with the right parameters, as in:

```
sed -e 'somesedcommand' < '%s' > '%f'
```

As those parameters are used internally to create temporary files, you cannot use them to modify the name of the final document for example. But you can redirect the standard output to a named file, if you do not want to override the current document, as in:

```
sed -e 'somesedcommand' < '%s' 1 > 'namedfile'
```

Here is an example to get rid of hard-coded /usr in a source file:

Procedure VII.7. Adding a Commands menu item

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **External programs** tab to display the **Utilities and filters** panel.
3. Click on the **Add** button. A new line will be shown, with an **Untitled** label.
4. Double-click on the label to allow editing, and enter the string you want to appear in the **External** menu.
5. Double-click in the **Command** zone and enter:

```
sed -e 's|\/usr|${PREFIX}|g' < '%s' > '%f'.
```



We need to escape the slash in /usr with a backslash to avoid interpretation by the shell.

6. Click on the **OK** button to save and close the panel.

8.3. Customizing Ouputbox menu

Items within the **External** → **Outputbox** submenu allow for programs to give feedback by opening an output box at the bottom

of Bluefish's main window.

Here is an example showing the output box after using the **External** → **Outputbox** → **tidy HTML validator** item on an html file with an on purpose error:

Filename	Line	Output
	3	Warning: <table> attribute "summary" lacks value
		Info: Doctype given is "-//W3C//DTD HTML 4.01 Transitional//EN"
		Info: Document content looks like HTML 4.01 Transitional

Figure VII.42. The tidy Output Box

The contents of the resulting output box are based upon scanning the output of the supplied command, as it appears in an xterm, with a given regular expression and filling in the various fields of the output box with the desired parts of that regular expression. The **Output parsers** tab of the **Edit preferences** panel provides you with a model to do that:

Outputbox						
Name	Pattern	File #	Line #	Output #	Command	Show all output
make	([a-zA-Z0-9/_-]+):([0-9]+):(.*)	1	2	3	make	<input checked="" type="checkbox"/>
weblint HTML checker	([a-zA-Z0-9/_-]+)\([([0-9]+)\):(.*)	1	2	3	weblint '%s'	<input checked="" type="checkbox"/>
tidy HTML validator	line ([0-9]+) column [0-9]+ - (.*)	-1	1	2	tidy -qe '%s'	<input type="checkbox"/>
javac	([a-zA-Z0-9/_-]+):([0-9]+):(.*)	1	2	3	javac '%s'	<input type="checkbox"/>
xmllint XML checker	([a-zA-Z0-9/_-]+)\([([0-9]+)\):(.*)	1	2	3	xmllint --noout --valid '%s'	<input checked="" type="checkbox"/>
php	([a-zA-Z0-9/_-]+):([0-9]+):(.*)	1	2	3	php '%s'	<input checked="" type="checkbox"/>

Figure VII.43. The Output Parsers Tab in Preferences Panel

The **Outputbox** panel comprises 7 fields:

- The **Name** field, a character string which will appear as the item in the **Outputbox** menu.
- The **Pattern** field, a Perl regular expression which describes the command output, so that some of its parts could be used in the following fields.

Let's use an example: say you have a ruby script named `foo.rb` with the following line in it:

```
put Hello Word
```

When executing **ruby -d foo.rb** in an xterm, the output is:

```
Exception `NoMethodError' at foo.rb:1 - undefined method `put' for main:Object
foo.rb:1: undefined method `put' for main:Object (NoMethodError)
```

The second line can be parsed with the following Perl regular expression:

```
([a-zA-Z0-9/_-]+):([0-9]+):(.*)
```

The first part embedded into parentheses will match the script name, i.e. `foo.rb`; the second part will match the line, i.e. `1`; the third part will match the remaining. See [Section 5.3, “Find and Replace Using Regular Expressions” \[54\]](#) for some explanation on using regular expressions within bluefish.

- The **File #** field, a part number matching the filename in the Perl regular expression given in the **Pattern** field. Note that the first part is numbered 1, the second 2, etc. If you do not want that the part be shown, put -1 in it.
- The **Line #**, a part number matching the line number in the regular expression, here it will be 2, as same rules apply as in the **Filename #** field.
- The **Output #** field, a part number matching the desired part in the regular expression, typically the remaining of the line, here it will be the third and last part. Again, same rules apply as in the **Filename #** field.
- The **Command** field, the command to execute on the current document, internally named `%s`. Here it will be: **ruby -d '%s'**. Notice that you should embed the reference to the current document, if any, within parentheses to avoid interpretation at run time.
- The **Show all output** check box, which can be checked to show all output not matching the Perl regular expression. Here it is not needed, since the regular expression matches all output.

You add, modify, delete, move output boxes the same way as described in [Section 8.1, “Customizing browsers” \[75\]](#).

Procedure VII.8. Adding an Outputbox menu item

1. Execute the desired command in an xterm with some error either in the command or in the file which it is applied on, in order to know how the errors are outputting.
2. Build a Perl regular expression based on the output, so that the filename, the line number and the error message be retrieved.
3. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
4. Click on the **Output parsers** tab to display the **Outputbox** panel.
5. Click on the Add button. A new line will be shown, with an **Untitled** label.
6. Click "Add" to add a new item.
7. Double-click on the **Name** field to give the command a name.
8. Double-click on the **Pattern** field and fill it in with the Perl regular expression you have built previously.
9. Double-click on the **File #** field and give the number for the subpattern matching the filename (-1 for none).
10. Double-click on the **Line #** field and give the number for the subpattern matching the line number (-1 for none).
11. Double-click on the **Output #** field and give the number for the subpattern matching the actual error message (-1 for none).
12. Double-click on the **Command** field and enter the command to execute in the form **command options '%s'**, %s being the current filename.
13. Toggle the "Show all output" check box to show output NOT matching the regular expression, if needed.



Of course, it is also possible to add these items by editing the file named `~/.bluefish/rcfile_v2` found in the user's home directory. The fields are delimited by colons and correspond to those found in the GUI.

Chapter VIII. Customising Bluefish

We have already seen how to customize the [quick bar \[67\]](#), the [Custom menu \[68\]](#), and the [External menu \[74\]](#). Here are some other possibilities, most of them being made through the [Edit preferences](#) panel, accessible from the [Preferences...](#) icon in the main tool bar or from the [Edit](#) → [Preferences](#) menu item.

1. Modifying shortcut keys

Many menu entries are accessible via key combination, also called a shortcut. For example, pressing the **Ctrl-S** keys saves the current file to disk. If available, shortcut key combinations are shown on the right of the menu entry.

To add or change a shortcut, move the mouse over the desired menu entry, and press the key combination you would like to use. Immediately this combination will show up on the right of the menu entry.

Here's a shortcut added to the [File](#) → [Open URL...](#) menu item:



Figure VIII.1. Adding a Shortcut to a Menu Item

To remove a shortcut, press the backspace key when you move the mouse over a menu entry to remove the shortcut.

To save the shortcut key combinations for later Bluefish sessions, use [Edit](#) → [Save Shortcut Keys](#). This will store the settings in the `~/ .bluefish/menudump_2`.



If you want to restore the default combinations simply remove this file and restart Bluefish.



Be aware that if you give a menu entry the same shortcut as another one, the shortcut of the latter will be lost.

2. Showing hidden files and folders

By default, invisible files and folders are not shown in the file browser tab of the side panel.

If you want to see them at a given level of the files system hierarchy, right click on the desired folder name in the file browser within the side panel and toggle [Show hidden files](#) in the contextual menu.

Here is how to view all visible files and folders in the whole system:

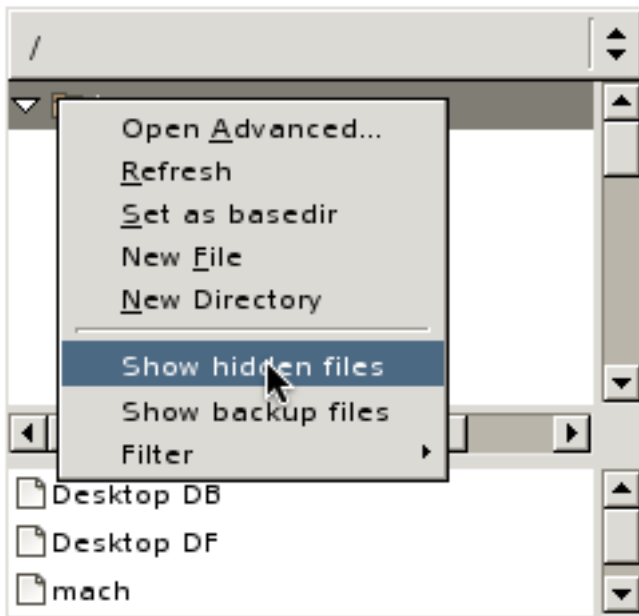


Figure VIII.2. Turning Files and Folders Visibility ON



This feature is very convenient for Mac users when used with caution, since combined with the [Delete](#) contextual menu in the file browser, it allows you, for example, to get rid of files generated by cvs on conflicts within bluefish.

3. Showing backup files

By default, backup files are not shown in the file browser tab of the side panel.

You may turn on their visibility at a given level of the file system by right clicking on the desired folder name in the file browser within the side panel and toggle **Show backup files** in the contextual menu.

4. Editor appearance

Most of the editor appearance depends on your GKT theme, which may be customized through the `~/.gtkrc-2.0` resource file.

Parts that you may want to customize through that resource file are among others:

- the background color of the editor
- the colors of GUI elements
- the position of arrows in a drop down list

You will find examples of themes resource files while searching for a `gtkrc` file in a `gtk-2.0` folder within the various directories under `$prefix/share/themes/`, where `$prefix` is your installation prefix (it may be `/usr`, `/usr/local`, `/sw`, `/opt`, etc.).



You should not customize those files, instead customize `~/.gtkrc-2.0`. If the file does not already exist in your home directory, just create it with: **`touch ~/.gtkrc-2.0`**

Here is an example made on a Crux theme:

```
style "bluefish"
{
  # For up and down arrows grouped together at right side
  GtkNotebook::has_secondary_forward_stepper = 1
  GtkNotebook::has_secondary_backward_stepper = 1

  # Editor background color
  # (background of editor view)
  base[NORMAL]="#fcfff5"

  # GUI normal background color
  # (most of the GUI)
  bg[NORMAL]="#dbe9e9"

  # GUI highlighted background color
  # (GUI when mouse over elements)
  bg[PRELIGHT]="#c6e9e9"

  # GUI unactive background color
  # (GUI disabled elements)
  bg[INSENSITIVE]="#9fb2b2"

  # GUI active background color
  # (GUI enabled elements)
  bg[ACTIVE]="#c7d4d4"
}
class "GtkWidget" style "bluefish"
```



You may give any name to the style on the first line, provided that you use the same on the last line. The customization applies to any Gtk application.

It will give this appearance to bluefish:

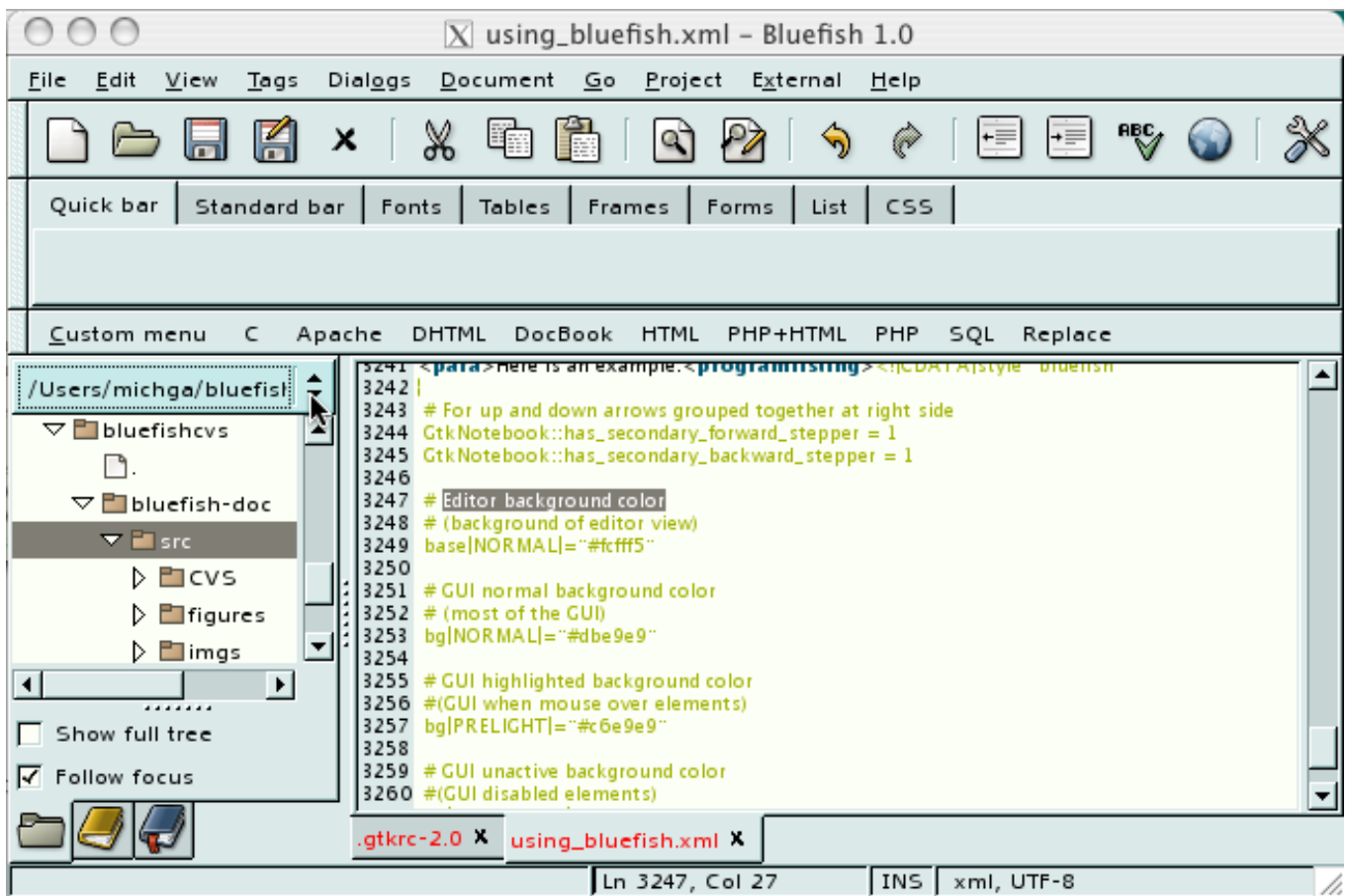


Figure VIII.3. Bluefish with a customized Gtk Theme

Other options for the Editor are available in the **Editor** tab of the **Edit preferences** panel accessible via the **Edit preferences...** button in the main tool bar. In particular you may want to customize the font of the editor, the end of line wrapping, and the undo history size:

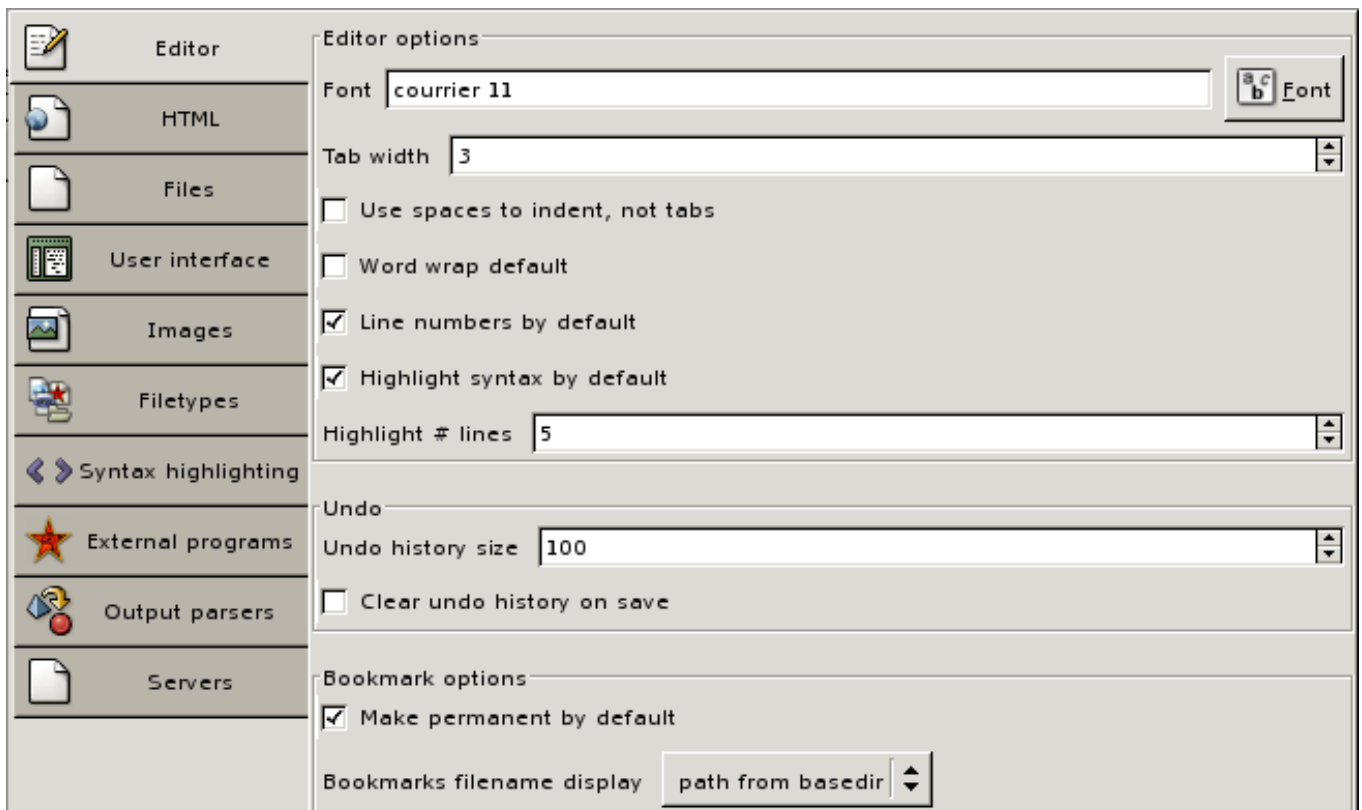


Figure VIII.4. The Editor Tab in Preferences

5. Customizing the bookmarks path

When you add bookmarks to document, the name of the file it refers to is displayed from the base directory. You can choose another path from the **Bookmarks filename display** pop up menu in the **Editor** tab of the **Edit preferences** panel:

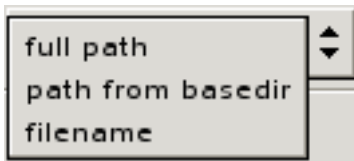


Figure VIII.5. The Bookmarks Path Pop up Menu in Preferences

6. Customizing the html tags style

The **HTML** tab of the **Edit preferences** panel provides you with some options to change the style of the html tags:

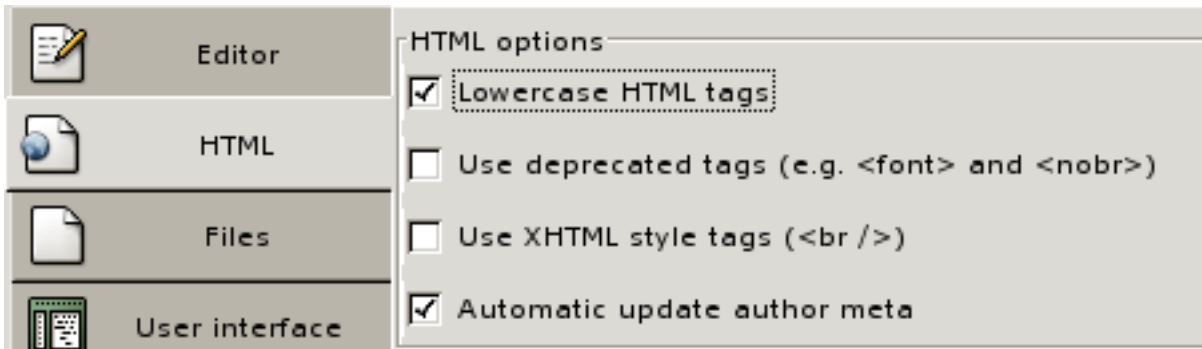


Figure VIII.6. The HTML Tab in Preferences

7. Changing the author meta tag on the fly

One interesting feature in the **HTML** tab of the **Edit preferences** panel is that you can let bluefish update the author meta tag on save.

Let's say you created an html file with an author meta tag while you were logged in as user *foo*. On save bluefish will fill up the contents attribute of the author meta tag with the full name associated with the *foo* user:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta name="generator" content="Bluefish">
<meta name="author" content="Michèle Garoche">
<title>A test for author meta tag's update</title>
</head>
<body>
<p>Just a test.</p>
</body>
</html>
```

Figure VIII.7. The Author Meta Tag filled in on Save

You share this html file with another user *bar* or you change the owner of the file to *bar*. When you modify the html file while logged in as user *bar*, the author meta tag is updated to reflect the new author on save, providing that the user *bar* has write permission on the file:


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta name="generator" content="Bluefish">
<meta name="author" content="Kristell Aubry">
<title>A test for author meta tag's update</title>
</head>
<body>
<p>Just a test.</p>
</body>
</html>
```

Figure VIII.8. Update of the Author Meta Tag on Save



If you do not want that the author meta tag be changed while editing the file under another user's login, uncheck the box.

8. Customizing files handling and browsing

The **Files** tab of the **Edit preferences** panel allows you to set some options related to the way files are handled and displayed in the file browser.

Figure VIII.9. The Files Preference Panel

8.1. Setting the encoding meta tag on save

Apart from setting the default character encoding in the **Files** tab of the **Edit preferences** panel, you may also instruct bluefish to set the encoding meta tag when you modify the document character set encoding.

Note that, if the encoding meta tag does not exist, it is inserted in the file, otherwise it is changed. Either modification occurs

immediately.

8.2. Setting the default base directory

You can set a default base directory in the **Files** tab of the **Edit preferences** panel.

This directory will serve as the initial point for the file browser.

8.3. Merging file browser views

By default, the file browser uses separate views for files and directories.

You can have a single view by unchecking the **Use separate file and directory view** option in the **Files** tab of the **Edit preferences** panel.

8.4. Backup files

By default, a backup file is created on save in the same directory as the original file based on the same filename with the exception that a ~ suffix is added. This backup file is deleted on closing the file.

You can change this behaviour in the **Files** tab of the **Edit preferences** panel.

When the backup fails to be created, you can choose what to do:



Figure VIII.10. Choosing an Action on Backup Failure

8.5. Using multiple instances of a file

A nice feature of bluefish is it allows you to open multiple instances of a file. Combined with either launching two instances of bluefish or opening the same file in two windows, it eases the modification of a file in one window while browsing it in another one.

This feature can be disabled in the **Files** tab of the **Edit preferences** panel.



Be aware that the last closed instance of the file wins. Hence it is important that you remember which instance is the modified one. You can, for example, always open the file to be modified on the left side of your screen, the file to be browsed on the right side.

9. Customizing the user interface

The **User interface** tab of the **Edit preferences** panel allows you to customize most part of the user interface:

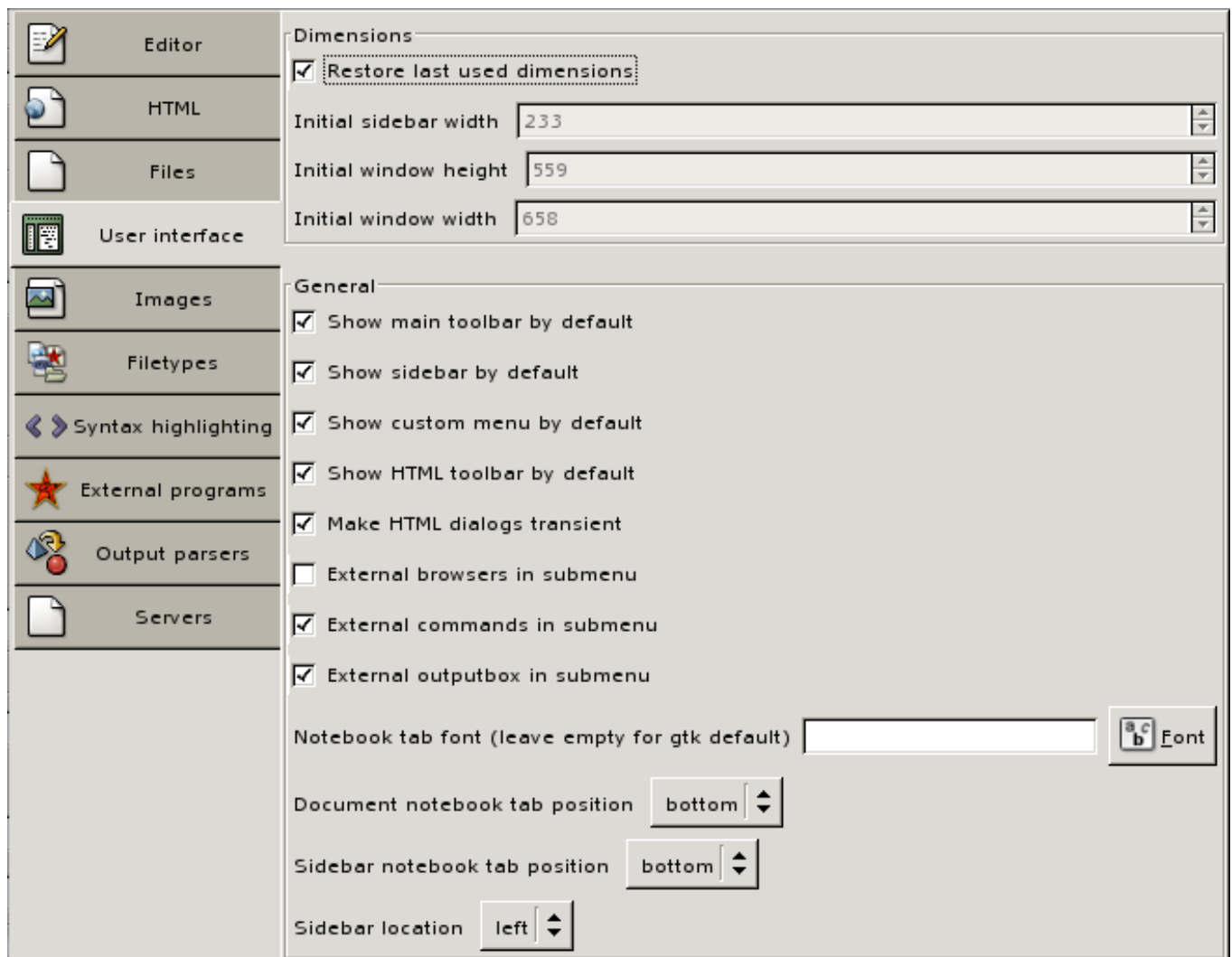


Figure VIII.11. The User Interface Preference Panel

10. Modifying file types

In the **Filetypes** tab of the **Edit preferences** panel you can define all file types that should be recognized by bluefish.

The file types consist of:

1. a label (this label is also used in the file filters, and in the highlighting patterns).
2. a list of extensions, separated by a colon (:).
3. the highlighting update characters. Upon a key press of one of these characters, the highlighting engine will refresh the highlighting around the cursor. If this field is empty, any character will force the highlighting engine to refresh. Special characters like the tab and the newline can be entered as `\t` and `\n`, the backslash itself is entered as `\\`.
4. the icon location for this file type.
5. whether this file type is editable by Bluefish (whether or not Bluefish should try to open it after a double click).
6. a regular expression that can be used to detect the file type if a file without extension is loaded.
7. the auto-tag-closing mode. A value of 0 means that Bluefish should not close XML/HTML tags, a value of 1 means it should close the tags XML style (`
`), a value of 2 means HTML style.

You add, modify, delete, or move file types the same way it is described in [Section 8.1, “Customizing browsers” \[75\]](#).

Example VIII.1. Adding a file type

Let's say you use DocBook xsl stylesheets. Those files are recognized by bluefish as xml files, but they do not appear with the xml icon in the file browser as their extension (.xsl) is not listed in the **Extensions** field of the **Filetypes** tab of the **Edit preferences** panel.

On the other hand, adding them to the xml file type would impede to group them into a stylesheet filter, where they belong from a semantical point of view. And you cannot add them to the provided stylesheet filetype made for css stylesheet, since

the highlighting patterns are different.

To add an xsl stylesheet file type, execute the following steps:

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **Filetypes** tab to display the **Filetypes** panel.
3. Click on the Add button in the **Filetypes** part. A new line will be shown, with an **Untitled** label.
4. Double-click on the label to allow editing, and enter the string you want to appear in the **Label** field. Here enter **xsl stylesheet**.
5. Click in the **Extensions** zone and enter the extension: **.xsl**.
6. Click in the **Update chars** field of the **xml** filetype line to copy and paste this field into the corresponding field of the **xsl stylesheet** filetype line. Once the field is highlighted, use **Ctrl-C** to copy the field. Click again in the **Update chars** field of the **xsl stylesheet** filetype line and use **Ctrl-V** to paste the field.
7. For the icon field, you can either use the xml icon path used in the **Icon** field of the **xml** filetype line or better create a new icon based on the xml one by changing its colors with the Colormap Rotation filter of gimp, located under the **FiltersColorsMap...** menu.

To do it, first copy the xml icon on your Desktop, apply the filter on it, and save it under **bluefish_icon_xsl.png** in a dedicated folder in your home directory, for example **~/Pictures** for Mac users.

Whichever icon you decided to use, click on the **Icon** field to enter its path.

8. Check the **Editable** box, if it is not already checked.
9. Copy and paste **Content regex** field of the **xml** filetype line into the corresponding field of the **xsl stylesheet** filetype line.
10. Set the **Auto close tags mode** to 1.
11. Click on the OK button to save and close the panel.



If you want to enter more than one extension in the **Extensions** field, you should separate them with a colon.

When you define a new filetype, you should also provide new highlighting patterns.

11. Modifying the files filters

The files filters allow you to group files types from the usage point of view. Once a file filter is created, you can view, hide, or open files based on a filter in the **File browser** contextual menu.

The file filters consist of:

1. a label.
2. whether or not the filter as defined in the **Filetypes in filter** hides the retrieved files or shows them.
3. a list of filetypes, as defined in the **Filetypes** part, separated by a colon.

You add, modify, delete, or move file types the same way it is described in [Section 8.1, “Customizing browsers” \[75\]](#).

Example VIII.2. Adding a file filter

Following with our example in [Section 10, “Modifying file types” \[87\]](#), we can add a stylesheet filter to group css and xsl stylesheets together.

To add a stylesheet filter, execute the following steps:

1. Click on the **Preferences...** icon in the main tool bar to access the **Edit preferences** panel.
2. Click on the **Filetypes** tab to display the **Filetypes** panel.
3. Click on the Add button in the **Filefilters** part at the bottom. A new line will be shown, with an **Untitled** label.
4. Double-click on the label to allow editing, and enter the string you want to appear in the **Label** field. Here enter **All stylesheets**.
5. Check the **Inverse filter** box.
6. Click in the **Filetypes in filter** field and enter the filetypes you want to group together, separated with a colon. Here it is **stylesheet:xsl stylesheet**.
7. Click on the OK button to save and close the panel.



The file types used in the **Filetypes in filter** match those defined in the **Filetypes** part. Do not confuse them with the file extensions. For example the **C programming** file filter matches **c** and **image** filetypes, i.e. files whose extensions are **.c**, **.h**, etc...

12. Modifying the highlighting patterns

The highlight patterns are build from Perl compatible regular expressions. A pattern has options for coloring and styling the text it matches. Within a match other patterns can be used to color parts of that match. There are three types of patterns:

1. Start pattern and end pattern: that is two distinct patterns, match from the start pattern to the end pattern
2. Only start pattern: that is a unique pattern that matches from start to end
3. Subpattern from parent: that is a subpattern from the parent pattern, specified by the range in the parent pattern.

One specific pattern can also be used within several other parent patterns. The parent-match option is a regular expression that defines all parents for a certain pattern. If empty it will default to `^top$`, so basically it will be on the top level.

So how does it work? Lets take a look at a little example text, a piece of PHP code within some HTML code:

```
<p align="center">
<?php
// this is a comment ?>
?>
```

The first thing the highlighting engine does is finding the pattern that has the lowest match. Using the default patterns for PHP, the pattern named **HTML**:

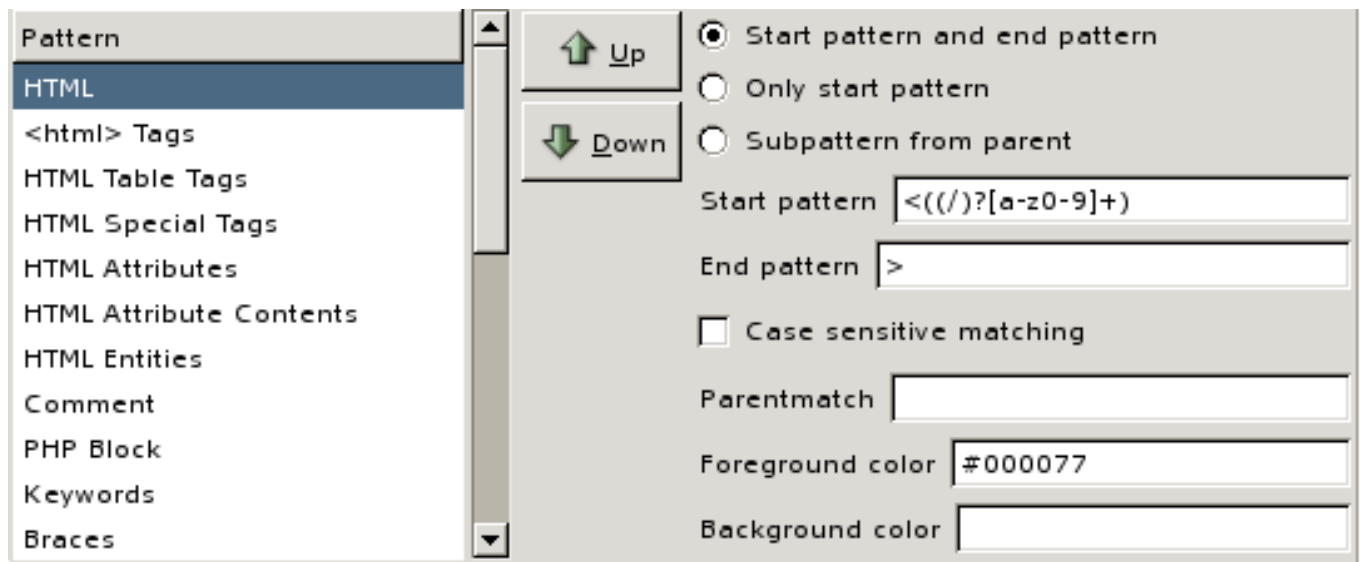


Figure VIII.12. The HTML Pattern

has a match at position 0:

```
<p align="center">
```

So now the highlighting engine searches for the lowest match in all subpatterns of **HTML**, in the region matched by a type 2 pattern. Again, the lowest match will count. The pattern named **<html> Tags**:

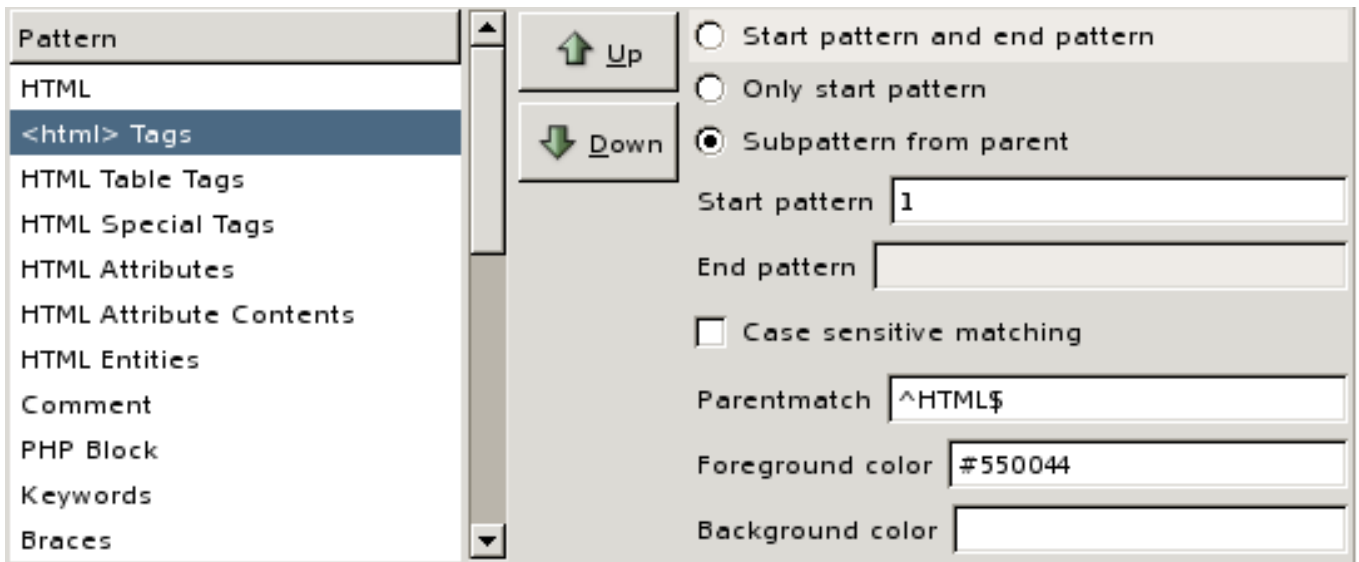


Figure VIII.13. The <html> Tags Pattern

has a match at position 1. This pattern is a type 3 pattern, so it matches a subpattern of the parent:

P

The match from subpattern <html> Tags ends at position 2 and it does not have any child patterns, so the highlighting engine continues at position 2 with all subpatterns from HTML. A type 2 pattern named HTML Attributes:

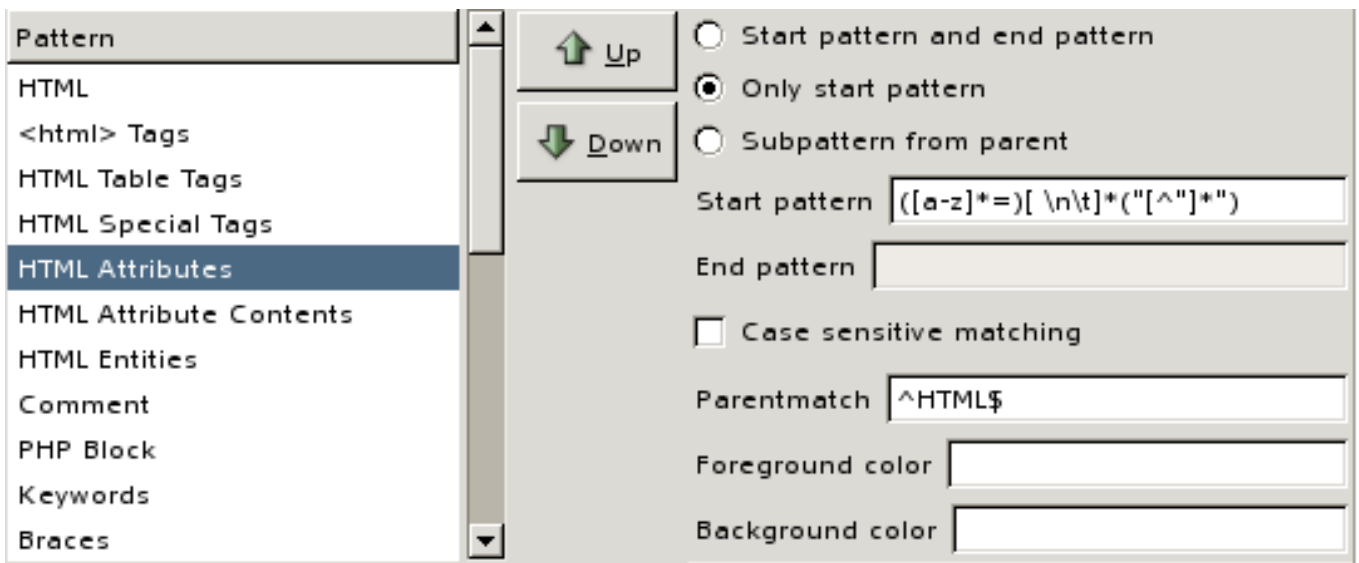


Figure VIII.14. The HTML Attributes Pattern

has the lowest match:

align="center"

This pattern does have a child pattern, again a type 3 pattern called HTML Attribute Contents:

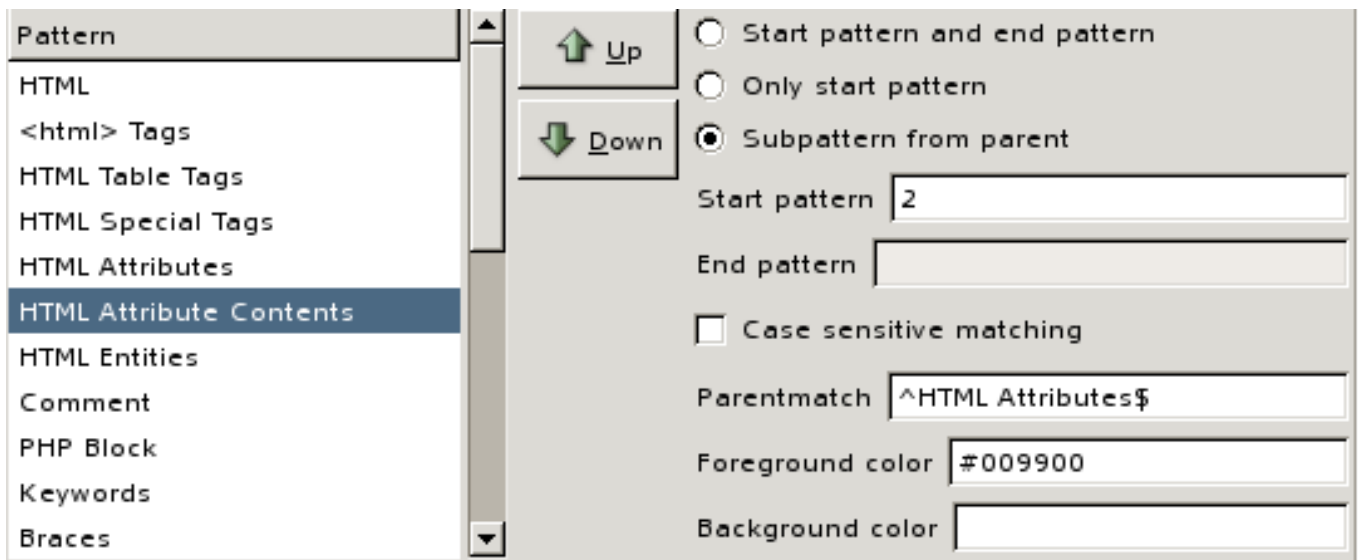


Figure VIII.15. The HTML Attribute Contents Pattern

matching:

"center"

The pattern **HTML Attribute Contents** does not have any child patterns, and subpatterns of **HTML Attributes** do not have any more matches, and also **HTML** subpatterns do not have any more matches. So we are back on the main level, the remaining code to highlight is:

```
<?php
// this is a comment ?>
?>
```

Now a pattern named **PHP Block**:

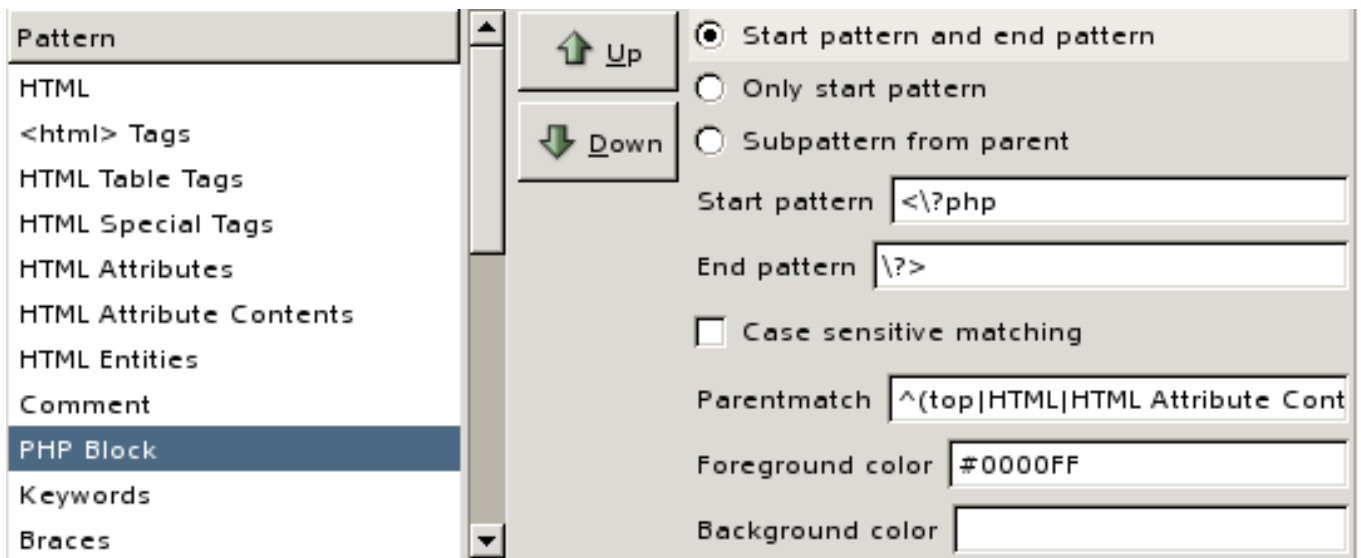


Figure VIII.16. The PHP Block Pattern

has the lowest match. This is a type 1 pattern, so the highlight engine continues with all the remaining code, but it will not only search for the lowest match of the child patterns of **PHP Block**, but it will also use it for the end pattern of **PHP Block**. The lowest match in this example is a pattern named **Comment (C++/single line)**:

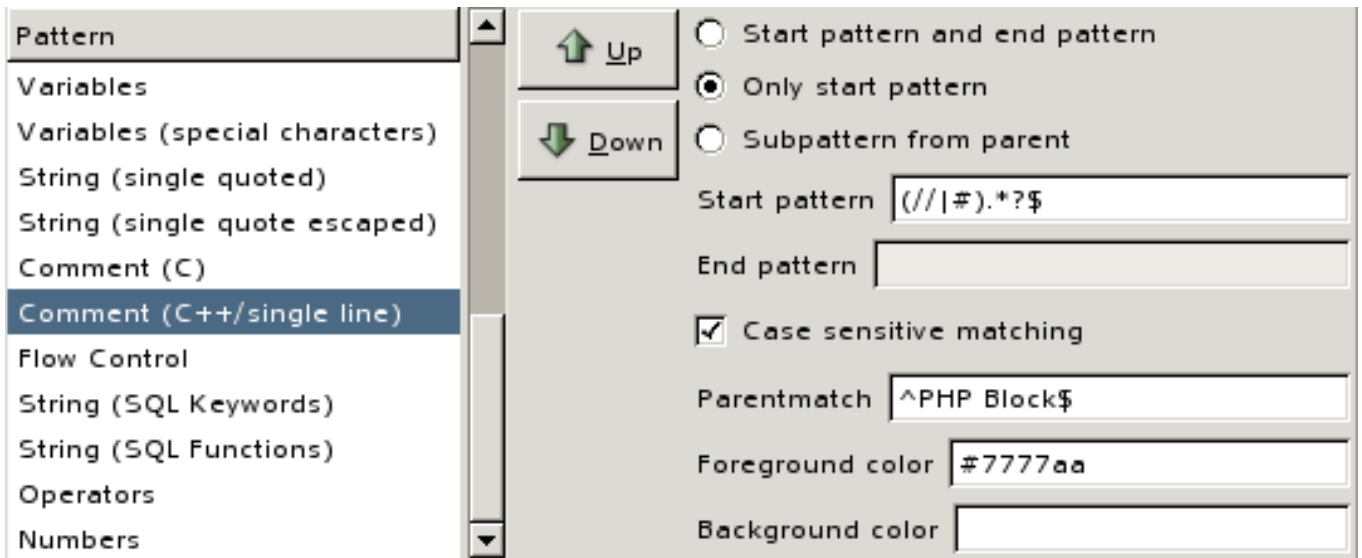


Figure VIII.17. The Comment (C++/single line) Pattern

As you can see the `?>` within the comment does not end the php pattern, because it lies within a subpattern of **PHP Block**:

```
// this is a comment ?>
```

The pattern **Comment (C++/single line)** does not have any child patterns, so the remaining code for the **PHP** subpatterns is:

```
?>
```

It is very obvious now, the lowest match will be the end pattern of the **php** pattern, so we're back on the main level, and we have matched all of the code!

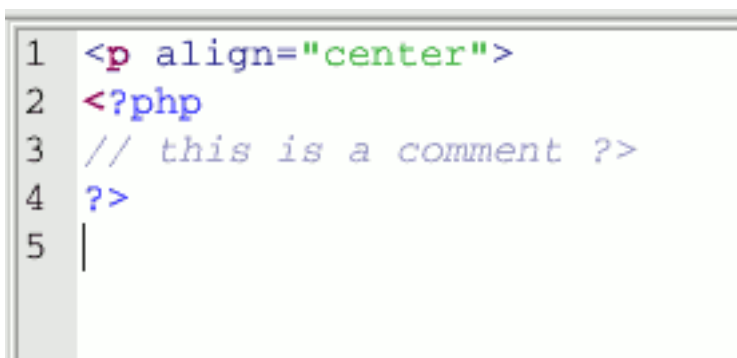


Figure VIII.18. Syntax Highlighting Example

The config file for highlighting is a colon separated array with the following content:

```
mode:
patternname:
case_sensitive(0-on/1-off):
start reg-ex:
end reg-ex:
start & end pattern(1), only start(2), subpattern(3):
parent-match:
foreground-color:
background-color:
don't change weight(0), non-bold(1), bold(2):
don't change style(0), non-italic(1), italic(2):
```

The same options are found in the syntax highlighting preferences.

As an exercise you may want to add the highlighting patterns for the xsl stylesheet file type created previously. They will be based on the xml patterns with just small changes.



If you check the force bold weight check box, you should also check that the font you use has a bold variant in the **Editor** tab of the **Preferences** panel.

Chapter IX. Shortcuts available within Bluefish

Here we will try to reference all the shortcuts available within Bluefish to perform a given action.

The here-below tables list the editor window accelerator keys (EWA keys), the contextual menus keys (CM keys), and the toolbars' icons. Some shortcuts appear in several tables to make easy the search for a given action.

You may always change the editor window accelerator keys, as explained in the [Section 1, “Modifying shortcut keys” \[81\]](#).

1. General Shortcuts

Here are the shortcuts that apply to all types of documents.



For the Filebrowser contextual (FileB. in the table), we distinguish the directory part with D and the files part with F. When no letter appears after the name, the shortcut applies to both parts.



The contextual menus shortcuts listed below are only valid for English language.

Table IX.1. Shortcuts for all Document Types

Action		EWA		CM	
		Keys	Menus	Keys	Loc.
Create files	New	Ctrl-N	File	F	FileB.
	New Window	Shift-Ctrl-N	File		
	New Directory			N	FileB.
Open a file	Open...	Ctrl-O	File	O	FileB. F
	Open Advanced...	Shift-Ctrl-O	File	A	FileB. D
Save files	Save	Ctrl-S	File		
	Save as...	Shift-Ctrl-S	File		
	Rename...	F2	File	m	FileB. F
Delete Files	Delete			D	FileB. F
Close files	Close	Ctrl-W	File	x ^a	Notebook
	Close All	Shift-Ctrl-W	File		
Quit		Ctrl-Q	File		
Undo/Redo	Undo	Ctrl-Z	Edit		
	Redo	Shift-Ctrl-Z	Edit		
Copy/Cut/Paste	Cut	Ctrl-X	Edit	t	Document
	Copy	Ctrl-C	Edit	C	Document
	Paste	Ctrl-V	Edit	P	Document
Select All				A	Document
Find/Replace	Find...	Ctrl-F	Edit		
	Find Again	Ctrl-G	Edit		
	Replace...	Ctrl-H	Edit		
	Replace Again	Shift-Ctrl-H	Edit		







Action		EWA		CM	
		Keys	Menus	Keys	Loc.
Shift	Shift Right	Shift-.	Edit		
	Shift Left	Shift-,	Edit		
Bookmark	Add Bookmark	Ctrl-D	Edit		
View Sidebar		F9	View		
Special signs	Less-than sign	Ctrl-Alt-,	Tags → Special → Math-Science		
	Greater-than sign	Ctrl-Alt-.	Tags → Special → Math-Science		
Layout	Non-Breaking Space	Ctrl-Alt-N	Tags → Format general		
Comment		Ctrl-Alt-C	Tags		
Autoclose Tags		Ctrl-T	Document		
Tag editing		F3	Dialogs		
Update Highlighting		F5	Document		
Documents navigation	Previous document	Ctrl-Page Up	Go		
	Next document	Ctrl-Page Down	Go		
	First document	Shift-Ctrl-Page Up	Go		
	Last document	Shift-Ctrl-Page Down	Go		
	Goto Line	Ctrl-L	Go		



















^aLeft click on the button.

2. HTML Shortcuts




Here are the shortcuts that apply to documents containing HTML markup.








Table IX.2. Shortcuts for HTML Markup

Action		EWA		Tool Bars	
		Keys	Menus	Icon	Bar
Headings	Quickstart...	Shift-Alt-Q	Dialogs → General		Std
	Meta...	Shift-Alt-M	Dialogs → General		
	Body...	Shift-Alt-B	Dialogs → General		Std
Header tags	H1	Ctrl-Alt-1	Tags → Headings		Std ^a
	H2	Ctrl-Alt-2	Tags → Headings		Std ^a
	H3	Ctrl-Alt-3	Tags → Headings		Std ^a
	H4	Ctrl-Alt-4	Tags → Headings		Std ^a

Action		EWA		Tool Bars	
		Keys	Menus	Ic.	Bar
	H5	Ctrl-Alt-5	Tags → Headings		Std ^a
	H6	Ctrl-Alt-6	Tags → Headings		Std ^a
Layout	Bold	Ctrl-Alt-B	Tags → Format by layout		Std
	Italic	Ctrl-Alt-I	Tags → Format by layout		Std
	Underline	Ctrl-Alt-U	Tags → Format by layout		
	Strikeout	Ctrl-Alt-S	Tags → Format by layout		
	Strong	Ctrl-Alt-G	Tags → Format by context		Font
	Emphasis	Ctrl-Alt-E	Tags → Format by context		Font
	Subscript				Font
	Superscript				Font
	Context Formatting				Font
	Paragraph	Ctrl-Alt-P	Tags → Format general		Std
	Break	Ctrl-Alt-K	Tags → Format general		Std
	Break and Clear				Std
	Non-Breaking Space	Ctrl-Alt-N	Tags → Format general		Std
	Preformatted Text	Ctrl-Alt-F	Tags → Format general		Font
	Rule...	Shift-Alt-R	Dialogs → General		Std
	Center				Std
	Align right	Ctrl-Alt-R	Tags → Format general		Std
Special signs	Division sign	Ctrl-Alt-/	Tags → Special → Math-Science		
	Less-than sign	Ctrl-Alt-,	Tags → Special → Math-Science		
	Greater-than sign	Ctrl-Alt.,	Tags → Special → Math-Science		
Fonts	Font...	Shift-Alt-F	Dialogs → General		Font

Action		EWA		Tool Bars	
		Keys	Menus	Ic.	Bar
	Base Font Size...				Font
	Font Size + 1	Ctrl-Alt=	Tags → Format general		Font
	Font Size - 1	Ctrl-Alt--	Tags → Format general		Font
Table	Table Wizard...				Tbl
	Table...	Shift-Alt-T	Dialogs → Table		Tbl
	Table Row...				Tbl
	Table Header...				Tbl
	Table Data...				Tbl
	Table	Ctrl-Alt-T	Tags → Table		Tbl
	Table Row				Tbl
	Table Header				Tbl
	Table Data				Tbl
	Table Caption				Tbl
Frame	Frame Wizard...				Fr.
	Frameset...				Fr.
	Frame...				Fr.
	Frameset				Fr.
	Frame				Fr.
	Noframes				Fr.
	Target				Fr.

Action		EWA		Tool Bars	
		Keys	Menus	Ik.	Bar
Forms	Form...				Form
	Input Button...				Form
	Input Text...				Form
	Input Hidden...				Form
	Textarea...				Form
	Input Radio Button...				Form
	Input Check Box...				Form
	Select...				Form
	Option...				Form
	Option Group...				Form
	Button...				Form
Lists	Quicklist...	Shift-Alt-L	Dialogs → General		List
	Unordered List	Ctrl-Alt-L	Tags → List		List
	Ordered List	Ctrl-Alt-O	Tags → List		List
	List Item	Ctrl-Alt-M	Tags → List		List
	Definition List				List
	Definition Term				List
	Definition				List
Anchors	Anchor...	Shift-Alt-A	Dialogs → General		Std
Email	Email...	Shift-Alt-E	Dialogs → General		Std

Action		EWA		Tool Bars	
		Keys	Menus	Ic.	Bar
Images	Insert Image...	Shift-Alt-I	Dialogs → General		Std
	Insert Thumbnail...	Shift-Alt-N	Dialogs → General		Std
	Multi Thumbnails...				Std
Comment		Ctrl-Alt-C	Tags		Std
Stylesheets	Create Style...	Shift-Alt-S	Dialogs → CSS		CSS
	Div...	Shift-Alt-D	Dialogs → CSS		CSS
	Span...				CSS
Tags	Editing	F3	Dialogs		
	Autoclose	Ctrl-T	Document		

^aThis is a pop up icon. Corresponding H1, H2, etc... icons are also accessible from the Fonts toolbar.

Part 3. Developing Bluefish

In this part, you will find how to produce a bug report, where preferences options are defined, and guidelines for development.

[Chapter X, *Debugging Bluefish* \[101\]](#)

[Chapter XI, *Reference* \[103\]](#)

[Chapter XII, *Development guidelines* \[107\]](#)

Chapter X. Debugging Bluefish

1. Using the Debugger

Here are the detailed steps for sending a useful backtrace to the Bluefish Developer Team.

Procedure X.1. Running bluefish under gdb

1. Get the latest CVS release (see [Section 4, “Latest Developmental Version” \[4\]](#) for info)
2. From `bluefish-gtk2`, the top directory of the bluefish source, run: `autoconf`
You may have to set some environment variables before running `autoconf`, as well as providing `autoconf` with some flags. Also, some patches may need to be applied, depending on your system.
3. Then, run `./configure --with-debugging-output`
Again, you may have to add some flags, depending on your system.
4. Once you succeed in configuring bluefish, run: `make clean` in order to remove all unnecessary files.
5. Then, run `make` to compile bluefish.
Do not run `make install` since it strips the debugging symbols from the executable.
6. Execute bluefish under gdb with: `gdb src/bluefish`. This way, you will get access to a non stripped version of bluefish, which is not the case if you run `gdb bluefish` or `gdb /usr/local/bin/bluefish`, since those binaries do not have any debugging symbols anymore.
7. Once `gdb` has started, type `r` to start the debugging session.
8. Try to reproduce the crash in bluefish.
9. Copy and paste the last 50 lines of debugging output to a text file.
10. Type `bt` in gdb to get the backtrace, and copy it to the text file too. If the backtrace is huge, copy only the first 50 lines.
11. Save the text file and gzip it by running `gzip textfile` in a terminal.
12. Quit gdb with `q`
13. Fill in the bug report (see [Section 2, “Filling a bug report” \[101\]](#) for instructions).

2. Filling a bug report

First make sure you have followed the instructions at [Section 1, “Using the Debugger” \[101\]](#) in case of bluefish crash.

You can use bug-buddy or go to <http://bugzilla.gnome.org/browse.cgi?product=bluefish>, the bluefish bugzilla page on bugzilla org gnome, to fill in the bug report.

The following instructions are related to bug-buddy. Nevertheless, though the form differs when filling a bug report from the bluefish bugzilla page, the necessary bits of information are the same.

Chapter XI. Reference

Here you will find the list of the configuration files and configuration names which match the options in the Preferences panel.

Each preferences tab is given a different table, and options in a table are sorted in the same order as they appear in the tab, which may not be the same order as they appear in the corresponding configuration file.

1. Options in the Editor Tab

Table XI.1. Editor Tab Reference

Option		Configuration	
Category	Label	File	Name
Editor startup options	Show empty quickbar warning	show_quickbar_tip	\$HOME/.bluefish/rcfile_v2
Editor options	Font	editor_font_string	\$HOME/.bluefish/rcfile_v2
Editor options	Tab width	editor_tab_width	\$HOME/.bluefish/rcfile_v2
Editor options	Smart cursor positioning at line start	editor_smart_cursor	\$HOME/.bluefish/rcfile_v2
Editor options	Use spaces to indent, not tabs	editor_indent_wspaces	\$HOME/.bluefish/rcfile_v2
Editor options	Word wrap default	word_wrap	\$HOME/.bluefish/rcfile_v2
Editor options	Line numbers by default	view_line_numbers	\$HOME/.bluefish/rcfile_v2
Editor options	Highlight syntax by default	defaulthighlight	\$HOME/.bluefish/rcfile_v2
Editor options	Enable highlighting PCRE UTF-8 support	highlight_utf8	\$HOME/.bluefish/rcfile_v2
Editor options	Highlight # lines	highlight_num_lines_count	\$HOME/.bluefish/rcfile_v2
Undo	Undo history size	num_undo_levels	\$HOME/.bluefish/rcfile_v2
Undo	Clear undo history on save	clear_undo_on_save	\$HOME/.bluefish/rcfile_v2
Bookmark options	Make permanent by default	bookmarks_default_store	\$HOME/.bluefish/rcfile_v2
Bookmark options	Bookmarks filename display	bookmarks_filename_mode	\$HOME/.bluefish/rcfile_v2

2. Options in the HTML Tab

Table XI.2. HTML Tab Reference

Option		Configuration	
Category	Label	File	Name
HTML options	Lowercase HTML tags	lowercase_tags	\$HOME/.bluefish/rcfile_v2
HTML options	Use deprecated tags	allow_the_use_of_font	\$HOME/.bluefish/rcfile_v2
HTML options	Use XHTML style tags	use_xhtml	\$HOME/.bluefish/rcfile_v2
HTML options	Automatically update author meta tag	auto_update_meta_author	\$HOME/.bluefish/rcfile_v2
HTML options	Automatically update date meta tag	auto_update_meta_date	\$HOME/.bluefish/rcfile_v2

Option		Configuration	
Category	Label	File	Name
HTML options	Automatically update generator meta tag	auto_update_meta_generator	\$HOME/.bluefish/rcfile_v2

3. Options in the Files Tab

Table XI.3. HTML Tab Reference

Option		Configuration	
Category	Label	File	Name
Encoding	Default character set	newfile_default_encoding	\$HOME/.bluefish/rcfile_v2
Encoding	Auto set <meta> encoding tag on change	auto_set_encoding_meta	\$HOME/.bluefish/rcfile_v2
Backup	Create backup on save	backup_file	\$HOME/.bluefish/rcfile_v2
Backup	Backup file suffix	backupfile_string	\$HOME/.bluefish/rcfile_v2
Backup	Action on backup failure	backup_abort_action	\$HOME/.bluefish/rcfile_v2
Backup	Remove backupfile on close	backup_cleanuponclose	\$HOME/.bluefish/rcfile_v2
Misc	Allow multi instances of a file	allow_multi_instances	\$HOME/.bluefish/rcfile_v2
Misc	Open files in already running bluefish window	open_in_running_bluefish	\$HOME/.bluefish/rcfile_v2
Misc	File modified on disk check	modified_check_type	\$HOME/.bluefish/rcfile_v2
Misc	Number of files in "Open recent"	max_recent_files	\$HOME/.bluefish/rcfile_v2
File browser	Default basedir	default_basedir	\$HOME/.bluefish/rcfile_v2
File browser	Use separate file and directory view	fb_two_pane_view	\$HOME/.bluefish/rcfile_v2
File browser	Unknown icon	fb_unknown_icon	\$HOME/.bluefish/rcfile_v2
File browser	Directory icon	fb_dir_icon	\$HOME/.bluefish/rcfile_v2

4. Options in the User interface Tab

Table XI.4. User interface Tab Reference

Option		Configuration	
Category	Label	File	Name
Dimensions	Restore last used dimensions	restore_dimensions	\$HOME/.bluefish/rcfile_v2
Dimensions	Initial sidebar width	left_panel_width	\$HOME/.bluefish/rcfile_v2
Dimensions	Initial window height	main_window_height	\$HOME/.bluefish/session
Dimensions	Initial window width	main_window_width	\$HOME/.bluefish/session
General	Make HTML dialogs transient	transient_htdialogs	\$HOME/.bluefish/rcfile_v2
General	External browsers in submenu	ext_browsers_in_submenu	\$HOME/.bluefish/rcfile_v2
General	External commands in submenu	ext_commands_in_submenu	\$HOME/.bluefish/rcfile_v2
General	External outputbox in submenu	ext_outputbox_in_submenu	\$HOME/.bluefish/rcfile_v2
General	Notebook tab font	tab_font_string	\$HOME/.bluefish/rcfile_v2

Option		Configuration	
Category	Label	File	Name
		allow_the_use_of_font	
General	Document notebook tab position	document_tabposition	\$HOME/.bluefish/rcfile_v2
General	Sidebar notebook tab position	leftpanel_tabposition	\$HOME/.bluefish/rcfile_v2
General	Sidebar location	left_panel_left	\$HOME/.bluefish/rcfile_v2

5. Options in the Images Tab

Table XI.5. Images Tab Reference

Option		Configuration	
Category	Label	File	Name
Thumbnails	Thumbnails suffix	image_thumbnailstring	\$HOME/.bluefish/rcfile_v2
Thumbnails	Thumbnails filetype	image_thumbnailtype	\$HOME/.bluefish/rcfile_v2

Chapter XII. Development guidelines

Work hard but have fun!

1. Indenting and formating style

Indenting can be done with the **indent** command line tool. Bluefish uses tabs - not spaces, and I'll explain why.

Some programmers prefer a lot of indenting, 8 characters, some prefer less, 3 characters. If Bluefish code was indented with spaces, these programmers had a problem, they would have to change the files to view it in their favourite layout. But because we use tabs, these programmers can simply set the tab width to a different value, and without changing the files it looks good for both programmers!

To indent properly with **indent**, issue this command:

```
$ indent --line-length 100 --k-and-r-style --tab-size 4 \
-bbo --ignore-newlines bluefishcode.c
```

Comment all public functions like it is done in `bf_lib.c` and `gtk_easy.c` (javadoc style, with some small differences), this can be used to create a function reference.

2. Naming

For non-local functions, the name should preferably include a prefix that shows the part of bluefish it is used for. There are, furthermore, many often used abbreviations in the bluefish code, such as:

Abbreviations used in the Bluefish code

`doc`

A function for handling a specific document

`bfwin`

A function for handling a specific Bluefish window

`cb`

Callback, a function called after a button click or some other event

`lcb`

Local callback, a function called after an event, only used in this .c file

Here are such function names that show where they are from, what they handle, and/or what they do

Examples of function names in Bluefish

`bmark_set_for_doc`

Bookmark code, sets bookmarks for a document

`spell_check_cb`

Spell check code, this is a callback function (for a button)

`project_open_from_file`

Project code, opens a new project from a given file name

3. Declaring procedures

All local functions should be static!

Callback functions (called for events such as button clicks) should have prefix `_cb`, or `_lcb` for local callbacks.

For GTK callback functions, use the name of the signal in the name.

4. Header files

Only functions that are used from outside the file itself should be in the header file, in the order in which they are found in the .c file itself. Basically these are all non-static functions in the .c file.

5. New files

About new files

6. File reference

References

7. Patches

Before starting to code:

1. Update your CVS tree, or alternatively download the latest snapshot
2. Copy this original tree, so you can make a patch against this tree

Before creating the patch:

1. Run `make distclean && ./configure && make` and test if it runs successfully
2. If you have the possibility do this both with `gcc-2.95` and `gcc-3.x` as compiler

Now create the patch. Assuming that you have two directories, `original-tree` and `my-tree`:

1. Run `make distclean` in both trees
2. `cd` to the parent directory of both trees
3. Run `diff -Naur original-tree my-tree | bzip2 -9c > patchbla.diff.bz2`

8. Translations

8.1. Introduction

Bluefish has been translated into more than 15 different languages and this is only the beginning.

Translation process is not a difficult task but you will need some time because there are more than one thousand strings to be translated. The good news are you don't need to be a programmer to make Bluefish speak your language and the only tool you need is a text editor (Vim, Emacs, bluefish, etc.)

Bluefish uses po (Portable Object) files. A po file is just a plain text file that you can edit with your favorite text editor.

8.2. PO files basics

In a typical po file there are five major types of entries:

1. Those which begin with `"#:"` showing the places in the source code that contains the string being translated (there may be one or more) as: `'#: ../src/about.c:123'`
2. Those which begin with `"#,"` containing some flags (not always present) as: `'#, c-format'`
3. Those which begin with `"msgid"` containing the English string being translated (it may be spanned in several lines) as: `'msgid "Authentication is required for %s."'`
4. Those which begin with `"msgstr"` containing the translated string as: `'msgstr "Une autorisation est requise pour accéder à %s."'`
5. Those which begin with `"#~ "` containing obsolete strings as: `'#~ msgid "Save document as"'`



When an entry is tagged as fuzzy (i.e. when the line begins with `"#, fuzzy"`), that means it is probably incorrect.

You have to make sure the translation is correct and then delete either the `"#, fuzzy"` line if this is the only flag on the line, or the `", fuzzy"` part of the line if there are some other flags on the same line, like in `"#, fuzzy, c-format"`.

Remember that as long as a translation is marked "fuzzy", it will NOT actually be used!

As far as obsolete strings are concerned, it is up to you to decide if you want to remove them. On one hand they can be reused in a latter version of the po file, on the other hand they make the po file bigger.

Hence, your task as a translator is to:

1. Translate all empty `msgstr` entries
2. Check all fuzzy entries, correct them if they are wrong and remove all fuzzy tags

3. Optionally, remove obsolete strings
4. Check that the po file ends with a blank line

8.3. Shortcut keys

Shortcut keys, known as hotkeys or even accelerator keys, are defined as follows (look at the underscore, please):

```
# src/toolbars.c:482
#: ../src/filebrowser.c:1453
msgid "/_Refresh"
msgstr "/_Actualizar"
```

It means that in the English locale the user have to press **Alt-R** to activate this particular GUI element. On the other hand if your locale is Spanish your shortcut key will be **Alt-A**.



You have to keep in mind that two GUI elements must not have the same shortcut key at the same level.

8.4. How to contribute

It is really easy. Just drop me a line at <wecharri(at)arnet.com.ar> and I will send you your po file ready to be translated. When you have finished the translation work, just send it me back (use gzip or bzip2 if possible, please). Then I will check it and if everything is right I will add it into CVS.

About ten days before a new release I will send a new fresh po copy to each translator to repeat this process.

All po files will be named as follows:

```
date-foo.po.gz (date: day-month-year)
```

Example:

```
12-12-2004.es.po.gz (for Spanish po file)
```

Please, remember:

- Do not rename it (I need it for tracking stuff)
- Send it me back as soon as possible in zipped format too.
- Do not mix it without any local copy you have.
- Remember they are in UTF-8 format
- Subject in my mail will be ***New Bluefish PO File !***

And at last, do not start a new translation before contacting me or contact Olivier and do not post your po file at the list, please.

If you have some doubts, do not hesitate contact me at <wecharri(at)arnet.com.ar>.

9. Some tips

Development tips

10. Making releases

What is to do

11. Useful stuff

Links and so

Appendix A. Credits

1. Bluefish developers

Here are the developers for release 1.0.x:

- Olivier Sessink
- Jim Hayward
- Oskar Swida
- Eugene Morenko
- Alastair Porter
- Daniel Leidert
- David Marrs
- Michèle Garoche

Developers for previous releases are:

- Chris Mazuc
- Neil Millar
- Gero Takke
- Bo Forslund
- David Arno
- Pablo De Napoli
- Santiago Capel Torres
- Rasmus Toftdahl Olesen
- Roland Steinbach
- Christian Tellefsen
- Antti-Juhani Kaijanaho

2. Bluefish package maintainers

The following people maintain bluefish packages for various systems:

- Debian: Daniel Leidert (thanks to Davide Puricelli for the long Debian package maintainership)
- Redhat: Matthias Haase
- Mandrake: Todd Lyons
- Fink: Michèle Garoche

3. Bluefish translators

Translators for the 1.0 release are:

Languages	Translators
Basque	Hizkuntza Politikarako Sailburuordetza
Brazilian Portuguese	Anderson Rocha
Bulgarian	Peio Popov
Chinese	Ting Yang (Dormouse)
Chinese, Traditional	OLS3
Danish	Rasmus Toftdahl Olesen
Finnish	Juho Roukala
French	Michèle Garoche
German	Roland Steinbach

Languages	Translators
Hungarian	Péter Sáska
Italian	Stefano Canepa
Japanese	Takeshi Hamasaki
Norwegian	Christian Tellefsen
Polish	Oskar Swida
Portuguese	Lopo Pizarro
Russian	Eugene Rupakov
Serbian	Marko Milenovic
Slovak	Vladimir VASIL
Spanish	Walter Oscar Echarri
Swedish	David Smeringe
Tamil	Murugapandian Barathee
Turkish	Oguz Eren

4. Supporters to bluefish

Supporters

Appendix B. Bluefish change history

Release-Changelog for the current 1.0.x series:

Revision History

Revision 1.0.6	2006-09-26	oli4, jimh6583, dleidert, michga
<ul style="list-style-type: none">• Updated translations: Finnish, French, German, Japanese, Norwegian, Polish, Swedish.• Configure: fixes host recognition for new Mac Intel.• Custom menu: fixes a typo in docbook informal table rowsep, adds distinct tags for XML and SGML images and links.• Data filetypes: adds the .pm extension for Perl files. Closes Bluefish bug #343569 [http://bugzilla.gnome.org/show_bug.cgi?id=343569].• Highlighting: small fix to XML highlighting (adding general support for namespaces). Highlight numbers in Java.• Man-pages: fixes distclean target.• Adds XTHTM 1.1 DOCTYPE in creators. Closes Bluefish bug #338586 [http://bugzilla.gnome.org/show_bug.cgi?id=338586].• Fixes the recognition of GNOME_VFS >= 2.6.• Fixes a bug where the Bluefish icon was not displayed if the splash screen was disabled.• Fixes a bug in smart cursor positioning.• Fixes a bug where the old file was deleted before the new file was saved successfully.• Removes redundant show hidden checkbox from file chooser dialog. Closes Bluefish bug #316298 [http://bugzilla.gnome.org/show_bug.cgi?id=316298].• Fixes Definition list formatting.• Fixes a bug where icons were missing in open/save dialogs. Closes Bluefish bug #323634 [http://bugzilla.gnome.org/show_bug.cgi?id=323634].• Fixes a bug where the backup file was not deleted after renaming it. Closes Bluefish bug #324788 [http://bugzilla.gnome.org/show_bug.cgi?id=324788].• Fixes a critical bug where Bluefish crashed when closing and saving a project. Closes Bluefish bug #330544 [http://bugzilla.gnome.org/show_bug.cgi?id=330544].• Fixes a bug where backup project file was not deleted when closing a project. Closes Bluefish bug #330575 [http://bugzilla.gnome.org/show_bug.cgi?id=330575].• Fixes a bug where savedir was not initialized correctly when creating a new project on a remote server. Closes Bluefish bug #330781 [http://bugzilla.gnome.org/show_bug.cgi?id=330781].• Fixes a bug where syntax highlighting was not refreshed after executing an external command. Closes Bluefish bug #337564 [http://bugzilla.gnome.org/show_bug.cgi?id=337564].• Fixes a critical bug where Bluefish crashed when pressing the Esc key in the Search:no match found dialog. Closes Bluefish bug #337739 [http://bugzilla.gnome.org/show_bug.cgi?id=337739].• Fixes a bug where undoing did not behave correctly for text pasted from the clipboard over other text. Closes Bluefish bug #340262 [http://bugzilla.gnome.org/show_bug.cgi?id=340262].• Fixes a critical bug where undoing after inserting a file into an unsaved file lead to loss of data. Closes Bluefish bug #344698 [http://bugzilla.gnome.org/show_bug.cgi?id=344698].• Fixes a bug in python syntax highlighting. Closes Bluefish bug #349662 [http://bugzilla.gnome.org/show_bug.cgi?id=349662].• Fixes a critical bug where Bluefish crashed when pressing the green info dialog button. Closes Bluefish bug #349743 [http://bugzilla.gnome.org/show_bug.cgi?id=349743] and Ubuntu bug #55256 [https://launchpad.net/distros/ubuntu/+source/bluefish/+bug/55256].• Fixes a critical bug where Bluefish used a high CPU load when searching for empty lines with Perl regexp syntax. Closes Bluefish bug #350721 [http://bugzilla.gnome.org/show_bug.cgi?id=350721].• Fixes a critical bug where Bluefish crashed when performing search with bookmark on unsaved document. Closes Bluefish bug #350754 [http://bugzilla.gnome.org/show_bug.cgi?id=350754].• Fixes a bug in syntax highlighting where escaped quotes were not highlighted correctly. Closes Bluefish Bug #352701 [http://bugzilla.gnome.org/show_bug.cgi?id=352701].• Fixes a critical bug in the replace dialog which caused data corruption when replacing text in "all open files" with files that contained UTF-8 characters. Closes Bluefish Bug #352762 [http://bugzilla.gnome.org/show_bug.cgi?id=352762].• Fixes a bug where clicking in folder browser did not draw new selection bar until a file from that folder was loaded. Closes Ubuntu bug #45927 [https://launchpad.net/distros/ubuntu/+source/bluefish/+bug/45927].		

Revision 1.0.5	2006-02-04	oli4, jimh6583, dleidert
<ul style="list-style-type: none"> New translations: Basque, Slovak. Updated translations: Chinese, French, German, Italian, Japanese, Polish, Portuguese, Spanish, Swedish. Fixes a critical feature bug in the Undo/Redo function, which can lead to data loss. Partially closes Bluefish bug #327432 [http://bugzilla.gnome.org/show_bug.cgi?id=327432] and Gentoo bug #119246 [https://bugs.gentoo.org/show_bug.cgi?id=119246]. Man-pages are now installed by default in every distribution not only Debian. Closes Bluefish bug #309404 [http://bugzilla.gnome.org/show_bug.cgi?id=309404]. Improves behaviour of font drop down list. Closes Bluefish bug #317974 [http://bugzilla.gnome.org/show_bug.cgi?id=317974]. Improves the “Link to stylesheet” dialog. Closes Bluefish bug #319753 [http://bugzilla.gnome.org/show_bug.cgi?id=319753]. Makes the splash screen a runtime user option. Closes Bluefish bug #324647 [http://bugzilla.gnome.org/show_bug.cgi?id=324647]. Fixes the untitled windows issue, when starting bluefish several times using the -n option. Closes Bluefish bug #317878 [http://bugzilla.gnome.org/show_bug.cgi?id=317878] and Ubuntu bug #1426 [https://launchpad.net/distros/ubuntu/+source/bluefish/+bug/1426]. Fixes and updates to C, ChangeLog, CSS, SQL, Perl and PHP highlighting patterns. Closes Bluefish bug #313000 [http://bugzilla.gnome.org/show_bug.cgi?id=313000], #315324 [http://bugzilla.gnome.org/show_bug.cgi?id=315324], #317769 [http://bugzilla.gnome.org/show_bug.cgi?id=317769], #326887 [http://bugzilla.gnome.org/show_bug.cgi?id=326887] and #329182 [http://bugzilla.gnome.org/show_bug.cgi?id=329182]. Fixes an issue where the “Highlight Syntax” menu entry could show a wrong state. Closes Bluefish bug #324517 [http://bugzilla.gnome.org/show_bug.cgi?id=324517]. Fixes a crash in the spellcheck dialog. Closes Bluefish bug #322126 [http://bugzilla.gnome.org/show_bug.cgi?id=322126] and Debian bug #340768 [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=340768]. Fixes for the quickstart dialog, especially a crash observed with GTK >= 2.8. Closes Bluefish bug #324836 [http://bugzilla.gnome.org/show_bug.cgi?id=324836] and Debian bug #344435 [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=344435]. Fixes for the find dialog. So choosing “Bookmark results” does not longer lead to a search in the whole document even if “Search selection” is used. Closes Bluefish bug #327099 [http://bugzilla.gnome.org/show_bug.cgi?id=327099]. Fixes the moving replace dialog. Closes Bluefish bug #323183 [http://bugzilla.gnome.org/show_bug.cgi?id=323183]. Fixes and improvements for the PCRE UTF-8 support. Closes Bluefish bug #316866 [http://bugzilla.gnome.org/show_bug.cgi?id=316866]. Fixes an inconsistency in the analysis of LC_MESSAGES and LANG leading to menus in different languages. Closes Bluefish bug #323408 [http://bugzilla.gnome.org/show_bug.cgi?id=323408]. Fixes and language additions for GNOME 2.12 (and above) integration and the menu tooltip. Closes Ubuntu bug #1166 [https://launchpad.net/distros/ubuntu/+source/bluefish/+bug/1166]. Fixes for the <link>, <meta> and <http-equiv> dialogs. Fixes for the custom menu. Closes Bluefish bug #326597 [http://bugzilla.gnome.org/show_bug.cgi?id=326597]. We do not longer ship the Debian packaging files with the source. 		
Revision 1.0.4	2005-08-28	oli4, jimh6583, dleidert
<ul style="list-style-type: none"> Translation updates: Spanish, Japanese. Highlighting fixes to HTML, PHP highlighting patterns. Fix a compile issue when compiling without GnomeVFS support. Closes Bluefish bug #314174 [http://bugzilla.gnome.org/show_bug.cgi?id=314174]. Update bluefish.desktop to current standards. Fix a bug in the highlighting engine that incorrectly matched multi byte characters. 		
Revision 1.0.3	2005-08-16	oli4, jimh6583, dleidert
<ul style="list-style-type: none"> Translation updates: Danish, German, Italian, Japanese, Chinese. Patch from Eduard Carreras i Nadal: Single-line selected text becomes the search-string in the Find/Replace dialogs. Add an option to disable the Quickbar Tip dialog. Closes Bluefish bug #309401 [http://bugzilla.gnome.org/show_bug.cgi?id=309401] and Debian bug #249639 [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=249639]. After inserting a text block from the custom menu, Bluefish should scroll the editor window to make cursor visible. Closes Bluefish bug #310531 [http://bugzilla.gnome.org/show_bug.cgi?id=310531]. Formatstring before/after in custom menu editor does not follow cursor. Closes Bluefish bug #310534 [http://bugzilla.gnome.org/show_bug.cgi?id=310534]. 		

- Highlighting fixes to C, Python highlighting patterns. Closes [Bluefish bug #312537](http://bugzilla.gnome.org/show_bug.cgi?id=312537) [http://bugzilla.gnome.org/show_bug.cgi?id=312537].
- Fix a bug where the user was not prompted to overwrite an existing file on a remote file system.
- Misc custom menu fixes.
- GUI improvements to the custom menu dialogs.
- Extend e-mail dialog: Add CC and BCC fields. Subject and body are now automatically URL-encoded.
- Extend time insert dialog: Add possibility to insert ISO8601 date/time as recommended by the W3C for the date meta tag. Rewrite a part of the auto-update of meta tags. User can now choose, which meta tags should be updated. Add possibility to auto-update the date meta tag. Partially fixes [Bluefish bug #309402](http://bugzilla.gnome.org/show_bug.cgi?id=309402) [http://bugzilla.gnome.org/show_bug.cgi?id=309402]. Closes [Debian bug #185158](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=185158) [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=185158].
- Clean up translatable strings. Fix a few menu entries.
- Set an empty title in message dialogs as described in the GNOME HIG.
- Add an option to the QuickStart dialog that allows the user to specify whether a new document should be opened.
- Misc small bug fixes and improvements (refer to ChangeLog).

Revision 1.0.2	2005-07-03	oli4, jimh6583, dleidert
----------------	------------	--------------------------

- Added ability to remove multiple meta tags in Quickstart dialog.
- Updated Debian packaging files (standards version and build-dependencies).
- Updated config scripts.
- Updated for traditional Chinese translation.
- Fixed Ctrl-Z/undo bug.
- Fixed outdated configure script and aspell detection.
- Fixed format specification bug in Bulgarian translation. Closes [Debian bug #308849](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=308849) [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=308849].
- Fixed translation bugs and typos in German translation. Closes [Debian bug #313674](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=313674) [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=313674].

Revision 1.0.1	2005-06-13	oli4, jimh6583, dleidert
----------------	------------	--------------------------

- New QuickStart dialog.
- Spaces not being used to instead of tabs.
- Updated Debian packaging files (build-depends).
- Updated and improved spell checking.
- Updated and improved highlighting patterns.
- Fixed custom menu.
- Fixed bookmarks handling.
- Fixed missing CSS dialog properties.
- Fixed improperly opened color dialog when opening the CSS dialog.
- Fixed project dialog segmentation fault when clicking the “Browse” button. Closes [Debian bug #295536](http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=295536) [http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=295536].
- Fixed not retained “View” menu item settings after closing/restarting.
- Fixed several build issues (debugging output, configure, ...)

Appendix C. Guidelines for Writing this Manual

1. Introduction to DocBook

The Bluefish manual is written in [DocBook XML](http://www.docbook.org) [http://www.docbook.org], which is a set of standards for writing documentation. Originally, DocBook was intended for computer software documentation, but is now used for many other document types.

2. Manual building requirements

To generate HTML, PDF or PostScript files out of the source XML, you will need the following:

- the Bluefish manual source files via CVS
- DocBook 4.4
- DocBook XSL style sheets 1.71.0
- XSLT Processors and Parsers: we use **xsltproc** for HTML production, and **fop** for PDF and PostScript production.
- **xmllint** for validating all files

Here are the procedures to install the required files:

Procedure C.1. Getting the Bluefish manual source files

1. First, think about where in the filesystem you would like to build the manual. Let us assume you choose your home directory. So, from your home directory, login to Bluefish's CVS repository by issuing the command::

```
$ cvs -d :pserver:anonymous@bluefish.cvs.sourceforge.net:/cvsroot/bluefish \
login
```

2. Hit enter at the prompt when asked for your password.
3. Checkout the directory containing the Bluefish documentation:

```
$ cvs -z3 -d :pserver:anonymous@bluefish.cvs.sourceforge.net:/cvsroot/bluefish \
co bluefish-doc
```

This will download the bluefish-doc CVS module, containing the Bluefish manual source files, to your system in a newly created directory bluefish-doc.

Procedure C.2. Installing DocBook and DocBook XSL

1. Install DocBook 4.4 for your distribution
2. Install DocBook XSL version 1.71.0 if it is available for your distribution

Otherwise, get it from the [DocBook Project pages](http://sourceforge.net/projects/docbook/) [http://sourceforge.net/projects/docbook/] on SourceForge.net and unpack it into the bluefish-doc directory.

Procedure C.3. Installing the xslt processors and parsers

1. Install libxslt if needed.

xsltproc is provided by the [libxslt](http://xmlsoft.org/XSLT/) [http://xmlsoft.org/XSLT/], distributed as part of the GNOME desktop environment and is packaged for most Linux distributions. [Fink](http://fink.sourceforge.net/) [http://fink.sourceforge.net/] provides the package for Mac OS X.

2. If you wish to build PDF or PostScript versions of the manual, you will need the Formatting Objects Processor (FOP) package from Apache. If it is not already installed on your system, get the latest binary distribution from the [FOP download](http://xml.apache.org/fop/download.html) [http://xml.apache.org/fop/download.html] page on the Apache web site.
3. Unpack it into the bluefish-doc directory:

```
$ tar zxvf fop-0.20.5-bin.tar.gz -C ~/bluefish-doc
```

The files will be unpacked to a directory called fop-0.20.5.

4. Rename fop-0.20.5 to fop:

```
$ mv ~/bluefish-doc/fop-0.20.5 ~/bluefish-doc/fop
```

5. FOP does not yet support embedding PNG images in pdf files. To get PNG support, we need from [Java Advanced Imageing \(JAI\)](http://java.sun.com/products/java-media/jai/downloads/download-1_1_2.html) [http://java.sun.com/products/java-media/jai/downloads/download-1_1_2.html] or the [JIMI Software Development Kit](http://java.sun.com/products/jimi/) [http://java.sun.com/products/jimi/] (especially for Debian and Ubuntu users, read /usr/share/doc/fop/README.Debian).

For Linux, download the CLASSPATH version, jai-1_1_2-lib-linux-i586.tar.gz. Unpack and copy the files jai_core.jar and jai_codec.jar files to ~/bluefish-doc/fop/lib. Debian and Ubuntu users running fop with JIMI-support download jimi1_0.(zip|tar.Z) and place jimi-1.0.jar into /usr/share/java.



JAI and JIMI support are available for FOP release 0.20.5 and later.

3. Make HTML/PDF/PostScript Versions of the Bluefish manual

Use the following options to make and remove the various files, which will be output to a newly created bluefish-doc/built-doc directory, except for the tarballs which will be output in another newly created bluefish-doc/tarballs directory:

Options to make the Bluefish manual

make html or make html-one-big or make html-all

Produces the book in html format, with numerous html files or with a unique html file or with both.

make pdf-a4 or make pdf-us

Produces the book in PDF format suitable for A4 or USLetter paper.

make ps-a4 or make ps-us

Produces the book in PostScript format suitable for A4 or USLetter paper.

make pspdf-a4 or make pspdf-us

Produces the PDF and PostScript versions for A4 or USLetter paper.

make all

Produces all formats of the book.

clean html

Removes the book in html format (both outputs).

clean pdf-a4 or clean pdf-us

Removes the book in PDF format suitable for A4 or USLetter paper.

clean ps-a4 or clean ps-us

Removes the book in PostScript format suitable for A4 or USLetter paper.

clean pdfps-a4 or clean pspdf-us

Removes the PDF and PostScript versions for A4 or USLetter paper.

clean all

Removes all versions of the book.

make validate-all

Validate the whole book.

make tarball-all

Produces tarballs suitable to put on the Bluefish web site or for individual use.

If you want to create a new format, you will have to make a copy of the titlepage-a4.xml and pdf-ps-a4.xsl files in the bluefish-doc/stylesheets directory, rename them according to the new format, and modify them.

4. Conventions for Writing this Manual

We recommend writing the manual with bluefish. It has most of the tags used in the manual in the DocBook custom menu. By unchecking the **Use spaces to indent, not tabs** and checking **Word wrap default** in the Preferences **Editor** panel, you will ensure that no unnecessary white space will be produced when processing the files.

The DocBook rules are strict and must be maintained in order for the manual to build using xsltproc. Thus, you should always validate the whole book before sending or committing any change. To do it, just issue:

```
$ make validate-all
```

in the bluefish-doc/src directory.

However, there are some rules we like to follow to make editing the manual more efficient and organized.

4.1. The id Attribute

We use `id` on chapter, appendix, section, figure, and procedure. This provides a convenient way to reference them in the text as well and to get them listed in the table of contents.

If you need to reference some chunk of text embedded in a tag different from those already mentioned, you can also use an `id` on this tag, since DocBook allows `id` on all tags.

Separate words in the `id` with hyphens.

Finally, include a word or two describing the content in the section. For example, a chapter entitled `Using Bluefish` would have the `id` `bluefish-using`. And, a section within that chapter called `Keyboard Shortcuts` could have the name `bluefish-using-shortcuts`.

The main thing is that all `id`'s must be unique or processing will fail. To ensure that all `id`'s are unique, just run **make validate-all** before committing the changes.

Also, be careful when renaming `id`'s, since the name could be used in links within other parts of the manual. It is best to do a global search for an `id` in all the manual's files before changing an `id`.

4.2. Using Screen shots

All screen shots are `png` files. They should be placed in the `bluefish-doc/src/figures` directory. They are inserted in the `xml` files with the following tags:

```
<para>
<figure id="figure-file-menu">
<title id="figure-file-menu-title">Bluefish File Menu</title>
<screenshot>
<mediaobject>
<imageobject>
<imagedata fileref="figures/file_menu.png" format="PNG"/>
</imageobject>
<textobject>
<phrase>A screen shot of the Bluefish File Menu</phrase>
</textobject>
</mediaobject>
</screenshot>
</figure>
</para>
```

Notice that the `figure id`, the `title id`, and the `imagedata fileref` are very similar. The former ones use hyphens, while the later uses underscore to separate the `id` parts. They have in common the significant part. Do not forget to put in the `phrase` tag, a sentence meaningful for blind people.

4.3. Referencing Bluefish interface elements

We use the following DocBook GUI tags:

Interface elements

Isolated shortcut

```
<keycombo>
<keycap>Ctrl</keycap>
<keycap>S</keycap>
</keycombo>
```

Isolated menu

```
<guimenu>File</guimenu>
```

Menu with submenuitem

```
<menuchoice>
<guimenu>File</guimenu>
<guimenuitem>Open...</guimenuitem>
</menuchoice>
```

Menu with submenu

```
<menuchoice>
<guimenu>Edit</guimenu>
<guisubmenu>Replace special</guisubmenu>
</menuchoice>
```

Menu with submenuitem and shortcut

```
<menuchoice>
<shortcut>
<keycombo>
<keycap>Ctrl</keycap>
<keycap>O</keycap>
</keycombo></shortcut>
<guimenu>File</guimenu>
<guimenuitem>Open...</guimenuitem>
</menuchoice>
```

Label

```
<guilabel>Use spaces to indent, not tabs</guilabel>
```

4.4. Using procedures

When you want to explain some process, use procedures; this way, the user benefits of a clear step by step guidance:

```
<procedure id="installing-docbook-xsl">
<title>Installing docbook-xsl</title>
<step>
<para>Install them for your distribution</para>
</step>
<step>
<para>Put a copy ... <filename>bluefish-doctools/tools</filename></para>
</step>
</procedure>
```

If the explanation consists mainly in orders, you may want to use ordered list instead.

Keep in mind that steps and ordered lists are automatically numbered, thus words such as first, then, etc. are useless.

4.5. Using notes, tips, warnings

Be aware that DocBook is picky about their usage inside variable list: you can put them either just after the title or inside a list item. They are used as follows:

```
<warning>
<para>You have to keep in mind...</para>
</warning>
```

4.6. Using links

To reference an external link, we use:

```
<ulink url="http://www.sourceforge.net"/>
```

Or:

```
<ulink url="http://xmlsoft.org/XSLT/">libxslt</ulink>
```



Note that the first form produces in html and ps/pdf formats a clickable link with the url as text, while the second form produces in html format a clickable link with the url as underlying url and the text as viewable text, and in ps/pdf formats a clickable link with the text as text and add the url into brackets after the text, so that the url is not lost when printing on paper. Hence it is important to use the second form when the text does not match the url.

To reference an internal link (i.e. internal to the book), we use:

```
<xref linkend="getting-bluefish-updates" />
```

This generates a linked text similar to: Section x.y.z “Getting Bluefish updates”. This is the preferred form, but it may not be always suitable; in this case, you can use:

```
<link linkend="getting-bluefish-updates">here</link>
```



Note that you may have to prepend the appropriated article if needed.

To reference a chapter by number, we use:

```
Chapter <xref linkend="getting-bluefish" xrefstyle="template:%n" />
```

4.7. Others tags

To highlight command line tools or small applications, we use:

```
<command>make</command>
```

To emphasize file names or directories, we use:

```
<filename>make</filename>
```

For user's instructions, use either:

```
<screen><prompt>$</prompt> make install</screen>
```

or:

```
Run the command <userinput>make</userinput>
```

Be aware that the former is shown alone on its proper line, while the latter is embedded within the line flow. If you use the `screen` tag, you should prepend either a `$` or a `#` followed by a space before the instruction, depending on how the command should be run, as non root for the former, as root for the latter. Moreover, with the `screen` command, we should check that the line is not too long, split it if needed, and add a backslash to indicate the splitting.

To embed chunk of code, we use:

```
<programlisting>
<![CDATA[
Run the command <userinput>make</userinput>]]>
</programlisting>
```

As a workaround a bug in **fop**, we use a special processing instruction to insert page breaks for PDF production. If the break is the same for A4 and USLetter format, the instruction is:

```
<?pagebreak?>
```

If it is only for A4 format, the instruction is:

```
<?pagebreaka4?>
```

Likewise for USLetter format only, it is:

```
<?pagebreakus?>
```

Similar processing instructions are used to insert line breaks for PDF production:

```
<?linebreak?>
<?linebreaka4?>
<?linebreakus?>
```

4.8. Recommendation

Do not use `simplesect` as it messes the table of contents.

Avoid to add blank lines or unnecessary white spaces in the files, it may break the files production and has the disadvantage to increase the files size.

A chapter should at least contains an `id`, a `title`, and either a `para` or section tag. Be aware that you cannot use an isolated `para` tag after a section.

All list items should use a `para` tag to embed their contents. If all of the items contents are very short, i.e. fit into one line, you may want to use the following attribute to suppress the additional line between items:

```
<itemizedlist spacing="compact">
```

The same applies to ordered lists.

Avoid contractions. Use *you will* instead of *you'll*.

Use the spell checker to correct any misspelling.

4.9. Contact us

If you find errors in the manual, or just want to add more, please [contact \[xix\]](#) the manual's maintainer, as well as if you have questions on how to edit the manual that are not addressed in this appendix. Often, you can look to the sources to see how things are done.

Appendix D. GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software

distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.