MiscSearchText

Category Of:	Text
Declared In:	MiscSearchText.h
Adopts protocols:	SearchableText

Category Description

The MiscSearchText category implements the SearchableText protocol for NeXT's AppKit's Text class.

This is a complete implementation, providing all functional aspects of the SearchableText protocol, including literal and regular expression pattern searching, replacing all instances of a match in a section of the text, backwards and forwards searching, and more.

[*These instructions assume that you are not linking the MiscKit library into your application.*] To incorporate this category into your project, instantly adding searching functionality to your existing Text objects, all you need to do is add the files *MiscSearchText.m, MiscTBMK.c,* and *regexpr.c* to "Other Sources" in your project file with ProjectBuilder, then add the files *MiscSearchText.h, MiscTBMK.h,* and *regexpr.h* to "Headers".

This category contains one method that is not part of the SearchableText protocol. The method **setRegExprSyntax:** calls the **re_set_syntax()** function of the regular expression package with the parameter passed to it. This function should probably be called once (to set the syntax) before any searching is done, and then not changed. The regular expression package is compatible with the GNU regular expression package. To get the most out of the regular expression searching facilities, the sections on regular expression syntax should be read in the documentation of the GNU regular expression package (which is not included with this package, but can be obtained via anonymous ftp from prep.ai.mit.edu in

/pub/gnu/regex-0.12.tar.gz, and no doubt other sites as well).

Two features of the search routines of this category should be mentioned, two prevent inexplicable trouble or confusing behavior in a project. The first allows the user to abort the **replaceAll:with:mode:regexpr:cases:** and **searchFor:mode:reverse:regexpr:cases:position:size:** methods with Command-period, the user-abort command key. The function **NXUserAborted()** is called in a number of places in these functions, but the abort may not be instantaneous. The second feature is that the **replaceAll:with:mode:regexpr:cases:** method does not make the changes directly to the text of the Text object during the operation, but the text is changed just before the method returns. Thus if a user aborts the operation, no changes will be made. Currently however, this also makes the replace all operation fairly slow relative to the "change-in-place" algorithm when few changes are made (all but a tiny fraction of this time is the time needed to read the modified text back into the text object).

A feature or two of the regular expression searching also bears mentioning: the text matched is the largest string that can be matched, on a line starting from the current position being examined. This has two effects. The first is that much more text than what you might intuitively expect ends up being selected in some cases. After you understand why the first is occuring, then comes the second: that much less text

than what you might have intuitively expected is being selected in some cases. In particular, forward regular expression matching from position A (assuming a successful search) will match text from point B to point C, where B and C are "after" point A. Reverse searching from a point D, after point C, may not make the same match, B to C, made by the forward search. Also, the regular expression searching/matching may (or may not) ignore the boundaries of the previously selected text; i.e. text which was previously selected may be part of the next match, a phenomenon especially noticable in reverse searches. To see these problems, search for the regular expression "(.*)" in a sample of C code. This version of the MiscSearchText category does not correct for these effects; subsequent versions may correct the latter problem.

And finally, here is an implementation of the Text class's **findText:ignoreCase:backwards:wrap:** instance method, implemented to use the fast Misc_TBMK stream searching routines. This cannot be added to the category, because NeXT put the **findText:ignoreCase:backwards:wrap:** method in a category, and category methods cannot reliably override methods in other categories. However, if you are subclassing Text, you may find this method useful. There is an important caveat however: the code below, as of the time of the current release of this MiscFindPanel package, had not been tested. Not one bit. You may need to do some debugging of your own.

```
- (BOOL)findText:(const char *)string ignoreCase:(BOOL)ignoreCaseflag backwards:(BOOL)backwardsflag wrap:(BOOL)wrapflag
{
   Misc_TBMKpattern pat;
   int s1, e1, s2, e2, mp, pos, r;

   s1 = s2 = e1 = e2 = 0;
   if (backwardsflag && sp0.cp<0)
      {s1 = textLength; e1 = -textLength;}
   else if (backwardsflag)
      {s1 = sp0.cp; e1 = -sp0.cp; s2 = textLength; e2 = spN.cp-textLength;}
   else if (sp0.cp<0)
      {s1 = 0; e1 = textLength;}
   else
      {s1 = spN.cp; e1 = textLength-spN.cp; s2 = 0; e2 = sp0.cp;}
   [self stream];
   if (textStream==0)
      return NO;
   pat = Misc_TBMKpattern_alloc(string, strlen(string), backwardsflag, ignoreCaseflag);
   if (pat==0)
      return NO;
   NXSeek(textStream, s1, NX_FROMSTART);
   r = Misc_TBMKsearch_stream(pat, textStream, e1, 0, &mp);
```

```
    pos = mp+s1;
    if (!r && e2>0 && wrapflag) {
        NXSeek(textStream, s2, NX_FROMSTART);
        r = Misc_TBMKsearch_stream(pat, textStream, e2, 0, &mp);
        pos = mp+s2;
    }
    Misc_TBMKpattern_free(&pat);
    if (!r)
        return NO;
    [self setSel:pos :pos+strlen(string)];
    [self scrollSelToVisible];
    return YES;
}
```