NAME

wafeapp – how to use and configure wafe perl written wafe applications

DESCRIPTION

A typical *wafe* application consists of two parts: the front-end *wafe* process and an application program, implemented in an arbitrary programming language, which talks to the front-end via stdio. Because *wafe* was developed using the extensible *tcl* shell (from John Ousterhout), an application program can dynamically submit requests to the front-end to build up the graphical user interface; the application can even down-load application specific procedures into the front-end.

The contributed perl written application programs load the *wafe.pl* package, which allows a uniform configuration for various applications. Per default wafe.pl tries to load during the startup of an application the configuration files *.wafe_options*, *$WafeLib/system.waferc* and *.waferc* in the home directory of the current user. The first file can be used to set certain options for one or all applications (you can for instance disable three dimmensional buttons for all applications by writing in your *.wafe_options* file the line:

```
$opt_3 = 1;
```

The options set here might influence the evaluation of the other configuration files. If a *.waferc* file exists it can be used to redefine variables defined in in the application program or in *$WafeLib/system.waferc*. The standard configuration file, *system.waferc*, is situated in the *wafe* library, which is generally */usr/lib/X11/wafe*.

The configuration files are written in *perl*(1L) syntax (from Larry Wall), with the restriction that a compound statement (between curly brackets) must NOT contain empty lines.

The same startup files are loaded from several applications to achieve similar configurations and colors. The startup files may also be used to pass variables to a specific application. For this purpose, *wafe.pl* sets the *perl*(1L) variable *$appName* according to the filename of the executed program. If a users wants to start the *xwafemail* application always with the option **–F** he might add the following line into his *.waferc*:

```
$opt_F = 1 if $appName eq "wafemail";
```

COLORS AND FONTS

If a *wafe* application *xapp* starts it is registered as an toolkit application with the name *xapp* in the class *XApp* and the according resources are loaded. In addition to the standard X resource management using *app-defaults* and *.Xdefaults*, *wafe* offers an additional instrument to give applications a similar appearance. Each wafe application loading *wafe.pl* accepts the command line option **–c** <*color mode*>, where <*color mode*> is one of **blueish**, **grayish**, **greenish** or **mono**, or of the form <*color string*>**–ish** where <*color string*> is a colorname defined in *rgb.txt*, available as *colorname1* to *colorname4*. If a user wants to give an application an wheat-ish appearance he can use the command line argument **–c wheat-ish** to use the colors **wheat1** to **wheat4**.

If the command line argument **–c mono** is specified, no colors are set in *system.waferc*, and only the standard X resource management is used.

The variables for color setting are defined in *system.waferc*. If the **–c option is used,** appropriate *perl*(1L) variables are set which might be used in all application programs for background, emphasized text, information, and so on. The *system.waferc* configuration file below sets besides the color options described above variables for font settings, configurations of widgets that should be configured the same way in various wafe applications. Finally, some parameters for certain applications follow, which will differ from site to site.

The supplied sample, *system.waferc*, has the following content:

```
;# -*-perl-*-
;# the following fonts are used by
;# xwafeftp, xwafenews, xwafemail, xwafecf, xwafemc, wafecf, wafetel, wafeora
;## if you have linked wafe with libXaw3d you can uncomment the following 2 li
nes
;## to get 3d effects for buttons, scrollbars etc.
;## $opt_3 = 1 if $wafe'Packages !˜ /\ bXAW3D\ b/ || $opt_c eq 'mono';
;## $threeD = $opt_3 ? : "" : "shadowWidth 3";
$opt_3 = 1 if $wafe'Packages !˜ /XAW3D/ || $opt_c eq 'mono';
$threeD = $opt_3 ? "" : "shadowWidth 3";
$boldFont = 'font -b&h-lucida-bold-r-*-*-14-*-*-*-p-*-iso8859-*'
unless $boldFont;
$normalFont = 'font -b&h-lucida-medium-r-*-*-14-*-*-*-p-*-iso8859-*'
unless $normalFont;
$buttonFont = $threeD ?
   'font -b&h-lucida-bold-r-*-*-12-*-*-*-p-*-iso8859-*' :
   'font -b&h-*-medium-r-*-*-12-*-*-*-p-*-iso8859-*'
unless $buttonFont;
$textFont = 'font -b&h-*-medium-r-*-*-12-*-*-*-m-*-*-*'
unless $textFont;
$opt_c = $opt_c || "grayish";  # default color mode
;## $defaultResources = "*bottomShadow #2e92b0 *topShadow #c9ecf5"; # blueish
;## $defaultResources = "*bottomShadow #679d67 *topShadow #cdfdcd"; # greenish#
;## $defaultResources = "*bottomShadow gray60 *topShadow gray90"; # greyish
COLOR: {
   if ($opt_c eq "blueish") {
$buttonColors = $threeD ?
   "background PowderBlue foreground black" :
   "background steelblue foreground white";
$backGround = "background PowderBlue";
$highLight = "background cyan";
      $roColors = "background aliceblue foreground black";
      $sbColors = "foreground black background steelblue $threeD";
      $infoColors = "foreground black background lightsteelblue $threeD";
last COLOR;
   }
   if ($opt_c eq "greenish") {
$buttonColors = $threeD ?
  "background MediumSeaGreen foreground black" :
  "background SeaGreen foreground white";
$backGround = "background MediumSeaGreen";
$infoColors = "background DarkSeaGreen2 foreground black $threeD";
      $roColors = "background DarkSeaGreen1 foreground black";
      $sbColors = "foreground black background DarkSeaGreen4 $threeD";
$highLight = "background mediumSpringGreen";
last COLOR;
   }
   if ($opt_c =˜ /ˆ(.*)-ish/) {
```

```
$buttonColors = "background ${1}4 foreground white";
$backGround = "background ${1}3";
$highLight = "background ${1}2";
        $roColors = "background ${1}1 foreground black";
        $sbColors = "foreground black background ${1}4 $threeD";
        $infoColors = "foreground black background ${1}2 $threeD";
last COLOR;
    }
    if ($opt_c =~ /gr[ae]yish/) {
$buttonColors = $threeD ?
    "background gainsboro foreground black" :
    "background gray55 foreground white";
$backGround = "background gainsboro";
$highLight = "background mediumSpringGreen";
$roColors = "background gray95 foreground black";
$sbColors = "foreground black background gray55 $threeD";
$infoColors = "foreground black background gray75 $threeD";
last COLOR;
    }
    print "no such colormode <$opt_c>, using mono" if $opt_c ne 'mono';
    if ($opt_c eq "mono") {
$highLightTextFont = 'font -b&h-*-bold-*-*-*-12-*-*-*-m-*-*-*';
last COLOR;
    }
}
;# subject colors for mail
%subjColors = (
   'Ordinary-UnRead', 'foreground black background white',
   'Ordinary-Read', 'foreground gray60 background white',
   'Urgent-UnRead', 'foreground red background white',
   'Urgent-Read', 'foreground orange background white',
   ) if $opt_c ne "mono";
$menueAtts = $highLight;
if ($threeD) {
    $threeDNarrow = "$threeD internalWidth 1 internalHeight 0";
    $defaultResources .= " *.highlightThickness 1";
    $twoD = "shadowWidth 0 internalHeight 3";
}
$buttonAtts =  $threeD ?
    "$buttonFont $threeDNarrow $buttonColors" :
    "$buttonFont shapeStyle oval $buttonColors";
%textResources = (
'*Text*insertFile*Form*Command', $buttonAtts,
'*Text*insertFile*Form*Label', "$backGround $boldFont",
'*Text*insertFile*Form*Text', $roColors,
'*Text*insertFile*Form', $backGround,
'*Text*search*Form*Command', $buttonAtts,
'*Text*search*Form*Label', "$backGround $boldFont",
'*Text*search*Form*Text', $roColors,
'*Text*search*Form', $backGround,
'*Scrollbar', $sbColors,
);
;# why does :Meta<Key>a not work for me?
$meta      = 'Meta';       # some people need 'Mod1'
```

```
$delete    = 'apLineDel';  # some people need 'Delete'
$backspace = 'Delete';     # some people need 'Backspace'
@genericTextActions = (
"~Shift $meta<Key>a: insert-string()",
"Shift $meta<Key>A: insert-string(Î)",
"~Shift $meta<Key>o: insert-string()",
"Shift $meta<Key>O: insert-string(^)",
"~Shift $meta<Key>u: insert-string()",
"Shift $meta<Key>U: insert-string()",
        "$meta<Key>s: insert-string()",
"None<Key>Prior: previous-page()",
"None<Key>Next: next-page()",
"$meta<Key>Prior: beginning-of-file()",
"$meta<Key>Next: end-of-file()",
"Shift<Key>$backspace: delete-previous-word()",
"Shift<Key>$delete: delete-next-word()",
"$meta<Key>Right: forward-word()",
"$meta<Key>Left: backward-word()",
"$meta<Key>Down: scroll-one-line-down()",
"$meta<Key>Up: scroll-one-line-up()",
"None<Key>$delete: delete-next-character()",
"Ctrl<Key>$backspace: exec(delLine w)",
"$meta<Key>d: kill-selection()",
"<Key>Find: search(forward)",
"<Key>Insert: insert-file()",
"None<Key>$backspace: exec(backSpace w)",
        "<Key>End: end-of-line()",
        "<Key>Home: beginning-of-line()",
        "$meta<Key>$delete: exec(delLine w)",
        "$meta<Key>Insert: exec(copyCurrentLineToNext w)",
);
@textActions = (
@genericTextActions,
        "$meta<Key>$backspace: exec(smartBackSpace w)",
        "Shift<Key>braceright: exec(smartCloseBrace w)",
        "None<Key>Return: exec(newlineAndIndent w)",
        "$meta<Key>Return: newline-and-indent()",
);
$defaultResources .= " *foreground  black  *shadowWidth 0";
;# Mac like backspace
&TclProc('
    proc backSpace {w} {
      textGetSelectionPos $w from to;
      if $from!=$to {
         callActionProc $w {} kill-selection;
 textSetSelection $w $from $from
       }
       {callActionProc $w {} delete-previous-character}
     }
    proc delLine {w} {
        callActionProc $w {} insert-string "a";
        callActionProc $w {} beginning-of-line;
        callActionProc $w {} kill-to-end-of-line;
        callActionProc $w {} delete-next-character
```

```
}
proc getChar {w} {
    set ip [textGetInsertionPoint $w];
    textSourceRead $w $ip txt 1;
    return [string range $txt(ptr) 0 0];
}
proc getCurrentLine {w} {
    set ip [textGetInsertionPoint $w];
    set bl [textSourceScan $w $ip EOL left 1 false];
    set el [textSourceScan $w $ip EOL right 1 false];
    textSourceRead $w $bl txt [expr $el-$bl];
    return [string range $txt(ptr) 0 [expr $el-$bl-1]]
}
proc getColumn {w} {
    set ip [textGetInsertionPoint $w];
    set bl [textSourceScan $w $ip EOL left 1 false];
    return [expr $ip-$bl];
}
proc gotoColumn {w column} {
    set ip [textGetInsertionPoint $w];
    set bl [textSourceScan $w $ip EOL left 1 false];
    set el [textSourceScan $w $ip EOL right 1 false];
    if {ip+column<el} {
        set ip [expr $bl+column]
    }
    {
        set ip $el
    }
    textSetInsertionPoint $w $ip;
}
proc setCurrentLine {w s} {
    set ip [textGetInsertionPoint $w];
    set bl [textSourceScan $w $ip EOL left 1 false];
    set el [textSourceScan $w $ip EOL right 1 false];
    set txt(ptr) "$s";
    set txt(firstPos) 0;
    set txt(length) [string length $s];
    textReplace $w $bl $el txt;
}
proc copyCurrentLineToNext {w} {
    set cl [getCurrentLine $w];
    set ip [textGetInsertionPoint $w];
    textDisplayCaret $w false;
    callActionProc $w {} end-of-line;
    callActionProc $w {} insert-string $cl;
    textSetInsertionPoint $w $ip;
    textDisplayCaret $w true;
}
proc newlineAndIndent {w} {
    textDisplayCaret $w false;
    set cl [getCurrentLine $w];
    set column [expr [getColumn $w]-1];
    callActionProc $w {} newline-and-indent;
    callActionProc $w {} insert-string "a";
```

```
        callActionProc $w {} delete-previous-word;
        if { [string compare  [string range $cl $column $column]] == 0 } {
            callActionProc $w {} insert-string  "    "
        };
        textDisplayCaret $w true
    }
    proc smartCloseBrace {w} {
        textDisplayCaret $w false;
        set ip [textGetInsertionPoint $w];
        set l [getCurrentLine $w];
        set column [getColumn $w];
        set ll [string range $l 0 [expr $column-1]];
        set lr [string range $l $column end];
        ;
        if { [regsub "    " $ll {} ll] != 0 } {
            set ip [expr $ip-3]
        }
        { set ip [expr $ip+1] };
        setCurrentLine $w $ll$lr;
        textSetInsertionPoint $w $ip;
        ;
        textDisplayCaret $w true;
    }
    proc smartBackSpace {w} {
        set ip [textGetInsertionPoint $w];
        set l [getCurrentLine $w];
        set column [getColumn $w];
        set ll [string range $l 0 [expr $column-1]];
        set lr [string range $l $column end];
        if { [regsub "    " $ll {} ll] != 0 } {
            set ip [expr $ip-4];
            setCurrentLine $w $ll$lr;
            textSetInsertionPoint $w $ip;
        }
        { callActionProc $w {} delete-previous-character; };
    }
');
;#
;# for reply, forward and followup
$mailIncludePrefix = " |> ";
;#
;# default mail host, if no alias exists and no address is given
$defaultMailHost = wi;
;#
;# default encoding for sending mail (TeX, Swe, Mime, None)
$defaultMailEncoding = 'Mime';
;#
;# configuration for xwafeftp
;# the following FTP servers appear in the server menu
@ftpServers = (
  'ftp.wu-wien.ac.at',
  'wuarchive.wustl.edu',
  'ftp.uu.net',
  'gatekeeper.dec.com',
```

```
  'prep.ai.mit.edu',
  'export.lcs.mit.edu',
      );
;#
;# configuration for news reader
$defaultNntpHost = "nestroy";
$defaultNntpOrganization = "WU-Wien";
$defaultNntpDistribution = "world";
;#
;# configuration for wafecf
$wafecfSearchPath = "bib-sample.db:$WafeLib/bib-sample.db";
;#
;# configuration for xwafetel
$wafetelDB = $wafetelDBalt = 'demo/demo@T:dec1.wu-wien.ac.at:wi';
;#
;# configuration for xwafeora
# 345 ""
%wafeoraDB = (
   'Ablage', 'demo/demo@T:dec1.wu-wien.ac.at:wi',
   'Paper', 'demo/demo@T:dec1.wu-wien.ac.at:wi',
   'Ordner', 'demo/demo@T:dec1.wu-wien.ac.at:wi',
   );
```

WAFELIB	location of the wafe library (typically /usr/lib/X11/wafe)
HOME	login directory of the current user

FILES

```
$WAFELIB/system.waferc
$HOME/.waferc
```

SEE ALSO

wafe(n),
X(1),
xwafemail(n),
xwafenews(n),
xwafeftp(n),
xwafeora(n),
xwafetel(n),
xdirtree(n),
xdirtree(n),
xprojektor(n),
xbm(n),
xwafemc(n),
xruptimes(n),
xnetstats(n),
xiostats(n),
xvmstats(n),
xiostats(n),
xwafeping(n),
xwafecf(n)

COPYRIGHT

Wirtschaftsuniversitaet Wien,
Abteilung fuer Wirtschaftsinformatik
Augasse 2-6,
A-1090 Vienna, Austria
neumann@wu-wien.ac.at, nusser@wu-wien.ac.at

AUTHORS

Gustaf Neumann, Stefan Nusser