

```

#!/bin/bash
# shellcheck source=kz_common.sh
#####
# Change GUI.
#
# Written by Karel Zimmer <info@karelzimmer.nl>, CC0 1.0 Universal
# <https://creativecommons.org/publicdomain/zero/1.0>, 2015-2023.
#####
#####
# Import
#####

program_path=$(cd "$(dirname "$(realpath "$0")")" && pwd)
source "$program_path"/kz_common.sh 2> >(systemd-cat) || exit 1

#####
# Variables
#####

program_name='kz-gset'
program_desc=$(gettext 'Change GUI')
display_name=${program_name/kz-/kz }

usage="$(eval_gettext "Usage: \${display_name} [-a|--addfavaft FAVORITE] \
[-b|--addfavbef FAVORITE]
          [-d|--delfav FAVORITE] [-l|--list] [-r|--resetfavs]
          [-f|--addappfolder APPFOLDER] [-x|--delappfolder APPFOLDER]
          \${options_usage}")"
help="$(eval_gettext "Usage: \${display_name} [OPTION...]"")

$program_desc.

$(gettext 'Options:')
$(gettext "Mandatory arguments to long options are mandatory for short \
options too.")

-a, --addfavaft FAVORITE
          add FAVORITE at the end
-b, --addfavbef FAVORITE
          add FAVORITE at the beginning
-d, --delfav FAVORITE
          delete FAVORITE
-l, --list      show favorites and app folders
-r, --resetfavs
          reset favorites
-f, --addappfolder APPFOLDER
          add APPFOLDER
-x, --delappfolder APPFOLDER
          delete APPFOLDER')

$options_help"
options_short+='a:b:d:lrf:x:'
options_long+=",addfavaft:,addfavbef:,delfav:,list,resetfavs,\
addappfolder:,delappfolder:"

appfolder=''
config_a=''
config_b=''
desktopfile=''
favorite_argument=''
folder_argument=''
option_addappfolder=false
option_addfavaft=false

```

```
option_addfavbef=false
option_delappfolder=false
option_delfav=false
option_list=false
option_resetfavs=false
```

```
#####
# Functions
#####
```

```
function check_input {
    local -i rc=0
    local parsed=''

    parsed=$(
        getopt --alternative          \
               --options             "$options_short" \
               --longoptions         "$options_long"  \
               --name                 "$display_name" \
               --                     "$@"
    ) || rc=$?
    if [[ $rc -ne $ok ]]; then
        printf '%s\n' "$usage_line"
        exit $error
    fi
    eval set -- "$parsed"
    process_option "$@"

    while true; do
        case $1 in
            -a|--addfavaf)
                if $option_addfavaf; then
                    printf "$display_name: $1 $2: %s\n$usage_line\n" \
                           "$(gettext 'too many options')"
                    exit $error
                fi
                option_addfavaf=true
                favorite_argument=$2
                shift 2
                ;;
            -b|--addfavbef)
                if $option_addfavbef; then
                    printf "$display_name: $1 $2: %s\n$usage_line\n" \
                           "$(gettext 'too many options')"
                    exit $error
                fi
                option_addfavbef=true
                favorite_argument=$2
                shift 2
                ;;
            -d|--delfav)
                if $option_delfav; then
                    printf "$display_name: $1 $2: %s\n$usage_line\n" \
                           "$(gettext 'too many options')"
                    exit $error
                fi
                option_delfav=true
                favorite_argument=$2
                shift 2
                ;;
            -l|--list)
                option_list=true
                shift
        esac
    done
}
```

```

        ;;
-r|--resetfavs)
    option_resetfavs=true
    shift
    ;;
-f|--addappfolder)
    if $option_addappfolder; then
        printf "$display_name: $1 $2: %s\n$usage_line\n" \
            "$(gettext 'too many options')"
        exit $error
    fi
    option_addappfolder=true
    folder_argument=$2
    shift 2
    ;;
-x|--delappfolder)
    if $option_delappfolder; then
        printf "$display_name: $1 $2: %s\n$usage_line\n" \
            "$(gettext 'too many options')"
        exit $error
    fi
    option_delappfolder=true
    folder_argument=$2
    shift 2
    ;;
--)
    shift
    break
    ;;
*)
    shift
    ;;
esac
done

if [[ -n "$*" ]]; then
    printf "$display_name: $*: %s\n$usage_line\n" \
        "$(gettext 'arguments are not allowed')"
    exit $error
fi

appfolder=$folder_argument
desktopfile=$favorite_argument.desktop
}

```

```

function process_input {
    if $option_addfavaft; then
        process_option_addfavaft
    elif $option_addfavbef; then
        process_option_addfavbef
    elif $option_delfav; then
        process_option_delfav
    elif $option_list; then
        process_option_list
    elif $option_resetfavs; then
        process_option_resetfavs
    elif $option_addappfolder; then
        process_option_addappfolder
    elif $option_delappfolder; then
        process_option_delappfolder
    else
        capture_gui_changes
    fi
}

```

```
}
```

```
function process_option_addfavft {
    local favorite_apps=''

    favorite_apps=$(gsettings get org.gnome.shell favorite-apps)
    if [[ $favorite_apps = '@as []' ]]; then
        gsettings set org.gnome.shell favorite-apps "['$desktopfile']"
    elif ! printf '%s' "$favorite_apps" |
        grep --quiet --word-regexp --regexp="$desktopfile"; then
        gsettings set org.gnome.shell favorite-apps "$(
            printf '%s' "$favorite_apps" |
            cut --delimiter=']' --fields=1
        ), '$desktopfile']"
    fi
}
```

```
function process_option_addfavbef {
    local favorite_apps=''

    favorite_apps=$(gsettings get org.gnome.shell favorite-apps)
    if [[ $favorite_apps = '@as []' ]]; then
        gsettings set org.gnome.shell favorite-apps "['$desktopfile']"
    elif ! printf '%s' "$favorite_apps" |
        grep --quiet --word-regexp --regexp="$desktopfile"; then
        gsettings set org.gnome.shell favorite-apps \
            "['$desktopfile', $(
                printf '%s' "$favorite_apps" | cut --delimiter='[' --fields=2
            )]"
    fi
}
```

```
function process_option_delfav {
    local favorite_apps=''

    favorite_apps=$(gsettings get org.gnome.shell favorite-apps)
    if [[ $favorite_apps = '@as []' ]]; then
        :
    elif [[ $favorite_apps = "['$desktopfile']" ]]; then
        gsettings set org.gnome.shell favorite-apps "[]"
    else gsettings set org.gnome.shell favorite-apps "$(
        printf '%s' "$favorite_apps" |
        sed --expression="s/'$desktopfile', //g" |
        sed --expression="s/, '$desktopfile'//"
    )"
    fi
}
```

```
function process_option_list {
    gsettings get org.gnome.shell favorite-apps
    gsettings get org.gnome.desktop.app-folders folder-children
}
```

```
function process_option_resetfavs {
    gsettings reset org.gnome.shell favorite-apps
}
```

```
function process_option_addappfolder {
```

```

local app_folders=''
local schema=''
local path=''

app_folders=$(gsettings get org.gnome.desktop.app-folders folder-children)

if [[ $app_folders = '@as []' ]]; then
    set org.gnome.desktop.app-folders folder-children ["'$appfolder'"]
elif ! printf '%s' "$app_folders" |
    grep --quiet \
        --word-regexp \
        --regexp="$appfolder"; then
    gsettings set org.gnome.desktop.app-folders folder-children "$(
        printf '%s' "$app_folders" | cut --delimiter=']' --fields=1
    ), '$appfolder']"
fi
schema=org.gnome.desktop.app-folders.folder
path=/org/gnome/desktop/app-folders/folders/$appfolder/
gsettings set "$schema":"$path" name "$appfolder"
gsettings set "$schema":"$path" translate true
gsettings set "$schema":"$path" categories ["'$appfolder'"]
}

function process_option_delappfolder {
    local app_folders=''
    local schema=''
    local path=''

    app_folders=$(gsettings get org.gnome.desktop.app-folders folder-children)

    if [[ $app_folders = '@as []' ]]; then
        :
    elif [[ $app_folders = ["'$appfolder'"] ]]; then
        gsettings set org.gnome.desktop.app-folders folder-children [""]
    else gsettings set org.gnome.desktop.app-folders folder-children "$(
        printf '%s' "$app_folders" | sed --expression="s/, '$appfolder'//"
    )"
    fi
    schema=org.gnome.desktop.app-folders.folder
    path=/org/gnome/desktop/app-folders/folders/$appfolder/
    gsettings reset-recursively "$schema":"$path"
}

function capture_gui_changes {
    config_a=$(mktemp -t "$program_name-A-XXXXXXXXXX.lst")
    save_configuration_database A "$config_a"
    info "$(gettext 'Now make the change in the graphical workspace.')"
    wait_for_enter
    config_b=$(mktemp -t "$program_name-B-XXXXXXXXXX.lst")
    save_configuration_database B "$config_b"
    report_database_changes "$config_a" "$config_b"
}

function save_configuration_database {
    # shellcheck disable=SC2034
    local fase=${1:-unknown}
    local output_file=${2:-unknown}

    gsettings list-recursively > "$output_file" 2> >($logcmd)

    sort --unique \

```

```

        --output="$output_file" \
        "$output_file"

    info "$(eval_gettext "Config database content captured (\$fase).")"
}

function report_database_changes {
    local output_file1=${1:-unknown}
    local output_file2=${2:-unknown}

    info "$(
gettext 'CHANGES in the configuration database,
< is the old setting (A),
> is the new setting (B):')"
    diff "$output_file1" "$output_file2" | grep --regexp='[>|<]' || true
    info "
$(eval_gettext "When CHANGES execute: \${blue}gsettings set CHANGE\${normal}")"
    rm "$config_a" "$config_b"
}

function term_script {
    exit $ok
}

#####
# Main
#####

function main {
    init_script "$@"
    check_input "$@"
    process_input
    term_script
}

main "$@"

```