Advanced Scripting with (k)dcop and kdialog

Stephan Binner <binner@kde.org>

2004-08-28, aKademy, Ludwigsburg

What is DCOP?

- Desktop COmmunication Protocol
- Interprocess communication
- Usage with C++, Bash, Perl, Python, ...
- Required to make KDE “work”
- Additional functions for KDE users
- Interfaces are light and easy to add
- If you miss a useful function, request it!

dcop

- Console DCOP client
- Usage: `dcop[options][application [object[function[arg1][arg2]..]]]`
- `dcop` lists registered applications
- `dcop <application>` lists app's objects
- `dcop <application> <object>` lists functions and their signatures

dcop Usage (1)

```
user@host:~> dcop
  kwin
  kicker
  kded
  knotify
  kio_uiserver
  klauncher
  kdesktop
  klipper
  ksmserver
  kaccess
  konsole-4358
```

dcop Usage (2)

```
user@host:~> dcop klipper
  qt
  KDebug
  MainApplication-Interface
  klipper
```

dcop Usage (3)

```
user@host:~> dcop klipper klipper
  QCStringList interfaces()
  QCStringList functions()
  QString getClipboardContents()
  void setClipboardContents(QString s)
  void clearClipboardContents()
  void clearClipboardHistory()
  QStringList getClipboardHistoryMenu()
  QString getClipboardHistoryItem(int i)
  int newInstance()
  void quitProcess()
```

dcop Usage (4)

```
host@user:~>dcop klipper klipper
  getClipboardContents↵
  Hello World!

host@user:~>dcop klipper klipper
  setClipboardContents "Just testing..."↵

host@user:~>dcop klipper klipper
  getClipboardContents↵
  Just testing...
```

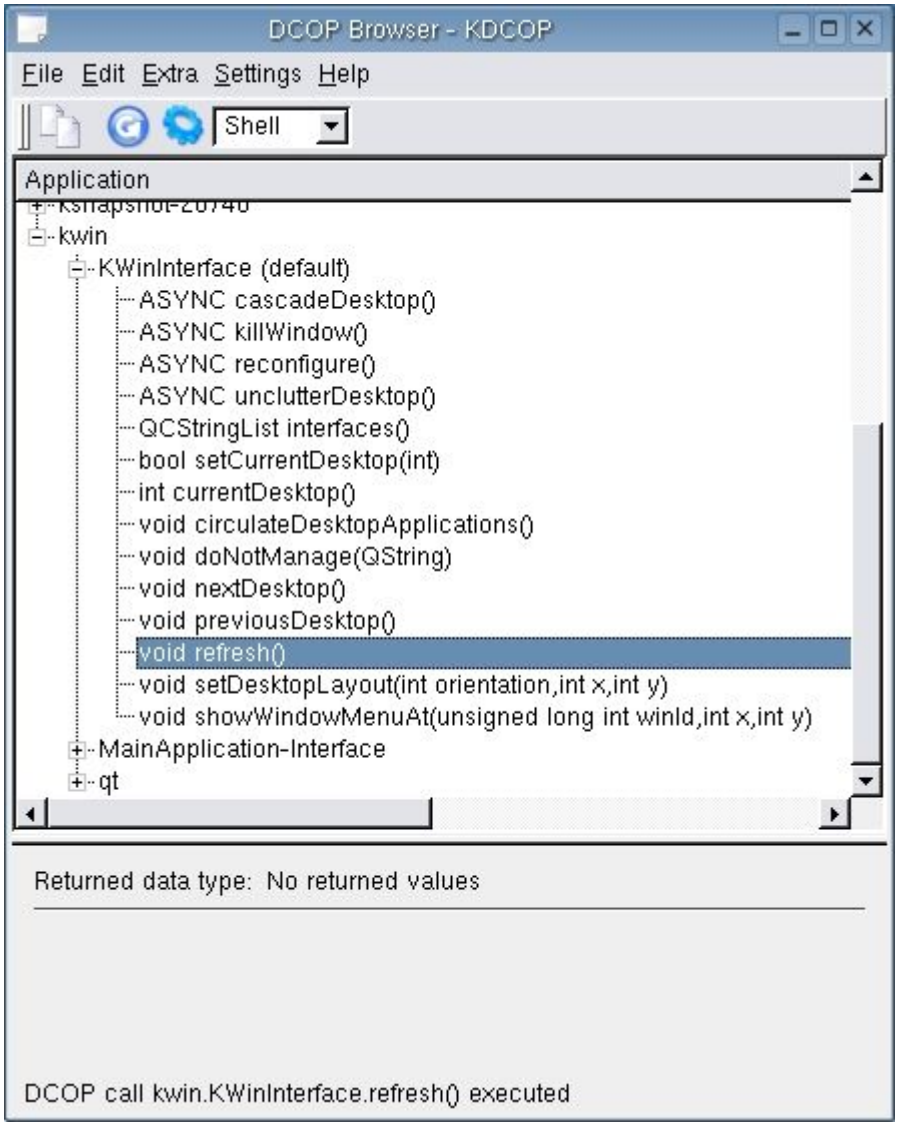
dcopstart, dcopfind

- `dcopstart <app>`
  - Starts <app> & returns <appid> on stdout
- `dcopfind[-l][-a]<appid>[<object-id> [<select_func> [args]]]`
  - Finds an existing DCOP application/object
  - <app-id>/<object-id> may end with a '*' as wildcard
  - Returns by default a 'DCOP reference'
- `dcop <dcopref> <function> <args>`

DCOP Reference/Utilities

- Form: `DCOPRef(<appid>, <objectid>)`
- `dcopref <appid> <objectid>`
  - Creates DCOPRef from appid and object
- `dcopclient <dcopref>`
  - Extracts the appid from dcopref
- `dcopobject <dcopref>`
  - Extracts the objectid from dcopref

kdcop


DCOP Browser - KDCOP
File Edit Extra Settings Help
Shell
Application
kwin
KWinInterface (default)
ASYNC cascadeDesktop()
ASYNC killWindow()
ASYNC reconfigure()
ASYNC unclutterDesktop()
QCStringList interfaces()
bool setCurrentDesktop(int)
int currentDesktop()
void circulateDesktopApplications()
void doNotManage(QString)
void nextDesktop()
void previousDesktop()
void refresh()
void setDesktopLayout(int orientation,int x,int y)
void showWindowMenuAt(unsigned long int winId,int x,int y)
MainApplication-Interface
qt
Returned data type: No returned values
DCOP call kwin.KWinInterface.refresh() executed

KDialog

- In the tradition of dialog and Xdialog
- Makes KDE dialogs available to scripts
- Includes different types of message boxes, input dialogs, file dialogs and passive popup
- New in KDE 3.3: Progress dialog
  - Requires dcop for communication

Progress Dialog Example

```
dcopRef=`kdialog --progressbar "Press Cancel
  at Any time" 10`

dcop $dcopRef showCancelButton true

until test "true" == `dcop $dcopRef
  wasCancelled`; do

  sleep 1

  inc=$((`dcop $dcopRef progress` + 1))

  dcop $dcopRef setProgress $inc;

done

dcop $dcopRef close
```

Demonstration

References

- “Shell Scripting with KDE Dialogs” Tutorial
  - http://developer.kde.org/documentation/tutorials/kdialog/t1.html
- “Connect KDE applications using DCOP”
  - http://www-106.ibm.com/developerworks/linux/library/l-dcop/index.html?ca=dgr-lnxw12ConnectKDE
- “Scripting with DCOP – Boost Your Efficiency”
  - http://www.linux-magazine.com/issue/36/KDE_Scripting_DCOP.pdf
- “Automation of KDE 2”
  - http://developer.kde.org/documentation/tutorials/automation/