Implementing context-sensitive help on Symbian

Overview

This code example describes how to implement context-sensitive help for an application.

IMPORTANT: In order to use context-sensitive help, you need to apply for a protected UID from Symbian for the help file (if you use an uprotected UID, the help file(s) of the application cannot be launched from the application). The UID3 of the actual application binary does not necessarily have to be protected. However, it should be different from the UID of the help file.

This snippet can be self-signed (the protected UID of the Help file does not prevent self-signing).

Steps

1. Create a directory for help files (for example, `help`).

2. Write the help file (.rtf). Use the `cshelp2000.dot` file as a template. The file can be found under the SDK (for example, `C:\Symbian\9.2\S60_3rd_FP1\Epoc32\cshlpcmp_template\cshelp2000.dot`).

3. Open the rtf file in a text editor and delete the formatting instruction that Word has created:

- `{\*\wgrffmtfilter XXXX}` in Word version 11.8134.8132, or
- `{\*\generator Microsoft Word 10.0.6612;}` in Word version 10.0.6612.

4. Write the XML file (`help\help.xml`) for the application:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml:stylesheet href="\epoc32\tools\cshlpcmp\xsl\CSHproj.xsl"
    title="CS Help project" type="text/xsl"?>
<!DOCTYPE cshproj SYSTEM "\epoc32\tools\cshlpcmp\dtd\CSHproj.dtd">

<cshproj>
    <helpfileUID>0x2000E190</helpfileUID>   <!-- From help file -->
    <directories>
        <input></input>
        <output></output>
        <working></working>
    </directories>
    <files>
        <source>
            <file>help.rtf</file>
        </source>
        <destination>help.hlp</destination>
        <customization>custom.xml</customization>
    </files>
</cshproj>
```

5. Write the XML file (`help\custom.xml`) for style declarations:

```
<?xml version="1.0"?>
<!DOCTYPE cshcust SYSTEM "\epoc32\tools\cshlpcmp\dtd\CSHcust.dtd">
<?xml:stylesheet href="\epoc32\tools\cshlpcmp\xsl\cshcust.xsl"
    title="CS Help customization" type="text/xsl"?>

<cshcust>
    <parastyle name="body" font="sansserif" size="10"/>
    <parastyle name="note" font="sansserif" size="10" left="20">
        <b>Note:</b>
```


(C) Copyright Nokia 2014. All rights reserved.

```
    </parastyle>
    <body style="body"/>
    <titlestyle fontstyle="sansserif" size="10"/>
    <listbullet1style bulletchar="8226"/>
    <listbullet2style bulletchar="8226"/>
    <lists leftindent="20"/>
</cshcust>
```

6. Create a makefile (`group\help.mk`) that compiles the help files:

```
makmake :
    cshlpcmp ..\help\help.xml

ifeq (WINS, $(findstring WINS, $(PLATFORM)))
    copy ..\help\help.hlp $(EPOCROOT)\epoc32\$(PLATFORM)\c\resource\help
endif


clean :
    del ..\help\help.hlp
    del ..\help\help.hlp.hrh

bld :
    cshlpcmp ..\help\help.xml

ifeq (WINS, $(findstring WINS, $(PLATFORM)))
    copy ..\help\help.hlp $(EPOCROOT)\epoc32\$(PLATFORM)\c\resource\help
endif


freeze lib cleanlib final resource savespace releasables :
```

Note: When editing the makefile, make sure that you use tabulators instead of spaces, or you get the error "HELP.MK:27: *** missing separator. Stop."

7. Edit `group\bld.inf` to contain the makefile created above:

```
#ifdef EKA2 //3rd edition
gnumakefile help.mk
Application_S60_3rd_ed.mmp
#else       //1st and 2nd edition
Application_S60_2nd_ed.mmp
#endif
```

Note: Remember to run `bldmake bldfiles` after editing the `bld.inf` file.

8. Add the help files and the help launcher library to the mmp file of the application (`group\[app].mmp`):

```
USERINCLUDE       ..\help

SOURCEPATH        ..\help
DOCUMENT help.rtf
DOCUMENT help.xml
DOCUMENT custom.xml
```


(C) Copyright Nokia 2014. All rights reserved.

```
LIBRARY hlplch.lib  // help
```

9. Add .hlp file to the package file of the application (`sis\[app].pkg`):

```
"..\help\help.hlp" - "!:\resource\help\help.hlp"
```

10. Add the constant for the help command (`EClientHelp`) to `inc\[app].hrh` file:

```
// Client enumerate command codes
enum TClientIds
    {
    EClientHelp = 1 // start value must not be 0
    };
```

11. Add UID to `inc\[app]Application.h`:

```
const TUid KUidHelpFile = {0x2000E190};  // From the help file
```

12. Add the localization strings to localized data files (for example, `data\[app]_loc.l01`):

```
#define STRING_r_app_options_help  "Help"
```

13. Add the menu item to the `RESOURCE MENU_PANE` section of the application resource file (`data\[app].rss`):

```
MENU_ITEM
    {
    command = EClientHelp;
    txt = STRING_r_app_options_help;
    }
```

14. Add the code associated to the help command to the `HandleCommandL` function of the AppUI (`src\[app]AppUi.cpp`) file:

```
#include <hlplch.h>  // HlpLauncher
```

```
case EClientHelp:
    {
    CArrayFix <TCoeHelpContext>* buf = CCoeAppUi::AppHelpContextL();
    HlpLauncher::LaunchHelpApplicationL(iEikonEnv->WsSession(), buf);
    break;
    }
```

15. Override the `HelpContextL` function:

(`inc\[app]AppUi.h` part):

```
private: // from CAknAppUi

    /*
     * HelpContextL()
     *
```


(C) Copyright Nokia 2014. All rights reserved.

```
     * Returns the help context for this application
     *
     * Returns:
     *     A pointer to the help context
     *
     */
    CArrayFix<TCoeHelpContext>* HelpContextL() const;
```

(`src\[app]AppUi.cpp` part):

```
#include "[app]Application.h"
#include "help.hlp.hrh"
```

```
// ----------------------------------------------------------------------------
// CClientAppUi::HelpContextL()
//
// Return the help context for this application.
// ----------------------------------------------------------------------------
CArrayFix <TCoeHelpContext>* CClientAppUi::HelpContextL() const
    {
    CArrayFixFlat <TCoeHelpContext>* array =
        new (ELeave) CArrayFixFlat <TCoeHelpContext>(1);
    CleanupStack::PushL(array);
    // KContextApplication below should refer to the context declared in
    // help.rtf
    array->AppendL(TCoeHelpContext(KUidHelpFile, KContextApplication));
    CleanupStack::Pop(array);
    return array;
    }
```

16. Override the `GetHelpContext` function:

(`inc\[app]AppView.h` part):

```
public:

    /**
     * From CoeControl
     * Identify the help context so that the framework can look up
     * the corresponding help topic
     *
     * @param aContext The help context
     */
    void GetHelpContext(TCoeHelpContext& aContext) const;
```

(`src\[app]AppView.cpp` part):

```
#include "[app]Application.h"
#include "help.hlp.hrh"
```

```
// ----------------------------------------------------------------------------
// CClientAppView::GetHelpContext()
```


(C) Copyright Nokia 2014. All rights reserved.

```
// Gets the control's help context.
// -------------------------------------------------------------------------------

void CClientAppView::GetHelpContext(TCoeHelpContext& aContext) const
    {
    // Get the help context for the application
    aContext.iMajor = KUidHelpFile;

    // KContextApplication below should refer to the context declared in
    // help.rtf
    aContext.iContext = KContextApplication;
    }
```

See also

- Symbian Context-sensitive Help Basics

(C) Copyright Nokia 2014. All rights reserved.