Listening for incoming SMS messages using Symbian C++

Overview

This snippet demonstrates how to listen for SMS messages that arrive in the inbox folder.

MMP file

The following libraries and capabilities are required:

```
CAPABILITY ReadUSerData WriteUserData UserEnvironment
LIBRARY msgs.lib
LIBRARY smcm.lib
LIBRARY gsmu.lib
```

Header file

```
#include <e32base.h>
#include <msvapi.h>           // MMsvSessionObserver
#include <msvids.h>           // Folder Ids
#include <smut.h>             // KUidMsgTypeSMS
#include <txtrich.h>          // CRichText


//  CONSTANTS
const TInt KSmsMessageLength = 512;
const TInt KAddressLength = 64;

class MSmsEngineObserver
    {
    public:
        virtual void MessageReceived(TDesC& aMsg, TDesC& aAddr) = 0;
    };


class CYourSmsEngine :    public CBase,
                              public MMsvSessionObserver
    {
    public:
        static CYourSmsEngine* NewL(MSmsEngineObserver* aObserver);
        static CYourSmsEngine* NewLC(MSmsEngineObserver* aObserver);
        virtual ~CYourSmsEngine();

    private:
        void HandleSessionEventL(TMsvSessionEvent aEvent,
        TAny* aArg1, TAny* aArg2, TAny* aArg3);

    private:
        void ConstructL();
        CYourSmsEngine(MSmsEngineObserver* aObserver);

    private:
        // Observers SmsEngine states
        MSmsEngineObserver*     iObserver;

        // Message body
        TBuf<KSmsMessageLength> iMessage;

        // Address (phonenumber)
        TBuf<KAddressLength>    iAddress;

        // Session with the messaging server
        CMsvSession*            iMsvSession;

        // CMsvEntry accesses and acts upon a particular Message Server entry
        CMsvEntry*              iMsvEntry;

        // Id of a new message
        TMsvId iNewMessageId;

        // Id of the sent message
        TMsvId iSentMessageId;
    };
```

Source file

```
#include "cyoursmsengine.h"
```


(C) Copyright Nokia 2013. All rights reserved.

```
#ifdef __WINS__
    const TMsvId KObservedFolderId = KMsvDraftEntryId;
#else
    const TMsvId KObservedFolderId =  KMsvGlobalInBoxIndexEntryId;
#endif

const TMsvId KInbox = KMsvGlobalInBoxIndexEntryId;


CYourSmsEngine* CYourSmsEngine::NewL(MSmsEngineObserver* aObserver)
    {
    CYourSmsEngine* self = NewLC(aObserver);
    CleanupStack::Pop(self);
    return self;
    }

CYourSmsEngine* CYourSmsEngine::NewLC(MSmsEngineObserver* aObserver)
    {
    CYourSmsEngine* self = new (ELeave) CYourSmsEngine(aObserver);
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
    }

void CYourSmsEngine::ConstructL()
    {
    // SMS automatic receiving needs a session to the messaging server
    iMsvSession = CMsvSession::OpenAsyncL(*this);
    }

CYourSmsEngine::CYourSmsEngine(MSmsEngineObserver* aObserver)
: iObserver(aObserver)
    {
    }

CYourSmsEngine::~CYourSmsEngine()
    {
    delete iMsvEntry;

    if (iMsvSession)
        iMsvSession->Cancel();

    delete iMsvSession;
    }


void CYourSmsEngine::HandleSessionEventL(
TMsvSessionEvent aEvent, TAny* aArg1, TAny* aArg2, TAny* /*aArg3*/)
    {
    switch (aEvent)
        {
        case EMsvServerReady:
            {
            // Initialise iMsvEntry
            if (!iMsvEntry)
                {
                iMsvEntry = CMsvEntry::NewL(*iMsvSession, KInbox,
                                            TMsvSelectionOrdering());
                }
            break;
            }
        case EMsvEntriesCreated:
            {
            // Only look for changes in the Inbox
            if (aArg2 &&  *(static_cast<TMsvId*>(aArg2)) == KObservedFolderId)
                {
                CMsvEntrySelection* entries =
                static_cast<CMsvEntrySelection*>(aArg1);
                if( entries->Count() >= 1 )
                    {
                    iNewMessageId = entries->At(0);
                    }
                else
                    {
                    return;
                    }
                }
            break;
            }
        case EMsvEntriesChanged:
            {
            // Look for changes. When using the emulator
            // observed folder is drafts, otherwise inbox
            if (aArg2 &&  *(static_cast<TMsvId*>(aArg2)) == KObservedFolderId)
                {
                CMsvEntrySelection* entries =
                static_cast<CMsvEntrySelection*>(aArg1);
                if(!entries && entries->Count() < 1 )
                    {
                    return;
                    }
                else if (iNewMessageId == entries->At(0))
                    {
                    if( !iMsvEntry )
                        {
                        return;
                        }

                    // Set entry context to the new message
                    iMsvEntry->SetEntryL(iNewMessageId);

                    // Check the type of the arrived message and
                    // that the message is complete
                    // Only SMS's are our consern
                    if (iMsvEntry->Entry().iMtm != KUidMsgTypeSMS ||
                        !iMsvEntry->Entry().Complete())
                        {
                        return;
                        }

                    // Read-only store
                    CMsvStore* store = iMsvEntry->ReadStoreL();
                    CleanupStack::PushL(store);

                    // Get address of received message.
                    iAddress.Copy(iMsvEntry->Entry().iDetails);

                    // Body text
                    if (store->HasBodyTextL())
                        {
```


(C) Copyright Nokia 2013. All rights reserved.

```
                        CRichText* richText = CRichText::NewL(
                           CEikonEnv::Static()->SystemParaFormatLayerL(),
                           CEikonEnv::Static()->SystemCharFormatLayerL());
                        CleanupStack::PushL(richText);

                        store->RestoreBodyTextL(*richText);
                        TPtrC ptr =
                        richText->Read(0, richText->DocumentLength());
                        iMessage.Copy(ptr);

                        CleanupStack::PopAndDestroy(richText);
                        CleanupStack::PopAndDestroy(store);

                        // Send message and phone number to caller
                        iObserver->MessageReceived(iMessage, iAddress);
                        }
                    else
                        {
                        CleanupStack::PopAndDestroy(store);
                        }
                    }
                }
            break;
            }
        default:
            {
            break;
            }
        }
    }
```

Postconditions

The incoming SMS message is received and the message and phone number are parsed.

See also

Deleting an incoming SMS message using Symbian C++

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.