Showing a global list query using CAknGlobalListQuery

Overview

This code snippet shows how to create a global list query `CAknGlobalListQuery`. A global list query is constructed by calling the `NewL()` method and is executed using the `ShowListQueryL()` method. An active object has to be constructed to handle the dismissal of the dialog. This example uses the generic class `CGlobalQueryHandlerAO` that is also used with other global query code snippets. The class `CGlobalQueryContainer` is used to launch the query and it is also the observer that implements the interface `MGlobalQueryObserver`.

Note: `CAknGlobalListQuery` can only hold very simple items. Tabulators can not be used in the items.

This snippet can be self-signed.

MMP file

The following libraries are required:

```
LIBRARY cone.lib            //CEikonEnv, CCoeEnv
LIBRARY avkon.lib           //Avkon resources
LIBRARY euser.lib           //CActive, CleanupStack
LIBRARY bafl.lib            //CDesCArray
```

Resource file

.rss

```
#include <eikon.rh>
#include <avkon.rsg>
#include <avkon.rh>

//...
RESOURCE ARRAY r_global_query_item_list_array
    {
    items=
        {
        LBUF {txt="<item 1>"; },
        LBUF {txt="<item 2>"; },
        LBUF {txt="<item 3>"; },
        LBUF {txt="<item 4>"; }
        };
    }
```

Header files

- observer interface

```
#ifndef GLOBALQUERYOBSERVER_H
#define GLOBALQUERYOBSERVER_H

class MGlobalQueryObserver
    {
public:
```


(C) Copyright Nokia 2014. All rights reserved.

```
    //keep these methods short running because they are called
    //from within an active object
    virtual void ProcessOkOptionL() = 0;
    virtual void ProcessCancelOptionL() = 0;
    virtual void ProcessDoneOptionL() = 0;
    virtual void ProcessSelectedItemL(const TInt aItem) = 0;
    };

#endif  // GLOBALQUERYOBSERVER_H
```

- handler active object

```
#ifndef GLOBALQUERYHANDLERAO_H
#define GLOBALQUERYHANDLERAO_H

#include <e32base.h>  //CActive
#include "GlobalQueryObserver.h"

class CGlobalQueryHandlerAO : public CActive
    {
public:
    ~CGlobalQueryHandlerAO();
    static CGlobalQueryHandlerAO* NewL(MGlobalQueryObserver& aObserver);
    static CGlobalQueryHandlerAO* NewLC(MGlobalQueryObserver& aObserver);
public:
    void StartHandler();
private:
    CGlobalQueryHandlerAO(MGlobalQueryObserver& aObserver)
            : CActive (EPriorityStandard), iObserver(aObserver){};
    void ConstructL();
private: // from CActive
    void RunL();
    void DoCancel();
private: // data
    MGlobalQueryObserver& iObserver;
    };
#endif //GLOBALQUERYHANDLERAO_H
```

- container / observer

```
#ifndef GLOBALQUERYCONTAINER_H
#define GLOBALQUERYCONTAINER_H

#include <badesca.h> // CDesCArray
#include <coecntrl.h> // CCoeControl
#include "GlobalQueryObserver.h"
#include <akngloballistquery.h> //CAknGlobalListQuery
class CGlobalQueryHandlerAO;

class CGlobalQueryContainer : public CCoeControl, MGlobalQueryObserver
    {
public:
//...
void MakeExampleQueryL();
public: // from MGlobalQueryObserver
```


(C) Copyright Nokia 2014. All rights reserved.

```
    void ProcessOkOptionL(); //not used
    void ProcessCancelOptionL(); //not used
    void ProcessDoneOptionL(); //not used
    void ProcessSelectedItemL(const TInt aItem);
//...
private:
    CGlobalQueryHandlerAO*       iQueryHandlerAO;
    CAknGlobalListQuery*         iGlobalListQuery;
    };
#endif  // GLOBALQUERYCONTAINER_H
```

Source files

- handler active object

```
#include "GlobalQueryHandlerAO.h"
#include <avkon.hrh> //EAknSoftkeyOk, EAknSoftkeyCancel, EAknSoftkeyDone

CGlobalQueryHandlerAO::~CGlobalQueryHandlerAO() { Cancel(); }

void CGlobalQueryHandlerAO::ConstructL()
    {
    CActiveScheduler::Add(this);
    }

CGlobalQueryHandlerAO* CGlobalQueryHandlerAO::NewLC(MGlobalQueryObserver& aObserver)
    {
    CGlobalQueryHandlerAO* self = new (ELeave) CGlobalQueryHandlerAO(aObserver);
    CleanupStack::PushL(self);
    self->ConstructL();
    return self;
    }

CGlobalQueryHandlerAO* CGlobalQueryHandlerAO::NewL(MGlobalQueryObserver& aObserver)
    {
    CGlobalQueryHandlerAO* self = CGlobalQueryHandlerAO::NewLC(aObserver);
    CleanupStack::Pop(self);
    return self;
    }

void CGlobalQueryHandlerAO::DoCancel(){}

void CGlobalQueryHandlerAO::StartHandler() { SetActive(); }

void CGlobalQueryHandlerAO::RunL()
    {
    if (iStatus == EAknSoftkeyOk)
        iObserver.ProcessOkOptionL();
    else if(iStatus == EAknSoftkeyCancel)
        iObserver.ProcessCancelOptionL();
    else if(iStatus == EAknSoftkeyDone)
        iObserver.ProcessDoneOptionL();
    else
        iObserver.ProcessSelectedItemL(iStatus.Int());
```


(C) Copyright Nokia 2014. All rights reserved.

```
    Cancel();
    }
```

- container / observer

```
#include "GlobalQueryContainer.h" //CGlobalQueryContainer
#include "GlobalQueryHandlerAO.h" //CGlobalQueryHandlerAO
#include <GlobalQuery_0xED0C36BE.rsg> //Resources

void CGlobalQueryContainer::ConstructL(const TRect& aRect)
    {
    CreateWindowL();
    SetRect(aRect);
    ActivateL();

    MakeExampleQueryL();
    }

void CGlobalQueryContainer::MakeExampleQueryL()
    {
    //start the query handler and show global list query
    iQueryHandlerAO = CGlobalQueryHandlerAO::NewL(*this);
    iQueryHandlerAO->StartHandler();
    iGlobalListQuery = CAknGlobalListQuery::NewL();

    CDesCArray* itemArray = iCoeEnv->ReadDesCArrayResourceL(
                                    R_GLOBAL_QUERY_ITEM_LIST_ARRAY );
    CleanupStack::PushL(itemArray);

    iGlobalListQuery->ShowListQueryL(itemArray, iQueryHandlerAO->iStatus);

    CleanupStack::PopAndDestroy(itemArray);
    }

CGlobalQueryContainer::~CGlobalQueryContainer()
    {
    //...
    delete iQueryHandlerAO;
    delete iGlobalListQuery;
    }

void CGlobalQueryContainer::ProcessOkOptionL()
    {
    //not used
    }
void CGlobalQueryContainer::ProcessCancelOptionL()
    {
    //not used
    }
void CGlobalQueryContainer::ProcessDoneOptionL()
    {
    //not used
    }
void CGlobalQueryContainer::ProcessSelectedItemL(const TInt aItem)
    {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    //Item selected, do something...
    TBuf<10> item;
    item.AppendNum(aItem);
    iEikonEnv->InfoWinL(_L("Process Item:"),item);
    }
```

Postconditions

The example code shows how to create a global list query, handle its dismissal, and how the active object handler informs an observer class that the dialog has been closed.

See also

- Showing a global message query using CAknGlobalMsgQuery
- Showing a global confirmation query using CAknGlobalConfirmationQuery
- Showing a global progress dialog using CAknGlobalProgressDialog

(C) Copyright Nokia 2014. All rights reserved.