Capturing foreground window server events on Symbian

This article illustrates a Symbian C++ class that captures window server events sent to the foreground application.

Introduction

`CFgrObserver` illustrates how to capture foreground events in a background application. This implementation should work with all platform versions, but with S60 3rd Edition you need to have the SwEvent capability.

The usage is quite simple. Just implement the defined callback interface in your class and construct an instance of the `CFgrObserver`. Whenever a GUI application is taking focus you will get a notification through the callback functions. The `aAppUid` argument will have the UID of the application getting the focus.

Note that when constructing this class the `aWsSession` session variable should be connected. Otherwise the construction process will fail.

If you are only interested in your own application's focus changes, you could overwrite the Application user interface class `HandleForegroundEventL()` method.

Source code

Header required:

```
#include <w32std.h>
#include <APGWGNAM.H> //CApaWindowGroupName
```

Library Required:

```
LIBRARY   ws32.lib
LIBRARY   apgrfx.lib //CApaWindowGroupName
```

Capability Required:

```
capability    SwEvent
```

ForeGroundObserver.cpp

```
CFgrObserver* CFgrObserver::NewL(RWsSession& aWsSession,MFgrCallBack& aCallBack)
 {
 CFgrObserver* self = CFgrObserver::NewLC(aWsSession,aCallBack);
 CleanupStack::Pop(self);
 return self;
 }


CFgrObserver* CFgrObserver::NewLC(RWsSession& aWsSession,MFgrCallBack& aCallBack)
 {
 CFgrObserver* self = new (ELeave) CFgrObserver(aWsSession,aCallBack);
 CleanupStack::PushL(self);
 self->ConstructL();
 return self;
 }


CFgrObserver::CFgrObserver(RWsSession& aWsSession,MFgrCallBack& aCallBack)
:CActive(EPriorityHigh),iCallBack(aCallBack), iWsSession(aWsSession), iWg(aWsSession)
{
}

CFgrObserver::~CFgrObserver()
{
 Cancel();
 iWg.Close();
}

void CFgrObserver::ConstructL()
{
 CActiveScheduler::Add(this);

 User::LeaveIfError(iWg.Construct((TUint32)&iWg, EFalse));
 iWg.SetOrdinalPosition(-1);
 iWg.EnableReceiptOfFocus(EFalse);

 CApaWindowGroupName* wn=CApaWindowGroupName::NewLC(iWsSession);
 wn->SetHidden(ETrue);
 wn->SetWindowGroupName(iWg);
 CleanupStack::PopAndDestroy();

 User::LeaveIfError(iWg.EnableFocusChangeEvents());

 Listen();
}
```


(C) Copyright Nokia 2013. All rights reserved.

```
void CFgrObserver::RunL()
{
 if (iStatus == KErrNone)
 {
  TWsEvent e;
  iWsSession.GetEvent(e);
 }

 TInt wgid = iWsSession.GetFocusWindowGroup();

 CApaWindowGroupName* gn;
 gn = CApaWindowGroupName::NewLC(iWsSession, wgid);

 iCallBack.ForegroundEventL(gn->AppUid());

 CleanupStack::PopAndDestroy(); // gn

 if (iStatus != KErrCancel)
 {
  Listen();
 }
}

void CFgrObserver::DoCancel()
{
 iWsSession.EventReadyCancel();
}

void CFgrObserver::Listen()
{
 iWsSession.EventReady(&iStatus);
 SetActive();
}
```

ForeGroundObserver.h

```
class MFgrCallBack
{
public:
 virtual void ForegroundEventL(TUid aAppUid) = 0;
};

class CFgrObserver : public CActive
{
public:
 static CFgrObserver* NewL(RWsSession& aWsSession,MFgrCallBack& aCallBack);
 static CFgrObserver* NewLC(RWsSession& aWsSession,MFgrCallBack& aCallBack);
 virtual ~CFgrObserver();
private:
 CFgrObserver(RWsSession& aWsSession,MFgrCallBack& aCallBack);
 void ConstructL();
 void RunL();
 void DoCancel();
 void Listen();
private:
 MFgrCallBack&  iCallBack;
 RWsSession&     iWsSession;
 RWindowGroup    iWg;
};
```

Capturing Foreground events in a non-GUI application

If you wish to capture foreground events using non-GUI applications, the above solution will not work fully as most of the time you will receive an `aAppUid` of 0 for the application that caused the foreground event. An alternate solution would be monitoring `EEventWindowGroupsChanged` events by changing the `ConstructL` method, change the line

```
User::LeaveIfError(iWg.EnableFocusChangeEvents());
```

to

```
User::LeaveIfError(iWg.EnableGroupChangeEvents());
```

As you will receive an event callback every time a **EEventWindowGroupsChanged** will occur, you will need to check if the window group id of the application in the foreground is the same as the one of the application which caused the event:

```
void CFgrObserver::RunL()
{
 if (iStatus == KErrNone)
 {
  TWsEvent e;
  iWsSession.GetEvent(e);
 }

 TInt wgid = iWsSession.GetFocusWindowGroup();

 CApaWindowGroupName* gn;
 gn = CApaWindowGroupName::NewLC(iWsSession, wgid);
      ///retrieve the task in the foreground
 TApaTask app =  iTaskList->FindByPos(0);
      ///compare the window group ids, call the callback only if the ids are equal
      if (app.WgId() == wgid)
              iCallBack.ForegroundEventL(gn->AppUid());


 CleanupStack::PopAndDestroy(); // gn

 if (iStatus != KErrCancel)
 {
  Listen();
 }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Note the usage of `iTaskList` which is a `TApaTaskList*` member, can be initialized in the `ConstructL()` method:

```
iTaskList = new (ELeave) TApaTaskList(iWsSession);
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.