Capturing all keys in Non-GUI applications

`CKeyCapturer2` illustrates how to capture all keys inside an application that doesn't implement the standard application frameworks and thus can't use the normal means on getting key events. If you want to capture keys in an application that implements the normal application framework, you can utilize the standard `OfferKeyEventL()` method implemented in your `CCoeControl` derived class.

This example implementation makes a window that takes focus and thus gets all key events as any other application taking focus would. Also since it is taking focus, other applications are not receiving any key input. If you just want to capture a few HW keys and let other applications still receive key events you might be better off using the CKeyCapturer implementation.

Header required:

```
#include <w32std.h>
```

Library Required:

```
LIBRARY   ws32.lib
```

Capability Required:

```
capability    SwEvent
```

CapturingKeys2.cpp

```
CKeyCapturer2* CKeyCapturer2::NewL(MKeyCallBack& aObserver)
 {
 CKeyCapturer2* self = CKeyCapturer2::NewLC(aObserver);
 CleanupStack::Pop(self);
 return self;
 }

CKeyCapturer2* CKeyCapturer2::NewLC(MKeyCallBack& aObserver)
 {
 CKeyCapturer2* self = new (ELeave) CKeyCapturer2(aObserver);
 CleanupStack::PushL(self);
 self->ConstructL();
 return self;
 }


CKeyCapturer2::CKeyCapturer2(MKeyCallBack& aObserver)
:CActive(EPriorityHigh),iObserver(aObserver)
{

}

CKeyCapturer2::~CKeyCapturer2()
{
 Cancel();
 iWg.Close();
 iWsSession.Close();
}

void CKeyCapturer2::ConstructL()
{
 User::LeaveIfError(iWsSession.Connect());

 CActiveScheduler::Add(this);

 iWg=RWindowGroup(iWsSession);
 User::LeaveIfError(iWg.Construct((TUint32)&iWg, EFalse));

 iWg.SetOrdinalPosition(1, ECoeWinPriorityAlwaysAtFront+2);
 iWg.EnableReceiptOfFocus(ETrue);

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

 Listen();
}

void CKeyCapturer2::RunL()
{
 if (iStatus == KErrNone)
 {
  TWsEvent e;
  iWsSession.GetEvent(e);
  TInt type = e.Type();

  switch (type)
  {
  case EEventKey:
   if(iObserver.KeyCapturedL(e))
   {
    TInt wgId = iWsSession.GetFocusWindowGroup();
    iWsSession.SendEventToWindowGroup(wgId, e);
   }
   break;
  case EEventKeyUp:
  case EEventKeyDown:
   break;
  };
```


(C) Copyright Nokia 2013. All rights reserved.

```
 }

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

void CKeyCapturer2::DoCancel()
{
 iWsSession.EventReadyCancel();
}


void CKeyCapturer2::Listen()
{
 iWsSession.EventReady(&iStatus);
 SetActive();
}
```

CapturingKeys2.h

```
class MKeyCallBack
{
public:
 virtual TBool KeyCapturedL(TWsEvent aEvent) = 0;
};



class CKeyCapturer2 : public CActive
{
public:
 static CKeyCapturer2* NewL(MKeyCallBack& aObserver);
 static CKeyCapturer2* NewLC(MKeyCallBack& aObserver);
 virtual ~CKeyCapturer2();
private:
 CKeyCapturer2(MKeyCallBack& aObserver);
 void ConstructL();
 void RunL();
 void DoCancel();
 void Listen();
private:
 MKeyCallBack&  iObserver;
 RWsSession      iWsSession;
 RWindowGroup    iWg;
};
```

Related Links:

- Capturing keys in background

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.