



ID	CS000859	Creation date	March 25, 2008
Platform	S60 3rd Edition, FP1	Tested on devices	Nokia N95
Category	Symbian C++	Subcategory	UI

Keywords (APIs, classes, methods, functions): CCoeControl

Overview

Symbian OS provides a large set of ready-made controls but sometimes that is not enough. You can build your own UI control by deriving `CCoeControl` base class for controls.

This example shows how to define your own simple custom control that has the `CEikLabel`.

Start by deriving `CCoeControl` and implement basic methods

- construction
- layout
- size
- drawing the control
- contained controls (if any)

MMP file

The following libraries are needed:

```

LIBRARY euser.lib
LIBRARY apparc.lib
LIBRARY cone.lib
LIBRARY eikcore.lib
LIBRARY avkon.lib
LIBRARY commonengine.lib
LIBRARY eikcoctl.lib
LIBRARY gdi.lib
  
```

Header

```
#include <coectrl.h>
#include <eiklabel.h>

class CMyControl : public CCoeControl
{
public:
    static CMyControl* NewL(const TRect& aRect,const CCoeControl* aParent=NULL);
    static CMyControl* NewLC(const TRect& aRect,const CCoeControl* aParent=NULL);
    virtual ~CMyControl();

public: // from CCoeControl
    TSize MinimumSize();

private: // from CCoeControl
    void Draw(const TRect& aRect) const;
    void SizeChanged();

private: // own methods
    CMyControl();
    void ConstructL(const TRect& aRect,const CCoeControl* aParent = NULL);

private: // data
    CEikLabel* iStatusText;
};
```

Source

```
CMyControl* CMyControl::NewL(const TRect& aRect,const CCoeControl* aParent)
{
    CMyControl* self = CMyControl::NewLC(aRect,aParent);
    CleanupStack::Pop(self);
    return self;
}

CMyControl* CMyControl::NewLC(const TRect& aRect,const CCoeControl* aParent)
{
    CMyControl* self = new(ELeave) CMyControl();
    CleanupStack::PushL(self);
    self->ConstructL(aRect,aParent);
    return self;
}

CMyControl::CMyControl()
{
}

CMyControl::~CMyControl()
{
    // NOTE: Does not delete iStatusText because we do not own it
}

void CMyControl::ConstructL(const TRect& aRect,const CCoeControl* aParent)
{
    // No owner, so create an own window
    if(aParent == NULL)
    {
        CreateWindowL();
    }
}
```

CS000859_-_Custom_control

```
    }
    // Use Parent's window
    else
    {
        // This is component in a compound control
        SetContainerWindowL(*aParent);
    }

    // Initialize component array
    InitComponentArrayL();

    // Create contained controls
    iStatusText = new (ELeave) CEikLabel;
    iStatusText->SetContainerWindowL(*this);
    iStatusText->SetTextL(_L("HelloWorld"));

    // Store component to component array
    Components().AppendLC(iStatusText);
    CleanupStack::Pop(iStatusText);

    SetRect(aRect); // or
    //SetExtentToWholeScreen(); //NOTE: Can not see CBA buttons

    // The application should call this function on
    // all controls that are not components in a compound control
    if(aParent == NULL)
    {
        ActivateL();
    }
}

TSize CMyControl::MinimumSize()
{
    // Get CEikLabel minium size and grow it
    // that is this control MinimumSize.
    // Custom control also needs a few other methods so it can be laid out
    // and drawn. For example, custom controls usually implement MinimumSize(),
    // SizeChanged() and Draw() methods.

    // When using control in container control, set the minium size very small
    TRect rect = iStatusText->MinimumSize();
    rect.Grow(TSize(2,2));
    return rect.Size();

    // When using the control in a dialog, set the control size large
    //return Rect().Size();
}

void CMyControl::SizeChanged()
{
    // Responds to size changes to set the size and position of the contents
    // of this control. For a simple control this might include text or
    // graphics. For a compound control this sets the size and position of the
    // component. It has an empty default implementation and should be
    // implemented by the CCoeControl-derived class.
    // The function is called whenever SetExtent(), SetSize(), SetRect(),
    // SetCornerAndSize(), or SetExtentToWholeScreen() are called on
    // the control.
    if (iStatusText)
    {
        TRect labelRect(Rect());
        labelRect.Shrink(TSize(5,2));
    }
}
```

```

        iStatusText->SetRect (labelRect);
    }
}

void CMyControl::Draw(const TRect& /*aRect*/) const
{
    CWindowGc& gc = SystemGc();
    gc.SetBrushColor (KRgbBlue);
    gc.Clear(Rect());
}

```

Using CCoeControl

The CMyControl custom control was added to the (S60 3rd FP1) MultiViews example and these are the changes in .cpp.

```

void CMultiViewsView1::DoActivateL( const TVwsViewId& /*aPrevViewId*/,
                                     TUid /*aCustomMessageId*/,
                                     const TDesC8& /*aCustomMessage*/)
{
    iControl = CMyControl::NewL(ClientRect());
}

void CMultiViewsView1::DoDeactivate()
{
    if (iControl)
    {
        AppUi()->RemoveFromStack(iControl);
        delete iControl;
        iControl = NULL;
    }
}

void CMultiViewsView1::HandleSizeChange( TInt aType )
{
    if( iControl )
    {
        iControl->HandleResourceChange( aType );
        if ( aType==KEikDynamicLayoutVariantSwitch )
        {
            TRect rect;
            AknLayoutUtils::LayoutMetricsRect (AknLayoutUtils::EMainPane, rect);
            iControl->SetRect (rect);
        }
    }
}

```

Postconditions

CMyControl can be inside another CCoeControl (compound control) or be as a single control without a parent.

See also

Custom Control Series:

- [CS000860 - Custom control: Construct from resource](#) Creating a control from a resource
- [CS000861 - Custom control: Container control](#) Creating a container control
- [CS000868 - Custom control: Focusing](#) Handling key events and changing active custom control focus
- [CS000869 - Custom control: Scrollbars](#) Adding scroll bar to custom control
- [CS000870 - Custom control: In dialog](#) Adding a custom control into CAknDialog
- [File:CustomControl.zip](#) Example code patch

Other useful articles:

- [S60 application views](#)
- [How to work with views and view architecture](#)