

Controles Compostos - Exemplo

Controles são usados para o usuário interagir com o programa. Todos os controles são derivados da classe CCoeControl, seja diretamente ou indiretamente. O Symbian dispõe de alguns controles padronizados como gridds, listas, dialogs, formulários, etc. Um view é responsável por exibir o controle para o usuário, porém em aplicações mais simples o controle pode ser o próprio view, como no exemplo abaixo.

O exemplo abaixo mostra a utilização de dois controles simples gerenciados por um controle pai (controle composto), um dos controles exibe um retângulo vermelho na tela e o outro um amarelo, a posição relativa desses controles é referente ao controle Pai.

Composto.h

```
#include <coecntrl.h>

class CControleSimples;
class CControleSimples2;

class CControleComposto : public CCoeControl
{
public:
    static CControleComposto* NewL(const TRect& aRect);
    static CControleComposto* NewLC(const TRect& aRect);
    virtual ~CControleComposto();
public:
    void Draw(const TRect& aRect) const;
    void SizeChanged();
    TInt CountComponentControls() const;
    CCoeControl* ComponentControl(TInt aIndex) const;
private:
    void ConstructL(const TRect& aRect);
    CControleComposto();
    CControleSimples* iTop;
    CControleSimples2* iBaixo;
    TRect iTopRet;
    TRect iBaixoRet;
    void AjustarRetangulo();
};
}
```

Composto.cpp

```
#include <coemain.h>

CControleComposto* CControleComposto::NewL(const TRect& aRect)
{
    CControleComposto* self = CControleComposto::NewLC(aRect);
    CleanupStack::Pop(self);
    return self;
}

CControleComposto* CControleComposto::NewLC(const TRect& aRect)
{
    CControleComposto* self = new (ELeave) CControleComposto;
    CleanupStack::PushL(self);
```

```
self->ConstructL(aRect);
return self;
}

CControleComposto::CControleComposto()
{
}

CControleComposto::~CControleComposto()
{
    if(iTop)
    {
        delete iTop;
        iTop = NULL;
    }
    if(iBaixo)
    {
        delete iBaixo;
        iBaixo = NULL;
    }
}

void CControleComposto::ConstructL(const TRect& aRect)
{
    CreateWindowL();
    AjustarRetangulo();
    iTop = CControleSimples::NewL(iTopRet, this);
    iBaixo = CControleSimples2::NewL(iBaixoRet, this);
    SetRect(aRect);
    ActivateL();
}

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

void CControleComposto::SizeChanged()
{
    AjustarRetangulo();
    iTop->SetRect(iTopRet);
    iBaixo->SetRect(iBaixoRet);
}

void CControleComposto::AjustarRetangulo()
{
    TRect retPrincipal = Rect();
    const TInt retPrincipalLargura = retPrincipal.Width();
    const TInt retPrincipalAltura = retPrincipal.Height() / 2;
    iTopRet.SetRect(retPrincipal.iTl, TSize(retPrincipalLargura, retPrincipalAltura));
    iBaixoRet.SetRect(TPoint(0,retPrincipalAltura), TSize(retPrincipalLargura,
    retPrincipalAltura));
}

TInt CControleComposto::CountComponentControls() const
{
```

```
//quantidade de controles
return 2;
}

CCoeControl* CControleComposto::ComponentControl(TInt aIndex) const
{
    //retorna o ponteiro de cada controle
switch(aIndex)
{
case 0:
    return iTop;
case 1:
    return iBaixo;
default:
    return NULL;
}

}
```

Simples.h

```
#include <coecntrl.h>

class CControleSimples : public CCoeControl
{
public:
    static CControleSimples* NewL(const TRect& aRect, const CCoeControl* aParent);
    static CControleSimples* NewLC(const TRect& aRect, const CCoeControl* aParent);
    virtual ~CControleSimples();
public:
    void Draw(const TRect& aRect) const;
    void SizeChanged();
private:
    void ConstructL(const TRect& aRect, const CCoeControl* aParent);
    CControleSimples();
};

}
```

Simples.cpp

```
#include <coemain.h>
#include "Simples.h"

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

CControleSimples* CControleSimples::NewLC(const TRect& aRect, const CCoeControl*
aParent)
{
    CControleSimples* self = new (ELeave) CControleSimples;
```

```
CleanupStack::PushL(self);
self->ConstructL(aRect, aParent);
return self;
}

CControleSimples::CControleSimples()
{
}

CControleSimples::~CControleSimples()
{
}

void CControleSimples::ConstructL(const TRect& aRect, const CCoeControl* aParent)
{
    SetContainerWindowL(*aParent);
    SetRect(aRect);
    ActivateL();
}

void CControleSimples::Draw(const TRect& /*aRect*/) const
{
    CWindowGc& gc = SystemGc();
    gc.Clear(Rect());
    TRect retPrincipal = Rect();
    const TInt retPrincipalLargura = retPrincipal.Width();
    const TInt retPrincipalAltura = retPrincipal.Height();

    TRect retangulo(TPoint(0,0), TSize(retPrincipalLargura,retPrincipalAltura));

    gc.SetBrushColor(KRgbRed);
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);
    gc.DrawRect(retangulo);
}

void CControleSimples::SizeChanged()
{
    //não precisa ser implementado
}
```

Simples2.h

```
#include <coecntrl.h>

class CControleSimples2 : public CCoeControl
{
public:
    static CControleSimples2* NewL(const TRect& aRect, const CCoeControl* aParent);
    static CControleSimples2* NewLC(const TRect& aRect, const CCoeControl* aParent);
    virtual ~CControleSimples2();
public:
    void Draw(const TRect& aRect) const;
    void SizeChanged();
private:
```

```
void ConstructL(const TRect& aRect, const CCoeControl* aParent);  
CControleSimples2();  
};
```

Simples2.cpp

```
#include "Simples2.h"  
  
CControleSimples2* CControleSimples2::NewL(const TRect& aRect, const CCoeControl*  
aParent)  
{  
    CControleSimples2* self = CControleSimples2::NewLC(aRect, aParent);  
    CleanupStack::Pop(self);  
    return self;  
}  
  
CControleSimples2* CControleSimples2::NewLC(const TRect& aRect, const CCoeControl*  
aParent)  
{  
    CControleSimples2* self = new (ELeave) CControleSimples2;  
    CleanupStack::PushL(self);  
    self->ConstructL(aRect, aParent);  
    return self;  
}  
  
CControleSimples2::CControleSimples2()  
{  
}  
  
CControleSimples2::~CControleSimples2()  
{  
}  
  
void CControleSimples2::ConstructL(const TRect& aRect, const CCoeControl* aParent)  
{  
    SetContainerWindowL(*aParent);  
    SetRect(aRect);  
    ActivateL();  
}  
  
void CControleSimples2::Draw(const TRect& /*aRect*/) const  
{  
    CWindowGc& gc = SystemGc();  
    gc.Clear(Rect());  
    TRect retPrincipal = Rect();  
    const TInt retPrincipalLargura = retPrincipal.Width();  
    const TInt retPrincipalAltura = retPrincipal.Height();  
  
    TRect retangulo(TPoint(0,120), TSize(retPrincipalLargura,retPrincipalAltura));  
  
    gc.SetBrushColor(KRgbYellow);  
    gc.SetBrushStyle(CGraphicsContext::ESolidBrush);  
    gc.DrawRect(retangulo);  
}
```

```
void CControleSimples2::SizeChanged()
{
}
```

ControlesAppUi.h

```
#include <aknappui.h>

class CControleComposto;

class CControlesAppUi : public CAknAppUi
{
public:
    void ConstructL();
    CControlesAppUi();
    virtual ~CControlesAppUi();

private:
    void HandleCommandL( TInt aCommand );

private:
    CControleComposto* iAppView;
};

}
```

ControlesAppUi.cpp

```
#include "Composto.h"

void CControlesAppUi::ConstructL()
{
    BaseConstructL(CAknnEnableSkin);
    iAppView = CControleComposto::NewL( ClientRect() );
}

CControlesAppUi::CControlesAppUi()
{
}

CControlesAppUi::~CControlesAppUi()
{
    if ( iAppView )
    {
        delete iAppView;
        iAppView = NULL;
    }
}

void CControlesAppUi::HandleCommandL( TInt aCommand )
{
    switch( aCommand )
    {
        case EEikCmdExit:
```

```
case EAknSoftkeyExit:  
    Exit();  
    break;  
default:  
    Panic( EControlesUi );  
    break;  
}  
}  
  
void CControlesAppUi::HandleStatusPaneSizeChange()  
{  
    iAppView->SetRect( ClientRect() );  
}
```