



FACULTAD DE INFORMÁTICA

Aplicaciones gráficas en C++ Builder Modelo Vista Controlador

Baltasar Fernández Manjón

<http://www.fdi.ucm.es/profesor/balta/>

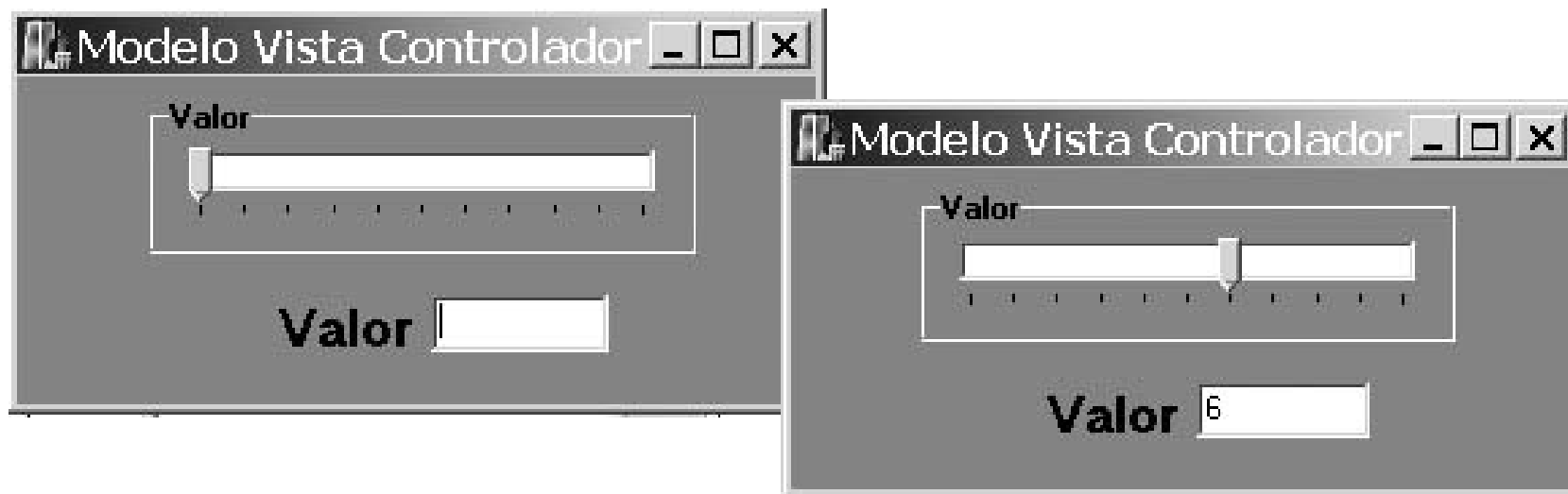


Aplicación ejemplo

Un único valor de tipo entero que se presenta simultáneamente de dos formas

- Mediante un deslizador
- Como un campo de texto

Ambos valores están siempre sincronizados de modo que el cambio de uno se refleja automáticamente en el otro





Aplicación monolítica

```
#include <vcl.h>
#pragma hdrstop
USERES("Project1.res");
USEFORM("Unit1.cpp", FormularioPrincipal);
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TFormularioPrincipal), &FormularioPrincipal);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
```

Programa principal

Project1.cpp



Aplicación monolítica

Formulario Principal

Unit1.h

```
//-----  
  
#ifndef Unit1H  
#define Unit1H  
//-----  
  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include <ComCtrls.hpp>
```

```
class TFormularioPrincipal : public TForm  
{  
    __published: // IDE-managed Components  
        TEdit *CampoValor;  
        TLabel *EtiquetaValor;  
        TTrackBar *BarraValor;  
        TGroupBox *GrupoValor;  
        void __fastcall CampoValorChange(TObject *Sender);  
        void __fastcall BarraValorChange(TObject *Sender);  
  
private: // User declarations  
        int valor; // modelo en solución monolítica  
  
public: // User declarations  
        __fastcall TFormularioPrincipal(TComponent* Owner);  
};  
  
//-----  
extern PACKAGE TFormularioPrincipal *FormularioPrincipal;  
  
//-----  
#endif
```



Aplicación monolítica

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormularioPrincipal *FormularioPrincipal;
//-----
__fastcall TFormularioPrincipal::TFormularioPrincipal(TComponent* Owner)
    : TForm(Owner)
{
}
```

Unit1.cpp

```
void __fastcall TFormularioPrincipal::CampoValorChange(TObject *Sender)
{
    // cuando cambia el valor del campo se informa al controlador
    // que debe modificar el modelo y actualizar la vista
    int valorAux;
    valorAux = StrToInt(CampoValor->Text);
    // actualizamos el modelo
    valor= valorAux;
    // actualizamos la otra visualización en la vista
    BarraValor->Position=valor;
}
```

```
void __fastcall TFormularioPrincipal::BarraValorChange(TObject *Sender)
{
    // cuando cambia el valor de la barra se informa al controlador
    // que debe modificar el modelo y actualizar la vista
    int valorAux;
    valorAux = BarraValor->Position;
    // actualizamos el modelo
    valor= valorAux;
    // actualizamos la otra visualización en la vista
    CampoValor->Text= IntToStr(valor);
}
```



Modelo Vista Controlador

```
#ifndef modeloH
#define modeloH
//-----
class Modelo {
public:
    void setValor(int num) { valor = num;}
    int getValor() {return valor;}
private:
    int valor;
};
#endif
```

modelo.h



Modelo Vista Controlador

```
#ifndef controladorH
#define controladorH
#include "vista.h"
#include "modelo.h"
class Controlador {
public:
    void ActualizarVista() { _vista->ActualizarVista(_modelo->getValor()); }
    void ActualizarModelo(int valor) {
        _modelo->setValor(valor);
        ActualizarVista();
    }
    void RegistraVista(TFormularioPrincipal *form) {_vista = form;}
    void RegistraModelo (Modelo *modelo) {_modelo = modelo;}
private:
    TFormularioPrincipal *_vista;
    Modelo *_modelo;
};
#endif
```

controlador.h



Modelo Vista Controlador

```
//-----  
#ifndef vistaH  
#define vistaH  
//-----  
#include <Classes.hpp>  
#include <Controls.hpp>  
#include <StdCtrls.hpp>  
#include <Forms.hpp>  
#include <ComCtrls.hpp>  
  
class Controlador; // para evitar referencia circular  
                    // de inclusion  
  
//-----
```

vista.h

```
class TFormularioPrincipal : public TForm  
{  
    __published: // IDE-managed Components  
        TTrackBar *BarraValor;  
        TLabel *Label1;  
        TGroupBox *GroupBox1;  
        TEdit *CampoValor;  
        void __fastcall CampoValorChange(TObject *Sender);  
        void __fastcall BarraValorChange(TObject *Sender);  
private:      // User declarations  
        Controlador *_control;  
public:      // User declarations  
        __fastcall TFormularioPrincipal(TComponent* Owner);  
        void ActualizarVista(int valor);  
        void RegControlador(Controlador *control) {_control =  
control;};  
};  
extern PACKAGE TFormularioPrincipal *FormularioPrincipal;  
#endif
```



Modelo Vista Controlador

```
#include <vcl.h>
#pragma hdrstop
#include "vista.h"
#include "controlador.h"
#pragma package(smart_init)
#pragma resource "*.dfm"
TFormularioPrincipal *FormularioPrincipal;
__fastcall TFormularioPrincipal::TFormularioPrincipal(TComponent* Owner)
: TForm(Owner){ }
void TFormularioPrincipal::ActualizarVista(int valor) {
    BarraValor->Position = valor;
    CampoValor->Text = valor;
}
void __fastcall TFormularioPrincipal::CampoValorChange(TObject *Sender) {
    _control->ActualizarModelo(StrToInt(CampoValor->Text));
}
void __fastcall TFormularioPrincipal::BarraValorChange(TObject *Sender){
    _control->ActualizarModelo(BarraValor->Position);
}
```

vista.cpp



Modelo Vista Controlador

```
#include <vcl.h>
#pragma hdrstop
USERES("proyectoMVC.res");
USEFORM("vista.cpp", FormularioPrincipal);
USEUNIT("modelo.cpp");
USEUNIT("controlador.cpp");
//-----
#include "modelo.h"
#include "controlador.h"
//-----
Modelo *model;
Controlador *control;
```

proyectoMVC.cpp

```
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
    try
    {
        Application->Initialize();
        Application->CreateForm(__classid(TFormularioPrincipal), &FormularioPrincipal);
        model = new Modelo;
        control = new Controlador;
        control->RegistraVista(FormularioPrincipal);
        control->RegistraModelo(model);
        FormularioPrincipal->RegControlador(control);
        Application->Run();
    }
    catch (Exception &exception)
    {
        Application->ShowException(&exception);
    }
    return 0;
}
```