# 使用C1ReportDesigner控件

C1ReportDesigner 在设计时显示报表，并允许用户拖动，复制，改变报表字段和区域的尺寸。该控件同时还提供了一个无限制撤销/重做的堆栈，以及一个可以支持Visual Studio自带的PropertyGrid控件的选择机制。

您可以通过使用C1ReportDesigner 控件包含一些报表设计功能在您的应用程序中，也可以写出完整的报表设计应用程序。我们随着Reports for WinForms提供了C1ReportDesigner 应用程序的完整源代码，并广泛地使用了C1ReportDesigner 控件。

写一个您自己的自定义报表设计器在许多情况下是有用的，例如：

* + 您可能想紧密地融入设计器到您的应用程序中，而不是运行一个单独的应用程序。（例如，参见微软Access报表设计器）。
	+ 您可能想自定义提供给用户的数据源，或者可以添加到报表的字段类型。（例如，你可能想使用应用程序定义的自定义数据源对象）。
	+ 您可能希望提供一个库存报表定义的菜单，在您的应用程序的的范围内是有意义的，并允许用户定义每一个库存报表的某些方面。（例如，参见微软Outlook的打印选项）。
	+ 您可以写一个比你现在所用的更好的，更强大的报表设计器应用，这使得它更容易地做对您或者您的同事更重要的事情。（例如，想报表添加字段分组）。

使用C1ReportDesigner 控件，仅需要简单地将其添加到一个窗体，添加一个包含您希望编辑的报表的C1Report组件，并在设计器控件中设置Report属性。

运行项目时，你会看到处于设计模式的报表定义。您将能够选择，移动和调整报表字段和区域的大。通过设计器所做的任何更改都将反映并存储在C1Report控件的报表定义中。您可以在任何时候使用C1Report.Save方法保存报表，或者使用C1Report.Document属性加上一个Preview 控件预览该报表。

为了建立一个完整的设计器，您必须添加其他用户界面元素：

* + 一个关联到设计器区域的**PropertyGrid**控件，因此用户可以更改字段和区域的属性。
	+ 一个数据源的选择机制，因此用户可以编辑和更改报表的数据源。
	+ 如果您希望允许用户创建，移除或者编辑报表分组，则还需要一个编辑分组的对话框。
	+ 一个用来创建新报表的向导。
	+ 通常的文件和编辑命令，用户可以加载和保存报表，使用剪贴板，并访问由C1ReportDesigner 控件提供的Undo/Redo机制。

大多数这些元素是可选的，可以根据您的需要进行省略。报表设计器应用程序源代码实现了所有这些，您可以使用源代码做为您实现的基础。

**关于本节**

本节介绍了如何使用C1ReportDesigner 控件实现一个简单的报表设计器。提供示例设计器的目的是演示如何将C1ReportDesigner 控件继承在一个设计器应用程序中。它支持多个报表加载和保存文件，编辑和预览报表，从文件中添加或删除报表，以及报告编辑撤消/重做并支持剪贴板。

大多数基于C1ReportDesigner 控件的设计器应用程序会和这里描述的这一个有着相似的功能。如果你遵循这些步骤，您将逐渐熟悉C1ReportDesigner控件的全部基本功能。

本实例设计器没有提供一些高级功能，比如说导入/导出，数据源选择/编辑，以及编辑分组。所有这些功能都通过C1ReportDesigner应用的完整版本进行支持，关于如何实现这些功能的详细介绍，您可以参考源代码。

下面的章节将描述如何逐步实现这个示例设计器。

**可用的示例报表**

完整的工程，请参见安装在ComponentOne 示例文件夹下的SimpleDesigner示例。

## 步骤一：创建并生成主窗体

该示例设计器由单个的窗体组成，并包含以下主要组件：

|  |  |  |
| --- | --- | --- |
| **控件** | **控件名称** | **描述** |
| **ListBox** | **\_list** | 显示当前加载的报表列表的列表框控件。 |
| **C1PrintPreview** | **\_c1ppv** | 用来预览报表的**C1PrintPreview**控件。 |
| **C1ReportDesigner** | **\_c1rd** | 用做设计并编辑报表的**C1ReportDesigner**控件。 |
| **PropertyGrid** | **\_ppg** | 用做对设计器选择的对象的属性进行编辑的**PropertyGrid**控件。 |
| **ToolBar** | **\_tb** | 包含对应着每一个命令按钮的工具栏控件。 |
| **C1Report** | **\_c1r** | 用做在\_**c1ppv**控件中呈现报表的**C1Report**组件。 |

窗体同时包含其他一些控件，比如说标签和分隔线，用来提高布局可用性。注意，控件的编号以及标签和分隔线的名称如下面的图片所示。该窗体应当看起来像这样：



参见窗体上的标签以定位以下控件：

 工具栏控件\_tb出现在表格的顶端。

 报表列表\_list 出现在左侧，位于PropertyGrid \_ppg的上方。

 PropertyGrid \_ppg。

 在右侧，C1ReportDesigner控件\_c1rd 填充满窗体的整个客户区。

* 预览控件\_c1ppv在设计模式是不可见的。在预览模式下，它将变成可见，同时将隐藏报表设计器。

在该示例设计器中，工具栏包含18项（14个按钮以及4个分隔线）。如果你是从头开始创建项目，在这里不用担心图像的问题。只需添加项目到\_tb控件（这可以很容易地使用ToolBarButton集合编辑器完成）并设置其名称为每个实现的命令。

## 步骤二：添加类变量和常量

在这一步中，将下面的代码添加到您的简单设计器的工程，以添加类变量以及常量：

* Visual Basic

'字段

Private \_fileName As String '当前文件的名称

Private \_dirty As Boolean '当前的文件已发生变化

'在窗体Caption区域显示的标题

Dim \_appName As String = "C1ReportDesigner Demo"

|  |  |  |
| --- | --- | --- |
|  | C# |  |
|  | // 字段private string | \_fileName; | //当前文件的名称 |
|  | private bool | \_dirty; | //当前的文件已发生变化 |

//在窗体Caption区域显示的标题

private const string \_appName = "C1ReportDesigner Demo";

## 步骤三：添加代码以更新用户界面

该简单的设计器具有可以切换启用或者禁用状态的按钮，取决于剪贴板以及Undo缓冲区是否为空，是否有文件被加载，等等。所有这些功能在一个单一的方法中实现，称为**UpdateUI**。

**UpdateUI**被频繁调用以确保UI真实地反映应用程序的状态。第一次调用应该在响应**Form\_Load**事件的方法中，用来初始化工具栏以及窗体的标题。在将下面的代码粘贴到工程之后，请记得将工具栏控件内的按钮名称匹配**UpdateUI**例程中使用到的名称。

将下面的代码添加进来以更新用户界面：

* Visual Basic

'在启动时更新界面，以显示窗体标题以并禁用剪贴板以及

'撤销/重做按钮

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

UpdateUI() End Sub

Private Sub UpdateUI()

'更新标题

\_fileName = \_appName

If \_fileName.Length > 0 Then

\_fileName = String.Format("{0} - [{1}]", \_appName, \_fileName)

If \_dirty Then \_fileName = \_fileName + " \*"

End If

'按下/释放表示设计或者预览模式的按钮

Dim design As Boolean = \_c1rd.Visible AndAlso (Not

IsNothing(\_c1rd.Report))

\_btnDesign.Pushed = design

\_btnPreview.Pushed = Not design

'启用/禁用按钮

\_btnCut.Enabled = design AndAlso \_c1rd.ClipboardHandler.CanCut

\_btnCopy.Enabled = design AndAlso \_c1rd.ClipboardHandler.CanCut

\_btnPaste.Enabled = design AndAlso \_c1rd.ClipboardHandler.CanPaste

\_btnUndo.Enabled = design AndAlso \_c1rd.UndoStack.CanUndo

\_btnRedo.Enabled = design AndAlso \_c1rd.UndoStack.CanRedo

Dim reportSelected As Boolean = design AndAlso Not (IsNothing(\_list.SelectedItem))

\_btnAddReport.Enabled = \_c1rd.Visible

\_btnDelReport.Enabled = reportSelected

\_btnAddField.Enabled = reportSelected

\_btnAddLabel.Enabled = reportSelected End Sub

* C#

//在启动时更新界面，以显示窗体标题以并禁用剪贴板以及

//撤销/重做按钮

private void Form1\_Load(object sender, System.EventArgs e)

{

UpdateUI();

}

private void UpdateUI()

{

"")

//更新标题

Text = (\_fileName != null && \_fileName.Length > 0)

? string.Format("{0} - [{1}] {2}", \_appName, \_fileName, \_dirty? "\*":

: \_appName;

//按下/释放表示设计或者预览模式的按钮

bool design = \_c1rd.Visible && \_c1rd.Report != null;

\_btnDesign.Pushed = design;

\_btnPreview.Pushed = !design;

//启用/禁用按钮

\_btnCut.Enabled = design && \_c1rd.ClipboardHandler.CanCut;

\_btnCopy.Enabled = design && \_c1rd.ClipboardHandler.CanCut;

\_btnPaste.Enabled = design && \_c1rd.ClipboardHandler.CanPaste;

\_btnUndo.Enabled = design && \_c1rd.UndoStack.CanUndo;

\_btnRedo.Enabled = design && \_c1rd.UndoStack.CanRedo;

bool reportSelected = design && \_list.SelectedItem != null;

\_btnAddReport.Enabled = \_c1rd.Visible;

\_btnDelReport.Enabled = reportSelected;

\_btnAddField.Enabled = reportSelected;

\_btnAddLabel.Enabled = reportSelected;

}

}

请注意在这里**UpdateUI**方法是如何通过使用 CanCut，CanP Aste，CanUndo，以及CanRedo属性启用或禁用工具栏按钮的。

## 步骤四：添加代码以处理工具栏命令

为了处理工具栏按钮的单击事件，并将它们派发到相应的处理程序，请使用以下代码：

* VisualB Asic

'在工具栏按钮上处理单击事件

Private Sub \_tb\_ButtonClick(ByVal sender As System.Object, ByVale As

System.Windows.Forms.ToolBarButtonClickEventArgs) Handles \_tb.ButtonClick

'设计/预览模式

If e.Button.Equals(\_btnDesign) Then

SetDesignMode(True)

End If

If e.Button.Equals(\_btnPreview) Then

SetDesignMode(False)

'文件命令

If e.Button.Equals(\_btnNew) Then

NewFile()

If e.Button.Equals(\_btnOpen) Then OpenFile()

If e.Button.Equals(\_btnSave) Then SaveFile()

'允许用户撤销剪贴板操作

If e.Button.Equals(\_btnCut) Or e.Button.Equals(\_btnP Aste) Then

\_c1rd.UndoStack.SaveState()

End If

'剪贴板

If e.Button.Equals(\_btnCut) Then

\_c1rd.ClipboardHandler.Cut()

If e.Button.Equals(\_btnCopy) Then

\_c1rd.ClipboardHandler.Copy()

If e.Button.Equals(\_btnP Aste) Then

\_c1rd.ClipboardHandler.P Aste()

'撤消/重做

If e.Button.Equals(\_btnUndo) Then

\_c1rd.UndoStack.Undo()

If e.Button.Equals(\_btnRedo) Then

\_c1rd.UndoStack.Redo()

'添加/删除报表

If e.Button.Equals(\_btnAddReport) Then

NewReport()

If e.Button.Equals(\_btnDelReport) Then

DeleteReport()

'添加字段

'（只需要设置创建信息并等待来自设计器的CreateField事件）

If e.Button.Equals(\_btnAddField) Then

\_c1rd.CreateFieldInfo= e.Button

End If

If e.Button.Equals(\_btnAddLabel) Then

\_c1rd.CreateFieldInfo= e.Button

End If

End Sub

* C#

//在工具栏按钮上处理单击事件

private void \_tb\_ButtonClick(objectsender,

System.Windows.Forms.ToolBarButtonClickEventArgse)

{

//设计/预览模式

If (e.Button== \_btnDesign) SetDesignMode(true);

If (e.Button== \_btnPreview) SetDesignMode(false);

//文件命令

If (e.Button== \_btnNew) NewFile();

If (e.Button== \_btnOpen) OpenFile();

If (e.Button== \_btnSave) SaveFile();

//允许用户撤销剪贴板操作

If (e.Button== \_btnCut|| e.Button== \_btnP Aste)

\_c1rd.UndoStack.SaveState();

//剪贴板

If (e.Button== \_btnCut) \_c1rd.ClipboardHandler.Cut();

If (e.Button== \_btnCopy) \_c1rd.ClipboardHandler.Copy();

If (e.Button== \_btnP Aste) \_c1rd.ClipboardHandler.P Aste();

//撤销/重做

If (e.Button== \_btnUndo) \_c1rd.UndoStack.Undo();

If (e.Button== \_btnRedo) \_c1rd.UndoStack.Redo();

//添加/删除报表

If (e.Button== \_btnAddReport) NewReport();

If (e.Button== \_btnDelReport) DeleteReport();

//添加字段

//（只需要设置创建信息并等待来自设计器的CreateField事件）

If (e.Button== \_btnAddField) \_c1rd.CreateFieldInfo= e.Button;

If (e.Button== \_btnAddLabel) \_c1rd.CreateFieldInfo= e.Button;

}

这个程序将大约一半的命令分配给专门的处理程序。这些将在后面介绍。另一半（剪贴板，撤消/重做）是由C1ReportDesigner 控件直接处理。

请注意，在调用剪切和粘贴的方法之前，该代码调用SaveState 方法保存报表的当前状态。这允许用户撤销和重做剪贴板操作。在一般情况下，您的代码应该在变更的报表之前调用SaveState。

## 步骤五：实现SetDesignMode 方法

该简单的设计器有两种模式：报表设计模式和预览模式。当用户选择了一个新的报表或点击工具栏上的Design按钮时，该应用程序显示设计器控件。当用户单击“Preview ”按钮时，应用程序将当前报表呈现到预览控件并显示结果。

添加以下代码以实现**SetDesignMode**方法：

* VisualB Asic

Private Sub SetDesignMode(ByValdesign As Boolean)

'显示/隐藏预览/设计面板

\_c1rd.Visible = design

\_c1ppv.Visible= Not design

'在预览模式不显示属性

If Not design Then

\_lblPropGrid.Text= "Properties"

\_ppg.SelectedObject= Nothing

End If

'将报表的一个副本关联到预览控件，

'（因此脚本所引发的变更不会保存）

If Not designThen

\_c1ppv.Document= Nothing

\_c1r.CopyFrom(\_c1rd.Report)

Cursor= Cursors.WaitCursor

\_c1r.Render()

If \_c1r.PageImages.Count>0 Then

\_c1ppv.Document= \_c1r.Document

End If

Cursor= Cursors.Default End If

'完成，更新UI

UpdateUI()

End Sub

* C#

private void SetDesignMode(booldesign)

{

//显示/隐藏预览/设计面板

\_c1rd.Visible = design;

\_c1ppv.Visible= !design;

//在预览模式不显示属性

If (!design)

{

\_lblPropGrid.Text= "Properties";

\_ppg.SelectedObject= null;

}

//将报表的一个副本关联到预览控件，

//（因此脚本所引发的变更不会保存）

If (!design)

{

\_c1ppv.Document= null;

\_c1r.CopyFrom(\_c1rd.Report);

Cursor= Cursors.WaitCursor;

\_c1r.Render();

If (\_c1r.PageImages.Count>0)

\_c1ppv.Document= \_c1r.Document; Cursor= Cursors.Default;

}

//完成，更新UI

UpdateUI();

}

切换到设计模式非常容易，您所要做的就是显示设计器并隐藏预览控件。切换到预览模式则稍微有点复杂，因为它还需要呈现报表。

请注意，在开始呈现之前，该报表被复制到一个独立的**C1Report**组件。这是必要的操作，因为报表本身可能包含脚本代码，这可能会修改报表定义（字段颜色，可见性，等等），我们不想让这些更改应用到报表定义。

## 步骤六：实现文件操作方法

该简单的设计器具有三个支持文件操作的命令：**New**，**Open**，以及**Save**。NewFile将清除类的变量，报表列表，预览以及设计器控件，并更新界面。

将下面的代码添加到**NewFile**的方法实现：

* Visual Basic

Private Sub NewFile()

\_fileName= ""

\_dirty= False

\_list.Items.Clear()

\_c1ppv.Document= Nothing

\_c1rd.Report= Nothing UpdateUI()

End Sub

* C#

private void NewFile()

{

\_fileName= "";

\_dirty= false;

\_list.Items.Clear();

\_c1ppv.Document= null;

\_c1rd.Report= null;

UpdateUI();

}

OpenFile提示用户选择打开一个报表定义文件，并使用C1Report组件获取选中的文件中的报表名称列表。每个报表加载到一个新的C1Report控件，并添加到报表列表（\_list控件）。

该代码使用了一个ReportHolder 的封装类，而不是直接地将C1Report 组件添加到列表框中。ReportHolder 类的唯一功能是重写了ToString方法，因此列表框可以显示报表名称。

将下面的代码添加到OpenFile的方法实现：

* Visual Basic

Public Sub OpenFile()

'获取打开文件的名称

Dim dlg As New OpenFileDialog

dlg.FileName = "\*.xml"

dlg.Title = "Open Report Definition File" If dlg.ShowDialog() <> DialogResult.OK Then

Return

End If

'检查所选的文件

Try

reports = \_c1r.GetReportInfo(dlg.FileName)

Catch

If IsNothing(reports) OrElse reports.Length = 0 Then

MessageBox.Show("Invalid (or empty) report definition file") Return

End If End Try

'清除列表

NewFile()

'加载新的文件

Cursor = Cursors.WaitCursor

\_fileName = dlg.FileName

Dim reportName As String

For Each reportName In reports

Dim rpt As New C1Report()

rpt.Load(\_fileName, reportName)

\_list.Items.Add(New ReportHolder(rpt))

Next

Cursor = Cursors.Default

'选择第一个报表

\_list.SelectedIndex = 0 End Sub

' ReportHolder

'用作在列表框中保存报表的辅助类。

'它所做的主要的事情就是复写了ToString()方法，以呈现报表名称Public Class ReportHolder

Public Sub New(ByVal report As C1Report)

Me.Report = report

End Sub

Public Overrides Function ToString() As String Dim text As String = Me.Report.ReportName

If text.Length = 0 Then text = "Unnamed Report" Return text

End Function

Public ReadOnly Report As C1Report End Class

* C#

public void OpenFile()

{

//获取打开文件的名称

OpenFileDialog dlg = new OpenFileDialog();

dlg.FileName = "\*.xml";

dlg.Title = "Open Report Definition File";

if (dlg.ShowDialog() != DialogResult.OK) return;

//检查选中的文件

string[] reports = null; try

{

reports = \_c1r.GetReportInfo(dlg.FileName);

}

catch {}

if (reports == null || reports.Length == 0)

{

MessageBox.Show("Invalid (or empty) report definition file");

return;

}

//清除列表

NewFile();

// 加载新文件

Cursor = Cursors.WaitCursor;

\_fileName = dlg.FileName;

foreach (string reportName in reports)

{

C1Report rpt = new C1Report();

rpt.Load(\_fileName, reportName);

\_list.Items.Add(new ReportHolder(rpt));

}

Cursor = Cursors.Default;

// 选择第一个report

\_list.SelectedIndex = 0;

}

// ReportHolder

//用作在列表框中保存报表的辅助类。

//它所做的主要的事情就是复写了ToString()方法，以呈现报表名称。

public class ReportHolder

{

public readonly C1Report Report;

public ReportHolder(C1Report report)

{

Report = report;

}

override public string ToString()

{

string s = Report.ReportName;

return (s != null && s.Length > 0)? s: "Unnamed Report";

}

}

最后，SaveFile方法提示用户选择一个文件名，并使用一个XmlWriter将每一个报表通过C1Report.Save方法保存到一个新文件。将下面的代码添加到SaveFile 的方法实现：

* Visual Basic

Public Sub SaveFile()

'获取打开文件的名称

Dim dlg As New SaveFileDialog()

dlg.FileName = \_fileName

dlg.Title = "Save Report Definition File"

If dlg.ShowDialog() <> Windows.Forms.DialogResult.OK Then Return

' 保存文件

Dim w As New XmlTextWriter(dlg.FileName, System.Text.Encoding.Default) w.Formatting = Formatting.Indented

w.Indentation = 2 w.WriteStartDocument()

' 写入所有报表

Cursor = Cursors.WaitCursor w.WriteStartElement("Reports") Dim rh As ReportHolder

For Each rh In \_list.Items rh.Report.Save(w) 'rh.Report.ReportName

Next

w.WriteEndElement()

Cursor = Cursors.Default

' 关闭文件

w.Close()

' 完成

\_fileName = dlg.FileName

\_dirty = False UpdateUI()

End Sub

* C#

public void SaveFile()

{

//获取要保存的文件名

SaveFileDialog dlg = new SaveFileDialog(); dlg.FileName = \_fileName;

dlg.Title = "Save Report Definition File"; if (dlg.ShowDialog() != DialogResult.OK)

return;

//保存文件

XmlTextWriter w = new XmlTextWriter(dlg.FileName,

System.Text.Encoding.Default);

w.Formatting = Formatting.Indented;

w.Indentation = 2; w.WriteStartDocument();

//将全部的报表写入文件

Cursor = Cursors.WaitCursor; w.WriteStartElement("Reports");

foreach (ReportHolder rh in \_list.Items)

rh.Report.Save(w); //rh.Report.ReportName;

w.WriteEndElement();

Cursor = Cursors.Default;

// 关闭文件

w.Close();

// 完成

\_fileName = dlg.FileName;

\_dirty = false;

UpdateUI();

}

## 步骤七：连接控件

下一步是添加事件处理程序，将所有的控件连接在一起。

这里是\_list 控件的SelectedIndexChanged事件的处理程序。添加以下代码，当用户从列表中选择一个新的报表，该代码在设计模式显示它：

* Visual Basic

'选中了一个新报表：切换到设计模式并显示

Private Sub \_list\_SelectedIndexChanged(ByVal sender As System.Object,

ByVal e As System.EventArgs) Handles \_list.SelectedIndexChanged

'切换到设计模式

SetDesignMode(True)

'将选中的报表关联到设计器以及预览控件

\_c1rd.Report = Nothing

\_c1ppv.Document = Nothing

If \_list.SelectedIndex > -1 Then

\_c1rd.Report = \_list.SelectedItem.Report

End If End Sub

* C#

//选中了一个新报表：切换到设计模式并显示

private void \_list\_SelectedIndexChanged(object sender, System.EventArgs e)

{

// 切换到设计模式

SetDesignMode(true);

//将选中的报表关联到设计器以及预览控件

\_c1rd.Report = null;

\_c1ppv.Document = null;

if (\_list.SelectedItem != null)

\_c1rd.Report = ((ReportHolder)\_list.SelectedItem).Report;

}

设计器使用一个PropertyGrid控件（\_ppg）将在设计器中选中的元素的属性暴露出来。这通过设置该PropertyGrid控件的SelectedObject属性完成；做为反馈，该控件触发一个SelectionChanged 事件。

当用户在设计器控件中选中了一个报表字段或者区域时，它将触发SelectionChanged事件。该事件的处理程序检查新的选择元素并将其设置给PropertyGrid控件。这是一个强大的机制。选中的元素可以是单个报表字段，一个字段分组，一个区域，或者整个报表。

将下面的代码添加到SelectionChanged 的方法实现：

* Visual Basic

'选择发生变化，需要更新PropertyGrid并显示选中对象的属性

Private Sub \_c1rd\_SelectionChanged(ByVal sender As Object, ByVal e As

System.EventArgs) Handles \_c1rd.SelectionChanged

Dim sel As Object() = \_c1rd.SelectedFields

If (sel.Length > 0) Then

\_lblPropGrid.Text = "Field Properties"

\_ppg.SelectedObjects = sel

ElseIf Not IsNothing(\_c1rd.SelectedSection) Then

\_lblPropGrid.Text = "Section Properties"

\_ppg.SelectedObject = \_c1rd.SelectedSection

ElseIf Not IsNothing(\_c1rd.Report) Then

\_lblPropGrid.Text = "Report Properties"

\_ppg.SelectedObject = \_c1rd.Report

'没有对象选中

Else

\_lblPropGrid.Text = "Properties"

\_ppg.SelectedObject = Nothing

End If

'完成

UpdateUI()

End Sub

* C#

//选择发生变化，需要更新PropertyGrid并显示选中对象的属性

private void \_c1rd\_SelectionChanged(object sender, System.EventArgs e)

{

object[] sel = \_c1rd.SelectedFields;

if (sel.Length > 0)

{

\_lblPropGrid.Text = "Field Properties";

\_ppg.SelectedObjects = sel;

}

else if (\_c1rd.SelectedSection != null)

{

\_lblPropGrid.Text = "Section Properties";

\_ppg.SelectedObject = \_c1rd.SelectedSection;

}

else if (\_c1rd.Report != null)

{

\_lblPropGrid.Text = "Report Properties";

\_ppg.SelectedObject = \_c1rd.Report;

}

else // 没有对象选中

{

\_lblPropGrid.Text = "Properties";

\_ppg.SelectedObject = null;

}

// 完成

UpdateUI();

}

PropertyGrid控件（\_ppg）显示在设计器（\_c1rd）中间选中的对象的属性。当用户在PropertyGrid控件中修改了一个对象的属性时，应当通知设计器使其可以更新显示。相反，当用户通过设计器编辑一个对象时，PropertyGrid也需要收到通知更新其显示的内容。

将以下代码添加到\_ppg控件的PropertyValueChanged事件以及\_c1rd控件的ValuesChanged事件的处理函数的实现：

* Visual Basic

 '当一个属性通过属性窗体发生变化时，刷新设计器以显示变化

Private Sub \_ppg\_PropertyValueChanged(ByVal s As Object, ByVal e As

System.Windows.Forms.PropertyValueChangedEventArgs) Handles

\_ppg.PropertyValueChanged

\_c1rd.Refresh()

\_dirty = True

UpdateUI()

End Sub

'当选中对象的属性在设计器中发生变化时，更新属性窗体以反映该变化

Private Sub \_c1rd\_ValuesChanged(ByVal sender As Object, ByVal e As

System.EventArgs) Handles \_c1rd.ValuesChanged

\_c1rd.Refresh()

\_dirty = True UpdateUI()

End Sub

* C#

 //当一个属性通过属性窗体发生变化时，刷新设计器以显示变化

private void \_ppg\_PropertyValueChanged(object s,

Systems.Windows.Forms.PropertyValueChangedEventArgs e)

{

\_c1rd.Refresh();

\_dirty = true; UpdateUI();

}

'当选中对象的属性在设计器中发生变化时，更新属性窗体以反映该变化

private void \_c1rd\_ValuesChanged(object sender, System.EventArgs e)

{

\_ppg.Refresh();

\_dirty = true;

UpdateUI();

}

## 步骤八：添加代码以创建或删除报告

通过使用DeleteReport 方法从列表中移除报表。该DeleteReport 方法简单地从报表列表移除选中的项目，从设计器控件清除Report属性，接下来如果该列表不为空，则进行新的选择。

添加以下代码以通过使用DeleteReport 方法移除报表：

* Visual Basic

' 从列表中移除当前报表

Private Sub DeleteReport()

' a report must be selected

Dim index As Integer = \_list.SelectedIndex If (index < 0) Then Return

' 从设计器以及列表中移除报表

\_c1rd.Report = Nothing

\_list.Items.RemoveAt(index)

' 如果可以，选择另一个报表

If (index > \_list.Items.Count – 1) Then

index = \_list.Items.Count - 1

If (index > - 1) Then

\_list.SelectedIndex = index

End If

End If

' 完成

\_dirty = True UpdateUI()

End Sub

* C#

// 从列表中移除当前报表

private void DeleteReport()

{

// a report must be selected

int index = \_list.SelectedIndex;

if (index < 0) return;

// 从设计器以及列表中移除报表

\_c1rd.Report = null;

\_list.Items.RemoveAt(index);

// 如果可以，选择另一个报表

if (index > \_list.Items.Count-1)

index = \_list.Items.Count-1; if (index > -1)

\_list.SelectedIndex = index;

// 完成

\_dirty = true;

UpdateUI();

}

AddReport 则稍微复杂一点。在完整版本的报表设计器中，该命令调用一个向导，允许用户选择一个数据源，决定分组选项，布局以及样式。当实现您自己的设计器时，您可以原样使用该项带代码，或者自定义以满足需求。

和仅创建一个空白的新报表不同，该简单设计器将提示用户选择一个MDB文件，从中选择可以找到的第一个表，接下来选中前五个字段，并基于以上数据创建一个报表。

添加以下代码以通过AddReport方法创建一个报表：

* Visual Basic

Private Sub NewReport()

' 选择数据源（在这个示例只允许MDB文件）

Dim dlg As New OpenFileDialog() dlg.FileName = "\*.mdb"

dlg.Title = "Select report data source"

If dlg.ShowDialog() <> Windows.Forms.DialogResult.OK Then Return

' 选择数据源中第一张表

Dim connString As String =

String.Format("Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0}",

dlg.FileName)

Dim tableName As String = GetFirstTable(connString)

If tableName.Length = 0 Then

MessageBox.Show("Failed to retrieve data from the selected source.") Return

End If

' 新建报表

Dim rpt As New C1Report() rpt.ReportName = tableName

' 设置数据源

rpt.DataSource.ConnectionString = connString rpt.DataSource.RecordSource = tableName

' 添加title字段

Dim s As Section = rpt.Sections(SectionTypeEnum.Header)

s.Visible = True

s.Height = 600

Dim f As Field = s.Fields.Add("TitleField", tableName, 0, 0, 4000, 600) f.Font.Bold = True

f.Font.Size = 24 f.ForeColor = Color.Navy

' 添加最多五个字段

Dim fieldNames As String() = rpt.DataSource.GetDBFieldList(True)

Dim cnt As Integer = Math.Min(5, fieldNames.Length)

' 添加页眉

s = rpt.Sections(SectionTypeEnum.PageHeader)

s.Visible = True

s.Height = 400

Dim rc As New Rectangle(0, 0, 1000, s.Height)

Dim i As Integer

For i = 0 To cnt - 1

f = s.Fields.Add("TitleField", fieldNames(i), rc)

f.Font.Bold = True rc.Offset(rc.Width, 0)

Next

' 添加detail区域

s = rpt.Sections(SectionTypeEnum.Detail) s.Visible = True

s.Height = 300

rc = New Rectangle(0, 0, 1000, s.Height)

For i = 0 To cnt - 1

f = s.Fields.Add("TitleField", fieldNames(i), rc) f.Calculated = True

rc.Offset(rc.Width, 0) Next

' 向列表添加新报表然后选中它

\_list.Items.Add(New ReportHolder(rpt))

\_list.SelectedIndex = \_list.Items.Count - 1

' 完成

\_dirty = True UpdateUI()

End Sub

* C#

private void NewReport()

{

// 选择数据源（在这个示例只允许MDB文件）

OpenFileDialog dlg = new OpenFileDialog();

dlg.FileName = "\*.mdb";

dlg.Title = "Select report data source";

if (dlg.ShowDialog() != DialogResult.OK) return;

// 选择数据源中第一张表

string connString =

string.Format(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source={0};",

dlg.FileName);

string tableName = GetFirstTable(connString);

if (tableName == null || tableName.Length == 0)

{

source.");

}

MessageBox.Show("Failed to retrieve data from the selected

return;

// 新建报表

C1Report rpt = new C1Report();

rpt.ReportName = tableName;

// 设置数据源rpt.DataSource.ConnectionString = connString; rpt.DataSource.RecordSource = tableName;

// 添加title字段

Section s = rpt.Sections[SectionTypeEnum.Header];

s.Visible = true;

s.Height = 600;

Field f = s.Fields.Add("TitleField", tableName, 0, 0, 4000, 600); f.Font.Bold = true;

f.Font.Size = 24; f.ForeColor = Color.Navy;

// 添加最多五个字段

string[] fieldNames = rpt.DataSource.GetDBFieldList(true);

int cnt = Math.Min(5, fieldNames.Length);

// 添加页眉

s = rpt.Sections[SectionTypeEnum.PageHeader];

s.Visible = true;

s.Height = 400;

Rectangle rc = new Rectangle(0, 0, 1000, (int)s.Height);

for (int i = 0; i < cnt; i++)

{

f = s.Fields.Add("TitleField", fieldNames[i], rc);

f.Font.Bold = true;

rc.Offset(rc.Width, 0);

}

// 添加detail区域

s = rpt.Sections[SectionTypeEnum.Detail];

s.Visible = true;

s.Height = 300;

rc = new Rectangle(0, 0, 1000, (int)s.Height);

for (int i = 0; i < cnt; i++)

{

f = s.Fields.Add("TitleField", fieldNames[i], rc);

f.Calculated = true; rc.Offset(rc.Width, 0);

}

// 向列表添加新报表然后选中它

\_list.Items.Add(new ReportHolder(rpt));

\_list.SelectedIndex = \_list.Items.Count-1;

// 完成

\_dirty = true;

UpdateUI();

}

以下代码使用一个辅助函数GetFirstTable打开一个连接，获取数据库架构，并返回其搜索到的第一个表。添加以下代码：

* Visual Basic

Private Function GetFirstTable(connString As String) As String Dim conn As New OleDbConnection(connString)

Try

' 获取数据库架构

conn.Open()

Dim dt As DataTable =

conn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, Nothing)

Dim dr As DataRow

Then

For Each dr In dt.Rows

' 检查表类型

Dim type As String = dr("TABLE\_TYPE").ToString().ToUpper()

If (type <> "TABLE" AndAlso type <> "VIEW" AndAlso type <> "LINK"

'跳过这个

Else

' 获取表名

tableName = dr("TABLE\_NAME").ToString() Exit For

End If Next

' 完成

conn.Close()

Catch

End Try

' 返回找到的第一张表

Return tableName End Function

* C#

private string GetFirstTable(string connString)

{

string tableName = null;

OleDbConnection conn = new OleDbConnection(connString);

try

{

// 获取数据库架构

conn.Open();

null);

DataTable dt = conn.GetOleDbSchemaTable(OleDbSchemaGuid.Tables,

foreach (DataRow dr in dt.Rows)

{

// 检查表类型

string type = dr["TABLE\_TYPE"].ToString().ToUpper();

if (type != "TABLE" && type != "VIEW" && type != "LINK")

continue;

// 获取表名

tableName = dr["TABLE\_NAME"].ToString();

break;

}

// 完成

conn.Close();

}

catch {}

// 返回找到的第一张表

return tableName;

}

## 步骤九：添加代码以创建字段

这个简单的设计器接近完成；目前唯一缺少的部分是用来在报表中创建新字段的代码。

查看我们在工具栏事件处理程序写的代码，您会发现它设置了设计器的CreateFieldInfo 属性，表示将等待处理设计器的CreateField事件。

添加以下代码，以便在报表中创建新的字段：

* Visual Basic

' 在工具栏按钮上处理单击事件

Private Sub \_tb\_ButtonClick(ByVal sender As System.Object, ByVal e As

System.Windows.Forms.ToolBarButtonClickEventArgs) Handles \_tb.ButtonClick

' …

' 添加字段

' （仅设置创建信息，并等待设计器的CreateField事件）

If e.Button.Equals(\_btnAddField) Then

\_c1rd.CreateFieldInfo = e.Button If e.Button.Equals(\_btnAddLabel) Then

\_c1rd.CreateFieldInfo = e.Button

End Sub

* C#

// 在工具栏按钮上处理单击事件

private void \_tb\_ButtonClick(object sender,

System.Windows.Forms.ToolBarButtonClickEventArgs e)

{

// …

// 添加字段

// （仅设置创建信息，并等待设计器的CreateField事件）

if (e.Button == \_btnAddField) \_c1rd.CreateFieldInfo = e.Button;

if (e.Button == \_btnAddLabel) \_c1rd.CreateFieldInfo = e.Button;

}

CreateFieldInfo属性可以设置为任意的非空对象，以通知设计器您希望创建一个新的字段。设计器不知道您想要什么类型的字段或你想如何初始化它，所以它跟踪鼠标，允许用户在一个区域内绘制出字段的边框位置。它接着触发CreateField事件，并传递给您所需要的创建字段的信息。

将下面的代码添加到事件处理函数以处理CreateField事件：

* Visual Basic

Dim \_ctr As Integer

Private Sub \_c1rd\_CreateField(ByVal sender As Object, ByVal e As C1.Win.C1ReportDesigner.CreateFieldEventArgs) Handles \_c1rd.CreateField

' 保存撤销信息

\_c1rd.UndoStack.SaveState()

' 添加label字段

\_ctr = \_ctr + 1

Dim fieldName As String = String.Format("NewField{0}", \_ctr)

Dim fieldText As String = fieldName

Dim f As Field = e.Section.Fields.Add(fieldName, fieldText,

e.FieldBounds)

' 如果这是一个计算字段,

' 改变Text和Calculated属性

If e.CreateFieldInfo.Equals(\_btnAddField) Then

Dim fieldNames As String() =

\_c1rd.Report.DataSource.GetDBFieldList(True)

If (fieldNames.Length > 0) Then

f.Text = fieldNames(0) f.Calculated = True

End If End If

End Sub

* C#

int \_ctr = 0;

private void \_c1rd\_CreateField(object sender, C1.Win.C1ReportDesigner.CreateFieldEventArgs e)

{

// 保存撤销信息

\_c1rd.UndoStack.SaveState();

// 添加label字段

string fieldName = string.Format("NewField{0}", ++\_ctr);

string fieldText = fieldName;

Field f = e.Section.Fields.Add(fieldName, fieldText, e.FieldBounds);

// 如果这是一个计算字段,

// 改变Text和Calculated属性

if (e.CreateFieldInfo == \_btnAddField)

{

string[] fieldNames =

\_c1rd.Report.DataSource.GetDBFieldList(true);

if (fieldNames.Length > 0)

{

f.Text = fieldNames[0];

f.Calculated = true;

}

}

}

注意代码如何在开始位置调用设计器的SaveState方法，因此用户可以撤销创建字段行为。在此之后，字段创建完成，CreateFieldInfo参数用作自定义新的字段，并且使其表现为一个标签或者计算字段。

总结简单设计器程序：一个对于如何操作C1ReportDesigner控件的介绍。