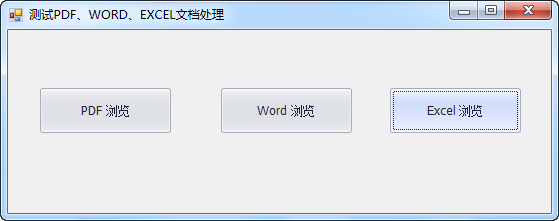
**[基于DevExpress实现对PDF、Word、Excel文档的预览及操作处理](http://www.cnblogs.com/wuhuacong/p/4175266.html)**

在一般的管理系统模块里面，越来越多的设计到一些常用文档的上传保存操作，其中如PDF、Word、Excel等文档，有时候是通过分布式的WCF技术实现数据的显示和处理，因此希望直接预览而不需要下载文件，这样能够给我们提供很多的方便。在DevExpress里面，提供了相应的控件来显示和处理这些文档，本文主要介绍如何利用DevExpress的控件实现对PDF、Word、Excel文档的预览和操作处理。

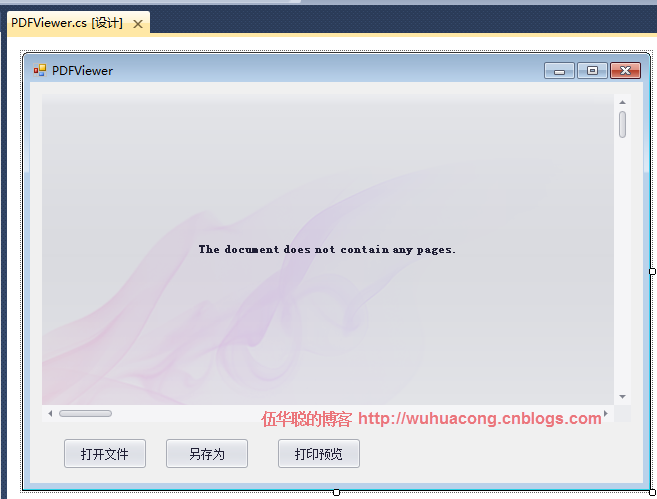
**1、PDF的预览和操作**

在较早的DevExpress的控件里面，已经提供了对应的PDF文档的显示控件，不过由于其对PDF格式支持不是很好，有些文档是Office导出的，也不是很正常阅读，因此很少使用，本文介绍的DevExpress的PDF查看控件是基于14.1的，测试过很多文档，好像都能正常打开，因此也想在系统中广泛使用了。

为了演示这些控件的处理，我单独编写了一个例子，用来实现对PDF、Word、Excel等文档的处理。



为了显示PDF文档，我们需要在界面里面添加一个XtraPdfViewer.PdfViewer的控件，这个主要是用来显示PDF的，它有很多属性方法，用来实现对PDF的处理操作，测试界面设计好如下所示。



对PDF，我们一般主要是用来打开文件，另存为，或者预览就可以了。相关的操作代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// PDF测试显示窗体

/// </summary>

public partial class PDFViewer : Form

{

//记录窗体的名称

readonly string mainFormText;

public PDFViewer()

{

InitializeComponent();

//记录窗体的名称，并实现文档变化事件的处理，方便显示新的文件名称

mainFormText = this.Text;

pdfViewer1.DocumentChanged += new DevExpress.XtraPdfViewer.PdfDocumentChangedEventHandler(pdfViewer1\_DocumentChanged);

}

/// <summary>

/// PDF文档变化后，实现对新文件名称的显示

/// </summary>

void pdfViewer1\_DocumentChanged(object sender, DevExpress.XtraPdfViewer.PdfDocumentChangedEventArgs e)

{

string fileName = Path.GetFileName(e.DocumentFilePath);

if (String.IsNullOrEmpty(fileName))

{

Text = mainFormText;

}

else

{

Text = fileName + " - " + mainFormText;

}

}

/// <summary>

/// 打开PDF文件

/// </summary>

private void btnOpenFile\_Click(object sender, EventArgs e)

{

string filePath = FileDialogHelper.OpenPdf();

if (!string.IsNullOrEmpty(filePath))

{

this.pdfViewer1.LoadDocument(filePath);

}

}

/// <summary>

/// 另存为PDF文件

/// </summary>

private void btnSaveAs\_Click(object sender, EventArgs e)

{

string dir = System.Environment.CurrentDirectory;

string filePath = FileDialogHelper.SavePdf("", dir);

if (!string.IsNullOrEmpty(filePath))

{

try

{

this.pdfViewer1.SaveDocument(filePath);

MessageUtil.ShowTips("保存成功");

}

catch (Exception ex)

{

LogTextHelper.Error(ex);

MessageUtil.ShowError(ex.Message);

}

}

}

/// <summary>

/// PDF文件打印

/// </summary>

private void btnPreview\_Click(object sender, EventArgs e)

{

this.pdfViewer1.Print();

}

}

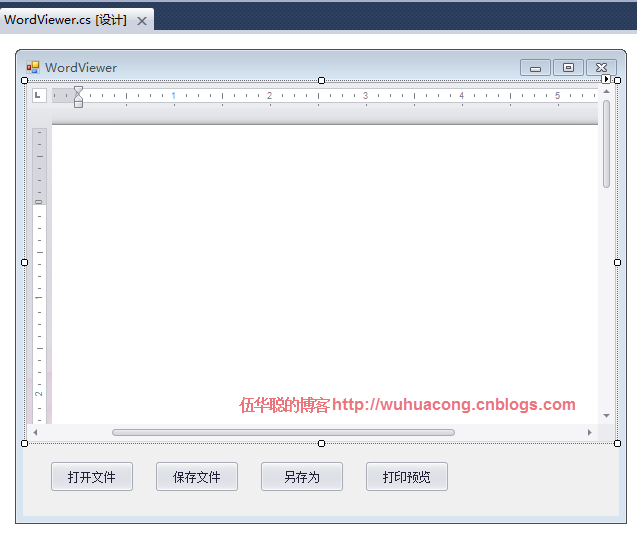
[复制代码](javascript:void(0);)

从上面的代码，我们可以看到，对于PDF，我们操作起来很方便，主要就是在界面里面加载文件后，就可以对PDFViewer对象实现相关的操作了。

**2、WORD文档的预览和操作**

类似于PDF文档，我们对WORD文档，也是通过使用RichEditControl实现文档的显示，不过和PDFViewer不同，这个控件可以实现对文档的修改和保存操作，这种对于我们提供用户对文档进行编辑很方便。

测试例子的界面如下所示。



相关的操作代码，也和PDF的操作类似，不同的是，它在文档变化后，不能很容易从参数里面获取到对应的文档的路径，需要特殊的处理才能得到。

[复制代码](javascript:void(0);)

/// <summary>

/// WORD控件的测试例子

/// </summary>

public partial class WordViewer : Form

{

//记录窗体的名称

readonly string mainFormText;

public WordViewer()

{

InitializeComponent();

//记录窗体的名称，并实现文档变化事件的处理，方便显示新的文件名称

mainFormText = this.Text;

this.richEditControl1.DocumentLoaded += new EventHandler(richEditControl1\_DocumentLoaded);

}

/// <summary>

/// WORD文档变化后，实现对新文件名称的显示

/// </summary>

void richEditControl1\_DocumentLoaded(object sender, EventArgs e)

{

string fileName = Path.GetFileName(this.richEditControl1.Options.DocumentSaveOptions.CurrentFileName);

if (String.IsNullOrEmpty(fileName))

{

Text = mainFormText;

}

else

{

Text = fileName + " - " + mainFormText;

}

//修改默认字体

DocumentRange range = richEditControl1.Document.Range;

CharacterProperties cp = this.richEditControl1.Document.BeginUpdateCharacters(range);

cp.FontName = "新宋体";

//cp.FontSize = 12;

this.richEditControl1.Document.EndUpdateCharacters(cp);

}

/// <summary>

/// 打开WORD文件

/// </summary>

private void btnOpenFile\_Click(object sender, EventArgs e)

{

string filePath = FileDialogHelper.OpenWord();

if (!string.IsNullOrEmpty(filePath))

{

richEditControl1.LoadDocument(filePath);//, DocumentFormat.Doc);

}

}

/// <summary>

/// 保存WORD文件

/// </summary>

private void btnSaveFile\_Click(object sender, EventArgs e)

{

this.richEditControl1.SaveDocument();

}

/// <summary>

/// 另存为WORD文件

/// </summary>

private void btnSaveAs\_Click(object sender, EventArgs e)

{

try

{

richEditControl1.SaveDocumentAs();

MessageUtil.ShowTips("保存成功");

}

catch (Exception ex)

{

LogTextHelper.Error(ex);

MessageUtil.ShowError(ex.Message);

}

}

/// <summary>

/// WORD文件打印

/// </summary>

private void btnPreview\_Click(object sender, EventArgs e)

{

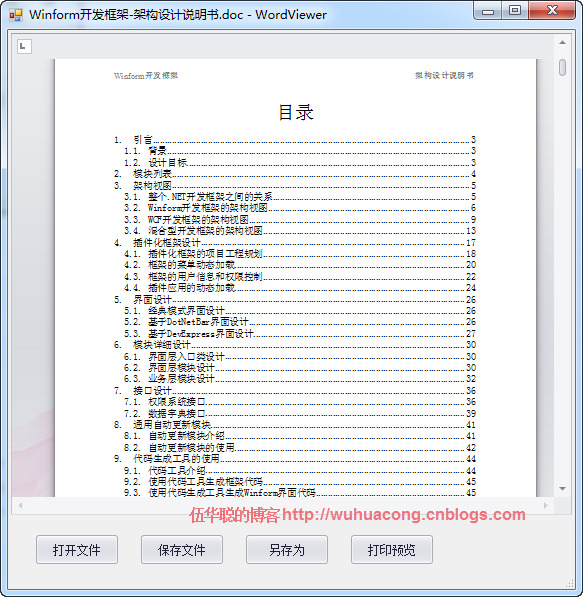
this.richEditControl1.ShowPrintPreview();

}

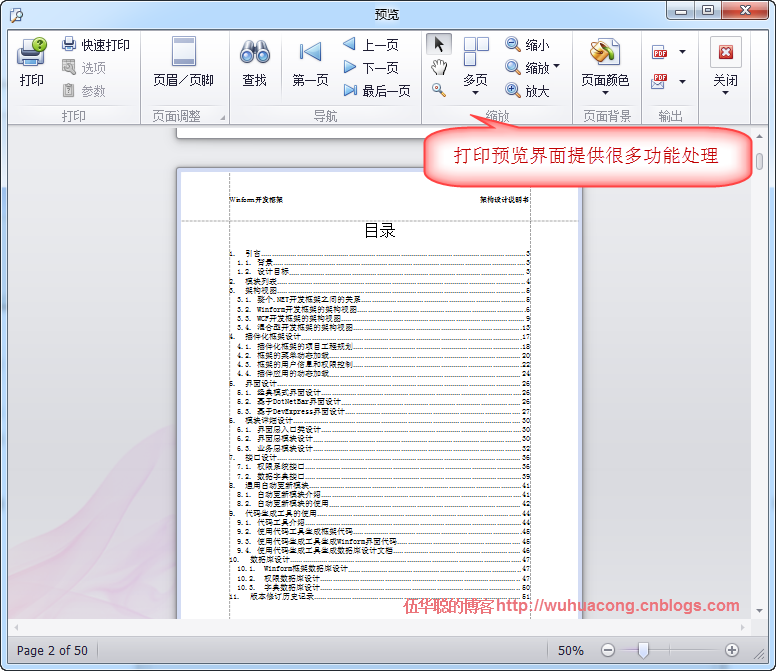
}

[复制代码](javascript:void(0);)

加载文档后，界面显示内容如下所示：



文档控件很容易支持打印预览功能，打印预览的界面如下所示

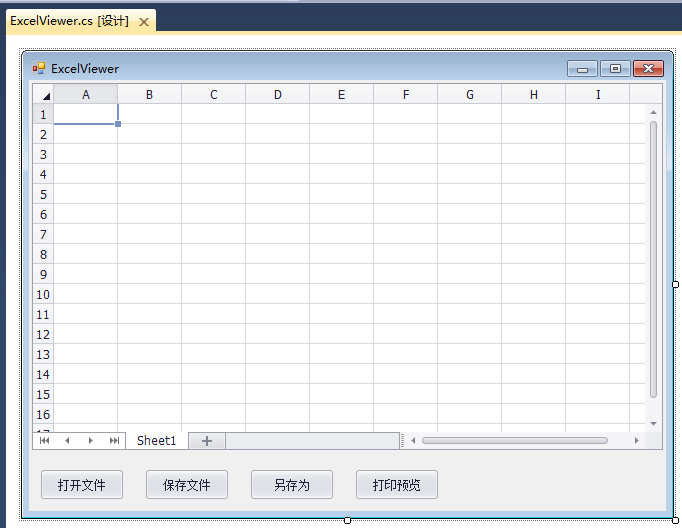


不过话说回来，这个RichEditControl虽然能够较好显示Word文档，不过也有一些字体显示的不是很好，格式和微软的Word软件显示的有些不太一样，格式有所损失。

因此如果对于格式要求比较严谨的，建议还是只是做显示为佳，不要保存原有的文档。如果对格式不是特别严格，倒是可以作为一个文档服务器实现文档的新建、保存处理。

**3、Excel文档的预览和操作**

对于Excel文档的预览和操作，DevExpress控件在最近版本中增加的XtraSpreadsheet.SpreadsheetControl控件就可以实现Excel的显示和处理操作，这个控件很强大，可以处理很复杂格式的Excel文档，虽然我原来使用了另外一个FarPoint Spread控件组，不过这个XtraSpreadsheet控件组，如果集成在DevExpress也就很方便了。



这个DevExpress的控件，可以在其中进行Excel的新建、保存、打印预览等操作，当然也可以打开我们已有的Excel文件了。

打开文件后，界面效果如下所示。



界面的相关功能操作代码如下所示。

[复制代码](javascript:void(0);)

/// <summary>

/// Excel控件的测试例子

/// </summary>

public partial class ExcelViewer : Form

{

//记录窗体的名称

readonly string mainFormText;

public ExcelViewer()

{

InitializeComponent();

//记录窗体的名称，并实现文档变化事件的处理，方便显示新的文件名称

mainFormText = this.Text;

this.spreadsheetControl1.DocumentLoaded += new EventHandler(spreadsheetControl1\_DocumentLoaded);

}

/// <summary>

/// 文档变化后，实现对新文件名称的显示

/// </summary>

void spreadsheetControl1\_DocumentLoaded(object sender, EventArgs e)

{

string fileName = Path.GetFileName(this.spreadsheetControl1.Document.Path);

if (String.IsNullOrEmpty(fileName))

{

Text = mainFormText;

}

else

{

Text = fileName + " - " + mainFormText;

}

}

/// <summary>

/// 打开Excel文件

/// </summary>

private void btnOpenFile\_Click(object sender, EventArgs e)

{

string filePath = FileDialogHelper.OpenExcel();

if (!string.IsNullOrEmpty(filePath))

{

IWorkbook workbook = spreadsheetControl1.Document;

workbook.LoadDocument(filePath);

}

}

/// <summary>

/// 保存Excel文件

/// </summary>

private void btnSaveFile\_Click(object sender, EventArgs e)

{

spreadsheetControl1.SaveDocument();

}

/// <summary>

/// 另存为Excel文件

/// </summary>

private void btnSaveAs\_Click(object sender, EventArgs e)

{

string dir = System.Environment.CurrentDirectory;

string filePath = FileDialogHelper.SaveExcel("", dir);

if (!string.IsNullOrEmpty(filePath))

{

try

{

IWorkbook workbook = spreadsheetControl1.Document;

workbook.SaveDocument(filePath);

MessageUtil.ShowTips("保存成功");

}

catch (Exception ex)

{

LogTextHelper.Error(ex);

MessageUtil.ShowError(ex.Message);

}

}

}

/// <summary>

/// Excel文件打印

/// </summary>

private void btnPreview\_Click(object sender, EventArgs e)

{

this.spreadsheetControl1.ShowPrintPreview();

}

}

[复制代码](javascript:void(0);)

预览也很方便，和Word的预览操作类似。



以上就是几个常用文档的显示和操作案例，有了这些我们很容易整合到我们的附件管理里面了。

如我在我的《Winform开发框架》、《混合式开发框架里面》使用的通用附件管理模块，就是基于这些特点，实现图片、Excel文档、Word文档和PDF等文档的在线预览和管理操作，界面截图如下所示。

