# 使用 C1MultiDocument

C1MultiDocument组件被设计用作允许创建，保存以及导出大型文档，用来解决由于内存限制单个C1PrintDocument 无法处理的使用场景。

C1MultiDocument对象提供了一个Items集合，该集合可以包含一个或多个C1MultiDocumentItem类型的元素。每一个这样的元素代表了一个C1PrintDocument。通过使用压缩以及临时的磁盘存储，可以允许将几个C1PrintDocument合并为一个大的多重文档，这个过程可能会造成内存溢出，如果所有页面隶属于单个的C1PrintDocument。下面的代码片段演示了如何创建并预览一个一个多重文档：

* VisualBasic

DimmdocAs NewC1MultiDocument() mdoc.Items.Add(C1PrintDocument.FromFile("myDoc1.c1dx")) mdoc.Items.Add(C1PrintDocument.FromFile("myDoc2.c1dx")) mdoc.Items.Add(C1PrintDocument.FromFile("myDoc3.c1dx")) DimpviewAs New C1PrintPreviewDialog()

pview.Document=mdocpview.ShowDialog()

* C#

C1MultiDocumentmdoc= newC1MultiDocument(); mdoc.Items.Add(C1PrintDocument.FromFile("myDoc1.c1dx")); mdoc.Items.Add(C1PrintDocument.FromFile("myDoc2.c1dx")); mdoc.Items.Add(C1PrintDocument.FromFile("myDoc3.c1dx")); C1PrintPreviewDialogpview= new C1PrintPreviewDialog(); pview.Document=mdoc;

pview.ShowDialog();

C1MultiDocument支持所包含文档之间的链接，共通的TOC目录，共通的页面编码，以及整体的页数计算。

请注意，C1MultiDocument不存储到C1PrintDocument 对象的引用，而是将他们序列化（做为.cid/x文件格式）并存储该结果。因此，您可以创建非常大的多重文档而不会导致内存耗尽，当然您的代码如果引用了单独的C1PrintDocument，那么在添加到一个C1MultiDocument之后，这个引用关系将不会被保持。所以，当使用C1MultiDocument时，请确保不要将引用添加到多重文档之中的单个文档。

C1MultiDocument可以保存为“C1 Open XML Multi Document”格式，后缀名为.c1mdx。通过使用任意的Export方法的重载，C1MultiDocument可以导出成多种格式。详细信息请参见[导出一个 C1MultiDocument 文件](#_bookmark206)。

C1MultiDocument 可以通过任何Print以及PrintDialog进行打印。更多细节，请参见[打印一个C1MultiDocument 文件](#_bookmark207)。

## C1MultiDocument 限制

提供C1MultiDocument组件的主要目的是处理由于内存限制而无法被正常创建/导出/打印的巨大文档。有关于的文档的大小和数量没有限制。但是您可能需要使用磁盘存储而不是内存（默认），详见SetStorage方法。

一个多重文档存在一个限制，按照具体应用的不同，这个限制可能会也可能不会非常重要：它不支持您访问其中包含的C1PrintDocuments 的对象模型。换句话说，您可以用下面的代码：

* VisualBasic

DimmdocAs NewC1MultiDocument() mdoc.Items.Add(C1PrintDocument.FromFile("file1.c1dx")) mdoc.Items.Add(C1PrintDocument.FromFile("file2.c1dx")) mdoc.Items.Add(C1PrintDocument.FromFile("file3.c1dx"))

* C#

C1MultiDocumentmdoc= newC1MultiDocument(); mdoc.Items.Add(C1PrintDocument.FromFile("file1.c1dx")); mdoc.Items.Add(C1PrintDocument.FromFile("file2.c1dx")); mdoc.Items.Add(C1PrintDocument.FromFile("file3.c1dx"));

但是您不能再添加如下的内容：

* VisualBasic

Dim doc As C1PrintDocument=mdoc.Items(1)

* C#

C1PrintDocumentdoc=mdoc.Items[1];

如果这种限制对于特定的应用程序不存在问题，甚至您可以使用C1MultiDocument组件“模块化”您的应用程序，即使不存在任何的内存使用问题。

## 创建并预览C1MultiDocument 文件

您可以通过Add方法向C1MultiDocumentItemCollection集合添加一个项目。您可以通过使用Load方法加载一个文件至C1MultiDocument组件。您可以使用Clear方法删除一个文件。这种方法可以清除任何之前加载到C1MultiDocument组件中的文件。

您可以通过Add方法向C1MultiDocumentItemCollection集合添加一个项目。完成以下步骤：

在设计视图中，双击以打开代码编辑器。

1. 将下面的代码添加到Load事件：
	* VisualBasic

Dimppc As New C1PrintPreviewControl Controls.Add(ppc)

ppc.Dock=DockStyle.Fill

DimpdocAs New C1PrintDocument Dim pdoc2As New C1PrintDocument DimmdocAs New C1MultiDocument

pdoc.Body.Children.Add(NewC1.C1Preview.RenderText("Hello!")) pdoc2.Body.Children.Add(NewC1.C1Preview.RenderText("World!"))

mdoc.Items.Add(pdoc) mdoc.Items.Add(pdoc2) ppc.Document=mdocmdoc.Generate()

* + C#

C1PrintPreviewControlppc= newC1PrintPreviewControl(); Controls.Add(ppc);

ppc.Dock=DockStyle.Fill;

C1PrintDocumentpdoc= newC1PrintDocument(); C1PrintDocumentpdoc2= new C1PrintDocument(); C1MultiDocumentmdoc= new C1MultiDocument();

pdoc.Body.Children.Add(newC1.C1Preview.RenderText("Hello!")); pdoc2.Body.Children.Add(newC1.C1Preview.RenderText("World!"));

mdoc.Items.Add(pdoc);

mdoc.Items.Add(pdoc2); ppc.Document=mdoc; mdoc.Generate();

以上代码加载两个C1PrintDocument至C1MultiDocument 组件，并在运行时在一个C1PrintPreviewControl 中显示这些文档。

## 导出C1MultiDocument文件

通过任意的Export方法冲在，可以导出C1MultiDocument为多种格式。例如，在下面的例子中将导出C1MultiDocument 到PDF文件。一个为True的布尔值参数表示应当在导出过程中显示一个进度对话框。

* VisualBasic

Me.C1MultiDocument1.Export("C:\exportedfile.pdf",True)

* C#

this.c1MultiDocument1.Export(@"C:\exportedfile.pdf", true);

如果您把以上代码包含在按钮的Click事件的处理程序中，则会在运行时单击该按钮时，导出C1MultiDocument的内容到一个PDF文件。

## 打印C1MultiDocument 文件

C1MultiDocument 可以通过任何Print以及PrintDialog进行打印。例如，下面的代码将打开打印对话框。

* VisualBasic

Me.C1MultiDocument1.PrintDialog()

* C#

this.c1MultiDocument1.PrintDialog();

如果您把以上代码包含在按钮的Click事件的处理程序中，则会在运行时单击该按钮时，出现一个打印对话框。

## C1MultiDocument 大纲

C1MultiDocument 包括对大纲视图的支持。可以通过Outlines指定多重文档的大纲节点。产生的大纲（比如预览）由该集合中的大纲节点以及所包含文档的大纲节点组合而成。提纲可以通过编程方式调用MakeOutlines()方法构建。

将首先处理多重文档自身的Outlines集合，来自于该集合中的节点将包含在结果大纲中。如果一个节点同时指定为OutlineNode的值根据多文档项目的NestedOutlinesMode属性的值最后，没有出现在多文档Outlines集合的由items表示的文档和报表中的大纲节点将自动地按顺序添加至结果大纲。

大纲支持由以下属性和方法提供：

* Outlines属性
* MakeOutlines方法
* Outlines属性
* OutlineNode属性