Legacy Example NOTICE

This article predates the introduction of our PdfDocument class. PdfDocument has a Separate method which is much more efficient for separating two or more PDF files. Support strongly recommends using this newer class for this task

HOWTO: Separate a Multi-Page PDF With Automatic Repair of Damaged PDFs

HOWTO: Combine Multiple PDFs with Automatic Repair of Damaged PDFs

HOWTO: Combine Multiple PDF Files With Fillable Forms (AcroForms)

Main Article Content

When splitting a large pdf into smaller pdf files, it can be difficult to manage the memory needed to complete the operation. Normally the FileSystemImageSource is sufficent to save a large pdf file while not keeping all the frames in memory. However, this prevents you from limiting the frames that are used. Below is an overload of the FileSystemImageSource that allows the setting of a start and finish property.

C#

```
ublic class RangedFileSystemImageSource: FileSystemImageSource { private int start; public
int Start { get { return start; } set { start = value; Reset(); } } private int finish;
public int Finish { get { return finish; } set { finish = value; Reset(); } } public
RangedFileSystemImageSource(string[] FileNames, bool doAllFrames, int Start, int Finish) :
base(FileNames, doAllFrames) { count = 0; start = Start; finish = Finish;
this.ImmediateUnload = true; } private int count; protected override void LowLevelReset() {
count = 0; this.Flush(); base.LowLevelReset(); } protected override ImageSourceNode
LowLevelAcquire(int index) { count++; return base.LowLevelAcquire(index+start); } protected
override bool LowLevelHasMoreImages() { if (count > finish - start) return false; return
base.LowLevelHasMoreImages(); } protected override bool LowLevelTotalImagesKnown() { return
true; } protected override int LowLevelTotalImages() { return Math.Max((finish - start) + 1,
0); /* Return the number of images we intend to return, unless it would be < 0, then just
return 0 */ }}
```

VB.NET

```
ublic Class RangedFileSystemImageSource Inherits FileSystemImageSource Private m_start As
Integer Public Property Start() As Integer Get Return m_start End Get Set(ByVal value As
Integer) m_start = value Reset() End Set End Property Private m_finish As Integer Public
```

```
Property Finish() As Integer Get Return m_finish End Get Set(ByVal value As Integer) m_finish
= value Reset() End Set End Property Public Sub New(ByVal FileNames As String(), ByVal
doAllFrames As Boolean, ByVal Start As Integer, ByVal Finish As Integer)
MyBase.New(FileNames, doAllFrames) count = 0 m_start = Start m_finish = Finish
Me.ImmediateUnload = True End Sub Private count As Integer Protected Overloads Overrides Sub
LowLevelReset() count = 0 Me.Flush() MyBase.LowLevelReset() End Sub Protected Overloads
Overrides Function LowLevelAcquire(ByVal index As Integer) As ImageSourceNode count += 1
Return MyBase.LowLevelAcquire(index + m_start) End Function Protected Overloads Overrides
Function LowLevelHasMoreImages() As Boolean If count > m_finish - m_start Then Return False
End If Return MyBase.LowLevelHasMoreImages() End Function Protected Overloads Overrides
Function LowLevelTotalImagesKnown() As Boolean Return True End Function Protected Overloads
Overrides Function LowLevelTotalImages() As Integer Return Math.Max((m_finish - m_start) + 1,
0) 'Return the number of images we intend to return, 'unless it would be < 0, then just
return 0 End Function End Class
```

Original Article:

Q10278 - HOWTO: Split a large pdf file into smaller pdf files while conserving memory

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50194/HOWTO-Split-a-Large-PDF-File-Into-Sm...