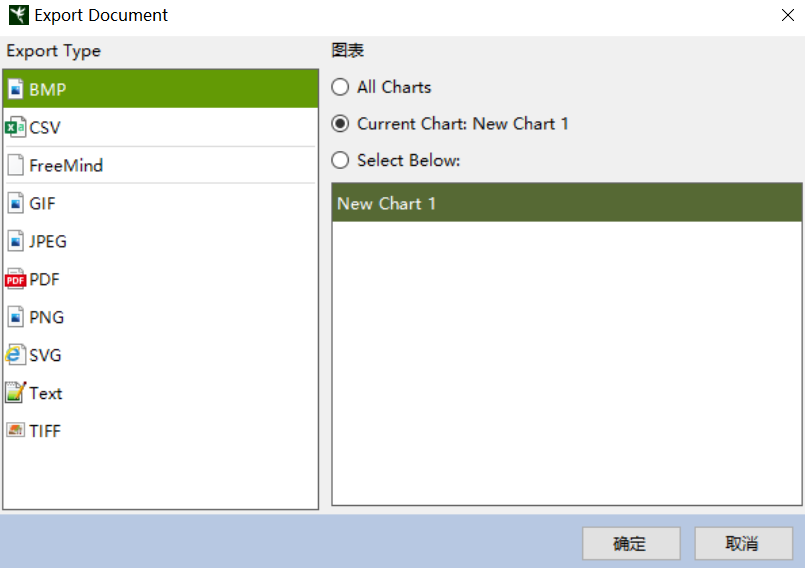
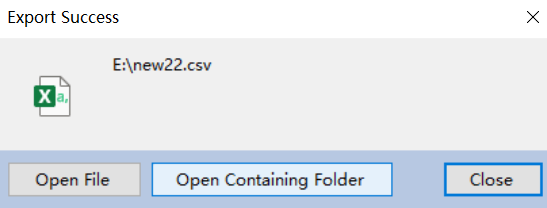
**文件导出分析报告**

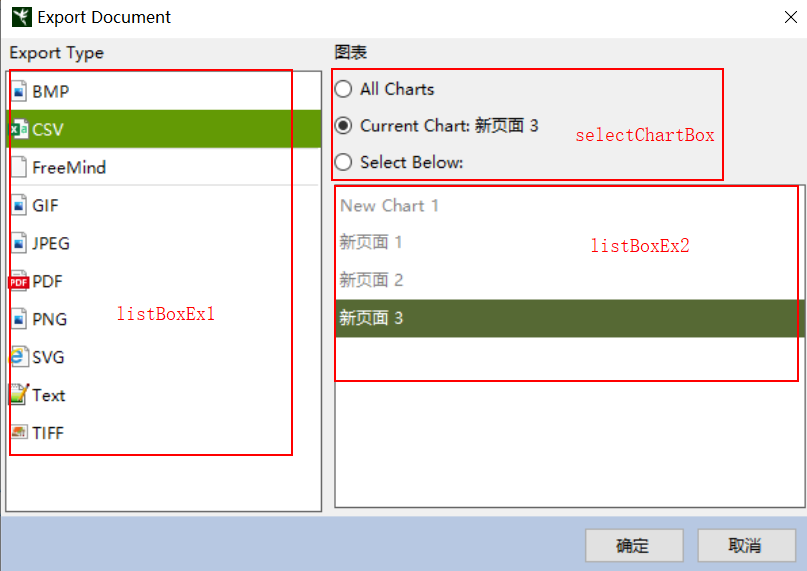
# 一、文件导出功能描述

Blumind的文件导出功能主要包括四大类10种导出文件类型以及根据需要选择导出当前图表文件、导出所有图表文件、导出已有文件中的任意一个。本报告主要分析文件导出的界面设计、功能实现。

# 效果预览







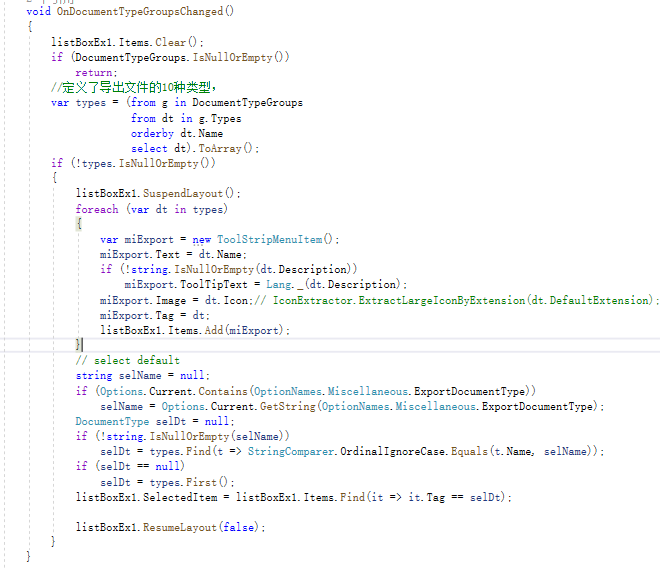
# 三、功能分析

## 3.1 左侧Export Type列表加载

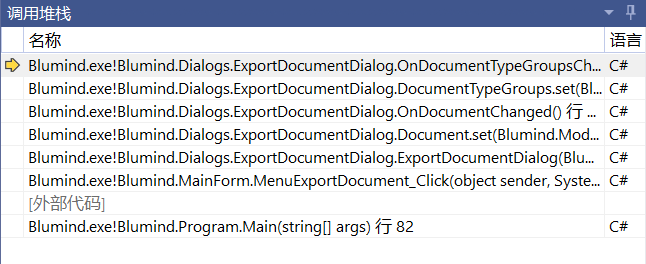
左侧导出类型列表listBoxEx1是一个菜单栏效果的单选控件

### 3.1.1 入口函数定位

以ExportDocumentDialog.OnDocumentTypeGroupsChanged()作为程序的入口函数，这里的函数名起的稍有歧义



### 3.1.2 脉络梳理



该函数的主要调用堆栈如下：

MenuExportDocument\_Click()->ExportDocumentDialog()->Document.set()->OnDocumentChanged()->DocumentTypeGroups.set()->OnDocumentTypeGroupsChanged()

先在主界面MainForm上触发导出点击事件，首先根据读取当前文档所有chart页面与对应的当前chart页面，最后触发OnDocumentTypeGroupsChanged()完成左侧导出类型的菜单栏构建

### 3.1.3 细节分析

其实左侧导出列表的构建是分为两大块，一是定义控件大小与位置，二是加载控件的具体内容，这里主要是介绍控件的具体加载内容

（1）为了构建构建左侧列表，首先定义了一个DocumentTypeGroups集合，它有两层结构，第一层代表name:PDF,Image,Xml,Text，每个name对应不同的types。

0:pdf:pdf,

1:image :PNG/JPEG/BMP/GIF/TIFF

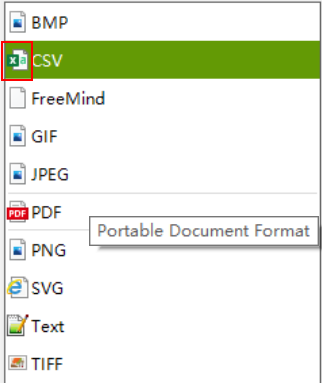
2:xml :SVG/FreeMind

3:Text: text/csv

这里使用LINQ代表语言集成查询，它允许我们用SQL查询数据库的方式来查询数据的集合，使用它，你可以从数据库、程序对象的集合以及XML文档中查询数据，通过这个查询语句获取所有的文档导出类型即dt。

1. **void** OnDocumentTypeGroupsChanged()
2. {
3. ……
4. var types = (from g in DocumentTypeGroups
5. from dt in g.Types
6. orderby dt.Name
7. select dt).ToArray();
8. ……
9. }

（2）文档导出类型具体的定义在Core.DocumentType中，其中定义导出类型的Name、TypeMine、Description、Icon等，这个主要介绍图标的生成过程



1. **public** DocumentType(string name, string typeMime, IEnumerable<string> extensions, string desc = null)
2. {
3. Name = name;   //姓名
4. TypeMime = typeMime;
5. Extensions = extensions;  //扩展名
6. Description = desc;   //描述信息，鼠标浮动在上面时展示
7. Icon = IconExtractor.ExtractSmallIconByExtension(DefaultExtension);  //图标提取
8. }

IconExtractor. ExtractIconByExtension()通过传入函数的扩展名，创建一个对应的文件，例如：通过“.csv”后缀名创建一个文件，通过读取该临时文件文件名的图标来生成左侧列表的文件导出类型对应的图标，读取完成后就删除该临时文件。

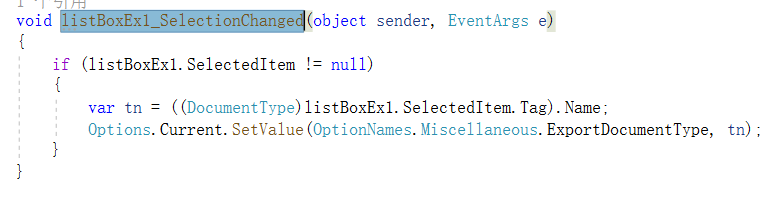
1. public static Bitmap ExtractIconByExtension(string extension, bool small)
2. {
3. Bitmap icon = null;
4. **if** (!string.IsNullOrEmpty(extension))
5. {
6. string tempFile = Path.Combine(Path.GetTempPath(), string.Format("blumind\_test{0}", extension));
7. **try**
8. {
9. **if** (!File.Exists(tempFile))
10. {  //创建文件
11. using (FileStream stream = new FileStream(tempFile, FileMode.Create, FileAccess.Write))
12. {
13. stream.Close();
14. }
15. }
17. icon = IconExtractor.ExtractAssociatedIconImage(tempFile, small);
18. }
19. **finally**
20. {
21. **if** (File.Exists(tempFile))
22. File.Delete(tempFile);
23. }
24. }

## 3.2 Export Type类型切换

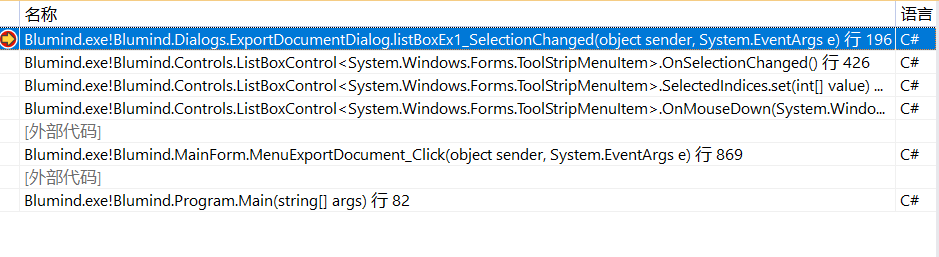
点击左侧导出类型列表，切换不同的列表类型，进行导出操作。

### 3.2.1 入口函数

在点击切换时触发了listBoxEx1\_SelectionChanged()函数，这里以这个函数作为入口



### 3.2.1 脉络梳理



主要函数调用如下：

MenuExportDocument\_Click()->ListBoxControl.OnMouseDown()->ListBoxControl.SelectedIndices.set()->OnSelectionChanged()->listBoxEx1\_SelectionChanged()

如果在ListBoxControl.OnMouseDown()触发鼠标点击事件后在ListBoxControl.SelectedIndices函数中判断前后两个值是否相等，如果不相等的话调用listBoxEx1\_SelectionChanged函数，进行切换后的值选中

### 3.2.2 细节分析

（1）在这里切换功能实现在在左侧列表所对应的listBoxEx1控件上绑定了一个事件SelectionChanged，在事件SelectionChanged上绑定了一个函数listBoxEx1\_SelectionChanged()来实现

* 在ListBoxControl中创建事件:

public event System.EventHandler SelectionChanged

* 在ExportDocumentDialog中绑定函数:

listBoxEx1.SelectionChanged += listBoxEx1\_SelectionChanged

（2）通过SetValue进行值改变，key代表ExportDocumentType他的value是原始的值，value代表变化后的值，如果两个值不一样的话，就对data字典里面的key:ExportDocumentType进行更新

1. **public** **void** SetValue(string key, object value)
2. {
3. **if** (key == null)
4. **throw** **new** ArgumentNullException();
6. object old = GetValue(key);
7. **if** (old != value)
8. {
9. **if** (!ChangedItems.ContainsKey(key))
10. ChangedItems.Add(key, old);
11. Data[key] = value;
12. OnOptionChanged(key, old, value);
13. Modified = **true**;
14. }
15. }

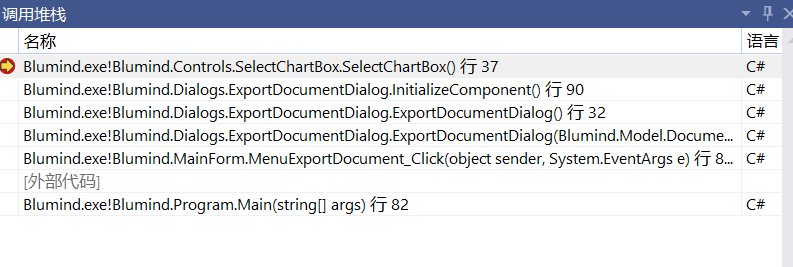
## 3.3 右侧charts加载

### 3.3.1 入口函数

右侧SelectChartBox是由3个radioButton,1个listBoxEx2组成，先加载外部导出窗口然后加载内部selectChartBox()

1. **public** SelectChartBox()
2. {
3. InitializeComponent();
5. LanguageManage.CurrentChanged += LanguageManage\_CurrentChanged;
6. }

### 3.3.2 脉络梳理



主要函数调用如下：

先调用MenuExportDocument\_Click触发导出事件然后在ExportDocumentDialog上加载控件时涉及到右侧单选控件加载引发了SelectChartBox内部加载函数InitializeComponent()

### 3.3.3 细节分析

文件导出时单选项Current Chart这一选项会在后面带出目前选中的chart名称。

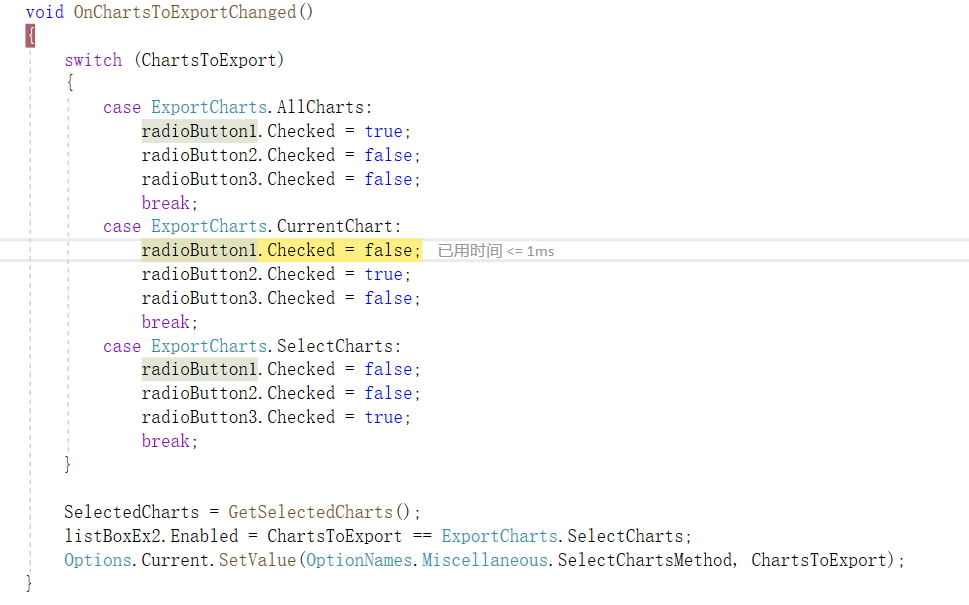


1. void OnCurrentChartChanged()
2. {
3. **if** (CurrentChart != null)
4. radioButton2.Text = string.Format("{0}: {1}", Lang.\_("Current Chart"), CurrentChart.Name);
5. **else**
6. radioButton2.Text = Lang.\_("Current Chart");
7. }

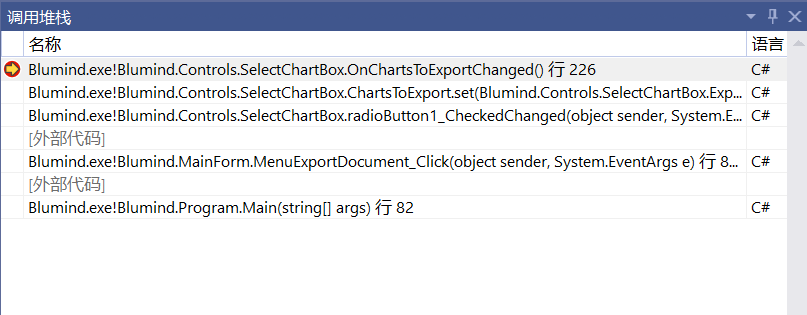
## 3.4 charts切换

### 3.4.1 入口函数

根据函数名猜想OnChartsToExportChanged函数会涉及到charts切换，经过验证后果然会触发断点



### 3.4.2 脉络梳理



主要函数调用如下:

radioButton1\_CheckedChanged()->ChartsToExport.set()->OnChartsToExportChanged()

右侧图表选择到对应的radioButton引发了radioButton\_CheckedChanged事件，最后到函数OnChartsToExportChanged()实现按钮的切换功能

### 3.4.3 细节分析

（1）不同radioButton的切换主要是使用了一个switch循环，判断每一个radioButton.Checked的值为true是选中状态，false是非选中状态，列举出3种不同的可能选中的状态。根据不同的选择状态返回不同的数据类型值

1. ChartPage[] GetSelectedCharts()
2. {
3. switch (ChartsToExport)
4. {
5. case ExportCharts.AllCharts:
6. **return** Document.Charts.ToArray();
7. case ExportCharts.CurrentChart:
8. **if** (CurrentChart != null)
9. **return** new ChartPage[] { CurrentChart };
10. **else**
11. **return** null;
12. case ExportCharts.SelectCharts:
13. **return** listBoxEx2.SelectedItems;
14. default:
15. **return** null;
16. }
17. }

（2）在点击其他选项时selectBelow下对应的listBoxEx2是默认的无法使用的状态，当点选到selectBelow时，listBoxEx2是可选状态，主要是以下语句实现

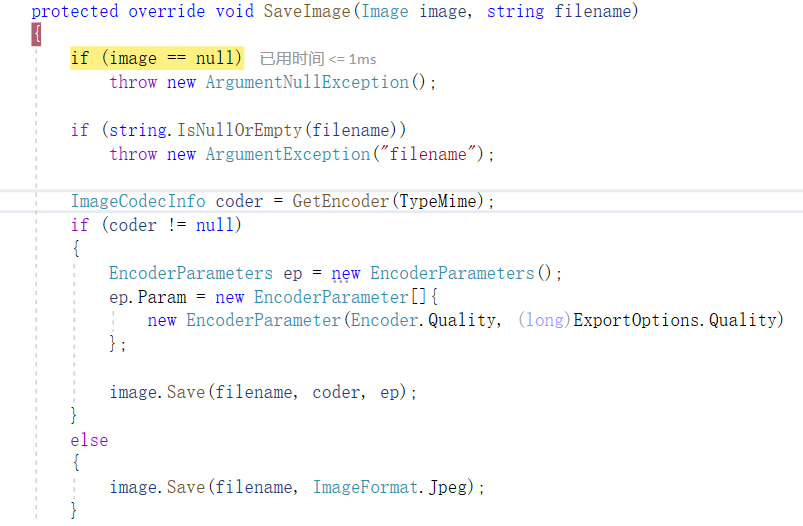
**listBoxEx2.Enabled = ChartsToExport == ExportCharts.SelectCharts**

## 3.5 文件导出

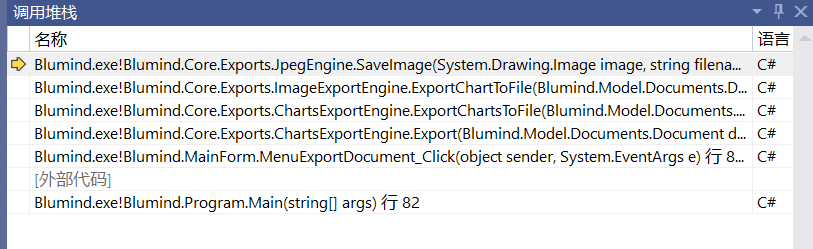
(1)导出图片类型数据，其中导出png、tiff、jpeg、gif、bmp实现步骤类似，这里主要介绍导出JPG类型图片

### 3.5.1入口函数

所有导出文件都在Exports目录下面，根据文件名JpegEngine.cs推测为导出jpeg相关的代码文件。这里经验证入口函数为SaveImage()



### 3.5.2 脉络梳理



主要函数调用如下：

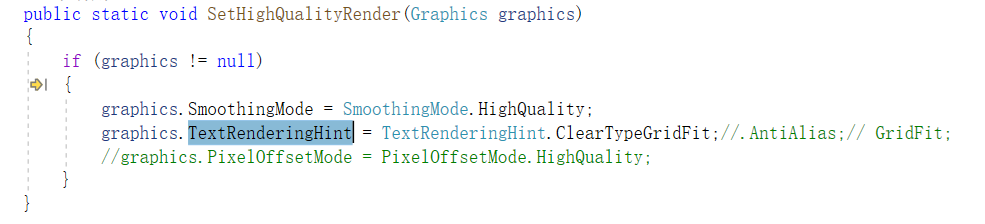
ChartsExportEngine.Export()->ChartsExportEngine.ExportChartsToFile->ImageExportEngine.ExportChartToFile()->JpegEngine.SaveImage()

先确定导出的类型和导出文档的个数，这里是假设导出文档的个数为1时，调用ImageExportEngine导出图片

### 3.5.3 细节分析

（1）实现文档转化成图片的方法主要是在ImageExportEngine.ExportChartToFile，将文件转化成图片类型，使用位图存储(Bitmap)方式，bmp代表图片的md5值

1. **protected** override **bool** ExportChartToFile(Document document, ChartPage chart, string filename)
2. **……**
3. Size contentSize = chart.GetContentSize();   //获取文档的分辨率，1124\*796
4. **if** (contentSize.Width <= 0 || contentSize.Height <= 0)
5. **return** **false**;
7. Bitmap bmp = **new** Bitmap(contentSize.Width, contentSize.Height);
8. **using** (Graphics grf = Graphics.FromImage(bmp))
9. {
10. PaintHelper.SetHighQualityRender(grf);   //设置高质量渲染
11. **if** (!TransparentBackground)   //是否需要背景色
12. {
13. grf.Clear(chart.BackColor);
14. }
15. var render = **new** GeneralRender();
16. var args = **new** RenderArgs(RenderMode.Export, grf, (MindMap)chart, ChartBox.DefaultChartFont);
17. render.Paint((MindMap)chart, args);
18. }
19. }



（2）图片的绘画主要是通过paint函数来实现的，一共分为两种，一种是节点的绘画，一种是连接线的绘画

**PaintNode(map.Root, args);**

**PaintLinkLines(map.Root, args);**

(2)其他导出类型入csv,txt,freemind,pdf,svg，除去具体文件中的导出代码有调整，其大致的调用过程基本一致

ChartsExportEngine.Export()->ChartsExportEngine.ExportChartsToFile->FreeMindEngine.ExportChartToFile()/CsvEngine. ExportChartToFile等