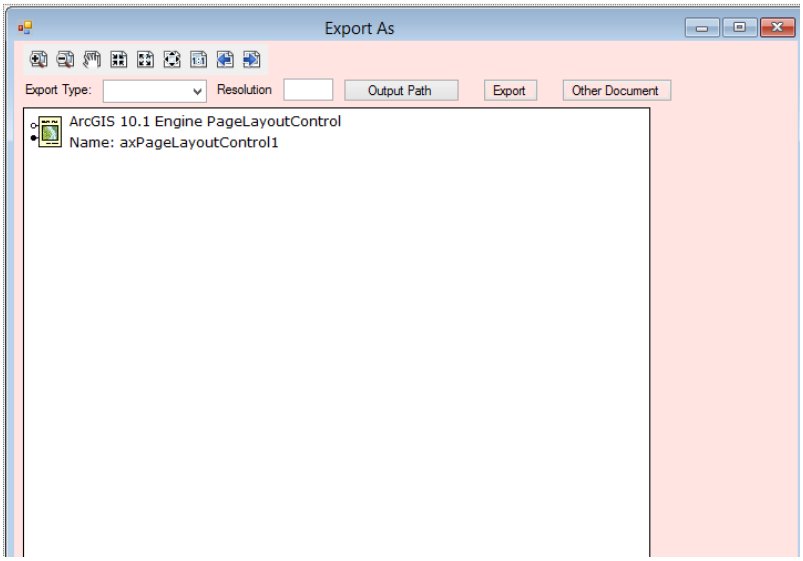
Chapter 9 Output and Print

1. Add a Windows Form “Export” to the project:



GUI implementation detail could be found at “Export.Designer.cs” file.

2. Implementation of GetSymbol:

- Construction method:

```
public Export(IPageLayout pagelayout)
{
    InitializeComponent();
    axPageLayoutControl1.PageLayout = pagelayout;
}
```

- Add click event of “Other Document”:

```
private void btnOpenFile_Click(object sender, EventArgs e)
{
    Stream myStream;
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Filter = "template files (*.mxt)|*.mxt|mxd files (*.mxd)|*.mxd";
    openFileDialog1.FilterIndex = 2;
    openFileDialog1.RestoreDirectory = true;
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        if ((myStream = openFileDialog1.OpenFile()) != null)
        {
            string fileName = openFileDialog1.FileName;
            if (axPageLayoutControl1.CheckMxFile(fileName))
            {
                axPageLayoutControl1.LoadMxFile(fileName, "");
            }
            myStream.Close();
        }
    }
}
```

- Add click event of “Output Path”:

```
string exportFileName = "C:\\Temp\\Test.bmp";
string fileExtension = "BMP";
private void btnOutputFile_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        exportFileName = saveFileDialog1.FileName + "." + fileExtension;
    }
}
```

- SelectedIndexChanged event of cbxOutputType:

```
string strOutputType = "ExportBMP";
IExport export = new ExportBMP() as IExport;

1 reference | Yuhui Wu, 15 hours ago | 1 change
private void cbxOutputType_SelectedIndexChanged(object sender, EventArgs e)
{
    if (cbxOutputType.SelectedItem == null) return;
    strOutputType = cbxOutputType.SelectedItem.ToString();
    try
    {
        switch (strOutputType)
        {
            case "ExportBMP":
                export = new ExportBMP() as IExport;
                fileExtension = "BMP";
                break;
            case "ExportGIF":  //ArcPressPrinter need ArcGIS Desktop License
                export = new ExportGIF() as IExport;
                fileExtension = "GIF";
                break;
            case "ExportJPEG":
                export = new ExportJPEG() as IExport;
                fileExtension = "JPG";
                break;
            case "ExportPNG":
                export = new ExportPNG() as IExport;
                fileExtension = "PNG";
                break;

            case "ExportTIFF":
                export = new ExportTIFF() as IExport;
                fileExtension = "TIFF";
                break;
            case "ExportAI":
                export = new ExportAI() as IExport;
                fileExtension = "AI";
                break;
            case "ExportEMF":
                export = new ExportEMF() as IExport;
                fileExtension = "EMF";
                break;
            case "ExportPDF":
                export = new ExportPDF() as IExport;
                fileExtension = "PDF";
                break;
            case "ExportPS":
                export = new ExportPS() as IExport;
                fileExtension = "PS";
                break;
            case "ExportSVG":
                export = new ExportSVG() as IExport;
                fileExtension = "SVG";
                break;
            default:
                break;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

- Click event of “Export”:

```
IOutputRasterSettings rasterSettings;
double iOutputResolution = 300;

private void btnPrint_Click(object sender, EventArgs e)
{
    if (export is IOutputRasterSettings)
    {
        rasterSettings = export as IOutputRasterSettings;
        rasterSettings.ResampleRatio = 1;
    }
    if (Information.IsNumeric(txtResolution.Text))
        iOutputResolution = Convert.ToInt32(txtResolution.Text);
    else
        iOutputResolution = 300;
    IActiveView pActiveView = axPageLayoutControl1.ActiveView;
    double iScreenResolution = pActiveView.ScreenDisplay.
        DisplayTransformation.Resolution;
    tagRECT exportRECT;exportRECT.left = 0; exportRECT.top = 0;
    exportRECT.right = Convert.ToInt32(Math.Ceiling(pActiveView.ExportFrame.
        right * (iOutputResolution / iScreenResolution)));
    exportRECT.bottom = Convert.ToInt32(Math.Round(pActiveView.ExportFrame.
        bottom * (iOutputResolution / iScreenResolution)));
    IEnvelope pPixelBoundsEnv = new Envelope() as IEnvelope;
    pPixelBoundsEnv.PutCoords(exportRECT.left, exportRECT.top, exportRECT.right
        , exportRECT.bottom);
    export.Resolution = iOutputResolution;
    export.PixelBounds = pPixelBoundsEnv;
    export.ExportFileName = exportFileName;
    int hDC = export.StartExporting();
    pActiveView.Output(hDC, (int)export.Resolution, ref exportRECT,
        null, null);
    export.FinishExporting();
    export.Cleanup();
    MessageBox.Show("Export Success");
}
```

3. Add A Base Command Class: ExportView:

- Add Class member:

```
public sealed class ExportView : BaseCommand
{
    COM Registration Function(s)

    private IHookHelper m_hookHelper = null;
    IPageLayoutControl3 pPageLayoutControl;
```

- Add code in the OnCreate method:

```
pPageLayoutControl = m_hookHelper.Hook as IPageLayoutControl3;
```

- OnClick Method:

```
public override void OnClick()
{
    // TODO: Add ExportView.OnClick implementation
    Export exportActiveView = new Export(pPageLayoutControl.PageLayout);
    exportActiveView.ShowDialog();
}
```

4. Back to MainForm, add a menu content and its click event:

```
private void exportAsToolStripMenuItem_Click(object sender, EventArgs e)
{
    ICommand export = new ExportView();
    export.OnCreate(axPageLayoutControl1.Object);
    export.OnClick();
    axPageLayoutControl1.CurrentTool = export as ITool;
}
```