

```
1 using MahApps.Metro.Controls;
2 using Microsoft.Data.Sqlite;
3 using Microsoft.Win32.TaskScheduler;
4 using PlotterDataGH.Properties;
5 using SendFileTo;
6 using System;
7 using System.Collections.Generic;
8 using System.Data;
9 using System.Diagnostics;
10 using System.IO;
11 using System.Linq;
12 using System.Reflection;
13 using System.Text;
14 using System.Threading.Tasks;
15 using System.Windows;
16 using System.Windows.Controls;
17
18 namespace WpfApp2
19 {
20     /// <summary>
21     /// Interaction logic for MainWindow.xaml
22     /// </summary>
23     public partial class MainWindow : MetroWindow
24     {
25         int addedRows = 0;
26         DataTable dataTable = new DataTable();
27
28         public MainWindow()
29         {
30             InitializeComponent();
31             LoadData();
32         }
33
34         //While a scan is in progress disable the scanning button
35         public void DisableWhileScanning()
36         {
37             foreach (UserControl1 row in fillerGrid.Children)
38             {
39                 row.btnScan.IsEnabled = false;
40             }
41         }
42
43         //Load the data from the local database file
44         public void LoadData()
45         {
46             dataTable.Clear();
47             SqliteConnection cnn;
48             SqliteCommand cmd = null;
49             cnn = new SqliteConnection("Data Source=plotterData.db;");
50             cnn.Open();
51
52             string query = "SELECT m1.*, models.plotter_type FROM printer_data
53                             m1 LEFT JOIN printer_data m2 ON (m1.serial_number =
54                             m2.serial_number AND m1.id < m2.id) INNER JOIN models on
55                             models.id = m1.model_id WHERE m2.id IS NULL";
56
57             cmd = new SqliteCommand(query, cnn);
```

```

54
55     SqlDataReader reader = cmd.ExecuteReader();
56     dataTable.Load(reader);
57
58     foreach (DataRow row in dataTable.Rows)
59     {
60         RowDefinition rd = new RowDefinition();
61         rd.Height = GridLength.Auto;
62         fillerGrid.RowDefinitions.Add(rd);
63         UserControl1 userControl1 = new UserControl1();
64         userControl1.lblMeterstand.Content = string.Format(row  ➤
65             ["meters_printed"].ToString());
66         userControl1.lblNaam.Content = string.Format(row  ➤
67             ["naam"].ToString());
68         userControl1.plotterId = Convert.ToInt32(row["id"]);
69         userControl1.lblMerk.Content = string.Format(row  ➤
70             ["plotter_type"].ToString());
71         userControl1.lblMerk.Uid = string.Format(row  ➤
72             ["model_id"].ToString());
73         userControl1.plotterIp = string.Format(row["ip"].ToString());
74         userControl1.lblSerialNumber.Content = "S/N: " + string.Format  ➤
75             (row["serial_number"].ToString());
76         userControl1.serialnm = string.Format(row  ➤
77             ["serial_number"].ToString());
78
79         var dt = DateTime.Parse((string)row["datetime"]);
80
81         userControl1.lblTime.Content = dt.ToString("dd/MM/yy H:mm");
82         userControl1.loadData();
83         userControl1.ParentForm = this;
84         fillerGrid.Children.Add(userControl1);
85         Grid.SetRow(userControl1, fillerGrid.RowDefinitions.Count -  ➤
86             1);
87         addedRows++;
88     }
89
90     //Create a CSV file for mailing
91     try
92     {
93         StringBuilder sb = new StringBuilder();
94
95         IEnumerable<string> columnNames =  ➤
96             dataTable.Columns.Cast<DataColumn>().  ➤
97                 Select(column =>  ➤
98                     column.ColumnName);
99         sb.AppendLine(string.Join(",", columnNames));
100
101         DataTable cartridgeTable = new DataTable();
102
103         foreach (DataRow row in dataTable.Rows)
104         {
105             IEnumerable<string> fields = row.ItemArray.Select(field =>  ➤
106                 field.ToString());
107             sb.AppendLine(string.Join(",", fields));

```

```
100
101         SqliteConnection cnn1;
102         SqliteCommand cmd1 = null;
103         cnn1 = new SqliteConnection("Data Source=plotterData.db;");
104         cnn1.Open();
105
106         string query1 = string.Format("SELECT * FROM `cartridge_reading` where `parent_id` = {0}", row[0]);
107         cmd1 = new SqliteCommand(query1, cnn1);
108
109         SqliteDataReader reader1 = cmd1.ExecuteReader();
110
111         cartridgeTable.Load(reader1);
112     }
113
114     File.WriteAllText("plotterData.csv", sb.ToString());
115
116     StringBuilder sb1 = new StringBuilder();
117
118     IEnumerable<string> columnNames1 =
119         cartridgeTable.Columns.Cast<DataColumn>().
120             Select(column =>
121                 column.ColumnName);
122     sb1.AppendLine(string.Join(",", columnNames1));
123
124     foreach (DataRow row1 in cartridgeTable.Rows)
125     {
126         IEnumerable<string> fields = row1.ItemArray.Select(field
127             => field.ToString());
128         sb1.AppendLine(string.Join(",", fields));
129     }
130     File.WriteAllText("cartridgeData.csv", sb1.ToString());
131 }
132 catch (System.IO.IOException)
133 {
134     MessageBox.Show("Please close Excel");
135 }
136
137 //Add another plotter
138 private void btnAdd_Click(object sender, RoutedEventArgs e)
139 {
140     AddPlotter addPlotter = new AddPlotter();
141     addPlotter.ParentForm = this;
142     addPlotter.Show();
143 }
144
145 //Opens the settings page
146 private void Button_Click(object sender, RoutedEventArgs e)
147 {
148     SettingsPage sp = new SettingsPage();
149     sp.Show();
150 }
```

```
151         this.Close();
152     }
153
154     #region Mailer
155
156     private void btnSendMail_Click(object sender, RoutedEventArgs e)
157     {
158         MAPI mapi = new MAPI();
159
160         //mapi.AddAttachment("plotterData.db");
161         mapi.AddRecipientTo("Helpdesk@goedhart-its.com");
162         mapi.AddAttachment(Environment.CurrentDirectory + "\\  \nplotterData.csv");
163         mapi.AddAttachment(Environment.CurrentDirectory + "\\  \n  \ncartridgeData.csv");
164         mapi.SendMailPopup("Testen plotter data", getMailData());
165     }
166
167     private string getMailData()
168     {
169         dataTable.Clear();
170         SqlConnection cnn;
171         SqlCommand cmd = null;
172         cnn = new SqlConnection("Data Source=plotterData.db;");
173         cnn.Open();
174
175         string query = "SELECT m1.*, models.plotter_type FROM printer_data
176             m1 LEFT JOIN printer_data m2 ON (m1.serial_number =
177             m2.serial_number AND m1.id < m2.id) INNER JOIN models on
178             models.id = m1.model_id WHERE m2.id IS NULL";
179
180         cmd = new SqlCommand(query, cnn);
181
182         string mailBody = "";
183
184         foreach (DataRow row in dataTable.Rows)
185         {
186             mailBody += "Plotters: \n";
187             mailBody += string.Format(row["serial_number"].ToString());
188             mailBody += "\n";
189             mailBody += string.Format(row["meters_printed"].ToString());
190             mailBody += "\n";
191             mailBody += string.Format(row["plotter_type"].ToString());
192             mailBody += "\n";
193             mailBody += "\n";
194             mailBody += "Cartridges: \n";
195
196             DataTable dataTableCartridge = new DataTable();
197             SqlCommand cmd1 = null;
198
199             string query1 = string.Format("SELECT * FROM
200                 `cartridge_reading` where `parent_id` = {0}", row["id"]);
201             cmd1 = new SqlCommand(query1, cnn);
202
203             SqlDataReader reader1 = cmd1.ExecuteReader();
204             dataTableCartridge.Load(reader1);
```

```
201
202         foreach (DataRow rowCartridge in dataTableCartridge.Rows)
203         {
204             mailBody += string.Format(rowCartridge
205                                     ["cartridge_model"].ToString());
206             mailBody += "\n";
207             mailBody += string.Format(rowCartridge["volume"].ToString
208                                     ());
209             mailBody += "\n";
210             mailBody += "\n";
211         }
212     }
213     return mailBody;
214 }
215
216 #endregion
217
218 public void TaskCreator()
219 {
220     // Get the service on the local machine
221     using (TaskService ts = new TaskService())
222     {
223         if (ts.GetTask("Plotter Scanner") != null)
224         {
225             ts.RootFolder.DeleteTask("Plotter Scanner");
226         }
227     }
228
229     foreach (UserControl1 row in fillerGrid.Children)
230     {
231         // Get the service on the local machine
232         using (TaskService ts = new TaskService())
233         {
234
235             var debugField = System.IO.Path.GetDirectoryName(
236                 Assembly.GetExecutingAssembly().GetName().CodeBase);
237
238             debugField = debugField.Substring(6);
239
240             var filename = debugField + @"/ghWebscraper.exe";
241
242             //Start the Converted python file and pass the paramater
243             string arguments = string.Format(@"{0} {1} {2} {3}",
244                                     row.lblMerk.Uid.ToString(), row.plotterIp,
245                                     Settings.Default.bedrijfsNaam, row.lblNaam.Content);
246
247             TaskDefinition td = ts.NewTask();
248
249             if (ts.GetTask("Plotter Scanner") != null)
250             {
251                 td = ts.GetTask("Plotter Scanner").Definition;
252             }
253
254             // Create a new task definition and assign properties
```

```
253
254         td.RegistrationInfo.Description = "Scans plotter";
255         td.RegistrationInfo.Author = "Goedhart Groep";
256
257         if (td.Triggers.Count == 0)
258         {
259             // Create a trigger that will fire the task at this time every day
260             td.Triggers.Add(new DailyTrigger { DaysInterval = 1 });
261         }
262
263         // Create an action that will launch Notepad whenever the trigger fires
264         td.Actions.Add(new ExecAction(filename, arguments, debugField));
265
266         // Register the task in the root folder
267         ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner", td);
268
269     }
270
271 }
272
273 using (TaskService ts = new TaskService())
274 {
275     var debugField = System.IO.Path.GetDirectoryName(
276         Assembly.GetExecutingAssembly().GetName().CodeBase);
277
278     debugField = debugField.Substring(6);
279
280     var filename = debugField + @"NewWay.exe";
281
282     TaskDefinition td = ts.NewTask();
283
284     if (ts.GetTask("Plotter Scanner") != null)
285     {
286         td = ts.GetTask("Plotter Scanner").Definition;
287     }
288
289     // Create a new task definition and assign properties
290
291     td.RegistrationInfo.Description = "Scans plotter";
292     td.RegistrationInfo.Author = "Goedhart Groep";
293
294     if (td.Triggers.Count == 0)
295     {
296         // Create a trigger that will fire the task at this time every day
297         td.Triggers.Add(new DailyTrigger { DaysInterval = 1 });
298     }
299
300     // Create an action that will launch Notepad whenever the trigger fires
```

```
302         td.Actions.Add(new ExecAction(filename, null, debugField));
303
304         // Register the task in the root folder
305         ts.RootFolder.RegisterTaskDefinition(@"Plotter Scanner", td);
306     }
307 }
308
309 #region Scanning
310 public void RunScan(string Merk, string IP, string Naam)
311 {
312     var debugField = System.IO.Path.GetDirectoryName(
313     Assembly.GetExecutingAssembly().GetName().CodeBase);
314
315     debugField = debugField.Substring(6);
316
317     var filename = debugField + @"/ghWebscraper.exe";
318
319     //Start the Converted python file and pass the paramater
320     string arguments = string.Format(@"{0} {1} {2} {3}", Merk, IP,
321     Settings.Default.bedrijfsNaam, Naam);
322
323     //Process myProcess = new Process();
324     //myProcess.Exited += new EventHandler(myProcess_Exited);
325     //myProcess.StartInfo.FileName = filename;
326     //myProcess.StartInfo.Arguments = arguments;
327     //myProcess.Start();
328
329     doStuff(filename, arguments);
330 }
331
332 async System.Threading.Tasks.Task doStuff(string fileName, string
333     args)
334 {
335     DisableWhileScanning();
336     await RunProcessAsync(fileName, args);
337
338     //MahApps.Metro.IconPacks.PackIconFontAwesome fe = btnScan.Content
339     //as MahApps.Metro.IconPacks.PackIconFontAwesome;
340     //fe.Kind =
341     MahApps.Metro.IconPacks.PackIconFontAwesomeKind.BinocularsSolid;
342     //btnScan.IsEnabled = true;
343
344     fillerGrid.RowDefinitions.Clear();
345     fillerGrid.Children.Clear();
346     LoadData();
347 }
348
349 public static async Task<int> RunProcessAsync(string fileName, string
350     args)
351 {
352     using (var process = new Process
353     {
354         StartInfo =
355     {
356         FileName = fileName, Arguments = args,
357         UseShellExecute = false, CreateNoWindow = true,
```

```
353         RedirectStandardOutput = true, RedirectStandardError = true
354     },
355         EnableRaisingEvents = true
356     })
357     {
358         return await RunProcessAsync(process).ConfigureAwait(false);
359     }
360 }
361 private static Task<int> RunProcessAsync(Process process)
362 {
363     var tcs = new TaskCompletionSource<int>();
364
365     process.Exited += (s, ea) => tcs.SetResult(process.ExitCode);
366     process.OutputDataReceived += (s, ea) => Console.WriteLine      ↗
367         (ea.Data);
368     process.ErrorDataReceived += (s, ea) => Console.WriteLine("ERR: " ↗
369         + ea.Data);
370
371     bool started = process.Start();
372     if (!started)
373     {
374         //you may allow for the process to be re-used (started =      ↗
375             false)
376         //but I'm not sure about the guarantees of the Exited event in ↗
377             such a case
378         throw new InvalidOperationException("Could not start process: ↗
379             " + process);
380     }
381
382     process.BeginOutputReadLine();
383     process.BeginErrorReadLine();
384
385     return tcs.Task;
386 }
387 #endregion
}
```