* ERD

|  |
| --- |
|  |
|  |

* 테이블 명세서

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

* 소스코드

|  |
| --- |
| * using MiniERP.Model.DAO.Message; * using System; * using System.Collections.Generic; * using System.ComponentModel; * using System.Data; * using System.Drawing; * using System.Linq; * using System.Text; * using System.Threading.Tasks; * using System.Windows.Forms; * namespace MiniERP.View * { * public partial class RealTimeMonitor : Form * { * Machine\_Monitoring machine\_Server; * public RealTimeMonitor() * { * InitializeComponent(); * this.ShowInTaskbar = false; // 작업표시줄X * this.FormBorderStyle = FormBorderStyle.SizableToolWindow; // 프로그램 전환기 숨기기 * } * private void RealTimeMonitor\_Load(object sender, EventArgs e) * { * machine\_Server = new Machine\_Monitoring(txt\_Log); * machine\_Server.Start(); * } * private void txt\_Log\_TextChanged(object sender, EventArgs e) * { * txt\_Log.SelectionStart = txt\_Log.Text.Length; * richTextBox1.AppendText(txt\_Log.Text + Environment.NewLine); * //ClientConnectingCheck(txt\_Log.Text); * } * private void ClientConnectingCheck(string msg) * { * string temp = ""; * if (msg.Contains("[command]") && msg.Contains("is connecting")) * temp = msg.Replace("[command]", "").Replace("is connecting", ""); //[pc1] * foreach (Control item in panel1.Controls) * { * if (item.Text == temp) * { * ((CheckBox)item).Checked = true; * } * } * } * private void RealTimeMonitor\_FormClosing(object sender, FormClosingEventArgs e) * { * machine\_Server.CloseSever(); * } * private void btn\_inputCountRequest\_Click(object sender, EventArgs e) * { * string command = "[command]"; * switch (((Button)sender).Text) * { * case "종료": command += selectPc + "exit"; machine\_Server.SendMsg(command); break; * case "재부팅": command += selectPc + "restart"; machine\_Server.SendMsg(command); break; * case "투입 자재 개수": command += selectPc + "barcode"; machine\_Server.SendMsg(command); break; * case "머신 설정 변경": Machine\_Info\_Change change = new Machine\_Info\_Change(machine\_Server, selectPc); change.ShowDialog(); break; * default: * break; * } * } * string selectPc = ""; bool clickSwitch = false; * private void pic\_pc1\_MouseClick(object sender, MouseEventArgs e) * { * if (e.Button == MouseButtons.Left && clickSwitch == false) * { * if (((PictureBox)sender).BackColor != SystemColors.ButtonShadow) * { * ((PictureBox)sender).BackColor = SystemColors.ButtonShadow; * selectPc = "[" + ((PictureBox)sender).Name + "]"; * } * clickSwitch = true; * } * else if (((PictureBox)sender).BackColor == SystemColors.ButtonShadow && clickSwitch == true) * { * clickSwitch = false; * if (((PictureBox)sender).BackColor == SystemColors.ButtonShadow) * { * ((PictureBox)sender).BackColor = SystemColors.ButtonHighlight; * } * } * else * MessageBox.Show("한번에 하나의 명령만 가능합니다."); * } * /// <summary> * /// 리치텍스트박스 변경시 일어날 메소드 * /// </summary> * private void richTextBox1\_TextChanged(object sender, EventArgs e) * { * ServerStateChecker(); * } * /// <summary> * /// 텍스트박스의 서버의 커맨드를 이용해 체크박스 체크해줍니다. * /// </summary> * private void ServerStateChecker() * { * string temp = txt\_Log.Text; * if (temp.Contains("[command]") && temp.Contains("is connecting")) * { * temp = temp.Replace("[command]", "").Replace("is connecting", ""); * foreach (Control item in panel1.Controls) * { * if (item.Text == temp) * { * ((CheckBox)item).Checked = true; * } * } * } * else if (temp.Contains("[command]") && temp.Contains("is endconnecting")) * { * temp = temp.Replace("[command]", "").Replace("is endconnecting", ""); * foreach (Control item in panel1.Controls) * { * if (item.Text == temp) * { * ((CheckBox)item).Checked = false; * } * } * } * } * /// <summary> * /// 텍스트박스의 서버 상태를 체크하고, 버튼의 enable속성을 변경해줍니다. * /// </summary> * private void check\_pc1\_CheckedChanged(object sender, EventArgs e) * { * string temp = ((Control)sender).Text; // [pc1] * temp = temp.Replace("[", "").Replace("]", ""); // pc1 * foreach (Control item in panel1.Controls) * { * if (item.Name == temp&&((CheckBox)sender).Checked) * ((PictureBox)item).Enabled = Enabled; * else if (item.Name == temp && !((CheckBox)sender).Checked) * ((PictureBox)item).Enabled = false; * } * } * // 테스트 위한 메서드 * private void btn\_Tester\_Click(object sender, EventArgs e) * { * pc1.Enabled = true; * pc2.Enabled = true; * pc3.Enabled = true; * pc4.Enabled = true; * pc5.Enabled = true; * } * } * } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Sockets;  using System.Text;  using System.Threading;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace MiniERP.Model.DAO.Message  {  public class Machine\_Monitoring  {  TcpClient client = new TcpClient();//tcpclient를 미리 초기화해놓음  NetworkStream stream = default(NetworkStream);  Thread thread;  object txtBox;      string ip = "192.168.0.6";  public string Ip { get { return ip; } set { this.ip = value; } }  string readData = null;  public Machine\_Monitoring(object txtBox)  {  this.txtBox = txtBox;  }  public void Start()  {  IAsyncResult access = null;  try  {  access = client.BeginConnect(ip, 4444, null, null);  var result = access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));  stream = client.GetStream();  byte[] name = Encoding.UTF8.GetBytes("master"); // 접속 닉네임? 주라고하드라 추후수정  stream.Write(name, 0, name.Length);  stream.Flush();  thread = new Thread(getMsg);  thread.Start();  }  catch (Exception)  {  return;  }  }  private void getMsg()  {  while (true)  {  stream = client.GetStream();  Byte[] byteFrom = new byte[client.SendBufferSize];  stream.Read(byteFrom, 0, client.SendBufferSize);  readData = Encoding.UTF8.GetString(byteFrom); // getString  ((TextBox)txtBox).Text = readData;  }  }  public void SendMsg(string msg)  {  Byte[] byteFrom = Encoding.UTF8.GetBytes(msg);  stream.Write(byteFrom, 0, byteFrom.Length);  stream.Flush();  }  public void CloseSever()  {  byte[] msgTemp = Encoding.UTF8.GetBytes("접속종료합니다");  stream.Write(msgTemp, 0, msgTemp.Length);  stream.Flush();  client.Close();  stream.Close();  }  }  } |
| using MiniERP.Model.DAO.Message;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace MiniERP.View  {  public partial class Machine\_Info\_Change : Form  {  string name;  string ip;  Machine\_Monitoring server;  public Machine\_Info\_Change(Machine\_Monitoring server,string name)  {  InitializeComponent();  this.name = name;  this.ip = server.Ip;  this.server = server;  }  private void radioButton1\_CheckedChanged(object sender, EventArgs e)  {  if (radio\_Name.Checked == true)  {  txt\_Name.Enabled = true;  txt\_Ip.Enabled = false;  }  else  {  txt\_Name.Enabled = false;  txt\_Ip.Enabled = true;  }  }  private void Machine\_Info\_Change\_Load(object sender, EventArgs e)  {  txt\_Name.Enabled = true;  txt\_Ip.Enabled = false;  txt\_Name.Text = this.name;  txt\_Ip.Text = this.ip;  }  private void btn\_Submit\_Click(object sender, EventArgs e)  {  string command = "[command]";  if (radio\_Name.Checked == true)  {  command += this.name + "[name]" + txt\_Name.Text;  server.SendMsg(command);  }  else  {  command += this.name + "[ip]" + txt\_Ip.Text;  server.SendMsg(command);  }  Close();  }  }  } |
| using MiniERP.Model.DAO;  using MiniERP.VO;  using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace MiniERP.View  {  public partial class FrmDashBoard : Form  {  //private bool menu\_OpenChk = false;  List<Trade> trades = new List<Trade>();  public FrmDashBoard()  {  InitializeComponent();  lbl\_ToDay.Text = "오늘은 " + DateTime.Today.ToShortDateString() + " 입니다.";  TradeListShow();  }  /// <summary>  /// 트레이드 리스트 다시그립니다.  /// </summary>  public void TradeListShow()  {  trades.Clear();  trades = GetTreade();  ControlAdd(trades);  }  /// <summary>  /// '판매' 거래에대한 정보를 리스트에 초기화  /// </summary>  private List<Trade> GetTreade()  {  TradeDAO tradeDAO = new TradeDAO();  return tradeDAO.GetProgTrade("판매");  }  /// <summary>  /// 거래리스트에 있는 모든 정보에 대한 컨트롤을 동적생성합니다.  /// </summary>  private void ControlAdd(List<Trade> list)  {  panel\_TradeList.Controls.Clear();  int x = 11; int y = 5;  foreach (var item in list)  {  ToDoList temp = new ToDoList(item,split.Panel2,this);  temp.Location = new Point(x, y);  y += 48;  panel\_TradeList.Controls.Add(temp);  temp.Show();  }  }  private void FrmDashBoard\_Load(object sender, EventArgs e)  {  panel1.AutoScroll = true;  }  private void timer1\_Tick(object sender, EventArgs e)  {  lbl\_Time.Text = "현재시간 : " + DateTime.Now.ToLongTimeString();  }  private void btn\_Refresh\_Click(object sender, EventArgs e)  {  TradeListShow();  }  }  } |
| using System;  using System.Collections;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Net;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  // taskkill /im minierp\_client\_jsu.exe /f  namespace MiniErp\_Client\_jsu  {  public partial class Form1 : Form  {  Machine machine = new Machine();  Chatting chatting;  // erro , command 리스트  List<Erro> erros = new List<Erro>();  List<Command> commands = new List<Command>();  List<Barcode> codes = new List<Barcode>();  public Form1()  {  InitializeComponent();  this.Text = machine.Name;  this.TopMost = true;  chatting = new Chatting(machine.Ip, machine.Name, codes);  //StringBuilder sb = new StringBuilder();  //sb.AppendLine("---------[투입 현황]");  //sb.AppendLine("---------[2019-03-02 6:44:50]");  //sb.AppendLine("\*[1234] 1");  //sb.AppendLine("\*[1235] 1");  //sb.AppendLine("----------------------------");  //// 바코드 문자열잘라줄때쓰자  //string temp = sb.ToString().Remove(0, sb.ToString().IndexOf('\*')).Replace("-", "").Trim();  //string[] temp2 = temp.Split('\*');  //MessageBox.Show(temp);  //foreach (var item in temp2)  //{  // MessageBox.Show(item);  //}  }  private void txt\_Barcode\_KeyDown(object sender, KeyEventArgs e)  {  if (e.KeyCode == Keys.Enter)  {  string temp = txt\_Barcode.Text;  txt\_Barcode.Clear();  foreach (var item in codes)  {  if (item.Barcode\_Code == temp)  {  item.Barcode\_Count += 1; // 같다면 카운트  return;  }  }  // 없다면 추가  codes.Add(new Barcode(temp));  }    }  private void Form1\_Load(object sender, EventArgs e)  {  #region erro test 모듈  //Erro testErro = new Erro(1);  //MessageBox.Show(testErro.Erro\_Code + "\n" + testErro.Head + testErro.Erro\_String);  #endregion  if (chatting.Start())  chk\_Server.Checked = true;  toolStripTextBox1.Text = AppConfiguration.GetAppConfig("name");  toolStripTextBox2.Text = AppConfiguration.GetAppConfig("ip");  }  private void BarcodeMonitor()  {  StringBuilder sb = new StringBuilder();  sb.AppendLine("---------[투입 현황]"+machine.Name);  sb.AppendLine(NowTime());  foreach (var item in codes)  {  txt\_Log.Clear();  sb.AppendLine("\*["+item.Barcode\_Code+"]" + "\t" + item.Barcode\_Count);  }  sb.AppendLine("----------------------------");  txt\_Log.Text = sb.ToString();  this.txt\_Barcode.Focus();  }  private string NowTime()  {  DateTime dt = DateTime.Now;  return  "---------[" + dt.ToShortDateString() + " " + dt.Hour + ":" + dt.Minute + ":" + dt.Second + "]";  }  private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  chatting.CloseServer();  }  private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)  {  switch (e.ClickedItem.ToString())  {  case "화면고정":  if (this.TopMost != true)  this.TopMost = true;  else  this.TopMost = false;  break;  case "종료": this.Close(); break;  case "재시작": Application.Restart(); break;  case "자재투입현황": BarcodeMonitor(); break;  default:  break;  }  }  private void toolStripTextBox1\_KeyDown(object sender, KeyEventArgs e)  { //name 변경  if (e.KeyCode == Keys.Enter)  {  AppConfiguration.SetAppConfig("name", toolStripTextBox1.Text);  MessageBox.Show("변경되었습니다.");  }  }  private void toolStripTextBox2\_KeyDown(object sender, KeyEventArgs e)  { // ip 변경  if (e.KeyCode == Keys.Enter)  {  if (IsValidIp(toolStripTextBox2.Text))  {  AppConfiguration.SetAppConfig("ip", toolStripTextBox2.Text);  MessageBox.Show("변경되었습니다.");  }  else  {  toolStripTextBox2.Text = AppConfiguration.GetAppConfig("ip");  MessageBox.Show("올바른 IP 주소가 아닙니다.");  return;  }    }  }  public bool IsValidIp(string addr)  {  IPAddress ip;  bool valid = !string.IsNullOrEmpty(addr) && IPAddress.TryParse(addr, out ip);  return valid;  }  private void checkBox1\_CheckedChanged(object sender, EventArgs e)  {  if (chatting.Client == null)  {  txt\_Log.Text += "erro server is not connected.\n";  return;  }  if (((CheckBox)sender).Text == "err\_1")  {  Erro tempErro = new Erro(1);  erros.Add(tempErro);  chatting.SendMsg(tempErro.Erro\_String);  txt\_Log.Text += "erro\_1 Exeption\n";  }else if (((CheckBox)sender).Text == "err\_2")  {  Erro tempErro = new Erro(2);  erros.Add(tempErro);  chatting.SendMsg(tempErro.Erro\_String);  txt\_Log.Text += "erro\_2 Exeption\n";  }    }  private void button1\_Click\_1(object sender, EventArgs e)  {    }  }  } |
| using System;  using System.Collections.Generic;  using System.Configuration;  using System.Linq;  using System.Net.Sockets;  using System.Text;  using System.Threading.Tasks;  namespace MiniErp\_Client\_jsu  {  class Machine  {  private string name = AppConfiguration.GetAppConfig("name");  public string Name  {  get { return name; }  set { name = value; }  }  private string ip = AppConfiguration.GetAppConfig("ip");  public string Ip  {  get { return ip; }  set { ip = value; }  }  public Machine()  {  }    }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MiniErp\_Client\_jsu  {  class Erro  {  private string head;  public string Head  {  get { return head; }  set { head = value; }  }  private string erro\_String;  public string Erro\_String  {  get { return erro\_String; }  set { erro\_String = value; }  }  private int erro\_Code;  public int Erro\_Code  {  get { return erro\_Code; }  set { erro\_Code = value; }  }  public Erro(int err\_Code)  {  this.erro\_Code = err\_Code;  this.head = "[erro] ";  ErroSetting();  }  private void ErroSetting()  {  switch (this.erro\_Code)  {  case 1: this.erro\_String = "[erro]" + AppConfiguration.GetAppConfig("name") + "작업자 일시정지";break;  case 2: this.erro\_String = "[erro]" + AppConfiguration.GetAppConfig("name") + "라인 에러 발생";break;  default:  break;  }  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  // 커맨드 생성  // 올바른 커멘드인지 체크  // 커멘드 펑션 실행 ( 정지, 재시작 )  namespace MiniErp\_Client\_jsu  {  class Command  {  private string head; // [command]  public string Head  {  get { return head; }  set { head = value; }  }  private string name; // [pc1]  public string Name  {  get { return name; }  set { name = value; }  }  private string command\_Value; // 프로그램종료  public string Command\_Value  {  get { return command\_Value; }  set { command\_Value = value; }  }  private Chatting chatinfo;  private List<Barcode> barcodes;  public Command(string name, string command\_Head, string command\_Value,Chatting chatinfo,object barcodelist)  {  this.name = name;  this.head = command\_Head;  this.command\_Value = command\_Value;  this.chatinfo = chatinfo;  this.barcodes = (List<Barcode>)barcodelist;  }  public Command(string command\_Value)  {  this.command\_Value = command\_Value;  }    // 바코드 리스트를 보내는 메소드  public void BarcodeMsgMaker(List<Barcode> barcodes)  {  StringBuilder sb = new StringBuilder();  sb.AppendLine("---------[투입 현황]" + this.Name);  sb.AppendLine(NowTime());  foreach (var item in barcodes)  {  sb.AppendLine("\*[" + item.Barcode\_Code + "]" + "\t" + item.Barcode\_Count);  }  sb.AppendLine("----------------------------");  chatinfo.SendMsg(sb.ToString());  }  /// <summary>  /// 현재 시간의 스트링을 만드는 메서드  /// </summary>  private string NowTime()  {  DateTime dt = DateTime.Now;  return  "---------[" + dt.ToShortDateString() + " " + dt.Hour + ":" + dt.Minute + ":" + dt.Second + "]";  }  public void ChangeIp()  {  // [command][this.name][ip]192.168.0.8  if (new Form1().IsValidIp(this.command\_Value))  {  AppConfiguration.SetAppConfig("ip", this.command\_Value);  chatinfo.SendMsg(this.name + "ip change ok");  }  else  {  chatinfo.SendMsg(this.name + "ip change not ok");  }  }  public void ChangeName()  {  // [command][this.name][name]pc2  AppConfiguration.SetAppConfig("name", this.command\_Value);  }  public void CommandRunning()  {  if (this.command\_Value.Contains("[ip]")) // ip변경 커맨드  {  this.command\_Value = Command\_Value.Replace("[ip]", "");  ChangeIp();  return;  }  else if (this.command\_Value.Contains("[name]")) // 이름변경 커맨드  {  this.command\_Value = Command\_Value.Replace("[name]", "");  ChangeName();  Application.Restart(); // 이름변경후 재시작  return;  }  switch (this.command\_Value) // 커맨드 선택부  {  case "test\_module": System.Windows.Forms.MessageBox.Show("Test"); break;  case "exit": Application.Exit(); break;  case "restart": Application.Restart(); break;  case "barcode": BarcodeMsgMaker(barcodes); break;  default:  break;  }  }  }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Net.Sockets;  using System.Text;  using System.Threading;  using System.Threading.Tasks;  using System.Windows.Forms;  namespace MiniErp\_Client\_jsu  {  class Chatting  {  #region 멤버 변수  TcpClient client;  NetworkStream stream = default(NetworkStream); // 기본값 할당(해당 객체의 기본값 참조형은 null)  Thread thread; // 서버 쓰레드  string readData = null; // 서버의 메시지  private string ip;  private string name;  object barcodeList; // 바코드정보를 담는 리스트  List<Command> command = new List<Command>(); // 명령어 리스트  List<string> erro = new List<string>(); // 에러 리스트  public List<Command> Command { get { return command; } }  public List<string> Erro { get { return erro; } }  public TcpClient Client { get { return client; } }  #endregion  public Chatting(string ipAddr, string name, object barcodelist)  {  this.ip = ipAddr; this.name = name;  this.barcodeList = barcodelist;  }  /// <summary>  /// 머신 클라이언트를 시작합니다.  /// 머신 서버와 접속합니다..  /// </summary>  public bool Start()  {  IAsyncResult access = null;  try  {  if (client == null)  {  client = new TcpClient();  access = client.BeginConnect(this.ip, 4444, null, null);  var result = access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));  stream = client.GetStream();  SendMsg(this.name); // 접속클라이언트 이름 보냄  }  else if (client.Connected == false)  {  access = client.BeginConnect(this.ip, 4444, null, null);  var result = access.AsyncWaitHandle.WaitOne(TimeSpan.FromSeconds(1));  stream = client.GetStream();  }  if (thread == null)  {  thread = new Thread(getMsg);  thread.Start();  if (client.Connected)  SendMsg("[command]" + this.name + "is connecting"); //접속성공했다면 메시지  }  return true;  }  catch (Exception)  {  return false;  }  }  private void DisplayText(string text)  {  Byte[] outStream = Encoding.UTF8.GetBytes(text);  stream.Write(outStream, 0, outStream.Length);  stream.Flush();  }  /// <summary>  /// 서버가 보내오는 메시지를 수신합니다. 쓰레드호출 메서드  /// </summary>  private void getMsg()  {  while (true)  {  stream = client.GetStream();  Byte[] byteFrom = new byte[client.SendBufferSize];  stream.Read(byteFrom, 0, client.SendBufferSize);  readData = Encoding.UTF8.GetString(byteFrom);  readData = readData.Replace("\0", ""); // 바이트배열에 빈값(쓰레기값 제거)  CommandChacker(readData);// 올바른커맨드 판별  }  }  /// <summary>  /// 올바른 command 인지 인식합니다.  /// 올바르다면 list에 추가함  /// </summary>  /// <param name="readData">서버에서 msg</param>  private void CommandChacker(string readData)  {  if (readData.Contains("[command]") != true || readData.Contains(this.name) != true)  return;  else if (readData.Contains("접속") == true) // 서버접속시 접속이라고 보내기에 이를 무시  return;  string tempHead = readData.Substring(readData.IndexOf("[command]"), "[command]".Length);  string tempName = readData.Substring(readData.IndexOf(this.name), this.name.Length);  string tempValue = readData.Replace(tempHead, "").Replace(tempName, "").Replace("\0","").Trim();  // 임시커맨드 객체 생성후 리스트에 삽입, 그 후 커맨드 실행함  Command tempCommand = new Command(tempHead, tempName, tempValue, this, barcodeList);  command.Add(new Command(tempHead, tempName, tempValue, this, barcodeList));  tempCommand.CommandRunning();  }  /// <summary>  /// 메시지 보내기  /// </summary>  public void SendMsg(string msg)  {  stream = client.GetStream();  Byte[] byteFrom = Encoding.UTF8.GetBytes(msg);  stream.Write(byteFrom, 0, byteFrom.Length);  stream.Flush();  }  // 서버종료  public void CloseServer()  {  // ex) [command][pc1]is endconnecting  if (client.Connected != false)  {  byte[] msgTemp = Encoding.UTF8.GetBytes("[command]" + this.name + "is endconnecting");  stream = client.GetStream();  stream.Write(msgTemp, 0, msgTemp.Length);  stream.Flush();  client.Close();  stream.Close();  }    }    }  } |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MiniErp\_Client\_jsu  {  class Barcode  {  private string barcode\_Code;  public string Barcode\_Code  {  get { return barcode\_Code; }  set { barcode\_Code = value; }  }  private int barcode\_Count;  public Barcode(string barcode\_Code)  {  this.barcode\_Code = barcode\_Code;  this.barcode\_Count = 1;  }  public int Barcode\_Count  {  get { return barcode\_Count; }  set { barcode\_Count = value; }  }  }  } |
| using System;  using System.Collections.Generic;  using System.Configuration;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace MiniErp\_Client\_jsu  {  class AppConfiguration  {  public static string GetAppConfig(string key)  {  return ConfigurationManager.AppSettings[key];  }  public static void SetAppConfig(string key, string value)  {  Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);  KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;  cfgCollection.Remove(key);  cfgCollection.Add(key, value);  config.Save(ConfigurationSaveMode.Modified);  ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);  }  public static void AddAppConfig(string key, string value)  {  Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);  KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;  cfgCollection.Add(key, value);  config.Save(ConfigurationSaveMode.Modified);  ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);  }  public static void RemoveAppConfig(string key)  {  Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);  KeyValueConfigurationCollection cfgCollection = config.AppSettings.Settings;  try  {  cfgCollection.Remove(key);  config.Save(ConfigurationSaveMode.Modified);  ConfigurationManager.RefreshSection(config.AppSettings.SectionInformation.Name);  }  catch { }  }  }  } |
| using System;  using System.Collections.Generic;  using System.Diagnostics;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using BarcodeLib;  /\*  \* License : Apache-2.0 license  \* 해당 라이브러리 프로젝트 주소  \* https://github.com/barnhill/barcodelib  \*  \*/  namespace MiniERP.VO  {  class Barcode\_Module  {  /// <summary>  /// 바코드 이미지생성함  /// </summary>  /// <param name="bacodeValue">바코드 값</param>  /// <param name="lblChk">바코드에 코드번호 스위치</param>  /// <param name="size">바코드 사이즈</param>  /// <returns>바코드 이미지를 반환</returns>  public Image MakeBarcode(string bacodeValue,bool lblChk,Size size)  {  Debug.WriteLine("bacode : MakeBarcode is running");  Barcode code = new Barcode();  Image barcode\_img;  if (String.IsNullOrEmpty(bacodeValue)) // 바코드값이 널일경우  {  Debug.WriteLine("err\_bacode : bacodeValue is null or empty");  return null;  }  if (lblChk != false) // 바코드 라벨  {  code.IncludeLabel = lblChk;  code.LabelPosition = BarcodeLib.LabelPositions.BOTTOMCENTER;  }  try  {  code.RawData = bacodeValue; // 바코드 정보  code.Encode(TYPE.CODE128, bacodeValue, Color.Black, Color.White, size.Width, size.Height);  }  catch (Exception)  {  Debug.WriteLine("err\_bacode : bacode Encoding Exception");  throw;  }  barcode\_img = code.EncodedImage;  Debug.WriteLine("bacode : bacode image make finish");  return barcode\_img;  }    }    } |
| using System;  using System.Collections.Generic;  using System.ComponentModel;  using System.Data;  using System.Drawing;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  using System.Windows.Forms;  using MiniERP.Model.DAO;  using MiniERP.VO;  namespace MiniERP.View  {  public partial class Frm\_PrintDisplay : Form  {  Barcode\_Module barcode = new Barcode\_Module();  List<Item> items = new List<Item>();  public Frm\_PrintDisplay()  {  InitializeComponent();  saveFileDialog1.FileName = DateTime.Today.ToShortDateString() + "\_Barcodes";  for (int i = 1; i < 43; i++)  {  combo\_Count.Items.Add(i);  }  }  private void Frm\_PrintDisplay\_Load(object sender, EventArgs e)  {  items = new ItemDAO().GetItems("");  Display();  }  /// <summary>  /// 현재 클래스의 List를 이용해 DataGridView에 내용을 출력합니다.  /// </summary>  private void Display()  {  DataTable dataTable = new DataTable();  DataColumn[] dataColumns = new DataColumn[4]  {  new DataColumn("아이템코드"),  new DataColumn("아이템명"),  new DataColumn("규격"),  new DataColumn("단위")  };  dataTable.Columns.AddRange(dataColumns);  foreach (var item in items)  {  DataRow dataRow = dataTable.NewRow();  dataRow["아이템코드"] = item.Item\_code;  dataRow["아이템명"] = item.Item\_name;  dataRow["규격"] = item.Item\_standard;  dataRow["단위"] = item.Item\_unit;  dataTable.Rows.Add(dataRow);  }  dataGridView1.DataSource = dataTable;  for (int i = 0; i < dataGridView1.Columns.Count; i++)  {  dataGridView1.Columns[i].Width = dataGridView1.Size.Width / dataGridView1.Columns.Count-1;  }  }  private void dataGridView1\_CellContentDoubleClick(object sender, DataGridViewCellEventArgs e)  {  foreach (var item in items)  {  if (item.Item\_code == dataGridView1.SelectedRows[0].Cells["아이템코드"].Value.ToString())  {  pictureBox1.Image = null;  pictureBox1.Image = barcode.MakeBarcode(item.Item\_code, true, new Size(300, 50));  break;  }  }  }  private void btn\_Search\_Click(object sender, EventArgs e)  {  items = new ItemDAO().GetItems(txt\_Search.Text);  Display();  }  private void txt\_Search\_KeyDown(object sender, KeyEventArgs e)  {  if (e.KeyCode == Keys.Enter)  {  btn\_Search\_Click(null, null);  }  }  /// <summary>  /// 바코드 이미지 내보냅니다.  /// 출력 경로 지정할 것.  /// </summary>  private void btn\_Print\_Click(object sender, EventArgs e)  {  try  {  if (pictureBox1.Image == null)  {  MessageBox.Show("바코드를 선택하여 주세요");  return;  } // 이미지 예외분기  if (Int32.Parse(combo\_Count.Text) > 43)  {  MessageBox.Show("42개 이상불가능합니다.");  return;  } // 카운트 예외분기  else if (Int32.Parse(combo\_Count.Text) == 0)  {  MessageBox.Show("0은 입력이 불가능합니다.");  combo\_Count.Text = "1";  return;  }  if (saveFileDialog1.ShowDialog() != DialogResult.OK)  {  return;  }  #region 이미지 이어붙이기  Bitmap A4 = new Bitmap(1240, 1754); // a4 용지 크기  Size size = pictureBox1.Image.Size;  Image img = pictureBox1.Image;  Graphics g = Graphics.FromImage(A4);  int y = 0; int x = 0;  for (int i = 0; i < Int32.Parse(combo\_Count.Text); i++)  {  // 한 줄에 16개씩 찍히도록..  if (i == 14)  { x += 400; y = 0; }  else if (i == 28)  { x += 400; y = 0; }  else if (i == 42) // 3줄 ,갯수 42개 끝  { x += 400; y = 0; }  g.DrawImage(img, x, y, size.Width, size.Height);  y += 130;  }    A4.Save(saveFileDialog1.FileName);  MessageBox.Show("완료");  #endregion  }  catch (Exception)  {  MessageBox.Show("숫자만 입력해주세요");  }  }  }  } |

* 액티비티 다이어그램

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

* 실행 사진

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |