using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Visifire.Charts;

using Xceed.Wpf.AvalonDock;

using Xceed.Wpf.Toolkit;

using System.Data.OleDb;

using System.Data;

using Microsoft.Research.DynamicDataDisplay;

using Microsoft.Research.DynamicDataDisplay.DataSources;

using Microsoft.Win32;

namespace Monitor

{

/// <summary>

/// Interaction logic for HistoryStatic.xaml

/// </summary>

public partial class HistoryStatic : UserControl

{

StartPage wnd;

OleDbConnection con;

LineGraph line;//当前曲线对象

LineGraph line\_upper;//当前曲线上限对象

LineGraph line\_lower;//当前曲线下限对象

//数据源

CompositeDataSource composite;//组合的最终数据源

EnumerableDataSource<DateTime> x;//x轴数据源

EnumerableDataSource<double> y;//y轴数据源

List<DateTime> temp\_datelist;//暂时存放时间x轴数据

List<double> temp\_ylist;///暂时存放y轴数据

//upper数据源

CompositeDataSource composite\_upper;//组合的最终数据源

EnumerableDataSource<DateTime> x\_upper;//x轴数据源

EnumerableDataSource<double> y\_upper;//y轴数据源

List<DateTime> temp\_datelist\_upper;//暂时存放时间x轴数据

List<double> temp\_ylist\_upper;///暂时存放y轴数据

//lower数据源

CompositeDataSource composite\_lower;//组合的最终数据源

EnumerableDataSource<DateTime> x\_lower;//x轴数据源

EnumerableDataSource<double> y\_lower;//y轴数据源

List<DateTime> temp\_datelist\_lower;//暂时存放时间x轴数据

List<double> temp\_ylist\_lower;///暂时存放y轴数据

DataTable dt;//参数定义表格

public List<string> list = new List<string>();//下拉列表数据存储链表

int nowSelectIndex = 0;//当前下拉列表选择索引

//饼图数据

List<string> pie\_x ;

List<string> pie\_y ;

bool isFirst = true;

public HistoryStatic()

{

InitializeComponent();

wnd = (StartPage)Application.Current.MainWindow;//获取主窗口

con = wnd.con;//获取主链接

string strSelect = "select \* from tb\_lhf\_measure where saveindex is not null";

if (con.State == ConnectionState.Open)

{

OleDbCommand cmd = new OleDbCommand(strSelect, con);

OleDbDataAdapter adp = new OleDbDataAdapter(cmd);

DataSet set = new DataSet();

adp.Fill(set);

dt = set.Tables[0];

for (int i = 0; i < dt.Rows.Count; i++)

{

list.Add(Convert.ToString(dt.Rows[i].Field<object>("name")));

}

preSelectIndex.ItemsSource = list;//设置组合框数据源

preSelectIndex.SelectedIndex = nowSelectIndex;

temp\_datelist = new List<DateTime>();

temp\_ylist = new List<double>();

x = new EnumerableDataSource<DateTime>(temp\_datelist);

x.SetXMapping(x => xAxis.ConvertToDouble(x));

y = new EnumerableDataSource<double>(temp\_ylist);

y.SetYMapping(y => y);

composite = new CompositeDataSource(x, y);

line = plotter.AddLineGraph(composite, Colors.Green, 2, list.ElementAt(nowSelectIndex));//添加曲线

if (dt.Rows[nowSelectIndex].Field<object>("upper") != null)

{

temp\_datelist\_upper = new List<DateTime>();

temp\_ylist\_upper = new List<double>();

x\_upper = new EnumerableDataSource<DateTime>(temp\_datelist\_upper);

x\_upper.SetXMapping(x\_upper => xAxis.ConvertToDouble(x\_upper));

y\_upper = new EnumerableDataSource<double>(temp\_ylist\_upper);

y\_upper.SetYMapping(y\_upper => y\_upper);

composite\_upper = new CompositeDataSource(x\_upper, y\_upper);

Pen pen = new Pen();

pen.Brush = new SolidColorBrush(Colors.Red);

pen.DashStyle = new DashStyle(new double[] { 2, 5 }, 2);

pen.Thickness = 2;//新建虚线画笔

PenDescription description = new PenDescription("上限");

line\_upper = plotter.AddLineGraph(composite\_upper, pen, description);//添加上曲线

}

if (dt.Rows[nowSelectIndex].Field<object>("lower") != null)

{

temp\_datelist\_lower = new List<DateTime>();

temp\_ylist\_lower = new List<double>();

x\_lower = new EnumerableDataSource<DateTime>(temp\_datelist\_lower);

x\_lower.SetXMapping(x\_lower => xAxis.ConvertToDouble(x\_lower));

y\_lower = new EnumerableDataSource<double>(temp\_ylist\_lower);

y\_lower.SetYMapping(y\_lower => y\_lower);

composite\_lower = new CompositeDataSource(x\_lower, y\_lower);

Pen pen = new Pen();

pen.Brush = new SolidColorBrush(Colors.Red);

pen.DashStyle = new DashStyle(new double[] { 2, 5 }, 2);

pen.Thickness = 2;//新建虚线画笔

PenDescription description = new PenDescription("下限");

line\_upper = plotter.AddLineGraph(composite\_lower, pen, description);//添加下限曲线

}

pie\_x = new List<string>();

pie\_y = new List<string>();

pie\_x.Add("正常率");

pie\_x.Add("过高率");

pie\_x.Add("过低率");

pie\_y.Add("0");

pie\_y.Add("0");

pie\_y.Add("0");

chart.Width = 580;

chart.Height = 380;

CreateChartPie("", pie\_x, pie\_y);

// chart.Width = 20;

}

else

{

return;

}

}

public void CreateChartPie(string name, List<string> valuex, List<string> valuey)

{

//设置图标的宽度和高度

// chart.Width = 580;

//chart.Height = 380;

// chart.Margin = new Thickness(100, 5, 10, 5);

//是否启用打印和保持图片

chart.ToolBarEnabled = true;

//设置图标的属性

//chart.ScrollingEnabled = true;//是否启用或禁用滚动

chart.View3D = true;//3D效果显示

//创建一个标题的对象

Title title = new Title();

//设置标题的名称

title.Text = name;

// title.Padding = new Thickness(0, 10, 5, 0);

//向图标添加标题

chart.Titles.Clear();

chart.Titles.Add(title);

//Axis yAxis = new Axis();

////设置图标中Y轴的最小值永远为0

//yAxis.AxisMinimum = 0;

////设置图表中Y轴的后缀

//yAxis.Suffix = "斤";

//chart.AxesY.Add(yAxis);

// 创建一个新的数据线。

DataSeries dataSeries = new DataSeries();

// 设置数据线的格式

dataSeries.RenderAs = RenderAs.Pie;//柱状Stacked

// 设置数据点

DataPoint dataPoint;

for (int i = 0; i < valuex.Count; i++)

{

// 创建一个数据点的实例。

dataPoint = new DataPoint();

// 设置X轴点

dataPoint.AxisXLabel = valuex[i];

dataPoint.LegendText = "##" + valuex[i];

//设置Y轴点

dataPoint.YValue = double.Parse(valuey[i]);

//添加一个点击事件

//dataPoint.MouseLeftButtonDown += new MouseButtonEventHandler(dataPoint\_MouseLeftButtonDown);

//添加数据点

dataSeries.DataPoints.Add(dataPoint);

}

// 添加数据线到数据序列。

chart.Series.Add(dataSeries);

}

public void UpdateChartPie(string name, List<string> valuey)

{

for(int i=0;i<chart.Series[0].DataPoints.Count;i++)

{

chart.Series[0].DataPoints[i].YValue = double.Parse(valuey[i]);

}

chart.Titles[0].Text = name;

}

private void OnSerch(object sender, RoutedEventArgs e)

{

try

{

string name = preSelectIndex.SelectedValue.ToString();

if (line != null)

line.Description = new PenDescription(name);

int nowSaveIndex = Convert.ToInt32(dt.Rows[nowSelectIndex].Field<object>("saveindex"));//当前存储索引

string startStr = startTime.Text;

string endStr = endTime.Text;

if (startStr == null)

throw new Exception("请输入起始时间！");

if (endStr == null)

throw new Exception("请输入结束时间！");

string strSelect = "select \* from tb\_lhf\_data where cytime between '" + startStr + "' and '" + endStr + "'"+ " order by cytime asc";

string strSelectCorro = "select \* from tb\_lhf\_corro where cytime between '" + startStr + "' and '" + endStr + "'" + " and T=" + TControl.Text + " order by cytime asc";

if (con.State == ConnectionState.Open)

{

DataTable temp\_dt = GetTable(strSelect, con);

DataTable temp\_dtCorro = GetTable(strSelectCorro, con);

temp\_datelist.Clear();

temp\_ylist.Clear();

temp\_datelist\_upper.Clear();

temp\_ylist\_upper.Clear();

temp\_datelist\_lower.Clear();

temp\_ylist\_lower.Clear();

double upper = 999999;

double lower = -999999;

int dtCount = 0;

if (nowSelectIndex != 2)

dtCount = temp\_dt.Rows.Count;

else if (nowSelectIndex == 2)

dtCount = temp\_dtCorro.Rows.Count;

for (int i = 1; i <= dtCount; i++)

{

double temp\_data = 0;

string str = "";

if (nowSelectIndex != 2)

{

temp\_data = Convert.ToDouble(temp\_dt.Rows[i - 1].Field<object>("v" + nowSaveIndex));

str = Convert.ToString(temp\_dt.Rows[i - 1].Field<object>("cytime"));

}

else if (nowSelectIndex == 2)

{

temp\_data = Convert.ToDouble(temp\_dtCorro.Rows[i - 1].Field<object>("corro"));

str = Convert.ToString(temp\_dtCorro.Rows[i - 1].Field<object>("cytime"));

}

DateTime date = DateTime.Parse(str);

temp\_datelist.Add(date);

temp\_ylist.Add(temp\_data);

if (dt.Rows[nowSelectIndex].Field<object>("upper") != null)

{

upper = Convert.ToDouble(dt.Rows[nowSelectIndex].Field<object>("upper"));

temp\_datelist\_upper.Add(date);

temp\_ylist\_upper.Add(upper);

}

if (dt.Rows[nowSelectIndex].Field<object>("lower") != null)

{

lower = Convert.ToDouble(dt.Rows[nowSelectIndex].Field<object>("lower"));

temp\_datelist\_lower.Add(date);

temp\_ylist\_lower.Add(lower);

}

}

x.RaiseDataChanged();//更新数据源

y.RaiseDataChanged();

if (dt.Rows[nowSelectIndex].Field<object>("upper") != null)

{

Dispatcher.Invoke(new Action(() => { x\_upper.RaiseDataChanged(); }));//更新x轴数据源

Dispatcher.Invoke(new Action(() => { y\_upper.RaiseDataChanged(); }));//更新y轴数据源

}

if (dt.Rows[nowSelectIndex].Field<object>("lower") != null)

{

Dispatcher.Invoke(new Action(() => { x\_lower.RaiseDataChanged(); }));//更新x轴数据源

Dispatcher.Invoke(new Action(() => { x\_lower.RaiseDataChanged(); }));//更新y轴数据源

}

int NormalNum = 0;

int UpperNum = 0;

int LowerNum = 0;

for (int i = 0; i < dtCount; i++)

{

double temp\_data = 0;

if (nowSelectIndex != 2)

temp\_data = Convert.ToDouble(temp\_dt.Rows[i].Field<object>("v" + nowSaveIndex));//数据

else if (nowSelectIndex == 2)

temp\_data = Convert.ToDouble(temp\_dtCorro.Rows[i].Field<object>("corro"));//数据

if (temp\_data > upper)

UpperNum++;

else if (temp\_data < LowerNum)

LowerNum++;

else

NormalNum++;

}

pie\_y.Clear();

pie\_y.Add(((double)NormalNum / temp\_dt.Rows.Count).ToString());

pie\_y.Add(((double)UpperNum / temp\_dt.Rows.Count).ToString());

pie\_y.Add(((double)LowerNum / temp\_dt.Rows.Count).ToString());

UpdateChartPie(name, pie\_y);

}

}

catch(Exception ex)

{

System.Windows.MessageBox.Show(ex.Message);

}

}

private void OnSelectionChanged(object sender, SelectionChangedEventArgs e)

{

nowSelectIndex = preSelectIndex.SelectedIndex;

}

private void OnLoaded(object sender, RoutedEventArgs e)

{

}

private void OnGotFocus(object sender, RoutedEventArgs e)

{

// chart.Width = 200;

//double a= chart.Height;

}

private void OnSizeChanged(object sender, RoutedEventArgs e)

{

chart.Width = LowerWnd.ActualWidth;

chart.Height = (LowerWnd.ActualHeight-PieLable.ActualHeight)\*1;

// chart.Height =200;

}

private DataTable GetTable(string strSql, OleDbConnection con)

{

OleDbCommand cmd = new OleDbCommand(strSql, con);

DataSet set = new DataSet();

OleDbDataAdapter adpCorro = new OleDbDataAdapter(cmd);

adpCorro.Fill(set);

DataTable temp\_dt = set.Tables[0];//获取参数数据表格

return temp\_dt;

}

private void OnSave(object sender, RoutedEventArgs e)

{

try

{

string startStr = startTime.Text;

string endStr = endTime.Text;

if (startStr == null)

throw new Exception("请输入起始时间！");

if (endStr == null)

throw new Exception("请输入结束时间！");

string strSelect = "select \* from tb\_lhf\_data where cytime between '" + startStr + "' and '" + endStr + "'" + " order by cytime asc";

string strSelectCorro = "select \* from tb\_lhf\_corro where cytime between '" + startStr + "' and '" + endStr + "'" + " and T=" + TControl.Text + " order by cytime asc";

if (con.State == ConnectionState.Open)

{

DataTable temp\_dt = GetTable(strSelect, con);

DataTable temp\_dtCorro = GetTable(strSelectCorro, con);

//创建一个保存文件式的对话框

SaveFileDialog sfd = new SaveFileDialog();

//设置这个对话框的起始保存路径

sfd.InitialDirectory = System.Environment.CurrentDirectory+@"\";

//设置保存的文件的类型，注意过滤器的语法

sfd.Filter = "Excel文件|\*.xlsx";

//调用ShowDialog()方法显示该对话框，该方法的返回值代表用户是否点击了确定按钮

if (sfd.ShowDialog() == true)

{

OperateExcel operateExcel = new OperateExcel(sfd.FileName);

operateExcel.AddToSheet(temp\_dt);

operateExcel.AddToSheet(temp\_dtCorro);

System.Windows.MessageBox.Show("保存成功");

}

else

{

System.Windows.MessageBox.Show("取消保存");

}

}

}

catch(Exception ex)

{

System.Windows.MessageBox.Show(ex.Message);

}

// OperateExcel operateExcel = new OperateExcel();

}

}

}