//应用程序启动类

<Application x:Class="ATNET.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

StartupUri="Window1.xaml">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/Shared.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/ScrollBar.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/Expander.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/GroupBox.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/ToolTip.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/ScrollViewer.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/GlassButtonStyle.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/Canvas.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/ScrollViewerRuler.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Common/BlendStyleMenuItem.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Graph;component/Resources/DesignerItem.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Graph;component/Resources/ToolboxItem.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Graph;component/Resources/Toolbox.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Graph;component/Resources/Connection.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Customization/FlowChartStencils.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Customization/ShapeStencils.xaml"/>

<ResourceDictionary Source="pack://application:,,,/SoftArt.WPF.Styles;component/Customization/SymbolStencils.xaml"/>

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

//新建项目类

<Window

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" x:Class="ATNET.NewProjectWindow"

Title="新建项目" Height="300" Width="500"

WindowStartupLocation="CenterScreen" WindowStyle="SingleBorderWindow"

ResizeMode="NoResize" ShowInTaskbar="True" Topmost="True" mc:Ignorable="d">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="0.333\*"/>

<RowDefinition Height="0.403\*"/>

<RowDefinition Height="0.264\*"/>

</Grid.RowDefinitions>

<GroupBox x:Name="projTypeGroup" Header="项目类型" Height="150" VerticalAlignment="Top">

<TreeView x:Name="projTypeTree" VerticalAlignment="Top" Height="150">

<TreeViewItem Header="楼宇" Tag="楼宇的说明"/>

<TreeViewItem Header="小区" Tag="小区的说明"/>

<TreeViewItem Header="楼层" Tag="楼层的说明"/>

<TreeViewItem Header="机房" Tag="机房的说明"/>

<TreeViewItem Header="机柜" Tag="机柜的说明"/>

</TreeView>

</GroupBox>

<TextBox Grid.Row="1" Height="30" HorizontalAlignment="Stretch"

Text="{Binding ElementName=projTypeTree,Path=SelectedItem.Tag}" IsEnabled="False" Margin="0,5.833,0,5.791" d:LayoutOverrides="VerticalMargin"/>

<Grid Grid.Row="2" Margin="0">

<Grid.RowDefinitions>

<RowDefinition Height="24"/>

<RowDefinition Height="24"/>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Label Content="名称" Grid.Row="0" Grid.Column="0" Width="40" />

<TextBox x:Name="projName" Grid.Row="0" Grid.Column="1" Width="180" HorizontalAlignment="Left" />

<Label Content="位置" Grid.Row="1" Grid.Column="0" Width="40" />

<TextBox x:Name="projPath" Grid.Row="1" Grid.Column="1" Width="280" HorizontalAlignment="Left" />

<Button x:Name="btnBrower" Content="浏览" Grid.Row="1" Grid.Column="2" Width="80" Click="button\_Click"/>

</Grid>

<Canvas Grid.Row="3">

<Button x:Name="btnOK" Content="确定" Canvas.Right="100" Width="80" Click="button\_Click"/>

<Button x:Name="btnCancel" Content="取消" Canvas.Right="20" Width="80" Click="button\_Click"/>

</Canvas>

</Grid>

</Window>

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.Shapes;

using System.Windows.Forms;

using System.IO;

namespace ATNET

{

/// <summary>

/// NewProjectWindow.xaml 的交互逻辑

/// </summary>

public partial class NewProjectWindow : Window

{

public NewProjectWindow()

{

InitializeComponent();

projPath.Text = AppDomain.CurrentDomain.BaseDirectory + "projects";

}

private void button\_Click(object sender, RoutedEventArgs e)

{

var btn = sender as System.Windows.Controls.Button;

if (btn.Content.ToString() == "浏览")

{

FolderBrowserDialog dialog = new FolderBrowserDialog();

dialog.RootFolder = Environment.SpecialFolder.MyDocuments;

dialog.ShowDialog();

projPath.Text = dialog.SelectedPath;

}

else if (btn.Content.ToString() == "确定")

{

if (string.IsNullOrEmpty(projName.Text))

{

System.Windows.MessageBox.Show("请输入工程的名字!");

return;

}

else

{

//验证工程的名字是否合适

//MessageBox.Show("请输入工程的名字!");

//return;

}

if (string.IsNullOrEmpty(projPath.Text))

{

System.Windows.MessageBox.Show("请输入工程保存的路径!");

return;

}

else

{

if (!Directory.Exists(projPath.Text))

{

System.Windows.MessageBox.Show("输入的路径不正确，请重新输入!");

return;

}

}

//建立新工程，进入主界面

ATNetProject.SaveProjectFile(projName.Text, projPath.Text);

this.Close();

}

else

{

this.Close();

}

}

}

}

//启动窗口

<Window x:Class="ATNET.StartWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Title="ATNET" Height="300" Width="500" WindowStartupLocation="CenterOwner"

WindowStyle="SingleBorderWindow" ShowInTaskbar="False" Background="{x:Static SystemColors.WindowBrush}"

Topmost="True" ResizeMode="NoResize">

<Grid Margin="10">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<GroupBox Grid.Row="0" Header="选项">

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<RadioButton x:Name="openRecentRadio" Grid.Row="0" Content="最近打开的文件" IsChecked="True"/>

<ListView Grid.Row="1" Height="100" x:Name="recentProjectLv"/>

<RadioButton x:Name="openOtherRadion" Grid.Row="2" Content="打开其他文件"/>

<RadioButton Grid.Row="3" x:Name="newRadion" Content="建立新工程"/>

</Grid>

</GroupBox>

<StackPanel Orientation="Horizontal" Grid.Row="1">

<StackPanel.Resources>

<Style TargetType="{x:Type Button}">

<Setter Property="Width" Value="60"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Margin" Value="2"/>

</Style>

</StackPanel.Resources>

<Button Content="确定" x:Name="btnOK" Click="button\_Click"/>

<Button Content="取消" x:Name="btnCancel" Click="button\_Click"/>

<Button Content="帮助" x:Name="btnHelp" Click="button\_Click"/>

</StackPanel>

</Grid>

</Window>

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.Shapes;

using System.IO;

namespace ATNET {

/// <summary>

/// StartWindow.xaml 的交互逻辑

/// </summary>

public partial class StartWindow : Window

{

public StartWindow()

{

InitializeComponent();

InitRecentProjectFiles();

}

public static string OpenProjectFile {

get;

set;

}

private bool InitRecentProjectFiles() {

List<string> files = new List<string>();

files.Add("c:\\Windows\\1.atnprj");

files.Add("d:\\winodws\\2.atnprj");

foreach (string file in files) {

recentProjectLv.Items.Add(file);

}

return true;

}

private void button\_Click(object sender, RoutedEventArgs e)

{

var btn = sender as Button;

if (btn.Name == "btnOK")

{

if ((bool)openRecentRadio.IsChecked) //Open Recent Project

{

string fileName = recentProjectLv.SelectedItem as string;

if (!File.Exists(fileName)) {//project file not exist

MessageBox.Show("Project File is not existed!");

return;

}

OpenProjectFile = fileName;

}

else if ((bool)openOtherRadion.IsChecked) {//Open Other Project

System.Windows.Forms.OpenFileDialog openFileDialog = new System.Windows.Forms.OpenFileDialog();

openFileDialog.InitialDirectory = "projects";

openFileDialog.DefaultExt = "atnprj";

openFileDialog.Filter = "Atnet Project files (\*.atnprj)|\*.atnprj";

openFileDialog.ShowDialog();

OpenProjectFile = openFileDialog.FileName;

}

else {//new project

System.Windows.Forms.SaveFileDialog saveFileDialog = new System.Windows.Forms.SaveFileDialog();

saveFileDialog.InitialDirectory = "projects";

saveFileDialog.DefaultExt = "atnprj";

saveFileDialog.Filter = "Atnet Project files (\*.atnprj)|\*.atnprj";

saveFileDialog.ShowDialog();

ATNetProject.SaveProjectFile(saveFileDialog.FileName);

OpenProjectFile = saveFileDialog.FileName;

}

this.Close();

}

else if (btn.Name == "btnCancel")

{

this.Close();

}

else

{

//Help Document

}

}

}

}

//画布Document类

<ad:DocumentContent x:Class="ATNET.CanvasDocument"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:g="clr-namespace:SoftArt.WPF.Graph;assembly=SoftArt.WPF.Graph"

xmlns:ad="clr-namespace:AvalonDock;assembly=AvalonDock"

xmlns:local="clr-namespace:ATNET">

<ScrollViewer Style="{StaticResource scrollViewerRuler}" HorizontalScrollBarVisibility="Visible"

VerticalScrollBarVisibility="Visible" Margin="0" x:Name="mainScrollViewer">

<g:DesignerCanvas Width="2000" Height="2000" Background="{StaticResource canvasBrushResource}"

Margin="-1,-1,0,0" x:Name="mainCanvas" PreviewMouseMove="mainCanvas\_PreviewMouseMove">

</g:DesignerCanvas>

</ScrollViewer>

</ad:DocumentContent>

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 AvalonDock;

using SoftArt.WPF.Styles.Common;

namespace ATNET

{

/// <summary>

/// CanvasDocument.xaml 的交互逻辑

/// </summary>

public partial class CanvasDocument : DocumentContent

{

public CanvasDocument()

{

InitializeComponent();

}

private void mainCanvas\_PreviewMouseMove(object sender, MouseEventArgs e)

{

Point mousePoint = e.GetPosition(mainCanvas);

RulerRectangle horizontalRect = mainScrollViewer.Template.FindName("horizontalRuler", mainScrollViewer) as RulerRectangle;

//horizontalRect.RulerLine = mousePoint;

//horizontalRect.InvalidateVisual();

RulerRectangle verticalRect = mainScrollViewer.Template.FindName("verticalRuler", mainScrollViewer) as RulerRectangle;

//verticalRect.RulerLine = mousePoint;

//verticalRect.InvalidateVisual();

}

}

}

//Project工程类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Controls;

using System.Xml;

using System.Xml.Linq;

using AvalonDock;

namespace ATNET

{

/// <summary>

/// ATNET的工程类

/// </summary>

internal class ATNetProject

{

private string projectName;

private Guid projectGuid;

private ATNETProjectType projectType;

private string configFile;

private List<ProjectItem> projectItemList = new List<ProjectItem>();

protected CanvasDocument projectDocument;

private ProjectStatus status = ProjectStatus.None;

private TreeView itemTreeView = new TreeView();

private DockingManager projectDockingManager;

/// <summary>

/// Project的名字

/// </summary>

public string ProjectName

{

get { return this.projectName; }

set { this.projectName = value; }

}

/// <summary>

/// Project的GUID

/// </summary>

public Guid ProjectGuid

{

get { return this.projectGuid; }

}

/// <summary>

/// Project的类型

/// </summary>

public ATNETProjectType ProjectType

{

get { return this.projectType; }

set { this.projectType = value; }

}

/// <summary>

/// Project的配置文件

/// </summary>

public string ConfigFile

{

get { return this.configFile; }

set { this.configFile = value; }

}

/// <summary>

/// Project工程的子项

/// </summary>

public List<ProjectItem> ProjectItemList

{

get { return projectItemList; }

}

/// <summary>

/// Project工程包含的画布

/// </summary>

public CanvasDocument ProjectDocument

{

get { return projectDocument; }

set { projectDocument = value; }

}

/// <summary>

/// 工程当前的状态

/// </summary>

public ProjectStatus Status

{

get { return status; }

set { status = value; }

}

/// <summary>

/// 工程的TreeView视图

/// </summary>

public TreeView ItemTreeView

{

get { return itemTreeView; }

}

public DockingManager ProjectDockingManager

{

get { return projectDockingManager; }

set { projectDockingManager = value; }

}

public static ATNetProject ProjectInstance

{

get { return instance; }

}

private static object helper = new object();

private static ATNetProject instance = null;

private ATNetProject(string name, CanvasDocument projectDocument)

{

this.projectGuid = Guid.NewGuid();

this.projectName = name;

this.projectDocument = projectDocument;

this.projectDocument.Title = "工程'" + name + "'";

this.projectDocument.Name = name;

TreeViewItem baseItem = new TreeViewItem();

baseItem.Header = "工程'" + name + "'";

baseItem.Name = name;

this.itemTreeView.Items.Add(baseItem);

this.AddNewProjectItemCompletedEvent += new AddNewProjectItemCompletedEventHandler(ATNetProject\_AddNewDocumentCompletedEvent);

}

protected void ATNetProject\_AddNewDocumentCompletedEvent(object sender, ProjectItem newItem)

{

TreeViewItem item = itemTreeView.Items[0] as TreeViewItem;

TreeViewItem addItem = new TreeViewItem();

addItem.Header = newItem.ItemName;

int index = item.Items.Add(addItem);

addItem.IsSelected = true;

}

/// <summary>

/// 建立一个新的工程

/// </summary>

/// <param name="name">工程名称</param>

/// <param name="projectDocument">工程所对应的Canvas</param>

/// <returns></returns>

public static ATNetProject GetProjectInstance(string name, CanvasDocument projectDocument)

{

lock (helper)

{

if (instance == null)

{

lock (helper)

{

instance = new ATNetProject(name, projectDocument);

return instance;

}

}

}

return instance;

}

/// <summary>

/// 添加新的工程子项的委托

/// </summary>

/// <param name="sender"></param>

/// <param name="newItem"></param>

private delegate void AddNewProjectItemCompletedEventHandler(object sender, ProjectItem newItem);

/// <summary>

/// 添加新的工程子项的事件

/// </summary>

private event AddNewProjectItemCompletedEventHandler AddNewProjectItemCompletedEvent;

protected void OnAddNewProjectItemCompleted(ProjectItem newItem)

{

if (AddNewProjectItemCompletedEvent != null)

{

AddNewProjectItemCompletedEvent(this, newItem);

}

}

/// <summary>

/// 添加新的工程子项

/// </summary>

/// <param name="newItem"></param>

/// <returns></returns>

public bool AddProjectItem(ProjectItem newItem)

{

projectItemList.Add(newItem);

OnAddNewProjectItemCompleted(newItem);

return true;

}

public static bool SaveProjectFile(string saveFielName)

{

int index = saveFielName.LastIndexOf("\\");

string fileName = saveFielName.Substring(index + 1);

int index1 = fileName.IndexOf(".");

fileName = fileName.Substring(0, index1);

ATNetProject project = GetProjectInstance(fileName, new CanvasDocument());

XElement xElement = new XElement("Project",

new XElement("Name", project.ProjectName),

new XElement("GUID", project.ProjectGuid),

new XElement("ProjectType", project.ProjectType),

new XElement("ProjectItems", ""));

xElement.Save(saveFielName);

project.Status = ProjectStatus.New;

return true;

}

/// <summary>

/// 保存工程文件

/// </summary>

/// <param name="saveFielName">保存工程文件的文件名</param>

/// <param name="fielPath">保存的路径</param>

/// <returns>保存是否成功</returns>

public static bool SaveProjectFile(string saveFielName, string fielPath)

{

ATNetProject project = GetProjectInstance(saveFielName, new CanvasDocument());

XElement xElement = new XElement("Project",

new XElement("Name", project.ProjectName),

new XElement("GUID", project.ProjectGuid),

new XElement("ProjectType", project.ProjectType),

new XElement("ProjectItems", ""));

xElement.Save(fielPath + "\\" + saveFielName + ".atnprj");

project.Status = ProjectStatus.New;

return true;

}

/// <summary>

/// 打开工程文件

/// </summary>

/// <param name="openFielName">打开工程的文件</param>

/// <returns>打开是否成功</returns>

public static bool OpenProjectFile(string openFielName)

{

XElement root = XElement.Load(@openFielName);

IEnumerable<XElement> element = from el in root.Elements() select el;

ATNetProject project = GetProjectInstance(element.Where(el => el.Name == "Name").First<XElement>().Value, new CanvasDocument());

project.projectGuid = new Guid(element.Where(el => el.Name == "GUID").First<XElement>().Value);

project.projectType = GetProjectType(element.Where(el => el.Name == "ProjectType").First<XElement>().Value);

return true;

}

private static ATNETProjectType GetProjectType(string type)

{

if (type == "StartFromBuilding")

{

return ATNETProjectType.StartFromBuilding;

}

else if (type == "StartFromBuilingFloor")

{

return ATNETProjectType.StartFromBuilingFloor;

}

else if (type == "StartFromPark")

{

return ATNETProjectType.StartFromPark;

}

else if (type == "StartFromRoom")

{

return ATNETProjectType.StartFromRoom;

}

return ATNETProjectType.Other;

}

public static bool AddNewProjectItem()

{

return true;

}

}

/// <summary>

/// ATNET工程的类型

/// </summary>

internal enum ATNETProjectType

{

/// <summary>

/// 从楼宇开始

/// </summary>

StartFromBuilding,

/// <summary>

/// 从楼层开始

/// </summary>

StartFromBuilingFloor,

/// <summary>

/// 从园区开始

/// </summary>

StartFromPark,

/// <summary>

/// 从机房开始

/// </summary>

StartFromRoom,

/// <summary>

/// 其他

/// </summary>

Other

}

/// <summary>

/// 工程的状态

/// </summary>

internal enum ProjectStatus

{

/// <summary>

/// 新建工程

/// </summary>

New,

/// <summary>

/// 打开工程

/// </summary>

Open,

/// <summary>

/// 关闭工程

/// </summary>

Close,

/// <summary>

/// 删除工程

/// </summary>

Delete,

/// <summary>

/// 正在使用工程

/// </summary>

Work,

/// <summary>

/// 没有操作

/// </summary>

None

}

}

//坐标转换类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Data;

using System.Globalization;

namespace ATNET

{

public class CanvasMousePointConverter:IValueConverter

{

public object Convert(object value, Type targertType, object parameter, CultureInfo culture)

{

if (value is double)

{

if ((double)value > 0)

{

return (int)((double)value);

}

}

return 0;

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

}

//ProjectItem工程子项类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ATNET

{

/// <summary>

/// 工程的子项类

/// </summary>

internal class ProjectItem

{

private string itemName;

private Guid itemGuid;

private CanvasDocument itemDocument;

private ProjectItemType itemType;

private List<ProjectItem> projectItemList = new List<ProjectItem>();

/// <summary>

/// 子项的名称

/// </summary>

public string ItemName

{

get { return itemName; }

set { itemName = value; }

}

/// <summary>

/// 子项的GUID

/// </summary>

public Guid ItemGuid

{

get { return itemGuid; }

}

/// <summary>

/// 工程子项的类别

/// </summary>

public ProjectItemType ItemType

{

get { return itemType; }

set { itemType = value; }

}

/// <summary>

/// 工程子项的子项

/// </summary>

public List<ProjectItem> ProjectItemList

{

get { return projectItemList; }

}

/// <summary>

/// 子项的画布

/// </summary>

public CanvasDocument ItemDocument

{

get { return itemDocument; }

}

public ProjectItem(string name, CanvasDocument itemDocument)

{

this.itemName = name;

this.itemDocument = itemDocument;

this.itemGuid = new Guid();

}

public bool AddNewProjectItem(ProjectItem newItem)

{

projectItemList.Add(newItem);

return true;

}

}

internal enum ProjectItemType

{

/// <summary>

/// 楼宇

/// </summary>

Building,

/// <summary>

/// 楼层

/// </summary>

Floor,

/// <summary>

/// 机房

/// </summary>

Room,

/// <summary>

/// 机柜

/// </summary>

Cabinet

}

}

//实体类

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存线缆信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存线缆信息

/// </summary>

[Serializable(),Description("Primary:CableID")]

public class tblCables

{

#region 构造函数

/// <summary>

/// 实体 保存线缆信息

/// </summary>

public tblCables(){}

#endregion

#region 私有变量

private string \_cableid;

private string \_catypeid;

private string \_cablename;

private string \_cablecode;

private string \_cableproperty;

private string \_shapeproperty;

private string \_equipoutput;

private string \_equipinput;

private string \_cablerouting;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 CableID (必填字段)

/// </summary>

[DataObjectField(true)]

public string CableID

{

set{ \_cableid=value;}

get{return \_cableid;}

}

/// <summary>

/// CaTypeID

/// </summary>

[DataObjectField(false)]

public string CaTypeID

{

set{ \_catypeid=value;}

get{return \_catypeid;}

}

/// <summary>

/// CableName (必填字段)

/// </summary>

[DataObjectField(false)]

public string CableName

{

set{ \_cablename=value;}

get{return \_cablename;}

}

/// <summary>

/// CableCode

/// </summary>

[DataObjectField(false)]

public string CableCode

{

set{ \_cablecode=value;}

get{return \_cablecode;}

}

/// <summary>

/// CableProperty

/// </summary>

[DataObjectField(false)]

public string CableProperty

{

set{ \_cableproperty=value;}

get{return \_cableproperty;}

}

/// <summary>

/// ShapeProperty

/// </summary>

[DataObjectField(false)]

public string ShapeProperty

{

set{ \_shapeproperty=value;}

get{return \_shapeproperty;}

}

/// <summary>

/// EquipOutput

/// </summary>

[DataObjectField(false)]

public string EquipOutput

{

set{ \_equipoutput=value;}

get{return \_equipoutput;}

}

/// <summary>

/// EquipInput (必填字段)

/// </summary>

[DataObjectField(false)]

public string EquipInput

{

set{ \_equipinput=value;}

get{return \_equipinput;}

}

/// <summary>

/// CableRouting

/// </summary>

[DataObjectField(false)]

public string CableRouting

{

set{ \_cablerouting=value;}

get{return \_cablerouting;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 线缆类型

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 线缆类型

/// </summary>

[Serializable(),Description("Primary:CaTypeID")]

public class tblCableType

{

#region 构造函数

/// <summary>

/// 实体 线缆类型

/// </summary>

public tblCableType(){}

#endregion

#region 私有变量

private string \_catypeid;

private string \_catypename;

private string \_catypecode;

private string \_catypeproperty;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 CaTypeID (必填字段)

/// </summary>

[DataObjectField(true)]

public string CaTypeID

{

set{ \_catypeid=value;}

get{return \_catypeid;}

}

/// <summary>

/// CaTypeName (必填字段)

/// </summary>

[DataObjectField(false)]

public string CaTypeName

{

set{ \_catypename=value;}

get{return \_catypename;}

}

/// <summary>

/// CaTypeCode

/// </summary>

[DataObjectField(false)]

public string CaTypeCode

{

set{ \_catypecode=value;}

get{return \_catypecode;}

}

/// <summary>

/// CaTypeProperty

/// </summary>

[DataObjectField(false)]

public string CaTypeProperty

{

set{ \_catypeproperty=value;}

get{return \_catypeproperty;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存公司信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存公司信息

/// </summary>

[Serializable(),Description("Primary:CompanyID")]

public class tblCompany

{

#region 构造函数

/// <summary>

/// 实体 保存公司信息

/// </summary>

public tblCompany(){}

#endregion

#region 私有变量

private string \_companyid;

private string \_superior;

private string \_companycode;

private string \_companyname;

private string \_adress;

private string \_telephone;

private string \_isvalid;

private string \_creator;

private DateTime \_createdate;

private string \_remark;

#endregion

#region 公共属性

/// <summary>

/// 主键 CompanyID (必填字段)

/// </summary>

[DataObjectField(true)]

public string CompanyID

{

set{ \_companyid=value;}

get{return \_companyid;}

}

/// <summary>

/// Superior

/// </summary>

[DataObjectField(false)]

public string Superior

{

set{ \_superior=value;}

get{return \_superior;}

}

/// <summary>

/// CompanyCode

/// </summary>

[DataObjectField(false)]

public string CompanyCode

{

set{ \_companycode=value;}

get{return \_companycode;}

}

/// <summary>

/// CompanyName (必填字段)

/// </summary>

[DataObjectField(false)]

public string CompanyName

{

set{ \_companyname=value;}

get{return \_companyname;}

}

/// <summary>

/// Adress

/// </summary>

[DataObjectField(false)]

public string Adress

{

set{ \_adress=value;}

get{return \_adress;}

}

/// <summary>

/// Telephone

/// </summary>

[DataObjectField(false)]

public string Telephone

{

set{ \_telephone=value;}

get{return \_telephone;}

}

/// <summary>

/// IsValid (必填字段)

/// </summary>

[DataObjectField(false)]

public string IsValid

{

set{ \_isvalid=value;}

get{return \_isvalid;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// remark

/// </summary>

[DataObjectField(false)]

public string remark

{

set{ \_remark=value;}

get{return \_remark;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 容器信息

容器是广义的 包括 城市、园区、楼宇、楼层、机房、机柜等

自连接上一级

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 容器信息

/// 容器是广义的 包括 城市、园区、楼宇、楼层、机房、机柜等

/// 自连接上一级

/// </summary>

[Serializable(),Description("Primary:ContainerID")]

public class tblContainers

{

#region 构造函数

/// <summary>

/// 实体 容器信息

/// 容器是广义的 包括 城市、园区、楼宇、楼层、机房、机柜等

/// 自连接上一级

/// </summary>

public tblContainers(){}

#endregion

#region 私有变量

private string \_containerid;

private string \_companyid;

private string \_ctypeid;

private string \_father;

private string \_containername;

private string \_containercode;

private string \_containerproperty;

private string \_shapeproperty;

private string \_creator;

private DateTime \_createdate;

#endregion

#region 公共属性

/// <summary>

/// 主键 ContainerID (必填字段)

/// </summary>

[DataObjectField(true)]

public string ContainerID

{

set{ \_containerid=value;}

get{return \_containerid;}

}

/// <summary>

/// CompanyID

/// </summary>

[DataObjectField(false)]

public string CompanyID

{

set{ \_companyid=value;}

get{return \_companyid;}

}

/// <summary>

/// CTypeID

/// </summary>

[DataObjectField(false)]

public string CTypeID

{

set{ \_ctypeid=value;}

get{return \_ctypeid;}

}

/// <summary>

/// Father

/// </summary>

[DataObjectField(false)]

public string Father

{

set{ \_father=value;}

get{return \_father;}

}

/// <summary>

/// ContainerName (必填字段)

/// </summary>

[DataObjectField(false)]

public string ContainerName

{

set{ \_containername=value;}

get{return \_containername;}

}

/// <summary>

/// ContainerCode

/// </summary>

[DataObjectField(false)]

public string ContainerCode

{

set{ \_containercode=value;}

get{return \_containercode;}

}

/// <summary>

/// ContainerProperty

/// </summary>

[DataObjectField(false)]

public string ContainerProperty

{

set{ \_containerproperty=value;}

get{return \_containerproperty;}

}

/// <summary>

/// ShapeProperty

/// </summary>

[DataObjectField(false)]

public string ShapeProperty

{

set{ \_shapeproperty=value;}

get{return \_shapeproperty;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存容器的类型

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存容器的类型

/// </summary>

[Serializable(),Description("Primary:CTypeID")]

public class tblContainerType

{

#region 构造函数

/// <summary>

/// 实体 保存容器的类型

/// </summary>

public tblContainerType(){}

#endregion

#region 私有变量

private string \_ctypeid;

private string \_ctypename;

private string \_ctypecode;

private string \_ctypeproperty;

private string \_creator;

private DateTime \_createdate;

#endregion

#region 公共属性

/// <summary>

/// 主键 CTypeID (必填字段)

/// </summary>

[DataObjectField(true)]

public string CTypeID

{

set{ \_ctypeid=value;}

get{return \_ctypeid;}

}

/// <summary>

/// CTypeName (必填字段)

/// </summary>

[DataObjectField(false)]

public string CTypeName

{

set{ \_ctypename=value;}

get{return \_ctypename;}

}

/// <summary>

/// CTypeCode (必填字段)

/// </summary>

[DataObjectField(false)]

public string CTypeCode

{

set{ \_ctypecode=value;}

get{return \_ctypecode;}

}

/// <summary>

/// CTypeProperty (必填字段)

/// </summary>

[DataObjectField(false)]

public string CTypeProperty

{

set{ \_ctypeproperty=value;}

get{return \_ctypeproperty;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 tblDataDictionary

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 tblDataDictionary

/// </summary>

[Serializable(),Description("Primary:DDID")]

public class tblDataDictionary

{

#region 构造函数

/// <summary>

/// 实体 tblDataDictionary

/// </summary>

public tblDataDictionary(){}

#endregion

#region 私有变量

private string \_ddid;

private string \_ddtype;

private string \_ddcode;

private string \_ddname;

#endregion

#region 公共属性

/// <summary>

/// 主键 DDID (必填字段)

/// </summary>

[DataObjectField(true)]

public string DDID

{

set{ \_ddid=value;}

get{return \_ddid;}

}

/// <summary>

/// DDType (必填字段)

/// </summary>

[DataObjectField(false)]

public string DDType

{

set{ \_ddtype=value;}

get{return \_ddtype;}

}

/// <summary>

/// DDCode (必填字段)

/// </summary>

[DataObjectField(false)]

public string DDCode

{

set{ \_ddcode=value;}

get{return \_ddcode;}

}

/// <summary>

/// DDName (必填字段)

/// </summary>

[DataObjectField(false)]

public string DDName

{

set{ \_ddname=value;}

get{return \_ddname;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存设备信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存设备信息

/// </summary>

[Serializable(),Description("Primary:EquipID")]

public class tblEuipments

{

#region 构造函数

/// <summary>

/// 实体 保存设备信息

/// </summary>

public tblEuipments(){}

#endregion

#region 私有变量

private string \_equipid;

private string \_containerid;

private string \_etypeid;

private string \_equipname;

private string \_equipcode;

private string \_equipproperty;

private string \_shapeproperty;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 EquipID (必填字段)

/// </summary>

[DataObjectField(true)]

public string EquipID

{

set{ \_equipid=value;}

get{return \_equipid;}

}

/// <summary>

/// ContainerID

/// </summary>

[DataObjectField(false)]

public string ContainerID

{

set{ \_containerid=value;}

get{return \_containerid;}

}

/// <summary>

/// ETypeID

/// </summary>

[DataObjectField(false)]

public string ETypeID

{

set{ \_etypeid=value;}

get{return \_etypeid;}

}

/// <summary>

/// EquipName (必填字段)

/// </summary>

[DataObjectField(false)]

public string EquipName

{

set{ \_equipname=value;}

get{return \_equipname;}

}

/// <summary>

/// EquipCode

/// </summary>

[DataObjectField(false)]

public string EquipCode

{

set{ \_equipcode=value;}

get{return \_equipcode;}

}

/// <summary>

/// EquipProperty

/// </summary>

[DataObjectField(false)]

public string EquipProperty

{

set{ \_equipproperty=value;}

get{return \_equipproperty;}

}

/// <summary>

/// ShapeProperty

/// </summary>

[DataObjectField(false)]

public string ShapeProperty

{

set{ \_shapeproperty=value;}

get{return \_shapeproperty;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 tblEuipType

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 tblEuipType

/// </summary>

[Serializable(),Description("Primary:ETypeID")]

public class tblEuipType

{

#region 构造函数

/// <summary>

/// 实体 tblEuipType

/// </summary>

public tblEuipType(){}

#endregion

#region 私有变量

private string \_etypeid;

private string \_etypename;

private string \_etypecode;

private string \_etypeproperty;

private string \_creator;

private DateTime \_createdate;

#endregion

#region 公共属性

/// <summary>

/// 主键 ETypeID (必填字段)

/// </summary>

[DataObjectField(true)]

public string ETypeID

{

set{ \_etypeid=value;}

get{return \_etypeid;}

}

/// <summary>

/// ETypeName (必填字段)

/// </summary>

[DataObjectField(false)]

public string ETypeName

{

set{ \_etypename=value;}

get{return \_etypename;}

}

/// <summary>

/// ETypeCode (必填字段)

/// </summary>

[DataObjectField(false)]

public string ETypeCode

{

set{ \_etypecode=value;}

get{return \_etypecode;}

}

/// <summary>

/// ETypeProperty (必填字段)

/// </summary>

[DataObjectField(false)]

public string ETypeProperty

{

set{ \_etypeproperty=value;}

get{return \_etypeproperty;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存固有的字段名称 比如：IP 、Manufacturer等

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存固有的字段名称 比如：IP 、Manufacturer等

/// </summary>

[Serializable(),Description("Primary:InnerColID")]

public class tblInnerColumn

{

#region 构造函数

/// <summary>

/// 实体 保存固有的字段名称 比如：IP 、Manufacturer等

/// </summary>

public tblInnerColumn(){}

#endregion

#region 私有变量

private string \_innercolid;

private string \_innercolname;

private string \_innercolproperty;

private string \_creator;

private DateTime \_createdate;

#endregion

#region 公共属性

/// <summary>

/// 主键 InnerColID (必填字段)

/// </summary>

[DataObjectField(true)]

public string InnerColID

{

set{ \_innercolid=value;}

get{return \_innercolid;}

}

/// <summary>

/// InnerColName (必填字段)

/// </summary>

[DataObjectField(false)]

public string InnerColName

{

set{ \_innercolname=value;}

get{return \_innercolname;}

}

/// <summary>

/// InnerColProperty (必填字段)

/// </summary>

[DataObjectField(false)]

public string InnerColProperty

{

set{ \_innercolproperty=value;}

get{return \_innercolproperty;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存标识信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存标识信息

/// </summary>

[Serializable(),Description("Primary:LabelID")]

public class tblLabels

{

#region 构造函数

/// <summary>

/// 实体 保存标识信息

/// </summary>

public tblLabels(){}

#endregion

#region 私有变量

private string \_labelid;

private string \_cableid;

private string \_ltypeid;

private string \_labelcode;

private string \_labelcontent;

private string \_isprint;

private string \_printer;

private DateTime \_printdate;

private Int32 \_printcount;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 LabelID (必填字段)

/// </summary>

[DataObjectField(true)]

public string LabelID

{

set{ \_labelid=value;}

get{return \_labelid;}

}

/// <summary>

/// CableID

/// </summary>

[DataObjectField(false)]

public string CableID

{

set{ \_cableid=value;}

get{return \_cableid;}

}

/// <summary>

/// LTypeID

/// </summary>

[DataObjectField(false)]

public string LTypeID

{

set{ \_ltypeid=value;}

get{return \_ltypeid;}

}

/// <summary>

/// LabelCode (必填字段)

/// </summary>

[DataObjectField(false)]

public string LabelCode

{

set{ \_labelcode=value;}

get{return \_labelcode;}

}

/// <summary>

/// LabelContent (必填字段)

/// </summary>

[DataObjectField(false)]

public string LabelContent

{

set{ \_labelcontent=value;}

get{return \_labelcontent;}

}

/// <summary>

/// IsPrint

/// </summary>

[DataObjectField(false)]

public string IsPrint

{

set{ \_isprint=value;}

get{return \_isprint;}

}

/// <summary>

/// Printer

/// </summary>

[DataObjectField(false)]

public string Printer

{

set{ \_printer=value;}

get{return \_printer;}

}

/// <summary>

/// PrintDate

/// </summary>

[DataObjectField(false)]

public DateTime PrintDate

{

set{ \_printdate=value;}

get{return \_printdate;}

}

/// <summary>

/// PrintCount (必填字段)

/// </summary>

[DataObjectField(false)]

public Int32 PrintCount

{

set{ \_printcount=value;}

get{return \_printcount;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 标识类型

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 标识类型

/// </summary>

[Serializable(),Description("Primary:LTypeID")]

public class tblLabelType

{

#region 构造函数

/// <summary>

/// 实体 标识类型

/// </summary>

public tblLabelType(){}

#endregion

#region 私有变量

private string \_ltypeid;

private string \_ltypename;

private string \_ltypecode;

private string \_shapeproperty;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 LTypeID (必填字段)

/// </summary>

[DataObjectField(true)]

public string LTypeID

{

set{ \_ltypeid=value;}

get{return \_ltypeid;}

}

/// <summary>

/// LTypeName (必填字段)

/// </summary>

[DataObjectField(false)]

public string LTypeName

{

set{ \_ltypename=value;}

get{return \_ltypename;}

}

/// <summary>

/// LTypeCode

/// </summary>

[DataObjectField(false)]

public string LTypeCode

{

set{ \_ltypecode=value;}

get{return \_ltypecode;}

}

/// <summary>

/// ShapeProperty

/// </summary>

[DataObjectField(false)]

public string ShapeProperty

{

set{ \_shapeproperty=value;}

get{return \_shapeproperty;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 tblLogs

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 tblLogs

/// </summary>

[Serializable(),Description("Primary:LogID")]

public class tblLogs

{

#region 构造函数

/// <summary>

/// 实体 tblLogs

/// </summary>

public tblLogs(){}

#endregion

#region 私有变量

private string \_logid;

private string \_logcode;

private string \_logtype;

private string \_logresult;

private string \_logtable;

private string \_logmodule;

private string \_creator;

private DateTime \_createdate;

#endregion

#region 公共属性

/// <summary>

/// 主键 LogID (必填字段)

/// </summary>

[DataObjectField(true)]

public string LogID

{

set{ \_logid=value;}

get{return \_logid;}

}

/// <summary>

/// LogCode (必填字段)

/// </summary>

[DataObjectField(false)]

public string LogCode

{

set{ \_logcode=value;}

get{return \_logcode;}

}

/// <summary>

/// LogType (必填字段)

/// </summary>

[DataObjectField(false)]

public string LogType

{

set{ \_logtype=value;}

get{return \_logtype;}

}

/// <summary>

/// LogResult (必填字段)

/// </summary>

[DataObjectField(false)]

public string LogResult

{

set{ \_logresult=value;}

get{return \_logresult;}

}

/// <summary>

/// LogTable

/// </summary>

[DataObjectField(false)]

public string LogTable

{

set{ \_logtable=value;}

get{return \_logtable;}

}

/// <summary>

/// LogModule

/// </summary>

[DataObjectField(false)]

public string LogModule

{

set{ \_logmodule=value;}

get{return \_logmodule;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 模块信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 模块信息

/// </summary>

[Serializable(),Description("Primary:ModuleID")]

public class tblModule

{

#region 构造函数

/// <summary>

/// 实体 模块信息

/// </summary>

public tblModule(){}

#endregion

#region 私有变量

private string \_moduleid;

private string \_superior;

private string \_modulename;

private string \_moduleurl;

private string \_creator;

private DateTime \_createdate;

private string \_remark;

#endregion

#region 公共属性

/// <summary>

/// 主键 ModuleID (必填字段)

/// </summary>

[DataObjectField(true)]

public string ModuleID

{

set{ \_moduleid=value;}

get{return \_moduleid;}

}

/// <summary>

/// Superior

/// </summary>

[DataObjectField(false)]

public string Superior

{

set{ \_superior=value;}

get{return \_superior;}

}

/// <summary>

/// ModuleName (必填字段)

/// </summary>

[DataObjectField(false)]

public string ModuleName

{

set{ \_modulename=value;}

get{return \_modulename;}

}

/// <summary>

/// ModuleURL (必填字段)

/// </summary>

[DataObjectField(false)]

public string ModuleURL

{

set{ \_moduleurl=value;}

get{return \_moduleurl;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// remark

/// </summary>

[DataObjectField(false)]

public string remark

{

set{ \_remark=value;}

get{return \_remark;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存角色权限

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存角色权限

/// </summary>

[Serializable(),Description("Primary:RoleRightID")]

public class tblRoleRight

{

#region 构造函数

/// <summary>

/// 实体 保存角色权限

/// </summary>

public tblRoleRight(){}

#endregion

#region 私有变量

private string \_rolerightid;

private string \_moduleid;

private string \_roleid;

private DateTime \_createdate;

private string \_creator;

#endregion

#region 公共属性

/// <summary>

/// 主键 RoleRightID (必填字段)

/// </summary>

[DataObjectField(true)]

public string RoleRightID

{

set{ \_rolerightid=value;}

get{return \_rolerightid;}

}

/// <summary>

/// ModuleID

/// </summary>

[DataObjectField(false)]

public string ModuleID

{

set{ \_moduleid=value;}

get{return \_moduleid;}

}

/// <summary>

/// RoleID

/// </summary>

[DataObjectField(false)]

public string RoleID

{

set{ \_roleid=value;}

get{return \_roleid;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存用户角色

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存用户角色

/// </summary>

[Serializable(),Description("Primary:RoleID")]

public class tblRoles

{

#region 构造函数

/// <summary>

/// 实体 保存用户角色

/// </summary>

public tblRoles(){}

#endregion

#region 私有变量

private string \_roleid;

private string \_rolename;

private string \_rolecode;

private string \_creator;

private DateTime \_createdate;

private string \_remark;

#endregion

#region 公共属性

/// <summary>

/// 主键 RoleID (必填字段)

/// </summary>

[DataObjectField(true)]

public string RoleID

{

set{ \_roleid=value;}

get{return \_roleid;}

}

/// <summary>

/// RoleName (必填字段)

/// </summary>

[DataObjectField(false)]

public string RoleName

{

set{ \_rolename=value;}

get{return \_rolename;}

}

/// <summary>

/// RoleCode

/// </summary>

[DataObjectField(false)]

public string RoleCode

{

set{ \_rolecode=value;}

get{return \_rolecode;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// remark

/// </summary>

[DataObjectField(false)]

public string remark

{

set{ \_remark=value;}

get{return \_remark;}

}

#endregion

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* 模块名称：实体 保存用户信息

\* 当前版本：1.0

\* 开发人员：Eric

\* 完成时间：2010-4-29

\* 版本历史：此代码由 VB/C#.Net实体代码生成工具(EntitysCodeGenerate v4.1) 自动生成。

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Data;

using System.Collections;

using System.Collections.Specialized;

using System.ComponentModel;

namespace ATNET.Entity.Common

{

/// <summary>

/// 实体 保存用户信息

/// </summary>

[Serializable(),Description("Primary:UserID")]

public class tblUsers

{

#region 构造函数

/// <summary>

/// 实体 保存用户信息

/// </summary>

public tblUsers(){}

#endregion

#region 私有变量

private string \_userid;

private string \_usercom;

private string \_userrole;

private string \_username;

private string \_useraccount;

private byte[] \_userpsw;

private string \_telephone;

private string \_isvalid;

private string \_creator;

private DateTime \_createdate;

private string \_remark;

#endregion

#region 公共属性

/// <summary>

/// 主键 UserID (必填字段)

/// </summary>

[DataObjectField(true)]

public string UserID

{

set{ \_userid=value;}

get{return \_userid;}

}

/// <summary>

/// UserCom (必填字段)

/// </summary>

[DataObjectField(false)]

public string UserCom

{

set{ \_usercom=value;}

get{return \_usercom;}

}

/// <summary>

/// UserRole

/// </summary>

[DataObjectField(false)]

public string UserRole

{

set{ \_userrole=value;}

get{return \_userrole;}

}

/// <summary>

/// UserName (必填字段)

/// </summary>

[DataObjectField(false)]

public string UserName

{

set{ \_username=value;}

get{return \_username;}

}

/// <summary>

/// UserAccount (必填字段)

/// </summary>

[DataObjectField(false)]

public string UserAccount

{

set{ \_useraccount=value;}

get{return \_useraccount;}

}

/// <summary>

/// UserPSW (必填字段)

/// </summary>

[DataObjectField(false)]

public byte[] UserPSW

{

set{ \_userpsw=value;}

get{return \_userpsw;}

}

/// <summary>

/// Telephone

/// </summary>

[DataObjectField(false)]

public string Telephone

{

set{ \_telephone=value;}

get{return \_telephone;}

}

/// <summary>

/// IsValid (必填字段)

/// </summary>

[DataObjectField(false)]

public string IsValid

{

set{ \_isvalid=value;}

get{return \_isvalid;}

}

/// <summary>

/// Creator (必填字段)

/// </summary>

[DataObjectField(false)]

public string Creator

{

set{ \_creator=value;}

get{return \_creator;}

}

/// <summary>

/// CreateDate (必填字段)

/// </summary>

[DataObjectField(false)]

public DateTime CreateDate

{

set{ \_createdate=value;}

get{return \_createdate;}

}

/// <summary>

/// remark

/// </summary>

[DataObjectField(false)]

public string remark

{

set{ \_remark=value;}

get{return \_remark;}

}

#endregion

}

}

//Canvas类

<ResourceDictionary

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" >

<DrawingBrush x:Key="canvasBrushResource" Viewport="0,0,20,20" ViewportUnits="Absolute" TileMode="Tile">

<DrawingBrush.Drawing>

<DrawingGroup>

<DrawingGroup.Children>

<GeometryDrawing Brush="White">

<!-- 这里是格子填充颜色 -->

<GeometryDrawing.Geometry>

<RectangleGeometry Rect="0,0,1,1" />

</GeometryDrawing.Geometry>

</GeometryDrawing>

<!--<GeometryDrawing Geometry="M0,0 L1,0 1,0.1, 0,0.1Z">

<GeometryDrawing.Pen>

<Pen Brush="Black" Thickness="0.01"/>

</GeometryDrawing.Pen>

</GeometryDrawing>-->

<!-- 这里是横线 -->

<!--<GeometryDrawing Geometry="M0,0 L0,1 0.1,1, 0.1,0Z" Brush="Black" />-->

<GeometryDrawing Brush="Black">

<GeometryDrawing.Geometry>

<EllipseGeometry Center="0,0" RadiusX="0.03" RadiusY="0.03"/>

</GeometryDrawing.Geometry>

</GeometryDrawing>

<!-- 这里是竖线 -->

</DrawingGroup.Children>

</DrawingGroup>

</DrawingBrush.Drawing>

</DrawingBrush>

<!-- 这里是外框线 -->

<Style x:Key="canvasBorderStyle">

<Setter Property="Border.Background" Value="{StaticResource canvasBrushResource}"/>

<Setter Property="Border.HorizontalAlignment" Value="Center"/>

<Setter Property="Border.VerticalAlignment" Value="Top"/>

<Setter Property="Border.BorderBrush" Value="Black"/>

<Setter Property="Border.BorderThickness" Value="1"/>

</Style>

</ResourceDictionary>

//Rectangle类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Shapes;

using System.Windows.Media;

using System.Windows;

using System.Globalization;

using System.Windows.Input;

using System.Windows.Threading;

namespace SoftArt.WPF.Styles.Common

{

public class RulerRectangle:Shape

{

public RulerOrientation Orientation

{

get { return (RulerOrientation)GetValue(OrientationProperty); }

set { SetValue(OrientationProperty, value); }

}

public static DependencyProperty OrientationProperty = DependencyProperty.Register("Orientation", typeof(RulerOrientation), typeof(RulerRectangle));

protected override System.Windows.Media.Geometry DefiningGeometry

{

get

{

Geometry g = new RectangleGeometry(new Rect(new Size(this.ActualWidth, this.ActualHeight)));

return g;

}

}

public static readonly DependencyProperty RulerLineProperty = DependencyProperty.Register("RulerLine", typeof(Point), typeof(RulerRectangle));

/// <summary>

/// 标尺的坐标

/// </summary>

public Point RulerLine

{

get { return (Point)GetValue(RulerLineProperty); }

set

{

SetValue(RulerLineProperty, value);

OnRulerLineChanged(value);

}

}

private delegate void RulerLineChangedEventHandler(object sender, Point point);

private event RulerLineChangedEventHandler RulerLineChangedEvent;

protected void OnRulerLineChanged(Point point)

{

if (RulerLineChangedEvent != null)

{

RulerLineChangedEvent(this, point);

}

}

public RulerRectangle()

{

this.RulerLineChangedEvent += new RulerLineChangedEventHandler(RulerRectangle\_RulerLineChangedEvent);

}

protected void RulerRectangle\_RulerLineChangedEvent(object sendeisr, Point point)

{

this.InvalidateVisual();

}

private delegate void ArgsEventHandle(DrawingContext drawingContext);

protected override void OnRender(DrawingContext drawingContext)

{

//base.OnRender(drawingContext);

Pen pen = new Pen();

SolidColorBrush penColor = new SolidColorBrush(Colors.Black);

penColor.Freeze();

pen.Brush = penColor;

pen.Thickness = 1;

pen.Freeze();

if (this.Orientation == RulerOrientation.Vertical)

{

DrawVerticalRuler(drawingContext);

drawingContext.DrawLine(pen, new Point(15, 0), new Point(15, this.ActualHeight));

}

else

{

DrawHorizontalRuler(drawingContext);

drawingContext.DrawLine(pen, new Point(0, 15), new Point(this.ActualWidth, 15));

}

if (RulerLine.X > 0 && Orientation == RulerOrientation.Horizontal)

{

DrawHorizontalMovingRuler(drawingContext);

}

if (RulerLine.Y > 0 && Orientation == RulerOrientation.Vertical)

{

DrawVerticalMovingRuler(drawingContext);

}

}

private void DrawHorizontalMovingRuler(DrawingContext drawingContext)

{

Pen pen = new Pen();

SolidColorBrush penColor = new SolidColorBrush(Colors.Red);

pen.Brush = penColor;

pen.Thickness = 1;

penColor.Freeze();

pen.Freeze();

//drawingContext.DrawLine(pen, new Point(RulerLine.X, 0), new Point(RulerLine.X, 15));

drawingContext.DrawGeometry(Brushes.Red, pen, new LineGeometry(new Point(RulerLine.X, 0), new Point(RulerLine.X, 15)));

}

private void DrawVerticalMovingRuler(DrawingContext drawingContext)

{

Pen pen = new Pen();

SolidColorBrush penColor = new SolidColorBrush(Colors.Red);

pen.Brush = penColor;

pen.Thickness = 1;

penColor.Freeze();

pen.Freeze();

//drawingContext.DrawLine(pen, new Point(0, RulerLine.Y), new Point(15, RulerLine.Y));

drawingContext.DrawGeometry(Brushes.Red, pen, new LineGeometry(new Point(0, RulerLine.Y), new Point(15, RulerLine.Y)));

}

private void DrawVerticalRuler(DrawingContext drawingContext)

{

int yCount = (int)this.ActualHeight;

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

{

Pen p = new Pen(new SolidColorBrush(Colors.Black), 0.5);

int lineLenth = 5;

if (i % 40 == 0)

{

lineLenth = 15;

p.Thickness = 1;

//drawingContext.DrawLine(p, new Point(0, i), new Point(lineLenth, i));

drawingContext.DrawGeometry(Brushes.Black, p, new LineGeometry(new Point(0, i), new Point(lineLenth, i)));

}

else if (i % 20 == 0)

{

//drawingContext.DrawLine(p, new Point(10, i), new Point(10 + lineLenth, i));

drawingContext.DrawGeometry(Brushes.Black, p, new LineGeometry(new Point(10, i), new Point(10 + lineLenth, i)));

}

if (i % 40 == 0)

{

int label = i / 40;

drawingContext.DrawText(new FormattedText(label.ToString(),

CultureInfo.GetCultureInfo("en-us"),

FlowDirection.LeftToRight,

new Typeface("Verdana"),

8, Brushes.Black),

new Point(lineLenth - 10, i + 5));

}

}

}

private void DrawHorizontalRuler(DrawingContext drawingContext)

{

int yCount = (int)this.ActualWidth;

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

{

Pen p = new Pen(new SolidColorBrush(Colors.Black), 0.5);

int lineLenth = 5;

if (i % 40 == 0)

{

p = new Pen(Brushes.Black, 1);

lineLenth = 15;

//drawingContext.DrawLine(p, new Point(i, 0), new Point(i, lineLenth));

drawingContext.DrawGeometry(Brushes.Black, p, new LineGeometry(new Point(i, 0), new Point(i, lineLenth)));

}

else if (i % 20 == 0)

{

//drawingContext.DrawLine(p, new Point(i, 10), new Point(i, lineLenth + 10));

drawingContext.DrawGeometry(Brushes.Black, p, new LineGeometry(new Point(i, 10), new Point(i, lineLenth + 10)));

}

if (i % 40 == 0)

{

int label = i / 40;

drawingContext.DrawText(new FormattedText(label.ToString(),

CultureInfo.GetCultureInfo("en-us"),

FlowDirection.LeftToRight,

new Typeface("Verdana"),

8, Brushes.Black),

new Point(i + 5, lineLenth - 10));

}

}

}

}

public enum RulerOrientation

{

Horizontal,

Vertical

}

}

//ScrollViewer类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows.Shapes;

using System.Windows.Media;

using System.Windows;

using System.Globalization;

using System.Windows.Controls;

using System.Windows.Markup;

using System.Xml;

namespace SoftArt.WPF.Styles.Common

{

public class ScrollViewerRuler : ScrollViewer

{

public static readonly DependencyProperty MousePointProperty = DependencyProperty.Register("MousePoint", typeof(Point), typeof(ScrollViewerRuler));

public Point MousePoint

{

get { return (Point)GetValue(MousePointProperty); }

set { SetValue(MousePointProperty, value); }

}

protected override void OnPreviewMouseMove(System.Windows.Input.MouseEventArgs e)

{

base.OnPreviewMouseMove(e);

MousePoint = e.GetPosition(this);

}

}

}

//Adorner类

using System.Collections.Generic;

using System.Windows;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

namespace SoftArt.WPF.Graph

{

public class ConnectorAdorner : Adorner

{

private PathGeometry pathGeometry;

private DesignerCanvas designerCanvas;

private Connector sourceConnector;

private Pen drawingPen;

private DesignerItem hitDesignerItem;

private DesignerItem HitDesignerItem

{

get { return hitDesignerItem; }

set

{

if (hitDesignerItem != value)

{

if (hitDesignerItem != null)

hitDesignerItem.IsDragConnectionOver = false;

hitDesignerItem = value;

if (hitDesignerItem != null)

hitDesignerItem.IsDragConnectionOver = true;

}

}

}

private Connector hitConnector;

private Connector HitConnector

{

get { return hitConnector; }

set

{

if (hitConnector != value)

{

hitConnector = value;

}

}

}

public ConnectorAdorner(DesignerCanvas designer, Connector sourceConnector)

: base(designer)

{

this.designerCanvas = designer;

this.sourceConnector = sourceConnector;

drawingPen = new Pen(Brushes.LightSlateGray, 1);

drawingPen.LineJoin = PenLineJoin.Round;

this.Cursor = Cursors.Cross;

}

protected override void OnMouseUp(MouseButtonEventArgs e)

{

if (HitConnector != null)

{

Connector sourceConnector = this.sourceConnector;

Connector sinkConnector = this.HitConnector;

Connection newConnection = new Connection(sourceConnector, sinkConnector);

// connections are added with z-index of zero

this.designerCanvas.Children.Insert(0, newConnection);

}

if (HitDesignerItem != null)

{

this.HitDesignerItem.IsDragConnectionOver = false;

}

if (this.IsMouseCaptured) this.ReleaseMouseCapture();

AdornerLayer adornerLayer = AdornerLayer.GetAdornerLayer(this.designerCanvas);

if (adornerLayer != null)

{

adornerLayer.Remove(this);

}

}

protected override void OnMouseMove(MouseEventArgs e)

{

if (e.LeftButton == MouseButtonState.Pressed)

{

if (!this.IsMouseCaptured) this.CaptureMouse();

HitTesting(e.GetPosition(this));

this.pathGeometry = GetPathGeometry(e.GetPosition(this));

this.InvalidateVisual();

}

else

{

if (this.IsMouseCaptured) this.ReleaseMouseCapture();

}

}

protected override void OnRender(DrawingContext dc)

{

base.OnRender(dc);

dc.DrawGeometry(null, drawingPen, this.pathGeometry);

// without a background the OnMouseMove event would not be fired

// Alternative: implement a Canvas as a child of this adorner, like

// the ConnectionAdorner does.

dc.DrawRectangle(Brushes.Transparent, null, new Rect(RenderSize));

}

private PathGeometry GetPathGeometry(Point position)

{

PathGeometry geometry = new PathGeometry();

ConnectorOrientation targetOrientation;

if (HitConnector != null)

targetOrientation = HitConnector.Orientation;

else

targetOrientation = ConnectorOrientation.None;

List<Point> pathPoints = PathFinderHelper.GetConnectionLine(sourceConnector.GetInfo(), position, targetOrientation);

if (pathPoints.Count > 0)

{

PathFigure figure = new PathFigure();

figure.StartPoint = pathPoints[0];

pathPoints.Remove(pathPoints[0]);

figure.Segments.Add(new PolyLineSegment(pathPoints, true));

geometry.Figures.Add(figure);

}

return geometry;

}

private void HitTesting(Point hitPoint)

{

bool hitConnectorFlag = false;

DependencyObject hitObject = designerCanvas.InputHitTest(hitPoint) as DependencyObject;

while (hitObject != null &&

hitObject != sourceConnector.ParentDesignerItem &&

hitObject.GetType() != typeof(DesignerCanvas))

{

if (hitObject is Connector)

{

HitConnector = hitObject as Connector;

hitConnectorFlag = true;

}

if (hitObject is DesignerItem)

{

HitDesignerItem = hitObject as DesignerItem;

if (!hitConnectorFlag)

HitConnector = null;

return;

}

hitObject = VisualTreeHelper.GetParent(hitObject);

}

HitConnector = null;

HitDesignerItem = null;

}

}

}

using System.Collections.Generic;

using System.ComponentModel;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Controls.Primitives;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

namespace SoftArt.WPF.Graph

{

public class ConnectionAdorner:Adorner

{

private DesignerCanvas designerCanvas;

private Canvas adornerCanvas;

private Connection connection;

private PathGeometry pathGeometry;

private Connector fixConnector, dragConnector;

private Thumb sourceDragThumb, sinkDragThumb;

private Pen drawingPen;

private DesignerItem hitDesignerItem;

private DesignerItem HitDesignerItem

{

get { return hitDesignerItem; }

set

{

if (hitDesignerItem != value)

{

if (hitDesignerItem != null)

hitDesignerItem.IsDragConnectionOver = false;

hitDesignerItem = value;

if (hitDesignerItem != null)

hitDesignerItem.IsDragConnectionOver = true;

}

}

}

private Connector hitConnector;

private Connector HitConnector

{

get { return hitConnector; }

set

{

if (hitConnector != value)

{

hitConnector = value;

}

}

}

private VisualCollection visualChildren;

protected override int VisualChildrenCount

{

get

{

return this.visualChildren.Count;

}

}

protected override Visual GetVisualChild(int index)

{

return this.visualChildren[index];

}

public ConnectionAdorner(DesignerCanvas designer, Connection connection)

: base(designer)

{

this.designerCanvas = designer;

adornerCanvas = new Canvas();

this.visualChildren = new VisualCollection(this);

this.visualChildren.Add(adornerCanvas);

this.connection = connection;

this.connection.PropertyChanged += new PropertyChangedEventHandler(AnchorPositionChanged);

InitializeDragThumbs();

drawingPen = new Pen(Brushes.LightSlateGray, 1);

drawingPen.LineJoin = PenLineJoin.Round;

}

private void InitializeDragThumbs()

{

Style dragThumbStyle = connection.FindResource("ConnectionAdornerThumbStyle") as Style;

//source drag thumb

sourceDragThumb = new Thumb();

Canvas.SetLeft(sourceDragThumb, connection.AnchorPositionSource.X);

Canvas.SetTop(sourceDragThumb, connection.AnchorPositionSource.Y);

this.adornerCanvas.Children.Add(sourceDragThumb);

if (dragThumbStyle != null)

sourceDragThumb.Style = dragThumbStyle;

sourceDragThumb.DragDelta += new DragDeltaEventHandler(thumbDragThumb\_DragDelta);

sourceDragThumb.DragStarted += new DragStartedEventHandler(thumbDragThumb\_DragStarted);

sourceDragThumb.DragCompleted += new DragCompletedEventHandler(thumbDragThumb\_DragCompleted);

// sink drag thumb

sinkDragThumb = new Thumb();

Canvas.SetLeft(sinkDragThumb, connection.AnchorPositionSink.X);

Canvas.SetTop(sinkDragThumb, connection.AnchorPositionSink.Y);

this.adornerCanvas.Children.Add(sinkDragThumb);

if (dragThumbStyle != null)

sinkDragThumb.Style = dragThumbStyle;

sinkDragThumb.DragDelta += new DragDeltaEventHandler(thumbDragThumb\_DragDelta);

sinkDragThumb.DragStarted += new DragStartedEventHandler(thumbDragThumb\_DragStarted);

sinkDragThumb.DragCompleted += new DragCompletedEventHandler(thumbDragThumb\_DragCompleted);

}

void AnchorPositionChanged(object sender, PropertyChangedEventArgs e)

{

if (e.PropertyName.Equals("AnchorPositionSource"))

{

Canvas.SetLeft(sourceDragThumb, connection.AnchorPositionSource.X);

Canvas.SetTop(sourceDragThumb, connection.AnchorPositionSource.Y);

}

if (e.PropertyName.Equals("AnchorPositionSink"))

{

Canvas.SetLeft(sinkDragThumb, connection.AnchorPositionSink.X);

Canvas.SetTop(sinkDragThumb, connection.AnchorPositionSink.Y);

}

}

void thumbDragThumb\_DragCompleted(object sender, DragCompletedEventArgs e)

{

if (HitConnector != null)

{

if (connection != null)

{

if (connection.Source == fixConnector)

connection.Sink = this.HitConnector;

else

connection.Source = this.HitConnector;

}

}

this.HitDesignerItem = null;

this.HitConnector = null;

this.pathGeometry = null;

this.connection.StrokeDashArray = null;

this.InvalidateVisual();

}

void thumbDragThumb\_DragStarted(object sender, DragStartedEventArgs e)

{

this.HitDesignerItem = null;

this.HitConnector = null;

this.pathGeometry = null;

this.Cursor = Cursors.Cross;

this.connection.StrokeDashArray = new DoubleCollection(new double[] { 1, 2 });

if (sender == sourceDragThumb)

{

fixConnector = connection.Sink;

dragConnector = connection.Source;

}

else if (sender == sinkDragThumb)

{

dragConnector = connection.Sink;

fixConnector = connection.Source;

}

}

void thumbDragThumb\_DragDelta(object sender, DragDeltaEventArgs e)

{

Point currentPosition = Mouse.GetPosition(this);

this.HitTesting(currentPosition);

this.pathGeometry = UpdatePathGeometry(currentPosition);

this.InvalidateVisual();

}

protected override void OnRender(DrawingContext dc)

{

base.OnRender(dc);

dc.DrawGeometry(null, drawingPen, this.pathGeometry);

}

protected override Size ArrangeOverride(Size finalSize)

{

adornerCanvas.Arrange(new Rect(0, 0, this.designerCanvas.ActualWidth, this.designerCanvas.ActualHeight));

return finalSize;

}

private PathGeometry UpdatePathGeometry(Point position)

{

PathGeometry geometry = new PathGeometry();

ConnectorOrientation targetOrientation;

if (HitConnector != null)

targetOrientation = HitConnector.Orientation;

else

targetOrientation = dragConnector.Orientation;

List<Point> linePoints = PathFinderHelper.GetConnectionLine(fixConnector.GetInfo(), position, targetOrientation);

if (linePoints.Count > 0)

{

PathFigure figure = new PathFigure();

figure.StartPoint = linePoints[0];

linePoints.Remove(linePoints[0]);

figure.Segments.Add(new PolyLineSegment(linePoints, true));

geometry.Figures.Add(figure);

}

return geometry;

}

private void HitTesting(Point hitPoint)

{

bool hitConnectorFlag = false;

DependencyObject hitObject = designerCanvas.InputHitTest(hitPoint) as DependencyObject;

while (hitObject != null &&

hitObject != fixConnector.ParentDesignerItem &&

hitObject.GetType() != typeof(DesignerCanvas))

{

if (hitObject is Connector)

{

HitConnector = hitObject as Connector;

hitConnectorFlag = true;

}

if (hitObject is DesignerItem)

{

HitDesignerItem = hitObject as DesignerItem;

if (!hitConnectorFlag)

HitConnector = null;

return;

}

hitObject = VisualTreeHelper.GetParent(hitObject);

}

HitConnector = null;

HitDesignerItem = null;

}

}

}