代码清单

—SE2018—G04简易查

****

**小组：G04**

**骆佳俊 徐双铅 吕迪**

**2018.6.26**

**修订表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **生成版本** | **修订人** | **修订章节与内容** | **修订日期** |
| 1 | 0.5 | 吕迪 | 编写内容 | 2018.6.2 |
| 2 | 0.6 | 吕迪 | 增加源代码清单列表 | 2018.6.10 |
| 3 | 0.7 | 吕迪 | 增加预警小灯泡代码 | 2018.6.19 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**目录**

[1 引言 4](#_Toc516517822)

[1.1 编写目的 4](#_Toc516517823)

[1.2 背景 4](#_Toc516517824)

[1.3 定义 4](#_Toc516517825)

[1.4 参考资料 5](#_Toc516517826)

[2 源代码清单 5](#_Toc516517827)

[2.1 源代码列表 5](#_Toc516517828)

[2.2 源代码清单 6](#_Toc516517829)

[2.2.1 AssemblyInfo.cs 6](#_Toc516517830)

[2.2.2 Resources.Designer.cs 8](#_Toc516517831)

[2.2.3 CheckerFactory 10](#_Toc516517832)

[2.2.4 ParamCheckers 11](#_Toc516517833)

[2.2.5 StockDataAnalyzer 14](#_Toc516517834)

[2.2.6 AppCommands 18](#_Toc516517835)

[2.2.7 CollectionCommands 19](#_Toc516517836)

[2.2.8 GlobalView.xaml 20](#_Toc516517837)

[2.2.9 Graph.html 21](#_Toc516517838)

[2.2.10 ParameterAlert.xaml 30](#_Toc516517839)

[2.2.11 StockAnalyze.xaml 35](#_Toc516517840)

[2.2.12 StockAnalyze\_2.cs 43](#_Toc516517841)

[2.2.13 StockAnalyze2.xaml 45](#_Toc516517842)

[2.2.14 StockDetail.xaml 51](#_Toc516517843)

[2.2.15 StockChangeConverter.cs 58](#_Toc516517844)

[2.2.16 DisplayText.cs 59](#_Toc516517845)

[2.2.17 ExtensionMethods.cs 60](#_Toc516517846)

[2.2.18 UtilMethods.cs 63](#_Toc516517847)

[2.2.19 AlertConditionList.cs 64](#_Toc516517848)

[2.2.20 AppConfig.cs 67](#_Toc516517849)

[2.2.21 RealTimeQuotationModel.cs 69](#_Toc516517850)

[2.2.22 StockInfo.cs 72](#_Toc516517851)

[2.2.23 StockQueryModel.cs 73](#_Toc516517852)

[2.2.24 DataTemplates.xaml 74](#_Toc516517853)

[2.2.25 Styles.xaml 75](#_Toc516517854)

[2.2.26 Api.cs 76](#_Toc516517855)

[2.2.27 AppConfigManager.cs 79](#_Toc516517856)

# 引言

伴随着资金的增长和对富有的向往，越来越多的人开始加入炒股的行列。股市的行情是瞬息多变的，掌握的实时的资料是十分必要的，我们做一个可以查看股价，并提供监控提醒。方便股民在工作空余时间实时查看股价，在股民事务繁忙情况下提供监控实时提醒。股民根据信息作出判断。

## 编写目的

设计测试方案是测试阶段的关键技术问题。白盒测试技术中包含逻辑覆盖和控制结构测试这两种设计测试数据的典型技术。白盒测试用例报告是使用白盒测试技术来设计测试数据和预计的输出结构形成测试用例，在项目完成编码以后使用测试用例来发现软件中的错误。

## 背景

1. 待开发软件的名称为“简易查”
2. 本项目提出者和开发者均为SE2018春-G04小组（组长骆佳俊，组员吕迪，徐双铅），目标人群设定为炒股初入门者还有特殊用户杨老师。
3. 相互来往关系：本软件通过调用阿里云的实时股票API和历史股票API，已K线图形式展现给使用者，并筛选出必要数据进行比较得出结果。

## 定义

API：Application Programming Interface,应用程序编程接口

## 参考资料

1. 张海藩,牟永敏.软件工程导论（第六版）[M].北京：清华大学出版社, 2013:162-171
2. 源代码清单模版 （豆丁网） 2018.6.10

<http://www.docin.com/p-957089027.html>

1. 软件工程源代码清单（豆丁网） 2018.6.10

<http://www.docin.com/p-703687587.html>

# 源代码清单

## 源代码列表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 功能模块名称 | 程序名 | 说明 | 代码行数 |
| 1 | 程序集信息 | AssemblyInfo.cs | 自动生成 | 64 |
| 2 | 资源设计器 | Resources.Designer.cs | 自动生成 | 72 |
| 3 | 类型检查 | CheckerFactory |  | 55 |
| 4 | 参数预警测试类 | ParamCheckers.cs | 参数预警多个类 | 146 |
| 5 | 股票数据分析器 | StockDataAnalyzer.cs | 实现了IBackgroundTask接口 | 142 |
| 6 | 本程序命令 | AppCommands.cs |  | 32 |
| 7 | 集合操作专用命令 | CollectionCommands.cs |  | 36 |
| 8 | 大盘UI | GlobalView.xaml. | UI部件 | 62 |
| 9 | 股票图UI | Graph.html | 网页 | 378 |
| 10 | 参数预警UI | ParameterAlert.xaml | UI部件 | 216 |
| 11 | 股票分析UI | StockAnalyze.xaml | UI部件 | 274 |
| 12 | 股票分析后台代码 | StockAnalyze\_2.cs |  | 86 |
| 13 | 股票分析UI | StockAnalyze2.xaml |  | 259 |
| 14 | 详细数据UI | StockDetail.xaml | UI部件 | 296 |
| 15 | 股票数值转换器 | StockChangeConverter.cs | 股票涨跌值的XAML值转换器 | 47 |
| 16 |  | DisplayText.cs |  | 30 |
| 17 | 扩展方法 | ExtensionMethods.cs | 工具方法 | 95 |
| 18 | 辅助方法 | UtilMethods.cs | 工具方法 | 66 |
| 19 | 参数预警列表模型类 | AlertConditionList.cs | 模型类 | 117 |
| 20 | 程序配置模型类 | AppConfig.cs | 模型类 | 74 |
| 21 | 实时行情模型类 | RealTimeQuotationModel.cs | 模型类 | 130 |
| 22 | 股票信息模型类 | StockInfo.cs | 模型类 | 35 |
| 23 | 股票查询模型类 | StockQueryModel.cs | 模型类 | 29 |
| 24 | 数据现实模板 | DataTemplates.xaml |  | 68 |
| 25 | UI样式 | Styles.xaml |  | 74 |
| 26 | API | Api.cs |  | 109 |
| 27 | 程序配置管理器 | AppConfigManager.cs |  | 97 |
| 28 | 程序入口 | App.xaml.cs | 自动生成 | 18 |
| 29 | 主窗口实现 | Mainwindow.xaml | 窗口 | 160 |
| 30 | 主窗口后台代码 | Mainwindow.xaml.cs | 窗口 | 154 |
| 总代码量 | | 3061 | | |

## 源代码清单

### AssemblyInfo.cs

using System.Reflection;

using System.Resources;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using System.Windows;

using System.Windows.Markup;

// 有关程序集的一般信息由以下

// 控制。更改这些特性值可修改

// 与程序集关联的信息。

[assembly: AssemblyTitle("SimpleTracer")]

[assembly: AssemblyDescription("")]

[assembly: AssemblyConfiguration("")]

[assembly: AssemblyCompany("")]

[assembly: AssemblyProduct("SimpleTracer")]

[assembly: AssemblyCopyright("Copyright © 2018 LoveKicher")]

[assembly: AssemblyTrademark("")]

[assembly: AssemblyCulture("")]

// 将 ComVisible 设置为 false 会使此程序集中的类型

//对 COM 组件不可见。如果需要从 COM 访问此程序集中的类型

//请将此类型的 ComVisible 特性设置为 true。

[assembly: ComVisible(false)]

//若要开始生成可本地化的应用程序，请设置

//.csproj 文件中的 <UICulture>CultureYouAreCodingWith</UICulture>

//例如，如果您在源文件中使用的是美国英语，

//使用的是美国英语，请将 <UICulture> 设置为 en-US。 然后取消

//对以下 NeutralResourceLanguage 特性的注释。 更新

//以下行中的“en-US”以匹配项目文件中的 UICulture 设置。

//[assembly: NeutralResourcesLanguage("en-US", UltimateResourceFallbackLocation.Satellite)]

[assembly: ThemeInfo(

ResourceDictionaryLocation.None, //主题特定资源词典所处位置

//(未在页面中找到资源时使用，

//或应用程序资源字典中找到时使用)

ResourceDictionaryLocation.SourceAssembly //常规资源词典所处位置

//(未在页面中找到资源时使用，

//、应用程序或任何主题专用资源字典中找到时使用)

)]

// 程序集的版本信息由下列四个值组成:

//

// 主版本

// 次版本

// 生成号

// 修订号

//

// 可以指定所有值，也可以使用以下所示的 "\*" 预置版本号和修订号

// 方法是按如下所示使用“\*”: :

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyFileVersion("1.0.0.0")]

[assembly: XmlnsPrefix("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "st")]

[assembly: XmlnsDefinition("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "SimpleTracer")]

[assembly: XmlnsDefinition("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "SimpleTracer.Controls")]

[assembly: XmlnsDefinition("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "SimpleTracer.Converters")]

[assembly: XmlnsDefinition("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "SimpleTracer.Commands")]

[assembly: XmlnsDefinition("https://schemas.suwaworld.xyz/xaml/2018/simpletracer", "SimpleTracer.Models")]

### Resources.Designer.cs

//------------------------------------------------------------------------------

// <auto-generated>

// 此代码由工具生成。

// 运行时版本: 4.0.30319.42000

//

// 对此文件的更改可能导致不正确的行为，如果

// 重新生成代码，则所做更改将丢失。

// </auto-generated>

//------------------------------------------------------------------------------

namespace SimpleTracer.Properties

{

/// <summary>

/// 强类型资源类，用于查找本地化字符串等。

/// </summary>

// 此类是由 StronglyTypedResourceBuilder

// 类通过类似于 ResGen 或 Visual Studio 的工具自动生成的。

// 若要添加或删除成员，请编辑 .ResX 文件，然后重新运行 ResGen

// (以 /str 作为命令选项)，或重新生成 VS 项目。

[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyTypedResourceBuilder", "4.0.0.0")]

[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]

[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]

internal class Resources

{

private static global::System.Resources.ResourceManager resourceMan;

private static global::System.Globalization.CultureInfo resourceCulture;

[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance", "CA1811:AvoidUncalledPrivateCode")]

internal Resources()

{

}

/// <summary>

/// 返回此类使用的缓存 ResourceManager 实例。

/// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

internal static global::System.Resources.ResourceManager ResourceManager

{

get

{

if ((resourceMan == null))

{

global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("SimpleTracer.Properties.Resources", typeof(Resources).Assembly);

resourceMan = temp;

}

return resourceMan;

}

}

/// <summary>

/// 覆盖当前线程的 CurrentUICulture 属性

/// 使用此强类型的资源类的资源查找。

/// </summary>

[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.EditorBrowsableState.Advanced)]

internal static global::System.Globalization.CultureInfo Culture

{

get

{

return resourceCulture;

}

set

{

resourceCulture = value;

}

}

}}

### CheckerFactory

using PropertyChanged;

using SimpleTracer.Extension;

using SimpleTracer.Models;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

namespace SimpleTracer.Analysis

{

//使用工厂（Factory）设计模式，利用类工厂根据参数生产指定的类实例

/// <summary>

/// 生成参数检查器的工厂类

/// </summary>

public class CheckerFactory

{

/// <summary>

/// 根据参数名生成对应的<see cref="ParamCheckerBase"/>类

/// </summary>

/// <param name="paramName"></param>

/// <returns></returns>

public static ParamCheckerBase Create(string paramName)

{

Assembly asm = Assembly.GetExecutingAssembly();

try

{

//通过反射查找指定类型

var types = (from t in asm.GetTypes()

where t.IsSubclassOf(typeof(ParamCheckerBase))

&& t.Name.Replace("Checker", "").ToLower() == paramName.ToLower()

select t).ToArray();

if (types.Length > 0)

{

return Activator.CreateInstance(types[0]) as ParamCheckerBase;

}

}

catch

{ }

return null;

}

}

/// <summary>

/// 参数检查器的基类

/// </summary>

public abstract class ParamCheckerBase

{

public abstract bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue);

}

}

### ParamCheckers

using SimpleTracer.Extension;

using SimpleTracer.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static SimpleTracer.Extension.UtilMethods;

namespace SimpleTracer.Analysis

{

//使用策略（Strategy）设计模式，针对不同参数提供不同的报警规则，但又符合共同的接口约束。

/// <summary>

/// 股价超过指定值的报警规则

/// </summary>

public class PriceOverChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

alertValue = last.last\_px;

return last.last\_px >= condition.alertValue;

}

return false;

}

}

/// <summary>

/// 股价低于指定值的报警规则

/// </summary>

public class PriceUnderChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

alertValue = last.last\_px;

return last.last\_px <= condition.alertValue;

}

return false;

}

}

/// <summary>

/// 日涨幅超过指定值的报警规则

/// </summary>

public class DailyRateUpChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

alertValue = last.px\_change\_rate;

return last.px\_change\_rate > 0

&& last.px\_change\_rate >= condition.alertValue \* 0.01;

}

return false;

}

}

/// <summary>

/// 日跌幅超过指定值的报警规则

/// </summary>

public class DailyRateDownChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

alertValue = last.px\_change\_rate;

return last.px\_change\_rate < 0

&& -last.px\_change\_rate >= condition.alertValue \* 0.01;

}

return false;

}

}

/// <summary>

/// 5分钟涨幅超过指定值的报警规则

/// </summary>

public class FiveMinRateUpChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

//筛选离 最后一次请求 过去300 ± 29秒的数据，然后取最近的那个

var before = (from p in data

where IsBetween(last.data\_timestamp - p.Key,

5 \* 60 - 29, 5 \* 60 + 29)

orderby p.Key

select p.Value)

.LastOrDefault();

if (before != null)

{

var rate = (last.last\_px - before.last\_px) / before.last\_px;

alertValue = rate;

return rate > 0 && rate >= condition.alertValue \* 0.01;

}

}

return false;

}

}

/// <summary>

/// 5分钟跌幅超过指定值的报警规则

/// </summary>

public class FiveMinRateDownChecker : ParamCheckerBase

{

public override bool Check(Dictionary<double, RealTimeQuotationModel> data, AlertCondition condition, out double alertValue)

{

alertValue = 0;

if (data.Count > 0)

{

var last = data[data.Keys.Max()];

//筛选离 最后一次请求 过去300 ± 29秒的数据，然后取最近的那个

var before = (from p in data

where IsBetween(last.data\_timestamp - p.Key,

5 \* 60 - 29, 5 \* 60 + 29)

orderby p.Key

select p.Value)

.LastOrDefault();

if (before != null)

{

var rate = (last.last\_px - before.last\_px) / before.last\_px;

alertValue = rate;

return rate < 0 && -rate >= condition.alertValue \* 0.01;

}

}

return false;

}

}

}

### StockDataAnalyzer

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using SimpleTracer.Models;

using System.Timers;

using SimpleTracer.Tasks;

namespace SimpleTracer.Analysis

{

/// <summary>

/// 股票数据分析器

/// </summary>

public class StockDataAnalyzer : IBackgroundTask

{

public StockDataAnalyzer()

{

timer.Elapsed += Timer\_Elapsed;

DataDownloader.DownloadComplete += DataDownloader\_DownloadComplete;

}

static Timer timer =

#if DEBUG

new Timer(10 \* 1000)

#else

new Timer(60 \* 1000)//每1分钟一次

#endif

{

AutoReset = true,

Enabled = false

};

/// <summary>

/// 股票数据，Key是对应的时间戳

/// </summary>

public static Dictionary<double, RealTimeQuotationModel> StockData { get; set; } =

new Dictionary<double, RealTimeQuotationModel>();

private void Timer\_Elapsed(object sender, ElapsedEventArgs e)

{

Logger.Log(nameof(StockDataAnalyzer) + "触发计时器");

AnalyzeAllAsync();

}

private void DataDownloader\_DownloadComplete(object sender, DownloadCompleteEventArgs<List<RealTimeQuotationModel>> e)

{

foreach (var item in e.ContentModel)

{

AppendData(item);

}

}

/// <summary>

/// 分析指定股票的参数

/// </summary>

/// <param name="stockID"></param>

public void Analyze(string stockID)

{

var conditions = AppConfigManager.GetManager().Config.alertList[stockID];

var result = new AlertConditionList(noDefaultList: true);

foreach (var condition in conditions)

{

//利用LINQ查询筛选指定股票代码的数据

var data = (from p in StockData

where p.Value.prod\_code == stockID

orderby p.Key

select p)

.ToDictionary(p => p.Key, p => p.Value);//生成新字典

//如果达到预警条件

if (condition.CheckAlert(data, out var current))

{

condition.currentValue = current;

result.Add(condition);

}

}

if (result.Count > 0)

{

Notificator.Notify(stockID, result);

}

}

/// <summary>

/// 异步分析所有股票

/// </summary>

public async void AnalyzeAllAsync()

{

await Task.Run(() =>

{

foreach (var alertInfo in AppConfigManager.GetManager().Config.alertList)

{

Analyze(alertInfo.Key);

}

});

}

/// <summary>

/// 添加新的实时数据

/// </summary>

/// <param name="model"></param>

public void AppendData(RealTimeQuotationModel model)

{

//HACK:防止因为请求时间过于接近导致key重复

while (StockData.ContainsKey(model.data\_timestamp))

{

model.data\_timestamp += 0.001;

}

StockData.Add(model.data\_timestamp, model);

//Analyze(model.prod\_code);

}

public void Start()

{

timer.Enabled = true;

Logger.Log(nameof(StockDataAnalyzer) + "已启动任务");

}

public void Stop()

{

timer.Enabled = false;

}

}

}

### AppCommands

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace SimpleTracer.Commands

{

/// <summary>

/// 包含此程序专用的命令集合

/// </summary>

public class AppCommands

{

private static RoutedUICommand showDetailCommand =

new RoutedUICommand("显示详细数据", "ShowDetail", typeof(AppCommands));

/// <summary>

/// 显示详细信息的命令

/// </summary>

public static RoutedUICommand ShowDetail => showDetailCommand;

private static RoutedUICommand alertCommand =

new RoutedUICommand("设置报警规则", "SetAlert", typeof(AppCommands));

/// <summary>

/// 设置报警规则的命令

/// </summary>

public static RoutedUICommand SetAlert => alertCommand;

}

}

### CollectionCommands

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace SimpleTracer.Commands

{

public class CollectionCommands

{

private static RoutedUICommand addCommand =

new RoutedUICommand("添加", "Add", typeof(CollectionCommands));

private static RoutedUICommand removeCommand =

new RoutedUICommand("移除", "Remove", typeof(CollectionCommands),

new InputGestureCollection { new KeyGesture(Key.Delete) });

private static RoutedUICommand clearCommand =

new RoutedUICommand("清除", "Clear", typeof(CollectionCommands));

/// <summary>

/// 向集合添加项目

/// </summary>

public static RoutedUICommand Add => addCommand;

/// <summary>

/// 从集合移除项目

/// </summary>

public static RoutedUICommand Remove => removeCommand;

/// <summary>

/// 清空集合

/// </summary>

public static RoutedUICommand Clear => clearCommand;

}

}

### GlobalView.xaml

<UserControl

x:Class="SimpleTracer.Controls.GlobalView"

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:local="clr-namespace:SimpleTracer.Controls"

xmlns:m="clr-namespace:SimpleTracer.Models"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

d:DesignHeight="300"

d:DesignWidth="300"

Loaded="UserControl\_Loaded"

mc:Ignorable="d">

<UserControl.Resources>

<DataTemplate DataType="{x:Type m:StockInfo}">

<StackPanel Orientation="Horizontal">

<md:PackIcon Kind="ChevronRight" />

<TextBlock Text="{Binding Name}" />

</StackPanel>

</DataTemplate>

</UserControl.Resources>

<Grid>

<md:Card

Margin="10"

HorizontalAlignment="Center"

VerticalAlignment="Stretch">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<TextBlock

Margin="10"

HorizontalAlignment="Center"

VerticalAlignment="Center"

FontSize="36"

Foreground="{DynamicResource SecondaryAccentBrush}">

大盘走势

</TextBlock>

<ListBox

x:Name="list"

Grid.Row="1"

Background="{DynamicResource PrimaryHueLightBrush}"

Foreground="{DynamicResource PrimaryHueLightForegroundBrush}"

ItemsSource="{Binding Indexes, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type local:GlobalView}}}"

SelectedIndex="0"

SelectedItem="{Binding CurrentItem, Mode=OneWayToSource, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type local:GlobalView}}}" />

<WebBrowser

x:Name="browser"

Grid.RowSpan="2"

Grid.Column="1"

Tag="{Binding SelectedItem, ElementName=list}" />

</Grid>

</md:Card>

</Grid>

</UserControl>

### Graph.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />

<meta name="viewport" content="width=device-width,height=device-height" />

<title>股票图</title>

</head>

<body>

<!--[if lt IE 10]>

<style> #main { display:none; } </style>

<center><h2>您的浏览器版本过低，无法正确显示此页内容。</h2></center>

<![endif]-->

<!--[if IE 6]><center><h2>当前版本：IE 6</h2></center><![endif]-->

<!--[if IE 7]><center><h2>当前版本：IE 7</h2></center><![endif]-->

<!--[if IE 8]><center><h2>当前版本：IE 8</h2></center><![endif]-->

<!--[if IE 9]><center><h2>当前版本：IE 9</h2></center><![endif]-->

<style>

::-webkit-scrollbar {

display: none;

}

html,

body {

overflow: hidden;

height: 100%;

font-family: 'Microsoft YaHei UI', 'Segoe UI', Tahoma, Geneva, SimHei, Verdana, sans-serif;

}

.center {

margin-left: auto;

margin-right: auto;

display: block;

}

@-webkit-keyframes circle {

from {

-webkit-transform: rotate(0deg);

}

to {

-webkit-transform: rotate(360deg);

}

}

@-ms-keyframes circle {

from {

-moz-transform: rotate(0deg);

}

to {

-moz-transform: rotate(360deg);

}

}

@keyframes circle {

from {

transform: rotate(0deg);

}

to {

transform: rotate(360deg);

}

}

.wait {

height: 15px;

width: 15px;

-webkit-animation: circle 1s infinite linear;

-moz-animation: circle 1s infinite linear;

animation: circle 1s infinite linear;

}

.wait-circle {

border-radius: 50%;

border: 4px solid transparent;

border-left: 4px solid #009688;

border-top: 4px solid #009688;

}

.wait-large {

height: 36px;

width: 36px;

}

#wrapper {

position: absolute;

top: 56px;

width: 100%;

height: 100%;

z-index: 66;

background-color: black;

opacity: 0.6;

display: -webkit-flex;

display: -ms-flexbox;

display: flex;

-webkit-align-items: center;

-ms-flex-align: center;

align-items: center;

}

#wrapper div {

-moz-user-select: none;

-ms-user-select: none;

-webkit-user-select: none;

user-select: none;

font-size: 64px;

z-index: 100;

vertical-align: middle;

text-align: center;

color: lightgray;

}

</style>

<div id="main" style="position:relative;">

<div>

<h1 id="title" style="font-size: 36px;text-align: center">上证指数</h1>

</div>

<div id="graph">

<div id="mountNode"></div>

</div>

<div>

<div id="wrapper" style="display:none;">

<h2 id="err" class="center" style="display:none;color: red">发生网络错误</h2>

<div class="center wait wait-circle wait-large" id="load"></div>

</div>

</div>

</div>

<script>/\*Fixing iframe window.innerHeight 0 issue in Safari\*/document.body.clientHeight;</script>

<script src="https://gw.alipayobjects.com/os/antv/assets/g2/3.0.9/g2.js"></script>

<script src="https://gw.alipayobjects.com/os/antv/assets/data-set/0.8.7/data-set.min.js"></script>

<script src="https://gw.alipayobjects.com/os/antv/assets/lib/jquery-3.2.1.min.js"></script>

<script src="https://gw.alipayobjects.com/os/antv/assets/g2-plugin-slider/2.0.2/g2-plugin-slider.js"></script>

<script>

var data = [];

function updateData(stock) {

var \_o = JSON.parse(stock);

if (\_o) {

$("#wrapper").show();

$("#title").text(\_o.Name);

$.ajax({

url: "https://stock.api51.cn/kline?candle\_period=5&get\_type=offset&data\_count=300&prod\_code=" + \_o.StockID,

method: "GET",

beforeSend: function (xhr) {

xhr.setRequestHeader("Authorization", "APPCODE d33be00202dd45bcadbd7d2a01357858");

},

dataType: "json",

success: function (data) {

$("#wrapper").hide();

preProcess(data, \_o.StockID);

draw();

},

error: function (xhr) {

$("#load").hide();

$("#err").show();

}

});

}

}

var startTime = "2018-1-1", endTime = "2018-12-31";

function preProcess(d, id) {

console.log(d);

data=[];

var raw = d.data.candle;

var fields = raw["fields"];

var kline = raw[id];

if (kline.length > 0) {

for (var j = 0; j < kline.length; j++) {

data.push({

time: (function (v) {

var \_s = v.toString();

var \_r = \_s[0] + \_s[1] + \_s[2] + \_s[3] + "-" + \_s[4] + \_s[5] + "-" + \_s[6] + \_s[7] + " " + \_s[8] + \_s[9] + ":" + \_s[10] + \_s[11];

if (j == 0) startTime = \_r;

if (j == kline.length - 1) endTime = \_r;

//console.log(\_r);

return \_r;

})(kline[j][fields.indexOf("min\_time")]),

start: kline[j][fields.indexOf("open\_px")],

max: kline[j][fields.indexOf("high\_px")],

min: kline[j][fields.indexOf("low\_px")],

end: kline[j][fields.indexOf("close\_px")],

volumn: kline[j][fields.indexOf("business\_amount")],

money: kline[j][fields.indexOf("business\_balance")]

});

}

}

}

updateData(JSON.stringify({

Name: "上证指数",

StockID: "000001.SS"

}));

</script>

<script>

function draw() {

$("#graph").empty();

$("#graph").html('<div id="mountNode"></div><div id="slider"></div>');

//$('<div id="slider"></div>').insertAfter('#mountNode');

// 设置状态量，时间格式建议转换为时间戳，转换为时间戳时请注意区间

var ds = new DataSet({

state: {

start: startTime,

end: endTime

}

});

var dv = ds.createView();

dv.source(data)

.transform({

type: 'filter',

callback: function (obj) {

var date = obj.time;

return date <= ds.state.end && date >= ds.state.start;

}

})

.transform({

type: 'map',

callback: function (obj) {

obj.trend = (obj.start <= obj.end) ? '上涨' : '下跌';

obj.range = [obj.start, obj.end, obj.max, obj.min];

return obj;

}

});

var chart = new G2.Chart({

container: 'mountNode',

forceFit: true,

height: window.innerHeight - 150,

animate: false,

padding: [10, 40, 40, 40],

});

chart.source(dv, {

'time': {

type: 'timeCat',

nice: false,

range: [0, 1]

},

trend: {

values: ['上涨', '下跌']

},

'volumn': { alias: '成交量' },

'start': { alias: '开盘价' },

'end': { alias: '收盘价' },

'max': { alias: '最高价' },

'min': { alias: '最低价' },

'range': { alias: '股票价格' }

});

chart.legend({

offset: 20

});

chart.tooltip({

showTitle: false,

itemTpl: '<li data-index={index}>'

+ '<span style="background-color:{color};" class="g2-tooltip-marker"></span>'

+ '{name}{value}</li>',

});

var kView = chart.view({

end: {

x: 1,

y: 0.5

}

});

kView.source(dv);

kView.schema()

.position('time\*range')

.color('trend', function (val) {

if (val === '上涨') {

return '#f04864';

}

if (val === '下跌') {

return '#2fc25b';

}

})

.shape('candle')

.tooltip('time\*start\*end\*max\*min', function (time, start, end, max, min) {

return {

name: time,

value: '<br><span style="padding-left: 16px">开盘价：' + start + '</span><br/>'

+ '<span style="padding-left: 16px">收盘价：' + end + '</span><br/>'

+ '<span style="padding-left: 16px">最高价：' + max + '</span><br/>'

+ '<span style="padding-left: 16px">最低价：' + min + '</span>'

};

});

var barView = chart.view({

start: {

x: 0,

y: 0.65

}

});

barView.source(dv, {

volumn: {

tickCount: 2

}

});

barView.axis('time', {

tickLine: null,

label: null

});

barView.axis('volumn', {

label: {

formatter: function (val) {

return parseInt(val / 1000, 10) + 'k';

}

}

});

barView.interval()

.position('time\*volumn')

.color('trend', function (val) {

if (val === '上涨') {

return '#f04864';

}

if (val === '下跌') {

return '#2fc25b';

}

})

.tooltip('time\*volumn', function (time, volumn) {

return {

name: time,

value: '<br/><span style="padding-left: 16px">成交量：' + volumn + '</span><br/>'

};

});

chart.render();

// 生成 slider

var slider = new Slider({

container: 'slider', // DOM id

width: 'auto',

height: 26,

padding: [20, 40, 20, 40],

start: ds.state.start, // 和状态量对应

end: ds.state.end,

data: data, // 源数据

xAxis: 'time', // 背景图的横轴对应字段，同时为数据筛选的字段

yAxis: 'volumn', // 背景图的纵轴对应字段，同时为数据筛选的字段

scales: {

time: {

type: 'timeCat',

nice: false,

}

},

onChange: function (/\*{ startText, endText }\*/obj) {

ds.setState('start', obj.startText);

ds.setState('end', obj.endText);

}

});

slider.render();

}

</script>

</body>

</html>

### ParameterAlert.xaml

<UserControl

x:Class="SimpleTracer.Controls.ParameterAlert"

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

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

xmlns:com="clr-namespace:SimpleTracer.Commands"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:SimpleTracer.Controls"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:mdl="clr-namespace:SimpleTracer.Models"

xmlns:validation="clr-namespace:SimpleTracer.Validation"

x:Name="userControl"

d:DesignHeight="600"

d:DesignWidth="800"

mc:Ignorable="d">

<UserControl.CommandBindings>

<CommandBinding

x:Name="saveCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static ApplicationCommands.Save}"

Executed="saveCommand\_Executed" />

<CommandBinding

x:Name="clearCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:CollectionCommands.Clear}"

Executed="clearCommand\_Executed" />

</UserControl.CommandBindings>

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/SimpleTracer;component/Resources/Styles.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid>

<GroupBox>

<GroupBox.Header>

<Grid>

<StackPanel Orientation="Horizontal">

<Button Style="{StaticResource backButtonStyle}" />

<TextBlock

VerticalAlignment="Center"

Foreground="White"

Style="{DynamicResource MaterialDesignTitleTextBlock}">

预警规则

</TextBlock>

</StackPanel>

</Grid>

</GroupBox.Header>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="\*" />

<RowDefinition Height="Auto" />

</Grid.RowDefinitions>

<Grid

x:Name="container"

Margin="80,20"

DataContext="{Binding CurrentAlertConditions, ElementName=userControl}"

Grid.IsSharedSizeScope="True">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

<RowDefinition />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<Grid.Resources>

<Style TargetType="StackPanel">

<Setter Property="Orientation" Value="Horizontal" />

</Style>

<Style BasedOn="{StaticResource {x:Type TextBox}}" TargetType="TextBox">

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="MinWidth" Value="200" />

<Setter Property="MaxWidth" Value="500" />

<Setter Property="Margin" Value="5,0" />

</Style>

<Style TargetType="TextBlock">

<Setter Property="VerticalAlignment" Value="Center" />

<Setter Property="Margin" Value="10,0,5,0" />

</Style>

<Style BasedOn="{StaticResource MaterialDesignSwitchToggleButton}" TargetType="CheckBox">

<Setter Property="Grid.Column" Value="1" />

<Setter Property="Margin" Value="20,0" />

<Setter Property="ToolTip" Value="启用/禁用参数" />

</Style>

</Grid.Resources>

<StackPanel>

<TextBlock Text="股价超过 " />

<!-- 下面这些神奇的绑定表达式是简写形式，理解请注意前面元素的DataContext -->

<TextBox x:Name="priceOver" Foreground="Red">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[0].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>元</TextBlock>

</StackPanel>

<CheckBox IsChecked="{Binding [0].enabled}" />

<StackPanel Grid.Row="1">

<TextBlock Text="股价低于 " />

<TextBox x:Name="priceUnder" Foreground="LimeGreen">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[1].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>元</TextBlock>

</StackPanel>

<CheckBox Grid.Row="1" IsChecked="{Binding [1].enabled}" />

<StackPanel Grid.Row="2">

<TextBlock Text="日涨幅超过 " />

<TextBox x:Name="dailyRateUp" Foreground="Red">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[2].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>%</TextBlock>

</StackPanel>

<CheckBox Grid.Row="2" IsChecked="{Binding [2].enabled}" />

<StackPanel Grid.Row="3">

<TextBlock Text="日跌幅超过 " />

<TextBox x:Name="dailyRateDown" Foreground="LimeGreen">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[3].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>%</TextBlock>

</StackPanel>

<CheckBox Grid.Row="3" IsChecked="{Binding [3].enabled}" />

<StackPanel Grid.Row="4">

<TextBlock>5分钟涨幅超过</TextBlock>

<TextBox x:Name="fiveMinRateUp" Foreground="Red">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[4].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>%</TextBlock>

</StackPanel>

<CheckBox Grid.Row="4" IsChecked="{Binding [4].enabled}" />

<StackPanel Grid.Row="5">

<TextBlock>5分钟跌幅超过</TextBlock>

<TextBox x:Name="fiveMinRateDown" Foreground="LimeGreen">

<TextBox.Text>

<Binding NotifyOnValidationError="True" Path="[5].alertValue">

<Binding.ValidationRules>

<validation:NoNegativeValidationRule />

</Binding.ValidationRules>

</Binding>

</TextBox.Text>

</TextBox>

<TextBlock>%</TextBlock>

</StackPanel>

<CheckBox Grid.Row="5" IsChecked="{Binding [5].enabled}" />

</Grid>

<Separator

Margin="8,0"

VerticalAlignment="Bottom"

Style="{StaticResource MaterialDesignDarkSeparator}" />

<Grid Grid.Row="1">

<StackPanel HorizontalAlignment="Right" Orientation="Horizontal">

<Button

x:Name="clear"

Command="{x:Static com:CollectionCommands.Clear}"

Content="清空"

Style="{DynamicResource dialogButtonStyle}" />

<Button

x:Name="save"

Command="{x:Static ApplicationCommands.Save}"

Content="保存"

Style="{DynamicResource dialogDefaultButtonStyle}" />

</StackPanel>

</Grid>

</Grid>

</GroupBox>

</Grid>

</Grid>

</UserControl>

### StockAnalyze.xaml

<UserControl

x:Class="SimpleTracer.Controls.StockAnalyze"

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

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

xmlns:com="clr-namespace:SimpleTracer.Commands"

xmlns:controls="clr-namespace:SimpleTracer.Controls"

xmlns:cov="clr-namespace:SimpleTracer.Converters"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:mdl="clr-namespace:SimpleTracer.Models"

xmlns:validation="clr-namespace:SimpleTracer.Validation"

x:Name="control"

d:DesignHeight="300"

d:DesignWidth="800"

mc:Ignorable="d">

<UserControl.CommandBindings>

<!-- 定义所有命令绑定的事件 -->

<CommandBinding

x:Name="addCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:CollectionCommands.Add}"

Executed="addCommand\_Executed" />

<CommandBinding

x:Name="removeCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:CollectionCommands.Remove}"

Executed="removeCommand\_Executed" />

<CommandBinding

x:Name="detailCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:AppCommands.ShowDetail}"

Executed="detailCommand\_Executed" />

<CommandBinding

x:Name="alertCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:AppCommands.SetAlert}"

Executed="alertCommand\_Executed" />

</UserControl.CommandBindings>

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/SimpleTracer;component/Resources/DataTemplates.xaml" />

<ResourceDictionary Source="/SimpleTracer;component/Resources/Styles.xaml" />

</ResourceDictionary.MergedDictionaries>

<BooleanToVisibilityConverter x:Key="bc" />

<cov:StockChangeConverter x:Key="covChange" />

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<md:ColorZone Padding="12" Mode="Light">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Button

VerticalAlignment="Center"

IsDefault="True"

Style="{DynamicResource MaterialDesignToolButton}">

<md:PackIcon

Width="24"

Height="24"

Kind="Magnify"

Opacity="0.56" />

</Button>

<ComboBox

x:Name="search"

Grid.Column="1"

MinWidth="200"

Margin="24,0,32,0"

md:HintAssist.Hint="搜索股票代码"

FontSize="18"

IsEditable="True">

<ComboBox.Text>

<Binding

Delay="500"

ElementName="control"

NotifyOnValidationError="True"

Path="SearchText">

<Binding.ValidationRules>

<validation:RegexValidationRule RegexPattern="^[0-9]{1,6}$" />

</Binding.ValidationRules>

</Binding>

</ComboBox.Text>

<ComboBoxItem

x:Name="searching"

HorizontalAlignment="Center"

IsEnabled="False"

Visibility="Collapsed">

<ProgressBar

Width="36"

Height="36"

IsIndeterminate="True"

Style="{StaticResource MaterialDesignCircularProgressBar}"

Value="60" />

</ComboBoxItem>

</ComboBox>

</Grid>

</md:ColorZone>

<md:Card

Grid.Row="1"

Margin="5"

VerticalAlignment="Stretch">

<Grid>

<md:DialogHost x:Name="dialog">

<md:DialogHost.DialogContent>

<StackPanel

Margin="16"

TextElement.FontFamily="Microsoft YaHei UI"

TextElement.FontSize="18">

<StackPanel Orientation="Horizontal">

<md:PackIcon

Width="24"

Height="24"

Kind="Alert" />

<TextBlock>确实要从自选列表删除吗？</TextBlock>

</StackPanel>

<StackPanel HorizontalAlignment="Right" Orientation="Horizontal">

<Button

Margin="0,8,8,0"

Command="{x:Static com:CollectionCommands.Remove}"

CommandParameter="{Binding DataContext, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type md:DialogHost}}}"

IsDefault="True"

Style="{StaticResource MaterialDesignFlatButton}">

确定

</Button>

<Button

Margin="0,8,8,0"

Command="md:DialogHost.CloseDialogCommand"

CommandParameter="False"

IsCancel="True"

Style="{StaticResource MaterialDesignFlatButton}">

取消

</Button>

</StackPanel>

</StackPanel>

</md:DialogHost.DialogContent>

<md:Transitioner x:Name="transitioner" SelectedIndex="0">

<md:TransitionerSlide>

<Grid x:Name="page0">

<ListView

x:Name="list"

IsSynchronizedWithCurrentItem="True"

ItemsSource="{Binding StockList, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type controls:StockAnalyze}}}">

<ListView.View>

<GridView>

<GridViewColumn DisplayMemberBinding="{Binding prod\_code}" Header="股票代码" />

<GridViewColumn DisplayMemberBinding="{Binding prod\_name}" Header="股票名称" />

<GridViewColumn DisplayMemberBinding="{Binding open\_px}" Header="开盘" />

<GridViewColumn Header="最高">

<GridViewColumn.CellTemplate>

<DataTemplate>

<TextBlock Foreground="Red" Text="{Binding high\_px}" />

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn Header="最低">

<GridViewColumn.CellTemplate>

<DataTemplate>

<TextBlock Foreground="LimeGreen" Text="{Binding low\_px}" />

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn Header="当前">

<GridViewColumn.CellTemplate>

<DataTemplate>

<TextBlock Foreground="{Binding px\_change, Converter={StaticResource covChange}}" Text="{Binding last\_px}" />

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn Header="涨跌">

<GridViewColumn.CellTemplate>

<DataTemplate>

<TextBlock Foreground="{Binding px\_change, Converter={StaticResource covChange}}" Text="{Binding px\_change}" />

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

<GridViewColumn DisplayMemberBinding="{Binding turnover\_ratio}" Header="换手" />

<!--<GridViewColumn DisplayMemberBinding="{Binding business\_balance}" Header="成交" />-->

<GridViewColumn Width="200" Header="操作">

<GridViewColumn.CellTemplate>

<DataTemplate>

<Grid>

<StackPanel Orientation="Horizontal">

<StackPanel.Resources>

<Style BasedOn="{StaticResource MaterialDesignFloatingActionMiniButton}" TargetType="Button">

<Setter Property="Padding" Value="5" />

<Setter Property="Margin" Value="5,0" />

<Setter Property="Width" Value="32" />

<Setter Property="Height" Value="32" />

<Setter Property="ToolTip" Value="{Binding Command.Text, RelativeSource={RelativeSource Self}}" />

</Style>

</StackPanel.Resources>

<ToggleButton

Margin="5"

md:ShadowAssist.ShadowDepth="Depth2"

Command="{x:Static com:AppCommands.SetAlert}"

CommandParameter="{Binding}"

Style="{StaticResource MaterialDesignActionToggleButton}">

<md:PackIcon Kind="Bell" />

</ToggleButton>

<Button

Click="delete\_Click"

Command="md:DialogHost.OpenDialogCommand"

DataContext="{Binding}"

ToolTip="删除项目">

<md:PackIcon Kind="Delete" />

</Button>

<Button Command="{x:Static com:AppCommands.ShowDetail}" CommandParameter="{Binding}">

<md:PackIcon Kind="DotsHorizontal" />

</Button>

<Button ToolTip="实时K线">

<md:PackIcon Kind="ChartLine" />

</Button>

</StackPanel>

</Grid>

</DataTemplate>

</GridViewColumn.CellTemplate>

</GridViewColumn>

</GridView>

</ListView.View>

</ListView>

</Grid>

</md:TransitionerSlide>

<md:TransitionerSlide>

<Grid x:Name="page1">

<controls:ParameterAlert x:Name="alertControl" />

</Grid>

</md:TransitionerSlide>

<md:TransitionerSlide>

<Grid x:Name="page2">

<controls:StockDetail x:Name="detailControl" />

</Grid>

</md:TransitionerSlide>

</md:Transitioner>

</md:DialogHost>

<Border

x:Name="loading"

d:IsHidden="True"

Background="#66000000"

Visibility="{Binding IsLoading, Converter={StaticResource bc}, ElementName=control}">

<ProgressBar

Width="36"

Height="36"

IsIndeterminate="True"

Style="{StaticResource MaterialDesignCircularProgressBar}"

Value="60" />

</Border>

</Grid>

</md:Card>

</Grid>

</UserControl>

### StockAnalyze\_2.cs

using SimpleTracer.Models;

using SimpleTracer.Tasks;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

namespace SimpleTracer.Controls

{

public partial class StockAnalyze

{

/// <summary>

/// StockAnalyze的构造函数

/// </summary>

public StockAnalyze()

{

InitializeComponent();

Init();

}

/// <summary>

/// 初始化

/// </summary>

private void Init()

{

StockList = new ObservableCollection<RealTimeQuotationModel>();

foreach (var item in AppConfigManager.GetManager().Config?.watchList)

{

StockList.Add(new RealTimeQuotationModel

{

prod\_code = item.StockID,

prod\_name = item.Name

});

}

View = CollectionViewSource.GetDefaultView(StockList) as ListCollectionView;

#if DEBUG

View.CurrentChanged += View\_CurrentChanged;

#endif

DataDownloader.DownloadComplete += DataDownloader\_DownloadComplete;

DataDownloader.DownloadFailed += DataDownloader\_DownloadFailed;

if (View.Count > 0)

{

View.MoveCurrentToFirst();

InitAllStockInfo();

}

}

// Using a DependencyProperty as the backing store for IsLoading. This enables animation, styling, binding, etc...

public static readonly DependencyProperty IsLoadingProperty =

DependencyProperty.Register("IsLoading", typeof(bool), typeof(StockAnalyze),

new PropertyMetadata(true));

// Using a DependencyProperty as the backing store for SearchText. This enables animation, styling, binding, etc...

public static readonly DependencyProperty SearchTextProperty =

DependencyProperty.Register("SearchText", typeof(string), typeof(StockAnalyze),

new PropertyMetadata("", OnSearchText\_PropertyChanged));

// Using a DependencyProperty as the backing store for StockList. This enables animation, styling, binding, etc...

public static readonly DependencyProperty StockListProperty =

DependencyProperty.Register("StockList", typeof(ObservableCollection<RealTimeQuotationModel>),

typeof(StockAnalyze), new PropertyMetadata(null));

//HACK:为了解决对话框调用的数据上下文对象转移难题，临时将当前Model类传递给dialog。

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

{

dialog.DataContext = (sender as Control)?.DataContext;

}

}

}

### StockAnalyze2.xaml

<UserControl

x:Class="SimpleTracer.Controls.StockAnalyze"

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

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

xmlns:com="clr-namespace:SimpleTracer.Commands"

xmlns:cov="clr-namespace:SimpleTracer.Converters"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:SimpleTracer.Controls"

xmlns:m="clr-namespace:SimpleTracer.Models"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

x:Name="control"

d:DesignHeight="300"

d:DesignWidth="300"

Loaded="UserControl\_Loaded"

mc:Ignorable="d">

<UserControl.CommandBindings>

<CommandBinding

x:Name="addCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:CollectionCommands.Add}"

Executed="addCommand\_Executed" />

<CommandBinding

x:Name="removeCommand"

CanExecute="Command\_CanExecute"

Command="{x:Static com:CollectionCommands.Remove}"

Executed="removeCommand\_Executed" />

</UserControl.CommandBindings>

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/SimpleTracer;component/Resources/DataTemplates.xaml" />

</ResourceDictionary.MergedDictionaries>

<cov:StockChangeConverter x:Key="covChange" />

<Style BasedOn="{StaticResource MaterialDesignDarkSeparator}" TargetType="{x:Type Separator}">

<Setter Property="VerticalAlignment" Value="Bottom" />

<Setter Property="Margin" Value="8,0" />

</Style>

<DataTemplate DataType="{x:Type m:StockInfo}">

<DockPanel LastChildFill="True">

<Button

Width="24"

Height="24"

HorizontalAlignment="Right"

Command="{x:Static com:CollectionCommands.Remove}"

CommandParameter="{Binding}"

DockPanel.Dock="Right"

Style="{DynamicResource MaterialDesignFloatingActionMiniAccentButton}"

ToolTip="从自选列表移除">

<md:PackIcon Kind="Minus" />

</Button>

<TextBlock

VerticalAlignment="Center"

FontSize="18"

Text="{Binding Name}" />

</DockPanel>

</DataTemplate>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<md:ColorZone Padding="12" Mode="Light">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Button

VerticalAlignment="Center"

IsDefault="True"

Style="{DynamicResource MaterialDesignToolButton}">

<md:PackIcon

Width="24"

Height="24"

Kind="Magnify"

Opacity="0.56" />

<!-- Click="Button\_Click" -->

</Button>

<ComboBox

x:Name="search"

Grid.Column="1"

MinWidth="200"

Margin="24,0,32,0"

md:HintAssist.Hint="搜索股票代码"

FontSize="18"

IsEditable="True"

Text="{Binding SearchText, ElementName=control, Mode=OneWayToSource, Delay=500}">

<!-- Text="{Binding SearchText, ElementName=control, Mode=OneWayToSource, Delay=500}" -->

<ComboBoxItem

x:Name="searching"

HorizontalAlignment="Center"

IsEnabled="False"

Visibility="Collapsed">

<ProgressBar

Width="36"

Height="36"

IsIndeterminate="True"

Style="{StaticResource MaterialDesignCircularProgressBar}"

Value="60" />

</ComboBoxItem>

</ComboBox>

</Grid>

</md:ColorZone>

<md:Card

Grid.Row="1"

Margin="5"

VerticalAlignment="Stretch">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="3\*" />

</Grid.ColumnDefinitions>

<ListBox

x:Name="list"

HorizontalContentAlignment="Stretch"

Background="{DynamicResource PrimaryHueLightBrush}"

Foreground="{DynamicResource PrimaryHueLightForegroundBrush}"

IsSynchronizedWithCurrentItem="True"

ItemsSource="{Binding StockList, ElementName=control}" />

<Grid

x:Name="container"

Grid.Column="1"

DataContext=""

TextElement.FontSize="18">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="4\*" />

</Grid.RowDefinitions>

<Grid.Resources>

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="20,10" />

<Setter Property="VerticalAlignment" Value="Center" />

</Style>

</Grid.Resources>

<Border Grid.ColumnSpan="2" Margin="0,0,0,15">

<TextBlock Margin="20,10" VerticalAlignment="Center">

<Run FontSize="32" Text="{Binding SelectedItem.Name, ElementName=list}" />

<LineBreak />

<Span Foreground="#AAAAAA">

<Run Text="{Binding SelectedItem.StockID, ElementName=list}" />

</Span>

</TextBlock>

</Border>

<Border Grid.Column="2" Margin="0,0,0,15">

<TextBlock Margin="20,10" Foreground="{Binding px\_change, Converter={StaticResource covChange}}">

<Run

FontSize="32"

FontWeight="Bold"

Text="{Binding last\_px}" />

<LineBreak />

<Span><Run Text="{Binding px\_change, Converter={StaticResource covChange}}" /><Run Text="{Binding px\_change\_rate, Converter={StaticResource covChange}, StringFormat=({0}), ConverterParameter=%}" /></Span>

</TextBlock>

</Border>

<Separator Grid.ColumnSpan="3" />

<Border Grid.Row="1" Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">开盘</Run>

<Run Text="{Binding open\_px}" />

</TextBlock>

</Border>

<Border

Grid.Row="1"

Grid.Column="1"

Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">最高</Run>

<Run Foreground="Red" Text="{Binding high\_px}" />

</TextBlock>

</Border>

<Border

Grid.Row="1"

Grid.Column="2"

Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">换手</Run>

<Run Text="{Binding turnover\_ratio}" />

</TextBlock>

</Border>

<Separator

Grid.Row="1"

Grid.ColumnSpan="3"

Height="5" />

<Border Grid.Row="2" Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">成交</Run>

<Run Text="{Binding business\_balance, StringFormat={}{0:N0}}" />

</TextBlock>

</Border>

<Border

Grid.Row="2"

Grid.Column="1"

Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">最低</Run>

<Run Foreground="LimeGreen" Text="{Binding low\_px}" />

</TextBlock>

</Border>

<Border

Grid.Row="2"

Grid.Column="2"

Margin="0,0,0,5">

<TextBlock>

<Run Foreground="#AAAAAA">量比</Run>

<Run Text="{Binding vol\_ratio}" />

</TextBlock>

</Border>

<Separator

Grid.Row="2"

Grid.ColumnSpan="3"

Height="5" />

<Border

x:Name="loading"

Grid.RowSpan="4"

Grid.ColumnSpan="3"

d:IsHidden="True"

Background="#66000000">

<ProgressBar

Width="36"

Height="36"

IsIndeterminate="True"

Style="{StaticResource MaterialDesignCircularProgressBar}"

Value="60" />

</Border>

</Grid>

<md:Snackbar

x:Name="errInfo"

Grid.ColumnSpan="2"

IsActive="{Binding HasException, ElementName=control}"

MessageQueue="{md:MessageQueue}">

<md:SnackbarMessage

ActionClick="SnackbarMessage\_ActionClick"

ActionContent="确定"

Content="{Binding Path=ErrorInfo, ElementName=control}" />

</md:Snackbar>

</Grid>

</md:Card>

</Grid>

</UserControl>

### StockDetail.xaml

<UserControl

x:Class="SimpleTracer.Controls.StockDetail"

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

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

xmlns:cov="clr-namespace:SimpleTracer.Converters"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:SimpleTracer.Controls"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:mdl="clr-namespace:SimpleTracer.Models"

d:DataContext="{d:DesignInstance Type={x:Type mdl:RealTimeQuotationModel}}"

d:DesignHeight="360"

d:DesignWidth="640"

mc:Ignorable="d">

<UserControl.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="/SimpleTracer;component/Resources/DataTemplates.xaml" />

<ResourceDictionary Source="/SimpleTracer;component/Resources/Styles.xaml" />

</ResourceDictionary.MergedDictionaries>

<cov:StockChangeConverter x:Key="covChange" />

<Style BasedOn="{StaticResource MaterialDesignDarkSeparator}" TargetType="{x:Type Separator}">

<Setter Property="VerticalAlignment" Value="Bottom" />

<Setter Property="Margin" Value="8,0" />

</Style>

</ResourceDictionary>

</UserControl.Resources>

<Grid>

<GroupBox>

<GroupBox.Header>

<Grid>

<StackPanel Orientation="Horizontal">

<Button Command="{x:Static md:Transitioner.MoveFirstCommand}" Style="{StaticResource backButtonStyle}" />

<TextBlock

VerticalAlignment="Center"

Foreground="White"

Style="{DynamicResource MaterialDesignTitleTextBlock}">

详细信息

</TextBlock>

</StackPanel>

</Grid>

</GroupBox.Header>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="5\*" />

<ColumnDefinition Width="4\*" />

</Grid.ColumnDefinitions>

<md:Card Margin="5" VerticalAlignment="Stretch">

<Grid x:Name="container" TextElement.FontSize="18">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="2\*" />

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<Grid.Resources>

<Style TargetType="TextBlock">

<Setter Property="Margin" Value="10,5" />

<Setter Property="VerticalAlignment" Value="Center" />

</Style>

<Style TargetType="ContentControl">

<Setter Property="ContentTemplate">

<Setter.Value>

<DataTemplate>

<Grid Margin="12,0,0,0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" MinWidth="75" />

<ColumnDefinition Width="\*" />

</Grid.ColumnDefinitions>

<TextBlock

VerticalAlignment="Center"

Foreground="#AAAAAA"

Text="{Binding Tag, RelativeSource={RelativeSource FindAncestor, AncestorType={x:Type ContentControl}}}" />

<ContentPresenter Grid.Column="1" Content="{Binding}" />

</Grid>

</DataTemplate>

</Setter.Value>

</Setter>

</Style>

</Grid.Resources>

<Border Grid.ColumnSpan="2" Margin="0,0,0,15">

<TextBlock Margin="20,10" VerticalAlignment="Center">

<Run FontSize="32" Text="{Binding prod\_name}" />

<LineBreak />

<Span Foreground="#AAAAAA">

<Run Text="{Binding prod\_code}" />

</Span>

</TextBlock>

</Border>

<Border Grid.Column="1" Margin="0,0,0,15">

<TextBlock Margin="20,10" Foreground="{Binding px\_change, Converter={StaticResource covChange}}">

<Run

FontSize="36"

FontWeight="Bold"

Text="{Binding last\_px}" />

<LineBreak />

<Span>

<Run Text="{Binding px\_change, Converter={StaticResource covChange}}" />

<Run Text="{Binding px\_change\_rate, Converter={StaticResource covChange}, StringFormat=({0}), ConverterParameter=%}" />

</Span>

</TextBlock>

</Border>

<Separator Grid.ColumnSpan="2" />

<ContentControl

Grid.Row="1"

Margin="0,0,0,5"

Tag="开盘">

<TextBlock Text="{Binding open\_px}" />

</ContentControl>

<ContentControl

Grid.Row="1"

Grid.Column="1"

Margin="0,0,0,5"

Tag="最高">

<TextBlock>

<Run Foreground="Red" Text="{Binding high\_px}" />

</TextBlock>

</ContentControl>

<ContentControl

Grid.Row="4"

Margin="0,0,0,5"

Tag="换手">

<TextBlock>

<Run Text="{Binding turnover\_ratio, StringFormat=\{0\}%}" />

</TextBlock>

</ContentControl>

<Separator

Grid.Row="1"

Grid.ColumnSpan="2"

Height="5" />

<ContentControl

Grid.Row="2"

Margin="0,0,0,5"

Tag="昨收">

<TextBlock>

<Run Text="{Binding preclose\_px}" />

</TextBlock>

</ContentControl>

<ContentControl

Grid.Row="2"

Grid.Column="1"

Margin="0,0,0,5"

Tag="最低">

<TextBlock>

<Run Foreground="LimeGreen" Text="{Binding low\_px}" />

</TextBlock>

</ContentControl>

<ContentControl

Grid.Row="4"

Grid.Column="1"

Margin="0,0,0,5"

Tag="量比">

<TextBlock>

<Run Text="{Binding vol\_ratio}" />

</TextBlock>

</ContentControl>

<Separator

Grid.Row="2"

Grid.ColumnSpan="2"

Height="5" />

<ContentControl

Grid.Row="3"

Margin="0,0,0,5"

Tag="成交量">

<TextBlock>

<Run Text="{Binding business\_amount, StringFormat=\{0:N0\}}" />

</TextBlock>

</ContentControl>

<ContentControl

Grid.Row="3"

Grid.Column="1"

Margin="0,0,0,5"

Tag="成交额">

<TextBlock Language="zh-cn">

<Run Text="{Binding business\_balance, StringFormat=\{0:C\}}" />

</TextBlock>

</ContentControl>

<Separator

Grid.Row="3"

Grid.ColumnSpan="2"

Height="5" />

</Grid>

</md:Card>

<!-- 右侧 -->

<md:Card

Grid.Column="1"

Margin="5"

VerticalAlignment="Stretch">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<Grid Margin="10" TextElement.FontSize="24">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<StackPanel

HorizontalAlignment="Center"

Orientation="Horizontal"

TextElement.Foreground="{DynamicResource PrimaryHueMidBrush}">

<md:PackIcon

Width="24"

Height="24"

VerticalAlignment="Center"

Kind="PlusBox" />

<TextBlock Text="买入" />

</StackPanel>

<StackPanel

Grid.Column="1"

HorizontalAlignment="Center"

Orientation="Horizontal"

TextElement.Foreground="{DynamicResource SecondaryAccentBrush}">

<md:PackIcon

Width="24"

Height="24"

VerticalAlignment="Center"

Kind="MinusBox" />

<TextBlock Text="卖出" />

</StackPanel>

</Grid>

<Grid

Grid.Row="1"

Grid.RowSpan="8"

Grid.Column="2">

<Grid.ColumnDefinitions>

<ColumnDefinition />

<ColumnDefinition />

</Grid.ColumnDefinitions>

<Separator

Width="16"

Height="Auto"

Margin="0,8,-8,8"

HorizontalAlignment="Right"

VerticalAlignment="Stretch" />

<ListView ItemsSource="{Binding bid\_grp}">

<ListView.View>

<GridView>

<GridViewColumn DisplayMemberBinding="{Binding id, StringFormat=买\{0\}}" />

<GridViewColumn DisplayMemberBinding="{Binding transaction\_count}" Header="数量" />

<GridViewColumn DisplayMemberBinding="{Binding price}" Header="价格" />

</GridView>

</ListView.View>

</ListView>

<ListView Grid.Column="1" ItemsSource="{Binding offer\_grp}">

<ListView.View>

<GridView>

<GridViewColumn DisplayMemberBinding="{Binding id, StringFormat=卖\{0\}}" />

<GridViewColumn DisplayMemberBinding="{Binding transaction\_count}" Header="数量" />

<GridViewColumn DisplayMemberBinding="{Binding price}" Header="价格" />

</GridView>

</ListView.View>

</ListView>

</Grid>

</Grid>

</md:Card>

<!-- loading图标 -->

<ContentControl

x:Name="loading"

Background="#66000000"

Visibility="Collapsed">

<ProgressBar

Width="36"

Height="36"

IsIndeterminate="True"

Style="{StaticResource MaterialDesignCircularProgressBar}"

Value="60" />

</ContentControl>

</Grid>

</GroupBox>

</Grid>

</UserControl>

### StockChangeConverter.cs

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Data;

using static Microsoft.VisualBasic.Information;

using static Microsoft.VisualBasic.Conversion;

using System.Windows.Media;

namespace SimpleTracer.Converters

{

/// <summary>

/// 股票涨跌值的XAML值转换器

/// </summary>

public class StockChangeConverter : IValueConverter

{

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

{

if (IsNumeric(value))

{

var v = Val(value);

//转换涨跌值到颜色

if (targetType == typeof(Brush) || targetType.IsSubclassOf(typeof(Brush)))

{

return v > 0 ? Brushes.Red

: v < 0 ? Brushes.LimeGreen

: Brushes.Black;

}

//格式化涨跌值

else if (targetType == typeof(string))

{

return (v > 0 ? ("+" + value) : value ) +

(parameter?.ToString() == "%" ? "%" : "");

}

}

throw new InvalidCastException($"无法执行从 '{value?.GetType().FullName}' 到 '{targetType.FullName}' 的转换。");

}

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

{

throw new NotImplementedException();

}

}

}

### DisplayText.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace SimpleTracer.Extension

{

public sealed class DisplayText:DependencyObject

{

public static string GetDisplayText(DependencyObject obj)

{

return (string)obj.GetValue(DisplayTextProperty);

}

public static void SetDisplayText(DependencyObject obj, string value)

{

obj.SetValue(DisplayTextProperty, value);

}

// Using a DependencyProperty as the backing store for DisplayText. This enables animation, styling, binding, etc...

public static readonly DependencyProperty DisplayTextProperty =

DependencyProperty.RegisterAttached("DisplayText", typeof(string),

typeof(DisplayText), new PropertyMetadata(""));

}

}

### ExtensionMethods.cs

using Newtonsoft.Json;

using SimpleTracer.Controls;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace SimpleTracer.Extension

{

/// <summary>

/// 包含了一些实用的扩展方法

/// </summary>

static class ExtensionMethods

{

/// <summary>

/// 设置是否阻止WebBrowser报告JavaScript错误

/// </summary>

/// <param name="webBrowser"></param>

/// <param name="silent">是否忽略JavaScript错误</param>

public static void SetSilentState(this WebBrowser webBrowser, bool silent)

{

FieldInfo fi = typeof(WebBrowser).GetField("\_axIWebBrowser2", BindingFlags.Instance | BindingFlags.NonPublic);

if (fi != null)

{

object browser = fi.GetValue(webBrowser);

if (browser != null)

browser.GetType().InvokeMember("Silent", BindingFlags.SetProperty, null, browser, new object[] { silent });

}

}

/// <summary>

/// 向最接近的处理SnackBar消息的父级元素设置SnackBar的内容

/// </summary>

/// <param name="element">开始查找的元素</param>

/// <param name="info">要设置的消息内容</param>

/// <returns>如果找到，返回<see langword="true"/>，否则返回<see langword="false"/></returns>

public static bool SetSnackBarInfo(this FrameworkElement element, object info)

{

DependencyObject parent = element;

do

{

if (parent is MainWindow s)

{

s.InfoContent = info;

return true;

}

} while ((parent = LogicalTreeHelper.GetParent(parent)) != null);

return false;

}

/// <summary>

/// 对简单对象进行深复制

/// </summary>

/// <typeparam name="T">对象的类型</typeparam>

/// <param name="plainObject">要复制的对象</param>

/// <returns></returns>

public static T DeepClone<T>(this T plainObject)

{

return JsonConvert.DeserializeObject<T>(JsonConvert.SerializeObject(plainObject));

}

/// <summary>

/// 通过使用默认的相等比较器对值进行比较生成一个字典和一个字典中值序列的差集。

/// </summary>

/// <typeparam name="TKey"></typeparam>

/// <typeparam name="TValue"></typeparam>

/// <param name="dictionary"></param>

/// <param name="values"></param>

/// <returns></returns>

public static IEnumerable<KeyValuePair<TKey,TValue>> Except<TKey, TValue>(

this IEnumerable<KeyValuePair<TKey, TValue>> dictionary, IEnumerable<TValue> values

)

{

foreach (var p in dictionary)

{

if (!values.Contains(p.Value))

{

yield return p;

}

}

}

public static T As<T>(this object obj) where T : class

{

return obj as T;

}

}

}

### UtilMethods.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SimpleTracer.Extension

{

/// <summary>

/// 包含了一些辅助方法

/// </summary>

public static class UtilMethods

{

/// <summary>

/// 将对象<paramref name="obj"/>的所有公共属性值赋给对象<paramref name="target"/>。

/// 结果相当于在不改变对<paramref name="target"/>对象任何引用的情况下，

/// 用<paramref name="obj"/>替换了该对象。<br />

/// </summary>

/// <typeparam name="T">对象的类型，必须有无参构造函数。</typeparam>

/// <param name="target">被赋值的对象</param>

/// <param name="obj">提供值的对象</param>

public static void Assign<T>(ref T target, T obj) where T : new()

{

var type = typeof(T);

var props = type.GetProperties();

foreach (var p in props)

{

try

{

p.SetValue(target, p.GetValue(obj));

}

catch (Exception e)

{

Logger.Log($"'{type.Name}' 上的属性 '{p?.Name}' 赋值失败！\r\n" + e.ToString());

}

}

}

/// <summary>

/// 把时间戳转换为<see cref="DateTime"/>

/// </summary>

/// <param name="timestamp"></param>

/// <returns></returns>

public static DateTime ToDateTime(int timestamp)

{

return DateTimeOffset.FromUnixTimeSeconds(timestamp).DateTime;

}

/// <summary>

/// 把<see cref="DateTime"/>转换为时间戳

/// </summary>

/// <param name="d"></param>

/// <returns></returns>

public static int ToTimestamp(DateTime d)

{

return (int)new DateTimeOffset(d).ToUnixTimeSeconds();

}

public static bool IsBetween(double value, double lower, double upper)

{

return value >= lower && value <= upper;

}

}

}

### AlertConditionList.cs

using Newtonsoft.Json;

using PropertyChanged;

using SimpleTracer.Analysis;

using SimpleTracer.Extension;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SimpleTracer.Models

{

/// <summary>

/// 预警条件列表

/// </summary>

public class AlertConditionList : ObservableCollection<AlertCondition>

{

public AlertConditionList()

{

//添加默认的参数

Add(new AlertCondition("priceOver", "当前股价超过", 100));

Add(new AlertCondition("priceUnder", "当前股价低于", 100)

{ paramType = AlertParamType.Less });

Add(new AlertCondition ("dailyRateUp", "日涨幅超过", 1.5 ));

Add(new AlertCondition ("dailyRateDown", "日跌幅超过", 0.01 ));

Add(new AlertCondition ("fiveMinRateUp", "5分钟涨幅超过", 0.8 ));

Add(new AlertCondition ("fiveMinRateDown", "5分钟跌幅超过", 1 ));

}

public AlertConditionList (bool noDefaultList)

{

}

protected override void InsertItem(int index, AlertCondition item)

{

AlertCondition condition;

if ((condition = Items.Where(c => c.paramName == item.paramName).FirstOrDefault()) != null)

{

//以下代码来自基类方法

//CheckReentrancy();

UtilMethods.Assign(ref condition, item);

//OnPropertyChanged(new PropertyChangedEventArgs("Count"));

//OnPropertyChanged(new PropertyChangedEventArgs("Item[]"));

//OnCollectionChanged(new NotifyCollectionChangedEventArgs(NotifyCollectionChangedAction.Add, item, index));

}

else

base.InsertItem(index, item);

}

}

/// <summary>

/// 预警条件模型

/// </summary>

[AddINotifyPropertyChangedInterface]

public class AlertCondition

{

/// <summary>参数名称</summary>

public string paramName { get; set; }

/// <summary>参数显示名称</summary>

[JsonIgnore]

public string DisplayText { get; set; }

/// <summary>预警参数值</summary>

public double alertValue { get; set; }

/// <summary>当前参数值</summary>

public double currentValue { get; set; }

/// <summary>是否启用</summary>

public bool enabled { get; set; } = false;

/// <summary>参数类型</summary>

public AlertParamType paramType { get; set; }

public AlertCondition() { }

public AlertCondition (string name, string displayText, double value)

{

DisplayText = displayText;

paramName = name;

alertValue = value;

}

/// <summary>

/// 检查参数是否达到预警

/// </summary>

/// <param name="values">筛选过的股票数据</param>

/// <param name="alertValue">out(引用)参数，返回一个达到预警的值</param>

/// <returns></returns>

public bool CheckAlert(Dictionary<double, RealTimeQuotationModel> values, out double alertValue)

{

alertValue = 0;

if (!enabled)

return false;

return CheckerFactory.Create(paramName)?.Check(values, this, out alertValue) ?? false;

}

}

/// <summary>参数类型</summary>

public enum AlertParamType

{

/// <summary>大于</summary>

Greater = 0,

/// <summary>小于</summary>

Less = 1

}

}

### AppConfig.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using PropertyChanged;

using System.Diagnostics;

using SimpleTracer.Extension;

using System.ComponentModel;

using System.Collections.Specialized;

namespace SimpleTracer.Models

{

/// <summary>

/// 程序配置对象，所有属性使用camel命名法以符合JSON规范

/// </summary>

[AddINotifyPropertyChangedInterface]

public class AppConfig

{

/// <summary>

/// 监控股票列表

/// </summary>

public List<StockInfo> watchList { get; set; } =

new List<StockInfo>();

/// <summary>

/// 推送API key

/// </summary>

public string pushKey { get; set; }

/// <summary>

/// 预警参数列表

/// </summary>

public StockAlert alertList { get; set; }

}

/// <summary>

/// 股票预警信息类

/// </summary>

[AddINotifyPropertyChangedInterface]

public class StockAlert : Dictionary<string, AlertConditionList>

{

/// <summary>

/// 通过给定的股票代码获取或设置对应的预警条件列表，

/// 当<paramref name="stockId"/>不存在时会自动创建项

/// </summary>

/// <param name="stockId">股票代码</param>

/// <returns></returns>

public new AlertConditionList this[string stockId]

{

get

{

if (!ContainsKey(stockId))

{

base[stockId] = new AlertConditionList();

return base[stockId];

}

return base[stockId];

}

set

{

base[stockId] = value;

}

}

}

}

### RealTimeQuotationModel.cs

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using PropertyChanged;

namespace SimpleTracer.Models

{

#pragma warning disable CS1591

/// <summary>

/// 实时行情API返回的Model类

/// </summary>

[AddINotifyPropertyChangedInterface]

public class RealTimeQuotationModel

{

/// <summary>股票代码</summary>

public string prod\_code { get; set; }

/// <summary>股票名称</summary>

public string prod\_name { get; set; }

/// <summary>请求时间戳</summary>

public double data\_timestamp { get; set; }

public double open\_px { get; set; }

public double high\_px { get; set; }

public double low\_px { get; set; }

public double last\_px { get; set; }

public double preclose\_px { get; set; }

public long business\_amount { get; set; }

public double business\_balance { get; set; }

public int shares\_per\_hand { get; set; }

public long business\_amount\_in { get; set; }

public long business\_amount\_out { get; set; }

public double up\_px { get; set; }

public double down\_px { get; set; }

public double amplitude { get; set; }

public long circulation\_value { get; set; }

public double eps { get; set; }

public double pe\_rate { get; set; }

public double px\_change { get; set; }

public double px\_change\_rate { get; set; }

public double dyn\_pb\_rate { get; set; }

public double turnover\_ratio { get; set; }

public double vol\_ratio { get; set; }

public TransactionList bid\_grp { get; set; }

public TransactionList offer\_grp { get; set; }

}

[JsonConverter(typeof(TransactionListConverter))]

public class TransactionList : List<TransactionInfo>

{

}

[AddINotifyPropertyChangedInterface]

public class TransactionInfo

{

public decimal price { get; set; }

public int transaction\_count { get; set; }

public int id { get; set; }

}

/// <summary>

/// 用于JSON数据序列化和反序列化<see cref="TransactionList"/>的类

/// </summary>

public class TransactionListConverter : JsonConverter<TransactionList>

{

public override TransactionList ReadJson(JsonReader reader, Type objectType, TransactionList existingValue, bool hasExistingValue, JsonSerializer serializer)

{

var value = reader.Value.ToString();

var v = value.Split(new[] { "," }, StringSplitOptions.RemoveEmptyEntries);

if (v.Length == 0 || v.Length % 3 != 0)

throw new ArgumentException("无效的元素个数");

var result = new TransactionList();

for (int i = 0; i < v.Length; i += 3)

{

var info = new TransactionInfo

{

price = decimal.Parse(v[i]),

transaction\_count = int.Parse(v[i + 1]),

id = i / 3 + 1

};

result.Add(info);

}

return result;

}

public override void WriteJson(JsonWriter writer, TransactionList value, JsonSerializer serializer)

{

string result = "";

foreach (var item in value)

{

result += $"{item.price},{item.transaction\_count},0,";

}

writer.WriteValue(result);

}

}

}

### StockInfo.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using PropertyChanged;

namespace SimpleTracer.Models

{

/// <summary>

/// 简单股票信息的Model类

/// </summary>

[AddINotifyPropertyChangedInterface]

public class StockInfo

{

/// <summary>

/// 股票名称

/// </summary>

public string Name { get; set; }

/// <summary>

/// 股票代码

/// </summary>

public string StockID { get; set; }

public override string ToString()

{

return $"{{Name = {Name},StockID = {StockID}}}";

}

}

}

### StockQueryModel.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

using PropertyChanged;

namespace SimpleTracer.Models

{

/// <summary>

/// 简单股票信息的Model类

/// </summary>

[AddINotifyPropertyChangedInterface]

public class StockInfo

{

/// <summary>

/// 股票名称

/// </summary>

public string Name { get; set; }

/// <summary>

/// 股票代码

/// </summary>

public string StockID { get; set; }

public override string ToString()

{

return $"{{Name = {Name},StockID = {StockID}}}";

}

}

}

### DataTemplates.xaml

<ResourceDictionary

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

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

xmlns:com="clr-namespace:SimpleTracer.Commands"

xmlns:m="clr-namespace:SimpleTracer.Models"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

xmlns:sys="clr-namespace:System;assembly=mscorlib">

<DataTemplate DataType="{x:Type m:StockQueryModel}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto" />

<ColumnDefinition />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<TextBlock

Margin="0,0,20,0"

VerticalAlignment="Center"

Text="{Binding prod\_code}" />

<TextBlock

Grid.Column="1"

VerticalAlignment="Center"

Foreground="#AAAAAA"

Text="{Binding prod\_name}" />

<Button

Grid.Column="2"

Command="{x:Static com:CollectionCommands.Add}"

CommandParameter="{Binding}"

Style="{DynamicResource MaterialDesignFloatingActionMiniAccentButton}"

ToolTip="添加到自选列表">

<md:PackIcon Kind="Plus" />

</Button>

</Grid>

</DataTemplate>

<DataTemplate DataType="{x:Type sys:Exception}">

<DockPanel LastChildFill="True">

<md:PackIcon

Margin="0,0,8,0"

VerticalAlignment="Center"

Kind="AlertCircle" />

<TextBlock xml:space="preserve" TextTrimming="WordEllipsis">发生错误：<Run Text="{Binding Message, Mode=OneWay}" /></TextBlock>

</DockPanel>

</DataTemplate>

<DataTemplate DataType="{x:Type m:NotificationData}">

<StackPanel>

<TextBlock Margin="10,5" FontSize="16">

<Run Text="{Binding StockInfo.StockID}" />

<Run Text="{Binding StockInfo.Name, StringFormat=(\{0\})}" />

</TextBlock>

<ItemsControl Margin="16,0,0,0" ItemsSource="{Binding Conditions}" />

</StackPanel>

</DataTemplate>

<DataTemplate DataType="{x:Type m:AlertCondition}">

<StackPanel Orientation="Horizontal">

<TextBlock Margin="6,3">

<Span>

参数'<Run Text="{Binding paramName}" />

'的值'<Run Foreground="Red" Text="{Binding currentValue}" />

'超过了预警值'<Run Text="{Binding alertValue}" />

'</Span>

</TextBlock>

</StackPanel>

</DataTemplate>

</ResourceDictionary>

### Styles.xaml

<ResourceDictionary

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

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

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes">

<!-- 这是包含各种样式（皮肤）的文件 -->

<Style

x:Key="backButtonStyle"

BasedOn="{StaticResource MaterialDesignFloatingActionMiniAccentButton}"

TargetType="Button">

<Setter Property="Margin" Value="0,0,8,0" />

<Setter Property="Command" Value="{x:Static md:Transitioner.MovePreviousCommand}" />

<Setter Property="Width" Value="28" />

<Setter Property="Height" Value="28" />

<Setter Property="Margin" Value="0,0,8,0" />

<Setter Property="Content">

<Setter.Value>

<md:PackIcon

Width="20"

Height="20"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Foreground="White"

Kind="ArrowLeft" />

</Setter.Value>

</Setter>

</Style>

<Style

x:Key="dialogButtonStyle"

BasedOn="{StaticResource {x:Type Button}}"

TargetType="Button">

<Setter Property="HorizontalAlignment" Value="Right" />

<Setter Property="Margin" Value="10,8" />

</Style>

<Style

x:Key="dialogDefaultButtonStyle"

BasedOn="{StaticResource MaterialDesignRaisedAccentButton}"

TargetType="Button">

<Setter Property="HorizontalAlignment" Value="Right" />

<Setter Property="Margin" Value="12,8" />

</Style>

</ResourceDictionary>

### Api.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Collections.ObjectModel;

using SimpleTracer.Models;

using System.Net.Http;

using System.Net.Http.Headers;

using Newtonsoft.Json;

using Newtonsoft.Json.Linq;

using System.Diagnostics;

using System.Net;

using System.Reflection;

namespace SimpleTracer

{

/// <summary>

/// 提供股票相关数据的API

/// </summary>

public class Api

{

internal const string AppKey = "24850853";

internal const string AppSecret = "864f0b05c694b0466ad8757ec6f8a981";

internal const string AppCode = "d33be00202dd45bcadbd7d2a01357858";

const string BaseUrl = "http://stock.api51.cn/";

/// <summary>

/// 异步获取实时行情数据

/// </summary>

/// <param name="stockCode">股票代码</param>

/// <param name="fields">要查询的字段，可用的字段列表参见<see cref="RealTimeQuotationModel"/>类的属性。</param>

/// <returns>行情数据模型</returns>

public static async Task<RealTimeQuotationModel> GetRealTimeQuotationAsync(

string stockCode, string[] fields)

{

HttpClient client = new HttpClient();

client.DefaultRequestHeaders.Add("Authorization", "APPCODE " + AppCode);

var rsp = await client.GetAsync($"{BaseUrl}real?en\_prod\_code={stockCode}&fields=" +

string.Join("%2C", fields));

if (rsp.IsSuccessStatusCode)

{

var json = rsp.Content.ReadAsStringAsync().Result;

var data = JObject.Parse(json)["data"]?["snapshot"];

var fieldList = data?["fields"]?.ToObject<List<string>>();

var stockinfo = data[stockCode];

//填充Model类

var result = new RealTimeQuotationModel();

result.prod\_code = stockCode;

var t = typeof(RealTimeQuotationModel);

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

{

PropertyInfo p = null;

try

{

p = t.GetProperty(fieldList[i]);

p.SetValue(result, stockinfo[i].ToObject(p.PropertyType));

}

catch(Exception e)

{

Logger.Log($"'{nameof(RealTimeQuotationModel)}' 上的属性 '{p?.Name}' 设置失败！\r\n"+e.ToString());

}

}

return result;

}

Logger.Log(rsp.Content.ReadAsStringAsync().Result);

throw new WebException((int)rsp.StatusCode + " " + rsp.ReasonPhrase);

}

/// <summary>

/// 异步查询股票代码

/// </summary>

/// <param name="keyWord">要查询的股票代码关键字</param>

/// <returns></returns>

public static async Task<List<StockQueryModel>> GetStockQueryAsync(string keyWord)

{

HttpClient client = new HttpClient();

client.DefaultRequestHeaders.Add("Authorization", "APPCODE " + AppCode);

var rsp = await client.GetAsync($"{BaseUrl}wizard?data\_count=50&en\_finance\_mic=SS,SZ&prod\_code={keyWord}");

if (rsp.IsSuccessStatusCode)

{

var json = rsp.Content.ReadAsStringAsync().Result;

var data = JObject.Parse(json)?["data"];

return data.ToObject<List<StockQueryModel>>();

}

Logger.Log(rsp.Content.ReadAsStringAsync().Result);

throw new WebException((int)rsp.StatusCode + " " + rsp.ReasonPhrase);

}

public static void Main()

{

var r = GetStockQueryAsync("000001").Result;

Debug.Print(JsonConvert.SerializeObject(r));

}

}

}

### AppConfigManager.cs

using SimpleTracer.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Newtonsoft.Json;

using System.IO;

namespace SimpleTracer

{

//使用了单实例（Singleton）设计模式，确保AppConfigManager有且只有一个实例

/// <summary>

/// 管理程序配置

/// </summary>

public sealed class AppConfigManager

{

/// <summary>

/// 私有构造函数，阻止外界创建实例

/// </summary>

private AppConfigManager() { }

const string fileName = "config.json";

//唯一实例

private static AppConfigManager manager = new AppConfigManager();

bool isLoaded = false;

/// <summary>

/// 返回管理器的唯一实例

/// </summary>

/// <returns></returns>

public static AppConfigManager GetManager()

{

if (!manager.isLoaded)

manager.Load();

return manager;

}

/// <summary>

/// 获取配置对象

/// </summary>

public AppConfig Config { get; private set; }

/// <summary>

/// 加载配置

/// </summary>

public void Load()

{

try

{

if (!File.Exists(fileName))

{

Logger.Log("找不到配置文件，尝试创建新文件。");

Config = new AppConfig();

Save();

isLoaded = true;

}

else

{

var json = File.ReadAllText(fileName);

Config = JsonConvert.DeserializeObject<AppConfig>(json);

isLoaded = true;

Logger.Log("加载配置成功\n"+ JsonConvert.SerializeObject(Config));

}

}

catch (Exception e)

{

Logger.Log("加载配置发生错误：\n" + e);

}

}

/// <summary>

/// 保存配置

/// </summary>

public void Save()

{

try

{

if (Config != null)

{

File.WriteAllText(fileName, JsonConvert.SerializeObject(Config));

Logger.Log("保存配置成功");

}

}

catch (Exception e)

{

Logger.Log("保存配置发生错误：\n" + e);

}

}

}

}

### App.xaml.cs

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Runtime.ExceptionServices;

using System.Threading.Tasks;

using System.Windows;

namespace SimpleTracer

{

/// <summary>

/// App.xaml 的交互逻辑

/// </summary>

public partial class App : Application

{

public App()

{

AppDomain.CurrentDomain.UnhandledException += CurrentDomain\_UnhandledException;

Global.Main(null);

Global.CurrentDispatcher = Dispatcher;

}

[HandleProcessCorruptedStateExceptions]

private void CurrentDomain\_UnhandledException(object sender, UnhandledExceptionEventArgs e)

{

var ex = e.ExceptionObject as Exception;

var r = MessageBox.Show("发生未处理的异常：\r\n" + ex?.ToString() +

"\r\n\r\n继续运行可能发生不可预计的行为。是否终止程序？",

"错误", MessageBoxButton.YesNo, MessageBoxImage.Error, MessageBoxResult.OK);

if (r == MessageBoxResult.OK)

{

Shutdown(ex?.HResult ?? -1);

}

}

}

}

### Mainwindow.xaml

<Window

x:Class="SimpleTracer.MainWindow"

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

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

xmlns:Controls="clr-namespace:SimpleTracer.Controls"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:SimpleTracer"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:md="http://materialdesigninxaml.net/winfx/xaml/themes"

x:Name="window"

Title="简易查"

Width="960"

Height="640"

Background="{x:Null}"

FontSize="18"

mc:Ignorable="d">

<Window.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Resources/DataTemplates.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Window.Resources>

<WindowChrome.WindowChrome>

<WindowChrome CaptionHeight="72" GlassFrameThickness="3.2,72,3.2,3.2" />

</WindowChrome.WindowChrome>

<Grid>

<md:DrawerHost IsLeftDrawerOpen="{Binding IsChecked, ElementName=menu}">

<md:DrawerHost.LeftDrawerContent>

<Border Width="200">

<StackPanel>

<ToggleButton

Margin="16"

HorizontalAlignment="Right"

IsChecked="{Binding IsChecked, ElementName=menu}"

Style="{StaticResource MaterialDesignHamburgerToggleButton}"

WindowChrome.IsHitTestVisibleInChrome="True" />

<ListBox x:Name="panelList" SelectionChanged="panelList\_SelectionChanged">

<ListBoxItem Tag="{x:Type Controls:GlobalView}">

<StackPanel Orientation="Horizontal">

<md:PackIcon Margin="0,0,8,0" Kind="TrendingUp" />

<TextBlock>大盘走势</TextBlock>

</StackPanel>

</ListBoxItem>

<ListBoxItem Tag="{x:Type Controls:StockAnalyze}">

<StackPanel Orientation="Horizontal">

<md:PackIcon Margin="0,0,8,0" Kind="ChartTimeline" />

<TextBlock>自选分析</TextBlock>

</StackPanel>

</ListBoxItem>

<ListBoxItem>

<StackPanel Orientation="Horizontal">

<md:PackIcon Margin="0,0,8,0" Kind="Settings" />

<TextBlock>软件设置</TextBlock>

</StackPanel>

</ListBoxItem>

</ListBox>

</StackPanel>

</Border>

</md:DrawerHost.LeftDrawerContent>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition />

</Grid.RowDefinitions>

<md:ColorZone

Height="72"

Padding="16"

Background="{x:Null}"

Foreground="White">

<md:ColorZone.Effect>

<DropShadowEffect ShadowDepth="0" Color="Gray" />

</md:ColorZone.Effect>

<!-- Mode="PrimaryLight" -->

<DockPanel>

<ToggleButton

x:Name="menu"

Style="{DynamicResource MaterialDesignHamburgerToggleButton}"

WindowChrome.IsHitTestVisibleInChrome="True" />

<md:PopupBox

VerticalAlignment="Bottom"

DockPanel.Dock="Right"

WindowChrome.IsHitTestVisibleInChrome="True" />

<TextBlock

Margin="16,0,0,0"

VerticalAlignment="Center"

FontSize="22"><Run Text="简易查 股票助手" /></TextBlock>

</DockPanel>

</md:ColorZone>

<Grid Grid.Row="1" Background="{DynamicResource MaterialDesignPaper}">

<Border x:Name="mainContent">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<StackPanel

Margin="10,10,10,0"

HorizontalAlignment="Center"

VerticalAlignment="Center"

Orientation="Horizontal">

<md:PackIcon

Width="36"

Height="36"

VerticalAlignment="Center"

Foreground="{DynamicResource SecondaryAccentBrush}"

Kind="Home" />

<TextBlock FontSize="36" Foreground="{DynamicResource SecondaryAccentBrush}">主页</TextBlock>

</StackPanel>

<Grid Grid.Row="1">

<md:Card Margin="10" VerticalAlignment="Stretch">

<Border Margin="100,60" VerticalAlignment="Top">

<Grid>

<StackPanel>

<TextBlock FontSize="36">关于</TextBlock>

<Separator Style="{DynamicResource MaterialDesignDarkSeparator}" />

<TextBlock Margin="0,20,0,0" FontSize="18">

<Span FontSize="24">简易查 股票助手</Span>

<LineBreak />

<Span Foreground="Gray">ver 0.5.0 Alpha</Span>

<LineBreak />

<LineBreak />

<Span>Copyright © 2018 SwingCosmic</Span>

<LineBreak />

<LineBreak />

<Bold>Powered By:</Bold>

<LineBreak />

<Span>

Material Design In Xaml Toolkit

</Span>

</TextBlock>

</StackPanel>

</Grid>

</Border>

</md:Card>

</Grid>

</Grid>

</Border>

<md:Snackbar

x:Name="infoSnackbar"

Margin="0,0,0,10"

IsActive="{Binding HasInfo, ElementName=window}"

MessageQueue="{md:MessageQueue}">

<md:SnackbarMessage

ActionClick="closeSnackbar\_ActionClick"

ActionContent="确定"

Content="{Binding InfoContent, ElementName=window}" />

</md:Snackbar>

</Grid>

</Grid>

</md:DrawerHost>

</Grid>

</Window>

### Mainwindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

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 System.Collections.ObjectModel;

using System.Globalization;

using System.Diagnostics;

using SimpleTracer.Extension;

using System.Media;

namespace SimpleTracer

{

/// <summary>

/// MainWindow.xaml 的交互逻辑

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

//SystemParameters.StaticPropertyChanged += SystemParameters\_StaticPropertyChanged;

notification = new NotificationWindow();

Tasks.Notificator.AlertOccurred += Notificator\_AlertOccurred;

}

private NotificationWindow notification;

private void Notificator\_AlertOccurred(object sender, Tasks.AlertEventArgs e)

{

Dispatcher.InvokeAsync(() =>

{

//出现在桌面右下角

notification.Left = SystemParameters.WorkArea.Width - notification.Width;

notification.Top = SystemParameters.WorkArea.Height - notification.Height;

notification.NotificationList.Add(e.AlertContent);

//显示并激活窗口，播放提示音

if (!notification.IsVisible)

notification.Show();

notification.Activate();

SystemSounds.Exclamation.Play();

});

}

public ObservableCollection<UserControl> AppControls { get; }

= new ObservableCollection<UserControl>();

/// <summary>

/// 获取或者设置提示信息内容

/// </summary>

public object InfoContent

{

get { return (string)GetValue(InfoContentProperty); }

set { SetValue(InfoContentProperty, value); }

}

// Using a DependencyProperty as the backing store for InfoContent. This enables animation, styling, binding, etc...

public static readonly DependencyProperty InfoContentProperty =

DependencyProperty.Register("InfoContent", typeof(object), typeof(MainWindow),

new PropertyMetadata(null, (s, e) =>

{

if (e.NewValue != null || (e.NewValue is string str && str != ""))

{

(s as MainWindow)?.SetValue(HasInfoPropertyKey, true);

}

else

{

(s as MainWindow)?.SetValue(HasInfoPropertyKey, false);

}

}));

/// <summary>

/// 获取是否有提示信息（只读）

/// </summary>

public bool HasInfo => (bool)GetValue(HasInfoProperty);

// Using a DependencyProperty as the backing store for HasInfo. This enables animation, styling, binding, etc...

private static readonly DependencyPropertyKey HasInfoPropertyKey =

DependencyProperty.RegisterReadOnly("HasInfo", typeof(bool), typeof(MainWindow),

new PropertyMetadata(false));

public static readonly DependencyProperty HasInfoProperty = HasInfoPropertyKey.DependencyProperty;

//private void SystemParameters\_StaticPropertyChanged(object sender, System.ComponentModel.PropertyChangedEventArgs e)

//{

// if (e.PropertyName == "WindowGlassBrush")

// {

// }

//}

private void panelList\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

var current = e.AddedItems[0] as ListBoxItem;

if (current?.Tag is Type type)

{

try

{

var controls = from ctl in AppControls

where ctl.GetType() == type

select ctl;

//如果已经存在指定类型的元素

if (controls.Count() > 0)

mainContent.Child = controls.First();

else

{

//通过反射创建类型

if (Activator.CreateInstance(type) is UserControl newControl)

{

AppControls.Add(newControl);

Logger.Log($"创建UserControl '{type.Name}' 成功");

mainContent.Child = newControl;

}

}

}

catch (Exception ex)

{

Logger.Log($"创建新控件 '{type.Name}' 失败\r\n{ex}");

}

}

menu.IsChecked = false;

}

private void closeSnackbar\_ActionClick(object sender, RoutedEventArgs e)

{

this.SetSnackBarInfo(null);

}

}

}