***Project List:***

2. NTest.DataHelper ------ 数据操作库
3. NTest.Framework ------ 基础函数库
4. NTest.DeviceHelper --- 设备函数库
5. Style

***Page***

Sign In ---------------------- 登录界面、锁屏界面、账号切换

Automatic Test ---------- 自动测试界面

Test Sequence ------------ 测试序列设计界面

Model Manage ---------- 产品模型管理界面

Product Manage -------- 产品管理界面

System Setting ----------- 系统参数设置

Device Setting ----------- 设备参数设置

Datalog -------------------- 数据记录查询、任务日志查询、错误日志查询

Tools ----------------------- 小工具集合界面

Help ------------------------ 帮助界面、使用说明、开发说明、功能扩展说明

*Note：Red --- 固定功能，Green --- 可变功能*

***File***:

System.ini

Device.ini

***Flowchart***

Model Create

Test Plan Create

Product Create

Instrument Create

Auto Test

***NameSpace***

1. 工程名称： N Test System
2. 项目名称: ’NTS.’+Name
3. 数据模型: ‘Name’+Model
4. ViewModel: ‘Name’+ViewModel
5. 视 图: ‘Name’+View
6. 扩展方法: ‘Name’+Extensions
7. 转 换 器: ’Name’+Converter
8. 控 制 器: ’Name’+Controller

System.ini content

1. Database Type
2. Database Link
3. Title
4. Icon

Device.ini content

1. Name
2. Type
3. Config
4. Commands
5. Termination
6. TimeOut

***Interface***

1. IDevice:设备接口，方法-Open，Close、Write、Query，属性-IsAsync，IsOpen，IsIdle
2. IDeviceService
3. IDevConfiguration
4. IData
5. IDataService

***Model***

1. DevCfgModel：设备配置模型，属性：Name，Type，Config，Termination，Commands，TimeOut，继承IDevConfiguration

Page Design

* Sign In
* Automatic Test
* Test Sequence
* Model Manage
* Product Manage
* System Setting
* Device Setting
* DataLog
* Tools
* Help

Model Design

* DevCfgModel

程序需求注入：

1. 配置信息采用程序配置文件

nTestSystem.DatabaseHelper:数据库相关操作模块

其中定义：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 类名称 | 类型 | 描述 |
| 1 | IDatabaseHelper | Interface | 数据库操作接口定义 |
| 2 | ISqlCommand | Interface | SQL语句接口定义 |
| 3 | SqlExpression | Class | SQL表达式处理函数定义 |
| 4 | ConditionalExpression | Class | SQL条件表达式函数定义 |
| 5 | SqlCommandExtensions | Class | ISqlCommand扩展方法定义 |
| 6 | BaseHelper | Class | 数据库操作基类定义 |
| 7 | DbFactory | Class | 数据库工厂 |
| 8 | MSSQLHelper | Class | 微软数据库操作函数定义 |
| 9 | SQLiteHelper | Class | TBD |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |

数据库数据操作：

1. 添加引用DatabaseHelper、DataHelper
2. 添加Using nTestSystem.DatabaseHelper、Using nTestSystem.DataHelper
3. 类继承IDataHelper

数据模型：

定义要求：Class需声明Database特性-表名称、属性需声明Column特性且默认值不能为null

[Database("nTestSystem") ]

public class Test

{

[Column(0)]

public string Guid { get; set; } = "";

[Column(1)]

public string PartNumber { get; set; } = "";

[Column(2)]

public string SerialNumber { get; set; } = "";

}

查询：无条件全字段

SQL语句：查询所有数据

var sql = new SqlExpression().Select<Test>();

结果：返回Result<Test>

var Data = this.GetData<Test>(sql);

查询：有条件自定义字段

SQL语句：查询Guid=‘123’的PartNumber

var condition = new ConditionalExpression<Test>().Eq(0, "123");

var sql = new SqlExpression().Select<Test>(new[] { 1 }, new[] { condition });

结果：返回Result<Test>

var Data = this.GetData<Test>(sql);

插入：单条数据

SQL语句：插入所有字段

var data = new Test() { Guid = Guid.NewGuid().Guid16(), PartNumber="Q123AS",

SerialNumber="SER1254" };

sql = new SqlExpression().Insert(null, data);

结果：返回Result<int>

var res = this.SaveData(sql);

SQL语句：插入指定字段

var data = new Test() { Guid = Guid.NewGuid().Guid16(), PartNumber="Q123AS",

SerialNumber="SER1254" };

sql = new SqlExpression().Insert(new[]{ 0, 1, 2 }, data);

结果：返回Result<int>

var res = this.SaveData(sql);

插入：多条数据

SQL语句：插入多条数据

var sqls = new List<ISqlCommand>();

for (var i = 0; i < 10; i++)

{

data = new Test() { Guid = Guid.NewGuid().Guid16() };

data.PartNumber = $"QSE234{i}";

data.SerialNumber = $"SERIL{i}";

sqls.Add(new SqlExpression().Insert(null, data));

}

结果：返回Result<Exception>

var res1 = this.SaveData(sqls);

更新：单条数据

SQL语句：更新Serial Number为‘SER1254’的Part Number为‘Test1’

condition = new ConditionalExpression<Test>().Eq(2, "SER1254");

sql = new SqlExpression().Update<Test>(new[] { 1 }, new[] { "Test1" }, new[]

{ condition });

结果：返回Result<int>

var res = this.SaveData(sql);

以Test类为值进行更新

SQL语句：更新Serial Number为‘SERIL0’的PartNumber和Guid

condition = new ConditionalExpression<Test>().Eq(2, "SERIL0");

data = new Test() { Guid = Guid.NewGuid().Guid16(), PartNumber = "Test2" };

sql = new SqlExpression().Update(new[] { 0, 1 }, data, new[] { condition });

结果：返回Result<int>

var res = this.SaveData(sql);

更新：更新多条数据

SQL语句：将SerialNumber为‘SERIL i’的PartNumber更新为’TTTTTT i’

sqls.Clear();

for (var i = 0; i < 10; i++)

{

condition = new ConditionalExpression<Test>().Eq(2, $"SERIL{i}");

sqls.Add(new SqlExpression().Update<Test>(new[] { 1 }, new[]

{ $"TTTTTT{i}" }, new[] { condition }));

}

结果：返回Result<Exception>

var res1 = this.SaveData(sqls);

删除：单条语句

SQL语句：删除Serial Number为‘SERIL0’的数据

condition = new ConditionalExpression<Test>().Eq(2, "SERIL0");

sql = new SqlExpression().Delete<Test>(new[] { condition });

结果：返回Result<int>

var res = this.SaveData(sql);

删除：多条语句

SQL语句：删除Serial Number为‘SERIL’系列的数据

sqls.Clear();

for (var i = 0; i < 10; i++)

{

condition = new ConditionalExpression<Test>().Eq(2, $"SERIL{i}");

sqls.Add(new SqlExpression().Delete<Test>( new[] { condition }));

}

结果：返回Result<Exception>

var res1 = this.SaveData(sqls);

最大自定义窗口丢失阴影效果解决方案：

<WindowChrome.WindowChrome>

<WindowChrome

CaptionHeight="0"

ResizeBorderThickness="5" />

</WindowChrome.WindowChrome>

Note:

1. BindingFlags用法

BindingFlags.Static/BindingFlags.Instance这两个中的一个必须要使用一个；BindingFlags.NonPublic/BindingFlags.Public这两个中的一个必须要使用一个；

也就是说，静态成员或者实例成员，是否Public访问级别这两个维度都需要指定，如果只是指定某一个维度就会出问题，也可以不使用BindingFlags来指定查找的范围，默认是查找Public的成员(包括Instance和static成员)，也就是说默认BindingFlags.Static | BindingFlags.Public|BindingFlags.Instance。

1. 中英文切换

app.xaml.cs代码

protected override void OnStartup(StartupEventArgs e)

{

base.OnStartup(e);

GetLanguage();

}

private void Application\_Exit(object sender, ExitEventArgs e)

{

SaveLanguage();

//关闭所有线程，即关闭此进程

System.Environment.Exit(0);

//MessageBoxManager.Unregister();

}

#region Method

/// <summary>

/// 开机启动默认的语言

/// </summary>

private void GetLanguage()

{

Language = string.Empty;

try

{

Language = CANTest.Properties.Settings.Default.Language.Trim();

}

catch (Exception)

{

}

Language = string.IsNullOrEmpty(Language) ? "en-US" : Language;

//update Language

UpdateLanguage();

}

/// <summary>

/// 保存语言设置

/// </summary>

private void SaveLanguage()

{

try

{

CANTest.Properties.Settings.Default.Language = Language;

CANTest.Properties.Settings.Default.Save();

}

catch (Exception)

{

}

}

/// <summary>

/// 更换语言包

/// </summary>

public static void UpdateLanguage()

{

List<ResourceDictionary> dictionaryList = new List<ResourceDictionary>();

foreach (ResourceDictionary dictionary in Application.Current.Resources.MergedDictionaries)

{

dictionaryList.Add(dictionary);

}

string requestedLanguage = string.Format(@"Language\StringResource.{0}.xaml", Language);

ResourceDictionary resourceDictionary = dictionaryList.FirstOrDefault(d => d.Source.OriginalString.Equals(requestedLanguage));

if (resourceDictionary == null)

{

requestedLanguage = @"Language\StringResource.en-US.xaml";

resourceDictionary = dictionaryList.FirstOrDefault(d => d.Source.OriginalString.Equals(requestedLanguage));

}

if (resourceDictionary != null)

{

Application.Current.Resources.MergedDictionaries.Remove(resourceDictionary);

Application.Current.Resources.MergedDictionaries.Add(resourceDictionary);

}

}

#endregion

xaml代码：

zh-cn：

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

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

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

<sys:String x:Key="biFile">文件</sys:String>

<sys:String x:Key="biHelp">帮助</sys:String>

<sys:String x:Key="biAbout">关于</sys:String>

<sys:String x:Key="biName">产品名称：</sys:String>

<sys:String x:Key="biCopyRight">版权所有:</sys:String>

<sys:String x:Key="biWeb">公司网址:</sys:String>

<sys:String x:Key="biVersion">版本号:</sys:String>

<sys:String x:Key="biSelectDevice">选择设备</sys:String>

<sys:String x:Key="biClose">关闭</sys:String>

<sys:String x:Key="biSetting">设置</sys:String>

<sys:String x:Key="biLanguage">语言</sys:String>

</ResourceDictionary>

en-us：

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

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

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

<sys:String x:Key="biFile">File</sys:String>

<sys:String x:Key="biHelp">Help</sys:String>

<sys:String x:Key="biAbout">About</sys:String>

<sys:String x:Key="biName">Product Name：</sys:String>

<sys:String x:Key="biCopyRight">CopyRight:</sys:String>

<sys:String x:Key="biWeb">Web:</sys:String>

<sys:String x:Key="biVersion">Version:</sys:String>

<sys:String x:Key="biSelectDevice">Choose Device</sys:String>

<sys:String x:Key="biClose">Close</sys:String>

<sys:String x:Key="biSetting">Setting</sys:String>

<sys:String x:Key="biLanguage">Language</sys:String>

</ResourceDictionary>

app.xaml定义，这里注意下，因为两个字典中有同样字符，如果没有动态更改，默认后添加的生效

<Application x:Class="CANTest.App"

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

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

xmlns:dxe="http://schemas.devexpress.com/winfx/2008/xaml/editors"

xmlns:dx="http://schemas.devexpress.com/winfx/2008/xaml/core"

StartupUri="MainWindow.xaml" Startup="OnAppStartup\_UpdateThemeName" Exit="Application\_Exit">

<Application.Resources>

<ResourceDictionary>

<ResourceDictionary.MergedDictionaries>

<ResourceDictionary Source="Language\StringResource.zh-CN.xaml" />

<ResourceDictionary Source="Language\StringResource.en-US.xaml" />

</ResourceDictionary.MergedDictionaries>

</ResourceDictionary>

</Application.Resources>

</Application>

如何调用资源字典的样式 前端我们一般DynamicResource +key，后台则是FindResource（"资源字典的KEY"）as string，但是在MVVM 模式下是得不到这个方法的 所以应该用到Application.Current.TryFindResource("资源字典的KEY) as string;这个方法来获取资源字典的内容