TMcraft Node Tutorial Multi-Language

Original Instructions

Document version: 1.0

Release date: 2023-06-28

Table of Contents

Revision History

Revision	Date	Description
1.0	2023-06-28	Original release

1. Introduction

Developers of TMcraft Node might want to enrich user experience by adding different cultures (languages). This document describes how to add multiple languages to TMcraft Node by using resource files. Before reading this document, readers should have the following requisites:

- Basic knowledge of programming with C# and WPF
- Understanding of the WPF resource concept preferred
- Having read *TMcraft Node Tutorial: Basic* and familiar with the TMcraft Node Program structure

Based on TMcraft.dll version 1.14.1100, this tutorial.

2. Add Multiple Languages by Resource Files

There are a few ways to add multiple languages to the TMcraft Node UI. Take a simple UI as an example. Users can use several text files in JSON or XML format to organize the UI Content as {key, value} so that the program can load the UI with different language content. Nevertheless, the most commonly used WPF method is by adding resources. Users may follow these steps:

1. When building the UI, assign Dynamic keys to the Content of all related Controls, i.e.{DynamicResource *keyName*}.

```
<Button x:Name="Btn_Close" … Content="{DynamicResource Close}" FontFamily="Microsoft YaHei
UI Light" FontSize="56" Foreground="White" FontWeight="Bold" … />
```

2. Create the resource files (StringResources.xaml) and add them to Resources directory. Here is an example of StringResource.en-US .

```
<ResourceDictionary
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:system="clr-namespace:System;assembly=mscorlib">
        <system:String x:Key="Close">Close</system:String>
</ResourceDictionary>
```

3. Add the resource when loading UI.
   Generally, a program gets the system culture with a function like

Thread.CurrentThread.CurrentCulture(); however, TMcraft Node UI should follow the locale setting of TMflow, which is usually different from the locale setting. It has much to recommend that users use the TMcraft function SystemProvider.GetCurrentLanguageCulture (out culture). After that, set up the resource directory by the locale setting. To fulfill the tasks mentioned above, users can define a function SetLanguageDictonary() calling during the project run. Refer to the example below.

```
private void SetLanguageDictionary(){
    ResourceDictionary dict = new ResourceDictionary();
    string culture = String.Empty; unit result = 0;

    result = TMNodeEditor.SystemProvider.GetCurrentLanguageCulture(out culture);
    switch (culture){
        case "en-US":
        dict.Source = new Uri("pack://application:,,,/{Name of the User Control Library
        Project};component/Resources/{String Resource file}", UriKind.Absolute);
        break;
        …//Other cases
    }
    this.Resources.MergedDictionaries.Add(dict);
```

IMPORTANT:
It is necessary to use the absolute path ("pack://application:,,,/*{Name of the User Control Library Project}*;component/Resources/*{String Resource file}*) instead of the relative path to access resources; otherwise, it will not work.

3. Example

In this example, it will make a simple TMcraft Node to change the language setting in TMflow. Please note that the source code in this example is available in the SDK package.

First, assign the dynamic keys to the UI Controls.

Control Name	Dynamic key
Label_Title	Title
Label_Slogan	Slogan
Btn_Close	Close

The code within MainPage.xaml goes like below.

```
<Label x:Name="Label_Title" … Content="{DynamicResource Title}" … />
<Label x:Name="Label_Slogan" … Content="{DynamicResource Slogan}" … />
<Button x:Name="Btn_Close" … Content="{DynamicResource Close}" … />
```

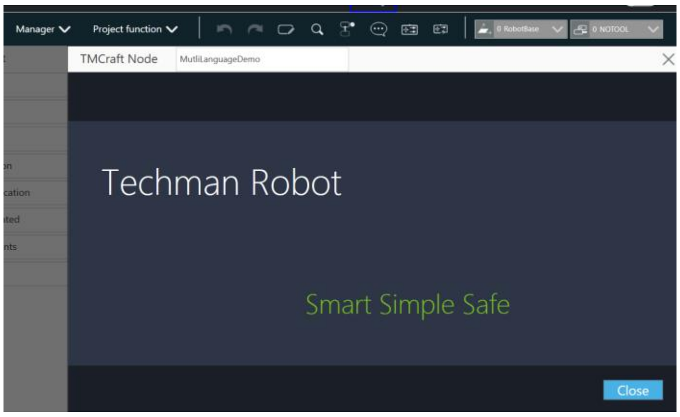
Create the two string resource files of en-US and zh-TW, respectively. English will apply to other cultures.

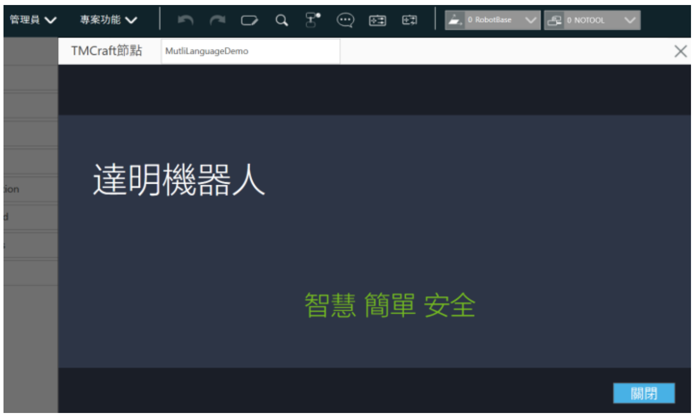
```
<ResourceDictionary
    xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
    xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”
    xmlns:system="clr-namespace:System;assembly=mscorlib">
        <system:String x:Key="Title”>Techman Robot</system:String>
        <system:String x:Key="Slogan”>Smart    Simple    Safe</system:String>
        <system:String x:Key="Close">Close</system:String>
</ResourceDictionary>

==================================================================
<ResourceDictionary
    xmlns=”http://schemas.microsoft.com/winfx/2006/xaml/presentation”
    xmlns:x=”http://schemas.microsoft.com/winfx/2006/xaml”
    xmlns:system="clr-namespace:System;assembly=mscorlib">
        <system:String x:Key="Title”>達明機器人</system:String>
        <system:String x:Key="Slogan”>智慧    簡單    安全</system:String>
        <system:String x:Key="Close">關閉</system:String>
</ResourceDictionary>
```

Next, define the function SetLanguageDictionary() and call it within the initialization function UserControl_Loaded().

```
private void UserControl_Loaded(object sender, RoutedEventArgs e){
    try{ SetLanguageDictionary();}
    catch (Exception ex) { MessageBox.Show("UserControl fails loading: " + ex.Message); }
}

private void SetLanguageDictionary(){
    ResourceDictionary dict = new ResourceDictionary();
    string culture = String.Empty; unit result = 0;

    //this is a simpler usage of TMcraft Provider function without checking the connection
    result = TMNodeEditor.SystemProvider.GetCurrentLanguageCulture(out culture);

    switch (culture){
        case "en-US":
        dict.Source = new Uri("pack://application:,,,/
        /MutliLanguageDemo;component/Resources/StringResource.en-US.xaml",
        UriKind.Absolute);
        break;

        case "zh-TW":
        dict.Source = new Uri("pack://application:,,,/
        /MutliLanguageDemo;component/Resources/StringResource.zh-TW.xaml",
        UriKind.Absolute);
        break;

        default:
        dict.Source = new Uri("pack://application:,,,/
        /MutliLanguageDemo;component/Resources/StringResource.en-US.xaml",
        UriKind.Absolute);
        break;
    }
    this.Resources.MergedDictionaries.Add(dict);
```

Compile the program to a User Control Library file and build it into TMcraft Node. Users will see the UI changes its locale by the language setting of TMflow.





TECHMAN
ROBOT

www.tm-robot.com