Loxodon Framework Fody

(English)

Developed by Clark

Requires Unity 2018.4 or higher.

这是一个静态织入代码的工具，已将Fody整合到Unity项目中，可以利用Fody丰富的插件来简化代码，提高开发效率。目前我已经将PropertyChanged.Fody和ToString.Fody插件也发布为Unity的package.

PropertyChanged.Fody是一个注入INotifyPropertyChanged相关代码的插件，通过为ViewModel类Model类添加注解，自动生成"INotifyPropertyChanged"接口和相关的代码。关于PropertyChanged.Fody注解的使用请查看官方文档。

ToString.Fody能够为类自动生成ToString函数。只要类添加了[ToString]注解，就会重载类的ToString函数。

安装

从OpenUPM安装

OpenUPM 中提供了很多的Unity插件，自动管理依赖，推荐从OpenUPM仓库安装本插件.

命令行方式安装，要求 nodejs's npm and openupm-cli, 如果没有安装nodejs命令行环境请先安装nodejs。

```
# Install openupm-cli,please ignore if it is already installed.
npm install -g openupm-cli

#Go to the root directory of your project
cd F:/workspace/New Unity Project

#Install loxodon-framework-fody-propertychanged
openupm add com.vovgou.loxodon-framework-fody-propertychanged

#Install loxodon-framework-fody-tostring
openupm add com.vovgou.loxodon-framework-fody-tostring
```

通过修改 Packages/manifest.json 文件安装插件(推荐)

在Unity项目的Packages目录中找到manifest.json 文件，增加第三方仓库"https://package.openupm.com"或者"https://registry.npmjs.org"到配置文件中，然后增加"com.vovgou.loxodon-framework-fody" 到dependencies节点下，Unity会自动下载插件，使用这种方式安装也相当方便，且省去了安装nodejs和openm-cli客户端的麻烦。

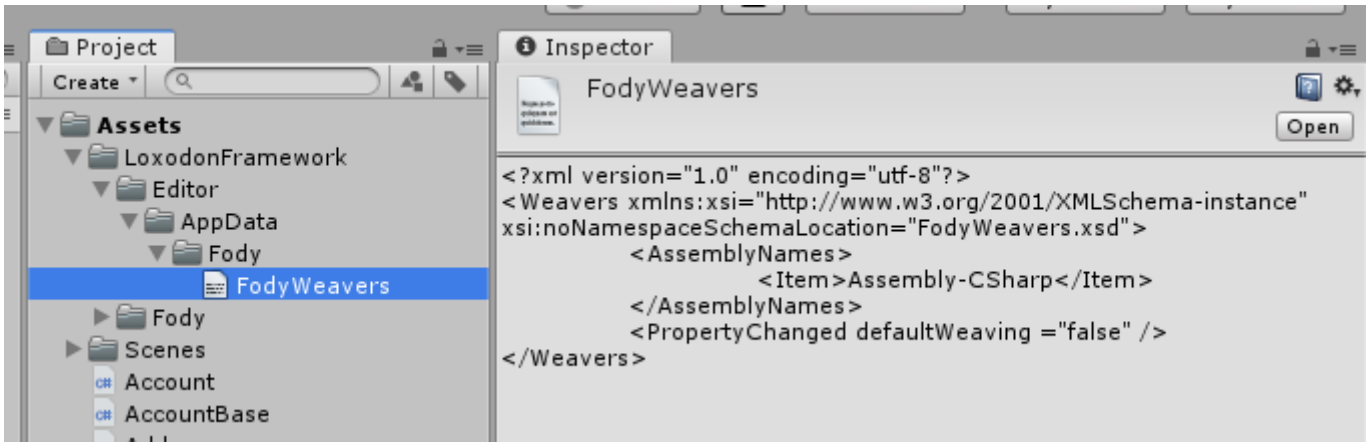
```
{
  "dependencies": {
    ...
    "com.unity.modules.xr": "1.0.0",
    "com.vovgou.loxodon-framework-fody": "2.4.9",
    "com.vovgou.loxodon-framework-fody-propertychanged": "2.4.9",
    "com.vovgou.loxodon-framework-fody-tostring": "2.4.9"
  },
  "scopedRegistries": [
    {
      "name": "package.openupm.com",
      "url": "https://package.openupm.com",
      "scopes": [
        "com.vovgou",
        "com.openupm"
      ]
    }
  ]
}
```

快速开始

PropertyChanged.Fody

插件导入到项目后，会在Assets\LoxodonFramework\Editor\AppData\Fody目录下自动生成FodyWeavers.xml文件。修改这个文件，添加需要织入代码的程序集名称即可。XML文件中的PropertyChanged节点是关于PropertyChanged.Fody插件的配置，具体可以查看PropertyChanged.Fody的文档。

PropertyChanged默认会织入所有继承了INotifyPropertyChanged或者添加了AddINotifyPropertyChangedInterface注解的类，如果某个类不想被织入代码，可以使用DoNotNotify注解排除。老的项目引入此插件后，会导致所有已经添加了属性通知的ViewModel类再次被织入RaisePropertyChanged函数，造成重复触发通知事件的情况，因此我重写了PropertyChanged.Fody插件的部分方法，为xml配置文件PropertyChanged节点增加了一个属性defaultWeaving。当defaultWeaving=false时，只会为添加了AddINotifyPropertyChangedInterface注解的类织入通知代码，避免老的ViewModel类被重复的织入RaisePropertyChanged函数。



FodyWeavers.xml

```
<Weavers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Fc
        <AssemblyNames>
                <Item>Assembly-CSharp</Item>
        </AssemblyNames>
        <PropertyChanged defaultWeaving ="true" />
</Weavers>
```

在项目中创建一个User类，添加注解"AddINotifyPropertyChangedInterface", 代码如下:

```
[AddINotifyPropertyChangedInterface]
public class User
{
    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string FullName => $"{FirstName} {LastName}";
}
```

在代码被Unity编译后，PropertyChanged.Fody会自动织入INotifyPropertyChanged接口相关的代码，所有的属性都会增加RaisePropertyChanged或者OnPropertyChanged函数触发属性改变通知事件。使用ILSpy反编译工具打开Assembly-CSharp.dll程序集，User类的代码如下：

```
public class User : INotifyPropertyChanged
{
        public string FirstName
        {
                [CompilerGenerated]
                get
                {
                        return FirstName;
                }
                [CompilerGenerated]
                set
                {
                        if (!string.Equals(FirstName, value, StringComparison.Ordinal))
                        {
                                FirstName = value;
                                <>OnPropertyChanged(<>PropertyChangedEventArgs.FullName);
                                <>OnPropertyChanged(<>PropertyChangedEventArgs.FirstName);
                        }
                }
        }

        public string LastName
        {
                [CompilerGenerated]
                get
                {
                        return LastName;
                }
                [CompilerGenerated]
                set
                {
                        if (!string.Equals(LastName, value, StringComparison.Ordinal))
                        {
                                LastName = value;
                                <>OnPropertyChanged(<>PropertyChangedEventArgs.FullName);
                                <>OnPropertyChanged(<>PropertyChangedEventArgs.LastName);
                        }
                }
        }

        public string FullName => FirstName + " " + LastName;

        [field: NonSerialized]
        public event PropertyChangedEventHandler PropertyChanged;

        [GeneratedCode("PropertyChanged.Fody", "3.4.1.0")]
        [DebuggerNonUserCode]
        protected void <>OnPropertyChanged(PropertyChangedEventArgs eventArgs)
        {
                this.PropertyChanged?.Invoke(this, eventArgs);
```

```
        }
}
```

ToString.Fody

增加注解 `<ToString/>` to FodyWeavers.xml

```
<Weavers xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="Fc
        <AssemblyNames>
                <Item>Assembly-CSharp</Item>
        </AssemblyNames>
        <ToString />
</Weavers>
```

User类代码如下，为User类添加[ToString]注解。

```
[ToString]
public class User
{
    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string FullName => $"{FirstName} {LastName}";
}
```

织入后的代码如下，自动生成了ToString函数。

```
public class User
{
    public string FirstName { get; set; }

    public string LastName { get; set; }

    public string FullName => $"{FirstName} {LastName}";

        [GeneratedCode("Fody.ToString", "1.11.1.0")]
        [DebuggerNonUserCode]
        public override string ToString()
        {
                return string.Format(CultureInfo.InvariantCulture, "{T: 'User', FirstName: '{0}'
                {
                        FirstName ?? "null",
                        LastName ?? "null",
                        FullName ?? "null"
                });
        }
}
```

Contact Us

Email: yangpc.china@gmail.com

Website: https://vovgou.github.io/loxodon-framework/

QQ Group: 622321589