.NET嵌入资源

资源指图片，音视频，文档等形式的文件。

可将资源嵌入程序集，程序集执行到访问资源的代码时，会去存储资源的程序集寻找资源。这种资源称为嵌入资源。

嵌入资源的优点和缺点

优点：资源嵌入到程序集，不会出现资源丢失导致访问不到资源的错误。

缺点：程序集的体积会因嵌入资源而迅速增大，导致加载程序集的时间变长。

嵌入资源的使用方法

方法一：利用.resx文件嵌入资源

xxx.resx是一种具有特定格式的XML文件，按照规则在XML文件中填入资源的路径,类型等信息，resgen.exe作用于xxx.resx文件，根据XML中的路径找到资源，然后将资源转换成xxx.resources二进制文件，最后csc.exe将xxx.resources和cs文件编译成程序集。资源在二进制文件中的组织形式像字典，key是资源名称，value是资源的字节数组形式。根据资源所在的.resources名称和资源名称，能在程序集的资源段的.resources中，检索到资源。

添加嵌入资源

有两种添加方式，第一种是比较原始的方式，第二种是利用Visual Studio提供的便捷方式。

方式一：手工编辑xxx.resx

假设我们将图片card.jpg和一些字符串嵌入进程序集作为嵌入资源。

1. 新建.resx文件　　D:\asm\xxx.resx
2. 在D:\asm\new_dir下放入图片card.jpg
3. 按照.NET预定的格式编辑xxx.resx，主要是填入路径信息

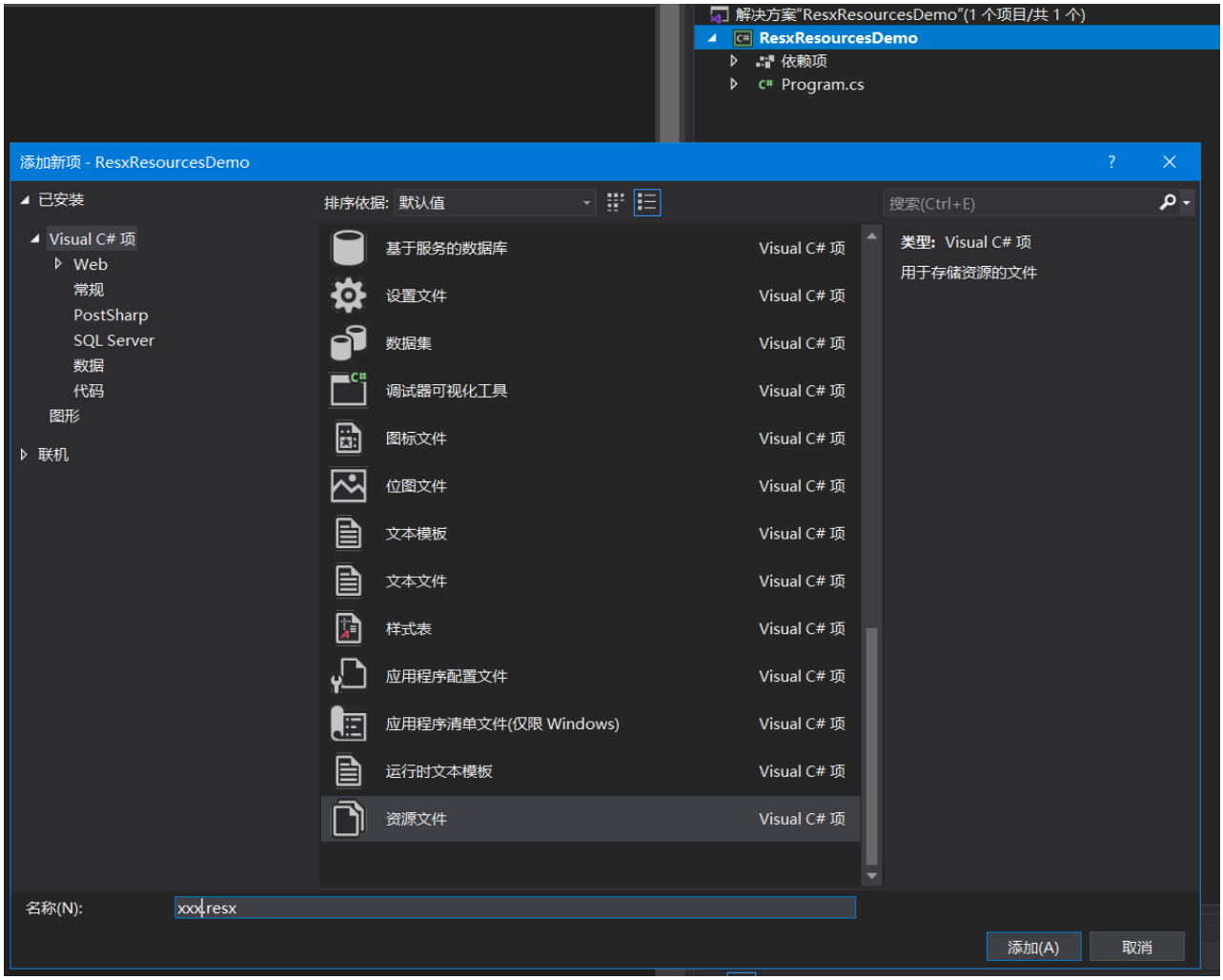
```
  <data name="btn1" xml:space="preserve">
    <value>打开</value>
  </data>
  <data name="btn2" xml:space="preserve">
    <value>关闭</value>
  </data>
  <assembly alias="System.Windows.Forms" name="System.Windows.Forms, Version=4.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" />
  <data name="idCardIcon" type="System.Resources.ResXFileRef, System.Windows.Forms">
    <value>new_dir\card.jpg;System.Byte[], mscorlib, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089</value>
  </data>
```

4. 运行resgen.exe xxx.resx，会生成xxx.resources。这一步，在xxx.resx中按照路径找到资源【假设当前的工作目录是new_dir的父目录，会正常的去new_dir下找到card.jpg；如果xml中配置的value是card.jpg，而不是new_dir\card.jpg，则会找不到card.jpg，除非当前的工作目录切换成new_dir】，然后提取所有的资源转换成.resources二进制文件。注意，二进制文件中并不包含资源的路径和文件名信息，只是简单的【资源key名 --- 资源二进制】，使用资源时，通过key拿到一个二进制块。
5. csc.exe /resource:xxx.resources Program.cs，将资源和cs源代码文件编译成程序集。

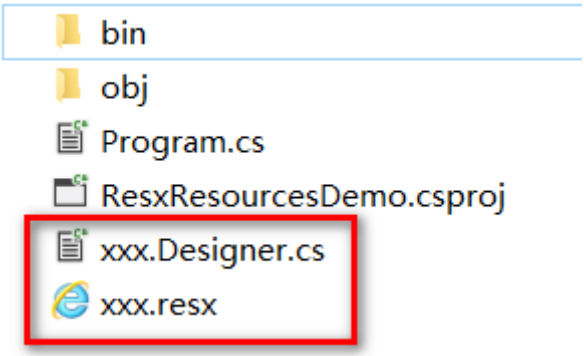
方式二：可视化编辑xxx.resx

方式一有两点不便，xxx.resx的格式是有限定的，容易填错；还要敲命令编译.resources，难度太高。所以，visual studio提供了可视化编辑resx文件的方法。

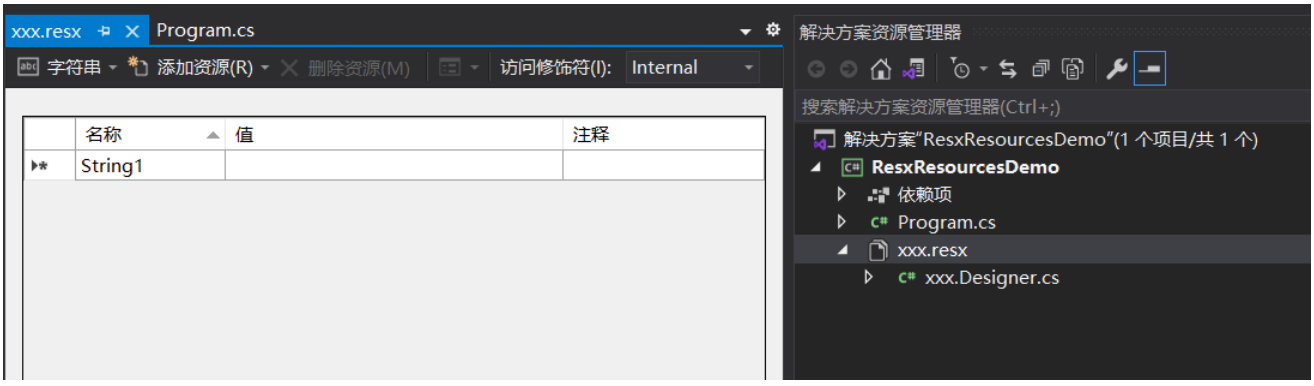
1. 添加资源文件



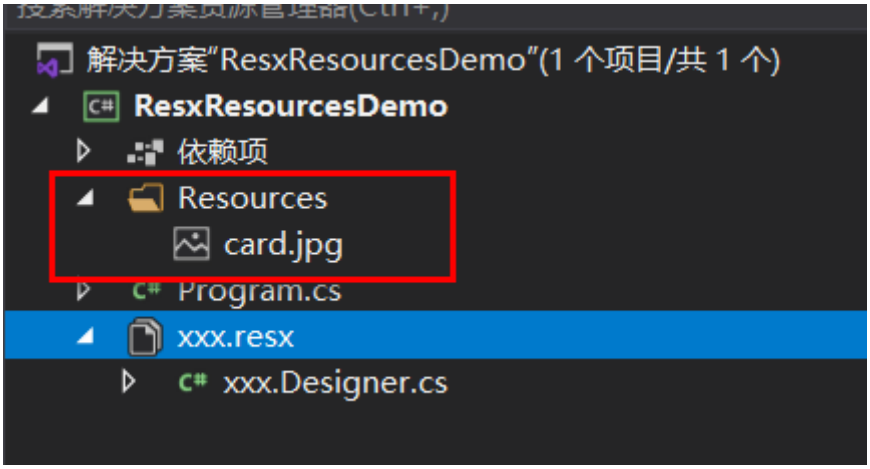
2. 项目会多出两个文件



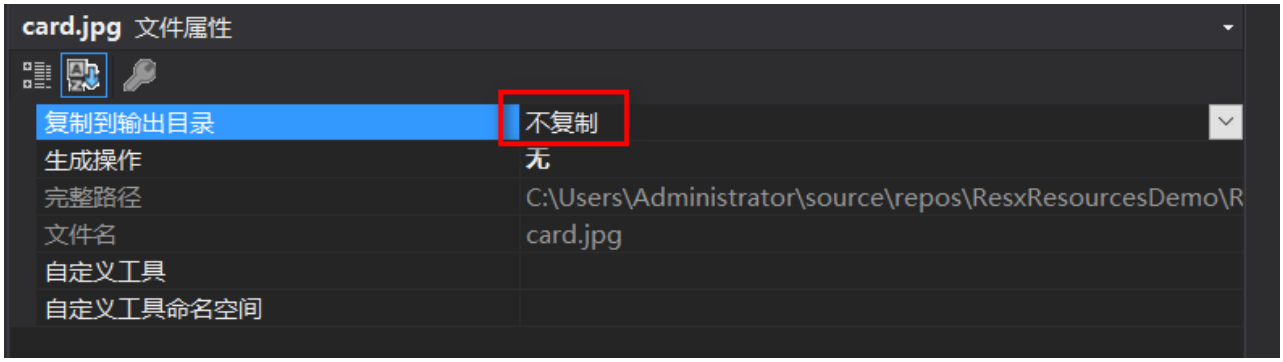
3. 双击xxx.resx，打开可视化画面，增删资源



4. 添加资源后，项目会自动将资源拷贝到Resources文件夹



5. 设置资源属性为【不复制】。因为资源被嵌入进程序集，程序也从程序集内部寻找资源，所以没必要把资源拷贝到根目录下。



6. 编译

编译时的工作目录是项目的一级目录，所以编译时，能找到资源和源代码。

编译完成后，资源共存在4个位置

（1）资源原来所在的位置

（2）项目目录结构下的Resources文件夹

（3）ResxResourcesDemo.xxx.resources（本应是xxx.resources，但是.NET自己加了手脚，就是.resources默认追加上项目的默认命名空间）

（4）程序集内部

- 编译完成后，我们可以删除位置（1）（2）（3）的资源，程序集也能正常工作。但是我们重新编译程序集，会重新检索.resx，搜索Resources文件夹下的资源，重新生成.resources并嵌入。所以我们增删资源【一定】要通过可视化.resx，如果只是简单的在Resources文件夹下增删，重新编译，解析.resx的资源路径，会因为找不到资源导致出错！

强调

- 一个.resx生成一个.resources文件
- .resx只包含资源的路径信息，.resources包含真正的资源并嵌入进程序集
- 一个项目可以包含多个.rex，所以一个程序集可以嵌入多个.resources
- 多个.resx的资源会被统一拷贝到Resource文件夹下
- 因为所有的.resx的资源都会被拷贝到Resource文件夹下，如果资源很多，想分类管理，那么可以通过新建多个.resx达到分类的目的
- .resources二进制资源被嵌入到程序集的资源段
- .resources文件名是 项目默认命名空间 + .resx文件名 + .resources
- 访问资源需要的信息包括：资源所在程序集，资源所在的.resources名，资源的key名

访问嵌入资源

```
/* 读取指定程序集中的指定.resources(.resx)的资源 */

// 1.命名空间.(resx文件名 | .resource文件名) 2.程序集
ResourceManager resource = new ResourceManager("MyNamespace.xxx",
Assembly.GetExecutingAssembly());
// 读取字符串或文本
string str = resource.GetString("str");
// 读取图片
Bitmap bitmap = resource.GetObject("pic") as Bitmap;
// 读取其他文件为字节数组
UnmanagedMemoryStream stream = resource.GetStream("file");
byte[] bytes = new byte[1024];
stream.Read(bytes, 0, 1024);
stream.Dispose();
```

```
// 遍历一个程序集内的所有资源
string[] resNames = Assembly.GetExecutingAssembly().GetManifestResourceNames();
foreach (string resName in resNames)
{
    ResourceManager resource1 = new ResourceManager(resName, Assembly.GetExecutingAssembly());
    using (ResourceSet set = resource1.GetResourceSet(CultureInfo.CurrentCulture, true, true))
    {
        foreach (DictionaryEntry res in set)
        {
            Console.WriteLine(res.Key);
            Console.WriteLine(res.Value);
        }
    }
}
```

我们注意到.NET还为每个.resx自动生成一个xxx.Designer.cs，该文件只是封装了ResourceManager访问资源的API，使我们更加容易的访问资源，不必自己再写一堆代码。

```
internal static global::System.Resources.ResourceManager ResourceManager {
    get {
        if (object.ReferenceEquals(resourceMan, null)) {
            global::System.Resources.ResourceManager temp = new global::System.Resources.ResourceManager("ResourcesDemo.Resource1", typeof(Resource1).Assembly);
            resourceMan = temp;
        }
        return resourceMan;
    }
}
```

```
internal static System.Drawing.Bitmap gzl {
    get {
        object obj = ResourceManager.GetObject("gzl", resourceCulture);
        return ((System.Drawing.Bitmap)(obj));
    }
}
```

```
internal static byte[] test_resource {
    get {
        object obj = ResourceManager.GetObject("test_resource", resourceCulture);
        return ((byte[])(obj));
    }
}
```

类名是.resx文件名，资源作为静态成员存在，权限为internal，我们可以修改权限，可以让程序集的资源能被其他程序集访问。

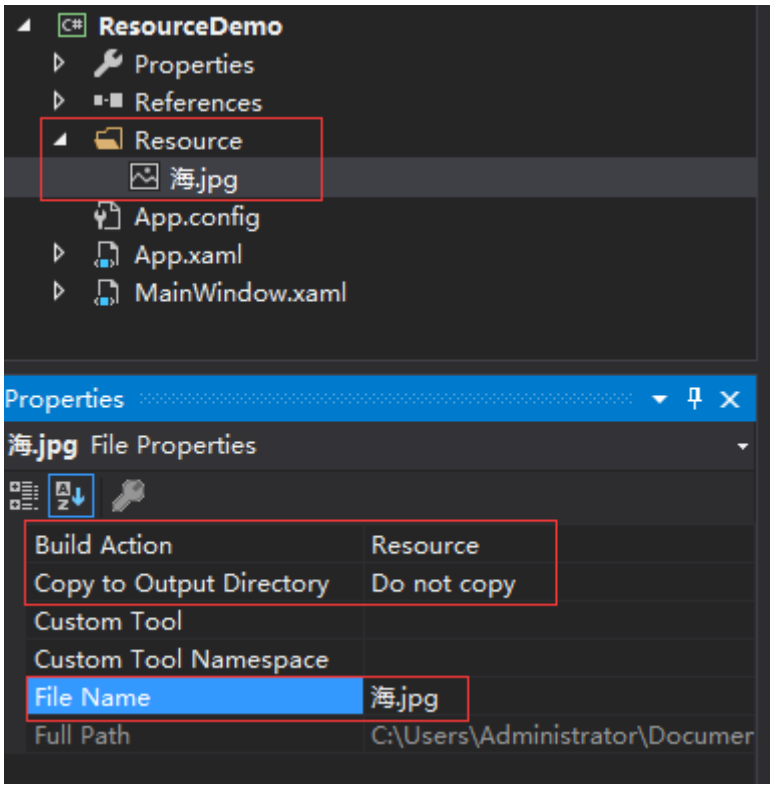
修改权限的方法：

方法二：利用uri机制嵌入资源

添加引用和命名空间

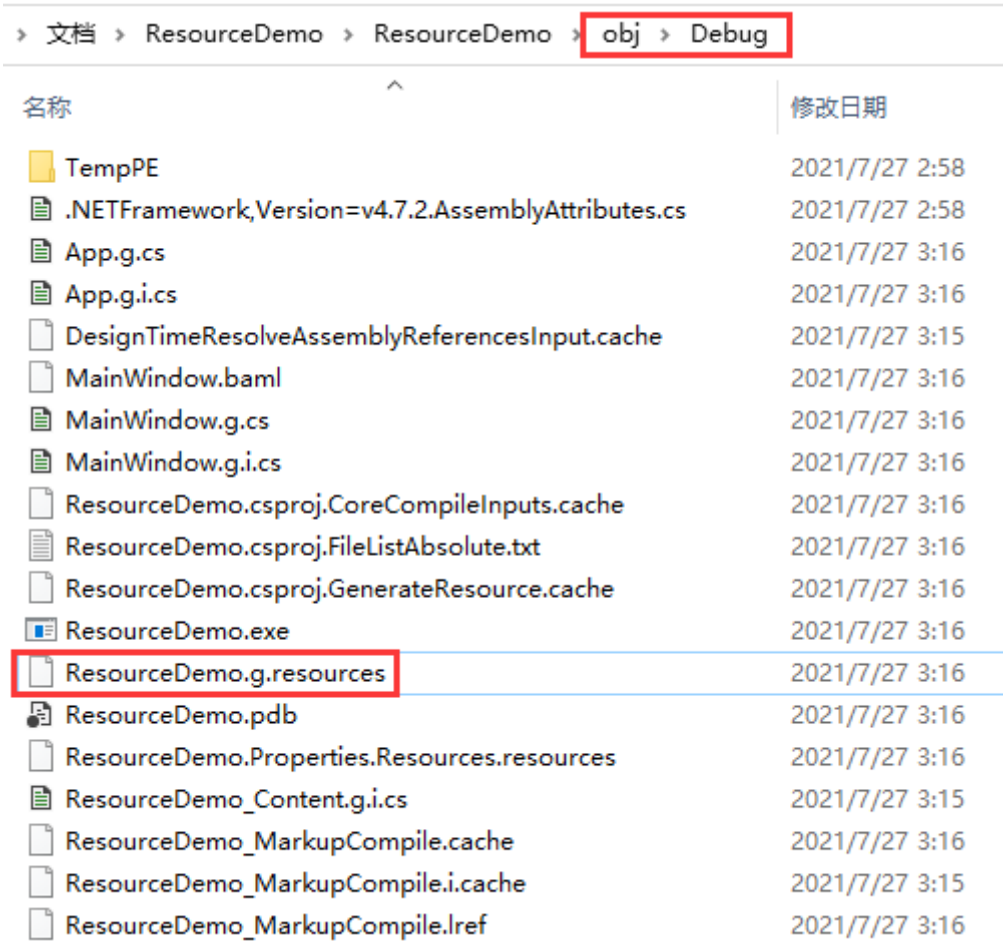
dll: PresentationFramework　　namespace: using System.Windows.Resources

添加文件夹和资源，设置资源属性



Build Action [Resource]表示资源会被嵌入到程序集, 代码访问资源时,直接去程序集内部寻找,所以我们不需要将资源拷贝到应用程序的根目录,故Copy to Output Directory为[Do not copy],因为复制是没有必要的.

编译后,资源存在4个位置(1) 资源原来的位置 (2)项目目录结构 (3) .resource文件 (4) 程序集内部



二进制资源名称 : 程序集名称.g.resources

编译后,我们可以删除位置（1） （2） （3）的资源,因为资源已经被编译到程序集中,代码直接从程序集寻找资源.

当我们想添加或删除资源时,直接在位置（2）添加或删除资源,重新编译即可. 如果我们把所有资源删除,再次编译时,便不会生成.resource文件, 程序集也不会嵌入任何资源.

访问资源

- 通过uri访问资源

  XAML

```
<!-- <Image Source="Resource/海.jpg"/> -->
<Image Source="pack://application:,,,/Resource/海.jpg"/>
```

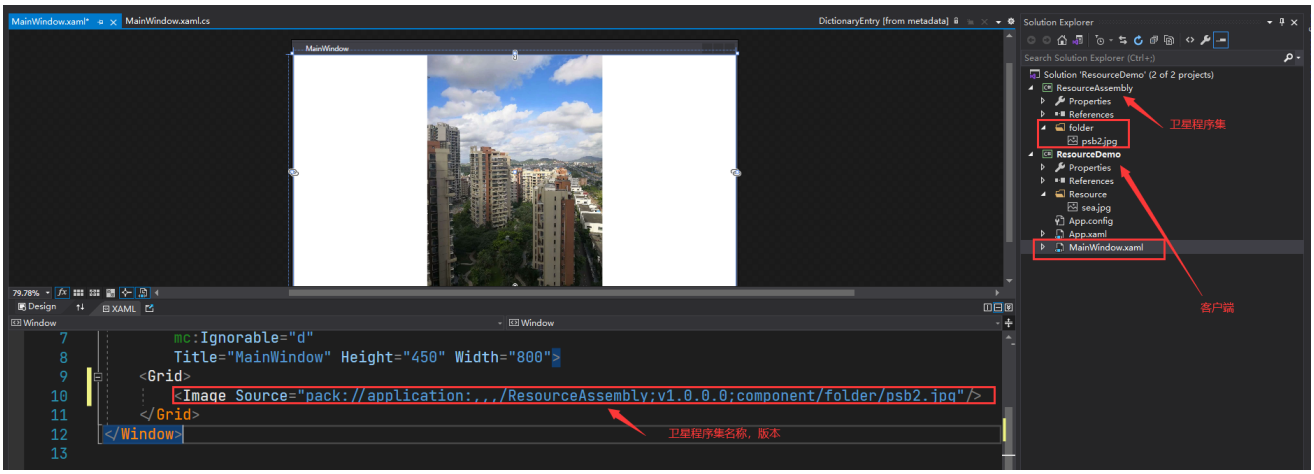
  CSharp

```
// Uri uri = new Uri("/Resource/海.jpg", UriKind.Relative);
Uri uri = new Uri("pack://application:,,,/Resource/海.jpg", UriKind.Absolute);
StreamResourceInfo info = Application.GetResourceStream(uri);
BitmapImage bitmapImage = new BitmapImage();
bitmapImage.BeginInit();
bitmapImage.StreamSource = info.Stream;
bitmapImage.EndInit();
img.Source = bitmapImage;
```

扩展:遍历所有嵌入资源

```
Assembly assembly = Assembly.GetAssembly(this.GetType());
ResourceManager rm = new ResourceManager(assembly.GetName().Name + ".g", assembly);
using (ResourceSet set = rm.GetResourceSet(CultureInfo.CurrentCulture, true, true))
{
    foreach (DictionaryEntry res in set)
    {
        MessageBox.Show(res.Key.ToString());
    }
}
```

卫星程序集的URI




```
<Image Source="pack://application:,,,/ResourceAssembly;v1.0.0.0;component/folder/psb2.jpg"/>
```

.NET内容资源

内容资源：当资源体积很大时，尤其是动辄几十M的音视频文件，不宜嵌入进程序集。所以内容资源，资源本身并没有嵌入程序集，编译时，资源同它们的目录结构一起会被自动的拷贝到应用程序的根目录。应用程序执行时，一般都设置当前工作目录是应用程序根目录，所以代码中的相对路径，也是相对应用程序的根目录，故编译时的自动拷贝内容资源到应用程序根目录，也是Visual Studio的一种“默契”。内容资源的添加方法和嵌入资源相似，只是Build Action设置成Content，Copy to Output Directory设置成true。对于URI，嵌入资源支持绝对路径，相对路径，卫星程序集，本地程序集。内容资源只支持本地程序集的绝对和相对路径。

```
Uri uri = new Uri("pack://application:,,,/Resource/海.jpg", UriKind.Absolute);
StreamResourceInfo info = Application.GetContentStream(uri);
```