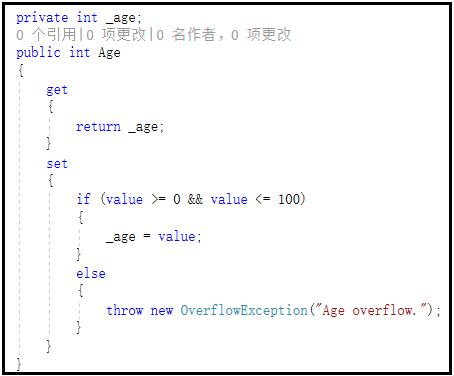
# CLR属性

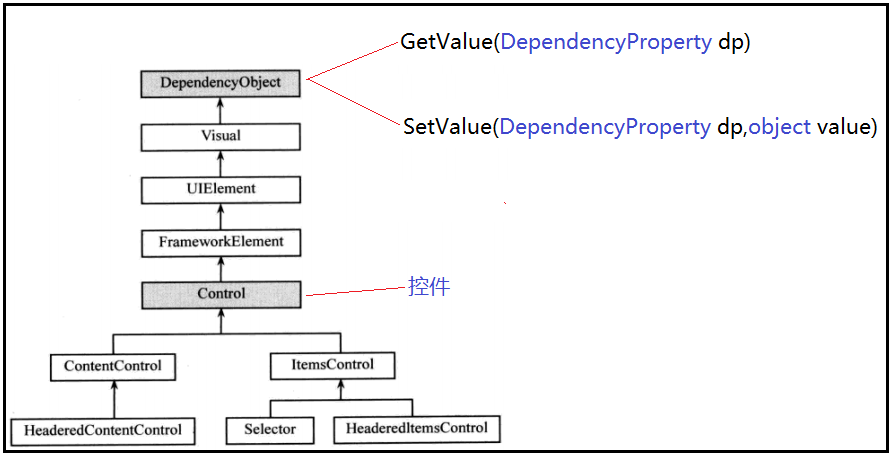
****

**在传统的类库中使用SetAge()\GetAge()等函数对私有字段\_age的读写进行限制、但是书写麻烦、微软在.NET Framework中将Get\Set这对方法合并成了属性(Property)。**

**.NET属性又称为CLR属性(CLR,Common Language Runtime).可以说CLR属性是Private字段的安全访问包装。**

# 依赖项属性

每个CLR属性背后都有一个静态字段存在，每个控件都有很多属性，一个GUI程序有大量的控件势必会占用大量内存，为了解决这些问题引入了依赖项属性。



属性属于类，那么依赖属性也一样，只是依赖属性的依赖对象是DependencyObject，依赖属性值的修改必须通过依赖对象的GetValue\SetValue接口来实现。

为了节省内存引入了依赖对象、那么依赖对象肯定不能和CLR属性一样随意的创建实例，依赖属性使用

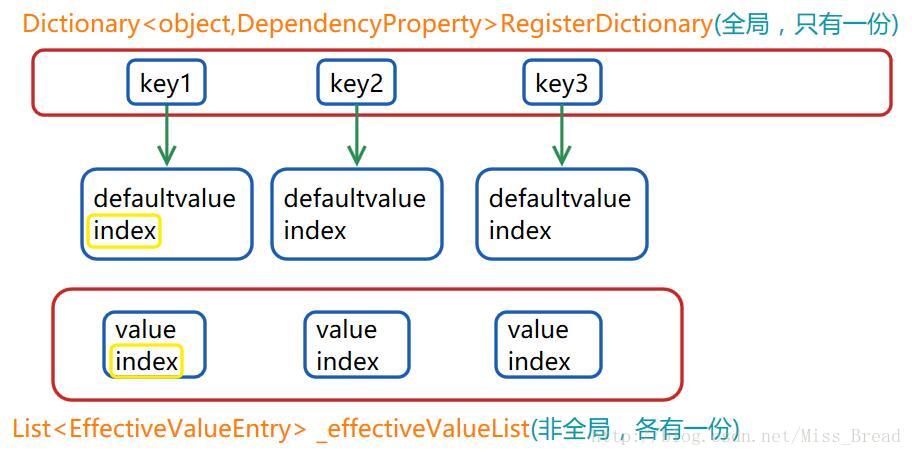
Public static readonly \*\*property;修饰，所以依赖属性是属于类的静态成员，静态变量的好处就是全局可见、方便信息的传递、、。

**依赖对象的实例：**



依赖属性使用DependencyProperty.Register("CLR包装属性名", typeof(CLR包装属性类型), typeof(依赖对象类型));来创建依赖属性的实例。

**依赖属性Regest过程：**



* 程序有一个全局字典RegisterDictionary，里面存放所有的依赖属性
* Key是字典关键字，使用CLR属性名和依赖对象类型的哈希值异或产生
* Value，依赖属性的实例，其中包含默认值、属性改变事件的回调函数、、
* Index，依赖属性实现了GetHashCode()函数，可以获取依赖属性实例的HashCode，也就是GlobalIndex

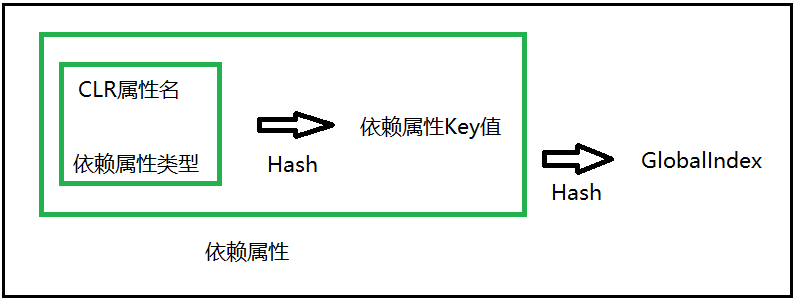
总而言之注册依赖属性的过程就是

1. 使用CLR属性名和依赖对象类型生成一个HashCode做为Key值，
2. 使用key值将依赖属性实例存放到一个全局的字典容器中，生成一个依赖属性的HashCode🡪GlobalIndex

**依赖属性的实际值存储：**

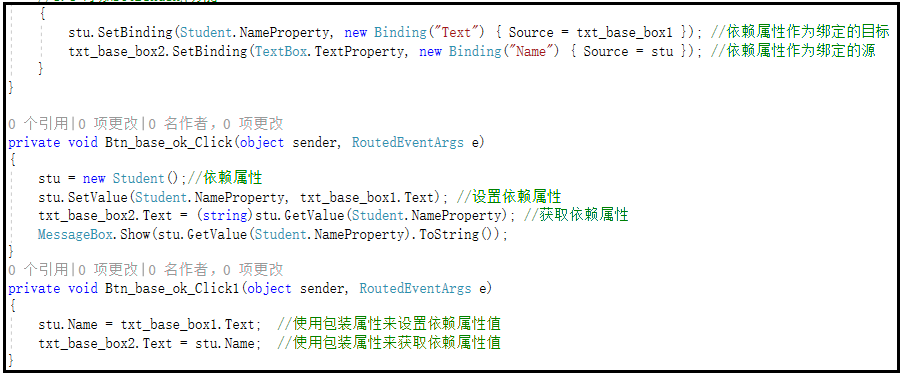
* 每个DependencyObject实例都有一个EffectiveValueEntry类型数组，存储依赖属性的GlobalIndex和属性值
* 这个数组的插入、删除、排序都有专门的算法
* 依赖属性的具体值存在于每个依赖对象的EffectiveValueEntry数组中

**ReadOnly关键字：**



依赖属性产生globalIndex是Hashcode，所以使用Readonly修饰来保证Hashcode的稳定性

**依赖属性的使用：**



**设计理念：**

**以public static类型的变量作为标记，以这个标记为索引进行对象的存储、访问、修改、删除、、。**