[C# where用法](http://www.cnblogs.com/soundcode/p/5798769.html)

where 子句用于指定类型约束，这些约束可以作为泛型声明中定义的类型参数的变量。  
 1.接口约束。  
 例如，可以声明一个泛型类 MyGenericClass，这样，类型参数 T 就可以实现 IComparable<T> 接口：

public class MyGenericClass<T> where T:IComparable { }

 2.基类约束：指出某个类型必须将指定的类作为基类（或者就是该类本身），才能用作该泛型类型的类型参数。  
 这样的约束一经使用，就必须出现在该类型参数的所有其他约束之前。

class MyClassy<T, U>  
 where T : class  
 where U : struct  
{  
}

 3.where 子句还可以包括构造函数约束。  
 可以使用 new 运算符创建类型参数的实例；但类型参数为此必须受构造函数约束 new() 的约束。new() 约束可以让编译器知道：提供的任何类型参数都必须具有可访问的无参数（或默认）构造函数。例如：

public class MyGenericClass <T> where T: IComparable, new()  
{  
 // The following line is not possible without new() constraint:  
 T item = new T();  
}

new() 约束出现在 where 子句的最后。  
  
 4.对于多个类型参数，每个类型参数都使用一个 where 子句，  
 例如：

interface MyI { }  
class Dictionary<TKey,TVal>  
where TKey: IComparable, IEnumerable  
where TVal: MyI  
{  
 public void Add(TKey key, TVal val)  
 {  
 }  
}

5.还可以将约束附加到泛型方法的类型参数，例如：

public bool MyMethod<T>(T t) where T : IMyInterface { }

请注意，对于委托和方法两者来说，描述类型参数约束的语法是一样的：

delegate T MyDelegate<T>() where T : new()