```
    *** Func<T,TResult> ***

    using System;
    namespace FuncDemo
    {
        internal class Program
        {
            private static void Main()
            {
                //类似委托功能
                Func<InputArgs, Result> func = TsetFunction;
                Console.WriteLine("第一种方式：");
                Console.WriteLine(func(new InputArgs("zhangqs008", "123456")));

                Console.WriteLine("第二种方式：");
                CallMethod(func, new InputArgs("zhangqs008", "1234567")); //或者
                CallMethod(TsetFunction, new InputArgs("zhangqs008", "1234567"));
            }

            public static Result TsetFunction(InputArgs input)
            {
                Result result = new Result();
                result.Flag = String.Compare("zhangqs008", input.UserName, StringComparison.OrdinalIgnoreCase) == 0 &
                    String.Compare("123456", input.Password, StringComparison.OrdinalIgnoreCase) == 0;
                result.Msg = "当前调用者：" + input.UserName;
                return result;
            }

            public static void CallMethod<T>(Func<T, Result> func, T item)
            {
                Result result = func(item);
                Console.WriteLine(result.ToString());
            }

            /// <summary>
            /// 方法参数
            /// </summary>
            public class InputArgs
            {
                public InputArgs(string userName, string password)
                {
                    UserName = userName;
                    Password = password;
                }

                public string UserName { get; set; }
                public string Password { get; set; }
            }

            /// <summary>
            /// 方法结果
            /// </summary>
            public class Result
            {
                public string Msg { get; set; }
                public bool Flag { get; set; }
                public override string ToString()
                {
                    return string.Format("Flag:{0},Msg:{1}", Flag, Msg);
                }
            }
        }
    }

-------------------------------------------------------------------------------------------

public static int StrToInt(String s)
{
    return int.Parset(s);
}
public static void Main(String[] argv)
{
    String[] s = { "123", "345", "567" };
            IEnumerable<int> t=s.Select<String, int>(StrToInt);
            foreach (int i in t)
            {
                Console.WriteLine(i);
            }
}
```

而是用Func<T>的话:

```
public static void Main(String[] argv)
{
    String[] s = { "123", "345", "567" };
            Func<string, int> f = (x) => int.Parse(x);       //简化了很多，所以说它只是提供便利而已
            IEnumerable<int> t=s.Select<String, int>(f);
            foreach (int i in t)
            {
                Console.WriteLine(i);
            }
}
```

```
-------------------------------------------------------------------------------------------
```

C#提供的Func泛型委托的多个重载版本:

例如:

```
    Func<in T1,out T>
    Func<in T1,in T2,out T>
    Func<in T1,in T2,in T3,out T>
```

```
-------------------------------------------------------------------------------------------
```