Resharper 上手指南

Resharper 进阶一：简要介绍

Resharper 进阶二：快速定位

Resharper 进阶三：快速完成

Resharper 进阶四：万能的 Alt+Enter

Resharper 进阶五：高效的代码结构调整

Resharper 进阶六：重构才是王道（上）

Resharper 进阶七：重构才是王道（下）

Resharper 进阶八：增强的浏览功能

这一系列不是对每个功能的中文帮助，而是我对开发中可能涉及的功能需求，利用 Resharper 来完成。有些是 Visual Studio 有的，但是做的不好，有些是没有而 Resharper 发明的。总的目的都只有一个，就是加快你编写代码的速度。在熟悉了它以后，你使用鼠标或者是无效击键的频率会大大地降低。当然这也需要对 visual studio 本身的使用有一定的基础，毕竟它 Resharper 是建立在这么强大的一个 IDE 基础上的。

Resharper 进阶一：简要介绍

面对这样一个问题：为什么.net 能够比 java 更加快速的完成开发？恐怕最大的优势在于 Microsoft 提供了一个无比强大的 IDE。它的强大在于适用于各个层次的开发人员。当你还是一只小小鸟的时候，可以学会 drag&drop，当你开始写代码的时候，你会发现设计器生成的代码是多么愚蠢和臃肿。当你自己能够写出干净的代码时，可能再也不想切换到设计器中去了（切换/F7）。我相信我们中的大部分人，即使是最资深的 Windows 开发人员， 用 VS 写代码比用记事本的效率要高很多 ,因为我们毕竟需要 IDE 的帮助以提高我们的生产力。但是在那个没有 Resharper 的年代里，你的生产力顶多停留在两个轮子的时代，另两个轮

子，只有在你熟悉了 Resharper 以后才能转动起来。

Resharper 能带给你的效率提升如此之高，以至于在我订阅的所有.net 专家的 blog 里面，所有的人都在关注它。要知道他们都不是一般的 coder，大多数是 CTO 或者技术总监。当我三个月前开始使用 Resharper 时，想要搜到一些中文的介绍，或者是经验性的文章却十分困难。大多数人似乎不关心它的存在。博客园里充斥着因为安装 Resharper 而拖慢系统，导致 VS 不稳定，机器响应速度下降，VS 启动速度下降等等的论调。幸运的是， 这些在我这里都没有遇到，也许是因为2G 的内存的缘故，反正 devenv.exe 所占的200~300m 内存并没有使我感觉机器在变慢，而且我的 VS 几乎是一直运行中，没有一会关闭一会打开的时候，所以我只需要忍受 Resharper 在加载时的半分钟初始化一次就够了。相比于它能带给你的效率提升，这些问题都是微不足道的（一家之言）。

如果你在 VS 环境下写代码时几乎不用快捷键，那我觉得你没有必要往下看了。因为 Resharper 是通过快捷键来提升你的效率的。但是相信我，如果你曾经在星际中一遍一遍被人虐的话，你应该能体会到快捷键带来多么大的变化。

首先在安装完 Resharper 之后，你主要通过三种方式用到或看到它：

一是在菜单栏上，如果是3.0版本的话，它会默认的屏蔽掉“重构”菜单栏，也就是说你按 Alt+R 的时候，弹出的是 Resharper 的下拉菜单而不是“重构”，为什么？因为你不再需要“重构”菜单了。如果是以前版本，比如2.0，似乎需要您手工的完成这一步，方法如下：工具——自定义——重排命令，然后把“重构”菜单删除，毫不犹豫地！

你可以简单地这样体验一下：Alt+R，N，回车，就可以在当前项目中添加一个类，还记得以前你是怎么做的吗？在解决方案资源管理器中选中一个项目，然后右键，在长长的弹出菜单中选择：添加——类，如果你的电脑够慢的话，在3秒钟之后才会弹出一个对话框来问你文件名。

另一种方式是：看到竖直滚动条了吗？在它的右边多出一个边框来，上方是一个“绿色/黄色/红色”的方框，而边框上会出现一道一道红的橙的横杠。什么 意思：当红杠出现时，表明你的代码在那个位置出现了错误，如果是橙杠，表明那是一个警告，多半是没有检测 null 值或者声明后未使用之类。

如果当前文档的所有错误和警告都得到了适当的处理，则不会再出现横杠，而最顶上的方框也会变成绿色；

如果没有错误但有警告，则是黄色，这时编译可以通过；

如果那个方框是红色，则表示有编译通不过的错误，这时你应该通过点击红杠，去修改您的代码。如果你即使编译，也会报错并无法生成。这样做使你在写代码时就能及时得发现你的错误，而不需要等到编译时。这样做也使你的效率得到了提高。因为编译至少要耗去您半分钟的时间，并且强烈的读写您的硬盘，特别在你的硬盘转速慢时，比如笔记本上，这一过程是那么的令人心痛。

第三种接触到 Resharper 的自然是快捷键了。右键菜单当然也有很多的功能，但是那太弱智了，太慢了。我们需要的是专业，我们用的工具定位不是打毛衣的大妈，而是开发的专业人员，如果你连快捷键都无法掌握，那真的得对你的开发技巧打一个大大的问号。试想一下这个场景，当你脑子里冒出这个念头， 啊，这个类，应该提出接口来，要这个方法，这个，和这个，还有这两个事件。啊，不行，我现在就得提，不然就忘了。我得新建一个接口，点哪个来着。啊，在哪 里新建……建好了，对了我要提取哪几个方法来着。（我晕死）许多时候你的念头都是一闪而过的，需要你的操作也相应的跟上。这个操作在 Resharper 就是，把光标移动到类名上，然后点 Ctrl+Shift+R，除了你想要的提取接口，其他的重构功能也一目了然。而且远比 VS 自带的要快。

至于右键，如果你刚开始实在记不住快捷键，只能使用它了，或者 Alt+R，实际上它也要比

右键菜单快一点点。

真正激动人心的是在随着你逐渐熟悉 Resharper 以后，能够带来的诸多方便之处，我将会在接下来的一系列随笔中讲述。

Resharper 进阶二：快速定位

摘要

快速代码定位的核心就是三个功能：

转到定义：Ctrl+B;

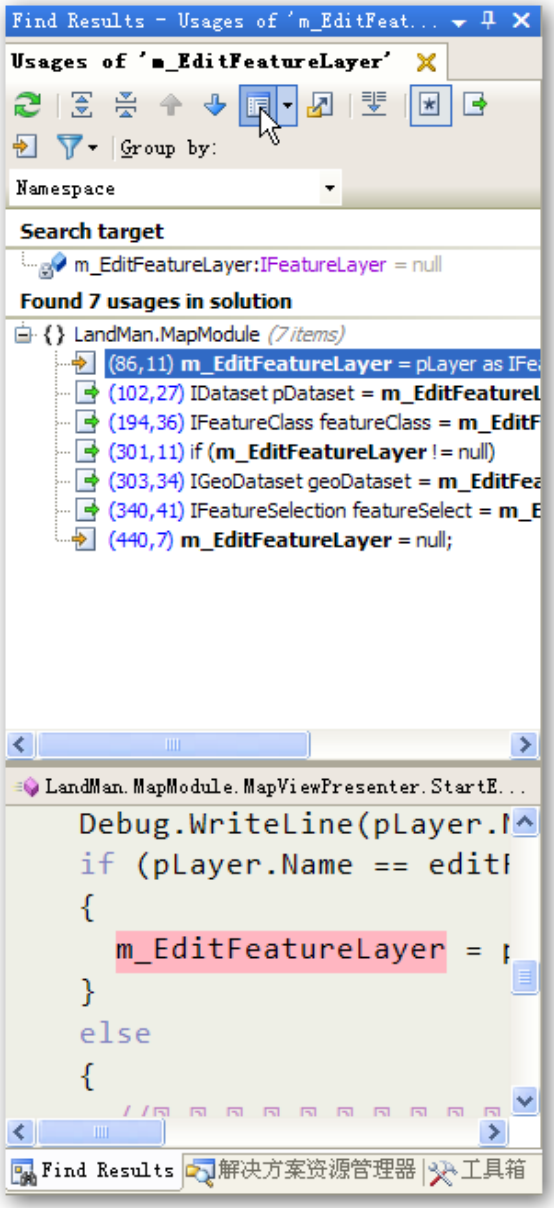
查找所有使用：Alt+F7;

从这里浏览：Ctrl+Shift+G.

转到定义

当你的光标在一个变量上时，按下 Ctrl+B 就会跳转到这个变量声明的地方。这可能是某一个字段，某一个方法的参数或者一个局部变量。当你的光标在一个类型上时，按下 Ctrl+B 可能发生两种情况，一是当这个类型的定义在你的项目中有源代码时，会跳转到这个源代码文件。如果没有源代码，则会打开一个对象浏览器。有一些使用者更期望跳转到元数据文件，但是我更喜欢对象浏览器，这样类型的公有属性、方法、事件都被列出来了，可以一目了然。就我的经验，列表永远比下拉框，选框更加易于查找。

但是这个功能有一个副作用，多年使用 Windows 的经验告诉我们，按住 Ctrl 的时候点鼠标左键，默认的操作是选中一整个单词，但是在安装了 Resharper 以后，这个操作实际上等于 Ctrl+B，也就是说可能你想选择这个变量的名字，然后复制到别处去，可是却跳转到了这个变量的声明处，不由得让你一阵恼火。目前我还没有找到选项屏蔽掉这个操作，所以，只有用鼠标双击来选择单词了（提示：如果跳转到了别处，想返回，按 Ctrl+-）



Alt+F7，有感觉吗？不错，就是 TotalCommand 里搜索的快捷键，没有用过 TC，还在用资源管理器？那可不应该。简而言之， Alt+F7将你光标所在位置的变量的所有使用以列表的方式显示出来，显示结果的窗体可以像其他窗体那样停靠。

它的优点包括：

可以从所有使用中挑选只显示 read usage 或者 write usage，有时我们只是想知道某个变量在哪里被改变了。找到的位置前的图标也告诉你这点。

可以在下方预览，即使我们列出所有使用，也不想跳转到每个使用它的地方，这时预览

可以帮你大忙。

当你在代码编辑器中改动了某些使用时，比如删除了某行，那么在查找结果的窗体中，会用删除线表示出来。

默认的是寻找解决方案中所有的使用，并且按照命名空间来组织，非常便于选择。

我现在已经记不起来在没有 Alt+F7之前我是怎么查找的。反正现在我几乎不怎么样 Ctrl+F 了，除非我忘记了某个变量的名字。如果是这样，多半这个名字需要 refactor，那也是 Resharper 的另一大块功能所在。也许有人对这个功能嗤之以鼻，但是用过 CAB 的人都知道，订阅和发布某个事件的签名，完全是字符串，如果你不用搜索来找到它的话，你都不知道这个控件的鼠标点下去，到底有多少个处理程序在背后开始工作了。用了 Alt+F7来搜索这个字符串，等于在查找背后所有的调用者。

不过提示你，当光标停留在一个类型上时，要慎用 Alt+F7，假设是一个 string，你应该能想象到得找到多少个使用。

实在不知道用什么中文来翻译这个 navigate from here...，只好直译了。你可能在这些时候需要它：

当你要找这个类的所有继承者，或者接口的所有实现者时，按住 Ctrl+Shift+G 会弹出一个菜单，其中有 Inheritor，用方向键来选择并回车，如果只有一个实现，那么直接跳转到这个实现，如果有多于一个，或者因为使用了 partial 分布到好几个文件中，会再弹出一个菜单来供你选择。与此类似，如果你选择 base，则会跳转到基类或接口中去。

同时这个功能也囊括了前面的转到定义和查找使用 如果你偷懒的话 ,只记住这个就可以了，虽然 Ctrl+Shift+G 的快捷键有一点生僻，但是用着用着你就发现顺手了。

这个功能不可不谓是用来熟悉一套框架的利器，可以让你迅速找到某个接口的所有实现，和 Alt+F7合用，可以让你在一个庞大的解决方案中如鱼得水。

下一次，我们该来讲一讲威力无比的 Alt+回车了

Resharper 进阶三：快速完成

常常写代码的人，应该都对 Visual Studio 的智能感知有一定的好感，通过它，在输入比较长的对象名称时不觉得吃力了。恐怕这是.net 平台比其他平台的代码更加易读的一个因素，因为每个变量名称都比较有意义，而不是晦涩的 varIdx，pElemDisp 之类了。尽管这样看起来很酷。

在没有 Resharper 之前，Visual Studio 的自动完成功能就很强大了。只是它定义的快捷键实在是让人不顺手。Alt+右方向键，我怎么都无法不低头按它俩。于是我把快捷键改成了 Ctrl+;(分号)这样按起来方便多了。

Resharper 则在 Visual Studio 的基础上又增添了一些功能。比如说，在原生的自动完成中，关键字，比如 private,override 这些，是没有提供自动完成的（有吗， 我已经不记得了）。

还有这下面一些，则是 Visual Studio 原生确实没有的：

当你写一个新的字段时，比如 string _field; Resharper 会自动地在前面为你加上 private，也许有的人觉得多余，但我觉得是应该的。

当你输入 foreach 的时候，模板会自动的出现，方便你输入集合还有子项的类型与名称，回车之后就进入到块中。这类的模板你可以自行定义，并在团队中共享。这样在处理某些特定的场景时，代码的执行基本一致。

在你需要输入{的时候，Resharper 能够自动的为你加上}，并且光标位于其中，如果你敲回车键，两个括号和你准备接下来要写的块内的代码都缩进对齐了。这个功能远远不只说起来那么简单，当你发现自己可以因此忘记每次括回去的时候，是多么的惬意。

同理，在你输入[或者(的时候，Resharper 也如此为你添加成对的括号。

这样也有不方便的时候，比如你在调用某个方法的时候，其实你只是想输入(，然后开始写

参数，再写)，然后写分号，可是它偏偏帮你写了，特别是当参数还是一个字符串的时候，你输入"，它又自动的帮你写了"，并且把光标置于其间，好多时候搞得你很恼火。因为你要敲 end 键挪到行尾，再写;而 end 键也很难在 你不低头的情况下按准。实际上，这时你的选择还不如老老实实写上")，也比敲 end 来的快。所以为方法自动的添加括号的功能，只有在不带参数的情况下使用 才最惬意。

总的说，自动完成函数的输入工作，还是一个很实用的功能，现在你需要输入 ToString() 的话，只需要最多敲四个了 ToS 和;就可以了，其他的部分 Resharper 都自动为您完成。

最后再卖弄一下，将自动完成的快捷键换成 Ctrl+;的话真的非常棒，即使你没有安装 Resharper ,也可能考虑把原生的快捷键更改掉。在我所有用的电脑上 ,我都偷偷的把 Ctrl+; 添加进去了。因为这两个键实在是太好摁了。

Resharper 进阶四：万能的 Alt+Enter

万能的 Alt+Enter 能够帮你完成很多编写代码过程中的 dirty work，总结起来大概是这么些：

帮你实现某个接口或抽象基类的方法；

提供你处理当前警告的一些建议；

为你提供处理当前错误的一些建议（不一定是真的错误）；

为你简化当前的臃肿代码；

帮你实现某个接口或抽象基类的方法

这个功能 Visual Studio 也已经帮你提供了 ,就是每次你在类名的后面加上:ISomeInterface 的时候，它会提示你按 Tab 键就生成接口中所有方法的存根。如果你这时没按，后来，你还可以把鼠标悬停在这个接口名，会出现一个小小的，小的不能再小的，费死劲才点得中的

智能感知符上，然后生成方法。在安装了 Resharper 以后，这个功能被视为与其它警告一样的处理办法。如果你:ISomeInterface 以后，又没有实现它的方法，这一行代码会打上波浪号，表示有警告或者错误发生。按下 Alt+Enter，则会为你生成这些方法。

不要觉得我小题大作，在你修改了 ISomeInterface 这个接口以后，比如添加了一个 Initial 方法，这时最快的办法是：在 ISomeInterface 上点 Ctrl+Shift+G 浏览到这个接口的实现类，然后 Alt+Enter，这个类中立即就添加了 Initial 这个方法，并可开始编写代码。

提供你处理当前警告的一些建议

有的时候你可能会忽略的一些细节，Resharper 不会忽略，比如你用了隐式类型转化：

```
Button btn = sender as Button;          //隐式类型转化
```

而没有接下来检测它是否为空，就直接使用。这时 Resharper 会提示你一个警告。如果你按下 Alt+Enter，它会把你之后的代码包括在一个 if 语句中：

```
if (btn != null)
{
//你接下来的代码
}
```

不过令我不满意的是，我时常要的是这样的效果：

```
if (btn == null)
return;
```

不知道这个行为能不能在配置中修改。

为你提供处理当前错误的一些建议

被 Resharper 标记为红色的即为错误了，有时错误也可能是你有意为之。比如你随意地在 View.cs 中写下了：

_presenter.GetAllLayers();

而实际上你的 Presenter.cs 中还没有这个方法，那么按下 Alt+Enter 就会立即在 Presenter.cs 中添加这个方法，并跳转到该方法，如果 Presenter 不止分布在一个文件中，会弹出对话框让你选择。相比于 Visual Studio 经常臃肿的对话框，Resharper 的弹出对话框往往十分直接，你只需立即用方向键选你需要的并回车。

为你简化当前的臃肿代码

你的代码如果被 Resharper 视为多余的，那么会以灰色标识出来，例如：

```
this.Text = "标题栏文本"     //this 为灰色
btnOK.Click += new EventHandler(btnOK_Click)   //new EventHandler 为灰色
```

多余的代码并无害，只是显得不那么简洁，如果你头一次在安装了 Reshaper 之后打开你之前编写过的代码会发现大量的灰色代码。如果你查看设计器生成的代码，会发现 Reshaper 的竖直条几乎变成了橙色的一条了。基本上，你都可以用 Reshaper 为你自己的代码瘦瘦身，设计器产生的，比如窗体，或者 Dataset 还是不要动的好。

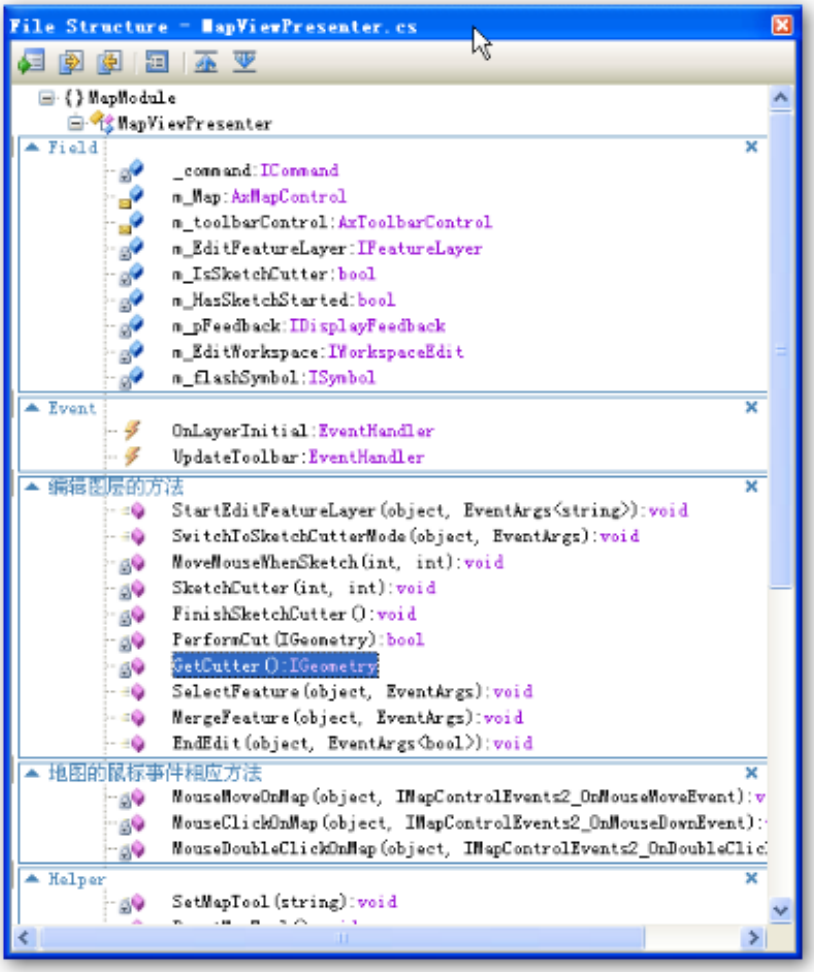
Resharper 进阶五：高效的代码结构调整

通过我之前介绍过的 Alt+F7和 Ctrl+B，你可以很快的在一个代码文件中知道函数的调用情况。但是有的时候，代码文件非常复杂，里头有几十个函数都算是小儿科，甚至一个构造函数就有 N 多个重载。这时你想清楚的了解文件中各个函数，属性，字段，事件等等，就不是那么轻而易举的事情了。

在 Visual Studio 中，你可以通过这些方式组织或审视你的代码：

用 region，这是最常使用的，我最喜欢的分法是 :Field, Property, Event, EventHander, 然后根据实际情况，Constructors(如果构造函数重载多的话),Helper(如果有许多公用的流程的话），再就是跟某个特定任务想关联的一系 列方法组成一类。

当你用 region 把你的代码组织好了以后，确实显得很简洁。但是问题来了，当一个同事 check 了你的代码，然后准备修改，这时，他想要理清你的思路， 于是在你的代码里开始奋战，啊，这个调用了那个，噢，这个地方可以接受0个或者1个或者三个参数……过了一个小时，他总算可以开始写他要改的那个部分了。 region 的缺陷就是让写的人很明白，而读的人还是很费劲。



用对象浏览器可以把你的方法结构展现出来，但致命的是它是按照字母排序的，对于我们中国人来说，实在是难以找到像 FilterSelectableLayerByName 这样一个其实自说明性很强的方法来。而实际上你又把它装进了一个“图层控制”的 region 里面，而你的读者却无法参透你的用意来。（如果你常常下载 codeproject 上的代码的话，会知道这种事情常有。）

说了这么多，其实就是想把对象浏览器和 region 的长处结合起来，既可以清晰的分类，又

能一目了然的找到需要的方法。Resharper 这时帮上你的大忙了。用 Ctrl+F12，就弹出一个像右边这样的窗口来。

这里面，按照你的 region 来显示，这样读你的代码的人也受益了。每个方法的参数，返回值都如 UML 一样列出来。

如果需要浏览到某个方法，直接双击它的名字；

如果要把某几个方法装进一个新的 region，则可以选中方法，点工具栏上的像框的那个图标；点叉则会删除这个 region 并把相应的方法移到外面来。

如果要调整某个方法的位置，比如把它移到别的 region 里面去，只需要在这里拖动这个方法即可。

更可喜的是，你想要的从这里浏览、找到所有使用和重构的功能也在这里提供了，在某个方法上右键你就能开始操作。

这绝对是 Resharper 里面最酷的功能，你快打开 Visual Studio 试一试吧。

Resharper 进阶六：重构才是王道（上）

重构是一种精神，证明你在致力于提供高效的、精炼的、健壮的代码，而不是凌乱的、晦涩的、漏洞百出的代码。

在 Visual Studio 2005中，微软第一次提供了重构工具。但是不够，远远不够。我们需要的重构是非常广义的，我们想要对代码进行快速的调整，快到我在想什么我的工具就能做什么。这才是追求重构的境界。所以在这个意义上，几乎 Resharper 为你提供了巨大的生产力。

Visual Studio 2005提供的重构包括了如下：

1. 封装字段
2. 提取方法

3 提取接口

4 提升局部变量

5 移除参数

6 重命名

7 重新排列参数

这些方法在 Resharper 中全部都支持（但 Resharper 的重构远不止这些），它们对应的变成了：

8 封装字段 —— Introduce Field

9 提取方法 —— Extract Method

10 提取接口 —— Extract Interface （另增加了 Extract Superclass 提取为基类）

11 提升局部变量 —— Introduce Variable

12 移除参数 —— 移到 Change Signature（改变方法签名）中

13 重命名 —— Rename （Resharper 会根据对象的类型名称，提供你几个可选的最合适的名称）

14 重新排列参数 —— 移到 Change Signature（改变方法签名）中

我知道很多人都声称自己 E 文不好，但是，这确实都是很简单的单词，难不倒任何人的。这些重构的功能是人所共知的，下面就告诉大家一些 Resharper 特有的，首先，重构的快捷键是 Ctrl+Shift+R：

1、对于类，除了提取接口、基类，你还可以移动它到其他的命名空间和移动到别的文件里，这是一个实用的功能，也许你不信，但是我这真的有个人，把所有的 business entity 都写在一个 DataObject.cs 里面。你难以想象，我打开它时嘴张了多大。

2、对于字段，提供了：

Safe Delete，会检测所有使用到的地方，并询问如何删除；

Pull Member Up 和 Push Member Down，可以把这个字段在基类和继承类中移动；

Use base type where possible，尽可能的使用基类，由于 ArcGIS 平台是基于 com 组件的，很多时候我们需要的是 IGeometry, IPointCollection 这样的接口所公开的属性或者方法，于是你没有必要保存一个 polygon 对象，而可以使用基类型；

Encapsulate Field，封装字段，但是这个功能远没有另一个提供同样功能的操作有用。我可以在后文中来讲。

3、对于方法，提供了：

与字段类似的功能，此外；

Change Signature，更改函数签名，包括更改名称，返回值类型，参数的各种信息，添加和删除参数，相当实用。如果你是在重写方法上操作，会提示你是否到基类中更改。

Make Static，如果 Resharper 检测到这个方法并没有与非静态成员相关联的话，往往会自动地提示你（以黄色横杠的形式出现）可以改为 static，如果你自作主张的对一些方法进行修改也无不妥，但后果自负。

Extract class from parameter，如果你的参数有七个八个，那是否考虑用一个类来封装这些参数呢，于是这个功能应运而生。

Method to Property，顾名思义，如果还在使用 GetField()或者 SetField(..)的话，你一定是从非.net 星来的。

4、在方法体内部：

Extract Method，不用介绍了吧。

Introduce Variable/Parameter/Field，取决于你光标所在的对象，可以提供转化的功能。

Inline Variable：就是把：

```
IPoint point = new PointClass();
point.PutCoords(_point.X, _point.Y);
```

变成这样子：

```
new PointClass().PutCoords(_point.X, _point.Y);      //这是个糟糕的例子
```

5、重命名：

为什么重命名值得挑出来讲，因为 Resharper 提供了命名建议这一金子般的功能。于是，想改名为易读性强的名字，不是那么费脑子的事情了。 Resharper 会根据这个变量的类型，为你提供几个备选名字，名字列表是列在光标位置上的（对方法重命名会弹出对话框），你只需要用方向键选择并敲回车即可，这种名字多是将类型的名字首字母改为小写得来的，甚至刨根到基类的类型名，你还可以在此基础上加以改进。如果你还在用 i,j 这种晦涩的名称，请迅速的把他们改为 outIndex, pointCount 之类可读的名称。

Resharper 其实提供了更先进的功能，在你命名一个变量时，就有快捷键为你提供备选名字，但是 Ctrl+Space 是我们宝贵的输入法切换键，于是，我对变量名的敲定，往往是先起了一个较烂的，然后重命名的。

还有一些更广义上的，帮助你对代码进行调整的功能，我另写一篇吧，不然太长了。

Resharper 进阶七：重构才是王道（下）

插入代码

Resharper 的 Alt+Insert 快捷键提供给你插入代码的功能。由于这两个键非常难按（这是我的感受），真正在使用的时候，我用的是 Alt- R-C-G，意指打开 Resharper 菜单——Code——Generate，都只需要你的左手，这样你可以右手一边比划，一边还在写代码，多酷啊。

生成的代码中最常用的是构造函数和属性，当你没有私有字段的时候，只会生成一个空

的默认构造函数，而且没有生成属性的功能。在你有私有字段的情况下，生成之前会让你选择哪些私有字段需要作为构造函数的参数，并生成初始化的代码，这样编写重载极其方便。生成属性也类似。

再次常用的就是重写基类或者接口的方法了。选择 Implement Interface Member 或者 Override Inheritate Member，Resharper 会查找当前类的基类或接口，然后按继承层次列出来，根据你的选择重写或实现这些方法。

不是太常用的是生成 Equals 和 GetHashCode 方法，在我的应用场景中很少重写它们。但是根据《.NET 设计规范》，不管是值类型还是引用类型的 Equals 都建议重写，并且应该重写 GetHaseCode 方法，因为它们相互依赖。如果你有这个需求，那么生成这三个函数一定能够帮你的大忙。

包围代码

Visual Studio 也提供了外侧代码这个功能，你可以按 Ctrl+K,Ctrl+S 来激活这个功能，虽然我并没有任何鄙视 Visual Studio 的意思，但是 Resharper 的快捷键确实更加合理（我在按下 Ctrl 的时候真的很难按下 S），条目也更加清晰。Resharper 中这个功能的快捷键是 Ctrl+Alt+J，然后你就可以选择将当前行的代码包围到 try-catch 块或者 using 中了。这是很高效的方法，我们倾向于在开 发的早期尽量不捕获异常，而在中后期才加入异常处理机制。于是你某一个时期有大量的工作是把他们扩到 try-catch 块中。而你要使用支持 dispose 对象时，最好的方法是使用 using 块。（卖蛋糕的，当我知道我的代码不是最优的时候，我总是寝食难安），这里自然也有把代码扩到 region 块中的功能，也是常用功能之一。

调整方法的位置

前面我曾说过，如果要调整方法的位置，可以在代码结构窗口中拖放操作。如果你觉得只是

把一个方法移动到前面去，却不得不打开代码结构窗口太过重量级，那么有轻量级的方法：当光标位于方法的名称上时，用 Ctrl+Shift+上下键就可以移动方法的位置，包括方法的 xml 注释，但如果你用的不是三个/的 xml 注释而是两个/的，那么就对不起了。

其他琐碎的功能

你肯定常常会复制粘贴当前行的代码，例如在使用 StringBuilder.Append 的时候，Ctrl+D 可以简化你 Ctrl+C,Ctrl+V 的工作。

曾经有一个组合键可以注释掉当前行，还有另一个是取消注释，但是我已经淡忘了，因为 Ctrl+/才应该是真正属于它的快捷键，再次按下就可以取消注释。

关于 Resharper 的重构功能就是这些，我可能天真地把很多额外功能都算在重构里了，但是它确实能够帮助你快速的对代码进行调整和优化。所以，请不要深究我对重构的概念认识是不是混乱。

Resharper 进阶八：增强的浏览功能

浏览参数的方式

输入方法的时候，我们已经习惯了由 IDE 提供给我们的参数提示，极大了方便了我们选择重载方法。在没有 Resharper 的环境下，Visual Studio 已经做到了。那么为什么 Resharper 还要增强这个功能并大获好评的。试问，Visual Studio 那窄窄的一行参数提示有没有让你觉得憋屈。我们有19寸的大屏幕，1600的分辨率，却不得不盯着那窄条条，小心翼翼的按着上下键寻找我们需要的重载。至少，开发 Resharper 的家伙是受不了这种憋屈的，于是大开大阖版的参数列表出现了，长长的参数重载被以列表的形式展现出来，当你在使用 GDT+方法，看到巨大的参数重载时，你会从心底里发出感叹：卖蛋糕的。

同时，Resharper 展示参数的快捷键变成了 Ctrl+P，如果你觉得屏蔽了打印的快捷键简直是在开玩笑的话，那么问问你自己有多少次打印过自己的代码。

浏览打开过的文档

我窃以为你已经知道了在 Visual Studio 中切换文档的方式，它们包括：

Ctrl+Alt+上下方向键，可以在打开的文档中切换；

Ctrl+Tab，不仅可以在文档之前切换，并可以切换到解决方案文件夹，属性视图去，需要按左右键。

但是怎么样打开最近编辑后关闭的文件呢，Visual Studio 很客气的又没有提供此功能，于是留给了 Resharper。在我这里这个快捷键是 Ctrl+E,Ctrl+E，没错，按两次。如果你的不是， 那么在 Resharper-View-Recent Files 菜单下看看它是什么。因为你会时常用到。打开一个文件的列表，用方向键选择并回车就会在编辑器中打开。

很多人说 Resharper 的性能问题，我想，一个可能的原因是打开的文档太多了，如果你有时刻关闭不需要的文档的习惯，性能或许不会那么差，并且你可以随时打开这些你关闭了的文档，就像在已经打开的文档中切换一样的方便。

我的团队中没有用到敏捷开发那些高级的东西，但是我们还是保持着每次改动都仅涉及两三个文件的好习惯，并且频繁的 commite 到源代码服务器上去。所以，我每次真正要编辑的文件不多，性能不是问题。

和大家分享了很多 Resharper 使用的技巧，点点滴滴都已经融入我日常的开发工作中了。

当然很不全面，例如与 Nunit 的集成，由于我们不是测试驱动，所以没有任何体验，自然也不敢大放厥词；也很主观，我觉得它好，你可能觉得它不好，萝卜青菜各有所爱。再说，它也不是没有白痴的地方，在文档上点右键 增加的一个 Close All 功能，可以关闭所有打开的文档，关闭了干什么，对着一个空白的屏幕发呆么？我觉得原生的“除此之外全部关闭”就够了。还有一个定位的功能 （Locate in Solution Explorer），真是没用，如果你在 VS 选项中设置了，在解决方案管理器中跟踪活动项，那么 VS 自动就给你定位了。