清華
电脑学堂

C# 2012 程序设计
实践教程

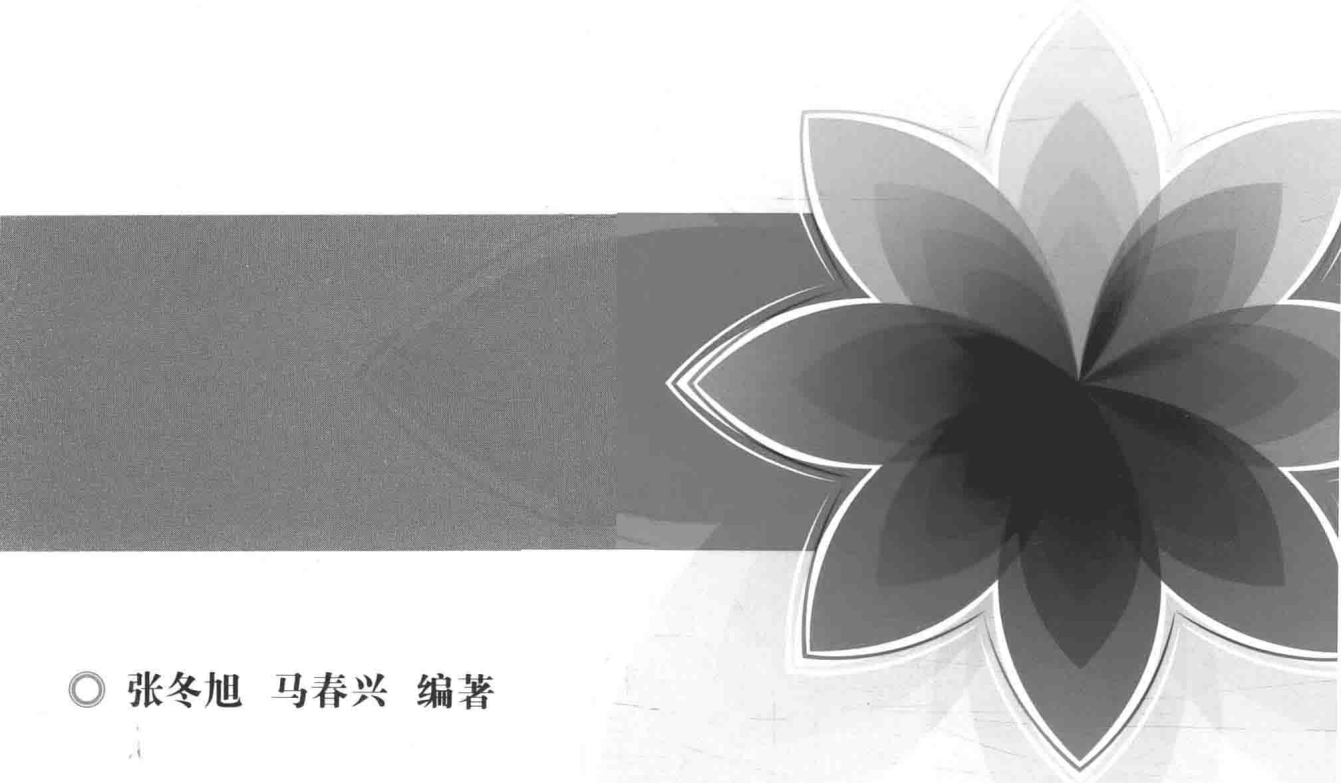
- **构思独特**，所有案例来自一线实战场景；
- **实用性强**，将抽象的理论结合到实战案例上；
- **内容全面**，结构清晰，体例丰富；
- **视频教学**，专业教学视频帮助读者快速上手。

张冬旭 马春兴 编著

清华大学出版社

清華
电脑学堂

C# 2012 程序设计实践教程



◎ 张冬旭 马春兴 编著

清华大学出版社

内 容 简 介

C#在编程语言排行中始终处于领先位置，从4.5版本开始运用新的架构和模块，使C#的编写更加灵活和智能化。本书主要讲述C#的理论和应用。全书共分为17章，内容包括：.NET Framework，C# 5.0功能、数据类型、变量、常量、类型转换、运算符和控制语句，类、对象、结构、枚举和接口，数组、集合、自定义集合和泛型，String类、StringBuilder类、DateTime结构、TimeSpan结构、Math类、Random类和Regex类，委托、事件和异常，LINQ简单查询和LINQ to SQL查询，WPF的发展历史、WPF 4.5新增功能、WPF体系结构、XAML和Application类，WPF的常用控件、依赖项属性、附加属性、路由事件和附加事件，绘制基本图形、画刷、动画、图像和多媒体，以及WPF中的数据绑定技术等。本书最后综合所学的C#知识制作了简单的文件资源管理器。

本书可作为在校大学生学习使用C#进行课程设计的参考资料，也适合作为高等院校相关专业的教学参考书，还可以作为非计算机专业学生学习C#语言的参考书。

图书在版编目（CIP）数据

C# 2012程序设计实践教程/张冬旭，马春兴编著. —北京：清华大学出版社，2016
（清华电脑学堂）
ISBN 978-7-302-41848-1

Ⅰ. ①C…　Ⅱ. ①张…　②马…　Ⅲ. ①C语言-程序设计-教材　Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2015）第252088号

责任编辑： 夏兆彦
封面设计： 张　阳
责任校对： 徐俊伟
责任印制： 沈　露

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 清华大学印刷厂
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　张：** 25.75　　**字　数：** 611千字
版　　次： 2016年6月第1版　　**印　次：** 2016年6月第1次印刷
印　　数： 1～2000
定　　价： 49.00元

产品编号：060062-01

前 言

FOREWORD

C#自面世以来，以其易学易用、功能强大的优势被广泛应用。它是微软公司为 Visual Studio 开发平台推出的一种简洁、类型安全的面向对象编程语言，开发者通过 C#可以编写在.NET Framework 上运行的各种安全可靠的应用程序。C#目前是主流的编程开发语言之一，Web 开发者通过使用 C#语言，不仅可以开发集声音、动画、视频为一体的多媒体应用程序和网络应用程序，而且可以和数据库结合开发出功能强大的管理系统。

目前最新的 C#版本是 5.0，支持.NET 4.5 框架，最新的开发工具是 Visual Studio 2012。本书向读者详细介绍了 C# 5.0 的基础知识，还将介绍基于.NET 4.5 框架之上的 4 种开发技术，即 LINQ、WPF、WCF 和 WF。

1. 本书内容

本书以目前主流的 C# 5.0 及 Visual Studio 2012 为例进行介绍。全书共分为 17 章，主要内容如下。

第 1 章　搭建 C# 2012 的开发框架。本章详细介绍了.NET Framework、C#语言和 Visual Studio 2012 这三部分内容。

第 2 章　C#入门语法。本章详细介绍了 C#的入门语法，包括数据类型、变量、常量、类型转换、运算符和控制语句等相关内容。

第 3 章　C#面向对象基础。本章重点介绍了面向对象编程的类，包括类、类的对象、构造函数、析构函数、类的成员变量、面向对象编程的三大特性以及常用的一些可选修饰符等内容。

第 4 章　C#面向对象的其他知识。本章详细介绍了 C#中的常用结构、枚举和接口，具体内容包括它们的定义、结构成员、与类的区别以及具体实现等。

第 5 章　数组、集合和泛型。本章介绍了数组、集合和泛型的概念及其使用，主要内容包括一维数组、二维数组、静态数组、动态数组、常用的内置集合类、自定义集合类以及常用的泛型等。

第 6 章　C#中常用的处理类。本章介绍了 C#中经常使用到的一些内置操作类，例如操作字符串的 String 和 StringBuilder 类，操作日期和时间的 DateTime 和 TimeSpan 结构，与数学工具相关的 Math 和 Random 类，以及正则表达式的匹配和常用的 Regex 类。

第 7 章　委托和异常。本章从 C#中的委托开始介绍，然后介绍 C#中的事件，最后介绍 C#中如何处理异常。

第 8 章　LINQ 简单查询。本章详细介绍了 LINQ 的组成部分、各子句的应用以及一些常规查询的实现。

第 9 章　LINQ to SQL。本章详细介绍了 LINQ 查询数据库数据的方法，即 LINQ to SQL。主要内容包括 LINQ 对象关系设计器、DateContext 类以及如何插入数据、更新数

据和删除数据等。

第 10 章　WPF 基础入门。本章从 WPF 的诞生开始，详细地向读者介绍了 WPF 开发所需的基础知识，包括 WPF 的体系结构、WPF 项目的创建、XAML 标记基础以及 Application 类的介绍。

第 11 章　WPF 控件布局。本章从 WPF 布局控件开始介绍，然后又详细介绍了 WPF 中提供的内容控件和标准控件。

第 12 章　WPF 的属性和事件。本章着重介绍了 WPF 中的属性和事件，包括依赖项属性、附加属性、路由事件和附加事件等内容。

第 13 章　WPF 图形和多媒体。本章详细介绍了 WPF 中图形和多媒体的使用，包括图形的绘制、颜色的控制、动画的制作、图像的处理和多媒体的使用等。

第 14 章　WPF 数据绑定技术。本章详细讨论了 WPF 中有关数据绑定的内容，包括如何绑定到单个属性、更改绑定模型、绑定到多个属性、绑定不可见的元素，以及绑定数据库中的数据。

第 15 章　WCF 概述。本章简单介绍了 WCF 的基础知识，包括 WCF 的概念、使用优势、技术要素和应用场景等。

第 16 章　WF 框架。本章简单 WF 的基础知识，包括工作流介绍、数据模型、活动以及工作流的创建和使用等。

第 17 章　WPF 制作文件资源管理器。本章将以 Visual Studio 2012 作为开发工具，Microsoft SQL Server 2012 作为开发数据库，C# 5.0 作为开发语言，利用 WPF 制作简单的文件资源管理器。实现功能包括目录和文件的查看、搜索、复制、剪切、粘贴、排序、创建以及根据日期分类等。

2. 本书特色

本书是针对初、中级读者量身订做，由浅入深地讲解了 C#语言的应用。书中采用大量的范例进行讲解，力求通过实际操作帮助读者更容易地使用 C#开发应用程序。

1）知识点全面

本书紧紧围绕 C# 5.0 的基础知识开发展开讲解，具有很强的逻辑性和系统性。另外，还介绍了基于 C# 5.0 开发的 LINQ、WPF、WCF 和 WF 技术。

2）基于理论，注重实践

本书不仅介绍了理论知识，而且还介绍了开发过程。在章节的合适位置安排了综合应用实例或者小型应用程序，将理论应用到实践当中，以加强读者实际应用能力，巩固开发基础和知识。

3）随书光盘

本书为每一章的范例和综合案例配备了视频教学文件，读者可以通过视频文件更加直观地学习 C# 5.0 的知识。

4）网站技术支持

读者在学习或者工作的过程中，如果遇到实际问题，可以直接登录 www.ztydata.com.cn 与作者取得联系，作者会在第一时间内给予帮助。

3. 读者对象

本书可作为在校大学生学习使用C#进行课程设计的参考资料，也适合作为高等院校相关专业的教学参考书，还可以作为非计算机专业学生学习C#语言的参考书。

（1）想学习C# 5.0开发技术的人员。

（2）C# 5.0的初级和中级开发人员。

（3）想使用WPF进行应用程序开发的人员。

（4）准备从事与C#语言开发有关的人员。

（5）各大中专院校的在校学生和相关授课老师。

除了封面署名人员之外，参与本书编写的人员还有李海庆、王咏梅、康显丽、王黎、汤莉、倪宝童、赵俊昌、方宁、郭晓俊、杨宁宁、王健、连彩霞、丁国庆、牛红惠、石磊、王慧、李卫平、张丽莉、王丹花、王超英、王新伟等。书中难免存在疏漏及不足之处，欢迎读者通过清华大学出版社网站 www.tup.tsinghua.edu.cn 与作者联系，帮助我们改正提高。

目 录

CONTENTS

第 1 章　搭建 C# 2012 的开发框架

C#的全称是 Microsoft Visual C#，它是 Microsoft 提供的一种强大的、面向对象的开发语言。C#在.NET Framework 中扮演着重要角色，一些人甚至将它与 C 在 UNIX 开发中的地位相提并论。C#也是目前最流行的开发语言之一，由于 C#语言的类库全部封装在.NET 框架中，因此本章在介绍 C#语言之前，会简单介绍.NET 框架。

本章主要包括三部分内容：.NET Framework、C#语言和 Visual Studio 2012 开发工具。通过本章的学习，读者可以了解和熟悉.NET 框架和 C#语言的知识，也可以熟练地通过 Visual Studio 2012 开发工具创建控制台应用程序。

本章学习要点：

- 了解.NET 框架的实现目标
- 熟悉公共语言运行时和类库
- 了解 C#语言的特色优势
- 熟悉 C# 5.0 的增强功能和修改功能
- 掌握 Visual Studio 2012 的安装
- 掌握 Visual Studio 2012 的使用
- 掌握如何创建控制台应用程序
- 熟悉如何引用命名空间

1.1 .NET Framework 概述

.NET Framework 即 Microsoft .NET Framework，又被称为.NET 框架。它是支持生成和运行下一代应用程序与 Web 服务内容的 Windows 组件。.NET Framework 提供了托管执行环境、简化开发和部署以及与各种编程语言的集成。

简单来说，如果想要开发和运行.NET 运行程序，就必须首先安装.NET Framework。

1.1.1 .NET Framework 组件

.NET Framework 是一种技术，该技术支持生成和运行下一代应用程序与 XML Web 服务。.NET Framework 旨在实现以下几个目标。

（1）提供一个一致的面向对象的编程环境，而无论对象代码是在本地存储和执行，还是在本地执行但在 Internet 上分布，或者是在远程执行的。

（2）提供一个将软件部署和版本控制冲突最小化的代码执行环境。

（3）提供一个可提高代码（包括由未知的或不完全受信任的第三方创建的代码）执行安全性的代码执行环境。

（4）提供一个可消除脚本环境或解释环境的性能问题的代码执行环境。

（5）使开发者的经验在面对类型大不相同的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）时保持一致。

（6）按照工业标准生成所有通信，以确保基于.NET Framework 的代码可与任何其他代码集成。

.NET Framework 包含两个组件：公共语言运行时和.NET Framework 类库。公共语言运行时是.NET Framework 的基础。.NET Framework 类库是一个综合性的面向对象的可重用类型集合。如图 1-1 所示为公共语言运行时、.NET Framework 类库与应用程序以及整个系统之间的关系。

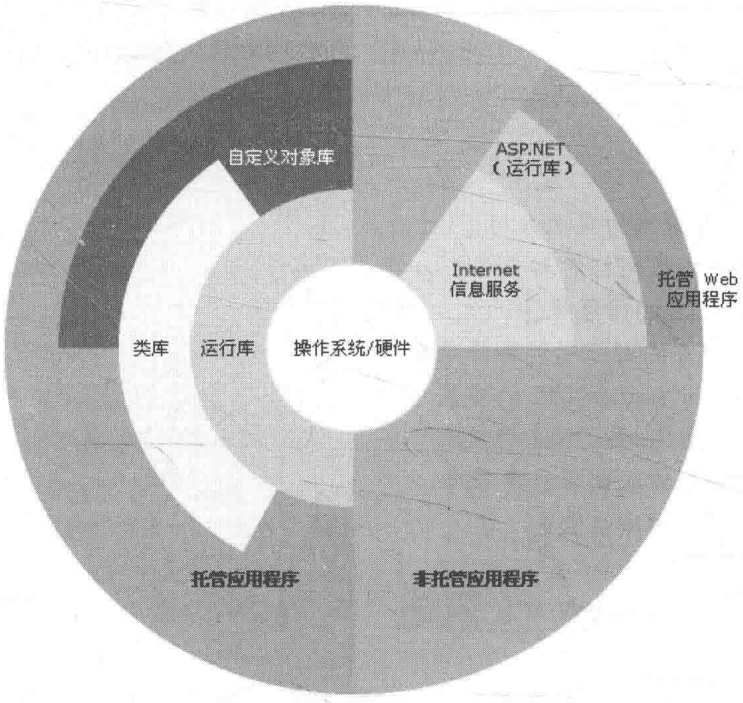


图 1-1　公共语言运行时、类库和应用程序以及整个系统的关系

目前，.NET Framework 4.5 是相当稳定的版本。.NET Framework 4.5 在之前版本（如.NET Framework 4.0）的基础上新增了多个功能，并且改进了之前版本的部分功能。如实现程序集的自动绑定重定向、可以在垃圾回收过程中显式压缩大对象堆、支持对区域性字符串排序以及比较数据进行版本控制。除这些基本功能外，还对 WPF、WCF 和 WF 等应用程序的功能进行了添加和更改。

1.1.2　公共语言运行时

.NET Framework 提供了一个称为公共语言运行时（Command Language Runtime，CLR）的运行环境，它运行代码并提供使开发过程更轻松的服务。作为.NET Framework 的核心组件，它是执行时管理代码的代理，并提供内存管理、线程管理和远程处理等核

心服务。

公共语言运行时通过通用类型系统（Common Type System，CTS）和公共语言规范（Common Language Specification，CLS）定义标准数据类型和语言之间相互操作的规则。Just-In-Time（JIT）编辑器在运行应用程序之前把中间语言（Intermediate Language，IL）代码转换为可执行代码。公共语言运行时还管理应用程序，在应用程序运行时为其分配内存和解除分配内存。

1. 通用类型系统

通用类型系统定义如何在 CLR 中声明、使用和管理类型，同时也是 CLR 支持跨语言集成的一个重要组成部分。通用类型系统支持.NET Framework 提供的两种常用类型，即值类型和引用类型，每一种类型又可以包含多种类型。

通用类型系统的实现功能如下。

（1）建立一个支持跨语言的集成、类型安全和高性能代码执行的框架。

（2）提供一个支持完整实现多种编程语言的面向对象的模型。

（3）定义各语言必须遵守的规则，有助于确保用不同语言编写的对象能够交互作用。

2. 公共语言规范

公共语言规范是一组结构和限制条件，作为库开发者和编译器编写者的指南。公共语言规范定义所有基于.NET Framework 的语言都必须支持的最小功能集，定义规则包括以下几种。

（1）CLS 定义命名变量的标准规则。例如，与 CLS 兼容的变量名称都必须以字母开始，并且不能包含空格。除了变量名之间的大小写区别之外，还要有其他区别。

（2）CLS 定义原数据类型，如 Int32、Int64、Double 和 Boolean 等。

（3）CLS 禁止无符号数值数据类型。有符号数值数据类型的一个数据位被保留来指示数值的正负，而无符号数据类型没有保留这个数据位。

（4）CLS 定义对支持基于 0 的数组的支持。

（5）CLS 指定函数参数列表的规则，以及参数传递给函数的方式。例如，CLS 禁止使用可选的参数。

（6）CLS 定义事件名和参数传递给事件的规则。

（7）CLS 禁止内存指针和函数指针，但是可以通过委托提供类型安全的指针。

3. 中间语言

使用.NET 开发的任何应用程序在执行之前都会编译为目标计算机能够理解的语言，即本机代码。在.NET Framework 下这个过程分为两个阶段：首先把应用程序编译成一种称为中间语言（Intermediate Language，IL）的独立于硬件的格式；然后就是使用 JIT 编辑器的阶段，它把中间语言编译为专门用于目标操作系统和目标机器结构的本机代码，

只有这样目标操作系统才能执行应用程序。

4. 托管执行过程

公共语言运行时执行的代码称为托管代码（Managed Code），其作用之一就是防止一个应用程序干扰另外一个应用程序的运行，这个过程称为安全性。使用类型安全的托管代码，一个应用程序就不会覆盖另一个应用程序分配的内容。创建托管代码的步骤如下。

（1）选择一个合适的编译器，它能够生成适合 CLR 执行的代码，并且使用.NET Framework 提供的资源。

（2）把应用程序编译为独立于机器的中间语言。

（3）在执行时，把中间语言代码转换为本机可执行文件。

（4）在应用程序执行时，调用.NET Framework 和 CLR 提供的资源。

5. 自动内存管理

自动内存管理是公共语言运行时在托管执行过程中提供的服务之一。公共语言运行时的垃圾回收器为应用程序管理内存的分配和释放。对于开发者而言，这意味着在开发托管应用程序时不必编写执行内存管理任务的代码。自动内存管理解决了常见的一些问题，例如忘记释放对象并导致内存泄漏，或者尝试访问已释放对象的内存。

1.1.3 .NET Framework 类库

.NET Framework 类库是一个由 Microsoft .NET Framework SDK 中包含的类、接口和值类型组成的库，提供对系统功能的访问。.NET Framework 类库是建立.NET Framework 应用程序、组件和控件的基础。.NET Framework 类库中还包含.NET Framework 中定义的所有类型。

.NET Framework 类库由命名空间组成，每个命名空间都包含可以在程序中使用的类型，如类、结构、枚举、委托和接口等。命名空间名称是类型的完全限定名（namespace.typename）的一部分，所有 Microsoft 提供的命名空间都以 System 或 Microsoft 开头，如表 1-1 所示列出了.NET Framework 类库中提供的一些常见命名空间。

表 1-1 .NET Framework 类库中提供的一些常见命名空间

命名空间	说明
Microsoft.JScript	Microsoft.JScript 命名空间包含具有以下功能的类：支持用 JScript 语言生成代码和进行编译
Microsoft.Win32	Microsoft.Win32 命名空间提供具有以下功能的类型：处理操作系统引发的事件、操纵系统注册表、代表文件和操作系统句柄
System	包含允许将 URI 与 URI 模板和 URI 模板组进行匹配的类
System.Collections	包含具有定义各种标准的、专门的和通用的集合对象等功能的类
System.Data	包含访问和管理多种不同来源的数据的数类
System.Dynamic	提供支持动态语言运行时的类和接口
System.Drawing	包含提供与 Windows 图形设备接口的接口类

续表

命名空间	说明
System.IO	包含支持输入和输出的类，包括以同步或异步方式在流中读取和写入数据、压缩流中的数据、创建和使用独立存储区以及处理出入串行端口的数据流等
System.Windows.Forms	定义包含工具箱中的控件及窗体自身的类
System.Net	包含用于网络通信的类或命名空间
System.Linq	该命名空间下的类支持使用语言集成查询（LINQ）的查询
System.Text	包含用于字符编码和字符串操作的类型
System.XML	该命名空间包含用于处理 XML 类型的数据

1.2 C#语言概述

.NET Framework 支持多种开发语言，如 C#、Visual Basic（VB）和 F#等。在众多支持的开发语言中，C#和 VB 最流行，VB 一般用来快速开发，在小型的 Windows 应用系统中最为常用。C#语言是 Microsoft 重点推出的开发语言，它结合了 C 语言和 C++语言的一些优点，然后又去除了指针等难以理解的概念，是一门方便开发的语言。

1.2.1 C#语言的特点

C#是一种简洁、类型安全的面向对象的语言。与其他语言相比，C#语言具备简单、方便和快速开发等优点，主要特色优势如下。

（1）C#支持面向对象开发，并且有.NET 底层类库的支持，可以轻松创建对象。

（2）C#的开发效率高。C#的开发工具 Visual Studio 2012 支持拖放式添加控件，开发者可以轻松完成桌面布局。

（3）C#通过内置的服务，使组件可以转化为 XML 网络服务。

（4）C#提供了对 XML 的强大支持，可以轻松地创建 XML，也可以将 XML 数据应用到程序中。

（5）C#具有自动回收功能，它不用像 C++一样，为程序运行中的内存管理伤脑筋。

（6）C#提供类型安全机制，可以避免一些常见的类型问题，如类型转换和数组越界等。

（7）在.NET Framework 中，C#可以自由地和其他语言（如 VB）进行切换。

1.2.2 C# 5.0 新增功能

在 Visual Studio 2012 开发工具中，自带的版本是 C# 5.0。换句话说，.NET Framework 4.5、C# 5.0 伴随着 Visual Studio 2012 一起正式发布。与之前版本的 C#（如 C# 4.0）相比，C# 5.0 新增了一些功能。

1. 异步编程

使用异步功能可以更轻松、更直观地编写异步代码，这使异步编程几乎和同步编程一样简单。在.NET Framework 4.5 中，通过 async 和 await 两个关键字，引入了一种新的基于任务的异步编程模型（TAP）。在这种方式下，可以通过类似同步方式编写异步代码，极大地简化了异步编程模式。

【范例 1】

如下代码演示了一个异步编程的例子。

```
static async void DownloadStringAsync2(Uri uri) {
    var webClient = new WebClient();
    var result = await webClient.DownloadStringTaskAsync(uri);
    Console.WriteLine(result);
}
```

如果不使用上述代码，使用之前同步编程的方式编写的代码如下。

```
static void DownloadStringAsync(Uri uri) {
    var webClient = new WebClient();
    webClient.DownloadStringCompleted += (s, e) => {
          Console.WriteLine(e.Result);
    };
    webClient.DownloadStringAsync(uri);
}
```


2. 调用方信息

C# 5.0 版本方便获取有关方法的调用方信息。使用调用方信息属性可以标识源代码、源代码和调用方的成员名称的文件路径。该信息可用于跟踪，用于调试以及创建诊断工具。

大多数时候，开发者需要在运行过程中记录一些调测的日志信息，可使用以下代码。

```
public void DoProcessing() {
    TraceMessage("Something happened.");
}
```

为了调测方便，除了事件信息外，往往还需要知道发生该事件的代码位置以及调用栈信息。在 C++中，开发者可以通过定义一个宏，然后在宏中通过_FILE_和_LINE_来获取当前代码的位置，但是 C#并不支持宏，只能通过 StackTrace 实现这一功能，但是 StackTrace 又不是很可靠。针对上述描述，在.NET Framework 4.5 中引入了三个属性：CallerMemberName、CallerFilePath 和 CallerLineNumber。在编译器的配合下，上述三个属性可以分别获取到调用函数的名称（即成员名称）、调用文件和调用行号。例如，前面提到的 TraceMessage()函数可以通过以下代码实现。

```
public void TraceMessage(string message,
```