Skill 3.1: Validate application input

Choose the appropriate data collection type

- `Array`
  - e.g.

```
//migrate an array into a larger one.
int[] data = {1,2,3,4};
int temp = new int[5];
//destination,position
data.CopyTo(temp,0);
data = temp;
```

  - fixed size
  - An array of value types (for example an array of integers) holds the values themselves within the array, whereas for an array of reference types (for example an array of objects) each element in the array holds a reference to the object.Array自己是引用类型。
  - When an array is created, each element in the array is initialized to the default value for that type. Numeric elements are initialized to 0, reference elements to null, and Boolean elements to false.
  - **Multi-dimensional arrays**

```
string [,] compass = new string[3, 3]
{
    { "NW","N","NE" },
    {"W", "C", "E" },
    { "SW", "S", "SE" }
};

Console.WriteLine(compass[0, 0]);  // prints NW
Console.WriteLine(compass[2, 2]);  // prints SE
```

  - **Jagged arrays**

```
int[][] jaggedArray = new int[][]
{
    new int[] {1,2,3,4 },
    new int[] {5,6,7},
    new int[] {11,12}
}
```

> 是一个二维数据，每一行的长度不等

- `ArrayList`
  - 保存的是item的object类型的引用，所以一个ArrayList可以存放不同类型的对象，但是使用的时候需要cast

```
class Program
    {
        static void Main(string[] args)
        {
            ArrayList arrayList = new ArrayList();
            arrayList.Add(new C1());
            arrayList.Add(new C2());
            foreach (var item in arrayList)
            {
                Console.WriteLine(item.ToString());
                //Temp.C1
                //Temp.C2
            }
        }
    }
    class C1
    {
    }
    class C2
    {
    }
```

- `List` : use generic,the number of items being stored is not known in advance

```
List<string> list = new List<string>();
list.Add("add to end of list");    // add to the end of the list
list.Insert(0, "insert at start"); // insert an item at the start
list.Insert(1, "insert new item 1"); // insert at position
list.InsertRange(2, new string[] { "Rob", "Immy" }); // insert a range
list.Remove("Rob");               // remove first occurrence of "Rob"
list.RemoveAt(0);                 // remove element at the start
list.RemoveRange(1, 2);           // remove two elements
list.Clear();                     // clear entire list
```

- `Queue`: FIFO
  - `Enqueue(some data);`
  - `Dequeue(some data);`
- `Stack`:LIFO
  - Push(some data);
  - Pop();
- `LinkedList<T>`:perform frequent insertions and deletions of data items
- `Dictionary<TKey,TValue>`

```
Dictionary<int, string> dictionary = new Dictionary<int, string>();
dictionary.Add(1, "Rob Miles");  // add an entry
dictionary.Remove(1);            // remove the entry with the given key
```

e.g.统计字数

```
Dictionary<string, int> counters = new Dictionary<string, int>();

string text = File.ReadAllText("input.txt");
string[] words = text.Split(new char[] { ' ', '.', ',' },
    StringSplitOptions.RemoveEmptyEntries);

foreach (string word in words)
{
    string lowWord = word.ToLower();
    if (counters.ContainsKey(lowWord))
            counters[lowWord]++;
        else
            counters.Add(lowWord, 1);
}
```

- Dictionary不带Sort方法，但是可以用LINQ排序

```
var items = from pair in counters
            orderby pair.Value descending
            select pair;

foreach (var pair in items)
{
    Console.WriteLine("{0}: {1}", pair.Key, pair.Value);
}
```

- `HashSet`
  - **A set is an unordered collection of items.**
  - **Each of the items in a set will be present only once.**
  - e.g.1

```
HashSet<string> set = new HashSet<string>();
set.Add("Rob Miles");    // add an item
set.Remove("Rob Miles"); // remove an item
set.RemoveWhere(x => x.StartsWith("R")); // remove all items that start with 'R'
```

  - e.g.2

```
HashSet<string> t1Styles = new HashSet<string>();
t1Styles.Add("Electronic");
t1Styles.Add("Disco");
t1Styles.Add("Fast" );

HashSet<string> t2Styles = new HashSet<string>();
t2Styles.Add("Orchestral");
t2Styles.Add("Classical");
t2Styles.Add("Fast");

HashSet<string> search = new HashSet<string>();
search.Add("Fast");
search.Add("Disco");

if (search.IsSubsetOf(t1Styles))
    Console.WriteLine("All search styles present in T1");

if (search.IsSubsetOf(t2Styles))
    Console.WriteLine("All search styles present in T2");
```

- `StringCollection` and `StringList`: store only string

Initialize a collection

Initialize a collection

```
int[] arrayInit = { 1, 2, 3, 4 };

ArrayList arrayListInit =   new ArrayList { 1, "Rob Miles", new ArrayList() };
List<int> listInit = new List<int>{ 1, 2, 3, 4 };

Dictionary<int, string> dictionaryInit =  new Dictionary<int, string> {
        {1, "Rob" },
        {2, "Immy" } };

HashSet<string> setInit = new HashSet<string> { "Electronic", "Disco", "Fast" };

Queue<string> queueInit = new Queue<string>( new string [] {"Rob", "Immy" });

Stack <string> stackInit = new Stack<string>(new string[] { "Rob", "Immy" });
```

Custom collection

- **create a new type that implements the ICollection interface.**
- **use an existing collection class as the base (parent) class of a new collection type.**

```
class TrackStore : List<MusicTrack>
{
    public int RemoveArtist(string removeName)
    {
        List<MusicTrack> removeList = new List<MusicTrack>
();
        foreach (MusicTrack track in this)
            if (track.Artist == removeName)
                removeList.Add(track);

        foreach (MusicTrack track in removeList)
            this.Remove(track);

        return removeList.Count;
    }
}
```

regular expression

- `replace()`:

```
string input = "Rob    Mary David   Jenny  Chris   Imogen       Rodney";

string regularExpressionToMatch = " +";

string patternToReplace = ",";

string replaced = Regex.Replace(input, regularExpressionToMatch, patternToReplace);
```

> output: Rob,Mary,David,Jenny,Chris,Imogen,Rodney

- common regular signs:
  - `+`: one or more times
  - `.` :match any character

    e.g.

    ```
    string input = "Rob Miles:My Way:120";
    string regexToMatch = ".+:.+:.+";
    if (Regex.IsMatch(input,regexToMacth))
        Console.WriteLine("Valide input")
    ```

  - `[ch-ch]` : match any character in that range, so the sequence `[0-9]` will match any digit, and the sequence `[0-9]+` will match one or more digits.
  - `^`:the start of a line
  - `$`: the end of a line
  - `|` :or

    ```
    string regexToMatch = @"^([a-z]|[A-Z]| )+:([a-z]|[A-Z]| )+:
    [0-9]+$";
    ```

Reading values

- `public static bool TryParse (string s, out int result);`

- `public static int ToInt32 (object value);`
- Here are some items to keep in mind whenconsidering the three different ways of converting values:
  - `int.Parse` will throw an exception if the supplied argument is `null` or if a string does not contain text that represents a valid value.
  - `int.TryParse` will return false if the supplied argument is null or if a string does not contain text that represents a valid value.
  - `Convert.ToInt32` will throw an exception if the supplied string argument does not contain text that represents a valid value. It will not, however, throw an exception if the supplied argument is `null`. It instead returns the default value for that type. If the supplied argument is null the `ToInt32` method returns 0.

Skill 3.2: Perform symmetric and asymmetric encryption

symmetric encryption

- the key at each side of the conversation is exactly the same—the book
- **AES**: Any new systems requiring data encryption should use AES
  - e.g.

```
using System;
using System.IO;
using System.Security.Cryptography;

namespace LISTING_3_14_AES_encryption
{
    class Program
    {
        static void DumpBytes(string title, byte [] bytes)
        {
            Console.Write(title);
            foreach (byte b in bytes)
            {
                Console.Write("{0:X} ", b);
            }
            Console.WriteLine();
        }

        static void Main(string[] args)
        {
            string plainText = "This is my super secret data";

            // byte array to hold the encrypted message
            byte[] cipherText;

            // byte array to hold the key that was used for encryption
            byte[] key;

            // byte array to hold the initialization vector that was used for encryption
            byte[] initializationVector;

            // Create an Aes instance
            // This creates a random key and initialization vector

            using (Aes aes = Aes.Create())
            {
                // copy the key and the initialization vector
                key = aes.Key;
                initializationVector = aes.IV;

                // create an encryptor to encrypt some data
                // should be wrapped in using for production code
                ICryptoTransform encryptor = aes.CreateEncryptor();

                // Create a new memory stream to receive the
                // encrypted data.

                using (MemoryStream encryptMemoryStream = new MemoryStream())
                {
                    // create a CryptoStream, tell it the stream to write to
                    // and the encryptor to use. Also set the mode
                    using (CryptoStream encryptCryptoStream =
                            new CryptoStream(encryptMemoryStream,
                                encryptor, CryptoStreamMode.Write))
                    {
                        // make a stream writer from the cryptostream
                        using (StreamWriter swEncrypt =
                                new StreamWriter(encryptCryptoStream))
                        {
                            //Write the secret message to the stream.
```

```
                    swEncrypt.Write(plainText);
                }
                // get the encrypted message from the stream
                cipherText= encryptMemoryStream.ToArray();
            }
        }
    }

    // Dump out our data
    Console.WriteLine("String to encrypt: {0}", plainText);
    dumpBytes("Key: ", key);
    dumpBytes("Initialization Vector: ", initializationVector);
    dumpBytes("Encrypted: ", cipherText);

    Console.ReadKey();
        }
    }
}
```

- **DES**: replaced by AES
- **RC2**: Insecure now
- **Rijndael**: AES is implemented as a subset of Rijndael
- **TripleDES**: The electronic payment industry makes use of TripleDES

asymmetric encryption

- 用非对称加密来传递信息：公钥发给对方，对方用自己公钥加密的信息只有自己的私钥才能解密，中途的人只能得到加密后的信息。（自己的私钥用来解密）
- 用非对称加密来签名：自己用自己的私钥加密信息，对方（其他所有人）只有用自己的公钥才能解密，对方解密成功就知道信息一定来自自己。（自己的私钥用来加密）
    - 我发布文件给对方的同时，我也发布了我自己的公钥和用自己私钥加密后的"message digest"如checksum，对方可以用我的公钥解密message digest之后和自己计算的message digest比较，比较结果一致则说明文件没有被篡改。

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace LISTING_3_16_RSA_encryption
{
    class Program
    {
        static void Main(string[] args)
        {
            string plainText = "This is my super secret data";
            Console.WriteLine("Plain text: {0}", plainText);

            // RSA works on byte arrays, not strings of text
            // This will convert our input string into bytes and back
            ASCIIEncoding converter = new ASCIIEncoding();

            // Convert the plain text into a byte array
            byte[] plainBytes = converter.GetBytes(plainText);

            dumpBytes("Plain bytes: ", plainBytes);

            byte[] encryptedBytes;
            byte[] decryptedBytes;

            // Create a new RSA to encrypt the data
            // should be wrapped in using for production code
            RSACryptoServiceProvider rsaEncrypt = new RSACryptoServiceProvider();

            // get the keys out of the encryptor
            string publicKey = rsaEncrypt.ToXmlString(includePrivateParameters: false);
            Console.WriteLine("Public key: {0}", publicKey);
            string privateKey = rsaEncrypt.ToXmlString(includePrivateParameters: true);
            Console.WriteLine("Private key: {0}", privateKey);

            // Now tell the encyryptor to use the public key to encrypt the data
            rsaEncrypt.FromXmlString(privateKey);

            // Use the encryptor to encrypt the data. The fOAEP parameter
            // specifies how the output is "padded" with extra bytes
            // For maximum compatibility with receiving systems, set this as
            // false
            encryptedBytes = rsaEncrypt.Encrypt(plainBytes, fOAEP:false);

            dumpBytes("Encrypted bytes: ", encryptedBytes);

            // Now do the decode - use the private key for this
            // We have sent someone our public key and they
            // have used this to encrypt data that they are sending to us

            // Create a new RSA to decrypt the data
            // should be wrapped in using for production code
            RSACryptoServiceProvider rsaDecrypt = new RSACryptoServiceProvider();

            // Configure the decryptor from the XML in the private key
            rsaDecrypt.FromXmlString(privateKey);

            decryptedBytes = rsaDecrypt.Decrypt(encryptedBytes, fOAEP: false);

            dumpBytes("Decrypted bytes: ", decryptedBytes);
            Console.WriteLine("Decrypted string: {0}",
                              converter.GetString(decryptedBytes));

            Console.ReadKey();
        }
    }
}
```

  - 公钥私钥保存问题：新建`RSACryptoServiceProvider`的时候可以告诉他公钥私钥保存在哪里：`CspParameters`类，`CspParameters.Flags = CspProviderFlags.UserMachineKeyStore`会用`Machine level key storage`，把密钥存放在`C:\ProgramData\Microsoft\Crypto\RSA\MachineKeys`

- Digital signatures and certificates
  - 问题：Bob给Alice发送信息，Eve可以冒充Bob给Alice发送Eve自己的公钥私钥，或者Eve可以冒充Alice收Bob的信息，然后装作是Alice，用Bob给的公钥加密信息后给Bob回复。解决方法：大家去certification authority注册，向certification authority提交自己的信息
  - VS developer command prompt中 使用`makecert democert.cer`创建测试用的certificate ()

    本来是`new RSACryPtoServiceProvider()`，现在用`certificate.PrivateKey as RSACryPtoServiceProvider`

```
using System;
using System.Security.Cryptography.X509Certificates;
using System.Security.Cryptography;
using System.Text;

namespace LISTING_3_20_Signing_data
{
    class Program
    {
        static void Main(string[] args)
        {
            // This will convert our input string into bytes and back
            ASCIIEncoding converter = new ASCIIEncoding();

            // Get a crypto provider out of the certificate store
            // should be wrapped in using for production code
            X509Store store = new X509Store("demoCertStore", StoreLocation.CurrentUser);

            store.Open(OpenFlags.ReadOnly);

            // should be wrapped in using for production code
            X509Certificate2 certificate = store.Certificates[0];

            // should be wrapped in using for production code
            RSACryptoServiceProvider encryptProvider =
                        certificate.PrivateKey as RSACryptoServiceProvider;

            string  messageToSign = "This is the message I want to sign";
            Console.WriteLine("Message: {0}", messageToSign);

            byte[] messageToSignBytes = converter.GetBytes(messageToSign);
            dumpBytes("Message to sign in bytes: ", messageToSignBytes);

            // need to calculate a hash for this message - this will go into the
            // signature and be used to verify the message
            // Create an implementation of the hashing algorithm we are going to use
            // should be wrapped in using for production code
            HashAlgorithm hasher = new SHA1Managed();
            // Use the hasher to hash the message
            byte[] hash = hasher.ComputeHash(messageToSignBytes);
            dumpBytes("Hash for message: ", hash);

            // Now sign the hash to create a signature
            byte[] signature = encryptProvider.SignHash(hash,CryptoConfig.
            MapNameToOID("SHA1"));
            dumpBytes("Signature: ", messageToSignBytes);

            // We can send the signature along with the message to authenticate it
            // Create a decryptor that uses the public key
            // should be wrapped in using for production code

            RSACryptoServiceProvider decryptProvider =
                        certificate.PublicKey.Key as RSACryptoServiceProvider;

            // Now use the signature to perform a successful validation of the message
            bool validSignature = decryptProvider.VerifyHash(hash,
                 CryptoConfig.MapNameToOID("SHA1"), signature);
            Console.WriteLine("Correct signature validated OK: {0}", validSignature);

            // Change one byte of the signature
            signature[0] = 99;
            // Now try the using the incorrect signature to validate the message
            bool invalidSignature = decryptProvider.VerifyHash(hash,
                                    CryptoConfig.MapNameToOID("SHA1"), signature);
            Console.WriteLine("Incorrect signature validated OK: {0}",
                              invalidSignature);

            Console.ReadKey();


        }
    }
}
```

Implement the System.Security namespace

Data integrity by hashing data

- You “hash” a large lump of data to create a, hopefully small, lump of data that is distinctive for the large lump.A hashing algorithm will weight the values in the source data according to their positions in the data, so that these kinds of changes result in different hash codes being calculated.
- 所有的C#对象都有`GetHash()`方法，计算哈希值基于对象在内存中的位置，可以override这个方法。

```
static void showHash(object source)
{
    Console.WriteLine("Hash for {0} is: {1:X}", source, source.GetHashCode());
}
```

```
Hash for Hello world is: 9D031388
Hash for world Hello is: FC27011E
Hash for Hemmm world is: 1446FEC7
```

- 自带的`GetHash()`可以用来hash value of an object for indexing and searching，但是用来cryptographic applications you need a hashing algorithm that produces a larger hash value.
- 常见哈希算法：
  - MD5: produces a hash code that is 16 bytes (128) bits in size. It has been shown that it is possible to create different documents that both have the same MD5 hash code, 所以不安全了
  - SHA1:160bits，还是不安全
  - **SHA2: 224,256,384,512 bits,** 安全
    -

```
using System;
using System.Security.Cryptography;
using System.Text;

namespace LISTING_3_23_SHA2
{
    class Program
    {
        static byte[] calculateHash(string source)
        {
            // This will convert our input string into bytes and back
            ASCIIEncoding converter = new ASCIIEncoding();
            byte[] sourceBytes = converter.GetBytes(source);

            HashAlgorithm hasher = SHA256.Create();
            byte[] hash = hasher.ComputeHash(sourceBytes);
            return hash;
        }

        static void showHash(string source)
        {
            Console.Write("Hash for {0} is: ", source);

            byte[] hash = calculateHash(source);

            foreach (byte b in hash)
                Console.Write("{0:X} ", b);
            Console.WriteLine();
        }

        static void Main(string[] args)
        {
            showHash("Hello world");
            showHash("world Hello");
            showHash("Hemmm world");

            Console.ReadKey();
        }
    }
}
```

    - 输出：

```
Hash for Hello world is: 64 EC 88 CA 0 B2 68 E5 BA 1A 35 67 8
 12 F4 F3 66 B2 47 72 32 53 4A 8A EC A3 7F 3C Hash for world
 70 A EE E8 36 FD A3 6 9D A4 D1 E1 6D 1B 4C 3C 60 D0 55 C4 A2
 C9 A9 D0 D6 71 B2 Hash for Hemmm world is: A1 FB 33 49 44 B
 72 A5 2B 8D 87 63 54 10 92 DE D6 EA 28 4C A9 78 48 4E 5E D9
```

- **Encrypting streams**：有一个例子

Skill 3.3: Manage assemblies

- You can regard an assembly as the output of a Visual Studio project. A Visual Studio solution can contain multiple projects.

- the contents of an assembly file are independent of the language that was used to create it. (语言无关性，可以是C# VB F#)，产生的MSIL也不是直接运行的，对应不同的平台把MSIL编译成不同的机器码执行(平台无关)

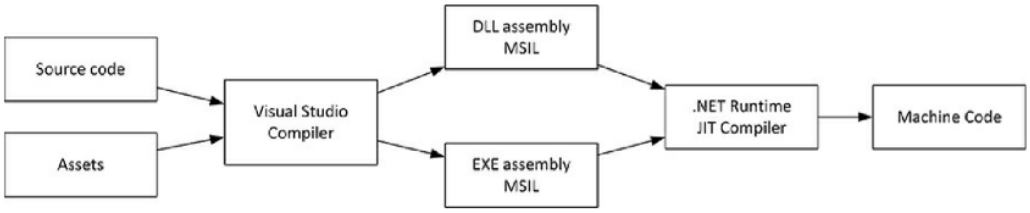


FIGURE 3-17 Assemblies in .NET

Version assemblies

- AssemblyInfo.cs 在.NET Core中变为MyProject.AssemblyInfo.cs

Sign assemblies using strong names

> However, strong-names are still required in applications in some rare situations, most of which are called out on this page: Strong-Named Assemblies.
>
> how to sign assembly

Strong-named assemblies are useful in the following scenarios:

- You want to enable your assemblies to be referenced by strong-named assemblies, or you want to give `friend` access to your assemblies from other strong-named assemblies.
- An app needs access to different versions of the same assembly. This means you need different versions of an assembly to load side by side in the same app domain without conflict. For example, if different extensions of an API exist in assemblies that have the same simple name, strong-naming provides a unique identity for each version of the assembly.
- You do not want to negatively affect performance of apps using your assembly, so you want the assembly to be domain neutral. This requires strong-naming because a domain-neutral assembly must be installed in the global assembly cache.
- You want to centralize servicing for your app by applying publisher policy, which means the assembly must be installed in the global assembly cache.

Skill 3.4: Debug an application

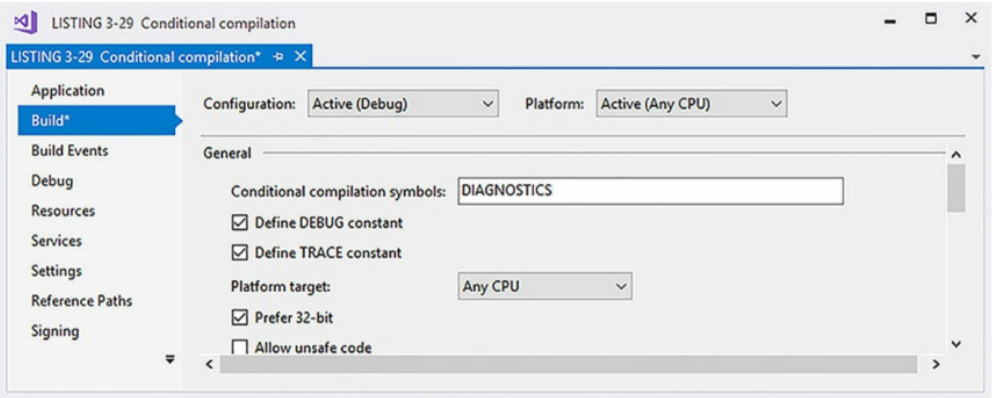
preprocessor compiler directives

- `#if`, `#define`

```
#define DIAGNOSTICS

using System;

namespace LISTING_3_29_Conditional_compilation
{
    public class MusicTrack
    {
        public static bool DebugMode = false;

        public string Artist { get; set; }
        public string Title { get; set; }
        public int Length { get; set; }

        // ToString that overrides the behavior in the base class
        public override string ToString()
        {
            return Artist + " " + Title + " " + Length.ToString() + " seconds long";
        }

        public MusicTrack(string artist, string title, int length)
        {
            Artist = artist;
            Title = title;
            Length = length;
#if DIAGNOSTICS
                Console.WriteLine("Music track created: {0}", this.ToString());
#endif
        }
    }
}
```

- Conditional compilation symbols can also be defined in the properties of an application.

  > 现在有一个Symble `DIAGNOSTICS`定义了，如果要定义多个用semicolon分隔



- `#else`,`#elif`,`#endif`

```
#if TERSE
            Console.WriteLine("Hello");
#elif NORMAL
            Console.WriteLine("Hello Rob");
#elif CHATTY
            Console.WriteLine("Hello Rob. And how are you");
#endif
```

- You can also use the `#undef` directive to undefine any symbols that might have been defined outside your program source.
- Skill 2.5的`Conditional` Attribute

```
//#undef DEBUG

using System;
using System.Diagnostics;

namespace LISTING_3_30_Conditional_attribute
{
    class Program
    {
        [Conditional("DEBUG")]
        static void display(string message)
        {
            Console.WriteLine(message);
        }
        static void Main(string[] args)
```

- `Obsolete` directive

> 第一个参数是Message, 第二个参数是是否产生warning

```
[Obsolete ("This method is obsolete. Call  NewMethod instead.", false)]
public string OldMethod()
```

- `#warning` and `#error`

`#warning`产生warning, `#error` 产生编译错误阻止程序继续执行。

```
#warning This version of the library is no longer maintained.
```

```
#if DIAGNOSTICS && DEBUG
#error Cannot run with both diagnostics and debug enabled
#endif
```

- **pragma directive**
  - 以下的方法本来有个warning因为异步方法没有await，但是现在阻止了这个方法产生warning.

```
#pragma warning disable

        public async Task<IActionResult> Resume()
        {
            return View();
        }
#pragma warning restore
```

  - 只阻止一种类型的warning:

```
#pragma warning disable CS1998
```

- **Identify error positions with `#line`**
  - 重新指定了错误产生的行数：`#line 1 "kapow.ninja"`表示错误在第一行产生，错误的文件是kapow.ninja,
  - `#line default`表示恢复默认

```
using System;

namespace LISTING_3_31_Line_numbers
{
    class Program
    {
        static void Exploder()
        {
#line 1 "kapow.ninja"
            throw new Exception("Bang");
#line default
        }
        static void Main(string[] args)
        {
            try
            {
                Exploder();
            }
            catch (Exception e)
            {
                Console.WriteLine(e.StackTrace);
            }
            Console.ReadKey();
        }
    }
}
```

```
   at LISTING_3_31_Line_numbers.Program.Exploder() in C:\User
Sample-Code\LISTING 3-31 Line numbers\LISTING 3-
31 Line numbers\kapow.ninja:line 1
```

- **Hide code using `#line`**

> debug中step debug的时候可以用来跳过自动产生的代码

```
#line hidden
// The debugger will not step through these statements
Console.WriteLine("You haven't seen me");
#line default
```

- `[DebuggerStepThrough] attribute`

debug单步调试的时候会把整个方法作为一步

```
[DebuggerStepThrough]
public void Update()
{
    ….
}
```

> 如果放在class上表示这个类里的所有方法都不会被`step through`

Choose an appropriate build configuration

> VS自带两种**preset build configurations**: `Debug`和`Release`

Release版本不含有声明了但是没有用到的变量，如果在Release Build里面有断点，会触发**Just My Code** 功能。

> *Just My Code* is a Visual Studio debugging feature that automatically steps over calls to system, framework, and other non-user code. In the **Call Stack** window, Just My Code collapses these calls into **[External Code]** frames.

Release 版本中If a method body is very short the compiler may decide that the program runs faster if the method body was copied “inline” each time the method is called, rather than generating the instructions that manage a method call.

Specify symbol (.pdb) and source files in the Visual Studio debugger

https://docs.microsoft.com/en-us/visualstudio/debugger/specify-symbol-dot-pdb-and-source-files-in-the-visual-studio-debugger?view=vs-2019

Skill 3.5: Implement diagnostics in an application

- `Debug.WriteLine()`, `Debug.WriteLineIf()`输出错误到Output window

  > 如果不是Debug Compiled, 语句不会被输出。

```
using System.Diagnostics;

namespace LISTING_3_33_Debug_code_tracing
{
    class Program
    {
        static void Main(string[] args)
        {
            Debug.WriteLine("Starting the program");
            Debug.Indent();
            Debug.WriteLine("Inside a function");
            Debug.Unindent();
            Debug.WriteLine("Outside a function");
            string customerName = "Rob";
            Debug.WriteLineIf(string.IsNullOrEmpty(customerName), "The name is empty");
        }
    }
}
```

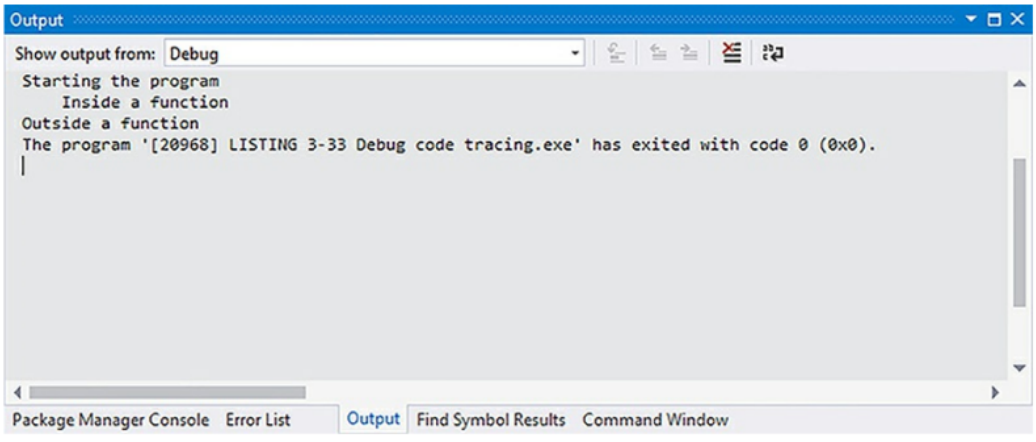


FIGURE 3-42 Debug tracing output

- **Tarce Object**

> 和上面的Debug一样，但是是在Release Build

```
Trace.WriteLine("Starting the program");
Trace.TraceInformation("This is an information message");
Trace.TraceWarning("This is a warning message");
Trace.TraceError("This is an error message");
```

- **Use assertions in Debug and Trace**

```
string customerName = "Rob";
Debug.Assert(!string.IsNullOrWhiteSpace(customerName));

customerName = "";
Debug.Assert(!string.IsNullOrWhiteSpace(customerName));
```

第一个Assert成功，第二个Assert 失败，产生如下窗口

- **Use listeners to gather tracing information**

> 默认情况下Debug和Trace输出到output window, 但是可以指定如下Listener

TABLE 3-1 Types of TraceListener

Class Name	Output
ConsoleTraceListener	Sends the output to the console.
DelimitedTextTraceListener	Sends the output to a TextWriter
EventLogTraceListener	Sends the output to the Event log
EventSchemaTraceListener	Sends the output to an XML encoded file compliant with the Event log schema
TextWriterTraceListener	Sends the output to a given TextWriter
XMLWriterTraceListener	Sends XML formatted output to an XML writer

  - e.g.

```
TraceListener consoleListener = new ConsoleTraceListener();
Trace.Listeners.Add(consoleListener);
Trace.TraceInformation("This is an information message");
Trace.TraceWarning("This is a warning message");
Trace.TraceError("This is an error message");
```

Profiling applications

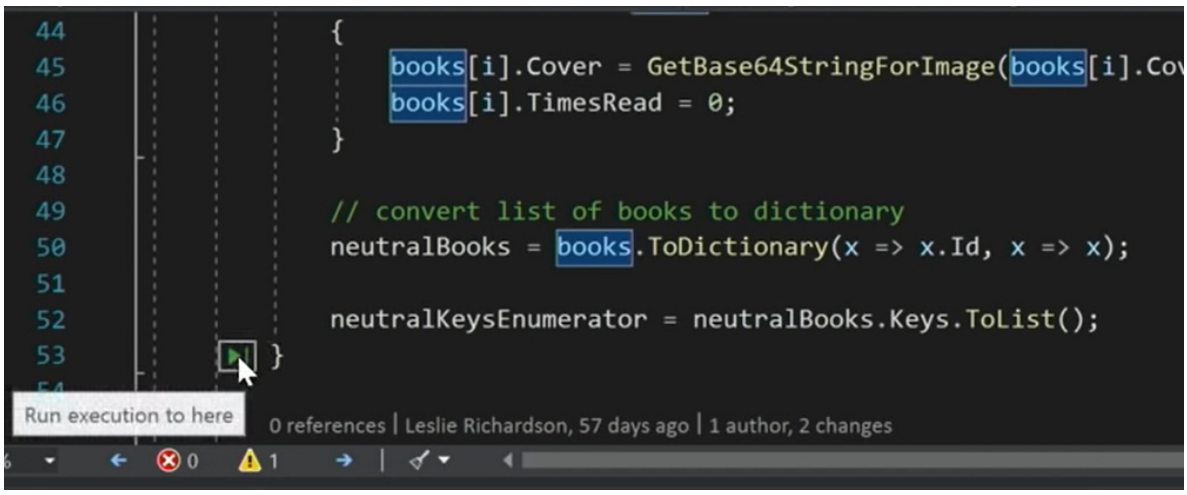
- **The StopWatch class**

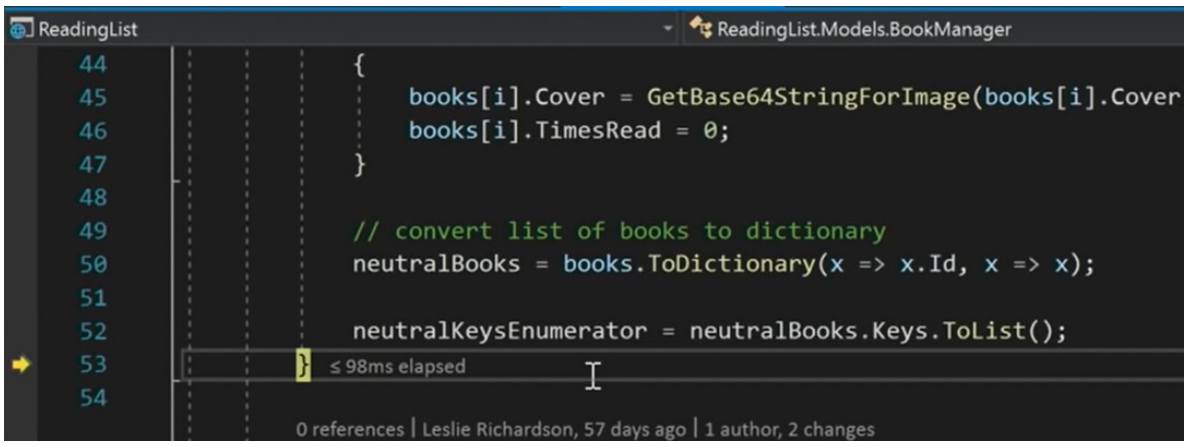
> **The StopWatch class provides `Start`, `Stop`, `Reset`, and `Restart` methods. The `Restart` method resets the stopwatch and starts it running again.**

```
Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();
sequentialTest();
stopwatch.Stop();
Console.WriteLine("Sequential time in milliseconds: {0}",
                  stopwatch.ElapsedMilliseconds);
stopwatch.Restart();
parallelTest();
stopwatch.Stop();
Console.WriteLine("Parallel loop time in milliseconds: {0}",
                   stopwatch.ElapsedMilliseconds);
```

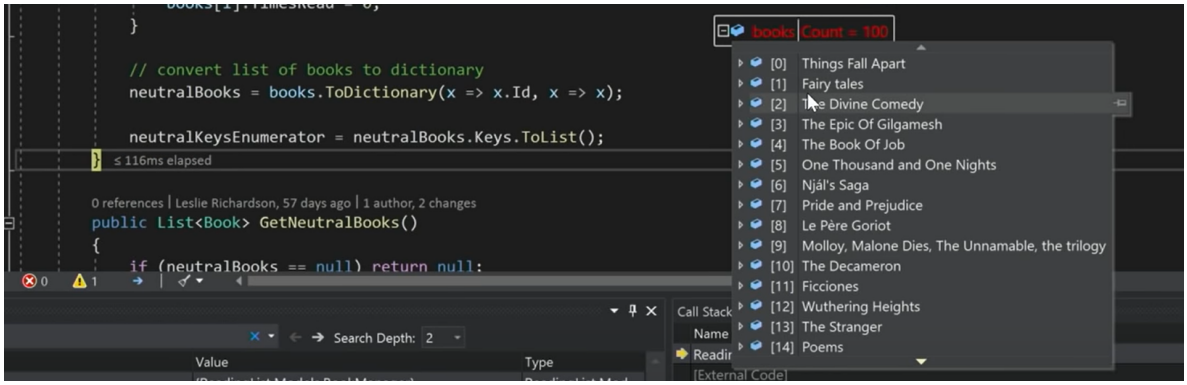
执行到这行

相当于在那行设断点，再继续执行到那行。

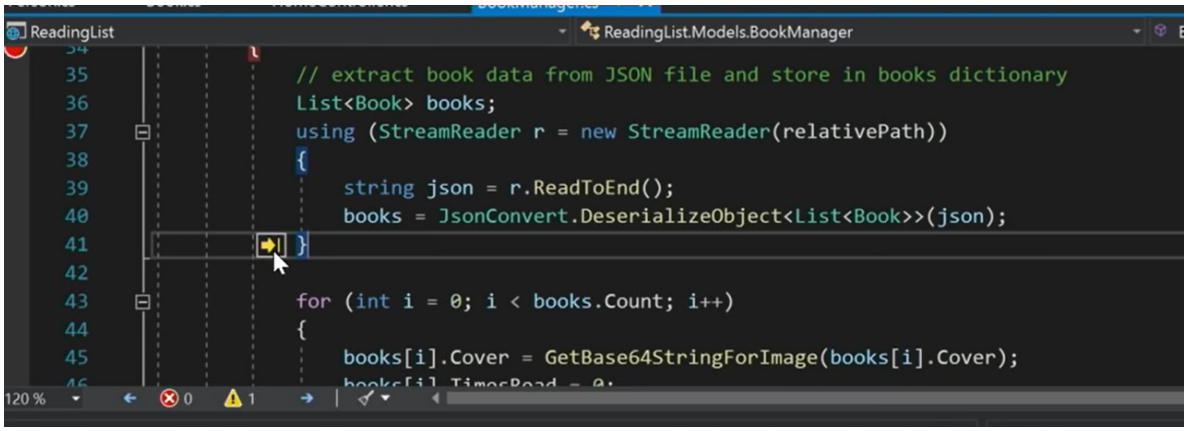




写一个DebuggerDisplay属性，可以选择要显示object的什么Property



下一行要执行的语句由黄箭头表示，如果，可以**向上**拖动小箭头或者在执行过的语句选择如下，实现回过去执行**前面执行过的**语句，而不需要整个重新运行，但是有corrupt风险。



Text Visualier