```
1  /*====================================================================
2  File:      FindType.cs

3  Summary:   Command Line utility to help find and display information
4  on .NET types.

5  Classes:   FindType
6  Indented Writer

7  Origin:    .NET Samples Team
8  ====================================================================*/

9  using System;
10 using System.Reflection;
11 using System.IO;

12 namespace SubMain.Sample {
13   public class App {
14     private const int SHOW_INTERFACES = 0x01;

15     private IndentedWriter myWriter   = new IndentedWriter();
16     private bool exactMatchOnly       = false;
17     private ArrayList DirList         = new ArrayList();

18     // Do an exact match of passed types - the name passed must
19     // be the same as the fully qualified type name.
20     public bool ExactMatchOnly {
21       get {
22         return exactMatchOnly;
23       }
24       set {
25         exactMatchOnly = value;
26       }
27     }

28     public static void Main(string[] args) {
29       FindType ft = new FindType();
30       try {
31         SetOptions(args, ft);
32         ft.Search();
33       }
34       catch(SecurityException ex) {
35         Console.WriteLine("This sample has failed to run due to a security
             limitation!\n"+
36         "Try running the sample from a local drive or using CasPol.exe to turn
             off security.");
37       }
38     }

39     // Processes all of the options specified in the arguments
40     private static void SetOptions(string[] args, FindType ft) {
41       try {
42         char[] backslash = new char[1];
43         backslash[0] = '\\';

44         for( int i = 0; i < args.Length; i++ ) {
```

```
45          string curArg = args[i];

46          if ( curArg[0] == '-' || curArg[0] == '/'  ) {
47            if (Char.ToUpper(curArg[1]) == 'D') {
48              if (curArg.Length > 2 && curArg[2] == ':') {
49                String dir = curArg.Substring(3).TrimEnd(backslash).ToUpper();

50                if (dir != "")
51                  ft.DirAdd(dir);
52                else
53                  Console.WriteLine("Directory not specified");
54              }
55              else
56                Console.WriteLine("Directory not specified");
57              } else {
58              for ( int j = 1; j < curArg.Length; j++ ) {
59                switch( Char.ToUpper(curArg[j]) ) {
60                  case 'A':
61                    ft.ShowAll();
62                    break;
63                  case 'D':
64                    Console.WriteLine("Directory not specified");
65                    break;
66                  default:
67                    Console.WriteLine("Invald Option[{0}]", curArg[j] );

68                    break;
69                }
70              }
71            }
72          } else {
73            ft.ClassAdd( curArg );
74          }
75        }
76      }
77      catch(ReflectionTypeLoadException rcle) {
78        // The following line would output the TypeLoadException.
79        // myWriter.WriteLine("Unable to load a type because {0}",
            exceptions[exceptionCount] );
80        exceptionCount++;
81      }
82      catch(FileNotFoundException fnfe) {
83        myVerboseWriter.WriteLine(fnfe.Message);
84      }
85      catch {
86      }
87    }
```

```
88    // Receive chat from remote client
89    private void ReceiveTalk() {
90      int readMode = 1;

91      while(readMode != 0) {
92        if(reader.Read(oneBuffer, 0, 1)==0) {
93          readMode = 0;
94          continue;
```

```
95          }
96          switch(readMode) {
97            case 1:
98              if(counter == commandBuffer.Length) {
99                readMode = 0;
100               continue;
101             }
102             if(oneBuffer[0] != ':') {
103               commandBuffer[counter++] = oneBuffer[0];
104             } else {
105               counter = Convert.ToInt32(new String(commandBuffer, 1, counter-1)
                    ;
106               if(counter>0) {
107                 readMode = 2;
108                 text.Length = 0;
109               } else if(commandBuffer[0] == 'R') {
110                 prevReceiveText = String.Empty;
111                 Notifications(Notification.ReceivedRefresh, prevReceiveText);
112               }
113             }
114             break;
115           case 2:
116             text.Append(oneBuffer[0]);
117             if(--counter == 0) {
118               switch(commandBuffer[0]) {
119                 case 'R':
120                   prevReceiveText = text.ToString();
121                   Notifications(Notification.ReceivedRefresh, prevReceiveText);
                      break;
123               default:
124                   string newText = text.ToString();
125                   Notifications(Notification.ReceivedAppend, newText);
                      break;
127               }
128               readMode = 1;
129             }
130             break;
131           default:
132             readMode = 0;
133             continue;
134         }
135       }
136       foreach (FileInfo f in dir.GetFiles(Path.GetFileName(switchValue))) {
137         pathnames.Add(f.FullName);
138       }
139     }
140   }
141 }
```

```
1  using System.Reflection;
2  using System.Runtime.CompilerServices;

3  //
4  // General Information about an assembly is controlled through the following
5  // set of attributes. Change these attribute values to modify the information
6  // associated with an assembly.
7  //

8  [assembly: AssemblyTitle("SampleProject.CS")]
9  [assembly: AssemblyDescription("SampleProject.CS")]
10 [assembly: AssemblyConfiguration("")]
11 [assembly: AssemblyCompany("Sub Main(), a Division of vbCity.com, LLC")]
12 [assembly: AssemblyProduct("SampleProject.CS")]
13 [assembly: AssemblyCopyright("vbCity.com, LLC")]
14 [assembly: AssemblyTrademark("vbCity.com, LLC")]
15 [assembly: AssemblyCulture("")]

16 //
17 // Version information for an assembly consists of the following four values:
18 //
19 //      Major Version
20 //      Minor Version
21 //      Build Number
22 //      Revision
23 //
24 // You can specify all the values or you can default the Revision and Build
     Numbers
25 // by using the '*' as shown below:

26 [assembly: AssemblyVersion("1.0.*")]

27 //
28 // In order to sign your assembly you must specify a key to use. Refer to the
29 // Microsoft .NET Framework documentation for more information on assembly
     signing.
30 //
31 // Use the attributes below to control which key is used for signing.
32 //
33 // Notes:
34 //   (*) If no key is specified, the assembly is not signed.
35 //   (*) KeyName refers to a key that has been installed in the Crypto Service
36 //       Provider (CSP) on your machine. KeyFile refers to a file which contains
37 //       a key.
38 //   (*) If the KeyFile and the KeyName values are both specified, the
39 //       following processing occurs:
40 //       (1) If the KeyName can be found in the CSP, that key is used.
41 //       (2) If the KeyName does not exist and the KeyFile does exist, the key
42 //           in the KeyFile is installed into the CSP and used.
43 //   (*) In order to create a KeyFile, you can use the sn.exe (Strong Name)
     utility.
44 //       When specifying the KeyFile, the location of the KeyFile should be
45 //       relative to the project output directory which is
46 //       %Project Directory%\obj\<configuration>. For example, if your KeyFile
     is
47 //       located in the project directory, you would specify the AssemblyKeyFil
```

```
48  //       attribute as [assembly: AssemblyKeyFile("..\\..\\mykey.snk")]
49  //   (*) Delay Signing is an advanced option - see the Microsoft .NET Framework
50  //       documentation for more information on this.
51  //
52  [assembly: AssemblyDelaySign(false)]
53  [assembly: AssemblyKeyFile("")]
54  [assembly: AssemblyKeyName("")]
```

Index

sample.cs
Namespace SubMain.Sample 1
Class App 1
Constant SHOW_INTERFACES 1
Field myWriter 1
Field exactMatchOnly 1
Field DirList 1
Property ExactMatchOnly 1
Method Main 1
Method SetOptions 1
Method ReceiveTalk 2

AssemblyInfo.cs