Chapter 1

Page 10

Replace figure 1.2 with the following:

Page 16, Para 5

Change "Table 1.2" to "Table 1.3".

Chapter 2

Page 44, Code After Para 1

Replace with the following:

```
string logFileName2 = "c:\\Projects\\MyGreatApp\\error.log";
```

Page 45, Code After Last Paragraph

Replace with the following:

```
static void Main(string[] args)
{
    Console.WriteLine("Your option is: {0}", args[0]);

    Console.ReadKey();
}
```

Chapter 3

Page 56, Last Line of Code on Page

Replace with:

```
            isEven = (oddMask & someByte) == 0; // isEven is
false
```

Chapter 4

Page 89, 90

Figures 4.7, 4.8, and 4.9 are out of order.

Figure 4.8 should be Figure 4.7.
Figure 4.9 should be Figure 4.8.
Figure 4.7 should be Figure 4.9.

Page 93, Para 2, Sentence 3

Change "Chapter 3, Writing C# Expressions and Statements" to "Chapter 2, Getting Started with C# and Visual Studio 2008, Table 2.6"

Page 94, Para 6, Sentence 1

Change "Chapter 3" to "Chapter 2"

Page 100, Table 4.2

D (short date), T (short time), F (short time), G (short time), M (left), S, U (first/sortable), Y (left) should be lower case.

Page 101, Table 4.3

All D, S, Y, H (regular hour), M (minutes) should be lower case.

Chapter 5

Page 106, Para 3, Sentence 2

Change "Chapter 1, Introducing the .NET Platform" to "Chapter 2, Getting Started with C# and Visual Studio 2008"

Page 106, Code After Para 3

Replace with the following:

```
 string str2 = "string 2";

string formatString = "{0,15}";

strResult = string.Format(formatString, str2);
```

```
            Console.WriteLine("string.Format({0}, {1}) =
[{2}]\n",
              formatString, str2, strResult);
```

With the following output:

string.Format({0,15}, string 2) = [string 2]

Page 107, Output After Para 3

Replace with the following:

string.Format({0,-15}, string 2) = [string 2   ]

Page 114, Para 8 (last output example)

Replace with:

filePath.LastIndexOf(@"\"): 43

Page 114, Para 9, Sentence 1

Replace sentence with:

This example was another opportunity to show you the verbatim string literal symbol, shown in @"c:\Windows\Microsoft.NET\Framework\v3.5.x.x\csc.exe".

Page 115, Output After Para 6

Replace with:

str1.PadLeft(15): [string 1]

Page 116, Para 7, Sentence 2

Replace "nonwhitespace" with " nonwhitespace" (leaving two spaces to the left of the word)

Page 116, Para 8, Sentence 2

Replace "nonwhitespace " with "nonwhitespace " (leaving two spaces to the right of the word)

Chapter 6

Page 135, Code Before First Para

Replace with the following:

```
int dimOneSize = 2;
int dimTwoSize = 2;

for (int i = 0; i < dimOneSize; i++)
{
    for (int j = 0; j < dimTwoSize; j++)
    {
        Console.WriteLine(
            "exclusiveOr[{0}, {1}]: {2}",
            i, j, exclusiveOr[i, j]);
    }
}
```

Page Code at Bottom of Page 137, Continuing on Page 138

Replace with:

```
for (int i = exclusiveOr.GetLowerBound(0);
         i <= exclusiveOr.GetUpperBound(0); i++)
{
    for (int j = exclusiveOr.GetLowerBound(1);
             j <= exclusiveOr.GetUpperBound(1); j++)
    {
        Console.WriteLine(
            "exclusiveOr[{0}, {1}]: {2}",
            i, j, exclusiveOr[i, j]);
    }
}
```

Page 138, Code After Para 3

Replace with the following:

```
Array kelvinTemperature = Array.CreateInstance(
    typeof(double),
    new int[] { 100 },
    new int[] { -100 });
```

Page 143, Para 8

Change "GetObject" to "ToObject".

Chapter 7

Page 148, Para 1, Sentence 2

Replace with:

The Mulitiply method multiplies a couple numbers and then returns them.

Chapter 9

Page 198, Para 1, Sentence 1

Append " for value types" to the end of the sentence.

Chapter 12

Page 261, Para 1, Sentence 1

Replace with:

"Listing 12.3 contains the DelegatesAndEvents class. In the Main method of DelegatesAndEvents, you'll see references to MenuItem (Listing 12.2), SiteManager (Listing 12.4), and Menu (Listing 12.5).

Chapter 14

Page 296, Listing 14.3

Change:

```
FinancialAdvisor finad = new FinancialAdvisor();
```

To:

```
IBroker finad = new FinancialAdvisor();
```

Page 297, First Paragraph After Listing 14.3 Output

Add the following as a 3rd sentence:

Because FinanceCompany implements IBroker, you can assign the new instance to a variable of type IBroker; which is finco in Listing 14.3.

Chapter 15

Page 324, 325

Replace the entire section titled "Inheritance and Order of Instantiation with this:

*** Begin Replacement Text ***

Inheritance and Order of Instantiation

When instantiating classes in an inheritance chain, you need a deterministic way to know which classes instantiate first. This section clarifies what you need to know about the order of invocation for constructors and some tricky scenarios you'll want to know about.

There are three scenarios I'll walk through to demonstrate order of instantiation with inheritance: default constructors, parameterized constructors, and controlling which base class constructor is called. The first scenario below demonstrates the sequence of instantiation among classes; Base class constructors execute before derived class constructors:

```
public class Base1
{
    public Base1()
    {
        Console.WriteLine("Called Base1().");
    }
}

public class Derived1 : Base1
{
    public Derived1()
    {
        Console.WriteLine("Called Derived1().");
    }
}

public class InitializationSequence
{
    static void Main()
    {
        Derived1 der1 = new Derived1();
```

```
        Console.WriteLine();

        Console.ReadLine();
    }
}
```

And here's the output:

Called Base1().
Called Derived1().

To understand what is happening above, start with the Main method, where it instantiates Derived1. During instantiation, the default constructor for the class at the top of the inheritance hierarchy, Base1, runs first. Then each constructor from the top down executes. You can see in the output above that the Base1 constructor executed first and then the Derived1 constructor executed. This sequence ensures that your base class is properly instantiated before derived classes that might depend on base class state.

Next, you'll see how parameterized constructors behave. This part is a little tricky because many developers initially think that a derived constructor call will initiate a call on the base class constructor with the same signature, but that isn't true. The real behavior is that the default constructor in the base class is called, as shown below:

```
public class Base2
{
       public Base2()
       {
              Console.WriteLine("Called Base2().");
       }

       public Base2(string someString)
       {
              Console.WriteLine("Called Base2(string
someString).");
       }
}

public class Derived2 : Base2
{
       public Derived2() : this ("some string")
       {
              Console.WriteLine("Called Derived2().");
       }

       public Derived2(string someString)
```

```
        {
                Console.WriteLine("Called Derived2(string
someString).");
        }
}
public class InitializationSequence
{
    static void Main()
    {
        Derived2 der2 = new Derived2();
        Console.WriteLine();

        Console.ReadLine();
    }
}
```

And here's the output:

Called Base2().
Called Derived2(string someString).
Called Derived2().

To follow how the code above works, observe that the default constructor is used in Main to instantiate Derived2. Then, look at how the Derived2 default constructor uses the this keyword to invoke the Derived2(string) constructor. Since Base2 has a matching constructor overload with a string parameter, you might think that Base2(string) would be called, but that assumption would be incorrect. As shown in the output above, the default Base2 constructor is called instead. Then Derived2(string) runs, followed by the Derived2 default constructor.

If you're having trouble wrapping your brain around this one, my recommendation is to set a break point in Main and step through the code. This is an important concept to remember because trying to instantiate a base class that doesn't have a default constructor results in a compiler error. You can see this by commenting out the default constructor in Base2 and trying to compile the code.

Remember, C# will automatically generate a default constructor for you if no other constructors are defined. However, if there are other constructors defined and you haven't defined a default constructor, C# will not generate the default constructor for you.

This leads us to the third scenario, where you might want to have some control over which base class constructor should be invoked. The

following example uses the base keyword in the constructor signature to ensure that the parameterized base class constructor is called:

```
public class Base3
{
    public Base3(string someString)
    {
        Console.WriteLine("Called Base3(string someString).");
    }
}

public class Derived3 : Base3
{
    public Derived3()
        : this("some string")
    {
        Console.WriteLine("Called Derived3().");
    }

    // must call single parameter base constructor
    // because there isn't a default constructor
    // comment out call to base to see compile-time error.
    public Derived3(string someString)
        : base(someString)
    {
        Console.WriteLine("Called Derived3(string someString).");
    }
}

public class InitializationSequence
{
    static void Main()
    {
        Derived3 der3 = new Derived3();
        Console.WriteLine();

        Console.ReadLine();
    }
}
```

And here's the output:

```
Called Base3(string someString).
Called Derived3(string someString).
Called Derived3().
```

The Base3 class above doesn't have a default constructor, eliminating the default instantiation sequence as a possibility in this case. To fix this problem (avoid the compiler error), use the base keyword, with a

parameter list that matches the signature of a base class constructor that you want to call instead. In the example above, Derived3(string) uses the base keyword to pass the someString constructor parameter as an argument to the Base3(string) constructor.

A classic example of when to use the base keyword is when creating a custom exception. In Chapter 11, “Error and Exception Handling”, Listing 11-6 and supporting discussion demonstrates how the TooManyItemsException constructor calls the base class constructor that takes a string parameter. For your convenience, here’s a snippet of the code:

```
public class TooManyItemsException : ApplicationException
{
    public TooManyItemsException() : base(@"
        **TooManyItemsException**  You added too many items
        to this container.  Try specifying a smaller number
        or increasing the container size.")
    {
    }
}
```

The default constructor above uses the base keyword to call the ApplicationException(string) constructor. This was a simple example of what you could do to ensure the proper base class constructor executes, but when designing your own class, you’ll want to add as many constructors as necessary and ensure they invoke base class constructors in a way that makes sense for your program.

This section concentrates on instance fields, but what about static fields? Static fields shouldn't be accessed within instance constructors because that could possibly create redundant code to execute every time a new instance is created or leave a class in an inconsistent state when subsequent static methods, if any, are invoked. A good rule of thumb is to keep static field initialization in static constructors and instance field initialization in instance constructors. The next section covers static constructors.

*** End Replacement Text ***

Chapter 17

Page 376, Paragraph 1, Sentence 1

Replace with:

If you scan the definitions of each member i Listing 17.2, you'll see a commonality where the object type is represented by the T parameter.

Chapter 18

Page 398, Code after 2nd Paragraph

Replace with:

([parameter list]) => [expression];

Chapter 20

Page 448, Paragraph 4

Replace with:

To compile the code above, ensure the connection string appears on a single line and include a using directive for the System.Data.SqlClient namespace.

Chapter 21

Page 462, Paragraph 3, Sentence 1 and 3

Change XMLTextWriter to XmlTextWriter

Chapter 22

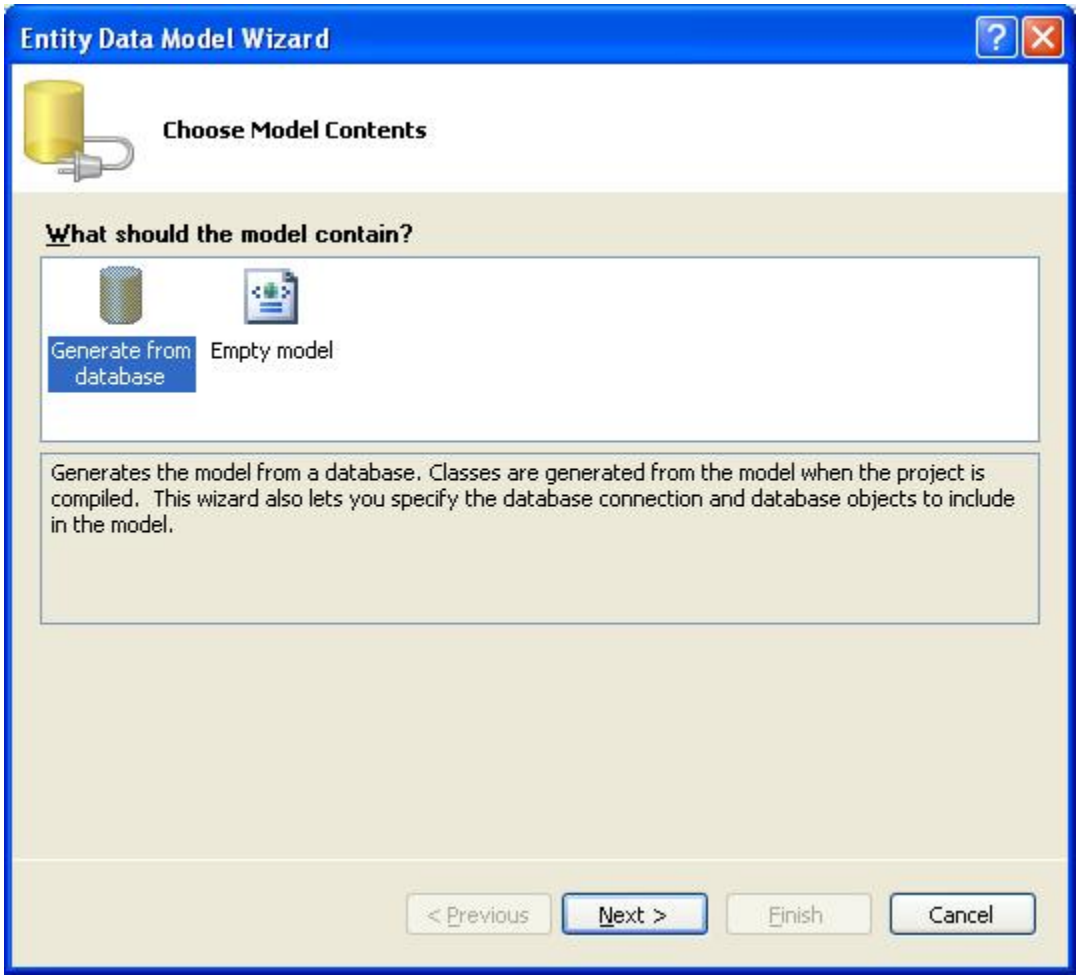
Page 476, Para 6, Sentence 2

Replace with:

In this chapter, we use the same Hospital database as used in Chapter 19, "Accessing Data with LINQ."

Page 477, Figure 22.2

Replace with:



Page 485, Steps 4.b and 4d

Remove both steps and renumber step 4c as 4b.

Page 485, Steps 6.b and 6d

Remove both steps and renumber step 6c as 6b.

Page 485, Step 9

Delete

Page 485, Step 10

Change CustomHospital.Target to CustomHospital.Store

Page 485, Step 8 and 10

Switch Steps - Step 8 becomes Step 10 and Step 10 becomes Step 8.

Page 485, Step 12

Change ?? to <empty string>.

Page 486, Code Snippet after Para 1

Replace with:

```
        using (var custHospital = new CustomHospitalContainer())
        {
            ObjectQuery<Doctor> doctorQuery =
                new ObjectQuery<Doctor>(
                    "select value Doc from
CustomHospitalContainer.Doctors as Doc",
                    custHospital);

            foreach (var doc in doctorQuery)
            {
                Console.WriteLine("Doctor: {0}", doc.Name);
            }
        }
```

Page 486, Para 2

Replace with:

This is the same type of query you saw in the previous section, except that the ObjectQuery type parameter is Doctor, the ObjectContext is named CustomHospitalContainer, and the select statement accesses the fully qualified name CustomHospitalContainer.Doctors.

Pages 486, 487, and 488 Code Examples

Change CustomHospitalEntites to CustomHospitalContainer.

Page 488, Last Code Example

Change "90210" to "90210-0000".

Chapter 23

From Beta to RTM Changes to Make Throughout Chapter (including code)

Change WebDataService to DataService
Change IWebDataServiceConfiguration to IDataServiceConfiguration
Change SetResourceContainerAccessRule to SetEntitySetAccessRule
Change ResourceContainerRights to EntitySetRights
Change Microsoft.Data.WebClient to System.Data.Services.Client.dll

You'll notice that the XML ATOM format from query results (when you view source in IE7) has changed, but the same instructions and concepts apply.

Page 492, Para 1

Change HospitalDataService.svx to HospitalDataService.svc.

Page 500, Para 6

Change the following command-line that starts with Webdatagen.exe to:

datasvcutil.exe /out:HospitalT ypes.cs
/uri:http://localhost:5011/HospitalDataService.svc

Page 501 & 502, Updating Entities Section

Replace code example with this:

```
            var webCtx = new DataServiceContext(
                new
Uri("http://localhost:5011/HospitalDataService.svc"));

            webCtx.MergeOption = MergeOption.AppendOnly;

            var docs = webCtx.Execute<HospitalStaff>(new
Uri("HospitalStaff?$filter=Name eq 'No'&$top=1",
UriKind.Relative));

            var drNo = docs.First();
            drNo.Position = "Nurse";

            webCtx.UpdateObject(drNo);

            webCtx.SaveChanges();

            docs =
webCtx.CreateQuery<HospitalStaff>("/HospitalStaff");

            docs.ToList().ForEach(
```

```
                doc => Console.WriteLine(
                    "Update - Name: {0}, Position: {1}",
                    doc.Name, doc.Position));
```

Page 502, Note on PreRelease Software Glitches

Delete note in its entirety because problem is resolved in RTM.

Page 502, Deleting Entities

Replace code example with this:

```
             var webCtx = new DataServiceContext(
                new
Uri("http://localhost:5011/HospitalDataService.svc"));

            webCtx.MergeOption = MergeOption.AppendOnly;

            var docs = webCtx.Execute<HospitalStaff>(new
Uri("HospitalStaff?$filter=Name eq 'No'&$top=1",
UriKind.Relative));

            docs.ToList().ForEach(
                doc => webCtx.DeleteObject(doc));

            webCtx.SaveChanges();

            docs =
webCtx.CreateQuery<HospitalStaff>("/HospitalStaff");

            docs.ToList().ForEach(
                doc => Console.WriteLine(
                    "Update - Name: {0}, Position: {1}",
                    doc.Name, doc.Position));
```

Page 503, Para 2, Sentence 1

Replace with:

All the examples in this section called CreateQuery or Execute with parameters that match the data service URL parameters.

Page 503, Querying Entities with LINQ to DataService

Replace code example with:

```
            var hospital = new DataServiceContext(
                new
Uri("http://localhost:5011/HospitalDataService.svc"));

            var docs =
                from doc in
hospital.CreateQuery<HospitalStaff>("/HospitalStaff")
                where doc.Position == "Doctor"
                select doc;

            docs.ToList().ForEach(
                doc => Console.WriteLine(
                    "LINQ - Name: {0}", doc.Name));
```

Page 504, Para 1

Change the following command-line that starts with Webdatagen.exe to:

datasvcutil.exe /out:HospitalT ypes.cs /uri:http://localhost:5011/HospitalDataService.svc

Page 504, Para 2

Replace code after the paragraph with the following:

```
            var hospital = new HospitalEntities(
                new
Uri("http://localhost:5011/HospitalDataService.svc"));

            var docs =
                from doc in hospital.HospitalStaff
                where doc.Position == "Doctor"
                select doc;

            docs.ToList().ForEach(
                doc => Console.WriteLine(
                    "LINQ - Name: {0}", doc.Name));
```

Chapter 25

Page 524, Para 2, Sentence 2

Change VS2005 to VS2008.

Chapter 26

Page 552, Para 5, Sentence 1

Change Figure 26.2 to Figure 26.3.

Chapter 27

Page 605, Step 4

Change pr-defined to pre-defined.

Chapter 29

Pages 651, First XAML Code that Starts at Bottom of 650

Replace with:

```
- <UserControl
    xmlns:data="clr-
    namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data" x:Class="Hospital.Page"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Width="400" Height="300">
- <Grid x:Name="LayoutRoot" Background="White">
- <StackPanel>
  <TextBlock
    x:Name="SilverlightMessage"
    Text="SilverlightControl"
    MouseEnter="SilverlightMessage_MouseEnter"
    MouseLeave="SilverlightMessage_MouseLeave" />
  <Button
    x:Name="LoadButton"
    Content="Load"
    Click="LoadButton_Click" />
  <data:DataGrid
    x:Name="HospitalDataGrid"
    AutoGenerateColumns="True"
    Width="400" Height="200" />
    </StackPanel>
    </Grid>
```

```
</UserControl>
```

Chapter 32

Page 700, Para 4, First Sentence

Delete "first".

Page 701, First Para

Delete entire paragraph.

Chapter 38

Page 807, Step 5, First Sentence

Change Iappointment to IAppointment.

Page 808, Para 1, Second Sentence

Change Iappointment to IAppointment.

Page 809, Para 1, Second Sentence

Change Iappointment to IAppointment

Page 811, Steps 5, 9, 14, and 19

Change StartState to TargetStateName

Chapter 39

Page 824, Running Code in a Thread with Less Code

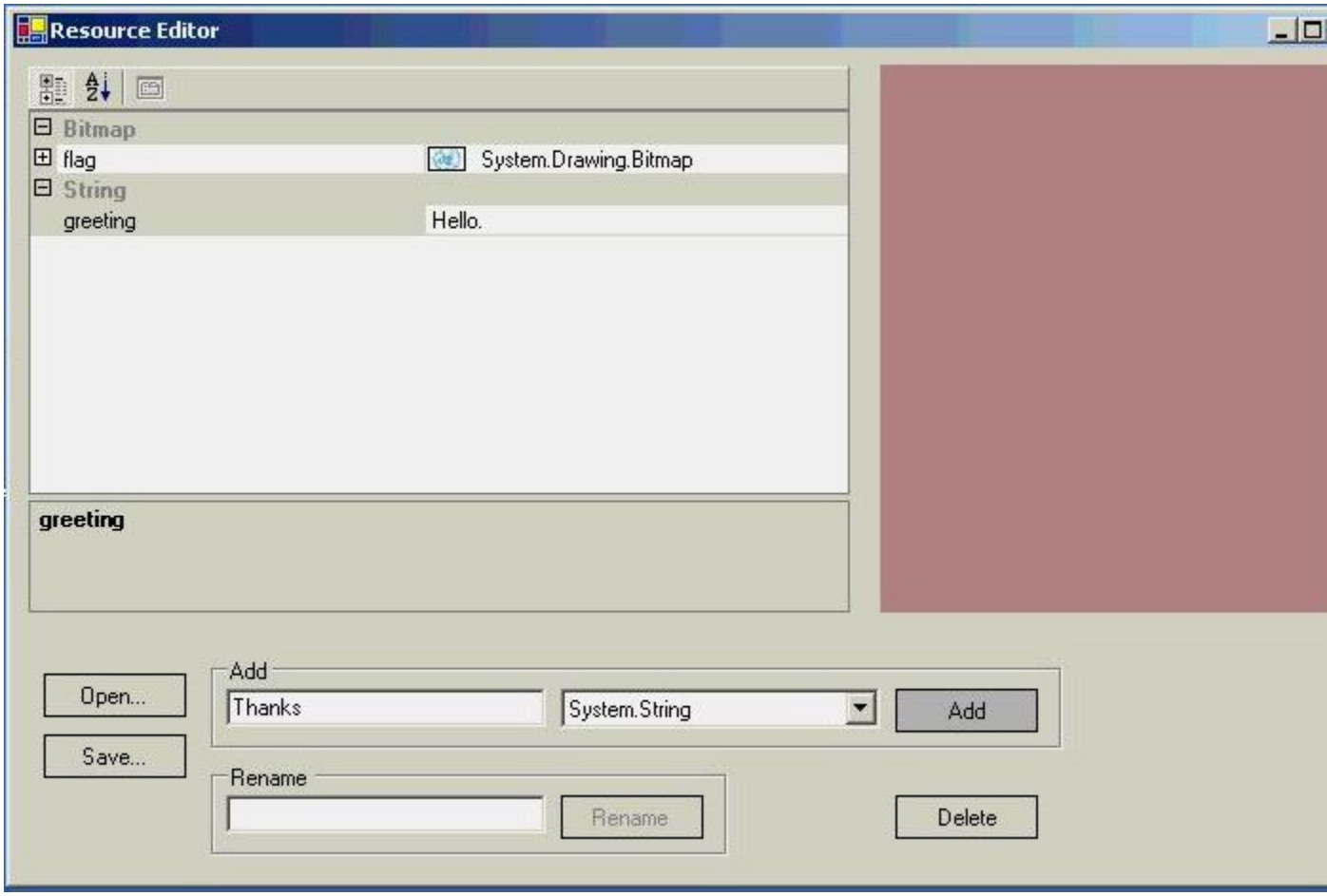
Add the following code after the paragraph of this section:

```
new Thread(
    () => Console.WriteLine(
        "Hello from a single thread via Lambda."))
.Start();
```

Chapter 40

Page 840, Figure 40.1

Replace with:



Chapter 44

Page 938, Table 44.4

Replace with:

Permission	Description
Environment	Read and write environment variables
FileDialog	Read access to a file
FileIO	Append, read, or write to a file
IsolatedStorageFile	Controls access and amount of virtual file system
IsolatedStorage	Control access and amount of generic isolated storage

Principal	Role-based security checks
PublisherIdentity	Access for a software publisher
Reflection	Can use C# reflection
Registry	Access operating system registry
Security	Security permissions that can be invoked
SiteIdentity	Access to software from a specific Web site
StrongName	Access to assembly with a specific strong name
UI	User interface and clipboard
URL	Access to software from a location on the Internet
Zone	Access to specified zones