Chapter 5

Metro

Navigating .NET Framework Documentation

Learning Objectives

- *Use Microsoft Docs to search for C# & .NET Framework documentation*
- *Use Microsoft Developer Network (MSDN) to search for .NET Framework documentation*
- *State the definition of the term "Application Programming Interface" (API)*
- *List and describe the contents of the .NET Framework namespaces used heavily in this book*
- *Demonstrate your ability to navigate a class inheritance hierarchy*
- *Define the term Obsolete API*
- *Explain why you should avoid data types flagged as obsolete in the .NET Framework*
- *Use Microsoft's TechNet website to obtain technical information about Microsoft products*
- *Use Google to speed up your search for .NET Framework documentation*
- *List additional resources available to assist you when you get stuck*

Introduction

When programming in C#, or any .NET programming language, a lot of your work is already done for you in the form of the .NET Framework class library. The .NET Framework class library supplies interfaces, classes, structures, delegates, and enumerations that serve as the foundation upon which all .NET applications are built, and provides advanced functionality to help you create feature-rich applications.

The .NET Framework class library, also referred to as the .NET Application Programming Interface (.NET API), is both a blessing and a curse. It's a blessing because just about every conceivable thing you would want to do in your programs, from creating and managing collections of objects to writing complex client-server networked database applications, is available for immediate use in the form of preexisting interfaces, classes, and other data types. It's a curse in that you must expend considerable effort in learning how to use these data types in your programs. Just discovering and remembering what is available to you requires a fair amount of time, and each new framework release introduces new data types, improves upon or deprecates existing data types.

The purpose of this chapter is to help you jump start your mastery of the .NET API by showing you how to look up information using Microsoft Docs and the Microsoft Developer Network (MSDN) websites. The successful and quick navigation of the .NET API reference material is a fundamental programming skill employed everyday by programmers around the world.

In fact, you will spend much more time learning how to use the .NET API than you will learning C# language fundamentals. This is due largely to the shear number of data types the API contains and partly because the API continuously evolves. But don't panic. You can get started programming complex, professional-looking C# applications with the help of only a small handful of classes, interfaces, and delegates.

MSDN And Microsoft Docs

The Microsoft Developer Network (MSDN) used to be the place to find detailed information about the .NET Framework Application Programming Interface (API), but the Microsoft documentation landscape is changing. In 2016 Microsoft launched Microsoft Docs citing the aging MSDN technology stack which made it difficult to update and publish. Still, when you search for information about the .NET API and other Microsoft technologies, you will interact with both MSDN and Microsoft Docs, at least for the foreseeable future. Let's start by using Microsoft Docs to look up information on the System.String class.

Launch the Microsoft Docs website by navigating to https://docs.microsoft.com. The homepage will look similar to figure 5-1.

Referring to figure 5-1 — Click on the .NET block. This will take you to the .NET documentation page located at https://docs.microsoft.com/en-us/dotnet. This page will look similar to figure 5-2..

Referring to figure 5-2 — This page contains links to a ton of great information, which can be overwhelming to those just starting out on their programming odyssey. (Programmers don't take journeys, we embark on odysseys!) For now, let's focus on the .NET Framework. Click on the .NET Framework API Reference link. This will take you to the .NET API Browser, the API reference page for the latest version of the .NET Framework, which at the time I write this is version 4.7. The page will look similar to figure 5-3.

Referring to figure 5-3 — Scroll down the page. What you see listed here are all the namespaces included with this version of the .NET Framework. Each namespace contains classes, structures, and other data types that you can use to create programs.To select a previous version of the .NET Framework, return to the top of the page and select the desired version from the dropdown list. On this site you can only go back as far as version 4.5.

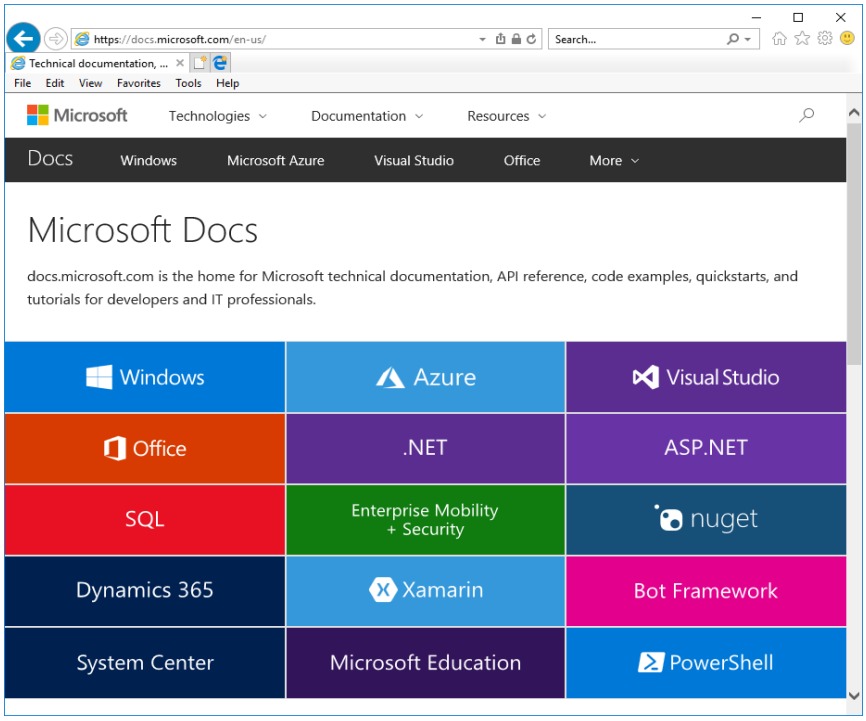

Figure 5-1: Microsoft Docs Homepage

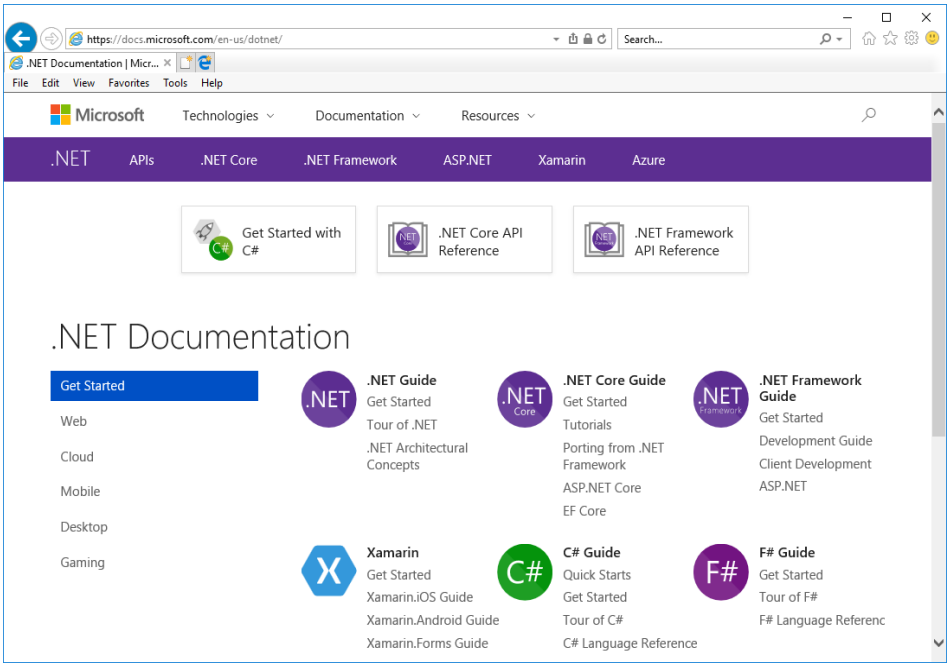

Figure 5-2: .NET Documentation Page

As you scroll down you'll notice the namespace names are listed in the left column, and a short description of that namespace appears in the right column. Continue scrolling down the page of namespaces until you find the **System** namespace. When you find the System namespace, click the link to navigate to the System namespace page. It should look similar to figure 5-4.

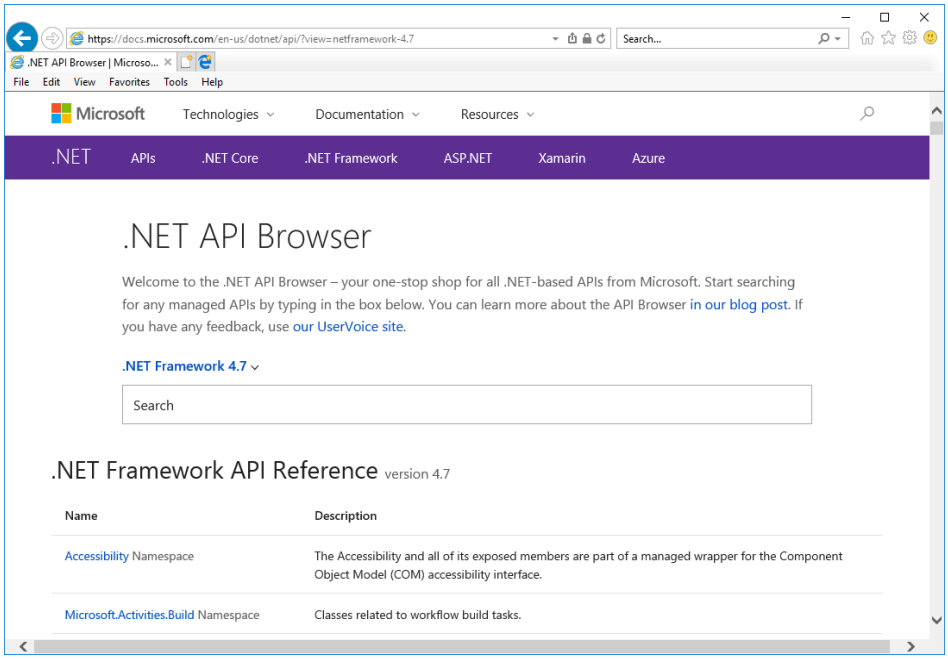

Figure 5-3: .NET API Browser

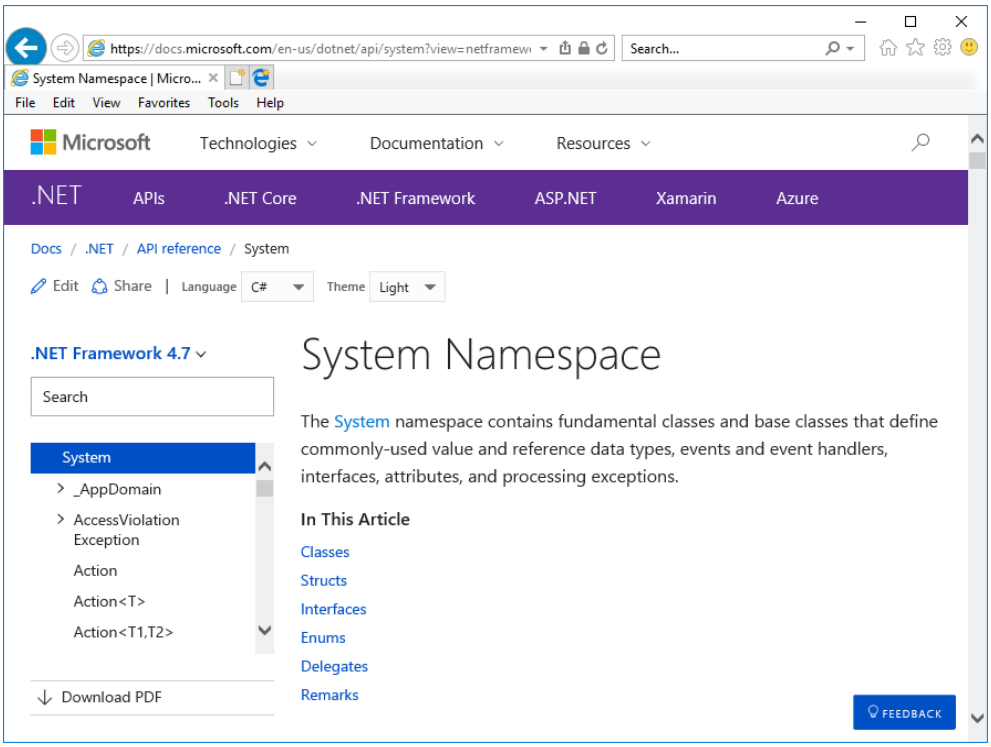

Figure 5-4: System Namespace Page

Referring to figure 5-4 — On this page you'll find links to all the classes, structures, and other data types contained within the System namespace. As you progress through this book you'll spend a lot of time researching the different data types found within the System namespace, and you'll use a lot of them in your programs. The next section shows you how to get information about the String class, which is located

in the System namespace. Knowing how to read the String class page will enable you to get valuable information about all the other data types in the .NET Framework.

Quick Review

Microsoft Docs is becoming Microsoft's central repository for all things documentation. Microsoft Docs will eventually replace MSDN and TechNet but all three sites contain great information about C# and .NET Framework development.

Discovering Information About Classes

If you look closely at figure 5-4 you'll see in the right-hand frame under System Namespace a short paragraph describing its contents. Below that you'll see the title **In This Article**. Below it are several links: Classes, Structs, Interfaces, Enums, Delegates, and Remarks. As you explore the System namespace, you'll quickly realize it contains a lot of stuff. Indeed, many of the data types found in the System namespace belong to the Base Class Library (BCL).

To get a feel for how to navigate API documentation, let's take a look at the String class. Click the **Classes** link to reveal all the System namespace classes. Scroll down until you find the String class. Click the String class link to open a window that looks similar to the one shown in figure 5-5.

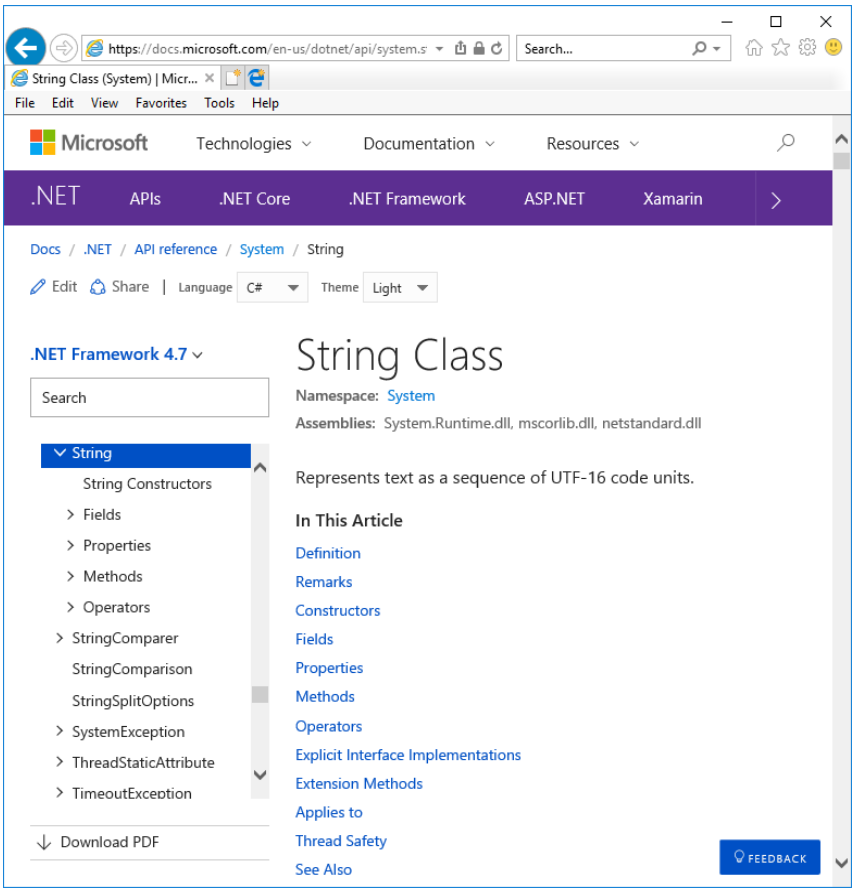

Figure 5-5: String Class Page

Referring to figure 5-5 — What you are looking at here is the String class overview page. There's a lot of great information on the overview page. First you'll notice the class name at the top of the page, followed by the namespace in which it resides, and the assemblies that contain the class definition. This is followed by a short description of the class. Under the title **In This Article**, you'll see links to other sections on the page. You'll want to scroll down the page as I discuss each section below.

The **Definition** section, shown in figure 5-6, shows the class declaration, its inheritance hierarchy, attributes applied to the class, and interfaces the class implements.

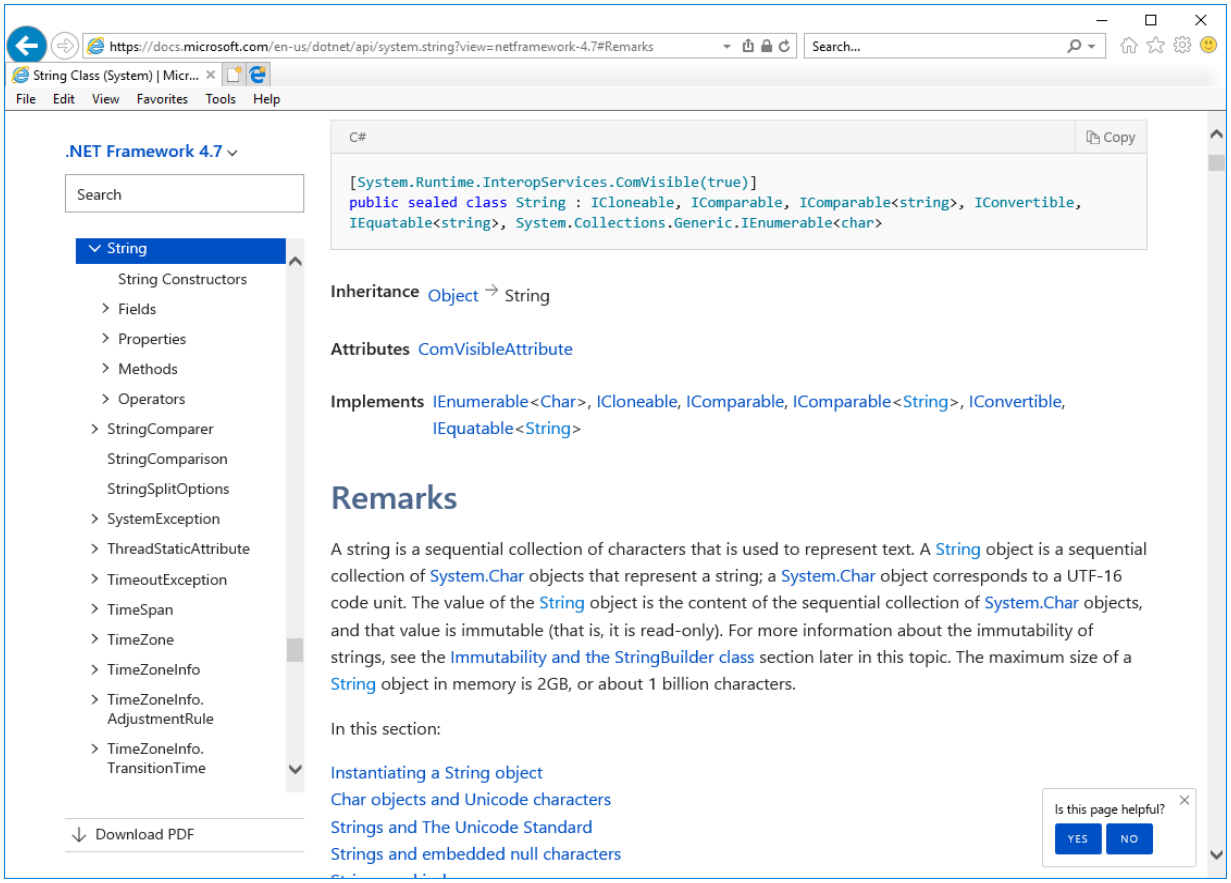

Figure 5-6: String Class Definition and Remarks Sections

Referring to figure 5-6 — This information is handy to have when you're trying to figure out what a class can do. At the top of the definition section under C# is the String class declaration. It might look like gibberish to you now, but with time you'll come to appreciate this section as one of the most valuable ones on the page. By reading the class declaration, you can determine at a glance that the String class is ComVisible, descends directly from System.Object, and implements a boatload of interfaces including ICloneable, IComparable, IComparable<string>, and IEquatable<string>, to name a few. To really get a feel for what the String class is capable of, you should look up each of the interfaces it implements and read up on what functionality they provide.

The **Inheritance** subsection shows you what classes this particular class extends or inherits. As you can see, the String class extends the Object class. The **Attributes** subsection lists any attributes applied to this class. Attributes grant special powers to data types by tagging the type with additional metadata. You'll learn more about attributes as you progress through the book. The **Implements** subsection lists the interfaces the String class implements. An interface specifies what an object can do, but not how to actually do it. You will become an interface ninja by the time you finish this book. All these subsections essen-

tially repeat what's communicated via the class declaration. When you learn how to read and understand a .NET Framework type declaration, you'll have the keys to the kingdom.

The **Remarks** section is super helpful. It contains information on how to use the class along with detailed code examples, and discusses issues you should be aware of when using the class in your code.

The **Thread Safety** section, not shown in figure 5-6, offers advice on using the class in multithreaded programs. Sometimes this can be a very terse statement like "This type is thread safe." You'll learn more about thread safety in chapter 16.

The **See Also** section, also not shown in figure 5-6, provides links to other elements within the .NET Framework that share a relationship with this data type. For example, here you would find links to the numerous interfaces implemented by the String class.

Class Member Documentation

A class contains members. These members can include constructor methods (constructors), fields, properties, methods, extension methods, and operators, just to name a few. For example, to see how many different ways there are to create an instance of a String, click on the Constructors link found under the **In This Article** section. The **Constructors** section shows you the list of constructor methods the class implements and notes on how to use them. Figure 5-7 shows the String class's Constructors section.

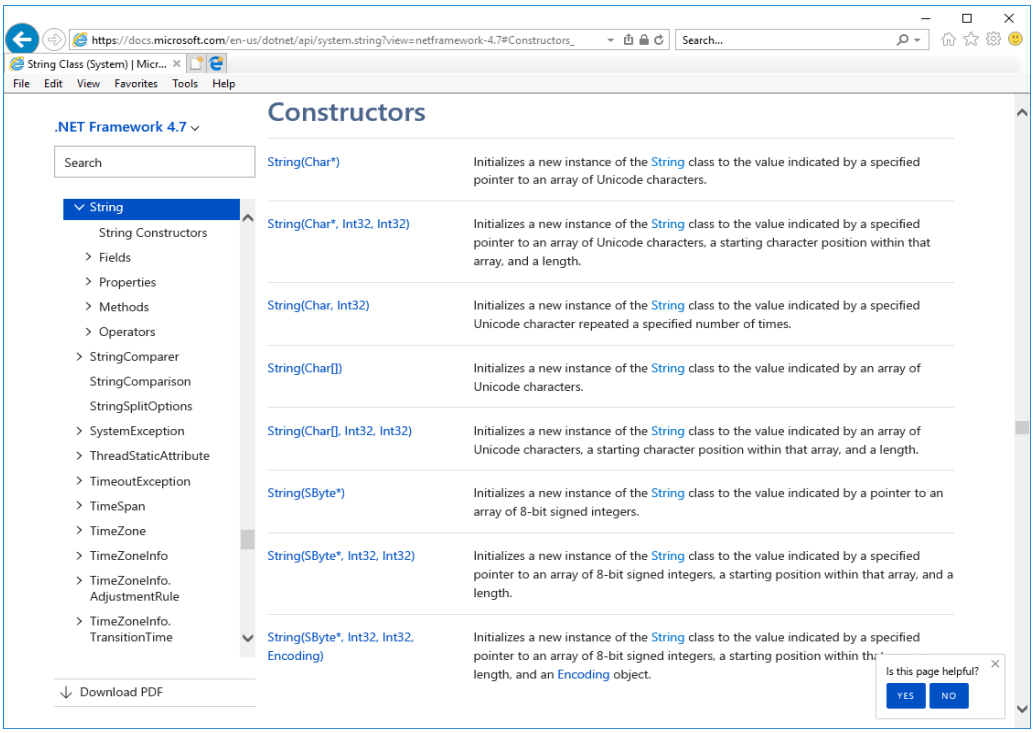

Figure 5-7: String Class's Constructors Section Partial Listing

Referring to figure 5-7 — To get more information about a particular constructor, click its link.

The **Fields** section lists any public fields the class makes available for use. The String class has only one field: Empty. The Empty field represents an empty string.

The **Properties** section lists any public or protected properties the class may have. The same hold for the **Methods** section. A partial listing of the String class's Methods section is shown in figure 5-8.

Referring to figure 5-8 — Click a particular method's link to get more information. For example, scroll down the String Methods section, find the SubString(Int32, Int32) method, and click its link. This will take you to the String.SubString(Int32, Int32) method details page, as is shown in figure 5-9.

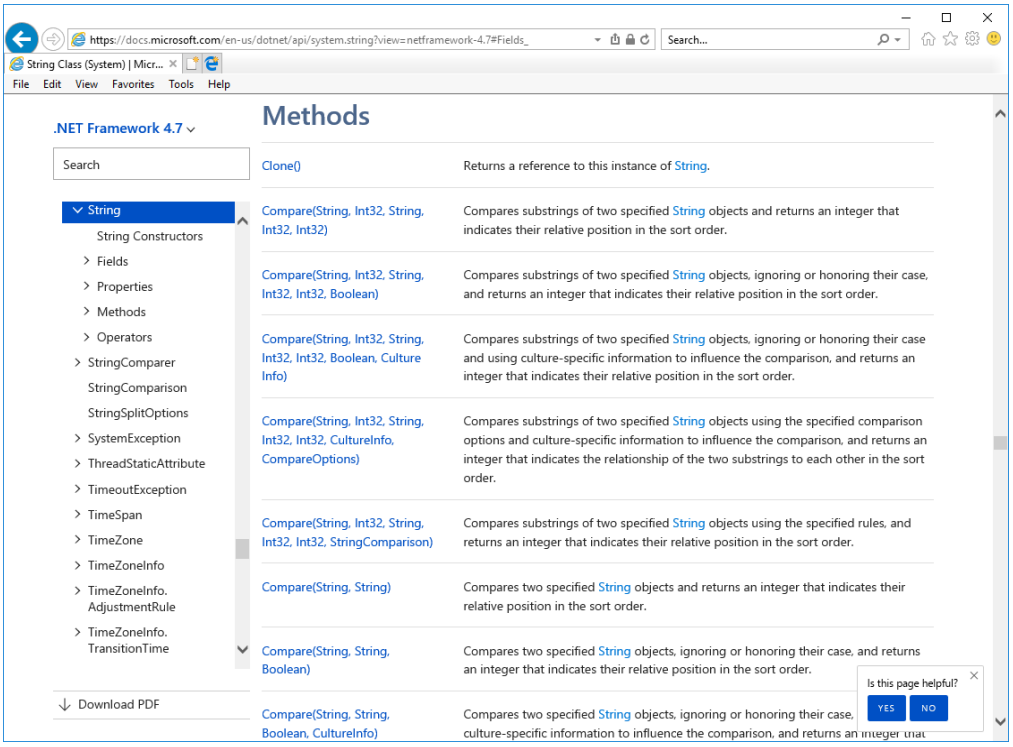

Figure 5-8: String Class's Methods Section Partial Listing

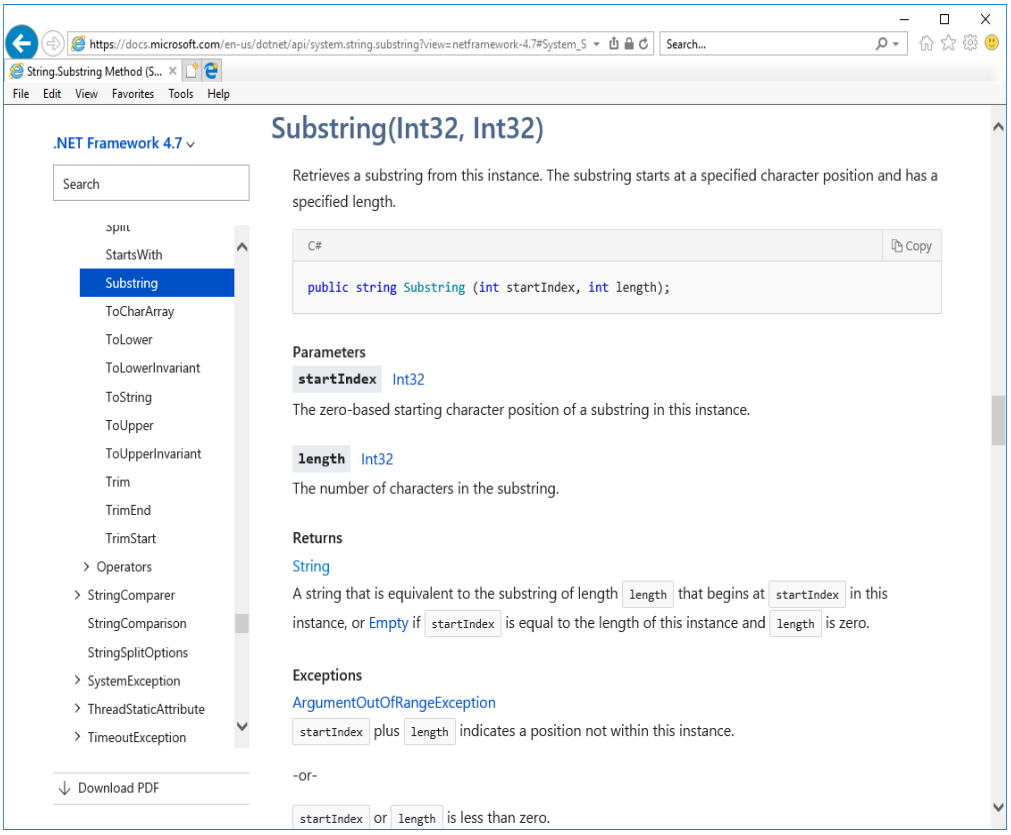

Figure 5-9: String.SubString(Int32, Int32) Method Page

Referring to figure 5-9 — Like the String class page, you'll find a Definition section for the SubString(Int32, Int32) method. You can see at a glance that the first parameter represents the startIndex, and the second parameter represents the length. The **Returns** subsection provides an explanation of what is returned when the method is called. You'll learn more about method parameters and return types in chapter 9.

Scroll down further to find the **Examples** section. You'll find this part of the page most interesting. It provides several examples demonstrating the use of the method in a program. Figure 5-10 shows a partial listing of the String.Substring(Int32, Int32) method's Examples section.

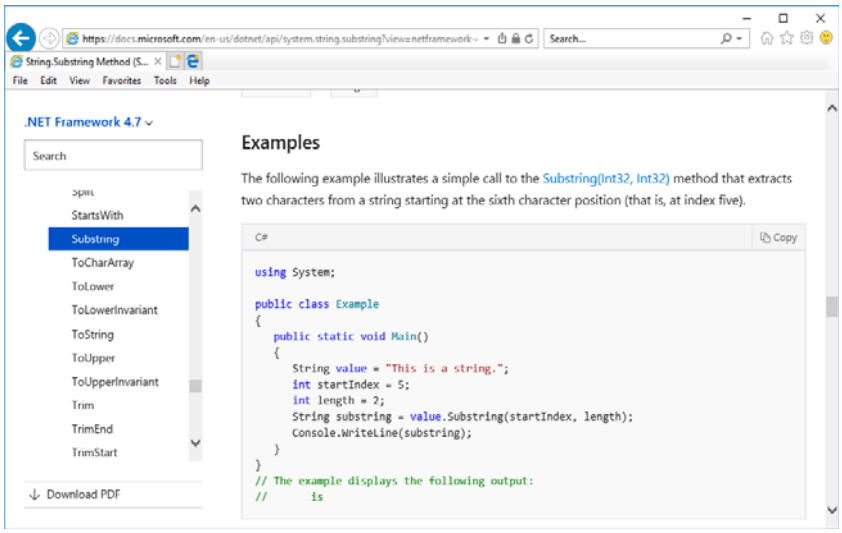

Figure 5-10: String.SubString Examples Section Expanded Showing Example Code

Getting Information On Other Class Members

Additional information on a data types' fields, properties, and other members can be obtained in the same way as information on methods. Just follow the links. At this time you will find it helpful to simply explore the .NET Framework documentation. Browse the System namespace and its many sub-namespaces to get a feel for the data types it contains. Don't be put off if you don't completely understand what you're looking at. The important thing to do is to simply get familiar with .NET Framework documentation. Doing so will pay huge dividends in the very near future.

Quick Review

Use Microsoft Docs to find useful information about the .NET Framework API. Microsoft Docs is slowly replacing MSDN as Microsoft's central documentation repository.

Namespaces Used Heavily In This Book

This book will draw heavily from the namespaces listed in table 5-1.

Namespace	Description
System	This is a fundamental namespace that includes all value type structures, the String class, math functionality, the DateTime class, and much, much more.
System.Collections.Generic	Provides many different types of generic collection classes.
System.Data System.Data.SQL	Provides the capability to write applications that access a relational database using ADO.NET and Structured Query Language (SQL)
System.Drawing	Provides graphics drawing and manipulation classes.
System.IO	Supports file, console, serial port, and interprocess input and output.
System.Linq	Provides classes and interfaces that support Language-Integrated Query (LINQ).
System.Net System.Net.Sockets	Provides classes and interfaces used to write networked applications.
System.Runtime.Remoting	Provides classes and interfaces necessary to write distributed applications that call methods on remote objects.
System.Runtime.Serialization	Supports the serialization and deserialization of objects.
System.Runtime.Serialization.Formatters.Binary	Supports the binary serialization and deserialization of objects.
System.Text System.Text.RegularExpressions	Contain types that support character encoding and string manipulation.
System.Threading	Provides multithreaded application support.
System.Windows.Forms System.Windows.Forms.Layout	Provides a large collection of classes and interfaces used to create Windows Graphical User Interface (GUI) applications.

Table 5-1: .NET Framework Namespaces Used Heavily In This Book

Referring to table 5-1 — As you explore the .NET Framework documentation pay particular attention to the namespaces listed above, but don't let this table limit your curiosity.

Navigating An Inheritance Hierarchy

In this section, I want to show you how to navigate an inheritance hierarchy. I'll continue to use the String class as an example. Figure 5-11 shows a Unified Modeling Language (UML) diagram for the String class's inheritance hierarchy.

Referring to figure 5-11 — The UML diagram shown is referred to as a "class" diagram. A class diagram shows static relationships between classes, interfaces and other system artifacts. In this case, the String class inherits from the Object class. How do we know this? By referring to the String class overview page available in the .NET Framework documentation. The inheritance hierarchy is listed in the Inheri-

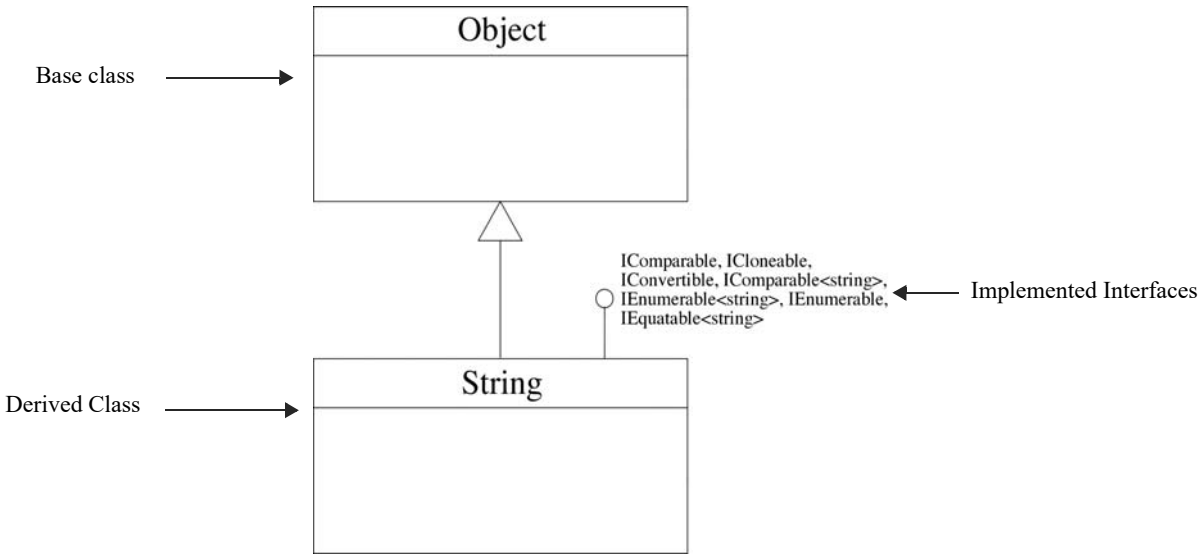

Figure 5-11: String Class Inheritance Hierarchy

tance section as was shown in figure 5-6. Unfortunately, the Inheritance section only gives part of the picture. You need to study the Definition section of the class overview page to learn what interfaces a class implements and any attributes, such as *SerializableAttribute* or simply *Serializable*, it supports. The String class declaration as given in the Definition section of the String class documentation page appears in example 5-1.

5.1 String Class Declaration

```
1   [SerializableAttribute]
2   [ComVisibleAttribute(true)]
3   public sealed class String : IComparable, ICloneable, IConvertible,
4   IEnumerable, IComparable<string>, IEnumerable<char>, IEquatable<string>
```

Referring to example 5.1 — Actually, this code comes from the MSDN version of the String class documentation page. (**Note:** The Microsoft Docs version is incomplete in my opinion because it does not contain the [SerializableAttribute]. The reason, I believe, for the missing attribute is that the String class is defined in the .NET Standard and the .NET Standard does not tag any basic type as being Serializable.)

The String class declaration does not explicitly inherit from Object. Rather, all C# types (*i.e.*, reference types [classes] and value types [structures]) implicitly extend Object, although value type structures implicitly extend System.ValueType, which extends System.Object, and behave differently from reference types.

OK, so what's the use of tracking down the base classes and interfaces of a class? Simple: class behavior is the sum of all behaviors inherited from base classes, from interface implementations, or applied attributes. (*e.g.*, SerializableAttribute is an example of an attribute.) To fully understand all that a particular class can do, you must navigate up the inheritance hierarchy, visit each base class, and in turn visit each interface to learn what powers an implementation of that interface gives to the data type.

In short, a String is an Object. Any methods declared public in the Object class can also be called on a String object. A String object is not only serializable and comvisible, it is also comparable, cloneable, convertible, enumerable, and equatable.

Do not panic if you don't yet understand all the terminology used in this chapter. By chapter 11 it will all start to make sense. However, it would still be a good exercise to visit the String class reference page and follow the links to all its related interfaces. Also, visit the Object class and see what it has to offer.

Quick Review

Trace a data type's inheritance hierarchy to discover its complete range of functionality. Visit each base class, interface, and attribute reference page to learn what functionality they provide to the data type in question. As you gain experience, the need to perform this exercise will gradually diminish to where you would only need to do it for data types with which you are unfamiliar.

Beware Obsolete APIs

As the .NET Framework evolves, there will be times when some of what came before will be rendered obsolete. Though using an obsolete API component does not immediately spell disaster, it's a good idea to avoid using them when possible for the sake of forward compatibility. You can get the most recent list of obsolete or deprecated .NET Framework types from Microsoft Docs as is shown in figure 5-12.

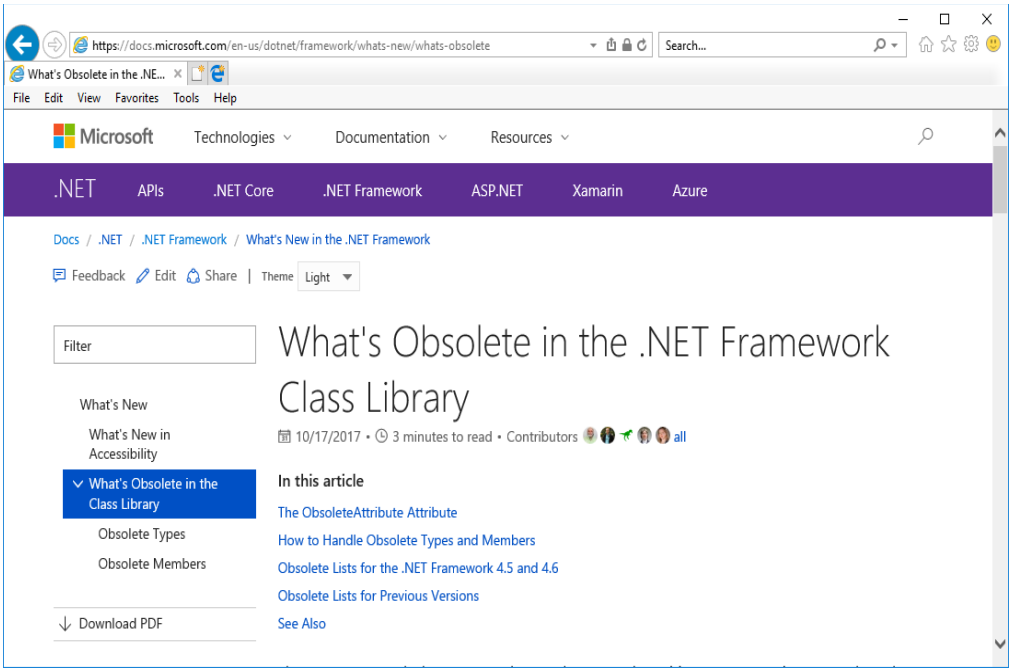

Figure 5-12: Obsolete .NET Framework Version API Page on Microsoft Docs

Referring to figure 5-12 — The Obsolete API page will offer suggestions on how to proceed if you've discovered obsolete API usage in your code. This may affect you, for example, if you're planning to migrate legacy code to a newer version of the .NET Framework.

Quick Review

Beware the usage of obsolete .NET Framework APIs. If you use obsolete API functionality in your program, the compiler will give you a warning. Refer to the Obsolete API page to determine the best course of action in the context of your particular situation.

Google For Faster Access To .NET Framework API Information

OK, having showed you how to go through Microsoft Docs to get .NET Framework API information, I'll now show you THE fastest way to get to some useful stuff, because after all, you want to develop at light speed, and clicking around Microsoft's websites can be a real pain the ass.

Just pull up Google and type in what you're looking for. The results may surprise you. For example, Google "System namespace". Figure 5-13 shows the results I get from this search.

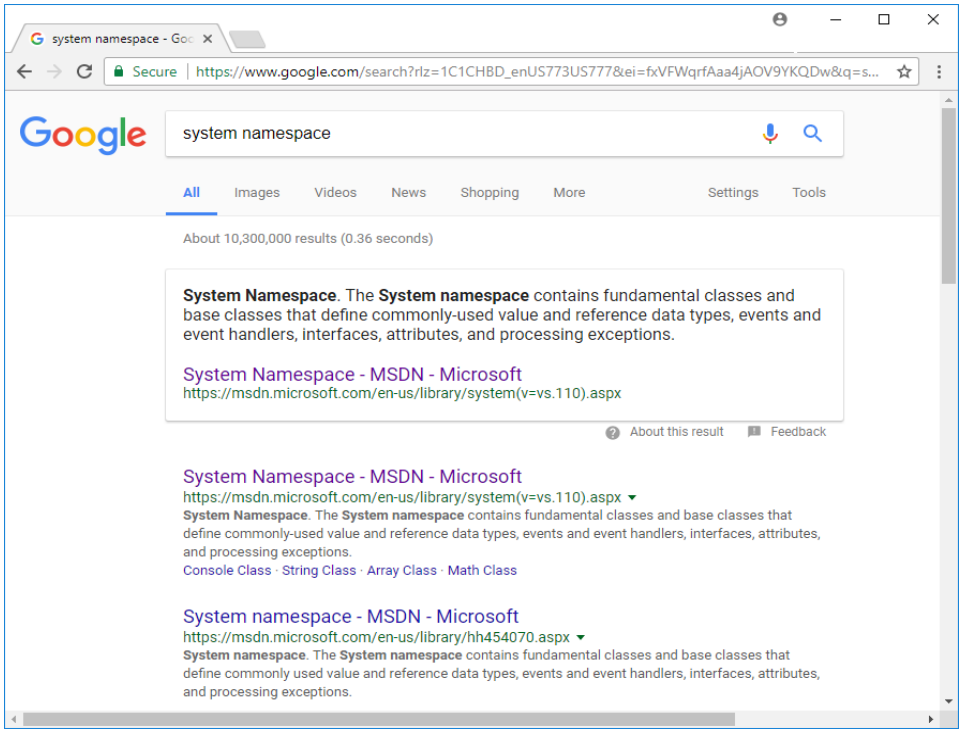

Figure 5-13: Google Results for System Namespace

Referring to figure 5-13 — Note that most of the results point to MSDN. Click the first link at the top of the result stack. This will take you to the System Namespace page on MSDN as is shown in figure 5-14.

Generally speaking, when you search for .NET Framework documentation via Google, the first result that comes back is the one you're looking. Take the time now to explore MSDN's version of the .NET Framework documentation. In a lot of ways, I find the MSDN site to be way more helpful, but I suspect Microsoft Docs will eventually catch up, and perhaps by the time you read this, MSDN docs will no longer be available.

As you gain experience using Google to search for a .NET Framework documentation and answers to your programming questions, your Google-Fu will become quite strong, grasshopper.

What If You Get Stuck?

There will be times when you are just stumped, even when you've worked as a programmer for many years. Usually the best way to get help on an issue is to search for a solution to your immediate problem on Google. There are millions of programmers around the world using C#, and you can rest assured you are not the first to be stumped by your particular problem.

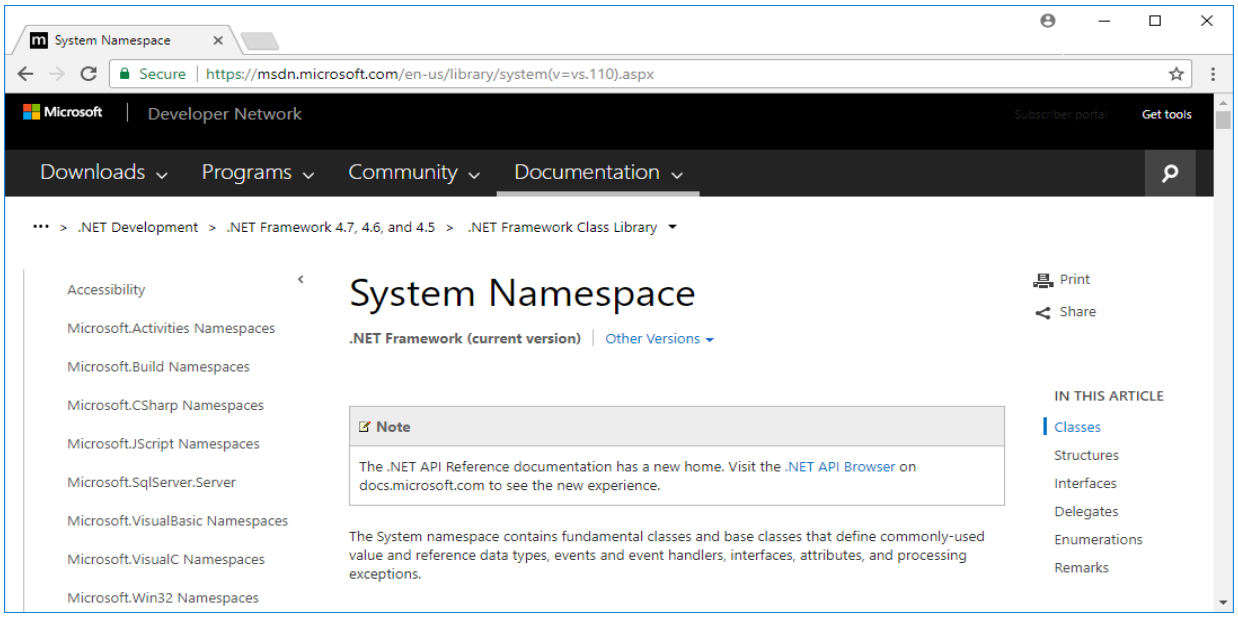

Figure 5-14: System Namespace Page on MSDN

One site I find provides the most helpful solutions to vexing programming problems is StackOverflow.com https://stackoverflow.com. But don't put up a general homework question or they'll shame you straight off that site. Generally, the answer to your question already exists on StackOverflow.com, so you may never need to post a specific question because someone else has already done so.

Summary

Microsoft Docs is becoming Microsoft's central repository for all things documentation. Microsoft Docs will eventually replace MSDN and TechNet but all three sites contain great information about C# and .NET Framework development. You'll spend most of your time on Microsoft Docs in the .NET Framework Class Library section researching the many data types that appear in various namespaces.

Trace a data type's inheritance hierarchy to discover its complete range of functionality. Visit each base class, interface, and attribute reference page to learn what functionality they provide to the data type in question. As you gain experience, the need to perform this exercise will gradually diminish to where you will only need to do it for data types with which you are unfamiliar.

Beware the usage of obsolete .NET Framework APIs. If you use obsolete API functionality in your program, the compiler will give you a warning. Refer to the Obsolete API page to determine the best course of action in the context of your particular situation.

Use Google to quickly find .NET API documentation.

Skill-Building Exercises

1. **API Drill:** Visit the Microsoft Docs homepage and explore the many links provided.

2. **API Drill:** Explore the System namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief descrip-

tion of that functionality using your own words.

3. **API Drill:** Explore the System.Text namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

4. **API Drill:** Explore the System.Collections namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

5. **API Drill:** Explore the System.Collections.Generic namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

6. **API Drill:** Explore the System.IO namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

7. **API Drill:** Explore the System.Net namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

8. **API Drill:** Explore the System.Threading namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

9. **API Drill:** Explore the System.Drawing namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

10. **API Drill:** Explore the System.Windows.Forms namespace. Write down a list of all classes, interfaces, exceptions, events, and delegates you find there. Research the functionality each provides and write a brief description of that functionality using your own words.

Suggested Projects

1. **Navigate Inheritance Hierarchy:** Navigate the inheritance hierarchy for the System.String class. Follow the links for its base class and all implemented interfaces. Write down the functionality provided by each interface. Make a note of any overridden methods found in the class.

2. **Navigate Inheritance Hierarchy:** Navigate the inheritance hierarchy for the System.Int32 structure. Follow the links for its base class and all implemented interfaces. Write down the functionality provided by each interface. Make a note of any overridden methods found in the class.

3. **Navigate Inheritance Hierarchy:** Navigate the inheritance hierarchy for the System.DateTime structure. Follow the links for its base class and all implemented interfaces. Write down the functionality provided by each interface. Make a note of any overridden methods found in the class.

4. **Navigate Inheritance Hierarchy:** Navigate the inheritance hierarchy for the System.Windows.Forms.Button class. Follow the links for its base classes and all implemented interfaces. Write down the functionality provided by each interface. Make a note of any overridden methods found in the class.

5. **Navigate Inheritance Hierarchy:** Navigate the inheritance hierarchy for the System.Convert class. Follow the links for its base class and all implemented interfaces. Write down the functionality provided by each interface. Make a note of any overridden methods found in the class.

Self-Test Questions

1. Where can you find the most recent version of .NET Framework documentation?

2. What types of information can you find on a class overview page?

3. How would you find some example code showing the use of a particular class method?

4. What's the purpose of knowing how to navigate a class inheritance hierarchy?

References

Microsoft Docs website https://docs.microsoft.com

Microsoft Developer Network (MSDN) website https://msdn.microsoft.com

StackOverflow.com https://stackoverflow.com

Microsoft TechNet https://technet.microsoft.com

Notes

