

CHAPTER 5



Contax T3 / Kodak Tri-X

BRANCHES

NAVIGATING .NET FRAMEWORK DOCUMENTATION

LEARNING OBJECTIVES

- *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 BASE CLASS LIBRARIES OF THE .NET FRAMEWORK API*
- *DEMONSTRATE YOUR ABILITY TO NAVIGATE A CLASS INHERITANCE HIERARCHY*

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, and value types 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 class library, also referred to as the .NET Framework application programming interface (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 pre-existing interfaces and classes. It's a curse in that you must expend considerable effort in learning how to use these interfaces and classes in your programs.

The purpose of this chapter is to help you jumpstart your usage of .NET API classes by showing you how to look up API information on Microsoft's MSDN website and navigate class inheritance hierarchies. 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 sheer number of classes 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 API classes.

MSDN: THE DEFINITIVE SOURCE FOR API INFORMATION

The definitive source for .NET API information is the Microsoft Developer Network (MSDN) website [www.msdn.com]. If you don't have an internet connection but did buy Microsoft Visual Studio.NET or get the C#.NET Express Edition CD, then you can access the .NET API reference documentation that came with those products.

On the MSDN website, navigate to the .NET API reference section by following the links .NET Framework -> Class Library Reference. This will bring you to a page that looks similar to that shown in Figure 5-1.

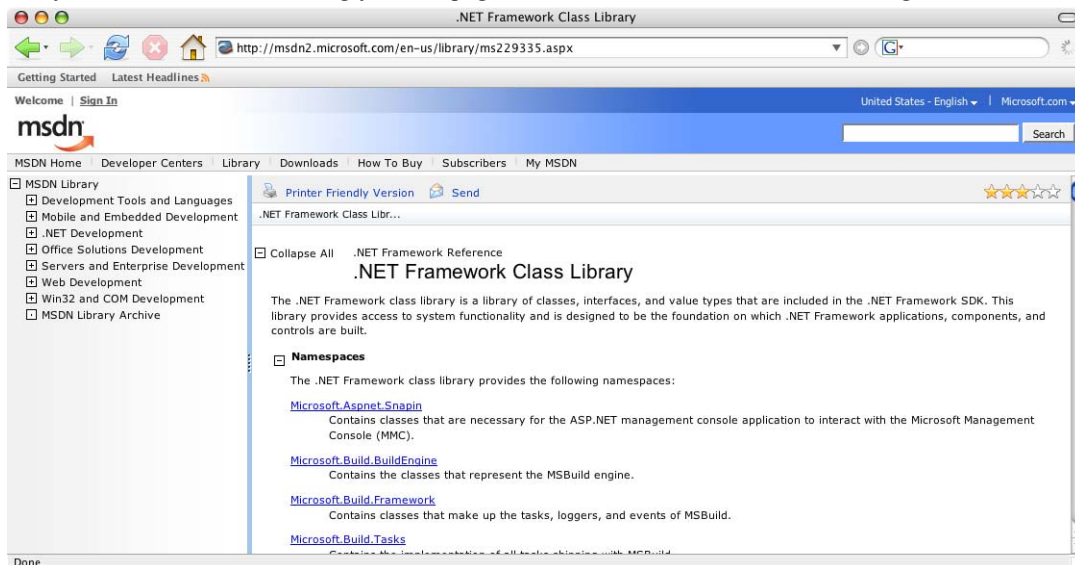


Figure 5-1: .NET Framework Class Library Reference Page

All of the .NET Framework development information you can access from this page can be overwhelming, but for the purposes of this book, you only need concern yourself with a very small part of the API reference. In the left frame, click the .NET Development link to expand it as shown in Figure 5-2.

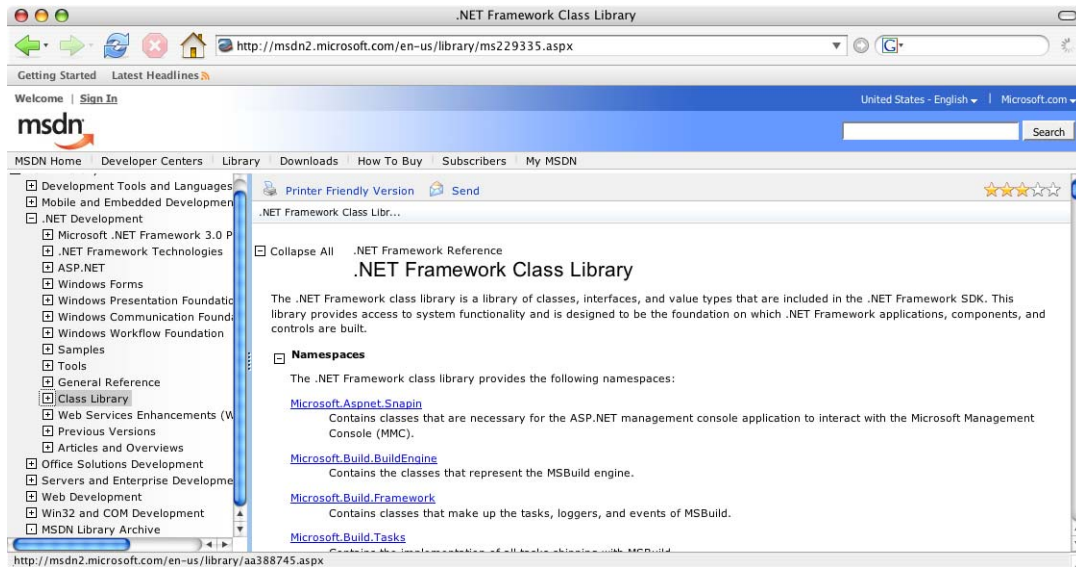


Figure 5-2: .NET Development Link Expanded and Class Library Link Highlighted

Again in the left frame, click the Class Library link to expand it, and scroll down until you find the System namespace as shown in Figure 5-3. It is in the Class Library section of the .NET API reference that you will spend most of your time researching and referencing the value types, interfaces, and classes found in the System namespace and its sub namespaces.

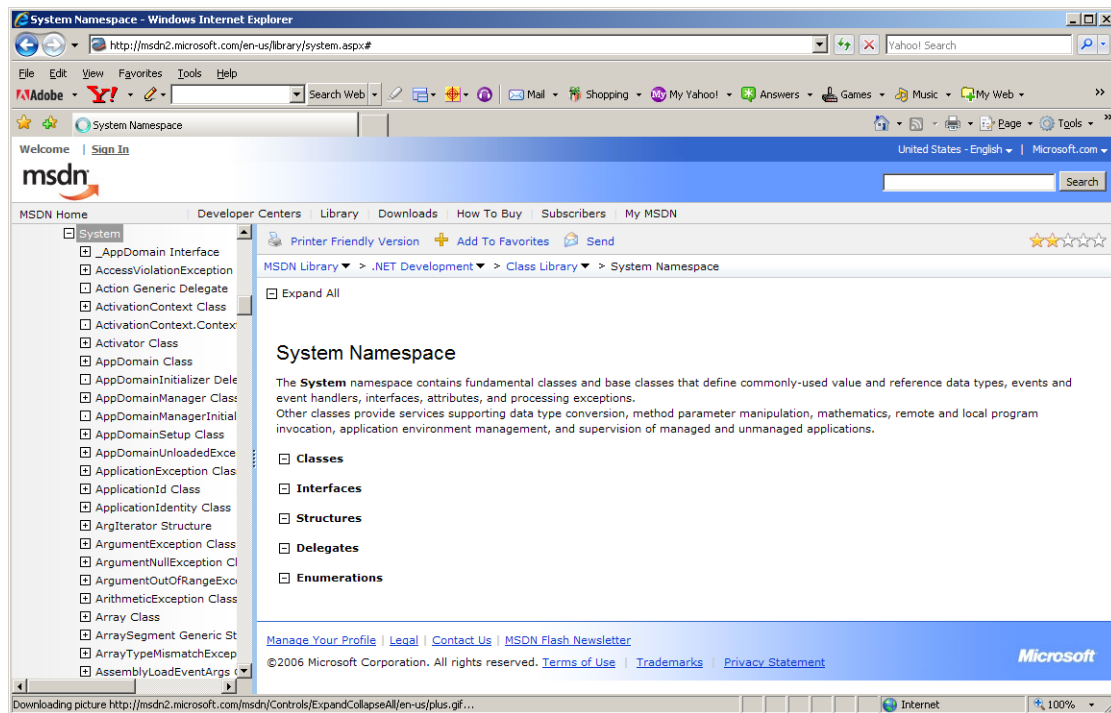


Figure 5-3: Class Library Link Expanded and System Namespace Highlighted

DISCOVERING INFORMATION ABOUT CLASSES

If you look closely at Figure 5-3 you'll see in the right frame under System Namespace a short paragraph describing its contents. Below that you'll see several subheadings (collapsed in the figure). The subheadings include Classes, Interfaces, Structures, Delegates, and Enumerations. The System namespace contains a lot of stuff; indeed, the System namespace is the primary library in the .NET Framework. To get a feel for how to navigate API information, let's take a look at the String class. Click the Classes subheading 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 that shown in Figure 5-4.

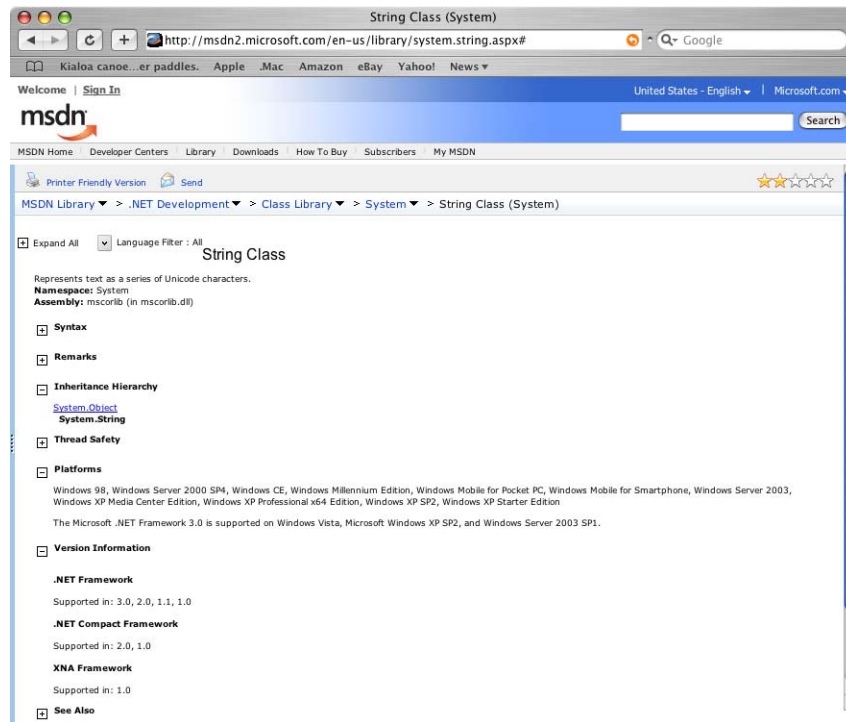


Figure 5-4: String Class API Reference Overview Page

GENERAL OVERVIEW PAGE

Referring to Figure 5-4 — What you are looking at here is the class information overview page. The overview page contains a lot of good general information arranged in subheadings or sections. I have collapsed a few of the subheadings for the purpose of this figure, but note that the API reference pages normally load with all subheadings fully expanded.

The Syntax section shows the class declaration. This is handy when you are trying to navigate a class's inheritance hierarchy, which I will discuss in greater detail later in this chapter.

The Remarks section contains information on how to use the class, and discusses issues you should be aware of when using the class in your code.

The Inheritance Hierarchy section shows you what classes this particular class extends or inherits. As you can see, the String class extends Object. **Note:** The Inheritance Hierarchy section does not show what interfaces the String class implements. For that you need to examine the Syntax section.

The Thread Safety section offers some advice on using the class in multithreaded programs.

The Platforms and Version Information sections show the Microsoft Windows operating system platforms that support this class and the different .NET Framework versions in which it appears. Not all .NET Framework classes, interfaces, etc., are supported on all platforms.

The See Also section provides links to other elements within the .NET Framework that share a relationship with this entry. For example, here you would find links to the numerous interfaces implemented by the String class.

Class Member Page

A class contains members. These members can include constructor methods (constructors), fields, properties, and methods. What you'll see on the members page is information about public members. Public members are those class members that you have access to when you use objects of this type in your code. If a class can be extended, meaning it's not sealed, you'll see its protected members as well. Figure 5-5 shows the String Members page with the subheadings collapsed.

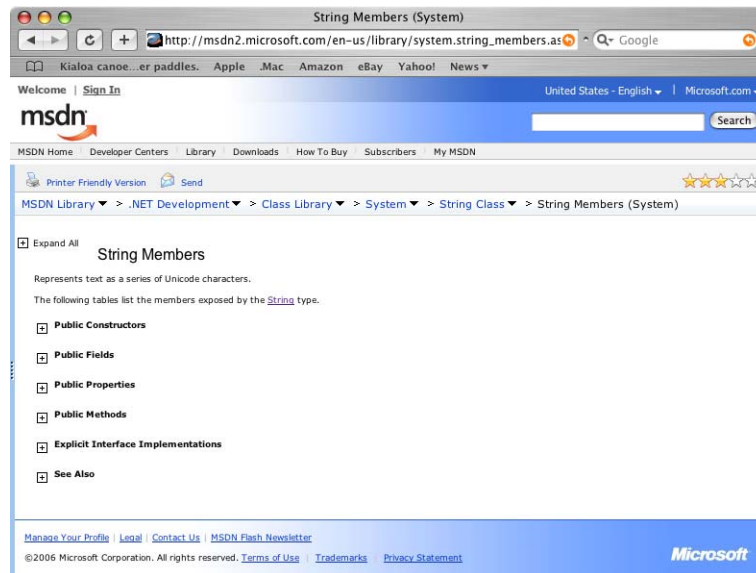


Figure 5-5: String Members Page

The Public Constructors section shows you the list of public constructor methods the class contains and notes on how to use them. Figure 5-6 shows a partial listing of the String class's Public Constructors section.

The Fields section lists any public fields the class makes available for use. The Properties section lists any public properties the class may have. The same hold for the Methods section. A partial listing of the String class's Methods page is shown in Figure 5-7.

Click a particular method's link to get more information. For example, scroll down the String Methods page, find the SubString method, and click its link. This will take you to the String.SubString Method page, as is shown in Figure 5-8.

Referring to Figure 5-8 — notice there are two versions of the SubString method. This means the SubString method has been overloaded to perform a similar operation in two different ways. Click one of the method links to learn more about how to use that particular method version, as Figure 5-9 illustrates.

Referring to Figure 5-9 — notice there are several subheadings on the method details page. The Syntax section shows how the method is declared in several different .NET programming languages. The Exceptions section lists and describes any exceptions the method may throw if something goes wrong when it's called. The Remarks section provides a few words of guidance on the method's use. The Platforms and Version sections contain the same types of information as they do on other API pages.

The Example section is the one part of the page you will be most interested in studying. It provides examples of how to use the method in several different programming languages. Figure 5-10 shows the Examples section expanded to reveal the C# code section.

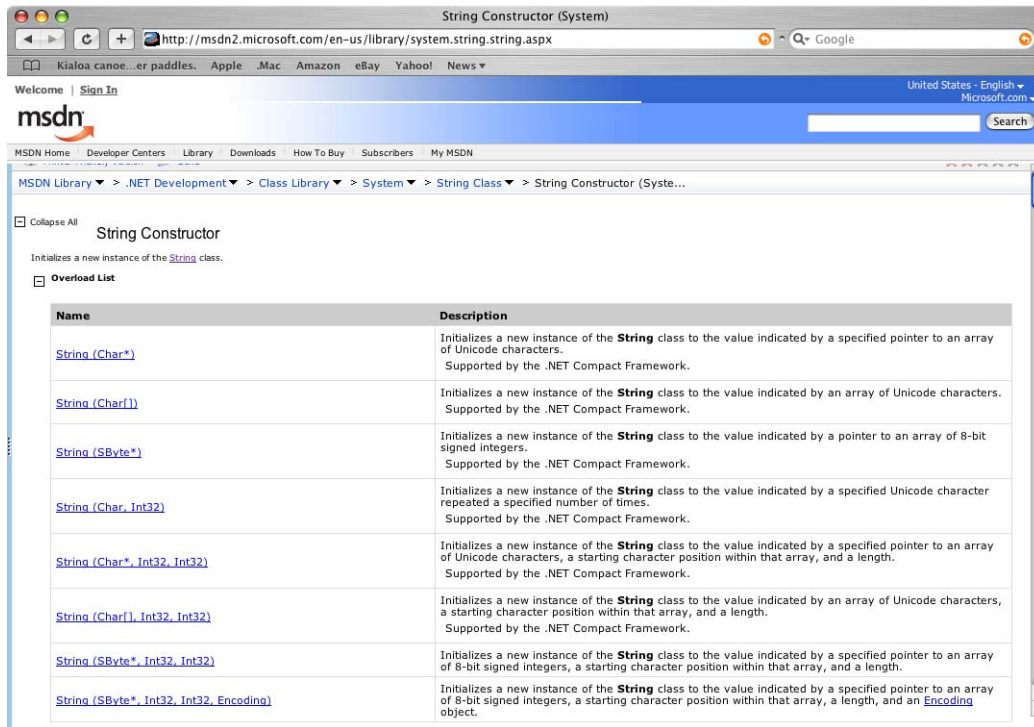


Figure 5-6: String Class's Public Constructors Partial Listing

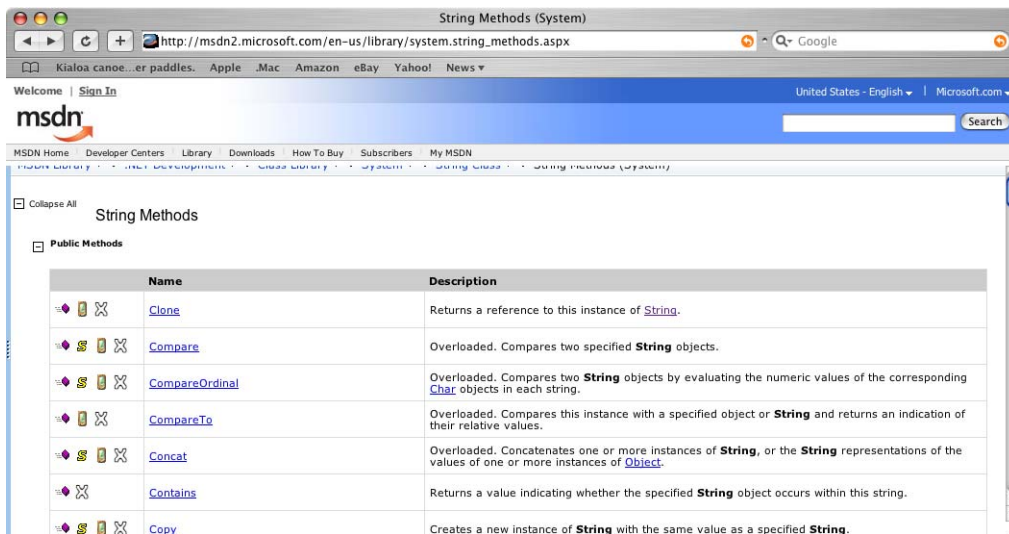


Figure 5-7: String Class's Methods Page Partial Listing

GETTING INFORMATION ON OTHER CLASS MEMBERS

Additional information on class fields, properties, constructors, etc., can be obtained in the same way as information on class methods; Just follow the links. At this time, you may find it extremely helpful just to wander around the .NET Framework documentation. Explore the System namespace and its many sub namespaces and see what types of classes and interfaces they contain. Don't be put off by not understanding what it is you are 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.

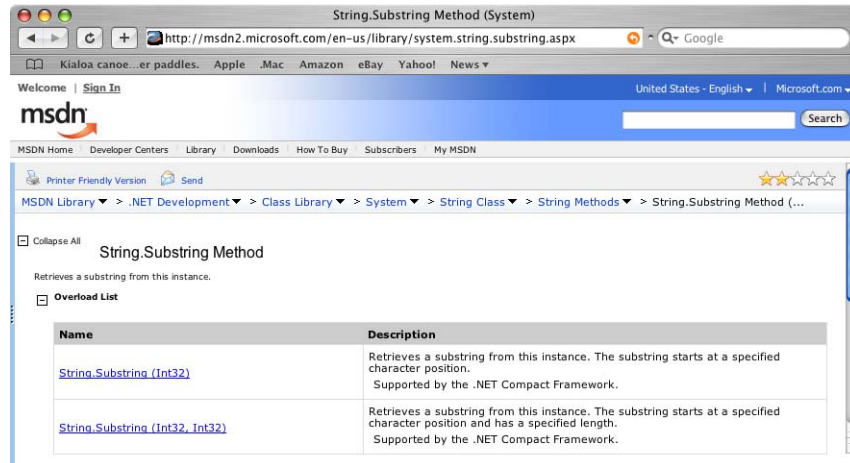


Figure 5-8: String.SubString Method Page

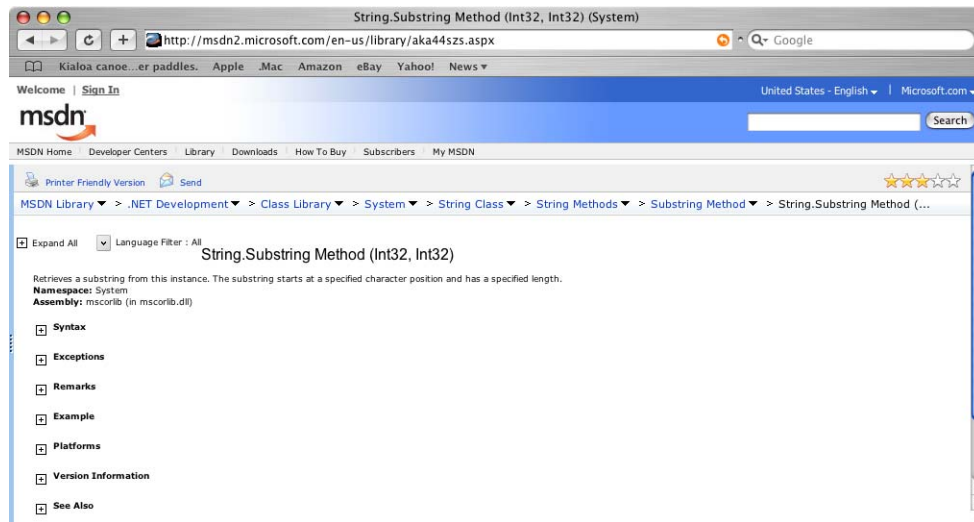


Figure 5-9: String.SubString Page with Collapsed Subheadings

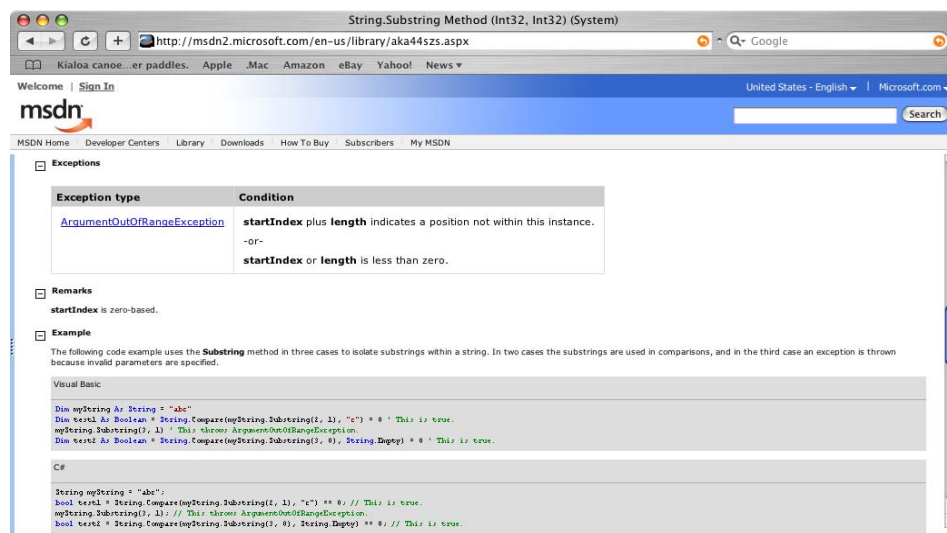


Figure 5-10: String.SubString Example Section Expanded Showing Example Code

Quick Review

The MSDN is the definitive source for .NET Framework API information. You'll spend most of your time in the Class Library section researching the many classes, interfaces, and value types that appear in the System namespace and its many subnamespaces.

THE BASE CLASS LIBRARIES (BCL)

Regardless of what type of C# application you build, your program will fundamentally depend upon a small core of classes that belong to the Base Class Libraries (BCL). The Base Class Libraries include the contents of the namespaces listed 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.CodeDom	Supports the ability to dynamically create and execute code.
System.Collections	Contains interfaces and classes that let you manipulate collections of objects. Includes data structures such as lists, hashtables, and dictionaries.
System.Diagnostics	Provides the ability to diagnose the performance of your application.
System.Globalization	Provides the capability to internationalize your application.
System.IO	Supports file, console, serial port, and interprocess input and output.
System.Resources	Provides classes and interfaces that allow you to store and manipulate culture-specific application resources.
System.Text	Provides the capability to manipulate sequences of ASCII, Unicode, UTF-7, and UTF-8 character encodings.
System.Text.RegularExpressions	Supports the use of regular expressions in your .NET applications.

Table 5-1: Base Class Library (BCL) Namespaces

In addition to these libraries, this book will draw heavily from the namespaces shown in Table 5-2.

Namespace	Description
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.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.

Table 5-2: Additional .NET Libraries Used Heavily In This Book

Namespace	Description
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 5-2: Additional .NET Libraries Used Heavily In This Book

Quick Review

The Base Class Libraries include those namespaces, and the classes they contain, that serve as the fundamental building blocks of the .NET Framework.

NAVIGATING AN INHERITANCE HIERARCHY

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

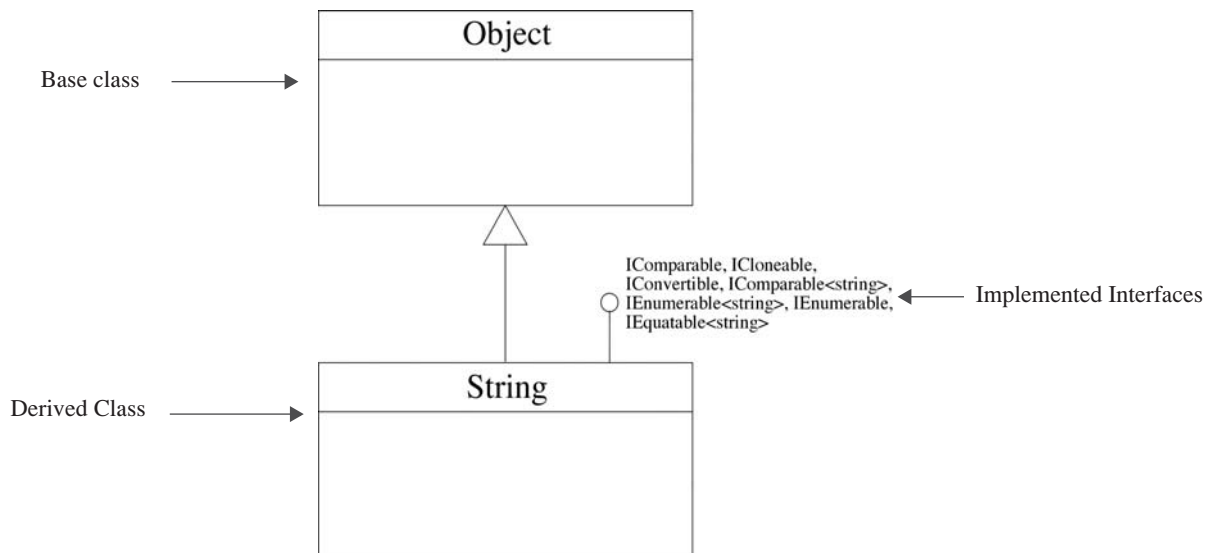


Figure 5-11: String Class Inheritance Hierarchy

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 shown in the Inheritance Hierarchy section as was shown in Figure 5-4. Unfortunately, the Inheritance Hierarchy section only gives part of the picture. You need to study the Syntax 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 Syntax 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, IComparable<string>,
    IEnumerable<string>, IEnumerable, IEquatable<string>
  
```

Referring to Example 5.1 — 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 it means to provide an implementation for it.

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.

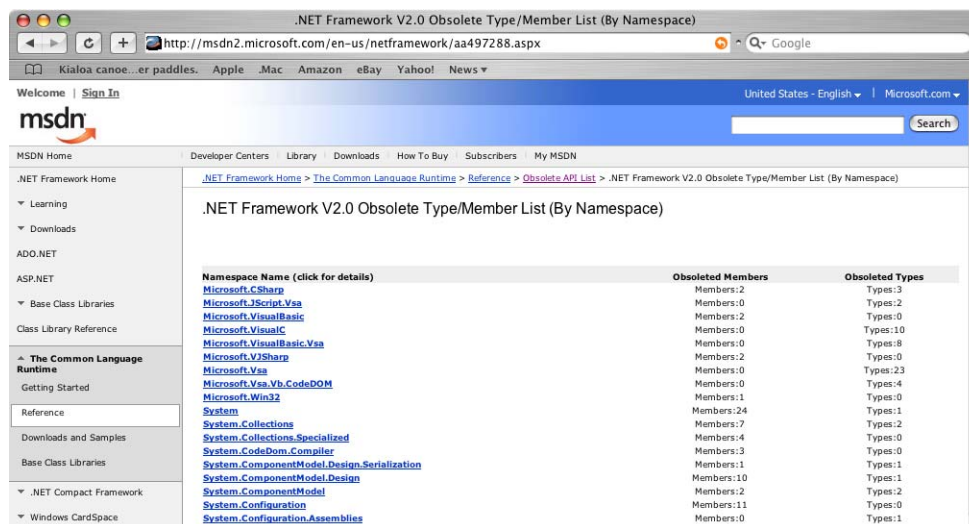
Do not panic if you don't know yet what all this mumbo jumbo means. By Chapter 11 it will all start to make perfect sense. However, it would still be a good exercise to visit the `String` class now and follow the links to all its related interfaces. Also, visit the `Object` class and see what it has to offer.

Quick Review

Trace a class's inheritance hierarchy to discover its complete range of functionality.

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 a listing of obsolete API members by visiting the MSDN home page and following these links: Common Language Runtime -> Reference -> .NET Framework V2.0 Obsolete API List. Figure 5-12 offers a partial listing of obsolete API components by Namespace.



Namespace Name (click for details)	Obsolete Members	Obsolete Types
Microsoft.CSharp	Members:2	Types:3
Microsoft.JScript.Vsa	Members:0	Types:2
Microsoft.VisualBasic	Members:2	Types:0
Microsoft.VisualBasic	Members:0	Types:10
Microsoft.VisualBasic.Vsa	Members:0	Types:8
Microsoft.VSSharp	Members:2	Types:0
Microsoft.Vsa	Members:0	Types:23
Microsoft.Vsa.Vb.CodeDOM	Members:0	Types:4
Microsoft.Win32	Members:1	Types:0
System	Members:24	Types:1
System.Collections	Members:7	Types:2
System.Collections.Specialized	Members:4	Types:0
System.CodeDom.Compiler	Members:3	Types:0
System.ComponentModel.Design.Serialization	Members:1	Types:1
System.ComponentModel.Design	Members:10	Types:1
System.ComponentModel	Members:2	Types:2
System.Configuration	Members:11	Types:0
System.Configuration.Assemblies	Members:0	Types:1

Figure 5-12: Obsolete .NET Framework Version 2.0 API Partial Listing by Namespace

SUMMARY

The Microsoft Developer Network (MSDN) is the definitive source for .NET Framework API information. You'll spend most of your time there in the Class Library section researching the many classes, interfaces, and value types that appear in the System namespace and its many subnamespaces.

The Base Class Libraries (BCL) include those namespaces, and the classes they contain, that serve as the fundamental building blocks of the .NET Framework.

Trace a class's inheritance hierarchy to discover its complete range of functionality.

Skill-Building Exercises

1. **API Drill:** Visit the MSDN 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 description 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 Developer Network website [<http://msdn.com>]

NOTES
