msdn® training

Module 1: Overview of the Microsoft .NET Framework

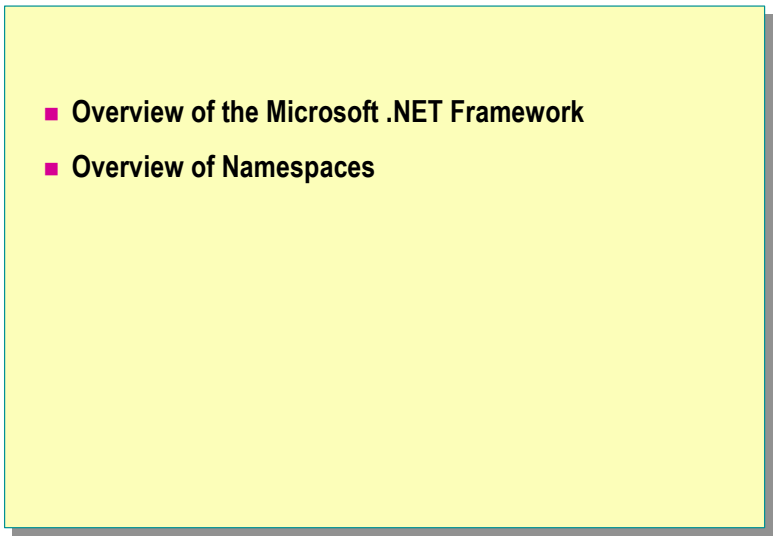
Contents

Microsoft®

Overview

Topic Objective
To provide an overview of the module topics and objectives.

Lead-in
In this module, you will be introduced to the .NET Framework. You will then learn about the namespaces and which modules teach certain namespaces.



The Microsoft® .NET Framework provides tools and technologies that you need to build distributed Web applications. In this module, you will learn the architecture of the .NET Framework. You will also learn how the .NET Framework class library is divided into namespaces, and which namespaces are taught in this course.

After completing this module, you will be able to:

- Describe the .NET Framework and its components.
- Explain the relationship between the .NET Framework class library and namespaces.

◆ Overview of the Microsoft .NET Framework

Topic Objective
To provide an overview of the .NET Framework topics.

Lead-in
In this section, you will learn about the Microsoft .NET Framework.

- **The .NET Framework**
- **Common Language Runtime**
- **The .NET Framework Class Library**
- **ADO.NET: Data and XML**
- **What is an XML Web Service?**
- **Web Forms and Services**

In this section, you will learn about the .NET Framework. The .NET Framework is a set of technologies that form an integral part of the Microsoft .NET platform. It provides the basic building blocks for developing Web applications and XML Web services.

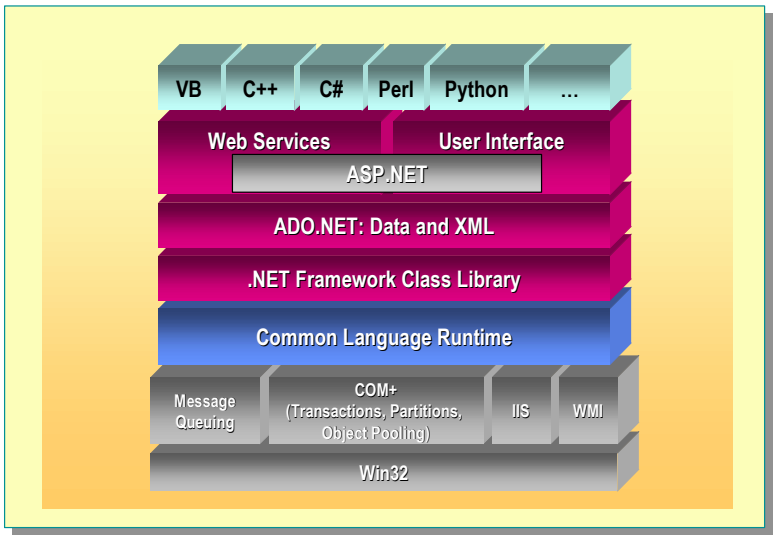
This section includes the following topics:

- The .NET Framework
- Common Language Runtime
- The .NET Framework Class Library
- ADO.NET: Data and XML
- What is an XML Web Service?
- Web Forms and Services

The .NET Framework

Topic Objective
To understand the architecture of the .NET Framework.

Lead-in
The .NET Framework is an architecture consisting of a runtime, the class library, and language support.



The .NET Framework

The .NET Framework provides the necessary compile time and runtime foundation to build and run .NET applications. The .NET Framework is the primary focus of this course.

Platform Substrate

The .NET Framework must run on an operating system. Currently, the .NET Framework is built to work on the Microsoft Win32® operating systems. In the future, the .NET Framework will be extended to run on other platforms, such as Microsoft Windows® CE.

Application Services

When running on Microsoft Windows 2000, application services, such as COM+, Message Queuing, Windows Internet Information Server (IIS), and Windows Management Instrumentation (WMI), are available to the developer. The .NET Framework exposes application services through classes in the .NET Framework class library.

Common Language Runtime

The common language runtime simplifies application development, provides a robust and secure execution environment, supports multiple languages, and simplifies application deployment and management.

The common language runtime environment is also referred to as a managed environment, in which common services, such as garbage collection and security, are automatically provided.

The .NET Framework Class Library

The .NET Framework class library exposes features of the runtime and provides other high-level services that every developer needs. The classes simplify development of .NET applications. Developers can extend them by creating their own libraries of classes.

ADO.NET

ADO.NET is the next generation of Microsoft ActiveX® Data Object (ADO) technology. ADO.NET provides improved support for the disconnected programming model. It also provides rich XML support.

ASP.NET

Microsoft ASP.NET is a programming framework that is built on the common language runtime. ASP.NET can be used on a server to build powerful Web applications. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web user interfaces (UI).

XML Web Services

The .NET Framework provides tools and classes for building, testing, and distributing XML Web services.

User Interfaces

The .NET Framework supports three types of user interfaces:

- Web Forms, which work through ASP.NET
- Windows Forms, which run on Win32 clients
- Console applications, which for simplicity, are used for most of the labs in this course

Languages

Any language that conforms to the Common Language Specification (CLS) can run on the common language runtime. In the .NET Framework, Microsoft provides Microsoft Visual Basic®, Microsoft Visual C++®, Microsoft Visual C#™, and Microsoft JScript® support. Third parties can provide additional languages.

Note C# has been submitted for standardization to ECMA, a vendor-neutral international standards organization committed to driving industry-wide adoption of information and communications technologies. This standardization will make it possible for any company which wishes to implement C# programming tools on any platform to do so. Microsoft has also submitted a subset of the Microsoft .NET Framework, called the Common Language Infrastructure (CLI), to ECMA. This will make it possible for other vendors to implement the CLI on a variety of platforms, so that software written using the basic architectural model presented by the .NET Framework can be created using a variety of tools on a variety of platforms.

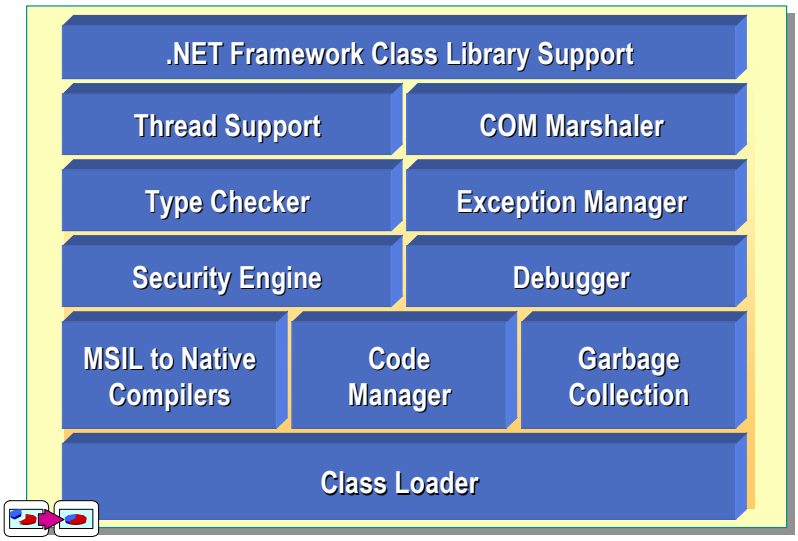
Building Components in the .NET Framework

In the .NET Framework, components are built on a common foundation. You no longer need to write the code to allow objects to interact directly with each other. In addition, you no longer need to write component wrappers in the .NET environment, because components do not use wrappers. The .NET Framework can interpret the constructs that developers are accustomed to using in object-oriented languages. The .NET Framework fully supports class, inheritance, methods, properties, events, polymorphism, constructors, and other object-oriented constructs.

Common Language Runtime

Topic Objective
To highlight some of the key components in the common language runtime.

Lead-in
This topic will give you an overview of the components of the common language runtime. I will briefly describe each component. As a C# developer, you will never see these discrete pieces, but this discussion will give you a better understanding of the richness of the runtime.



The common language runtime simplifies application development, provides a robust and secure execution environment, supports multiple languages, and simplifies application deployment and management.

The common language runtime environment is also referred to as a managed environment, one in which common services, such as garbage collection and security, are automatically provided. The common language runtime features are described in the following table.

Component	Description
Class loader	Manages metadata, and the loading and layout of classes.
Microsoft Intermediate Language (MSIL) to native compiler	Converts MSIL to native code on a just-in-time basis.
Code manager	Manages code execution.
Garbage collection	Provides automatic lifetime management of all of objects in the .NET Framework, garbage collection is a multiprocessor, scalable garbage collector.
Security engine	Provides evidence-based security, based on user identity and the origin of the code.
Debugger	Enables the developer to debug an application and trace the execution of code.
Type checker	Does not allow unsafe casts or uninitialized variables. MSIL can be verified to guarantee type safety.
Exception manager	Provides structured exception handling, which is integrated with Windows Structured Exception Handling (SEH). Error reporting has been improved.
Thread support	Provides classes and interfaces that enable multithreaded programming.
COM marshaler	Provides marshaling to and from COM.
.NET Framework class library support	Integrates code with the runtime that supports the .NET Framework class library.

The .NET Framework Class Library

Topic Objective
To provide an overview of the .NET Framework class library and the most common namespace: **System**.

Lead-in
In this topic, you will learn how .NET Framework class library exposes features of the runtime and provides other high-level services.

- **Spans All Programming Languages**
  - Enables cross-language inheritance and debugging
  - Integrates well with tools
- **Is Object-Oriented and Consistent**
  - Enhances developer productivity by reducing the number of APIs to learn
- **Has a Built-In Common Type System**
- **Is Extensible**
  - Makes it easy to add or modify framework features
- **Is Secure**
  - Allows creation of secure applications

The .NET Framework class library exposes features of the runtime and provides other high-level services that every developer needs.

Because there are hundreds of classes in the .NET Framework class library, classes are grouped into namespaces. The first part of the full name, which is located before the rightmost dot, is the namespace name. The last part of the name, which is located after the dot, is the type name.

For example, **System.Collections.ArrayList** represents the **ArrayList** class, which belongs to the **System.Collections** namespace. The types in **System.Collections** namespace can be used to manipulate collections of objects.

Spans All Programming Languages

The .NET Framework class library is language-independent so it enables cross-language inheritance and debugging. The .NET Framework class library also integrates fully with Microsoft Visual Studio® .NET, making it easy to develop applications with the library.

Is Object-Oriented and Consistent

Unlike flat APIs that are numerous and unorganized, the .NET Framework class library is organized into namespaces and classes. This object-oriented approach groups related functionality and data together and enables the developer to work with the library in a more natural way.

Has a Built-In Common Type System

The .NET Framework class library is type-safe. Type safety is ensured through the common type system, which is part of the common language runtime.

Is Extensible

You can extend the library by creating your own classes and compiling them into libraries. If designed properly, your class library will also be object-oriented and language-independent.

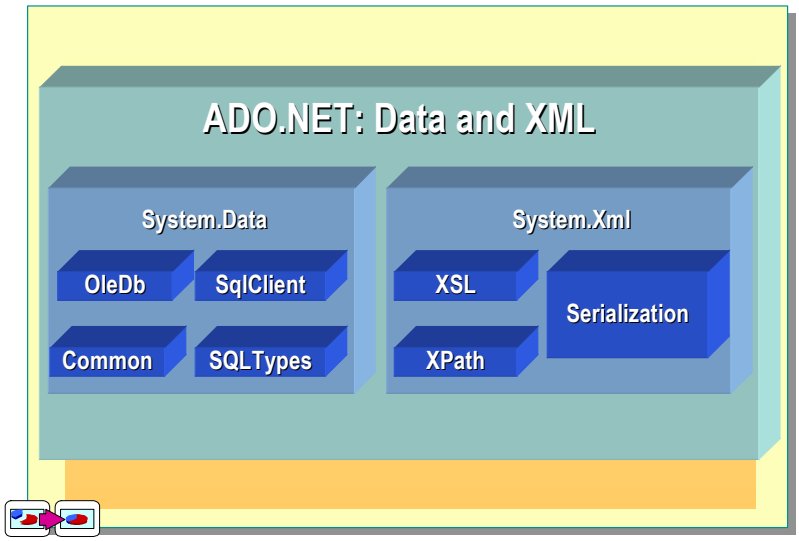
Is Secure

The .NET Framework class library provides rich security for your applications. You can use code access security and role-based security, and configure your own security policies. Furthermore, there are numerous security tools to assist in certificate creation, permission viewing, and so on.

ADO.NET: Data and XML

Topic Objective
To explain the data and XML support in the runtime.

Lead-in
The .NET Framework provides a new set of ADO.NET classes to handle data.



ADO.NET, the next generation of ADO technology provides improved support for disconnected programming. It also provides rich XML support in the **System.Xml** namespace.

System.Data Namespace

The **System.Data** namespace consists of classes that constitute the ADO.NET object model. At a high level, the ADO.NET object model is divided into two layers: the connected layer and the disconnected layer.

The **System.Data** namespace includes the **DataSet** class, which represents multiple tables and their relations. These data sets are completely self-contained data structures that can be populated from a variety of data sources. One data source could be XML; another data source could be an OLE DB; and a third data source could be the direct adapter for Microsoft SQL Server™.

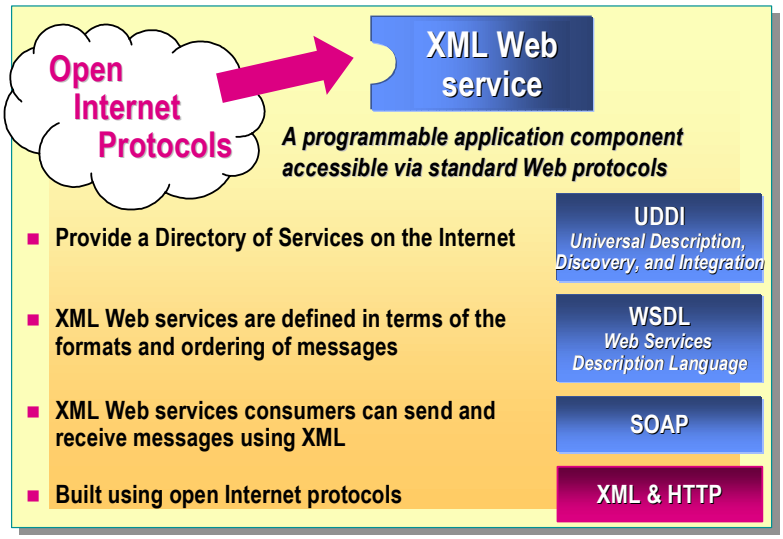
System.Xml Namespace

The **System.Xml** namespace provides support for XML. It includes an XML parser and a writer, which are W3C-compliant. The Extensible Stylesheet Language for Transformation (XSLT) is provided by the **System.Xml.Xsl** namespace. The implementation of XPath, a comprehensive language for document addressing, enables data graph navigation in XML. The **System.Xml.Serialization** namespace provides the entire core infrastructure for XML Web services, including such features as moving back and forth from objects to an XML representation.

What Is an XML Web Service?

Topic Objective
To define an XML Web service.

Lead-in
XML Web services are the fundamental building blocks of the .NET platform.



XML Web services are an integral part of the .NET platform. They are the fundamental mechanism for exposing and consuming data and functionality across Web applications, both inside organizations and across organizations.

XML Web services are units of application logic that provide data and services to other applications. Applications access XML Web services by means of industry standard Web protocols and data formats, such as HTTP, XML, and Simple Object Access Protocol (SOAP), regardless of how each XML Web service is implemented.

XML and HTTP

XML Web services are built by using XML and HTTP. Because they are built with XML and HTTP, XML Web services operate without firewall restrictions. Also, because XML and HTTP are industry standards, any platform supporting XML and HTTP can work with XML Web services.

SOAP

SOAP defines how messages are formatted, sent, and received when working with XML Web services. SOAP is also an industry standard that is built on XML and HTTP. Any platform that supports the SOAP standard can support XML Web services.

Web Services Description Language

The Web Services Description Language (WSDL) is an XML format for describing the network services that are offered by the server. You use WSDL to create a file that identifies the services that are provided by the server and the set of operations within each service that the server supports. For each of the operations, the WSDL file also describes the format that the client must follow when requesting an operation.

Universal Description, Discovery, and Integration

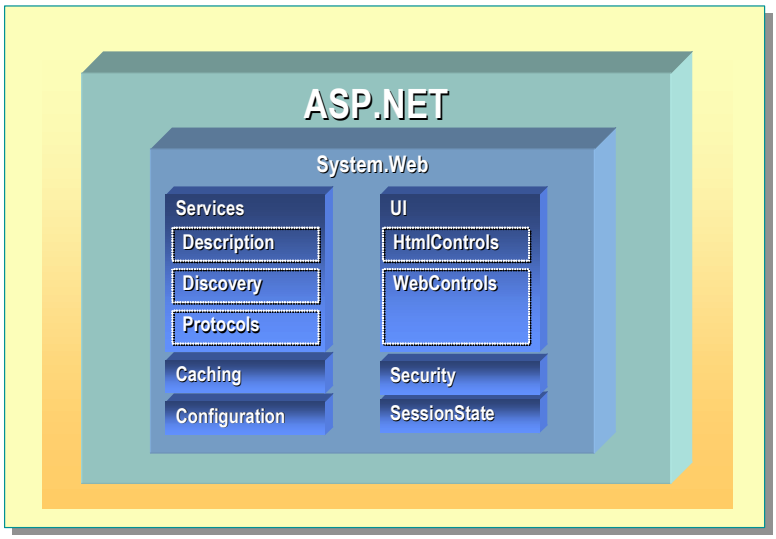
Universal Description, Discovery, and Integration (UDDI) is an industry standard for registering and searching for XML Web services. By using UDDI, developers can discover and use XML Web services that are available publicly over the Internet.

For more information on UDDI, see Web Service Discovery in Module 13, "Remoting and XML Web Services," in Course 2349B, *Programming with the Microsoft.NET Framework (Microsoft Visual C# .NET)*, and the UDDI Web site at http://www.uddi.org.

Web Forms and Services

Topic Objective
To show where Web Forms and XML Web services are found in the ASP.NET programming model.

Lead-in
The Internet is quickly evolving from today's Web sites that simply deliver UI pages to browsers to a next generation of programmable Web sites that link organizations, applications, services, and devices directly.



ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI pages. ASP.NET XML Web services provide the building blocks for constructing distributed Web-based applications. XML Web services are based on open Internet standards, such as HTTP and XML.

The common language runtime provides built-in support for creating and exposing XML Web services by using a programming abstraction that is consistent and familiar to both ASP Web Forms and Visual Basic developers. The resulting model is both scalable and extensible. This model is based on open Internet standards, such as HTTP, XML, SOAP, and WSDL, so it can be accessed and interpreted by any client or Internet-enabled device. Some of the more common ASP.NET classes are described in this topic as follows:

System.Web

In the **System.Web** namespace, there are lower-level services, such as caching, security, and configuration, which are shared between XML Web services and Web UIs.

System.Web.Services

The **System.Web.Services** namespace has classes that handle XML Web services, such as protocols and discovery.

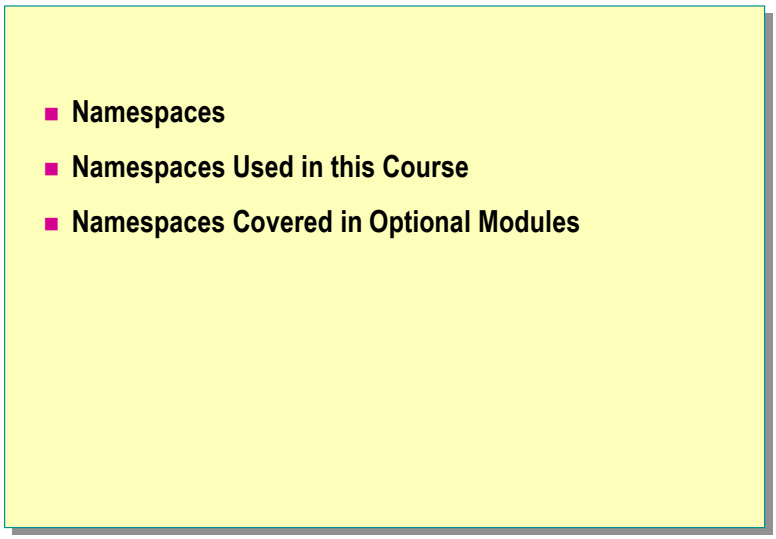
Controls

There are two types of controls: HTML controls and Web controls. The **System.Web.UI.HtmlControls** namespace gives you direct mapping of HTML tags, such as input. The **System.Web.UI.WebControls** namespace enables you to structure controls with templates, such as grid controls.

◆ Overview of Namespaces

Topic Objective
To provide an overview of the namespaces in the .NET Framework.

Lead-in
In this section, you will learn about the namespaces in the .NET Framework.



In this section, you will learn about the namespaces in the Microsoft .NET Framework. You will also learn about which namespaces are taught in this course.

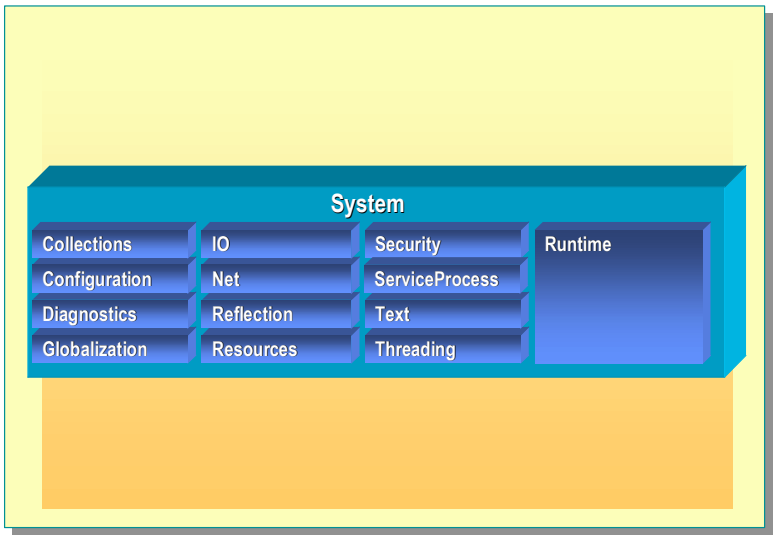
This section includes the following topics:

- Namespaces
- Namespaces Used in this Course
- Namespaces Covered in Optional Modules

Namespaces

Topic Objective
To understand how namespaces provide an easy-to-use hierarchy of types and functionality.

Lead-in
The .NET Framework includes a large set of class library assemblies, which contain hundreds of types. These assemblies provide access to system functionality in your development process.



The .NET Framework includes a large set of class library assemblies, which contain hundreds of types. These assemblies provide access to system functionality in your development process.

The Purpose of Namespaces

Because the .NET Framework class library includes definitions for so many types, the library is organized in a hierarchical namespace structure.

Namespaces use a dot-syntax naming scheme to group logically related classes together so that they can be easily searched and referenced. For example, the **System.Data** namespace contains the classes that constitute the ADO.NET architecture. The **System.Xml** namespace is the overall namespace for the XML classes that provide standards-based support for processing XML.

The System Namespace

The **System** namespace is the root namespace for types in the .NET Framework. The **System** namespace contains the base type **Object**, from which all other types are derived.

The **System** namespace also contains types for exception handling, garbage collection, console I/O, various tool types, format data types, random number generators, and math functions.

Namespaces Used in this Course

Topic Objective
To explain which namespaces are taught in this course, and which namespaces are not taught.

Lead-in
This course covers many of the **System** namespaces. Not all namespaces are covered, for example the **System.Security** namespaces.

Module 2
- System.Console

Module 3
- System.Windows.Forms
- System.Drawing

Module 4
- System.Reflection

Module 7
- System.Text
- System.Collections

Module 10
- System.IO

Module 11
- System.Net
- System.Net.Sockets

Module 12
- System.Runtime.Serialization

Module 13
- System.Runtime.Remoting.Channels
- System.Web.Services

This course covers many of the namespaces in the Microsoft .NET Framework. Module 2 teaches the **System.Console** namespace for printing output to the console. Module 3 teaches the **System.Windows.Forms** and **System.Drawing** namespaces for building a form with buttons that interacts with the user.

Module 4 teaches the **System.Reflection** namespace for storing version and key file information in an assembly. Module 7 teaches the **System.Text** namespace for advanced string management, and **System.Collections** for maintaining collections of data.

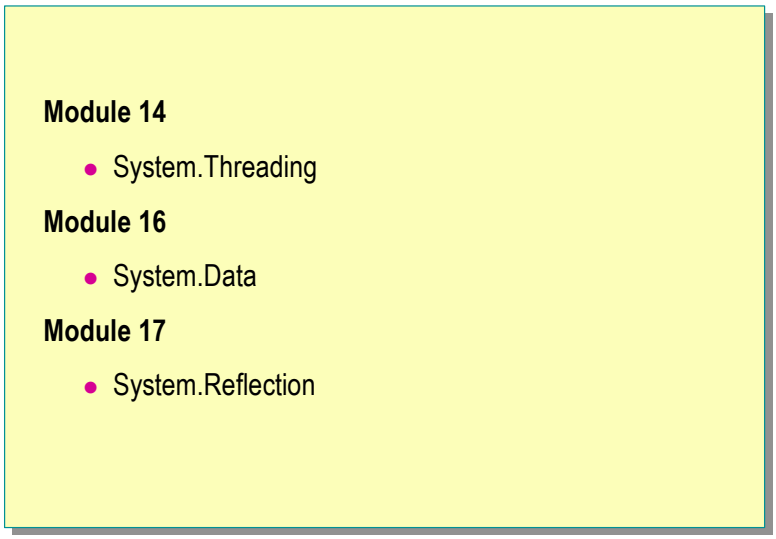
Module 10 teaches the **System.IO** namespace for reading and writing to files. Module 11 teaches the **System.Net** and **System.Net.Sockets** namespaces for transmitting data over the network.

Module 12 teaches the **System.Runtime.Serialization** namespace for persisting objects to storage. Module 13 teaches the **System.Runtime.Remoting.Channels** and **System.Web.Services** namespaces for invoking remote objects, and building XML Web services.

Namespaces Covered in Optional Modules

Topic Objective
To complete the namespace information that was presented in the preceding slide.

Lead-in
Here are some more namespaces that are covered in optional Modules 14 through 17 of this course.



Modules 14 through 17 are optional modules.

Module 14 teaches the **System.Threading** namespace for enabling multithreaded programming. Module 16 teaches the **System.Data** namespace, which provides the base objects and types for the ADO.NET programming model. ADO.NET also provides rich XML support in the **System.Xml** namespace. Finally, Module 17 teaches the **System.Reflection** namespace, which contains classes that you can use for examining metadata.

Namespaces Not Covered

This course does not teach the **System.Security** namespace. For more information about **System.Security** and related security namespaces, see Course 2350A, *Securing and Deploying Microsoft .NET Assemblies*.

Review

Topic Objective
To reinforce module objectives by reviewing key points.

Lead-in
The review questions cover some of the key concepts taught in the module.

- Overview of the Microsoft .NET Framework
- Overview of Namespaces

1. List the components of the .NET Framework.

   The common language runtime, .NET Framework class library, data and XML, XML Web services and Web Forms, and Windows Forms.

2. What is the purpose of the common language runtime?

   It provides an environment in which you can execute code.

3. What is the purpose of the common language specification?

   It defines a set of features that all .NET languages should support.

4. What is an XML Web service?

   An XML Web service is a programmable Web component that can be shared among applications on the Internet or an intranet.

5. What is a managed environment?

   A managed environment is an environment that provides services, such as garbage collection, security, and other related features.