

.NET Framework 3.5 Architecture

The .NET Framework version 3.5 builds upon the .NET Framework version 2.0 and the .NET Framework version 3.0, including service packs for the .NET Framework versions 2.0 and 3.0. This topic briefly describes the relationship of the .NET Framework versions 2.0, 3.0, and 3.5.

□ Relationship of the .NET Framework Versions 2.0, 3.0, and 3.5

The following are considered to be a part of the .NET Framework 3.5:

- .NET Framework 2.0
- .NET Framework 2.0 service pack 1, which updates assemblies that are included in the .NET Framework 2.0.
- .NET Framework 3.0, which uses the assemblies that exist in the .NET Framework 2.0, or .NET Framework 2.0 SP1 if it has been installed, and includes the assemblies that are necessary for the technologies that are introduced in the .NET Framework 3.0. For example, PresentationFramework.dll and PresentationCore.dll, which are necessary for Windows Presentation Foundation (WPF), are installed with the .NET Framework 3.0.

- .NET Framework 3.0 service pack 1, which updates the assemblies that were introduced in .NET Framework 3.0.
- New assemblies that provide additional functionality to the .NET Framework 2.0 and 3.0 and the technologies that are new to the .NET Framework 3.5.

If any of these are missing when the .NET Framework 3.5 is installed on a computer, they are installed automatically.

An application uses the same assemblies regardless of whether it targets the .NET Framework version 2.0, 3.0, or 3.5. For example, an application that uses WPF and targets the .NET Framework 3.0 uses the same instance of the mscorlib assembly as an application that uses Windows Forms and targets the .NET Framework 2.0. If the .NET Framework 2.0 SP1 has been installed on the computer, mscorlib.dll has been updated, and both applications use the updated version of mscorlib.dll.

Note:

The relationship of the .NET Framework versions 2.0, 3.0, and 3.5 differs from the relationship of versions 1.0, 1.1, and 2.0. The .NET Framework versions 1.0, 1.1, and 2.0 are completely separate from each other, and one version can be present on a computer regardless of whether the other versions are present. When versions 1.0, 1.1, and 2.0 are on the same computer, each version has its own common language runtime, class libraries, compiler, and so forth. Applications can choose whether to target version 1.0, 1.1, or 2.0. For more information, see [Side-by-Side Execution](#).

▣ What Is Included in the .NET Framework 3.5

This section summarizes the technologies that are in the .NET Framework 2.0, technologies that are in .NET Framework 3.0, and the features that are in the .NET Framework 3.5. This list is not exhaustive; it lists just some of the major technologies that ship in the .NET Framework.

.NET Framework 2.0

The following technologies shipped with the .NET Framework 2.0.

- Common language runtime (CLR).
- Support for generic types and methods.
- Compilers for C#, Visual Basic, C++, and J#.
- Base class libraries.
- ADO.NET.
- ASP.NET.
- Windows Forms.
- Web services.

.NET Framework 2.0 SP 1

The .NET Framework 2.0 service pack 1 updates the CLR and several assemblies that shipped with the .NET Framework 2.0 and can be installed independent of the .NET Framework 3.5. Most of the updates to .NET Framework 2.0 are nonbreaking changes, but there are

a few cases where new API elements are added or behavior has changed. If your application relies on new or changed functionality, it is recommended that your application target the .NET Framework 3.5. If your application relies on changes that shipped with .NET Framework 2.0 SP1, then you can have your application target the .NET Framework 2.0 and ask your customers to download the .NET Framework 2.0 SP1.

.NET Framework 3.0

The .NET Framework 3.0 requires the .NET Framework 2.0 to be installed on the computer. If a user installs the .NET Framework 3.0 on a computer that does not have the .NET Framework 2.0, the .NET Framework 2.0 is installed automatically.

The following technologies are introduced with the .NET Framework 3.0:

- Windows Presentation Foundation (WPF).
- Windows Communications Foundation (WCF).
- Windows Workflow Foundation (WF).

.NET Framework 3.0 SP 1

The .NET Framework 3.0 service pack 1 updates several assemblies that shipped with .NET Framework 3.0 and can be installed independent of the .NET Framework 3.5. The updates include nonbreaking changes, new API elements, and additional functionality to the

technologies that shipped with .NET Framework 3.0. If your application relies on new functionality, it is recommended that your application target the .NET Framework 3.5. If your application relies on changes that shipped with .NET Framework 3.0 SP1, then you can have your application target .NET Framework 3.0 and ask your customers to download the .NET Framework 3.0 SP1.

Installing the .NET Framework 3.0 SP1 installs the .NET Framework 2.0 SP1 if it is not already on the computer.

.NET Framework 3.5

The .NET Framework 3.5 introduces new features for the technologies in 2.0 and 3.0 and additional technologies in the form of new assemblies. The following technologies are introduced with the .NET Framework 3.5:

- LINQ.
- New compilers for C#, Visual Basic, and C++.
- ASP.NET AJAX.
- Additional types in the Base class library.

For a complete list of features new to the .NET Framework 3.5, see [What's New in the .NET Framework Version 3.5](#).

☐ See Also

Concepts

[What's New in the .NET Framework Version 3.5](#)

What's New in the .NET Framework Version 3.5

This topic contains information about new and enhanced features in the .NET Framework version 3.5.

☐ .NET Compact Framework

The .NET Compact Framework version 3.5 expands support for distributed mobile applications by including the Windows Communication Foundation (WCF) technology. It also adds new language features such as LINQ, new APIs based on community feedback, and improves debugging with updated diagnostic tools and features.

For details about these new features and enhancements, see [What's New in the .NET Compact Framework Version 3.5](#)

☐ ASP.NET

The .NET Framework 3.5 includes enhancements in targeted areas of ASP.NET and Visual Web Developer. The most significant advance is improved support for the development of AJAX-enabled Web sites. ASP.NET supports server-centric AJAX development with a set of new server controls and APIs. You can enable an existing ASP.NET 2.0 page for AJAX by adding a [ScriptManager](#) control and an [UpdatePanel](#) control so

that the page can update without requiring a full page refresh.

ASP.NET also supports client-centric AJAX development with a new client library called the Microsoft AJAX Library. The Microsoft AJAX Library supports client-centric, object-oriented development, which is browser-independent. By using the library classes in your ECMAScript (JavaScript) you can enable rich UI behaviors without roundtrips to the server. You can mix the degree of server-centric and client-centric development to meet the needs of your application. Furthermore, Visual Web Developer includes improved IntelliSense support for JavaScript and support for the Microsoft AJAX Library.

ASP.NET and Visual Web Developer now support the creation of both ASMX and WCF-based Web services and the seamless use of either implementation from Web pages using Microsoft AJAX Library. Furthermore, server-side application services including forms authentication, roles management, and profiles are now exposed as Web services that can be consumed in WCF-compatible applications, including client script and Window Forms clients. ASP.NET enables all Web-based applications to share these common application services.

Other improvements in ASP.NET include a new data control, [ListView](#), for displaying data; a new data source control, [LinqDataSource](#), that exposes Language Integrated Query (LINQ) to Web developers through the ASP.NET data source control architectures; a new tool, [ASP.NET Merge Tool \(Aspnet_merge.exe\)](#), for merging precompiled assemblies; and tight integration with IIS 7.0. [ListView](#) is a highly customizable control (using templates and styles) that also supports edit, insert, and delete operations, as well as sorting and paging functionality. The paging functionality for [ListView](#) is provided by a new control called [DataPager](#). You can use the merge tool to combine assemblies to support a range of deployment and release management scenarios. The integration of ASP.NET and IIS 7.0 includes the ability to use ASP.NET services, such as authentication and caching, for any content type. It also includes the ability to develop server pipeline modules in ASP.NET managed code and supports unified configuration of modules and handlers.

Other improvements in Visual Web Developer include multitargeting support, inclusion of Web Application Projects, a new Design view, new Cascading Style Sheets (CSS) design tools, and support for LINQ for SQL databases. Multitargeting enables you to use Visual Web Developer to target development of Web

applications to specific versions of the .NET Framework, including versions 2.0, 3.0, and 3.5.

For more information, see [What's New in ASP.NET and Web Development](#).

☐ Add-Ins and Extensibility

The System.AddIn.dll assembly in the .NET Framework 3.5 provides powerful and flexible support to developers of extensible applications. It introduces a new architecture and model that helps developers with the initial work to add extensibility to an application and by ensuring that their extensions continue working as the host application changes. The model provides the following features:

- Discovery

You can easily find and manage sets of add-ins in multiple locations on a computer with the [AddInStore](#) class. You can use this class to search for and obtain information about add-ins by their base types without having to load them.

- Activation

After an application chooses an add-in, the [AddInToken](#) class makes it easy to activate. Simply choose the isolation and sandboxing level and the system takes care of the rest.

- Isolation

There is built-in support for application domains and process isolation of add-ins. The isolation level for each add-in is in the control of the host. The system handles loading application domains and processes and shutting them down after their add-ins have stopped running.

- Sandboxing

You can easily configure add-ins with either a default or customized trust level. Support includes Internet, Intranet, Full Trust, and "same-as-host" permission sets, as well as overloads that let the host specify a custom permission set.

- UI Composition

The add-in model supports direct composition of Windows Presentation Foundation (WPF) controls that span application domain boundaries. You can easily allow add-ins to contribute directly to the UI of the host while still retaining the benefits of isolation, ability to unload, sandboxing, and versioning.

- Versioning

The add-in architecture makes it possible for hosts to introduce new versions of their object model without breaking existing add-ins or impacting the developer experience for new ones.

For more information, see [Add-ins and Extensibility](#).

☐ Common Language Runtime

Collections

[HashSet<Of <\(T\)>>](#) provides high performance set operations to the .NET Framework. A set is a collection that contains no duplicate elements, and whose elements are in no particular order. For more information, see [HashSet Collection Type](#).

Diagnostics

The [EventSchemaTraceListener](#) class provides tracing of end-to-end, schema-compliant events. You can use end-to-end tracing for a system that has heterogeneous components that cross thread, [AppDomain](#), process, and computer boundaries. A standardized event schema (see [Event Representation for Event Consumers](#)) has been defined to enable tracing across these boundaries. This schema is shared by various tracing technologies, including Windows Vista diagnostics tools such as Event Viewer. The schema also enables the addition of custom, schema-compliant elements.

The [EventSchemaTraceListener](#) class is tuned for logging performance with implicit support for lock-free tracing.

I/O and Pipes

Pipes provide interprocess communication between any processes running on the same computer, or on any

other Windows computer within a network. The .NET Framework provides access to two types of pipes: anonymous pipes and named pipes. For more information about pipes, see [Pipes](#).

Garbage Collection

The [GCSettings](#) class has a new [LatencyMode](#) property that you can use to adjust the time that the garbage collector intrudes in your application. You set this property to one of the values of the new [GCLatencyMode](#) enumeration.

The [GC](#) class has a new [Collect\(Int32, GCCollectionMode\)](#) method overload that you can use to adjust the behavior for a forced garbage collection. For example, you can use this overload to specify that the garbage collector should determine whether the current time is optimal to reclaim objects. This overload takes a value from the new [GCCollectionMode](#) enumeration.

Reflection and Reflection Emit in Partial Trust

Assemblies that run with partial trust can now emit code and execute it. Emitted code that calls only public types and methods needs no permissions beyond the permissions demanded by the types and methods that are accessed. The new [DynamicMethod\(String, Type, array<Type>\[\]\(\)\)](#) constructor makes it easy to emit such code.

When emitted code needs to access private data, the new [DynamicMethod\(String, Type, array<Type>\[\]\(\), Boolean\)](#) constructor allows restricted access. The host must grant [ReflectionPermission](#) with the new [RestrictedMemberAccess](#) flag to enable this feature, which gives emitted code the ability to access private data only for types and methods in assemblies with equal or lesser trust levels. See [Walkthrough: Emitting Code in Partial Trust Scenarios](#).

For reflection, a host grant of [RestrictedMemberAccess](#) similarly allows restricted use of methods that access private properties, call private methods, and so on, but only for target assemblies with equal or lesser trust levels.

Threading

Better Reader/Writer Lock

The new [ReaderWriterLockSlim](#) class provides performance that is significantly better than [ReaderWriterLock](#), and comparable with the **lock** statement (**SyncLock** in Visual Basic). Transitions between lock states have been simplified to make programming easier and to reduce the chances of deadlocks. The new class supports recursion to simplify migration from **lock** and from [ReaderWriterLock](#).

ThreadPool Performance Enhancements

Throughput for the dispatch of work items and I/O tasks in the managed thread pool is significantly improved. Dispatch is now handled in managed code, without transitions to unmanaged code and with fewer locks. The use of [ThreadPool](#) is recommended over application-specific thread pool implementations.

Time Zone Improvements

Two new types, [DateTimeOffset](#) and [TimeZoneInfo](#), improve support for time zones and make it easier to develop applications that work with dates and times in different time zones. For a discussion of which type to use in particular situations, see [Choosing Between DateTime, DateTimeOffset, and TimeZoneInfo](#).

TimeZoneInfo

The new [TimeZoneInfo](#) class largely supplants the existing [TimeZone](#) class. You can use [TimeZoneInfo](#) to retrieve any time zone defined in the registry, rather than just the local time zone and Coordinated Universal Time (UTC). You can also use this class to define custom time zones, to serialize and deserialize custom time zone data, and to convert times between time zones. For more information about developing applications that use the [TimeZoneInfo](#) class, see [Times and Time Zones](#).

DateTimeOffset

The new [DateTimeOffset](#) structure extends the [DateTime](#) structure to make working with times across time zones easier. The [DateTimeOffset](#) structure stores date and time information as a UTC date and time together with an offset value that indicates how much the time differs from UTC.

☐ Cryptography

ClickOnce Manifests

There are new cryptography classes for verifying and obtaining information about manifest signatures for ClickOnce applications. The [ManifestSignatureInformation](#) class obtains information about a manifest signature when you use its [VerifySignature](#) method overloads. You can use the [ManifestKinds](#) enumeration to specify which manifests to verify. The result of the verification is one of the [SignatureVerificationResult](#) enumeration values. The [ManifestSignatureInformationCollection](#) provides a read-only collection of [ManifestSignatureInformation](#) objects of the verified signatures. In addition, the following classes provide specific signature information:

- [StrongNameSignatureInformation](#)

Holds the strong name signature information for a manifest.

- [AuthenticodeSignatureInformation](#)

Represents the Authenticode signature information for a manifest.

- [TimestampInformation](#)

Contains information about the time stamp on an Authenticode signature.

- [TrustStatus](#)

Provides a simple way to check whether an Authenticode signature is trusted.

Suite B Support

The .NET Framework 3.5 supports the Suite B set of cryptographic algorithms published by the National Security Agency (NSA). For the NSA documentation, see www.nsa.gov/ia/industry/crypto_suite_b.cfm.

The following algorithms are included:

- Advanced Encryption Standard (AES) with key sizes of 128 and 256 bits for encryption.
- Secure Hash Algorithm (SHA-256 and SHA-384) for hashing.
- Elliptic Curve Digital Signature Algorithm (ECDSA) using curves of 256-bit and 384-bit prime moduli for signing. This algorithm is provided by the [ECDsaCng](#) class. It allows you to sign with a private key and verify with a public key.

- Elliptic Curve Diffie-Hellman (ECDH) using curves of 256 and 384-bit prime moduli for key exchange/secret agreement. This algorithm is provided by the [ECDiffieHellmanCng](#) class.

Managed code wrappers for the Federal Information Processing Standard (FIPS) certified implementations of the AES, SHA-256, and SHA-384 implementations are available in the new [AesCryptoServiceProvider](#), [SHA256CryptoServiceProvider](#), and [SHA384CryptoServiceProvider](#) classes.

The Cryptography Next Generation (CNG) classes provide a managed implementation of the native Crypto API (CAPI). Central to this group is the [CngKey](#) key container class, which abstracts the storage and use of CNG keys. This class allows you to store a key pair or a public key securely and refer to it using a simple string name. The [ECDsaCng](#) and [ECDiffieHellmanCng](#) classes use [CngKey](#) objects.

The [CngKey](#) class is used for a variety of additional operations, including opening, creating, deleting, and exporting keys. It also provides access to the underlying key handle to use when calling native APIs directly.

There are a variety of supporting CNG classes, such as [CngProvider](#), which maintains a key storage provider, [CngAlgorithm](#), which maintains a CNG algorithm, and

[CngProperty](#), which maintains commonly used key properties.

☐ Networking

Peer-to-Peer Networking

Peer-to-peer networking is a serverless networking technology that allows several network devices to share resources and communicate directly with each other.

The [System.Net.PeerToPeer](#) namespace provides a set of classes that support the Peer Name Resolution Protocol (PNRP) that allows the discovery of other peer nodes through [PeerName](#) objects registered within a peer-to-peer cloud. PNRP can resolve peer names to IPv6 or IPv4 IP addresses.

Collaboration Using Peer-to-Peer Networking

The [System.Net.PeerToPeer.Collaboration](#) namespace provides a set of classes that support collaboration using the Peer-to-Peer networking infrastructure. These classes simplify the process by which applications can:

- Track peer presence without a server.
- Send invitations to participants.
- Discover peers on the same subnet or LAN.
- Manage contacts.
- Interact with peers.

Microsoft's Peer-to-Peer collaboration infrastructure provides a peer-to-peer network-based framework for collaborative serverless activities. Use of this framework enables decentralized networking applications that use the collective power of computers over a subnet or the Internet. These types of applications can be used for activities such as collaborative planning, communication, content distribution, or even multiplayer game matchmaking.

Socket Performance Enhancements

The [Socket](#) class has been enhanced for use by applications that use asynchronous network I/O to achieve the highest performance. A series of new classes have been added as part of a set of enhancements to the [Socket](#) namespace. These classes provide an alternative asynchronous pattern that can be used by specialized high-performance socket applications. These enhancements were specifically designed for network server applications that require the high-performance.

☐ Windows Communication Foundation

WCF and WF Integration—Workflow Services

The .NET Framework 3.5 unifies the Windows Workflow Foundation (WF) and [Windows Communication Foundation](#) (WCF) frameworks so that you can use WF as a way to author WCF services or expose your

existing WF workflow as a service. This enables you to create services that can be persisted, can easily transfer data in and out of a workflow, and can enforce application-level protocols. For more information, see [Creating Workflow Services and Durable Services](#). For code samples, see [Workflow Services Samples](#).

Durable Services

The .NET Framework 3.5 also introduces support for WCF services that use the WF persistence model to persist the state information of the service. These durable services persist their state information on the application layer, so that if a session is torn down and re-created later, the state information for that service can be reloaded from the persistence store. For more information, see [Creating Workflow Services and Durable Services](#). For a code sample, see [Durable Service Sample](#).

WCF Web Programming Model

The WCF Web Programming Model enables developers to build Web-style services with WCF. The Web Programming Model includes rich URI processing capability, support for all HTTP verbs including GET, and a simple programming model for working with a wide variety of message formats (including XML, JSON, and opaque binary streams). For more information, see [Web](#)

[Programming Model](#). For code samples, see [Web Programming Model Samples](#).

WCF Syndication

WCF now includes a strongly typed object model for processing syndication feeds, including both the Atom 1.0 and RSS 2.0 formats. For more information, see [WCF Syndication](#). For code samples, see [Syndication Samples](#).

WCF and Partial Trust

In .NET Framework 3.5, applications running with reduced permissions can use a limited subset of WCF features. Server applications running with ASP.NET Medium Trust permissions can use the WCF Service Model to create basic HTTP services. Client applications running with Internet Zone permissions (such as XAML Browser Applications or unsigned applications deployed with ClickOnce) can use the WCF proxies to consume HTTP services. In addition, the WCF Web Programming Model features (including AJAX and Syndication) are available for use by partially trusted applications. For more information, see [Partial Trust](#). For code samples, see [Partial Trust WCF Samples](#).

WCF and ASP.NET AJAX Integration

The integration of WCF with the Asynchronous JavaScript and XML (AJAX) capabilities in ASP.NET provides an end-to-end programming model for building

Web applications that can use WCF services. In AJAX-style Web applications, the client (for example, the browser in a Web application) exchanges small amounts of data with the server by using asynchronous requests. Integration with AJAX features in ASP.NET provides an easy way to build WCF Web services that are accessible by using client JavaScript in the browser. For more information, see [AJAX Integration and JSON Support](#). For code samples, see [AJAX Samples](#).

Web Services Interoperability

In the .NET Framework 3.5, Microsoft maintains its commitment to interoperability and public standards and introduces support for the new secure, reliable, and transacted Web services standards:

- [Web Services Reliable Messaging v1.1](#)
- [Web Services Reliable Messaging Policy Assertion v1.1](#)
- [WS-SecureConversation v1.3](#)
- [WS-Trust v1.3](#)
- [WS-SecurityPolicy v1.2](#)
- [Web Services Atomic Transaction \(WS-AtomicTransaction\) Version 1.1](#)
- [Web Services Coordination \(WS-Coordination\) Version 1.1](#)

- [Web Services Policy 1.5 - Framework](#)
- [Web Services Policy 1.5 - Attachment](#)

Implementation of these protocols is made available using the new standard bindings, [<ws2007HttpBinding>](#) and [<ws2007FederationHttpBinding>](#), which are documented in the [Web Services Protocols Interoperability Guide](#). For a code sample, see [WS Binding Samples](#).

☐ Windows Presentation Foundation

In the .NET Framework 3.5, Windows Presentation Foundation contains changes and improvements in numerous areas, including versioning, the application model, data binding, controls, documents, annotations, and 3-D UI elements.

For details about these new features and enhancements, see [What's New in Windows Presentation Foundation Version 3.5](#).

☐ Windows Workflow Foundation

WCF and WF Integration—Workflow Services

The .NET Framework 3.5 unifies the Windows Workflow Foundation (WF) and Windows Communication Foundation (WF) frameworks so that you can use WF as a way to author WCF services or expose your existing WF workflow as a service. This enables you to create

services that can be persisted, can easily transfer data in and out of a workflow, and can enforce application-level protocols. For more information, see [Creating Workflow Services and Durable Services](#). For code samples, see [Workflow Services Samples \(WF\)](#).

Rules

The WF rules engine now supports extension methods, operator overloading, and the use of the new operator in your rules. For more information, see [Rule Changes in .NET Framework 3.5](#). For code samples, see [Rules and Conditions Samples](#).

☐ Windows Forms

ClickOnce Improvements

Several improvements have been made to ClickOnce. Improvements include deployment from multiple locations and third-party branding. For more information, see [Deploying ClickOnce Applications without Resigning](#) and [Creating ClickOnce Applications for Others to Deploy](#).

The Mage.exe tool, which is sometimes used together with ClickOnce, has been updated for the .NET Framework 3.5. For more information, see [Manifest Generation and Editing Tool \(Mage.exe\)](#).

Authentication, Roles, and Settings Services

Client application services are new in the .NET Framework 3.5 and enable Windows-based applications

(including Windows Forms and Windows Presentation Foundation applications) to easily access the ASP.NET login, roles, and profile services. These services enable you to authenticate users and retrieve user roles and application settings from a shared server.

You can enable client application services by specifying and configuring client service providers in your application configuration file or in the Visual Studio Project Designer. These providers plug into the Web extensibility model and enable you to access the Web services through existing .NET Framework login, roles, and settings APIs. Client application services also support occasional connectivity by storing and retrieving user information from a local data cache when the application is offline.

For more information, see [Client Application Services](#).

Windows Vista Support

Existing Windows Forms applications work seamlessly on Windows Vista, and they are upgraded to have the same appearance as applications written specifically for Windows Vista whenever possible. Common file dialog boxes are automatically updated to the Windows Vista version. The .NET Framework 3.5 also supports the User Account Control (UAC) Shield icon. For more information, see [FileDialog Class](#) and [Shield](#).

WPF support

You can use Windows Forms to host Windows Presentation Foundation (WPF) controls and content together with Windows Forms controls. You can also open WPF windows from a Windows Form. For more information about how to use Windows Forms and WPF together, see [Migration and Interoperability](#).

□ LINQ

Language-Integrated Query (LINQ) is a new feature in Visual Studio 2008 and the .NET Framework 3.5. LINQ extends powerful query capabilities to the language syntax of C# and Visual Basic in the form of standard, easily-learned query patterns. This technology can be extended to support potentially any kind of data store. The .NET Framework 3.5 includes LINQ provider assemblies that enable the use of LINQ for querying .NET Framework collections, SQL Server databases, ADO.NET Datasets, and XML documents.

The components of LINQ that are part of the .NET Framework 3.5 are:

- The [System.Linq](#) namespace, which contains the set of standard query operators and types and interfaces that are used in the infrastructure of a LINQ query. This namespace is in the System.Core.dll assembly.

- The [System.Data.Linq](#) namespace, which contains classes that support interaction with relational databases in LINQ to SQL applications.
- The [System.Data.Linq.Mapping](#) namespace, which contains classes that can be used to generate a LINQ to SQL object model that represents the structure and content of a relational database.
- The [System.Xml.Linq](#) namespace, which contains the classes for LINQ to XML. LINQ to XML is an in-memory XML programming interface that enables you to modify XML documents efficiently and easily. Using LINQ to XML, you can load XML, serialize XML, create XML trees from scratch, manipulate in-memory XML trees, and validate by using XSD. You can also use a combination of these features to transform XML trees from one shape into another.
- New types in the [System.Web.UI.WebControls](#) and [System.Web.UI.Design.WebControls](#) namespaces. These new types, such as [LinqDataSource](#), support the use of LINQ in ASP.NET Web pages through a data source control.
- The [DataRowComparer](#), [DataRowExtensions](#), and [DataTableExtensions](#) classes in the [System.Data](#) namespace support LINQ queries against ADO.NET [DataSet](#) objects.

In the class library, the LINQ extension methods that apply to a class are listed in the members page for the class, in the **Contents** pane, and in the **Index** pane.

☐ Expression Trees

Expression trees are new in the .NET Framework 3.5, and provide a way to represent language-level code in the form of data. The [System.Linq.Expressions](#) namespace contains the types that are the building blocks of expression trees. These types can be used to represent different types of code expressions, for example a method call or an equality comparison.

Expression trees are used extensively in LINQ queries that target remote data sources such as a SQL database. These queries are represented as expression trees, and this representation enables query providers to examine them and translate them into a domain-specific query language.

The [System.Linq.Expressions](#) namespace is in the System.Core.dll assembly.

☐ Programming Languages

Three Microsoft programming languages explicitly target the .NET Framework. For more information about new and enhanced features in these languages, see the following topics:

[What's New in Visual C#](#)

[What's New in Visual C++ 2008](#)

[What's New in the Visual Basic Language](#)

☐ See Also

Concepts

[What's New in ASP.NET and Web Development](#)

[What's New in the .NET Compact Framework Version](#)

[3.5](#)

[What's New in Windows Presentation Foundation](#)

[Version 3.5](#)

[What's New in Visual Studio 2008](#)