

Capítulo 1

Desarrollo de aplicaciones empresariales

Objetivo

Al finalizar el capítulo, el alumno

- Conocer el ambiente de desarrollo de Visual Studio.
- Identificar las características del .NET Framework.
- Entender las tecnologías desarrolladas por Microsoft para la implementación de soluciones empresariales.
- Comprender el uso de C#.NET y la POO.

Temas

1. Introducción al IDE Visual Studio
2. Características de .NET Framework
3. Tecnologías en la plataforma .NET
4. C# .NET y programación orientada a objetos
5. Nuevas características en las últimas versiones de C#

1. Introducción al IDE Visual Studio



Visual Studio es un IDE (Integrated Development Environment) que permite el desarrollo de aplicaciones de todo tipo (de escritorio, web y móviles).

Entre los lenguajes que soporta están C#, VB.NET, F#, C++, Javascript, Python pudiendo tener soporte a otros lenguajes por medio de extensiones (plugins); también, soporta XML/XSLT, HTML, CSS y variantes como Razor, Less y Sass.

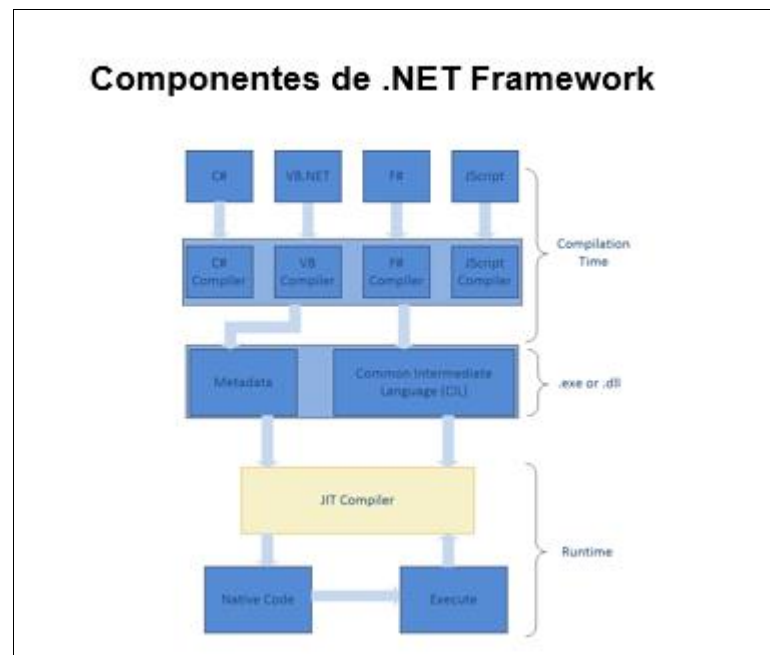
Su editor de código soporta características como autocompletado de código (intellisense) y refactoring. Tiene herramientas integradas para realizar debug de aplicaciones y editores visuales de clases, de interfaces web y de escritorio.

Ediciones de Visual Studio:

- Visual Studio Enterprise
- Visual Studio Profesional
- Visual Studio Test Professional
- Visual Studio Community
- Visual Studio Code
- Visual Studio Express
- Visual Studio Online

Es probable que las ediciones Express (gratuitas) salgan fuera del mercado ante la aparición de la versión Community que es mucho más completa (características equivalentes a la edición Pro); además de, ser gratuita y ofrecer soporte a complementos de terceros que le pueden dar más características.

2. Componentes de .NET Framework



Dentro de los componentes que posee el .Net Framework están:

- **Common Language Runtime (CLR):** el modelo de ejecución que propone la plataforma .NET se suele definir como "virtual", o "de máquina virtual", ya que las aplicaciones no son desarrolladas directamente contra las APIs de programación expuestas por el sistema operativo, ni es este el que se encarga de su ejecución y ciclo de vida, sino que .NET provee un entorno de ejecución (el CLR) que corre por sobre el sistema operativo y que es el encargado de ejecutar las aplicaciones y proveerles servicios en tiempo de ejecución. A los componentes de software que se ejecutan de esta manera se los conoce comúnmente como "componentes manejados", ya que su ejecución es controlada por un entorno intermedio.
- **Framework Class Library:** contiene la funcionalidad más comúnmente utilizada para el desarrollo de todo tipo de aplicaciones. Algunos ejemplos son el manejo de colecciones, cadenas de texto, entrada/salida, threading, operaciones matemáticas y dibujos 2D.
- **ADO.NET:** contiene un conjunto de clases que permiten interactuar con bases de datos relacionales y documentos XML como repositorios de información persistente.
- **LINQ:** también conocido como Language Integrated Query, permite escribir consultas de tipo seguro estructuradas sobre colecciones de objetos locales y fuentes de datos remotas. Es una característica presente desde C # 3.0 y .NET Framework 3.5.

LINQ permite consultar cualquier colección que implementa `IEnumerable<>`, ya sea una matriz, lista, XML DOM, o la fuente de datos remota (como una tabla en SQL Server). LINQ ofrece los beneficios de ambos, comprobación de tipos en tiempo de compilación y la composición de consulta dinámica.

- **EntityFramework:** es un ORM que facilita la tarea de crear una capa de acceso de datos, mediante la representación de los datos como un modelo conceptual, es decir, un conjunto de entidades y relaciones. La aplicación puede realizar las operaciones básicas CRUD (crear, leer, actualizar y eliminar) y gestionar fácilmente relaciones entre las entidades de uno a uno, uno a muchos, y muchos-a-muchos.
- **ASP.NET (WebForms, MVC, Web API):** tecnología dentro del .NET Framework para construir aplicaciones con interfaz de usuario Web (es decir, aplicaciones cuya lógica se encuentra centralizada en uno o varios servidores y que los clientes pueden acceder usando un browser o navegador mediante una serie de protocolos y estándares como HTTP y HTML).
- **Winforms:** permite crear aplicaciones con interfaz de usuario basada en formularios y ventanas Windows que se ejecutan directamente en los clientes.
- **WPF:** permite la creación de aplicaciones de escritorio.
- **WCF:** permite la creación de servicios basados en SOAP.
- **WF:** permite la creación de Workflows de aplicaciones de manera visual.
- **TPL Async:** permite realizar programación asíncrona.
- **Modern UI Runtime:** permite la ejecución de aplicaciones del Windows Store.

Características del .NET Framework

1. Compilación Just In Time (o Justo a tiempo): este es un proceso de 2 pasos:
 - a. En la compilación que se hace durante el desarrollo se obtiene un código intermedio conocido como MSIL que es equivalente en diferentes lenguajes de .NET como VB.NET y C#.
 - b. El CLR se encarga de compilar el código MSIL a instrucciones de CPU. Esto lo hace sin intervención alguna del desarrollador o el usuario, este paso se hace en tiempo de ejecución.
2. Gestión automática de memoria: el CLR abstrae a los desarrolladores de tener que pedir y liberar memoria explícitamente. Para esto, uno de sus componentes llamado Garbage Collector (Recolector de basura) se encarga de liberar periódicamente la memoria que ya no está siendo usada por ninguna aplicación. Por otra parte, el CLR también abstrae a los desarrolladores del uso de punteros y del acceso a memoria de bajo nivel.

Si bien estas características pueden ser consideradas poderosas, suelen hacer el desarrollo y mantenimiento de aplicaciones más propenso a errores y menos productivo.

3. Gestión de errores consistente: como las aplicaciones .NET no se ejecutan directamente contra el sistema operativo, cualquier error no manejado que ocurra en tiempo de ejecución será atrapado por el CLR en última instancia, no afectando a ninguna otra aplicación que se esté ejecutando ni teniendo efecto alguno sobre su estabilidad.
4. Ejecución basada en componentes: todas las aplicaciones .NET son empaquetadas en componentes reutilizables denominados genéricamente Assemblies, que el CLR se encarga de cargar en memoria y ejecutar.
5. Gestión de seguridad: el CLR provee una barrera más de contención a la hora de ejecutar aplicaciones manejadas, ya que permite establecer políticas de seguridad muy detalladas que las aplicaciones .NET que se ejecuten en una determinada computadora deberán cumplir.
6. Multithreading: el CLR provee un entorno de ejecución multi-hilos por sobre las capacidades del sistema operativo, así como también mecanismos para asegurar su sincronización y acceso concurrente a recursos compartidos.

NuGet Package Manager

NuGet es un gestor de paquetes de código libre y abierto para la plataforma de desarrollo de Microsoft. Se distribuye como una extensión de Visual Studio; a partir de Visual Studio 2012, viene preinstalado por defecto. También, se puede utilizar desde la línea de comandos y automatizado a través de scripts.

NuGet aporta todos los beneficios de cualquier gestor de paquetes: se ocupa de las cadenas de dependencia y conflictos de versiones en el momento de la instalación, y que facilita encontrar, instalar, actualizar y desinstalar paquetes en el nivel de aplicación.


3. C# y Programación orientada a objetos

C#.NET y Programación orientada a objetos

- Type Safe.
- Encapsulación, polimorfismo, herencia.
- Clases, interfaces, propiedades, métodos y eventos.

1 - 7

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



C# es un lenguaje de propósito general, es type safe y orientado a objetos. El objetivo del lenguaje es conseguir productividad. Junto con VB.NET estos lenguajes son independientes de plataformas, pero fueron escritos para trabajar de manera adecuada con el .NET Framework.

Características de C#

- Orientación a objetos

C# ofrece soporte al paradigma de orientación a objetos el que incluye encapsulación, polimorfismo y herencia. Ofrece formas de implementar directamente clases, interfaces, propiedades, métodos y eventos.

- Type Safe

El lenguaje requiere seguridad de tipos en tiempo de compilación. Esto quiere decir que las variables y propiedades deben tener un tipo específico (int, string, DateTime, etc.) y que no podemos asignarle valores de otros tipos a esas variables/propiedades. Esta característica elimina una gran cantidad de errores antes de que un programa sea ejecutado.

C# puede soportar la ausencia de tipos específicos mediante el uso de la palabra reservada **dynamic**. Sin embargo, esta debe ser usada en casos específicos.

Encapsulación, polimorfismo y herencia

Al ser lenguajes orientados a objetos, C# y VB.NET soportan las características de dicho paradigma, es más C# fue diseñado principalmente con esta intención mientras que VB.NET fue adaptado cumpliendo de buena manera con el paradigma.

- **Clases, interfaces, propiedades, métodos y eventos**

Todos estos elementos forman parte de ambos lenguajes; aunque la sintaxis varía en cada lenguaje al realizar el proceso de compilación, la sintaxis MSIL llega a ser equivalente.

4. Nuevas características en C#

Nuevas características en C#	
C# 4.0	C# 6.0
<ul style="list-style-type: none"> - Dynamic (palabra reservada que permite obviar tipos estáticos) - Parámetros opcionales y argumentos nombrados (brindan la ventaja de tener menos código y una sintaxis más fácil de leer) - Mejorar en la interoperabilidad COM 	<ul style="list-style-type: none"> - Expresiones nameof - String interpolation - Null conditional operator (Elvis operator) - Index initializers - Auto-property initializers - Getter-only auto properties - Expression-bodied function members
C# 5.0	
<ul style="list-style-type: none"> - Async/await (funciones asíncronas) 	

C# 4.0

- Dynamic (palabra reservada que permite obviar tipos estáticos).
- Parámetros opcionales y argumentos nombrados (brindan la ventaja de tener menos código y una sintaxis más fácil de leer).
- Mejorar en la interoperabilidad COM.

C# 5.0

- Async/await (funciones asíncronas).

C# 6.0

- Expresiones *nameof*: permite obtener el nombre de variables como strings con el fin de facilitar el mantenimiento de código.

```
throw new ArgumentNullException(nameof(param1));
```


- String interpolation: sintaxis más simple para dar formato a cadenas de caracteres.

```
//Antes  
Log.Information("Logged in {UserId}", loggedInUserId);  
//C# 6  
Log.Information($"Logged in {loggedInUserId}");
```

- Null conditional operator (Elvis operator)

```
var result = Foo()?.Length
```

- Auto-property initializers

```
public Guid Id { get; } = Guid.NewGuid();
```

- Getter-only auto properties

```
public int MyProp { get; }
```

- Expression-bodied function members: permite declarar métodos de manera mucho más reducida.

```
public string ToHex() => string.Format("#{0:X2}{1:X2}{2:X2}", Red, Green,  
Blue);
```