Sistemas distribuidos y de tiempo real

1.- Introducción a .NET Framework 1.1

Qué es .Net?

- Conjunto de nuevas tecnologías Microsoft
  - Mejorar sus sistemas operativos
  - Mejorar su modelo de componentes COM+
  - Obtener un entorno específicamente diseñado para el desarrollo y ejecución del software en forma de servicios que puedan ser tanto publicados como accedidos a través de Internet .
    - Independiente del lenguaje de programación
    - Modelo de objetos
    - Sistema operativo y hardware utilizados tanto para desarrollarlos como para publicarlos.

PLATAFORMA .NET

.net Framework – General

- **.NET Framework es un componente de Microsoft Windows que simplifica el desarrollo de aplicaciones en un entorno de red y particularmente Internet**

- Ofrece
  - Un entorno productivo y basado en estándares
    - HTTP
    - SOAP
    - XML
  - Flexibilidad para la implementación y aplicación en la empresa
- Compuesto por:
  - CLR : Common Language Runtime
  - Conjunto de bibliotecas de clases

.net Framework – Características

- Arquitectura flexible
- Desarrollo rápido
  - Uso de cualquier lenguaje de programación (integración de varios)
  - Escribir menos código
  - Servicios de aplicaciones de Windows
- Operaciones eficientes
  - Mejora del rendimiento
  - Simplificación de implementación
  - Generación de aplicaciones más confiables

.net Framework – Composición

- **Common Language Runtime(CLR)**

  Common Language Runtime (CLR) es responsable de los servicios en tiempo de ejecución, como por ejemplo, la integración de lenguajes, el cumplimiento de las normas de seguridad y la administración de la memoria, los procesos y los subprocesos.
  (Similar a JVM)

- **Bibliotecas de clases**

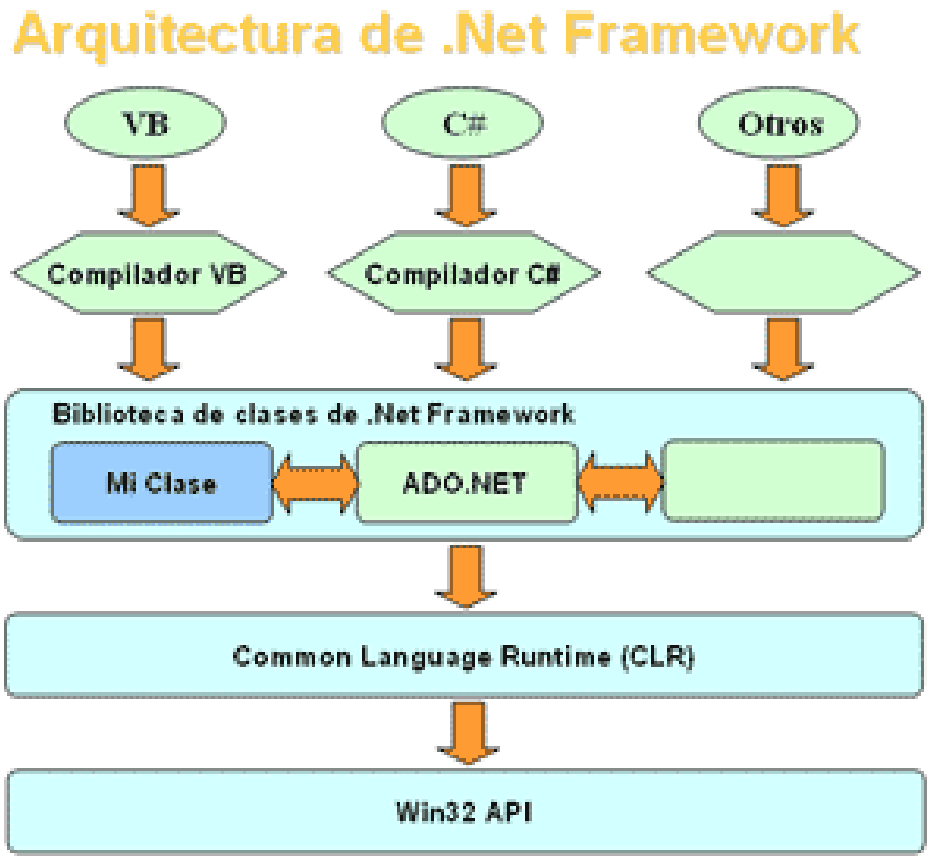
  Las clases base proporcionan funciones estándar, como las de entrada/salida, manipulación de cadenas, administración de seguridad, comunicaciones en red, administración de subprocesos, administración de textos y funciones de diseño de la interfaz de usuario.

  Además: ADO.NET, XML, ASP.NET, Windows Forms

.Net Framework – Servicios y facilidades

- Modelo de programación consistente y sencillo, completamente orientado a objetos.
- Código administrado
- Eliminación del "infierno de las DLL's"
- Ejecución multiplataforma
- Ejecución multilenguaje
- Compilación JIT : Just In Time
- Recolección de basura (Garbage Collector)
- Soporte multihilo
- Gestión del acceso a objetos remotos que permite el desarrollo de aplicaciones distribuidas de manera transparente a la ubicación real de cada uno de los objetos utilizados en las mismas.
- Seguridad avanzada, hasta el punto de que es posible limitar los permisos de ejecución del código en función de su procedencia (Internet, red local, CD-ROM, etc.), el usuario que lo ejecuta o la empresa que lo creó.
- Interoperabilidad con código preexistente

Arquitectura de .net Framework



¿OK?

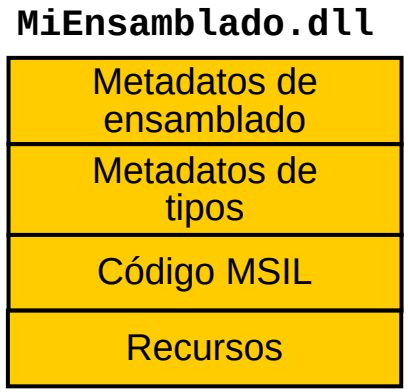
Ensamblados

- Problema
  - Diferentes archivos binarios (DLL's), registro, conectividad abierta a bases de datos (ODBC)
- Solución
  - Ensamblado: Ficheros EXE o DLL que contienen toda la funcionalidad de la aplicación encapsulada
    - No hay que registrar componentes → Almacenaje de toda la información dentro del propio ensamblado
    - Métodos y propiedades en forma de meta-datos + Información descriptiva (permisos, dependencias)
    - Ensamblados
      - Estáticos
      - Dinámicos

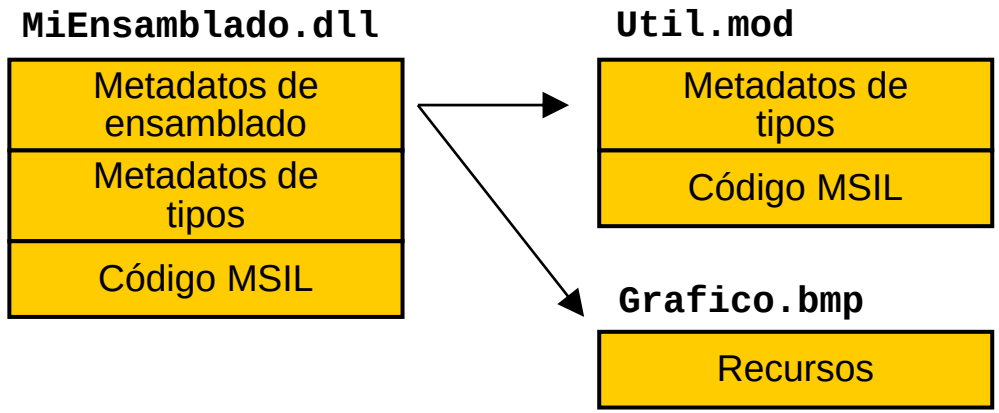
MANIFIESTO

Ensamblado estático

1. En un solo archivo

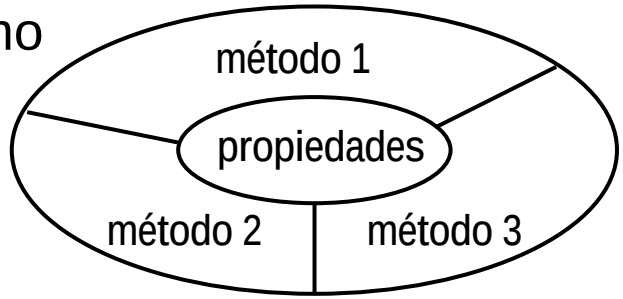


2. En múltiples archivos



Programación orientada a objetos

- Evolución de la programación procedural o procdimental
- POO : Agrupar código encapsulándolo y haciéndolo independiente
- La organización de una aplicación en POO se realiza mediante estructuras de código, también llamados objetos.
- Los objetos se crean a partir de una serie de especificaciones o normas que definen como va a ser el objeto (clase).
- Propiedades → Variables **(definen los datos o información del objeto)**
- Métodos → Funciones **(rutinas que definen el comportamiento del objeto**)
- Herencia y Polimorfismo



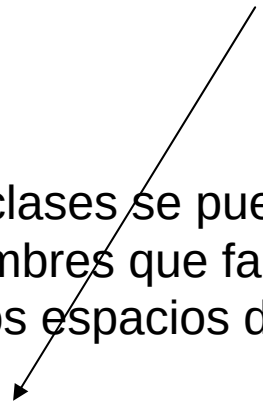
Lenguaje de programación C# - Generalidades

- Diseñado para ejecutarse sobre .NET
- Orientado a objetos como Java, C++
- Creado desde cero
- Fácil curva de aprendizaje
- Retiene gran parte de la sintaxis de C / C++
- Todo el código y datos en C# deben estar incluidos en una **clase**

Clases en C# (I)

```
namespace Nombre{
 class NombreClase
 {
   <[EA] tipoDato NombreVariable;>*
   <[EA] [tipoDatoRetorno] NombreMetodo (Parametros)
    {
      Cuerpo de la función o método
    }
   >*
 }
}
```

- Las clases se pueden agrupar en colecciones de clases llamados espacios de nombres que faciliten la organización.
- Varios espacios de nombres anidados



- Función : Devuelve información o resultados
- Método : No devuelve información (“Biblia de C#”)

} INDISTINTAMENTE

Clases en C# (II)

```
namespace Vehiculos{
 public class Bicicleta
 {
   public string Modelo;
   public int NumeroDeVelocidades
   private int Velocidad
         public void Acelerar(int km){
                  Velocidad = Velocidad + km;
         }
         public void Frenar(){
                  if (Velocidad > 0)
                  {
                           Velocidad = Velocidad - 1;
                  }
         }
         public int ConsultarVelocidad(){
                  return Velocidad;
         }
 }
}
```

Clases en C# (III)

- Los constructores de la clase se llaman como ella. Puede haber 0 o varios siempre que la lista de parámetros sea distinta
- Un único destructor, sin retorno ni parámetros
- El constructor se invoca con el operador **new**
- El destructor se invoca automáticamente por el recolector de basura cuando no quedan referencias sobre el objeto
- Las clases tienen dos tipos de miembros
  - Instancia
  - Compartidos: Se declaran por la cláusula **static**
- También se pueden declarar miembros estáticos (***static***) que se pueden invocar sin necesidad de instanciar un objeto de una clase.

Clases en C# (IV)

```
using Vehiculos;
Bicicleta miBici = new Bicicleta();
miBici.Acelerar(30);
miBici.ConsultarVelocidad();
miBici.Frenar();
```

```
Vehiculos.Bicicleta miBici = new Vehiculos.Bicicleta();
miBici.Acelerar(30);
miBici.ConsultarVelocidad();
miBici.Frenar();
```

```
using V = Vehiculos;
V.Bicicleta miBici = new V.Bicicleta();
miBici.Acelerar(30);
miBici.ConsultarVelocidad();
miBici.Frenar();
```

Especificadores de acceso

- **Public:** Visible en todos los ámbitos (Miembros y clases)
- **Protected:** Visible para la clase y sus derivadas (Solamente para miembros)
- **Private:** Clase privada sólo se puede usar en el ámbito que se ha definido. Miembros sólo se pueden usar dentro de la propia clase.
- **Internal:** Similar a *public*. Visible desde fuera del ámbito pero no desde fuera del ensamblado
- **Protected internal:** Los identificadores pueden usarse en el ensamblado que se definieron, así como en clases derivadas a pesar de que se encuentren en otros ensamblados.

Tipos de datos .NET

- Tipos de datos implementados como clases
- Tipos intrínsecos funcionan como si no fu
  objetos
- Mismos tipos de datos para todos los
  lenguajes .NET

```
String Cadena;
Int Longitud;
Cadena = "Datos";
Longitud = Cadena.Length();
```

- `int Num=5; Object RNum;`

```
Rnum=Num;                    //empaquetado
Num=(int) Rnum;              //desempaquetado
```

Nombre de la clase	Tipo de dato en C#	Descripción
Byte	Byte	Entero sin signo de 8 bit.
Sbyte	sbyte	Entero sin signo de 8bit (Tipo no acorde con el CLS)
Int16	short	Entero con signo de 16 bit.
Int32	int	Entero con signo de 32 bit.
Int64	long	Entero con signo de 64 bit.
Uint16	ushort	Entero sin signo de 16 bit. (Tipo no acorde con el CLS)
Uint32	uint	Entero sin signo de 32 bit. (Tipo no acorde con el CLS)
Uint64	ulong	Entero sin signo de 64 bit. (Tipo no acorde con el CLS)
Single	float	Numero con coma flotante de precisión simple, de 32 bit.
Double	double	Numero con coma flotante de precisión doble, de 64 bit.
Boolean	bool	Valor logico
Char	char	Carácter unicode de 16 bit.
Decimal	decimal	Valor decimal de 96 bit.
IntPtr	--	Entero con signo cuyo tamaño depende de la plataforma: 32 bit en plataformas de 32 bit y 64 bit en plataformas de 64 bit. (Tipo no acorde con el CLS)
UintPtr	--	Entero sin signo cuyo tamaño depende de la plataforma: 32 bit en plataformas de 32 bit y 64 bit en plataformas de 64 bit. (Tipo no acorde con el CLS)
String	string	Cadena de caracteres.

OBJECT

Declaración de variables

- Tipo C/C++

```
int numero = 0;
```

- Enumeraciones

```
enum colores = {Blanco, Azul, Rojo};
color = colores.Rojo;
```

- Estructuras

```
struct estructura{
 public estructura (int a, char b){ //Constructor
 x =a; y = b; }
 public int x;
 private char y;
}
```

Vectores y matrices

- Declaración

```
int [] vector = new int [10];
```

- Varias dimensiones

```
float [,] valores = new float [23,49];
```

- Clase **Array**
  - Rank
  - GetLength
  - Clone
  - Copy
  - Sort
  - Reverse
  - IndexOf

Métodos de la clase **Array**

Clases

Incluye clases para realizar todo tipo de programas en Windows

Inicialmente (en VS corresponde al tipo de proyecto Win32)

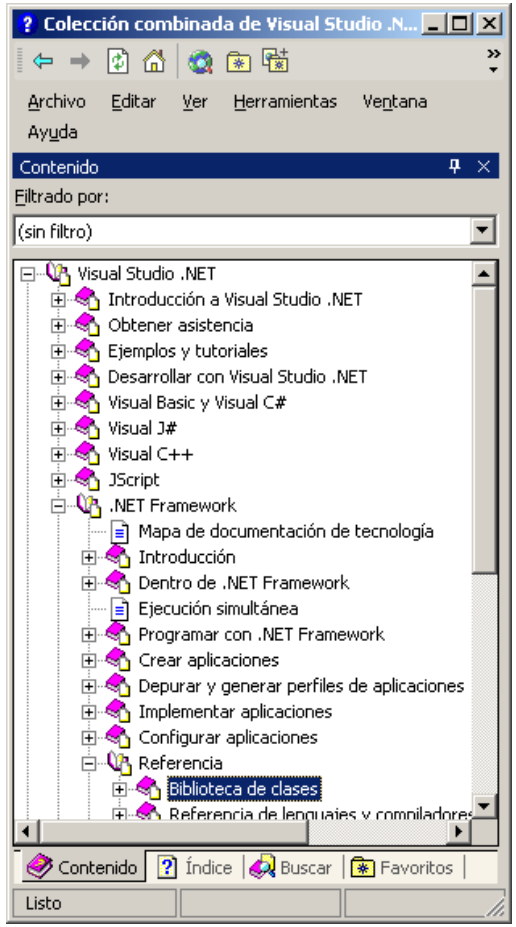
Se utiliza la API Win32 que da acceso a los servicios del SO
El modelo de programación es orientado a procedimientos

Después (en VS corresponde a los tipos de proyecto MFC y ATL)

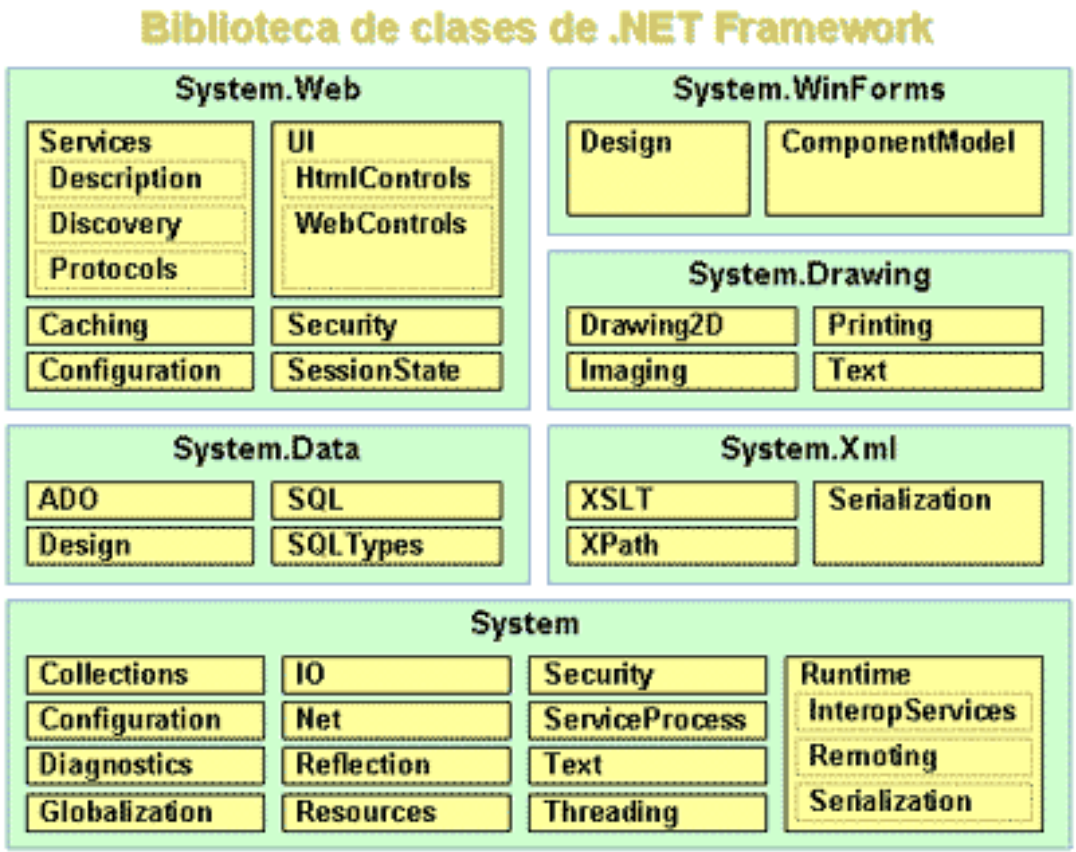
Se encapsula la API Win32 en la librería de clases MFC
Para las aplicaciones distribuidas se añade la librería ATL
El modelo de programación es orientado a objetos

Ahora (en VS corresponde al tipo de proyecto .NET)

Su usa la librería .NET para todas los tipos de aplicación
El modelo de programación es orientado a objetos



Biblioteca de clases de .net Framework



Espacios de nombres (I)

System

Contiene clases fundamentales y clases base que define tipos de datos de uso frecuente

System.Collections

Contiene interfaces y clases que definen varias colecciones de objetos como listas, colas, matrices de bits, tablas hash y diccionarios.

System.Data → Acceso a la capa de datos en SDs

Contiene las clases que permiten acceder a datos (bases de datos) usando la tecnología ADO.NET

System.Diagnostics

Contiene clases para interactuar con los procesos del sistema, registros de eventos y contadores de rendimiento

System.Drawing

Contiene clases que proporcionan acceso a la funcionalidad de manejo de gráficos GDI+

System.EnterpriseServices → Transacciones en SDs

Proporciona a los objetos .NET acceso a servicios COM+ usados aplicaciones empresariales distribuidas que procesan transacciones complejas

System.Globalization

Contiene clases que definen información útil para escribir aplicaciones globales o internacionales

Espacios de nombres (II)

System.IO
Contiene clases y tipos para la lectura y escritura de archivos y secuencias de datos (streams)

System.Management
Contiene clases para instrumentar y monitorizar procesos con la tecnología WMI (Windows Management Instrumentation)

System.Messaging
Contiene clases para usar las colas de mensajes del paquete MSMQ (Microsoft Message Queues)

System.Net → Comunicaciones en SDs
Contiene clases para usar fácilmente protocolos de comunicación de redes

System.Runtime.InteropServices
Contiene clases que posibilitan la interoperabilidad del código administrado con el no administrado

System.Runtime.Remoting → Básico para desarrollar SDs
Proporciona clases e interfaces que permiten crear y configurar aplicaciones distribuidas basadas en objetos que interactúan entre ellos

System.Runtime.Serialization
Contiene clases que permiten la serialización y deserialización de objetos

System.Security
Contiene clases que permiten utilizar la infraestructura de seguridad del CLR

Espacios de nombres (III)

System.ServiceProcess

Contiene clases para implementar, instalar y controlar las aplicaciones del tipo Servicios de Windows

System.Text

Contiene clases que representan codificaciones de caracteres y otras para realizar conversiones

System.Timer

Proporciona acceso a temporizadores

System.Web

Contiene clases que permiten la comunicación entre un explorador (cliente) y un servidor Web

System.Web.Services → Desarrollo servicios Web (RPC-XML)

Contiene clases para crear servicios Web XML usando ASP.NET y clientes que los usan

System.Web.UI → Desarrollo de interfaz de servidores Web

Contiene clases e interfaces para crear páginas de servidor activas ASP.NET y sus controles que constituyen la interfaz de usuario de las aplicaciones basadas en servidores Web

System.Windows.Forms

Contiene clases para crear Aplicaciones de Ventanas (ó de Windows) que pueden utilizar todos los recursos de interfaz de usuario proporcionados por el SO Windows

System.Xml

Contiene clases para procesar texto en formato XML

Sistemas distribuidos y de tiempo real

1. - Introducción a .NET Framework