UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Introducción al Lenguajes C#

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Agosto de 2017

HISTORIA DEL LENGUAJE C#.NET

HISTORIA DE C#.NET

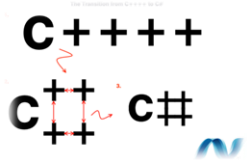
- C# (leído en inglés "C Sharp" y en español "C Almohadilla") es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET en el año de 1998
- Se empezó a elaborar como COOL y el año 2000 se libera la especificación de C#.
- Sus principales creadores son:
  - **Scott Wiltamuth y**
  - **Anders Hejlsberg** (también fue el diseñador del lenguaje Turbo Pascal y la herramienta RAD Delphi).

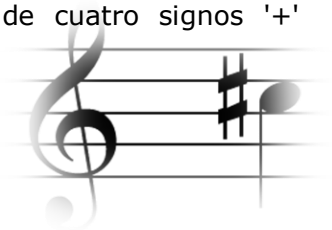
Anders Hejlsberg, arquitecto principal del lenguaje C#

Introducción al Lenguaje C# • FIUAS

HISTORIA DE C#.NET

- El nombre C Sharp fue inspirado por la notación musical, donde ♯ (sostenido, en inglés *sharp*) indica que la nota (*C* es la nota do en inglés) es un semitono más alta, sugiriendo que C♯ es superior a C/C++.
- C♯, como parte de la plataforma.NET, está normalizado por ECMA (*European Computer Manufacturers Association*) desde diciembre de 2001.
- Además, el signo '#' se compone de cuatro signos '+' pegados.





HISTORIA DE C#.NET

- C# es el único Lenguaje que ha sido diseñado especificamente para ser utilizado en la plataforma .NET,
- Programar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes incluidos en la plataforma .NET, ya que,
- C# carece de elementos heredados innecesarios en .NET. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET
- El .NET Framework, es la única parte que puede considerarse terminada, hasta el punto de que el propio Visual Studio .NET ha sido construido al 90% en C# y el 10% restante en C++.

Introducción al Lenguaje C# • FIUAS

HISTORIA DE C#.NET

- La sintaxis y estructuración de C# es muy similar a la C++, ya que la intención de Microsoft con C# es:
  - Facilitar la migración de códigos escritos en estos lenguajes a C# y
  - Facilitar su aprendizaje a los desarrolladores habituados a ellos.
- Sin embargo, su sencillez y el alto nivel de productividad son equivalentes a los de Visual Basic.

Historia de C#

- Un lenguaje que hubiese sido ideal utilizar para estos menesteres es Java, pero debido a problemas con la empresa creadora del mismo -Sun-, Microsoft ha tenido que desarrollar un nuevo lenguaje que se añada a las ya probadas virtudes de Java, las modificaciones que Microsoft tenía pensado añadirle para mejorarlo aún más y hacerlo un lenguaje orientado al desarrollo de componentes.

Introducción al Lenguaje C# • FIUAS

HISTORIA DE C#.NET

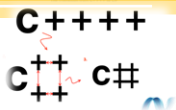
- En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como:
  - Visual Basic,
  - Pascal , Delphi
  - Java o C++ y las combina en uno solo.
- El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la Base Class Library (BCL) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el *.NET Framework SDK.*

Características de c#

- **Sencillez**
- **Modernidad**
- **Orientación a objetos**
- **Orientación a componentes**
- **Gestión automática de memoria**
- **Seguridad de tipos**
- **Instrucciones seguras**
- **Sistema de tipos unificado**
- **Extensibilidad de tipos básicos**
- **Extensibilidad de operadores**
- **Extensibilidad de modificadores**
- **Versionable**
- **Eficiente**
- **Compatible**

Introducción al Lenguaje C# • FIUAS

Sencillez



- C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Por ejemplo:
  - El código escrito en C# es **autocontenido**, lo que significa que no necesita de archivos adicionales al propio fuente tales como ficheros de cabecera o archivos IDL
  - No se incluyen elementos poco útiles de lenguajes como C++ tales como macros, herencia múltiple o la necesidad de un operador diferente del punto (**.**) acceder a miembros de espacios de nombres (**::**)

Modernidad

- C# incorpora en el propio lenguaje elementos que son muy útiles para el desarrollo de aplicaciones y que Java o C++ tienen algo simular, como un tipo básico **decimal** que permita realizar operaciones de alta precisión con reales de 128 bits (muy útil en el mundo financiero), la inclusión de una instrucción **foreach** que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico **string** para representar cadenas o la distinción de un tipo **bool** específico para representar valores lógicos.

Introducción al Lenguaje C# • FIUAS

Orientación a objetos

- C# es un lenguaje orientado a objetos.
- Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++ es que el de *C# es más puro en tanto que no admiten ni funciones ni variables globales* sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la **legibilidad del código.**
- C# soporta todas las características propias del paradigma de programación orientada a objetos: **encapsulación**, **herencia** y **polimorfismo**.

Orientación a objetos

- **Encapsulación:**
  - Contienen los modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a cuya definición precede sólo puede accederse desde su mismo ensamblado.
- **Herencia:**
  - A diferencia de C++ y al igual que Java- C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades

Introducción al Lenguaje C# • FIUAS

Orientación a componentes

- Los componentes proporcionan código reutilizable en forma de objetos.
- La sintaxis de C# incluye elementos propios del diseño de componentes.
- La sintaxis de C# permite definir cómodamente:
  - **Propiedades** (similares a campos de acceso controlado),
  - **Eventos**(asociación controlada de funciones de respuesta a notificaciones) o
  - **Atributos** (información sobre un tipo o sus miembros)

Gestión automática de memoria

- Todo lenguaje de .NET tiene a su disposición el **recolector de basura del CLR**. Esto tiene el efecto en el lenguaje de que no es necesario incluir instrucciones de destrucción de objetos.
- Sin embargo, dado que la destrucción de los objetos a través del **recolector de basura es indeterminista** y sólo se realiza cuando éste se active –ya sea por falta de memoria, finalización de la aplicación o solicitud explícita en el fuente.
- C# también proporciona un mecanismo de liberación de recursos determinista a través de la instrucción **using**.

Introducción al Lenguaje C# • FIUAS

Seguridad de tipos

- C# incluye mecanismos que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar. Para ello se toman ciertas medidas como:
  - ✓ Sólo se admiten **conversiones entre tipos compatibles**.
  - ✓ No se pueden usar **variables no inicializadas**.
  - ✓ Se comprueba que todo **acceso a los elementos de una tabla** se realice con índices que se encuentren dentro del rango de la misma.
  - ✓ Se puede controlar la **producción de desbordamientos** en operaciones aritméticas, informándose de ello con una excepción cuando ocurra.
  - ✓ A diferencia de Java, C# incluye **delegados**, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos,
  - ✓ Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

Instrucciones seguras

- Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes.
- Por ejemplo, la semántica de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador de igualdad (**==**) con el de asignación (**=**); y todo caso de un **switch** ha de terminar en un **break** o **goto** que indique cuál es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

Introducción al Lenguaje C# • FIUAS

Sistema de tipos unificado

- A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base común llamada **System.Object**, por lo que dispondrán de todos los miembros definidos en ésta clase (es decir, serán "objetos")

Extensibilidad de tipos básicos

- C# permite definir, a través de **estructuras**, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos.
- Es decir, que se puedan almacenar directamente en pila (luego su creación, destrucción y acceso serán más rápidos) y se asignen por valor y no por referencia.

Extensibilidad de operadores

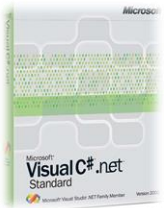
- Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores:
  - **++** y **--**
- El compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postfija; y
- Definiendo operadores simples (como **+**), el compilador deduce cómo aplicar su versión de asignación compuesta (+=).
- Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas

Extensibilidad de modificadores

- C# ofrece, a través del concepto de **atributos**, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente información adicional a la generada por el compilador que luego podrá ser consultada en tiempo ejecución a través de la librería de reflexión de .NET .
- Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

Versionable

- C# incluye una **política de versionado** que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijos previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos.



Eficiente

- En C# todo código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros.
- Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros.
- Para ello basta marcar regiones de código como inseguras (modificador **unsafe**) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad procesamiento muy grandes.

Introducción al Lenguaje C# • FIUAS

Compatible

- Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente en código escrito en C# fragmentos de código escrito en estos lenguajes, sino que el CLR también ofrece, a través de los llamados **PlatformInvocationServices** (**PInvoke**)
- También es posible acceder desde código escrito en C# a objetos COM. Para facilitar esto, el *.NET Framework SDK* incluye una herramientas llamadas **tlbimp** y**regasm** mediante las que es posible generar automáticamente clases proxy que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM.
- Finalmente, también se da la posibilidad de usar controles ActiveX desde código .NET y viceversa. Para lo primero se utiliza la utilidad **aximp**, mientras que para lo segundo se usa la ya mencionada **regasm**.

Aplicaciones que se pueden crear con C#

- Cliente de Windows tradicionales,
- Servicios Web XML,
- Componentes distribuidos,
- Aplicaciones cliente-servidor,
- Aplicaciones de base de datos,
- y mucho, mucho más.

.NET FRAMEWORK

C#

Los programas de C# se ejecutan en **.NET Framework:**

- Es un componente que forma parte de Windows y
- Incluye un sistema de ejecución virtual denominado Common Language Runtime (CLR) y
- Un conjunto unificado de bibliotecas de clases. CLR es la implementación comercial de Microsoft de CLI (Common Language Infrastructure),
- Un estándar internacional que constituye la base para crear entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin ningún problema.

¿Qué es el .NET Framework?

- Es el componente fundamental de la plataforma Microsoft .NET, necesario tanto para poder desarrollar aplicaciones como para poder ejecutarlas luego en entornos de prueba o producción.
- **Incluye:**
  - Entorno de Ejecución (Runtime)
  - Bibliotecas de Funcionalidad reutilizable (Class Library)
  - El motor de generación de interfaz de usuario (web y windows)
- Se distribuye en forma libre y gratuita.
- Está instalado por defecto en Windows.

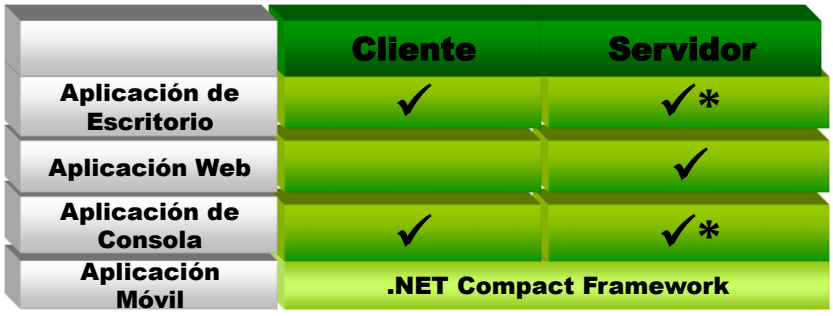
¿Qué es el .NET Framework?

- Existen tres variantes principales:
  - **.NET Framework Redistributable Package**, mínimo componente de la plataforma .NET que se necesita para poder ejecutar aplicaciones, que se instala en los entornos productivos, una vez que el desarrollo y las pruebas de la aplicación han finalizado.
  - **.NET Framework SDK**, contiene herramientas de desarrollo de línea de comandos (compiladores, depuradores, etc.), documentación de referencia, ejemplos y manuales para desarrolladores de aplicaciones(útil a los programadores)
  - **.NET Compact Framework**, versión reducida del .NET Framework Redistributable, especialmente pensada para ser instalada en dispositivos moviles como Pocket PC's y SmartPhones.

¿Dónde instalar el .NET Framework?

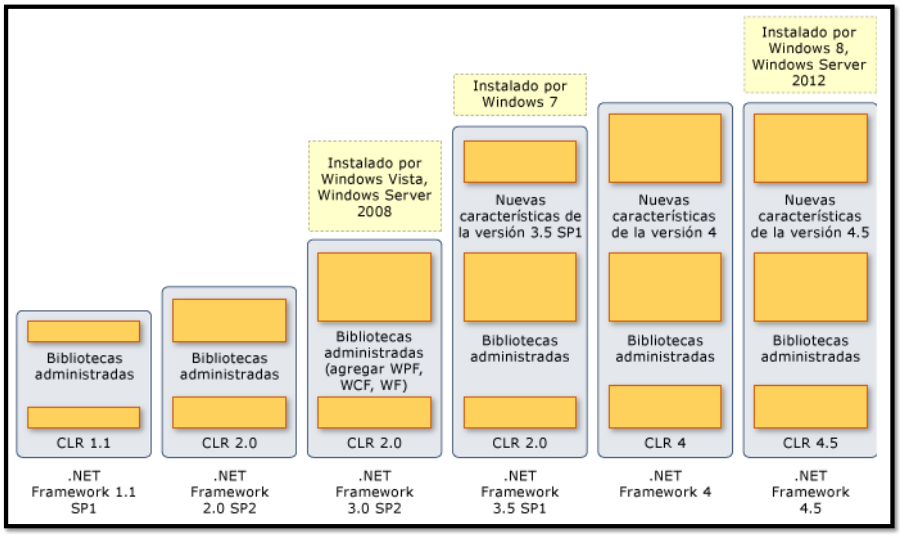
Debe estar instalado en cualquier dispositivo de hardware para que la ejecución de una aplicación .NET sea posible.

.NET Framework puede ser instalado en cualquier sistema operativo de la familia Windows, superior a Windows 98.



*** Sólo si la aplicación es distribuída**

Versiones y dependencias de .NET Framework



.NET 4.6 **Visual Studio 2015** Net 4.6.1 **Windows 10**

Compilación de un Programa Fuente

- El código fuente escrito en C# se compila en un lenguaje intermedio (IL) conforme con la especificación CLI.
- El código de lenguaje intermedio y recursos tales como mapas de bits y cadenas se almacenan en disco en un archivo ejecutable denominado ensamblado, cuya extensión es .exe o .dll generalmente.
- Un ensamblado contiene un manifiesto que proporciona información sobre los tipos, la versión, la referencia cultural y los requisitos de seguridad del ensamblado.

Cuando se ejecuta un programa de C#

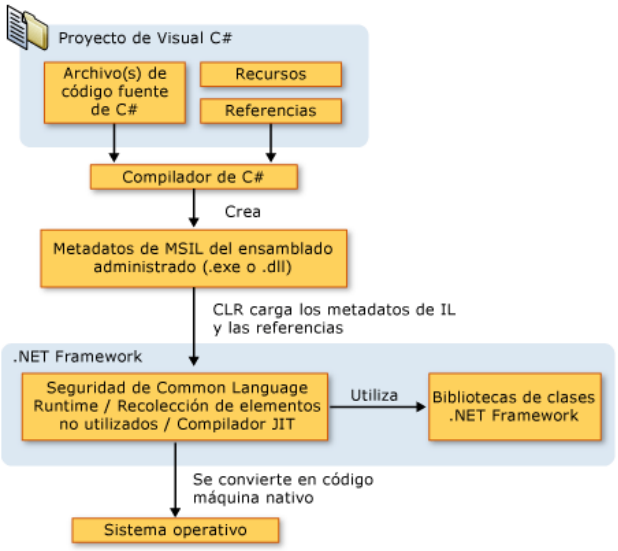
- El ensamblado se carga en CLR, con lo que se pueden realizar diversas acciones en función de la información del manifiesto.
- A continuación, si se cumplen los requisitos de seguridad, CLR realiza una compilación Just In Time (JIT) para convertir el código de lenguaje intermedio en instrucciones máquina nativas.
- CLR también proporciona otros servicios relacionados con la recolección de elementos no utilizados automática, el control de excepciones y la administración de recursos.

Introducción al Lenguaje C# • FIUAS

Cuando se ejecuta un programa de C#

- El código ejecutado por CLR se denomina algunas veces "código administrado", en contraposición al "código no administrado" que se compila en lenguaje máquina nativo destinado a un sistema específico.
- En el diagrama siguiente se muestran las relaciones en tiempo de compilación y tiempo de ejecución de los archivos de código fuente de C#, las bibliotecas de clases de .NET Framework, los ensamblados y CLR

Arquitectura de la plataforma .NET Framework

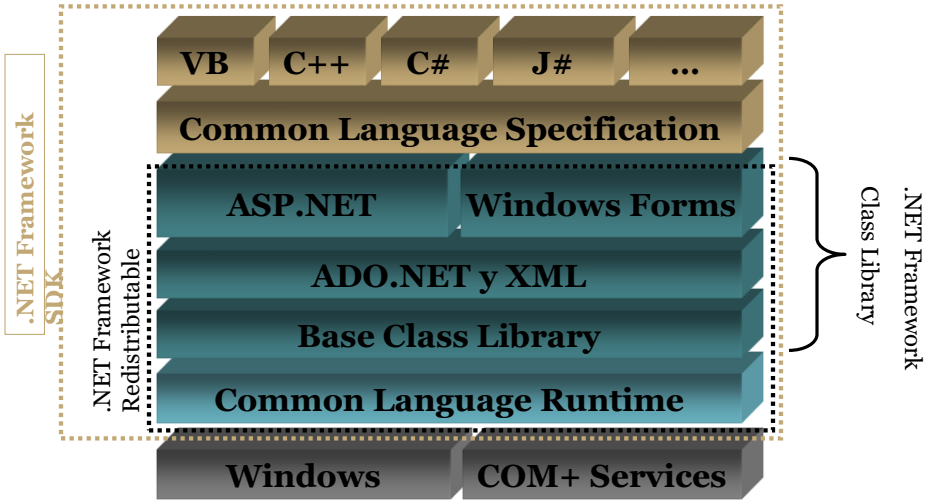


Introducción al Lenguaje C# • FIUAS

Temas a Tratar

- Componentes Fundamentales
  - Arquitectura del .NET Framework
  - CLR – Arquitectura de Ejecución de Aplicaciones
  - CLR – Common Language Runtime
  - CLR – Componentes Internos
  - CLR – Procesos de Compilación
  - CLR – Microsoft Intermediate Language (MISIL)
  - ¿Qué es un Assemblies?
  - Assemblies – Aplicaciones .NET
  - .NET Framework Class Library
  - Common Language Specification (CLS)
  - CLS – Elección del Lenguaje

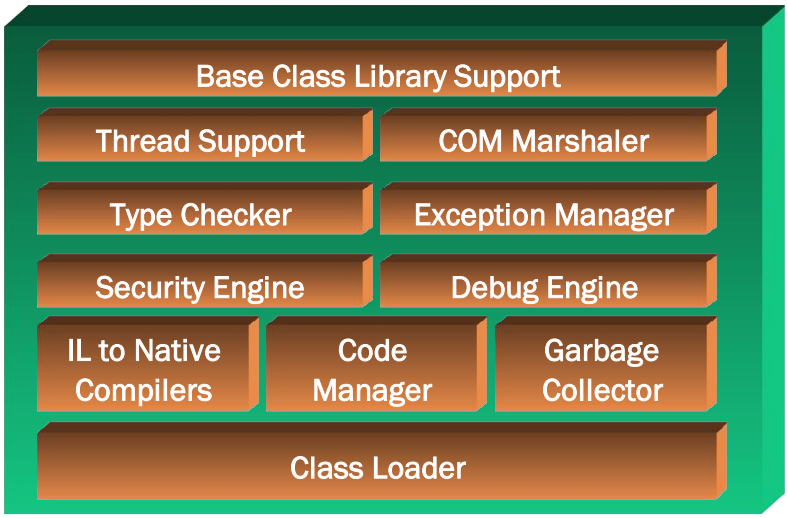
Arquitectura del .NET Framework



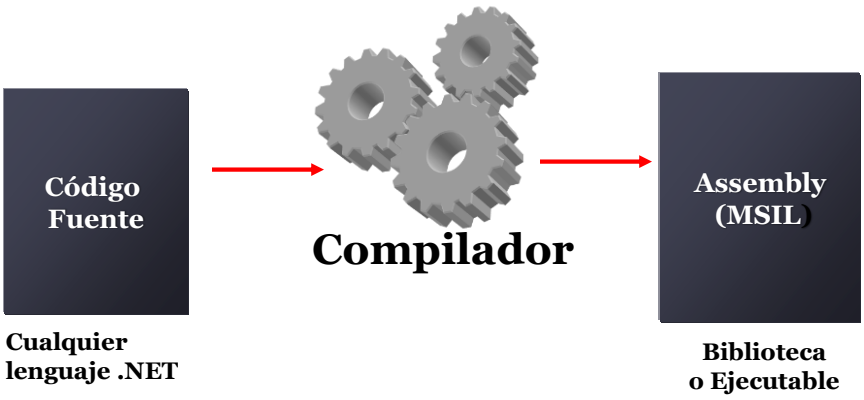
CLR - Common Language Runtime

- El CLR es el motor de ejecución (runtime) de .NET
  - Características
  - Compilación Just-In-Time (JIT)
  - Gestión automática de memoria (Garbage Collector)
  - Gestión de errores consistente (Excepciones)
  - Ejecución basada en componentes (Assemblies)
  - Gestión de Seguridad
  - Multithreading (multi-hilos)

CLR - Componentes Internos



CLR - Proceso de Compilación



CLR - MSIL

```
.method private hidebysig static void Main(string[] args) cil
      managed {
.entrypoint
maxstack 8
L_0000: ldstr "Hola Mundo"
L_0005: call void
      [mscorlib]System.Console::WriteLine(string)
L_000a: ret
}
```


¿Qué es un "Assembly"?

- Un Assembly es la unidad mínima de ejecución, distribución, instalación y versionado de aplicaciones .NET

Assemblies - Aplicaciones .NET

- Uno o más Assemblies
- Al ejecutar una aplicación, ¿cómo ubico los assemblies necesarios?
  - El Class Loader busca en el directorio local (preferido)
  - Global Assembly Cache (GAC)
- Diferentes aplicaciones pueden usar diferentes versiones
  - Actualizaciones más simples
  - Desinstalación más simple

.NET Framework Class Library

- Conjunto de Tipos básicos (clases, interfaces, etc.) que vienen incluídos en el .NET Framework
- Los tipos están organizados en jerarquías lógicas de nombres, denominados NAMESPACES
- Los tipos son INDEPENDIENTES del lenguaje de desarrollo
- Es extensible y totalmente orientada a objetos

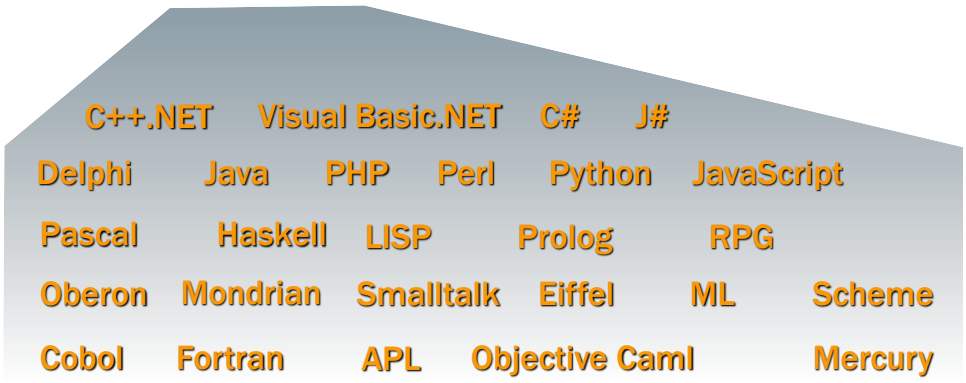
.NET Framework Class Library

Common Language Specification (CLS)

- Especificación que estandariza una serie de características soportadas por el CLR
- Contrato entre diseñadores de lenguajes de programación y autores de bibliotecas
- Permite la interoperabilidad entre lenguajes
- Microsoft provee implementaciones
  - Microsoft Visual Basic .NET
  - Microsoft Visual C# .NET
  - Microsoft Visual J#.NET
  - Microsoft Visual C++.NET

Common Language Specification (CLS)

- El resto de la industria y el sector académico han desarrollado más de 20 lenguajes compatibles con la especificación CLS



Introducción al Lenguaje C# • FIUAS

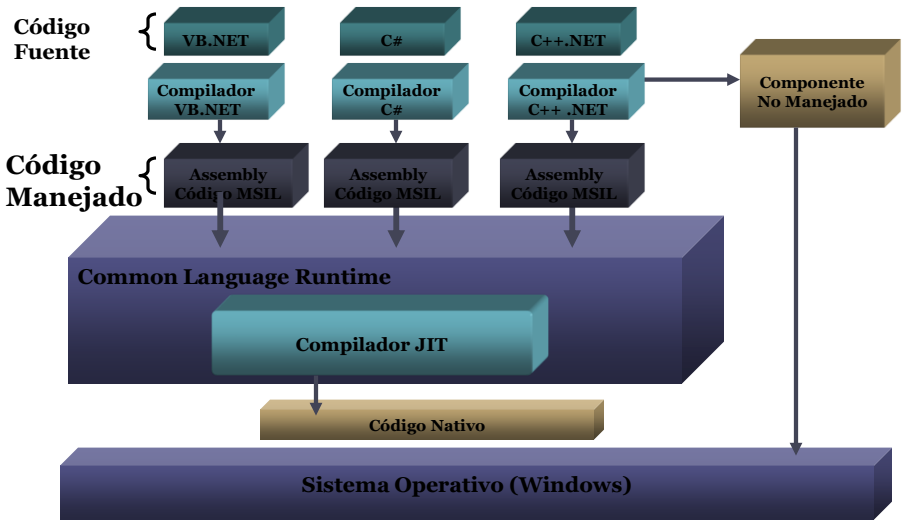
CLS - Elección del lenguaje

- .NET posee un único runtime (el CLR) y un único conjunto de bibliotecas para todos los lenguajes
- No hay diferencias notorias de performance entre los lenguajes provistos por Microsoft
- El lenguaje a utilizar, en gral., dependerá de su experiencia previa con otros lenguajes o de gustos personales:
  - Si conoce Java, Delphi, C++, etc. → C#
  - Si conoce Visual Basic o VBScript → VB.NET
- Los tipos de aplicaciones .NET son INDEPENDIENTES del lenguaje que elija

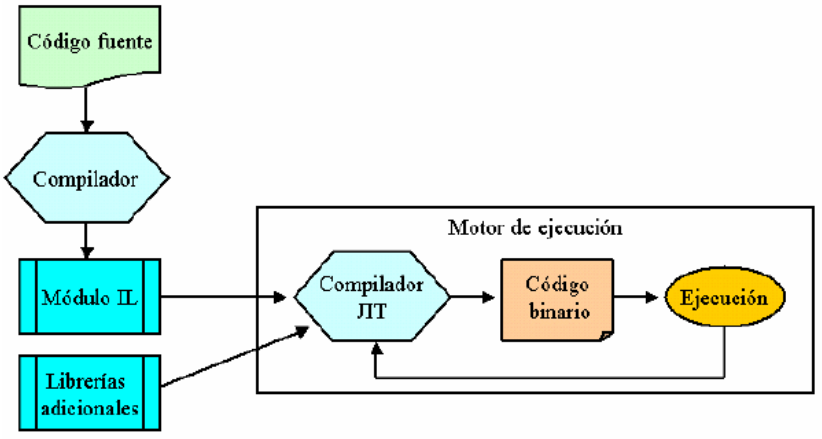
Temas a Tratar

- Funcionamiento Interno del CLR
  - Modelo de ejecución del CLR
  - Common Type System (CTS)
  - La memoria y los tipos de datos

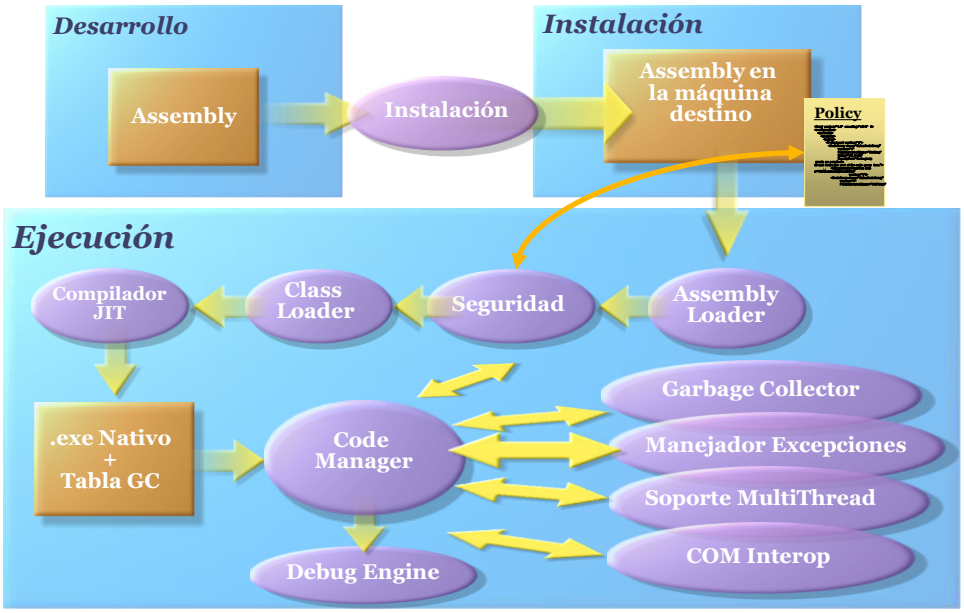
Modelo de Ejecución del CLR



Esquema con el proceso de compilación llevado a cabo por el compilador Just-In-Time (JIT)



Modelo de Ejecución del CLR



El CTS, Common Type System

El Sistema Común de Tipos o CTS (Common Type System), es el mecanismo del CLR que permite definir el modo en que los tipos serán creados y manipulados por el entorno de ejecución de .NET Framework.

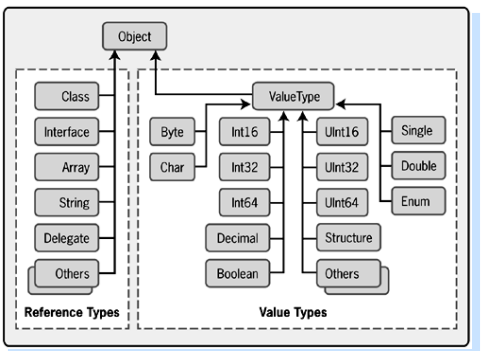
Entre las funcionalidades que comprende destacan:

- La integración de código escrito en diferentes lenguajes;
- Optimización del código en ejecución;
- Un modelo de tipos orientado a objeto, que soporta múltiples lenguajes; y
- Una serie de normas que aseguran la intercomunicación entre objetos.

CTS (Common Type System)

- Define un conjunto común de "tipos" de datos orientados a objetos
- Todo lenguaje de programación .NET debe implementar los tipos definidos por el CTS

- **Todo tipo hereda directa o indirectamente del tipo System.Object**
- **Define Tipos de VALOR y de REFERENCIA**



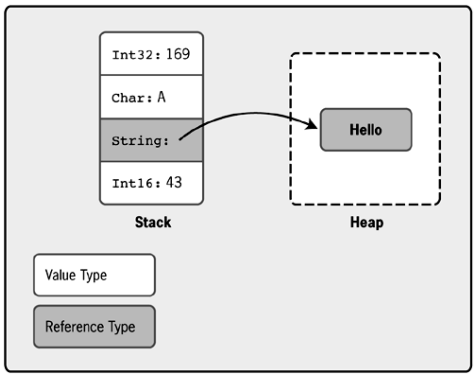
Ventajas del CTS

Desde un lenguaje como VB.NET, podemos instanciar un objeto de una clase escrita en otro lenguaje como C#; y al hacer una llamada a uno de los métodos del objeto, no es necesario realizar conversiones de tipos en los parámetros del método, funcionando todo de forma transparente.

La Memoria y los Tipos de Datos

- El CLR administra dos segmentos de memoria: **Stack (Pila)** y **Heap (Montón)**
- El **Stack** es liberado automáticamente y el **Heap** es administrado por el **GC** (**Garbage Collector**)

- **Los tipos VALOR se almacenan en el Stack**
- **Los tipos REFERENCIA se almacenan en el Heap**



Temas a Tratar

- Ventajas de .NET
  - Ventajas de .NET
  - Desarrollo simplificado
  - Entorno de ejecución robusto y seguro
  - Independencia del lenguaje
  - Extensibilidad
  - Interoperabilidad

Introducción al Lenguaje C# • FIUAS

Ventajas de .NET

- Unifica los modelos de programación
- Simplifica aún más el desarrollo
- Provee un Entorno de Ejecución robusto y seguro
- Es independiente del lenguaje de programación
- Interoperabilidad con código existente
- Simplifica la instalación y administración de las aplicaciones
- Es Extensible

Desarrollo Simplificado

- Alto nivel de abstracción
  - No mas accesos COM a bajo nivel
  - Orientado a Objetos desde el Núcleo
- Sistema de tipos unificado (CTS)
  - Todo es un objeto, no mas variants
- Componentes de Software
  - Propiedades, métodos, eventos, y atributos incluidos en la construcción de clases
- API organizada en forma Jerárquica

Entorno de Ejecución Robusto y Seguro

- Gestión automática de la memoria
  - Todos los objetos son administrados por el Garbage Collector
- Manejo de Excepciones
- Fuertemente tipado
  - Solo casteos seguros
  - Inicialización de variables obligatoria
- Instalación con Cero Impacto
  - No requiere registración en la Registry

Independencia del lenguaje

- Libertad en la elección del lenguaje
  - Todas las facilidades de la plataforma .NET están disponibles a todos los lenguajes de programación .NET
  - Los componentes de una aplicación .NET pueden ser escritos en distintos lenguajes de alto nivel compatibles con la plataforma
- Herramientas compartidas
  - Debuggers, profilers, analizadores de código, y otras trabajan para todos los lenguajes

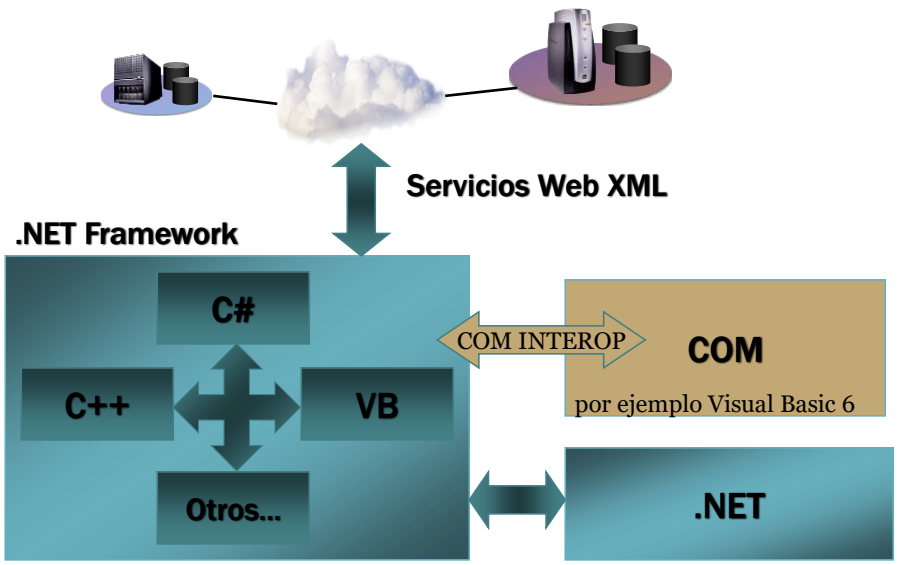
Instalación y Administración más simples

- Unidades de Ensamblado ("Assemblies")
  - Mínima unidad de distribución, versionado y administración de seguridad de aplicaciones .NET
  - Auto-descriptas a través de un manifiesto ("manifest")
- Instalaciones Cero-impacto
  - Aplicaciones y componentes pueden ser compartidas o privadas
- Versioning
  - Múltiples versiones del mismo componente pueden co-existir, aún en el mismo proceso

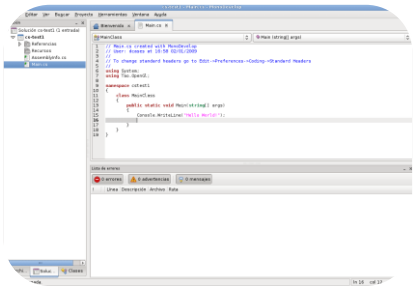
Extensibilidad

- El Framework no es una "caja negra"
- Sus clases pueden ser extendidas a través del mecanismo de herencia
  - A diferencia de COM, usamos y extendemos las clases en si mismas, no un "wrapper"
- Herencia entre distintos lenguajes

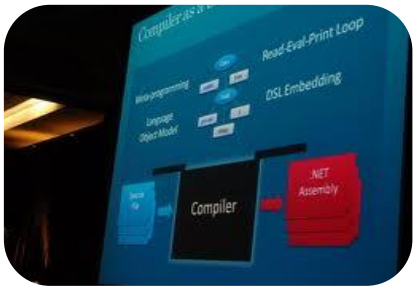
Interoperabilidad con otras aplicaciones



Herramientas de programación en C#



Entornos de desarrollo

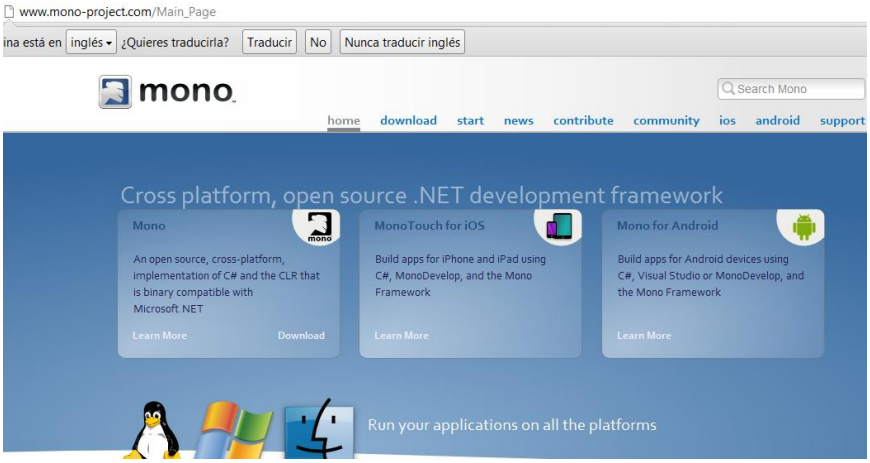


Compilador

Entornos de Desarrollo

- **Visual Studio**
- **SharpDevelop**
- **Mono**
- **MonoDevelop**
- **Eclipse y Emonic**
- **Notepad ++**

mono



Introducción al Lenguaje C# • FIUAS

Preguntas?