Introducción a Microsoft .NET

Componentes Fundamentales

Bibliotecas Principales

Ventajas de .NET

Herramientas de Desarrollo .NET

Novedades en .NET 2.0

¿Qué es .NET?

.NET Como evolución de COM

**¿Qué es .NET?**

Microsoft .NET es una plataforma de desarrollo y ejecución de aplicaciones. Esto quiere decir que no sólo nos brinda todas las herramientas y servicios que se necesitan para desarrollar modernas aplicaciones empresariales y de misión crítica, sino que también nos provee de mecanismos robustos, seguros y eficientes para asegurar que la ejecución de las mismas sea óptima. Los componentes principales de la plataforma .NET son:

* Un entorno de ejecución de aplicaciones, también llamado “Runtime”, que es un componente de software cuya función es la de ejecutar las aplicaciones .NET e interactuar con el sistema operativo ofreciendo sus servicios y recursos.
* Un conjunto de bibliotecas de funcionalidades y controles reutilizables, con una enorme cantidad de componentes ya programados listos para ser consumidos por otras aplicaciones.
* Un conjunto de lenguajes de programación de alto nivel, junto con sus compiladores y linkers, que permitirán el desarrollo de aplicaciones sobre la plataforma .NET.
* Un conjunto de utilitarios y herramientas de desarrollo para simplificar las tareas más comunes del proceso de desarrollo de aplicaciones
* Documentación y guías de arquitectura, que describen las mejores prácticas de diseño, organización, desarrollo, prueba e instalación de aplicaciones .NET

A lo largo de esta presentación se analizará cada uno de estos puntos con mayor detalle.

Por otra parte, .NET representa la evolución COM (Component Object Model), la plataforma de desarrollo de Microsoft anterior a .NET y sobre la cual se basaba el desarrollo de aplicaciones Visual Basic 6 (entre otros tantos lenguajes y versiones).

**Características de .NET**

Describiremos a continuación algunas de las características principales de la plataforma Microsoft .NET:

* Se dice que es una plataforma de ejecución intermedia, ya que las aplicaciones .NET no son ejecutadas directamente por el sistema operativo, como ocurre en el modelo tradicional de desarrollo. En su lugar, las aplicaciones .NET están diseñadas para ser ejecutadas contra un componente de software llamado Entorno de Ejecución (muchas veces también conocido como “Runtime”, o , “Máquina Virtual”). Este componente es el encargado de manejar el ciclo de vida de cualquier aplicación .NET, iniciándola, deteniéndola, interactuando con el Sistema Operativo y proveyéndole servicios y recursos en tiempo de ejecución.
* La plataforma Microsoft .NET está completamente basada en el paradigma de Orientación a Objetos (para más información acerca de este tema puede consultar el material de estudio de la Estrella 0 del programa)
* .NET es multi-lenguaje: esto quiere decir que para poder codificar aplicaciones sobre esta plataforma no necesitamos aprender un único lenguaje específico de programación de alto nivel, sino que se puede elegir de una amplia lista de opciones. Veremos este tema con mayor detalle más adelante en la presentación.
* .NET es una plataforma que permite el desarrollo de aplicaciones empresariales de misión crítica, entendiéndose por esto que permite la creación y ejecución de aplicaciones de porte corporativo que sean críticas para la operación de tipos variados de organizaciones. Si bien también es muy atrayente para desarrolladores no profesionales, estudiantes y entusiastas, su verdadero poder radica en su capacidad para soportar las aplicaciones más grandes y complejas.
* Net fue diseñado de manera tal de poder proveer un único modelo de programación, uniforme y consistente, para todo tipo de aplicaciones (ya sean de formularios Windows, de consola, aplicaciones Web, aplicaciones móviles, etc.) y para cualquier dispositivo de hardware (PC’s, Pocket PC’s, Teléfonos Celulares Inteligentes, también llamados “SmartPhones”, Tablet PC’s, etc.). Esto representa un gran cambio con respecto a las plataformas anteriores a .NET, las cuales tenían modelos de programación, bibliotecas, lenguajes y herramientas distintas según el tipo de aplicación y el dispositivo de hardware.
* Uno de los objetivos de diseño de .NET fue que tenga la posibilidad de interactuar e integrarse fácilmente con aplicaciones desarrolladas en plataformas anteriores, particularmente en COM, ya que aún hoy existen una gran cantidad de aplicaciones desarrolladas sobre esa base.
* .NET no sólo se integra fácilmente con aplicaciones desarrolladas en otras plataformas Microsoft, sino también con aquellas desarrolladas en otras plataformas de software, sistemas operativos o lenguajes de programación. Para esto hace un uso extensivo de numerosos estándares globales que son de uso extensivo en la industria, y acerca de los cuales iremos aprendiendo a lo largo del curso. Algunos ejemplos de estos estándares son XML, HTTP, SOAP, WSDL y UDDI.

**NET como evolución de COM**

Entorno de Ejecución (Runtime)

* + COM: Windows
  + .NET: Common Language Runtime

Librerías de Funcionalidad

* + COM: Algunas (ADO, FSO, etc.)
  + .NET: Muy extensa (.NET Framework Class Library)

Lenguajes de Programación

* + COM: VB, C++, VFP, ASP, J++
  + .NET: Common Language Specification

Entorno de Desarrollo (IDE)

* + COM: Uno para cada lenguaje

.NET: Uno independiente del lenguaje (VS.NET

**¿Qué es el .NET Framework?**

El .NET Framework (traducido como “Marco de Trabajo”) 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.

El .NET framework tiene tres variantes principales, todas descargables gratuitamente desde Internet

* + .NET Framework Redistributable Package: este es el mínimo componente de la plataforma .NET que se necesita para poder ejecutar aplicaciones. Normalmente ésta es la variante que se instala en los entornos productivos, una vez que el desarrollo y las pruebas de la aplicación han finalizado.

Está compuesto por:

* + - El entorno de ejecución de la plataforma .NET
    - Las bibliotecas de funcionalidad reutilizable
  + .NET Framework SDK: esta versión contiene herramientas de desarrollo de línea de comandos (compiladores, depuradores, etc.), documentación de referencia, ejemplos y manuales para desarrolladores de aplicaciones. Normalmente ésta variante se instala en los entornos de desarrollo de aplicaciones, y es más útil a los programadores que a los usuarios finales. Para poder instalar la versión SDK (Software Development Kit) es necesario instalar previamente el Redistributable Package.
  + .NET Compact Framework: esta es una versión reducida del .NET Framework Redistributable, especialmente pensada para ser instalada en dispositivos móviles como Pocket PC’s y SmartPhones.

El .NET Framework puede ser instalado en cualquier sistema operativo de la familia Windows superior a Windows 98. Para más información acerca de los prerrequisitos se puede consultar:

**Acceso a Bases de Datos Relacionales Escenario Conectado**

En la actualidad se plantean dos tipos de escenarios de acceso a bases de datos relacionales. El primero de ellos es el que se conoce como “Escenario Conectado”, ya que en él se requiere una conexión física establecida con el servidor de datos durante todo momento para poder efectuar cualquier consulta o actualización sobre los datos.

Esto tiene algunas ventajas y también sus desventajas.

Algunas Ventajas:

* Al haber una única conexión a la base de datos por usuario, o incluso a veces por aplicación, establecida permanentemente, puede llegar a resultar más sencillo administrar la seguridad y el acceso al servidor de datos. Lo mismo ocurre con el control de concurrencia: en un escenario donde múltiples usuarios se estuvieran conectando y desconectando permanentemente para realizar distintas acciones, este control sería más difícil de llevar.
* Siempre la aplicación tiene acceso a los datos actualizados

Algunas Desventajas:

* Se requiere una conexión abierta todo el tiempo con el servidor de base de datos, lo cual consume recursos innecesariamente si no se la está utilizando.
* La escalabilidad del acceso a los datos se ve limitada por la cantidad de conexiones establecidas simultáneamente contra el servidor de base de datos.

**Acceso a Bases de Datos Relacionales Escenario Desconectado**

El segundo escenario de acceso a bases de datos relacionales se conoce como “Escenario Desconectado”, ya que en él una parte de los datos del servidor central se copia localmente y puede luego ser consultada y actualizada sin contar con una conexión abierta. Luego si se desea puede establecerse una conexión con el servidor de base de datos para sincronizar los cambios efectuados sobre la copia local y actualizar los datos. Este tipo de funcionalidad es particularmente útil para escenarios de usuarios móviles, que salen de su oficina con una laptop, un SmartPhone o una PocketPC y desean poder continuar trabajando por más que no tengan conectividad física con el servidor de base de datos ubicado en la red interna de la empresa.

Algunas ventajas que provee un escenario de acceso a datos desconectado son:

* La posibilidad de trabajar sobre los datos independientemente del resto de los usuarios de la aplicación
* Mayor escalabilidad en el acceso a datos y utlización más óptima de recursos del servidor, ya que se mantiene en un mínimo indispensable la cantidad y duración de conexiones abiertas.
* Mayor performance, al trabajar con una copia local de los datos.

Algunas Desventajas:

* Puede ocurrir que en un momento dado un usuario no esté accediendo a los datos más actualizados del repositorio central
* Al momento de sincronizar los cambios efectuados localmente contra el repositorio central pueden surgir conflictos, los cuales deben ser resueltos manualmente.

ADO.NET soporta el acceso a datos tanto en escenarios conectados como desconectados.

**ADO.NET- Proveedores de Acceso a Datos**

Los proveedores de acceso a datos ADO.NET (conocidos como “Managed Data Providers”) representan conjuntos específicos de clases que permiten conectarse e interactuar con una base de datos, cada uno utilizando un protocolo particular. El .NET Framework incluye cuatro proveedores de acceso a datos, que en conjunto le permiten conectarse e interactuar virtualmente con cualquier base de datos existente en la actualidad:

* Data Provider For SQL Server: es el proveedor de acceso nativo a servidores de bases de datos Microsoft SQL Server 7.0 o superior, y Microsoft Access. Al conectarse via protocolos nativos de bajo nivel, povee la alternativa más performante para conexiones contra estos motores de bases de datos. Sus clases se encuentran en el namespace System.Data.SqlClient.
* Data Provider For OLE DB: es el proveedor de acceso a datos que permite interactuar via el protocolo estándar OLE DB con cualquier repositorio de datos que lo soporte. Sus clases se encuentran en el namespace System.Data.OleDb.
* Data Provider For ODBC: es el proveedor de acceso a datos que permite interactuar via el protocolo estándar ODBC con cualquier repositorio de datos que lo soporte. Sus clases se encuentran en el namespace System.Data.Odbc.
* Data Porvider For Oracle: es el proveedor de acceso nativo a bases de datos Oracle, desarrollado por Microsoft utilizando las herramientas de conectividad de Oracle. De esta forma puede lograrse un acceso más performante a bases de datos Oracle desde aplicaciones .NET que utilizando ODBC u OLE DB. Sus clases se encuentran en el namespace System.Data.OracleClient, y están compiladas en un assembly diferente al resto: System.Data.OracleClient.dll.

ADO.NET provee una arquitectura extensible, posibilitando que terceras partes creen sus propios proveedores de acceso nativo para aplicaciones .NET. Algunos ejemplos de esto son:

* Data Provider For DB2, desarrollado por IBM
* Oracle Data Provider For .NET, desarrollado por Oracle
* Providers de acceso nativo a bases de datos OpenSource, como MySQL y PostgreSQL

Es importante volver a destacar que utilizando estos proveedores de acceso a datos cualquier aplicación .NET puede utilizar casi cualquier base de datos relacional existente en la actualidad como respositorio de información persistente (esto incluye, además de MS SQL Server, a IBM DB2, Oracle, Sybase, Informix, TeraData, MySQL y PostgreSQL, entre otras).

**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

Una de las principales ventajas de .NET es que unifica los modelos de programación, bibliotecas de funcionalidad y entornos de ejecución que existían anteriormente para distintos tipos de aplicaciones y distintos dispositivos.

Anteriormente a .NET existían lenguajes, bibliotecas, entornos de ejecución y herramientas de desarrollo distintas y específicas para cada tipo de aplicación y dispositivo (Visual Basic, Visual C++, ASP/VBScript, Embedded Visual C++, etc.).

.NET unifica todos esos modelos de programación ofreciendo una única API, un único entorno de ejecución, un único conjunto de bibliotecas y una única herramienta de desarrollo para cualquier tipo de aplicación.

Ofrece un modelo de desarrollo simplificado, basado en objetos que utilizan un sistema unificado de tipos de datos y se empaquetan en componentes reutilizables y auto descriptivos (los assemblies).

Otra de las principales ventajas con las que cuenta la plataforma .NET es su robusto entorno de ejecución (el CLR), que provee servicios a las aplicaciones en ejecución y maneja su ciclo de vida reforzando la seguridad y abstrayendo a los programadores de optimizaciones y manejos de memoria de bajo nivel.

Otra de los grandes beneficios de .NET es su soporte a múltiples lenguajes de programación, lo cual acelera la curva de aprendizaje de los desarrolladores permitiendo que cada uno elija en base a sus gustos personales. Además, la posibilidad de utilizar las mismas herramientas de programación y tener las mismas capacidades de acceso a la plataforma independientemente del lenguaje le proporcionan una flexibilidad sin precedentes.

.NET también simplifica, gracias al uso de assemblies auto-descriptivos, la instalación y administración de aplicaciones resolviendo gran parte de los problemas existentes en COM en lo que respecta la registración de componentes, manejo de múltiples versiones en paralelo y compatibilidad de aplicaciones.

Todas las clases incluidas en el .NET Framework son extensibles mediante los mecanismos de herencia propios de la orientación a objetos. Esto posibilita que funcionalidades o controles gráficos que no cumplan exactamente con una determinada necesidad pueden ser extendidos para agregarle o modificarle comportamiento sin tener que escribir todo el código nuevamente.

**Herramientas de Desarrollo .NET**

**Visual Studio 2005 Express Editions:**

Herramientas de desarrollo gratuitas

Muy livianas, fáciles de usar y de aprender

Diseñadas para entusiastas, estudiantes y desarrolladores principiantes

Hay varias ediciones, según el tipo de aplicación y el lenguaje

Tiene características avanzadas, por ejemplo:

* + Refactoring
  + Code Snippets
  + Diseñadores WYSIWYG
  + Depuradores

**SQL Server 2005 Express**

SQL Server 2005 Express Edition es parte de la familia de productos del más nuevo y sofisticado motor de bases de datos relacional de Microsoft: SQL Server 2005.

Al igual que su predecesor, el MSDE, esta herramienta es gratuita y royalty free (se puede embeber en aplicaciones comerciales sin pagar regalías a Microsoft). Esta edición express tiene ciertas limitaciones que la hacen inapropiada para soportar la operatoria de una organización, pero puede ser utilizada perfectamente como base de datos de escritorio, prueba o desarrollo. A lo largo del curso utilizaremos esta versión de SQL Server, por se de acceso libre y gratuito, aunque normalmente se podrán realizar los ejericios y ver las aplicaciones de ejemplo en cualquier producto de la familia de SQL Server 2005 (en caso contrario se indicará apropiadamente).