CAPÍTULO 1

Plataforma .NET 2.0

Veamos las novedades que ha traído esta nueva plataforma de desarrollo de Microsoft, enfocándonos no sólo en ASP.NET, sino en todas las herramientas y entornos que lo rodean, como Visual Studio 2005, SQL Server 2005 y el .NET Framework 2.0.

El nuevo paradigma

→ A fines de 2005, Microsoft lanzó la nueva versión de su plataforma de desarrollo: "**.NET 2.0**". Dentro del paquete de actualizaciones, existieron cambios para todos los gustos:

Nuevos entornos de desarrollo
- Visual Studio 2005
- Visual Web Developer 2005 Express
- Otros productos "Express": Visual Basic Express, Visual C# Express, Visual J# Express, Visual C++ Express.

Nueva base de datos
- SQL Server 2005
- SQL Server 2005 Express

Nuevos lenguajes
- Visual Basic.NET 7.0
- C# 2.0

Nuevos frameworks
- .NET Framework 2.0
- .NET Compact Framework 2.0

Y, por supuesto, el lanzamiento de su nueva plataforma de desarrollo Web: **ASP.NET 2.0**, con muchos cambios desde su primera versión. Analicemos primero los cambios en el resto de las áreas y, luego, nos metemos de lleno en ASP.

.NET FRAMEWORK 2.0

Lo nuevo

El nuevo framework trae bastantes novedades, entre las que podemos mencionar:

- Soporte nativo para plataformas 64 bits.
- Nueva Data **Protection API** para encriptar claves, información o strings de conexión.

- Soporte de Edit and Continue: una característica que era propia de Visual Basic 6 y había sido eliminada en .NET 1 por la cual podemos hacer modificaciones en el código mientras estamos haciendo debugging y seguir ejecutando sin comenzar nuevamente la compilación.
- Se han incorporado nuevas funcionalidades al namespace **System.Net**, por ejemplo, la posibilidad de trabajar como cliente FTP, hacer caching en HTTP y descubrimiento de proxies.
- Ahora existe la clase **Ping** para descubrir si un equipo está respondiendo en la red, a través de una red de tipo IP.
- Es posible crear un pequeño web server desde una aplicación .NET al utilizar la clase **HttpListener**. Esta opción no se relaciona con el Internet Information Server, sino que es un servicio propio de .NET.
- Se han incorporado clases a **System.IO.Compression** para leer y escribir datos comprimidos en el estándar **GZIP**.

Lo mejorado

El framework también mejoró algunas cuestiones ya existentes, como:

- Ahora soporta **SOAP 1.2** en **Web Services.**
- La plataforma y todos los lenguajes soportan tipos básicos **nullables** (que pueden ser nulos), como int o float.
- Ahora es posible enviar e-mails con más de un destinatario, con formatos alternativos de distribución (texto plano y HTML, por ejemplo) y adjuntos.
- Se han implementado nuevos tipos de excepciones que dan mayor información en **System.Security.SecurityException**.
- Se ha incorporado un nuevo procesador de transformación de XML (**XSLT**), ahora se soportan tipos en **XmlReader**, **XmlWriter** y **XPathNavigator** y existe modo edición en **XPathNavigator**.

LA HISTORIA DE ASP

Si bien todavía se están haciendo desarrollos en ASP 3 (ahora llamado ASP Clásico), ASP.NET es una tecnología que ya tiene más de 5 años. Su curva de aprendizaje y el uso de un nuevo paradigma retrasó la adaptación de la tecnología por parte de los desarrolladores y de las empresas.

Los lenguajes

→ Analicemos los cambios que han sufrido los dos lenguajes estándar de la plataforma en esta nueva versión. Recordemos que las características nuevas del framework afectan automáticamente a todos los lenguajes compatibles con .NET 2.0.

GENERICS

Las clases genéricas son muy similares a los conocidos **templates** de **C++**. Si ya los hemos utilizado en C++ no tendremos inconvenientes en entender el concepto de clases genéricas en .NET.

Introducen en el framework el concepto de **type parameters**, o parámetros de tipos. Esto hace posible diferir la especificación de uno o más tipos hasta que la clase o método esté declarado e instanciado en el código del cliente.

La gran ventaja de esta clase de tipos es que no se incurre en gastos de moldeado (**cast**) en tiempo de ejecución u operaciones de **boxing/unboxing**, que son un mayor trabajo de codificación y de ejecución por parte del entorno.

Características

- Los Generics permiten la creación de colecciones de clases de tipo seguro.
- A diferencia del resto de las colecciones (que son siempre de **Object**), no pueden mezclar tipos de datos distintos.
- Pueden ser obligadas a soportar sólo ciertos tipos de datos.
- La información de tipos utilizados puede ser obtenida en tiempo de ejecución usando **reflection**.

Esto nos da tres factores en el trabajo con colecciones: Reusabilidad, seguridad de tipos y eficiencia de ejecución.

La version 2.0 del Framework provee un nuevo namespace llamado **System.Collections.Generics**. Estas coleccio-

OTROS LENGUAJES

Recordemos que .NET es una plataforma independiente del lenguaje que utilicemos y que terceras empresas crearon compiladores de otros lenguajes para .NET, como Delphi, Cobol o hasta PHP.

nes son más eficientes que sus pares no generics, existentes en .NET 1.x (y todavía vigentes también en 2.0).

La siguiente tabla muestra alguna de las colecciones de la versión 1.1 y su "par" en **Generics**, utilizando primero la sintaxis de C#.

COLECCIONES	
NET 1.X .	**NET 2.0**
ArrayList	List<T>
Hashtable	Dictionary<K,V>
SortedList	SortedList<K,V>
Stack	Snack<T>

Aún hay más, para una lista completa ver System.Collections.Generics.

En la versión anterior, se lograba la generalización a costa del moldeado de tipos desde y hacia **System.Object**. Utilizando clases genéricas se pueden crear colecciones que son de tipo seguro en tiempo de compilación.

Ejemplo

Veamos ahora cómo, utilizando una clase **Persona**, conseguimos mejores resultados mediante una colección genérica que utilizando el viejo modelo.

Si hiciéramos la colección sin Generics, el código sería el siguiente:

```
// Instanciar un System.Collections.ArrayList
System.Collections.ArrayList a = new System.Collections.ArrayList();

// Agregar un objeto de tipo Persona al ArrayList
a.Add(new Persona("Gabriel", "Bulfon"));

// Obtener un objeto de tipo Persona desde el ArrayList haciendo casting
Persona p = (Persona) a[0];
```

Ahora hagamos lo mismo, pero utilizando Generics en C# utilizaremos **List** de **System.Collections.Generics**.

```
// Instanciar un System.Collections.Generics.List
List<Persona> l = new List<Persona>();

// Agregar un objeto de tipo Persona a la lista
l.Add(new Persona("Gabriel", "Bulfon"));

// Obtener un objeto de tipo Persona desde la colección.
// Nótese la ausencia del moldeado (cast)
Persona p = l[0];
```

Como se aprecia, el código es más prolijo, elegante y eficiente.

Restricciones

Es posible aplicar restricciones (**constrains**) a los tipos (**type parameters**) en la declaración de las clases genéricas. Las restricciones en C# se declaran utilizando la palabra reservada **where**, y una interfaz que debe cumplir el tipo, como el siguiente ejemplo:

```
public class Trabajos<K>
      where K : IPersistible
{
private K _item;

public Trabajos(K item)
      {
         _item = item;
      }
```

CUIDADO CON INFORMACIÓN DE LAS BETAS

Las betas de ASP.NET incorporaban más funciones y controles que las que terminó teniendo en la versión final, por lo que es importante verificar cuando usamos alguna fuente de datos que no sea de las Betas.

```
public void Procesar()
    {
        // Al ser un tipo que cumple con IPersistible
        // podemos usar el método Save
        _item.Save();
    }

}
```

La sentencia:

```
_item.Save();
```

Fallaría si **K** no implementara la interface **IPersistible**. Por lo tanto es necesario comprobar que **K** es de un "tipo de tipos". Ya no es "cualquier" tipo, sino uno que implementa la interface **IPersistible**.

De esta manera podremos escribir código suponiendo que los tipos (**types parameters**) son de un tipo que si bien es desconocido en la declaración de la clase, asumimos que implementa la interfaz **IPersistible**.

Si la restricción no estuviera presente, sólo se puede asumir que **K** es del tipo **System.Object**. El compilador generará entonces un error ya que object no posee el método **Save()**.

Es posible implementar varias restricciones por tipo aunque sólo se permite una clase por tipo como restricción. El siguiente ejemplo expande la clase Trabajos con otro type parameter:

```
public class Empleados<K,U>
        where K : IEmpleado, IPersistible
        where U : Persona {

}
```

Se permite utilizar múltiples interfaces como **constrains**, pero sólo una clase.

Sintaxis

La sintaxis en C# 2.0 ya la estuvimos viendo en los ejemplos, pero formalmente sería la siguiente:

```
// Crear un objeto de una clase genérica
ClaseGenerica<Tipo> nombre;

// Definir una clase con genéricos
public class ClaseGenerica<tipos_separados_xcoma>

// Definir una clase con genéricos y restricciones
public class ClaseGenerica<tipos_separados_xcoma>
where tipo : interfaz
```

En Visual Basic 2005, la misma funcionalidad se logra con la siguiente sintaxis:

```
' Crear un objeto de una clase genérica
Dim nombre as New ClaseGenerica(Of Tipo);

' Definir una clase con genéricos
Public Class ClaseGenerica(Of Tipo)

' Definir una clase con genéricos – varios tipos
Public Class ClaseGenerica(Of Tipo1, Tipo2)

' Definir una clase con genéricos y restricciones
Public Class ClaseGenerica(Of Tipo As Interfaz)
```

NULLABLE TYPES

Los tipos habilitados para ser nulos, o **Nullable Types** son tipos de datos que podemos permitirles tener valor nulo y que, al no ser clases propiamente dichas, hasta ahora no tenían la posibilidad de serlo.

Por ejemplo, un entero (**int** en C# o **Integer** en VB) no tenía forma de tener un valor nulo. Esto es muy importante para unir datos con bases de datos, dado que allí sí pueden existir campos con datos nulos.

Así, se puede definir un tipo **nullable**, de la siguiente forma en Visual Basic (utilizando un tipo genérico):

```
Dim nul As Nullable(Of Integer)
If nul.HasValue Then
    ' Tiene un valor
End If
```

En C# la sintaxis es un poco más simple, ingresando un símbolo **?** al final del tipo de datos a utilizar, que es una abreviatura de **Nullable<tipo>**:

```
int? nul;
If (nul.HasValue)
    ' Tiene un valor
```

En C# aparece un nuevo operador **??** para asignar un valor por defecto a un tipo nullable. La propiedad **HasValue** de cualquier tipo nullable devuelve si tiene o no valor ingresado y **GetValueOrDefault** devuelve el valor o, si es nulo, un valor por defecto ingresado previamente.

Por ejemplo, el siguiente código en C# guardará en y el valor de x, salvo que este último sea nulo, entonces guardará un -1.

```
int? x;
int y = x ?? -1;
```

CHEQUEO DE TIPOS EN TIEMPO DE COMPILACIÓN

Las restricciones en los tipos genéricos, proveen chequeo de tipos en tiempo de compilación y mejora de la performance (en algunos casos), aunque restringen el potencial de las clases genéricas.

CLASES PARCIALES

Las clases parciales son un concepto simple, pero muy útil. Una **clase parcial** es una clase que físicamente su definición está dividida en más de un archivo. Para el compilador y el uso de la clase es exactamente igual. Sólo nos permite trabajar de mejor forma cuando trabajamos en capas o en trabajos en equipo, pudiendo tener bien separados ambos archivos. También podemos separar los atributos en un archivo y los métodos en otro, por ejemplo **nombre.atributos.vb** y **nombre.metodos.vb**.

En ASP.NET esto trajo una gran solución y es que una página Web de tipo **Code-Behind** trabaja internamente como una sola clase dividida en dos archivos, el archivo .aspx y el archivo .aspx.vb o .aspx.cs, según el lenguaje.

La sintaxis en C# 2.0 es la siguiente, incorporando el operador **partial**:

```
public partial class nombre {

}
```

En Visual Basic 2005 es la siguiente:

```
Partial Public Class nombre

End Class
```

C# 2.0

C# 2.0 trajo la mayoría de actualizaciones relacionadas con la base del lenguaje y con más eficientes métodos para desarrollar aplicaciones.

Iterators

Un **Iterator** es un patrón que define una interfaz para acceder secuencialmente a los elementos de un objeto. Los Iterators son una característica de C# 2.0 que permite la implementación del patrón Iterator. Dicho patrón permite recorrer una colección de elementos en forma secuencial.

La mejora que añade C# 2.0 radica en el hecho de que sólo tenemos que proveer el **Iterator Block** (escribirlo) y el compilador se encargará del resto. Gracias al nuevo compilador debemos programar mucho menos código para proveer la iteración de los elementos de una clase o colección.

Un **Iterator Block** es un bloque de código que contiene declaraciones **yield**. De esta manera, por la presencia de la sentencia **yield** queda definido un Iterator Block. La sentencia yield sigue utilizándose como un identificador, pero se ha modificado en C# 2.0 agregándole a continuación **return** o **break**.

La declaración **yield return** produce el próximo valor de la iteración.
La declaración **yield break** indica que la iteración está completa.

Un **Iterator Block** produce una secuencia de valores del mismo tipo, a este tipo se lo llama tipo **yield** del Iterator Block. El tipo de la sentencia yield debe poder ser convertido implícitamente al tipo del Iterator Block.
Un tipo **yield** es utilizado para implementar una función miembro que devuelve un objeto que implementa **IEnumerable** o **IEnumerator**.
Un tipo **yield** es utilizado para implementar una función miembro que devuelve un objeto que implementa **IEnumerable<T>** o **IEnumerator<T>**.

Veamos un ejemplo para intentar dar un poco de claridad a estos conceptos:

Convertiremos a la clase **AnimalesDeLaSelva** en un tipo enumerable que permite el recorrido de sus valores secuencialmente.

```
class AnimalesDeLaSelva {
        string[] _AnimalesDeLaSelva =
                {"Leon", "Elefante", "Jirafa", "Gorila"};

        public System.Collections.IEnumerator GetEnumerator() {
            for (int i = 0; i < _AnimalesDeLaSelva.Length; i++)  {
                yield return _AnimalesDeLaSelva[i];
            }
```

```
        }
}
```

La clase **AnimalesDeLaSelva** contiene un array de string que almacena los nombres de los animales.

La implementación del método **GetEnumerator()** permite que la clase sea recorrida utilizando **foreach**. No es necesario nada más.

El siguiente código muestra como utilizar la clase **AnimalesDeLaSelva**:

```
AnimalesDeLaSelva animales = new AnimalesDeLaSelva();

foreach (string animal in animales) {
        Response.Write(animal);
}
```

Otros Agregados

Clases Estáticas: Al mejor estilo de Java ahora es posible definir clases estáticas en las que todos sus métodos son estáticos. Ésta será una clase que no podrá ser instanciada. Y sólo podrán ser invocados los métodos estáticos a través del nombre de la clase.

```
// creamos la clase estática
public static class Sistema {
        public static String Nombre() {
                return "Sistema de Gestión 1.2"
        }
```

ITERATORS EN VISUAL BASIC

La funcionalidad de los iteradores no ha sido incorporada en el lenguaje Visual Basic 2005 y está en la lista de próximas incorporaciones al lenguaje en su próxima versión VB 9.0. Sin embargo recordemos que sí podemos recorrer con **For Each** una clase iterator realizada en C#.

```
}
// la utilizamos sin instanciar ningún objeto.
Response.Write(Sistema.Nombre());
```

Métodos anónimos: Permiten definir un bloque de código a ejecutarse por un delegado (que, por ejemplo, responderá a un evento de un objeto). De esta forma no necesitamos crear un método para ese efecto. Por ejemplo,

```
btnEnviar.Click += delegate(System.Object o, System.EventArgs e) {
        Response.Write("Has hecho click");
};
```

VISUAL BASIC 2005

Visual Basic 8, o VB 2005, como se lo ha nombrado es la nueva versión del conocido lenguaje de programación de Microsoft. Además de las características antes mencionadas que son a todos los lenguajes, como genéricos, Visual Basic trae los siguientes cambios.

Operador IsNot

Este nuevo operador simplifica muchas expresiones lógicas y equivale a utilizar Not Is, útil por ejemplo con Nothing, como en el ejemplo:

```
If objeto IsNot Nothing then
```

Definición de Arrays

Como ya sabemos, los vectores en .NET empiezan de la posición 0 y, en Visual Basic definirlo daba mucha confusión, si la definición daba la cantidad de elementos o el límite superior, entonces ahora es posible declarar un vector de la siguiente forma, más clara:

```
Dim vector(0 to 9) as String
```

Instrucción Continue

Indica en un bucle que continúe a la siguiente iteración.

Sobrecarga de Operadores

Una de las funciones avanzadas más pedidas era la sobrecarga de operadores, así ahora es posible definir operaciones, como la suma o negación sobre clases creadas por nosotros mismos.

Objeto My

El nuevo objeto **My** ("Mi", en inglés), permite acceder a distintas funcionalidades del .NET Framework de una forma más fácil y accesible. Pensemos en este objeto (ya instanciado) como accesos directos a distintos puntos del árbol jerárquico de clases .NET.

Por ejemplo, podemos leer el texto de un archivo de disco:

```
Dim texto as String =  My.Computer.FileSystem.ReadAllText("C:\Texto.txt")
```

En lo que respecto al desarrollo web, **My** nos ofrece las siguientes opciones:

My.Computer
My.Request
My.Response
My.User
My.Log

Por ejemplo, podemos acceder al nombre de usuario logueado con **My.User.Identity.Name**.

Comentarios XML

Ésta es una opción que ya estaba disponible oficialmente en C# y ahora se incorpora al lenguaje y entorno de Visual Basic. Permite definir comentarios que serán de utilidad para la generación de la documentación oficial del proyecto. Un ejemplo de un comentario XML lo vemos en el siguiente código, ubicándolo previo a la definición de cualquier método.

Además, esta información es utilizada por **Intellisense** para darle ayuda al usuario cuando intenta utilizar el método.

```
''' <summary>
''' Procesa el click de un usuario en el botón
''' </summary>
''' <param name="strNombre">Nombre del Boton</param>
''' <returns>Devuelve 1 si es correcto</returns>
```

Instrucción using

Permite definir un bloque de código que utilizará algún objeto y queremos liberarlo al finalizar dicho bloque de código. Por ejemplo, conexiones a bases de datos, apertura de archivos o gráficos requieren de la apertura y liberación de recursos y son candidatos a usar esta instrucción.

```
Using conexion as SqlConnection
        ' Aquí va el código que trabaja con la conexión
End using
```

Operador TryCast

Permite hacer un **cast** (moldeado) de un objeto hacia otra clase, pero que tiene la particularidad de devolver **Nothing** en caso de que el casting no se haya podido ejecutar.

```
Dim c as Cuadrado = TryCast(figura, Cuadrado)
```

LOS ENTORNOS

ASP siempre se caracterizó por no necesitar de un entorno propietario para poder desarrollar, desde las primeras versiones, con el Notepad era suficiente. No obstante, siempre los entornos de desarrollos (IDEs) nos aumentan notablemente el rendimiento, ayudándonos a escribir menos código, y a recordar los métodos y funciones disponibles.

ASP.NET 2.0 no es la excepción. Si queremos, podremos seguir desarro-

VALIDACIÓN DE ESTÁNDARES

Ahora es posible seleccionar bajo qué navegador queremos garantizar que es válido nuestro código XHTML. Las opciones de "target" las podemos ver en la parte inferior de la pantalla y podemos seleccionar IE 6, HTML 3.2, XHTML 1.0, Opera, Netscape o Compact HTML.

llando con Notepad pero, la verdad es que hay tantos controles, tantas propiedades y eventos disponibles que una herramienta de desarrollo que nos ayude resulta fundamental para programar páginas Web con esta plataforma en poco tiempo. Por suerte, tendremos una opción gratuita o muy económica.

Visual Studio 2005

Ésta es la herramienta más importante para crear aplicaciones bajo el entorno .NET. Ya sea herramientas de escritorio, para equipos móviles o para Web, Visual Studio 2005 tiene todo el entorno de desarrollo, de trabajo en equipos, de ayuda en línea y de soporte necesarios para facilitarnos la tarea.

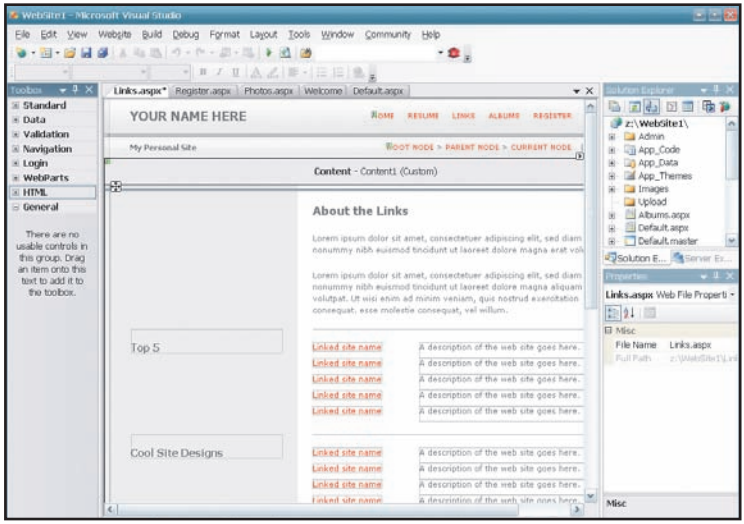

Figura 1. *Visual Studio 2005 es una herramienta que ha sufrido muchos cambios en su entorno para lograr mayor eficiencia en la codificación y desarrollo de aplicaciones.*

Visual Studio es un producto comercial y está disponible en distintas versiones según nuestra necesidad. Con él podremos desarrollar aplicaciones ASP.NET 2.0 en forma visual y codificando en **Visual Basic**, **C#**, u otro lenguaje instalado compatible con .NET 2.0 utilizando el **Visual Web Developer**, el entorno visual para desarrollar aplicaciones Web incluido en Visual Studio 2005. También es posible desarrollar aplicaciones de escritorio, librerías y aplicaciones para equipos móviles.

Algunas características de este entorno:

- Entorno unificado para toda la gama de desarrollo bajo .NET.
- Ayuda en línea y a través de la base de datos MSDN.
- Soporte de **Intellisense**, ayuda al programador en tiempo de codificación.
- Soporte de herramientas para trabajo en grupo, arquitectura y diseño de aplicaciones.
- Soporte de herramientas de **Refactoring.**

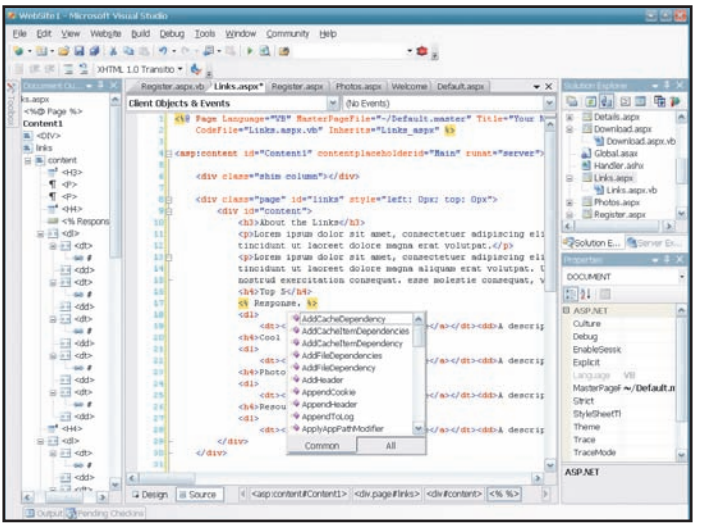

Figura 2. *Aquí vemos al entorno de desarrollo web y las características incluidas, como Intellisense, árbol de clases y soporte de debugging.*

Para aquellos que ya han utilizado Visual Studio 2003 previamente hay muchas novedades interesantes. Era común, en **ASP.NET 1.x** que aparecieran problemas

ARCHIVO DE PROYECTO

Visual Studio 2005 genera un archivo de proyecto, como en la versión anterior. Sin embargo, ya no utiliza este archivo para definir qué ítems (archivos o clases) son parte del proyecto. Para agregar un ítem al proyecto, sólo hace falta copiarlo a la carpeta raíz de éste.

repentinos, como que los eventos de los controles web dejaran de funcionar, que al agregar un control al ASPX desde una aplicación externa, Visual Studio no reconociera el nuevo control al querer compilar el proyecto y que tuviéramos que hacer mil piruetas para lograrlo. Ahora estos problemas ya fueron solucionados y las mejoras respecto al desarrollo de aplicaciones Web son las siguientes:

Figura 3. *Cassini, el servidor web incorporado en Visual Studio y VWD, permite ejecutar aplicaciones sin necesidad de instalar y configurar el IIS. Al trabajar sobre un puerto al azar, no interfiere con otros servidores instalados.*

- Incluye un **servidor Web**, por lo que ya no es necesario tener IIS instalado y funcionando.
- Es posible trabajar con proyectos por carpeta física en el disco, por carpetas virtuales del **IIS** y por carpetas **FTP**.
- Ahora VS no compila todo el proyecto en un solo **.DLL** (que era difícil de administrar en proyectos grandes). Se pueden generar compilaciones parciales o usar el nuevo sistema de compilación dinámica de ASP.NET.
- Permite copiar un sitio entero y publicarlo hacia otra carpeta, unidad de red o vía FTP a un servidor.
- Es posible abrir un archivo **.aspx** y editarlo en VS sin necesidad de abrir un proyecto, ni tenerlo compartido en el IIS. Intellisense funcionará sin problemas con un archivo independiente y abierto de esta forma.

ASP.NET EN LINUX

Tanto para ejecutar aplicaciones .NET en Linux, como en Mac y en otras plataformas, surgió el proyecto **Mono** que permite trabajar en estos sistemas operativos con proyectos compilados en .NET. Respecto a ASP.NET 2.0 ya tiene alguna de sus funcionalidades incorporadas.

- Si editamos código ASPX, al pasar a vista Diseño, VS ya no modificará el estilo, sintaxis, mayúsculas ni identación del código que creamos, dejándolo todo como lo hicimos nosotros.
- En cada **WebForm** podemos seleccionar trabajar del modo **CodeBehind** (como era en VS 2003) o del modo **CodeInside** (como lo hacía **WebMatrix**).
- **Intellisense** está disponible tanto en código de servidor, como en código HTML, ASPX, Javascript y código de servidor embebido en CodeInside.
- Ya no hay código oculto que genera VS, que no sabemos qué hace y que a la mínima modificación deja de funcionar. No hay más **event handlers** ni web controls definidos en CodeBehind.

En un mismo proyecto, ahora podemos trabajar algunas páginas en Visual Basic y otras en C#.

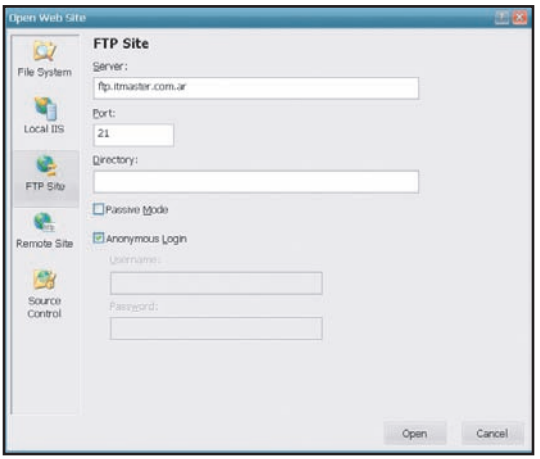

Figura 4. *Al abrir un nuevo proyecto web, tenemos distintas opciones, desde levantar una carpeta de IIS (única opción en VS 2003), abrir una carpeta del disco o una carpeta FTP.*

SEGURIDAD DE USUARIOS

Mientras al utilizar IIS el usuario con el que se ejecutan las operaciones es **ASPNET** o **NETWORK SERVICE**; utilizando el servidor embebido de Visual Studio, se utiliza el usuario actualmente identificado en el sistema. Hay que tenerlo en cuenta para los permisos a habilitar.

Visual Web Developer Express

Esta herramienta es una versión "reducida" de Visual Studio 2005 específicamente para desarrollar aplicaciones Web. Entra dentro del paquete de versiones "Express" de Visual Studio que, por política de Microsoft, se distribuyen con licencia gratuita por tiempo limitado y, se espera que, si llegan a convertirlo en producto comercial, tenga muy poco valor de mercado.

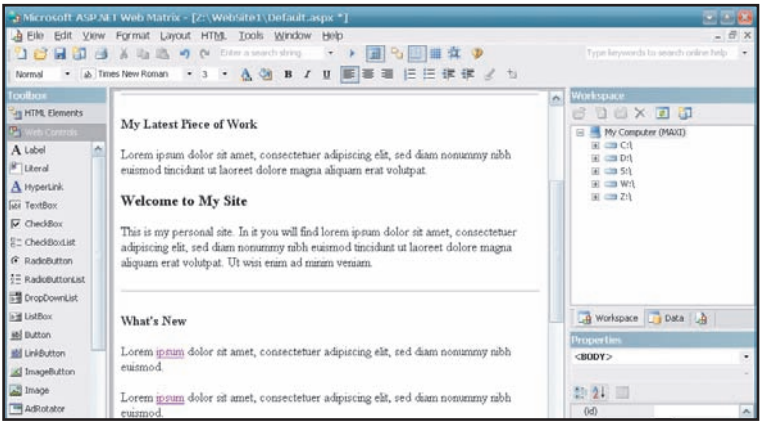

Figura 5. *El viejo amigo WebMatrix quedó fuera de la contienda al salir ASP.NET 2.0, por lo que no se espera ninguna nueva versión de este entorno de apenas 1.3Mb.*

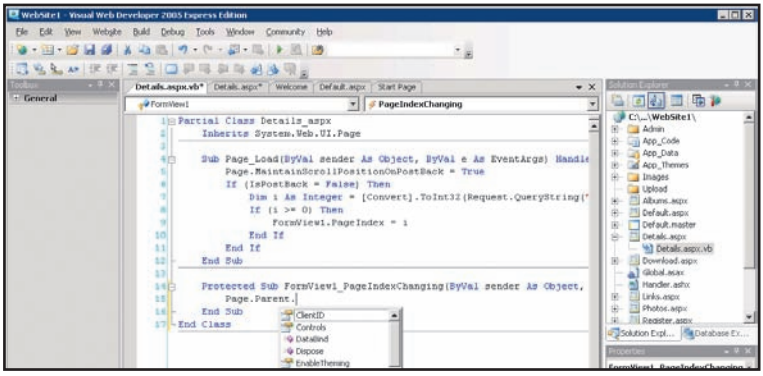

Figura 6. *La diferencia entre WebMatrix y Visual Web Developer está a la vista; el entorno es mucho más completo y profesional y el soporte de Intellisense es muy bienvenido.*

Aquellos que hayan conocido el producto gratuito **WebMatrix** para ASP.NET 1.x, verán que éste vendría a ser el reemplazante para la versión 2.0, pero sin comparación. **VWD**, como se lo conoce rápidamente es un entorno completo de desarrollo, posee todas las características de avanzada para la codificación y diseño de una página en ASP.NET, por lo que tiene poco que envidiarle a su versión mayor, si somos desarrolladores independientes. Según Microsoft, es un entorno ideal para hobbistas, entusiastas y estudiantes, pero perfectamente cabe en las necesidades de cualquier desarrollador independiente.

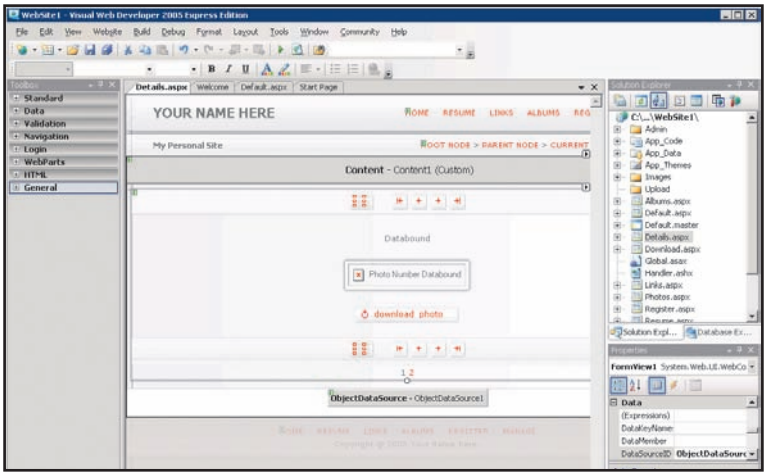

Figura 7. *El diseñador visual de formularios Web de Visual Web Developer posee todas las características de su hermano mayor Visual Studio.*

Sus características son:

- Edición Visual de Sitios Web.
- Soporte de XHTML, CSS y Javascript.

SERVIDOR WEB EMBEBIDO

El servidor Web incluido en VS 2005 y en VWD 2005 es una versión de "**Cassini**", el mini servidor que originalmente había sido incorporado en WebMatrix, el producto gratuito para desarrollar ASP.NET 1.x.

- Soporte de Web Services, RSS y XML.
- Diseñador de Datos Visuales.
- Debugging incorporado.
- Basado en el motor de Visual Studio 2005.
- Soporta Visual Basic y C# indistintamente.
- Incluye Intellisense.
- Permite trabajar directamente con bases de datos SQL Server y Access directamente desde el entorno.
- Soporte de Smart Tasks.
- Tag Navigator, permite navegar por la jerarquía de etiquetas del documento.
- Tiene el mismo servidor web embebido que Visual Studio.
- Permite trabajar con proyectos vía el protocolo FTP, carpetas de IIS o carpetas de disco local o de la red.
- En sólo 80Mb tenemos disponible todo el entorno, el .NET Framework y la base de datos SQL Server Express.

Sus limitaciones respecto a Visual Studio 2005 son:

- Sólo permite utilizar VB y C# y sólo para realizar proyectos web.
- No permite la administración de bases de datos remotas (aunque sí permite trabajar con ellas).
- Documentación reducida (existe un download adicional en Internet de 200MB: **MSDN Express**).
- No tiene diseñador de clases.
- No tiene soporte de edición avanzada de XML.
- No soporta macros ni addins.
- No soporta versionamiento de código.
- No tiene compilador para equipos de 64 bits.
- No soporta profiling, testeos de unidad, administración de proyectos, administración de casos de uso.

COMENTARIOS XML

Los comentarios XML que generan documentación automáticamente, ahora fueron incorporados a VS y VWD tanto en C# como en Visual Basic, en el que se realizan con triple apóstrofe.

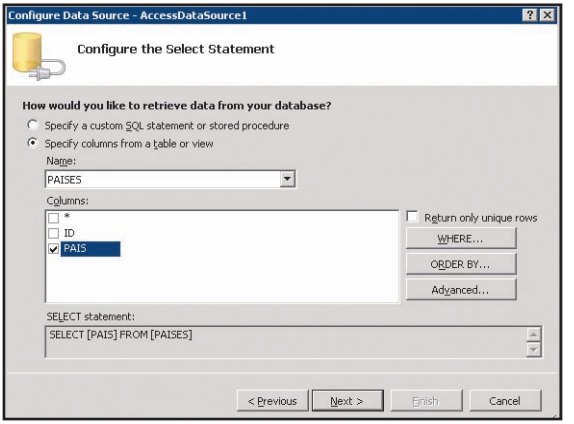

Figura 8. *El diseño de consultas sobre bases de datos es otro aspecto fundamental para todo entorno de desarrollo de aplicaciones; VWD no se queda atrás.*

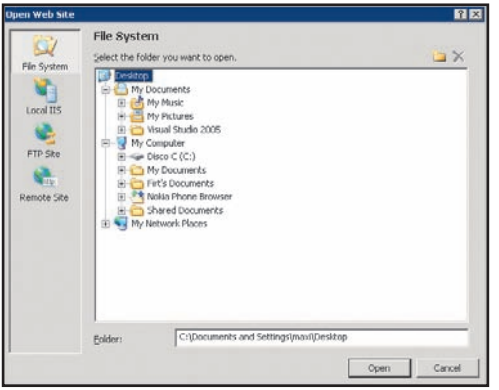

Figura 9. *Así como Visual Studio, Visual Web Developer permite trabajar con proyectos desde una carpeta de disco, un sitio remoto, una carpeta de IIS o vía FTP.*

¿VS O VWD?

Sacando el factor dinero, ¿alcanza VWD? En casos de desarrollos de un solo programador podría alcanzar, y cuando son proyectos con trabajo en equipo y que requieran de modelado, Visual Studio se hace fundamental.

Para más información sobre este producto y descarga la dirección a visitar es **http://msdn.microsoft.com/express/vwd**.

LA BASE DE DATOS

SQL Server 2005

Más de cinco años llevó a Microsoft el desarrollo de la nueva versión de su motor de base de datos: **SQL Server**. La última versión había sido la 2000 y, a fines de 2005, junto a la plataforma .NET 2.0, lanzó la nueva versión, con inmensa cantidad de novedades.

Una de las mayores novedades para los programadores .NET es que ahora es posible programar **Stored Procedures** para SQL Server directamente en **Visual Basic** o **C#**, además de **T-SQL**, dado que incorpora el .NET Framework como parte de la base de datos.

Sus nuevas características son:

- Mejoras en transact-SQL, el lenguaje de consultas y Stored Procedures de SQL Server 2005.
- Cambios en los tipos de datos.
- Soporte de .NET Framework.
- Mayor soporte de XML.
- Particionamiento de las bases de datos.
- Permite crear Web Services en la capa de datos.
- Servicios de Reporting.

Además, viene con el nuevo **SQL Server Management Studio**, que reemplaza al viejo **Enterprise Manager**. Este nuevo entorno se integra a Visual Studio, y permite trabajar y desarrollar con esta base de datos.

TAREAS INTELIGENTES

Tanto VS como VWD soportan **SmartTasks**, una herramienta que permite definir las propiedades más comunes de cada control Web directamente desde el entorno visual sin tener que recurrir a la ventana de propiedades.

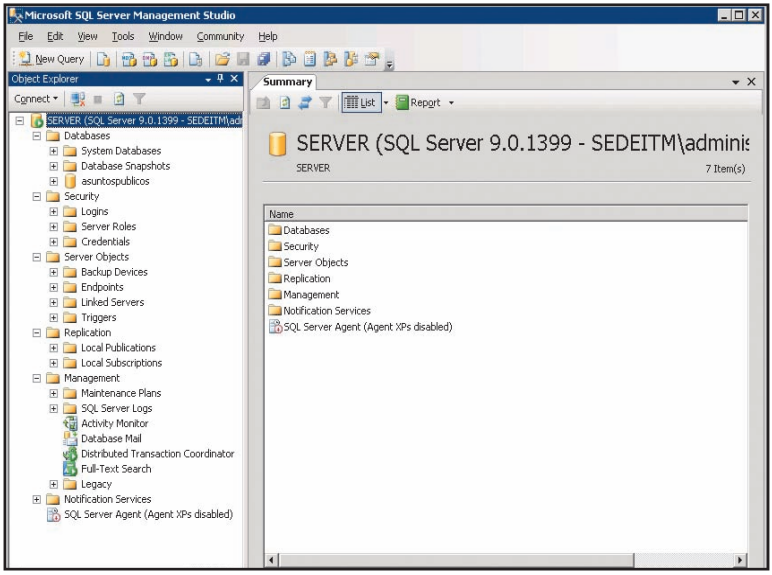

Figura 10. *El SQL Server Management Studio es la única vía de administración para las bases de datos 2005, dado que el Enterprise Manager no puede conectarse a esta nueva versión.*

Edición Express

SQL Server 2005 Express Edition es una versión de licencia gratuita de la base de datos de Microsoft. Está pensada para uso personal, de desarrollo y de producción de sitios que no requieran de grandes capacidades ni herramientas.

Es la evolución del **MSDE** (Microsoft Data Engine), que era una versión gratuita de SQL Server 2000, a diferencia de que esa versión tenía una limitación fundamental para usarse en producción: permitiría sólo cuatro usuarios conectados.

SQL Server 2005 Express no tiene ninguna limitación en cuanto a cantidad de usuarios conectados ni cantidad de bases de datos que puede soportar. Sus únicas limitaciones son:

- Soporta servidores de un procesador y hasta 1Gb de memoria.
- Cada base de datos no puede superar los 4Gb.
- No tiene servicios de Reporting ni **Business Intelligence** (OLAP).

Las características son:

- Tiene todo el poder de SQL Server 2005, no hay reducción de funcionalidad en cuanto a bases de datos.
- Permite migrar a SQL Server 2005 en caso de que sea necesario.
- Soporta **Stored Procedures**, **Triggers**, Tipos Personalizados y Funciones creados desde Visual Studio y VWD.
- Encriptación de datos.
- Capacidades avanzadas de autenticación, auditoría y autorización.
- Soporta Stored Procedures realizados en DLLs compiladas en .NET.
- Soporta servicios de integración de datos.
- Permite suscripciones, snapshot y publicaciones transaccionales.
- Cuenta con la herramienta gratuita **SQL Server Management Studio Express**.

SQL Server 2005 Express puede ser descargada gratuitamente de **http://msdn-.microsoft.com/vstudio/express/sql.**

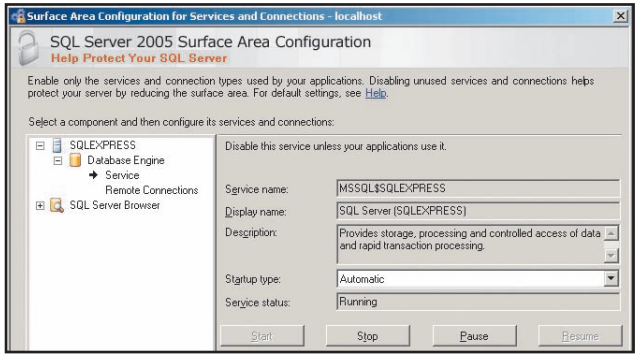

Figura 11. *Con SQL Server 2005 Express podemos montar sitios Web con bases de datos en producción con la seguridad de que tendrán mucho mejor rendimiento que una aplicación que use Access.*

BASES DE DATOS GRATUITAS

Si bien existen bases de datos Open Source, como MySQL y Firebird, muchas empresas grandes del mundo de las bases de datos comerciales han lanzado sus propias versiones "Express" o "personales" como Microsoft, IBM y Oracle. Estas versiones son gratuitas y se pueden descargar.