nº 41 octubre 2007 | 6,50 €

www.dotnetmania.com

dotNetManía

dedicada a los profesionales de la plataforma .NET

OCR.NET

Motor para el reconocimiento óptico de caracteres en .NET

Páginas asíncronas en ASP.NET 2.0
Seguridad de acceso a código
Migración de SharePoint 2003 a 2007
Microsoft Solutions Framework (II)
Configuración (Settings). El configurador que lo configure...

Dedicada a los profesionales de la plataforma .NET

Vol. III •Número 41 • Octubre 2007
Precio: 6,50 €

Editor
Paco Marín
(*paco.marin@netalia.es*)

Redactor jefe
Marino Posadas
(*marino.posadas@netalia.es*)

Editor técnico
Octavio Hernández
(*octavio.hernandez@netalia.es*)

Redacción
Dino Esposito, Guillermo 'Guille' Som, José Manuel Alarcón, Luis Miguel Blanco y Miguel Katrib (Grupo Weboo)

Empresas Colaboradoras

Alhambra-Eidos
Krasis
Plain Concepts
Raona
Solid Quality Learning

Además colaboran en este número
Alberto Población, Gustavo Vélez, Joan Llopart, Leonardo Paneque y Ludwig Leonard.

Corresponsal para América Latina
Pablo Tilotta

Ilustraciones
Mascota (Clico): Yamil Hernández
Portada: Javier Roldán

Fotografía
Roberto Mariscal

Atención al suscriptor
Pilar Pérez
(*pilar.perez@netalia.es*)

Edición, suscripciones y publicidad
.netalia
c/ Robledal, 135
28529 Rivas-Vaciamadrid (Madrid)
www.dotnetmania.com

Tf. (34) 91 666 74 77
Fax (34) 91 499 13 64

Imprime
Gráficas MARTE

ISSN
1698-5451

Depósito Legal
M-3.075-2004

Cuestión de luces

Bienvenido al número 41, de octubre de 2007, de **dotNetManía**.

Este mes pasado se ha producido el lanzamiento de Silverlight 1.0 y la confirmación del trabajo con Novell para ofrecer Moonlight (*luz de luna*, más romántica, pero menos brillante, ¿como la plataforma donde corre?), la versión de Silverlight 1.1 para Linux. Realmente, la noticia que nos interesa está por llegar, y es precisamente la salida de la versión 1.1 de Silverlight (aún en versión alfa) en la que podremos empezar a hacer desarrollos para la Web de aplicaciones multimedia y multiplataforma. Hasta ahora, Flash es un estándar de facto; sin embargo, Silverlight amenaza con entrar con fuerza y quién sabe si arrebatar pronto parte de esa exclusividad. ¿Cuántos programadores de ActionScript hay en el mundo? ¿Cuántos de lenguajes .NET? ¿Qué *luz* brillará más entre los programadores de .NET? Yo creo que está bien claro, nos quedaremos con la *luz plateada*.

Presentamos este mes la librería **OCR.NET** del **Grupo Weboo** de la Universidad de La Habana, cuyo código fuente está a disposición como *open source* desde la Web de la revista en *www.dotnetmania.com*. Su carácter genérico está orientado a la reusabilidad, de modo que los lectores puedan completarla y extenderla según sus intereses, incluso en aplicaciones que hagan reconocimiento de otras simbologías y no solo de caracteres.

José Manuel Alarcón nos ayuda a realizar aplicaciones ASP.NET 2.0 más escalables con su "Páginas asíncronas en ASP.NET 2.0", en el que aprenderemos cómo usar la ejecución asíncrona de páginas para evitar los cuellos de botella producidos por algunos procesos.

Alberto Población, nuestro invitado de este mes, al que doy la bienvenida, presenta "Seguridad de acceso a código", un artículo en el que se explica cómo .NET asigna los permisos a los ejecutables y cómo se puede modificar esos permisos para resolver problemas como que un programa que funcionaba perfectamente en fase de desarrollo no lo haga cuando se copia a una carpeta en producción.

Gustavo Vélez, autor del libro "Programación con SharePoint 2007", editado por Netalia, nos explica de forma práctica algunas consideraciones a tener en cuenta en una migración de SharePoint 2003 a SharePoint 2007.

También publicamos este mes la segunda parte de tres sobre Microsoft Solutions Framework, de **Joan Llopart**, dando continuación a la publicada el mes pasado acerca de esta metodología de Microsoft.

Para los usuarios menos avanzados, **Guillermo "Guille" Som** propone el artículo "Configuración (Settings). El configurador que lo configure..." que muestra cómo gestionar los datos de configuración de una aplicación usando Visual Studio 2005.

Todo lo antes mencionado, junto con nuestras secciones habituales, conforman este primer número de otoño, que espero que sea de su agrado.

Paco Marín

sumario 41

Microsoft lanza Silverlight 1.0 y extiende su soporte a la plataforma Linux

Microsoft presenta además Expression Encoder 1.0, sistema que permite importar, comprimir y publicar online de forma rápida vídeo digital a partir de una amplia variedad de formatos.

Microsoft ha lanzado **Silverlight 1.0**, la evolución del nuevo *plug-in* que soporta múltiples plataformas y exploradores, para Windows y para Mac. Para Linux, Microsoft ha confirmado que seguirá trabajando estrechamente con Novell para ofrecer Silverlight dentro de esta plataforma, en el proyecto conocido como **Moonlight** (*http://www.mono-project.com/Moonlight*), basado en Mono, la conocida alternativa de software libre de .NET, como ya adelantó **Miguel de Icaza**, vicepresidente de desarrollo de Novell, en el REMIX 2007 que tuvo lugar en París.

Microsoft ha desvelado nuevas posibilidades para el usuario, como se puede apreciar en la *preview* del videojuego Halo 3 (*http://www.microsoft.com/silverlight/halo3.aspx*) y en el proyecto de buscador **Tafiti.com**.

La compañía también ha anunciado **Silverlight Partner Initiative**, un programa diseñado para albergar las distintas colaboraciones entre proveedores, redes de contenido, vendedores de herramientas y estudios de diseño. Según esta iniciativa, más de 35 compañías de distinto tipo han decidido adoptar la tecnología Silverlight.

Además de la presentación de Silverlight 1.0, Microsoft ha lanzado **Expression Encoder 1.0** (antes conocido como Expression Media Encoder), una herramienta que facilita a los profesionales codificar, perfeccionar y publicar contenido innovador en Silverlight. Permite a los usuarios codificar una gran cantidad de formatos de contenido multimedia, basados en archivos dentro de la experiencia Silverlight. Expression Encoder también facilita la producción de eventos en directo con la opción de cambio de fuente y la publicación de contenido dentro de servicios como **Silverlight Streaming by Windows Live**.

Antonio Gómez, jefe de producto de Silverlight en Microsoft Ibérica, ha señalado que "con el lanzamiento de Silverlight 1.0 estamos haciendo posible que los desarrolladores y diseñadores creen el tipo de experiencias en alta definición que todos los usuarios demandan, mediante la integración de datos y servicios con un formato único. Silverlight acelerará el crecimiento de las aplicaciones interactivas, al ofrecer a desarrolladores y diseñadores nuevas opciones para crear experiencias innovadoras para la Web, PC, teléfono móvil y otros dispositivos".

Según **Scott Guthrie** (*http://weblogs.asp.net/scottgu*), "ahora que Silverlight 1.0 está en la calle, mi equipo está trabajando duro en la siguiente versión, Silverlight 1.1".

Silverlight 1.1 incluirá una versión multi-plataforma de .NET Framework que permitirá una experiencia rica de desarrollo .NET en el explorador. Incorporará el modelo de programación de WPF para la interfaz de usuario, incluyendo el soporte para un modelo de controles extensible, la gestión de la colocación, enlace a datos, *skinning* y un amplio conjunto de controles incorporados de serie. También incluirá un subconjunto de la librería de clases base de .NET Framework que usamos hoy, ofreciendo soporte para colecciones, genéricos, IO, múltiples hilos, globalización, comunicaciones (incluyendo sockets, servicios Web y soporte REST), HTML, DOM, XML, almacenamiento local y LINQ. Además, será posible utilizar cualquier lenguaje .NET para desarrollar aplicaciones Silverlight, incluyendo lenguajes dinámicos como Javascript, Python o Ruby, lo que realmente nos abrirá muchas nuevas oportunidades.

Para más información, visite la Web de Silverlight en *www.silverlight.net*. Si quiere ver algunos vídeos cortos (en inglés) sobre cómo desarrollar con él, puede hacerlo desde *http://silverlight.net/learn/learnvideos.aspx* o en *http://channel9.msdn.com/ShowPost.aspx?PostID=339594*, donde encontrará un vídeo del propio Scott Guthrie. Si utiliza Visual Studio 2008 o la edición gratuita Web Developer Express 2008, puede descargar una librería desde *http://weblogs.asp.net/scottgu/archive/2007/08/01/vs-2008-javascript-intellisense-for-silverlight.aspx* para obtener ayuda Intellisense para Silverlight 1.0. Y si tiene pensado desarrollar aplicaciones .NET usando Silverlight 1.1 Alpha, esta versión está disponible para su descarga en *http://silverlight.net/GetStarted*.

Liberado el Service Pack 1 de Microsoft .NET Micro Framework SDK 2.0

El pasado 17 de septiembre Microsoft liberó el Service Pack 1 de Microsoft .NET Micro Framework SDK 2.0.

Microsoft .NET Micro Framework SDK 2.0 combina la fiabilidad y eficiencia del código manejado con las facilidades que ofrece Visual Studio para hacer posible un aumento sustancial de la productividad en el desarrollo de aplicaciones embebidas para pequeños dispositivos. La descarga consiste en el SDK completo, que permite desarrollar desde Visual Studio 2005 código para .NET Micro Framework utilizando el lenguaje C# y un subconjunto de las librerías .NET, ejecutarlas sobre un emulador universal y luego desplegarlas sobre hardware real utilizando uno de los kits de despliegue concretos que se suministran.

.NET Micro Framework SDK 2.0 con el Service Pack 1 ofrece las siguientes características:

Integración con Visual Studio

El SDK de .NET Micro Framework se integra en el entorno de Visual Studio, permitiendo hacer uso, para el desarrollo en C#, de características de productividad como la depuración sobre el dispositivo, navegación de objetos, ayuda IntelliSense y compilación y despliegue integrados.

Librerías y drivers de código manejado

Una implementación para dispositivos de un subconjunto del CLR de.NET Framework, que lleva las ventajas del código manejado al desarrollo en para pequeños dispositivos. Es posible desarrollar controladores de dispositivos y rutinas de interrupción en C# para hardware conectado a través de interfaces estándar, como SPI, I2C, GPIO y USART.

Emulador extensible

Es posible extender el emulador básico incluido en el SDK para reflejar las posibilidades de un hardware concreto; por ejemplo, se puede añadir periféricos simulados como sensores, pantallas, botones, etc. Una vez hecho esto, podrá desplegar ese emulador personalizado y probar su código sobre el hardware virtual.

Protección de memoria flash

Para evitar la carga de código no autorizado, el Service Pack 1 añade la posibilidad de almacenar claves y firmas de aplicaciones en su dispositivo.

Herramienta de gestión de la memoria flash

MFDeploy, incluida como parte del SDK, ahora permite la gestión de claves y firmas de aplicaciones en los dispositivos.

Herramienta de generación de fuentes

TFConvert permite convertir fuentes .TTF para que funcionen en su dispositivo.

Más información y descargas en *http://www.microsoft.com/downloads/details.aspx?familyid=32f5df20-6c95-4fe8-a76c-0ed56a839ad2&displaylang=en*.

Alhambra-Eidos convoca su 3er Máster en desarrollo de software

Alhambra-Eidos, empresa especializada en facilitar soluciones a las necesidades empresariales en el ámbito de las TIC, inicia el próximo 19 de octubre de 2007 el tercer **Máster en Desarrollo de Software** en formato *blended*, el XIV de la historia de la compañía.

Tendrá como foco de atención las últimas versiones de los productos de Microsoft: Visual Studio 2005, SQL Server 2005, Microsoft Office Sharepoint Portal Server 2007 y Biztalk Server 2006. Además, sus contenidos facilitarán la certificación de más alto nivel de la carrera oficial para desarrolladores de Microsoft: el MCPD Enterprise Applications Developer.

El máster *blended* dará comienzo el 19 de octubre y finalizará el 11 de mayo. Esta modalidad combina las clases presenciales (en un 40%) con horas de autoestudio, tutorización y orientación (un 60%). En concreto, este año los alumnos recibirán 176 horas lectivas presenciales y 255 horas de autoestudio. Las clases presenciales del máster se impartirán los viernes por la tarde y los sábados por la mañana.

Más información en *http://www.alhambra-eidos.es*.

Microsoft PerformancePoint Server 2007

Poco más de un año después de la adquisición de Proclarity, Microsoft ha anunciado **Microsoft® Office PerformancePoint™ Server 2007**, una aplicación que permite a las compañías gestionar el rendimiento a través de la monitorización, análisis y planificación de sus negocios usando una solución integrada. Microsoft Business Intelligence permite a la mayoría de las personas de cualquier nivel de una organización tomar mejores decisiones y convierte en realidad la visión de que todos añadan valor a cada decisión.

Windows Mobile Briefing 2007

El pasado día 20 de septiembre, Microsoft celebró en Barcelona tres eventos paralelos orientados a clientes, *partners* y desarrolladores bajo el nombre de **Windows Mobile Briefing**.

Desde **dotNetManía** asistimos al bloque de desarrollo, en el que tuvimos la oportunidad de conocer de primera mano los aspectos más destacados en Windows Mobile 6.0 y la próxima aparición de Visual Studio 2008 y con ello .NET Compact Framework 3.5. Todo esto de la mano de dos habituales en eventos de desarrollo móvil en Europa, **John Wyer** y **David Goon**.

En la primera parte, John mostró las posibilidades de desarrollo que hoy por hoy ofrece la última versión de Windows Mobile, qué papel juega en ella la plataforma .NET y cuál es la hoja de ruta tanto del sistema operativo como de .NET Compact Framework. Antes de dar el relevo a David, y ya durante la comida que tuvo lugar en el mismo recinto, pudimos compartir mesa con John y David y nos hablaron de la enorme expectación que se está generando en torno a las nuevas características disponibles con la próxima aparición de Visual Studio 2008 las cuales serán, sin duda, uno de los temas estelares en el próximo Tech-Ed que tendrá lugar en Barcelona en el próximo mes de noviembre y en el que ambos estarán presentes.

En la segunda parte, David nos brindó una magistral demostración práctica de Windows Mobile 6.0 con .NET CF 3.5. Intercepción de SMS, integración con el teléfono —así como con las tareas y correo de Pocket Outlook— y buenas prácticas en el uso de notificaciones de sistema, hasta la utilización del Cellular Emulator y Fake GPS. No podía dejarnos sin mostrar un ejemplo —aunque fuera sencillo— de Compact WCF. Por primera vez tuvimos la oportunidad de ver `CfSvcutil.exe`, el generador cliente de servicios WCF para la versión Compact, y cómo de una forma rápida y sin demasiadas complicaciones podemos consumir servicios WCF.

Ya en la recta final, mostraron los aspectos más destacados de Mobile Client Software Factory (MCSF), destacando algunos *application blocks*, y cómo MCSF se está convirtiendo en un referente dentro del desarrollo de aplicaciones móviles, pasando a ser incluso el pilar para plataformas de aplicaciones como la de Microsoft Dynamics CRM 3.0 Mobile.

En definitiva, un evento sobresaliente, tanto por la organización como por el contenido y los ponentes, deseando que Microsoft vuelva de nuevo en futuras ediciones.

José Miguel Torres

Edición de otoño de Basta!

El pasado 17 al 21 de septiembre se celebró la edición de otoño de BASTA, en Mainz, Frankfurt. Con más de 600 asistentes, **BASTA** y **SQL Con** se han convertido en la conferencia por excelencia de Alemania en lo que se refiere a .NET y SQL Server. Hubo más de 70 ponentes, entre los cuales se encuentran algunos de mucho prestigio tales como **Dominick Baier**, **Christian Weyer**, **Andreas Kosch** o nuestro compañero **Dino Esposito**. Tal como comentaba el propio **Massoud Kamali**, director de Software & Support Verlag, organizadores del evento, la conferencia pretende ser un evento de la comunidad donde hay un alto grado de participación de muchos integrantes de ésta. Como muestra, desde hace un par de años, se obsequia a proyectos innovadores con hasta 20.000 euros, premio que se anuncia el último día de la conferencia y que esta ocasión fue a parar a **dynaTrace** (*www.dynatrace.com*). Sin embargo, para los organizadores todavía no ha terminado el trabajo. La semana del 24 al 28, ya con este número en imprenta, tendrá lugar **EKON/Euro DevCon** en su undécima edición, conferencia que se enfoca a desarrolladores Win32 y .NET. Este año promete ser interesante, ya que CodeGear (anteriormente Borland) recientemente ha lanzado al mercado **Highlander,** la última versión de su entorno RAD para .NET 3.0. Además como novedad, han sacado **3rdRail**, su nuevo entorno de desarrollo bajo *Ruby on Rails*, que promete ser todo un éxito.

Hadi Hariri

Presentamos ADS 8.0

Tu motor de base de datos

ADVANTAGE DATABASE SERVER

Mejoras y nuevas funcionalidades:

- Soporte a replicación
- Funcionalidad de copia de seguridad online
- Punto de guardado de operaciones
- Índices y comunicaciones codificadas
- Extensiones del lenguaje SQL
- Optimización de velocidad y rendimiento
- Mejora de la interfaz de Usuario y Operaciones de ADSSTAMP.EXE

ABOX

C/Manso 26-28, 2a planta · 08015 Barcelona
Telf.: 902 160 145 · Fax: 934 231 140
E-mail: abox@abox.com · Web: http://www.abox.com

SYBASE iAnywhere

José Manuel Alarcón

Páginas asíncronas en ASP.NET 2.0

En ocasiones, nuestras aplicaciones deben realizar tareas largas, que bloquean el hilo de ejecución principal a la espera de que otros procesos terminen. Este tipo de situaciones se debe evitar si queremos desarrollar sistemas escalables, capaces de atender a miles de usuarios. Como veremos, estas situaciones son más comunes de lo que cabría esperar, y la ejecución asíncrona de páginas es nuestro mejor aliado.

La programación asíncrona, como todo buen conocedor de los fundamentos de la plataforma .NET sabe, permite que un proceso disponga de múltiples subprocesos o hilos de ejecución que nos proporcionan la posibilidad de llevar a cabo más de una acción al mismo tiempo o ejecutar acciones en segundo plano.

En el caso de las aplicaciones Web ASP.NET, la programación asíncrona se utiliza para mejorar enormemente la eficiencia de las páginas Web que tardan mucho en ejecutarse debido a operaciones de entrada/salida que pueden llegar a ser largas. Esta mejora viene del modo en que Internet Information Server procesa las peticiones de páginas que se realizan por parte de los distintos usuarios.

Un barniz de cómo procesa las peticiones IIS

IIS solo es capaz de atender un número concreto de peticiones de páginas de manera simultánea, ya que a cada petición le asigna un hilo de ejecución propio. Estos hilos de ejecución los obtiene de un *pool* o repositorio que contiene una cantidad determinada de ellos. Por defecto se dispone de 100 hilos de ejecución por procesador, aunque esta cantidad es configurable, tal y como se indica en `http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/6b46c077-49c2-40d5-a6bc-56cdfa79d457.mspx?mfr=true`.

Cuando se alcanza ese límite y todos los hilos están ocupados, IIS encola las nuevas peticiones que lleguen para atenderlas en cuanto le sea posible. Mientras tanto, estas peticiones estarán pendientes y sin ofrecer respuesta a los usuarios. En casos extremos, los usuarios comenzarán a recibir códigos de estado `503` de "servicio no disponible" y no se procesarán las páginas.

Cuando una petición termina de ser atendida, su hilo correspondiente se libera y se devuelve al *pool* y queda disponible para procesar otra petición. En condiciones normales, el tiempo que uno de estos hilos está ocupado es muy pequeño y se pueden atender muchas peticiones prácticamente simultáneas sin problema. Las dificultades comienzan cuando una de nuestras páginas realiza alguna operación que tarda mucho tiempo en ejecutarse. Mientras se ejecuta esta operación larga, el hilo estará bloqueado, y a medida que se realicen peticiones a ésta u otras páginas de ejecución larga quedarán bloqueados más hilos, disminuyendo notablemente la escalabilidad de la aplicación, puesto que no habrá subprocesos disponibles para atender más peticiones. Debemos intentar evitar esta situación en la medida de lo posible.

Generalmente, estas operaciones largas se corresponden con operaciones de E/S, como esperar a que se procese una consulta larga en el servidor de datos o acceder a un servicio Web remoto que puede tener

José Manuel Alarcón Aguín es redactor de **dotNetManía**. Es ingeniero industrial y especialista en consultoría de empresa. Ha escrito varios libros, y ha publicado más de trescientos artículos sobre informática e ingeniería en revistas especializadas. Es MVP de ASP.NET, MCTS, MCPD, MCT y tutor de **campusMVP**. Visite su blog en ***www.jasoft.org***

problemas de tráfico. Todas estas tareas tienen asociados métodos nativos para hacer llamadas asíncronas que podemos utilizar en nuestras páginas con la infraestructura de asincronía de ASP.NET que ahora veremos. Al ejecutar de forma asíncrona las llamadas a esos procesos largos, conseguimos que el subproceso principal que atiende la petición quede liberado y se devuelva al *pool* para poder atender nuevas peticiones de otros usuarios. Al terminar la ejecución asíncrona, se recupera un hilo del repositorio para devolver los resultados al cliente.

En general, nos beneficiaremos más de la ejecución asíncrona en situaciones en las que el proceso largo se ejecute en otro servidor, como en los ejemplos citados. El motivo es que si lo único que hace nuestro código es esperar a que termine alguna tarea remota en el servidor de datos o en otro servidor Web, mientras se espera en segundo plano el impacto sobre la capacidad de proceso es prácticamente nulo, por lo que devolver el proceso a IIS asegura que se gestionen nuevas peticiones de páginas sin mermar el rendimiento. Sin embargo, si el proceso largo implica realizar una tarea intensiva en nuestro propio servidor (procesar un archivo muy grande o acceder a la base de datos local), liberamos el subproceso para atender nuevas llamadas, pero hay un procedimiento ejecutándose en segundo plano que compite con IIS por los recursos de la máquina. En este caso, tendremos una mayor capacidad de respuesta a peticiones (escalabilidad), pero probablemente el aumento de rendimiento no sea apreciable. Es importante tener claro este concepto.

Implementación de páginas asíncronas

Dado que la plataforma .NET ofrece un gran soporte para operaciones multi-subproceso asíncronas, siempre podemos hacer uso de ellas en nuestras aplicaciones para conseguir la ejecución en segundo plano de tareas. De hecho, en las versiones 1.x de ASP.NET no quedaba más remedio que hacerlo de

Figura 1

esta forma, lo que implicaba bajar de nivel en el código y añadía bastante complejidad a este tipo de operaciones. Por fortuna, ASP.NET 2.0 nos pone las cosas mucho más fáciles. Aunque tendremos que hacer algo de trabajo extra, éste es muy llevadero.

ASP.NET 2.0 introduce un nuevo atributo para las páginas llamado `Async`. Basta con colocarlo en la directiva de la página actual con el valor establecido en verdadero para que se cree de modo transparente para nosotros la infraestructura necesaria para ejecutar tareas de manera asíncrona. La figura 1 muestra el aspecto de este atributo aplicado a una página.

A continuación, se deben definir dos métodos que nos servirán para iniciar y terminar la acción "larga" que queremos implementar en segundo plano. Su definición es la siguiente (se puede variar el nombre sin problemas, como veremos en breve):

```
Public Function BeginProcessRequest(ByVal sender As Object, _
     ByVal e As System.EventArgs, ByVal cb As System.AsyncCallback, _
     ByVal extraData As Object) As System.IAsyncResult
End Function

Public Sub EndProcessRequest(ByVal result As System.IAsyncResult)

End Sub
```

Estos dos miembros son muy parecidos a los de la interfaz `IHttpAsyncHandler` de ASP.NET, pero cambian los dos primeros parámetros del método de inicio de la llamada.

Finalmente, solo resta indicar en la página que se tiene que llamar a estos dos métodos de manera asíncrona, para lo cual tenemos dos formas de conseguirlo:

1) Llamar al método `AddOnPreRenderCompleteAsync` de la página, pasándole referencias a estos dos métodos.
2) Llamar al método `RegisterAsyncTask`, indicándole la definición de una tarea asíncrona a llevar a cabo. Éste se suele utilizar cuando hay varias tareas asíncronas en la misma página, como enseguida estudiaremos.

De acuerdo. Hasta aquí la teoría, que puede parecer más complicada de lo que realmente es. Ahora veámoslo con un ejemplo práctico y se entenderá mucho mejor.

Llamada asíncrona a un método largo

En general, el patrón de llamadas asíncronas de .NET está implementado en todas las clases que implican entrada y salida y que en ocasiones pueden dar problemas de ejecución larga. Por ejemplo, las clases *proxy* que genera .NET para utilizar servicios Web exponen métodos específicos para realizar la llamada de manera asíncrona, a saber, `BeginXXX` y `EndXXX`, donde `XXX` representa el nombre común del método usado para la llamada síncrona, más habitual. Lo mismo ocurre en el caso de la E/S de archivos, las llamadas para ejecutar procedimientos de bases de datos, etc.

Este patrón de asincronía debería ser de sobra conocido para un programador de la plataforma .NET. Por este motivo, voy a hacer un ejemplo más sencillo en sus resultados, pero que al mismo tiempo es más complejo y completo porque ilustra cómo dotar de capacidad asincrónica a un método cualquiera que no tenga implementado "de serie" un patrón

asíncrono mediante `BeginXXX` y `EndXXX`, así que seguramente será más interesante para la mayoría de los lectores.

El ejemplo (que se puede descargar del sitio Web de **dotNetManía**) muestra una página con dos etiquetas. En la primera de ellas se apunta la hora exacta de comienzo de la ejecución. A continuación, se llama a un método que tarda cinco segundos en ejecutarse y que detiene la ejecución de la página, y finalmente se muestra la hora tras haberlo ejecutado (por lógica, cinco segundos después). En el código de ejemplo se ha implementado este esquema de forma normal (síncrona), y usando lo visto hasta ahora con asincronía. Si ejecutamos uno y luego el otro desde Internet Information Server y usamos una herramienta de prueba de carga de aplicaciones que simule decenas o cientos de usuarios, encontraremos una gran diferencia en los tiempos de respuesta de las páginas y en la escalabilidad de la aplicación, siendo la página asincrónica capaz de gestionar muchos más usuarios simultáneos que la página normal por los motivos expuestos anteriormente.

Vamos a analizar el código:

1.- Marcar la página como asíncrona

Lo primero es, como ya sabemos, marcar la página con el atributo de asincronía correspondiente, como se puede ver en la figura 1.

2.- Definir el método que tarda en ejecutarse

En este caso, simplemente metemos artificialmente un retardo de cinco segundos, poniendo el subproceso de ejecución actual en suspenso:

```
Private Sub hazAlgoLargo()
 System.Threading.Thread.Sleep(5000)
End Sub
```

3.- Definir el comienzo de la operación asíncrona

Debemos implementar la función `BeginProcessRequest`, que toma ciertos parámetros que ya hemos presentado y devuelve un manejador de resultados asíncronos (`IAsyncResult`). Para devolver un manejador de este tipo, debemos generar una ejecución asíncrona de nuestro método. La forma de hacerlo es a través de un delegado que lo represente (un delegado en .NET, como ya sabrá el lector, es asimilable a un puntero seguro a un método). Para ello, lo primero que hacemos es definir dentro de la clase un nuevo tipo de delegado que represente al método que queremos ejecutar, en este caso uno que no tiene parámetros y no devuelve nada, así:

```
Delegate Sub TareaSinParametros()
```

Así de fácil. Si tuviese parámetros y/o devolviera algún valor haríamos una declaración análoga del delgado para reflejarlos.

Creamos también una variable miembro privada en la página para guardar una referencia a la función a ejecutar, lo que será tan sencillo como esto:

```
Private tarea As TareaSinParametros
```

Ahora ya tenemos un delegado con el que representar nuestra tarea asíncrona, así que la definición del método `BeginProcessRequest` (o como le hayamos llamado) será:

```
Public Function BeginProcessRequest(ByVal sender As Object, ___
     ByVal e As System.EventArgs, ByVal cb As System.AsyncCallback, __
     ByVal extraData As Object) As System.IAsyncResult

    tarea = New TareaSinParametros(AddressOf hazAlgoLargo)
    Dim res As IAsyncResult = tarea.BeginInvoke(cb, extraData)
    Return res
End Function
```

Es decir, obtenemos una referencia/delegado al método que queremos ejecutar asíncronamente con `AddressOf` (en C# se pondría simplemente el nombre del método sin paréntesis) y comenzamos la ejecución asíncrona de la misma usando el método `BeginInvoke` que poseen todos los delegados. Éste devuelve un objeto de control de tipo `IAsyncResult`, que devolvemos inmediatamente como resultado de la función. Con esto hemos conseguido que al llamar a este método se empiece la ejecución asíncrona del método `hazAlgoLargo`.

4.- Definir el final del proceso asíncrono

Ahora debemos definir qué queremos hacer cuando nuestro método asíncrono se haya terminado de ejecutar. Nos falta, por tanto, implementar el método `EndProcessRequest` (o como le hayamos llamado). En nuestro caso, sería simplemente:

```
Public Sub EndProcessRequest(_
 ByVal result As _
       System.IAsyncResult)
 tarea.EndInvoke(result)
 HoraFin.Text = _
  String.Format( "Fin: {0}", _
  DateTime.Now.ToLongTimeString())
End Sub
```

Es decir, con el controlador asíncrono que nos entrega el método y el delegado de la función llamamos a `EndInvoke` para dar por terminada la ejecución asíncrona. Finalmente, apuntamos en la etiqueta la hora de finalización de la llamada asíncrona.

5.- Especificar los métodos asíncronos que se van a utilizar

En el evento `Load` de la página debemos indicar qué métodos de la misma se van a utilizar para lanzar y terminar la ejecución asíncrona. Generalmente utilizaremos para ello el método `AddOnPreRenderComplete-`

Async, por lo que nuestro evento Load quedaría así:

```
Protected Sub Page_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Me.Load
    HoraInicio.Text = String.Format("Inicio: {0}", _
      DateTime.Now.ToLongTimeString())
    Me.AddOnPreRenderCompleteAsync(AddressOf BeginProcessRequest, _
      AddressOf EndProcessRequest)
End Sub
```

¡Listo! Ahora, al ejecutar la página, el evento Load indica a la infraestructura de ASP.NET que se debe ejecutar el método BeginProcessRequest para lanzar una tarea asíncrona y el método EndProcessRequest cuando ésta termine, y todo funcionará de maravilla. Si prueba bien el código adjunto, comprenderá mejor su funcionamiento, que no es tan complicado como parece. Además, conviene tener en cuenta que en este caso hemos realizado un ejemplo complejo que consiste en llamar a un método que no ofrece soporte "de serie" para recibir llamadas asíncronas. Si lo que quisiésemos es llamar a una consulta larga contra una base de datos u otro método con soporte nativo incorporado, el código sería mucho más fácil todavía.

Modificación del ciclo de vida de la página

Con la ejecución asíncrona de tareas, lo que estamos consiguiendo por debajo es modificar el ciclo de vida de la página. La figura 2 ilustra perfectamente lo que quiero decir. En dicha figura, el diagrama de la izquierda nos muestra el ciclo de vida normal de una página (simplificado), ilustrando los distintos pasos/eventos que se van produciendo.

Cuando ejecutamos una tarea de forma asíncrona el flujo de eventos se modifica, tal y como se puede observar en el diagrama de la derecha de la figura 2. Tras haber ejecutado el evento PreRender de la página, se detiene el hilo de ejecución de la misma y se lanza un nuevo hilo asíncrono con la tarea especificada. Este hilo principal de la página, como ya he comentado, se devuelve al *pool* de IIS para atender otras peticiones. Al terminar la ejecución asíncrona, se retoma la ejecución de la página, pero ya en otro hilo diferente que se toma también del *pool* de IIS.

Ejecución de varias tareas asíncronas con ventaja

El último paso explicado antes utiliza el método AddonPreRenderCompleteAsync para registrar la tarea asíncrona. Existe una opción adicional, que consiste en envolver la información de la tarea mediante una clase específica denominada PageAsyncTask. El constructor de la misma acepta como argumentos los métodos para inicio y fin de

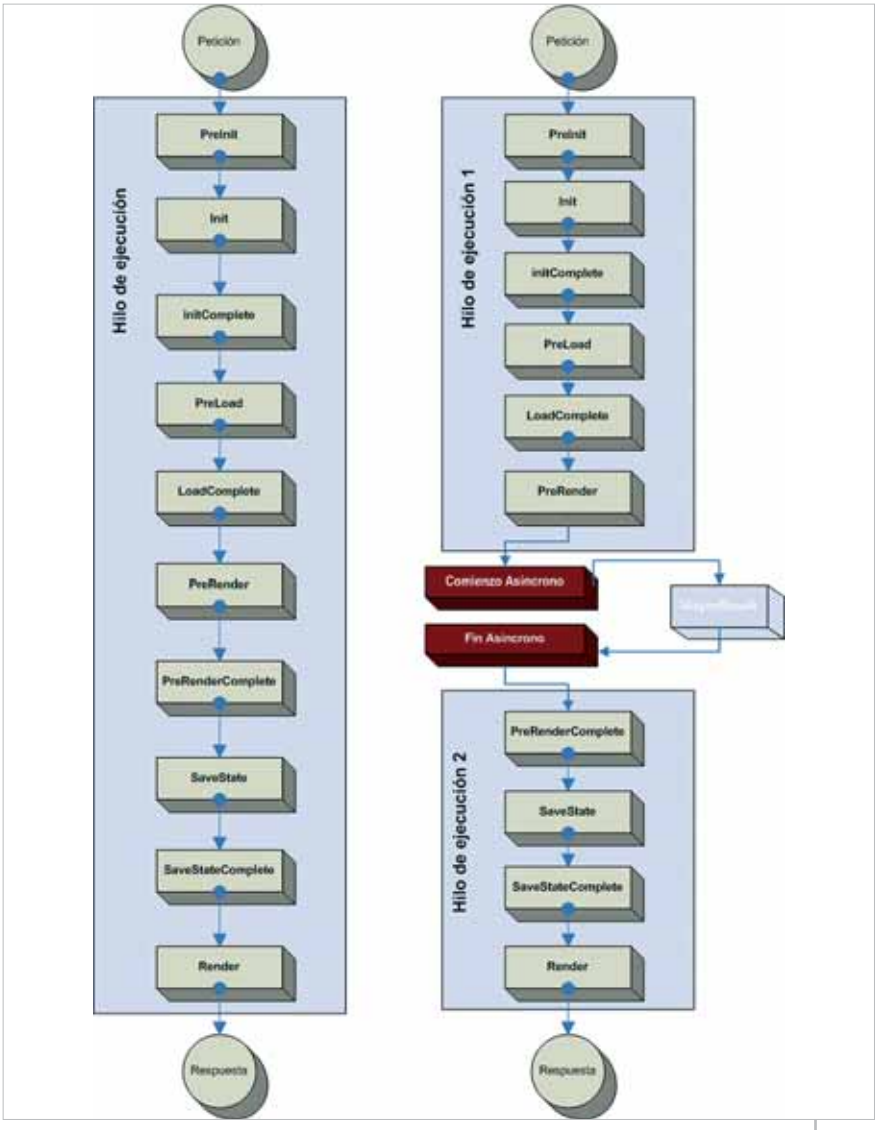


Figura 2

la tarea asíncrona, por lo que el siguiente código sería equivalente al del paso 5 anterior:

```
Protected Sub Page_Load( ByVal sender As Object, _
                         ByVal e As System.EventArgs) _
                         Handles Me.Load
    HoraInicio.Text = String.Format("Inicio: {0}", _
                              DateTime.Now.ToLongTimeString())
    Me.RegisterAsyncTask(New PageAsyncTask(_
                              AddressOf BeginProcessRequest, _
                              AddressOf EndProcessRequest, _
                              Nothing, Nothing))
End Sub
```

La diferencia es que usamos el método `RegisterAsyncTask` propio de la página actual y le pasamos una nueva instancia de la clase `PageAsyncTask` que describe la tarea a ejecutar.

Este método parece una forma más clara de lanzar tareas asíncronas que el —sin embargo— más habitual método anterior. A mí personalmente me gusta más. Además de esta claridad en el código, que puede ser una ventaja bastante subjetiva, ofrece algunas ventajas adicionales de índole técnica:

1.- La clase `PageAsyncTask` nos permite indicar también qué método queremos que se ejecute en caso de que se supere un determinado tiempo de ejecución. Para ello, en su tercer parámetro podemos pasar un delegado a un método del tipo `WebEndEventHandler`, que es idéntico en su definición al método de finalización de la tarea. En el ejemplo hemos prescindido de su uso, pasándole un valor nulo (`Nothing` en VB). Este tiempo máximo de espera se puede controlar con un atributo de la directiva de página llamado `AsyncTimeOut`. Éste es también el nombre de una propiedad de la página, así que podemos establecerlo mediante código. Por ejemplo si escribimos `AsyncTimeOut=3` en la directiva de la página estamos indicando que al cabo de tres segundos se debe llamar a este método de gestión de tiempo excedido, lo cual puede ser muy útil para cortar la ejecución y mostrar un mensaje de error, por ejemplo.

2.- El cuarto parámetro del constructor nos permite identificar de manera única a una tarea asíncrona de la página, en caso de utilizar varias. Normalmente le pasaremos una cadena identificativa, pero podemos pasar cualquier otro objeto que nos proporcione, por ejemplo, una forma de compartir información entre las tareas, ya que se pasará al método de inicio en su último parámetro de información adicional.

3.- Este método nos permite ejecutar diversas tareas asíncronas en la página. Para ello, simplemente tenemos que definir sus manejadores de comienzo y final, como hemos visto hasta ahora, y podemos registrar tantas tareas como deseemos mediante el método `RegisterAsyncTask`. Es más, existe otro constructor de `PageAsyncTask` que ofrece un parámetro adicional, de tipo booleano, que nos permite decidir si la tarea se ejecutará en paralelo con las demás asíncronas que hubiese o si, por el contrario, se ejecutará de manera exclusiva haciendo esperar a la siguiente que hayamos registrado. Esto puede ser de gran interés en muchas aplicaciones.

4.- Una ventaja fundamental de usar este método en lugar del anterior es que `RegisterAsyncTask` conserva dentro del método de finalización de la tarea asíncrona el estado de suplantación de usuarios, los ajustes de cultura y lenguaje actuales, y el contexto de la llamada actual (`HttpContext.Current`). Esto no ocurre con el método anterior.

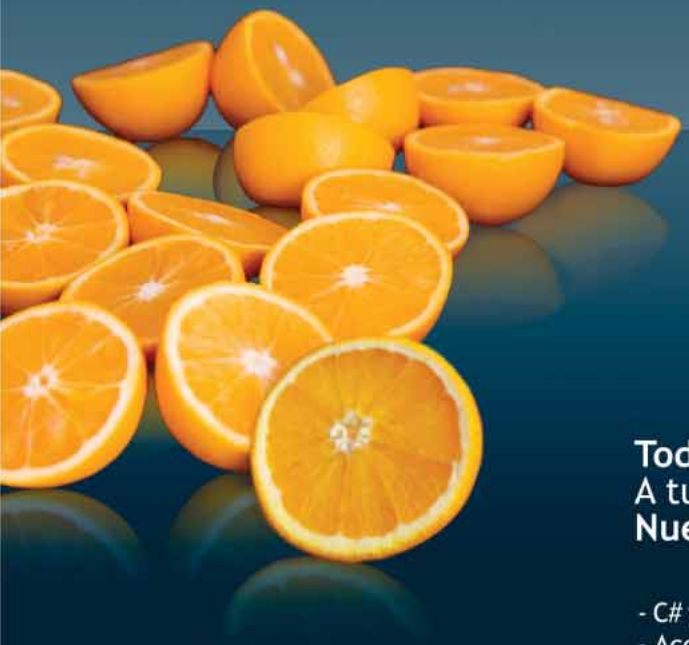
5.- Aunque la ejecución de la tarea asíncrona se lanza de manera automática en el evento `PreRender` de la página, con este método tenemos mayor control sobre el instante en que queremos que comience la ejecución de la misma gracias a que podemos utilizar el método `ExecuteRegisteredAsyncTasks` de la página. Éste lanza de inmediato la ejecución de las tareas asíncronas que tengamos registradas. No es necesario llamarlo, pero nos ofrece un control adicional que puede sernos de utilidad en algún momento.

En definitiva, podemos usar cualquiera de los dos métodos de lanzar tareas asíncronas (o incluso ambos a la vez), pero el uso de la clase `PageAsyncTask` nos ofrece ciertas ventajas objetivas interesantes y es recomendable si queremos lanzar varias tareas asíncronas al mismo tiempo o en secuencia. No obstante en la mayoría de los ejemplos y documentos que se encuentran en la red se usa el primer método.

En resumen

Lo que marca la diferencia entre una aplicación "del montón" y una aplicación profesional de carácter empresarial son precisamente los detalles que surgen cuando las exigencias sobre la misma comienzan a ser grandes: muchos usuarios, restricciones de memoria, tareas largas y complejas, etc. La escalabilidad es una cuestión que debería estar entre los primeros puestos de la lista de preocupaciones de una aplicación empresarial, junto a la seguridad, la facilidad de despliegue y mantenimiento o la usabilidad. Una de las técnicas avanzadas que permite crear aplicaciones escalables en circunstancias adversas es la creación de páginas con procesamiento asíncrono. En este artículo hemos presentado los fundamentos necesarios para poder sacar partido a esta práctica tan útil.

Muchos exprimieron el verano con nosotros... ¿y tú?



campus MVP

Todavía puedes ponerte al día en .NET
A tu ritmo con nuestros **cursos online**
Nuestros tutores MVP resolverán todas tus dudas

- C# y fundamentos de .NET
- Acceso a datos
- Desarrollo Web
- Aplicaciones empresariales
- Aplicaciones para Dispositivos Móviles
- SQL Server

Y además:
Cursos de Certificación Oficial de Microsoft

feed **your brain.**

infórmate ya:

902 876 475
www.campusmvp.com

krasis

Microsoft CERTIFIED Partner

Learning Solutions
Custom Development Solutions

Alberto Población

Seguridad de acceso a código

Una consulta frecuente entre los desarrolladores de aplicaciones .NET se refiere al hecho de que un programa que funcionaba perfectamente mientras se encontraba en desarrollo deja de funcionar cuando se pasa a producción por el método de copiarlo a una carpeta en un servidor y ejecutarlo desde ella. En este artículo examinamos la forma en que .NET determina los permisos de los ejecutables y cómo éstos se pueden modificar para resolver casos como el indicado. En un futuro artículo detallaremos cómo podemos aprovechar estos mecanismos de seguridad desde nuestro código.

Qué es la seguridad de acceso a código en .NET

La seguridad de acceso a código –conocida en los manuales en inglés como CAS (*Code Access Security*)– permite que los desarrolladores de aplicaciones y los administradores de sistemas restrinjan los recursos a los que puede acceder un programa ejecutable desarrollado con .NET.

Los archivos `.exe` desarrollados con las herramientas tradicionales anteriores a .NET no tienen ninguna limitación en cuanto a los recursos del sistema que pueden utilizar, aparte de las que se determinen por los permisos que el sistema conceda al usuario que ejecuta la aplicación. Por el contrario, los desarrollados con .NET, además de las restricciones impuestas por los permisos del sistema operativo, están sujetos a limitaciones adicionales establecidas por el entorno común de ejecución de .NET.

.NET Framework dispone de herramientas y clases para examinar y modificar los permisos que se conceden a cada ejecutable. En este artículo examinaremos el funcionamiento y configuración de la seguridad de acceso a código desde un punto de vista externo al programa, y en una próxima entrega nos detendremos en las clases y atributos que podemos utilizar dentro de nuestros programas para controlar su comportamiento.

Pruebas

El primer paso para determinar los permisos que recibe un ejecutable consiste en reunir lo que se denomina "Pruebas" (*Evidence* en inglés), que consisten en un conjunto de características referentes a la identidad y el origen del ensamblado. La concesión de permisos se realiza en base a estas Pruebas.

Las Pruebas que se examinan son el directorio de la aplicación, la inclusión del ensamblado en el GAC, el resumen criptográfico (*Hash*), el editor (*Publisher*) del software, el sitio de descarga, el nombre seguro (*Strong Name*) y la URL.

Cuando se carga en memoria un ensamblado, el motor de ejecución de .NET determina las distintas Pruebas que le son aplicables, y en base a ellas decide a qué grupo o grupos de código pertenece el ensamblado, tal como veremos más adelante.

Alberto Población lleva 27 años desarrollando software. Cuenta entre otras con las certificaciones MCSE, MCDBA, MCITP, MCSD, MCPD y MCT, además de ser MVP de C#. En la actualidad trabaja como consultor independiente, dedicándose principalmente a la formación, asesoramiento y desarrollo de aplicaciones.

Conjuntos de permisos

Las políticas de seguridad de acceso a código permiten que un administrador o un usuario final controlen los permisos concedidos a un ensamblado en función de las Pruebas de éste.

Con .NET Framework vienen ya predefinidos múltiples permisos que se pueden asignar al código, por ejemplo `UIPermission` (permiso para utilizar la interfaz de usuario), `FileIOPermission` (para trabajar con archivos), `RegistryPermission` (para el acceso al registro), etc. Adicionalmente, se pueden definir permisos por programación heredando de la clase `CodeAccessPermission`.

Estos permisos individuales pueden a su vez agruparse en conjuntos de permisos. La herramienta de configuración de .NET Framework, de la que hablaremos más adelante, nos permite definir y examinar los conjuntos de permisos. La figura 1 muestra un fragmento de la pantalla de esta herramienta, en la que se ven los conjuntos predefinidos. Haciendo clic sobre cualquiera de ellos podemos ver los permisos que contiene el conjunto, como nos muestra la figura 2. Algunos de los permisos admiten a su vez configuración adicional, similar al ejemplo de la figura 3.

Figura 1

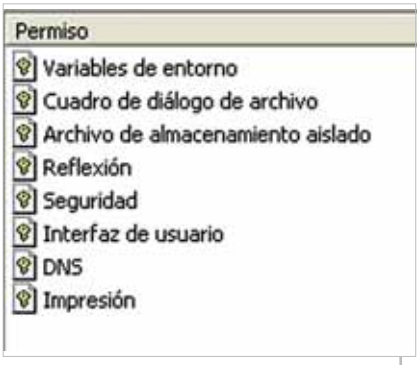


Figura 2

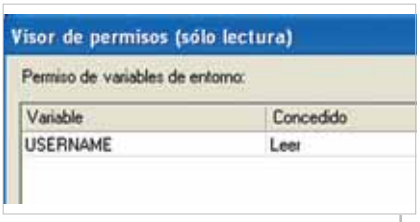


Figura 3

Grupos de código

Para decidir qué permisos deben concederse a un determinado ensamblado, primero hay que decidir a qué grupo o grupos de código pertenece. Estos grupos se pueden configurar por medio de la herramienta de configuración de .NET Framework. La configuración predeterminada se muestra en la figura 4, en la que podemos apreciar un grupo denominado `All_Code` ("todo el código") del que cuelgan varios grupos que coinciden con las Zonas configuradas en Internet Explorer.

Cada uno de estos grupos tiene asignada una condición de pertenencia, que es la que determina qué ensamblados pertenecen a este grupo. En la figura 5 podemos observar que el grupo `My_Computer_Zone` tiene como condición de pertenencia la Zona "Mi PC". De esta forma, un ensamblado que se carga desde la máquina local cumple la condición de pertenencia al grupo `My_Computer_Zone`, y en consecuencia recibe los permisos

Figura 4

asignados a este grupo. Estos permisos se configuran a través de la lengüeta "Conjunto de permisos" de la ventana de propiedades del grupo de código.

En la figura 6 vemos que el grupo que estamos tomando como ejemplo recibe los permisos del conjunto "Full Trust". En consecuencia, los ejecutables lanzados desde el disco local de nuestro ordenador (zona "Mi PC") reciben permisos ilimitados ("Full Trust"), salvo que se modifique la configuración de permisos predeterminada.

Los grupos de código se pueden configurar de forma jerárquica, tal como vemos en el árbol de la figura 4, en la que las diversas zonas cuelgan de la rama "All_Code". A la hora de determinar la pertenencia de un ensamblado a un grupo de código, se examina cada rama del árbol y si el ensamblado cumple la condición de pertenencia a esa rama se continúan examinando las ramas que cuelguen de ella. En caso contrario, se descarta esa rama sin entrar en las inferiores.

Un ensamblado puede pertenecer a múltiples grupos de código. Por ejemplo, podríamos haber definido un grupo cuya condición de pertenencia sea "todos los ensamblados descargados desde el servidor X", y otro grupo para "todos los ensamblados firmados con

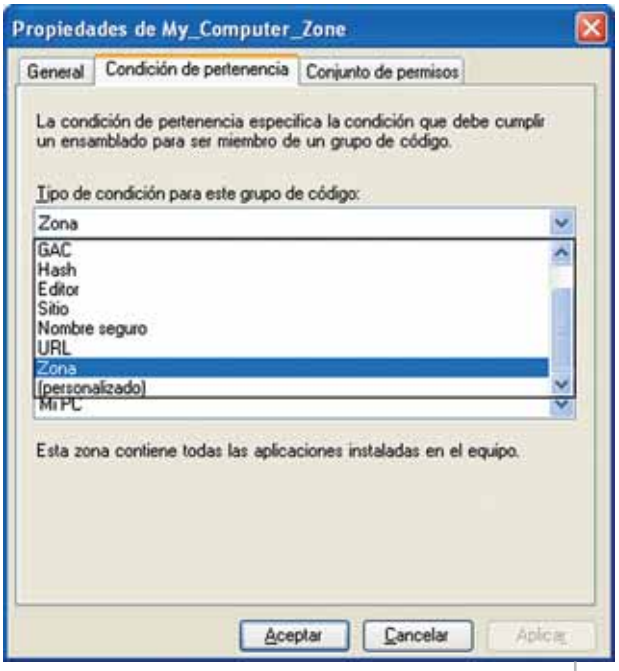


Figura 5

Figura 6

el certificado Y". Un ejecutable firmado con el certificado "Y" descargado del servidor "X" pertenecería a ambos grupos. Cuando un ensamblado pertenece a varios grupos, recibe la **unión** de todos los conjuntos de permisos de todos los grupos a los que pertenece.

Niveles de políticas de seguridad

Si nos fijamos una vez más en la figura 4, veremos que el árbol de las directivas de seguridad tiene tres ramas designadas como "Empresa", "Equipo" y "Usuario". De forma predeterminada, la única rama que viene configurada es la de Equipo. Cada una de estas ramas representa un Nivel de política de seguridad. Existe un cuarto nivel denominado "Dominio de Aplicación" que no dispone de herramientas administrativas, sino que corresponde con los permisos aplicados desde programa cuando se crea mediante código un Dominio de Aplicación.

Al cargar un ensamblado, cada uno de los niveles se evalúa por separado, determinando los permisos que el ensamblado debe recibir conforme con los grupos de código a los que pertenece dentro de ese nivel. Finalmente, se asigna al ensamblado la **intersección** de todos los conjuntos de permisos que le otorga cada nivel.

Así pues, cada uno de los niveles sirve para aplicar restricciones suplementarias sobre los permisos concedidos por los otros niveles. Por ejemplo, un usuario podría otorgar en su equipo, a nivel de Usuario, permisos de escritura en el disco local para los ejecutables .NET descargados desde Internet, pero si no se modifica a nivel de Equipo la política predeterminada, dichos ejecutables no podrán grabar sobre el disco local, puesto que la política a nivel de Equipo no lo permite.

NOTA

Es posible introducir dentro del código fuente de un programa directivas que le hagan rechazar parte de los permisos concedidos por la política de seguridad, logrando que el ejecutable no tenga esos permisos pese a que la política los conceda.

Atributos de los Grupos de código

Existe un par de de atributos que se pueden asignar a los grupos de código y que modifican el comportamiento descrito anteriormente. En la documentación se denominan "Exclusive" y "LevelFinal", y en la figura 7 podemos observar las dos casillas de verificación que permiten establecer estas características desde la herramienta administrativa.

Si a un grupo de código se le aplica el atributo `Exclusive`, lo que se logra es que a los ensamblados que pertenezcan a ese grupo se les apliquen única y exclusivamente los permisos de ese grupo, aunque el ensamblado pertenezca a más grupos. No es lícito que un ensamblado pertenezca a más de un grupo que esté marcado con el atributo `Exclusive`. Si esto ocurre, se produce un error del tipo `PolicyException`.

El atributo `LevelFinal` impide que se apliquen otras políticas establecidas a un nivel menor. El orden en el que se evalúan los niveles es el de Empresa, Equipo y Usuario. Así pues, si a nivel de Empresa se aplica un atributo `LevelFinal` en un grupo de permisos, los ensamblados que pertenezcan a ese grupo no serán evaluados en cuanto a su

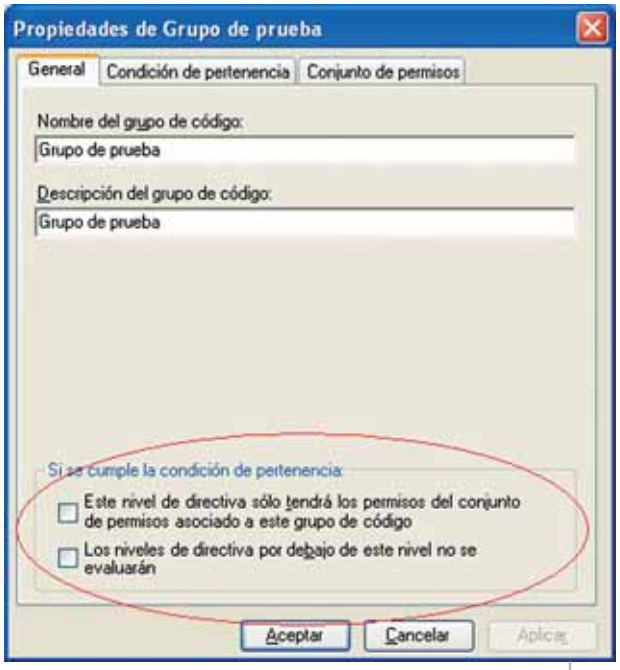


Figura 7

NOTA

En todos los casos en los que hablamos de los permisos concedidos por el motor de ejecución de .NET, siempre hay que tener en cuenta que los permisos efectivos disponibles para el ejecutable pueden ser menores, ya que los programas siguen sujetos a cualesquiera limitaciones sean impuestas por el sistema operativo y otros elementos de infraestructura al usuario que está ejecutando el código. Por ejemplo, aunque .NET conceda el permiso "E/S de archivo" sin restricciones, no significa que los programas puedan escribir sin restricciones en todo el disco, sino solo en aquellos archivos o carpetas en los que las Listas de Control de Acceso del sistema de archivos lo consientan.

pertenencia a grupos definidos a nivel de Equipo o de Usuario.

Es lícito aplicar a un grupo simultáneamente los dos atributos.

Modificación de las políticas de seguridad

Como hemos visto, de forma predeterminada los ejecutables lanzados desde el disco local gozan de los permisos definidos como "plena confianza", lo cual es consistente con el comportamiento esperado por quienes están acostumbrados a instalar y ejecutar programas desarrollados con tecnologías anteriores.

Sin embargo, hay ocasiones, sobre todo en entornos empresariales, en los que se decide ubicar los ejecutables de los programas en un directorio compartido en un servidor. Cuando se lanzan estos programas desde un puesto de trabajo, mediante una ruta de acceso del tipo `\\servidor\ruta\programa.exe`, reciben los permisos del conjunto `LocalIntranet`, que de forma predeterminada son sumamente limitados, no permitiendo operaciones tales como la escritura en disco o el acceso a bases de datos. Esta situación resulta sorprendente para quien espera del ejecutable el comportamiento "tradicional", sin ser consciente de la intervención de las políticas de seguridad de .NET. En estos casos, deberemos modificar las políticas de seguridad en los puestos de trabajo para lograr el comportamiento esperado.

Continuando con el ejemplo anterior del servidor corporativo en el que se instalan los programas que deben estar disponibles para toda la empresa, podríamos optar por agregar un grupo de código cuya condición de pertenencia sea "todos los programas descargados desde este servidor", y asignar a ese grupo el conjunto de permisos adecuado, que probablemente sería "Full_Trust" si el servidor efectivamente es de confianza. Esta política tendría que ser aplicada en todos los puestos de trabajo desde los que deban operar correctamente estos programas. Se dispone para ello de la herramienta de configuración de .NET Framework y la herramienta de línea de comandos CASPOL, de las que hablaremos a continuación.

La herramienta de configuración de .NET Framework

Esta herramienta es accesible desde las Herramientas Administrativas del Panel de Control. Para disponer de ella en la versión 2.0 del Framework no es suficiente con instalar el Framework por sí solo, sino que se necesita instalar algún producto adicional que la contenga, tal como Visual Studio 2005 o, al menos, el SDK del Framework.

Tras abrir la herramienta, ésta presenta un árbol de opciones, del cual nos interesa la rama denominada "Directiva de Seguridad en Tiempo de ejecución". En los epígrafes anteriores hemos hablado ya de su utilización.

El comando CASPOL

`CASPOL.EXE` es una herramienta de línea de comandos incluida con el SDK de .NET Framework (el nombre es una abreviatura de *Code Access Security POLicy*). Ejecutándolo con los parámetros adecuados, conseguiremos modificar los grupos de código y conjuntos de permisos de la misma forma que con la herramienta gráfica de configuración. La ventaja de CASPOL es que se pueden agrupar varias llamadas a este programa en un archivo de comandos, que luego se puede distribuir y ejecutar en múltiples equipos para reproducir la configuración de seguridad.

CASPOL se invoca escribiendo a continuación una serie de parámetros que indican las operaciones a realizar. El primero de los parámetros indica el

nivel de política que se aplicará a todos los parámetros que vengan a continuación. Los posibles valores son los siguientes:

- **-a:** Todos los niveles.
- **-en:** Nivel Empresa (*Enterprise*).
- **-m:** Nivel Equipo (*Machine*).
- **-u:** Nivel Usuario para el usuario que esté ejecutando el comando en ese momento. Este es el valor predeterminado para todos los usuarios que no sean administradores.

Las políticas de seguridad de acceso a código permiten que un administrador o un usuario final controlen los permisos concedidos a un ensamblado en función de las pruebas de éste.

Seguidamente, se añaden al comando los parámetros necesarios para indicar la operación deseada. Se puede consultar o modificar la configuración de las políticas. Para consultar se usan estos parámetros:

- **-lg:** Listar Grupos
- **-lp:** Listar conjuntos de Permisos
- **-l:** Listar ambos

Por ejemplo, para listar toda la información al nivel de políticas de usuario se usa este comando:

```
Caspol -u -l
```

Para añadir, modificar o borrar grupos de código, se utilizan los parámetros **-ag**, **-cg** y **-rg**, seguidos en cada caso de los parámetros necesarios para especificar la operación solicitada.

En la figura 8 vemos un ejemplo en el que se ha ordenado desde la línea de comandos la creación de un grupo de código llamado "Grupo prueba", colgando del nodo número 1 del árbol (se pueden averiguar los números con **caspol l**), con condición de pertenencia basada en URL y correspondiente con el directorio **c:\prueba**, asignando los permisos del grupo **LocalIntranet**, y aplicando el atributo **Exclusive**.

Este ejemplo es particularmente interesante porque nos proporciona una forma sencilla de probar los programas que estemos desarrollando y que deban funcionar con un conjunto de permisos concreto. Si copiamos el programa al directorio **c:\prueba** y lo ejecutamos desde ahí, se ejecutará con los permisos de **LocalIntranet**, a pesar de estar alojado en nuestro propio disco. La clave está en el atributo **Exclusive** que le hemos asignado, que hace que no se apliquen los permisos derivados de otros grupos de código a los que pertenezca el ensamblado. De no haberlo añadido, puesto que el ensamblado pertenece además al

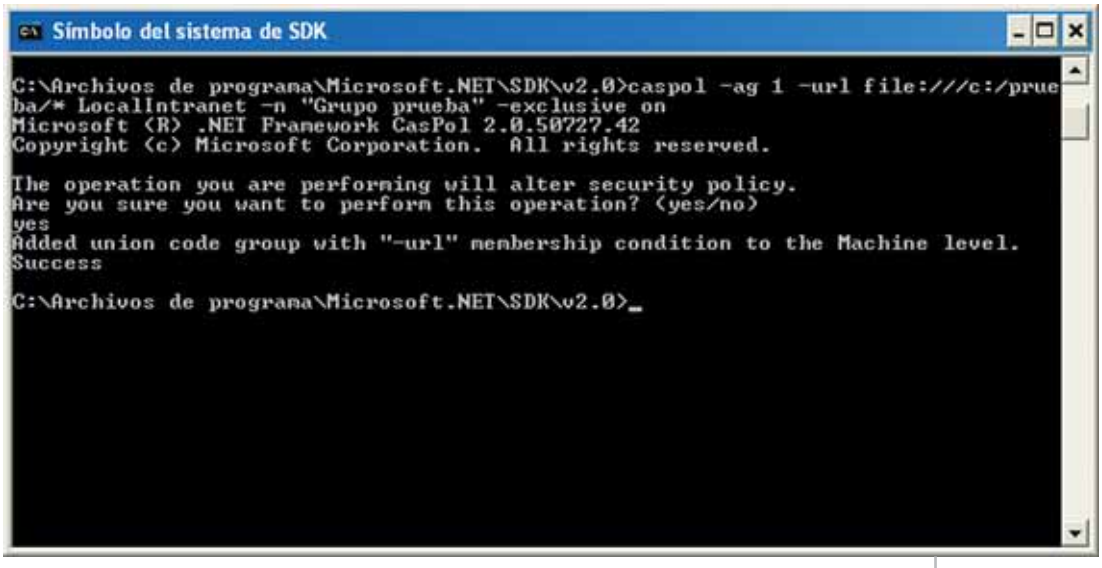


Figura 8

grupo `My_Computer_Zone`, se le aplicaría la unión de los permisos de este grupo (`FullTrust`) con los del "Grupo prueba" (`LocalIntranet`), resultando que el programa recibiría `FullTrust`.

Si no queremos que aparezca el mensaje de solicitud de confirmación (ante el que tenemos que contestar "yes"), podemos suprimirlo añadiendo el parámetro `-q` (*quiet*) al principio de la sentencia.

Escribiendo caspol `-rg "Grupo prueba"` podemos eliminar el grupo que acabamos de crear. Escribiendo `caspol -?` obtendremos una lista de los parámetros disponibles.

Otras opciones de utilidad son `-reset` para devolver las políticas de seguridad a su estado predeterminado, y `-af <nombre de ensamblado>` para conceder "Full Trust" a un ensamblado.

Aplicar una política a todos los equipos de una red

Con frecuencia ocurrirá que necesitemos aplicar un conjunto de políticas de seguridad a todo un conjunto de equipos en un entorno empresarial. En estos casos, configuraremos las políticas en un único equipo en el que dispongamos de la herramienta administrativa de .NET Framework a nivel de Empresa. A continuación seleccionaremos sobre la Directiva de Seguridad la opción "Crear Paquete de implementación", como se muestra en la figura 9.

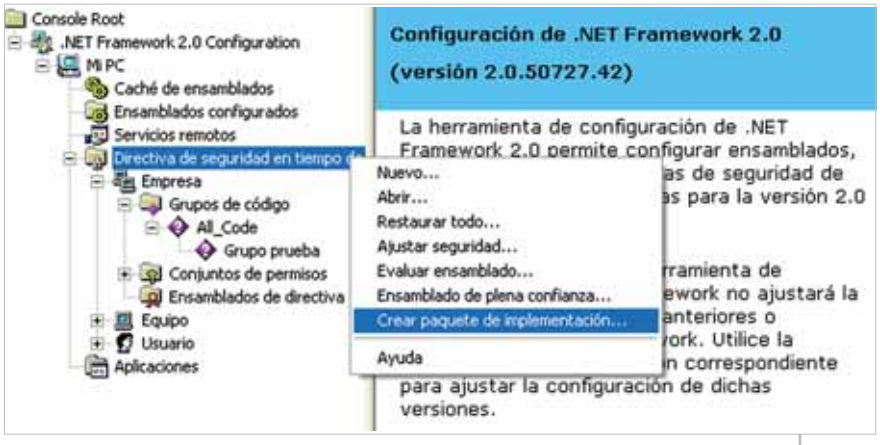


Figura 9

Nos preguntará el nivel de políticas para el que queremos generar una instalación, y el archivo que queremos generar. El resultado será un archivo `.msi` para Windows Installer, que podremos instalar en los equipos en los que deseemos crear la misma configuración de permisos. Si se trata de una instalación basada en Directorio Activo, se pueden usar las tecnologías de IntelliMirror para asignar la "aplicación" (el `.msi`) a un grupo de usuarios o equipos. De esta forma la política se aplica a los equipos necesarios dentro de la red.

Conclusión

En este artículo hemos dado un pequeño repaso a la forma en que el sistema de seguridad de .NET Framework utiliza la seguridad de acceso a código para controlar el nivel de acceso a los recursos externos que se conceden al código ejecutable. Gracias a este control de seguridad podemos limitar el riesgo que supone para un recurso informático la ejecución de un programa que no provenga de una fuente de completa confianza.

Referencias

La librería de MSDN contiene amplia información acerca de la administración de directivas de seguridad: `http://msdn2.microsoft.com/es-es/library/c1k0eed6(VS.80).aspx`.

Concesión de permisos a los ensamblados

El proceso que conduce a determinar los permisos que finalmente recibe un ensamblado tiene dos fases:

1) Calcular los permisos concedidos por la política de seguridad.

   Se concede la intersección de los permisos concedidos en cada uno de los niveles de seguridad (Empresa, Equipo y Usuario), los cuales a su vez son la unión de los permisos de todos los grupos a los que pertenece el código dentro del nivel correspondiente.

2) Restringir los anteriores permisos en base a las declaraciones de permisos opcionales y/o rechazados existentes dentro del ensamblado.

   Usualmente se realizan estas declaraciones en el código fuente mediante la inclusión de atributos del tipo `[assembly:SecurityPermission(…)]` o `[assembly:PermissionSet(…)]` con los parámetros adecuados. Estas declaraciones de seguridad no se tratan en el presente artículo.

dotNetManía

dedicada a los profesionales de la plataforma .NET

❑ Nº27 (6,50€)

❑ Nº28 (6,50€)

❑ Nº29 (6,50€)

❑ Nº30 (6,50€)

❑ Nº31 (6,50€)

❑ Nº32 (6,50€)

❑ Nº33 (6,50€)

❑ Nº34 (6,50€)

❑ Nº35 (6,50€)

❑ Nº36 (6,50€)

❑ Nº37 (6,50€)

❑ Nº38 (6,50€)

❑ Nº39 (6,50€)

❑ Nº40 (6,50€)

Oferta válida hasta el 31 de octubre de 2007 o hasta agotar existencias

❑ Deseo suscribirme a **dotNetManía** por un año (11 números) por un precio de **65,00€ IVA incluido**.

❑ Deseo que me envíen los **números atrasados marcados** según el precio de portada. Otros

(Solo para residentes en España)

DATOS DE ENVÍO

CIF/NIF ..... Empresa .....
Nombre y apellidos .....
Dirección .....
Población ..... Código Postal ..... Provincia .....
Teléfono ..... Fax ..... email .....

DATOS DE FACTURACIÓN (sólo si son distintos a los datos de envío)

CIF/NIF ..... Empresa .....
Nombre y apellidos .....
Dirección .....
Población ..... Código Postal ..... Provincia .....
Teléfono ..... Fax ..... email .....

FORMA DE PAGO

❑ Talón nominativo a nombre **NETALIA, S.L.**

❑ Transferencia bancaria a nombre de **NETALIA, S.L.** a:
La Caixa - Número de cuenta **2100 4315 48 2200014696** (Indique su nombre en la transferencia)

❑ Domiciliación Bancaria (con renovación automática, previo aviso)
Indique su número de cuenta: _____ _____ __ __________

❑ Tarjeta de crédito
❑ VISA ❑ MASTERCARD
Número de su tarjeta: _____ _____ _____ _____
Fecha de caducidad: __/__ (imprescindible)

Firma y/o sello

a de de 2007

Puede enviar sus datos por **Fax** al **91 499 13 64**, o por **teléfono** al **91 666 74 77**, o por **email** a la dirección **suscripciones@dotnetmania.com**, o también puede enviarlos por **correo** postal a la siguiente dirección:

Netalia, S.L.
C/ Robledal, 135
28529- Rivas Vaciamadrid (Madrid)

Usted autoriza a la mecanización de estos datos. El responsable y destinatario de éstos es Netalia, S.L. Usted tiene derecho a acceder a sus datos, modificarlos y cancelarlos cuando lo desee. Sus datos no serán cedidos en ninguna de las formas posibles a terceras partes y no se utilizarán más que para el buen funcionamiento de su suscripción a la revista **dotNetManía** y para informarle de las actividades comerciales que realice la editorial Netalia, S.L. Si no desea recibir información comercial de **dotNetManía** marque la casilla siguiente q



Leonardo Paneque,
Ludwig Leonard,
Miguel Katrib

Motor para el reconocimiento óptico de caracteres en .NET

OCR.NET

En este trabajo se presentan las características generales de una biblioteca "open source" en .NET que ofrece funcionalidades para el reconocimiento óptico de caracteres. Estas funcionalidades se sustentan en más de un centenar de tipos que han sido implementados basados en muchos de los algoritmos disponibles en la red y en otros desarrollados por los propios autores.

El reconocimiento de texto dentro de imágenes ha sido siempre un tema importante para el software. El *Reconocimiento Óptico de Caracteres* (OCR, por sus siglas en inglés) es un proceso que consta de varias etapas, que van desde la detección del área ocupada por el texto dentro de la imagen hasta su extracción y clasificación en cadenas de texto, llegando incluso a la reconstrucción de la estructura original de un documento, si lo que se tiene es una imagen de éste.

El presente trabajo desarrolla una biblioteca de código abierto sobre .NET que encapsula un motor para el reconocimiento óptico de caracteres. Dicho motor consta de interfaces programáticas y módulos para todas las etapas del proceso de reconocimiento. En dicha biblioteca se han incluido algoritmos de más de una veintena de investigadores [1-9], y también se han perfeccionado y combinado algoritmos ya existentes, incluyendo la incorporación de métodos nuevos producto del trabajo de investigación de los autores.

La intención es que esta biblioteca sea de utilidad a los desarrolladores de aplicaciones que quieran incorporar alguna capacidad de reconocimiento de texto como parte de su aplicación.

OCR.NET

Aunque es común aplicar la denominación de OCR sólo a la acción de clasificar la imagen de un carácter aislado, este proceso sólo es uno de los múltiples implicados en la tarea de reconocimiento. En cada etapa es posible aplicar una diversidad de métodos en dependencia de los resultados deseados y del caso particular de información a manejar.

El primer paso del proceso es, claro está, lograr la digitalización en imágenes de la información original, ya sea usando un escáner o una cámara digital. Es decir, partimos de que la fuente es una colección de imágenes que corresponden a una página de un documento (en el cual pueden haber fotos y figuras y no solo caracteres).

Etapas de OCR

- *Pre-procesamiento*. Eliminación de ruido en la imagen, alineación, contraste, binarización.
- *Análisis estructural*. Segmentación de la imagen original en columnas, párrafos, líneas, palabras y caracteres, así como identificación de imágenes correspondientes a fotos y figuras y de texto por separado.
- *Clasificación*. Clasificación de caracteres, formación de palabras, líneas y demás componentes del documento.
- *Post-procesamiento*. Reconstrucción de la estructura del documento, rectificación del texto usando diccionarios u otros métodos.

En dependencia del problema a tratar, se puede omitir alguna de las etapas. Durante el proceso de clasificación se pueden realizar acciones de rectificación usando diccionarios. También se podrían usar

Miguel Katrib es doctor y profesor jefe de programación del departamento de Ciencia de la Computación de la Universidad de La Habana. Miguel es líder del grupo WEBOO, dedicado a la orientación a objetos y la programación en la Web. Es entusiasta de .NET y redactor de dotNetManía.
Ludwig Leonard es estudiante de Ciencia de la Computación de la Universidad de La Habana y miembro del grupo WEBOO. Trabaja como instructor de la cátedra de Programación. Sus áreas de mayor interés son DirectX y gráficos en general .
Leonardo Paneque es graduado en Ciencia de la Computación de la Universidad de La Habana y colaborador del grupo WEBOO. Es un programador todoterreno en la tecnología .NETque gusta principalmente de las redes, los sistemas distribuidos y temas de Inteligencia Artificial

mecanismos de clasificación durante el proceso de análisis estructural, como en el trabajo de **Thomas E. Bayer** [10], que utiliza un clasificador para dividir caracteres conectados en una palabra.

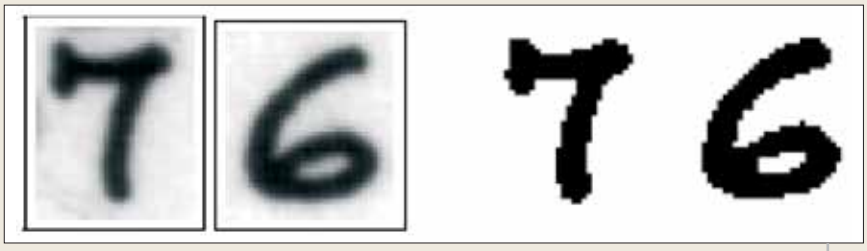


Figura 1. Imágenes antes y después de un filtro de binarización

Modelo de la biblioteca OCR.NET

La biblioteca OCR.NET está desarrollada sobre C# 2.0 para lograr un código más reusable y flexible gracias a la genericidad.

Pre-procesamiento

Los métodos usados para el pre-procesamiento de la imagen a reconocer consisten en la aplicación de filtros. Dichos filtros son implementados mediante la interfaz `IFilter`.

```
public interface IFilter
{
   Bitmap Apply(Bitmap bitmap);
}
```

Cada filtro programado recibe como parámetro de su método `Apply` una imagen (instancia del tipo `Bitmap`) y retorna en otra instancia la imagen resultante de la aplicación del filtro. Por cuestiones de optimización durante la implementación, se prepararon filtros para trabajar con imágenes de color real (24 bits) así como con imágenes en blanco y negro. Entre los filtros implementados en la biblioteca se encuentran:

- Threshold
- Grayscale
- Erosion
- Edge Detection
- Resize
  - Nearest Neighbor, Bilinear y Bicubic
- Rotate
  - Nearest Neighbor, Bilinear y Bicubic
- Invert
- Blur

Pero los filtros no solo son utilizados para eliminar defectos incluidos en la imagen, producto del proceso de digitalización que pasó mediante el escáner el documento del papel a un mapa de bits, como se muestra en la figura 1. La combinación de algunos de estos filtros puede ayudar, en la etapa de análisis estructural, a separar mejor los elementos de contenido.

Parte del pre-procesamiento consiste en detectar la inclinación del texto en la imagen o en el segmento de imagen a procesar, ya que dentro de una misma imagen puede haber textos con diferentes inclinaciones. Para incluir métodos de cálculo de inclinación se debe implementar la interfaz `ISkewMethod`.

```
public interface ISkewMethod
{
  double CalculateSkew(IImage image);
}
```

El método utilizado por la biblioteca OCR .NET para la detección del ángulo de inclinación de la imagen se basa en la transformada de **Hough** [11]. El método `CalculateSkew` retorna el ángulo de inclinación del texto en la imagen. Usando después el filtro `Rotate`, se procede a "enderezar" la imagen. El error común introducido en la inclinación durante el proceso de escaneo es de 2 a 7 grados. El método utilizado ha demostrado ser preciso (con solo un error de 0.5 grados) en inclinaciones de hasta 40 grados.

Note que la interfaz `ISkewMethod` no recibe como parámetro un mapa de bits, sino un objeto de tipo `IImage`. La interfaz `IImage`, implementada en la clase `OCR.Imaging.Image`, representa un envoltorio (*wrapper*) de un mapa de bits, que nos brinda un acceso más rápido y seguro a la información de la imagen, además de ofrecer más información al resto de los módulos del motor OCR.NET.

Análisis estructural

Esta es la etapa mediante la cual se determinan las regiones que pueden tener significado para el **clasificador**. Asimismo, se guardan de forma estructurada para una futura reconstrucción. Se ha diseñado una jerarquía de tipos (figura 2) para contener los diversos elementos de una página, donde se define la interfaz `IRegion`, que sirve como interfaz base a todos los elementos que pueden ser representados en el documento.

Cada una de las interfaces mostradas en la figura 2 cuenta con su respectiva implementación. Los mecanismos de análisis estructural son incorporados en OCR.NET mediante implementaciones de la interfaz `IRegionAnalizer`.

```
public interface IRegionAnalizer
{
 IRegion[] BuildRegions(
          IImage image,
          IImage original_image);
}
```

Se decidió también pasar como parámetro la imagen original, puesto que es posible que algunos algoritmos de deter-

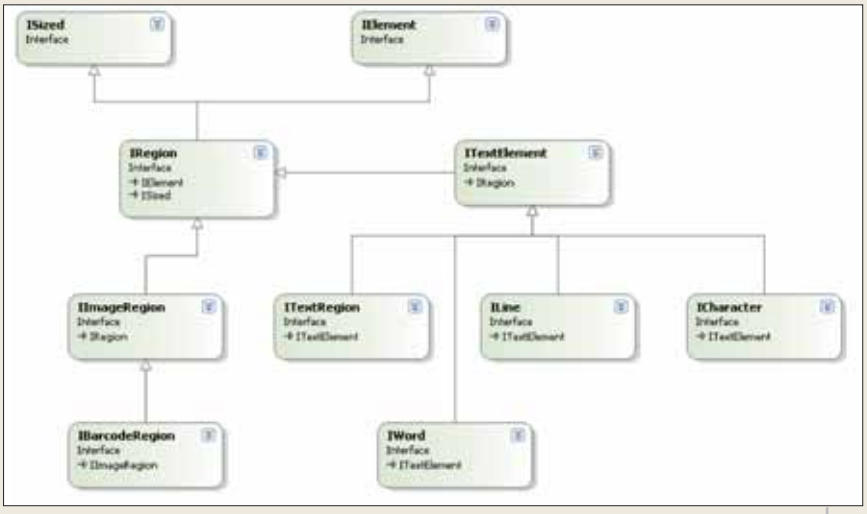


Figura 2. Interfaces utilizadas para representar elementos del documento

minación de regiones se basen en características del documento que se hayan transformado en el pre-procesamiento (por ejemplo, diversidad de colores). De igual manera, una vez detectadas las regiones imágenes, éstas deben extraer la porción correspondiente a la imagen original, y no a la procesada, para una futura reconstrucción.

Durante el reconocimiento de la imagen los analizadores realizan dos tareas principales:

- Clasificar las regiones de texto en caracteres, palabras, líneas y cuadros de texto.
- Localizar las imágenes y descartar áreas no útiles.

Durante el pre-procesamiento de la imagen, la misma quedará "binarizada" (proceso mediante el cual se transforman los píxeles de la imagen en blancos y negros de acuerdo al brillo que posean). De esta forma, los píxeles correspondientes al texto (generalmente oscuros) podrán ser "diferenciados" de los píxeles que forman el relleno o fondo del documento.

Detectar los grupos de píxeles negros conexos (*rounding boxes*) en la imagen "binarizada", partiendo de la primicia de que las letras en documentos impresos suelen estar separadas unas de las otras, es el primer acercamiento a los posibles caracteres de texto.

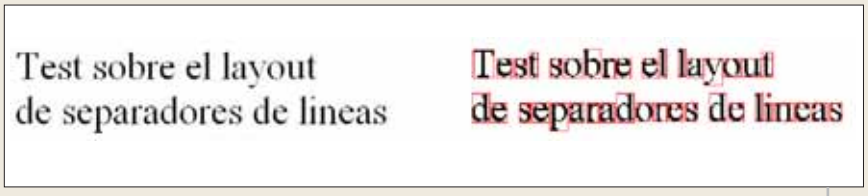


Figura 3. Texto con sus respectivos "rounding boxes"

La principal diferencia entre las regiones que corresponden a texto y a imágenes consiste en que el texto por lo general se muestra en agrupaciones alineadas con cierta regularidad en cuanto a tamaño, disposición, etc. Por tanto, se pueden considerar algunos criterios como descartar los rectángulos muy pequeños, o los muy alargados. Otro criterio que se sigue en este paso es determinar los "contenedores", grupos de píxeles que rodean áreas de posible texto.

Para esta etapa se brindan las clases:

```
class LeeRegionAnalizer :
          IRegionAnalizer { }
class DisjointBoxAnalizer :
          IRegionAnalizer { }
```

Ambas permiten determinar las componentes conexas: el primero por propagación, el segundo mediante operaciones sobre conjuntos disjuntos.

La clase `LeeRegionAnalizer` se basa en recorrer todos los píxeles y para cada píxel negro no marcado hacer un BFS, marcando todos los conexos a él y determinando así cada uno de los grupos conexos. Este algoritmo es simple y tiene un costo menor del doble de la cantidad de píxeles.

Por su parte, `DisjointBoxAnalizer` usa una implementación de una estructura de datos para el trabajo con conjuntos disjuntos (`DisjointSet`) que permite ejecutar la operación de unión en tiempo constante. Este algoritmo, al igual que el anterior, recorre dos veces el total de píxeles del documento, pero realiza un menor número de operaciones por píxel, lo que lo hace más eficiente en la práctica.

Por supuesto que no todos los píxeles en negro del documento después de la "binarización" corresponden a textos. Algunos corresponden a los píxeles más oscuros de las imágenes, bordes o incluso posibles manchas (muy común en documentos escaneados).

Para esta selección se hizo imprescindible una estructura que facilitara determinar qué regiones se encontraban en determinado rectángulo (`HashedBoxCollection`). Esta estructura suponía acceder directamente a las regiones utilizando un algoritmo de hash en forma bidimensional. Es decir, una rejilla bidimensional que referenciase a las regiones interceptadas con cada celda. De este modo, la cantidad de chequeos se hace relativa a la cantidad de regiones locales a las celdas en cuestión. De esta forma, se pueden

comprobar eficientemente algunos de los parámetros que se fijan como heurísticas para la clasificación en regiones-caracteres, regiones-ruido y regiones-contenedores.

Una vez se tienen los rectángulos correspondientes a posibles caracteres, se procede a determinar las palabras, líneas y cuadros de texto.

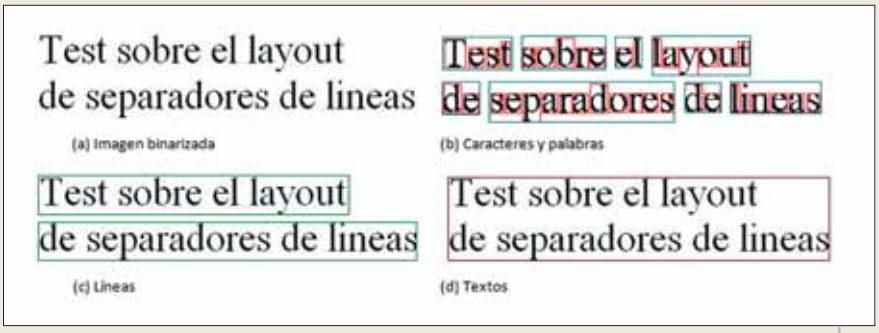


Figura 4. Elementos de un análisis estructural

Existen varios métodos para analizar la estructura del documento, clasificables en jerárquicos y no jerárquicos. Los jerárquicos son aquellos que asumen la existencia de una relación jerárquica entre los elementos de una página; por ejemplo la relación página/columna/párrafo/línea/palabra/carácter. Existen dos formas de reconstruir estas jerarquías: partiendo desde el menor elemento, o sea desde los rectángulos de los caracteres hasta llegar a los párrafos y columnas (*bottom-up* [2,3]), y a la inversa, o sea dividiendo la imagen progresivamente en columnas, párrafos, etc. hasta llegar a los caracteres (*top-down* [4,9]).

Bottom-up

Tomando cada rectángulo y ampliándolo cierta razón en dirección horizontal, se puede detectar si hay intersección con algún vecino. De esta forma se pueden determinar primero palabras, luego líneas y por último (ampliando los rectángulos correspondientes a las líneas en dirección vertical) los cuadros de texto. Esto resulta ventajoso por ser intuitivo, eficiente y por el hecho de que el propio análisis construye la estructura, pero no contempla posibles errores como la unión de caracteres y líneas. Además, resulta poco confiable para documentos "ruidosos" en los que han quedado secuelas (píxeles y marcas) de un mala digitalización.

Top-down

Un análisis *top-down* del problema se basa en un algoritmo para determinar rectángulos maximales [12]. Se trata de insertar en el documento rectángulos grandes y alargados, de modo que los cuadros de texto queden separados entre sí y de las imágenes, y puedan de este modo ser tratados localmente, a partir de proyecciones. Este método es ventajoso en documentos con ruido, pues un análisis del histograma horizontal de un párrafo ayudaría a la división por líneas de texto sin mucho riesgo de error, y un histograma vertical ayudaría a dividir palabras y caracteres. Claro está que para que este método funcione correctamente el fragmento a analizar debe encontrarse alineado.

Figura 5. Imagen dividida por maximales

Híbrido

En OCR.NET se desarrolló un híbrido entre estos dos métodos, aprovechando las ventajas y evitando las desventajas de cada uno de ellos. De esta forma, se tiene un algoritmo inicial para mezclar grupos de cajas cercanas, reduciendo el número de rectángulos, y luego se hace el análisis de arriba hacia abajo. Este híbrido se representa mediante la clase:

```
class HybridRegionAnalizer :
      IRegionAnalizer { }
```

La figura 6 muestra una aplicación de ejemplo en la que se puede observar la división realizada por el método híbrido. La parte superior de la figura muestra la imagen original escaneada, con líneas sobre ella determinando los diferentes niveles del análisis, es decir, párrafos, líneas, palabras y caracteres. En la parte inferior se muestra el texto que ha sido reconocido por el motor.

Clasificación

Una vez que han sido completadas las dos primeras etapas, se cuenta con una estructura del documento lista para reconocimiento. Ésta queda como muestra el árbol de la figura 7, donde el nodo raíz sería el documento o página y los nodos hojas los caracteres. Construido con las interfaces mostradas en la figura 2, es posible recorrer este árbol que hasta ahora no contiene líneas ni palabras o caracteres, sino rectángulos que determinan las zonas

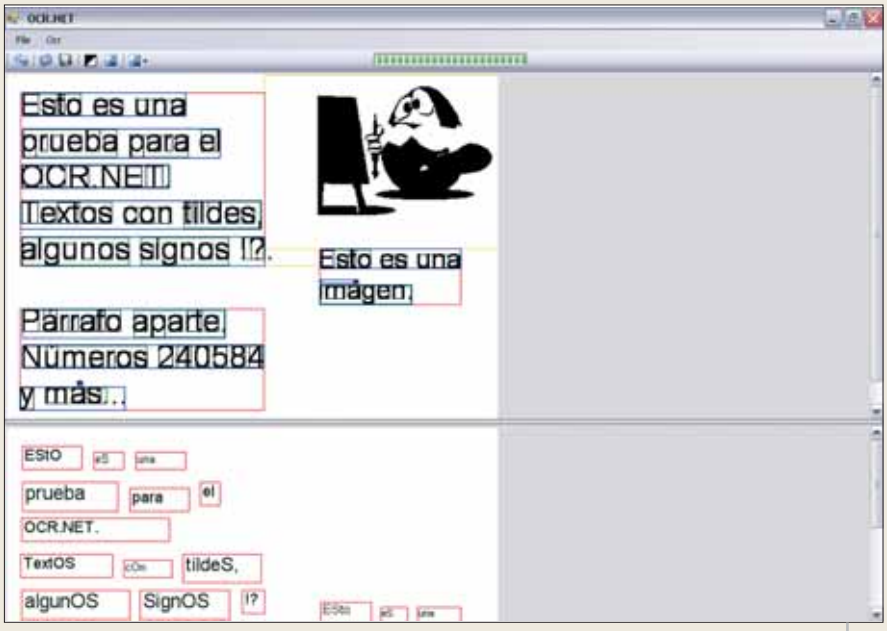


Figura 6. Aplicación de ejemplo que usa la biblioteca OCR.NET

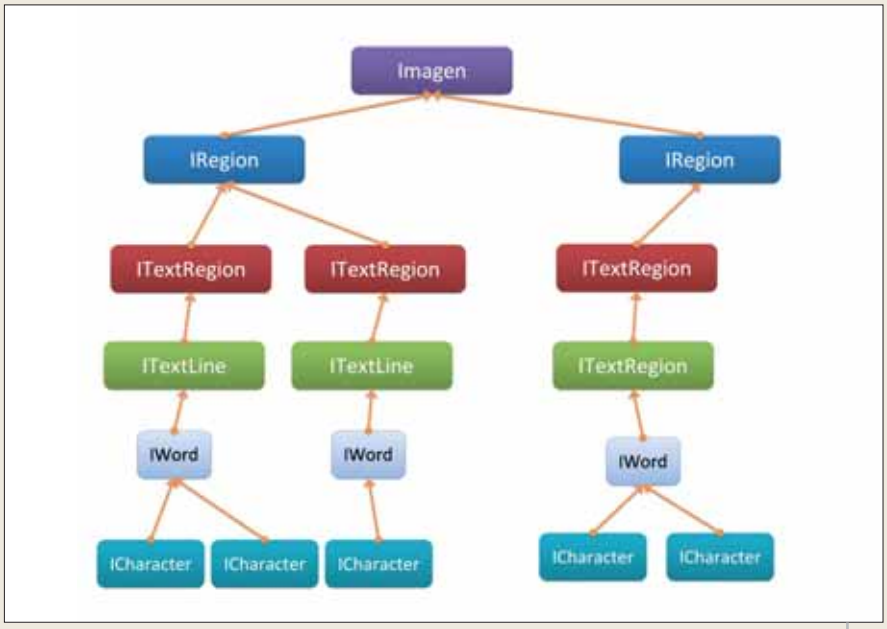


Figura 7. Estructura del árbol luego del análisis estructural

donde se encuentran enmarcados cada uno de estos elementos, como se muestra en la parte superior de la figura 6. El proceso de clasificación consiste en determinar, dada la imagen de un carácter, la "clase" a la que pertenece. Esta clase no es más que el valor del carácter como cadena; es decir, si se tiene la imagen de la letra "A", su clasificación debe retornar la cadena cuyo valor ASCII es 65, o sea el carácter o letra "A". Muchos clasificadores definen las clases usando el tipo `char`, pero en nuestro caso, anticipando el reconocimiento de caracteres unidos, se ha utilizado el tipo `string` para identificar la clase.

Evidentemente, la clasificación de la imagen resulta engorrosa a menos que se intenten extraer ciertas "características" de dicha imagen y luego trabajar en estas características.

Existen dos etapas en este proceso: primero, la extracción de características o creación del vector de características, y segundo, la clasificación del vector en una de las clases posibles. En el proceso de OCR, el conjunto de clases posibles está formado por los posibles símbolos a identificar; en el presente caso, símbolos alfanuméricos, signos de puntuación y demás. La forma en que se crean inicialmente estas clases se llama "entrenamiento" y será discutida posteriormente, dado que este proceso es muy dependiente del mecanismo de clasificación que se utilice.

Extracción de características

Los vectores de características se conforman aritméticamente por un vector numérico:

$$V = (c_1, c_2, .., c_n)$$

Donde `n` es el orden del vector y cada elemento c_i es el valor asociado a una característica del carácter o elemento a clasificar, como puede ser la altura de la letra, la densidad de píxeles negros dentro de la imagen o la cantidad de espacios en blancos, entre otras. Estos vectores son definidos mediante la interfaz `IFeatureVector`.

```
public interface IFeatureVector
{
   double[] Components { get;}
   FeatureVectorType Type { get;}
}
```

La creación de estos vectores de características está a cargo de las clases que implementen la interfaz `ICharacterFeatureExtractor`.

```
public interface ICharacterFeatureExtractor
    {
        IFeatureVector GetFeatures(IImage image, IRegion region);
    }
```

El método `GetFeatures` recibe como parámetros un objeto `IImage` y la región que contiene al carácter dentro de esa imagen. Retorna entonces un vector de características extraídas del carácter contenido en dicha región. En caso de que la región especificada sea `null`, se asume que toda la imagen representa un carácter y será procesada como tal.

Para la extracción de características es necesario que las muestras a procesar tengan el mismo tamaño, así como que la imagen esté centrada y acotada con los bordes. A este proceso se le llama normalización, y es posible dentro de la biblioteca mediante clases que implementen la interfaz `INormalizer`. La implementación que se ofrece de este método usa filtros de rotación y ajuste de tamaño.

Diversos métodos son utilizados para la extracción de características en imágenes: desde métodos simples, como considerar la matriz de puntos de la imagen como un vector, evaluación de perfiles horizontales y verticales (figura 8), hasta complejas evaluaciones como los momentos de Zernike o Lagrange.

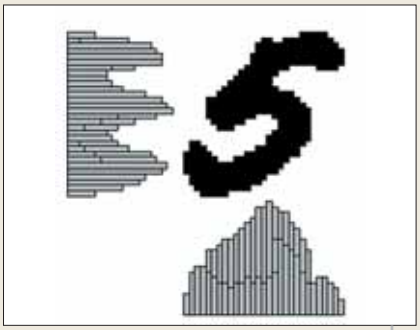


Figura 8. Perfiles V y H

OCR.NET ofrece implementaciones de varias de estas funciones, como son:

- Perfil de matriz.
- Perfil de proyecciones verticales y horizontales.
- Perfil de matriz con "grayscale zoning".
- Momentos de Lagrange.

Cada una de estas formas de extraer características posee ventajas y limitaciones. Los perfiles de proyección son calculados eficientemente, pero no generan la variabilidad necesaria para obtener una alta precisión a menos que se aumente el tamaño del vector y la cantidad de elementos de muestra; mientras que los momentos de Lagrange ofrecen una increíble variabilidad, pero son costosos de calcular. Por lo tanto, el uso de alguno de estos métodos o incluso su combinación en un vector híbrido será opción del cliente de la biblioteca que pretenda utilizar el motor para tareas más específicas.

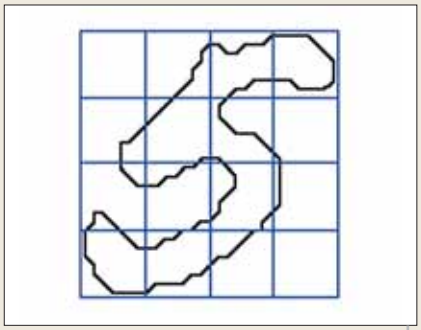


Figura 9. Perfil de matriz de 4x4

Clasificadores

Los clasificadores son una parte fundamental dentro del proceso de OCR. Los mecanismos de clasificación más conocidos son las redes neuronales, perceptrones multicapa, fuerza bruta, vecino más cercano y árboles de decisión. La combinación de varios de ellos puede ayudar a incrementar la precisión de la clasificación.

La presente biblioteca incluye como clasificador un árbol de búsqueda, que como variante para resolver el problema del vecino más cercano en R^N utiliza un ordenamiento por distancias relativas basado en el

presentado por **Tomas E. Portegys** de AT&T [1]. Se ha modificado dicho algoritmo para facilitar la búsqueda de los *N* vecinos más cercanos con la utilización de colas con prioridad.

Con idea de implementar dicho algoritmo, se hace necesario escribir soporte para el cálculo de distancia entre vectores; dicho soporte es incorporado a través de la interfaz `IDistanceFunction`.

```
public interface IDistanceFunction
{
 double Eval(IFeatureVector a, IFeatureVector b);
}
```

Varias funciones de distancia han sido implementadas. La más utilizada por la bibiloteca, y por las aplicaciones que se han creado hasta ahora, ha sido la distancia Manhattan [1]. Teniendo esta distancia como base se implementó el árbol de clasificación. Un clasificador es representado en código por la interfaz:

```
public interface IClassifier <T>
{
   IList<IClassMatch<T>>
      Classify(IFeatureVector vector, int max);
}
```

El método `Classify` recibe como parámetros un vector de características a clasificar y un entero que representa el mayor número de aciertos que serán devueltos. El clasificador es una interfaz genérica, por lo que es posible utilizar los clasificadores para otros elementos y no sólo para caracteres. Los resultados del proceso de clasificación son representados por instancias que implementan la interfaz `IClassMatch`. El valor de la propiedad `Confidence` representa el nivel de similitud con el vector, es decir, la distancia entre el vector que se le pasó al método `Classify` y el vector asociado al elemento almacenado en el árbol y cuya clase es retornada en la propiedad `ClassValue`.

```
public interface IClassMatch <T> :
                    IComparable<IClassMatch<T>>
{
    double Confidence { get;}
    T ClassValue { get;}
}
```

La clase siguiente es la que implementa el árbol que se utiliza para clasificar caracteres:

```
public class SearchTreeClassifier<T> :
      IClassifier<T>,ITrainableEngine<T> { }
```

Note que el árbol no solo es un clasificador genérico, sino que es además un motor capaz de ser entrenado o de aprender. Esto se realiza a través de la interfaz `ITrainableEngine`, que facilita el entrenamiento y persistencia del clasificador.

```
public interface ITrainableEngine<T>
{
    void Train(Dictionary<IFeatureVector,T>
               collection);
    void Save(Stream output);
    void Load(Stream input);
}
```

En el caso del árbol que se ha utilizado, el proceso de entrenamiento consiste en la inserción de los respectivos pares de vectores y clases dentro del mismo. Las clases en éste son representadas por el parámetro genérico `T`. En la biblioteca en cuestión se instancia el árbol usando como parámetro genérico a la interfaz `ICharInfo`.

```
public interface ICharInfo
{
    string FontFamily { get; }
    string Value { get;  }
    bool IsBold { get; }
    bool IsItalic { get; }
```

La interfaz `ICharInfo` contiene no sólo el valor en cadena del carácter reconocido (aunque podría ser una cadena de más de un carácter), sino que también contempla otros aspectos, como el tipo de letra y estilo. Una herramienta de entrenamiento (figura 10) provista junto con el código de OCR.NET ofrece la posibilidad de entrenar la biblioteca utilizando diversos tipos de letras de los instalados en el sistema. Permite además estudiar el comportamiento de los métodos de clasificación y de las funciones de extracción de características para diversos tamaños y tipos de letra. Es posible crear con esta aplicación

un juego de caracteres con diversos tipos de letras y luego crear un árbol y hacerlo persistente.

permite continuar "enseñando" a la biblioteca nuevos caracteres mientras se utiliza la aplicación.

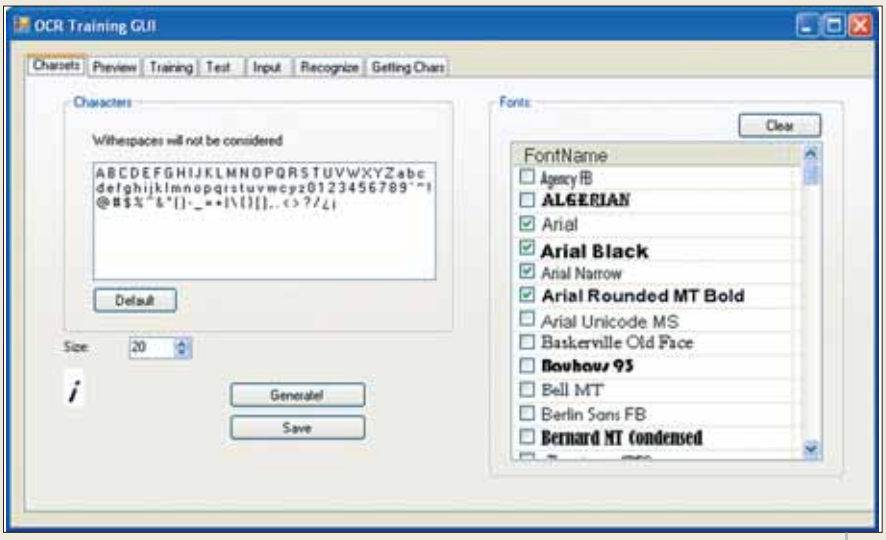


Figura 10. Interfaz de entrenamiento de OCR.NET

Esta pequeña aplicación sirve como paso inicial, pero una vez que se tiene entrenada la biblioteca es posible introducir nuevos elementos al árbol en cualquier momento. Por lo que la misma interfaz de reconocimiento de OCR.NET

Conclusiones

Sobre el motor implementado para el reconocimiento óptico de caracteres se han realizado pruebas con más de 100 documentos, con resultados satisfactorios, para un árbol entrenado con alrededor de 7000 patrones. Actualmente se trabaja para aumentar el rendimiento de la biblioteca e incorporar un mejor mecanismo para la separación de caracteres unidos.

Como consecuencia de su diseño genérico y estructurado, muchos de los métodos implementados pueden ser utilizados como base para otros mecanismos de reconocimiento que no sean caracteres, como por ejemplo códigos de barras.

La biblioteca desarrollada consta de más de un centenar de clases e interfaces, diseñadas para soportar no solo el reconocimiento de caracteres sino también el análisis estructural de documentos. Esta biblioteca incluye la implementación de varios algoritmos para cada uno de los procesos involucrados en la acción de reconocer caracteres.

Aunque se han implementado gran parte de los métodos, quedan aún muchos por incorporar. El código completo está disponible en el sitio Web de **dotNetManía** y se distribuye bajo licencia LGPL. Exhortamos a todos los entusiastas programadores de .NET a revisar el código, aportar ideas y completar algunas implementaciones.

Referencias

[1] **Thomas E. Portegys.** A Search Technique for Pattern Recognition Using Relative Distances. IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 17, NO. 9, september 1995.

[2] **D. Drivas** y **A. Amin**. A Page Segmentation and Classification Utilizing Bottom-Up Approach, Proc. Third Int'l Conf. Document Analysis and Recognition, pp. 610-614, 1995.

[3] **A. Simon**, **J. Pret** y **A. Johnson**. A Fast Algorithm for Bottom-Up Document Layout Analysis, IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, pp. 273-276, 1997.

[4] **J. Ha**, **R. Haralick** y **I. Phillips**. A Recursive X-Y Cut Using Bounding Boxes of Connected Components. Document Analysis and Recognition, pp. 952-955, 1995.

[5] **J. Ha**, **R. Haralick** y **I. Phillips**. A Document Page Decomposition by the Bounding-Box Projection Technique. Document Analysis and Recognition, pp. 1119-1122, 1995.

[6] **A. Jain** y **Y. Zhong**. A Page Segmentation Using Texture. Analysis, Pattern Recognition, vol. 29, pp. 743-770, 1996.

[7] **A. Jain** y **S. Bhattacharjee**. A Text Segmentation Using Gabor Filters for Automatic Document Processing, Machine Vision and Applications, vol. 5, pp. 169-184, 1992.

[8] **H. Cheng** y **C. Bouman**. A Trainable Context Model for Multiscale Segmentation. Image Processing, vol. 1, pp. 610-614, 1998.

[9] **R. G. Casey** y **G. Nagy**. Recursive segmentation and classification of composite character patterns. In Proc. 6th Int. Conf. Pattern Recognition, pages 1023-1026, Munich, 1982.

[10] **Thomas E. Bayer** y **H. Ulrich**. Cut Classification for Segmentation

[11] **S. C. Hinds**, **J. L. Fisher** y **D. P. D'Amato**. "A Document Skew Detection Method using run-length encoding and the Hough Transform", Conf of Pattern Recognition, 1990, pp 464-468.

[12] **M. Breuel**, **Thomas**. "Two geometric algorithms for Layout Analysis", pp. 2-5.

Microsoft®
TechEd Developers
2007

Si eres desarrollador, durante CINCO días de profundo contenido técnico, mejorarás tus habilidades para construir aplicaciones más seguras, escalables y con las últimas tecnologías con las herramientas de desarrollo de Microsoft.

Del 5 al 9 de noviembre de 2007 en Barcelona

www.microsoft.com/europe/teched-developers/

msdn subscriptions

Microsoft®

Gustavo Vélez

Migración de SharePoint 2003 a 2007

Puede ser fácil... pero no siempre

En este artículo se describen de una manera práctica algunas consideraciones técnicas y operacionales para una migración exitosa de SharePoint 2003 a SharePoint 2007. ¿Tenemos razones para hacerlo?

Introducción

Con la nueva ola de productos de Office 2007, Microsoft liberó también las nuevas versiones de Microsoft SharePoint Services (WSS) y Microsoft Office SharePoint Server (MOSS), los reemplazos de WSS y SPS 2003. SharePoint es una herramienta de colaboración que permite a los usuarios trabajar y compartir información de una forma eficiente y segura. La nueva versión incluye utilidades y procedimientos para migrar SharePoint 2003 a su equivalente de 2007.

¿Es necesario migrar?

WSS y MOSS 2007 introducen una serie de mejoras en comparación con las versiones 2003. Fuera de perfeccionamientos en las prestaciones por la utilización de los progresos en rendimiento que se puede realizar con SQL Server 2005, las mejoras en infraestructura y la mejor arquitectura resultan en un uso más eficiente de los recursos en las bases de datos: los portales ahora son colecciones de sitios, con solamente una base de datos para cada uno. Al mismo tiempo, el filtro ISAPI no es necesario por la utilización más efectiva de los fundamentos de IIS, lo que mejora considerablemente la reutilización de los servidores para ejecutar aplicaciones en paralelo y la estabilidad operacional del sistema.

La integridad de la información ha sido garantizada en la nueva versión por medio de la utilización de la papelera de reciclaje, que hace posible recuperar información eliminada por el usuario, y en casos extremos, por los administradores del sistema, y por mejoras en los sistemas de respaldo. Además, la máquina de búsqueda ha sido completamente rediseñada y, gracias a que WSS ya no necesita la opción de búsqueda de texto (*full-text search*), es posible utilizar versiones de 64 bits de SQL Server; además, las diferencias entre WSS y MOSS en cuanto a la forma de utilizar la máquina de búsqueda han desaparecido.

Para los usuarios los cambios son radicales: *blogs*, *wikis*, RSS, flujos de trabajo, integración de datos de negocios con el Catálogo de Datos Profesionales, los servicios integrados de Excel, formularios de InfoPath, son algunas de las mejoras y novedades. Para los desarrolladores también se abren nuevas posibilidades con la utilización de .NET 2.0 (con lo que se hace posible utilizar páginas maestras), las características y proyectos de SharePoint, y la ampliación y mejora de la API de programación, que permiten utilizar SharePoint más como una plataforma de desarrollo que como una simple aplicación.

Cómo migrar

Microsoft ha diseñado tres estrategias de migración: "En sitio", "Gradual" y "Migración de bases de datos".

Una migración "En sitio" se utiliza para actualizar todos los sitios y el portal de una sola vez y

Gustavo Vélez es ingeniero mecánico y electrónico, especializado en el diseño, desarrollo e implementación de software (MCSD) basado en tecnologías de Microsoft, especialmente SharePoint. Es creador y webmaster de http://www.gavd.net/servers, y trabaja como *senior developer* en **Winvision** (http://www.winvision.nl)

Tipo de migración	Ventajas	Desventajas
En sitio	- Fácil de realizar - Usa el hardware original	- El sistema no se puede utilizar durante la migración - Imposible regresar a la situación original
Gradual	- Fácil de regresar a la situación original - Migración de cada colección de sitios a la vez (gradual) - Usa el hardware original - Mejor que "En sitio" si se han hecho muchas personalizaciones	- Necesita mantener WSS o SPS 2003 instalado - Necesita mucho más espacio físico (disco duro) - Redirección de los sitios durante el proceso - Migración mucho más lenta
Base de Datos	- Conexión/desconexión de bases de datos fácil de realizar - Migración mucho más rápida - Una base de datos a la vez - Oportunidad para mejorar el hardware, si es necesario	- Solamente se pueden migrar bases de datos completas - Imposible regresar a la situación original - El sistema no se puede utilizar durante la migración - Búsqueda y personalizaciones no son migradas - Imposible migrar por colección de sitios

Tabla 1: Resumen de escenarios de migración

automáticamente. Es la mejor estrategia para instalaciones contenidas en un solo servidor o para granjas de pequeñas dimensiones. En este modo de migración, la versión antigua de la base de datos es sobrescrita con la versión nueva, y los archivos físicos de la aplicación son cambiados a la nueva estructura; al final, la estructura y archivos antiguos son eliminados. El proceso ocurre sobre el mismo hardware utilizado por la versión 2003, y no es posible volver a la situación original (a menos que se reinstale todo el sistema de nuevo con copias de reserva). Durante la migración, ningún sitio del portal podrá ser utilizado, y las direcciones de todas las páginas serán conservadas.

La migración "Gradual" o "Lado a lado" permite un mayor control sobre el proceso, pues se puede especificar qué sitios se deben migrar, en qué orden, y cuando hacerlo. En este escenario, el contenido de los sitios es copiado primero en una base de datos temporal y luego en la versión definitiva de 2007, conservando los datos originales en la base de datos de 2003. Los cambios se pueden revertir fácilmente, y la operación ocurre sobre el hardware original. Durante la migración, solamente los sitios que se están migrando en cada momento no estarán disponibles, y el proceso crea URL alternativas para los sitios originales.

Por último, una "Migración de Bases de Datos" es de hecho una migración "En sitio" que se realiza con copias de las bases de datos en servidores diferentes a los utilizados por SharePoint 2003. Las bases de datos de contenido se ponen fuera de conexión en el sistema con 2003, se copian en los nuevos servidores y se ponen de nuevo en conexión; luego, utilizando la Administración Central de SharePoint, se indica en la base de datos de configuración que hay que utilizar las BBDD de contenido recién copiadas y se inicia un proceso de migración "En sitio". El sistema original de 2003 debe ser detenido para evitar pérdidas de información durante el proceso de sincronización.

Preparación antes de migrar, la mitad del trabajo

Como en cualquier otro tipo de migración de sistemas, es muy importante hacer una buena preparación antes de realizar la operación para evitar la pérdida de información. Es indispensable crear copias de respaldo de todo el sistema, tanto de las bases de datos como de SharePoint mismo (las herramientas de que disponen SQL Server y SharePoint para esto son más que suficientes). Realice además copias de respaldo después de cada paso si está usando una migración "Gradual" o de "Bases de Datos".

Una buena estrategia de preparación es realizar primero una o más migraciones en equipos paralelos de prueba antes de realizarla en los servidores de producción (la virtualización es una manera ideal para efectuarlo). La comunicación con los usuarios, administradores, diseñadores y desarrolladores es muy importante para evitar malentendidos, sobre todo si la migración tomará bastante tiempo (migración "En sitio") o será realizada en una forma gradual.

En cuanto al hardware, hay que tomar las medidas necesarias para que los servidores tengan el sufi-

ciente espacio de disco duro, capacidad de cálculo y memoria. El proceso de migración requiere bastante tiempo, en el que la CPU de los servidores estará funcionando prácticamente al 100% constantemente (una migración puede durar varios días), y requerirá toda la memoria RAM disponible; si hay otras aplicaciones ejecutándose en el mismo hardware, probablemente los tiempos de reacción de éstas se deteriorarán considerablemente. Tenga en mente cosas básicas de infraestructura, como por ejemplo que no se produzca un corte de electricidad durante el proceso, sobre todo en una migración "En sitio".

En lo relativo al software, instale primero el *service pack* 2 de WSS y SPS 2003 y todos los *service packs* de SQL y Windows antes de empezar la migración, y asegúrese de que todos los sitios siguen funcionando sin problemas después de la instalación.

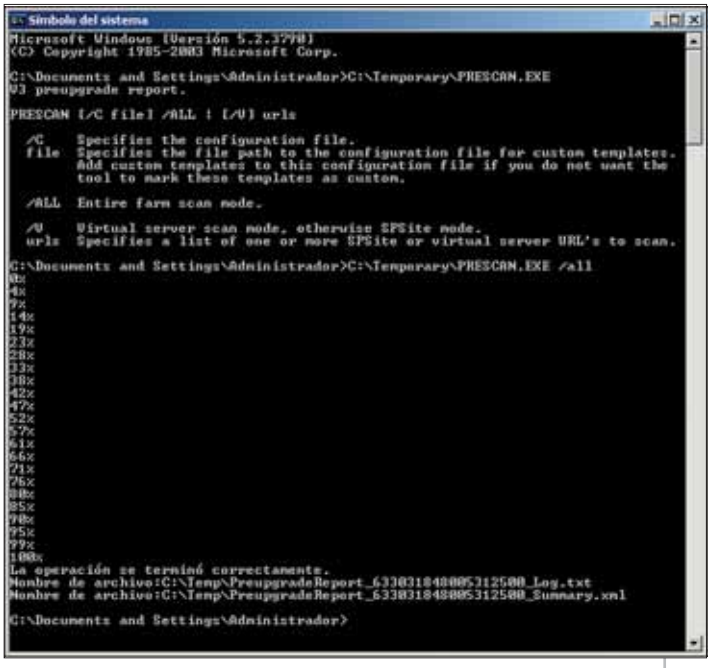


Figura 2: Resultado de la herramienta de análisis

Se puede migrar con poco trabajo...

Probablemente, la forma más fácil de migrar es "En sitio". Simplemente inicie la rutina de instalación de SharePoint 2007 en el (o los) servidor(es) de SharePoint 2003, y la rutina misma detectará automáticamente que existe una versión anterior, y preparará todo el terreno para la migración.

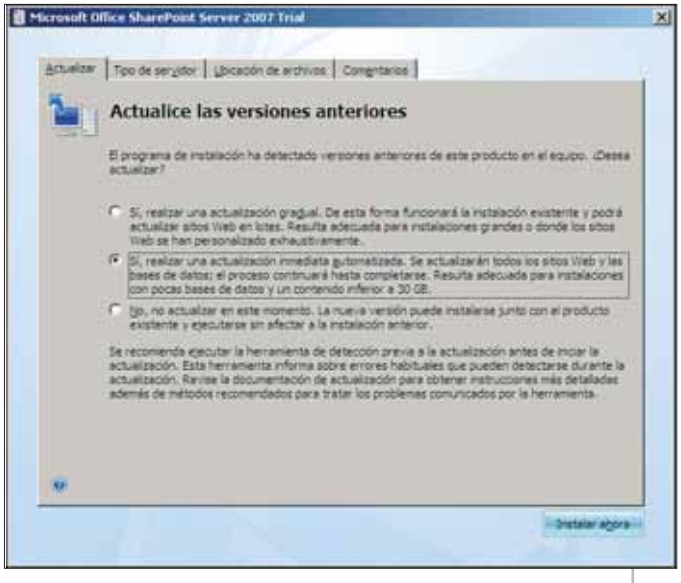


Figura 1: Rutina de instalación que ha detectado una posible migración desde SharePoint 2003

Durante la instalación, la herramienta de análisis `prescan.exe` es colocada automáticamente en el directorio `C:\Archivos de programa\Archivos comunes\Microsoft Shared\web server extensions\12\BIN` (se puede encontrar también en los discos de instalación, en el archivo comprimido `C:\Archivos de programa\MSECache\oserver12\Global\Wss\sts.cab`). Es muy importante ejecutar la herramienta antes de completar la migración para asegurarse que no hay problemas insolubles en SharePoint 2003 (sitios bloqueados, cuotas que se han sobrepasado, sitios huérfanos). La herramienta genera dos archivos de registro: un archivo `.txt` con un análisis extensivo, y otro `.xml` con un resumen de los resultados.

La rutina de instalación de SharePoint sigue el mismo procedimiento al seguido por una instalación normal, y solamente al final, bajo la Administración Central se pueden ver las diferencias. Una migración "En sitio" inicia de inmediato y automáticamente el proceso de actualización, creando toda la infraestructura necesaria para los sitios y aplicaciones. En una migración "Gradual" se deben crear primero los servicios compartidos necesarios, configurar los servicios de los servidores e iniciar manualmente la actualización desde la sección "Operaciones" | "Actualización" y migración de la Administración Central; luego hay que indicar qué colección de sitios se desea migrar. Evite crear nuevas aplicaciones Web antes de iniciar la actualización para soslayar conflictos con los sitios a migrar.

En cualquiera de los casos (para una migración de bases de datos el proceso es similar luego de

Figura 3: Estado de la migración

En general, una instalación de SharePoint 2003 (WSS o SPS) sin personalizaciones será migrada sin mayores problemas. Áreas, autorización y autenticación, sitios, WebParts (inclusive WebParts programadas para SharePoint 2003) y todos los documentos e información serán convertidos al nuevo sistema. Al finalizar la actualización, es necesario revisar extensivamente el portal para detectar posibles fallas. La figura 4 muestra un portal 2003 estándar que ha sido migrado a MOSS 2007: la navegación cambia consecuentemente, pero toda la funcionalidad está presente.

Aunque migrar puede significar también mucho trabajo…

Las WebParts creadas para SharePoint 2003 y compiladas con .NET 1.1 pueden ser utilizadas sin problemas con SharePoint 2007. Si se ha protegido el ensamblado con algún ofuscador, la WebPart no será reconocida en el nuevo sistema, y será necesario recompilarla. Si los ensamblados se encuentran originalmente en el directorio `bin` de IIS, es necesario instalarlos de nuevo en la estructura creada por SharePoint 2007 (si han sido instalados en la GAC

> La preparación de la migración es tan importante como la migración misma y debe ser planificada con el mismo cuidado

conectar las bases de datos respectivas), la Administración Central ofrece una página para seguir el proceso de actualización, con información sobre el estado, modo de migración, acciones, etc.

Si ocurre un error, o la rutina llega a un punto en el que no puede realizar alguna de sus acciones, el proceso se detiene indicando el problema, y dando la posibilidad de solucionarlo y continuar con la migración; en este caso, el archivo de registro indicado al principio del artículo contendrá información completa sobre las complicaciones encontradas. Al final, cuando la actualización se ha terminado con éxito, la página muestra un mensaje indicándolo.

Figura 4: Comparación de un Portal de SharePoint antes y después de la migración

no existe este problema), lo mismo que los archivos de recursos y los archivos `.dwp` con la definición de la WebPart. Si las WebParts utilizan clases o métodos obsoletos en .NET 2.0, es mejor cambiarlos por los utilizados en la nueva versión y recompilar.

Los cambios en el tema de sitios WSS deberán ser aplicados manualmente de nuevo, pues no serán migrados. Las páginas que hayan sido modificadas con FrontPage (páginas "ghost") pueden ser devueltas a su estado original, o migradas tal como están, aunque en este último caso los resultados pueden ser sorpresivos. Si es indispensable aplicar las modificaciones de nuevo, desde FrontPage se pueden exportar los cambios, y luego re-aplicarlos en el nuevo sistema.

> No es de esperar que ocurran problemas en la migración de sistemas sin personalizaciones

Puede ocurrir que modificaciones más complejas realizadas en SharePoint 2003 sean difíciles o incluso imposibles de migrar automáticamente. En general, los controles Web (utilizados principalmente para menús) creados especialmente y aplicados en el sistema original no son reconocidos ni migrados, y es necesario generarlos de nuevo con .NET 2.0 y aplicarlos manualmente en las páginas maestras del nuevo sistema. Lo mismo ocurre con páginas personalizadas localizadas en el directorio `layouts` y todas las modificaciones realizadas en las plantillas por defecto de SharePoint 2003. Cualquier modificación realizada en los archivos de configuración `onet.xml` y schema.xml no serán reconocidas ni migradas, y tienen que volverse a realizar en el sistema actualizado (SharePoint 2007 contintúa utilizando CAML en los archivos de configuración, pero la estructura del código ha sido cambiada para adaptarlo al sistema de páginas maestras). Los servicios Web creados especialmente tampoco son reconocidos ni actualizados al nuevo sistema, aunque probablemente continuarán funcionando sin problemas.

Los cambios realizados en "Mi Sitio" que van más allá de la instalación de WebParts no son migrados y deben ser realizados de nuevo manualmente. Los manejadores de eventos para las Librerías de Documentos sí son actualizados, pero debido a la posibilidad de utilizar los nuevos y mucho más poderosos Flujos de Trabajo, es su función relativamente obsoleta.

Dependiendo del tamaño y cantidad de documentos en el sistema, una migración puede durar entre algunas horas y algunos días. La migración "De Bases de Datos" es la forma más rápida, seguida por "En sitio", con "Gradual" como la forma más lenta debido a la cantidad de datos que es necesario copiar. Una estimación muy general es que se pueden migrar entre 15 y 20 Gb por hora, aunque esto depende completamente del tipo de hardware utilizado y la complejidad de los sitios. Sitios con estructura muy complicada o con grandes cantidades de documentos pueden ser migrados a una velocidad aproximada de 10 Gb por hora. La mejor forma de obtener una buena estimación es hacer una migración en un servidor paralelo de una parte de los datos, y revisar el archivo de registro (donde se registra el tiempo de cada operación). Hay que tener en cuenta que el proceso de preparación puede tomar mucho más tiempo que la migración física de los servidores, y se debe agregar en la planificación del proyecto. Igualmente, si el portal tiene gran cantidad de personalizaciones, hay que añadir el tiempo necesario para resolver todas las dificultades que se van a encontrar por el camino.

Conclusiones

Definitivamente, Microsoft ha realizado esta vez un mejor trabajo para crear estrategias de migración de SharePoint de 2003 a 2007, que el que hizo hace algunos años cuando era necesario migrar de SharePoint 2001 a 2003. Se pueden utilizar diferentes modos de actualización, dependiendo del tipo de instalación y las necesidades particulares, y la migración ocurre de una manera simple y segura (aunque lenta) siempre y cuando no se le hayan hecho modificaciones profundas al sistema de 2003.

Cuando el sistema ha sido personalizado de una forma extensiva, es más que seguro que se encontrarán problemas en la migración; con realizar algún trabajo de re-compilación y re-instalación de las personalizaciones se puede llegar bastante lejos, aunque no hay ninguna garantía de que todo pueda ser migrado; sobre todo si se han realizado modificaciones en los archivos por defecto de SharePoint 2003, se pueden esperar problemas técnicamente difíciles de solucionar, si bien siempre será posible migrar la información (documentos) al nuevo sistema, aunque las personalizaciones se pierdan, o haya que programarlas de nuevo.

Joan Llopart

Microsoft Solutions Framework (II)

Aplicar la metodología en el trabajo diario según Microsoft

Este mes damos continuación al primer artículo de esta serie, publicado el mes pasado, en el que se realizó un primer acercamiento a la metodología que Microsoft pone a disposición de los ingenieros y administradores para realizar de forma más eficiente y organizada su trabajo.

En el caso concreto de MSF, no está de más recordar que se trata de un conjunto de modelos y herramientas que describen pasos y perfiles necesarios en el desarrollo de proyectos, tanto para el desarrollo de software como para la implantación de sistemas informáticos.

Una vez se ha descrito el Modelo de Equipo (*Team Model*), que detalla los roles necesarios para contar con todos los puntos de vista necesarios para un proyecto, y se han descrito las tres disciplinas auxiliares que sirven como elementos de apoyo a los dos modelos principales, queda por adentrarse en la parte más compleja e interesante de MSF: el Modelo de Procesos (*Process Model*). Sin duda, se trata del elemento clave de la metodología, dado que describe las fases de que consta un proyecto y en qué forma deben ser gestionadas.

Dada la amplitud de esta metodología, estos dos artículos buscan proporcionar una visión global que permita introducirse en sus principios y aplicarla a la realidad cotidiana.

Principios básicos de MSF

Aplicar MSF implica conocer sus modelos, pero sobre todo tener en mente en todo momento los principios básicos que guían su utilización. Estos son los 8 principios fundacionales que describen la filosofía de MSF.

- Seguimiento claro, responsabilidad compartida
- Delegar en los miembros del equipo
- Centrarse en aportar valor
- Visión compartida del proyecto
- Mantenerse ágil y adaptarse al cambio
- Fomentar las comunicaciones abiertas
- Aprender de todas las experiencias
- Invertir en calidad

Visión inicial del Modelo de Procesos

La gente con experiencia en la gestión de proyectos suele tener una visión más o menos similar de las fases que lo componen y qué tareas deben realizarse para llevarlo a cabo correctamente. A primera vista, el Modelo de Procesos es bastante similar a otras metodologías ampliamente aceptadas, tanto por la cantidad como por la definición que realiza de las distintas fases. Pero lo que diferencia a MSF de otras metodologías tradicionales no es tanto su esquema básico como la forma en que se aplica.

Algunas metodologías se sustentan en el Modelo de Cascada (*Waterfall Model*), consistente en definir unos hitos (*milestones*) que indican la finalización de las tareas del proyecto y que se ejecutan secuencialmente durante el tiempo de desarrollo. Así, una fase precede a la siguiente y debe ser terminada completamente para pasar a realizar la tarea posterior. Este enfoque suele ser demasiado encorsetado para algunos proyectos, por lo que han aparecido metodologías con un modelo en espiral, donde las dis-

Joan Llopart es Project Manager y arquitecto de Raona. Joan es MCSE, MCDBA, MCT, ITPro y MSF Certified.

tintas funcionalidades se desarrollan al mismo tiempo y evolucionan cíclicamente. Este sistema es ágil y dinámico, ideal para desarrollos pequeños, pero es muy complejo de aplicar en proyectos grandes y con múltiples requerimientos.

Para intentar solventar estas limitaciones, MSF recoge lo mejor de ambos modelos y los fusiona, para dar un método generalista aplicable a todo tipo de proyectos. El control y seguimiento del proyecto se basa en el uso de hitos definidos que permiten conocer si el proyecto avanza dentro lo previsto, pero la ejecución de las tareas no se restringe a su cumplimiento. MSF recomienda iniciar cualquier tarea lo antes posible y cerrarla tan tarde como sea necesario, para así afrontar flexiblemente los cambios que afectan a todo proyecto y que suelen desembocar en retrasos. Uno de los principios de MSF es muy claro en este sentido: "stay agile, expect change", y ese dinamismo es una de las principales características de esta metodología.

Aplicación del modelo de procesos

Todo proyecto se divide en fases, y MSF no es una excepción. En este sentido, MSF no presenta excesivas diferencias en cuanto a su definición y separación respecto a metodologías tradicionales. Opta por definir cinco fases sucesivas que corresponden a los nombres de *envisioning*, *planning*, *developing*, *stabilizing* y *deploying*, identificadas por la imagen adjunta en la figura 1. Respecto a esta clasificación, cabe destacar la inclusión de una fase de despliegue en el entorno de producción como parte integrante del proyecto, dado que muchas veces no se contempla su realización y suele ser fuente de numerosos problemas de definición del alcance.

Estas fases son un guión adaptable a las necesidades del proyecto, dado que son más indicativas que restrictivas. Cada una de ellas finaliza con un *major milestone*, un hito general claro y evaluable que permite determinar claramente que la fase se ha completado con éxito. Adicionalmente, cada fase tiene

Figura 1
Esquema básico del *Process Model*

un conjunto de *interim milestones* definidos que identifican a grandes rasgos las tareas a realizar y cómo debe afrontarse su ejecución. En todo caso, MSF identifica sólo etapas generales que deben ser adaptadas a cada proyecto y asociar cómo serán llevadas a cabo.

Para entender correctamente este modelo de fases, es imprescindible tener en mente el concepto de *versioned release*: todo elemento del proyecto pasa por todas las fases del mismo y va evolucionando paralelamente hasta que se cierra en el momento de validar el hito correspondiente. Mientras, ese elemento pasa por varias versiones que evolucionan y crecen hasta completar todos los requerimientos asociados. De esta forma, se puede acotar más fácilmente el estado de cada entregable y prever las posibles desviaciones que pueda sufrir.

La aplicación de MSF requiere la implicación de todo el equipo y especialmente, del promotor del proyecto en la definición del alcance de cada uno de sus elementos. Así, MSF es totalmente adaptable a la realidad del proyecto y su ejecución variará completamente para cada entorno, por lo que todo equipo debe ser capaz de acordar los límites que deben permitir evaluar la ejecución del proyecto.

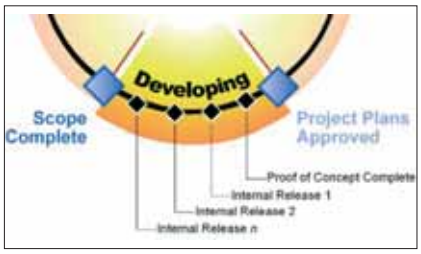
Detalle de las fases del proyecto

Las cinco fases de MSF presentan una guía general que debe adaptarse al proyecto, pero indican un conjunto de pasos necesarios que deben realizarse para garantizar el cumplimiento y la calidad del resultado final. Cada fase consta de una serie de pasos intermedios y un cierre de la misma; estas fases se describen a continuación.

Envisioning: se orienta a especificar los requerimientos de la solución propuesta, contando con la intervención de todos los implicados, tanto por parte del equipo del proyecto como del cliente. En esta fase se establece la visión global de la solución a implementar, pasando a acotar un alcance definido para el proyecto que permita establecer los recursos necesarios para satisfacer las expectativas. MSF ya contempla que este alcance varíe durante la vida del proyecto y especifica la forma de afrontar ágilmente estos cambios.

Planning: esta fase incluye el diseño y la planificación completa de la realización del proyecto. Por un lado, se realizan las labores habituales de diseño conceptual, lógico y físico del sistema, donde MSF no entra en detalle, permitiendo el uso de métodos ampliamente aceptados como es el caso de UML. Además, se contempla el diseño de elementos que integran la solución, como las infraestructuras, las comunicaciones, la seguridad, la puesta en operación o la formación. Esta fase finaliza con la apro-

bación del elemento clave: el Plan de Proyecto, que contempla las tareas a realizar durante las siguientes fases, la responsabilidad asociada a cada una y el momento de ejecución previsto.



Developing: en la fase de desarrollo del proyecto es donde MSF utiliza un enfoque más innovador sobre la forma de afrontar la implementación de los elementos de la solución. MSF utiliza el concepto de *release* para guiar esta fase y la siguiente a la consecución de los objetivos determinados en el proyecto. Esta orientación implica iniciar lo antes posible todas las tareas necesarias e intentar cerrarlas lo más tarde posible, de forma que todos los entregables avancen en paralelo y que la modificación de uno de ellos no implique cambiar algo que a priori ya está cerrado. De esta forma, se establece un método de trabajo basado en la elaboración de versiones parciales de los elementos del proyecto que pueden ser validados conjuntamente, formando lo que conoce como *internal release*. Estas versiones incluyen el código, las infraestructuras, la documentación y los materiales auxiliares. Cada una de estas versiones pasa internamente por el desarrollo de funcionalidades, su revisión de errores y las pruebas de integración hasta cerrar el cumplimiento de esta versión, como se puede observar en la figura 2.

Desde la primera versión, conocida como prueba de concepto (*proof of concept*), que sirve para validar la viabilidad del diseño realizado, se avanza en versiones que implican el desarrollo de nuevas funcionalidades, su validación y la integración entre sí para completar un nuevo hito del proyecto, que permite controlar su avance según lo establecido y el correcto funcionamiento de la solución. Esta fase finaliza con la implementación completa de las funcionalidades definidas en el alcance del proyecto.

Stabilizing: Una vez se han completado todas las funcionalidades necesarias para el proyecto, se pasa a una fase de validación del conjunto de la solución centrada en la corrección de errores y la optimización del resultado final. MSF establece dos puntos importantes en el control del resultado: la convergencia de errores (*bug convergence*), que es el momento en que el número de errores reportados es menor que el número de errores que se consiguen resolver; y el umbral sin errores (*zero bug bounce*), que sucede cuando el equipo consigue no tener ningún error abierto, independientemente que aparezcan posteriormente. Esto permite ir avanzando en la preparación de sucesivas *release candidates*, versiones prácticamente finalizadas de la solución. Finalmente, se realiza una prueba piloto en entorno real para verificar la puesta en producción de la solución implementada y cerrar definitivamente todos los elementos del proyecto.

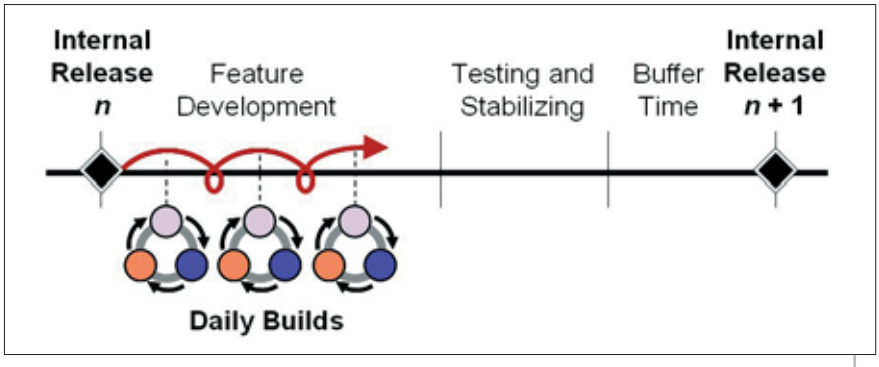
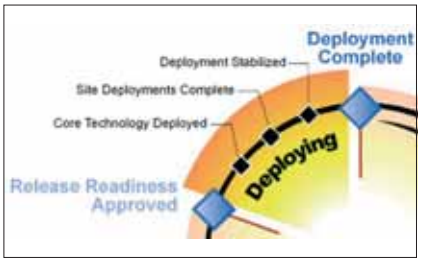
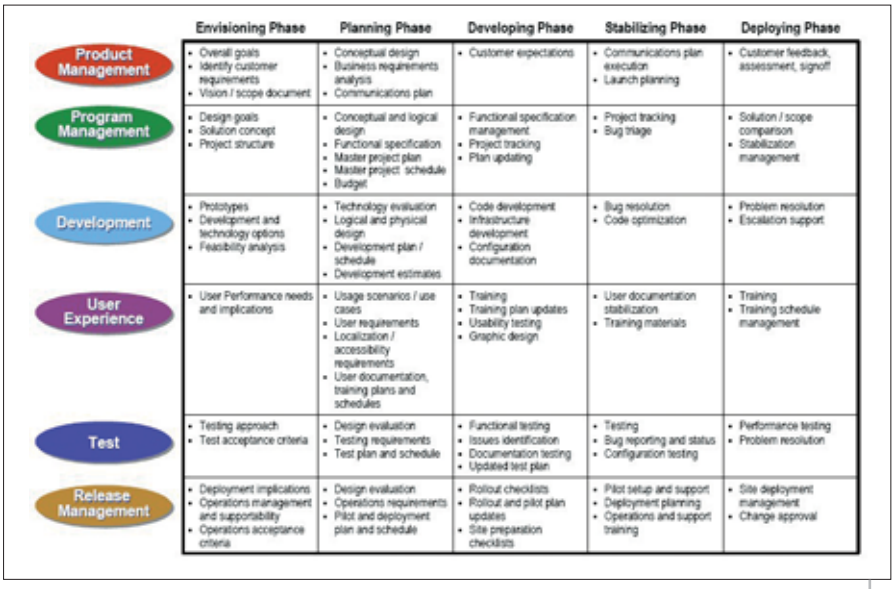


Figura 2. Ciclo de versiones en la fase de *Developing*

En la fase de desarrollo del proyecto es donde MSF utiliza un enfoque más innovador sobre la forma de afrontar la implementación de los elementos de la solución



Deploying: La última fase se centra totalmente en el paso a producción del proyecto realizado, teniendo en cuenta las necesidades específicas de adaptación al entorno de explotación existente y su despliegue a todos los usuarios del sistema. Implica la definición de las fases de instalación de clientes



	Envisioning Phase	Planning Phase	Developing Phase	Stabilizing Phase	Deploying Phase
Product Management	• Overall goals • Identify customer requirements • Vision / scope document	• Conceptual design • Business requirements analysis • Communications plan	• Customer expectations	• Communications plan execution • Launch planning	• Customer feedback, assessment, signoff
Program Management	• Design goals • Solution concept • Project structure	• Conceptual and logical design • Functional specification • Master project plan • Master project schedule • Budget	• Functional specification management • Project tracking • Plan updating	• Project tracking • Bug triage	• Solution / scope comparison • Stabilization management
Development	• Prototypes • Development and technology options • Feasibility analysis	• Technology evaluation • Logical and physical design • Development plan / schedule • Development estimates	• Code development • Infrastructure development • Configuration documentation	• Bug resolution • Code optimization	• Problem resolution • Escalation support
User Experience	• User Performance needs and implications	• Usage scenarios / use cases • User requirements • Localization / accessibility requirements • User documentation, training plans and schedules	• Training • Training plan updates • Usability testing • Graphic design	• User documentation stabilization • Training materials	• Training • Training schedule management
Test	• Testing approach • Test acceptance criteria	• Design evaluation • Testing requirements • Test plan and schedule	• Functional testing • Issues identification • Documentation testing • Updated test plan	• Testing • Bug reporting and status • Configuration testing	• Performance testing • Problem resolution
Release Management	• Deployment implications • Operations management and supportability • Operations acceptance criteria	• Design evaluation • Operations requirements • Pilot and deployment plan and schedule	• Rollout checklists • Rollout and pilot plan updates • Site preparation checklists	• Pilot setup and support • Deployment planning • Operations and support training	• Site deployment management • Change approval

Figura 3. Cuadro de relación entre los roles del *Team Model* y las fases del *Process Model*

Dar el salto a MSF v4

Estos artículos se basan en la versión v3 de MSF, que nos permite disponer de un punto de vista general y compartido de esta metodología. La cuarta versión de MSF apareció inicialmente junto con Visual Studio 2005 y con mucha vinculación a la *suite* de trabajo en equipos de Microsoft conocida como Team System, que aporta un sólido conjunto de útiles herramientas que permiten llevar la metodología del terreno teórico al práctico. La metodología sigue estando disponible en el sitio Web de Microsoft, sin que exista necesidad de adquirir este producto, de forma que será el siguiente paso para los lectores interesados en el tema.

La principal novedad de MSF v4 es que Microsoft ha intentado adaptar esta metodología a las dos tendencias más consolidadas en el mercado en cuanto a la gestión de proyectos: por un lado, los desarrollos ágiles, orientados a equipos pequeños y donde es más importante el seguimiento que la formalidad de la metodología; y por otro, las gestiones basadas en la formalidad y la calidad del proyecto, conocidas como modelos de madurez. Así, tenemos dos versiones de MSF v4. La más popular es sin duda **MSF for Agile Development**, que simplifica el modelo de la v3 para hacer más espiral todavía el modelo y que se centra en tareas identificadas que se asignan y resuelven de forma autónoma, con poca intervención de un jefe de proyectos y buscando el dinamismo y la colaboración del equipo. En cambio, **MSF for CMMI Development** se centra en reforzar los niveles de calidad y los estándares de documentación según el conocido modelo de madurez CMMI, siendo mucho más estricto y definido en cuanto a los flujos de información y control de calidad del resultado.

Con la base de términos y filosofía de aplicación de MSF presentada en esta serie de artículos, os recomiendo acercaros a estas dos variantes y probar su uso con Team System para ver que la metodología en entornos Microsoft ha dejado de ser algo teórico para convertirse en un auténtico pilar de toda empresa que realice proyectos de calidad y bien gestionados.

y servidores, los métodos de despliegue y la formación de los administradores y los usuarios finales. Esta fase contempla también la evaluación final del proyecto y su cierre, con la previsión de una nueva versión del mismo en mente.

Relación entre el Modelo de Equipos y el Modelo de Procesos

Después de conocer los dos modelos básicos que componen MSF, puede parecer a primera vista que se trata de visiones independientes del proyecto y que la relación entre sí es puramente superficial. Nada más lejos de la realidad, dado que MSF describe ampliamente la vinculación entre cada una de las fases del proyecto y las tareas a realizar por parte de cada uno de los roles existentes. Es imprescindible que cada miembro del equipo asuma las tareas a realizar en las distintas fases del proyecto, dado que todos los roles intervienen durante todo el desarrollo con mayor o menor dedicación.

La relación entre los roles y las fases de MSF se describe en el cuadro adjunto en la figura 3, detallando las responsabilidades de cada rol durante la vida del proyecto. Evidentemente, esta amplia lista de responsabilidades debe ser adaptada a cada proyecto, pero es necesario tener en cuenta cómo se llevará a cabo para no olvidar ningún elemento que pueda afectar a la calidad del resultado.

Conclusión

En el artículo del próximo mes se completará este acercamiento a las metodologías de Microsoft con la presentación de MOF, su método de gestión de la explotación de entornos informáticos. Mientras, los interesados en ampliar este artículo y conocer con más profundidad esta metodología pueden consultar la información disponible en el sitio Web de Microsoft, `http://www.microsoft.com/msf`.

Cuando tomamos decisiones, necesitamos información; esa información nos ayuda a gestionar mejor los procesos de nuestras organizaciones y a ser más competitivos, dándonos mejores posibilidades de supervivencia en el corto, medio y largo plazo. Aproximadamente el noventa por ciento de la información que necesitamos está dentro de nuestra propia organización, pero por desgracia la dispersión de la información (múltiples fuentes de datos, múltiples formatos, calidad del dato mostrado, etc.) hace que solo el treinta por ciento esté accesible en el formato y en el tiempo adecuado; el resto son datos por tratar o información no estructurada...

si no lo tienes claro...

Guillermo «Guille» Som

Configuración (Settings)

El configurador que lo configure...

Cualquier aplicación que se precie debe tener en cuenta las preferencias de los usuarios, pero incluso si la aplicación no es "amigable" con el usuario que la utiliza, usará con total seguridad datos de configuración. En este artículo veremos cómo podemos gestionar esos datos de configuración utilizando las facilidades que nos proporciona Visual Studio 2005.

Guillermo "Guille" Som es Microsoft MVP de Visual Basic desde 1997. Es redactor de **dotNetManía**, mentor de Solid Quality Iberoamericana, tutor de campusMVP, miembro de Ineta Speakers Bureau Latin America, y autor de los libros "Manual Imprescindible de Visual Basic .NET" y "Visual Basic 2005".
http://www.elguille.info

Configuración de aplicaciones en Visual Studio 2005

En Visual Studio 2005 se introdujo en las propiedades del proyecto un nuevo apartado, que es precisamente sobre lo que trata este artículo: la ficha "Settings" ("Configuración" en la versión en castellano). Por medio de esa ficha podemos definir los datos de configuración que usará nuestra aplicación. Esos datos de configuración pueden tener dos niveles, según queramos que sean a nivel de la aplicación (válida para todos los usuarios) o que esos datos sean particulares a cada usuario. En el primer caso, los datos serán de solo lectura; es decir, no podremos modificarlos en tiempo de ejecución, sino que solo podremos leer los valores asignados. Por otra parte, los datos de configuración a nivel de usuario son valores de lectura/escritura, y por tanto los podremos modificar en tiempo de ejecución, que es lo deseable, ya que así podremos guardar las preferencias de cada usuario, como por ejemplo el tamaño y posición de la ventana, algunos de los valores con los que normalmente trabajará, etc.

La razón de que los valores de configuración a nivel de aplicación sean de solo lectura es para que el usuario no pueda cambiar un valor que en realidad es "global" al resto de usuarios; por tanto, si necesitamos valores "personalizables" por cada usuario, usaremos el ámbito de usuario. Pero si necesitamos valores modificables a nivel de toda la aplicación, tendremos que hacerlo manualmente; Visual Studio no nos proporciona ninguna forma automatizada de que los valores de configuración con ámbito de aplicación se puedan modificar en tiempo de ejecución, y por tanto tendremos que usar nuestra propia forma de acceder y modificar esos valores, ya sea escribiendo directamente en el propio fichero de configuración de la aplicación o, mejor aún, creando nuestro propio fichero de configuración. En el código que acompaña a este artículo incluyo una clase (`ConfigXml`) que nos puede servir para realizar esas configuraciones personalizadas.

Tipos de datos de los valores de configuración

Los tipos de datos que podemos usar en la configuración son muy variados, e incluyen los tipos "normales" de .NET, y según estemos usando un lenguaje de programación u otro, esos tipos de datos se mostrarán según la definición de los mismos en el lenguaje; por ejemplo, en C# se nos mostrará `string`, `bool` o `int`, mientras que en Visual Basic será `String`, `Boolean` o `Integer`. También se incluyen tipos más complejos, como `StringCollection` o `Font`, aunque en realidad podemos usar prácticamente cualquier tipo de datos definido en .NET Framework e incluso en nuestro propio proyecto; en un momento veremos cómo.

Añadir valores de configuración a nuestras aplicaciones

Veamos cómo podemos añadir valores de configuración (y usarlos posteriormente) en nuestras aplicaciones.

NOTA

La forma de configurar (crear o modificar en tiempo de diseño) esos valores de configuración en Visual Basic es prácticamente la misma, solamente cambia el nombre de los tipos de datos. Sin embargo, la forma de acceder a los valores de configuración desde programas es diferente en los dos lenguajes; muy parecida, la verdad, pero en Visual Basic el acceso se simplifica gracias al objeto My.Settings. Pero como en este artículo (como casi todos los de la sección dnm.inicio) utilizo C#, pues... la forma de usar los valores de configuración en Visual Basic habrá que verla en el código que acompaña a este artículo.
En cualquier caso, y con permiso de los "puristas", en el código de ejemplo muestro cómo crear una clase estática en C# para "simular" el objeto My.Settings de Visual Basic.

Figura 1. La ficha de configuración

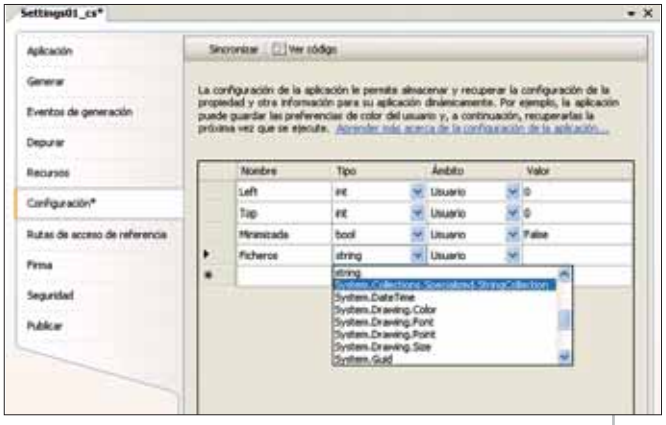


Figura 2. Tipos de datos en la configuración

Crear valores de configuración

Para crear valores que usaremos para almacenar datos de configuración debemos pulsar en las propiedades del proyecto ("My Project" en Visual Basic, "Properties" en C#) y seleccionar la ficha "Settings". Se nos mostrará algo como lo que aparece en la figura 1.

Viendo la figura 1 y dejándonos llevar por la intuición, podemos deducir que en "Nombre" pondremos cómo queremos que se llame ese valor de configuración, en "Tipo" el tipo de datos que tendrá (inicialmente siempre se muestra `string`), en "Ámbito" pondremos si queremos que sea una valor a nivel de "Usuario" o de "Aplicación" y en "Valor" el valor inicial que queremos que tenga. Ese valor debe ser adecuado para el tipo de datos; por ejemplo, en la figura 2 tenemos varios valores de configuración añadidos, entre los que hay uno de tipo `bool`, y en ese caso, las opciones que nos muestra son `True` y `False` (con la primera letra en mayúsculas, ya que no son tipos de C# sino valores que se guardarán como cadenas y después se convertirán adecuadamente).

Si necesitamos datos más complejos, por ejemplo una colección de datos, también podemos indicarlo. Tal como vemos en la figura 2, el valor de configuración "Ficheros" será del tipo `StringCollection`. Y en este caso, si queremos asignarle un valor inicial, veremos que el propio Visual Studio nos muestra una ventana en la que podemos indicar cada uno de los elementos que la colección tendrá (ver la figura 3).

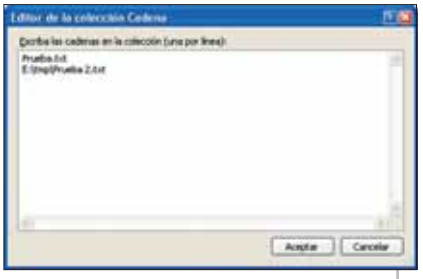


Figura 3. Editor de colecciones

Y si lo que necesitamos es algunos de los tipos de datos que no se han mostrado en la lista desplegable con los tipos a elegir, podemos seleccionar la última opción ("Examinar"); eso hará que se muestre una ventana con los tipos de datos que podemos seleccionar, tal como vemos en la figura 4. Y si el tipo que queremos usar no está, lo podemos indicar, como es el caso de la clase `Colegas` que tenemos en nuestro proyecto.

En este último caso, debemos tener en cuenta que al usar ese tipo que tenemos definido en nuestro proyecto, se agrega una referencia a nuestra aplicación; la solución es eliminar esa referencia y asunto arreglado.

Pero no vamos a complicarnos demasiado, ya que en ese caso no veremos las cosas importantes de todo esto de la configuración, que es lo que en realidad

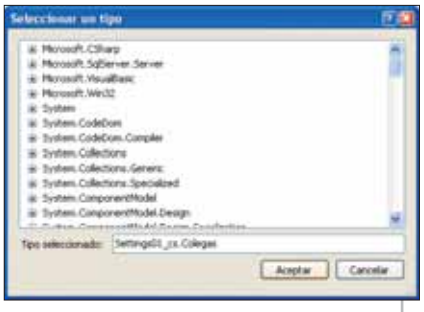


Figura 4. Seleccionar otros tipos de datos

importa. Así que veamos ahora cómo usar los datos de configuración.

Acceder a los datos de configuración

Una vez que hemos definido los valores que queremos usar en nuestra aplicación, ahora toca poder utilizarlos. En nuestro ejemplo vamos a usar los valores de configuración que podemos ver en la figura 2. Lo bueno del sistema de configuración generado por Visual Studio 2005 es que esos valores los usaremos como propiedades de una clase "especial" que se añade a nuestro proyecto. Esa clase se llama `Settings` y en C# está definida en un espacio de nombres llamado `Properties`, que su vez está definido dentro del espacio de nombres de nuestra aplicación. Y debido a que es una clase donde están las propiedades de los valores de configuración, podríamos pensar que tendríamos que crear una instancia de esa clase para poder accederlas; pero esto no es necesario, ya que el propio compilador de C# define una propiedad estática (compartida) en esa misma clase que nos permite acceder a esas propiedades que hacen referencia a los valores de configuración. Esa propiedad se llama `Default` e internamente accede a un campo que crea una instancia de la propia clase. Sabiendo esto, la forma de acceder a esos valores será el siguiente:

```
this.Left =
    Properties.Settings.Default.Left;
```

Al definirse como propiedades de una clase, la forma de usarlas es bastante intuitiva. Y como podemos comprobar, los tipos de datos de esas propiedades son los mismos que hemos definido en la configuración; en este ejemplo el tipo "interno" de la propiedad `Left` es `int`, por lo que no tenemos que hacer ningún tipo de conversión a la hora de asignar el valor.

Si los valores a los que hacen referencia esas propiedades han cambiado y queremos "persistirlas", lo primero que tenemos que hacer es asignar el valor, por ejemplo:

```
Properties.Settings.Default.Left =
      this.Left;
```

> Los valores de configuración se usan como propiedades de una clase "especial" que se añade a nuestro proyecto

Y para que esos datos se guarden tenemos que llamar al método `Save`:

```
Properties.Settings.Default.Save();
```

Es importante llamar al método `Save` para asegurarnos de que los datos se guardan. Y debemos llamarlo siempre que creamos conveniente, aunque lo recomendable es hacer esa llamada cuando el formulario se vaya a cerrar, es decir, en el evento `FormClosing`.

Sin querer confundir las cosas, me gustaría hacer una aclaración para los lectores que prefieren usar VB como lenguaje de programación, y es que en él de forma predeterminada no es necesario llamar explícitamente al método `Save`, ya que el compilador se encarga de añadir la llamada a ese método cuando el formulario se cierra. Al menos si así lo hemos indicado en las propiedades del proyecto, marcando la opción "Guardar My.Settings" al cerrar. Precisamente porque `My.Settings` es otra funcionalidad que ofrece Visual Basic para gestionar todo el tema de las configuraciones, y en realidad consiste en la definición de una propiedad llamada `Settings` que está definida en un módulo (y por tanto no es necesario indicar dónde está definida), y además este módulo está definido en el espacio de nombres `My`, por tanto, la forma de usar los valores de configuración en Visual Basic es la siguiente:

```
Me.Left = My.Settings.Left
```

En el fondo, esa propiedad `Settings` lo que hace es utilizar la propiedad "autoinstanciable" `Default`.

En C# podemos acceder fácilmente a esa propiedad `Default` sin necesidad de escribir tanto, definiendo una variable en el formulario que simplemente haga referencia a esa propiedad "predeterminada":

```
Properties.Settings misProp =
      Properties.Settings.Default;
```

De esta forma podremos acceder más fácilmente a las propiedades de configuración:

```
misProp.Top = this.Top;
```

Recibir notificaciones cuando cambien los valores de configuración

Otra funcionalidad que Visual Studio 2005 pone a nuestra disposición es la de saber cuándo se va a asignar un valor a una propiedad de configuración o cuándo se va a guardar o cuándo se han cargado los valores. Cuando queremos acceder a esos eventos de configuración, Visual Studio crea una clase parcial con instrucciones para usar dos de los cuatro eventos disponibles: `SettingChanging` y `SettingsSaving`.

El primero de los eventos anteriores se produce antes de que cambie uno de los valores de configuración, permitiendo cancelar la asignación. El segundo se produce antes de que se guarden los valores, y también podemos cancelarlo, aunque en este caso no tenemos acceso a los datos que se están guardando salvo que accedamos a las propiedades de la propia clase; sin embargo, con el evento que nos informa de los cambios de las propiedades, en la variable del segundo parámetro se pasa información sobre la propiedad afectada, el nuevo valor a asig-

nar, etc., lo que veremos a continuación con más detalle.

Los otros dos eventos son `PropertyChanged` y `SettingsLoaded`. En el primero se nos avisa cuando una propiedad ha cambiado, y la única información pasada en el segundo parámetro es el nombre de la propiedad que ha cambiado. El segundo evento se dispara después de la carga de los valores de configuración.

La forma de acceder a ese fichero de código es desde la ficha "Configuración" de las propiedades del proyecto y pulsando el botón "Ver código" que está en la parte superior de esa ficha, tal como podemos comprobar en la figura 1. Al pulsar ese botón, se abrirá un fichero con parte del código que normalmente usaremos; en particular, hay definidos dos de los métodos que interceptarán los eventos `SettingChanging` y `SettingsSaving`, y en el constructor de la clase están comentadas las instrucciones que ligarán esos métodos con los eventos. Si queremos interceptar los otros dos eventos, tendremos que escribir sus "manejadores" por nuestra cuenta; como ya vimos en el número 31 de **dotNetManía**, el propio editor de C# nos ayuda a la creación de los mismos.

Si nos decidimos a interceptar el evento `SettingChanging` para detectar los cambios antes de que se asignen a las propiedades, debemos tener en cuenta que los valores pasados en la variable del segundo parámetro no son tipos "explícitos"; es decir, si el cambio se produce en el valor `Left`, que como vimos es de tipo entero, en este evento se recibe dicho valor como `object`, porque ese mismo evento sirve para todos los valores de configuración, y como hemos comprobado podemos usar prácticamente cualquier tipo de datos.

Por tanto, si queremos detectar el cambio en esa propiedad, tendremos que hacer la conversión correspondiente, tal como vemos en el siguiente código:

```
if (e.SettingName == "Left")
{
    int i = (int)e.NewValue;
    if (i < 0)
    {
        e.Cancel = true;
    }
}
```

Como vemos, el nombre del valor de configuración lo averiguamos por medio de `SettingName`, y el nuevo valor que se va a asignar está en `NewValue`. Y si no queremos que ese nuevo valor se asigne, simplemente asignamos un valor verdadero a la propiedad `Cancel`.

> La forma de ligar cualquier propiedad a un valor de configuración es mediante la propiedad ApplicationSettings

Trabajar con colecciones genéricas personalizadas

Al definir los valores de configuración no podemos utilizar directamente tipos genéricos, pero nada impide que podamos definir un tipo de datos propio que esté derivado de un tipo genérico y lo utilicemos. Por ejemplo, en nuestro proyecto podemos tener creado un tipo de datos llamado `Colega`, y podemos querer almacenar instancias de esa clase en una colección genérica de tipo `List<Colega>`. Debido a que Visual Studio no nos permite usar ese tipo para un valor de configuración, lo que podemos hacer es definir nuestra clase/colección `Colegas` de esta forma:

```
public class Colegas : List<Colega>
{}
```

Esto simplemente hará que usemos la clase `Colegas` como una colección genérica, pudiendo añadir valores de esta forma:

```
Colegas cols = new Colegas();
Colega c = new Colega("Guille");
cols.Add(c);
cols.Add(new Colega("Pepe"));
```

Y si hemos definido en la configuración una propiedad que sea del tipo `Colegas` que hemos definido en nuestro proyecto, tal como vimos en la figura 4, podemos asignar valores de este tipo a la configuración de esta forma:

```
Properties.Settings.Default.Colegas =
    cols;
```

Y por supuesto, podemos acceder a ese valor de la forma habitual, además sin necesidad de hacer ningún tipo de conversión, ya que en las propiedades de la clase `Settings` siempre se almacenan los datos usando el tipo que hemos indicado. En el siguiente código obtenemos los valores de los "colegas" que tengamos en el fichero de configuración y los asignamos a un control `ListBox`:

```
Colegas cols =
 Properties.Settings.Default.Colegas;
this.listBox1.Items.AddRange(
                cols.ToArray());
```

Enlace de propiedades con valores de configuración

Algo muy habitual en el manejo de los valores de configuración es hacer *binding* de ciertas propiedades de los controles o del formulario a valores almacenados en la configuración de la aplicación. Tan habitual es, que es prácticamente casi todo lo que la documentación de Visual Studio muestra como ejemplos, y "casi" lo único que se podía hacer en las versiones anteriores de Visual Studio, aunque a diferencia de éstas, en Visual Studio 2005 esos valores pueden tener un ámbito de usuario en lugar de ser solo a nivel de aplicación, con lo cual se facilita su uso.

La forma de ligar cualquier propiedad de cualquier control (o del formulario) a un valor de configuración es mediante la ventana de propiedades, en particular de la propiedad "ApplicationSettings", que en el caso del formulario, mostrará los valores que vemos en la figura 5.

Figura 5. Indicar las propiedades enlazadas a la configuración

Como vemos en la figura 5, para el formulario hay dos propiedades preparadas para recibir el valor. Esas propiedades dependerán del control que estemos usando, pero si la propiedad que queremos "persistir" no está en esa lista de propiedades, podemos pulsar en "PropertyBinding" y desde el cuadro de diálogo que se nos mostrará podremos seleccionar la propiedad que queremos ligar con un valor de configuración. Si ya tenemos un valor de configuración del tipo adecuado, podemos usar esa propiedad o bien crear una nueva; en ese caso, como vemos en la figura 6, podemos crearla directamente desde ese mismo cuadro de diálogo.

Al crearla desde este cuadro de diálogo, podemos indicar el ámbito que queremos que tenga ese valor de configuración además de asignar los valores que queremos que tenga. Y tal como vemos en la figura 7, en la que asignamos el valor de la propiedad `ClientSize`, se mostrará el valor que actualmente tenga la propiedad que queremos usar.

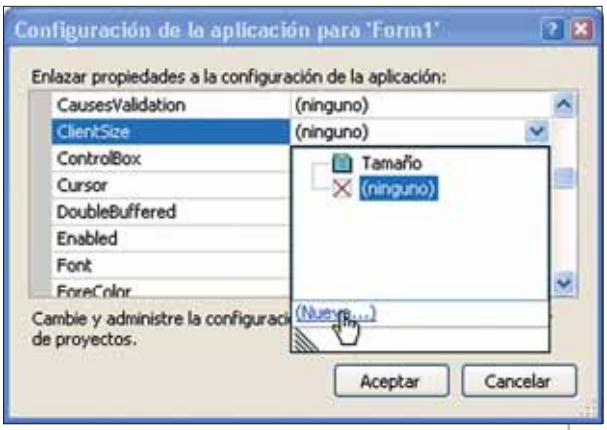


Figura 6. Enlace de propiedades

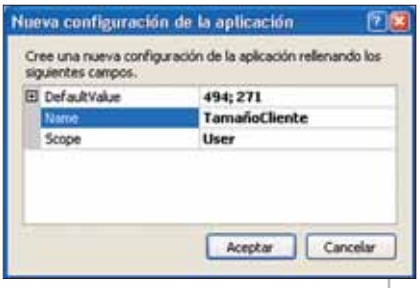


Figura 7. Crear un valor para una propiedad

Una vez que tenemos las propiedades enlazadas a valores de configuración, en la ventana de propiedades se nos mostrarán todas las que lo estén, tal como podemos comprobar en la figura 8.

Figura 8. En las propiedades se muestran las propiedades enlazadas

Y en cada una de esas propiedades enlazadas tendremos un icono que nos indicará esa situación "de enlace", además de indicársenos cuál es el valor de la configuración con el que la propiedad está enlazada, tal como vemos en la figura 9.

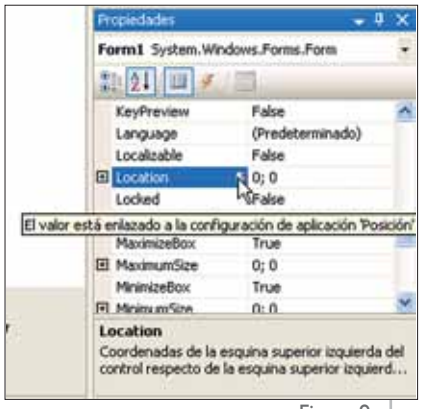


Figura 9. En las propiedades enlazadas se nos indica con qué valor lo está

Ni que decir tiene que es el propio compilador el que se encarga de implementar todos estos "enlaces", de forma que nosotros no tengamos que escribir código extra para que se hagan efectivos. Lo único que tendremos que escribir (si estamos usando C#) es la llamada al método `Save` de la clase `Settings` a la hora de guardar los datos de configuración, y como ya comenté antes, el mejor sitio en el que podemos llamar a ese método es en el evento `FormClosing`, para que se guarden los valores de configuración cuando el formulario se esté cerrando y no perdamos nada.

Como truco, decir que hay ciertas propiedades que tienen un comportamiento no esperado, como es el caso de la propiedad `Checked` de los controles `RadioButton`, ya que el comportamiento esperado de ese tipo de controles es que al seleccionar un control de un grupo, el resto se deseleccione, pero al estar la propiedad `Checked` enlazada, esa "magia" se pierde, y tendremos que ser nosotros mismos los que tengamos que asignar los valores adecuados a cada una de las opciones que estén en un mismo rango. Además de que para ese tipo de controles, al tener esa propiedad enlazada, el comportamiento en tiempo de ejecución no es... nada deseable. Por tanto, mi recomendación es la de no enlazar automáticamente las propiedades `Checked` de los controles `RadioButton` y si lo necesitamos, hacerlo manualmente, aunque, como es natural, usando los valores de configuración.

Configuraciones totalmente personalizadas

Esto de que el propio Visual Studio proporcione una forma automatizada de almacenar valores de configuración está muy bien. El problema principal es que los valores de configuración a nivel de aplicación son de solo lectura, es decir, no podemos almacenar nuevos valores. Por un lado es lógico ese comportamiento, ya que así un usuario no alterará los valores que todos los usuarios van a usar. Pero hay casos en los que nos puede interesar que sí se puedan modificar ciertos valores, por ejemplo, la localización de una base de datos o de un fichero con cierta información que todos los usuarios de la aplicación usarán.

En estos casos, podemos actuar de dos formas. Una es creando nuestro propio sistema de almacenar los datos de configuración; por ejemplo, yo suelo usar una clase llamada `ConfigXml` que utilizo en la mayoría de los casos en los que necesito que los valores de configuración estén compartidos con todos los usuarios (incluso la uso para que cada usuario tenga sus propios datos, pero eso más que nada es por costumbre de usar las cosas que me funcionan y sobre las que tengo más control).

La otra forma de modificar valores del propio fichero de configuración es accediendo directamente a dicho fichero, que suele tener el nombre de la aplicación con la extensión `.config` y que suele estar en el directorio del ejecutable (los cambios realizados en Windows Vista serán en el directorio `roaming`, pero para nuestro uso eso no afecta).

Como el fichero de configuración tiene formato XML, lo más fácil es usar una variable de tipo `XmlDocument` y sabiendo cómo se almacenan los datos, nos resultará fácil escribir un método que se encargue de hacer esas modificaciones por nosotros. En realidad, el acceso manual solo lo tenemos que hacer para almacenar los valores, ya que la lectura de esos valores es automática y se realiza al iniciarse la aplicación.

Veamos cómo podemos modificar esos valores. Lo primero es definir una variable a nivel del formulario, ya que la usaremos tanto en el evento `FormClosing` como desde un método que será el que se encargue de guardar los datos. Después definimos el método que se encarga de almacenar los datos en el fichero de configuración y finalmente, en el evento de cierre del formulario asignamos esos valores. Si no queremos tener que escribir código extra, esos valores los debemos tener "ligados" a las propiedades del formulario o de los controles que nos interese que reflejen los valores de la configuración. Una observación: si queremos "persistir" la posición del formulario, la propiedad `StartPosition` de éste debe tener el valor `Manual`. En el listado 1 tenemos todo el código necesario para hacer esto que acabamos de comentar. Para no alargar demasiado el artículo, en los comentarios del código se explica el truco para poder acceder correctamente a los valores de configuración.

```
private XmlDocument configXml = new XmlDocument();
// El método para guardar los valores
private void cfgSetValue(string clave, string valor)
{
    // La sección en realidad tendrá la forma:
    // configuration/applicationSettings/APLICACION.Properties.Settings/
    string secCS = this.GetType().Namespace + ".Properties.Settings";
    string seccion = "configuration/applicationSettings/" + secCS;
    XmlNode n = configXml.SelectSingleNode(
        seccion + "/setting[@name=\"" + clave + "\"]");
    if (n != null)
    {
        n["value"].InnerText = valor;
    }
}

private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    // Guardar los datos de configuración
    Properties.Settings.Default.Save();

    // El fichero de configuración
    string ficConfig = Application.ExecutablePath + ".config";
    // Cargamos el contenido
    configXml.Load(ficConfig);

    // Guardar los valores de configuración de la aplicación

    // Guardar el valor de la posición del formulario
    cfgSetValue("Posicion", this.Left + ", " + this.Top);

    // El nombre de la base de datos
    cfgSetValue("BaseDatos", this.textBox1.Text);

    configXml.Save(ficConfig);
}
```

Listado 1. Modificar los valores de configuración a nivel de aplicación

Conclusiones

En este artículo hemos visto cómo trabajar con los valores de configuración en nuestras aplicaciones creadas con Visual C# 2005, en las que aprovechamos la funcionalidad que nos da el entorno de desarrollo para automatizar todo el trabajo de asignación y recuperación de los valores de configuración. Y como no es plan de olvidarse de los usuarios que utilizan Visual Basic, en el ZIP con el código he incluido también proyectos creados con Visual Basic 2005 en los que se muestra cómo hacer todo lo comentado en el artículo, además de extras que por falta de espacio no he podido explicar, como es la posibilidad de usar la clase `ConfigXml` o cómo mostrar y modificar los valores de configuración al estilo de la ventana de propiedades del propio Visual Studio 2005.

Dino Esposito

todotNet.qa@dotnetmania.com

Web rica... pero Web...

En este número discutimos una antigua pero divertida característica de Google (detección automática de ubicación) e investigamos las formas de añadirla a nuestras aplicaciones personales. También discutiremos el estado de sesión y los envoltorios (wrappers) ASP.NET para el motor de Silverlight.

¿Ha visto alguna vez a Google en acción? No importa desde dónde uno se conecte, siempre parece saber dónde estamos. Y automáticamente muestra la página principal localizada. Si estamos en España, sale la página principal española; si estamos en Bulgaria, tenemos una página en alfabeto cirílico, y así sucesivamente. Me gustaría hacer lo mismo en el sitio Web de mi empresa. Si pudiera detectar la ubicación del usuario, podría redirigirle automáticamente a la página localizada adecuada.

¿Que si he visto a Google en acción? No podría vivir sin él. Creo que es una de las invenciones modernas que más significativamente ha afectado a nuestras vidas y a la forma en que hacemos las cosas. Así que sé exactamente lo que quieres decir con redirección automática a páginas localizadas. Si conectas a Google desde Bulgaria, se te envía a `www.google.bg`, donde toda la interfaz de usuario está escrita en caracteres cirílicos. Sin embargo, desde la misma página, puedes acceder al sitio principal (el de extensión `.com`, en inglés). Creo que es un excelente ejemplo de amigabilidad en la interfaz de usuario que muchos sitios Web deberían imitar. En una más que agradable obra de reciente publicación, **David Platt** —una de las leyendas vivas del software en Microsoft y reconocido autor de libros técnicos— indica precisamente este truco de Google como un camino a seguir. El libro en cuestión es "Why Software Sucks and What You Can Do About It" ("Por qué el software molesta y qué se puede hacer al respecto"), y está publicado por Addison-Wesley. Si tiene oportunidad, échelo un vistazo. No es muy técnico y ciertamente, no se trata de un libro de programación. Pero está lleno de humor y nos hace pensar y sonreír; a veces, amargamente.

En ese libro, David compara el sitio Web de Google con el de UPS. Ambos sitios son capaces de suministrar páginas localizadas basadas en la ubicación del usuario. Sin embargo, Google realiza una suposición sobre esa ubicación y lo hace correctamente. UPS, por su parte, espera que el usuario seleccione dónde se encuentra en una lista desplegable. Los patrones serían: "Déjame que imagine de dónde vienes" y "Dime de dónde vienes", respectivamente. No hay duda de que la segunda opción es mucho más sencilla de implementar, especialmente si los desarrolladores ni siquiera se molestan en guardar la localización seleccionada en una "cookie". ¿Cómo se haría la codificación de la primera opción, como hace Google?

La información clave que se necesita para deducir cuál es el país del usuario es la dirección IP asociada con la petición. En ASP.NET podemos acceder a esa dirección IP mediante la propiedad `UserHostAddress` del objeto `Request`, de esta forma:

Dino Esposito es mentor de **Solid Quality Learning**. Es ponente habitual en los eventos de la industria a nivel mundial. Visite su blog en: *http://weblogs.asp.net/despos*. (todotNet.QA@dotnetmania.com)

```
Response.Write(
        Request.UserHostAddress);
```

La dirección IP es el único punto de contacto del servidor Web con el usuario remoto que hace la llamada. No obstante, la dirección IP puede ser una rica mina de información si sabemos manejarla

La dirección IP es el único punto de contacto del servidor Web con el usuario remoto que hace la llamada. No obstante, la dirección IP puede ser una rica mina de información si sabemos manejarla. Una dirección Internet Protocol (IP) se presenta como una secuencia de cuatro números normalmente mostrados como `NNN.NNN.NNN.NNN`. Juntos, los cuatro números indican una ubicación geográfica codificada en tablas internacionales. Diseccionada de forma adecuada, la dirección IP puede revelar información similar a la que aportan los códigos telefónicos internacionales que todos conocemos.

Más concretamente, las direcciones IP son direcciones de 32 bits (un total de 4 bytes, o números entre 0 y 255) solo en la más comúnmente usada versión IPv4. IPv6 es el protocolo estándar emergente para Internet que Windows Vista, así como un gran número de implementaciones de Linux, soporta de forma nativa. Según IPv6, las direcciones IP son de 128 bits (16 bytes). Las direcciones IP se manejan y generan por la Internet Assigned Numbers Authority (IANA). IANA registra bloques de direcciones en Registros Regionales de Internet, que a su vez dan cabida a bloques más pequeños de proveedores de servicios y empresas.

¿Cómo mejorar su sitio mediante el soporte localizado geográfico para los usuarios? Es una cuestión de cuánto desea pagar por ello. Para una búsqueda rápida, puede venir a costarle entre 0 y 50$ por año. Si solamente se necesita reconocer usuarios de un pequeño conjunto de países, puede ir a sitios Web tales como `http://www.ipaddress-location.org`, interrogar acerca del rango de direcciones IP de un país dado, guardar la información en una base de datos propia, y consultarla después mediante código. Por supuesto, esto requiere trabajo, ya que hay que transformar los datos planos en algo manejable de forma rápida y efectiva en tiempo de ejecución. Pero es gratis. No lo he comprobado en profundidad, pero parece que no existe ninguna regla matemática intuitiva que simplifique la tarea. La otra posibilidad es comprar una base de datos ya existente y actualizada cada año que cuesta alrededor de 50$. Una se encuentra en `http://www.ip2location.com/free.asp`, desde donde se puede descargar una versión gratuita para probar y empezar a jugar con ella.

Soy un consultor de .NET y la mayor parte del tiempo trabajo en desarrollo de aplicaciones ASP.NET 2.0. Como tengo pasión por .NET, leo a menudo sus artículos. En uno de ellos, acerca de la gestión de estado en ASP.NET, se indica que el estado de la sesión no es un buen sitio para colocar DataSets grandes hasta que sean llamados por las páginas. De ahí mi pregunta: ¿Cuál sería un rango aceptable de tamaños para los datos de la sesión?

Como en las películas de acción, el policía bueno le dice al criminal: "Cualquier cosa que diga podrá ser utilizada en su contra". Así es como parece que sucede en este caso: "Cualquier cosa que escriba puede ser usada en contra mía". Y hacerme escribir más y más…, ya que más y más preguntas podrían hacerse y responderse sin fin, en un círculo del que es difícil decir si es virtuoso o vicioso ☺. Pero volvamos al estado de sesión en ASP.NET.

El manejo del estado de la sesión es una tarea que puede ser circunscrita a 3 pasos: asignar un ID de sesión, obtener datos de la sesión de un proveedor y añadirlos al contexto de la página. El módulo de estado de sesión HTTP maneja la ejecución de todas estas tareas. Al hacerlo, se aprovecha de un par de componentes adicionales: el generador de ID de sesión y el proveedor de estado de sesión.

Los datos almacenados en el estado de sesión afectan de una forma u otra la memoria del servidor. De forma que cuanto más pequeño, mejor. Por otra parte, el estado de la sesión no es algo nativo de la Web; lo inventó alguien, y fue porque era extremadamente útil.

En ASP.NET, cuando usas el estado de sesión en modo *InProc* (en el proceso), cualquier objeto almacenado en el estado de sesión lo es como instancia activa de una clase. No tiene lugar ninguna serialización o de-serialización, lo que supone que puedes almacenar cualquier objeto que desees (incluyendo objetos COM) y acceder a él sin ninguna sobrecarga significativa.

> Considera que para un escenario InProc solo pagamos en términos de ocupación de memoria. En un escenario Out-of-Process, tenemos un costo adicional de serialización

La situación es menos favorable si optas por un proveedor de estado *Out-of-Process* (fuera del proceso). En esta arquitectura, los valores de la sesión son copiados del medio de almacenamiento original a la memoria del dominio de aplicación (`AppDomain`) que procesa la petición. Se necesita una capa de serialización/de-serialización para llevar a cabo la tarea, y esto representa uno de los mayores costos para estos proveedores. ¿Cómo se ve afectado tu código? Primero, tendrías que asegurarte de que solo los objetos serializables se almacenan en el diccionario de sesión; como puede imaginarse, el estado no podría almacenarse.

Para realizar la serialización/de-serialización de tipos, ASP.NET usa dos métodos, cada cual suministrando distintos resultados en términos de rendimiento. Para los tipos básicos, se usa un serializador interno optimizado; para los demás tipos, incluyendo objetos y clases definidas por el usuario, ASP.NET hace uso del formateador binario, que es más lento. Tipos básicos son `String`, `DateTime`, `Guid`, `IntPtr`, `TimeSpan`, `Boolean`, `byte`, `char` y todos los tipos numéricos.

Es difícil encontrar un número clave por debajo del cual estemos seguros del rendimiento. Es una cuestión muy específica de cada aplicación el indicar un valor concreto. Sin embargo, considera que para un escenario *InProc* (el predeterminado) solo pagamos en términos de ocupación de memoria. En un escenario *Out-of-Process*, tenemos un costo adicional de serialización que supone que para cada petición necesitamos un 15% extra de tiempo en caso de una petición al servidor de procesos o un 25% si usamos una base de datos.

Tengo un interés especial en Silverlight. Tengo entendido que, en este momento, hay que inicializar el entorno utilizando código Javascript. Eso me suena horrible. ¿Hay alguna otra forma?

Esperar o hacérselo uno mismo. En la versión 1.0 de Silverlight no hay ninguna facilidad para el programador a la hora de inicializar el entorno sin necesidad de escribir funciones Javascript y llamarlas manualmente. El problema aquí no es tanto el escribir el código —es nuestro trabajo, después de todo— sino el modelo, y —déjeme decirlo— el estilo, que es muy feo. Silverlight es en sí mismo un SDK para acometer ciertas tareas, y hace su trabajo correctamente. Las utilidades deberíamos encontrarlas en algún otro lado, quizás en el Cuadro de herramientas de ASP.NET. En lo que respecta a ASP.NET 3.5, no hay nada, que yo sepa. Veremos en una actualización futura. Mientras tanto, en ASP.NET Futures (`http://www.asp.net/downloads/futures/`) hay un par de controles de servidor habilitados para Silverlight (Media y XAML) que pueden usarse en páginas ASP.NET. No están todavía en producción y no han sido publicados oficialmente.

Si quieres entretenerte un rato, puedes crear tu propio envoltorio ASP.NET para inicializar el entorno Silverlight. Yo crearía una especie de control `SilverlightManager` y le haría generar el script de salida que se necesita. No es difícil, créeme.

Traducido al castellano por Marino Posadas

Un programa de TV online para desarrolladores

Hace 4 años, cuando decidí montar *www.vb-mundo.com*, no imaginé el creciente tráfico mensual de este portal y foro, mucho menos haber superado los 50.000 usuarios registrados, y desde luego no pensé en crear un programa propio y exclusivo de TV para desarrolladores.

VB-MUNDO TV constituye hoy (junto a MSDN TV) las 2 únicas alternativas televisivas disponibles 100% para desarrolladores. En Internet pueden encontrarse cientos de miles de vídeos de *webcasts*, o archivos de *podcasts*, pero no hay vídeos con formato televisivo (bloques, secciones y un conductor frente a las cámaras).

Los *webcasts* constituyen un formato rico en contenido y realmente valiosos, pero nosotros además sumamos la personalización de alguien mirando a cámara explicándonos de manera sencilla cosas complicadas, añadiendo noticias, trucos y creando interacción entre los usuarios y el programa (a partir del tercer capítulo se leen algunos *emails* de usuarios y a partir del cuarto se contestarán preguntas puntuales). Todo esto se realiza cuidando los detalles de la edición, iluminación y musicalización.

VB-MUNDO TV consta de 4 bloques: saludo inicial, comentarios generales sobre el programa actual, comentarios breves o agradecimientos, pasando rápidamente al "caballo de batalla" del programa: la sección "How To". Existen 2 secciones "How To" (código explicado) durante cada capítulo y una sección llamada "El Producto" donde se muestran detalles generales de diversos productos relacionados con los desarrolladores.

Ya se ha visto desde creación de informes en .NET gracias a los ReportViewers, pasando por los usos del HelpProvider y del ErrorProvider, hasta segmentos tan interesantes como *hyperthreading* con *background workers*. Ya para el cuarto capítulo mostraremos la creación y ejecución de un *workflow* mediante Windows Workflow Foundation.

Planeamos también envíos especiales como la cobertura que han hecho durante abril 2007 del evento de Microsoft Cono Sur para su TechNet & MSDN Day, que puede descargarse también desde la página del sitio: *www.vb-mundo.com/tv.asp*.

Pablo Tilotta

¿Quieres trabajar en Microsoft?

Durante octubre y noviembre se buscan candidatos en Europa para cubrir una serie de puestos en Microsoft.

Who is a qualified candidate?

1. Someone who has (or will be completing this year (before April 2008) a bachelor's (four year or more) degree in Computer Science or a related field.
2. Someone who has strong abilities and passion to write code in C/C++, C#.
3. Someone who is driven and passionate about technology, desiring to make high-quality software products that will sell all over the world.
4. Someone with fluent English communication skills.

Si cumples estos requisitos y te gustaría intentarlo, haznos llegar tu currículo a *trabajarenmicrosoft@netalia.es*.

¡De primero, ensalada mixta!

Con este título, empezamos la nueva temporada en **OnobaNET** —celebrando nuestro 2º Aniversario— con sesiones como:

- "Novedades en C# 3.0", con **Diego Aragón** y **Marcelo Villacorta**.
- "Introducción a LINQ", con **Alberto Barroso** y **Fran Díaz**.
- "Introducción a Windows Mobile", con **Paco Carbajosa**.
- "Alta disponibilidad en SQL Server 2005", con **Miguel Rodríguez**.

Cada presentación la dividimos en dos, una para cada ponente; éstas a su vez contenían la presentación y una demo donde se explicaba todo. Después de cada demo hubo preguntas y se sorteó algún que otro regalillo.

Para celebrar este evento, y haciendo mención al título, a la hora de presentar cada sesión, los ponentes cumplieron con la seguridad laboral y se enfundaron su correspondiente delantal.

En este evento —celebrado por primera vez en viernes— hubo ponentes nuevos. La gente se animó mucho y participó en todas las demos, con diferentes preguntas y mini-charlas dentro de algunas de las sesiones que dimos.

Todas nuestras presentaciones están colgadas en la sección eventos de nuestra web *www.onobanet.com/eventos/eventos.aspx*.

Fran Díaz

eventos.eventos.eventos.eventos.eventos.eventos

Octavio Hernández

IdeaBlade DevForce

Este mes dedicamos nuestra columna a una de las múltiples herramientas ORM (para el mapeado objeto-relacional) disponibles para .NET en el mercado, DevForce (de la empresa californiana IdeaBlade), que aunque tal vez sea un poco menos popular que algún que otro de sus competidores, no tiene nada que envidiarles, ofreciendo un amplio conjunto de posibilidades que permiten acelerar en gran medida el desarrollo de aplicaciones corporativas.

Ficha técnica

Nombre: DevForce
Versión: 3.5
Fabricante: IdeaBlade
Sitio Web: http://www.ideablade.com/
Categoría: Marcos de trabajo, herramientas ORM
Precio:
- Edición Express: de libre utilización y redistribución de librerías.
- Resto desde 2.495 USD/desarrollador*

* - descuentos especiales para múltiples licencias, instituciones académicas y ONG.

El estudio de LINQ to SQL, tecnología que como seguramente conoce el lector habitual [1] estará disponible como parte de .NET Framework 3.5 y Visual Studio 2008, y que se define a sí misma como un ORM ligero (o sea, como una pieza de software que ofrece, además del acceso a las consultas integradas, capacidades básicas para el mapeado objeto-relacional) ha despertado últimamente mi interés por estas potentes herramientas que, sin lugar a dudas, elevan la productividad del desarrollo y ayudan a saltar el agujero de impedancia entre el mundo de la programación orientada a objetos y el de las bases de datos relacionales. Si bien la disponibilidad inmediata dentro de la plataforma de recursos básicos para tales fines hará que herramientas como la que presentamos este mes, **IdeaBlade DevForce**, deban reestructurarse para aprovechar esos recursos, no pienso que las herramientas líderes de este segmento vayan a perder su atractivo ni mucho menos a desaparecer, por cuanto ofrecen toda una serie de facilidades que van bastante más allá de lo que nos proporcionará LINQ to SQL (al menos, en esta primera versión).

Principales características de DevForce

DevForce se distribuye en tres ediciones diferentes: Express, Professional y Enterprise. La versión Express es de uso y distribución gratuitos, y si bien, como es lógico, no ofrece todas las facilidades de sus hermanas mayores, sí suministra todas las capacidades básicas relacionadas con la gestión y persistencia de objetos y todas las herramientas visuales asociadas, por lo que se invita al lector a descargarla del sitio Web del fabricante [2] y probarla.

Entre los componentes principales de DevForce podemos mencionar los siguientes:

- **Mapeado objeto-relacional**, que simplifica el acceso a datos, ofreciendo una interfaz orientada a objetos de las bases de datos, que elimina la necesidad de escribir consultas SQL complejas.
- **Gestión de la persistencia**, a través de una capa de acceso a datos de alto rendimiento que controla la forma en que los objetos se recuperan, guardan y almacenan en caché.
- **Enlace a datos declarativo**, que simplifica el desarrollo de interfaces de usuario y sincroniza la interfaz con los objetos de negocio sin necesidad de escribir código complejo para ello, encargándose igualmente del formato y validación de datos y la gestión de errores.
- **Herramientas visuales adecuadas**, que se integran dentro de Visual Studio y simplifican en gran medida el trabajo del desarrollador.
- **Servidor de objetos de negocio** (solo en edición Enterprise), que permite el despliegue de aplicaciones en un entorno de *N* capas altamente escalable, garantizando la seguridad y la monitorización centralizada, entre otras ventajas.

Octavio Hernández es Development Advisor de **Plain Concepts**, editor técnico de **dotNetManía** y tutor de **campusMVP**. Es MVP de C# desde 2004, MCSD y MCT.

Por otra parte, como características fundamentales del producto podemos mencionar las siguientes:

- **Arquitectura en múltiples capas lógicas y físicas**. DevForce produce aplicaciones con una arquitectura en cinco capas lógicas, que a la vez simplifica la separación en capas físicas según la conveniencia. Estas capas lógicas son:
  a) *Capa de persistencia de datos*. El marco de trabajo de DevForce soporta todos los SGBD corporativos más populares del mercado: SQL Server, Oracle, Sybase, DB2 e Informix, además de cualquier otra que soporte OLEDB.
  b) *Capa de acceso a datos*. La persistencia objeto-relacional de DevForce suministra la capa de acceso a datos, gestionando el flujo y la sincronización de los datos entre las entidades en el modelo de negocio y sus destinos definitivos en el almacén de datos.
  c) *Capa de lógica de negocio*. El modelo de objetos de negocio de DevForce encapsula las reglas y procesos aplicables al dominio de la aplicación. DevForce crea clases de negocio que se mapean al almacén de datos subyacente; el desarrollador puede extender esas clases con lógica personalizada.
  d) *Capas de interfaz de usuario*. La arquitectura de enlace a datos de DevForce facilita el intercambio y sincronización de datos entre las entidades del modelo de negocio y los controles visuales.
- **Persistencia orientada a objetos**. El mecanismo de mapeado objeto-relacional de DevForce constituye una solución robusta y probada en la práctica que hace posible mover la información entre las tablas de la base de datos y los objetos de la capa de negocio con un mínimo esfuerzo por parte del desarrollador, y programar de una manera orientada a objetos.
- **Caché de alto rendimiento y ejecución en modo desconectado**. DevForce almacena los objetos en el cliente de manera transparente al programador. Cuando un objeto es solicitado, el marco de trabajo lo busca inicialmente en la caché, y luego, solo en caso necesario, en el servidor. Adicionalmente, es posible diseñar las aplicaciones de modo que puedan operar en modo desconectado. En tal caso, la memoria caché se utiliza como almacén temporal para las inserciones, actualizaciones y borrados; esta caché puede ser serializada a disco, y luego restaurada en el momento oportuno para enviar esas modificaciones al servidor. El soporte transaccional ayuda a resolver los posibles conflictos de concurrencia.
- **Marco extensible para el enlace visual a datos**. El enlace a datos de DevForce automatiza el enlace entre los controles visuales y las propiedades de los objetos de negocio, de modo que los desarrolladores puedan dedicarse a otras tareas. DevForce ofrece una API potente y extensible para el enlace a datos que añade mucha funcionalidad a la que está presente en .NET de manera predefinida. Adicionalmente, esta API integra el soporte para librerías de componentes de terceros como DevExpress e Infragistics.
- **Topología híbrida para aplicaciones Windows y Web.** Al desarrollar aplicaciones que deberán tener tanto interfaz Windows como Web, es incuestionable la conveniencia de que ambas "vistas" compartan los mismos objetos y lógica de negocio. La infraestructura de DevForce hace eso posible. Los usuarios ocasionales podrán utilizar la aplicación a través de Internet, mientras que los usuarios avanzados podrán aprovechar las ventajas de una aplicación Windows Forms que opera a través de Internet. Ambas clases de usuario compartirán una misma lógica de negocio, manteniendo la consistencia de los datos y reduciendo la cantidad de codificación y mantenimiento.

Un recorrido rápido

Primeros pasos

En esta sección daremos un vistazo rápido al proceso de desarrollo de aplicaciones basadas en DevForce a través de la creación de una aplicación sencilla. Para ello, comenzaremos creando una solución en blanco desde el Asistente de proyectos de Visual Studio 2005. A continuación, lanzamos la herramienta **IdeaBlade Object Mapper** (Mapeador de objetos), que la instalación de DevForce añade al menú de "Tools" (Herramientas) del entorno integrado. El Mapeador de objetos (figura 1) es el centro de control desde el que se llevan a cabo todas las tareas asociadas con la definición de las clases de negocio (entidades) y su mapeado contra las bases de datos relacionales en las que la aplicación se apoya.

Para empezar a interactuar con la base de datos, es necesario crear una fuente de datos de DevForce, indicando la cadena de conexión correspondiente. Como muestra la figura 1, también pueden utilizarse servicios Web como orí-

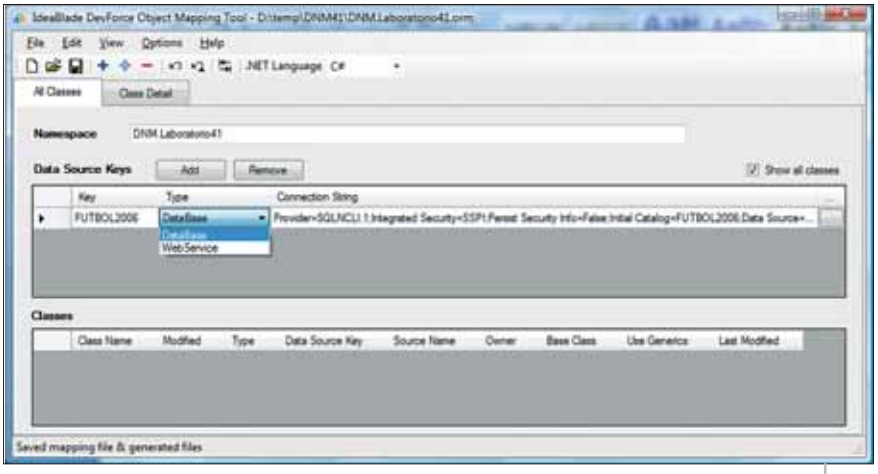


Figura 1

genes de datos. Para este ejemplo utilizaremos la base de datos `FUTBOL2006`, que almacena información sobre los equipos que tomaron parte en la Liga española de fútbol 2006-2007 y en la que se basa una buena parte de los ejemplos de mi próximo libro "C# 3.0 y LINQ" [3]. Tanto la base de datos como el código del ejemplo pueden descargarse del sitio Web de **dotNetManía**.

Siguiendo el nunca despreciable consejo de guardar pronto nuestro trabajo, seleccionamos la opción "File | Save as…" del Mapeador. Previamente deberemos establecer un espacio de nombres, que se utilizará como nombre para el proyecto que contendrá nuestras clases de negocio, a la vez que como espacio de nombre para ellas. El nombre de fichero que se nos propone para guardar la información que el mapeador utiliza es `DNM.Laboratorio41.orm`, y al guardar este fichero se generarán automáticamente dos proyectos de librerías de clases:

- `AppHelper`, proyecto colateral cuyo miembro más destacado es el fichero de configuración `IdeaBlade.ibconfig`, que cobra importancia a la hora del despliegue.
- `DNM.Laboratorio41`, que contendrá las definiciones de nuestras clases de negocio.

El mapeado y las clases de negocio

Para la creación de las clases de negocio, debemos seleccionar las tablas, vistas y procedimientos almacenados de la base de datos que deseamos incorporar a nuestro modelo. Podemos llevar a cabo esta labor pulsando el icono '+' de la barra de herramientas del Mapeador; para este ejemplo seleccionaremos las tablas `Futbolista`, `Club` y `Pais` (figura 2). Una vez agregadas las tres tablas, éstas se mostrarán en la lista "Classes" de la pestaña "All classes". Pero lo realmente importante es la labor de mapeado detallado que podemos llevar a cabo a continuación en la pestaña "Class Detail" sobre cada una de las tablas para establecer la manera en que se generarán las clases de negocio correspondientes. A continuación presentamos las principales posibilidades que ofrece DevForce en este sentido:

- En la pestaña "Class" se definen características generales como el nombre para la clase de entidad, su plural, la clase base de la que la clase debe heredar (caso de que se desee que algunas o todas las clases tengan un ancestro común), la columna a utilizar como identidad para las comprobaciones de concurrencia optimista o la cláusula `WHERE` de SQL a utilizar como filtro para las filas en el caso en que se trate de una tabla común para varias entidades que se discriminan por el valor de uno o más campos.
- La pestaña "Simple Properties" (figura 3) es en la que se lleva a cabo el grueso de la labor de mapeado. En ella podemos indicar, para cada columna de la tabla, cómo debe ser mapeada a la propiedad correspondiente de la clase de negocio: el nombre para la propiedad, tipo .NET a utilizar, si es de solo lectura o lectura/escritura, la posibilidad de asignación de valor nulo o el valor predeterminado, entre otras.
- Por su parte, en la pestaña "Relation Properties" se establecen las características de las propiedades que se añadirán a las clases de negocio para dar soporte a las relaciones entre entidades. Nuestro modelo relacional incluye dos relaciones *1:N* de las que la tabla `Futbolista` es destino. Del lado del "1" se utilizará para el nombre de la propiedad generada la variante en plural del nombre de la clase correspondiente.
- Por último, la pestaña "Enums" permite definir para una tabla un tipo enumerado en lugar de una clase de entidad; cada uno de los valores de la columna indicada de cada fila de la tabla se utilizará como valor del tipo enumerado. Esto es sumamente útil en el caso de tablas-nomencladores con un conjunto fijo de filas.

Al guardar de nuevo el fichero de mapeado, se generará automáticamente código para el proyecto de clases de negocio. La arquitectura de las clases de negocio que genera la herramienta es tal, que éstas siempre heredan de una clase base cuyo nombre es el mismo, pero con el aditivo "DataRow"; éstas últimas se apoyan a su vez en clases de la infraestructura de DevForce. De manera similar a como funcionan las clases parciales de .NET, el código generado automáticamente por la herramienta se concentra principalmente en las clases de más bajo nivel; y es en las clases herederas donde encontramos el

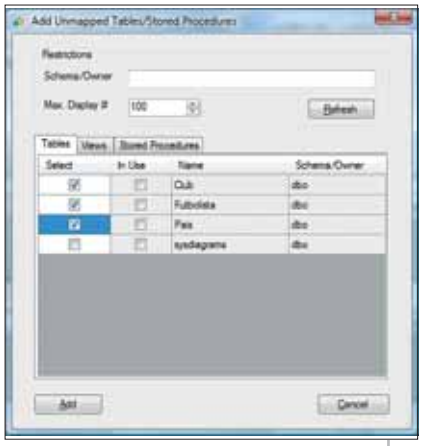


Figura 2

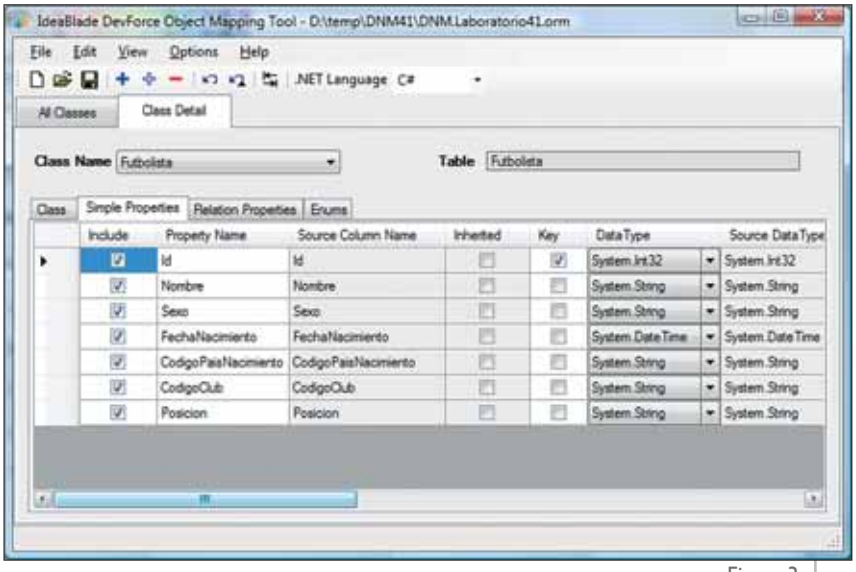


Figura 3

típico comentario "añada aquí su lógica de negocio" para que agreguemos la lógica específica de nuestras clases.

Generando la interfaz de usuario

DevForce no solo ofrece ayuda a la hora de generar los mapeados objeto-relacional y suministrar toda la infraestructura necesaria para la gestión transparente de esos datos; sino también a la hora de desarrollar la interfaz de usuario. Veamos cómo aprovechar esas facilidades, creando un nuevo proyecto de aplicación Windows Forms (que llamaremos `Demo`), al que agregaremos inmediatamente una referencia a la librería que contiene nuestras clases de negocio.

En este punto entra en juego un conjunto de componentes que DevForce instala en el Cuadro de herramientas de Visual Studio 2005 (figura 4). En particular, arrastraremos sobre el formulario del proyecto un objeto `ControlBindingManager`, encargado de gestionar los enlaces a datos entre los controles visuales del formulario y las colecciones de objetos de negocio gestionadas por DevForce. Entre las tareas que ofrece la etiqueta inteligente de este componente tenemos "Configure databindings", mediante la que podemos seleccionar la clase de negocio a presentar (figura 5, seleccionando la clase `Futbolista`) e indicar qué propiedades de esas clase (tanto simples, como de relación, como las que hayamos incorporado posteriormente de forma manual) queremos presentar en la interfaz, y mediante qué tipo de controles deseamos hacerlo (figura 6).

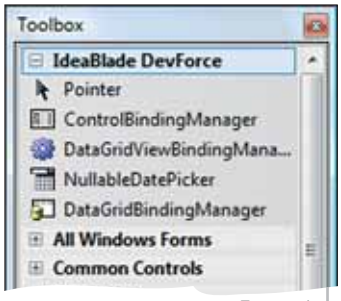


Figura 4

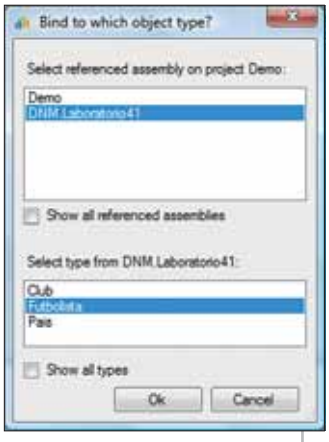


Figura 5

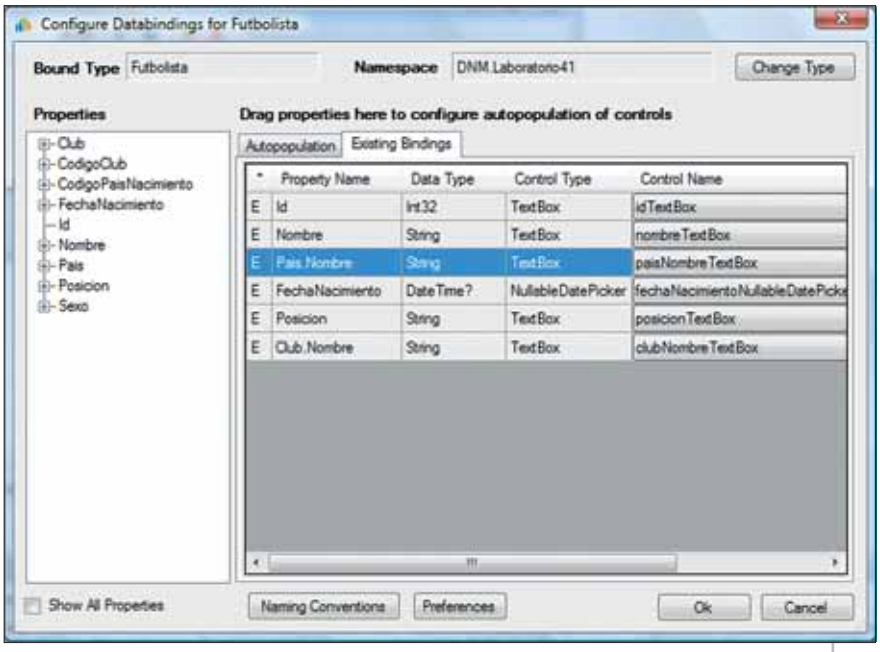


Figura 6

```
private void Form1_Load(object sender, EventArgs e)
{
  bindingSource1.DataSource =
              PersistenceManager.DefaultManager.GetEntities<Futbolista>();
}
```

Cuando se pulse el botón "Aceptar" de este diálogo, se generarán los controles indicados sobre la superficie del formulario. El resto del diseño de la interfaz es trabajo estándar con Windows Forms: arrastramos un `BindingSource` y conectamos a él el `ControlBindingManager`; un navegador tampoco viene mal para permitir al usuario desplazarse por los objetos. Solo nos restará entonces escribir un mínimo código basado en la potente librería de DevForce para garantizar la carga inicial de los objetos necesarios; dicho código se presenta en el listado 1.

Por supuesto, hay mucho más de lo que es posible presentar en el espacio del que disponemos. El sitio Web del fabricante [2] ofrece numerosos tutoriales y documentación al respecto.

Conclusiones

A lo largo de este artículo hemos intentado mostrar las principales posibilidades que ofrece IdeaBlade DevForce y cómo esta herramienta va más allá de ser un simple mapeador objeto-relacional para constituirse en un verdadero marco de trabajo para el desarrollo rápido y eficaz de aplicaciones corporativas con acceso a bases de datos de datos relacionales.

Referencias

[1] **Hernández O.**, "Lo que nos traerá Orcas: LINQ to SQL", en **dotNetManía** nº 36, abril de 2007

[2] Sitio Web de **IdeaBlade**: `http://www.ideablade.com`.

[3] **Hernández O.**, "C# 3.0 y LINQ", ISBN: 978-84-935489-1-9, Krasis Press (`http://www.krasispress.com`) de próxima aparición.

expoQA

4ª Jornadas Profesionales de Calidad y Testing de Software

26,27,28,29 y 30 de Noviembre en Madrid

www.expoqa.com

¡No ponga su negocio en juego!

¡No se la juegue!

Antes de lanzar su producto al mercado, asegúrese de su calidad.

Expo: QA se dirige a profesionales que necesitan mejorar procesos, cumplir plazos cada vez más cortos e incrementar la calidad de sus productos. Lo hace acercándolos a las tecnologías, servicios y tendencias actuales.

Cursos, Exposiciones y Presentaciones

Patrocinador Gold

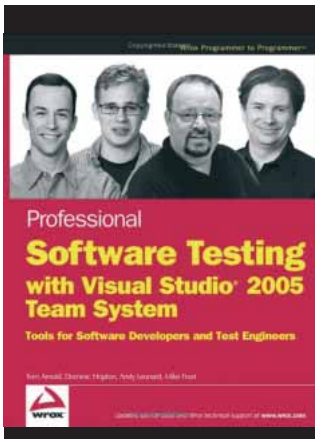
IBM

organizado por

SOGETI

Contactar:
contact@expoqa.com

Tel.:
+34 93 253 0188



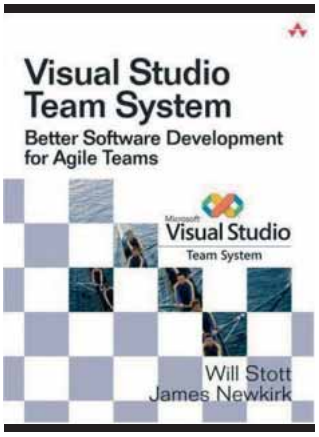
Professional Software Testing with Visual Studio 2005 Team System: Tools for Software Developers and Test Engineers

Tom Arnold, Dominic Hopton, Andy Leonard, Mike Frost

Editorial: Wrox
Páginas: 408
Publicado: septiembre 2007
ISBN: 978-0470149782
Idioma: inglés

Este libro es de los que muchos lectores, al verlo, pensamos: ¡por fin! Alguien se ha decidido a hacerlo. Es de esas obras que parecen necesarias para aprovechar en su integridad una tecnología y que, después de hojear su contenido, nos anima inmediatamente a mirarla con más profundidad.

La obra está teniendo una acogida espectacular, ya que cubre todas las fases del ciclo de vida de un proyecto de software, y muestra con claridad cómo implementar las buenas prácticas en la fase de pruebas, minimiza el tiempo necesario en esta fase del desarrollo, y en suma, ayuda a la mejor comprensión y utilización de VSTS como una herramienta integral de desarrollo. Los autores no son novatos, y en más de un caso han ayudado en el propio diseño de la herramienta. Ya conocíamos a **Andy Leonard**, por lo que no nos extraña el resultado.



Visual Studio Team System: Better Software Development for Agile Teams

Will Stott y James Newkirk

Editorial: Addison-Wesley Professional
Páginas: 864
Publicado: mayo 2007
ISBN: 978-0321418500
Idioma: inglés

Se trata de una obra de contenido similar al anterior, solo que abarca un espectro más amplio de la utilización de Visual Studio Team System (de ahí su longitud), que también está teniendo una excelente acogida en EE.UU. El enfoque de los autores se centra en enseñar a utilizar VSTS en entornos de programación ágiles reales, aprovechando el conjunto de herramientas integrado que posee VSTS. **Newkirk** es jefe de desarrollo en Microsoft, y en la actualidad, Product Unit Manager para CodePlex. **Will Stott**, consultor de estas técnicas, también en EE.UU. El recorrido del libro es, por tanto, eminentemente práctico, utilizando un caso de estudio del que se van desbrozando todos los problemas que aparecen en el desarrollo, y explicando las ofertas que la herramienta expone para su solución. Encontramos de especial valor la explicación de los mecanismos que permiten integrar las soluciones teóricas de desarrollo ágil en una herramienta como ésta.

novedades

MCITP Self-Paced Training Kit (Exam 70-441): Designing Database Solutions by Using SQL Server 2005

Dejan Sarka, Andy Leonard, Javier Loria y Adolfo Wiernik. Editorial: Microsoft Press. Páginas: 656. Publicado: septiembre 2007. ISBN: 978-0735623422.

Pro C# 2008 and the .NET 3.5 Framework

Andrew Troelsen. Editorial: APress. Páginas: 1000. Publicado: octubre 2007. ISBN: 978-1590598849.

TEXTO: MARINO POSADAS

Marino Posadas

Windows Live Translator, disponible

Aunque sus bondades de traducción son, como puede imaginarse, similares a las que podríamos encontrar en algunos otros servicios existentes por Internet, y se trata de una versión beta, lo cierto es que funciona bastante aceptablemente (deben revisar algunas construcciones en castellano, como las formas reflexivas) y es capaz de traducir buenas cantidades de texto. No está pensado para usarlo directamente, pero nos puede venir bien para comprender el sentido de ciertos textos que vemos en la Web. Dispone, además, de muchas opciones de traducción entre lenguas: ruso, árabe, inglés, chino, japonés, alemán, francés, coreano, portugués, holandés, y por supuesto castellano. Se puede probar en *http://translator.live.com*.

noticias.noticias.noticias

Windows 2008 RC0

Microsoft acaba de liberar la *release candidate* 0 de Windows Server 2008, que ya está disponible para su descarga en *http://www.microsoft.com/windowsserver2008/audsel.mspx*.

Por primera vez disponemos de una Community Technology Preview (CTP) de la tecnología de virtualización de Windows Server (*codename* Viridian). Windows Server 2008 incluye una capa pequeñísima (de menos de un megabyte) de virtualización de software basada en hipervisor que opera entre el hardware y el sistema operativo de Windows Server 2008.

Este RC0 es un paso importante en el camino hacia la entrega final de Windows Server 2008 y significa que el código del servidor está entrando en la etapa final de las pruebas, que se espera finalice en el primer trimestre de 2008.

La presentación oficial, junto a Visual Studio 2008 y SQL Server 2008, en diferentes estadios de desarrollo, se producirá a finales de febrero.

noticias.noticias.noticias

documentos en la red

Los amantes de los gráficos no pueden perderse este vídeo (disponible en *http://www.pcinpact.com/actu/news/38412-redimensionnement-image-photo-poids-photogra.htm*) sobre lo que los investigadores **Ariel Shamir** y **Shai Avidan**, del Efi Arazi School of Computer Science presentaron en la pasada edición de Siggraph (*Special Interest Group for Computer GRAPHics*). Se trata de una nueva técnica de redimensionamiento y modificación inteligente de las imágenes utilizando técnicas desarrolladas por ellos.

Vídeos sobre programación de API para Internet. Los chicos de **ProgrammableWeb** siguen publicando vídeos (cada vez tiene más éxito este formato para explicar procesos que requieren pasos discretos en una herramienta) sobre la integración de las API relacionadas con la Web. Ya tienen 19, incluyendo la programación de Yahoo, AOL y Skype, entre otras. Están disponibles en YouTube, pero también en *http://blog.programmableweb.com/2007/09/20/19-video-apis/funcionalidad*.

sitios del mes

Windows Scripting no es un *blog*, sino un subsitio con una ingente cantidad de material relacionado con la programación en general y los *scripts* en particular. Pertenece a **ASPFree**, pero contiene muy buenos artículos sobre una amplia gama de *programming affaires*, en sus propias palabras. El último recomendado, sobre WCF: *http://www.aspfree.com/c/a/Windows-Scripting/WCF-Essentials*.

DZone Snippets es una de esas fuentes donde podemos encontrar casi de todo. Pero relacionado con la programación y —concretamente— con el código fuente. Más de 8000 usuarios y casi 4000 rutinas de código para cubrir todas las necesidades (o muchas de ellas). Accesible en el sitio: *http://snippets.dzone.com*.

utilidades del mes

Freeware Files ofrece, entre muchas otras utilidades, una que es de especial interés para aquellos que tenemos que hacer demostraciones de código en público: **ZoomIt** permite ampliar cualquier parte del escritorio y arrastrarla a la zona que nos interese. Incluso permite dibujar sobre la zona ampliada (*http://www.freewarefiles.com/program_2_237_19697.html*).

Lina es una utilidad que permite, según sus autores, ejecutar cualquier programa hecho para Linux en Windows, Mac y Unix. Se encuentra en un sitio Web dedicado (*http://www.openlina.com/download.html*) y se trata de una delgada capa virtual que habilita al desarrollador para escribir y desarrollar código con las herramientas Linux ordinarias, pudiendo luego ejecutarlo en los sistemas antes citados.