Programación en ASP .NET

Beatriz Hernández Cruz
Reynaldo Hernández Hernández

Advertencia

Capitulo I

Introducción a ASP .NET

Fundamentos de la nueva arquitectura de programación

ASP .NET marca un antes y un después en la era de la programación por Internet. Microsoft ha dado un paso decidido y fundamental hacia la plena programación en entornos distribuidos, dejando atrás las limitaciones propias del uso de lenguajes script como VBScript.

Gracias a ASP .NET el programador puede usar todo el potencial que ofrecen lenguajes como Visual Basic .NET, Visual C .NET y Visual C# .NET. Ello se debe a la nueva arquitectura de programación que se ha establecido en el nuevo entorno de Visual Studio .NET.

La clave de la interoperatividad de los distintos lenguajes dentro de una misma aplicación radica en el nuevo motor de ejecución de lenguajes: Common Language Runtime (CLR). Se encuentra en el nivel inferior dentro de la arquitectura .NET.

El motor CLR se encarga de compilar el código antes de ejecutarlo, independientemente del lenguaje utilizado por el programador.

En vez de compilar a código binario (como es usual en cualquier lenguaje), CLR crea una representación a un lenguaje compartido dentro de la estructura .NET, el lenguaje Microsoft lntermediate Language (MSIL).

La primera vez que se ejecuta un código, el motor CLR invoca un compilador llamado Just In Time (JIT) que traduce el lenguaje MSIL en instrucciones propias al procesador del sistema que lo ejecuta, es decir, que la estructura .NET puede adaptarse y ejecutarse en distintos lenguajes y sistemas.

Cabe notar que la arquitectura .NET utiliza el mismo motor CLR para compilar cualquier tipo de código escrito en cualquiera de los lenguajes .NET, por ello, el rendimiento será el mismo, sea cual sea el lenguaje escogido a la hora de diseñar la aplicación.

Esquemáticamente, la estructura .NET es:

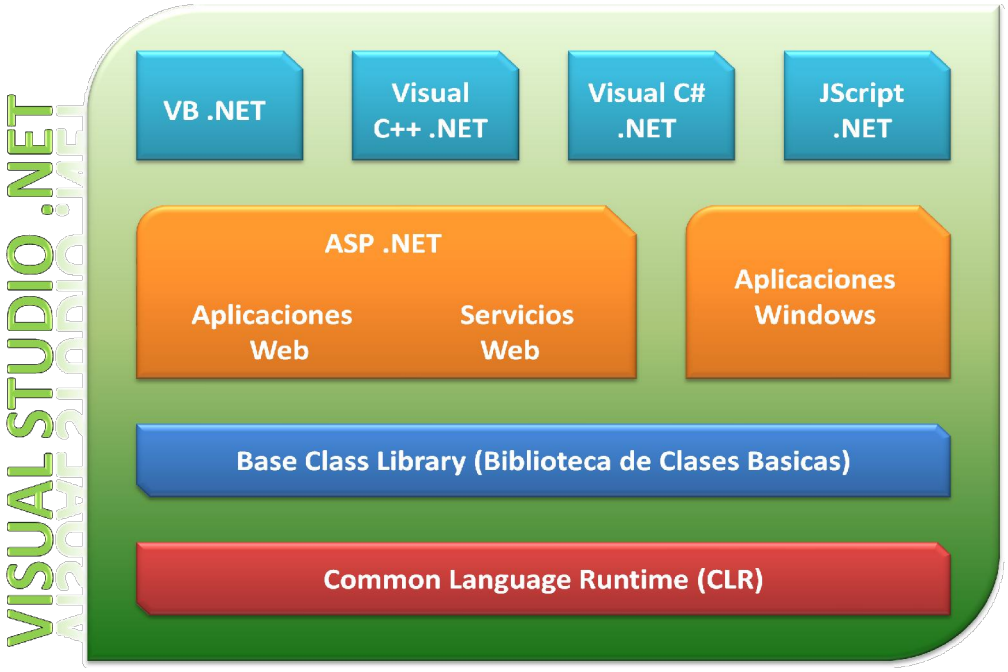

Figura 1. Estructura de Visual Studio .NET

La arquitectura .NET, se destaca por la completa compatibilidad entre los distintos lenguajes a la hora de programar aplicaciones o servicios ASP .NET. Dentro del motor CLR se ejecuta un sistema llamado Common Types System.

Base Class Library son unas bibliotecas de clases agrupadas por tipos, en función de las aplicaciones que tengan asignadas (seguridad, almacenamiento de datos, etc.). Cada uno de los distintos grupos de clases (Namespace) deriva de un grupo fundamental llamado System.

En un nivel superior es donde se diseñan las aplicaciones, que pueden ser de ASP .NET como las típicas de Microsoft (usando formularios para entornos locales de ejecución).

Dentro de la estructura ASP .NET, podemos ejecutar aplicaciones y/o servicios ASP .NET, aplicaciones de ejecución en red, tanto del lado del

servidor como del cliente usando para ello formularios Web y otras herramientas relacionadas con los servicios on-line.

Las aplicaciones ASP .NET se sirven de formularios Web para facilitar enormemente la tarea de diseño y creación. Únicamente con seleccionar y arrastrar encima del formulario Web un determinado control, Visual Studio .NET se encarga de crear el código HTML correspondiente. Una de las muchas ventajas que ofrece la estructura ASP .NET es que, automáticamente, se encarga de detectar el tipo de navegador utilizado por el cliente a la hora de realizar una petición a nuestro servidor y en consecuencia, determina la versión HTML que éste soporta. Por ello, el programador de aplicaciones ASP .NET no tiene que preocuparse por la compatibilidad con los navegadores, ya que ASP .NET se encargará de confeccionar la respuesta adecuada al tipo de navegador que realiza la consulta.

Los servicios Web son un tipo particular de aplicaciones ASP .NET pensadas para ser utilizadas dentro de otras aplicaciones ASP .NET. La idea es crear aplicaciones Web ASP .NET de acceso en red que sean accesibles a otras aplicaciones y de esta forma disminuir enormemente la cantidad de código necesario para realizar una aplicación. Por ejemplo, si queremos crear una aplicación ASP .NET encargada de realizar facturas a partir de los datos suministrados por un cliente y sabemos que existe un servicio Web que nos calcula el IVA (por ejemplo), lo podemos incluir dentro de nuestra aplicación (mediante llamadas) y ahorrarnos el trabajo de codificarlo nosotros.

Toda la estructura anterior está contenida en un entorno de desarrollo único llamado Visual Studio .NET. Este nuevo y completo entorno sustituye la anterior colección de entornos aislados como eran Visual Basic 6, Visual C++ y Visual InterDev. En un mismo entorno (VB .NET) un programador puede crear aplicaciones Web o locales, usar VB .NET o Visual C# .NET, diseñar con formularios Web o Windows, directamente o mediante HTML.

Capitulo II

Estructura de ASP .NET

El salto definitivo hacia la nueva estructura .NET vino motivado por el éxito comercial de la aplicación Java 2 Enterprise Edition (J2EE). Microsoft al verse a un segundo plano, creó una arquitectura integral que permitía al desarrollador de aplicaciones olvidarse completamente del sistema operativo, la gestión de memoria, etc., y mediante diferentes interfaces de programación soportadas por bibliotecas y plataformas de ejecución comunes, gestar aplicaciones y servicios Web o locales, esta arquitectura era .NET.

La plataforma .NET integra software de distintos lenguajes, además de programas por Internet y aplicaciones de servidores SQL Server. El objetivo es simplificar al máximo el código necesario para crear una aplicación.

Con tal fin, Microsoft se ha querido adelantar al futuro creando una arquitectura que permite la elaboración de aplicaciones ejecutables en Internet, teléfonos móviles, televisión digital, Intranets, etc. Éste es el éxito de .NET.

Fundamentos de ASP .NET

El entorno necesario para poder desarrollar aplicaciones ASP .NET es el nuevo producto de Microsoft: .NET Framework. Este entorno de programación permite tratar ASP .NET como un lenguaje del tipo orientado a objetos. En este punto es donde podemos afirmar que ASP .NET rompe completamente con las anteriores versiones de ASP.

Los puntos fundamentales de la nueva estructura ASP .NET son:

- Básicamente, los lenguajes para programar ASP .NET son: VB .NET, JScript y el nuevo Visual C# .NET, aunque realmente existen más de 20 (Perl .NET, Cobol .NET, etc.).

- ASP .NET forma parte de la estructura .NET (lenguajes orientados a objetos) y no es una versión ASP 4 (lenguajes interpretados).

- ASP .NET crea aplicaciones Web rápidas, escalables, manejables y flexibles, pero por encima de todo, son fáciles de entender y codificar.

- El código de las aplicaciones ASP .NET se compila a través del motor CLR, que compila JIT. Optimiza y almacena la compilación en memoria caché. Recuerde que el paso intermedio consistía en traducirlo a un lenguaje común MSIL.

- Los parámetros de configuración se almacenan en archivos de tipo XML, porque es de lectura universal y se puede generar con cualquier editor de textos.

- La seguridad de las aplicaciones ASP .NET es muy adaptable a las necesidades de cada situación, pues se basa en un conjunto de esquemas de autorización que puede configurarse ampliamente.

- ASP .NET puede acceder al potente grupo de librerías y clases que contiene .NET Framework para configurar transmisiones TCP/IP y Domain Name System (DNS), a través de XML y con los servicios Web.

Particularidades del lenguaje

Los tres lenguajes subministrados por Microsoft para la programación de aplicaciones Web con ASP .NET son JScript, VB .NET y Visual C# .NET. Otros fabricantes han enunciado muchos más: Python, Cobol .NET, Perl .NET, etc.

JScript ha sido modificado para comportarse como un lenguaje orientado a objetos. Antiguos desarrolladores notarán algunas diferencias, pero quedarán gratamente sorprendidos con los cambios efectuados.

VB .Net reemplaza VBScript como base en la programación ASP. El potencial de desarrollo, evidentemente, ha aumentado muchísimo pues no se ve limitado al reducido número de funciones que poseía el lenguaje script.

El nuevo lenguaje C# .NET en su estructura de programación se parece mucho a la del lenguaje C++. Se ha mejorado este último eliminando los errores y la optimización final se correspondería con C#.

Gracias al entorno .NET Framework, una aplicación Web escrita en un lenguaje determinado puede heredar funciones escritas en otros lenguajes. A su vez, dicha aplicación puede extenderse o referenciarse por otras aplicaciones escritas en otros lenguajes.

En definitiva, gracias al lenguaje común de compilación MSIL, la comunicación entre objetos y aplicaciones dentro de la arquitectura .NET no implica ningún problema añadido.

Hoy en día existen proyectos para ir más allá y poder extender el lenguaje MSIL entornos fuera de Windows (como hace Java). Son los proyectos Mono y Portable .NET.

Ejecución de los archivos ASP .NET

Cuando un visitante quiere acceder a un sitio Web, escribe la dirección URL en el navegador y éste realiza una petición HTML al servidor que está alojando ese sitio Web. En el momento en que el servidor recibe la petición, determina el tipo de archivo solicitado y lo envía a la aplicación correspondiente para que lo procese.

En el caso de paginas ASP .NET, éstas son compiladas (normalmente si es la primera vez que se seleccionan) y ejecutadas, reenviando al visitante los resultados de la consulta a través de su navegador.

La compilación realizada la primera vez implica un lapso de tiempo de reenvío mayor que con las anteriores versiones de ASP, pero, a diferencia de éstas, para todas las sucesivas peticiones de la misma página ASP .NET la respuesta será mucho más rápida.

Ejecución del lado del Cliente

En las aplicaciones ASP .NET se mezcla una parte de ejecución del lado del cliente y otra del lado del servidor. Cuando una página Web ASP .NET es descargada por el navegador de un visitante, en ella también se envía código para realizar comprobaciones e iniciar funciones del lado del cliente y así liberar de recursos al servidor. Previamente, el servidor ha determinado el tipo de navegador del cliente y en consecuencia, ha codificado las instrucciones a una versión HTML que el navegador pueda soportar.

Cuando el servidor recibe la respuesta de un formulario, los valores son guardados en una nueva herramienta de ASP .NET llamada bolsa de estado y son comprimidos y ocultados en una página llamada vista de estado. El objetivo es que, una vez enviado el formulario, éste recupere su apariencia anterior.

Debe de tener presente que los formularios Web no se comportan igual que un formulario Windows. En el último, si rellenamos una casilla o borramos un texto, lo escondemos detrás de otro formulario, etc. Windows lo recuperará automáticamente sin que tengamos que codificar ni una sola línea de código. En el caso de los formularios Web, esto no es así.

El procedimiento más habitual para que un navegador realice una petición a un servidor o le mande información es mediante el uso de los dos métodos HTML: GET y POST.

El método GET almacena toda la información que requiere dentro de la dirección URL.

Cuando un navegador envía información mediante el método POST, los datos se estructuran igual que en el método GET, pero se ubican en una cabecera HTML separada de la página, por lo que no son visibles. Por esta razón, en la mayoría de los casos los programadores prefieren este método. Cabe notar que en la cabecera también figura información útil como el tipo de navegador utilizado, etc.

Ejecución del lado del Servidor

Cuando el servidor recibe la petición, localiza la página usando la URL. A continuación y mediante especiales .DLL y objetos de la estructura .NET, compila y ejecuta la aplicación ASP .NET para generar la respuesta. La respuesta es reenviada al navegador traducida a código HTML y éste representa la respuesta en la pantalla del cliente.

El ciclo simplificado de ejecución del lado del servidor sería:

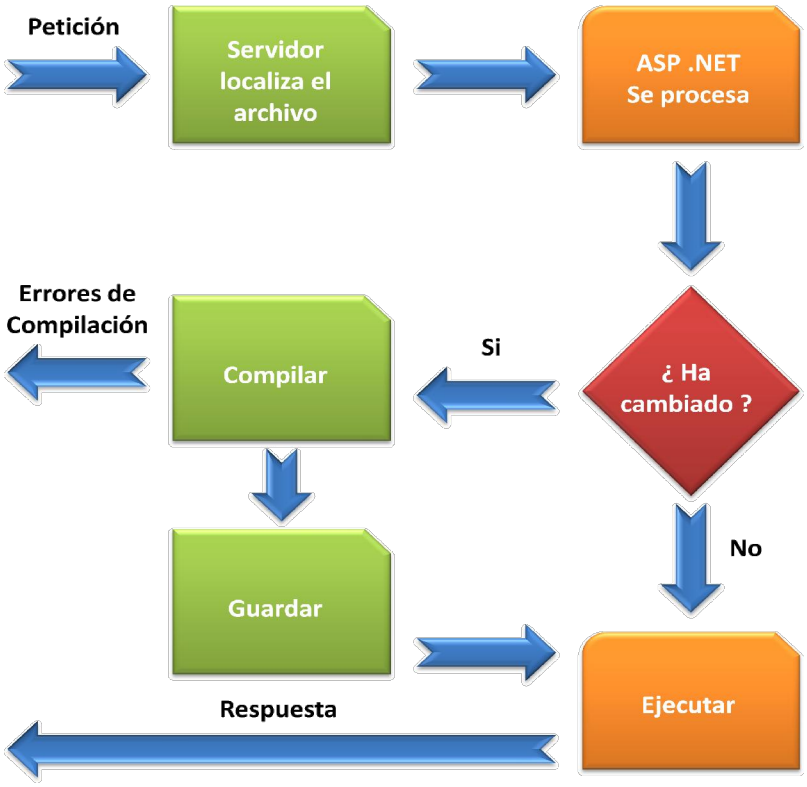

Figura 2. Ciclo simplificado de ejecución del lado servidor

Con un poco de detalle, los pasos que se siguen en el servidor desde que se recibe la petición hasta que se envía la respuesta son:

1. Internet Information Sever (IIS) compara la URL de la petición con una dirección física de del archivo en el sistema, traduciendo el directorio

virtual, por ejemplo: /tiempo/index.aspx en un directorio del sistema, por ejemplo: C:\Inetpub\wwwroot\tiempo\index.aspx.

2. Una vez se ha localizado el archivo, se identifica de qué tipo es, comparando la extensión .aspx con una lista que posee el sistema o porque lo identifica el propio cliente.

3. Si ésta es la primera vez que el cliente realiza una petición sobre esta página, ASP .NET la compila usando el motor CLR traduciéndola al lenguaje MSIL y posteriormente, al código binario, preparada para ejecutarse.

4. El código binario es una clase .DLL de la estructura .NET que se almacena en un archivo temporal.

5. La próxima vez que sea requerida esta página, el servidor comprobará si el código ha cambiado. Si es el mismo, entonces se omitirá la compilación y se procederá automáticamente a la ejecución. En caso contrario, la clase es eliminada y el código ASP .NET se vuelve a compilar.

6. El código compilado es ejecutado y los valores enviados en la petición (GET o POST) son interpretados.

7. El siguiente paso consiste en detectar el tipo de navegador que usa el cliente: Explorer, Netscape o teléfono móvil (Wireless Markup Lenguaje).

8. Se envía la respuesta al navegador del cliente.

Capitulo III

Instalación del Servidor Web

Estructura mínima necesaria

Para poder desarrollar aplicaciones Web con ASP .NET, es necesario tener configurado el ordenador como un servidor Web.

La estructura mínima necesaria para poder codificar, compilar y ejecutar páginas ASP .NET se basa en dos elementos: .NET Framework y IIS 5. Microsoft recomienda usar como sistemas operativos Windows XP Professional o Windows TI Server, aunque también puede funcionar en los sistemas Windows 2000 Professional o Server.

En todo caso, nosotros usaremos un entorno de desarrollo más amplio en el que queden integradas todas las funciones y lenguajes, así como las bibliotecas de clases, el motor de compilación CLR y demás herramientas. Nos referimos, evidentemente, a Visual Studio .NET.

Si el programador trabaja en un entorno Visual Studio .NET, no será necesario que ejecute las funciones en la línea de comandos (en MS-DOS) y además, podrá trabajar con otros sistemas operativos: Windows NT 4.0, Windows Me y Windows 98.

Instalación de elementos necesarios

El orden de instalación de los distintos elementos de software que formarán nuestro servidor Web es inalterable. Si no se respeta, el servidor no funcionará correctamente.

Instalaremos en el orden siguiente:

1. Sistema operativo (preferible Windows XP Profesional).
2. Navegador: mínimo Internet Explorer 5.5.
3. Internet Information Server 5 (IIS 5).
4. SQL Server.
5. .NET Framework.
6. Visual Studio .NET.

Instalación de IIS 5

Internet Information Server contiene el software necesario para que su ordenador pueda operar como un servidor Web. No es el más potente ni el único pero sí el más extendido y fácil de manejar. Usado junto a sistemas operativos como Windows XP Professional, está haciendo una fuerte competencia a servidores como Unix.

IIS es compatible con otras herramientas Microsoft: Word, Excel, Access, Power Point, ADO, ODBC, ASP .NET, etc., por lo que las posibilidades de construcción y publicación en Web son enormes.

Su punto débil es que no soporta un gran número de peticiones de clientes a la vez. Es muy útil para el aprendizaje y uso como banco de pruebas de aplicaciones de servidor, pero no para emplearlo en grandes empresas.

Junto con el paquete IIS 5, se incluye el servicio FTP de transferencia de archivos, el servicio SMTP de correo electrónico, los servicios OLEDB, ADO de acceso a base de datos, etc.

Antes de iniciar propiamente la instalación de IIS 5, asegúrese de que no está conectado a la red. Durante la instalación, su sistema es muy vulnerable a los ataques exteriores pues aún no ha configurado su seguridad.

Los pasos a seguir son:

1. Seleccione ***Inicio*** → ***Configuración*** → ***Panel de Control***.

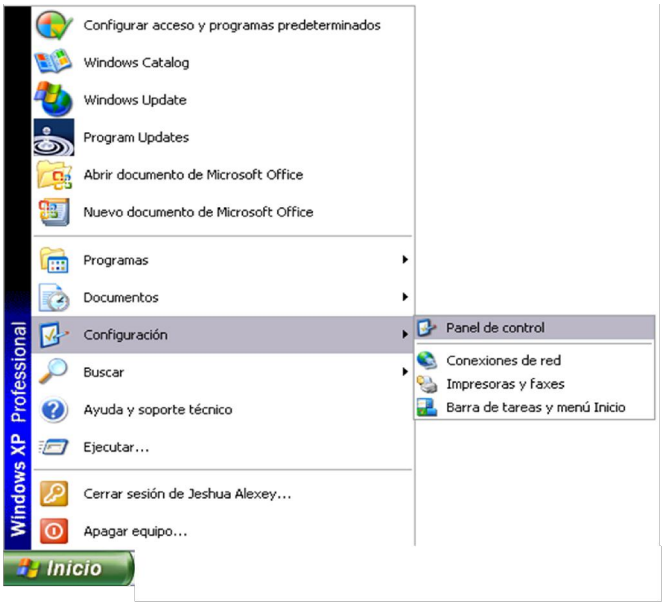

Figura 3. Inicio, Configuración, Panel de control

2. En el ***Panel de Control***, haga doble clic sobre el icono de ***Agregar o quitar programas***.

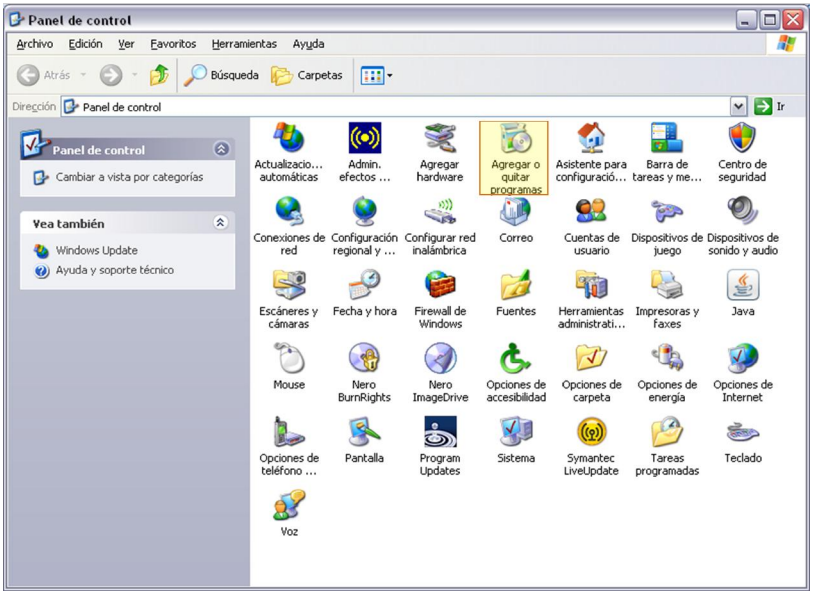

Figura 4. Panel de control, Agregar o quitar programas

3. Seleccione ***Agregar o quitar componentes de Windows*** al lado izquierdo de la pantalla.

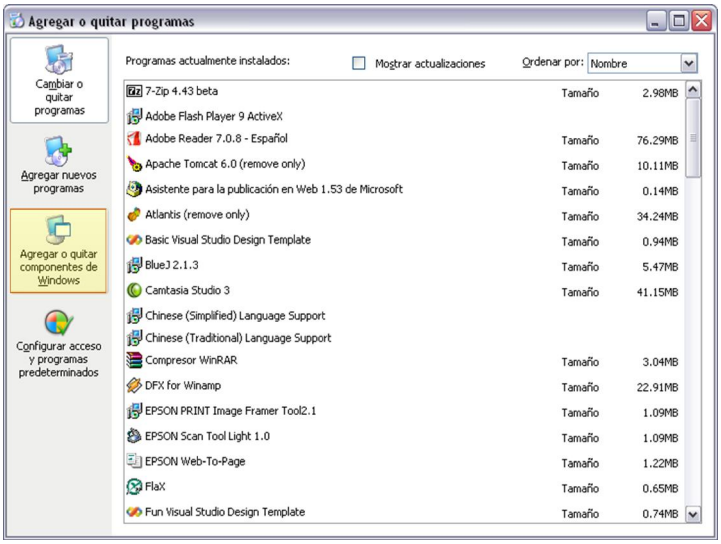

Figura 5. Agregar o quitar componentes de Windows

4. Seleccione ***Servicios de Internet Information Server (IIS)***.

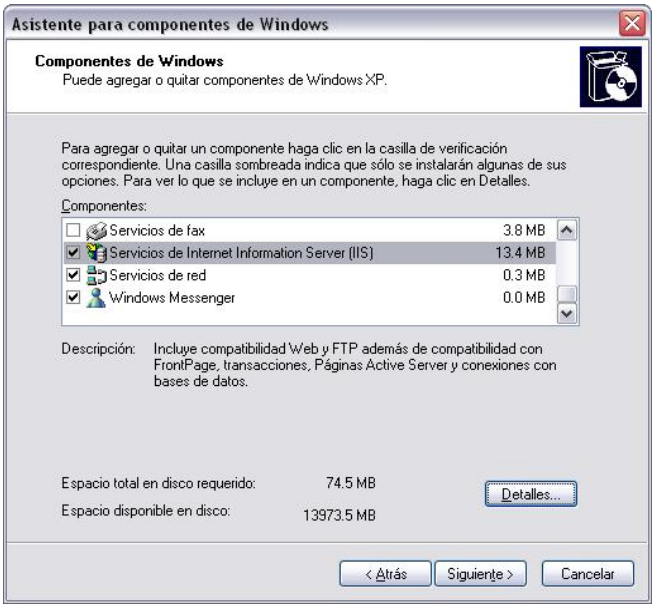

Figura 6. Asistente para componentes de Windows

5. Pulse el botón ***Detalles*** y seleccione los recursos que desee instalar en su servidor.

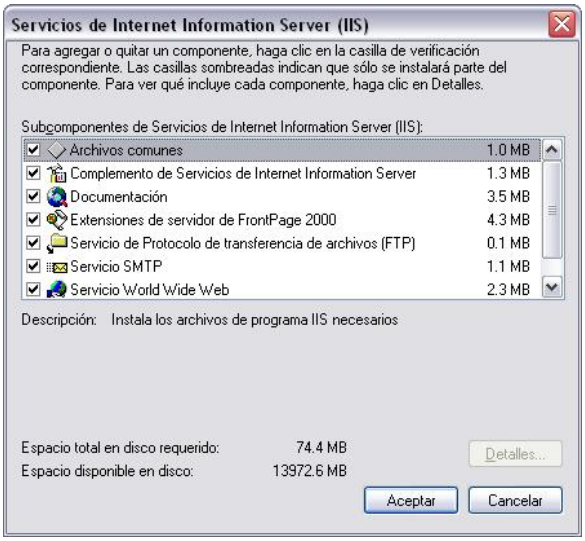

Figura 7. Servicios de Internet Information Server

6. Pulse ***Aceptar*** y luego ***Siguiente*** para proceder a la instalación. Recuerde que necesitará usar el CD-ROM del Sistema Operativo.

7. Al acabar, pulse ***Finalizar*** y cierre las ventanas de ***Agregar o quitar programas***.

Verificación de las extensiones de Servidor

El siguiente paso consistirá en verificar que las extensiones del servidor Front Page 2000 están correctamente configuradas. Este paso lo deben comprobar obligatoriamente aquellos servidores que utilizan archivos del tipo FAT, que son mucho más vulnerables a los hackers que los del tipo NTFS (los más comunes). Si en este momento no sabe qué tipo usa su servidor, pulse con el botón derecho sobre el icono de su disco duro (C:\) y en la ventana Propiedades aparecerá escrito.

En cualquier caso, los pasos a seguir son:

1. Seleccione ***Inicio*** → ***Configuración*** → ***Panel de Control***.

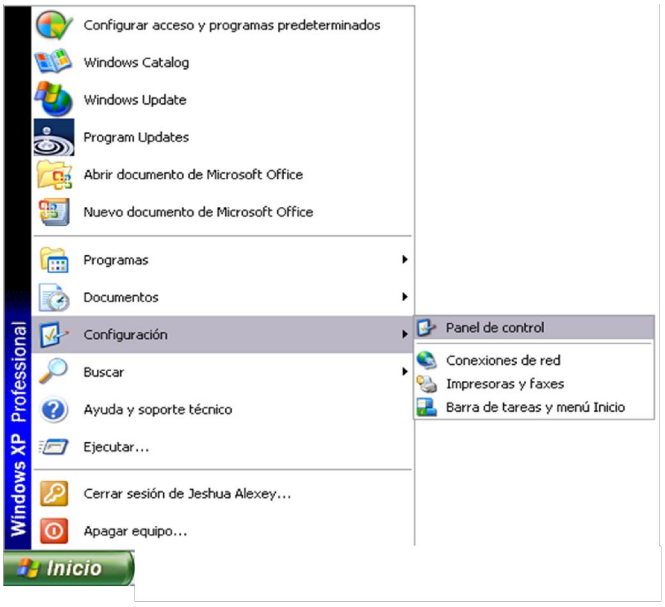

Figura 8. Inicio, Configuración, Panel de control

2. En el ***Panel de Control***, haga doble clic sobre el icono de ***Herramientas Administrativas***.

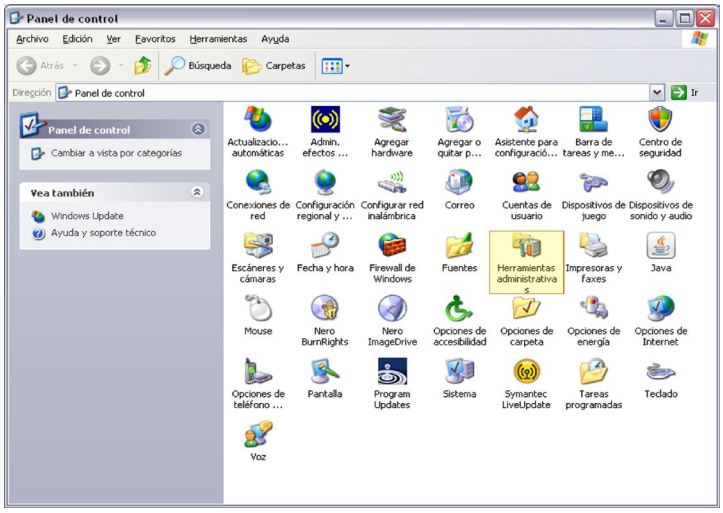

Figura 9. Panel de control, Herramientas administrativas

3. Haga doble clic sobre el icono de ***Administración de equipos***. Esto activará una instancia del programa Microsoft Management Console (MMC).

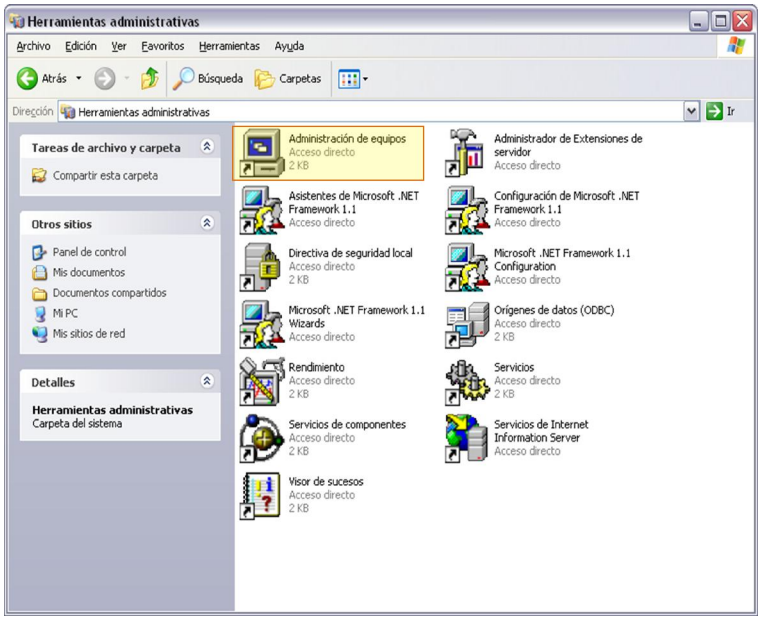

Figura 10. Herramientas administrativas, Administración de equipos

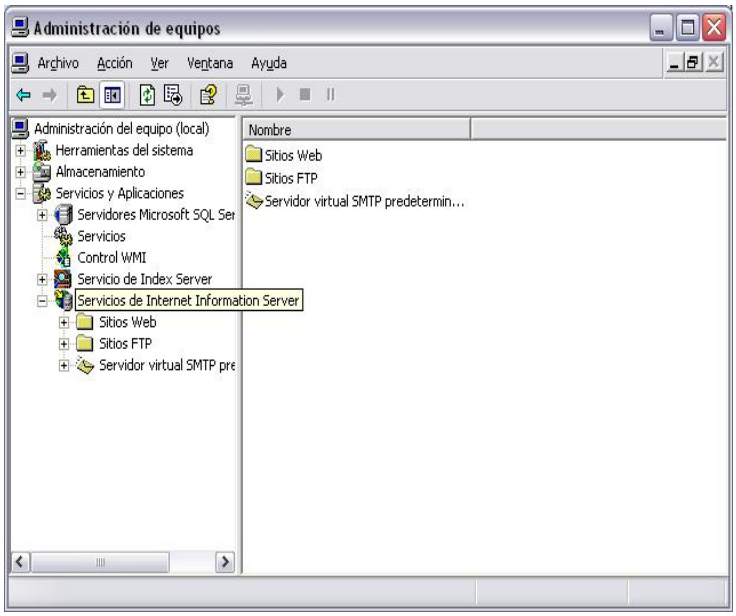

Figura 11. Administración de Equipos, Servicios de Internet Information Server

4. Expanda la carpeta ***Sitios Web***. Seleccione con el botón derecho del ratón ***Sitio Web predeterminado*** y a continuación ***Todas las tareas***, ***Comprobar extensiones de servidor***.

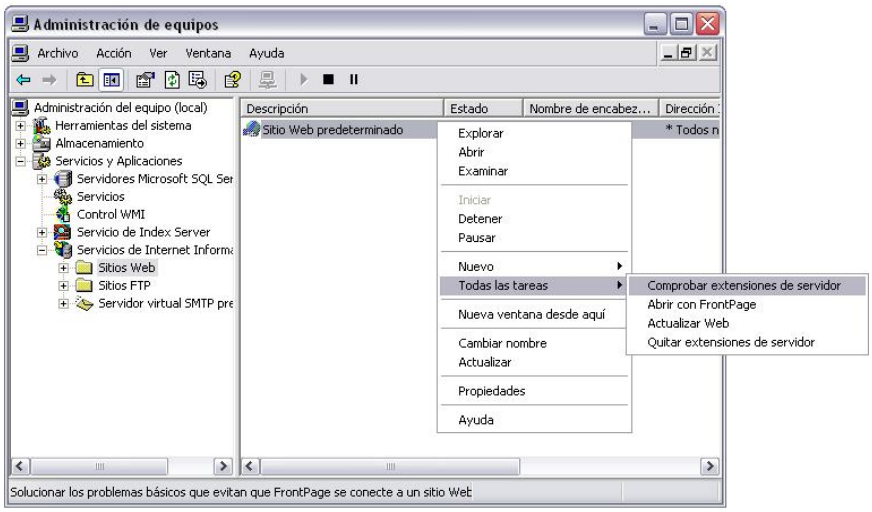

Figura 12. Administración de equipos, Comprobación de extensiones del servidor

5. Responda ***No*** a la pregunta y finalmente pulse ***Cerrar*** para salir.

Características de IIS

La primera característica que nos llama la atención es que no hace falta rebotar el servidor para detener y reiniciar el servicio de Internet. El control sobre el funcionamiento del servidor Web se controla desde MMC. Además, permite al administrador del servidor Web controlar el uso que de la CPU hacen los diferentes sitios Web y, en caso de requerirlo, puede limitar su uso. IIS puede manejar los mensajes de error para que sean más detallados. En caso de producirse un error en la ejecución de un sitio Web, IIS lanzará al navegador un mensaje de error que, normalmente, consistirá en traducir su código numérico a una expresión inteligible.

IIS permite que el administrador pueda acceder a él desde otro equipo mediante la red. La recomendación es que, aunque sea factible hacerlo, lo evite pues no deja de ser muy peligroso.

En cuanto a la seguridad, IIS ofrece una encriptación de contraseñas mediante hashing gracias a la cual el servidor añade información aleatoria alrededor del password. Así, en caso de ser interceptado, no se podrá usar, a diferencia de anteriores versiones en las que se encriptaba únicamente el password.

A nivel de comunicaciones, IIS soporta los sistemas Secure Sockets Layer (SSL) y Transport Layer Security (TLS) que restringen la comunicación entre clientes y servidores; con ello se logra que el servidor verifique quién es el cliente que intenta acceder al sistema antes de que éste se autentifique en el servidor.

Complementando los anteriores niveles de seguridad, también se pueden restringir el acceso al sistema a determinadas IP, o grupos de computadoras o dominios. Éste suele ser el caso de las Intranets que están al mismo tiempo en Internet. Dando únicamente permisos de acceso a los usuarios de la Intranet, evitamos que los usuarios de Internet tengan acceso a los documentos y archivos que pertenecen a la Intranet.

IIS normalmente ejecuta y controla los sitios Web en un entorno común. Pero, si queremos realizar pruebas con aplicaciones Web que pueden acarrear fallos en el sistema, IIS permite que se ejecuten Out of Process, aisladamente.

Una gran novedad es que IIS permite crear varios sitios Web con una sola IP. Gracias a las cabeceras del servidor host, una empresa puede hospedar distintos sitios Web con una sola IP merced al direccionamiento del IP, esto es, que cada cliente tendrá su propio dominio (realmente subdominio) y su IP particular.

Una limitación de IIS es el correo electrónico, pues sólo permite el envío de correo (SMTP, SimpleMail Transfer Protocol) y no la recepción (POP3, Post Office Protocol).

Se debe destacar la funcionalidad del servicio FrP para la publicación en red o descarga de archivo. En esta versión de IIS se permite continuar una descarga interrumpida.

La compresión de HTTP permite aumentar la velocidad de transmisión de sitios Web en el caso de ancho de banda limitado. Y, finalmente, el servicio Microsoft Index Server (MIS) permite indexar toda la información referente a los sitios Web que contiene un servidor IIS para facilitar su consulta. De este modo se puede configurar un buscador Web.

Acceder al Servidor Web

Podemos acceder al servidor Web para comprobar si se ha instalado correctamente IIS. Para ello simplemente debemos escribir http://localhost/localstart.asp en Internet Explorer y debería aparecer una página Web informando que IIS está correctamente instalado. Además, aparecerá la documentación de IIS en una ventana emergente, si es que fue instalada.

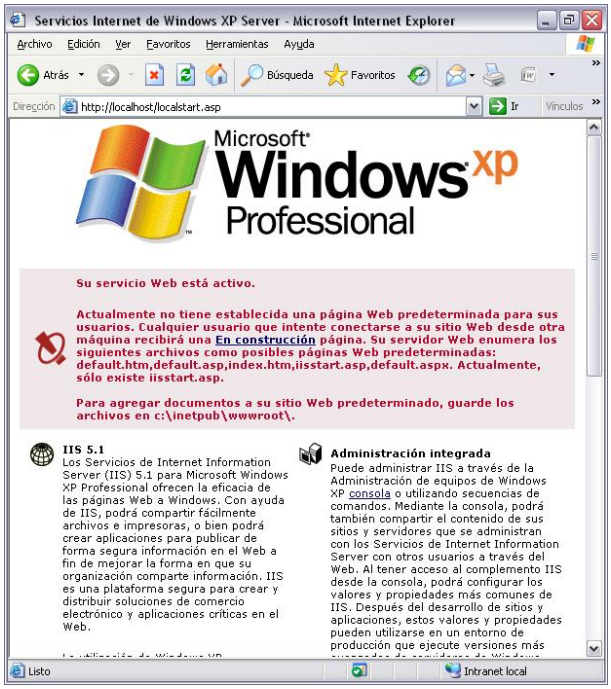

Figura 13. Página Web de información de IIS

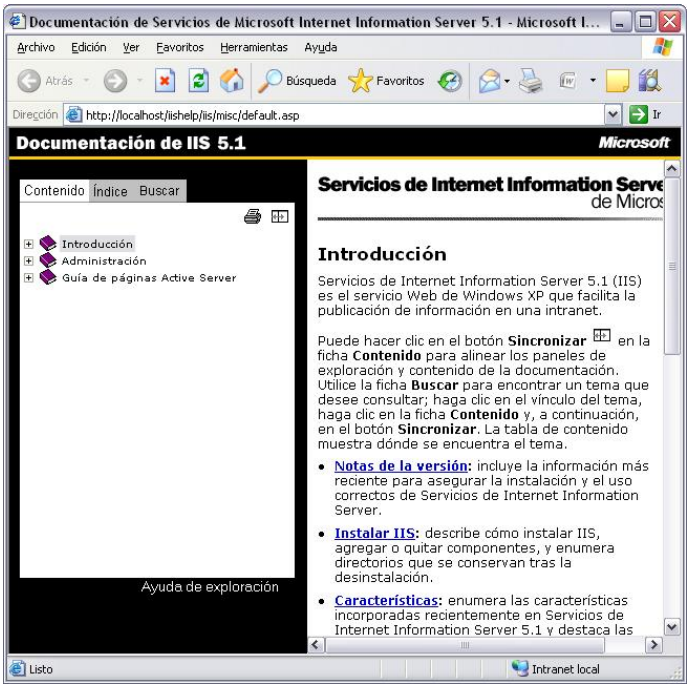

Figura 14. Documentación de IIS 5.1

Capitulo IV

Instalación de Visual Studio .NET

Preparación del entorno de trabajo

Antes de poder comenzar a escribir aplicaciones para .NET Framework, debemos instalar en nuestra máquina de trabajo las herramientas que nos permitirán el desarrollo de programas para este entorno de ejecución.

.NET Framework SDK

Se trata del kit de desarrollo de software para .NET Framework (Software Development Kit o SDK), que contiene la propia plataforma .NET y un conjunto de herramientas independientes, algunas funcionan en modo comando (en una ventana MS-DOS) y otras en modo gráfico. Los elementos imprescindibles para poder desarrollar aplicaciones para .NET están contenidos en este conjunto de herramientas.

Visual Studio .NET

Es la nueva versión de la familia de herramientas de desarrollo de software de Microsoft, naturalmente orientadas hacia su nuevo entorno de programación: .NET Framework.

Si bien es posible la escritura de programas empleando sólo el SDK de .NET Framework, este último, al estar compuesto de herramientas independientes, constituye un medio más cómodo de trabajo.

Visual Studio .NET al tratarse de un entorno de desarrollo integrado (Integrated Development Environment o IDE), aún a todas las herramientas del SDK: compiladores, editores, ayuda, etc., facilitando en gran medida la creación de programas.

Por este motivo, todas las explicaciones y ejemplos desarrollados a lo largo de este texto se harán basándose en este entorno de programación.

Requisitos Hardware

La siguiente tabla muestra una lista con las características mínimas y recomendadas que debe tener el equipo en el que instalemos Visual Studio .NET.

	Mínimo	Recomendado
Procesador	Pentium II – 450 MHz	Pentium IV 2.80 GHz
Memoria	128 MB	512 MB
Espacio en disco duro	3 GB	

Tabla 1. Características mínimas y recomendadas para instalar Visual Studio .NET

Sistema Operativo

Visual Studio .NET puede ser instalado en un equipo con uno de los siguientes sistemas operativos:

- Windows 2000 (se requiere tener instalado Service Pack 2).
- Windows NT 4.0. (se requiere tener instalado Service Pack 5).
- Windows Me.
- Windows 98.
- Windows XP Professional.

Para aprovechar todo el potencial de desarrollo de la plataforma, es recomendable usar como sistema operativo Windows XP Professional, ya que ciertos aspectos del entorno (las características avanzadas de gestión gráfica por ejemplo) no están disponibles si instalamos .NET en otro sistema con menos prestaciones.

Instalación de Visual Studio .NET

Microsoft Visual Studio .NET Enterprise Architect 2003 (Versión 7.1.6030), se compone de tres CD-ROM's de instalación del producto.

Procederemos insertando el disco de instalación rotulado como ***Microsoft Visual Studio .NET Enterprise Architect 2003 - 01***, el cuál detectará si es necesario actualizar algún componente a nivel del Sistema Operativo, pulsaremos sobre el paso 1 ***Visual Studio .NET Prerequisites***, en el que se nos pedirá el disco rotulado con el mismo nombre.

Figura 15. Selección de prerrequisitos para Visual Studio .NET

Una vez insertado el disco de Visual Studio .NET Prerequisites, se mostrará la pantalla de la *Figura 16*. En caso de aceptar el contrato de licencia para requisitos previos, haremos clic sobre ***Continuar***, para que el instalador detecte qué componentes faltan por actualizar.

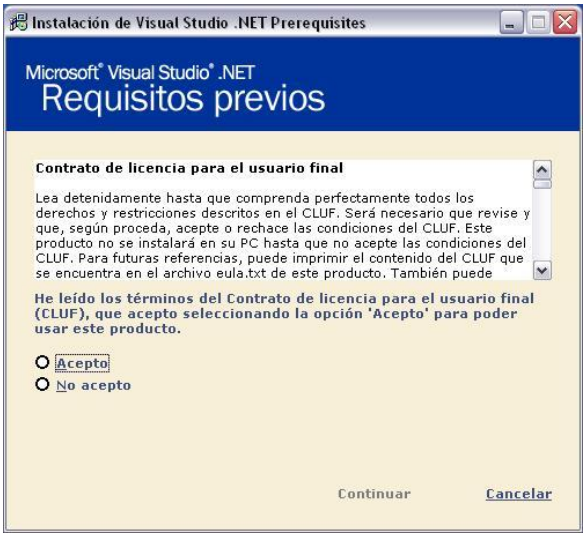

Figura 16. Contrato de licencia para requisitos previos

Una vez detectados los componentes que necesitan actualización, serán mostrados a continuación en la lista de la *Figura 17*, donde daremos clic sobre ***Instalar ahora***, con lo que se procederá a la actualización de los componentes de la lista. Una vez terminada esta actualización, daremos clic sobre ***Listo*** y seguiremos con la instalación normal de Visual Studio .NET, lo que nos requerirá de nuevo la introducción del disco de instalación rotulado como ***Microsoft Visual Studio .NET Enterprise Architect 2003 - 01***.

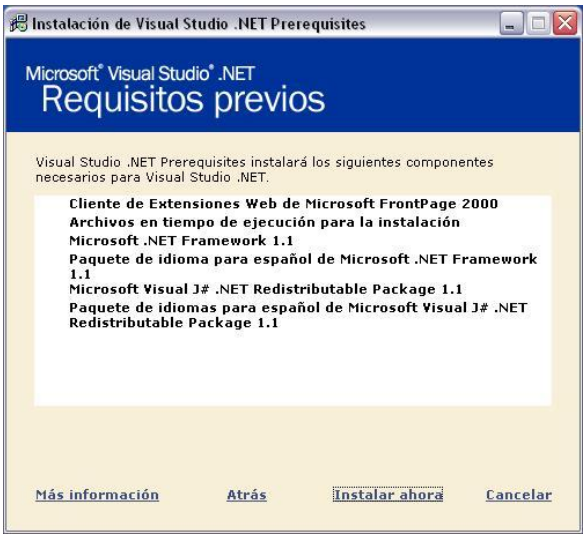

Figura 17. Lista de componentes necesarios para Visual Studio .NET

Una vez actualizado los componentes del sistema, el siguiente paso será la instalación de Visual Studio .NET, que pondremos en marcha al hacer clic sobre el paso 2 de la instalación, que tiene el nombre de ***Visual Studio .NET***. Ver *Figura 18*.

Figura 18. Instalación de Visual Studio .NET

A continuación se mostrará la pantalla con la información del contrato de licencia de Visual Studio .NET. En caso de estar de acuerdo con todos estos términos y aceptar el contrato, haremos clic sobre ***Continuar***. Ver *Figura 19*.

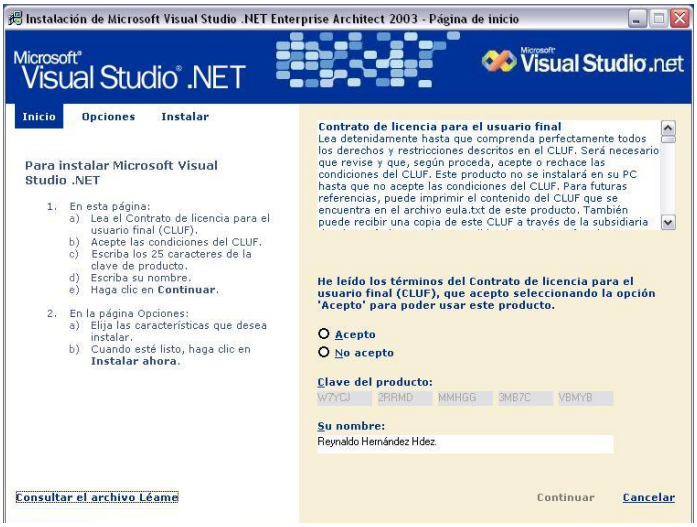

Figura 19. Información del contrato de licencia de Visual Studio .NET

A continuación debemos seleccionar aquellos elementos del producto que deseamos instalar, el entorno de ejecución, lenguajes, utilidades, ayuda, etc., y su ubicación en el disco duro, como muestra la *Figura 20*. Terminada la selección, pulsaremos sobre ***Instalar ahora*** para que comience el proceso.

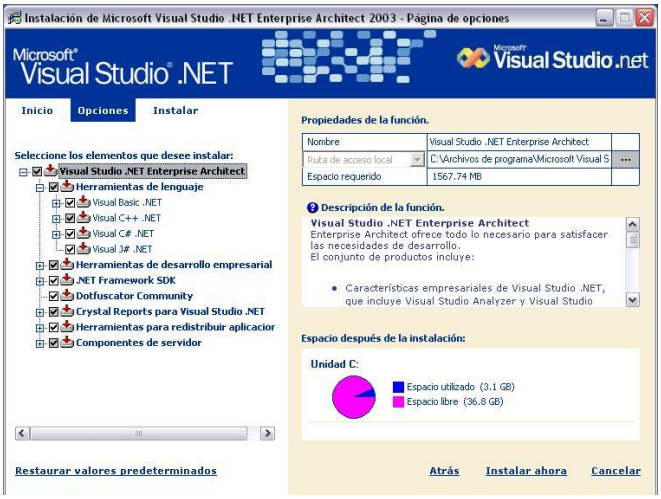

Figura 20. Selección de los elementos a instalar de Visual Studio .NET

Este proceso de instalación nos indica el archivo que se está instalando en cada momento, así como la información de su estado a través de una barra de progreso, el tiempo aproximado para realizar una instalación es de dos horas. Ver *Figura 21*.

Figura 21. Información sobre el progreso de la instalación

Durante la instalación, el programa solicitará el disco de instalación rotulado como ***Microsoft Visual Studio .NET Enterprise Architect 2003 - 02***.

Concluida la instalación, el programa nos informará de si se produjo alguna incidencia. En caso de que no se hayan producido errores, finalizaremos haciendo clic sobre ***Listo***, con lo que ya tendremos instalado Microsoft Visual Studio .NET Enterprise Architect 2003 en nuestro ordenador. Ver *Figura 22*.

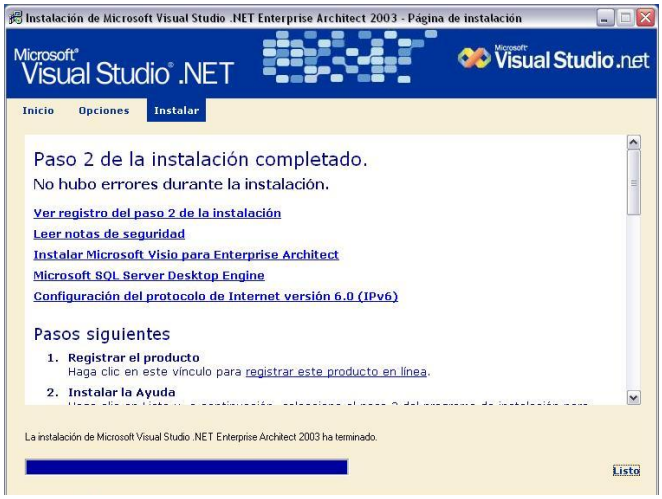

Figura 22. Instalación completa

Capitulo V

Uso de Visual Studio .NET

Introducción

Visual Studio .NET es el entorno de desarrollo global que se utiliza para crear potentes y fiables soluciones Web empresariales. Ofreciendo capacidades de desarrollo Web extremo a extremo y componentes de servidor escalables y reutilizables, Visual Studio .NET permite incrementar la productividad y ayudar a crear aplicaciones y sitios Web ASP.NET más eficazmente.

¿Por qué Visual Studio .NET?

Visual Studio .NET simplifica el desarrollo de soluciones Web empresariales potentes y fiables e incrementa la eficacia del desarrollador al proporcionar un entorno de desarrollo compartido y familiar. Los componentes ya desarrollados, los asistentes de programación y la posibilidad de reutilizar componentes escritos en cualquier lenguaje pueden reducir significativamente el tiempo de desarrollo. La opción de completar el código basada en Microsoft IntelliSense® permite producir código preciso más rápidamente. El potente soporte de depuración multilenguaje extremo a extremo ayuda a que las aplicaciones estén operativas más rápidamente.

Visual Studio .NET tiene un único IDE que proporciona un aspecto coherente, con independencia del lenguaje de programación que se utilice o el tipo de aplicación que se desarrolle. Las características que antes estaban disponibles para un único lenguaje, ahora están disponibles para todos los lenguajes.

Visual Studio .NET soporta el desarrollo en varios de los lenguajes basados en Microsoft .NET. Este soporte de diversos y distintos lenguajes permite a los desarrolladores trabajar con el lenguaje que prefieran, puesto que ya no necesitan aprender un nuevo lenguaje para cada nuevo proyecto.

Los lenguajes que se incluyen en Visual Studio .NET son:

- Microsoft Visual Basic® .NET
- C#

- J#
- Microsoft Visual C++®

Visual Studio .NET soporta el desarrollo de múltiples tipos de proyectos, que abarcan desde aplicaciones basadas en Microsoft Windows® hasta aplicaciones Web ASP.NET y servicios Web XML.

Este soporte para múltiples tipos de proyectos permite trabajar de forma simultánea en varios proyectos sin necesidad de cambiar el entorno de desarrollo o aprender nuevas interfaces de herramientas o lenguajes.

Visual Studio .NET contiene un navegador integrado basado en Microsoft Internet Explorer. El navegador está integrado en el IDE y puede accederse a él desde múltiples ventanas y menús.

Esta accesibilidad al navegador permite visualizar el sitio Web durante el ciclo de desarrollo sin tener que transferir a otro programa y volver a escribir las cadenas de la URL (Uniform Resource Locator).

Visual Studio .NET está diseñado para soportar depuración desde el código inicial hasta el despliegue de la aplicación. El soporte de depuración incluye puntos de interrupción, expresiones break, expresiones watch y la posibilidad de recorrer el código de instrucciones o procedimientos paso a paso.

El IDE de Visual Studio .NET puede personalizarse a nivel de ventanas y herramientas. Por tanto, podemos mostrar únicamente las herramientas o ventanas que utilicemos en un momento determinado y ocultar el resto.

Página de inicio

Cada vez que iniciamos Visual Studio .NET, se muestra la página de inicio. Esta página proporciona una ubicación centralizada para establecer preferencias, leer noticias sobre productos, acceder a discusiones con otros desarrolladores y obtener otra información que puede utilizarse para iniciarse en el entorno de Visual Studio .NET.

Podemos visualizar la página de inicio en cualquier momento mientras trabajamos en el entorno de desarrollo.

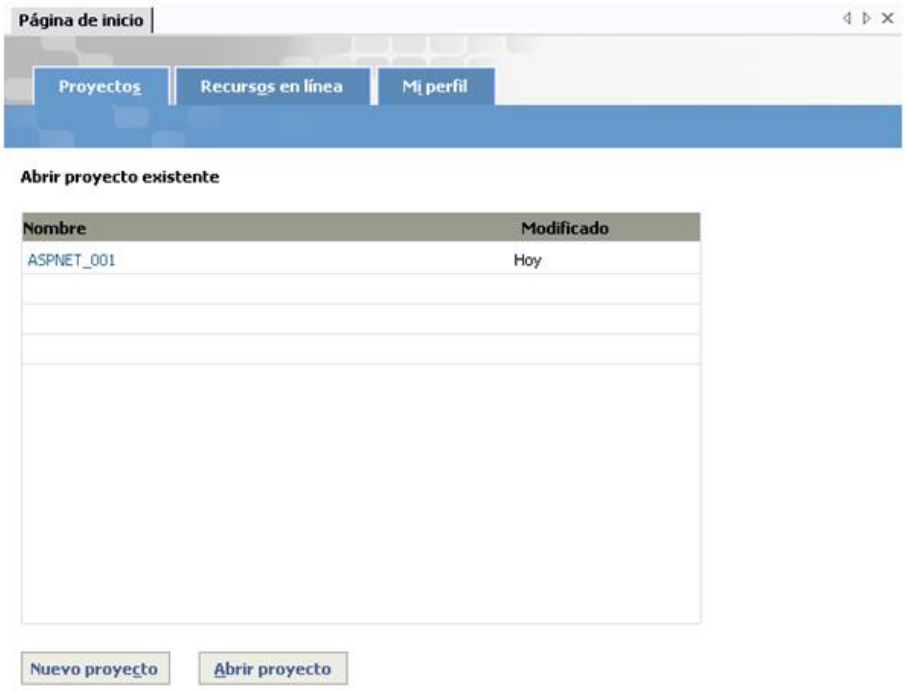

Figura 23. Página de inicio

Si hacemos clic en Introducción, la ayuda dinámica muestra temas sobre el inicio de nuevos proyectos, y se muestran las dos carpetas siguientes:

- **Proyectos**

  La carpeta Proyectos muestra enlaces a los últimos proyectos en los que se ha trabajado. Esta carpeta permite abrir rápidamente Visual Studio .NET y cargar todos los archivos relacionados con los proyectos actuales.

- **Recursos en línea**

  Hacer clic en Comunidad en línea proporciona acceso a los sitios de Microsoft Visual Studio .NET Web y los grupos de noticias relacionados.

- **Mi perfil**

  Hacer clic en Mi perfil permite establecer un perfil de usuario que ajusta el cuadro de herramientas, el nuevo proyecto predeterminado y la ayuda dinámica para adaptarlos a nuestras preferencias de programación. Podemos cambiar nuestro perfil en cualquier momento para modificar estas opciones de configuración personalizadas.

  Tenemos la opción de escoger un perfil ya existente, como Desarrollador Visual Basic, o modificar manualmente cada uno de los elementos del perfil.

Plantillas de proyectos disponibles

Visual Studio .NET proporciona plantillas que soportan la creación de los tipos de proyectos más habituales. Estas plantillas contienen todos los archivos necesarios y trabajan con nuestro perfil para ajustar el IDE a la configuración correcta del proyecto seleccionado.

Estas plantillas nos ayudan a aprovechar mejor el tiempo al permitir que nos concentremos en agregar funciones al proyecto y no tener que establecer la infraestructura cada vez que cambiamos de tipo de proyecto.

Cuando creamos un proyecto en Visual Studio .NET, también creamos un contenedor de mayor tamaño denominado Solución. Esta Solución puede contener múltiples proyectos, del mismo modo que un contenedor de proyectos puede contener múltiples páginas.

Las soluciones permiten que nos concentremos en el proyecto o conjunto de proyectos necesarios para desarrollar e implementar nuestra aplicación, en lugar de tener que centrarnos en los detalles de administración de los objetos y de los archivos que los definen. Al utilizar el concepto de Solución como contenedor, ésta nos permite:

- Trabajar con múltiples proyectos en una misma instancia del IDE.

- Trabajar con elementos, configuraciones y opciones aplicables a un grupo de proyectos.

- Administrar archivos misceláneos que se abren fuera del contexto de una Solución o de un Proyecto.

- Utilizar el Explorador de soluciones, que es una visualización gráfica de nuestra solución, para organizar y administrar todos los proyectos y archivos necesarios para diseñar, desarrollar e implementar una aplicación.

Visual Studio .NET incluye múltiples plantillas de proyectos clasificadas por lenguaje y tipo. Para seleccionar la plantilla correcta, es necesario especificar antes el lenguaje en el que trabajaremos.

Las plantillas de proyectos disponibles en Visual Basic y Microsoft Visual C#™ incluyen:

- **Aplicación para Windows**

  La plantilla de proyecto Aplicación para Windows se utiliza para crear aplicaciones Windows estándar. Esta plantilla agrega automáticamente las referencias y archivos de proyecto esenciales que la aplicación necesita como punto de inicio.

- **Biblioteca de clases**

  La plantilla Biblioteca de clases se utiliza para crear clases y componentes reutilizables que pueden compartirse con otros proyectos.

- **Biblioteca de controles de Windows**

  La plantilla Biblioteca de controles de Windows se utiliza para crear controles personalizados para usarlos en formularios Windows Forms.

- **Aplicación Web ASP.NET**

  La plantilla de proyecto Aplicación Web ASP.NET se utiliza para crear una aplicación Web ASP.NET en un equipo en el que se haya instalado Internet Information Server (IIS) versión 5.0 o posterior. La plantilla crea los archivos básicos que se necesitan en el servidor como ayuda para iniciar el diseño de la aplicación.

- **Servicio Web ASP.NET**

  La plantilla de proyecto Servicio Web ASP.NET se utiliza para escribir un servicio Web XML que pueda ser consumido por otros servicios o aplicaciones Web en una red. Los servicios Web XML son componentes disponibles en Internet y están diseñados para interactuar únicamente con otras aplicaciones Web.

- **Biblioteca de controles Web**

  La plantilla Biblioteca de controles Web se utiliza para crear controles de servidor Web personalizados. La plantilla agrega los elementos de proyecto necesarios para iniciar la creación de un control que pueda agregarse posteriormente a cualquier proyecto Web.

- **Aplicación de consola**

  Normalmente, las aplicaciones de consola se diseñan sin una IU gráfica y se compilan en un archivo ejecutable autónomo. Una aplicación de consola se ejecuta desde la línea de comandos intercambiando la información de entrada y de salida entre el símbolo de sistema y la aplicación que se está ejecutando.

- **Servicio de Windows**

  La plantilla Servicio de Windows se utiliza para crear aplicaciones de servicios de Windows, aplicaciones ejecutables de ejecución de larga duración que se ejecutan en su propia sesión de Windows.

- **Proyecto vacío**

  La plantilla Proyecto vacío se utiliza para crear un tipo de proyecto propio. La plantilla crea la estructura de archivos necesaria para almacenar la información sobre la aplicación. Las referencias, archivos o componentes deben agregarse a la plantilla manualmente.

- **Proyecto Web vacío**

  La plantilla Proyecto Web vacío es para usuarios avanzados que deseen empezar con un proyecto vacío. La plantilla crea la estructura de archivos necesaria para un proyecto basado en servidor en un servidor IIS. Las referencias y componentes (como páginas de formularios Web Forms) deben agregarse manualmente.

- **Nuevo proyecto en carpeta existente**

  La plantilla de proyecto Nuevo proyecto en carpeta existente se utiliza para crear un proyecto en blanco en un directorio de aplicación existente. Podemos decidir agregar los archivos desde el directorio de la aplicación existente en el nuevo proyecto haciendo clic con el botón derecho en cada uno de los elementos del Explorador de soluciones y seleccionando Incluir en el proyecto del menú contextual.

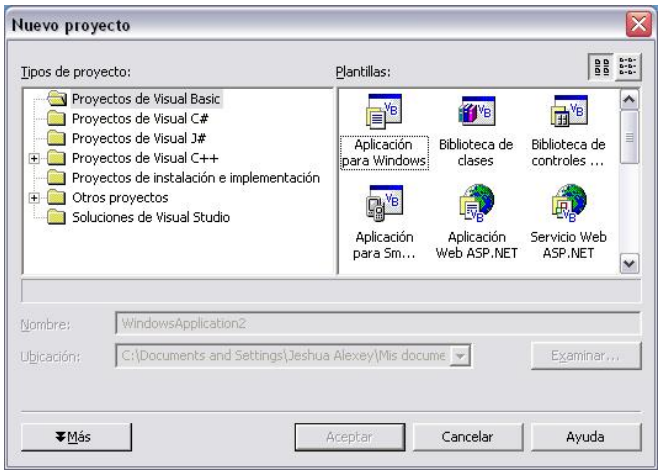

Figura 24. Ventana Nuevo proyecto

Entorno de desarrollo integrado (IDE)

El IDE de Visual Studio .NET contiene múltiples ventanas que proporcionan diversas herramientas y servicios. Muchas de las características de Visual Studio .NET están disponibles desde varias de las ventanas, menús y barras de herramientas del IDE.

No podemos mover ni ocultar las ventanas del IDE dependiendo de nuestras preferencias personales. El menú **Ver** se utiliza para seleccionar las ventanas a mostrar. Podemos hacer clic en el botón del alfiler **Ocultar automáticamente** para convertir ventanas estáticas en ventanas emergentes.

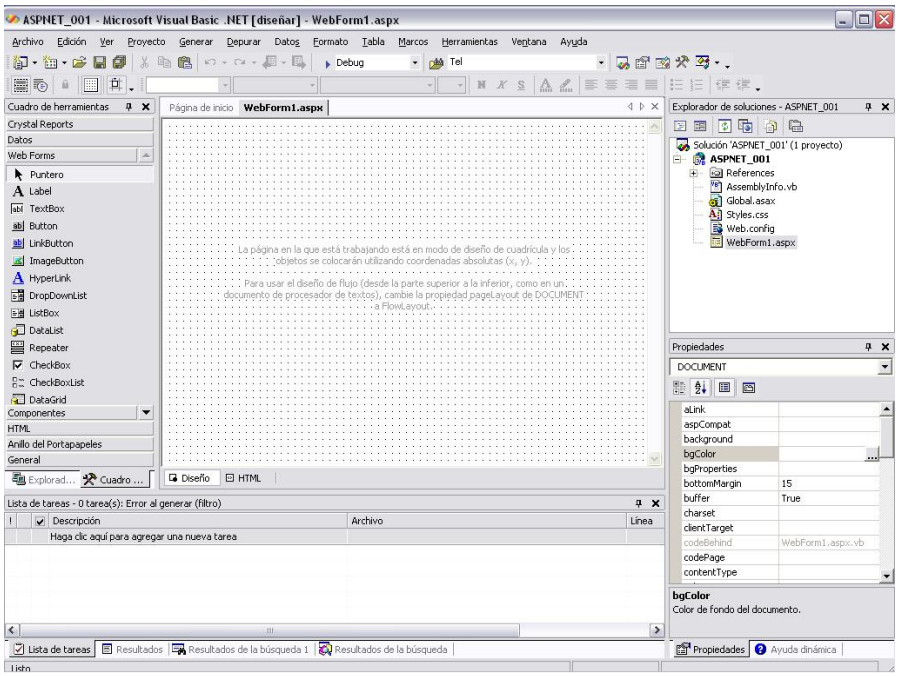

Figura 25. Entorno de Desarrollo Integrado de Microsoft Visual Studio .NET

Editor/navegador

El editor/navegador es la ventana de interfaz principal de Visual Studio .NET. En modo editor, el editor/navegador muestra el código para editar y

proporciona una interfaz gráfica qué usted ve es lo que usted consigue, para la ubicación de los controles. Podemos arrastrar y soltar para crear el diseño visual de nuestra aplicación. A continuación, podemos gestionar el diseño lógico de la aplicación modificando el código del control Web predeterminado.

Hay dos opciones para pantalla del editor: modo Diseño y modo HTML:

- **Modo Diseño**

  En modo Diseño, el editor permite mover los controles y los elementos gráficos por la ventana mediante una sencilla operación de arrastrar y soltar.

  Visual Studio .NET proporciona dos esquemas de posicionamiento de controles para diseñar páginas Web: FlowLayout y GridLayout. En FlowLayout, los controles se ubican en la página uno a continuación del otro, mientras que GridLayout permite posicionar con exactitud cada control, agregando automáticamente etiquetas DHTML (Dynamic Hypertext Markup Language) a los controles.

  Cuando agregamos un control a una página Web en modo Diseño, Visual Studio .NET agrega al formulario Web Form código de soporte y predetermina las propiedades. A continuación, podemos cambiar a modo HTML para mostrar el código y editarlo.

- **Modo HTML**

  En modo HTML, Visual Studio .NET destaca nuestro código para que los distintos elementos, como los nombres de variables y las palabras clave, sean identificables al instante. La característica de IntelliSense proporciona sugerencias para la finalización automática y permite generar funciones simplemente seleccionando desde listas de sintaxis disponibles.

Cuando utilizamos la ventana del editor en modo HTML, aparecen dos listas desplegables en la parte superior de la ventana: la lista Nombre de Clase, a la izquierda, y la lista Nombre de Método, a la derecha. La lista Nombre de Clase muestra todos los controles del formulario asociado. Si hacemos clic en el nombre de un control de la lista, la lista Nombre de Método mostrará todos los eventos de ese control. Los eventos son acciones que puede realizar el control y que pueden ser interpretadas por nuestra aplicación. Utilizando conjuntamente las listas Nombre de Clase y Nombre de Método, podemos localizar y editar rápidamente el código de nuestra aplicación.

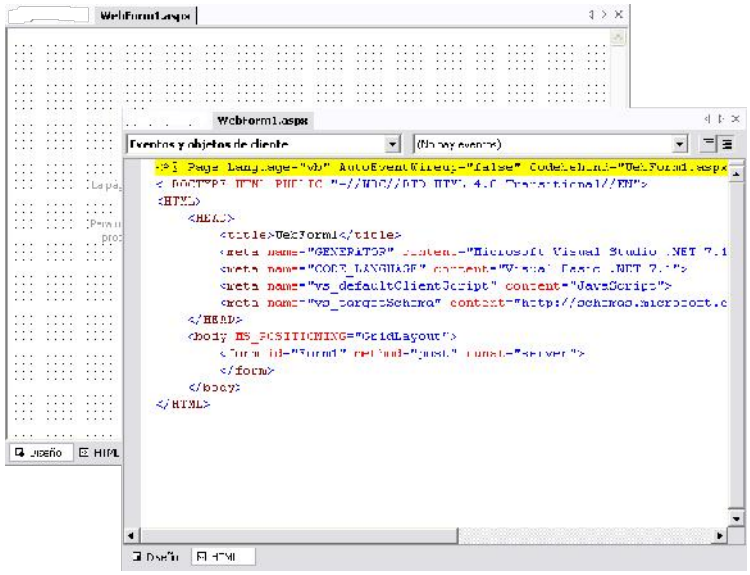

Figura 26. Editor/ Navegador

Examinador de objetos

El Examinador de objetos es una herramienta que proporciona información sobre los objetos y sus métodos, propiedades, eventos y constantes.

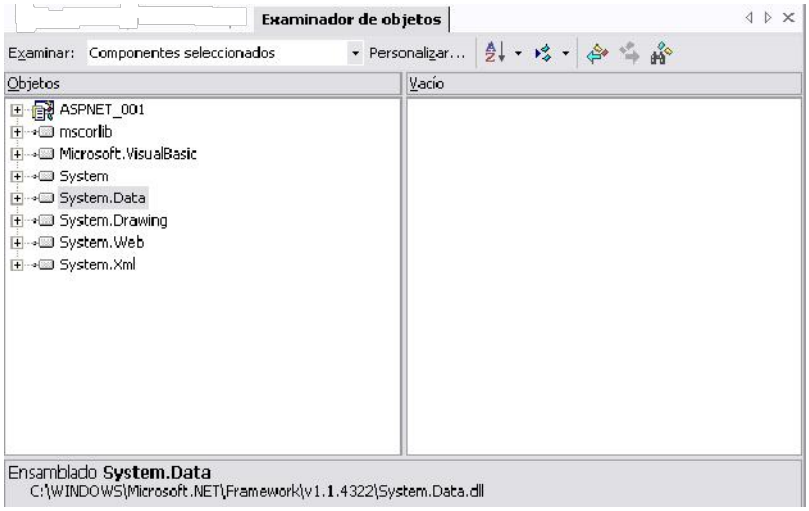

Figura 27. Examinador de objetos

Explorador de soluciones

El Explorador de soluciones muestra la jerarquía de los archivos del proyecto. Desde esta ventana, podemos mover y modificar archivos, por ejemplo:

- Utilizar una operación de arrastrar y soltar para reorganizar elementos.
- Seleccionar un elemento del Explorador de soluciones y la ventana Propiedades mostrará las propiedades de ese elemento. Esto permite cambiar las propiedades a nivel de proyecto o de página.
- Hacer clic con el botón derecho en el archivo, proyecto o solución para ver las opciones disponibles, incluyendo agregar, generar y editar páginas.

Los tipos de archivos que muestra el Explorador de soluciones incluyen:

- Referencias de proyectos que listan las clases que utiliza la página y los controles Web.
- Todos los formularios Web Forms del proyecto.

- Todas las páginas de código subyacente que contienen la lógica que soporta los formularios Web Forms.

- Carpetas relacionadas con proyectos y sub-elementos.

Figura 28. Explorador de soluciones

Ayuda dinámica

La ayuda dinámica proporciona acceso a temas de la ayuda local y en línea, en función de la configuración de Mi perfil, el Tipo de proyecto y la ubicación actual del cursor. A medida que nos movemos por el IDE o editamos código, las opciones disponibles en la ayuda dinámica se ajustan para corresponderse con nuestra actividad.

Figura 29. Ayuda dinámica

Propiedades

Visual Studio .NET permite ajustar las propiedades de documentos, clases y controles utilizando una ventana común de Propiedades. Cuando creamos o seleccionamos un elemento, la ventana Propiedades muestra automáticamente las propiedades relacionadas. Como muestra la siguiente ilustración, las propiedades disponibles se listan en la columna izquierda, mientras que las opciones de configuración se listan a la derecha.

Figura 30. Propiedades

Lista de tareas

La Lista de Tareas permite hacer un seguimiento del estado de las tareas mientras se desarrollan las aplicaciones. Visual Studio .NET también utiliza la Lista de tareas para señalar errores cuando se genera una aplicación.

Existen varias formas de agregar una tarea a la Lista de tareas, por ejemplo:

- Agregar tareas manualmente, haciendo clic en la Lista de tareas e introduciendo elementos. La tarea superior de la Lista de tareas en la siguiente imagen de pantalla es una tarea añadida manualmente.

- Visual Studio .NET agrega automáticamente una tarea con símbolos, como el comentario 'TODO en el código. La segunda tarea de la siguiente imagen de pantalla ha sido añadida automáticamente por Visual Studio .NET debido a un comentario 'TODO del código. Para acceder a esta sección del código, hacemos clic en el elemento de la Lista de tareas y Visual Studio .NET abrirá la página referida en esa línea de comentario. Existen varios símbolos (tokens) preestablecidos que pueden utilizarse en el código y que añadirán automáticamente una tarea a la Lista de tareas. Ver y agregar a la lista de símbolos:

  a) En el menú Herramientas, hacer clic en Opciones.

  b) b. En el cuadro de diálogo Opciones, en la carpeta Entorno, hacer clic en Lista de tareas.

- Visual Studio .NET agrega automáticamente los errores producidos en la generación a la Lista de tareas. La tarea inferior de la siguiente imagen de pantalla se añadió automáticamente cuando se generó la página. Para acceder a esta sección del código, hacer clic en el elemento de la lista de tareas y Visual Studio .NET abrirá la página referida en la línea que contiene el error.

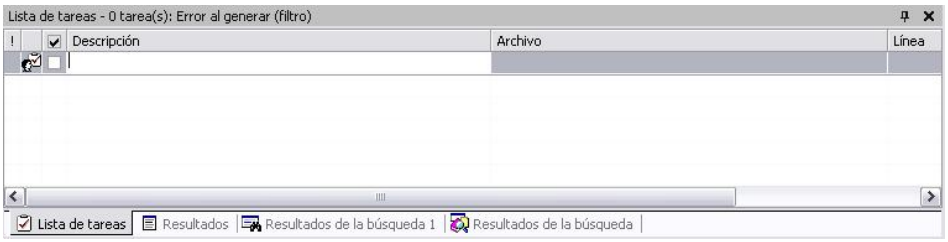

Figura 31. Lista de tareas

Explorador de servidores

El Explorador de servidores permite examinar conexiones a datos locales, servidores y servicios de Windows. El Explorador de servidores soporta la integración de servicios externos en nuestro sitio Web.

Figura 31. Explorador de servidores

Cuadro de herramientas

El Cuadro de herramientas permite arrastrar y soltar los controles que formarán nuestra aplicación.

Las herramientas disponibles se agrupan por categorías en los siguientes menús:

- **Datos**

  Esta categoría contiene objetos que permiten a la aplicación conectarse y acceder a datos de una base de datos Microsoft SQL Server™ y otras bases de datos.

- **Web Forms**

  Esta categoría contiene un conjunto de controles de servidor que pueden agregarse a páginas Web.

- **Componentes**

  Esta categoría contiene componentes que soportan la infraestructura de nuestra aplicación.

- **HTML**

  Esta categoría contiene un conjunto de controles HTML que pueden agregarse a la página Web. Estos controles pueden ejecutarse tanto en el lado servidor como en el lado cliente.

Figura 32. Cuadro de herramientas

Crear un proyecto de aplicación Web ASP.NET

Visual Studio .NET contiene todo lo necesario para generar una aplicación Web ASP.NET de principio a fin.

Crear una aplicación Web ASP.NET con Visual Studio .NET implica los siguientes pasos básicos:

1. **Crear una especificación de diseño.**

   La especificación de diseño es la guía que utilizaremos cuando creemos una aplicación Web. Debemos tomarnos tiempo antes de escribir código para diseñar la aplicación que crearemos. Aunque Visual Studio .NET proporciona herramientas que nos ayudan a desarrollar rápidamente una solución, tener una idea clara de las necesidades del

usuario y del conjunto inicial de características nos ayudará a ser más eficaces en nuestro trabajo de desarrollo. Empezar con una especificación de diseño también nos ayudará a ahorrar tiempo minimizando el potencial de reescritura de código debido a una pobre o inexistente especificación de diseño.

2. **Crear un nuevo proyecto.**

   Cuando seleccionamos una nueva plantilla de proyecto, Visual Studio .NET crea automáticamente los archivos y el código predeterminado necesarios para soportar el proyecto.

   Como parte de esta creación inicial del proyecto, deberíamos transferir las principales tareas de codificación desde nuestra especificación de diseño a la Lista de tareas de Visual Studio .NET. Esta transferencia permite hacer un seguimiento de nuestro desarrollo contra la especificación.

3. **Crear la interfaz.**

   Para crear la interfaz de nuestra aplicación Web, en primer lugar necesitaremos ubicar controles y objetos en las páginas Web utilizando la ventana Editor/Navegador en modo Diseño.

   A medida que agregamos objetos a un formulario, podemos establecer sus propiedades desde la tabla en la ventana Propiedades o como código en la ventana de Edición.

4. **Escribir código.**

   Tras establecer las propiedades iníciales del formulario Web Form ASP.NET y sus objetos, podremos escribir los procedimientos de eventos que se ejecutarán al realizar diferentes acciones sobre un control u objeto.

Es posible que también necesitemos escribir código para agregar lógica de negocio y para acceder a datos.

5. **Generar.**

   Cuando generamos un proyecto, compilamos todo el código de las páginas Web y demás archivos de clases en una librería de enlace dinámico (D*ynamic-Link Library*, DLL) denominada *ensamblado*.

   Visual Studio .NET tiene dos opciones de generación: *debug* y *release*. Cuando desarrollamos un proyecto por primera vez, generamos versiones de depuración. Cuando estamos preparados para liberar el proyecto, crearemos una versión *release* del proyecto.

6. **Probar y depurar.**

   Probar y depurar no es un paso a realizar una única vez, sino algo que se realiza repetidamente durante el proceso de desarrollo. Cada vez que realizamos un cambio importante, es necesario generar una versión de depuración de la aplicación para asegurarnos de que funciona según lo previsto.

   Visual Studio .NET ofrece numerosas herramientas de depuración que podemos utilizar para encontrar y solucionar errores de nuestra aplicación.

7. **Implementar.**

   Cuando un proyecto está totalmente depurado y se ha generado una versión *release*, podemos implantar los archivos generados en un servidor Web en producción.

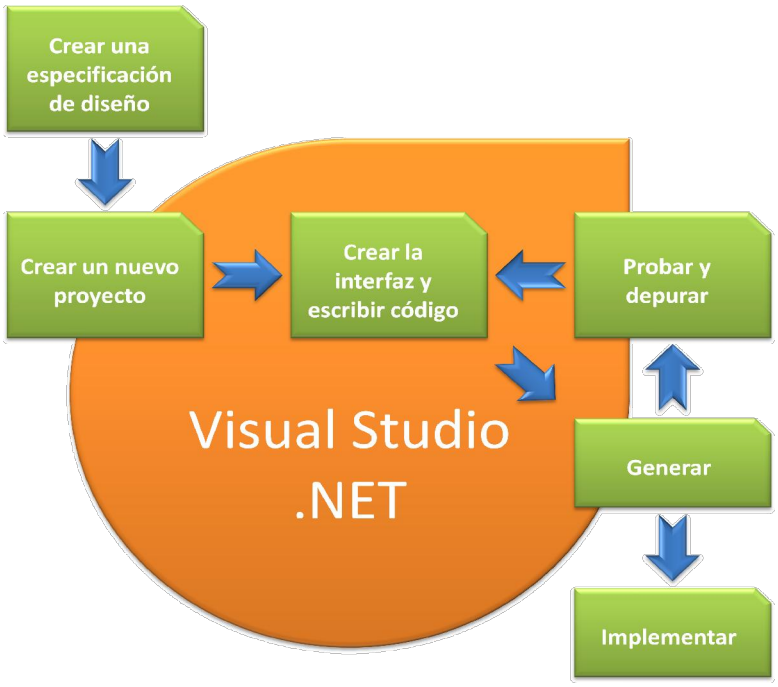

Figura 33. Crear un proyecto de aplicación Web con Visual Studio .NET

Archivos en aplicaciones Web

Cuando creamos un nuevo proyecto o trabajamos con proyectos existentes, Visual Studio .NET crea un determinado conjunto de archivos que soportan nuestro desarrollo.

Al crear un nuevo proyecto, también se crea una solución, aunque la solución únicamente tenga un proyecto. Se crea una carpeta para cada solución en la carpeta **\Mis documentos\Visual Studio Projects** que contiene los archivos .sln y .suo.

- **Archivos de solución (.sln)**

  La extensión de archivo NombreSolución.sln se utiliza para archivos de solución que enlazan uno o más proyectos, y almacena información global. Los archivos .sln son similares a los archivos de grupo Visual Basic (.vbg), que aparecen en versiones anteriores de Visual Basic.

- **Solution User Options (.suo)**

  La extensión de archivo NombreSolución.suo se utiliza para archivos que acompañan los registros de solución y las personalizaciones que agreguemos a nuestra solución. Este archivo guarda nuestra configuración, como puntos de interrupción y elementos de tareas, para que sean recuperados cada vez que abramos la solución.

Cada proyecto es una única aplicación Web almacenada en su propia carpeta. Dentro de la carpeta de proyecto se encuentra el archivo de configuración del proyecto y los archivos reales que constituyen el proyecto. El archivo de configuración del proyecto es un documento XML que contiene referencias a todos los elementos del proyecto, como formularios y clases, además de referencias de proyecto y opciones de compilación.

Los archivos de proyecto Visual Basic .NET utilizan una extensión .vbproj. Estas extensiones permiten diferenciar entre archivos escritos en otros lenguajes compatibles con .NET y facilitan la inclusión de múltiples proyectos basados en diferentes lenguajes en la misma solución.

Los proyectos de aplicaciones Web se crean en una nueva carpeta **\Inetpub\wwwroot**. Además, en IIS, se crea un directorio virtual que apunta a la carpeta del proyecto.

Visual Studio .NET soporta varios tipos de archivos de aplicaciones y extensiones:

- **Formularios Web Forms ASP.NET (.aspx)**

  Los formularios Web Forms ASP.NET se utilizan cuando es necesario generar sitios Web dinámicos a los que los usuarios accederán directamente. Los formularios Web Forms ASP.NET pueden estar soportados por una página de código subyacente diseñada por la extensión WebForm.aspx.vb.

- **Servicios Web ASP.NET (.asmx)**

Los servicios Web se utilizan cuando deseamos crear sitios Web dinámicos a los que únicamente accederán otros programas.

Los servicios Web ASP.NET pueden estar soportados por una página de código subyacente designada por la extensión WebService.asmx.vb o WebService.asmx.vb.

- **Clases y páginas de código subyacente (.vb)**

  Las versiones anteriores de Visual Basic utilizaban diferentes extensiones de archivo para distinguir entre clases (.cls), formularios (.frm), módulos (.bas), y controles de usuario (.ctl). Visual Basic .NET permite mezclar múltiples tipos en un único archivo .vb.

  Las páginas de código subyacente utilizan dos extensiones: el tipo de página (.aspx o .asmx) y la extensión de Visual Basic (.vb). Por ejemplo, el nombre de archivo completo para la página de código subyacente de un formulario Web Form ASP.NET predeterminado es WebForm1.aspx.vb para un proyecto Visual Basic .NET.

- **Archivos de descubrimiento (.disco y .vsdisco)**

  Los archivos de descubrimiento son archivos basados en XML que contienen enlaces (URLs) a recursos que proporcionan información para el descubrimiento programático de un servicio Web XML.

- **Clases de aplicación global (global.asax)**

  El archivo Global.asax, también conocido como el archivo de la aplicación ASP.NET, es un archivo opcional que contiene código para responder a eventos a nivel de aplicación provocados por ASP.NET o por HttpModules. En tiempo de ejecución, se parsea Global.asax y se compila en una clase .NET Framework generada dinámicamente y derivada de la clase base HttpApplication.

- **Archivos de recursos (.resx)**

  Un recurso es cualquier dato no ejecutable implantado lógicamente con una aplicación. Un recurso puede mostrarse en una aplicación como un mensaje de error o como parte de la IU. Los recursos pueden contener datos de diversos tipos, incluyendo cadenas, imágenes y objetos persistentes. Almacenar los datos en un archivo de recursos permite modificarlos sin necesidad de recompilar toda la aplicación.

- **Styles.css**

  Styles.css es el archivo de hojas de estilo predeterminado para la aplicación Web.

- **Archivo Web.config**

  Este archivo Web.config contiene opciones de configuración que el CLR lee, como políticas de enlace de ensamblados, objetos remoting, etc., y otras configuraciones que la aplicación puede leer. Los archivos Web.config también contienen las clases de aplicación global soportadas por un proyecto.

Los archivos que no estén basados en un lenguaje de programación tendrán sus propias extensiones. Por ejemplo, un archivo Crystal Reports utiliza la extensión .rpt y un archivo de texto utiliza .txt.

Cuando se compila un proyecto Web, se crean dos tipos adicionales de archivos:

- **Archivos ensamblados del proyecto (.DLL)**

  Todas las páginas de código subyacente (.aspx.vb y .aspx.cs) de un proyecto están compiladas en un único archivo ensamblado que se almacena como ProjectName.dll. Este archivo ensamblado del proyecto se ubica en el directorio /bin del sitio Web.

- **AssemblyInfo.vb o AssemblyInfo.cs**

  El archivo AssemblyInfo se utiliza para describir la información general del ensamblado, como la versión y los atributos del mismo.

Estructura de los archivos de una aplicación Web

Cuando creamos una aplicación Web ASP.NET, Visual Studio .NET crea dos carpetas para almacenar los archivos que soportan esa aplicación. Cuando compilamos un proyecto, se crea una tercera carpeta para almacenar el archivo .DLL resultante.

Visual Studio .NET crea una carpeta para la solución, Proyecto_A, que contiene el archivo Proyecto_A.sln. Este archivo es un mapa de los diversos archivos que soportan el proyecto.

También podemos crear una solución en blanco y agregarle proyectos. Si creamos una solución en blanco, tendremos una solución con un nombre distinto al del proyecto.

Visual Studio .NET también crea una carpeta denominada Proyecto_A, en la carpeta Inetpub\wwwroot, que contiene los archivos que requiere la aplicación Web. Estos archivos incluyen:

- El archivo de proyecto, Proyecto_A.vbproj, que es un documento XML que contiene referencias a todos los elementos del proyecto, como formularios y clases, además de referencias de proyecto y opciones de compilación.

- Formularios ASP.NET Web Forms, WebForm1.aspx, o servicios Web XML, WebService.asmx.

- Páginas de código subyacente, WebForm1.aspx.vb, WebService1.asmx.vb.

- Un archivo Web.config, que contiene las opciones de configuración de la aplicación Web.

- Un archivo Global.asax que gestiona los eventos que son invocados mientras la aplicación Web se está ejecutando.

Cuando generamos un proyecto de aplicación Web, Visual Studio .NET crea un ensamblado en la carpeta Inetpub\wwwroot\Proyecto_A\bin. Un ensamblado es un archivo .DLL que se crea desde todas las páginas de código subyacente que constituyen una aplicación Web.

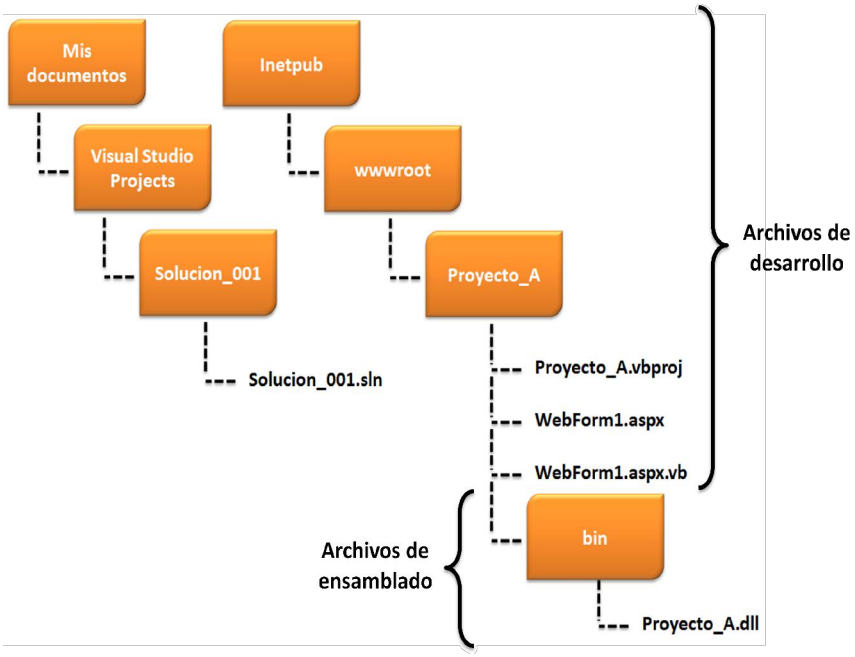

Figura 34. Estructura de los archivos de una aplicación Web

Capitulo V

Crear formularios Web Forms con Visual Studio .NET

Los formularios Web Forms son páginas Web programables que sirven como interfaz de usuario (IU) para un proyecto de aplicación Web utilizando ASP .NET. Un formulario Web Form presenta información al usuario visualizable en cualquier tipo de navegador, e implementa lógica de aplicación utilizando código ejecutable en el servidor.

¿Qué es un formulario Web Form?

Los formularios Web Forms están formados por una combinación de HTML, código y controles que se ejecutan en un servidor Web ejecutando Microsoft Internet Information Server (IIS). Los formularios Web Forms muestran una interfaz de usuario que genera HTML y que se envía al navegador, mientras que el código de soporte y los controles que la componen permanecen en el servidor Web. Esta división entre el interfaz en el cliente y el código en el servidor es una importante diferencia entre los formularios Web Forms y las páginas Web tradicionales. Mientras una página Web tradicional requiere que todo el código se envíe y se procese en el navegador, los formularios Web Forms únicamente necesitan enviar al navegador los controles de la interfaz, y el proceso de las páginas se mantiene en el servidor. Esta división entre IU y código aumenta el número de navegadores soportados e incrementa la seguridad y funcionalidad de la página Web.

Los formularios Web Forms se denominan habitualmente páginas ASP.NET o páginas ASPX. Los formularios Web Forms tienen una extensión .aspx y funcionan como contenedores para el texto y los controles que deseamos mostrar en el navegador.

Las páginas ASP.NET (.aspx) y Active Server Pages (ASP) (.asp) pueden coexistir en el mismo servidor. La extensión del archivo determina si la página la procesa ASP o ASP.NET.

Los formularios Web Forms están frecuentemente formados por dos archivos distintos: el archivo .aspx contiene la IU para el formulario Web Form, mientras que el archivo .aspx.vb, denominado página de código subyacente, contiene el código de soporte.

Las funciones de un formulario Web Form están definidas por tres niveles de atributos. Los atributos de página definen las funciones globales, los atributos de cuerpo definen cómo se mostrará una página y los atributos de formulario definen cómo se procesarán los grupos de controles.

La etiqueta <@ Page> define atributos específicos de la página que utiliza el parseador de páginas ASP.NET y el compilador. Únicamente podemos incluir una etiqueta <@ Page> por archivo .aspx. Los siguientes ejemplos son etiquetas <@ Page> típicas para Microsoft Visual Basic® .NET:

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="WebApplication1.WebForm1"%>
```

Los atributos de una etiqueta <@ Page> incluyen:

- **Language.**

  El atributo Language define el lenguaje en el que está escrito el script de la página Web. Algunos de los valores de este atributo son: vb, c# y JScript.

- **Codebehind.**

  El atributo de página Codebehind identifica la página de código subyacente que tiene la lógica que soporta el formulario Web Form. Cuando Visual Studio .NET crea un formulario Web Form, como WebForm1.aspx, también crea una página de código subyacente, WebForm1.aspx.vb.

- **SmartNavigation.**

  El atributo SmartNavigation de ASP.NET permite al navegador actualizar únicamente las secciones del formulario que han cambiado. Las ventajas de SmartNavigation son que la pantalla no parpadea mientras se actualiza; en lugar de ello, se mantiene la posición de desplazamiento y se mantiene la "última página" en el historial.

SmartNavigation únicamente está disponible para los usuarios con Microsoft Internet Explorer 5 o superior.

Los atributos de la etiqueta <body> definen el aspecto de los objetos que se muestran en el navegador del cliente. La siguiente es una etiqueta <body> típica:

```
<body MS_POSITIONING="GridLayout">
```

Los atributos de una etiqueta <body> incluyen:

- **PageLayout.**

  El atributo pageLayout (etiquetado como MS_POSITIONING) determina cómo se posicionan los controles y el texto en la página. Existen dos opciones para pageLayout:

  - ✓ **FlowLayout.**

    En FlowLayout, el texto, las imágenes y los controles fluyen por la pantalla, dependiendo del ancho de la ventana del navegador.

  - ✓ **GridLayout.**

    En GridLayout, los campos de texto, las imágenes y los controles de una página están fijados por coordinadas absolutas. GridLayout es el valor de pageLayout predeterminado para Visual Studio .NET.

La etiqueta <form> define cómo se procesarán los grupos de controles. La etiqueta <form> es diferente del término Web Form utilizado para definir la página Web completa. Los atributos de la etiqueta <form> definen cómo se procesarán los controles. Aunque podemos tener muchos formularios HTML en una página, únicamente podemos tener un formulario del lado del servidor en una página .aspx.

El siguiente ejemplo es de una etiqueta <form> típica:

```
<form id="Form1" method="post" runat="server">
...
</form>
```

Los atributos de una etiqueta <form> incluyen:

- **Method.**

  El atributo Method identifica el método para enviar valores de control de retorno al servidor. Las opciones de este atributo son:

  - ✓ **Post.**

    Los datos se pasan en pares nombre/valor dentro del cuerpo de la petición HTTP (Hypertext Transfer Protocol).

  - ✓ **Get.**

    Los datos se pasan en una cadena de consulta.

- **Runat.**

  Una de las principales características de un formulario Web Form es que los controles se ejecutan en el servidor. El atributo runat="server" hace que el formulario publique información de control de retorno a la página ASP.NET en el servidor donde se ejecuta el código de soporte. Si el atributo runat no está establecido en "server", el formulario funciona como un formulario HTML normal.

El siguiente código procede del formulario Web Form predeterminado de Visual Studio .NET al crear un nuevo proyecto de aplicación Web ASP.NET con Visual Basic .NET:

```
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="WebApplication1.WebForm1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<html>
  <head>
    <title>WebForm1</title>
    <meta name="GENERATOR" content="Microsoft Visual Studio
    .NET 7.1">
    <meta name="CODE_LANGUAGE" content="Visual Basic .NET
    7.1">
    <meta name=vs_defaultClientScript content="JavaScript">
    <meta name=vs_targetSchema
    content="http://schemas.microsoft.com/intellisense/ie5">
  </head>
  <body MS_POSITIONING="GridLayout">

    <form id="Form1" method="post" runat="server">

    </form>

  </body>
</html>
```

Crear un formulario Web Form con Visual Studio .NET

Dependiendo del punto en que nos encontremos en el proceso de desarrollo, existen varios modos de crear un formulario Web Form.

Cuando creamos un nuevo proyecto en Visual Studio .NET, se incluye automáticamente en el proyecto un formulario Web Form predeterminado denominado WebForm1.aspx.

- **Crear un nuevo proyecto de aplicación Web ASP.NET y un formulario Web Form predeterminado.**

    1. En Visual Studio .NET, en la Página de inicio, hacer clic en ***Nuevo proyecto***.

2. En el cuadro de diálogo ***Nuevo proyecto***, hacer clic en ***Aplicación Web ASP.NET***, escribir el nombre del proyecto en el campo ***Ubicación*** y hacer clic en ***Aceptar***.

Visual Studio .NET crea una nueva aplicación Web y un formulario Web Form predeterminado denominado WebForm1.aspx.

Si estamos expandiendo un proyecto existente, podemos utilizar el Explorador de soluciones para agregar rápidamente formularios Web Forms adicionales.

- **Agregar formularios Web Forms adicionales a un proyecto de aplicación Web.**

1. En la ventana del Explorador de soluciones, hacer clic con el botón derecho en el nombre del proyecto, seleccionar ***Agregar***, y hacer clic en ***Agregar Web Forms***. Se abrirá el cuadro de diálogo ***Agregar nuevo elemento*** - *NombreProyecto*.

2. En el cuadro de diálogo ***Agregar nuevo elemento*** - *NombreProyecto*, cambiar el nombre del formulario Web Form, y hacer clic en ***Abrir***.

Se creará un nuevo formulario Web Form y se agregará al proyecto.

Si estamos revisando un sitio Web existente, podemos importar páginas HTML a Visual Studio .NET y actualizar esas páginas a formularios Web Forms.

- **Actualizar páginas HTML existentes.**

1. En el Explorador de soluciones, hacer clic con el botón derecho en el nombre del proyecto, seleccionar ***Agregar*** y hacer clic en ***Agregar elemento existente***.

2. En el cuadro de diálogo ***Agregar elemento existente***, navegar hasta la ubicación del archivo HTML, hacer clic en el nombre del archivo y posteriormente en ***Abrir***.

3. Cambiar el nombre del archivo *nombreArchivo.htm* por *nombreArchivo.aspx*, y hacer clic en ***Sí*** a la pregunta de si estamos seguros de desear cambiar la extensión del archivo.

4. Cuando se nos pregunte si deseamos crear un nuevo archivo de clase, hacer clic en ***Sí***.

Uso de controles de servidor

Los controles de servidor ASP.NET son componentes que se ejecutan en el servidor y encapsulan la IU y demás funcionalidades relacionadas. Los controles de servidor se utilizan en páginas ASP.NET y en las clases de código subyacente. Los controles de servidor incluyen botones, cuadros de texto y listas desplegables.

El siguiente ejemplo es el de un control de servidor `Button`:

```
<asp:Button id="Button1" runat="server" Text="Submit" />
```

Los controles de servidor tienen un atributo runat="server", el mismo atributo que los formularios Web Forms. Esto significa que la lógica del control se ejecuta en el servidor y no en el navegador del usuario. Los controles de servidor se diferencian de los controles HTML en que éstos últimos únicamente se ejecutan en el navegador del cliente y no realizan ninguna acción en el servidor.

Otra característica de los controles de servidor es que el estado de la vista, las opciones de configuración y la entrada de datos de usuario en el control se guardan automáticamente cuando la página viaja entre el cliente y el servidor. Los controles HTML tradicionales no tienen estado y vuelven a su

configuración predeterminada cuando la página retorna del servidor al cliente.

La funcionalidad de un control es lo que se produce cuando el usuario hace clic en un botón o en un cuadro de lista. A estos procesos se denominan procedimientos de eventos. Como programadores del formulario Web Form, debemos determinar los procedimientos de eventos asociados a cada control de servidor.

En ASP.NET, los controles de servidor se basan en un modelo de objetos común, y por tanto, comparten varios atributos entre sí.

Cuando un navegador interpreta una página, los controles de servidor Web determinan el tipo de navegador que solicita la página, y envía el código HTML adecuado.

Existen numerosos tipos de controles de servidor disponibles en ASP.NET. Algunos controles de servidor se parecen mucho a los controles HTML tradicionales, mientras que otros son nuevos en ASP.NET. Esta amplia variedad de controles nos permite personalizar nuestro formulario Web Form para que se adapte a la aplicación que estamos creando.

Por defecto, el servidor no tiene disponibles los elementos HTML de una página de un formulario Web Form; los elementos HTML son tratados como texto opaco que pasa a través del navegador. Sin embargo, al agregar el atributo runat="server" se convierten los elementos HTML en controles de servidor HTML, exponiéndolos por tanto como elementos que podemos programar con código del lado del servidor.

Los controles de servidor Web no sólo incluyen controles de tipo formulario, como botones y cuadros de texto, sino también controles con funcionalidad especial, como el control calendario. Los controles de servidor Web son más abstractos que los controles de servidor HTML, porque su modelo de objetos no refleja necesariamente la sintaxis HTML.

Los controles de servidor Web se clasifican como:

- **Controles intrínsecos.**

  Los controles intrínsecos concuerdan con los sencillos elementos HTML, como botones o cajas de listas. Utilizamos estos controles del mismo modo que utilizamos los controles de servidor HTML.

- **Controles de validación.**

  Los controles de validación incorporan lógica que permite verificar la entrada de datos de un usuario. Para probar la entrada de un usuario, adjuntamos un control de validación al control de entrada y especificamos las condiciones de entrada de datos de usuario correctas.

- **Controles ricos estándar.**

  Los controles estándar son controles complejos que incluyen múltiples funciones. Ejemplos de controles estándar incluyen el control AdRotator, que se utiliza para mostrar una secuencia de anuncios o el control Calendar, que proporciona un calendario de citas.

- **Controles enlazados a listas.**

  Los controles enlazados a listas pueden mostrar listas de datos en una página ASP.NET. Estos controles nos permiten mostrar, reformatear, clasificar y editar datos.

- **Controles Web de Internet Explorer.**

  Los controles Web de Internet Explorer son un conjunto de controles complejos, como MultiPage, TabStrip, Toolbar y TreeView, que pueden descargarse desde Internet e integrarse en el entorno de Visual Studio .NET para ser reutilizados en cualquier aplicación Web con ASP.NET. Estos controles pueden ser interpretados en todos los navegadores utilizados habitualmente, y al mismo tiempo aprovechan las potentes

características soportadas por las versiones de Internet Explorer 5.5 o superior.

Controles de servidor HTML

Los controles HTML de un formulario Web Form no están disponibles en el servidor. Si convertimos los controles HTML en controles de servidor HTML, podemos exponerlos como elementos a nuestro código del lado del servidor. Esta conversión nos permite utilizar los controles para disparar eventos que son gestionados en el servidor.

Los controles de servidor HTML incluyen el atributo `runat="server"`, y deben residir en una etiqueta contenedora `<form …runat="server">…</form>`.

La ventaja de los controles de servidor HTML es que nos permiten actualizar rápidamente páginas existentes a formularios Web Forms. Además, podemos optimizar el rendimiento de una página ajustando qué controles deben funcionar localmente en el navegador y qué controles se procesan en el servidor.

Controles de servidor Web

Los controles de servidor Web son controles de servidor creados específicamente para ASP.NET. A diferencia de los controles de servidor HTML, los controles de servidor Web no funcionarán si falta el atributo `runat="server"`.

Los controles de servidor Web se basan en un modelo de objetos común; por ello, todos los controles comparten varios atributos, incluyendo la etiqueta `<asp:TipoControl...>`, y un atributo id. Los controles de servidor Web existen en el espacio de nombres `System.Web.UI.WebControls` y pueden utilizarse en cualquier formulario Web Form.

Capitulo VI

Agregar código a un formulario Web Form con ASP .NET

Podemos agregar código a nuestro formulario Web Form de los siguientes modos:

- **Código mezclado**.

  El código se encuentra en el mismo archivo que el contenido Web, entremezclado con el contenido Hypertext Markup Language (HTML). Este método es el menos elegido, ya que es difícil leer y trabajar con un archivo de este tipo. Sin embargo, es un método utilizado con frecuencia en páginas Active Server Pages (ASP).

- **Código en línea**.

  El código se encuentra en el mismo archivo en una sección **SCRIPT** distinta, al igual que el contenido HTML.

- **Código subyacente**.

  El código se encuentra en un archivo distinto del contenido HTML. El archivo de código se denomina página de código subyacente. Cuando se utiliza Visual Studio .NET, el método predeterminado es ubicar todo el código en una página de código subyacente.

Aunque el método predeterminado para implementar código en el lado del servidor en Visual Studio .NET es utilizar una página de código subyacente, podemos encontrar páginas que utilizan código en línea, concretamente páginas ASP.

Cuando se utiliza código en línea en una página Web, el HTML y el código se encuentran en secciones distintas de un único archivo .aspx. Esta separación se produce para ofrecer claridad en la lectura de la página; la funcionalidad, el código y HTML pueden coexistir en cualquier lugar de la página.

El siguiente código es un ejemplo de código en línea:

```
<HTML>
```

```
<asp:Button id="btn" runat="server"/>
...
</HTML>
<SCRIPT Language="vb" runat="server">
Sub btn_Click(s As Object, e As EventArgs) _
Handles btn.Click
...
End Sub
</SCRIPT>
```

¿Qué son las páginas de código subyacente?

El método predeterminado para implementar código en el lado del servidor en Visual Studio .NET es utilizar páginas de código subyacente. Cuando utilizamos páginas de código subyacente, la lógica de programación se encuentra en un archivo distinto de los elementos visuales de la página.

Separar la lógica del diseño permite a los desarrolladores trabajar en la página de código subyacente mientras los diseñadores de la interfaz de usuario (IU) trabajan en la página ASP.NET.

Las páginas de código subyacente contienen toda la lógica de programación para una sola página Web. Cada página de una aplicación Web tiene su propia página de código subyacente. De forma predeterminada, una página de código subyacente tiene el mismo nombre que la página Web a la que está asociada; sin embargo, la página de código subyacente también tiene una extensión .aspx.vb.

Por ejemplo la página Web Form1.aspx tendrá una página de código subyacente Microsoft Visual Basic® .NET denominada Form1.aspx.vb

Cómo funcionan las páginas de código subyacente

Para que las páginas de código subyacente funcionen, cada página .aspx debe estar asociada a una página de código subyacente, y esa página de código

subyacente debe estar compilada antes de que la información se envíe de vuelta a un navegador cliente que la solicite.

Aunque cada página Web Form está formada por dos archivos distintos (la página .aspx y la página de código subyacente), ambos archivos forman una única unidad cuando se ejecuta la aplicación Web. La página de código subyacente puede ser precompilada por Visual Studio .NET cuando generamos el proyecto de aplicación Web, o puede ser compilada just-in-time (JIT) la primera vez que un usuario accede a la página.

La página .aspx debe estar asociada a la página de código subyacente. Visual Studio .NET agrega los tres atributos siguientes a la directiva `@ Page` de la página .aspx para conseguir esta asociación:

- **Codebehind**.

  Es el atributo que Visual Studio .NET utiliza internamente para asociar los archivos.

- **Src**.

  Este atributo es el nombre de la página de código subyacente, y se utiliza si la aplicación Web no está precompilada.

- **Inherits**.

  Este atributo permite a la página .aspx heredar clases y objetos de la página de código subyacente.

El siguiente código muestra un ejemplo de directiva `@ Page` para un archivo denominado Page1.aspx:

```
<%@ Page Language="vb" Inherits="Project.WebForm1"
Codebehind="Page1.aspx.vb" Src="Page1.aspx.vb" %>
```

Cuando una página está compilada JIT, las páginas de código subyacente se compilan la primera vez que un cliente solicita la página .aspx. Tras la primera petición, las siguientes utilizan el archivo compilado existente. Por tanto, la primera petición de una página dura más, pero las siguientes peticiones son más rápidas.

Si se desea utilizar la compilación JIT para una página, debería utilizarse el atributo **Src** de la directiva `@ Page`.

Cuando un usuario solicita la página .aspx, el archivo DLL procesa la petición entrante y responde creando el código HTML y el scripting adecuados y devolviéndolos al navegador solicitante.

Si omitimos el atributo **Src** de la directiva `@ Page` en un archivo .aspx, la página se precompila cuando generamos la aplicación en Visual Studio .NET. De modo predeterminado, Visual Studio .NET no agrega el atributo **Src**; por ello, todas las páginas de código subyacente en los formularios Web Forms de un proyecto se compilan cuando se genera el proyecto. Este proceso ahorra un tiempo considerable de proceso en el servidor Web.

Precompilar páginas de código subyacente también simplifica la implantación del sitio Web ya que no es necesario implantar las páginas de código subyacente junto con las páginas .aspx.

Procedimientos de evento a controles de servidor Web

Los formularios Web Forms dinámicos e interactivos normalmente reaccionan a la entrada de datos del usuario. Los procedimientos de evento se utilizan para gestionar las interacciones de los usuarios en un formulario Web Form.

Cuando un usuario interactúa con un formulario Web Form, se genera un evento. Diseñamos nuestra aplicación Web para realizar una determinada tarea cuando se genera el evento. Un procedimiento de evento es la acción que ocurre en respuesta al evento generado.

Muchos formularios Web Forms permiten al usuario introducir información y hacer clic en un botón **Enviar**. Se genera un evento cuando el usuario hace clic en el botón **Enviar**. Por ejemplo, un procedimiento de evento podría ser enviar la información del usuario a una base de datos Microsoft SQL Server™.

Procedimientos de evento en el lado del cliente

Existen dos tipos de procedimientos de evento: en el lado del cliente y en el lado del servidor. Ambos tienen ventajas e inconvenientes.

Los procedimientos de evento en el lado del cliente son eventos gestionados en el equipo que solicita el formulario Web Form (el cliente). Cuando se genera un evento, no se envía ninguna información al servidor. En lugar de ello, el navegador del cliente interpreta el código y también realiza la acción.

Los procedimientos de evento en el lado del cliente únicamente pueden utilizarse con controles HTML. Además, los procedimientos de evento en el lado del cliente nunca tienen acceso a los recursos del servidor. Por ejemplo, no podemos utilizar scripts en el lado del cliente para acceder a una base de datos SQL Server.

Los procedimientos de evento en el lado del cliente resultan útiles para eventos que deseamos que ocurran inmediatamente porque no requieren un viaje de ida y vuelta al servidor Web (envío de información al servidor Web y espera de una respuesta). Por ejemplo, es posible validar información en un cuadro de texto antes de enviarla al servidor. Podemos utilizar scripts en el lado del cliente para validar la información rápida y eficazmente antes de enviar la información del usuario al servidor Web para continuar su proceso.

Especificamos un procedimiento de evento en el lado del cliente creando un bloque `<SCRIPT>` en la página Web, como muestra el siguiente código:

```
<SCRIPT language="javascript">
```

Procedimientos de evento en el lado del servidor

A diferencia de los procedimientos de evento en el lado del cliente, los procedimientos de evento en el lado del servidor requieren el envío de información al servidor Web para su proceso. Aunque el uso de procedimientos de evento en el lado del servidor supone un coste en tiempo, son mucho más potentes que los procedimientos de evento en el lado del cliente.

Los procedimientos de evento en el lado del servidor están formados por código compilado que reside en el servidor Web. Los procedimientos de evento en el lado del servidor pueden utilizarse para gestionar eventos que son generados desde los controles de servidor Web y HTML.

Los procedimientos de evento en el lado del servidor tienen acceso a recursos del servidor que normalmente no están disponibles para los procedimientos de evento en el lado del cliente.

Para especificar un procedimiento de evento en el lado del servidor, se utiliza el atributo `runat="server"` en la etiqueta `script`, como muestra el siguiente código:

```
<SCRIPT language="vb" runat="server">
```

Debido a que los procedimientos de evento en el lado del servidor requieren un viaje de ida y vuelta al servidor Web, existen un número limitado de tipos de eventos de control soportados.

Con los procedimientos de evento en el lado del cliente, podemos incluir código para procesar eventos asociados a botones de ratón y eventos `onChange`. Mientras los procedimientos de evento en el lado del servidor soportan eventos `click` y una versión especial del evento `onChange`, no pueden soportar eventos que ocurren frecuentemente, como eventos asociados a botones del ratón.

Crear procedimientos de evento

Crear un procedimiento de evento en el lado del servidor en Visual Studio .NET implica dos pasos. En el primer paso, creamos el control que genera el evento en el formulario Web Form. En el segundo, proporcionamos el código necesario en la página de código subyacente para procesar el evento.

Cuando hacemos doble clic en un control en Visual Studio .NET, Visual Studio .NET declara una variable (con el mismo nombre que el atributo `id` del control) y crea una plantilla para el procedimiento de evento. Cuando utilizamos Visual Basic .NET, Visual Studio .NET también agrega la palabra clave `Handles`, que adjunta el procedimiento de evento al control. La palabra clave `Handles` permite crear múltiples procedimientos de evento a un único evento.

El siguiente código HTML muestra un formulario Web Form que tiene un único botón con un atributo id igual a cmd1; el evento clic del botón se gestionará en el servidor:

```
<form id="form1" method="post" runat="server">
<asp:Button id="cmd1" runat="server"/>
</form>
```

El siguiente código Visual Basic .NET muestra la declaración de variables necesaria en la página de código subyacente.

```
Protected WithEvents cmd1 As _
System.Web.UI.WebControls.Button
```

En el código anterior, el nombre de la variable debe coincidir con el `id` del control Web, y debemos utilizar la palabra clave `WithEvents` para indicar que este control hace que se ejecuten los procedimientos de evento.

En el siguiente código Visual Basic .NET, que muestra el procedimiento de evento para el evento `Click`, la palabra clave `Handles` indica que el procedimiento de evento se ejecuta en respuesta al evento `Click` del control `cmd1`:

```
Private Sub cmd1_Click(ByVal s As System.Object, _
ByVal e As System.EventArgs) _
Handles cmd1.Click
...
End Sub
```

Todos los eventos pasan dos argumentos al procedimiento de evento: el remitente del evento y una instancia de la clase que guarda los datos del evento.

Normalmente, este último es del tipo `EventArgs`, y a menudo no contiene ninguna información adicional; sin embargo, para algunos controles, es de un tipo que es específico para ese control.

Por ejemplo, para un control Web `ImageButton`, el segundo argumento es de tipo `ImageClickEventArgs`, que incluye información sobre las coordenadas donde el usuario ha hecho clic. El siguiente procedimiento de evento envía las coordenadas de la ubicación donde ocurre un clic en un control `Label`:

```
Sub img_OnClick(ByVal s As System.Object, _
ByVal e As System.Web.UI.ImageClickEventArgs) _
Handles ImageButton1.Click
Label1.Text = e.X & ", " & e.Y
End Sub
```

Interactuar con controles en procedimientos de evento

En muchas aplicaciones Web, necesitamos leer y escribir a controles de un formulario. Podemos hacerlo dentro de los procedimientos de evento en el lado del servidor.

En un procedimiento de evento en el lado del servidor, podemos leer información de un control de servidor. Por ejemplo, si tenemos el siguiente formulario con un control `Textbox` y un control `Button`:

```
<FORM id="Form1" runat="server">
```

```
<asp:TextBox id="txtName" runat="server" />
<asp:Button id="cmd1" runat="server" />
</FORM>
```

Cuando el usuario hace clic en el botón, podemos leer el texto que el usuario ha escrito en el cuadro de texto. El siguiente código asigna la variable de cadena `strGreeting` a una concatenación del texto "`Hello`" y el texto del cuadro de texto `txtName`:

```
Dim strGreeting As String = "Hello " & txtName.Text
```

Podemos enviar información directamente a un control de servidor Web utilizando las propiedades del control. Por ejemplo, supongamos que tenemos un control de servidor Web denominado `Label` en la página ASP.NET, como sigue:

```
<asp:Label id="lblGreeting"
runat="server">Greeting</asp:Label>
```

El siguiente código en el lado del servidor asigna la propiedad `Text` del control de servidor Web `lblGreeting` a una cadena de texto:

```
lblGreeting.Text = "new text"
```

Ciclo de vida de los eventos de página

Cuando se solicita una página ASP.NET, se producen una serie de eventos de página. Estos eventos siempre ocurren en el mismo orden, denominado ciclo de vida de los eventos de página.

El ciclo de vida de los eventos de página consta de los siguientes eventos de página, que ocurren en el siguiente orden:

1. **Page_Init**.

Este evento de página inicializa la página creando e inicializando los controles de servidor Web de la página.

2. **Page_Load**.

   Este evento de página se ejecuta cada vez que se solicita la página.

3. Eventos **Control**.

   Este evento de página incluye eventos de cambio (por ejemplo, `Textbox1_Changed`) y eventos de acción (por ejemplo, `Button1_Click`).

4. **Page_Unload**.

   Este evento de página ocurre cuando la página se cierra o cuando el control pasa a otra página.

El final del ciclo de vida de los eventos de página incluye el borrado de la página en memoria.

La mayoría de eventos de control no ocurren hasta que el formulario Web Form se envíe de nuevo al servidor. Por ejemplo, los eventos `Change` se gestionan en orden aleatorio en el servidor después de que el formulario haya sido enviado. En cambio, los eventos `Click` pueden hacer que el formulario sea enviado al servidor inmediatamente.

Por ejemplo, si un usuario introduce texto en varios controles de un formulario y hace clic en un botón `Submit`, los eventos `Change` de los controles de texto no se procesarán hasta que el evento `Click` envíe el formulario al servidor.

En ASP .NET, los formularios están diseñados para enviar información de retorno a la página ASP.NET remitente para su procesamiento. Este proceso se denomina *postback*. Los postbacks pueden producirse por determinadas acciones del usuario. De modo predeterminado, únicamente los eventos

`click Button` hacen que el formulario sea enviado de nuevo al servidor. Sin embargo, si establecemos la propiedad `AutoPostBack` de un control a `true`, forzamos un postback para los eventos de ese control.

Por ejemplo, el siguiente código HTML es un ejemplo del uso de `AutoPostBack` en un cuadro de lista. Cada vez que el usuario modifica el valor del cuadro de lista, el evento `SelectedIndexChanged` se invoca en el servidor y actualiza el cuadro de texto:

```
<asp:DropDownList id="ListBox1" runat="server"
AutoPostBack="True">
<asp:ListItem>First Choice</asp:ListItem>
<asp:ListItem>Second Choice</asp:ListItem>
</asp:DropDownList>
El código de la página de código subyacente es como sigue:
Private Sub ListBox1_SelectedIndexChanged _
(ByVal s As System.Object, ByVal e As System.EventArgs) _
Handles ListBox1.SelectedIndexChanged
TextBox1.Text=ListBox1.SelectedItem.Value
End Sub
```

Gestión de los eventos Page.IsPostback

El evento `Page_Load` se ejecuta en cada petición de una página, tanto si es la primera petición de la página o un *postback*.

Debido a que el evento `Page_Load` se ejecuta con cada petición de una página, todo el código del evento `Page_Load` se ejecutará cada vez que la página sea solicitada. Sin embargo, cuando utilizamos eventos *postback*, es posible que no deseemos que se ejecute nuevamente todo el código. Si éste es el caso, podemos utilizar la propiedad `Page.IsPostBack` para controlar qué código se ejecuta únicamente cuando la página se solicita por primera vez, como muestra el siguiente código:

```
Private Sub Page_Load(ByVal s As System.Object, _
ByVal e As System.EventArgs) _
Handles MyBase.Load
If Not Page.IsPostBack Then
```

```
'ejecución sólo en la carga de página inicial
End If
'este código se ejecuta en cada petición
End Sub
```

Vínculo entre dos controles

Podemos vincular un control con el contenido de otro. Vincular es especialmente útil para mostrar valores de cuadros de lista o listas desplegables.

El siguiente código de ejemplo muestra cómo vincular un control `Label` al contenido de una lista desplegable. Utilizando las etiquetas de enlace `<%#` y `%>`, establecemos el atributo `Text` del control `Label` para el elemento seleccionado (`SelectedItem`) del cuadro de lista:

```
<asp:Label id="lblSelectedValue" runat="server"
Text="<%# lstOccupation.SelectedItem.Text %>" />
```

El siguiente ejemplo muestra el código de un formulario Web Form que se utiliza para vincular el control `Label` al cuadro de lista:

```
<form runat="server">
<asp:DropDownList id="lstOccupation"
autoPostBack="true" runat="server" >
<asp:ListItem>Program Manager</asp:ListItem>
<asp:ListItem>Tester</asp:ListItem>
<asp:ListItem>User Assistance</asp:ListItem>
</asp:DropDownList>
<p>You selected: <asp:Label id="lblSelectedValue"
Text="<%# lstOccupation.SelectedItem.Text %>"
runat="server" />
</p>
</form>
```

En el código anterior, la propiedad `AutoPostBack` de la lista desplegable está establecida a `True`, que provoca el *postback* automático cuando cambia el valor del cuadro de lista.

En el procedimiento de evento `Page_Load`, invocamos el método `DataBind` de toda la página o del control `Label`, como muestra el siguiente código:

```
Sub Page_Load (s As Object, e As EventArgs) _
Handles MyBase.Load
lblSelectedValue.DataBind()
End Sub
```