ENRIQUE GÓMEZ JIMÉNEZ

PROGRAMACIÓN AVANZADA EN WEB

Guía de estudio

Producción académica
y asesoría metodológica
Yendri Núñez Barrantes

© Enrique Gómez Jiménez
© Universidad Estatal a Distancia

Corrección de estilo
María Benavides González

Ilustraciones
Enrique Gómez Jiménez

Encargado de cátedra
Roy Aguilera Jinesta

Encargada de carrera
Erika Hernández Agüero

Esta guía de estudio ha sido confeccionada en la UNED, en el año 2019, para ser utilizada en la asignatura Programación Avanzada en Web, código 3101, que se imparte en la Licenciatura en Ingeniería Informática y Desarrollo de Aplicaciones Web, Escuela de Ciencias Exactas y Naturales.

Contenido

PRESENTACIÓN

Las empresas modernas requieren sistemas informáticos sólidos, rápidos y eficientes para llevar a cabo muchas de sus operaciones. Esas operaciones pueden ser transacciones rutinarias como pagos, compras, requisiciones, o complejas como la generación de escenarios de inversión o análisis de mercados. Disponer de sistemas inadecuados o, en el peor de los casos, no poseerlos, atentaría contra la competitividad de la empresa y la propia sobrevivencia en el mundo de los negocios. No es un secreto que las empresas de un mismo sector se vigilan entre sí para analizar sus prácticas comerciales y sus estrategias de negocios. Los sistemas de información coadyuvan a este tipo de análisis, arrojando resultados y proponiendo estrategias que contrarresten esas prácticas o las mejoren.

El desarrollo de *software* requiere de una formación adecuada en técnicas y herramientas que permita al profesional en esta disciplina crear soluciones de calidad y pertinencia para ayudar a la competitividad de las empresas actuales. Las técnicas para el desarrollo de software involucran procedimientos, reglas, normas o protocolos, entre otras, cuyo objetivo obtener los resultados exigidos por el cliente.

Las herramientas ayudan al desarrollador a crear los diseños, el código, las pruebas, la implementación y otras que generen el producto final.

La asignatura 03101, Programación Avanzada en Web, pretende guiarle en el aprendizaje de las técnicas y herramientas existentes para la creación de aplicativos web eficientes y modernos. Para lograr los objetivos propuestos, se cuenta con el libro de texto de la asignatura titulado *ASP.NET con C# en Visual Studio 2017, diseño y desarrollo de aplicaciones web*, del autor Brice-Arnaud Guérin. La presente guía le apoyará en el estudio de esta obra.

OBJETIVO GENERAL

Crear aplicaciones web, con capacidad de visualización en dispositivos móviles, que permitan la solución de problemas complejos de procesamiento de datos.

DESCRIPCIÓN DE LA GUÍA DE ESTUDIO

La guía de estudio para la asignatura Programación Avanzada en Web (03101) tiene como propósito orientarle en el avance de la temática. Cada tema está conformado por capítulos relacionados y lógicamente vinculados, los cuales se espera que coadyuven a la asimilación de los conocimientos necesarios para desarrollar las competencias requeridas para el desarrollo de aplicaciones web. A continuación, se resume el contenido de los temas de la asignatura.

Tema	Descripción
I. Generalidades de la plataforma para desarrollo de aplicaciones web	Se describen los procedimientos de instalación de una herramienta de desarrollo de aplicativos web, su funcionalidad, mecanismos de programación de código fuente, entre otras características.
II. Infraestructura de aplicaciones web y accesibilidad a	Se detalla la infraestructura sobre la cual se pueden crear aplicativos web: formularios web o mediante el patrón arquitectónico MVC. También se tratan los

Tema	Descripción
datos	servicios web, la accesibilidad a datos, entre otros temas.
III. Configuración, seguridad e implementación de aplicaciones web	Desarrolla el tema de la configuración y seguridad de un aplicativo web y su despliegue externo en un servidor web propio o de terceros.

Cada tema de la guía de estudio está estructurado de la siguiente manera:

- Sumario: enlista los conceptos que se abordarán en el tema.
- Objetivos específicos: son los que se espera que el o la estudiante logre al finalizar el estudio del tema.
- Introducción: ofrece un panorama general del estudio del tema.
- Guía de lectura: cuadro que indica las páginas del libro externo que deben estudiarse para el tema respectivo.
- Comentarios del tema: presenta las ideas principales desarrolladas en cada capítulo, complementadas con un esquema resumen del capítulo y recuadros con sitios de interés para ampliar el tema. Al final de cada capítulo, se plantean los ejercicios de autoevaluación.

Generalidades de la plataforma para desarrollo de aplicaciones web

I

Sumario

- ✓ Novedades de Visual Studio 2017
- ✓ C# 5 de un vistazo
- ✓ Las variantes de .NET
- ✓ El modelo de compilación
- ✓ El rol del servidor web

Objetivos específicos

Al finalizar el estudio de este tema, usted estará en capacidad de:

- Aplicar los procedimientos de instalación de una herramienta para el desarrollo de aplicativos web.
- Comprender la funcionalidad y administración de la interfaz del Integrated Development Environment (IDE) de una herramienta para el desarrollo de aplicaciones web.
- Elaborar el código fuente y los mecanismos de programación de una herramienta de creación de aplicaciones web que solucionen problemas informáticos.

Introducción

Este tema comprende los procedimientos de instalación de una herramienta informática que se puede utilizar para el desarrollo de aplicaciones web. Una vez instalada la herramienta, el estudiante debe comprender la funcionalidad del entorno de programación (Integrated Development Environment, IDE por sus siglas en inglés) y los mecanismos que ofrece para el desarrollo de código fuente y sus posibilidades de reutilización. También, se trata la tecnología de formularios web (Web Forms) para la creación de interfaces de usuarios, utilizando controles y estructuras de diseño gráfico.

Guía de lectura

Para el estudio de este tema, lea detalladamente las páginas del libro que se indican a continuación:

Capítulo	Páginas
Capítulo 1. Visual Studio 2017 y .NET	15-79
Capítulo 2. Los sitios web ASP.NET	81-118

Comentarios del tema

Capítulo 1. Visual Studio 2017 y .NET

1. Visual Studio 2017 y .NET

La nueva versión de Visual Studio 2017 incluye dos framework: el nativo y el nuevo .NET Core. El nuevo framework (.NET Core) comparte los mecanismos de ejecución del framework nativo, pero se diferencia por su modelo de aplicación ASP.NET. Este modelo se basa en una presentación de tres capas, permitiéndose desplegar en servidores no Windows.

Con Visual Studio 2017, se sobrepasa el marco de productividad individual, incluyendo la gestión del ciclo de vida del desarrollo del *software*. Se utiliza el espacio compartido de desarrollo en línea (Visual Studio Team Services, VSTS), el sistema de mantenimiento de código, según los protocolos Team Foundation Services (TFS), GitHub para el control de versiones, entre otras características.

En el libro de texto, páginas 15 a 17, se resumen algunas de las nuevas características de Visual Studio 2017.

1.1. Instalación

Para la utilización de Visual Studio 2017, se debe descargar el aplicativo y, posteriormente, instalar todos sus componentes. En esta versión, no se incluye SQL Server dado que la idea es que se instale de forma separada.

1.2. Interfaz del programa

Para las labores de programación, Visual Studio .NET presenta la interfaz que ha sido familiar durante las versiones anteriores. La **barra de menú de opciones** contiene todas las operaciones que es posible llevar a cabo con la herramienta, los menús de contexto, las acciones correspondientes a la operación seleccionada y la barra de iconos la funcionalidad que apoya, de forma rápida, las mismas opciones de la barra de menús. Lógicamente, si es la primera vez que se utiliza la herramienta, se deben estudiar las funcionalidades de cada operación o revisar las nuevas si ya es un estudiante avanzado.

En el libro de texto, páginas 18 a 39, se resumen las opciones del IDE de Visual Studio 2017, funcionalidades tales como codificación, depuración, pruebas unitarias, configuraciones, paqueterías NuGet, entre otras.

1.3. Gestión del código

Para la gestión del código, Visual Studio 2017 ofrece funcionalidades que permiten la facilidad de creación de programas. La figura 1.1. muestra las funcionalidades que presenta Visual Studio 2017 para que los desarrolladores puedan escribir código fuente con mayor facilidad de edición y reutilización.

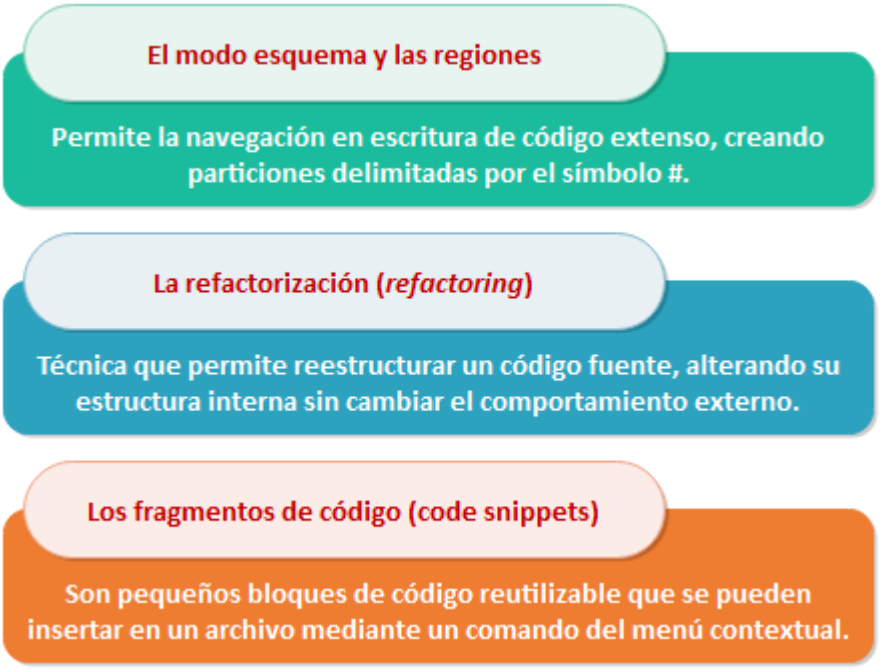


Figura 1.1. Herramientas de ayuda al programador de Visual Studio 2017

1.4. Documentación

Visual Studio 2017 permite documentar programas, extrayendo las anotaciones XML que el desarrollador escribe en los comentarios de su código fuente. Esto no es nuevo, puesto que ya existía en las versiones anteriores de Visual Studio.

1.5. Control de código fuente con Visual Studio Online

Mediante Visual Studio Online es posible establecer el control de código fuente generado. Es una herramienta gratuita que permite el control de código fuente para equipos de cinco personas. Es necesario registrarse mediante una cuenta de correo electrónico válida para descargar e instalar esta herramienta.

En el libro de texto, páginas 40 a 54, se explican los procesos de gestión del código generado en aplicaciones desarrolladas en Visual Studio 2017, así como su documentación y control de código fuente.

1.6. La herramienta MSBuild

MSBuild proporciona las herramientas esenciales para la compilación de aplicaciones administradas por el .NET Framework de Visual Studio 2017. Esta herramienta se incluía antes en el .NET Framework, pero actualmente está disponible para descarga independiente. Con MSBuild, es posible crear proyectos y soluciones de Visual Studio sin tenerlo instalado.

2. C# de un vistazo

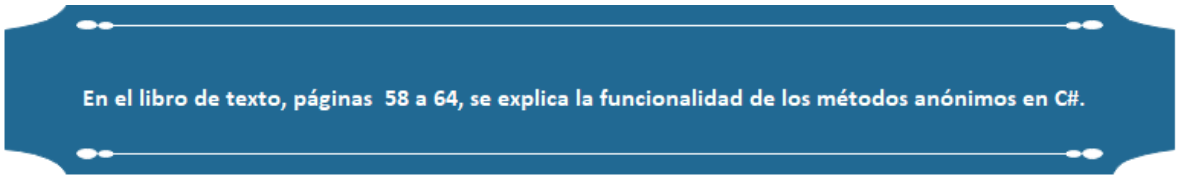
C# constituye uno de los lenguajes de programación más extendido y utilizado de la suite de Visual Studio .NET. A lo largo de la evolución de Visual Studio, C# ha ido creciendo exponencialmente hasta convertirse en el lenguaje predilecto de muchos desarrolladores.

2.1. Clases parciales

Las clases parciales en C# consisten en que su definición se pueda distribuir en más de un bloque de código. Esto brinda la ventaja de separar la definición de una clase en varios archivos. Por ejemplo, se podría crear el archivo *Program.cs* y el archivo *ProgramMethods.cs*, separando la programación de la clase. Si se trabaja en grandes proyectos de desarrollo, la distribución de una clase en archivos separados permite que varios programadores puedan utilizarla o modificarla al mismo tiempo.

2.2. Métodos anónimos

Un método anónimo carece de nombre y ofrece la posibilidad de pasar un bloque de código como un parámetro de un delegado. Como no tiene un nombre, el método anónimo se invoca mediante delegados, es decir, conjuga la encapsulación de un método por parte de un delegado.



2.3. Inferencia de tipo

Para este concepto, el libro de texto, en la página 65, lo define claramente como un mecanismo que el compilador utiliza para deducir el tipo de una expresión y asigne una variable que lo represente. Esto se trata, también, en LINQ, en el que las consultas pueden variar de una a otra, generando diversos resultados.

2.4. Las expresiones lambda

Una expresión lambda se utiliza para crear funciones anónimas que se asignan a un delegado o un árbol de expresión. Son anónimas porque no se le asigna un nombre para referirse a ellas dentro de un programa. Un delegado, por su parte, es un objeto que referencia a uno o más métodos. Estas expresiones lambda pueden utilizarse para escribir expresiones de consultas en LINQ, crear árboles de expresión, para pasar código que se ejecutará en métodos asincrónicos, entre otras funciones.

2.5. Clases dinámicas y tipos anónimos

Mientras que SQL aprovecha la indexación para generar consultas de datos, la programación de consultas en memoria, a través de clases se ve afectada por la reserva de espacio por cada atributo de esta, con datos o no.

Recordemos que una clase dinámica permite ejecutar propiedades y métodos a través de una de sus instancias. Los tipos anónimos, por su parte, proporcionan una manera conveniente de encapsular un conjunto de propiedades de solo lectura en un solo objeto sin tener que definir el tipo de primero.

2.6. Extensión de clases sin herencia

Generalmente, una extensión de clases puede utilizarse para múltiples propósitos, lo más común es la especialización mediante la cual la clase que se extiende define su propio comportamiento, convirtiéndose en una clase especializada de su

superclase. En C#, según el libro de texto, para definir una extensión de una clase sin derivarla es preciso crear un método estático en el mismo espacio de nombres donde se empleará. En las páginas 67 y 68 del libro de texto, se expone un caso concreto de implementación de este concepto.

2.7. Tipos *nullables*

Con Visual Studio 2017, se permite asignar nulos a las variables de tipo valor. Se pueden declarar tipos *nullables* utilizando *Nullable<t>*, siendo T un tipo. Los tipos "nullables" son instancias de la estructura *System.Nullable<T>*.

2.8. Iterador

Los iteradores se utilizan para enumerar los elementos de un conjunto de tipo tabla, colección, entre otros. Con ellos, se pueden recorrer estas estructuras mediante el bucle *foreach*. Es un método, un descriptor de acceso *get* o un operador que proporciona compatibilidad con *foreach* en una clase o estructura sin que se deba implementar la interfaz *IEnumerable*.

En el libro de texto, páginas 69 a 72, se detalla el concepto iterador, su evolución y su forma de uso.

2.9. Genericidad

Según el libro de texto, la genericidad es una herramienta de programación que

evita al programador tener que practicar secuencias de copiar-pegar, difíciles de mantener. Como base a todas las declarativas de clase, el tipo *object* ha constituido hasta ahora la forma como se enuncia una clase sin tipo definido.

En el libro de texto, páginas 72 a 76, se explica el concepto de la generidad, uso de tipos genéricos y la interpolación de tipos.

2.10. Las variantes de .NET

Existen dos variantes muy marcadas en las tecnologías de .NET. La figura 1.2. muestra estas dos variantes. La primera tecnología, .NET Core, está orientada a ejecutarse sobre múltiples plataformas (Windows, Linux, macOS, entre otros), creando aplicaciones basadas en iCloud (Azure, principalmente). La segunda, .NET Standard, son aplicaciones nativas de Microsoft® que se ejecutan en un entorno Windows, utilizando como servidor web el sistema Internet Information Services (IIS).

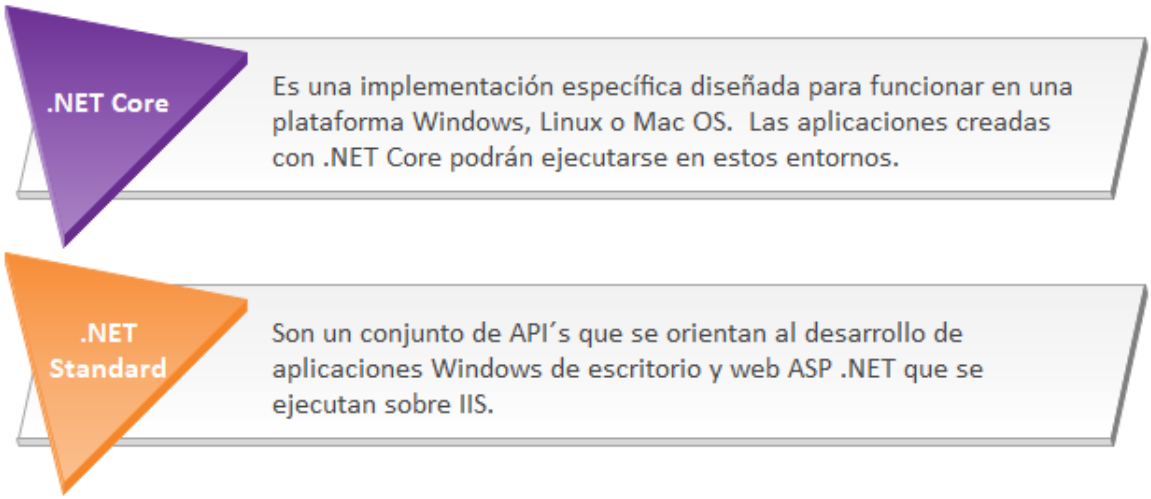


Figura 1.2. Variantes de .NET

Resumen del capítulo

Este capítulo describe las novedades de Visual Studio 2017. También, se tratan temas orientados al lenguaje de programación C# y las funcionalidades que ofrece al desarrollador de aplicaciones informáticas. No profundiza en muchos temas porque es un resumen o una citación de las propiedades de la plataforma, de las cuales ya muchas están implementadas en versiones anteriores.

Ejercicios de autoevaluación del capítulo 1

- Ejercicios complementarios:
  - ✓ Elabore un resumen acerca de las principales novedades de Visual Studio 2017 descritas en el libro de texto.
  - ✓ Explique resumidamente el proceso de Gestión de código que se lleva a cabo con Visual Studio 2017.

- ✓ Elabore una prueba codificada de interfaz de usuario que permita la grabación del proceso de entrada, proceso y salida para el cálculo del factorial de un numero (5!=120). Apóyese en la sección 1.2.5 del capítulo 1 del libro de texto.
- ✓ Defina, con sus propias palabras, los conceptos *clases parciales*, *métodos anónimos* y la *genericidad* en Visual Studio 2017.
- ✓ Explique brevemente la diferencia entre .NET Core y .NET Standard.

Capítulo 2. Los sitios web ASP.NET

1. El modelo de compilación

El modelo de compilación de ASP.NET se basa en el uso de múltiples librerías contenidas en el Framework que se provee con Visual Studio .NET. Entre estas librerías encontramos desde System.Web.UI hasta System.Data. También permite la compilación dinámica al construir carpetas de ejecución como App_Code, App-WebReferences, entre otros.

1.1. Del CGI al modelo ASP.NET 1.X

El modelo de compilación de ASP.NET evolucionó desde las técnicas Common Gateway Interface (CGI) hasta lo que es hoy día el motor ASP.NET que se encuentra dentro del .NET Framework de Visual Studio. Por su parte, CGI utilizaba *software* cliente para ejecutarse en la máquina del usuario y programas ejecutables en el servidor. ASP.NET posee diferentes capas y API que realizan tareas especializadas, como la que se dedica a la gestión de datos.

En el libro de texto, páginas 81 a 86, se detallan las técnicas de compilación de CGI y ASP .NET para aplicaciones de software desarrolladas por Visual Studio 2017.

1.2. Clases parciales para las páginas

Las clases ASPX funcionan aún mediante dos archivos separados: el que contiene el código C# y etiquetas <asp> y otro que almacena el código escrito por el

programador. Ambos archivos trabajan conjuntamente para dotar de la funcionalidad requerida al documento web.

En el libro de texto, páginas 87 a 89, se explica el funcionamiento de las clases parciales para documentos web en Visual Studio 2017.

1.3. El código compartido en *App_Code*

El *App_Code* contiene código fuente para todas aquellas clases que se comparten en el aplicativo web. Incluye, también, los archivos *.aspx* y *.cs* o *.vb* que se compilan con el aplicativo. Cada vez que hay cambios en el proyecto, ASP.NET recompila nuevamente esta carpeta para considerarlos.

1.4. Los ensamblados referenciados

La compilación de un proyecto de Visual Studio antes consideraba el archivo *.csproj*, sin embargo, el nuevo modelo solo lo hace para efectos de recompilación de la DLL que soporta el conjunto de clases del sitio web. En la actualidad, la descripción del proyecto se hace directamente en el archivo de solución. Algunos otros ensamblados de la solución tienen sus propias referencias, estas se pueden realizar dinámicamente o mediante el archivo web.config del proyecto.

1.5. Caché de construcción

El servidor de aplicaciones utiliza la carpeta llamada caché de construcción para almacenar las distintas versiones de las clases parciales y las DLL que representan el producto ejecutable de los sitios web ASP.Net.

1.6. Las aplicaciones web de Visual Studio

Visual Studio 2017 permite crear aplicaciones web que no están enlazadas a IIS, sino a un servidor auxiliar de la plataforma.

2. El rol del servidor web

El servidor web cumple el rol de ejecutar un aplicativo informático para que los usuarios lo puedan utilizar mediante un navegador web. Posee reglas de seguridad y accesibilidad que les permiten usarlo adecuadamente. Este servidor web recibe las peticiones de los usuarios (*request*), procesándolas y devolviendo una respuesta mediante un response.

2.1. El servidor IIS

IIS es un servidor web que posee, a su vez, un conjunto de servicios para administrar sitios y aplicaciones web en un sistema operativo Windows. En este servidor, se pueden publicar sitios y aplicaciones web, tanto de forma local como remota.

2.2. El servidor de desarrollo ASP.NET

Una alternativa a IIS, provista por Microsoft, es el servidor IIS Express. Es un servidor de desarrollo y de pruebas que no está previsto para la explotación del sitio web.

3. Pipeline HTTP de IIS

El pipeline HTTP de IIS es una máquina de estado que permite procesar diversos eventos como Begin_Request, Authenticate_Request, End_Request, entre otros. Este pipeline funciona desde que se recibe una solicitud del usuario, y se pasa por la máquina de estados hasta que finaliza la sección del cliente. El pipeline de IIS puede ejecutarse de dos modos: integrado y clásico (ISAPI); ambos tienen sus particularidades de ejecución.

3.1. Funcionamiento de IIS

IIS funciona a través de peticiones y respuesta entre un cliente (navegador) y el servidor. El cliente solicita algún recurso al servidor, éste evalúa la solicitud y devuelve una respuesta de acuerdo con el resultado de la evaluación. Toda esta comunicación se basa en el protocolo HTTP.

En el libro de texto, páginas 101 a 120, se explica el funcionamiento del protocolo HTTP en el manejo de sitios web ASP .NET.

3.2. Clase HttpContext

La clase HttpContext encapsula toda la información específica de HTTP sobre una solicitud individual HTTP. Expone la propiedad estática Current, la cual devuelve una gran cantidad de información que se relaciona con la petición actual.

En el libro de texto, páginas 105 a 106, se explica someramente el funcionamiento de la clase *HttpContext* y los principales objetos que la componen.

3.3. Clase HttpApplication

La clase HttpApplication permite definir los métodos, propiedades y eventos que son comunes para todos los objetos de una aplicación ASP.NET. Se considera la clase base para las aplicaciones que son definidas por el usuario en el archivo *Global.asax*. Cuando se ejecuta una consulta, la clase HttpApplication desencadena una serie de eventos que se gestionan mediante el archivo *Global.asax*.

En el libro de texto, páginas 106 a 113, se explica el funcionamiento de la clase *HttpApplication* y su vinculación al archivo *Global.asax*, en un aplicativo web ASP .NET. También destaca la creación de un módulo HTTP como derivación de *HttpApplication*.

3.4. Controladores (*handlers*) HTTP

Los controladores HTTP tienen la responsabilidad de responder a las peticiones que realice un usuario a través de un navegador. Por ejemplo, un controlador HTTP puede ser una instancia de System.Web. Page, la cual responde a una renderización de un documento HTML. ASP.NET cuenta con muchos controladores HTTP, adaptables a gran cantidad de situaciones que requieren una respuesta HTTP.

En el libro de texto, páginas 114 a 120, se explica el funcionamiento de los controladores (*handlers*) HTTP y su implementación como un controlador ASHX o en una DLL.

Resumen del capítulo

Este capítulo describe el modelo de compilación de un sitio web ASP.NET, los componentes generados de dicha compilación y su exposición en el aplicativo creado. También, se trata el funcionamiento del servidor IIS de Windows y el auxiliar IIS Express, de Visual Studio .NET. Finalmente, se desarrolla el tema de la clase HttpApplication, la cual desencadena una serie de eventos orientados al procesamiento del ciclo de vida de una aplicación web ASP.NET. En el archivo *Global.asax*, se describe una versión de esa clase.

Ejercicios de autoevaluación del capítulo 2

- Ejercicios complementarios:
  - ✓ Describa brevemente el modelo de compilación de ASP.NET.
  - ✓ Elabore una tabla comparativa donde muestre el rol que cumple el servidor IIS de Windows y el del servidor auxiliar IIS Express en un aplicativo web ASP.NET
  - ✓ Describa la funcionalidad de la clase HttpApplication en el procesamiento del ciclo de vida de una página ASP.NET.
  - ✓ Explique la funcionalidad del archivo Global.asax en un aplicativo web.

Infraestructura de aplicaciones web y accesibilidad a datos

II

Sumario

- Web Forms
- Los sitios web MVC
- ASP.NET Core
- El acceso a datos con ADO.NET
- Gestión del estado
- Los servicios web WCF y Rest

Objetivos específicos

Al finalizar el estudio de este tema usted, estará en capacidad de:

- Desarrollar aplicaciones web mediante formularios web (Web Forms) para la solución de problemas de procesamiento de datos.
- Desarrollar aplicaciones web mediante la utilización del patrón arquitectónico de diseño Model View Controller (MVC).
- Utilizar la tecnología ASP.NET Core para la creación de aplicaciones web, servicios web, aplicaciones IoT y backends móviles que se implementen en la nube o en servidores dedicados.
- Integrar la funcionalidad de la gestión de datos con ADO.NET en la implementación de un aplicativo web.
- Comprender la gestión del estado de una página web durante su ejecución en un navegador de internet.
- Integrar servicios web WCF en la implementación de aplicaciones web.
- Analizar el sistema Rest como protocolo de intercambio y manipulación de datos en servicios web.

Introducción

Este tema considera las arquitecturas, tecnologías y técnicas necesarias para la creación de aplicativos web que gestionen datos. Para la creación de los sitios y aplicaciones web, se utilizarán desde la conocida práctica de crear aplicaciones de internet empleando formularios web (*web forms*) hasta la más utilizada actualmente,

WCF. Se tratan los temas de ASP.NET Core, ADO.NET, WCF, Rest, entre otros que son importantes para la creación e implementación de aplicativos web robustos y seguros.

Guía de lectura

Para el estudio de este tema, lea detalladamente las páginas del libro que se indican a continuación:

Capítulo	Páginas
Capítulo 3. Los Web Forms	121-267
Capítulo 4. Los sitios web MVC	269-311
Capítulo 5. ASP.NET Core	315-329
Capítulo 6. El acceso a datos con ADO.NET	331-424
Capítulo 7. Gestión del estado	425-449
Capítulo 9. Los servicios web WCF y REST	511-528

Comentarios del tema

Capítulo 3. Los Web Forms

3. Presentación de los *Web Forms*

Los *Web Forms* son la técnica más conocida de Visual Studio para el diseño de aplicativos web. Realmente, se basan en un concepto de reparto de responsabilidades al estilo MVC (Model View Controller). En este caso, el HTML del formulario *.aspx* es la vista, la clase C# que gestiona los datos y los cálculos serían el modelo y el servidor de aplicaciones ASP.NET, el controlador.

3.1. Estructura de una página .ASPX

Desde el punto de vista de la compilación, una página *.aspx* está conformada por tres partes esenciales: las directivas Page, los DTD y el código XHTML relacionado. La figura 2.1. muestra esta conformación:

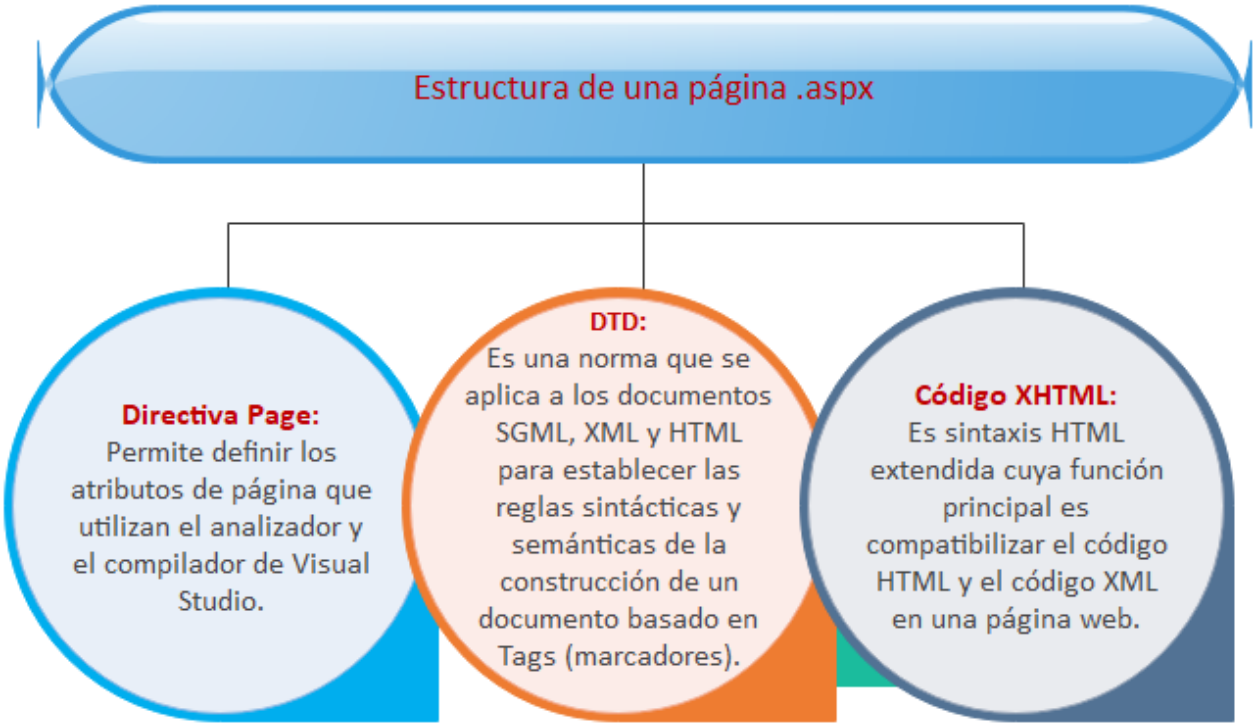


Figura 2.1. Estructura de una página .aspx

Las páginas dinámicas se pueden estilizar mediante HTML y CSS del lado del cliente e implementar la lógica del lado del servidor a través de C#.

También, del lado del cliente, el estilo de la presentación (el UI) puede ser delimitado por una secuencia de información y objetos en forma anidada, en línea o separada.

En el libro de texto, páginas 126 a 131, se detallan los tipos de organización de una página dinámica en ASP .NET y la sintaxis de los bloques de instrucciones que se pueden anidar dentro de una interfaz de cliente.

Un formulario web es un contenedor tanto de lógica de servidor (*code-behind*) como interfaces de usuario. Ambas facilidades conviven en un mismo archivo, separando, eso sí, la lógica que se implementa en el servidor como el diseño de la interfaz del usuario. En un formulario web, coexisten una serie de controles bajo una jerarquía bien definida y que se hereda desde System.Web.UI. Controls. Esta clase posee una cantidad importante de propiedades que permiten al desarrollador controlar su funcionamiento mientras se ejecuta.

Los formularios web poseen tres objetos importantes intrínsecos en su funcionalidad: *Request*, *Form* y *Response*. El objeto Request representa los parámetros enviados desde el navegador al servidor cuando se produce una solicitud tipo GET. Cuando la petición es de tipo POST Request, lo que hace es un postback, es decir, una devolución al servidor para realizar alguna tarea. El objeto Form, por su parte, es el contenedor de los objetos que pertenecen a un formulario y que el usuario utiliza. Finalmente, Response, se emplea para escribir una respuesta proveniente del servidor.

En el libro de texto, páginas 131 a 137, se explica la jerarquía de controles que existe en un formulario web y los objetos intrínsecos que se manejan en la ejecución de estos formularios: *Request*, *Form* y *Response*.

3.2. Ciclo de vida de una página

Una página web ASP.NET apoya su ciclo de vida en un proceso basado en eventos. Estos eventos tienen una secuencia definida que permite la interacción entre el usuario y el servidor donde se aloja el sitio web. La inicialización de la página se

realiza mediante el framework, el cual se encarga de crear los controles necesarios para usar dicha página. El programador también puede establecer eventos de inicialización del formulario a través del Load de la página. Todo este ciclo de vida de una página incluye gestión de eventos, liberación de recursos, identificación de peticiones, entre otros.

La figura 2.2. muestra el ciclo de vida de una página web ASP.NET, iniciando con la solicitud del usuario a través de una página ASP.NET, ejecutándose en un navegador. El servidor carga las DLL requeridas para el procesamiento de la solicitud, crea la página adecuada y la envía al usuario. En ese ciclo, ejecuta los métodos Application_Start, Session_Start y Page_Load. Una vez que el usuario abandona la sesión se destruye la página.

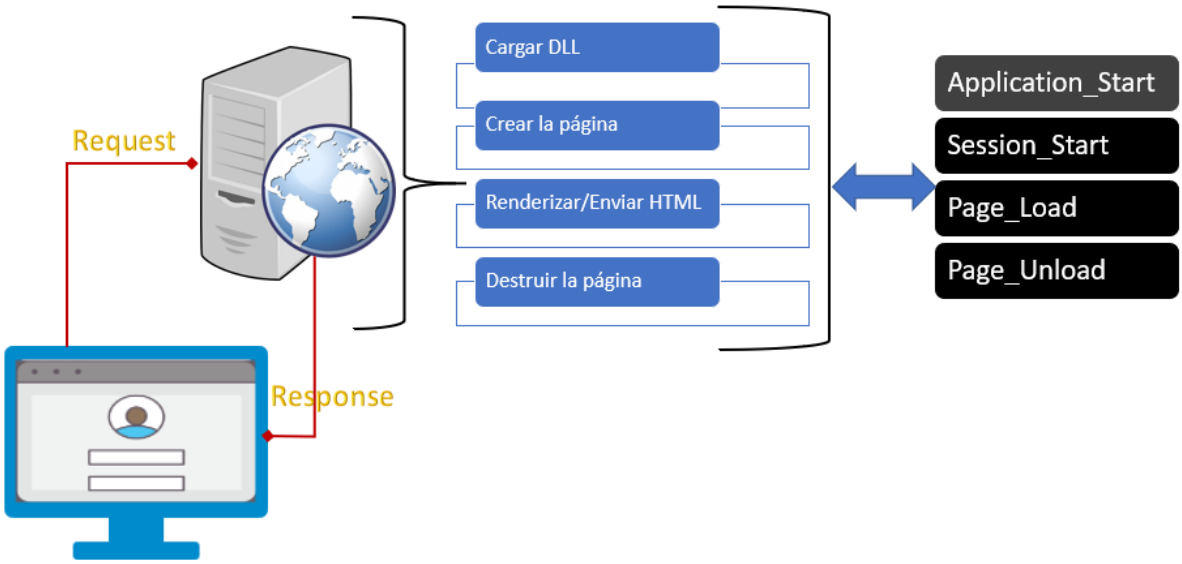


Figura 2.2. Ciclo de vida de una página web

3.3. Controles web

Microsoft® desarrolló su propio Document Object Model (DOM) para exponer sus

propios controles web en la interfaz de usuario. Mediante estos controles web, los usuarios pueden introducir valores o ver los resultados enviados por el servidor o producto de un cálculo. Estos coexisten con los controles HTML que también se incluyen en el framework. Los eventos que se integran a esos controles pueden ser gestionados mediante atributos, por ejemplo: <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Button"/>, o mediante código, el cual se inicializa en método OnInit() de la página.

En el libro de texto, páginas 141 a 151, se explican los controles web, las etiquetas y controles HTML que coexisten en una página ASPX, así como se gestionan los eventos que se implementan en ellos.

3.4. Navegación entre páginas

La navegación entre páginas ASPX se realiza fácilmente mediante enlaces de hipertexto. El control web HyperLink permite establecer la navegación entre páginas. Otra forma de hacerlo es utilizando la instrucción JavaScript *Window.location="dirección";*. Si la navegación se produce desde el lado del servidor, se emplean las instrucciones *Response.Redirect(URL)*, *Server.Execute(page)* o *Server.Transfer(page)*. La primera opción es la más aplicada.

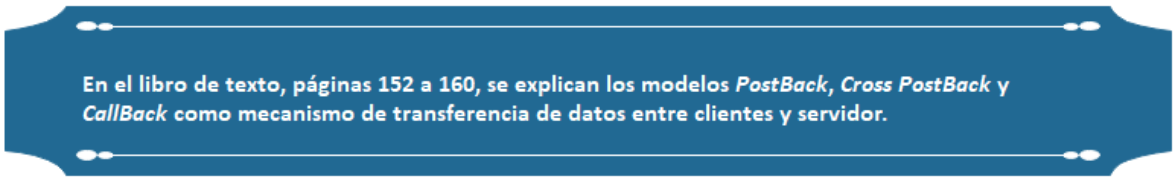
3.5. Postback y cross postback

Un postback es cuando una página web se está cargando en respuesta a un valor devuelto por el cliente o el servidor, o cuando se carga por primera vez y se puede acceder a esta. Es una acción producto de un evento (como el clic en un botón), por

lo cual viaja al servidor a realizar algún proceso. A veces, es necesario publicar los datos de un formulario web en otro; en estos casos, se utilizan variables de *session* para ir de un lado al otro. Sin embargo, puede utilizar un envío desde la página origen a otra página incluyendo el PostBackUrl e indicando hacia cuál página se está navegando. La página destino deberá comprobar la propiedad PreviousPage para saber si se ha invocado tras un cross postback.

3.6. Callback

Cuando utilizamos un postback, la información se envía al servidor y ahí se ejecutan los procesos requeridos. Los callback, por su parte, usan una función del lado del cliente que se encarga de enviar información al servidor donde se ejecutan los procesos requeridos. En este caso, se tiene que en el cliente hay un *event handler*, el cual invoca un evento definido en el cliente, enviando posteriormente la respuesta (por ejemplo, una validación) al servidor. Esto, lógicamente, mejora el desempeño del aplicativo web.



3.7. Validación de los datos introducidos por el usuario

Visual Studio 2017 incluye, en el suite de herramientas que posee, toda una gama de controles web de validación. Esto permite al desarrollador establecer controles que validan las entradas, formatos y salidas en controles de usuario en un

formulario web. Entre estos, RequiredFieldValidator, RangeValidator, ValidationSummary, entre otros.

En el libro de texto, páginas 161 a 172 se explican los controles de validación que ofrece Visual Studio 2017 y la forma en cómo se pueden implementar.

4. Organizar la presentación

Para la presentación de la interfaz de usuario que se mostrará al cliente web, Visual Studio 2017 cuenta con herramientas que facilitan el diseño. Entre estas, tenemos los temas y máscaras.

4.1. Temas y máscaras

Estas herramientas permiten controlar la apariencia de los elementos en una página ASPX, apoyándose en CSS. Un tema posee hojas de estilos, archivos .skin e imágenes que pueden ser utilizados en cualquier página ASPX. Los *skins* son un conjunto de propiedades textuales (colores, imágenes, estilos, entre otros), las cuales se aplican a los controles web que se utilizan en una página ASPX.

En el libro de texto, páginas 172 a 180, se explican los conceptos de máscaras y skins para el diseño de páginas ASP .NET y la forma en cómo se pueden crear.

4.2. Controles de usuario

Visual Studio 2017 ofrece la posibilidad de utilizar controles web de formulario para diseñar las interfaces de usuario; también, es posible crear controles web personalizados reutilizables en un aplicativo web. En realidad, un control de usuario es un tipo de control compuesto que funciona en formularios web ASP.NET Se pueden definir propiedades y métodos para el control y utilizar en una página ASP.NET como una instancia.

En el libro de texto, páginas 180 a 187, se define el concepto de controles de usuario, cómo diseñarlos (agregar propiedades, métodos y eventos) e incrustrarlos en una página web ASP .NET.

4.3. Las páginas maestras

Visual Studio 2017 permite la creación de páginas maestras en las que se puede estandarizar el diseño que tendrá la página principal del aplicativo web. Generalmente, en esta página se aloja el menú, los *sidebar*, información general de la empresa, entre otros elementos principales. También admite crear la página de contenido, la cual será la plantilla por utilizar por todos los documentos ASPX que serán, a su vez, las páginas internas del aplicativo web. Estas páginas de contenido albergan los formularios transaccionales o de vistas que tendrá el aplicativo.

En el libro de texto, páginas 188 a 196, se explica el concepto de página maestra y página de contenido y cómo diseñarlas para ser usadas en un aplicativo web ASP .NET.

5. Componentes personalizados

Los componentes personalizados, por su propósito, son diferentes a los controles de usuario. Un control personalizado puede, sin embargo, ser similar a un componente de usuario (.ascx) porque integra varios controles web. Los componentes personalizados es posible invocarlos desde varios proyectos web, dado que es un ensamblado .NET que se ubica en una librería de componentes.

En el libro de texto, páginas 196 a 210, se define el concepto de componentes personalizados y se describe el procedimiento para su creación y configuración. También, se explica cómo gestionar el *postback* del control, entre otras características técnicas.

3.3. Chart Control: componente gráfico que utiliza GDI+

El Chart Control es un componente que permite la creación de gráficos "ofimáticos". Se utiliza para generar y desplegar gráficos de distintos tipos. Los datos pueden ser obtenidos de una base de datos y generar gráficos en una página web ASP.NET

3.4. PictureBrowser, componente basado en una plantilla

El pictureBrowser se utiliza para la representación de imágenes de dos modos: miniatura o imagen por imagen. La lista de imágenes por mostrar se encuentra en una Data Table, la cual se puede inicializar mediante un archivo XML.

En el libro de texto, páginas 211 a 226, se explican los componentes *ChartControl* y *PictureBrowser*, utilizados para la generación de gráficos y la visualización de imágenes. En ambos casos, se explica su funcionamiento y la forma de crearlos.

6. AJAX

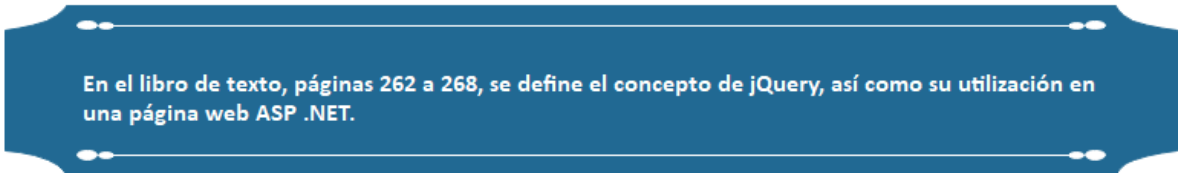
Se trató el tema de los callbacks en el punto 1.6., el cual es una solución técnica similar a AJAX. Microsoft® incorporó dos componentes muy importantes para el manejo de lógica de cliente y soporte de AJAX: el controlador de script y el panel de actualización. Se incorpora entonces un ScriptManager que se encarga de cargar las librerías de código JavaScript, requeridas para el funcionamiento de AJAX. También, incorpora UpdatePanel que posee la capacidad de realizar postback parcial, refrescando áreas del formulario que lo requieran. Este refrescamiento se produce gracias a la existencia de triggers, encargados de actualizar el contenido de los UpdatePanel. Hay otros componentes secundarios, los cuales son igualmente importantes, de AJAX: UpdateProgress y Timer.

En el libro de texto, páginas 228 a 262, se explica el concepto de AJAX, los componentes que lo integran y su íntima relación con JavaScript.

7. jQuery

jQuery es una biblioteca basada en JavaScript que permite la interacción con el árbol DOM (Document Object Model), manejar eventos, gestionar animaciones, agregar dinamismo con AJAX, entre otras funcionalidades. Es un software de código abierto que ahorra código comparado con el uso de JavaScript puro. Además, con el uso de jQuery, se logran grandes resultados en menos tiempo que si se usara el mismo JavaScript.

Para utilizar jQuery, se descarga de su sitio web (www.jquery.com). Así, el desarrollador tendrá la posibilidad de recorrer el árbol DOM de un archivo HTML y realizar operaciones sobre cada objeto que lo componga.



Resumen del capítulo

Este capítulo describe las principales características técnicas de Visual Studio 2017, las cuales permiten al desarrollador de aplicaciones web crear soluciones de alta complejidad. Se inicia con el tema de los formularios web (*web forms*) y todo lo que involucra su conformación sintáctica y semántica. Estos formularios web constituyen la denominada página ASPX, que cuenta con directivas Page, DTD (normas aplicables a documentos SGML, XML y HTML) y formato XHTML. Se continúa con el tema de las páginas dinámicas, donde converge el uso de HTML.

CSS, JavaScript, AJAX, jQuery, entre otros, como componentes ejecutables desde el lado del cliente. Por otro lado, se detallan los componentes de servidor (controles web con su propio DOM y controles de usuario) y toda la lógica inmersa.

Finalmente, se tratan los temas de presentación mediante temas y máscaras (skin) que utilizan muchas técnicas del lado del cliente. Cuestiones de diseño y estandarización por medio de páginas maestras (*page. master*), páginas de contenido (*page. content*) y las formas de navegación disponibles (*response.redirect*, *server.execute* y *server.transfer*).

Ejercicios de autoevaluación del capítulo 3

- Ejercicios complementarios:
  - ✓ Elabore un resumen acerca de la estructura de una página web.
  - ✓ ¿Qué significa que Microsoft® haya creado su propio árbol DOM (Document Object Model)?
  - ✓ Explique, en forma resumida, el ciclo de vida de una página ASP.NET.
  - ✓ Explique brevemente cuál es la funcionalidad de las páginas maestras y de contenido en la usabilidad de un aplicativo web ASP.NET.
  - ✓ Explique, brevemente, la importancia de los temas y las máscaras (skin) en el diseño de interfaces de un aplicativo web ASP.NET.

Capítulo 4. Los sitios web MVC

4. El enfoque MVC

Java revolucionó la forma como los sitios web se desenvolvían desde que separó las responsabilidades de los distintos niveles de un aplicativo. Esto lo consiguió mediante el diseño de framework como Struts o Spring Web MVC.

4.1. El patrón de diseño MVC

MVC es un patrón arquitectónico de diseño que permite separar las responsabilidades de un aplicativo web mediante modelos, controladores y vistas. Los modelos contienen toda la lógica de datos (estructura, comportamientos, validaciones, entre otros) que un aplicativo requiere para su funcionamiento. El controlador, por su parte, la navegación que se da entre las vistas, gestionando una comunicación con el modelo, el cual le facilita la información que les pasará a determinadas vistas. Finalmente, la vista, como interfaz del usuario, mostrará la información requerida por este y generará las entradas que el modelo necesita a través del controlador.

La figura 4.1. muestra la conformación de las capas MVC, se destaca el modelo que contiene bases de datos, clases, archivos, entre otros. Le sigue la capa del controlador que son clases tipo ActionResult, ContentResult, entre otros, que permiten la ejecución de métodos y renderización de vistas. Finalmente, las vistas, estas son archivos de interfaz que el usuario puede desplegar y utilizar en un navegador.

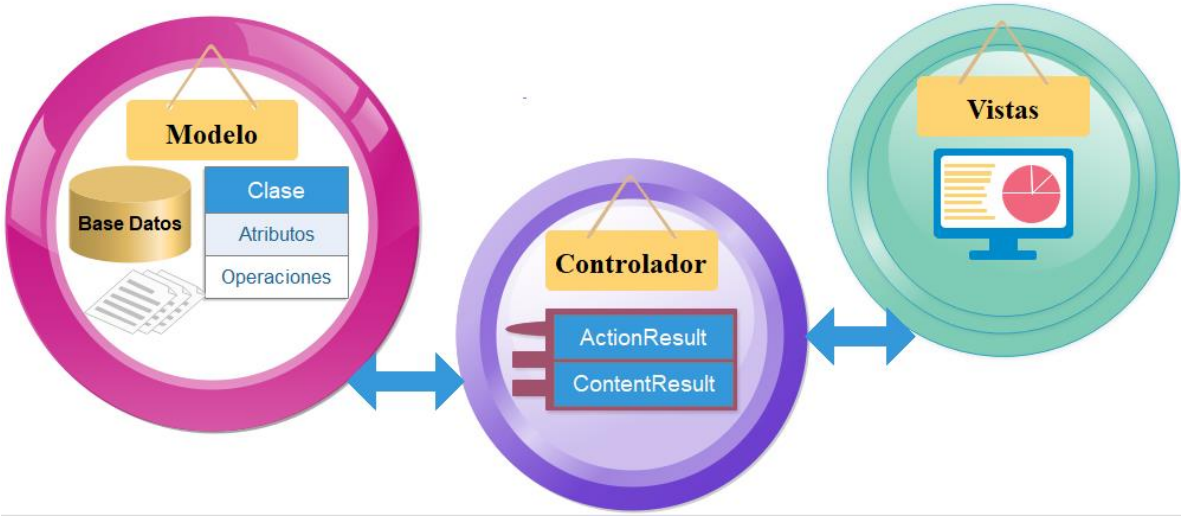


Figura 4.1. Capas de MVC

4.2. Evolución de MVC

MVC ha evolucionado a tal punto que la programación se ha vuelto más sencilla. Es posible, ahora, manejar varias acciones desde un mismo controlador, generando procesos, resultados y ejecutando vistas, según las necesidades del usuario.

5. Los sitios ASP.NET MVC

Los sitios MVC se crean de acuerdo con la arquitectura mostrada en la figura 4.1. Creado un sitio web ASP.NET MVC, el desarrollador iniciará la creación del modelo, el cual puede estar basado en clases, en ADO.NET Entity Framework, invocaciones a otros proyectos que contengan la lógica de datos, entre otras estrategias. Luego, establecerá los controladores que implementa la lógica aplicativa y, finalmente, las vistas que requieren ser ejecutadas por esos controladores.

En el libro de texto, páginas 271 a 291, se explica, mediante la creación de un sitio web ASP .NET MVC, las clases que se indican en la figura 4.1.

5. Motor de vistas Razor y las vistas

Razor permite la creación de páginas web mediante el uso de su propia sintaxis. Este motor de presentación se ve enriquecido con el uso de HTML, CSS, jQuery, Bootstrap, entre otros, para crear interfaces interactivas muy agradables para el usuario.

5.1. Sintaxis C# en las vistas CSHTML

Razor incluye, dentro de su sintaxis, la codificación mediante C# (también Visual Basic .NET), conjugándolo con sintaxis propias basada en el uso de etiquetas *scriplets*. Por ejemplo, en Razor es posible reconocer instrucciones C# tales como *@RenderBody()* o *@DateTime.Now*.

5.2. Estructura y organización de las vistas

Las vistas en MVC se organizan de tal manera que la modularidad sea un punto alto para la simplicidad y la facilidad de mantenimiento. Por un lado, se diseña un layout (cuyo objetivo es similar a las páginas maestras de Web Form) y, por otro, las vistas parciales (que son las páginas de contenido de la interfaz). La figura 4.2. muestra la conformación de las vistas en MVC.

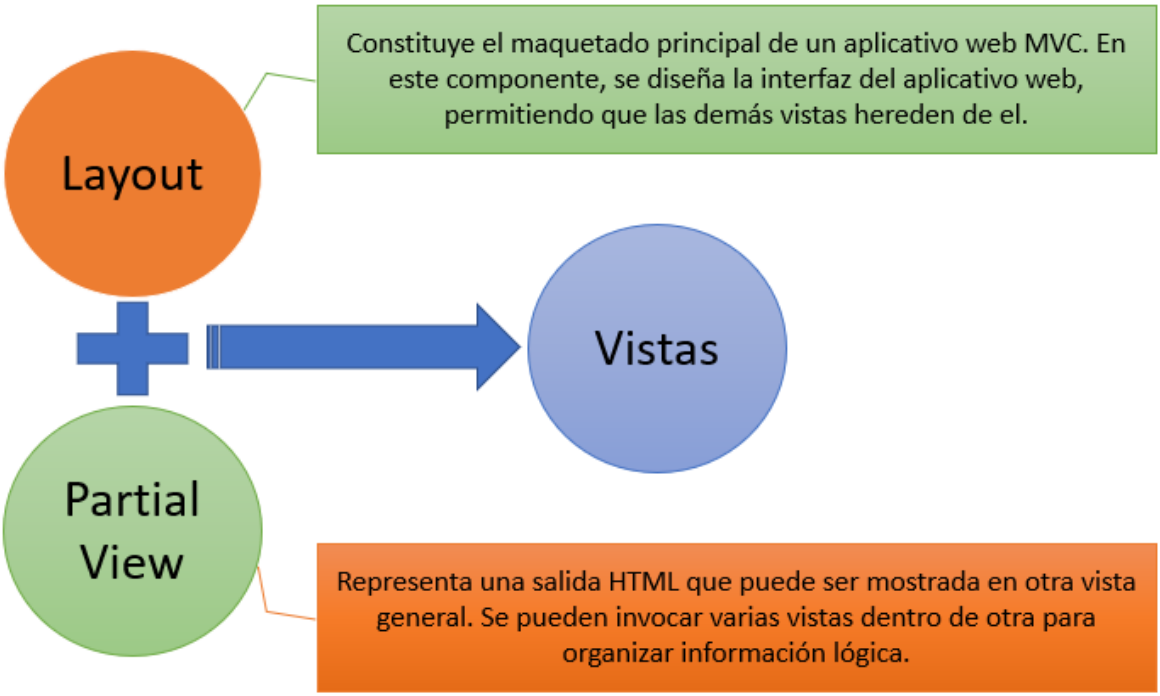


Figura 4.2. Vistas en MVC

Dentro de las vistas, es posible utilizar scripts y CSS para mejorar el diseño de la interfaz. Para ello, se puede emplear el concepto de *bundles*. Estos bundles son un mecanismo MVC para incluir archivos (scripts o CSS) que se relacionan entre sí, mediante bibliotecas. Es decir, se insertan dentro del BundleConfig del archivo *startup.cs* para usarlos en la vista posteriormente.

6. Securización de los sitios MVC

Una vez creado el sitio web MVC, Visual Studio permite configurar el proyecto para que utilice distintos modos de autenticación. Por medio del área *Account* (controlador, vistas, modelos), el desarrollador puede establecer la administración

de los usuarios del aplicativo; mediante *Startup.ConfigureAuth()* es posible definir las estrategias de seguridad para el aplicativo web.

Otra forma de securizar el aplicativo web es mediante la autorización. Esta se lleva a cabo utilizando el atributo Authorize, el cual limita el acceso a los usuarios a redirigirse a una página de conexión si son anónimos.

7. Definir áreas

Las áreas se utilizan, en un aplicativo web MVC, para agrupar en carpetas todos aquellos elementos relativos a un módulo funcional (controladores, vistas y modelos). Cada área agrupa todos los componentes y objetos que requiere el aplicativo web, desde un punto funcional y lógico: la carpeta *View* contendrá las vistas de la solución; la carpeta *Models,* las clases, referencias a bases de datos, archivos y otros que corresponden a la lógica de procesamiento de datos; y la carpeta *Controllers,* los controladores que interactuarán entre las vistas y el modelo.

8. Las aplicaciones Single Page Applications (SPA)

Una aplicación de una sola página (Single Page Application, SPA) es un aplicativo web que reescribe dinámicamente la página del cliente sin tener que hacerlo desde el servidor; esto lo hace con la ayuda de JavaScript. Con ello, se mantiene la experiencia del usuario, puesto que se conserva la interacción con el contenido que este ve; tal comportamiento se asemeja a un aplicativo de escritorio.

En el libro de texto, páginas 306 a 313, se explica el concepto de SPA (*Single Page Application*) y las ventajas que ofrece para el dinamismo de una página web. También, se resume el uso de *KnockOut* para enlazar datos y ser un complemento útil a SPA.

Resumen del capítulo

Este capítulo describe el modelo arquitectónico MVC para el desarrollo de aplicaciones web. Se consideran las tres capas básicas del modelo: modelo (para implementar la lógica de datos), el controlador (que establece la lógica de ejecución de las vistas y su relación con el modelo de datos) y las vistas (que implementan las interfaces del usuario). Se trata el tema de las Single Page Applications y su dinamismo en la interacción del usuario.

Se puede resumir la lógica funcional de MVC en un ejemplo concreto. Supóngase que se requiere mantenimiento de departamentos de una empresa. Los pasos básicos para establecer la lógica de este requerimiento (CRUD: Create, Read, Update, Delete) es el siguiente:

a. Creamos la base de datos y una tabla en ella que se denomine Departamentos.

b. En un proyecto MVC de Visual Studio .NET, creamos la clase que se vincula con la tabla departamentos (véase en la figura 4.2 la clase Departamento), Esta clase estará en una carpeta de Entidades y pertenece al Model del aplicativo.

c. Creamos un método que establezca la lógica del CRUD para nuestro aplicativo. Este método también pertenece al Model del aplicativo.

d. A continuación, creamos un controlador que se vinculará con el modelo por medio de un método. Luego, procederá a ejecutar una vista donde mostrará los datos que se han ejecutado a través de un ActionResult. Este controlador estará en la carpeta Controllers.

e. Finalmente, se crea la vista que llama el controlador para mostrar los datos obtenidos del modelo.

La figura 4.3. muestra la lógica funcional de la ejecución del CRUD para Departamentos, utilizando MVC.

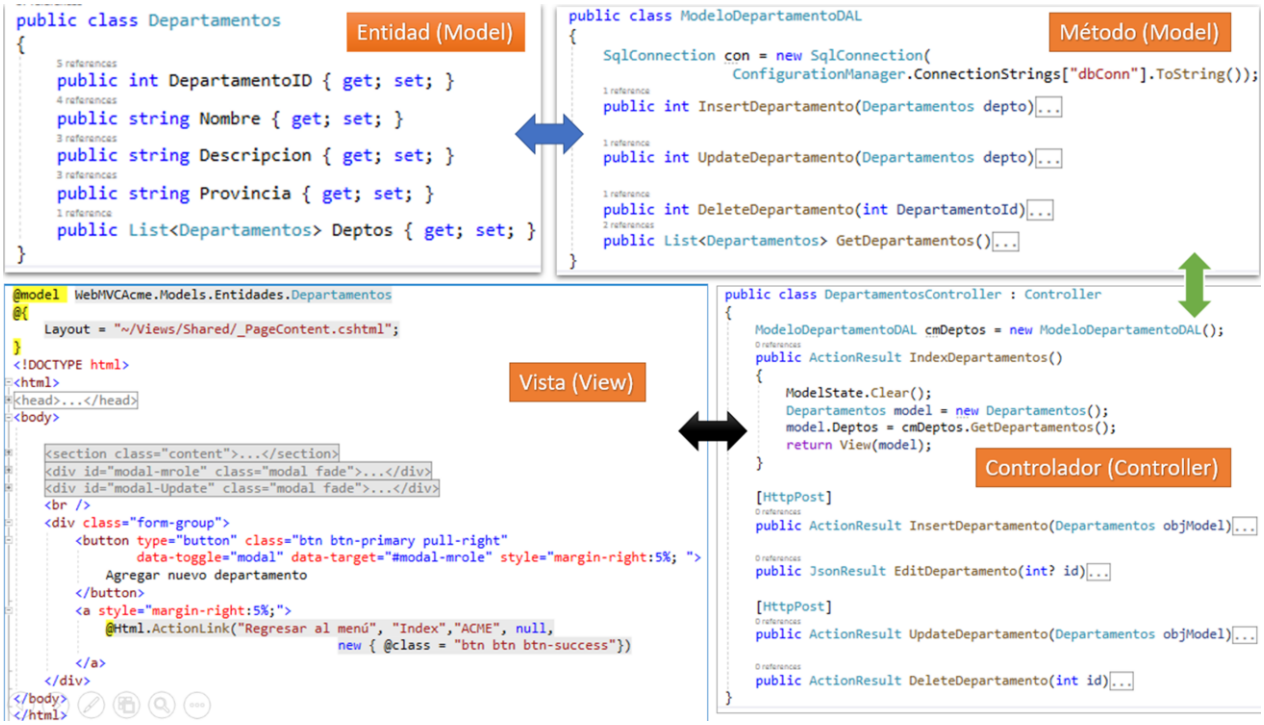


Figura 4.3. Lógica funcional de una ejecución MVC

La ejecución de los componentes mostrados en la figura 4.2. podría ser similar a la figura 4.3.

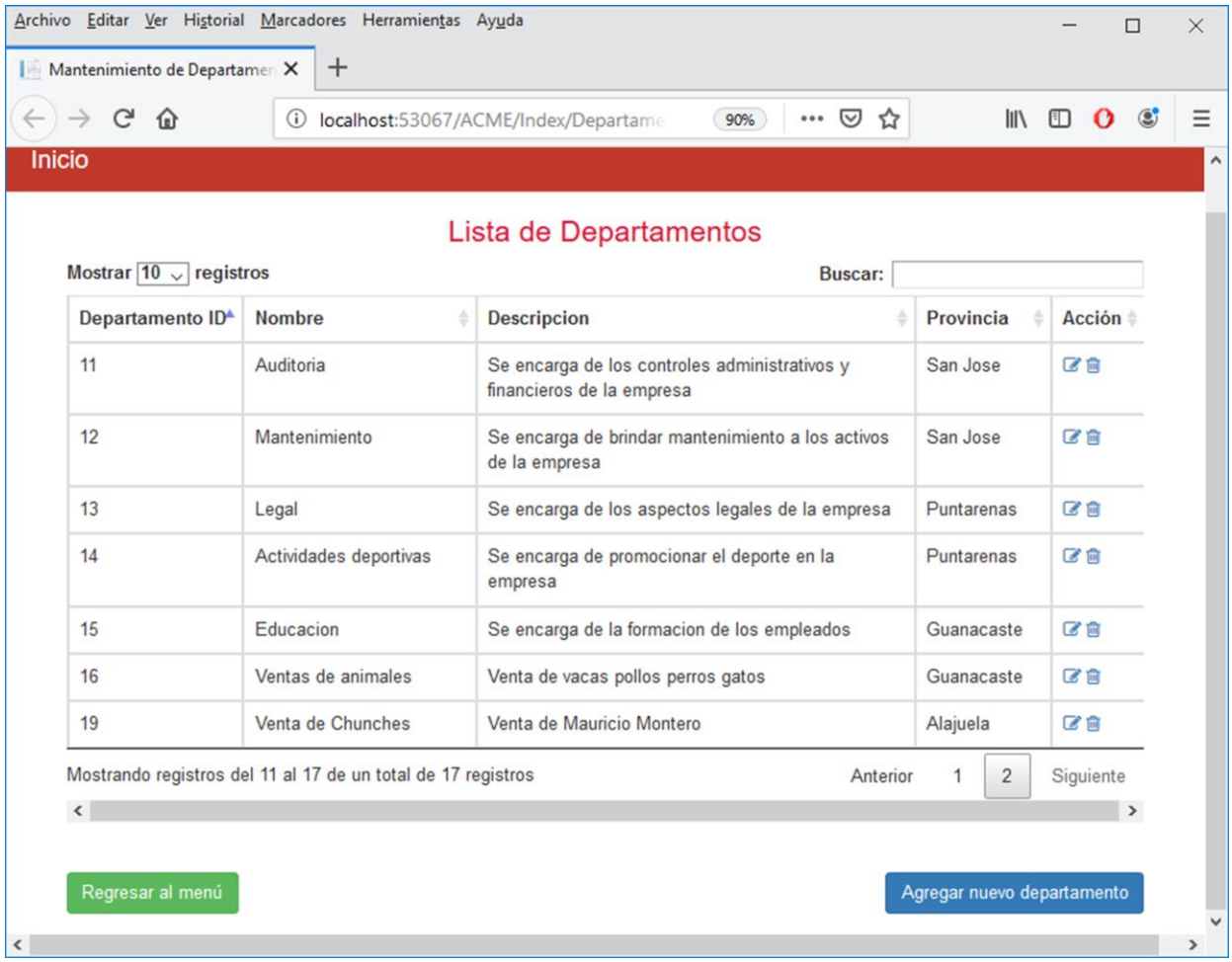


Figura 4.4. Ejecución de los componentes mostrados en la figura 4.2

Ejercicios de autoevaluación del capítulo 4

- Ejercicios complementarios:
  - ✓ Describa, brevemente, los componentes del patrón arquitectónico MVC.
  - ✓ Explique, brevemente, cuál es la funcionalidad e importancia de las vistas parciales en MVC.
  - ✓ Cite y explique brevemente, los métodos de formulario que provee el motor Razor.
  - ✓ Describa la importancia de los *bundles* en una aplicativo MVC.

- ✓ Explique, brevemente, la funcionalidad e importancia de Single Page Applications (SPA).

Capítulo 5. ASP.NET Core

1. Un sitio web ASP.NET Core

Podría afirmarse que Visual Studio .NET trae consigo tres tecnologías que, aunque comparten lenguajes de programación, clases y funcionalidades, son muy diferentes. Estas tecnologías son .NET Framework, .NET Core y Xamarin. Con .NET Standard se pueden crear aplicaciones de cualquier tipo: consola, escritorio, web y para móviles.

Sin embargo, las aplicaciones web solo están disponibles para ejecutarse en ambientes Windows. Por su parte, .NET Core permite el desarrollo de aplicaciones multiplataforma, de alto rendimiento y pensadas para la nube. Lo importante es que con .NET Core se pueden crear aplicaciones web con ASP.NET Core y ASP.NET Core MVC.

2. Configuración JSON

JavaScript Object Notation (JSON) constituye un formato que se basa en texto estándar y se utiliza para representar datos estructurados en sintaxis JavaScript. Se usa para transmitir datos en aplicaciones web (enviar datos desde el cliente al servidor o viceversa, por ejemplo). ASP.NET Core emplea esta notación para la configuración de múltiples objetos. La figura 4.5. muestra el uso JSON en la configuración de ASP.NET Core.

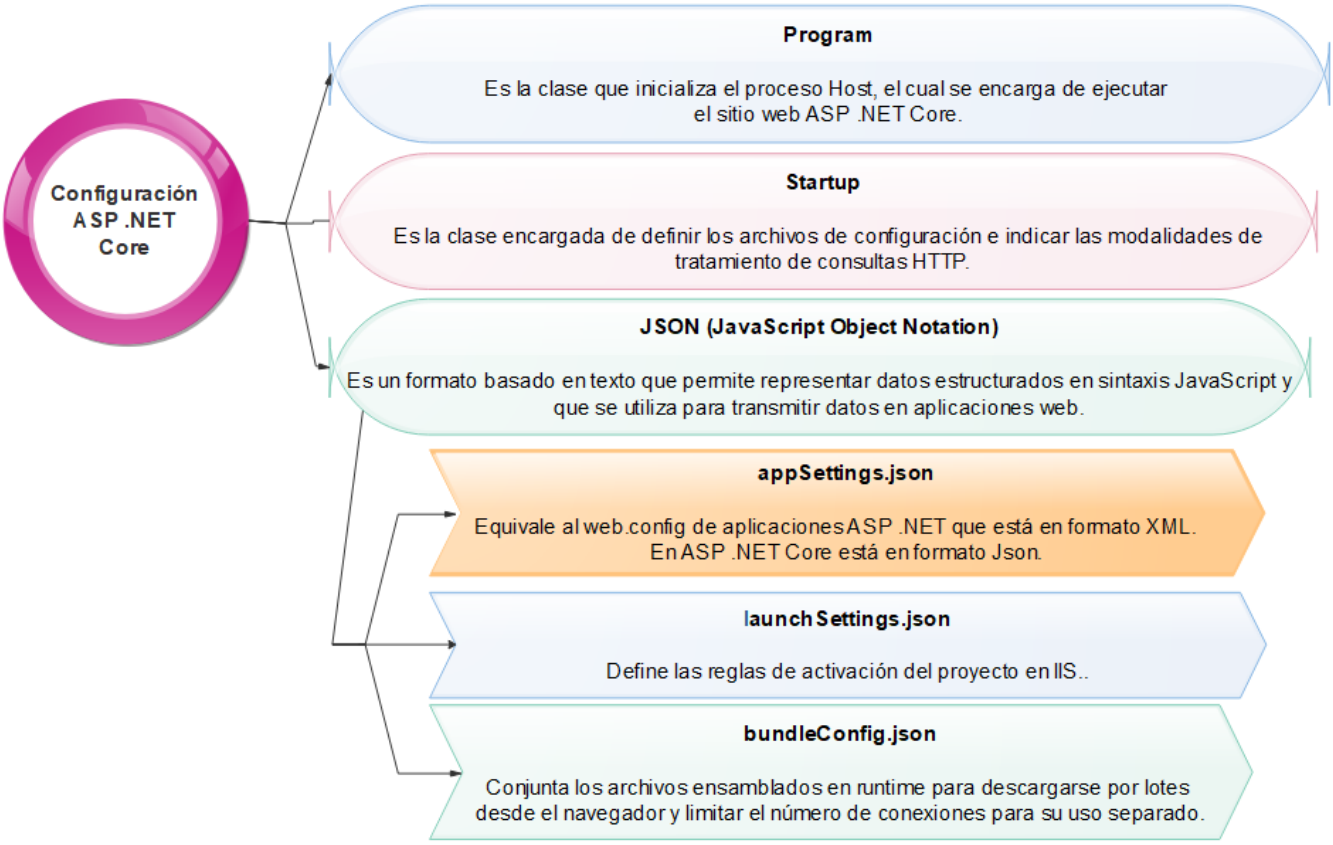


Figura 4.5. JSON en la configuración de sitios web ASP.NET Core

En el libro de texto, páginas 321 a 326, se demuestra, mediante ejemplos, el uso de Json en la configuración de sitios web ASP .NET Core. Asimismo, el uso de paquetes *Nuget*, *Bower* y *Bootstrap*.

3. Desarrollo MVC

El desarrollo MVC consta de controladores, modelos y vistas. Esas capas se interrelacionan entre sí para crear un aplicativo reutilizable y de muchas prestaciones funcionales. En las figuras 4.1, 4.2., 4.3 y 4.4, se mostraron someramente los componentes y la usabilidad de este patrón arquitectónico.

Resumen del capítulo

Este capítulo describe la estrategia de desarrollo web mediante ASP.NET Core y algunas diferencias con respecto a ASP.NET Estándar. Primero, se considera el ambiente de configuración; luego, la gestión de paquetes a través de NuGet y Bower, y, finalmente, un resumen del patrón arquitectónico MVC.

Ejercicios de autoevaluación del capítulo 5

- Ejercicios complementarios:
  - ✓ Investigue, con mayor detalle, la funcionalidad de JSON en aplicativos web.
  - ✓ Indague la importancia que tiene la gestión de paquetes NuGet y Bower para un aplicativo web.
  - ✓ Complemente el conocimiento esbozado en el libro con respecto al uso de JSON en lugar de formato XML en archivos de configuración de ASP.NET Core.

Capítulo 6. El acceso a datos con ADO.NET

1. Bases de datos ADO.NET

Activex Data Object (ADO) es la propuesta que Microsoft® estableció para la gestión de datos en aplicaciones de escritorio o web. Este framework está constituido por muchas librerías de clase y sus respectivos métodos, propiedades, entre otros. ADO.NET maneja dos formas de conexión a una base de datos: conectado y desconectado. La figura 4.6 muestra las clases principales de ADO.NET, considerando el modo conectado.

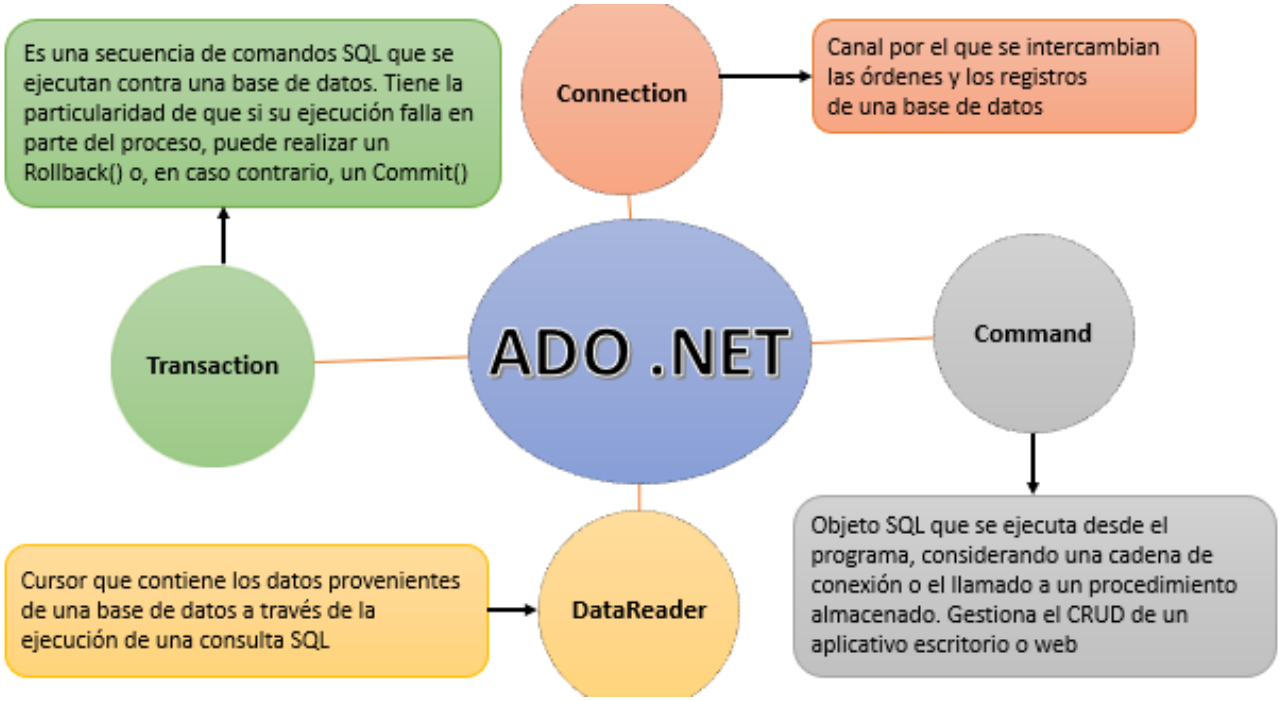


Figura 4.6. Clases principales de ADO.NET

En el libro de texto, páginas 331 a 344, se describe el concepto de ADO .NET y las principales clases y métodos con que cuenta para la gestión de bases de datos.

Para la gestión de bases de datos, lógicamente requerimos un DataBase Manager System (DBMS). Como motor nativo, las aplicaciones Visual Studio utilizan SQL Server; por medio del cual se pueden diseñar bases de datos y dentro de estas sus tablas, campos, relaciones, vistas y objetos que requieren programación como lo son procedimientos almacenados y funciones.

El modo conectado en ADO.NET significa que el aplicativo web estará la mayoría de las veces conectado a la fuente de datos, es decir, procesará la información directamente de la fuente de datos en el servidor. En este caso, abrirá la conexión a la base de datos cuando un usuario ingrese al aplicativo y la cerrará cuando salga de esta.

En el libro de texto, páginas 344 a 352, se explica cómo crear una base de datos, sus tablas, vistas y procedimientos almacenados, utilizando SQL Server.

El **modo desconectado,** en la gestión de datos de un aplicativo web, consiste en que los accesos directos a la fuente de datos sean los menos posibles. Componentes especiales de Microsoft® como el DataTable con un DataAdapter o un TableAdapter procuran extraer información de la base de datos y la puedan mostrar al usuario sin estar conectado directamente a la fuente. La manipulación de los datos se realiza en memoria y, cuando se requiera una actualización de la base de datos, se accede al servidor. Las clases empleadas por ADO.NET, en modo desconectado, son DataSet, DataTable, DataRow y DataColumn.

En el libro de texto, páginas 353 a 361, se explica el concepto de modo desconectado en un aplicativo web cuando gestiona datos. También, los componentes que se ven involucrados en este modo.

Asimismo, es posible utilizar las fábricas ADO.NET que se encargan del encapsulamiento que permite resolver la dependencia de proveedores genéricos como OleDB u ODBC para el acceso a bases de datos. Mediante el patrón de diseño *Factory* es posible instanciar clases de un driver determinado.

2. Acceso a los datos mediante proveedores

Cuando se trata del acceso a fuentes de datos por parte de aplicativos web ASP.NET, el tema se vuelve complicado. Sin embargo, ASP.NET ofrece varios niveles de desarrollo: modo conectado, modo desconectado y modo proveedor de datos. Cada uno provee su propio conjunto de librerías y componentes para ser utilizados por el desarrollador.

Existen controles especializados que permiten vincularse directamente a la base de datos, a saber, SqlDataSource y AccessDataSource. Estos orígenes de datos se especializan en el acceso a fuentes de datos SQL, permitiendo crear consultas de selección o de acción para proveer de información a objetos tales como un GridView. Con esos controles, es posible que se determine automáticamente la estrategia en el manejo del caché de datos, permitiendo recargarlos tras cada postback de la página contenedora. Por otro lado, se tiene el control ObjectDataSource, similar a los dos anteriores, pero en el cual el programador debe implementar la lógica de persistencia de los datos.

Entity Framework es un potente framework que permite los CRUD (Create, Read, Update y Delete) en una base de datos mediante un aplicativo web. Con este framework, es posible crear un modelo conceptual desde un modelo de base de datos relacional. En este mapa de contexto, las tablas del modelo relacional se comportan como clases y son accedidas por sentencias estilo Link to Entities. La figura 4.7. muestra los principales proveedores de datos de ADO.NET.

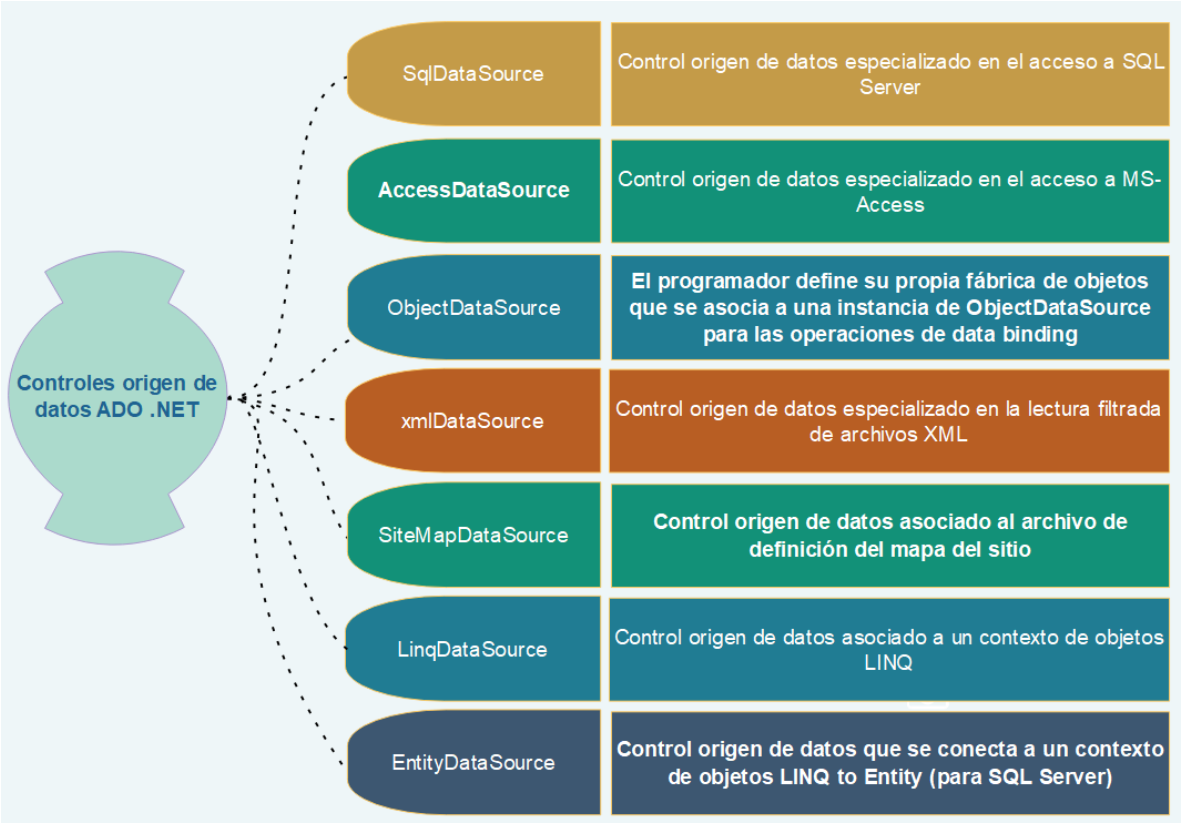


Figura 4.7. Principales proveedores de datos de ADO.NET

En el libro de texto, páginas 365 a 408, se trata el tema de los proveedores de datos que permiten actualizaciones y recuperación de información de una base de datos relacional. Se detalla el manejo funcional de cada proveedor, sus métodos y propiedades. Hace especial énfasis en el uso de Entity Framework para el modelado conceptual de una base de datos relacional.

3. Componentes gráficos de presentación de datos

Visual Studio .NET provee muchos componentes o controles que permiten la representación de datos en formularios web. Entre estos, encontramos el GridView, el DataList, el DetailsView, el FormView, el ListView y el Repeater. En el caso del GridView, es un control muy utilizado en aplicaciones web ASP.NET para mostrar datos en un formulario web. Posee muchas propiedades que posibilitan desde configurar la fuente de datos hasta las gestiones que se pueden realizar sobre estos.

En el libro de texto, páginas 409 a 422, se explica el control *GridView* de ASP .NET, sus propiedades y funcionalidad.

El control DetailsView, al contrario de GridView, trabaja sobre un registro a la vez. GridView presenta una serie de registros en forma vertical (cada fila es un registro), mientras DetailsView lo hace con registros individuales, también presentados verticalmente.

Además de la posibilidad de mostrar registros individuales, el DetailsView puede insertar, actualizar o eliminar registros en una tabla. Finalmente, el control FormView se utiliza para mostrar, al igual que DetailsView, un registro a la vez, proveniente de una fuente de datos. FormView usa plantillas para mostrar y editar valores enlazados a datos. Estas plantillas proveen controles, expresiones de enlace y formatos que permiten definir el aspecto y la funcionalidad del formulario.

Resumen del capítulo

Este capítulo describe la estrategia propuesta por Microsoft® para el acceso a los datos almacenados en una base de datos. Esta estrategia se denomina ADO.NET y permite la gestión de datos tanto en aplicaciones de escritorio como para la web. Se explican las clases y los controles existentes que permiten establecer la conectividad y el manejo de los datos, considerando los diversos sistemas de administración de bases de datos existentes. Entre los componentes que aporta Visual Studio .NET se tiene GridView y DetailsView que ofrecen grandes funcionalidades para el desarrollador web en cuanto la gestión de datos en formularios web.

Ejercicios de autoevaluación del capítulo 6

- Ejercicios complementarios:
  - ✓ Defina, con sus propias palabras, el concepto Modo Conectado y Modo Desconectado.
  - ✓ Cree una lista de controles que pertenecen al Modo Conectado y una breve descripción de su funcionalidad.
  - ✓ Determine la diferencia entre una consulta de selección y una consulta de actualización.
  - ✓ Programe un ejemplo sencillo que permita utilizar el concepto de fábrica de objetos (página 378 a 388 del libro de texto).
  - ✓ Explique la diferencia entre los controles GridView y DetailsView.
  - ✓ Programe un ejemplo sencillo que permita utilizar los controles GridView (página 409 a 422 del libro de texto) y DetailsView (página 422 a 424 del libro de texto)

Capítulo 7. Gestión del estado

1. Los distintos medios para mantener el estado

HTTP es un protocolo que trabaja de modo desconectado. De esta manera, no permite que el servidor web y la aplicación que se ejecuta del lado del cliente recuerden las peticiones que se producen. Por ende, es necesario que un aplicativo web procese la gestión de estados de tales peticiones. Existen mecanismos para este procedimiento tal como el uso de campos ocultos en un documento HTML, QueryString, variables de sesión, cookies o los ViewState en páginas de servidor ASPX.

El ViewState es un mecanismo que se maneja del lado del cliente. Este mecanismo almacena el valor en su propia página a través de campos ocultos. Ver la fuente de la página HTML permite encontrar la firma de ViewState, pero el valor almacenado se encuentra cifrado. Sin embargo, un programador experimentado podría descifrar ese valor.

QueryString es un conjunto de caracteres que se ingresan a una computadora o un navegador web. Mediante QueryString es posible recuperar los valores de las variables que han sido especificadas en una consulta HTTP. Esta cadena de consulta HTTP puede ser especificada mediante los valores que continúan al signo de interrogación (¿); también, se generan mediante el envío del formulario o cuando un usuario escribe directamente la consulta utilizando la barra de direcciones de un navegador web.

2. Sesiones

Las cookies y variables de sesión también son mecanismos que pueden ser utilizados para el paso de los valores contenidos en variables de un formulario web ASP.NET a otro; mientras que las cookies son segmentos de datos limitados (4 KB) y se manejan del lado del cliente. Las variables de sesión se manejan del lado del servidor y consumen pocos recursos; es posible que admitan cualquier tipo de objeto que se almacene y constituye una de las mejores técnicas para la administración del estado, permitiendo guardar datos para cada usuario de forma separada.

En el libro de texto, páginas 425 a 438 del libro de texto, se explican en detalle algunos medios que utiliza ASP .NET para la gestión del estado de los diferentes objetos que pertenecen a formularios web. Entre estos, se encuentran el uso de campos ocultos, el ViewState, variables de sesión, cookies, uso de QueryString, entre otros.

3. Objetos Application y Caché

El objeto Application se utiliza como un diccionario de términos identificados mediante claves. A diferencia de las variables de sesión, una variable de aplicación estará disponible para todos los usuarios de un sitio web. Supóngase que una variable de sesión se utiliza para almacenar el identificador y contraseña de un usuario particular. Estos datos serán específicos para este usuario y solo él los podrá ver o utilizar. Por otro lado, considérese un contador de visitas al sitio web, está bien podría ser una variable de aplicación para que todos los usuarios conozcan cuántos accesos se han producido.

Igual pasa con el objeto Caché. Estos objetos conservan los datos para todos los usuarios del sitio web. Si se producen dependencias asociativas entre objetos de este tipo, es posible eliminar sus datos. He ahí la complejidad de su implementación: precisar las dependencias que puedan existir entre objetos de tipo caché. Estas dependencias pueden provenir de archivos, con SQL Server, con HTML, entre otros.

Resumen del capítulo

Este capítulo desarrolla el tema de la gestión del estado de los objetos que se construyen en un formulario web. El protocolo HTTP es el medio por el cual las páginas web pueden realizar peticiones de datos y recursos, y para transmitir o actualizar documentos web. Sin embargo, no fue pensado para "recordar" los contenidos de los datos de los distintos objetos que se crean en esos documentos web. Para ello, ASP.NET se vale del uso de campos ocultos en HTML, los objetos *ViewState,* a través de QueryString, cookies, variables de aplicación o de sesión, entre otros para resolver esta carencia de HTTP.

Ejercicios de autoevaluación del capítulo 7

- Ejercicios complementarios:
  - ✓ Explique brevemente la diferencia entre variables de sesión y variables de aplicación.
  - ✓ Cree dos formularios web. En el primero, establezca los controles necesarios para una ventana de ingreso a un sitio web (con entrada de usuario y

contraseña) y en el otro muestre de la información ingresada en el primero formulario. Utilice variables de sesión o cookies.

- ✓ Explique brevemente la importancia del proceso w3wp (Worker Process).
- ✓ Explique brevemente la importancia del proceso Windows ASP.NET Service. ¿Qué pasaría si este servicio está deshabilitado en Windows?

Capítulo 9. Los servicios web WCF y REST

1. Los servicios web WCF

Simple Object Access Protocol (SOAP) es el enfoque dominante en Services Oriented Arquitecture (SOA). SOA se refiere al nexo que se establece entre las metas del negocio con el sistema de software. Aporta flexibilidad a los servicios empresariales facilitando automatización e infraestructura. SOAP, por su parte establece la interoperabilidad entre aplicaciones informáticas; sin embargo, se ha visto rezagada por una segunda alternativa de computación distribuida denominada Representational State Transfer (REST).

SOAP es un protocolo cuya especificación es completa y compleja. Es capaz de brindar solución a cualquier necesidad de comunicación entre componentes. Expone operaciones individuales que se denominan servicios web (*web services*). Estos servicios web son capaces de resolver o exponer soluciones a los aplicativos de la empresa o a terceros. Desde hace unos años, Microsoft® descontinuó los servicios web y comenzó a utilizar de Windows Communication Foundation (WCF); WCF es un conjunto de API que se orientan a la comunicación para lograr

que dos o más sistemas se puedan comunicar entre sí y resolver procesos. Sustituye a los servicios web ASMX, al control remoto .NET de objetos y algunos otros API.

En el libro de texto, páginas 512 a 525, se explica el concepto de SOAP (Simple Object Access Protocol) y los servicios web WCF (Windows Communication Foundation) como parte de la interoperabilidad de aplicaciones que se pueden establecer en ASP .NET. Se expone un ejemplo de la implementación y pruebas de servicios web WCF.

2. Servicios web REST

Representational State Transfer (REST) es un estilo arquitectónico que define un conjunto de recomendaciones o restricciones para crear servicios web utilizando el protocolo HTTP. En sí, un servicio web REST presenta una implementación similar a los demás servicios con algunas particularidades, por ejemplo, permite las cuatro operaciones de un servicio web: GET (consulta y lectura), POST (crear), DELETE (eliminar) y PUT (editar).

En el libro de texto, páginas 525 a 529, se explica el concepto de REST (Representational State Transfer) y su implementación básica en un servicio web en ASP .NET.

Resumen del capítulo

Este capítulo describe someramente el concepto de servicios web, considerando para ello el dialecto común de SOAP (Simple Object Access Protocol) y el uso de WCF (Windows Communication Foundation) y Rest (Representational State

Transfer). Los servicios web tienen como responsabilidad comunicar equipos que requieren intercambiar información; algunos resuelven lógica de negocios, otros son proveedores de información para terceros. Sea cual sea la función, siempre tendrán detrás una arquitectura que soportar mediante servicios SOAP o Rest.

Ejercicios de autoevaluación del capítulo 9

- Ejercicios complementarios:
  - ✓ Defina el concepto de SOA y SOAP, y establezca sus diferencias.
  - ✓ Conceptualice los servicios a través de SOAP y Rest, y determine cuáles son sus diferencias y virtudes.
  - ✓ Implemente un servicio web WCF que permita resolver operaciones sencillas de suma, resta, multiplicación y división.

Configuración, seguridad e implementación de aplicaciones web

III

Sumario

- Personalización y securización
- Configuración, despliegue y administración

Objetivos específicos

Al finalizar el estudio de este tema, usted estará en capacidad de:

- Securizar el acceso a las distintas páginas de un aplicativo web mediante configuración a niveles de sitio y de servidor.

- Configurar el despliegue de un aplicativo web considerando un proveedor externo de hosting.

Introducción

Este tema considera dos actividades importantes dentro de la creación de un aplicativo web. Por un lado, la seguridad del acceso por parte de los usuarios a las distintas páginas al aplicativo web y, por otro, el deploy en un servidor web externo. La seguridad del aplicativo web es un asunto importante que tratar y, por tanto, el desarrollador debe estar consciente de la responsabilidad que se tiene al configurarla. Una vez que se consigue la funcionalidad deseada y se establece la seguridad adecuada del aplicativo web es necesario publicarlo para ponerlo a disposición del público.

Guía de lectura

Para el estudio de este tema, lea detalladamente las páginas del libro que se indican a continuación:

Capítulo	Páginas
Capítulo 8. Personalización y securización	451-510
Capítulo 10. Configuración, despliegue y administración	531-553

Comentarios del tema

Capítulo 8. Personalización y securización

8. Securización de los sitios ASP.NET

Los sitios web, al estar expuestos al público, requieren definirse reglas de accesibilidad para todos los elementos del sitio. Esta seguridad puede basarse en los roles de las aplicaciones Windows (RBS: *Role Based Security*).

Para la securización de un sitio web, se pueden establecer sobre objetos de seguridad, a través de la identidad de los usuarios dentro de la base de datos estándar de Windows, de credenciales de un usuario dentro de una lista, autorización a través del web.config del aplicativo, entre otras estrategias.

A través de Windows, se puede securizar un aplicativo web mediante el archivo

web.config o, también, a través del Internet Information Services (IIS). Así mismo, es aconsejable configurar usuarios y grupos de Windows para realizar pruebas. Otro modo de securización es a modo *Forms.* La securización modo Forms es aprovechable para aplicaciones web que requieren autenticación, siempre que no sea de Windows. Se realiza en la configuración del web.config del aplicativo web.

En el libro de texto, páginas 451 a 481, se explican los métodos de securización que dispone Visual Studio .NET 2017 para los aplicativos web ASP .NET. Entre estos están la seguridad mediante Windows, a través de IIS o mediante funcionalidades propias de ASP .NET.

9. Presentación personalizada

La personalización de la interfaz de la cual gozará el usuario es otro aspecto por considerar al momento de diseñar un aplicativo web. Esto se logra creando perfiles de usuario, mediante el web.config. El archivo de configuración contiene el perfil del usuario que se intenta conectar al aplicativo web y de ahí expone propiedades esenciales como el tema de presentación de la interfaz (el skin, por ejemplo) o el mensaje de bienvenida.

Otro archivo importante dentro de la presentación del aplicativo web es Web.sitemap, el cual constituye el mapa del sitio web; en este, se detalla la jerarquía de las páginas que componen el aplicativo web. Es un archivo XML que agrupa elementos descendientes, al estilo de un árbol DOM. Para la navegación, dentro de esta jerarquía de páginas ASP.NET, se dispone de controles como Menú y TreeView.

El aspecto de internacionalización permite el acceso internacional al aplicativo web. En este caso, es posible formatear divisas, fechas y números según una cultura específica. La figura 8.1. muestra las posibilidades de culturización que permite Visual Studio .NET 2017 para internacionalizar las aplicaciones web que se desarrollen con este framework.

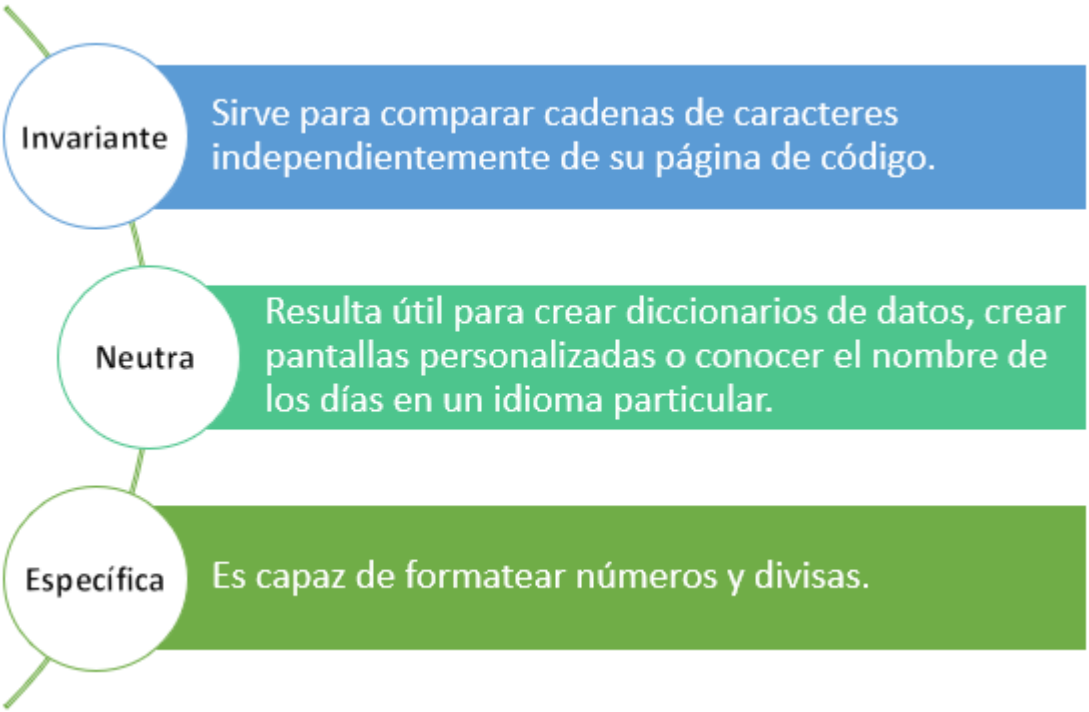


Figura 8.1. Tipos de culturas para la globalización de **aplicaciones web**

En el libro de texto, páginas 489 a 493, se resumen los mecanismos que ofrece Visual Studio .NET 2017 para la globalización de aplicaciones ASP .NET. Entre estos mecanismos se tienen los recursos globales y los locales de configuración, el componente localize, entre otros.

10. WebParts

Los WebParts permiten a un usuario personalizar la presentación, tanto de su disposición como de contenido. Por ende, generalmente, constituyen páginas de inicio configuradas con una determinada personalización de usuario. Son bloques de código que el usuario final puede decidir si lo muestra como una biblioteca de documentos, un gráfico o cualquier otro elemento, construyendo su propia página web.

Los portales web, por su parte, son una plataforma que recopila información de diferentes fuentes utilizando una única interfaz de usuario. Con el tiempo, la sencillez del inicio de la creación del portal genera tanta experiencia de usuario que se convierte en una verdadera plataforma de servicios. En estos portales existen componentes que son atinentes a su funcionalidad; entre los cuales tenemos el gestor WebPartManager, las zonas WebPartZone, CatalogZone y PageCatalogPart.

En el libro de texto, páginas 494 a 510, se explican los conceptos de WebParts y Portales. Dentro de la creación de portales existen componentes esenciales para su funcionamiento como *WebParts*, *WebPartManager*, *WebPartZone* y los controles *CatalogZone* y *PageCatalogPart*.

Resumen del capítulo

Este capítulo describe dos aspectos importantes dentro del desarrollo de un aplicativo web: la securización y la personalización. En primera instancia, Visual

Studio .NET 2017 ofrece varios mecanismos para implementar la seguridad en un aplicativo web. Desde la seguridad de Windows, el servidor IIS y la propia aplicación web, existen métodos eficaces para establecer la seguridad del aplicativo. En cuanto a la personalización, también Visual Studio .NET 2017 ofrece estrategias para configurar la interfaz que presentará un aplicativo web; también, ofrece estas características al desarrollador y al usuario final, desde la configuración de la internacionalización hasta la construcción de WebParts y Portales, Visual Studio .NET 2017.

Ejercicios de autoevaluación del capítulo 8

- Ejercicios complementarios:
  - ✓ Determine las diferencias existentes entre autenticación y autorización en aplicaciones Web ASP.NET.
  - ✓ Realice un resumen de la securización en modo Windows y la securización en modo Forms.
  - ✓ Explique brevemente la funcionalidad de AspNetSqlMembershipProvider.
  - ✓ Explique brevemente la funcionalidad de la carpeta Account.
  - ✓ Diseñe un ejemplo demostrativo de la creación de un portal. Puede replicar el que se explica en las páginas 494 a 510 del libro de texto.

Capítulo 10. Configuración, despliegue y administración

6. Configuración

Así como las aplicaciones Windows poseen un archivo de configuración (*app.config*), los aplicativos webs cuentan con el suyo (web.config). En el caso de web.config,

juega un papel más detallado, dado que cada carpeta virtual del proyecto puede declarar su propio archivo de configuración. Estos archivos particulares heredan parámetros que pueden estar definidos en el web.config de nivel superior. Los parámetros del archivo *machine.config* tienen preferencia en esta jerarquía de prioridad.

Visual Studio .NET tiene varios perfiles de compilación, siendo debug el que se encuentra por defecto. Esto quiere decir que el aplicativo web está preparado para que, si ocurre un error durante la fase de ejecución, se generen mensajes descriptivos de este y así el desarrollador tenga facilidades para el análisis y corrección. Por otro lado, existe también la compilación Release que produce ensamblados más compactos y el detalle de los errores acaecidos no presentarán mayor detalle. Por ende, el programador debe utilizar la estructura *try … catch* para la captura de excepciones.

7. Despliegue de aplicaciones ASP.NET

Una vez que se ha terminado el desarrollo de un aplicativo web corresponde realizar su despliegue. Esto es, publicarlo en un servidor de producción para que esté disponible para el usuario final. Este despliegue se puede realizar de forma manual en una carpeta virtual creada en IIS, por ejemplo; también, mediante un sistema de copia que se realiza a través del mismo Visual Studio .NET. Finalmente, se puede hacer la publicación del sitio mediante una cuenta en Azure, Somee o cualquier otro proveedor de internet.

En el libro de texto, páginas 535 a 550, se explica el proceso de publicación o despliegue de un aplicativo web, ya sea mediante implementación en IIS o utilizando una cuenta Azure.

8. Supervisión de aplicaciones ASP.NET

Los administradores de aplicativos web poseen herramientas que les facilitan el monitoreo del estado de estas aplicaciones en producción. Esto permite que el administrador revise mensajes enviados por la ejecución del aplicativo, mediante la infraestructura de supervisión Health Monitoring. Por medio de esta herramienta, se pueden monitorear señales como registros, mensajería, indicadores de rendimientos, entre otros.

Resumen del capítulo

Este capítulo describe dos aspectos importantes que suceden una vez terminada la fase de desarrollo de un aplicativo web. Por un lado, la configuración del aplicativo web a través del *machine.config* y el web.config, que ayudan a someter el aplicativo a normas de seguridad, conectividad, accesibilidad, entre otros. Y, por otro lado, se explica someramente el proceso de despliegue de un aplicativo web, en un servidor propio (en este caso IIS) o en uno externo (mediante Azure), para poner a disposición de los usuarios finales la funcionalidad del sitio web creado.

Ejercicios de autoevaluación del capítulo 10

- Ejercicios complementarios:
  - ✓ Describa brevemente los conceptos *machine.config* y web.config.
  - ✓ Explique brevemente la funcionalidad de compilación Release y Debug.
  - ✓ Vea, comprenda y aplique el procedimiento explicado en el video https://www.youtube.com/watch?v=oVFI4KiBPys para el despliegue de un aplicativo web en el servidor gratuito somee.com.
  - ✓ Explique brevemente el funcionamiento de la infraestructura Health Monitoring.