educación IT
CODE YOUR FUTURE

Programación Web .NET Core

Módulo 4

Descripción general de ASP.NET

Modelos de programación

- **ASP.NET 4.x:**
  - Web Pages.
  - Web Forms.
- **ASP.NET 4.x andASP.NETCore:**
  - MVC.
  - Web API.
- **ASP.NET Core:**
  - Razor Pages.

Páginas web

Disponible en **ASP.NET 4.x**.

Cuando se crea un sitio mediante páginas web, se puede escribir código mediante el lenguaje de programación **C#** (***.cshtml***) o **Visual Basic *(.vbhtml).*** Recursos, en el siguiente link.

Formularios web

Disponible en **ASP.NET 4.x**.

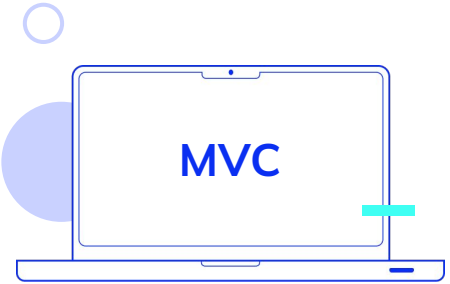
Cuando se construye un sitio con **formularios Web Forms**, se emplea un modelo de programación basado en eventos. El código se ejecuta del lado del servidor. Su extensión es ***.apsx***. Recurso.

MVC

Disponible en **ASP.NET Core** y **ASP.NET 4.x**.

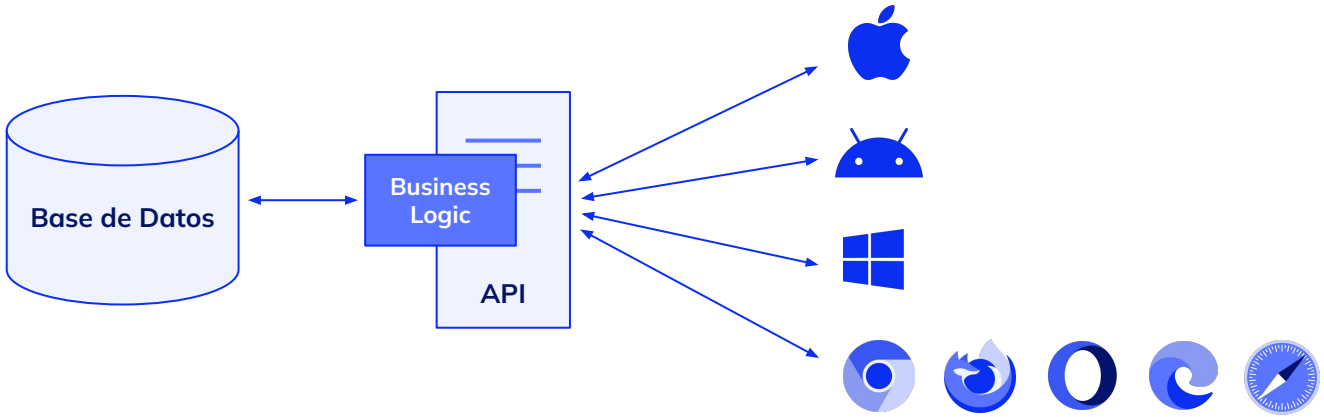
Cuando crea un sitio utilizando **MVC**, separa el código del lado del servidor en tres partes:

1. **Modelo:** Un modelo MVC define un conjunto de clases que representan los tipos de objetos que la aplicación web administra.
2. **Controladores:** Un controlador MVC es una clase que maneja la interacción del usuario, crea y modifica clases de modelo y selecciona las vistas adecuadas.
3. **Vistas:** Vista MVC es un componente que crea las páginas web que componen la aplicación web. Es la interfaz de usuario de la aplicación. Los controladores a menudo pasan una instancia de una clase de modelo a una vista.

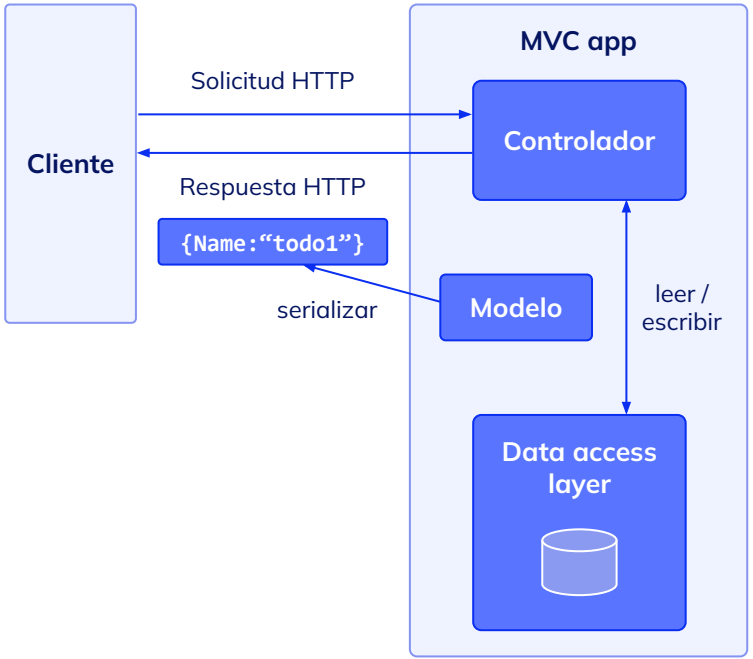


Web API

Disponible en **ASP.NET Core** y **ASP.NET 4.x**.



ASP.NET Web API es un marco para crear servicios **HTTP(RESTful)** a los que se puede acceder desde cualquier cliente, incluidos navegadores y dispositivos móviles. Gestiona las solicitudes **HTTP**, utiliza controladores como el modelo de **MVC** y se divide en distintas partes, que veremos en detalle en la siguiente diapositiva.



- **Controlador:** El controlador es una clase que maneja la solicitud del cliente que se envió al servidor. Accede o actualiza base de datos, recupera información, y devuelve la respuesta **HTTP** con un código de estado que indica si la acción se realizó correctamente y los datos, si es necesario.
- **Modelo:** Al igual que con **MVC**, es un conjunto de clases que representan los tipos de objetos que **Web API** gestiona.
- **Cliente:** El cliente envía solicitudes al servidor para ejecutar acciones específicas en el controlador de **Web API**. En el lado del servidor, crea una interfaz que consta de funciones a las que se puede acceder a través de **HTTP**. Esas llamadas se envían desde el cliente al servidor para recuperar información específica y hacer operaciones de lectura y escritura.

Páginas de Razor

- Las **páginas Razor** tienen una extensión ***.cshtml***.
- **Razor** nos ayuda a mezclar código de cliente en código de servidor.
- **Razor** es una sintaxis de programación **ASP.NET** utilizada para crear páginas web dinámicas con los lenguajes de programación **C#** o **Visual Basic .NET**.
- En **Razor C#** se usa el símbolo **@** para realizar la transición de **HTML** a **C#**. Razor evalúa las expresiones de **C#** y las representa en la salida **HTML**.

```
@* Some more Razor examples *@

<span>
    Price including Sale Tax: @Model.Price * 1.2
</span>
<span>
    Price including Sale Tax: @(Model.Price * 1.2)
</span>

@if (Model.Count > 5)
{
    <ol>
        @foreach (var item in Model)
        {
            <li>@item.Name</li>
        }
    </ol>
}
```

Descripción general de las páginas web

Páginas web

- Código en archivos **.CSHTML.**
- Control preciso sobre **HTML**.

El código de ejemplo en la derecha muestra cómo el **HTML** y **C#** son parte de las páginas web:

- El código muestra datos del objeto `item`, que es una clase modelo de **MVC**.
- Muestra de una colección de un modelo.
- Usa **`@foreach`** para recorrer cada modelo de la colección que recibe la vista.

```
<h2>  Special Offers </h2>
<p>
    Get the best possible value
    on Northwind specialty foods by
    taking advantage of these
    offers:
</p>

@foreach (var item in offers)
{
    <div class="offer-card">
        <div class="offer-picture">
            @if (!String.IsNullOrEmpty(item.PhotoUrl))
            {
                <img src="@Href(item.PhotoUrl) alt="@item.Title" />
            }
        </div>
    </div>
```

Descripción general de formularios web

Formularios web

- Dispone de código **HTML**.
- Variedad de controles de **ASP.NET** y controles **HTML**.
- Su extensión es .aspx y el archivo.subyacente es ***.aspx.cs***, cuando se usa **C#**.
- Crea una interfaz de usuario arrastrando los controles desde la vista de diseño.
- Los controles proporcionan propiedades y eventos enriquecidos.
- Simple enlace de controles a datos.

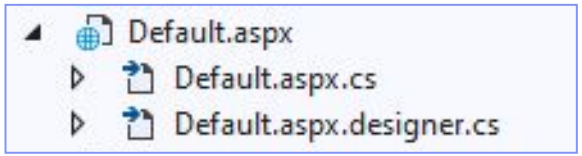
Web Forms

Una plantilla de proyecto para crear aplicaciones ASP.NET Forms. ASP.NET Forms le permite crear sitios web dinámicos con un modelo familiar controlado por eventos para arrastrar y colocar. Una superficie de diseño y cientos de controles y componentes le permiten crear rápidamente sofisticados y eficaces sitios controlador por la interfaz de usuario y con acceso a datos.

En una aplicación de formularios **Web Forms**, el **HTML** y el marcado de control se almacenan en archivos con extensión ***.aspx.***

Del lado del servidor el código en **C#**, generalmente, se escribe en un archivo ***.cs*** asociado llamado ***archivo de código subyacente.*** Por ejemplo, una página llamada *Default.aspx* tiene un archivo de código subyacente llamado *Default.aspx.cs*.

De manera similar, cuando escribe controles personalizados, almacena **HTML** y el marcado de control en un archivo ***.ascx*** y ***ascx.cs***.



Ventajas y desventajas de los formularios web

El modelo de programación de formularios Web Forms tiene las siguientes ventajas:

- Se puede diseñar una página, con controles de servidor, desde la vista de Diseño.
- Se cuenta con una amplia gama de controles que encapsulan funciones y propiedades.
- Es posible mostrar datos sin escribir muchas líneas de código del lado del servidor.
- La interfaz de usuario está en el ***.aspx*** y la lógica de negocios en el ***aspx.cs***.

El uso de un sitio de formularios Web Forms tiene algunas desventajas:

- El ciclo de vida de la página es una capa de abstracción sobre **HTTP** y puede comportarse de formas inesperadas. Debes tener una comprensión completa del ciclo de vida, para poder escribir código en los controladores de eventos correctos.
- No tiene un control preciso sobre el marcado generado por controles del lado del servidor.
- Los controles pueden agregar grandes cantidades de marcado e información de estado a la página **HTML** renderizada. Esta aumenta el tiempo de carga de las páginas.

Descripción general de MVC

MVC

- **MVC** es otro modelo de programación que está disponible en **ASP.NET 4.x**.
- Las aplicaciones **MVC** se caracterizan por una fuerte separación de lógica empresarial, datos, código de acceso y la interfaz de usuario en **modelos, controladores y vistas.**
- Es un avanzado modelo de programación para **Web Forms**, **Páginas Web**, y es adecuado para crear aplicaciones a gran escala.

MVC

Una plantilla de proyecto para crear aplicaciones ASP.NET MVC. ASP.NET MVC permite compilar aplicaciones mediante la arquitectura de controlador de vista de modelos. ASP.NET MVC incluye muchas características que permiten un desarrollo rápido orientado a pruebas para crear aplicaciones que usan los últimos estándares.

Modelos

Un modelo contiene lógica empresarial de la aplicación, validación y lógica de acceso a la base de datos. Cada sitio web presenta información sobre diferentes tipos de objetos para los visitantes del sitio. Por ejemplo, el sitio web de un editor puede presentar información sobre libros y autores.

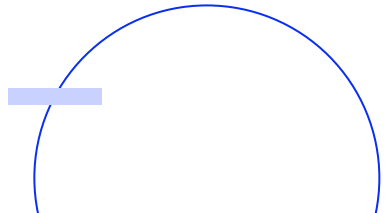
Vista

Cada sitio web debe representar a las páginas **HTML** que se puedan mostrar en un navegador. Esta representación se completa con vistas.

> Por ejemplo, en el sitio de publicación, una vista puede recuperar datos del modelo *Libro* y representarlos en una página web, para que el usuario vea todos los detalles.

Controladores

Cada sitio web debe interactuar con los usuarios cuando hacen clic en botones y enlaces. Los controladores realizan lo siguiente: responden a las acciones del usuario, cargan los datos desde un modelo y los pasan a una vista.



Ventajas y desventajas de MVC

El modelo de programación MVC posee las siguientes ventajas:

- Las vistas permiten al desarrollador tomar un control preciso del **HTML** que se representa.
- Puede utilizar el motor de enrutamiento para tomar un control preciso de las **URL**.
- La lógica empresarial, la lógica de entrada y la lógica de la interfaz de usuario se separan en modelos, controladores y vistas.
- Son posibles las técnicas de prueba unitaria y el desarrollo basado en pruebas (**TDD**).

El uso de un sitio MVC tiene desventajas:

- **MVC** es potencialmente más complejo de entender que las páginas web o los formularios web.
- **MVC** obliga a separar (modelos, vistas y controladores). Algunos programadores pueden considerarlo desafiante.
- No es posible crear de manera visual una interfaz de usuario, al arrastrar los controles a una página.
- Requiere un conocimiento completo de **HTML**, **CSS** y **JavaScript** para desarrollar vistas.

Características de ASP.NET 4.x

Características compartidas de ASP.NET 4.x

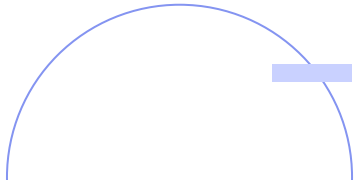
- Configuración.
- Autenticación.
- Membrecía y roles.
- Administración del Estado.
- Almacenamiento en caché.

ASP.NET 4.x incluye una variedad de funciones que están disponibles independientemente del modelo de programación que utilice. Esto significa que si está familiarizado con estas funciones el conocimiento se puede usar también en otro modelo de programación.

Configuración

Con los archivos ***Web.config***, se puede configurar una aplicación web, de forma independiente del modelo de programación. Los archivos llamados ***Web.config*** son archivos **XML** con etiquetas y atributos específicos que **ASP.NET 4.x** acepta en el tiempo de ejecución. En el archivo ***Web.config*** se puede acceder a la configuración a través del espacio de nombres ***System.Web.Configuration***.

> Por ejemplo, es posible configurar conexiones de base de datos y errores personalizados.



Autenticación y autorización

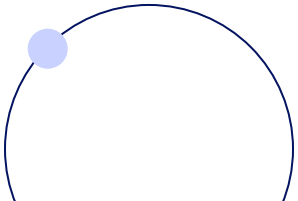
Muchos sitios web requieren que los usuarios inicien sesión ingresando un nombre de usuario y contraseña, o proporcionando información adicional. Se pueden utilizar proveedores de membrecía **ASP.NET 4.x** para autenticar y autorizar a los usuarios y restringir el acceso al contenido. También se pueden crear páginas que permitan membrecía y roles. ASP.NET 4.x admite varios proveedores de membrecía, como el denominado ***MemberShipProvider***, que usa una base de datos de **SQL Server** para almacenar el usuario. También puede crear un proveedor de membresía personalizado, al heredar de uno de los proveedores predeterminados si tiene requisitos únicos.

Administración del Estado

Los servidores web y los navegadores web se comunican a través de **HTTP**. Este es un protocolo sin estado en el que cada solicitud es independiente de las solicitudes anteriores y posteriores. Los valores de solicitudes anteriores no son recordados automáticamente.

Almacenamiento en caché

Una página **ASP.NET 4.x** se crea en forma dinámica durante el tiempo de ejecución en el servidor web. Por ejemplo, en una aplicación de páginas web, se ejecuta el código **C#** en la página para representar el HTML y devolverlo al navegador. Ese código **C#** , muchas veces, hace operaciones complejas que requieren mucho tiempo. Ejecuta varias consultas contra una base de datos o llama a servicios en servidores remotos. Para mitigar estos retrasos de tiempo se utilizan Cachés de **ASP.NET 4.x**.



educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!