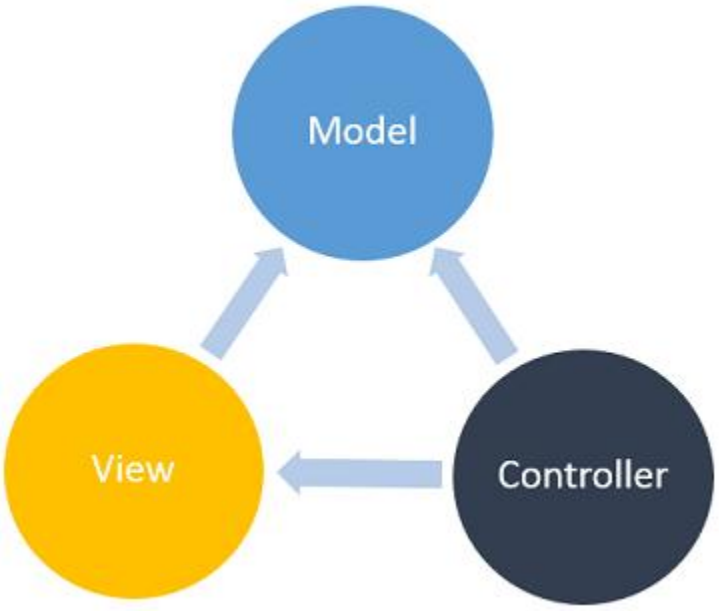
Laboratorio 3 – Primera Aplicación MVC .NET

¿Qué es MVC?

Modelo-vista-controlador (MVC) es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado, define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.



De manera genérica, los componentes de MVC se podrían definir como sigue:

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto, gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.12

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto, se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' (véase Middleware).

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto, requiere de dicho 'modelo' la información que debe representar como salida.

¿Cómo construir una Aplicación Web con MVC?

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación. Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón, estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Los primeros frameworks MVC para desarrollo web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor. En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor.

Como las tecnologías web han madurado, ahora existen frameworks como JavaScriptMVC, Backbone o jQuery14 que permiten que ciertos componentes MVC se ejecuten parcial o totalmente en el cliente. Para este ramo utilizaremos el framework ASP.NET MVC.

Solo como referencia, aquí hay los framework MVC utilizados para diferentes lenguajes.

Lenguaje	Framwork
C#	ASP.NET MVC
C#	Maverick.NET
C#	Spring.NET
C#	MonoRail
C++	treefrog
Objective C	Cocoa
Java	Spring MVC
Java	Struts 2
Java	Vaadin
Ruby	Ruby on Rails
PHP	Laravel
PHP	Symfony
Python	Django
JavaScript	AngularJS

ASP .NET MVC

El ASP.NET MVC Framework es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador (MVC).

Basado en ASP.NET, permite a los desarrolladores de software construir una aplicación web como una composición de tres funciones: modelo, vista y controlador.

En marzo de 2009 se hizo pública la primera versión de ASP.NET MVC. El patrón de arquitectura MVC (model-view-controller) no es nuevo (data de 1979) ni es algo que haya inventado Microsoft. Existen muchos frameworks de desarrollo web populares que utilizan MVC, como por ejemplo Ruby on Rails, Spring o Apache Struts. MVC es un patrón de arquitectura que como se indicó anteriormente nos ayuda a crear una separación lógica entre el modelo (información y lógica de negocio), la vista (la lógica de presentación) y el controlador (intermediario entre la vista y el modelo).

Uno de los pilares básicos de ASP.NET MVC es el concepto de enrutamiento (routing), lo que permite a las aplicaciones aceptar peticiones a URL que no se corresponden con ficheros físicos en el servidor. Por ejemplo, en ASP.NET Web Forms las URL tienen el siguiente formato “http://website/products.aspx?category=dvd” en el que físicamente existe un fichero products.aspx en la raíz del sitio web. En MVC la misma URL tendría el

siguiente aspecto “http://website/products/dvd” sin que el servidor web necesariamente contenga una carpeta products con una subcarpeta dvd. De forma predeterminada, ASP.NET MVC enruta las peticiones al controlador y a la vista adecuada en función de la URL. Es decir, en el ejemplo anterior, nos devolverá la vista dvd del controlador products.

Motor de Vistas MVC

Desde su primera versión ASP.NET MVC ha tenido el concepto de motor de vistas (View Engine). En ASP.NET MVC las vistas realizan tareas sólo de presentación. No contienen ningún tipo de lógica de negocio y no acceden a datos. Básicamente se limitan a mostrar datos y a solicitar datos nuevos al usuario. Si vienes del mundo de webforms, olvídate del concepto de Web Controls: no existen en MVC. No tenemos drag and drop, no configuramos propiedades. Las vistas son básicamente HTML. Y lo que no es HTML son pequeñas porciones de código de servidor destinadas a terminar generando HTML para mostrar información.

El equipo que desarrolló ASP.NET MVC tuvo la buena idea de permitir separar la sintaxis de servidor usada, del framework de ASP.NET MVC. ¿El resultado? Lo que llamamos un motor de vistas de ASP.NET MVC. En las versiones 1 y 2, el framework viene con un único motor de vistas, el llamado motor ASPX, que usa los clásicos archivos .aspx como vistas de ASP.NET MVC. Pero ¡ojo! Aunque estemos utilizando archivos .aspx, no son webforms (los webcontrols no funcionan, no tenemos viewstate ni el ciclo de vida de webforms. En las siguientes versiones del framework se incorpora un nuevo motor de vista llamado RAZOR, este se convertirá en el mas utilizado y que utilizaremos en este curso.

Desarrollo del Laboratorio

Primera Fase de Desarrollo Aplicación Web MVC, creación de interfaz y menú con routing.

Paso 1 (20 puntos). -

Crear Nueva aplicación ASP.NET MVC. Modificar el Menú con los ítems: Inicio, Acerca, Contáctanos y Productos, además de cambiar el nombre por defecto a la aplicación. esto se debe hacer desde el Layout de la solución.

El nombre de la aplicación, se debe escribir como AppSetting dentro del archivo WebConfig.

Paso 3(20 puntos).-

Dentro de la carpeta de modelo, debe crear el siguiente modelo de datos:

Producto

Propiedades:

- Nombre → String
- Codigo de Barra → String
- Precio → int
- Cantidad Disponible→ int
- Estado → bool (indica si el producto está disponible)

ProductoPerecible (hace herencia del objeto Producto)

Propiedades:

- Fecha de Vencimiento → Date
- DiasParaVencer →Int (Set Privado, se debe setear solo tomando la fecha de vencimiento y la fecha actual)

ProductoNoPerecible (hace herencia del objeto Producto)

Propiedades:

- Tipo de Envasado → Envasado

Envasado

Propiedades:

- Nombre →String
- Descripcion→String

Agregar la librería `System.ComponentModel.DataAnnotations,` al código de su clase y utilizar las etiquetas Display Name, String Length y Required (para todos las propiedades)

Paso 3 (20 puntos).-

Implemente las vistas que permitan listar todos los productos, además que en esta lista permita seleccionar un producto y ver sus características.

Agregar opción de agregar nuevo producto.