1.-

Hola a todos, mi nombre es Angel Núñez Salazar, me desempeño como consultor en tecnologías MS y en lo que se refiere a prácticas de ingeniería ágiles. En este día les vengo a presentar ASP.NET MVC, daremos un rápido recorrido a todas las características y beneficios que tiene esta framework y xq es la mejor opción para el desarrollo web dentro de la plataforma de MS. Entonces comencemos.

2.-

Cuando hablamos de una nueva framework o tecnología es casi inevitable hablar sobre cuáles son los patrones y principios que están detrás de estas cosas. Entonces…

3.-

Esta framework como su nombre también lo indica está basado sobre el ya conocido patrón MVC, que es principalmente un patrón de presentación, usado para separar las responsabilidades de nuestra aplicación en por lo menos 3 partes principales que son el Modelo , la vistas y los controladores. Lo cual beneficia a nuestra aplicación en términos de mantenibilidad, flexibilidad y testeabilidad.

Este patrón nos dice que existe un controller que es el encargado de recibir y manejar todas las peticiones que vienen del cliente, este manipula un modelo, un modelo de presentación el cual luego es pasado a la vista y por último esta se encarga de mostrar este modelo en un formato de salida adecuado.

Si nos damos cuenta, este es un patrón bastante simple y probablemente el patrón de arquitectra más antiguo que existe, fue formulado alrededor de los 70s por [Trygve Reenskaug](http://heim.ifi.uio.no/~trygver){tringvi riinskog} y desde esa época han surgido muchas implementaciones de este patrón en distintas plataformas, como por ejemplo ruby, python y NET tampoco ha sido la excepción.

4.-

En este punto es importante decir que ASP.NET MVC no es la primera implementación de este patrón sobre la plataforma .NET, desde hace muchos años han existido y existen muchas otras frameworks MVC, dentro de las más importantes podemos recalcar a MonoRails y FubuMVC siguen teniendo un desarrollo y evolución constante.

5.-

Entonces que es ASP.NET MVC, ASP.NET MVC es una framework desarrollada por MS y que ha sido liberada como open source , esta combina todas las ventajas y efectividad de arquitectura MVC, asimismo otra de sus principales características es su diseño, ya que nos permite muy fácilmente la aplicación de los principios, patrones, técnicas y prácticas que han ido surgiendo durante todo este tiempo que ha ido evolucionando el desarrollo de software, como por ejemplo las prácticas ágiles. Asimismo, esto aprovechando y utilizando todo lo que ya existe desde hace mucho tiempo en asp.net. Para aclarar esto último.

6.-

ASP.NET MVC es una nueva alternativa que nos presenta MS para el desarrollo de aplicaciones web dentro de la plataforma .NET. ASP.NET MVC no ha sido construida desde cero sino que se encuentra sobre el core de ASP.NET lo que nos permite seguir utilizando todas las características que esta tiene ( MasterPages, Sesiones, Seguridad, Caching) y que se encuentra dentro de un ensamblado diferente, el system.web.mvc. Hablando en términos un poco más técnicos ASP.NET MVC es más que nada un nuevo handler que se ejecuta sobre ASP.NET

8.-

Primero comencemos con la más obvia y razonable dentro del desarrollo de software. Si tú ya tienes una aplicación sólida, estable, que genera ganancias de manera contínua, no existe ninguna razón para cambiar a ASP.NET MVC o a ninguna otra herramienta. No habrá nada, por lo menos a corto plazó que te haga decír AHA, por todo el tiempo y dinero ahorraste al utilizar MVC.

Pero si miramos un poco más ayá dentro de 1 o 2 años y la aplicación va a cambiar por las necesidades del negocio y del mercado, o nuestro negocios se encuentra en crecimiento y necesita daptarse para continuar en ascenso; sabemos para estos casos, la seguridad de que todos los cambios sean introducidos con menor fricción e impacto posible es una necesidad para el negocio, y definitivamente MVC es algo a considerar. Esto no solo es aplicable en una visión a largo plazo, sino también para aplicaciones que se encuentran en plena construcción y se ajustan o necesitan ajustarse a un modelo de desarrollo ágil que abrazan el descubrimiento o apertura al cambio como una forma de dar valor al negocio, o tal vez aplicaciones en las cuales existe cierta incertidumbre a nivel de dominio. Para todos estos casos en los cuales necesitemos afrontar el cambio, que es la única constante dentro del software, ASP.NET MVC es la perfecta opción debido a su flexibilidad, testeabilidad y extensibilidad que brinda a las aplicaciones.

Considerar ASP.NET MVC, es muy importante para las personas de negocio que necesitan buscan y contratan nueva ayuda. Como probablemente tú que tiene un negocio grande o pequeñe siempre estarán en búsqueda de nuevas personas para tu negocios y cuando algo es tan simple, fácil de adoptar y aprender y basado en cosas muy conocidas y establecidas, pero sin perder nada de su potencia como lo es ASP.NET MVC esta tarea de búsqueda se facilita muchísimo, no solo por desarrolladores experimentados, no solo dentro de .NET sinó también de otras plataformas que se pueden adpatar muy rápidamente, sino también por gente joven que recién está entrando al negocio. Por lo tanto siempre podrás encontrar personas para que crescan juntu a ti y tu negocios.

Por último, sabemos que ninguna persona se siente tranquila en un trabajo aburrido, que no lo haga crecer profesionalmente, o que no lo llene como peersona, y esto suele ser una de las causas principales de la gran rotación de personas que existe, es por esto que ASP.NET MVC permite retar a todo tu equipo, no solo por todas las características que tiene, sino también a la facilidad con la cual se pueden aplicar tantas cosas nuevas que han surgido dentro del desarrollo de software, las cuales incrementan la productividad del equipo y también la solidez y estabilidad de la aplicación, lo que significa mayor dinero para el negocio.

Entonces si tu no quieres perder a las personas más importantes dentro tu empresa y a la vez incrementar las ganancias, ASP.NET MVC es para ti.

10.-

ASP.NET MVC no solo nos provee una gran separación de responsabilidades gracias a la adopción de la arquitectura MVC que es tan útil en el desarrollo web debido a que las aplicaciones suelen ser una combinación de responsabilidades y necesitamos una adecuada separación de responsabilidades.

Sino también está compuesto de manera interna por una arquitectura pipeline conformada por una serie de componentes independientes lo que nos permite crear nuestra propia implementación o extender una ya existente según nuestras necesidades.

Por ejemplo esto es bastante útiles desde casos simples en los cuales queramos reutilizar toda nuestra aplicación y únicamente devolver una vista diferente cuando nuestra aplicación se esté ejecutando desde un dispositivo mobil o inclusive vistas diferentes según la marca del dispositivo, o hasta casos más complejos en los cuales tengamos que realizar aplicaciones multitenant , es decir aplicaciones que comparten un mismo código base y una misma infraestructura, pero que atienden a multiples clientes con necesidades diferentes que van desde una presentación o vistas diferentes hasta reglas de negocio especiales, entonces nosotros podemos extender ASP.NET MVC para que pueda resolver de manera automática cuales son los componentes adecuados para cada tenant de acuerdo a su configuración o inclusive su host.

Casos similares a estos son muy comunes, pero ASP.NET MVC responde muy bien a estas necesidades gracias a gran extensibilidad y flexibilidad.

11.-

Otra de las principales características de ASP.NET MVC es su testeabilidad. Una arquitectura MVC nos brinda un gran punto de entrada para hacer nuestras aplicaciones mantenibles y testeables debido a la natural separación de responsabilidades que esta tiene.

Pero ASP.NET MVC no se quedó ahí, cada pieza fue diseñada para tener la seguridad que tenga una estructura idealmente adecuada para realizar test automatizados sobre ella.

De esta manera nosotros podemos escribir test unitarios sobre las diversas partes de la framework de manera muy fácil y limpia, utilizando la testing framwork de nuestra preferencia, sin tener que preocuparnos de lidiar con partes específicas de la framework como por ejemplos clases http request,response o binders, e inclusive dentro de los casos más complejos estas partes específicas son fácilmente mockeables.

Y cuando es tan natural de aplicar, siempre nos permite ir un paso más alla, como por ejemplo fácil inclusión de prácticas ágiles como las especificaciones ejecutables o test driven development

12.-

Sabemos que ingeniería de software es una ciencia bastante nuevo, pero desde sus inicio se ha encontrado en una constante evolución para poder asegurar que nuestras aplicaciones siempre mantengan un nivel de calidad tanto interna como externa. Es por esto que ahora escuchamos de la existencia de una gran cantidad de nuevas prácticas, principios, patrones, etc que buscan mejorar el software y la forma como lo hacemos.

ASP.NET MVC es totalmente compatible con todos estos conceptos, de tal manera que podemos aplicarlos muy rápidamente y empezar a captar todos sus beneficios desde el primer día.

13.-

ASP.NET MVC reconoce la importancia de contar con un buen HTML como pieza clave en el desarrollo web, es por esto que nos brinda todas las herramientas necesarias para podamos producir un html limpio que pueda ser fácilmente manipulado y extendido por lo desarrolladores y diseñadores;

Un html semántico de tal manera que podamos expresar claramente nuestra intención y la importancia de determinada información a través del markup, de tal manera que esta intención sea también entendida por los motores de búsqueda y asi mejorar nuestro posicionamiento y la búsqueda de información a través de esas herramientas.

Y por supuesto la compatibilidad con los estándares que es una prioridad en el desarrollo web actual.

Pero no solo hablemos hablemos de html, sino de todas las herramientas a nivel de cliene que nos permiten crear interfaces ricas de usuario, ASP.NET MVC tiene una estupenda integración con estas herramientas, e inclusive haciendo muy natural y fácil la transmisión de datos de manera asíncrona entre nuestras vistas y controladores que es necesario para una mejor interacción de los usuarios con nuestra aplicación.

Como muestra de esto ASP.NET MVC ya incluye dentro del proyecto por defecto varias de estas herramientas como por ejemplo Jquery.

14.-

Para terminar con este punto, sabemos que ASP.NET MVC se caracteriza por su extensibilidad, y otra muestra de esto es su fácil integración con las mejores view engines que existen, como por ejemplo spark o haml. Pero desde su última versión, trae una view engine completamente nueva que es Razor, cuya principal característica es romper la fricción que usualmente existe en nuestras vistas cuando mezclamos determinado código e un lenguaje de programación con etiquetas hmtl, de tal manera que facilite la manipulación de las vistas y también se incremente la productividad de los desarrolladores

15.-

Hoy en día, el diseño de las URLs que tendrán nuestras páginas es una tarea muy seria e importante, ya que es esencial permitir que los usuarios puedan manipular, transmitir y recordar fácilmente las urls y poderlas relacionarlas casi instintivamente con el contenido o información de las páginas. Esto no solo de cara a los usuarios, sino también hacia los motores de búsqueda que valoran mucho la url al indexar nuestra aplicación y darle mayor relevancia frente a determinadas palabras clave.

16.-

Debido a estas necesidades es que ASP.NET MVC nos provee con un sistema de routing para que tengamos un completo control de cómo se muestran las urls de nuestra aplicación.

Si recordamos estas Urls ya no corresponden a archivos físicos, sino a un determinado controller y acción que recibirá el request que la Url representa.

Debido a la importancia que tiene manejar las url, este sistema que fue construido para ASP.NET MVC fue separado en un ensamblado diferente permitiendo su uso también desde aplicaciones webforms.

17.-

Usualmente en nuestras aplicaciones tenemos que realizar algún tipo de configuración, y para esta tarea tenemos muchas opciones, pero la peor de todas es escribir xml, o lo que es aún peor toneladas de xml, ya que disminuye terriblemente la productividad (no se entiende rápidamente, difícil de modificar, refáctoring)

Para evitar esta práctica dañina o cualquier otra situación en la cual tengamos que escribir gran cantidad de código de manera repetitiva, existe la filosofía o técnica denominada “coc”, que nos dice que hay que evitar escribir configuraciones explícitas y por el contrario definamos ciertas reglas, normas o convenciones que todos podamos seguir. De tal manera que únicamente nos preocupemos de las partes no covencionales de nuestra aplicación.

Introducir un número pequeño de estas convenciones puede incrementar drásticamente la productividad pero sin perder la flexibilidad.

18.-

ASP.NET MVC explota esta filosofía ya que podemos verla aplicada en múltiples lugares de la framework.

Hemos visto la convención en la cual ubicamos y nombramos las vistas de acuerdo al nombre del controlador y la acción. También hemos visto como nuestros objetos del dominio son persistidos en la base de datos sin la necesidad de ningún mapeo objeto, base de datos relacional, únicamente utilizando la convención en la cual los nombres de las clases y sus propiedad debe ser el mismo nombre de la tabla y sus columnas.

Asimismo hemos visto cómo podemos definir ciertas convenciones en las rutas para que se ´puedan seleccionar automáticamente cuales son los controladores que recibirán las solicitud.

Pero aún no hemos visto cómo podemos utilizar convenciones para aplicar el principio de inversión de dependencias dentro de ASP.NET MVC

Para terminar este punto, esta filosofía de “coc” ya se ha establecido de manera general dentro de la plataforma de .net. Y por lo tanto vamos a encontrar que las mejores herramientas y frameworks la implementan.

19.-

ASP.NET MVC desde inclusive antes de su salida, tuvo una gran acogida por parte de la comunidad, que creó múltiples proyectos alrededor de MVC, lo que permitio que desde la versión 1 sea totalmente utilizable y poderosa. Ahora en en su versión 3 existen todo un ecosistema alrededor de ASP.NET MVC compuesto de herramientas, frameworks, aplicaciones, etc que los podemos descargar y utilizar inmediatamente como complemento a nuestras aplicaciones.

Asimismo se han formado múltiples comunidades que debaten y comparten todo tipo información centrada en ASP.NET MVC. Por ejemplo tenemos C4MVC que tiene una lista bastante activa y tiene reuniones virtuales de manera regular. También tenemos la conferencia anual que se transmite en línea de ASP.NET MVC que por cierto este año comienza este mismo lunes así que pueden buscar mvcconf en el navegador e inscribirse. Por otro lado tenemos

Una muestra de todo esto es que si realizamos una búsqueda en codeplex que es la comunidad que aloja proyectos open source orientados a tecnologías MS, encontraremos 241 resultados para las palabras claves ASP.NET MVC.

20.-

Una gran movida que realizó MS fue liberar ASP.NET MVC como open source bajo licencia Ms-pl, que es una licencia open source aprobada por la open source iniciative. Es por esto que nosotros podemos ir a la página de ASP.NET MVC y descargarnos todo el código fuentes, inclusive podemos realzar ciertas modificaciones según nuestras necesidades, volver a compilarlo y utilizarlo, dentro de la licencia no está permitido distribuir múltiples versiones personalizadas del ensamblado para evitar colisiones pero sí poder corregir algún bug, y utilizar los binarios.

Esto también brinda la posibilidad de navegar por el código y entender todo su funcionamiento por motivos de aprendizaje o para ver las posibilidades de desarrollo que existen.

Con esto hemos hablado de algunas de las ventajas que hacen de ASP.NET MVC una gran alternativa al momento de realizar aplicaciones web.

21.-

Ahora cambiando un poco de tema, cuando hablamos de ASP.NET MVC es casi inevitable hablar también sobre webforms.

ASP.NET significó un gran cambio cuando apareció por primera vez, no solo en términos del nuevo .NET Framework sino en que pretendía cerrar la brecha entre el desarrollo de Windows Forms (statefull) y el desarrollo web - HTML (stateless).

Se intentó ocultar toda la complejidad que significaba el manejo y conocimiento del HTTP y el HTML (en ese momento desconocido para muchos desarrolladores), mediante el modelado de la interfaz de usuario a través de una jerarquía de controles en el lado de servidor, que automáticamente se renderizaban como HTML y se conectaban con las acciones realizadas en el cliente, mediante eventos que se ejecutaban también en el lado del servidor.

Los desarrolladores no tenían nunca más que preocuparse de las llamadas y respuestas independientes HTTP y podrían pensar en términos de una UI que mantenía su estado.

WebForms representa una excelente capa de abstracción para trabajar con una clásica UI orientada a eventos sobre la Web.

Y esto fue un boom absoluto, se imaginan hace 10 años brindar a los desarrolladores de escritorio la capacidad de crear aplicaciones web sin cambiar su forma de programación en el momento que la web era relativamente nueva pero que ya empezaba a tomar gran fuerza.

22.-

Pero ahora luego de todos estos años ya no queremos algo que nos oculte el comportamiento de la web, sino queremos tener control de ella porque ahí existe un gran potencial. Asimismo muchas de estas cosas que nos abstraen el comportamiento de la web, pueden llegar a ser un problema si no se usan adecuadamente. Por ejemplo el famoso viewstate que es un campo oculto a través del cuál se guarda almacena el estado de una página en webforms, sino se usa responsablemente puede impactar en el tiempo de carga de la página, y lo peor de todo esto que pocas personas realmente saben hacer un uso adecuado de esto.

Lo mismo sucede con el pagelifecicle que es una gigantesca seríe de eventos que se ejecutan cada vez que creamos una página, que a diferencia de MVC que es un flujo sencillo, esto puede llegar a ser muy complicado de aprender para su aplicación correcta.

También tenemos los controles que nos quitan el control del html y probablemente en algunos casos renderizan un markup que no queremos.

Lo mismo sucede con la testeabilidad, cuando se diseño webforms nunca se pensó en la testeabilidad, es por esto que por defecto es imposible realizar test unitarios sobre ella, y para esto tenemos que integrarnos con otras frameworks e inclusive utilizar otros patrones para separar correctamente las responsabilidades dentro de webforms.

23.-

Luego de esto, hemos llegado a la eterna pregunta ¿webforms o mvC?, cuales son los contexto o escenarios en los cuales deberíamos elegir uno o el otro.

MVC no es un reemplazo de webforms, MS seguirá invirtiendo en webforms y cada versión seguirá mejorando y solucionando sus problemas e inclusive podemos mezclar ambas tecnologías en un mismo proyecto, pero de cierta manera se siente que webforms va quedando obsoleta.

Entonces cuál es la respuesta a la pregunta.

En realidad, y esto no solo es opinión mía sino tambiénde personas muy influyentes y con gran conocimiento, no es necesario formularse la pregunta. Son tantos los beneficios que tiene MVC, todas las cosas que trae, que hace la decisión sea casi obvia.