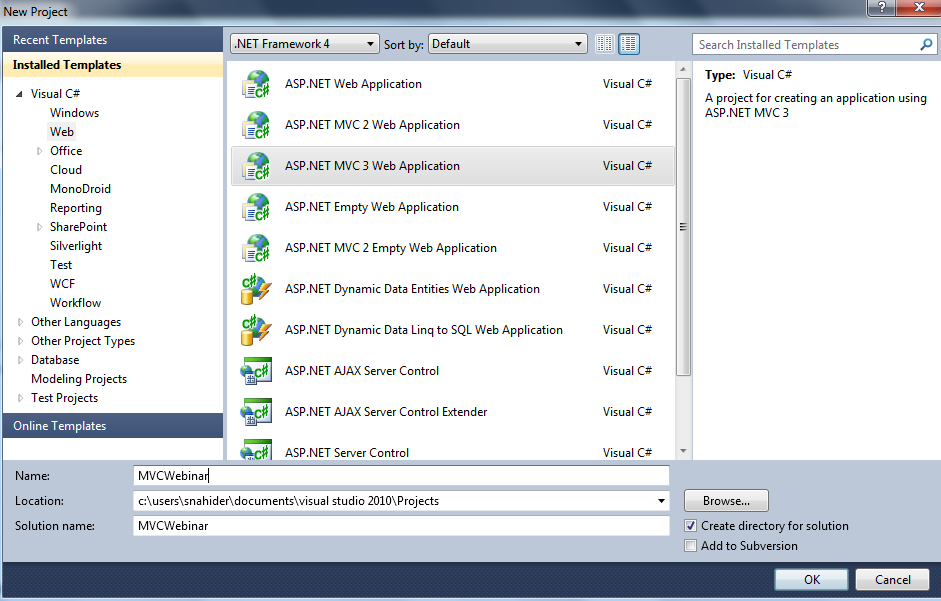
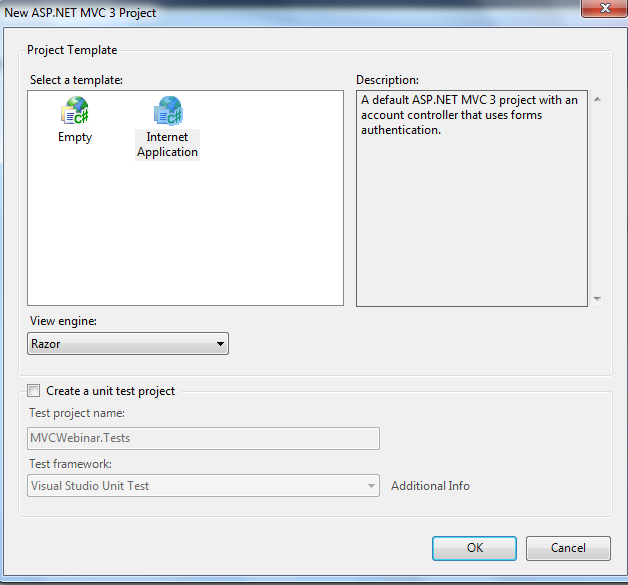


ASP.NET MVC está representado como un nuevo tipo de proyecto dentro del VS. Dependiendo de la versión que queramos utilizar esta puede venir desde caja en el VS o tendremos que instalarla aparte lo cual también es muy muy simple.

Entonces para crear un nuevo proyecto damos File|New|Project



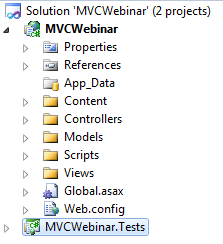
Seleccionamos la versión deseada en este caso la 3 versión 3 que acaba de salir.



Y lo primero que pasa es que nos sale una nueva ventana donde podemos indicarle algunas cosas adicionales. Una de ellas es seleccionar el template que queremos utilizar, el primero es un template de un proyecto completamente en blanco y el otro nos creará un proyecto con algunas cosas ya implementadas como login, registro, una hoja de estilos, etc. Seleccionamos el segundo.

Ahora nos toca seleccionar la View Engine que vamos a utilizar, en esta versión 3 se ha incorporado una view engine completamente nueva que es razor y desde esta versión para adelante es la view engine por defecto. Asimismo la view engine anterior que es WebForms pero también podemos utilizar cualquier otra View Engine como Haml, Spark o Velocity. Lo dejamos en Razor.

Y por último si queremos que nos cree un proyecto para realizar test unitarios en el cuál podemos seleccionar la framework de test unitarios, seleccionamos que queremos test unitarios y dejamos el resto como está.

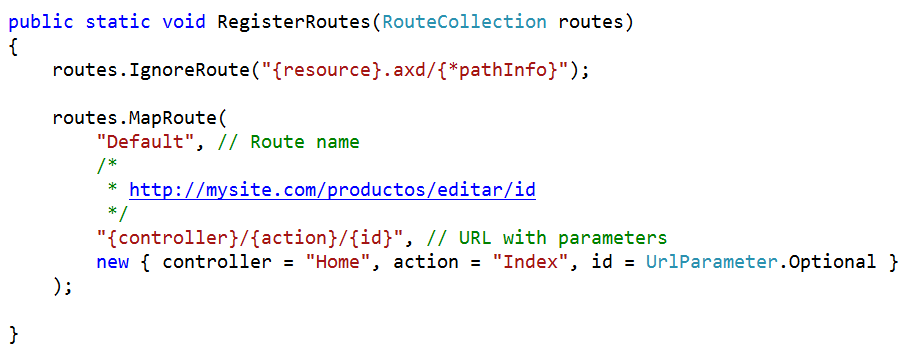


Nos crear una estructura de proyecto con todo lo necesario para ya poder ejecutar la aplicación.

Ahora nos dirigimos a la estructura del proyecto. MVC es muy fuerte en el uso de convenciones y de esta manera facilitar bastante el trabajo de los desarroladores. Uno de los primeros lugares donde podemos observar esto es en la estructura del proyecto donde tenemos por defecto todos estos directorios que nos dicen dónde podemos colocar determinados archivos, por supuesto podemos cambiar estas convenciones pero al seguirlas permiten comenzar de una manera más rápida. Por ejemplo tenemos un directorio para el contenido (imágenes y css), modelos (presentation models u objetos reales de negocio), scripts( que trae varios archivos por defecto entre ellos podemos reconocer a jquery), vistas ( en el cual podemos observar un directorio de vistas por cada controller, ejem: tenemos un controller account y una vista account).

Podemos reconocer algunos elementos que son familiares como el Global.asax, Master.site y el Web.config debido a como dijimos MVC está construido sobre ASP.NET y muchas de las cosas q ya conocemos las vamos a encontrar. Si los abrimos vamos a ver que son muy similares a los ya conocidos. Asi que si eres desarrollador ASP.NET vas a sentirte confortable xq no es tan diferente.

Ahora vamos a hablar de algunos de las partes más importantes de la arquitectura ASP.NET MVC. El primero de ellos es el routing.

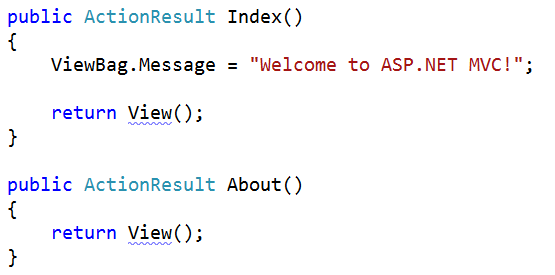


Es muy importante ya que a través de la configuración que este tenga es como se determinará cuál es el controller y el método dentro del controller que tomará el request.

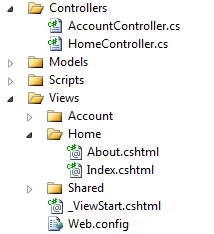
Por defecto se encuentra registrada esta única ruta que nos dice cual es el patrón que deberá tener la URL para identificarse con esta ruta. Por ejemplo para esta ruta si tenemos la siguiente URL(mysite.com…) el path de la URL hace un match con el patrón de la ruta entonces lo que nos dice el patrón es que la primera parte corresponderá al nombre del controller, la segunda a su acción y la tercera a un parámetro id dentro de la acción.

Pero que pasa con la URL sin path del sitio principal, aquí es donde entra la siguiente parte de la configuración, donde nosotros especificamos determinados valores por defecto. En este caso, si no se especificará dentro de la ruta valores para el controller y la acción estos serían Home e Index respectivamente y el id es un parámetro opcional que puede estar en la URL o no. Entonces si realizamos una petición con la url mysite.com está será recibida por el controller Home y la acción Index.

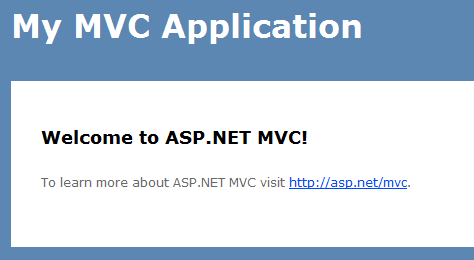
Antes de ver esto en el navegador vamos a revisar los controllers. Como sabemos los controllers son los responsables de responder a las solicitudes que ingresan a la aplicación.



Como vemos un controlador expone determinados métodos públicos, a estos métodos se les denomina acciones y son aquellos que realmente procesan e interactúan con las solicitudes que llegan y devuelven la respuesta adecuada para la misma. Una acción puede devolver múltiples tipos de respuesta, la más común es una vista pero también puede devolver Json, archivos u otras cosas. Por ejemplo aquí la acción Index simplemente está devolviendo una vista como respuesta a la solicitudy además está enviando un mensaje a esta vista.



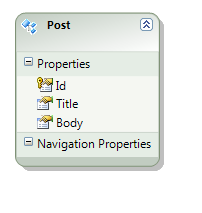
Como dijimos estas vistas se encuentran dentro del directorio views y son como las plantillas que nos renderizan el html tomando la información proveniente del controller. Estas tienen una característica en ASP.NET MVC y es q x convención se encuentran en un directorio con el mismo nombre del controller y se llaman igual que las acción del controller que la invocan. Por ejemplo la vista para la acción index del HomeController se encuentra dentro del directorio Home y se llama Index.cshtml.



Ahora veamos cómo funcionan todos estos en conjunto. Para esto ejecutamos la aplicación y observamos que la solicitud se ha dirigido hacia la acción Index del HomeController y nos ha devuelto su vista correspondiente. Esto xq si recordamos la configuración de las rutas habíamos definido que el HomeController y el Index action eran valores por defecto. En cambio si seleccionamos la pestaña de About veremos que la URL está compuesta por 2 partes en el path que es el Home y el About y que según nuestro patrón dentro de nuestra configuración de rutas la primera parte corresponde al controller y la segunda a la acción.

Aún no hemos visto nada del Modelo, ASP.NET MVC no provee una infraestructura en particular para el modelo, pero podemos utilizar cualquiera de la gran cantidad de opciones que tenemos a nuestra disposición.

Ahora vamos a ver como EF, que es un ORM desarrollado por MS y que nos permite trabajar en el modelo, se integra con ASP.NET MVC.



Para esto, primero que nada vamos a importar una bd a nuestro proyecto, que tiene únicamente una tabla, y luego de esto nos dirigimos a la carpeta models, agregamos un nuevo ítem al proyecto y seleccionamos Ado.net EF (seguimos todos los pasos). Entonces lo que vamos a hacer es mostrar la información de esta tabla en nuestra vista.

Para esto lo primero que hacemos es crear el controller, para lo cuál nos dirigimos a la carpeta controller y seleccionamos agregar controller, le poneos por nombre Posts y nos da la opción de seleccionar si queremos que se nos autogenere acciones para realizar operaciones CRUD en este caso no lo seleccionamos, lo vamos a hacer manualmente. (creamos las acciones index y detalle)

Ahora creamos las vistas para cada acción. (EXPLICAR LA CREACION DE VISTAS)