|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Revisión** | **Fecha** | **Descripción** | **Redactado por** | **Aprobado por** |
|  |  |  |  |  |
| 1.0 | 09/03/2018 |  | SR |  |

.NET MVC EF

|  |  |
| --- | --- |
| El siguiente documento sirve como guía para realizar capacitaciones |  |

|  |  |
| --- | --- |
| Capacitación  09/03/2018  Revisión | logoEway.gif  Cliente:  Proyecto: |

# Contenido

[1 Contenido 2](#_Toc508356270)

[2 Objetivo 3](#_Toc508356271)

[3 Audiencia 3](#_Toc508356272)

[4 Introducción 3](#_Toc508356273)

[5 Pre-requisito: C# 3](#_Toc508356274)

[6 Tutorial 1 – Getting Started 3](#_Toc508356275)

[6.1 Objetivo 3](#_Toc508356276)

[6.2 Conceptos a adquirir 3](#_Toc508356277)

# Objetivo

El objetivo de este documento es uniformizar una base de conocimientos para participar en los proyectos de .NET de la empresa.

# Audiencia

El siguiente documento se encuentra dirigido a desarrolladores.

# Introducción

Los siguientes tutoriales brindan las bases para poder participar en los proyectos de .NET de la empresa. Los conceptos introducidos fueron tomados para definir la metodología de desarrollo de Eway.

La capacitación de .NET de Eway está diseñada para ser completada en forma autónoma por cada desarrollador.

Se deben completar los tutoriales de este documento, desarrollando los ejemplos ahí mencionados, no solo limitándose a leer. Poner en práctica esos conceptos es lo que afirmará los mismos y permitirá aplicarlos en los proyecto.

# Pre-requisito: C#

El desarrollador debe estar familiarizado con el **lenguaje C#**.

El objetivo de esta capacitación no es enseñar el lenguaje, si no que explica la arquitectura MVC y EF.

Los desarrolladores que no conozcan el lenguaje C#, pueden completar previamente cualquier tutorial que encuentren en la web, o si lo desean, tienen disponible el curso oficial de Microsoft en el Teravault.

En la carpeta:

[\\teravault.eway.com.ar\DESARROLLO\Capacitaciones\MCSD - Microsoft Certified Solution Developer](file:///\\teravault.eway.com.ar\DESARROLLO\Capacitaciones\MCSD%20-%20Microsoft%20Certified%20Solution%20Developer)

Los documentos:

* 10266A-EN-Prog\_In\_MS\_VS\_2010\_TrainerHandbook1 - Programming in C(Sharp) with Microsoft Visual Studio 2010.pdf
* 10266A-EN-Prog\_In\_MS\_VS\_2010\_TrainerHandbook2 - Programming in C (Sharp) with Microsoft Visual Studio 2010.pdf

# Tutorial 1 – MVC

Getting Started with ASP.NET MVC 5

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/introduction/>

## Objetivo

* Completar los ejemplos siguiendo paso a paso los puntos del tutorial a lo largo de sus 11 capítulos.
* Lograr crear modelos, controladores y vistas.
* Completar un abm.
* Con este simple tutorial se tiene el 90% de lo que requiere un sistema: Seguridad, menú, ABMs, persistencia. Con muy poco código se tienen un sistema funcional y responsivo.

## Conceptos a adquirir

* Creando un proyecto de tipo MVC, se tiene de entrada un sitio web funcional, con inicio de sesión y menú principal. Las opciones de menú llevan a páginas estáticas que dan ayuda para desarrolladores novatos. El estilo del sitio utiliza las clases de Bootstrap y el sitio es responsive.
* Entender la arquitectura MVC y las responsabilidades de cada capa.
* Usar Scaffold para crear los controladores y vistas.
* Hay 3 estrategias para trabajar con Entity Framework (EF): CodeFirst, DatabaseFirst y ModelFirst. En este caso se exploró la opción CodeFirst y es la que utilizaremos en los proyectos de Eway.
* Siguiendo la estrategia CodeFirst, se definen las clases del modelo. Clases que en principio solo tienen propiedades y representan las entidades de negocio. (Por ejemplo la clase producto para un catálogo)
* MVC crea automáticamente el código para operaciones básicas CRUD (Create, Read, Update, Delete) para una clase de modelo. Con esto se cubre cualquier ABM sencillo. Se agrega al proyecto un “Controller con vistas MVC usando Entity Framework” y se le especifica con qué clase del modelo trabaja. Se crea automáticamente una clase Controller y cinco vistas: Index, Add, Edit, Details y Delete. Todo este código es completamente funcional y ya funciona el ABM para esa clase de modelo.
* A cada acción del controlador se le puede especificar si la operación es por GET o POST. Un método puede estar sobrecargado para tener ambas opciones. Por ejemplo, para Create, una opción GET para obtener el formulario de alta vacío cuando se ingresa la primera vez y una POST cuando se guarda. Se debe usar GET siempre que no se vaya a modificar.
* Las vistas no tienen nada de lógica. A lo sumo un for para cargar varios datos, no más.
* A la vista se le indica con qué clase modelo debe trabajar. Luego en qué parte del código utilizar cada propiedad de esa clase.
* El código de las vistas es similiar a ASP, solo que no queda casi nada de lógica.
* En las vistas se pueden utilizar clases Helper para escribir HTML o escribir HTML directamente. Los helpers serán ampliamente utilizados en los proyectos de Eway, incluso definiremos nuestros propios helpers.
* Las URL para navegar el sitio se arman de la siguiente forma /controlador/accion
* Siempre el código que se ejecuta primero es el del controlador y este decide que vista mostrar.
* Las validaciones básicas se manejan en forma muy sencilla con atributos en la clase modelo (dataannotations). Las validaciones corren tanto en el cliente con jQuery como en el servidor con la clase modelo. Sin tener que modificar nada, la web muestra los mensajes de validación al lado del campo que no valida.
* Para la seguridad se utiliza ASP Identity. Al crear el proyecto si se seleccionó que tenga inicio de sesión ya crea automáticamente una base que tiene usuarios y roles.
* La seguridad se aplica en forma muy sencilla en el controlador. Con el atributo [Autorize] se puede pedir que haya iniciado sesión o que esté en un rol específico para tener acceso ya sea a todo el controlado o a un método en particular.
* Ya vienen incorporados varios mecanismos de seguridad para ataques como pueden ser cross-site scripting u overposting attacks. Se deben adoptar estas prácticas para todos los desarrollos de Eway.
* Para utilizar separador de decimales con coma hay que utilizar un paquete de JQuery especial, Globalize.
* Para los despliegues de base de datos se debe habilitar las migraciones. Esto crea una clase específica para ello y crea un script inicial. Luego cada vez que se modifica el modelo se puede agregar una modificación y crea un nuevo script. Además esta clase tiene un método seed que sirve para cargar la base de datos de prueba.
* Transformar el index en un buscador es muy sencillo agregando parámetros al método del controlador y usándolos para filtrar la consulta.
* Las consultas se hacen con LINQ a los objetos. No se interactúa con la base de datos directamente. La sintaxis de LinkQ es muy similiar a SQL, por lo que es muy sencillo comenzar a utilizarla.
* En el caso de tener que agregar un campo, la base de datos se actualiza sola. Pero las vistas hay que actualizarlas a mano.

# Tutorial 2 – Entity Framework

Getting Started with Entity Framework 6 Code First using MVC 5

<https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/>

## Objetivo

Completar el ejemplo de sistema de Universidad siguiendo paso a paso los puntos del tutorial a lo largo de sus 12 capítulos.

## Conceptos a adquirir

* Cuando un objeto está relacionado a una colección de objetos de otra clase, se utilizan propiedades de navegación, que se definen como virtuales y pueden aprovechar Lazy Load. Esto es que cuando se obtiene el objeto la colección no está cargada, y se carga la primera vez que se la quiere acceder. Esta opción es conveniente y más performante dependiendo del caso. Se puede activar o desactivar Lazy Load.
* Convenciones codefirst:
  + <https://msdn.microsoft.com/en-us/data/jj679962>
* Transacciones: EF maneja transacciones por default y se asegura que todo lo que se mandó a guardar se haga en forma exitosa o se haga un rollback de todo.
  + Información mucho más detallada en:
    - <https://msdn.microsoft.com/en-US/data/dn456843>
* Orden de tablas. Para que las páginas Index se ordenen por columna, se agrega un parámetro en el controlador al método Index. Luego se modifica la consulta LinkQ para que ordene por el parámetro. Para no tener un switch para cada columna se pude usar Dynamic LINQ que permite armar el order by con texto en forma dinámica.
  + Más información de Dynamic LINQ:
    - <https://weblogs.asp.net/scottgu/dynamic-linq-part-1-using-the-linq-dynamic-query-library>
* Paginado de tablas.
  + En el controlador se necesita: using PagedList. Luego simplemente el método index debe recibir el número de página y se usa el método ToPagedList para devolver es página.
  + En la vista: Hay que incluir el paquete de NuGet PagedList.Mvc, que instala un helper para poner toda la botonera del paginado. Esto es opcional, puede utilizarse un html realizado por el/la diseñador/a.
* Connection resiliency:
  + Para que reintente solo ante errores de sql por conectividad.
  + Hay que capturar la excepción RetryLimitExceededException
* Command Interception: sirve para test del punto anterior y logs de todas las consultas realizadas a la base.
  + Mejores prácticas para Logs:
    - <https://docs.microsoft.com/en-us/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/monitoring-and-telemetry#log>
  + Para interceptar las consultas hay que heredar de la clase DbCommandInterceptor y se agrega en Application\_Start con DbInterception.Add, esto hace que EF ejecute la intercepción cada vez que hace una consulta a la base.
* Customización EF
  + EF puede crear las tablas y campos de la base automáticamente. Para esto sigue ciertas reglas.
  + Se puede alterar el comportamiento de EF para tener más control de lo que hace con la base con atributos como [Column] [Key] o [Foreign Key] [DatabaseGenerated]
* Llamadas asincrónicas. Se deben utilizar para liberar recursos del servidor. Un servidor web tiene una cantidad limitada deThreads. Mientras se espera que se completen operaciones I/O se puede liberar el proceso para que se atiendan otras solicitudes utilizando llamadas asincrónicas.
  + En el controlador hay que poner que el método es async. Luego al momento de consultar la base de datos se debe usar await y la versión asincrónica del método en cuestión. Solo hay que usar métodos asincrónicos en el momento que se consulta a la base de datos, por ejemplo con los métodos: tolistasync, findasync, saveasync
  + El código asincrónico no es “Thread safe” no se debe utilizar para hacer operaciones en paralelo.
* Stored procedures: Se puede modificar EF para que utilice SP
* Concurrencia
  + .NET maneja una técnica de concurrencia optimista en que le permite a 2 usuarios modificar la misma entidad. Al segundo que guarda le muestra un cartel indicando que otro usuario modificó esa entidad, le muestra los valores cambiados y si está seguro y vuelve a guardar ahí aplica los cambios del segundo usuario.
  + Para esto utiliza un campo rowversion, si al momento de guardar el valor en la base es diferente que en el objeto que se va a guardar entonces lanza la excepción OptimisticConcurrencyException
* Manejo de errores
  + Utiliza ModelState.AddModelError. El primer parámetro indica el campo. Si se pasa un campo el mensaje se mostrará junto a ese campo. Si no, es un mensaje genérico.
  + Ejemplo:
* catch (RetryLimitExceededException /\* dex \*/)
* {
* //Log the error (uncomment dex variable name and add a line here to write a log.
* ModelState.AddModelError("", "Unable to save changes. Try again, and if the problem persists, see your system administrator.");
* }
* EF permite hacer consultas directamente a la base. Se puede uar DbSet.SqlQuery para devolver objetos del modelo. Se puede usar Database.SqlQuery para devolver otras entidades que no son objetos del modelo. Y se puede usar Database.ExcecuteSqlCommand para comandos que no son select. También se puede deshabilitar el tracking de consultas con el método AsNoTracking
* Publicación en Azure
  + Para publicar en Azure, primero se debe crear la WebApp dentro de Azure y exportar el perfil.
  + Luego haciendo clic derecho en el proyecto de .NET, Publicar e importar el archivo anterior.
  + La implementación se hace con un solo clic. Se implementa tanto el sitio como la base de dato. No es necesario modificar webconfig para cambiar strings de conexión.
* Utilizar data annotations como DataType, DisplayFormat, Display, StringLength, RegularExpression, Column, Required, Key, ForeignKey, DatabaseGenerated.
  + <https://msdn.microsoft.com/library/system.componentmodel.dataannotations.aspx>
* Herencia y EF

# Tutorial 3 – Test y debug

Test and debug in ASP.NET Core

<https://docs.microsoft.com/en-us/aspnet/core/testing/>

## Objetivo

* Aprender a hacer pruebas unitarias y pruebas de integración.
* Testear controladores.

## Conceptos a adquirir

* Crear un proyecto de Test
* Crear pruebas unitarias con xUnit
* Utilización de Assert
* Crear pruebas de integración
* Crear test para controladores
* Utilización de Moqs