ENTITY FRAMEWORK - GUÍA 1: Persistencia de una clase simple usando Code First

NOTA: Para el siguiente tutorial utilizaremos la capacidad de EF de conectarse automáticamente a una instancia de SQLExpress. Por lo tanto verifique en los servicios locales que la instancia de SQL levantada sea la correspondiente a SQLExpress y no la de otra versión.

1. Crear el modelo de dominio:

Crear un nuevo proyecto de Consola, PruebaEF_01, agregarle una clase Estudiante, con atributos **EstudianteId**[1], nombre y edad.

2. Agregar el paquete Entity Framework al proyecto (NuGet):

Desde el explorador de soluciones, agregaremos el paquete de Entity Framework al proyecto. Para esto utilizaremos el gestionador de complementos NuGet incluido con Visual Studio 2012.
Sobre el proyecto, dar botón derecho, y seleccionar **Administrar Paquetes NuGet.**
Si Entity Framework no se encuentra en la opción de "Paquetes instalados" seleccionar la opción "En línea" y utilizar el buscador. Seleccionar el paquete **Entity Framework**, e instalarlo, aceptando las condiciones.

3. Agregar la clase para manejar el contexto (DbContext) y los conjuntos de entidades (DbSet):

Crearemos una clase derivada de *DbContext* que tendrá la responsabilidad de almacenar los cambios, ejecutar los procedimientos almacenados (si existen), obtener resultados desde la BD, manejar las transacciones, etc. Es un puente entre las entidades y la base de datos.
Crear en el proyecto una nueva clase y nombrarla **UniversidadContext**.
- Incluir la siguiente instrucción using:

```
using System.Data.Entity;
```

- Derivar la clase de DbContext:

```
public class UniversidadContext:DbContext
```

- Agregar el/los DbSet necesarios para representar los conjuntos de entidades.
  En el ejemplo, el único conjunto será el que agrupa las entidades Estudiante:

```
using System.Data.Entity;

namespace Prueba01_EntityFramework
{
    public class UniversidadContext:DbContext
    {
        public DbSet<Estudiante> Estudiantes { get; set; }
    }
}
```

4. Instanciar los objetos y persistirlos a través de Entity Framework:

[1] Es importante que la clase tenga un atributo que funcione como identificador único, y que tenga el sufijo "Id".

Probaremos desde el Main () el procedimiento para almacenar los objetos:

```
static void Main(string[] args)
{
    Estudiante p = new Estudiante { Nombre = "Coco", edad = 3 };
    UniversidadContext db = new UniversidadContext();
    db.Estudiantes.Add(p);
    db.SaveChanges();

}
```

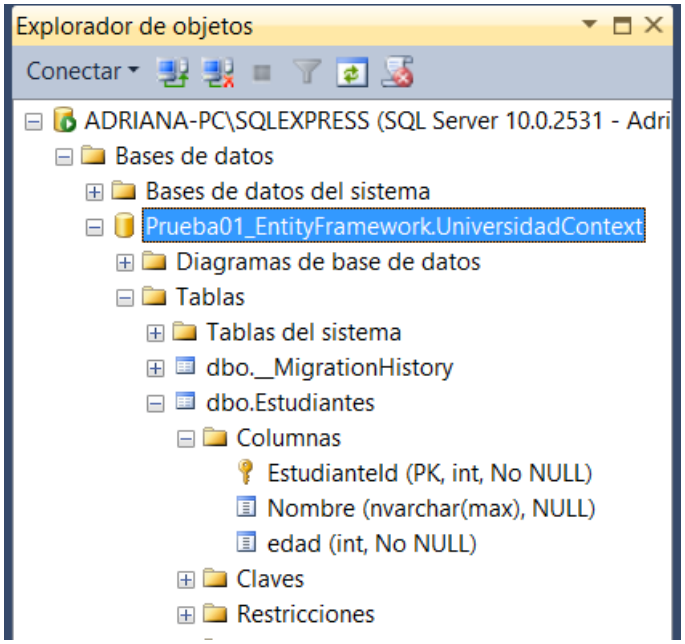
Observar:

i. El objeto se agrega al DbSet igual que a cualquier tipo de colección (Add).
ii. 1. El contexto el responsable de la persistencia a través del método SaveChanges().

5. Ejecutar la aplicación y comprobar que se creó la base de datos.

Al ejecutar la aplicación, se creó la base de datos. Para comprobarlo, abrir el Management Studio y actualizar las bases de datos (en caso de ser necesario). Observar que se creó una base de datos con las siguientes características:

- El nombre es el de la clase del contexto (UniversidadContext) (incluyendo el namespace).
- La tabla tiene el mismo nombre que el DbSet (Estudiantes).
- Las columnas tienen el mismo nombre que los atributos de la clase Estudiante.
- El atributo que en la clase Estudiante estaba sufijado con Id (EstudianteId) quedó como clave primaria.



6. Recuperar datos desde la BD.

Para recuperar la información utilizaremos la sintaxis propia de LinQ. Por el momento no profundizaremos en ella, solamente observaremos que su sintaxis es muy similar a la del lenguaje SQL.

En el Main escribir el siguiente código:

```
{
    Estudiante p = new Estudiante { Nombre = "Tita", edad = 88 };
    UniversidadContext db = new UniversidadContext();
    db.Estudiantes.Add(p);
    db.SaveChanges();

    db.Dispose();

    //Crear otro contexto, solamente para verificar que no se están
    //obteniendo los datos cargados en el contexto anterior sino que
    //estamos accediendo a la BD
    UniversidadContext otraDb = new UniversidadContext();
    //Obtener el DbSet desde la BD
    IEnumerable <Estudiante> losEstudiantes = from Estudiante e
                                              in otraDb.Estudiantes
                                              select e;

    //Una vez obtenidos los estudiantes en una lista, recorrerlos
    foreach (Estudiante  unE in losEstudiantes)
        Console.WriteLine(  unE.EstudianteId + " " + unE.Nombre
                          + "   " + unE.edad );
    Console.ReadKey();
}
```

LinQ

Observar que los datos son recuperados de la BD.

7. Modificar datos

Para actualizar los datos en la BD solamente es necesario obtener la referencia a la entidad a modificar. Para ello usaremos el método Find del DbSet, que permite obtener el elemento dada su clave primaria.

Una vez obtenido, solo es necesario modificar su estado y pedirle al contexto que guarde los cambios. Par probarlo, agregar el código recuadrado y luego de ejecutar verificar en la BD que los datos hayan sido modificados.

```
    //Una vez obtenidos los estudiantes en una li
    foreach (Estudiante  unE in losEstudiantes)
        Console.WriteLine(  unE.EstudianteId + " " + unE.Nombre
                          + "   " + unE.edad );
    Console.ReadKey();

    //Seleccionar un elemento para modificarlo, en el ejemplo el que tiene PK =1
    Estudiante estudianteBuscado = otraDb.Estudiantes.Find(1);
    if (estudianteBuscado != null)
    {
        estudianteBuscado.Nombre = "Nombre cambiado";
        otraDb.SaveChanges();
    }
}
```

8. Modificar datos

Para eliminar una fila, el mecanismo es análogo a la modificación: buscar el elemento a eliminar, quitarlo del DbSet y solicitarle al contexto que actualice los cambios.
Luego de obtener el estudiante (usando find u otra alternativa):

```
if (estudianteBuscado != null)
        {
            otraDb.Estudiantes.Remove(estudianteBuscado);
            otraDb.SaveChanges();
        }
```

9. ¿Cómo continuar?

Practique lo anterior para realizar el ingreso de clientes al sistema del restaurant (en una primera etapa no se registra su dirección).