EF-CODE FIRST: STORE PROCEDURES

INTRODUCCION

Vamos a crear una aplicación de consola sencilla con una clase Persona y realizaremos algunas operaciones utilizando store procedures en lugar de la funcionalidad autogenerada por EF.

SOPORTE PARA ENTITY FRAMEWORK

Ya terminamos con las clases de dominio. Ahora vamos a agregar soporte para EF en el proyecto (Administrador de paquetes NuGet→EntityFramework→Instalar→Aceptar).

CLASES DE NEGOCIO

Vamos a crear la clase Persona y utilizaremos anotaciones para definir el documento como clave. Recuerde incluir usings a las biblioteca: `System.ComponentModel.DataAnnotations`

```
     public class Persona
    {

        public string Nombre { get; set; }

        [Key]
        public virtual string Documento { get; set; }


    }
```

ADMINISTRACIÓN DE LA PERSISTENCIA

Cree una clase de contexto y sobrescriba su constructor para usar su propio string de conexión.

```
    public class PersonaContext:DbContext
    {
        public DbSet<Persona> Personas { get; set; }

        public PersonaContext():base("con"){}
    }


…
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <connectionStrings>
        <add name="con"
connectionString="Server=.\SQLEXPRESS;DATABASE=Personas;Trusted_Connection=true;"
providerName="System.Data.SqlClient"/>
    </connectionStrings>
</configuration>
```

CREAR LA BASE DE DATOS

Vamos a editar el método main para asegurarnos de que la base de datos este creada antes de solicitar la ejecución de un store procedure.

```
static void Main(string[] args){
            using (var db = new PersonaContext())
            {
                // Al usar CodeFirst necesitamos asegurarnos de que el modelo fue
construido antes de abrir la conexión
                // Si usaramos Model First (usando el diseñador) no sería necesario
                db.Database.Initialize(force: false);

             }

}
```

CREAR LOS STORE PROCEDURES

Ahora vamos a crear dos store procedures. Uno para leer todas las personas y otro para borrar una persona.

```
CREATE PROCEDURE [dbo].[DeletePersona]
         (
                  @doc nchar(15)
         )
AS
BEGIN
          delete from Personas where Documento=@doc

END

GO
CREATE PROCEDURE [dbo].[GetAllPersonas]

AS
BEGIN
          select * from Personas
         END
GO
```

EJECUCIÓN DE STORE PROCEDURES

Ahora vamos a modificar el método Main para invocar dos store procedures para leer todas las personas y borrar una Persona usando la clase de contexto.

```
static void Main(string[] args){
    using (var db = new PersonaContext()){
// Al usar CodeFirst necesitamos asegurarnos que el modelo fue construido
// antes de abrir la conexión
// Si usaramos Model First (usando el diseñador) no sería necesario
```

```
                db.Database.Initialize(force: false);
                // Creamos un comando para ejecutar el store procedure
                var cmd = db.Database.Connection.CreateCommand();
                cmd.CommandText = "[dbo].[GetAllPersonas]";
                try{
                    db.Database.Connection.Open();//abrimos la conexion
                    // ejecutamos el sp
                    var reader = cmd.ExecuteReader();
                    // leemos las personas
                    var personas = ((IObjectContextAdapter)db)
                        .ObjectContext
                        .Translate<Persona>(reader, "Personas",
                         MergeOption.AppendOnly);

                    foreach (var item in personas){
                        Console.WriteLine(item.Nombre +" Documento " +item.Documento);
                    }
                    reader.Close();

                    //Ahora vamos a usar un store procedure para borrar
                    Console.WriteLine("Ingrese el documento a borrar");
                    string documento=Console.ReadLine();

                    var p = new SqlParameter("@doc", documento);
                    db.Database.ExecuteSqlCommand("DeletePersona @doc", p);
//si hubiese más parámetros:
//db.Database.ExecuteSqlCommand("exec procedimiento @Nomp1,@Nomp2,Varp1,Varp2);
//si queremos asegurarnos de que el motor ejecute el sp puede incluir el comando exec:
                //db.Database.ExecuteSqlCommand("exec DeletePersona @doc", p);

                 }finally{
                    db.Database.Connection.Close();//cerramos la conexion
                }
                Console.ReadKey();
            }
```

Agregue puntos de interrupción para examinar el código y sus resultados.

TAREAS

Agregue dos nuevos store procedures para insertar y actualizar una persona.