Práctica 3: Model First

Contenido

1. Crear la aplicación

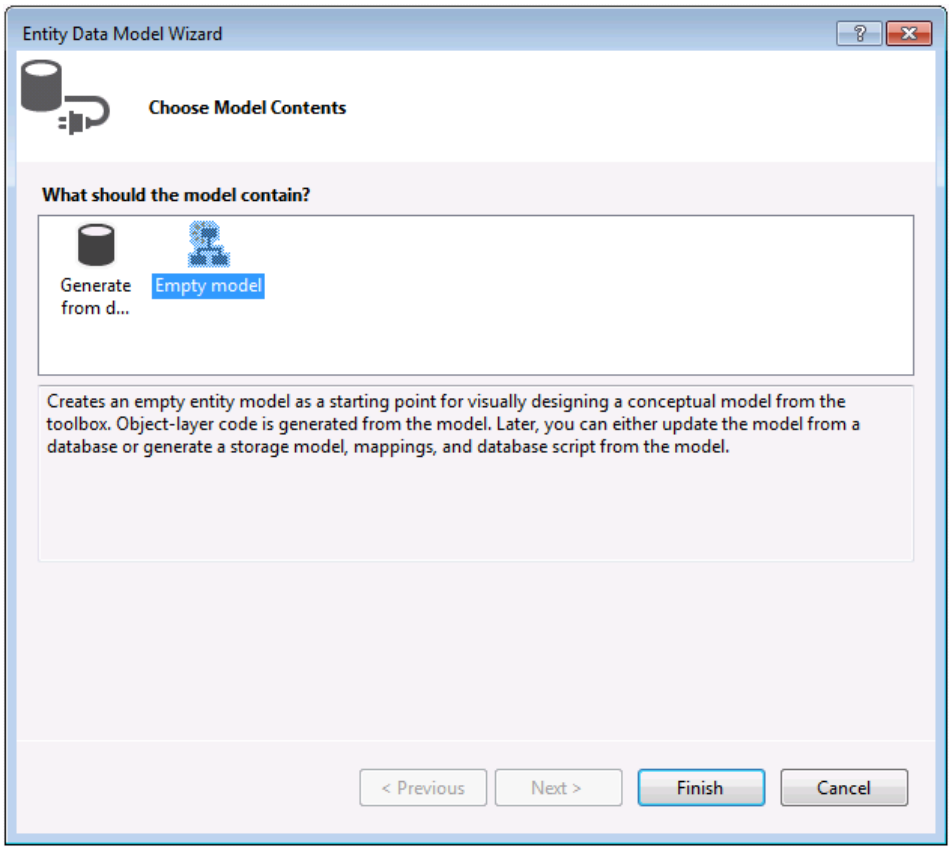
Para mantener la simplicidad, vamos a generar una aplicación de consola básica que use Model First para el acceso a los datos:

- Abra Visual Studio
- **Archivo -> Nuevo -> Proyecto**
- Seleccione **Windows** en el menú de la izquierda y **aplicación de consola**
- Escriba **ModelFirstSample** como nombre
- Seleccione **Aceptar**

2. Crear un modelo

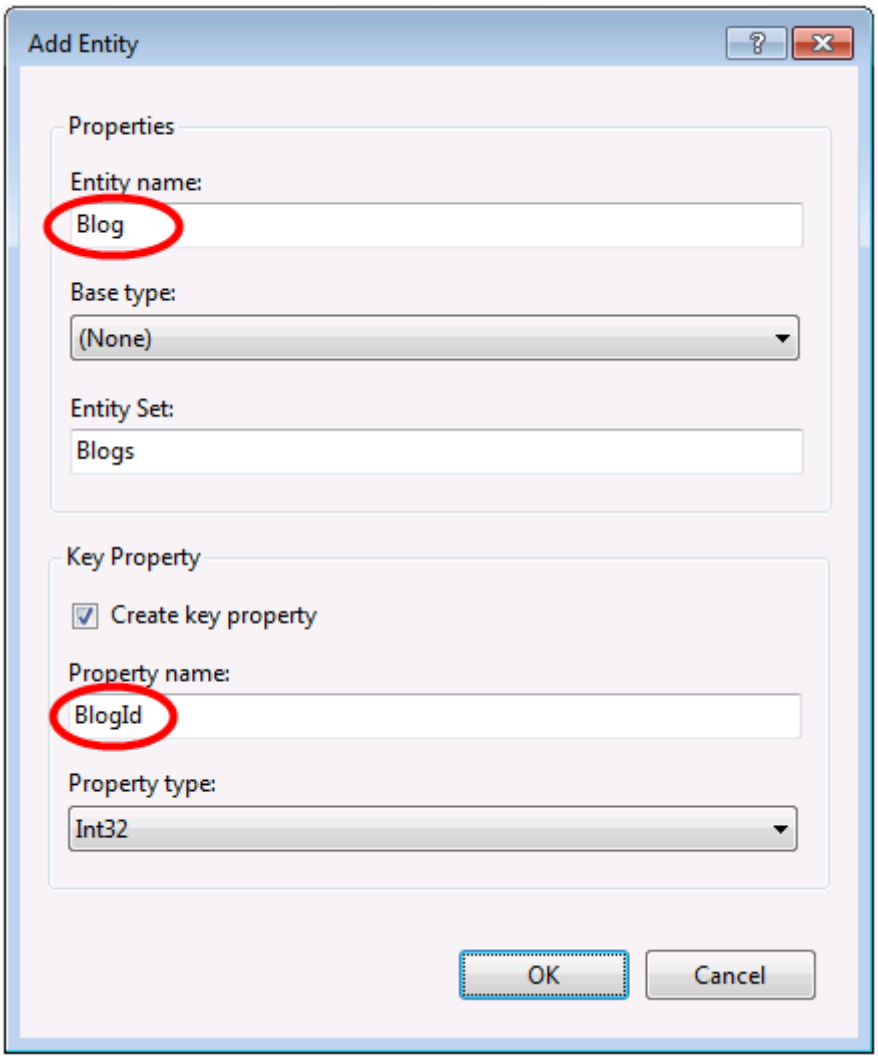
Vamos a usar Entity Framework Designer, que se incluye como parte de Visual Studio, para crear nuestro modelo.

- **Proyecto -> Agregar nuevo elemento**
- Seleccione **Datos** en el menú izquierdo y después **Entity Data Model de ADO.NET**
- Escriba **BloggingModel** como nombre y haga clic en **Aceptar**; de esta forma inicia el Asistente para Entity Data Model
- Seleccione **Modelo vacío** y haga clic en **Finalizar**

- 

Entity Framework Designer se abre con un modelo en blanco. Ahora podemos empezar a agregar las entidades, las propiedades y las asociaciones al modelo.

- Haga clic con el botón secundario en la superficie de diseño y seleccione **Propiedades**
- En la ventana Propiedades, cambie el **Nombre del contenedor de entidades** a **BloggingContext**
  Es el nombre del contexto derivado que se generará automáticamente, el contexto representa una sesión con la base de datos, que nos permite consultar y guardar los datos
- Haga clic con el botón secundario en la superficie de diseño y seleccione **Agregar nueva -> Entidad**
- Escriba **Blog** como nombre de entidad y **BlogId** como nombre de clave y haga clic en **Aceptar**

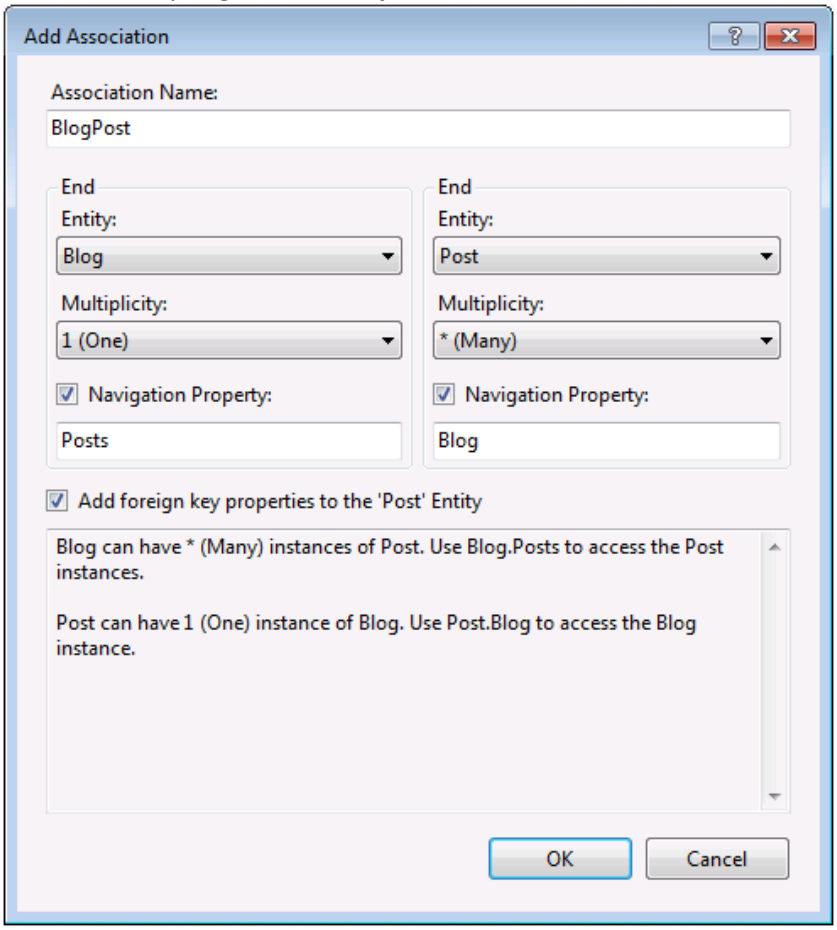
- 

- Haga clic con el botón secundario en la nueva entidad en la superficie de diseño y seleccione **Agregar nuevo -> Propiedad escalar**, escriba **Name** como nombre de la propiedad.
- Repita este proceso para agregar una propiedad **Dirección URL**.
- Haga clic con el botón secundario en la propiedad **Dirección URL** en la superficie de diseño y seleccione **Propiedades**, en la ventana Propiedades, cambie la opción **Acepta valores NULL** a **True**
  Esto nos permite guardar un blog en la base de datos sin asignarle una dirección URL
- Utilizando las técnicas que ha aprendido, agregue una entidad **Post** con una propiedad de clave **PostId**
- Agregue las propiedades escalares **Content** y **Title** a la entidad **Post**

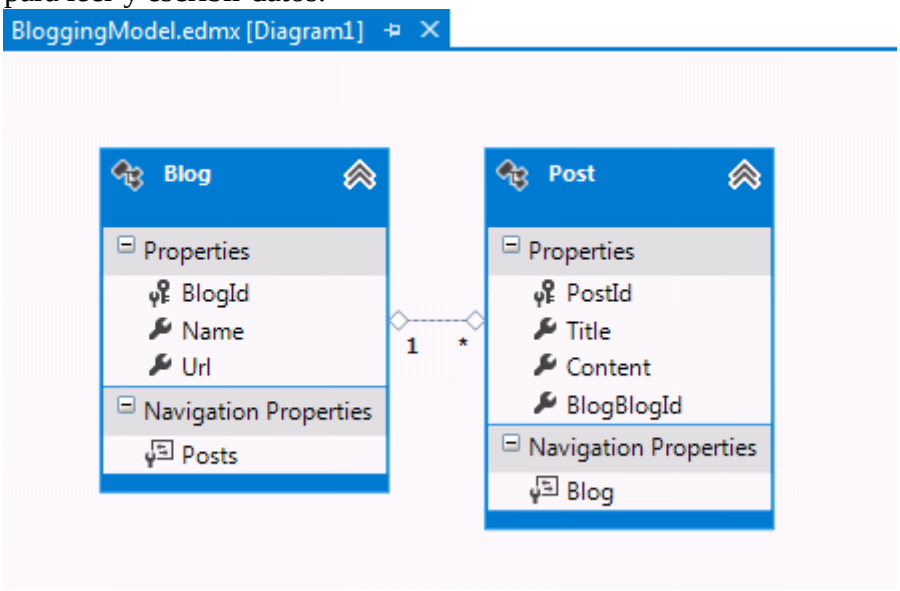
Ahora que tenemos un par de entidades, es hora de agregar una asociación (o relación) entre ellas.

- Haga clic con el botón secundario en la superficie de diseño y seleccione **Agregar nueva -> Asociación**
- Haga que un extremo de la relación señale a **Blog** con una multiplicidad de **Uno** y que el otro extremo señale a **Post** con una multiplicidad de **Muchos**
  Esto significa que un Blog tiene muchos Posts y un Post pertenece a un Blog

- Asegúrese de que el cuadro **Agregar propiedades de clave externa a la entidad 'Post'** esté activado y haga clic en **Aceptar**



Ahora tenemos un modelo simple con el que podemos generar una base de datos y usar para leer y escribir datos.



Pasos adicionales en Visual Studio 2010

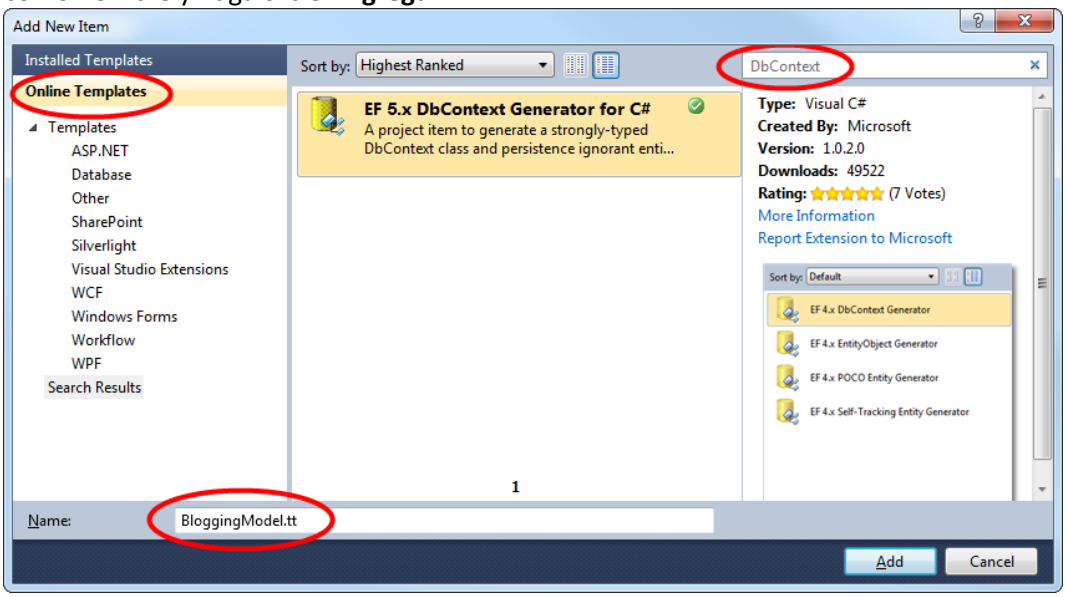
Si está trabajando en Visual Studio 2010, hay algunos pasos adicionales que debe seguir para actualizar a la versión más reciente de Entity Framework. La actualización es importante porque proporciona acceso a una superficie mejorada de la API, que es mucho más fácil de usar, así como a las correcciones de errores más recientes.

Primero, necesitamos obtener la versión más reciente de Entity Framework de NuGet.

- **Proyecto –> Administrar paquetes de NuGet**
  *Si no dispone de la opción **Administrar paquetes de NuGet**, debe instalar la versión más reciente de NuGet*
- Select the **Online**
- Select the **EntityFramework**
- Haga clic en **Instalar**

A continuación, tenemos que intercambiar nuestro modelo para generar el código que usa la API DbContext, que se incluyó en las versiones posteriores de Entity Framework.

- Haga clic con el botón secundario en un punto vacío del modelo en el EF Designer y seleccione **Agregar elemento de generación de código**
- Seleccione **Plantillas en línea** en el menú de la izquierda y busque **DbContext**
- Seleccione el **Generador de DbContext de EF 5.x para C#**, escriba **BloggingModel** como nombre y haga clic en **Agregar**



3. Generar la base de datos

En nuestro modelo, Entity Framework puede calcular un esquema de la base de datos que nos permitirá almacenar y recuperar los datos con el modelo.

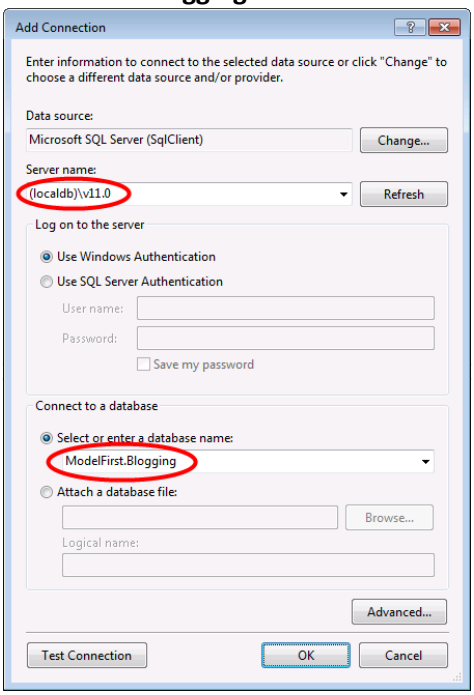
El servidor de bases de datos que se instala con Visual Studio varía en función de la versión de Visual Studio que se haya instalado:

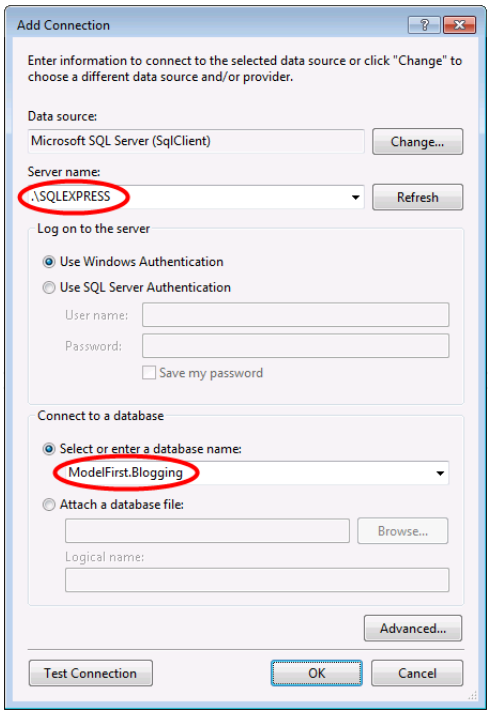
- Si usa Visual Studio 2010, creará una base de datos de SQL Express.

- Si usa Visual Studio 2012, creará una base de datos de LocalDb.

Continuemos y generemos la base de datos.

- Haga clic con el botón secundario en la superficie de diseño y seleccione **Generar base de datos desde modelo**
- Haga clic en **Nueva conexión** y especifique LocalDb (**(localdb)\v11.0**) o SQL Express (**.\SQLEXPRESS**), dependiendo de qué versión de Visual Studio use, escriba **ModelFirst.Blogging** como nombre de la base de datos.



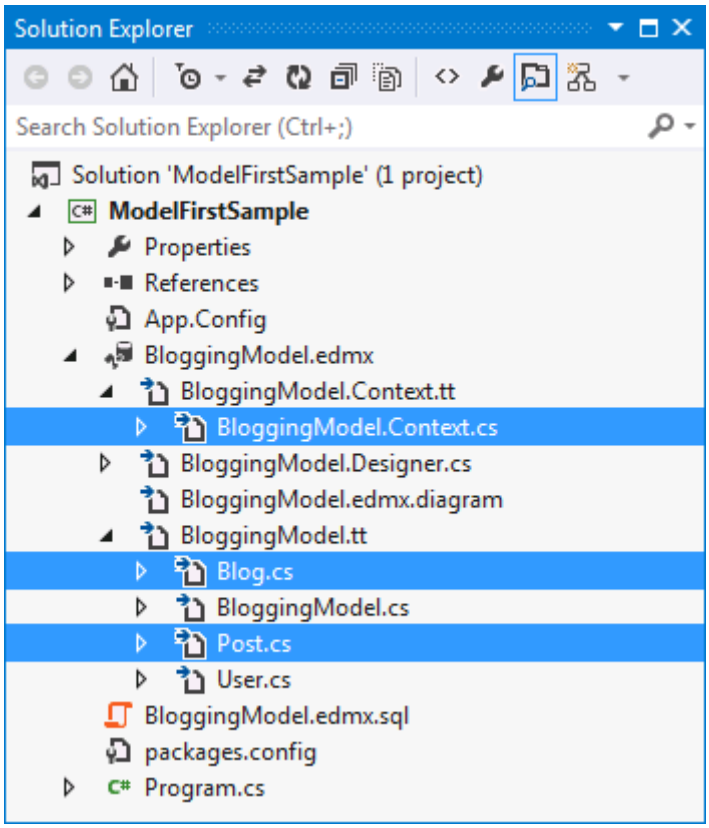


- Seleccione **Aceptar** y se le preguntará si desea crear una base de datos nueva; seleccione **Sí**
- Seleccione **Siguiente** y Entity Framework Designer calculará un script para crear el esquema de la base de datos
- Una vez que se muestre el script, haga clic en **Finalizar** y el script se agregará al proyecto y se abrirá
- Haga clic con el botón secundario en el script y seleccione **Ejecutar**. Se le pedirá que especifique la base de datos a la que conectarse. Especifique **(localdb)\v11.0** o **.\SQLEXPRESS**, según qué versión de Visual Studio use

4. Leer y escribir datos

Ahora que tenemos un modelo, es hora de usarlo para tener acceso a algunos datos. Las clases que vamos a usar para tener acceso a los datos se generan automáticamente según el archivo EDMX.

Esta captura de pantalla es de Visual Studio 2012, si usa Visual Studio 2010, los archivos BloggingModel.tt y BloggingModel.Context.tt estarán directamente debajo del proyecto en lugar de anidados en el archivo EDMX.



Implemente el método Main en Program.cs como se muestra a continuación. Este código crea una nueva instancia de nuestro contexto y la usa para insertar un nuevo blog. Entonces, usa una consulta LINQ para recuperar todos los blogs de la base de datos ordenados alfabéticamente por el título.

```
class Program
{
    static void Main(string[] args)
    {
        using (var db = new BloggingContext())
        {
            // Create and save a new Blog
            Console.Write("Enter a name for a new Blog: ");
            var name = Console.ReadLine();

            var blog = new Blog { Name = name };
            db.Blogs.Add(blog);
            db.SaveChanges();

            // Display all Blogs from the database
            var query = from b in db.Blogs
                        orderby b.Name
                        select b;

            Console.WriteLine("All blogs in the database:");
            foreach (var item in query)
            {
                Console.WriteLine(item.Name);
            }

            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }
    }
}
```

Ahora puede ejecutar la aplicación y probarla.

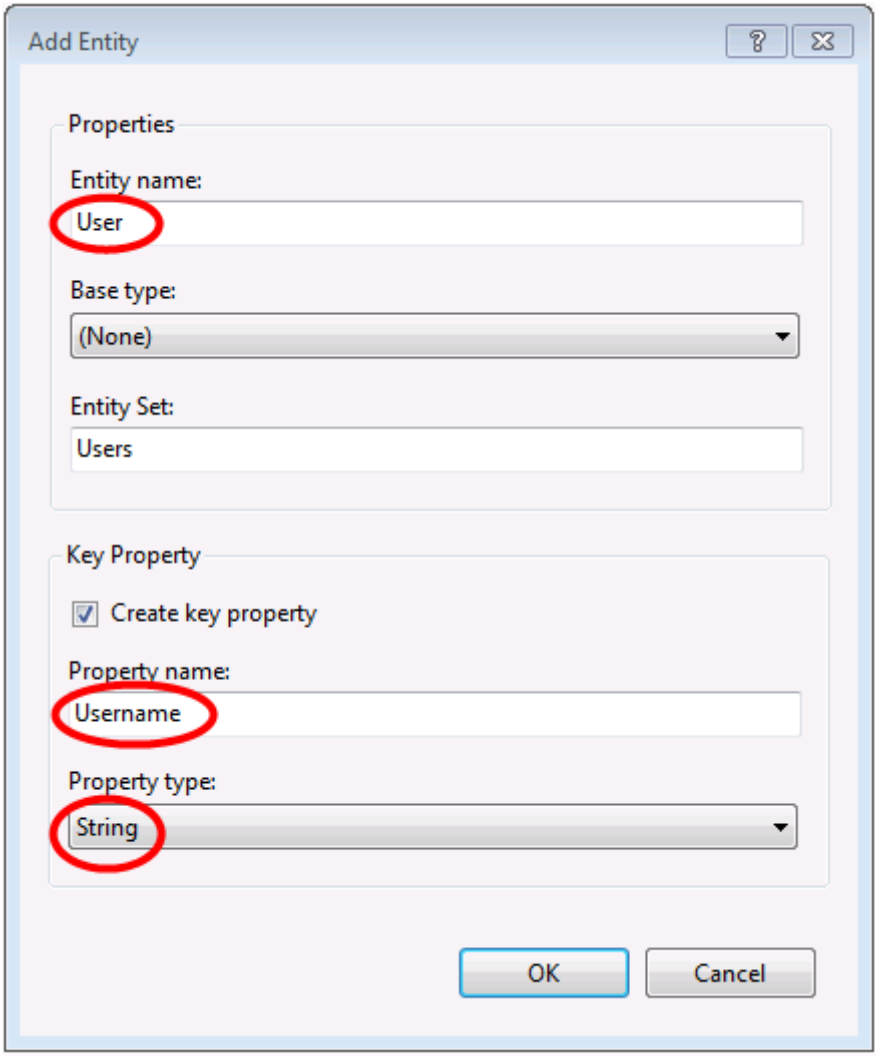
```
Escriba un nombre para el nuevo blog: Blog de ADO.NET
Todos los blogs de la base de datos:
Blog de ADO.NET
Presione cualquier tecla para salir.
```

5. Tratar los cambios en el modelo

Es el momento de realizar algunos cambios en el modelo; cuando realicemos dichos cambios, también necesitaremos actualizar el esquema de la base de datos.

Comenzaremos agregando una nueva entidad de usuario a nuestro modelo.

- Agregue un nuevo nombre de entidad **Usuario** con **Nombre de usuario** como nombre de clave y **String** como tipo de propiedad para la clave



- Haga clic con el botón secundario en la propiedad **Username** en la superficie de diseño y seleccione **Propiedades**, en la ventana Propiedades, cambie la opción **MaxLength** a **50**
  Esto limita los datos que se pueden almacenar en el nombre de usuario a 50 caracteres
- Agregue una propiedad escalar **DisplayName** a la entidad **Usuario**

Ahora tenemos un modelo actualizado y estamos listos para actualizar la base de datos para adaptarse a nuestro nuevo tipo de entidad de usuario.

- Haga clic con el botón secundario en la superficie de diseño y seleccione **Generar base de datos desde modelo**, Entity Framework calculará un script para volver a crear un esquema basado en el modelo actualizado.
- Haga clic en **Finalizar**
- Puede recibir advertencias sobre sobrescribir el script DDL existente y los elementos de almacenamiento y asignación del modelo, haga clic en **Sí** en ambas advertencias
- El script SQL actualizado para crear la base de datos se abre automáticamente
  El script generado quitará todas las tablas existentes y luego volverá a crear el esquema desde el principio. Esto puede funcionar para desarrollo local pero no es

viable para insertar cambios en una base de datos que ya se haya implementado. Si necesita publicar los cambios en una base de datos que ya se haya implementado, tendrá que modificar el script o usar una herramienta de comparación de esquemas a fin de calcular un script de migración.

- Haga clic con el botón secundario en el script y seleccione **Ejecutar**. Se le pedirá que especifique la base de datos a la que conectarse. Especifique **(localdb)\v11.0** o **.\SQLEXPRESS**, según qué versión de Visual Studio use

Resumen

En este tutorial examinamos el desarrollo de Model First, que nos permitió crear un modelo en EF Designer y después generar una base de datos a partir del modelo. Entonces usamos el modelo para leer y escribir algunos datos de la base de datos. Finalmente, actualizamos el modelo y luego reconstruimos el esquema de la base de datos para que coincidiera con el modelo.