Entity Framework 5 on SQLite

Oct 12, 2012

This walkthrough will get you started with an application that uses the Entity Framework (EF) to read and write data from a SQLite database. It is intended to be similar to the Code First to a New Database walkthrough.

There are currently two SQLite providers for EF that I know of: System.Data.SQLite and Devart's dotConnect for SQLite. Devart's provider has a much richer set of features, but it is also a commercial product. In the spirit of FOSS, we will be using the System.Data.SQLite provider for this walkthrough. I encourage you to keep the Devart provider in mind, however, if your project requires that extra level of support.

Create the Application

For simplicity, we will be using a Console Application, but the basic steps are the same regardless of project type.

1. Open Visual Studio
2. Select **File** -> **New** -> **Project…**
3. Select **Console Application**
4. Name the project
5. Click **OK**

Create the Model

For our model, we'll be borrowing pieces from the Chinook Database (a cross-platform, sample database). Specifically, we will be using Artists and Albums.

Add the following two classes to your project.

```
public class Artist
{
    public Artist()
    {
```

```
        Albums = new List<Album>();
    }

    public long ArtistId { get; set; }
    public string Name { get; set; }

    public virtual ICollection<Album> Albums { get; set; }
}

public class Album
{
    public long AlbumId { get; set; }
    public string Title { get; set; }

    public long ArtistId { get; set; }
    public virtual Artist Artist { get; set; }
}
```

Create a Context

In EF, the context becomes your main entry point into the database. Before we define our context though, we will need to install Entity Framework.

1. Select **Tools -> Library Package Manager -> Package Manager Console**
2. Inside the Package Manager Console (PMC) run `Install-Package EntityFramework`

Now, add the context class to your project.

```
    class ChinookContext : DbContext
{
    public DbSet<Artist> Artists { get; set; }
    public DbSet<Album> Albums { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        // Chinook Database does not pluralize table names
        modelBuilder.Conventions
            .Remove<PluralizingTableNameConvention>();
    }
}
```

Install the Provider

In order to connect to SQLite databases, we will need to install an appropriate ADO.NET and Entity Framework provider. Luckily, the provider we're using is available via NuGet.

1. Inside PMC, run `Install-Package System.Data.SQLite.x86`

We also need to register the provider. Open App.config, and anywhere inside the `configuration` element, add the following fragment.

```
<system.data>
  <DbProviderFactories>
    <add name="SQLite Data Provider"
         invariant="System.Data.SQLite"
         description="Data Provider for SQLite"
         type="System.Data.SQLite.SQLiteFactory, System.Data.SQLite" />
  </DbProviderFactories>
</system.data>
```

Add the Database

Unfortunately, System.Data.SQLite does not support creating databases. So instead of letting Code First create our database, we will need to manually add a database to our project. It's a good thing we're using a sample database that's available for SQLite!

1. Download and extract the SQLite version of the Chinook Database
2. Select **Project** -> **Add Existing Item…**
3. Browse to the folder where you extracted the database
4. Select **All Files** from the dropdown next to **File name**
5. Select the Chinook_Sqlite_AutoIncrementPKs.sqlite file
6. Click **Add**
7. Select the file in **Solution Explorer**
8. In **Properties**, set **Copy to Output Directory** to **Copy if newer**

Also, add a connection string to the App.Config that points to the database file. Anywhere inside the `configuration` element, add the following fragment.

```
<connectionStrings>
  <add name="ChinookContext"
       connectionString=
```

```
"Data Source=|DataDirectory|Chinook_Sqlite_AutoIncrementPKs.sqlite"
        providerName="System.Data.SQLite" />
</connectionStrings>
```

Start Coding

Ok, we should be ready to start coding our application. Let's see what artists exist in the database. Inside Program.cs, add the following to `Main`.

```
using (var context = new ChinookContext())
{
    var artists = from a in context.Artists
                  where a.Name.StartsWith("A")
                  orderby a.Name
                  select a;

    foreach (var artist in artists)
    {
        Console.WriteLine(artist.Name);
    }
}
```

Hmm, it looks like one of my favorite bands is missing. Let's add it.

```
using (var context = new ChinookContext())
{
    context.Artists.Add(
        new Artist
        {
            Name = "Anberlin",
            Albums =
            {
                new Album { Title = "Cities" },
                new Album { Title = "New Surrender" }
            }
        });
    context.SaveChanges();
}
```

We can also update and delete existing data like this.

```
using (var context = new ChinookContext())
{
    var police = context.Artists.Single(a => a.Name == "The Police");
    police.Name = "Police, The";

    var avril = context.Artists.Single(a => a.Name == "Avril Lavigne");
    context.Artists.Remove(avril);

    context.SaveChanges();
}
```

Conclusion

Hopefully by now, you have enough information to get started using the Entity Framework with a SQLite database. For many, many more articles on how to use EF, check out our team's official Getting Started page on MSDN.

comments powered by Disqus

Brice's Blog

Senior Software Engineer
bricelam@outlook.com

bricelam
bricelambs

Highlighting some of my more technical adventures