# Database First Using the Entity Framework

# Project Set Up

## Setup Steps for Visual Studio

From within VS (using point and click):

* Create new solution called **WorkingWithDatabases**
* Add console project called **DatabaseFirstEFDemo**.csproj to the solution
* Use NuGet to add a reference to the following packages:
  + Microsoft.EntityFrameworkCore
  + Microsoft.EntityFrameworkCore.SqlServer
  + Microsoft.EntityFrameworkCore.Design
  + Microsoft.EntityFrameworkCore.Tools
* Create the ORM wrapper classes by running the following command in Visual Studio’s Package Manager Console:

Scaffold -DbContext -provider Microsoft.EntityFrameworkCore.SqlServer -connection "Data Source=(local);Initial Catalog=Movies;trusted\_connection=true; ;MultipleActiveResultSets=true;Encrypt=False" -OutputDir "Models" -Project "**DatabaseFirstEFDemo**"

* **Note, the above script assumes you are using a local copy of the Community edition of SQL Serve**r. If you are using the **SQLExpress** edition then the Data Source may need to be altered to “**Data Source=.\SQLExpress**;…” or the server name or IP address. You may also need to modify the connection credentials to specify a username and password.

## Setup Steps for Visual Studio Code

From Command Prompt:

* Prepare solution by typing:

dotnet new solution --name WorkingWithDatabases

* Create console project and add to solution by typing:

dotnet new console --output DatabaseFirstEF-demo

dotnet sln add DatabaseFirstEF-demo

* Install the following packages by typing:

dotnet add package Microsoft.EntityFrameworkCore

dotnet add package Microsoft.EntityFrameworkCore.SqlServer

dotnet add package Microsoft.EntityFrameworkCore.Design

dotnet tool install --global dotnet-ef

* Create the ORM wrapper classes by running the following command in Visual Studio Code’s terminal window:

dotnet-ef dbcontext scaffold "Data Source=(local);Initial Catalog=Movies;trusted\_connection=true;TrustServerCertificate=True; ;MultipleActiveResultSets=true;Encrypt=False" -o "Models"

* **Note, the above script assumes you are using a local copy of the Community edition of SQL Serve**r. If you are using the **SQLExpress** edition then the Data Source may need to be altered to “**Data Source=.\SQLExpress**;…” or the server name or IP address. You may also need to modify the connection credentials to specify a username and password.

## Now do the following regardless of project type:

* Replace the Program.cs file’s content with the following:

using System;

using System.Linq;

using DatabaseFirstEFDemo.Models;

namespace DatabaseFirstEFDemo

{

internal class Program

{

static void Main(string[] args)

{

////Display first 50 biggest grossing movies

using (MoviesContext db = new MoviesContext())

{

var movies = db.Movies.OrderByDescending(m => m.Revenue).Take(50).ToList();

foreach (Movie m in movies)

{

Console.WriteLine($"{m.Title} - {m.Tagline} - {string.Format(new System.Globalization.CultureInfo("en-GB"), "{0:C}", m.Revenue)}");

}

}

Console.WriteLine("\n\nMovies starting with letter S\n");

// Movies starting with letter S

using (MoviesContext db = new MoviesContext())

{

var movies = db.Movies

.Where(m => m.Title.StartsWith("S"))

.ToList();

foreach (Movie m in movies)

{

Console.WriteLine($"{m.Title} - {m.Tagline}");

}

}

Console.WriteLine("\n\nOnly retrieve some columns (via select / projection)\n");

// Only retrieve some columns (via select / projection)

using (MoviesContext db = new MoviesContext())

{

var movies = db.Movies

.Where(m => m.Title.StartsWith("S"))

.Select(m => new { m.Title, m.ReleaseDate, m.Revenue })

.ToList();

foreach (var m in movies)

{

Console.WriteLine($"{m.Title} - {m.ReleaseDate:dd/MM/yyyy}- {string.Format(new System.Globalization.CultureInfo("en-GB"), "{0:C}", m.Revenue)}");

}

}

Console.WriteLine("\n\nCast of Movie with MoveID of 11 (lambda notation)\n");

// Movie\_Cast character and actor names and

using (MoviesContext db = new MoviesContext())

{

var cast = db.Movies

.Join(db.MovieCasts, m => m.MovieId, mc => mc.MovieId, (m, mc) => new { m, mc })

.Join(db.People, mcp => mcp.mc.PersonId, p => p.PersonId, (mcp, p) => new { mcp, p })

.Where(mcp => mcp.mcp.m.MovieId == 11)

.Select(p => new

{

Title = p.mcp.m.Title,

CharacterName = p.mcp.mc.CharacterName,

ActorName = p.p.PersonName

})

.ToList();

foreach (var cm in cast)

{

System.Console.WriteLine($"{cm.Title} - {cm.CharacterName} - {cm.ActorName}");

}

}

Console.WriteLine("\n\nCast of Movie with MoveID of 11 (query notation) \n");

// Movie\_Cast character and actor names and

using (MoviesContext db = new MoviesContext())

{

var cast = (from m in db.Movies

join mc in db.MovieCasts on m.MovieId equals mc.MovieId

join p in db.People on mc.PersonId equals p.PersonId

where m.MovieId == 11

select new { Title = m.Title, CharacterName = mc.CharacterName, ActorName = p.PersonName })

.ToList();

foreach (var cm in cast)

{

System.Console.WriteLine($"{cm.Title} - {cm.CharacterName} - {cm.ActorName}");

}

}

Console.WriteLine("\n\nMovie and size of cast\n");

// Movie and size of cast

using (MoviesContext db = new MoviesContext())

{

var movieCasts = db.MovieCasts

.Join(db.Movies, mc => mc.MovieId, m => m.MovieId, (mc, m) => new { mc, m })

.GroupBy(m => m.m.Title)

.Select(mid => new

{

Title = mid.Key,

SizeOfCast = mid.Count()

})

.ToList();

foreach (var m in movieCasts)

{

System.Console.WriteLine($"{m.Title} - {m.SizeOfCast}");

}

}

//DATA MAINTENANCE

// Creating a new movie

Movie mov = new Movie();

using (MoviesContext db = new MoviesContext())

{

mov.Title = "Hairy Spotter and the Potion of Doom";

mov.Tagline = "Hairy, Germione and Don get up to mischief in a potions class!";

mov.Overview = "Hairy hops off to Cakewalks and gets into a couple of scrapes but triumphs in the end";

mov.Runtime = int.MaxValue;

mov.Homepage = "https://www.youtube.com/watch?v=-\_vqx2BsSj0&t=125s";

db.Movies.Add(mov);

db.SaveChanges();

}

//Run following query in SQL Server Management Studio

//SELECT \* FROM Movie WHERE title Like 'Hairy%'

// Updating Movie tag line

using (MoviesContext db = new MoviesContext())

{

Movie m = db.Movies.Single(m => m.MovieId == mov.MovieId);

m.Tagline += "\*";

db.SaveChanges();

}

// Deleting a movie

using (MoviesContext db = new MoviesContext())

{

Movie m = db.Movies.Single(m => m.MovieId == mov.MovieId);

db.Movies.Remove(m);

db.SaveChanges();

}

}

}

}

* Review the code and try to work out what it’s doing.
* Build and run the code and confirm it behaves as you expected.