# Creating a Connection String and Working with SQL Server LocalDB

## Creating a Connection String and Working with SQL Server LocalDB

The MovieDBContext class you created handles the task of connecting to the database and mapping Movie objects to database records. One question you might ask, though, is how to specify which database it will connect to. You don't actually have to specify which database to use, Entity Framework will default to using [LocalDB](https://docs.microsoft.com/sql/database-engine/configure-windows/sql-server-2016-express-localdb). In this section we'll explicitly add a connection string in the Web.config file of the application.

## SQL Server Express LocalDB

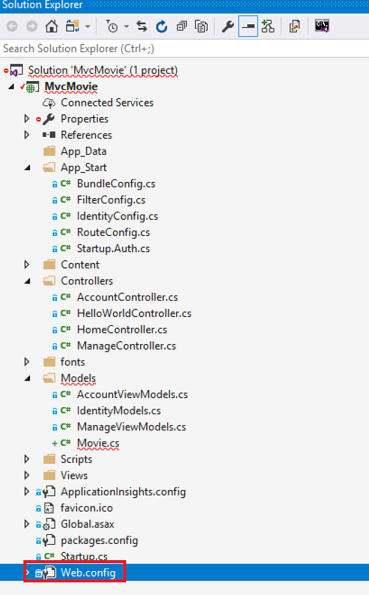
[LocalDB](https://docs.microsoft.com/sql/database-engine/configure-windows/sql-server-2016-express-localdb) is a lightweight version of the SQL Server Express Database Engine that starts on demand and runs in user mode. LocalDB runs in a special execution mode of SQL Server Express that enables you to work with databases as .mdf files. Typically, LocalDB database files are kept in the App\_Data folder of a web project.

SQL Server Express is not recommended for use in production web applications. LocalDB in particular should not be used for production with a web application because it is not designed to work with IIS. However, a LocalDB database can be easily migrated to SQL Server or SQL Azure.

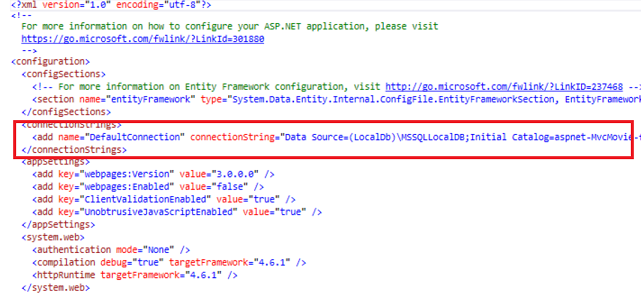
In Visual Studio 2017, LocalDB is installed by default with Visual Studio.

By default, the Entity Framework looks for a connection string named the same as the object context class (MovieDBContext for this project). For more information see [SQL Server Connection Strings for ASP.NET Web Applications](https://msdn.microsoft.com/en-us/library/jj653752.aspx).

Open the application root Web.config file shown below. (Not the Web.config file in the Views folder.)



Find the <connectionStrings> element:



Add the following connection string to the <connectionStrings> element in the Web.config file.

XML

<add name="MovieDBContext"

connectionString="Data Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=aspnet-MvcMovie;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\Movies.mdf"

providerName="System.Data.SqlClient"

/>

The following example shows a portion of the Web.config file with the new connection string added:

XML

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=aspnet-MvcMovie-fefdc1f0-bd81-4ce9-b712-93a062e01031;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\aspnet-MvcMovie-fefdc1f0-bd81-4ce9-b712-93a062e01031.mdf" providerName="System.Data.SqlClient" />

<add name="MovieDBContext" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;Initial Catalog=aspnet-MvcMovie;Integrated Security=SSPI;AttachDBFilename=|DataDirectory|\Movies.mdf" providerName="System.Data.SqlClient" />

</connectionStrings>

The two connection strings are very similar. The first connection string is named DefaultConnection and is used for the membership database to control who can access the application. The connection string you've added specifies a LocalDB database named Movie.mdf located in the App\_Data folder.

The name of the connection string must match the name of the [DbContext](https://msdn.microsoft.com/en-us/library/system.data.entity.dbcontext(v=vs.103).aspx) class.

C#

using System;

using System.Data.Entity;

namespace MvcMovie.Models

{

    public class Movie

    {

        public int ID { get; set; }

        public string Title { get; set; }

        public DateTime ReleaseDate { get; set; }

        public string Genre { get; set; }

        public decimal Price { get; set; }

    }

    public class MovieDBContext : DbContext

    {

        public DbSet<Movie> Movies { get; set; }

    }

}

You don't actually need to add the MovieDBContext connection string. If you don't specify a connection string, Entity Framework will create a LocalDB database in the users directory with the fully qualified name of the [DbContext](https://msdn.microsoft.com/en-us/library/system.data.entity.dbcontext(v=vs.103).aspx) class (in this case MvcMovie.Models.MovieDBContext). You can name the database anything you like, as long as it has the .MDF suffix. For example, we could name the database MyFilms.mdf.

Next, you'll build a new MoviesController class that you can use to display the movie data and allow users to create new movie listings.