

CP2011 Week11 - Practical 10

Topics for today

- Worked example of using Entity Framework to create a SQL database in VS2012
-

This practical is being marked

- 1.5 / 3 marks - for clarifying the task with your practical supervisor, and then demonstrating partially completed code solutions
- 3 / 3 marks - for completing all code solutions to the practical supervisor's satisfaction

Review Networked TicTacToe - Iteration 7

Focus:

- Create a simple prototype UDP networking program consisting of two clients and single server
- A UDP client sends simple request/response messages to the UDP server
- The UDP server broadcasts messages to the UDP clients using UDP multicasting

Verification:

- Check (using program tracing) that a request from a single client gets to the server, and check that the responses get back to the client from the server
- Make the server constantly broadcast messages and check that they are received by both clients

Progress: (written after the end of the prac)

- Many UdpClient programs can send data to a single UdpServer program - you only need a single UdpClient object on the server-side to handle this

Networked TicTacToe - Iteration 8

Focus:

- Create a simple two table database
- Add records to the database using the DbContext object
- Update records using DbContext object

Verification:

- Use LINQ queries and program tracing to verify that the database is correctly storing the records we add into it
- Make the server constantly broadcast messages and check that they are received by both clients

Progress: (written after the end of the prac)

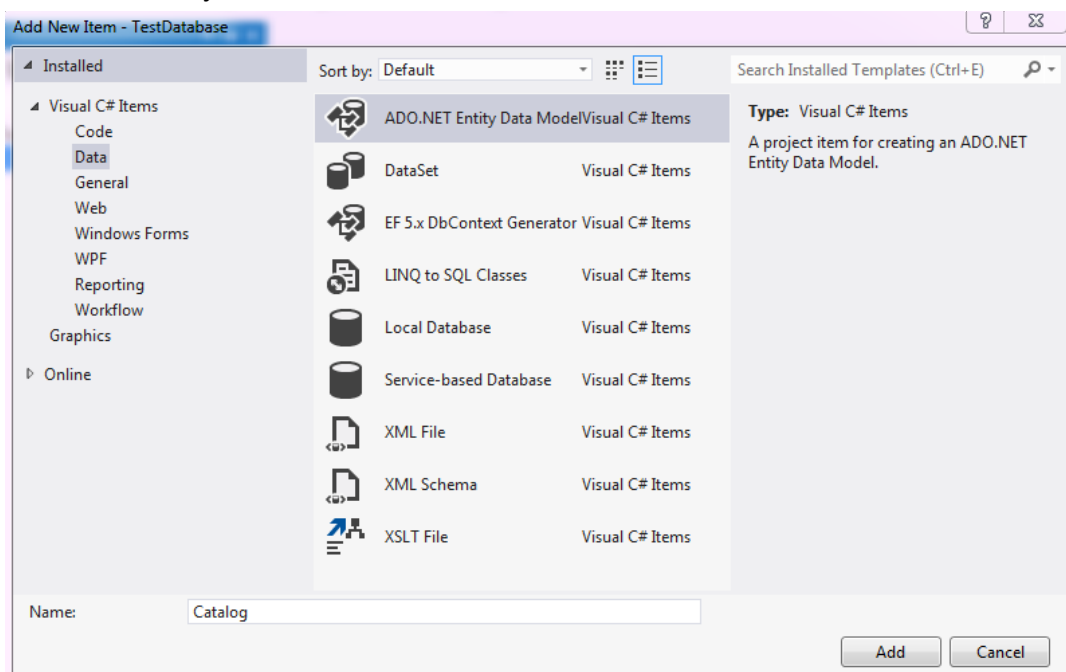
-

Steps

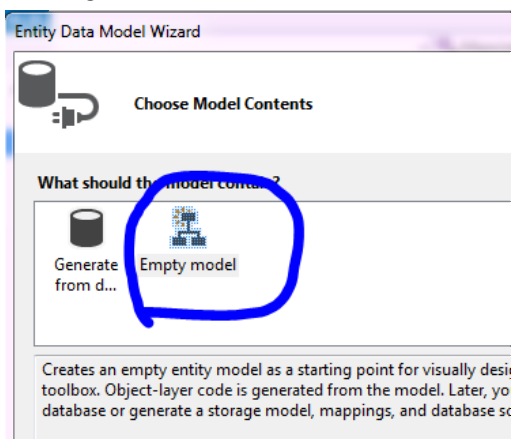
1. Hopefully, the idea of zipping up the project work before doing anything new is

becoming second-nature right!? :))

- a. Rather than implementing the prototype from last week's prac into the TicTacToe project, let's look at prototyping a simple database, since we are nearing the end of semester and it's important to see that now
2. Add a new Console Project to the TicTacToe solution called "TestDatabase"
 - a. We will use this to create an Entity model for a "Music Catalog"
 - b. Our Music Catalog will contain two entities Artist, and Song, such that one Artist makes many Songs
3. In VS2012 IDE, it's worthwhile going into the "View" menu and choosing "Server Explorer" and "SQL Server Object Explorer"
 - a. These are useful for dealing with database related activities in VS2012
4. Next, add a new item to your "TestDatabase" project called "Catalog" which is a ADO.NET Entity model:



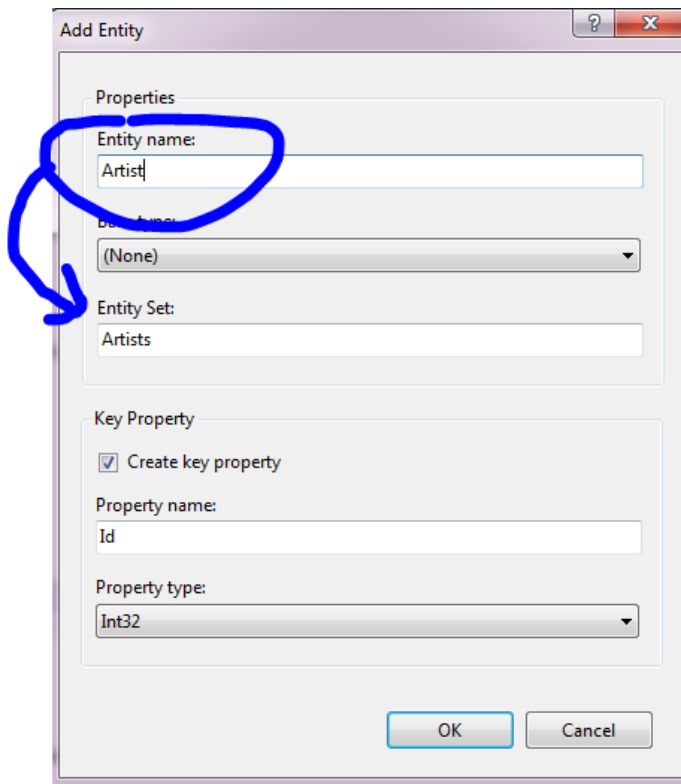
5. In the wizard dialog that appears, make sure you choose "Empty model" - we are starting from scratch in this case:



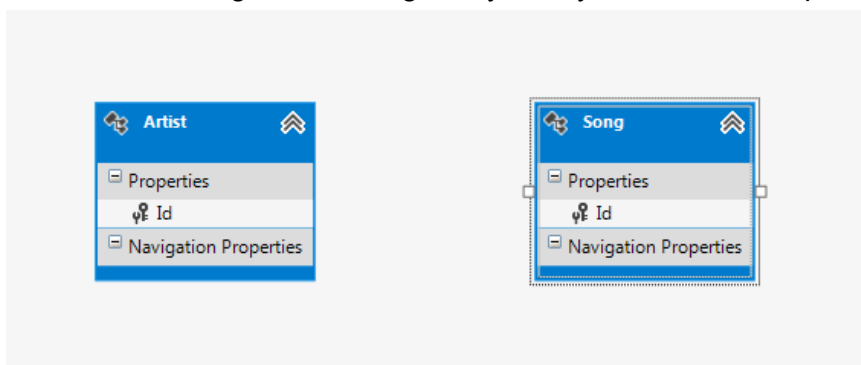
6. This should (after some time...) create a "Catalog.edmx" file
 - a. This is a WYSIWYG designer for Entity Models
 - b. We want to add two entities, one for Artists and the other for Songs, but we

name then in the singular, not plural!

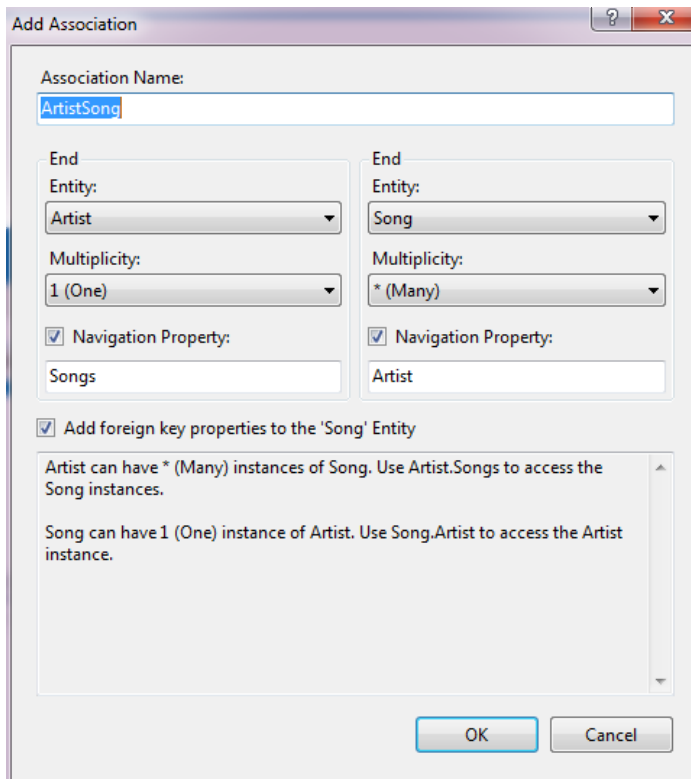
7. Right-mouse clicking anywhere inside the “Catalog.edmx” window and select “Add New... Entity”
 - a. Notice that the typing in “Artist” into the first textbox automatically sets the second textbox to be “Artists”
 - b. That is important! Because an Entity is a single record of data in a table, whereas the Entity Set is the table itself
 - c. Notice also the Key property is defined as “Id” and it an integer - this will be the primary key for the Artists table, you can leave this as it is...



8. Do the same thing for the Song Entity, and you should end up with with this:

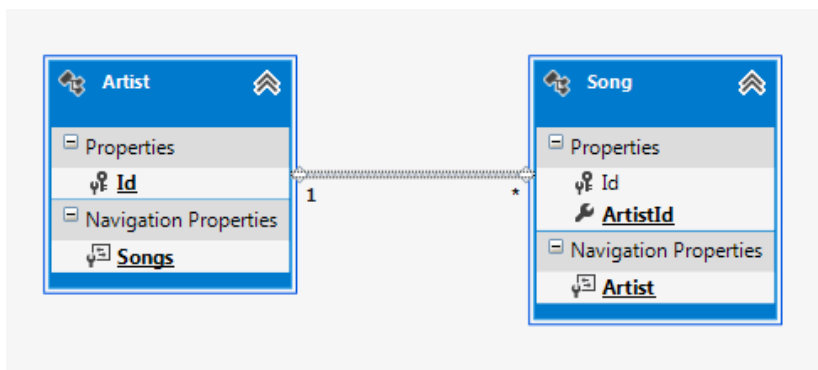


9. Now, we need to setup the one-to-many association between Artist and Song, so right-mouse click on a blank part of the Entity model window and select “Add new... association”
10. The “Add Association” dialog should show, make sure it looks like this:

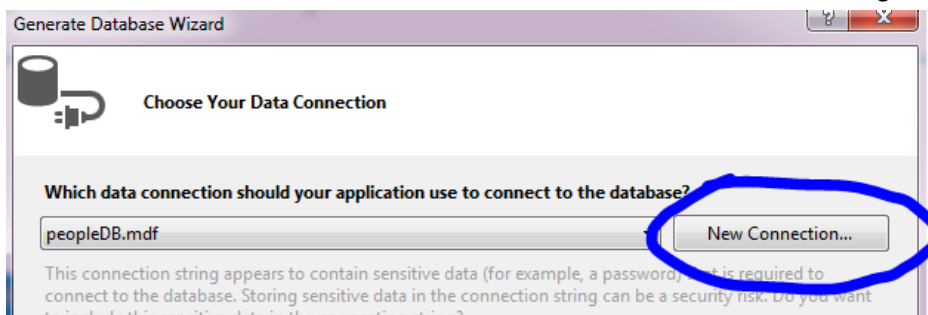


The 'Add Association' dialog box is shown. The 'Association Name' is 'ArtistSong'. The left end is 'Artist' with multiplicity '1 (One)' and a navigation property 'Songs'. The right end is 'Song' with multiplicity '* (Many)' and a navigation property 'Artist'. Both ends have 'Navigation Property' checked. The checkbox 'Add foreign key properties to the 'Song' Entity' is also checked. The text area shows: 'Artist can have * (Many) instances of Song. Use Artist.Songs to access the Song instances.' and 'Song can have 1 (One) instance of Artist. Use Song.Artist to access the Artist instance.'

11. The completed model should be like this:
- As you can see, this is easy to setup...



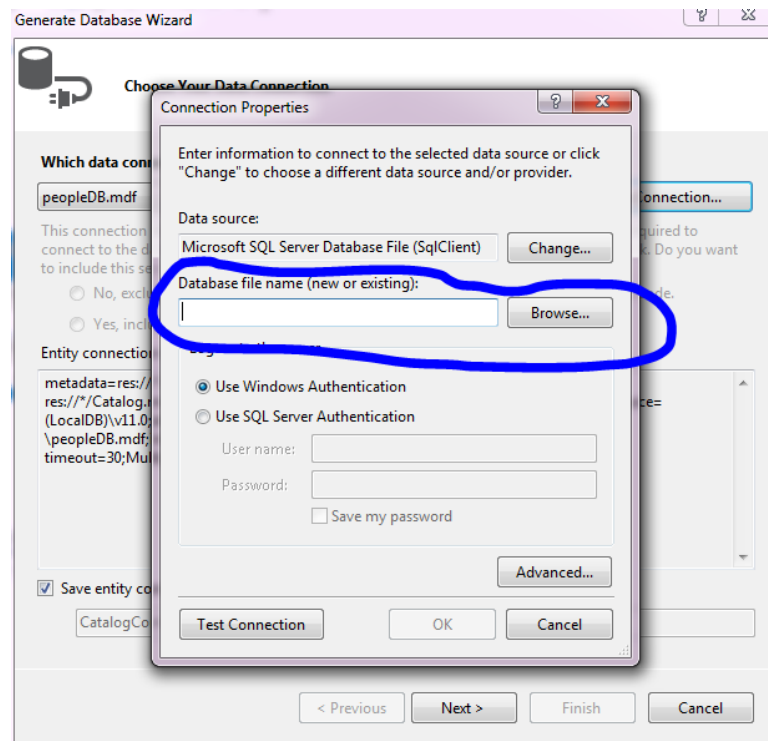
12. Save the entity model and then, right-mouse click on a blank part of the entity model window and select "generate database from model..."
- Amazingly, our Entity model will generate our database without us having to thinking about SQL statements!
 - You should now see the "Generate Database wizard" dialog like this:



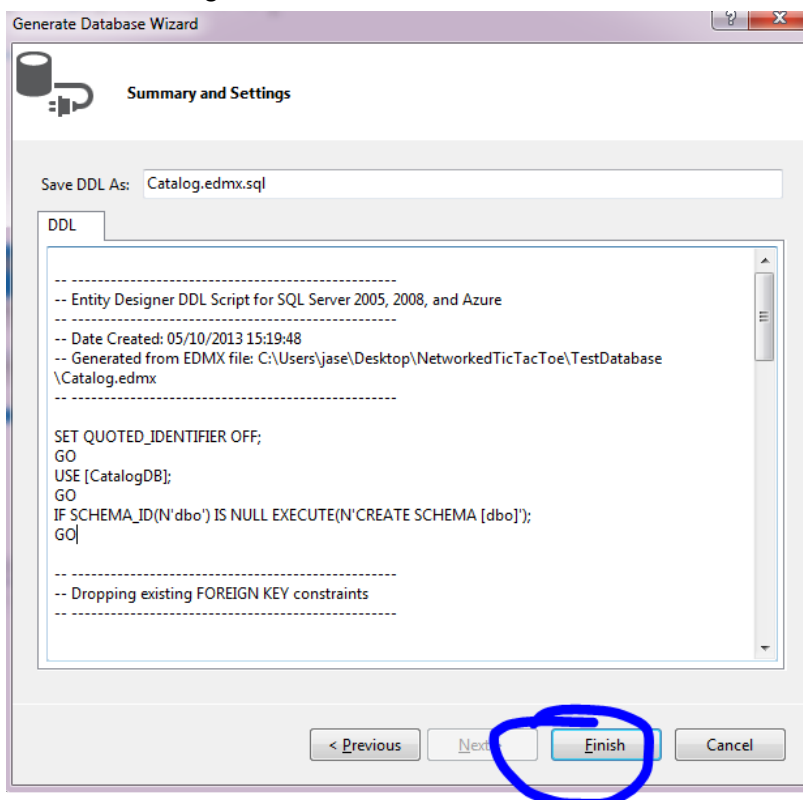
The 'Generate Database Wizard' dialog box is shown. The title is 'Choose Your Data Connection'. The question is 'Which data connection should your application use to connect to the database?'. The list shows 'peopleDB.mdf'. The 'New Connection...' button is circled in blue. A warning message at the bottom states: 'This connection string appears to contain sensitive data (for example, a password). It is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?'

13. Click on "New Connection" and browse for your project folder and create a new file called "CatalogDB" at that location, and then click "Okay"

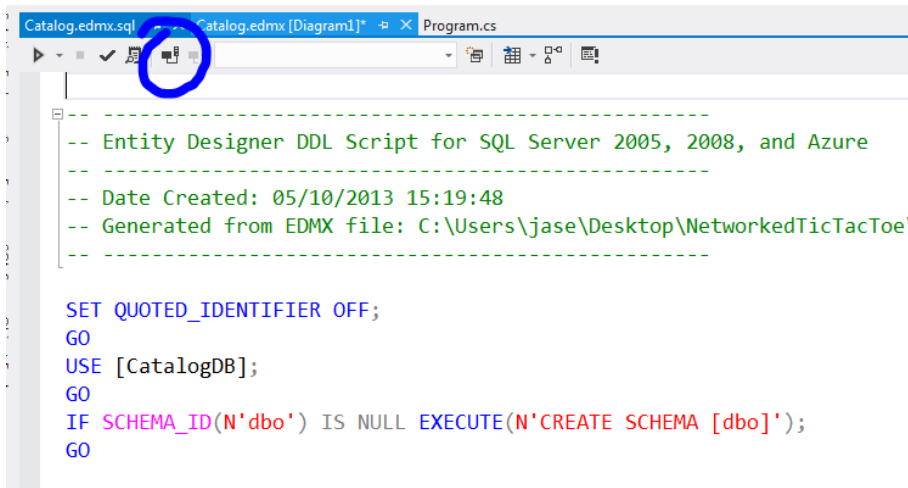
- a. This will allow us to create a new database file - this is just one of the forms a database can take in VS2012



14. Next we click “next” in the “Generate database wizard”, this will automatically create SQL for creating our tables



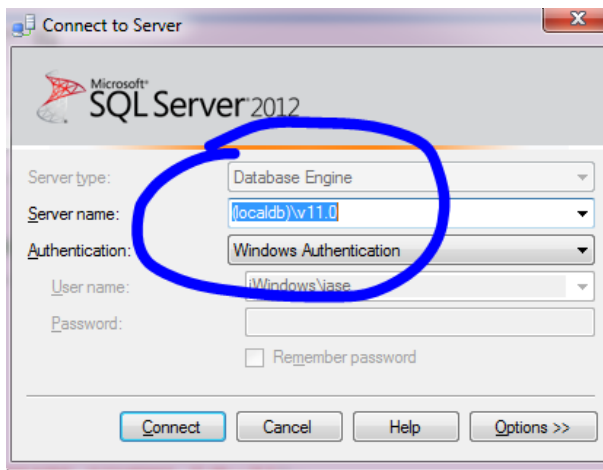
15. Next, click on the “connect to server” button:



```
-- Entity Designer DDL Script for SQL Server 2005, 2008, and Azure
--
-- Date Created: 05/10/2013 15:19:48
-- Generated from EDMX file: C:\Users\jase\Desktop\NetworkedTicTacToe\
--
SET QUOTED_IDENTIFIER OFF;
GO
USE [CatalogDB];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
GO
```

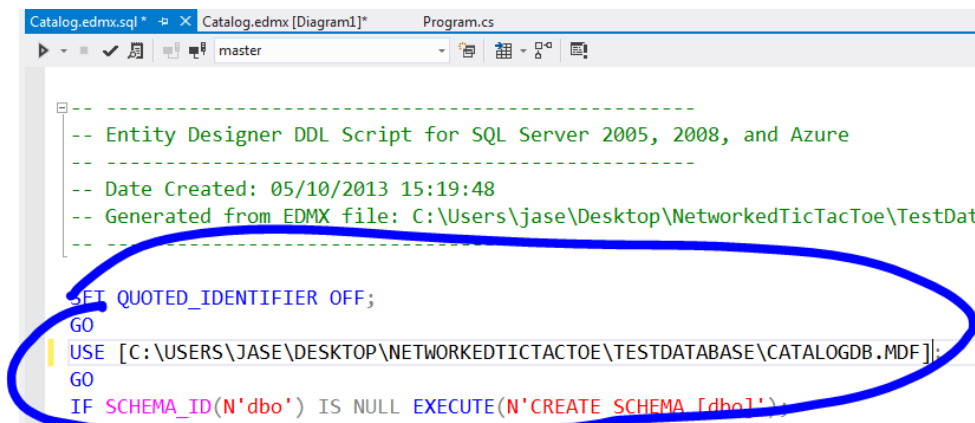
16. This will present you with the “Connect to Server” dialog - into which you must type “(localdb/v11.0)” into the server name

- a. This is the name of the VS2012 IDE’s inbuilt SQL database engine
- b. It used to be called “SQLExpress”, now it’s called “LocalDB”



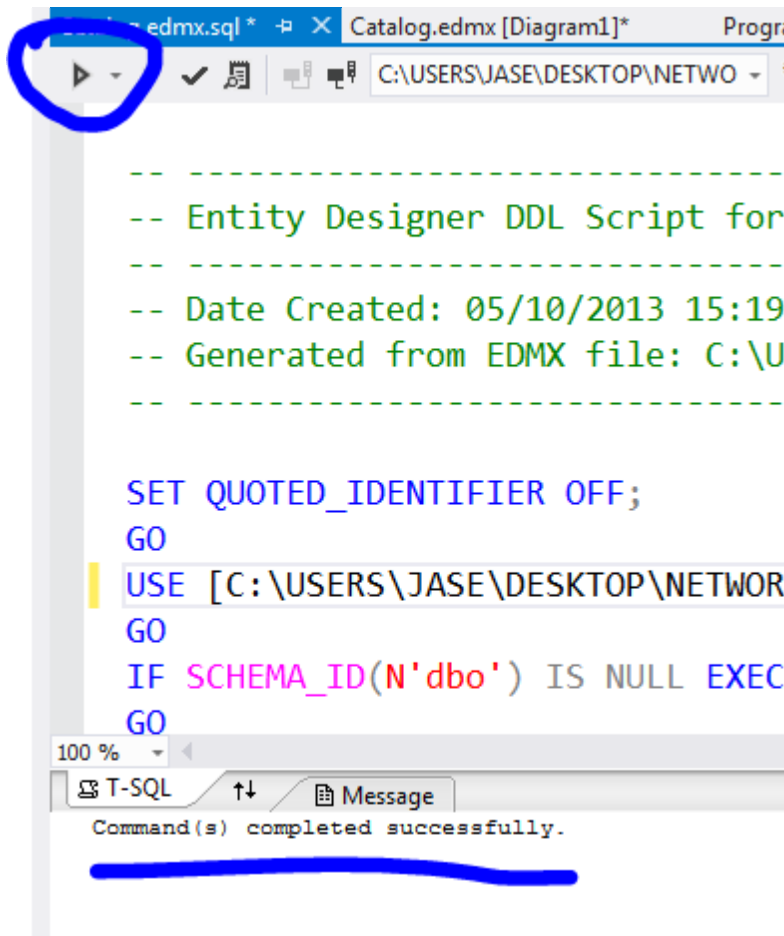
17. Finally, you need to change the “USE” directive in the SQL to be the absolute path to your new “CatalogDB” database file

- a. This is easy to do, just select what is there and CTRL-space to bring up a list of databases the system already knows about

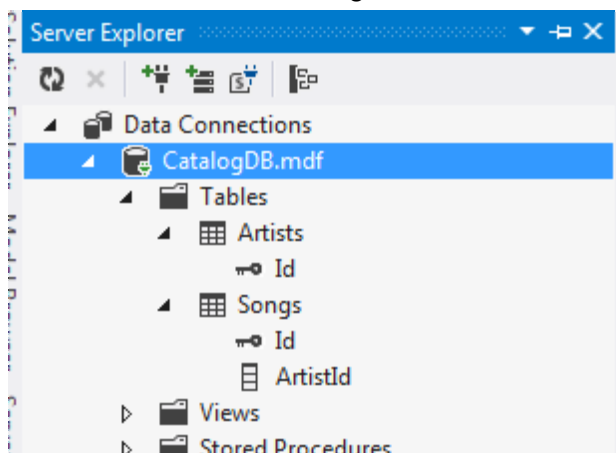


```
-- Entity Designer DDL Script for SQL Server 2005, 2008, and Azure
--
-- Date Created: 05/10/2013 15:19:48
-- Generated from EDMX file: C:\Users\jase\Desktop\NetworkedTicTacToe\TestDat
--
SET QUOTED_IDENTIFIER OFF;
GO
USE [C:\USERS\JASE\DESKTOP\NETWORKEDTICTACTOE\TESTDATABASE\CATALOGDB.MDF];
GO
IF SCHEMA_ID(N'dbo') IS NULL EXECUTE(N'CREATE SCHEMA [dbo]');
```

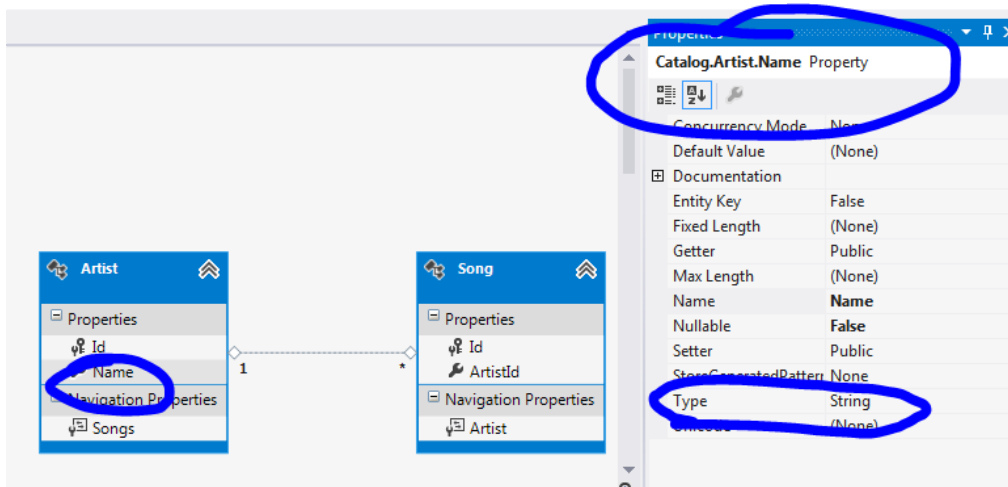
18. Click on the SQL run button and this will setup the tables in the database file for you



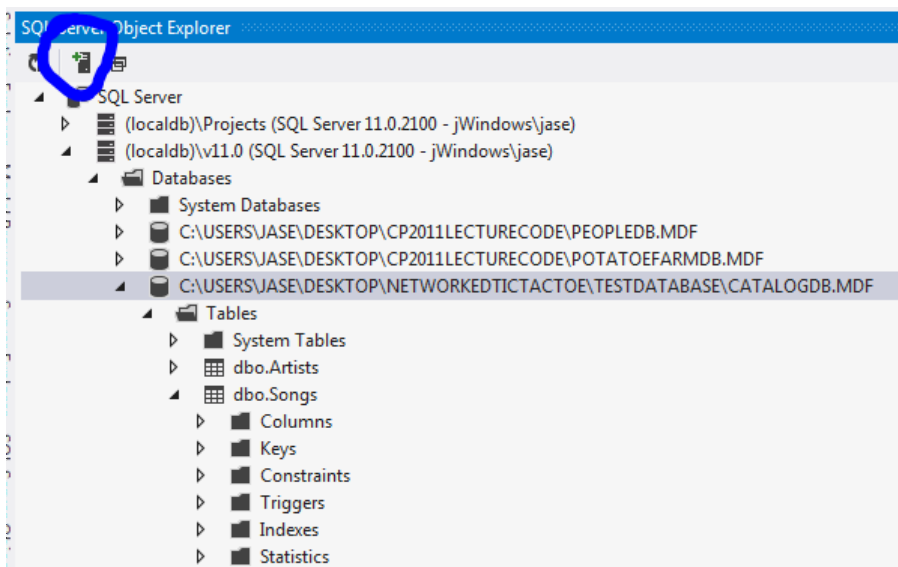
19. Now, let's verify that the database was setup okay, open the "Server Explorer" view and look inside the CatalogDB data connection, you should see:



20. Now we show add some properties to the Entities, an Artist needs a Name (string) and Country (string), a Song needs a Title (string) and Duration (integer).
- Right-mouse click on an entity and select "Add new scalar property"
 - The default type is string, but if you need to change the type, use the properties view



21. Now, re-generate the database from the entity model using the procedure from step 12 (although, you won't need to add connection again... but you will need to adjust the path to the database as discussed before)
22. You might also like to verify that the database has been created correctly using the "SQL server object explorer" view



23. In "Server Explorer" view, right-mouse click on the "Artists" table and select "Show table data"
 - a. This allows you to enter hard-coded data into the table
 - b. Perhaps add in a few artists - don't worry about setting the Id values though

dbo.Artists [Data]			
Catalog.edmx.sql			
Catalog.edmx [Di]			
Max Rows: 1000			
	Id	Name	Country
	1	Lindsey Stirling	USA
	2	Reggie Watts	Outerspace
	3	Prince	The80s
▶*	NULL	NULL	NULL

24. Now, let's verify that we can access the database from code using LINQ

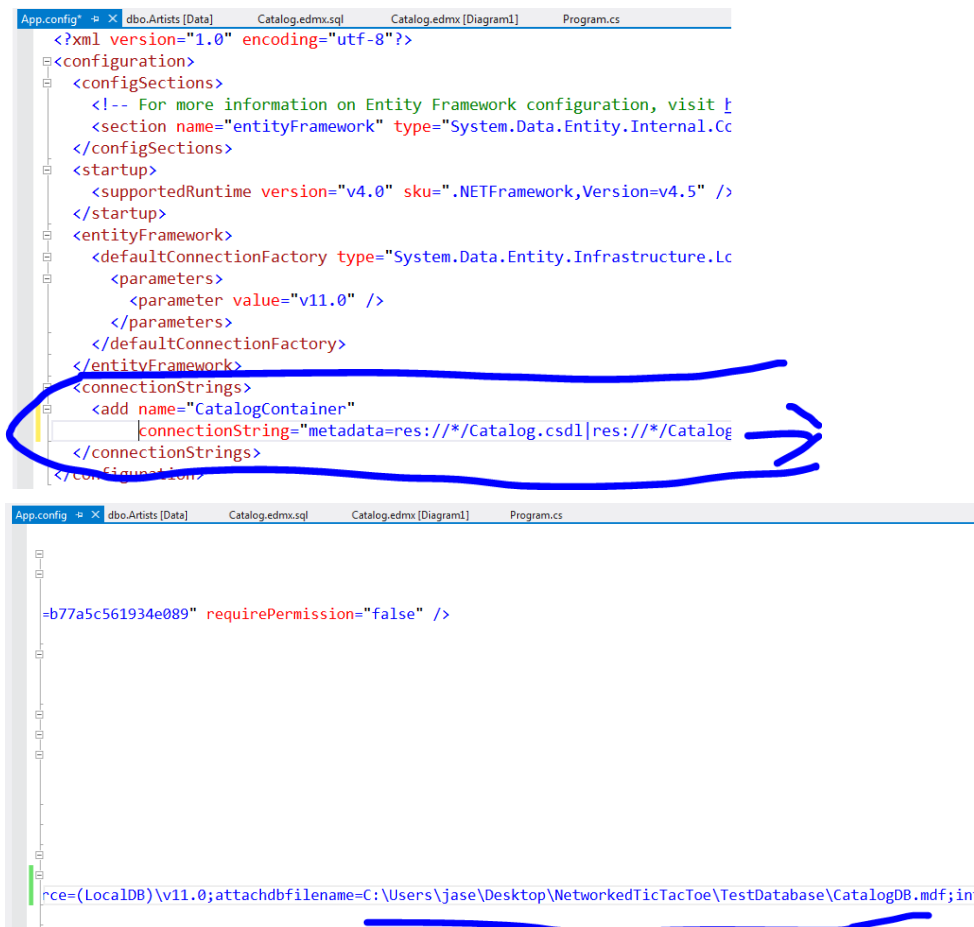
- a. Remember, LINQ is a declarative syntax included in C# to make it easier to access collections of data (arrays, lists, databases, ...)
- b. Try executing the following LINQ query in the Program class for the TestDatabase project

```
static void Main(string[] args)
{
    using (var context = new CatalogContainer())
    {
        var query = from artist in context.Artists select artist.Name;

        foreach (var result in query)
        {
            Console.WriteLine(result);
        }
    }
}
```

25. This will crash the program :) That's because VS2012 IDE created an XML configuration file called "App.Config" for your project, and the connection string is broken

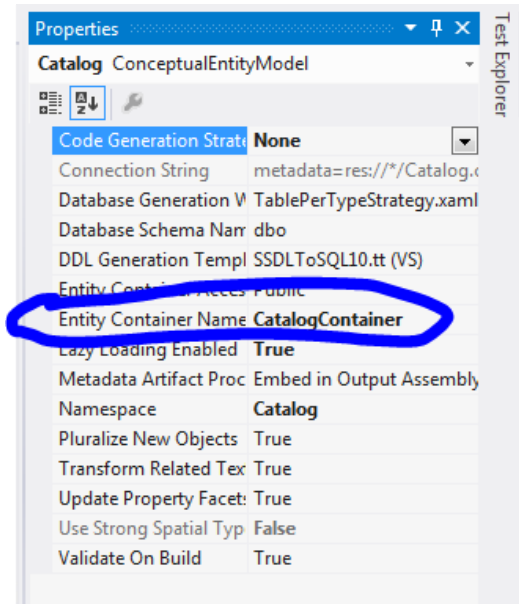
- a. It's important to see this now, as it is a source of confusion!
- b. A "fix" is to use the absolute database pathname in the configuration



26. Now, let's try adding data into the database using the DbContext object

- a. Notice that the name of our DbContext object is "CatalogContainer" - because

- we create the entity model using "Catalog.edmx" as the name
- b. This can be changed / seen in the properties for the Entity model:



27. Now, in the Program class, we can add new records to the database using the following code:

- a. use the "server explorer" to look at the table contents and verify that the table has changed

```
var artists = from artist in context.Artists select artist;
var song = new Song()
{
    Title = "Eye of the tiger",
    Duration = 1000,
    Artist = artists.First()
};
context.Songs.Add(song);
context.SaveChanges();
```

28. Finally, try updating the name of an artist in the Artists table and verifying that it works, try something like this:

- a. Make sure you place this code before the SaveChanges() method call

```
context.Artists.First().Country = "USA";

foreach (var artist in context.Artists)
{
    if (artist.Name == "Prince")
    {
        artist.Name = "Daivd Bowie";
    }
}
```