Entity Framework is an ORM

Object Relational Mapper

Tables Classes

Employee Employee

Id id

Name name

address

pub;ic actionResult Index(int rn , string name)

{

}

Limitations in ADO.Net

1. There is no mapping between column names & class properties
2. Queries are not checked at compile time , they are checked at runtime
3. Sql Injection issue is there
4. No intellisense

SqlConnection

SqlCommand

SqlDataReader

ADO.Net has 2 Architectures

Connected & Disconnected

In connected , when you establish connection from FE to BE, it will remain connected until you close it

SqlConnection

SqlCommand

SqlDataReader

In disconnected , connection is opened & closed automatically

We use DataSet , DataAdapter

In connected , when you establish connection from FE to BE, it will remain connected until you close it

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

namespace ConsoleApp6

{

class Program

{

static void Main(string[] args)

{

SqlConnection con = new SqlConnection("data source=LAPTOP-53S2KQS8;initial catalog=A;integrated security=true");

SqlCommand command = new SqlCommand("Select \* from Employee", con);

con.Open();

SqlDataReader reader = command.ExecuteReader();

if(reader.HasRows)

{

while(reader.Read())

{

Console.WriteLine(reader["id"].ToString() + " " + reader[1]);

}

}

reader.Close();

con.Close();

command.Dispose();

con.Dispose();

}

}

}

Disconnected

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SqlClient;

using System.Data;

namespace ConsoleApp6

{

class Program

{

static void Main(string[] args)

{

SqlConnection con = new SqlConnection("data source=LAPTOP-53S2KQS8;initial catalog=A;integrated security=true");

SqlCommand command = new SqlCommand("Select \* from Employee", con);

SqlDataAdapter ada = new SqlDataAdapter(command);

DataSet ds = new DataSet();

ada.Fill(ds);

foreach(DataRow dr in ds.Tables[0].Rows)

{

Console.WriteLine(dr["id"].ToString() + " " + dr["name"]);

}

command.Dispose();

con.Dispose();

}

}

}

Entity Framework : ORM

There are 3 approaches

Code First >Add classes , we get tables in database

Database First Approach > Add tables > we get classes

Model First Approach > You add model, you get tables in BE & classes in FE

In Code First Approach

1. Add a class , Add some properties

class Batch

{

public int Id { get; set; }

public string Name { get; set; }

public string Course { get; set; }

public int Size { get; set; }

}

1. Add a class BatchDbContext inherit this class DbContext (System.Data.Entity)
2. You add a property DbSet<Batch>

class BatchDbContext : DbContext

{

public DbSet<Batch> Batches { get; set; }

}

1. Add an entry in config file, with the same name BatchDbContext : DbContext

Add connection String there

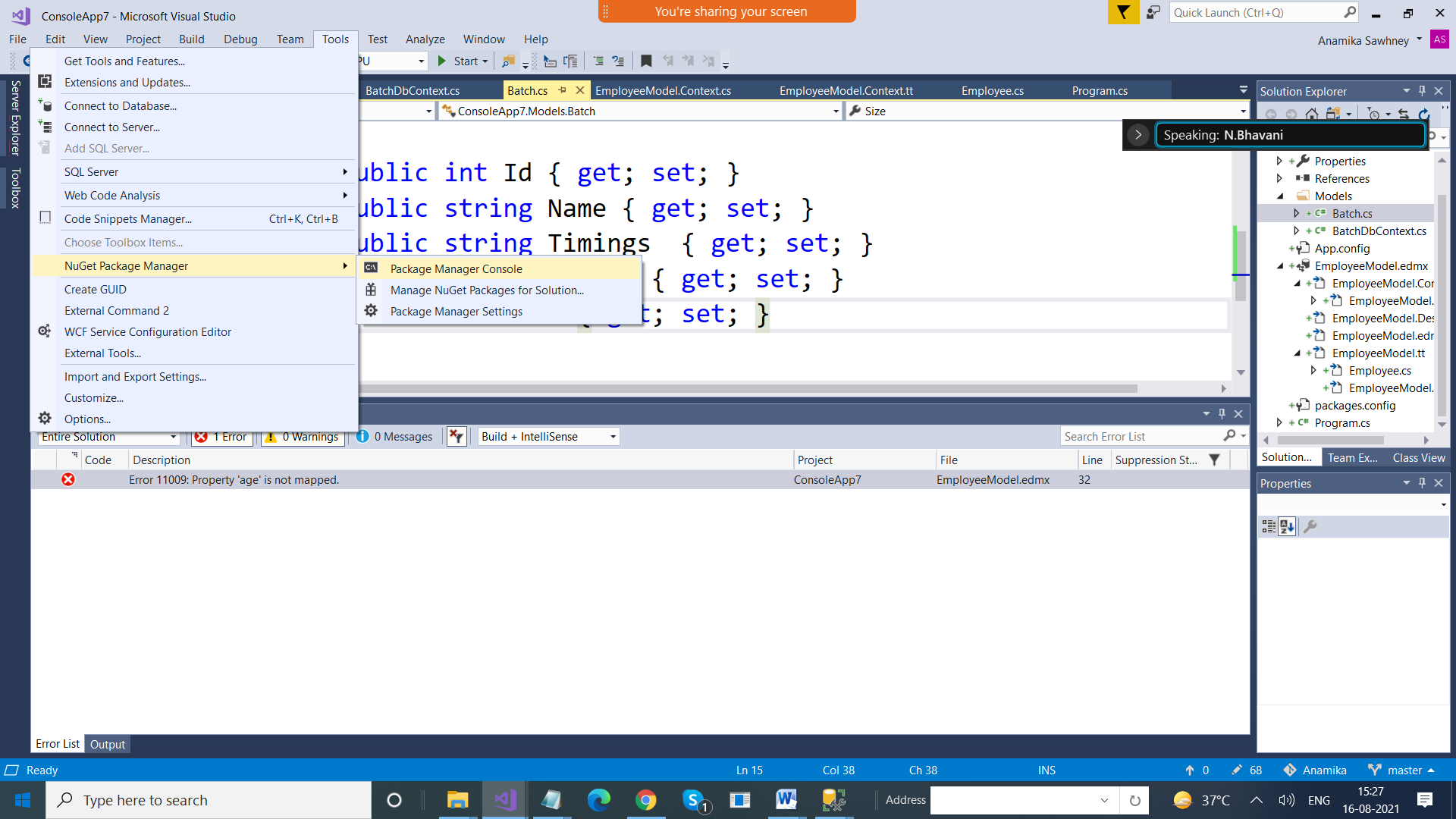
<connectionStrings>

<add name="BatchDbContext" connectionString="data source=LAPTOP-53S2KQS8;initial catalog=A;integrated security=true" providerName="System.Data.SqlClient"/>

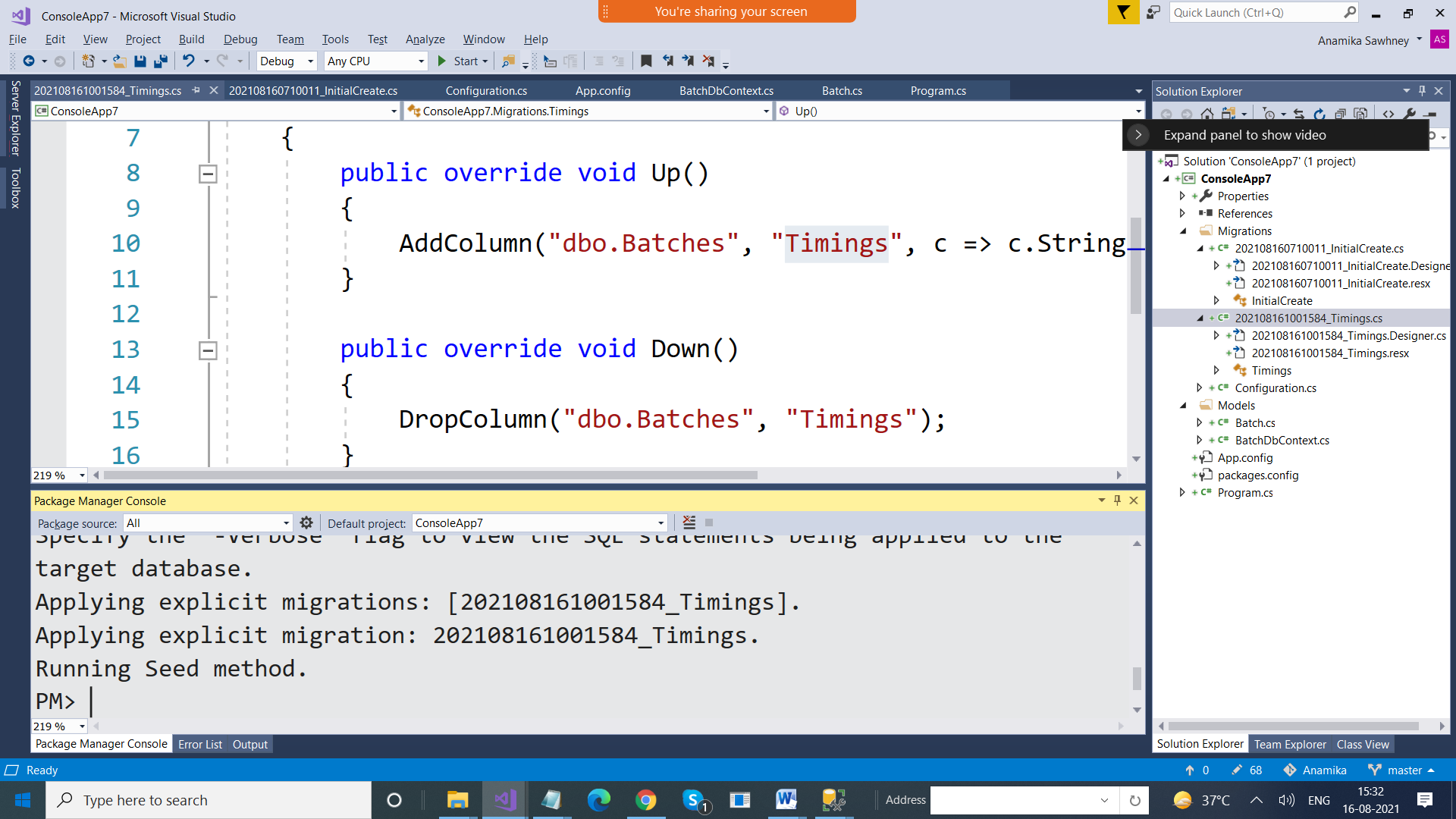
</connectionStrings>

Why use Migration > Its used to transfer changes done in the Model class to database

To use it, we have to first enable migration



Step 1: Enable-Migrations (only once)



After that whenever you make any changes in class, 2 steps are needed

Add-Migration “name”

Update-database