JOHN ABBOTT
CEGEP/COLLEGE

(420-PS4-AB)
Data Sources - ASP .NET MVC

Summer 2018

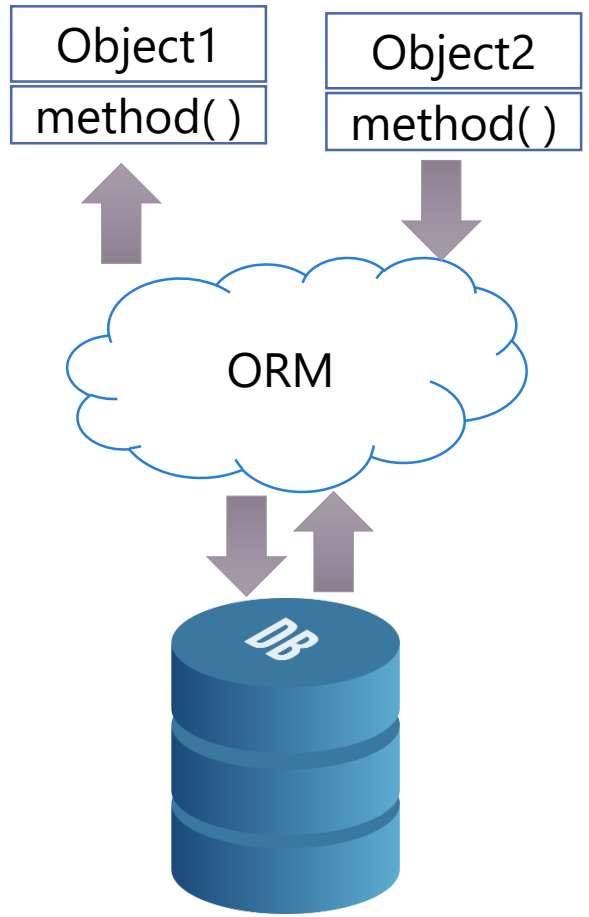
Outline

- Entity Framework & Code First
- Code First Migrations
- Seeding Database
- Overriding Conventions
- Querying Data
- Eager Loading

- Exercise Time

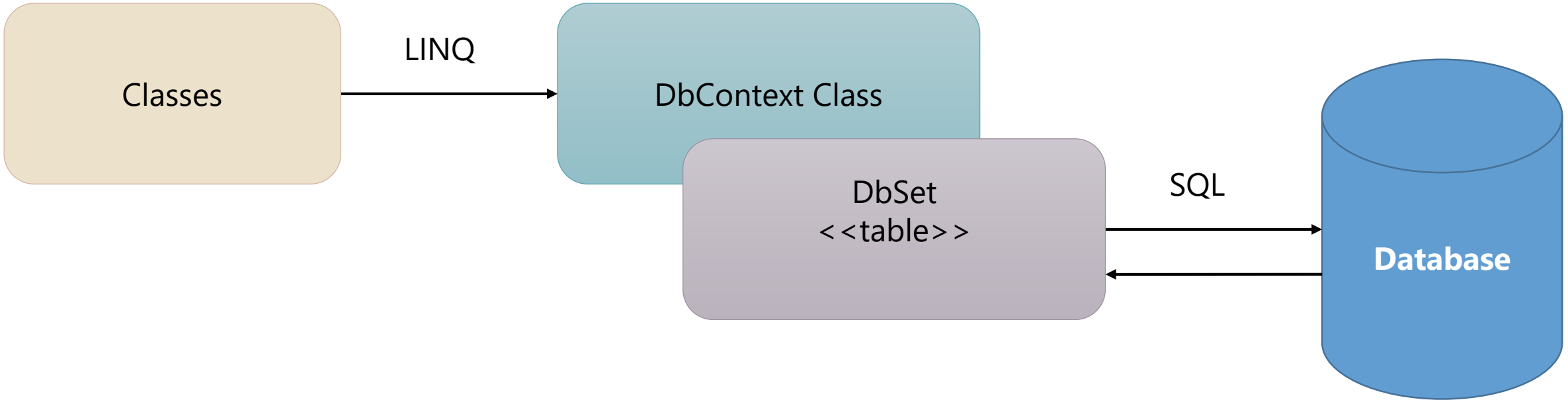
Entity Framework is an ORM

- The .NET Framework provides support for Object Relation Mapping (ORM)

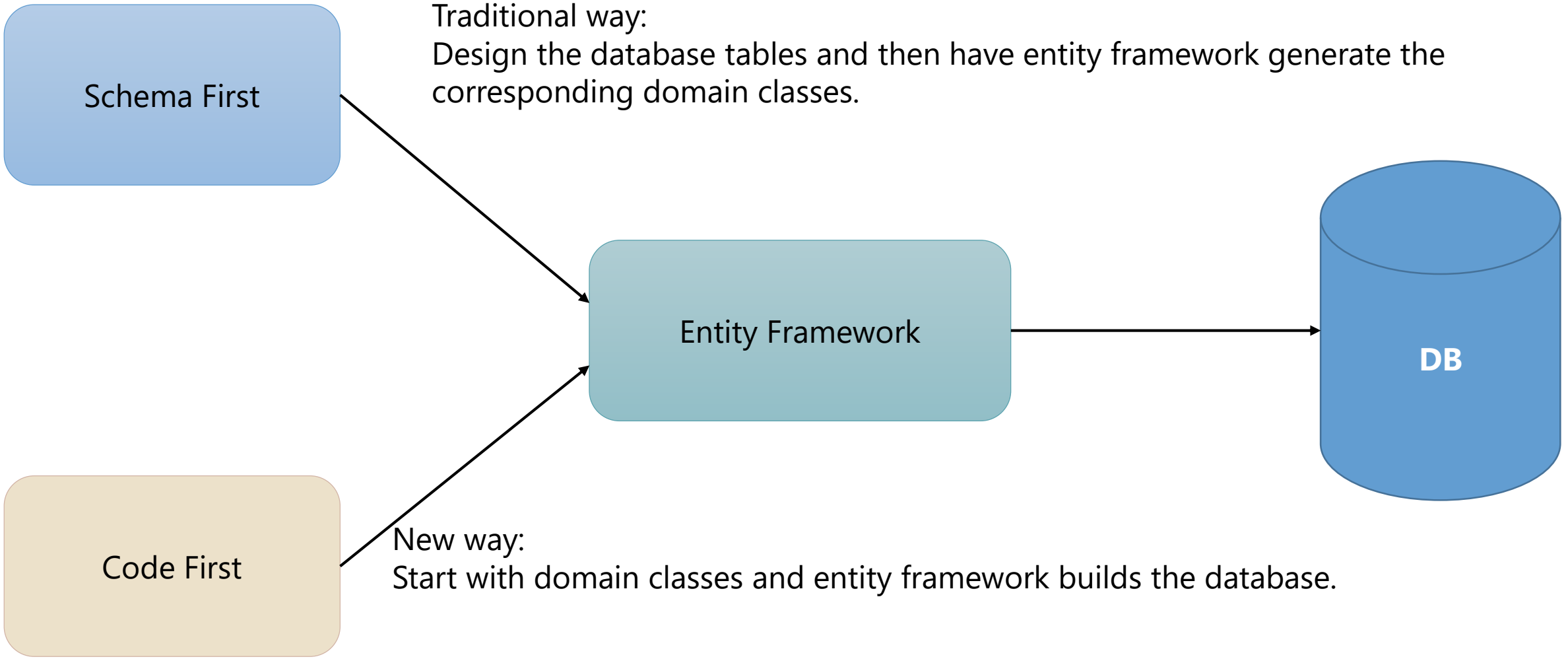


- Features
  - Automatically generate necessary SQL code
  - Map result sets to strongly typed objects
  - Persist object changes back to a database
  - Implicit support for transactions

Entity Framework Flow

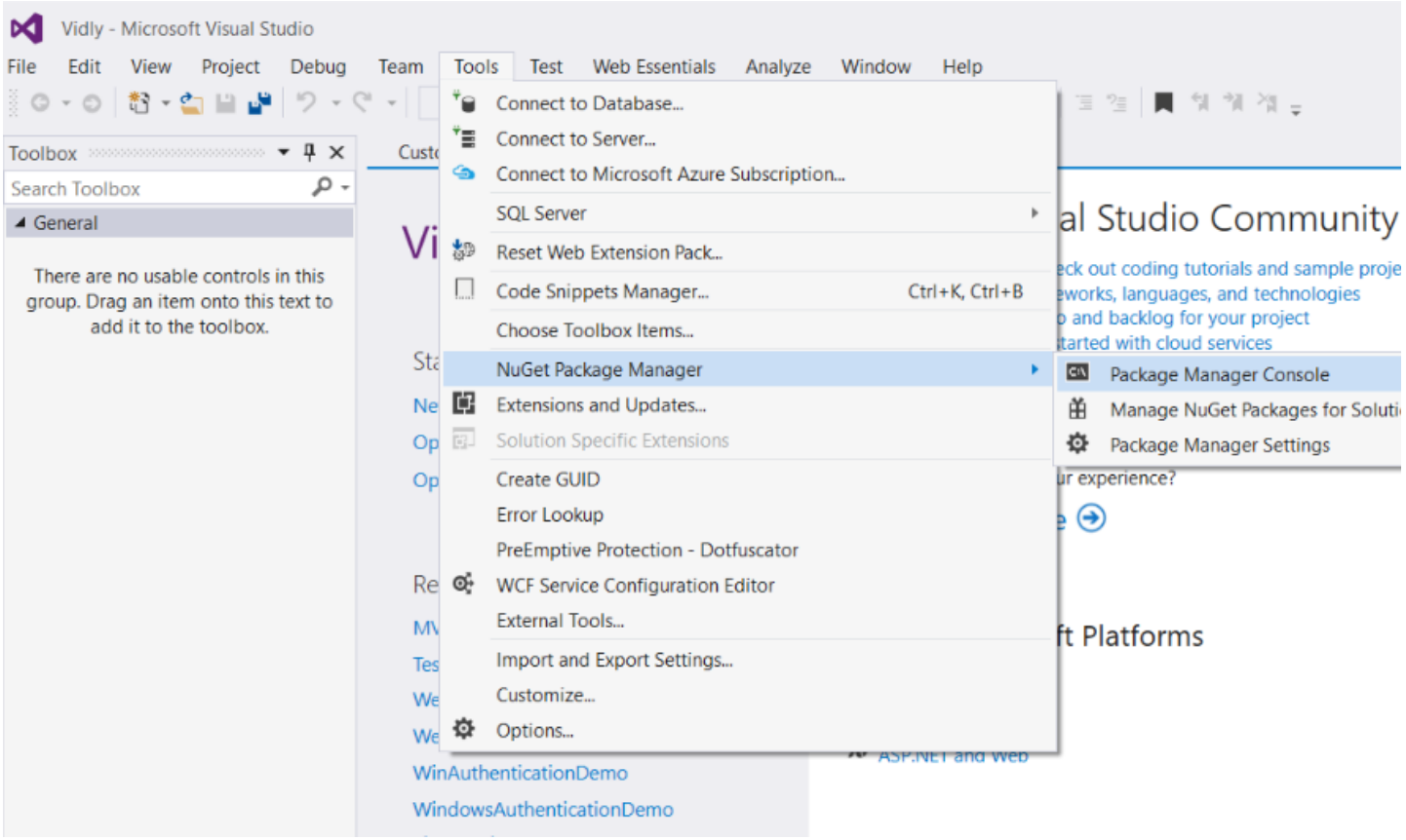


Entity Framework Features



Code First Migration

- Go to Package Manager Console



Code First Migration

- Enable migration: First time only
  - Type: enable-migrations
- Check solution explorer for “Migration” folder
  - Stores all migration history
- Creating first migration
  - Console: add-migration *UniqueName*
  - Use a useful name, for example: *InitialMigration*
- Examine Migration folder content again.
- Consider a migration as a restore point that you can come back to, so choose the name wisely.

Add New Models to the DbContext

- Under class ApplicationDbContext
  - Add property for each model you have

  Example:

  public DbSet<Customer> Customers {get; set;}

- In Console
  - Use add-migration (to reuse the same migration name use "-force"
  - To commit the migration: update-database
- Examine App-Data folder.

Adding Business Elements

- Examine the domain model and suggest changes to in accordance to you business plan (work flow).

- Example:

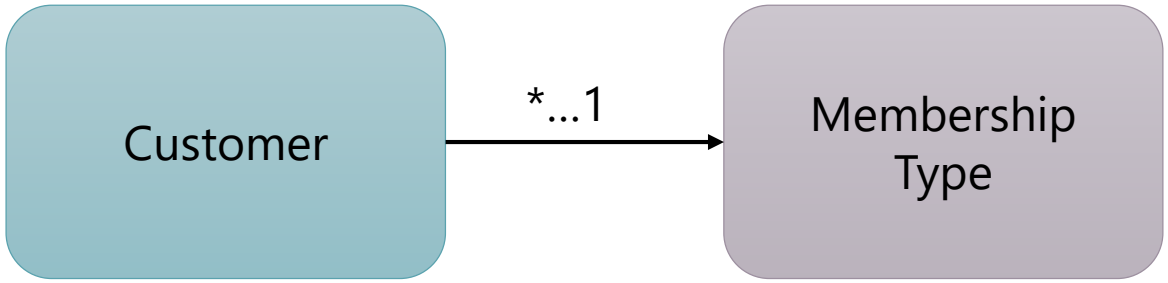
Customer Class in a video rental store requires some changes
→ Add new properties:

  - IsSubscribedToNewsLetter
  - MembershipType

Membership Class

- Memberships

Plan	Fees	Discount on rentals
Pay As You Go	Free	0%.
Monthly	10$	10%
Quarterly	30$	15%
Annual	100$	20%



- Design:
  - SignupFee
  - Duration
  - DiscountRate

Demo: Add changes to Customer Class

- Add changes one by one
- Add migrations to the database.

1. Add property IsSubscribedToNewsLetter
   a) Create a migration with new name & update-database.
2. Add Membership Class
   a) Create class
   b) Add object in customer class
   c) Create a migration with new name & update-database.

Demo: Membership

```
public class Memberships
 {
        public byte Id { get; set; }
        public short SignUpFee { get; set; }
        public byte DurationInMonths { get; set; }
       public byte DiscountRate { get; set; }
}
```

Seeding Database through Migrations

- Using migrations to seed the database is recommended when you want the data to be deployed while production.

- Process:
  - Add migration (most probably it will be an empty migration)
  - User Sql method to run queries to seed to the database.
  - Update database.

Demo: Seeding

- Console:
  - add-migration PopulateMembershipTypes
  - Migration will create a class for you
    - In the up method add required SQL code

```
Sql("INSERT INTO Memeberships
(Id, SignupFee, DurationInMonth, DiscountRate)
VALUES (1,0,0,0)");
```

  - Console: update-Database
- Add all sql needed code to seed the membership type with all our business logic provided earlier.

Overriding Conventions – Revisit

- Several conventions are built into entity framework that are used to determine the schema of the database.
- Example:
  - “Name” property in the Customer class in of String Value
  - In C#, string can have a null value with no limit of number of characters.
  - Entity frame work will define the column in the table as it can be null with max size. (type nvarchar(max))

Overriding Conventions – Revisit

- To override entity framework default convention, use
  "Data Annotation"
- Add namespace:
  `System.ComponentModel.DataAnnotations;`
- Above each property, apply a data "Annotation"
- Annotations are placed within square brackets.
- A single property can have more than one annotation.
  - Examples
    - [Required] : cannot be null
    - [StringLength(100)]

Demo: Data Annotation

- Add needed data annotations to the *Customer* class properties.
- Create a new migration named "ApplyAnnotationsToCustomerClass"
- Update the database

Querying Data

- To query the database, use an instance of the DBConext class.

- An instance maybe added as a private field in the controller.
  - Initialize in constructor.

Demo: Querying Data

- In the Customer Controller
  - Add a DbContext object to access the database
  - Adjust the code in Index action to read from the context object.
  - Test your application.

Demo: Load Data from different tables

- In the Index View we currently display the Customer name only.
  - Add a second column to add the discount rate of the related membership.
  - Run and check what exception you might get.

Eager Loading

- When selecting from a context we need to ask the framework to include foreign data.

- The "*Include*" method is applied on the DB context dataset:

```
_context.Customers.Include(c => c.MemebershipType).toList()
```

The Include method is provided by entity framework.

This is known as ***Eager Loading***

Exercise Time

Exercise (1)

- In the list of customers, replace discount rate with the Name of Memerships.
  - Memberships does not include a “Name” property.
    - Add Name to domain class MemershipType
    - Update the database
    - Create another migration to update existing records in the MembershipType using SQL statement → use Update statement

```
UPDATE MembershipTypes SET Name = 'Monthly' WHERE Id = 2
```

Customer	Membership Type
John Smith	Pay as You Go
Mary Williams	Monthly

Exercise (2)

Customers

Customer	Membership Type
John Smith	Pay as You Go
Mary Williams	Monthly

John Smith

- Membership Type: Pay as You Go
- Birthdate: 1/1/1980

Birthdate:
- A new property that needs to be added to Customer class.
- Birthdate is optional (you can use nonable)
- Property should be displayed only if it has value.

Mary Williams

- Membership Type: Monthly

Exercise (3.1)

- Populate the database with some records for available Medias in VidPlace.

- Make necessary changes to display the Media records in the Index View in the Media Controller.

Exercise (3.2)

- Add the following two reference data types to Media.
- Remember that these reference data are part of the business model and need to deployed the application. (populate using Migrations)

- ***MediaType***
  - Example:

  Movie, TV Show, Tutorial

  - Properties:

  Name

- ***Genre***
  - Example of values:

  Action, Thriller, Science Fiction, Family, Comedy, Romance, Horror & Documentary.

  - Properties:

  Name

Exercise (3.3)

- Modify the Media Index View to include Genre in the table a link to “Details” action in Media Controller (create the action)

- Add a Details View for the Media.
  - Add new Media properties

    ReleaseDate

    DateAdded
  - All are required properties.