Lecture 8. Introduction to database access with Entity Framework

Programming II

School of Business Informatics
Autumn 2016

(: One man's poor software is another man's full time job :)

Plan for the coming module

- Theoretical test: 7th November (during the lecture)
- HW3 - 8th to 25th November
- Team project - 21st November to 18th December, presentation on Dec 19-20

Topics:

- Database programming
- Version control systems
- Parallel programming
- Network interconnection
- Extras

Terminology

A **database** is a collection of data organized in a way to allow efficient access to it by many users.

A **database management system (DBMS)** is a special software that enables interaction between a user / user application and a database.

Why databases

Main tasks solved by a DBMS:

- Data storage
- Multiple access
- Caching
- Dealing with schema changes
- Data validation
- Log management
- Data backup and recovery

Why databases

Main tasks solved by a DBMS:

- Data storage
- Multiple access
- Caching
- Dealing with schema changes
- Data validation
- Log management
- Data backup and recovery

Without a DBMS all these tasks have to be solved by the user application.

Interfacing a database

The primary way to interface a relational database is to use SQL (Structured Query Language).

Old .NET applications managed a number of objects:

- Connection + Command + DataReader
- Connection + DataAdapter + DataSet

Link to an example

- Data is stored in tables
- Each table is a physical container of entities of a certain type
- Columns of the table correspond to attributes of the entity
- A table normally has a primary key (a field or a combination of fields, that uniquely identify a table row)

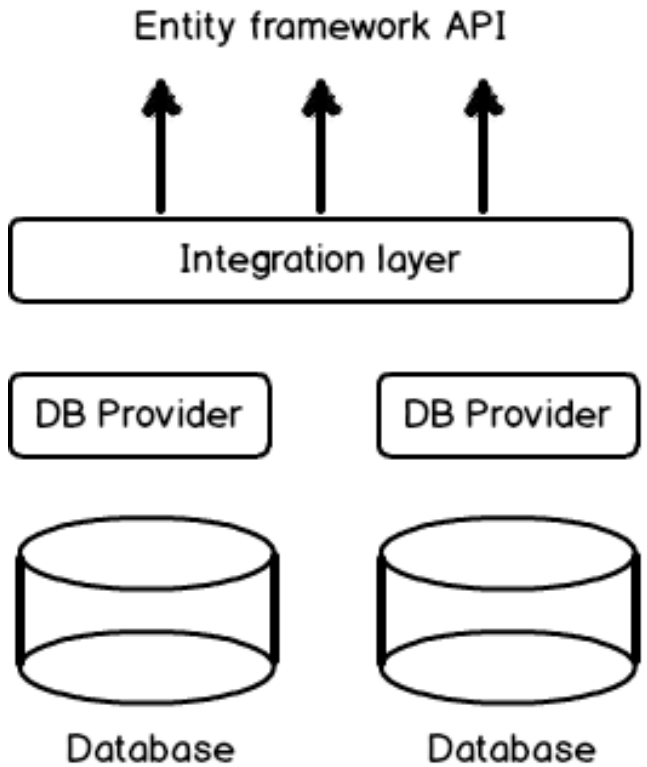
Object to relational mapping


```
class Employee
{
    public int Id {get; set;}
    public string Name {get; set;}
    public int Age {get; set;}
    public string Position {get; set;}
}

...

var e = new Employee(3, "James
Green", 34, "Technical director");
```

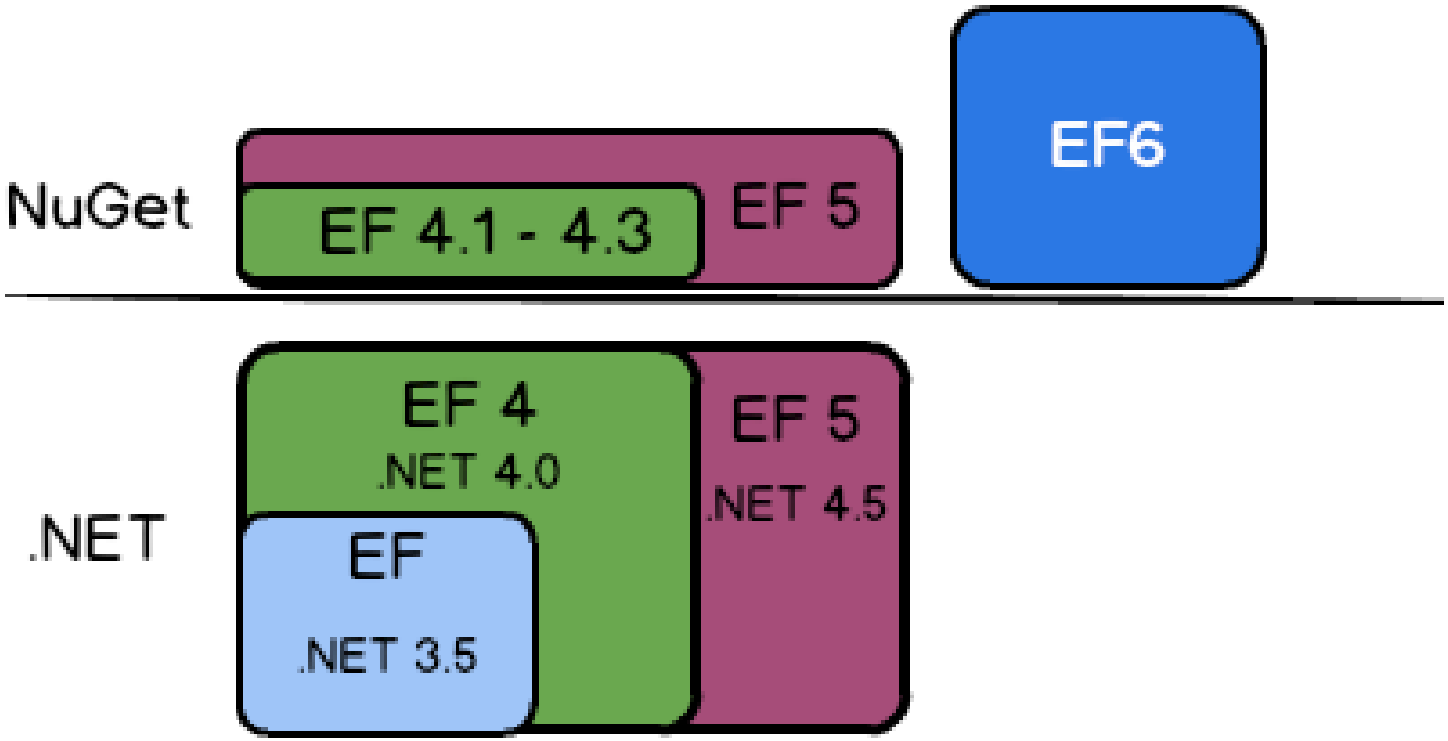
Id	Name	Age	Position
1	Mark Smith	37	CEO
3	James Green	34	Technical director
10	Ann Kaye	28	Financial manager
11	Dirk Brown	35	Lead software developer

Entity Framework

Entity Framework (EF) is an object-to-relational mapper, which for typical .NET applications is the preferred way to interface databases.



Entity Framework evolution

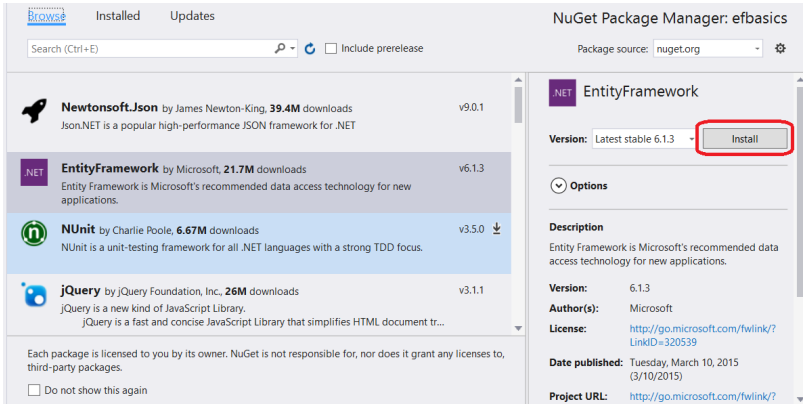


Now a separate EF Core branch is maintained in addition to the basic EF

Installing Entity Framework

Use one of the following approaches:

1. Solution explorer ->Right click project ->Manage NuGet Packages



2. Menu ->View ->Other Windows ->Package Manager Console. In the PM console type: Install-Package EntityFramework

3 ways to use Entity Framework

- Database first - used when a database already exists
- Model first - a special graphical designer is used to create a data model of the subject area
- Code first - database schema is created based on the usual C# code

Code-first

- Create entity classes with all necessary fields. Each entity should contain an integer Id property
- Declare a class derived from DbContext. Inside the class declare DbSet<> for all entities
- Whenever data is required, instantiate the context object and use it as a repository

```
1 class Context : DbContext
2 {
3     public DbSet<Employee> Employees {get; set;}
4     public DbSet<Department> Departments {get; set;}
5 }
```

Unless explicitly specified in code or configuration files, Entity Framework relies on a number of conventions:

- By default EF works with the local installation of the Sql Server
- Each entity (class) should contain an integer Id field, which becomes the primary key of the corresponding table
- If an entity contains a list of related entities they will be represented as one-to-many relationship through a foreign key
- And many others

Data validation

An application can perform several validations of its data:

- Inside the code
- In the DBMS

To enable DBMS validation constraints must be defined on class fields.

Constraints

- Primary key
- Required
- Maximal and minimal length for a string property
- Foreign key
- and others

Data annotations


```
1 public class Department
2 {
3     public int Id { get; set; }
4
5     [Required]
6     [MaxLength(30)]
7     public string Name { get; set; }
8 }
```

Second approach: override a protected method in the Context class

```
1  public class Context : DbContext
2  {
3      protected override void OnModelCreating(
           DbModelBuilder modelBuilder)
4      {
5          modelBuilder.Entity<Employee>()
6              .Property(e => e.Name)
7              .IsRequired()
8              .HasMaxLength(30);
9
10         base.OnModelCreating(modelBuilder);
11     }
12 }
```

Querying the database

Normal LINQ is used to query the database with Entity Framework:

```
1 var query = from s in context.Students
2             orderby s.Rating descending
3             select s;
```

IQueryable interface

- LINQ to SQL extension methods are defined on the IQueryable interface
- IQueryable is designed to build a complete request to the database
- As with IEnumerable, most IQueryable methods are “lazy” (executed only when the result is required).

Loading related entities I

In Entity Framework related entities are not loaded automatically from the database.

Possible ways of solving the problem:

1. Use the Include method:

```
1 var query = context.Employees
2                    .Include(e => e.Department);
3 // For each employee the related Department will
      be loaded
```

2. Declare reference properties as virtual:

```
1 public class Employee
2 {
3     public int Id { get; set; }
4     public string Name { get; set; }
5     public virtual Department Department { get;
          set; }
6 }
```

When the virtual property is accessed, data will be automatically fetched from the database

Page with more examples

After the database has been initialized for the first time, modifications of the entity classes are not possible, unless one of the following steps is taken:

- Manually change the database to match the new entity classes (should be avoided)
- Drop (delete) the old database
- Specify database initialization strategy
- **Use migrations (preferred way)**

- A database migration is very similar to a commit in VCS.
- A migration specifies changes to the database schema relatively to the previous version
- Using migrations with EF
  1. Run "Enable-Migrations" command in the Package Manager Console (only once)
  2. After each schema change execute the "Add-Migration <name>" command
  3. Execute "Update-Database" to export the changes to the database

Seeding

Seeding is the process of populating the database with some default data. Examples may include lists of:

- Countries
- Cities
- Currencies
- Clothes sizes
- i.e. items that are not likely to change

When migrations are enabled, a special "Configuration" class is created in the "Migrations" folder inside the project. Use its "Seed" method to populate the database.

DbContext internals

While in memory, the context instance keeps track of all changes inside entities.
An Entity can be in any of the following five states:

- Unchanged
- Added
- Modified
- Deleted
- Detached

Disconnected scenarios

- In some scenarios the context is not maintained throughout program execution
- After a context instance is disposed, entities are no longer connected to the database (changes are not tracked by the context)
- To apply changes to the newly created context, entities have to be attached manually

Handling concurrency exceptions

If your application updates data that has been edited by a different client, an exception occurs when saving changes
One of the following approaches can be used to resolve the conflict:

- Database wins
- Client wins
- Custom resolution

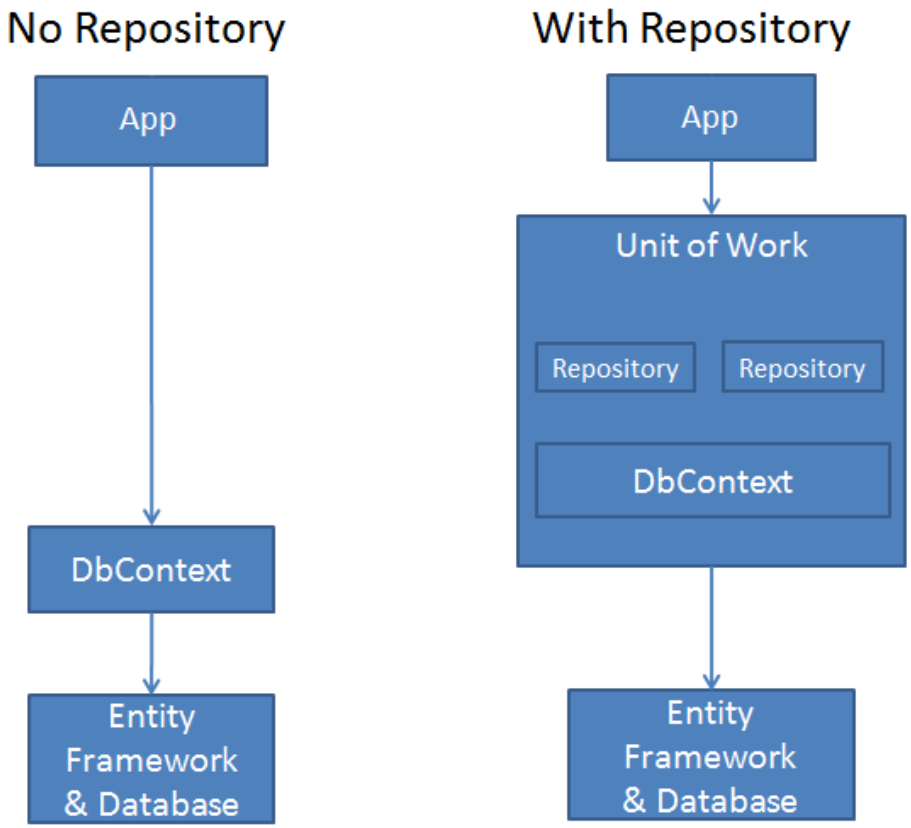
Link to MSDN

Extended concurrency tracking

To track every change to an individual row inside the database add a timestamp (Sql type - ROWVERSION):

```
1
2   public class Employee
3   {
4       public int Id { get; set; }
5       public string Name { get; set; }
6
7       [TimeStamp]
8       public byte[] Version { get; set; }
9       public virtual Department Department {get;set;}
10  }
```

Unit of Work pattern



M. Fowler: "The Unit of Work instance maintains a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems"