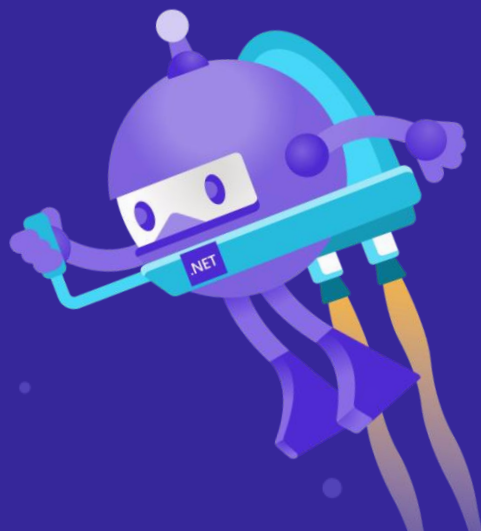
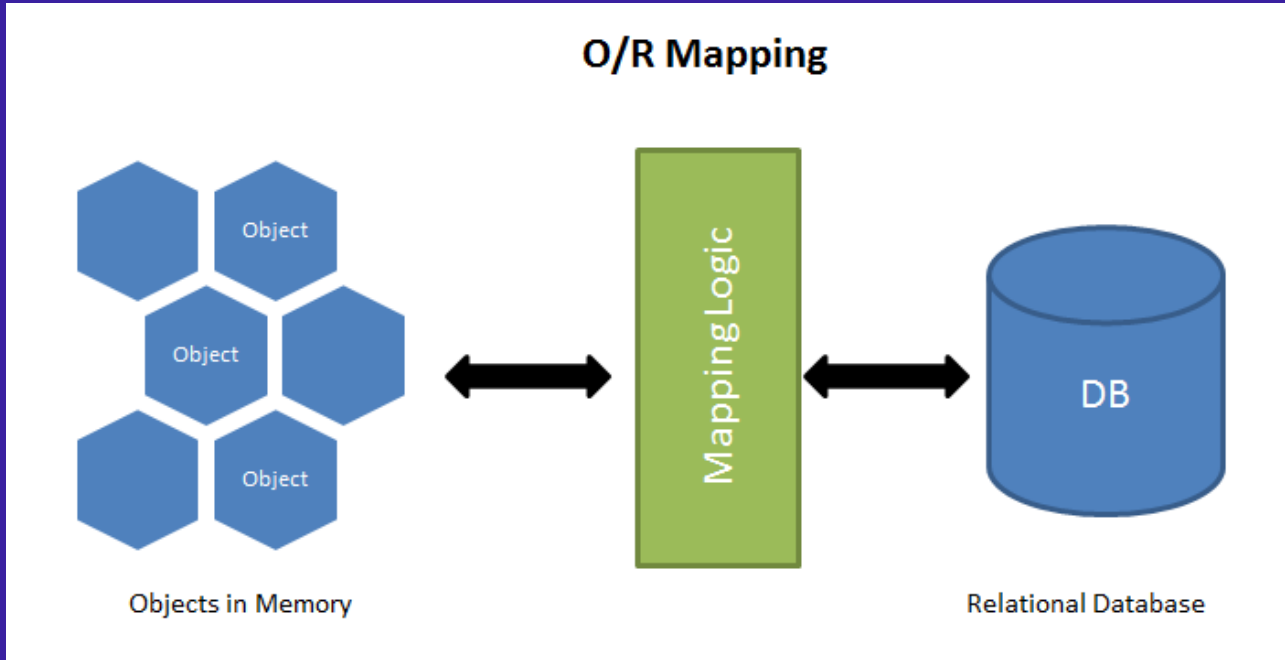
Day 12 : Entity Framework



Object Relational Mapping(ORM)

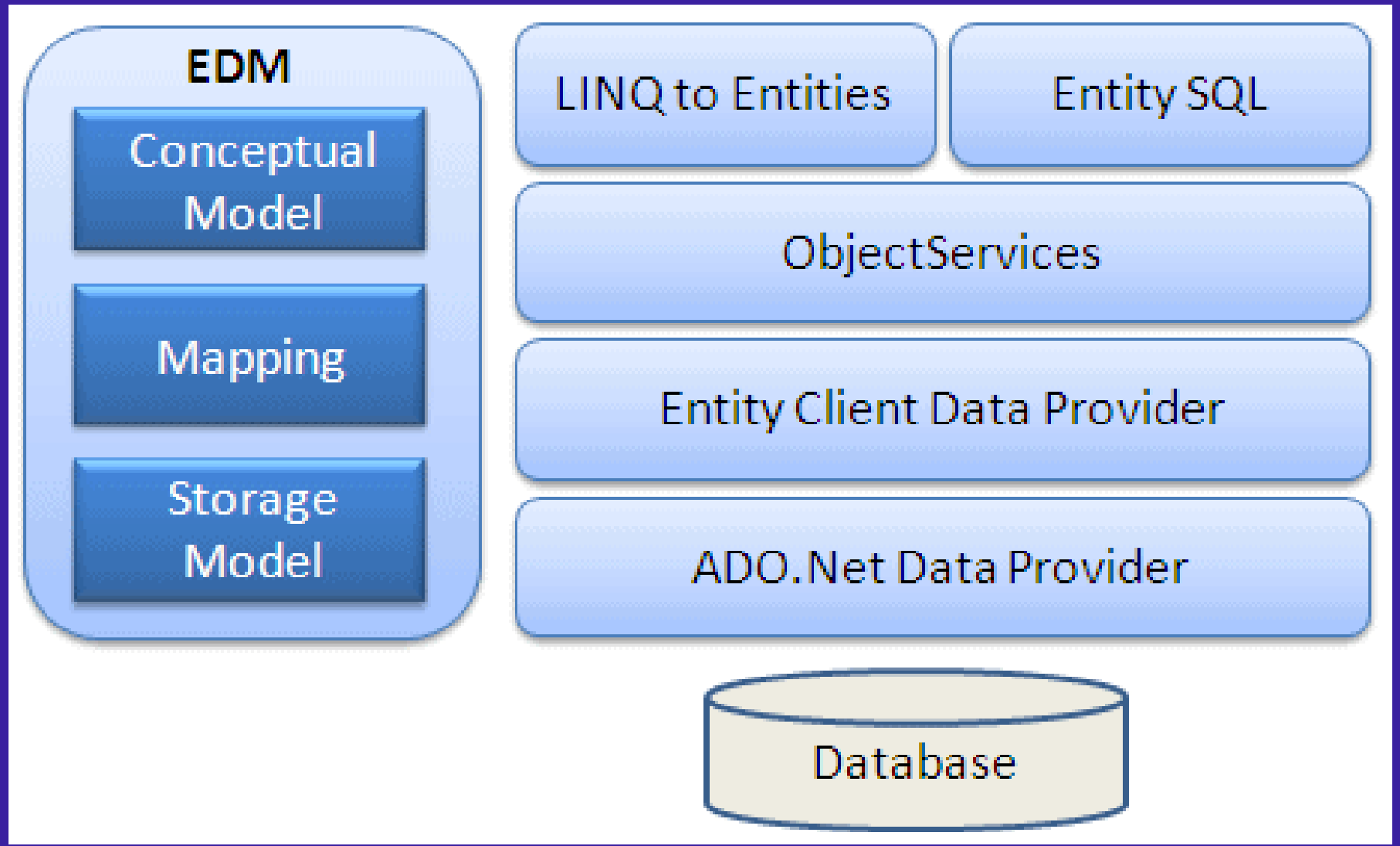
- ORM technology provides is to abstract database access from the rest of the application by encapsulating the database access with its own layer and representing the data from the database within the business objects



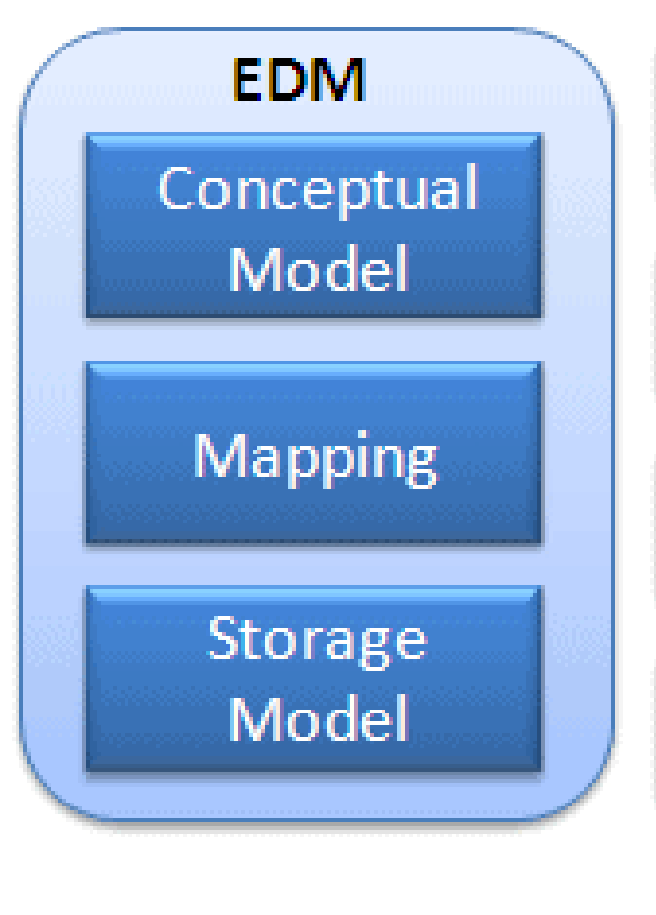
Entity Framework(EF)

- Object-relational mapper (O/RM) for .NET
- Reduces the impedance mismatch between the relational and object-oriented worlds
- Enables developers to write applications that interact with data stored in relational databases using strongly-typed .NET objects
- Available as a NuGet Package

EF Architecture

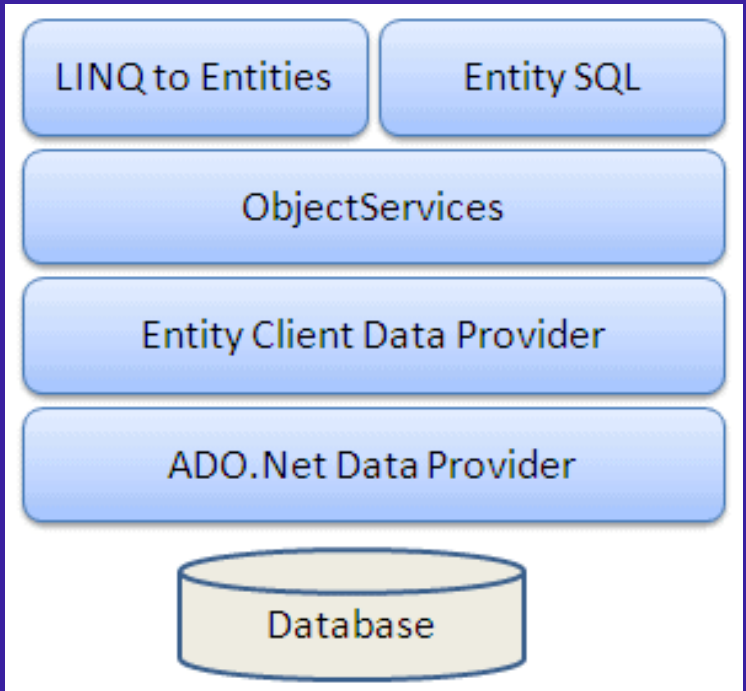


Entity Data Model(EDM)



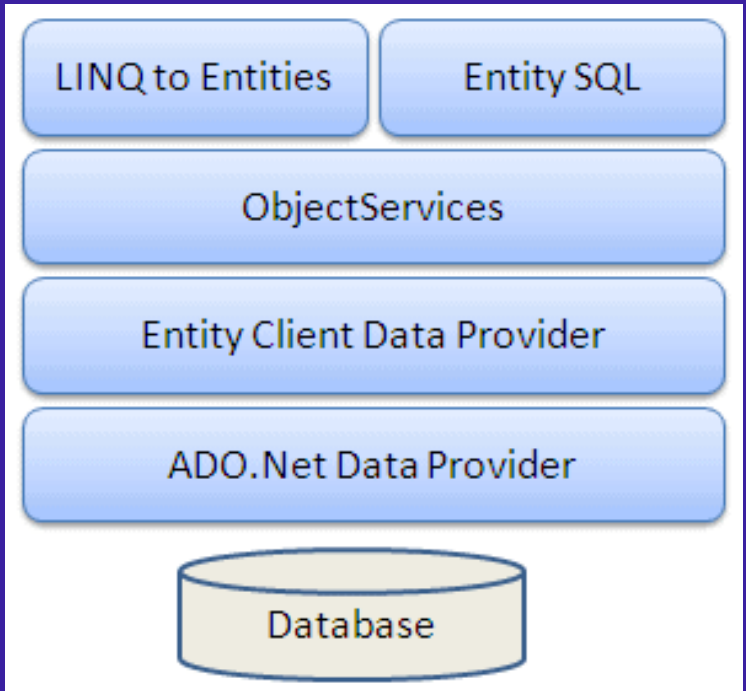
- Conceptual Model
  - Contains Entities and their relationships
- Mapping Model
  - Contains information about how conceptual model is mapped to storage model
- Storage/Logical Model
  - Represents the schema of the underlying database

Linq to Entities, Entity SQL



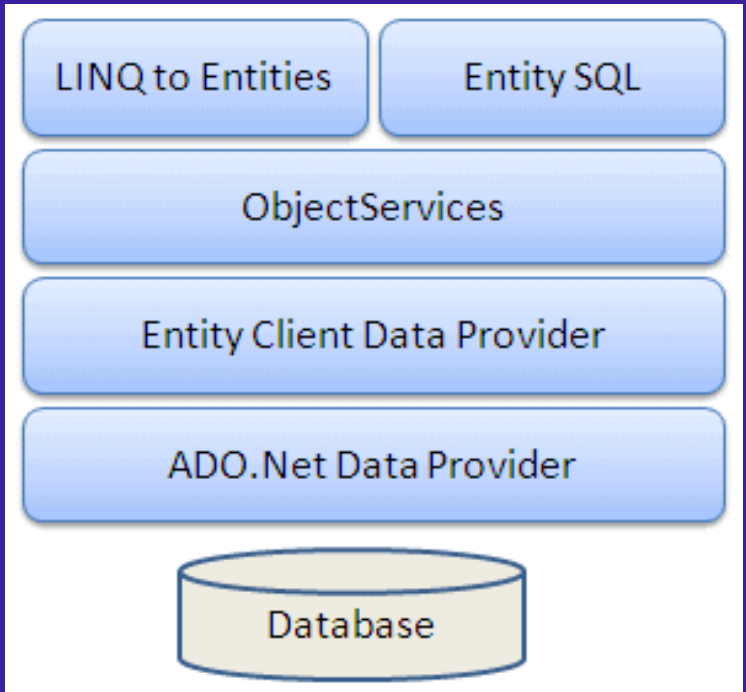
- Linq to Entities
  - Query language used to write queries against the object model
  - Returns entities which are defined in the conceptual model
  - Queries are written using LINQ
- Entity SQL
  - Another query language used in EF 6.0 and higher
  - Queries written using Entity SQL are internally translated to data store dependent SQL Queries

Object Service



- Represents the interaction between the applications and the database
- Provides following features
  - Change tracking
  - Lazy loading
  - Inheritance
  - Optimistic concurrency
  - Merging data
  - Identity resolution
  - Support for querying data using Entity SQL and LINQ to Entities

Data Provides



- Entity Client Data Provider
  - Main responsibility is to convert Linq to Entities or Entity SQL queries into a SQL Query
- ADO.NET Data Provider
  - Communicates with the underlying database using the standard ADO.NET library

Context Class

- Represents a session with the underlying database which can perform the CRUD(Create, Read, Update, Delete) Operations
- Is derived from the ***System.Data.Entity.DbContextDbContext***
- Instance of this class represents a Unit of Work or Repository Patterns
- It can combine multiple changes under a single database transaction

Context Class - Example

```
using System.Data.Entity;

public class EmployeeContext : DbContext
{
    public EmployeeContext()
    {

    }
    // Entities
    public DbSet<Employee> Employee { get; set; }
    public DbSet<Department> Department{ get; set; }
}
```

Entity Class

- Is a class that maps to a table in the database
- Must be included as a *DbSet<Entity>* type property in the Context class
- EF API maps each entity to a table and each property of an entity to a column in that table

Entity Class - Example

```
public class EmployeeContext : DbContext
{// Entities
    public DbSet<Employee> Employee { get; set; }
    public DbSet<Department> Department{ get; set; }
}
```

```
public class Employee
{
  public int EmployeeId { get; set; }
  public string FirstName{ get; set; }
  public Department Department { get;
set; }
```

```
public class Department
{
  public int Id { get; set; }
  public string Name{ get; set; }
  public ICollection<Employee> Employee{ get;
set; }
```

Entity Property Types

- Two types of properties are there
  - Scalar Property

    Primitive type properties are called Scalar properties
    Each one maps to column in the DB which stores the actual data

```
public int EmployeeId { get; set; }

public string FirstName{ get; set; }
```

  - Navigation Property

    Represents a relationship to another entity
    Two Types – Reference Navigation & Collection Navigation

Entity Property Type - Navigation

- Reference Navigation
  - If the entity includes a property of another entity type
  - Points to a single entity and represents a 1-to-1 relationship
  - EF API will create a Foreign Key in the table represented by the entity where its referred

```
public class Employee
{
  public int EmployeeId { get; set; }
  public string FirstName{ get; set; }
  public Department Department { get; set; }
```

Entity Property Type - Navigation

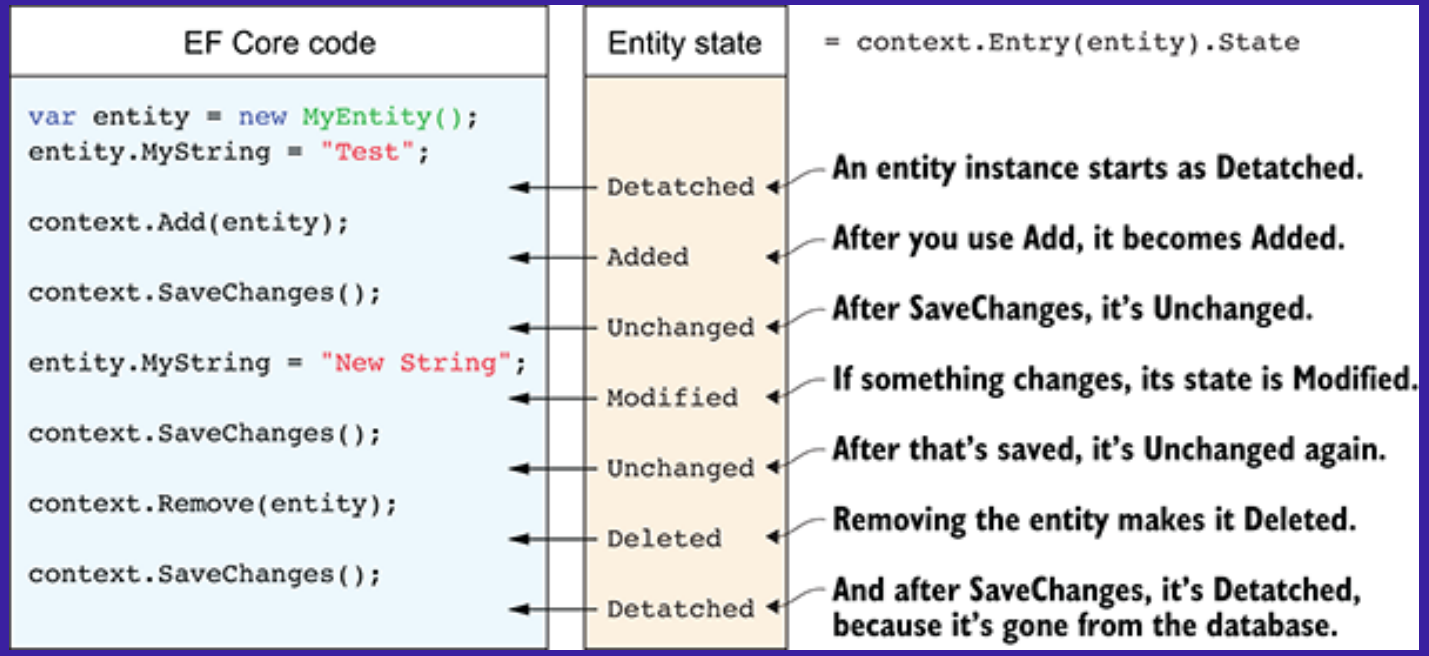
- Collection Navigation
  - If the entity includes a property of generic collection of an entity type
  - Points to a single entity and represents a 1-to-many relationship
  - EF API will create a Foreign Key in the table represented by the entity where its referred

```
public class Department
{
  public int Id { get; set; }
  public string Name{ get; set; }
  public ICollection<Employee> Employee{ get; set; }
```

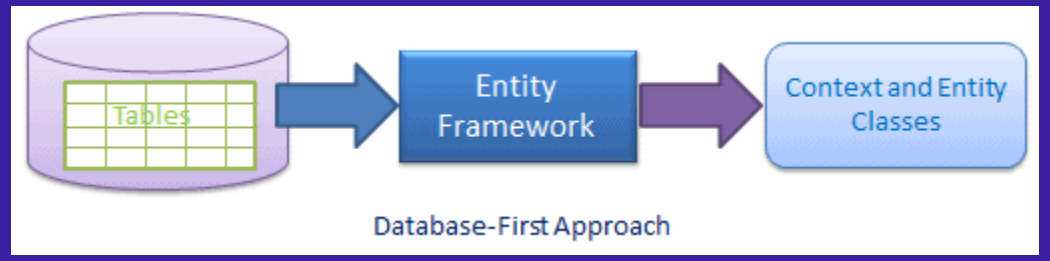
Entity States

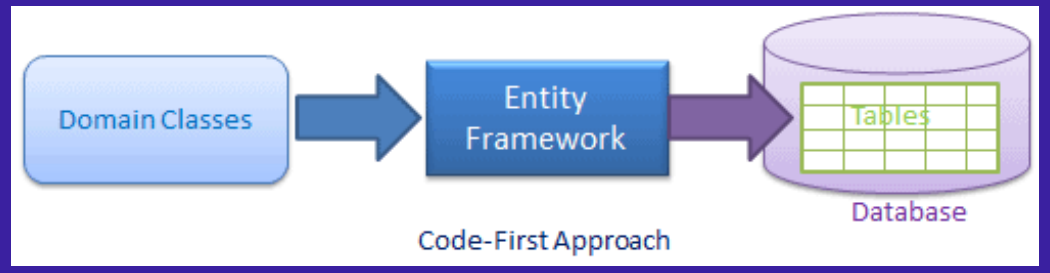
- EF maintains state of entity during its lifetime
- Each entity has a state based on the operation performed using the context class

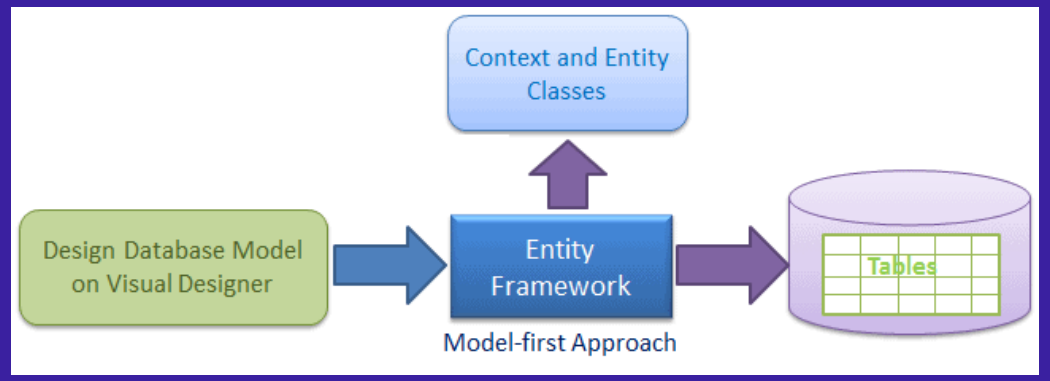
  - Added
  - Modified
  - Deleted
  - Unchanged
  - Detached



Development Approaches







Thanks for joining!