

LẬP TRÌNH C# 5

BÀI 2: DATA ANNOTATION, FLUENT API

- ◎ Entity Framework Code First - Data Annotations
- ◎ Entity Framework Code First- Fluent API



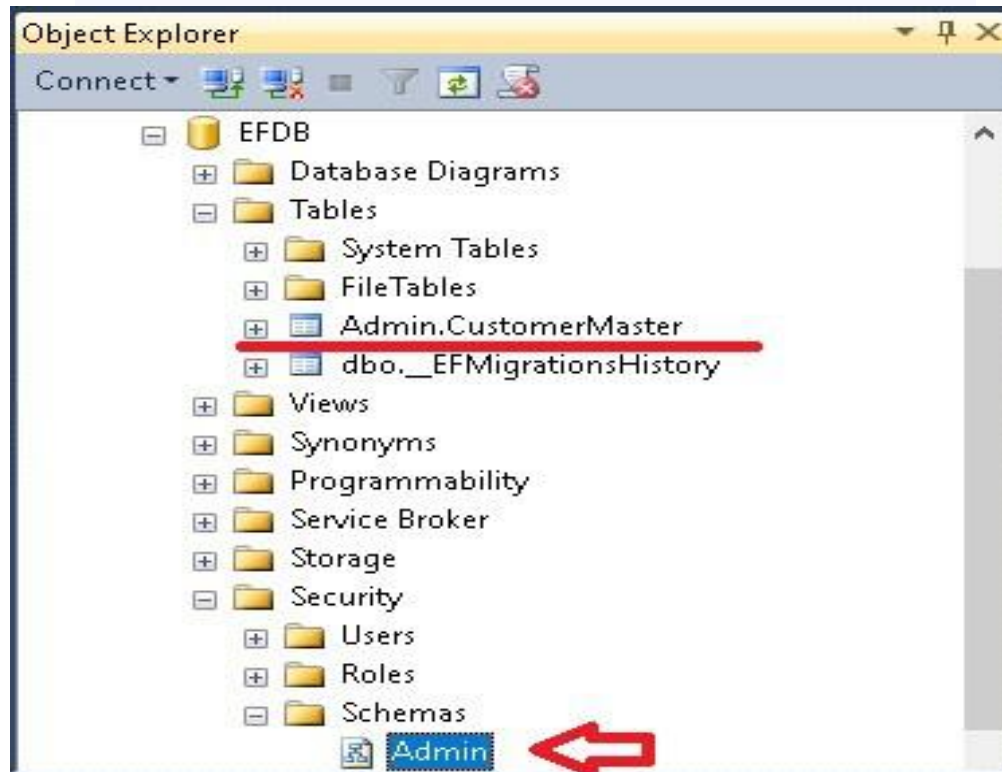
- ❑ Là một phần của siêu dữ liệu (metadata - loại dữ liệu cung cấp thêm thông tin về đối tượng nào đó) giúp chỉ định cấu trúc bảng dữ liệu khi dùng code first, Entity Framework sẽ sử dụng các thông tin này để tạo ra bảng dữ liệu có cấu trúc đúng như mô tả trong domain classes hoặc các properties của domain classes.
- ❑ Các Data annotation attribute có thể được truy xuất tra cứu ở thời điểm thực thi bằng kỹ thuật gọi là reflection, truy xuất ngược từ đối tượng biết được nguồn gốc mà đối tượng đó sinh ra (lớp).
- ❑ Data annotation attribute có thể được chia thành 2 nhóm
 - ❖ Data Modeling Attributes (Database Schema related Attributes)
 - ❖ Validation Related Attributes

- ❑ Khai báo trong domain class hoặc các properties của domain class xác định schema của csdl

Attribute	Description
<u>Table Attribute</u>	You can specify the name of the table to which the entity class maps to using table attributes
<u>Column Attribute</u>	Allows us to specify the name of the column.
<u>ComplexType Attribute</u>	This attribute specifies that the class is a complex type.
<u>DatabaseGenerated Attribute</u>	We use this attribute to specify that the database automatically updates the value of this property.
<u>ForeignKey Attribute</u>	We apply Foreign Key Attribute to a property, which participates as a foreign key in a relationship
<u>InverseProperty Attribute</u>	Specifies the inverse of a navigation property. We use this when we have multiple relationships between two entities. Apply it on a property which is at the other end of the relationship
<u>NotMapped Attribute</u>	Specifies the property, which you do not want to be in the database table. The Entity Framework will not create a column for this property.
Index	Specify this attribute on a property to indicate that this property should have an index in the database

- ❑ Table Attribute xác định Table name và Schema table

```
[Table("CustomerMaster", Schema = "Admin")]  
public class Customer  
{  
    public int CustomerID { get; set; }  
    public string Name { get; set; }  
}
```



- ❑ Column Attribute xác định Column Name, Data type, Column Order

```
[Table("Employee", Schema = "thepv")]
0 references
public class Employee
{
    [Column("ID", Order = 1)]
    0 references
    public int EmployeeId { get; set; }
    [Column("Name", Order = 2, TypeName = "Varchar(100)")]
    0 references
    public string EmployeeName { get; set; }
}
```

- ❑ Key Attribute xác định property là khóa chính cho cột trong table

- ❖ Xác định khóa chính
- ❖ Enabling / Disabling identity column

```
public class Customer
{
    public int CustomerID { get; set; }
    [Key]
    public string CustomeNo { get; set; }
    public string CustomerName { get; set; }
}
```

```
public class Customer
{
    public int CustomerID { get; set; }

    [Key, DatabaseGenerated(DatabaseGeneratedOption.None)]
    public int CustomerNo { get; set; }

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

- ❑ ForeignKey attribute: có 3 cách ràng buộc
 - ❖ Foreign Key property of the dependent class

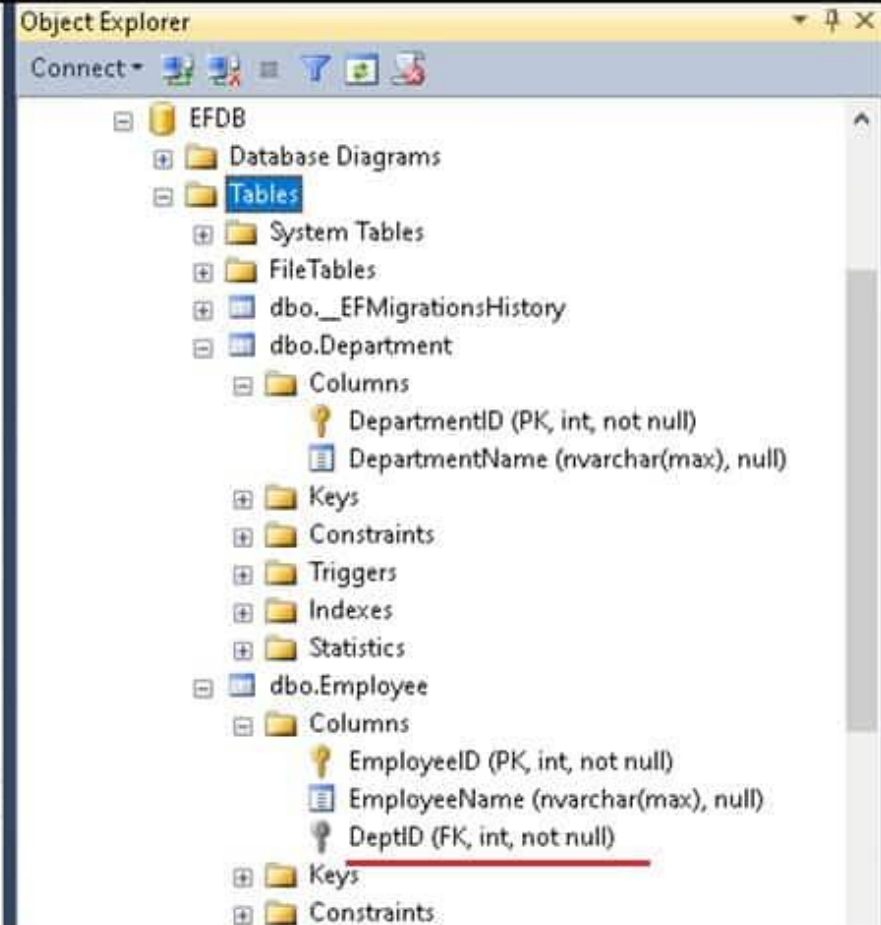
```
public class Employee
{
    public int EmployeeID { get; set; }
    public string EmployeeName { get; set; }

    [ForeignKey("Department")]
    public int DeptID { get; set; }

    //Navigation property
    public Department Department { get; set; }
}

public class Department
{
    public int DepartmentID { get; set; }
    public string DepartmentName { get; set; }

    //Navigation property
    public ICollection<Employee> Employees { get; set; }
}
```



- ❑ ForeignKey attribute: có 3 cách ràng buộc
 - ❖ Navigational Property of the dependent class


```
public class Employee
{
    public int EmployeeID { get; set; }
    public string EmployeeName { get; set; }

    public int DeptID { get; set; }

    //Navigation property
    [ForeignKey("DeptID")]
    public Department Department { get; set; }
}

public class Department
{
    public int DepartmentID { get; set; }
    public string DepartmentName { get; set; }

    //Navigation property
    public ICollection<Employee> Employees { get; set; }
}
```



- ❑ ForeignKey attribute: có 3 cách ràng buộc
 - ❖ Navigational Property of the Principal class


```
public class Employee
{
    public int EmployeeID { get; set; }
    public string EmployeeName { get; set; }

    public int DeptID { get; set; }

    //Navigation property
    public Department Department { get; set; }
}

public class Department
{
    public int DepartmentID { get; set; }
    public string DepartmentName { get; set; }

    //Navigation property
    [ForeignKey("DeptID")]
    public ICollection<Employee> Employees { get; set; }
}
```



- ❑ InverseProperty Attribute : xác định chính xác các ràng buộc khi 2 thực thể có nhiều ràng buộc

```
3 references
public class Flight
{
    0 references
    public int FlightID { get; set; }
    0 references
    public string Name { get; set; }
    0 references
    public Airport DepartureAirport { get; set; }
    0 references
    public Airport ArrivalAirport { get; set; }
}

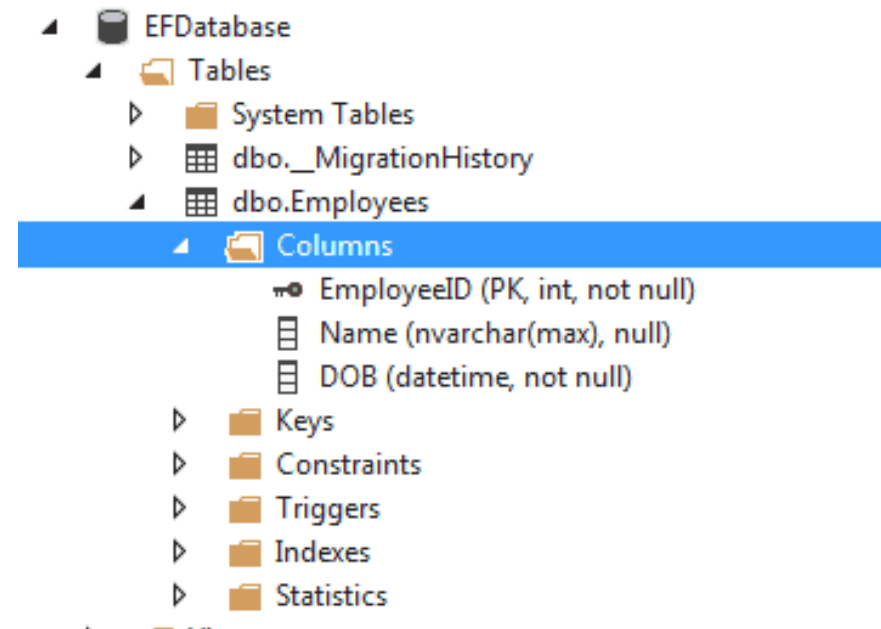
3 references
public class Airport
{
    0 references
    public int AirportID { get; set; }
    0 references
    public string Name { get; set; }
    [InverseProperty("DepartureAirport")]
    0 references
    public virtual ICollection<Flight> DepartingFlights { get; set; }
    [InverseProperty("ArrivalAirport")]
    0 references
    public virtual ICollection<Flight> ArrivingFlights { get; set; }
}
```



- ❑ NotMapped: Chỉ định properties không được ánh xạ thành column

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }
    public DateTime DOB { get; set; }

    [NotMapped]
    public int Age { get; set; }
}
```



- ❑ Khai báo trong domain class hoặc các properties của domain class để xác thực, kiểm tra các thực thể hoặc định nghĩa size, nullability & ...

ATTRIBUTE	DESCRIPTION
<u>ConcurrencyCheck Attribute</u>	Apply this attribute to a property, which participates in concurrency check validation while updating or deleting an entity
<u>Key Attribute</u>	Specify the properties, that are part of the primary key
<u>MaxLength Attribute</u>	This validation attribute specifies the max length of the column in the database
<u>MinLength Attribute</u>	This validation attribute specifies the minimum length of the data allowed in a string or array property.
<u>Required Attribute</u>	Specifies that a data field value is required. Specify the column as non-nullable
<u>StringLength Attribute</u>	Specifies the minimum and maximum length of characters that are allowed in a data field. This attribute is similar to MaxLength & MinLength attribute
<u>Timestamp Attribute</u>	Specifies the data type of the column as a row version.



DEMO

- Dùng code first demo Data Annotations

Lập trình C#5



LẬP TRÌNH C# 5

BÀI 1: DATA ANNOTATION, FLUENT API (P2)

- ❑ Là cách chỉ định cấu trúc cơ sở dữ liệu sử dụng class modelBuilder với chuỗi phương thức (method chaining) ghép nối với nhau.
- ❑ Trong lớp Context (kế thừa DbContext) cần ghi đè (override) phương thức OnModelCreating và chỉ định cấu hình cho từng property của lớp entity thông qua một object của lớp modelBuilder

```
public class EFContext : DbContext
{
    private const string connectionString = "Server=(localdb)\\mssqllocaldb;Database=

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(connectionString);
    }

    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        //Configure domain classes using modelBuilder here
    }
}
```

- ❑ Có thể phân chia EF Core Fluent API thành 3 loại:
- ❖ Model wide configuration (database) gồm các phương thức áp dụng cho toàn bộ model

Method	Description
HasDefaultSchema	Configures the default schema that database objects should be created in, if no schema is explicitly configured.
RegisterEntityType	Registers an entity type as part of the model
HasAnnotation	Adds or updates an annotation on the model. If an annotation with the key specified in annotation already exists its value will be updated.
HasChangeTrackingStrategy	Configures the default ChangeTrackingStrategy to be used for this model. This strategy indicates how the context detects changes to properties for an instance of an entity type.
<u>Ignore</u>	Excludes the given entity type from the model. This method is typically used to remove types from the model that were added by convention.
HasDbFunction	Configures a database function when targeting a relational database.
HasSequence	Configures a database sequence when targeting a relational database.

❑ Có thể phân chia EF Core Fluent API thành 3 loại:

❖ Entity Configuration gồm các phương thức áp dụng entity

Method	Description
Ignore	Exclude the entity from the Model.
ToTable	Sets the table name for the entity type
HasKey	Sets the properties that make up the primary key for this entity type.
HasMany	Configures a relationship where this entity type has a collection that contains instances of the other type in the relationship.
HasOne	Configures a relationship where this entity type has a reference that points to a single instance of the other type in the relationship.
HasAlternateKey	Adds or updates an annotation on the entity type. If an annotation with the key specified in annotation already exists its value will be updated
HasChangeTrackingStrategy	Configures the ChangeTrackingStrategy to be used for this entity type. This strategy indicates how the context detects changes to properties for an instance of the entity type.
HasIndex	Configures an index on the specified properties. If there is an existing index on the given set of properties, then the existing index will be returned for configuration.
OwnsOne	Configures a relationship where the target entity is owned by (or part of) this entity. The target entity key value is always propagated from the entity it belongs to.

❑ Có thể phân chia EF Core Fluent API thành 3 loại:

❖ Entity Configuration gồm các phương thức áp dụng entity

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.HasDefaultSchema("Admin");

    modelBuilder.Entity<Employee>().HasKey(e => e.EmployeeID);

    modelBuilder.Entity<Employee>().|
}

public DbSet<Employee> Employees { get; }
```

Entity<TEntityType> method

Methods available to Configure the Entity

❑ Có thể phân chia EF Core Fluent API thành 3 loại:

- ❖ Property Configuration cấu hình thuộc tính: column name, default value, nullability, Foreignkey, data type...

Method	Description
Ignore	Exclude the Property from the Model.
HasColumnName	Configures database column name of the property
HasColumnType	Configures the database column data type of the property
HasDefaultValue	Configures the default value for the column that the property maps to when targeting a relational database.
HasComputedColumnSql	Configures the property to map to a computed column when targeting a relational database.
HasField	Specifies the backing field to be used with a property.
HasMaxLength	Specifies the maximum length of the property.
IsConcurrencyToken	Enables the property to be used in an optimistic concurrency updates
IsFixedLength	Configures the property to be fixed length. Use HasMaxLength to set the length that the property is fixed to.
IsMaxLength	Configures the property to allow the maximum length supported by the database provider

❑ Vi dụ code first Fluent API

```

protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    //Write Fluent API configurations here

    //Entity Configuration
    modelBuilder.Entity<Country>().HasKey(s => s.PId);

    //Property Configurations
    modelBuilder.Entity<Country>(entity =>
    {
        entity.Property(e => e.Name)
            .HasColumnName("CountryName")
            .HasDefaultValue("USA")
            .IsRequired();

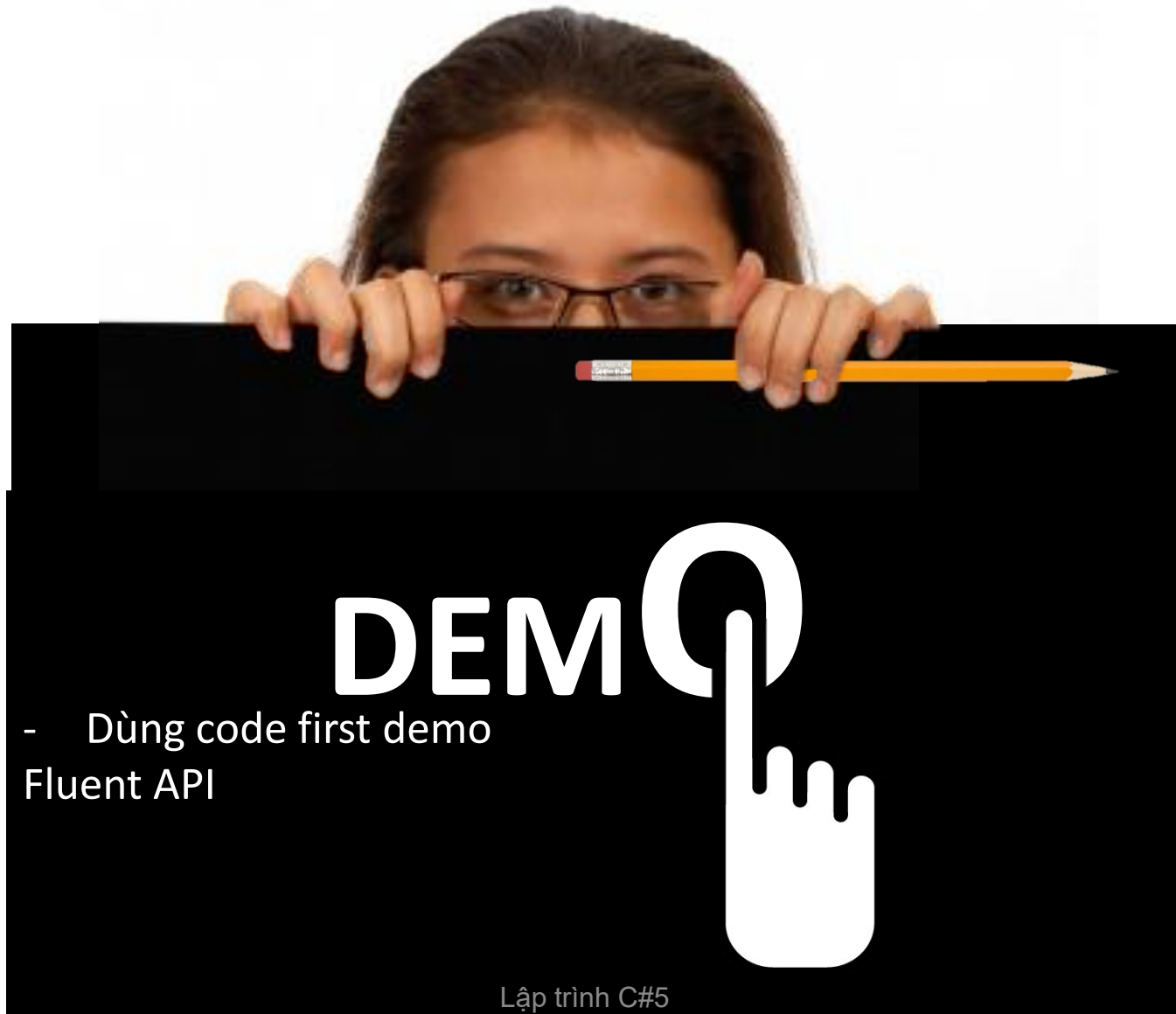
        entity.Property(e => e.AddedOn)
            .HasColumnType("date")
            .HasDefaultValueSql("(getdate())");
    });
    modelBuilder.Entity<Country>().Ignore(e => e.population);
}

```

public class Country
{
 public int PId { **get**; **set**; }
 public string Name { **get**; **set**; }
 public DateTime AddedOn { **get**; }
}

VAIO.Company - dbo.Country

	Column Name	Data Type	Allow Nulls
🔍	PId	int	<input type="checkbox"/>
	AddedOn	date	<input type="checkbox"/>
	CountryName	nvarchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>

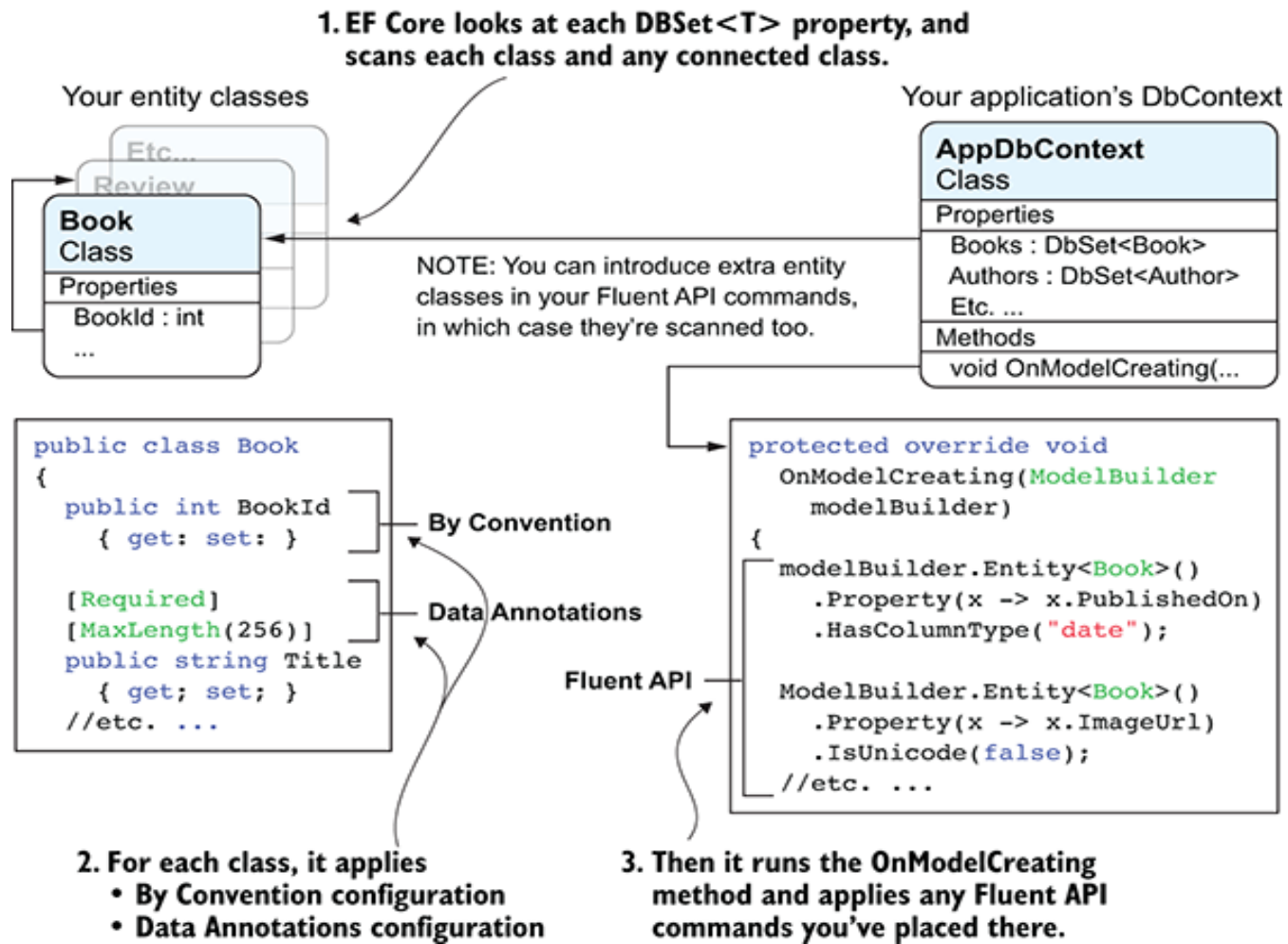


DEMO

- Dùng code first demo
Fluent API

Lập trình C#5

- Entity Framework code-first cung cấp các kỹ thuật để xây dựng tất cả các loại quan hệ (một – nhiều, nhiều – nhiều, một – một) dựa trên các phương pháp: Convention, Data Annotations, Fluent



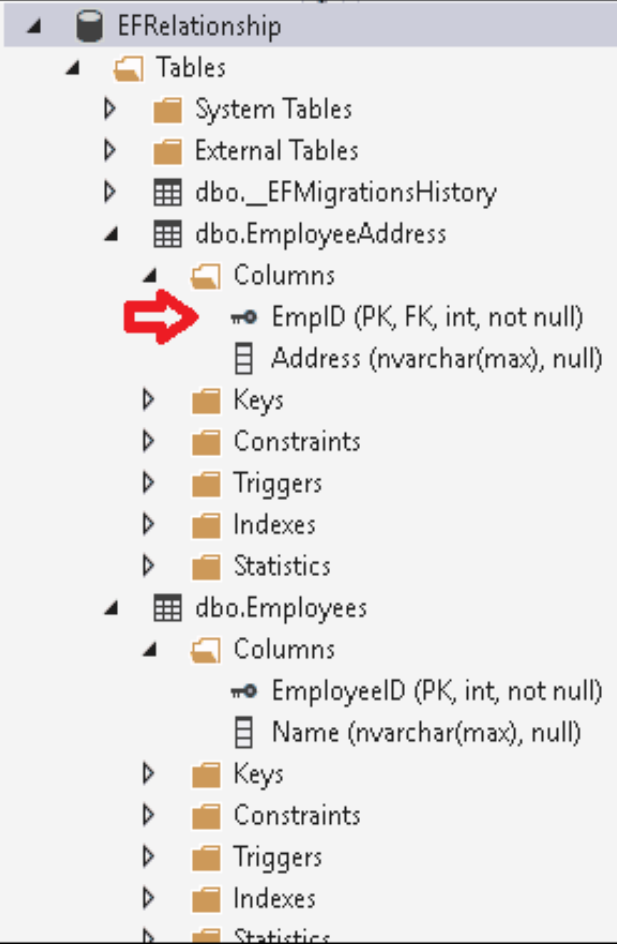
- ❑ Dùng Convention (xem bài 1)
- ❑ Dùng Data Annotations

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }

    //Navigation property Returns the Employee Address
    public EmployeeAddress EmployeeAddress { get; set; }
}

public class EmployeeAddress
{
    [Key, ForeignKey("Employee")]
    public int EmpID { get; set; }
    public string Address { get; set; }

    //Navigation property Returns the Employee object
    public Employee Employee { get; set; }
}
```



- ❑ Dùng Fluent API thông qua phương thức HasOne/WithOne
 - ❖ Bắt đầu với Employee class

```
{
    modelBuilder.Entity<EmployeeAddress>()
        .HasKey(s => s.EmployeeID);

    modelBuilder.Entity<Employee>()
        .HasOne<EmployeeAddress>(p => p.EmployeeAddress)
        .WithOne(s => s.Employee);
}

public DbSet<Employee> Employees { get; set; }
public DbSet<EmployeeAddress> EmployeeAddress { get; set; }
```

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }

    //Navigation property Returns the Employee Address
    public EmployeeAddress EmployeeAddress { get; set; }
}

public class EmployeeAddress
{
    public int EmployeeID { get; set; }
    public string Address { get; set; }

    //Navigation property Returns the Employee object
    public Employee Employee { get; set; }
}
```

- ❖ Bắt đầu với EmployeeAddress

```
modelBuilder.Entity<EmployeeAddress>()
    .HasOne<Employee>(p => p.Employee)
    .WithOne(s => s.EmployeeAddress);
```

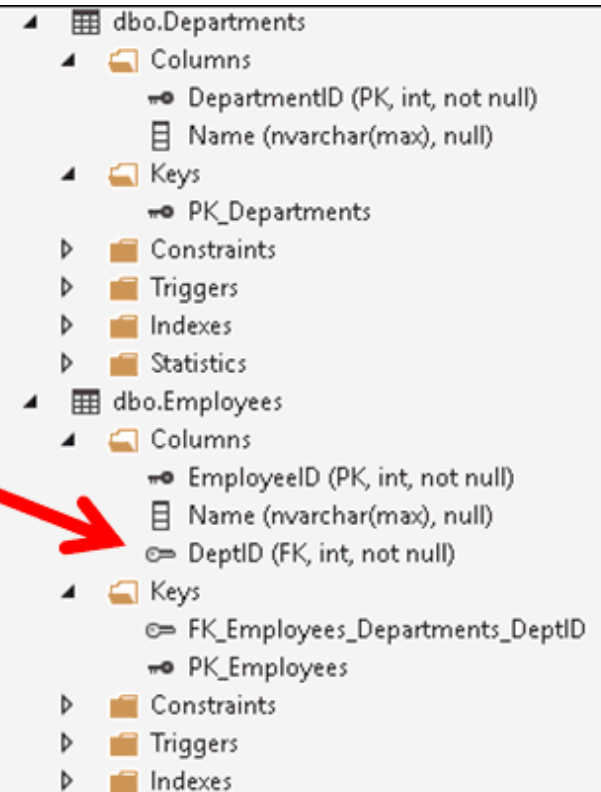

- ❑ Dùng Convention (xem bài 1)
- ❑ Dùng Data Annotations

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }

    [ForeignKey("Department")]
    public int DeptID { get; set; }
    public Department Department { get; set; }
}

public class Department
{
    public int DepartmentID { get; set; }
    public string Name { get; set; }

    public ICollection<Employee> Employees { get; set; }
}
```



❑ Dùng Fluent API qua phương thức HasOne/WithMany

```

modelBuilder.Entity<Employee>()
    .HasOne<Department>(e => e.Department)
    .WithMany(d => d.Employees)
    .HasForeignKey(e => e.DeptID);

public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }
    public int DeptID { get; set; }
    public Department Department { get; set; }
}

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

- ▶ **dbo.Departments**
 - ▶ Columns
 - DepartmentID (PK, int, not null)
 - Name (nvarchar(max), null)
 - ▶ Keys
 - PK_Departments
 - ▶ Constraints
 - ▶ Triggers
 - ▶ Indexes
 - ▶ Statistics
- ▶ **dbo.Employees**
 - ▶ Columns
 - EmployeeID (PK, int, not null)
 - Name (nvarchar(max), null)
 - DeptID (FK, int, not null)
 - ▶ Keys
 - FK_Employees_Departments_DeptID
 - PK_Employees
 - ▶ Constraints
 - ▶ Triggers
 - ▶ Indexes
 - ▶ Statistics

- ❑ Dùng Data Annotations: phát sinh thêm Entity class kết hợp từ 2 class có quan hệ nhiều – nhiều
- ❑ Ví dụ có model:

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }
    public virtual ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}

public class Project
{
    public int ProjectID { get; set; }
    public string Name { get; set; }
    public virtual ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}
```

MANY TO MANY RELATIONSHIP

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }

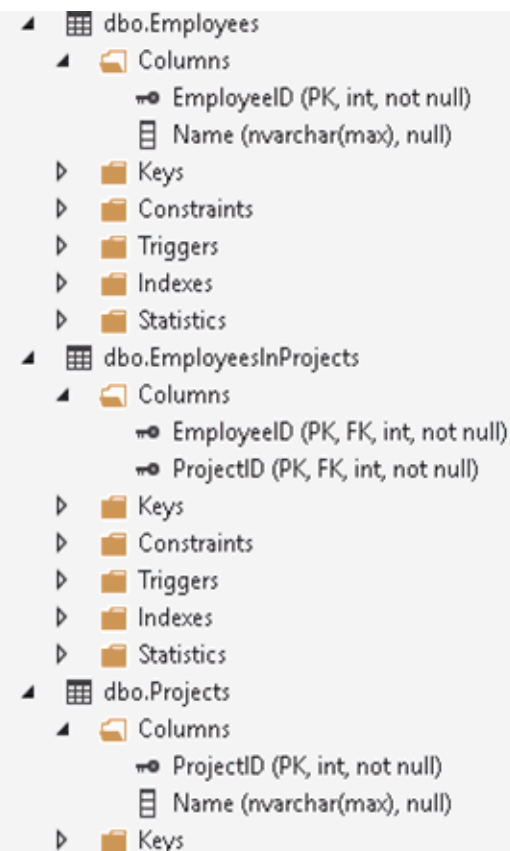
    [ForeignKey("EmployeeID")]
    public ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}

public class Project
{
    public int ProjectID { get; set; }
    public string Name { get; set; }

    [ForeignKey("ProjectID")]
    public ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}

public class EmployeesInProject
{
    public int EmployeeID { get; set; }
    public Employee Employee { get; set; }

    public int ProjectID { get; set; }
    public Project Project { get; set; }
}
```



```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<EmployeesInProject>()
        .HasKey(e => new { e.EmployeeID, e.ProjectID });
}
```

- ❑ Dùng Fluent API: kết hợp tạo Entity class kết hợp từ 2 class có quan hệ nhiều – nhiều và thiết lập 2 mỗi quan hệ một nhiều

```
public class Employee
{
    public int EmployeeID { get; set; }
    public string Name { get; set; }
    public virtual ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}

public class Project
{
    public int ProjectID { get; set; }
    public string Name { get; set; }
    public virtual ICollection<EmployeesInProject> EmployeesInProject { get; set; }
}

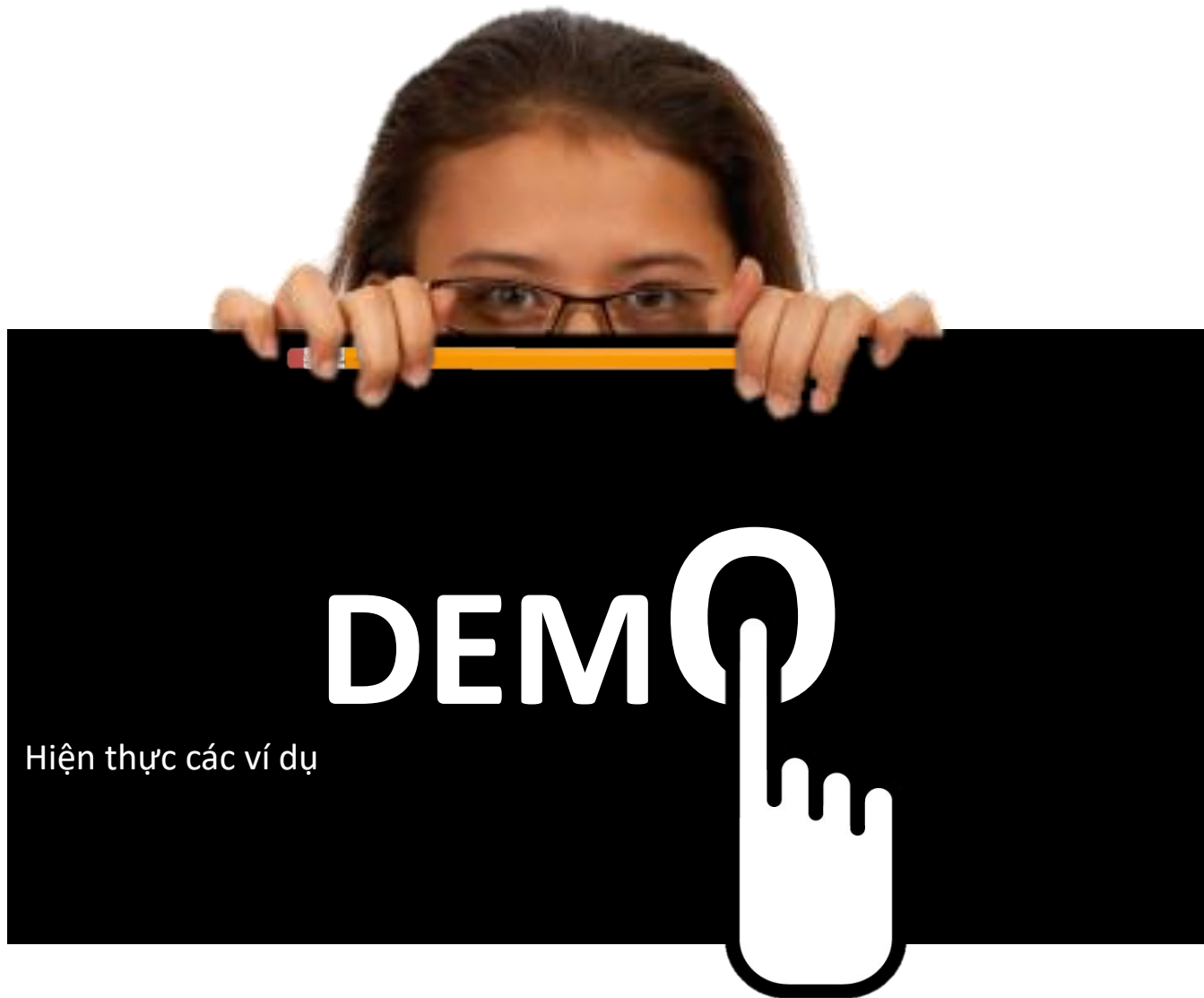
public class EmployeesInProject
{
    public int EmployeeID { get; set; }
    public Employee Employee { get; set; }

    public int ProjectID { get; set; }
    public Project Project { get; set; }
}
```

```
modelBuilder.Entity<EmployeesInProject>()
    .HasKey(e => new { e.EmployeeID, e.ProjectID });

modelBuilder.Entity<EmployeesInProject>()
    .HasOne<Employee>(e => e.Employee)
    .WithMany(p => p.EmployeesInProject);

modelBuilder.Entity<EmployeesInProject>()
    .HasOne<Project>(e => e.Project)
    .WithMany(p => p.EmployeesInProject);
```



Tổng kết bài học

- ◎ Entity Framework Code First - Data Annotations
- ◎ Entity Framework Code First- Fluent API





KẾT THÚC