


CÔNG NGHỆ WEB

MODELS

NỘI DUNG

- Giới thiệu
- EF
- Truy xuất Database
- Model Validation

GIỚI THIỆU

- Models chứa tất cả các xử lý:
  - Nghiệp vụ hệ thống
  - Logic hệ thống, truy xuất CSDL
  - Tính hợp lệ

GIỚI THIỆU

- Database ?
  - Database là tập hợp các bản ghi có liên quan với nhau. Thông tin trong database được lưu trữ sao cho dễ truy cập, quản trị và cập nhật.

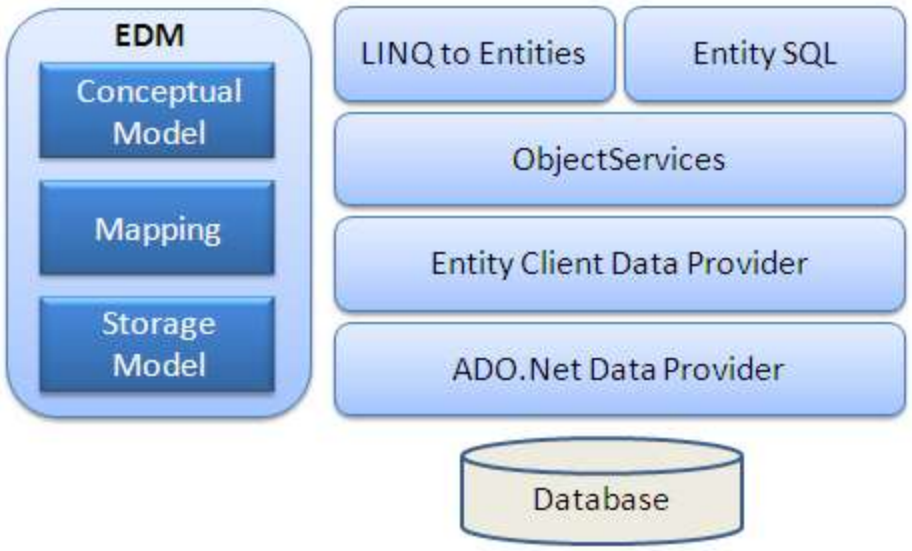
+ Các mô hình truy xuất:
  - ADO.NET
  - Entity Framework

ENTITY FRAMEWORK -EF

- Entity framework là Object/Relational Mapping (O/RM) framework:
  - Là sự phát triển nâng cao của ADO.NET cho tự động truy cập và lưu trữ dữ liệu trong database.
  - Làm việc với dữ liệu quan hệ như “domain-specific objects”
  - Có thể truy vấn sử dụng LINQ, sau đó truy xuất và thao tác dữ liệu như kiểu đối tượng

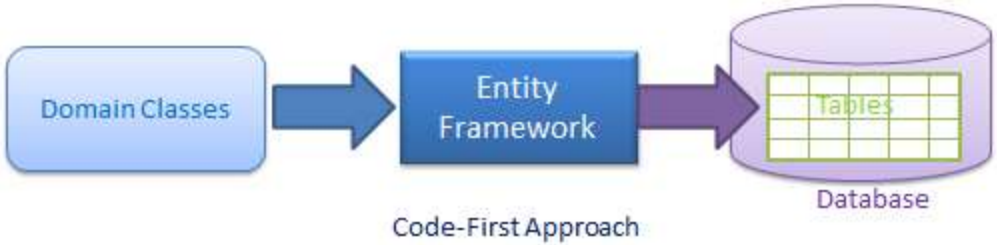
ENTITY FRAMEWORK -EF

- Entity Framework Architecture:
  - EDM (Entity Data Model)



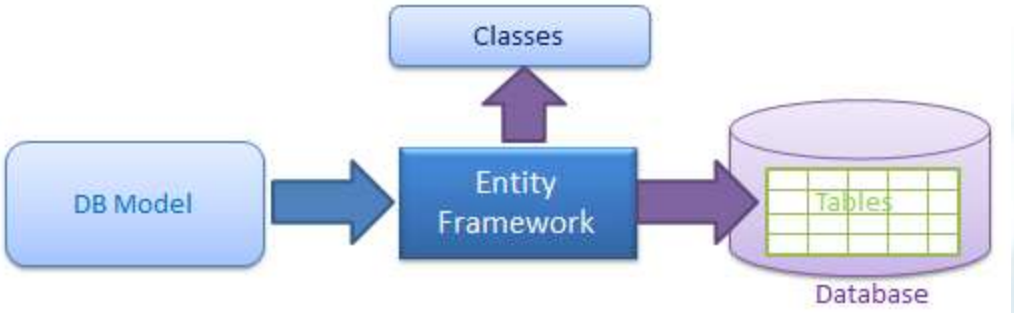
ENTITY FRAMEWORK -EF

- Entity framework cung cấp 3 cách tiếp cận:
  - + Code First
  - + Model First
  - + Database first
- Code First
  - + Viết các classes trước
  - + Sau tạo database từ classes đó.



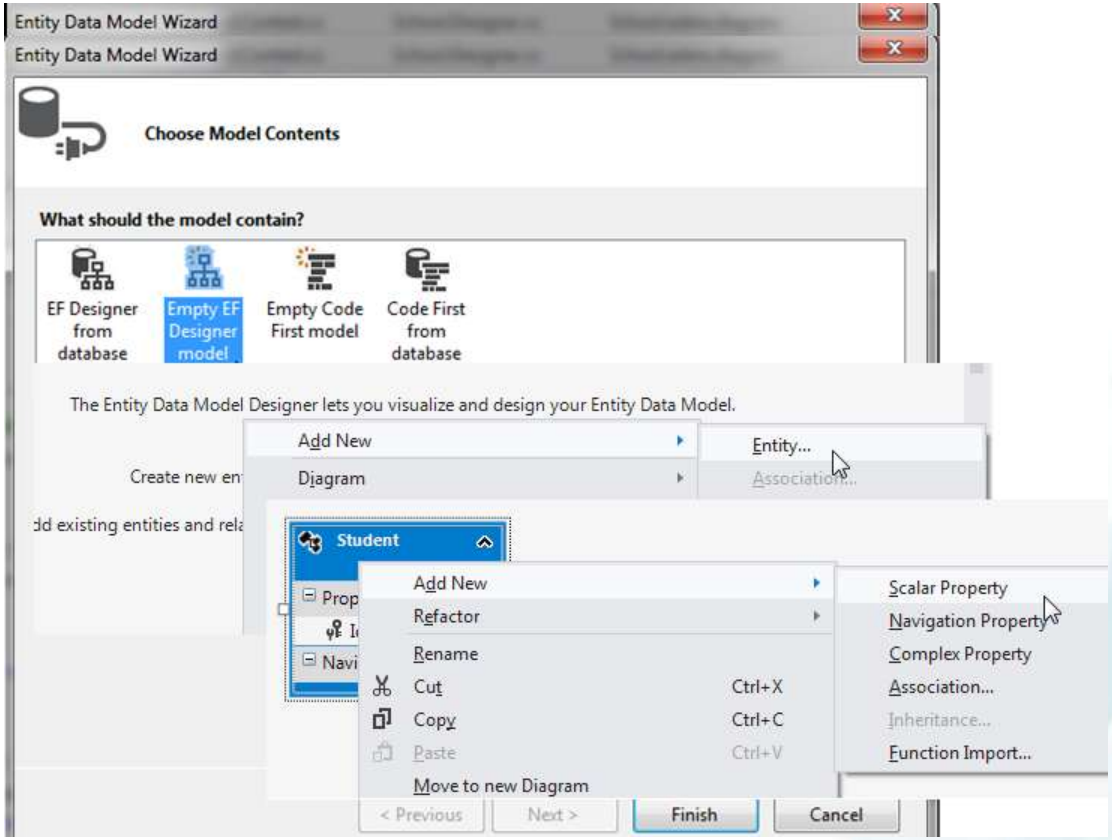
ENTITY FRAMEWORK -EF

- Model First
  - Tạo Entities, relationships, và các kế thừa trực tiếp từ cửa sổ thiết kế
  - Sau sinh database từ model này.

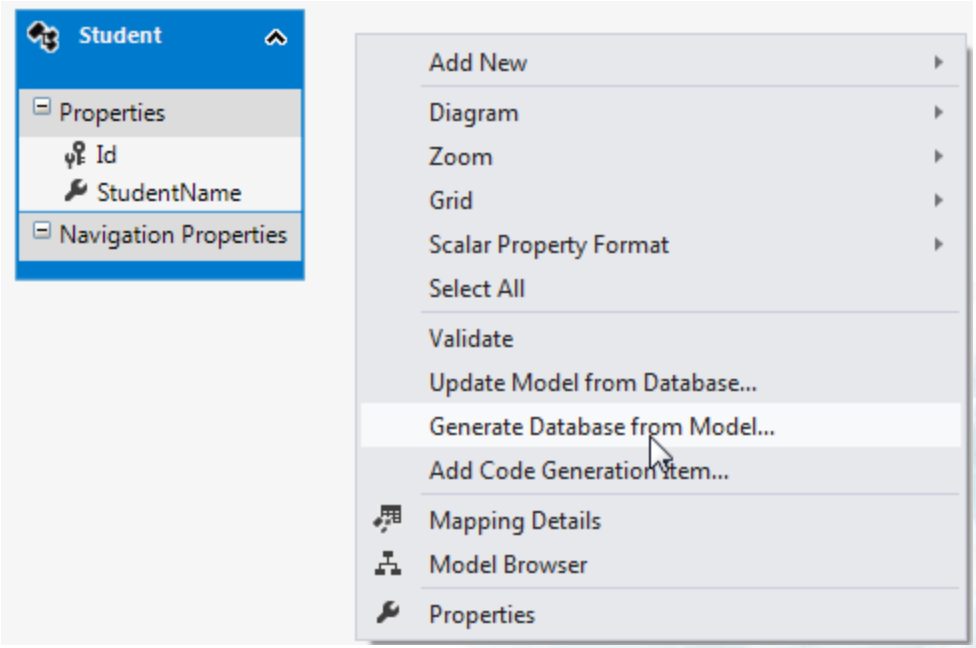


ENTITY FRAMEWORK -EF

- Model First



- Model First
  - + Tạo Database từ model

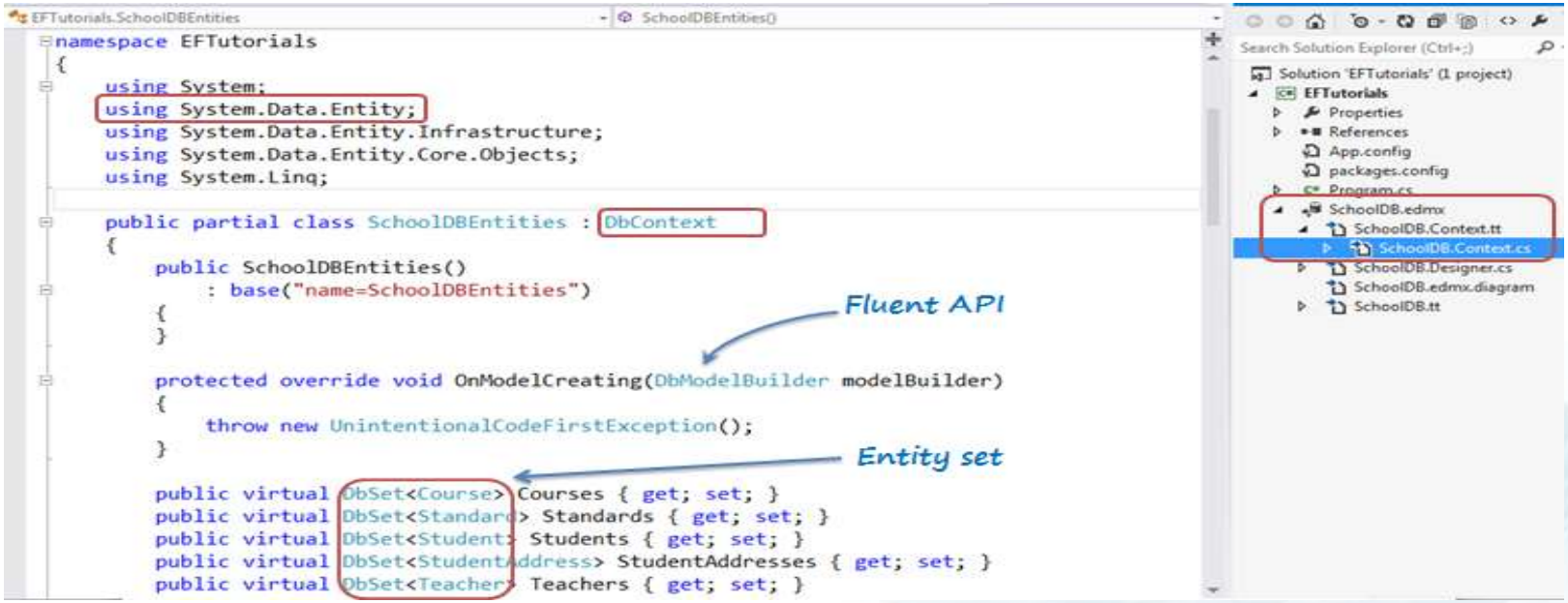


ENTITY FRAMEWORK -EF

- Database first
  - + Tạo EDM, context và entity classes từ database đã tồn tại

ENTITY FRAMEWORK -EF

- DBContext:
  - Các class trong EF được dẫn xuất từ class *System.Data.Entity.DbContext* .
  - Là cầu nối giữa domain or entity classes và database.



ENTITY FRAMEWORK -EF

- DBContext:
  - + DbContext tương tác dữ liệu gồm các hoạt động sau:

  - **EntitySet:** DbContext chứa tập các thực thể (DbSet<TEntity>) đối với tất cả các thực thể ánh xạ đến DB tables.

- DBContext:
  - **Querying:** DbContext chuyển LINQ-to-Entities queries sang SQL query và gửi chúng đến database.
  - **Change Tracking:** Theo dõi các thay đổi xuất hiện trong thực thể sau khi truy vấn từ database.
  - **Persisting Data:** Thực hiện các thao tác Insert, Update and Delete trên database, dựa trên trạng thái thực thể.
  - **Caching:** Lưu trữ các thực thể được truy xuất trong thời gian sống của context class.
  - **Manage Relationship:** DbContext quản lý các quan hệ.
  - **Object Materialization:** DbContext thực thể hóa table data thành entity objects.

ENTITY FRAMEWORK -EF

- DBContext:
  - Khởi tạo DBContext và truy xuất ObjectContext từ DBContext

```
using (var ctx = new SchoolDBEntities())
{
        var objectContext = (ctx as
System.Data.Entity.Infrastructure.IObjectContextAdapter).Obj
ectContext;
    //use objectContext here..
}
```

ENTITY FRAMEWORK -EF

- DBSet Class:
  - Là thể hiện cho tập các thực thể được dùng cho các thao tác:
    - create,
    - read,
    - update,
    - delete

```
EFBasicTutorials.SchoolDBEntities                    SchoolDBEntities()
namespace EFBasicTutorials
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;
    using System.Data.Entity.Core.Objects;
    using System.Linq;

    public partial class SchoolDBEntities : DbContext
    {
        public SchoolDBEntities()
            : base("name=SchoolDBEntities")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Course> Courses { get; set; }
        public virtual DbSet<Standard> Standards { get; set; }
        public virtual DbSet<Student> Students { get; set; }
        public virtual DbSet<StudentAddress> StudentAddresses { get; set; }
        public virtual DbSet<Teacher> Teachers { get; set; }
        public virtual DbSet<View_StudentCourse> View_StudentCourse { get; set; }
```

ENTITY FRAMEWORK -EF

- DBSet Class:
  - Là thể hiện cho tập các thực thể được dùng cho các thao tác:
    - create,
    - read,
    - update,
    - delete

```
EFBasicTutorials.SchoolDBEntities                    SchoolDBEntities()
namespace EFBasicTutorials
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;
    using System.Data.Entity.Core.Objects;
    using System.Linq;

    public partial class SchoolDBEntities : DbContext
    {
        public SchoolDBEntities()
            : base("name=SchoolDBEntities")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Course> Courses { get; set; }
        public virtual DbSet<Standard> Standards { get; set; }
        public virtual DbSet<Student> Students { get; set; }
        public virtual DbSet<StudentAddress> StudentAddresses { get; set; }
        public virtual DbSet<Teacher> Teachers { get; set; }
        public virtual DbSet<View_StudentCourse> View_StudentCourse { get; set; }
```

- DBSet Class:
  - Một số phương thức của class DBSet :
    - Add: Thêm một kiểu thực thể
    - AsNoTracking<Entity>: Trả về truy vấn mà các thực thể không được cached trong DbContext.
    - Attach(Entity): Attaches thực thể vào context với trạng thái Unchanged
    - Create: Tạo ra thể hiện mới của thực thể

- DBSet Class:
  - Một số phương thức của class DBSet :
    - Find(int): Tìm kiếm thực thể
    - Include: Trả về truy vấn ‘non generic LINQ to Entities’ trên DbContext. (kế thừa class DbQuery)
    - Remove: Đánh dấu xóa thực thể
    - SqlQuery: Trả về truy vấn SQL chứa tập các thực thể này.

- DBEntityEntry Class:
  - Được dùng để truy xuất thông tin về một thực thể.
  - Lấy thể hiện của DBEntityEntry sử dụng phương thức Entry của DBContext.

```
//get student whose StudentId is 1
var student = dbcontext.Students.Find(1);
//get DbEntityEntry object for student entity object
var entry = dbcontext.Entry(student);
 //get entity information e.g. full name
string fullname= entry.Entity.GetType().FullName);
```

- Thêm Entity sử dụng DBContext trong ngữ cảnh Disconnected:

```
// create new Student entity
var newStudent = new Student();
newStudent.StudentName = "Bill";
 //create DBContext object
using (var dbCtx = new SchoolDBEntities()) {
 //Add Student object into Students DBset
dbCtx.Students.Add(newStudent);
// call SaveChanges method to save student into database
dbCtx.SaveChanges();
}
```

- Thêm Entity sử dụng DBContext trong ngữ cảnh Disconnected: Hoặc sử dụng Entry

```
// create new Student entity object
 var newStudent = new Student();
//set student name
newStudent.StudentName = "Bill";
//create DBContext object
using (var dbCtx = new SchoolDBEntities()) {
 //Add newStudent entity into DbEntityEntry and mark EntityState to Added
dbCtx.Entry(newStudent).State =
System.Data.Entity.EntityState.Added;
 // call SaveChanges method to save new Student into database
dbCtx.SaveChanges(); }
```

- Sửa thông tin Entity sử dụng DBContext trong ngữ cảnh Disconnected:

```
Student stud;
//1. Get student from DB
using (var ctx = new SchoolDBEntities()) {
 stud = ctx.Students.Where(s => s.StudentName == "New Student1").FirstOrDefault<Student>(); }
//2. change student name in disconnected mode (out of ctx scope)
 if (stud != null) { stud.StudentName = "Updated Student1"; }
```

- Sửa thông tin Entity sử dụng DBContext trong ngữ cảnh Disconnected:

```
//save modified entity using new Context
using (var dbCtx = new SchoolDBEntities()) {
 //3. Mark entity as modified
dbCtx.Entry(stud).State = System.Data.Entity.EntityState.Modified;
//4. call SaveChanges
dbCtx.SaveChanges();
}
```

- Xóa thông tin Entity sử dụng DBContext trong ngữ cảnh Disconnected:

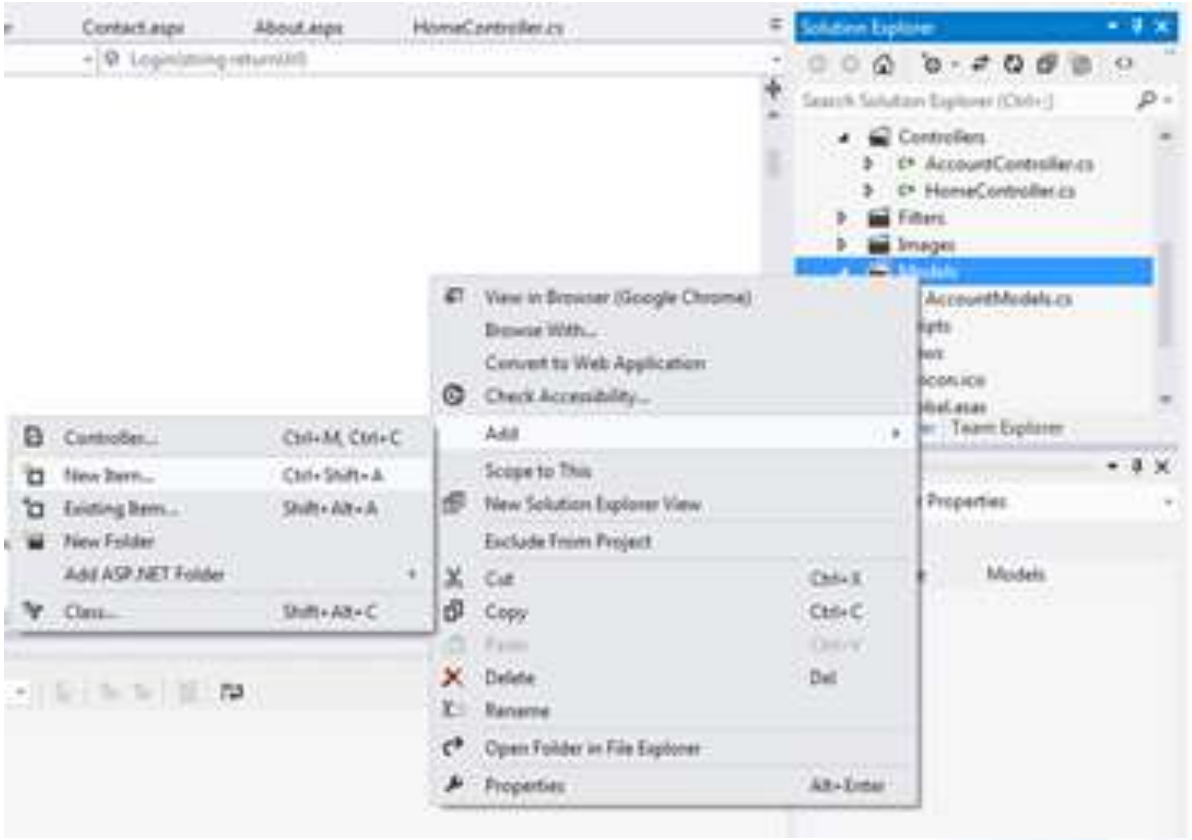
```
Student studentToDelete;

//1. Get student from DB
using (var ctx = new SchoolDBEntities()) {
 studentToDelete = ctx.Students.Where(s => s.StudentName ==
"Student1").FirstOrDefault<Student>(); }

//Create new context for disconnected scenario
using (var newContext = new SchoolDBEntities()) {
newContext.Entry(studentToDelete).State = System.Data.Entity.EntityState.Deleted;
newContext.SaveChanges(); }
```

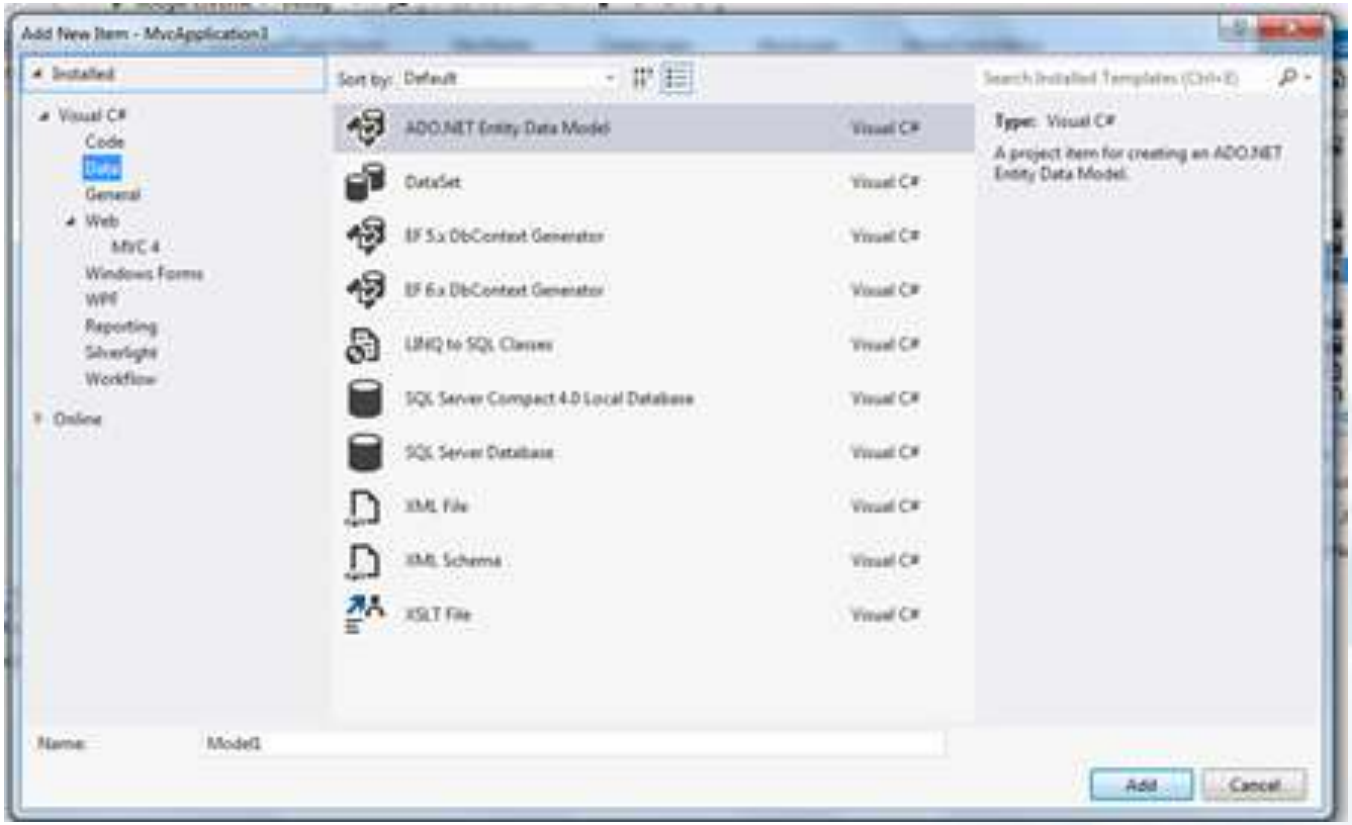
TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Tạo Models từ Database:



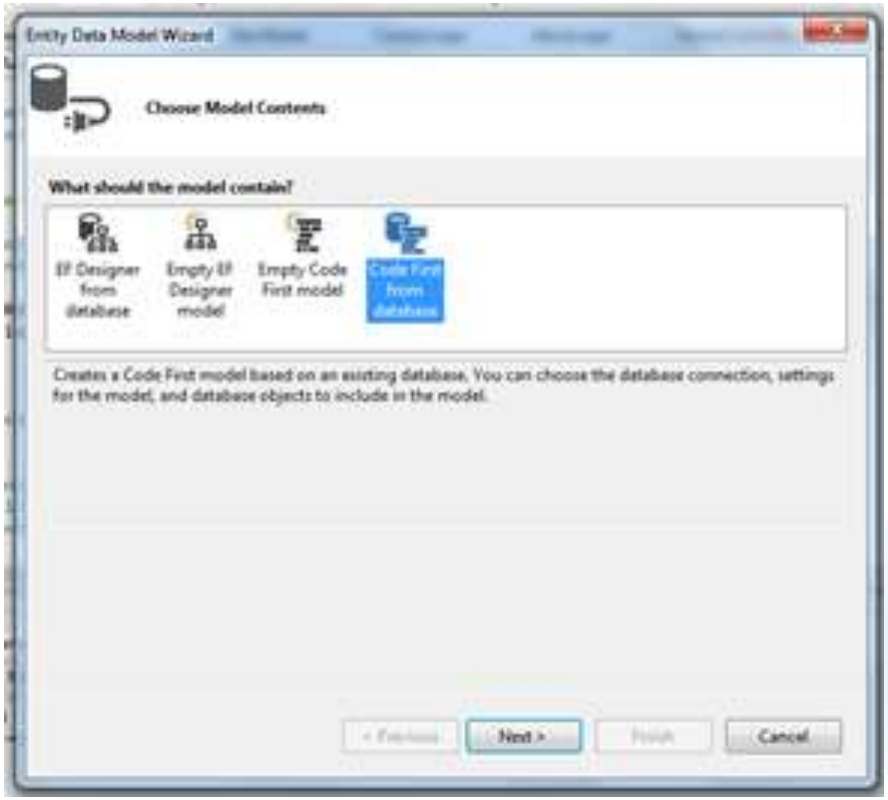
TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Tạo Models từ Database:



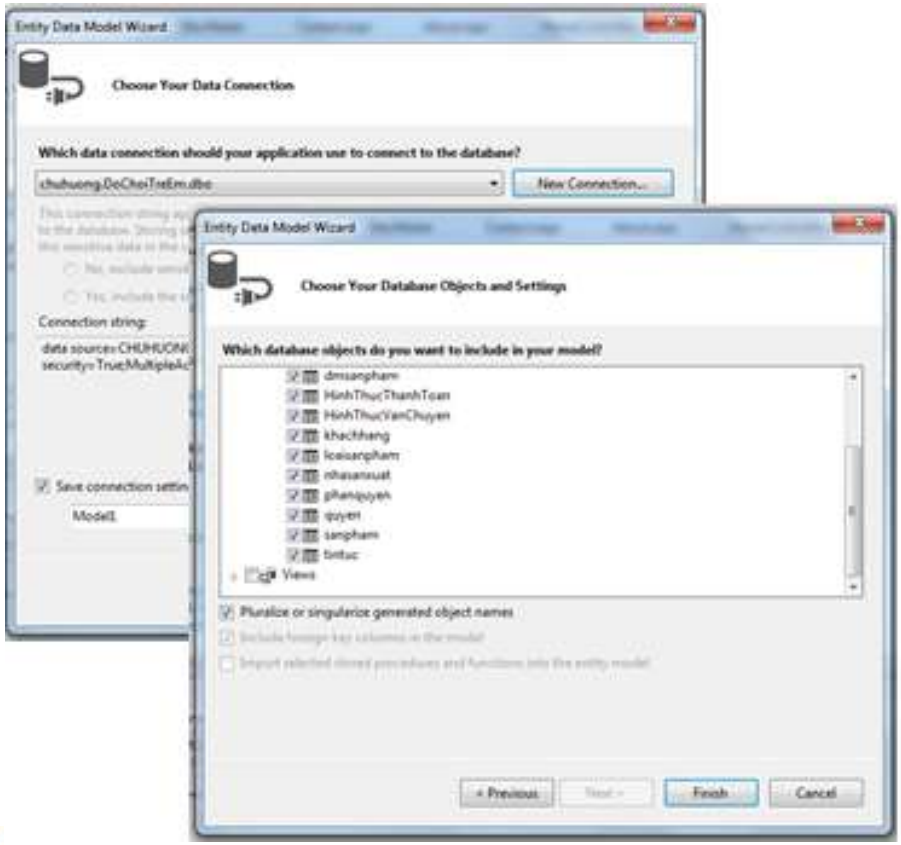
TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Tạo Models từ Database:



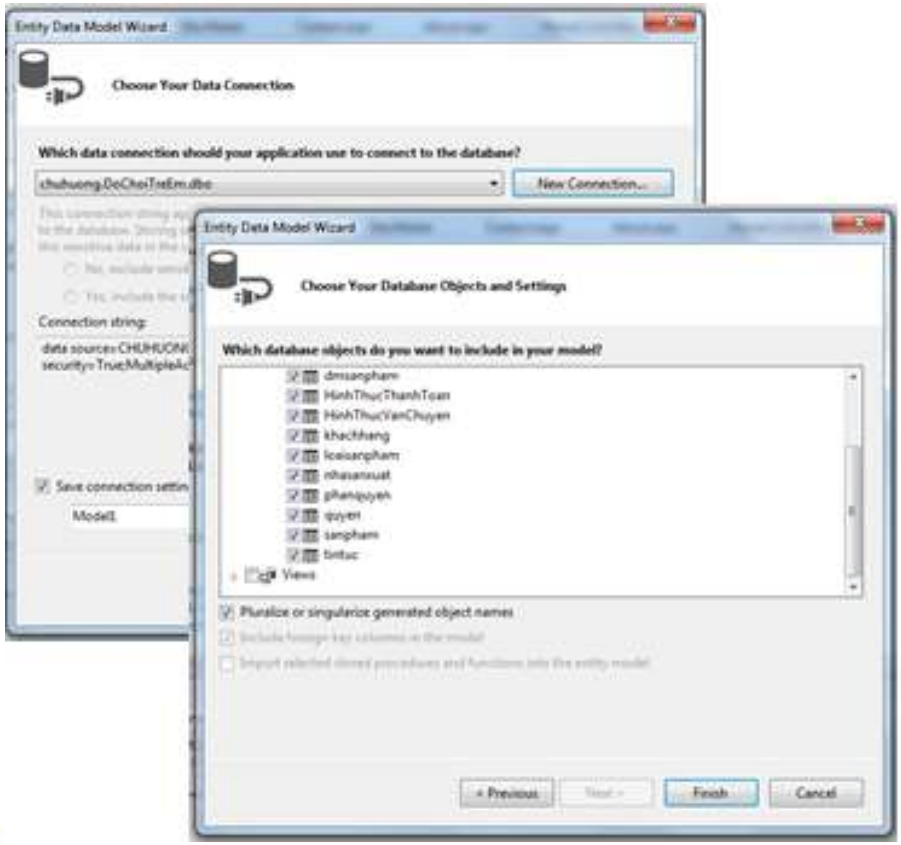
TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Tạo Models từ Database:



TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Tạo Models từ Database:



TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:

```
public partial class ModelTestMVC : DbContext{
    public ModelTestMVC(): base("name=ModelTestMVC") {}
    public virtual DbSet<SACH> SACHes { get; set; }
    public virtual DbSet<THELOAI> THELOAIs { get; set; }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {     modelBuilder.Entity<SACH>()
              .Property(e => e.MaSach)
              .IsUnicode(false);
     }
}
```

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - + Truy xuất dữ liệu với EF:

```
public class FTheLoai{
     private ModelTestMVC context = new ModelTestMVC();
     public IQueryable<THELOAI> TheLoais{
                get { return context.THELOAIs; }
     }
public string InsertTheLoai(THELOAI theloai){
     THELOAI dbEntry = context.THELOAIs.Find(theloai.MaTL);
     if (dbEntry != null){return null}
     context.THELOAIs.Add(theloai);
     context.SaveChanges();
     return theloai.MaTL
 }
```

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:

```
public string UpdateTheLoai(THELOAI theloai)
{
    THELOAI dbEntry = context.THELOAIs.Find(theloai.MaTL);
    if (dbEntry == null){return null;}
    dbEntry.MaTL= theloai.MaTL;
    dbEntry.TenTL = theloai.TenTL;
    context.SaveChanges();
    return theloai.MaTL
}
```

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:

```
public string DeleteTheLoai(THELOAI theloai){
    THELOAI dbEntry = context.THELOAIs.Find(theloai.MaTL);
        if (dbEntry != null){
        context.THELOAIs.Remove(dbEntry);
        context.SaveChanges();
        return theloai.MaTL   }
        return null;
 }
public THELOAI FindTheLoai(string MaTL){
        THELOAI dbEntry = context.THELOAIs.Find(MaTL);
        return dbEntry;
 }
```

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:
    - Sử dụng phương thức SqlQuery và ExecuteSqlCommand : Để thực thi các lệnh Sql.
      - <DataContext>.DataBase.SqlQuery
      - <DataContext>.<EntityName>.SqlQuery
      - <DataContext>.DataBase. ExecuteSqlCommand

    Chú ý: SqlQuery cho phép trả về các thực thể từ database (SELECT) còn ExecuteSqlCommand chỉ thực hiện câu lệnh và trả về trạng thái của code từ Database (INSERT, UPDATE, DELETE)

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:
    - Sử dụng phương thức SqlQuery và ExecuteSqlCommand

```
public List<SACH> GetDanhSach(string MaTL){
      SqlParameter[] idParam={
         new SqlParameter {ParameterName = "MaTL", Value = MaTL },
         new SqlParameter{ParameterName = "Ten", Value = MaTL }};
         var courseList = context.SACHes.SqlQuery("GetSach @MaTL,
@Ten", idParam).ToList<SACH>();
         return courseList;
}
```

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:

```
public List<SACH> GetDanhSach(string MaTL){
      SqlParameter[] idParam={
          new SqlParameter {ParameterName = "MaTL", Value = MaTL },
          new SqlParameter{ParameterName = "Ten", Value = MaTL }};
          var courseList = context.SACHes.SqlQuery("GetSach @MaTL,
@Ten", idParam).ToList<SACH>();
          return courseList;
}
```

TRUY XUẤT DATABASE

- Tạo và kết nối Database với Code First
  - Truy xuất dữ liệu với EF:

```
public int InsertPr(DanhMuc model) {
      DanhMuc dbEntry = context.DanhMucs.Find(model.MaDM);
      if (dbEntry != null) {
              return 0;
          }
       SqlParameter[] idParam ={
        new SqlParameter {ParameterName ="MaDM", Value = model.MaDM },
        new SqlParameter{ParameterName="TenDM",Value = model.TenDM }};
       int courseList = context.Database.ExecuteSqlCommand("EXEC
proInsertDM @MaDM,@TenDM", idParam);
            return courseList;
       }
```

TRUY XUẤT DATABASE

- Truy xuất dữ liệu giữa Model và Controller :

```
private FTheLoai theloaiF = new FTheLoai();

 // GET: /TheLoai/Edit/5
 public ActionResult Edit(string id){
     THELOAI theloai = theloaiF.FindTheLoai(id);
     return View(theloai);
}
// POST: /TheLoai/Edit/5
[HttpPost]
public ActionResult Edit(THELOAI theloai){
   try{   bool thongbao;
          theloaiF.UpdateTheLoai(theloai, out thongbao);
          return RedirectToAction("Index");}
    catch {  return View();}
}
```

MODEL VALIDATION

- Kiểm tra tính hợp lệ Model:

```
public ViewResult CreateHoaDon(HOADON entity){
    if (string.IsNullOrEmpty(entity.HoTen)){
         ModelState.AddModelError(“HoTen", “Chưa điền họ tên");}
    if(ModelState.IsValidField("Date") && DateTime.Now > entity.Date){
        ModelState.AddModelError("Date", “Điền ngày quá khứ");}
    if (!entity.Accepted) {
        ModelState.AddModelError("Accepted", “Bạn phải đồng ý ĐK");}
     if (ModelState.IsValid) {
                // statements to store new Appointment in a
                // repository would go here in a real project
           return View("Completed", entity);}
      else { return View();}
}
```

MODEL VALIDATION

- Hiển thị lỗi:
  - + Sử dụng Validation Summary:

```
@using (Html.BeginForm()){
    @Html.ValidationSummary()
    <p> Họ tên: @Html.EditorFor(m => m.HoTen)</p>
    <p> Ngày: @Html.EditorFor(m => m.Date)</p>
    <p>@Html.EditorFor(m => m.Accepted) Tôi chấp nhận các điều
khoản</p>
    <input type="submit" value="Make Booking" />
}
```

MODEL VALIDATION

- Hiển thị lỗi:
  - + Sử dụng ValidationMessageFor:

```
@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
    <p>@Html.ValidationMessageFor(m => m.HoTen)</p>
    <p>Your name: @Html.EditorFor(m => m.HoTen)</p>
    <p>@Html.ValidationMessageFor(m => m.Date)</p>
    <p> Ngày: @Html.EditorFor(m => m.Date)</p>
    <p>@Html.ValidationMessageFor(m => m. Accepted)</p>
    <p>@Html.EditorFor(m => m.Accepted) Tôi chấp nhận các điều
khoản </p>
    <input type="submit" value="Make Booking" />
}
```

MODEL VALIDATION

- Sử dụng Validation Annotations:

```
public class RegisterModel {
      [Required]
      [Display(Name = "User name")]
      public string UserName { get; set; }

      [Required]
      [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.",
MinimumLength = 6)]
      [DataType(DataType.Password)]
       [Display(Name = "Password")]
       public string Password { get; set; }

       [DataType(DataType.Password)]
       [Display(Name = "Confirm password")]
       [Compare("Password", ErrorMessage = "The password and confirmation password do not
match.")]
       public string ConfirmPassword { get; set; }
  }
```

MODEL VALIDATION

- Sử dụng Validation Annotations:
  - [Required]: Giá trị không được rỗng hoặc là các chuỗi chỉ có các spaces. Nếu muốn whitespace là hợp lệ, sử dụng

    [Required(AllowEmptyStrings = true)].

```
[Required]
public string UserName { get; set; }
```

MODEL VALIDATION

- Sử dụng Validation Annotations:
  - [StringLength(n)]: Chỉ định độ dài lớn nhất của dữ liệu. Ta có thể chỉ định độ dài tối thiểu sử dụng thuộc tính MinimumLength

```
[StringLength(30)]
public string UserName { get; set; }

[StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
public string Password { get; set; }
```

MODEL VALIDATION

- Sử dụng Validation Annotations:
  - [RegularExpression("pattern")]: Chỉ định dữ liệu phải đúng khuôn định dạng.

```
[RegularExpression(@"[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}")]
public string Email { get; set; }
```

  - [Range(n,m)] : Chỉ định dữ liệu kiểu số phải nằm trong khoảng. Ta có thể chỉ định biên một phía bằng cách sử dụng hằng số MinValue, MaxValue, ví dụ [Range(int.MinValue, 50)]

```
[Range(0,10)]
public int Marked { get; set; }
```

MODEL VALIDATION

- Sử dụng Validation Annotations:
  - [Compare("MyOtherProperty")]: Hai thuộc tính bắt buộc phải có cùng giá trị.

```
[Required]
public string Password { get; set; }

[Display(Name = "Confirm password")]
[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
public string ConfirmPassword { get; set; }
```

MODEL VALIDATION

- Sử dụng Validation Annotations:
  - Chú ý:
    - Sử dụng ErrorMessage property để chỉ định 'custom error message'.
    - DataType attribute cung cấp thông tin về kiểu dữ liệu tại thời điểm runtime.
    - Display attribute: Hiển thị tên cho thuộc tính model.

```
[DataType(DataType.Password)]
[Display(Name = "Confirm password")]
[Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
public string ConfirmPassword { get; set; }
```

MODEL VALIDATION

- Defining Self-Validating Models:

```
public class Appointment:IValidatableObject {
    public string Name { get; set; }
    [DataType(DataType.Date)]
    public DateTime Date { get; set; }
    public IEnumerable<ValidationResult> Validate(ValidationContext
validationContext) {
    List<ValidationResult> errors = new List<ValidationResult>();
    if (string.IsNullOrEmpty(Name)){
       errors.Add(new ValidationResult("Please enter your name"));}
    if (DateTime.Now > Date) {
       errors.Add(new ValidationResult("Please enter a date in the
future"));}
     return errors;
       }
}
```

Tương tác với SQL Server LocalDB

- SQL Server Express LocalDB:
  - SQL Server Express Database Engine có LocalDB. Nó nhẹ, tự động khởi động dựa trên và chạy trong chế độ người dùng.
  - SQL Server Express có chế độ thực thi đặc biệt với LocalDB cho khả năng làm việc với database như .mdf files. *.mdf file của database được lưu trữ trong thư mục App_data của ứng dụng.

Tương tác với SQL Server LocalDB

- Tạo chuỗi kết nối trong tệp web.config:

```
<connectionStrings>
    <add name="DefaultConnection"
providerName="System.Data.SqlClient" connectionString="Data
Source=(LocalDb)\v11.0;Initial Catalog=aspnet-TestMVC-
20160112095552;Integrated Security=SSPI; AttachDBFilename=
|DataDirectory|\aspnet-TestMVC-20160112095552.mdf" />
<add name="MyDBContext" providerName="System.Data.SqlClient"
connectionString="Data Source=(LocalDb)\v11.0;Initial Catalog=aspnet-
TestMVC-20160112095552;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|\MyDB.mdf" />
<add name="ModelTestMVC" connectionString="data
source=CHUHUONG;initial catalog=TestMVC;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
providerName="System.Data.SqlClient" />
  </connectionStrings>
```

THẢO LUẬN – CÂU HỎI