

LAB 5: XÂY DỰNG CHƯƠNG TRÌNH

Mục đích:

Bài Lab này giúp sinh viên xây dựng chương trình theo mô hình 3 tầng từ các thiết kế dữ liệu, giao diện và xử lý.

Yêu cầu:

- Sinh viên ngồi theo nhóm để thực hành, có thể thực hành thêm ở nhà.
- Nhóm phải thực hiện xong bài Lab 4.

Kiến thức lý thuyết

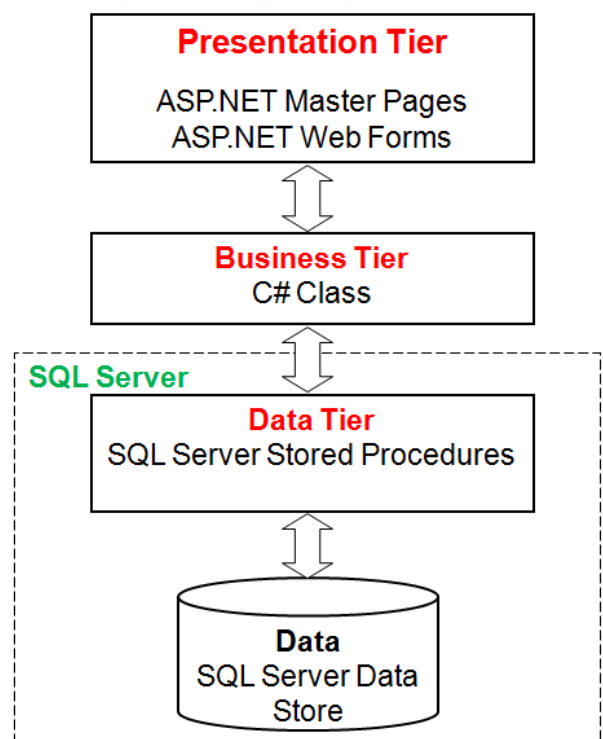
1. Mô hình 3 tầng

Mô hình 3 tầng giúp chia nhỏ chức năng của ứng dụng thành các thành phần dựa trên công việc mà thành phần đó thực hiện, sau đó nhóm mỗi thành phần đó thành một tầng. Mô hình kiểu này cho phép các lập trình viên có thể cập nhật, thay đổi chức năng của mỗi tầng riêng biệt mà không phá vỡ code của các tầng khác.

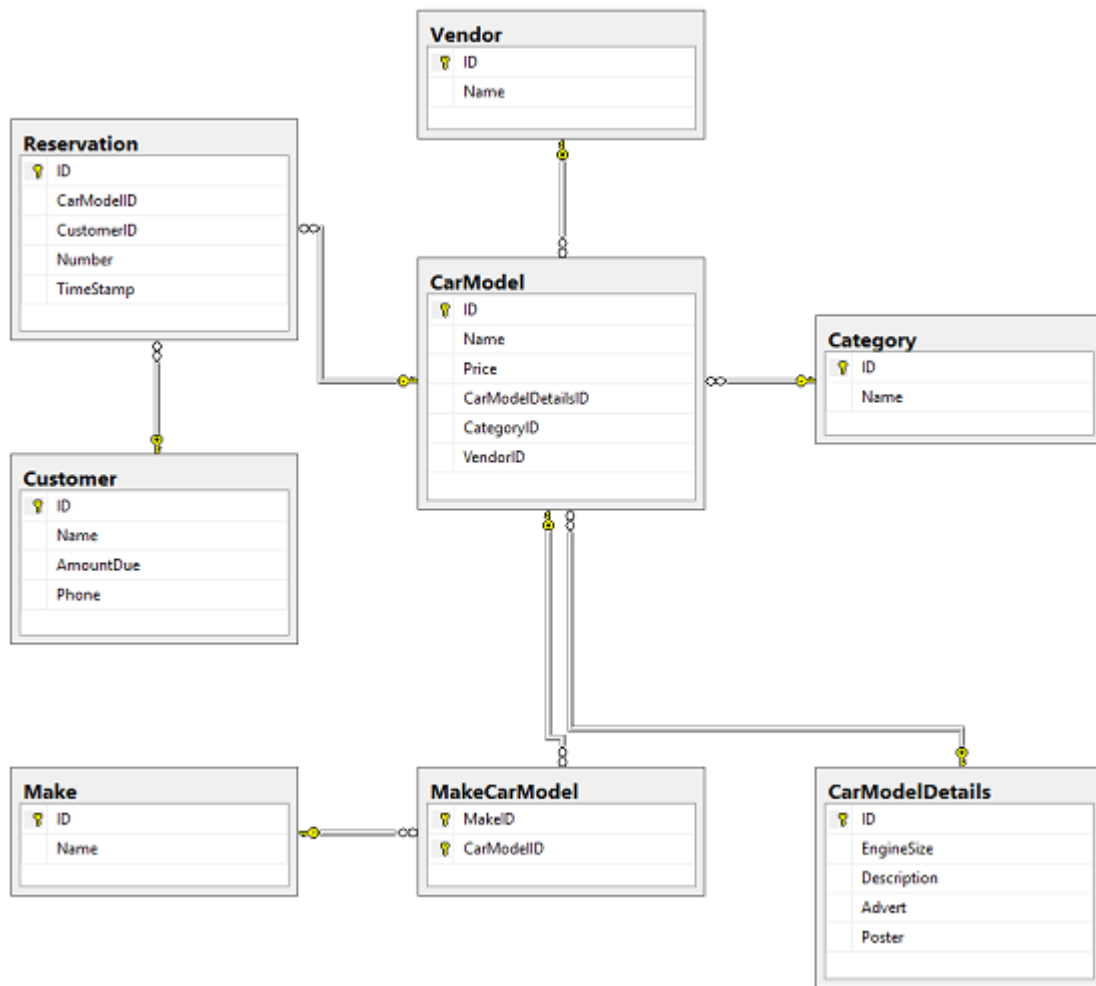
Người ta thường chia thành ba tầng với các code liên quan như sau:

- Tầng Presentation (Giao diện): cho phép kết nối giao diện
- Tầng Business (Giao dịch): cho phép thực hiện các giao dịch
- Tầng Access (Kết nối): thực hiện các kết nối với CSDL.

Trên bình diện thực tế, người ta có thể tạo ra các code với các kiểu mô hình như: MVC, mô hình n-tier, ...

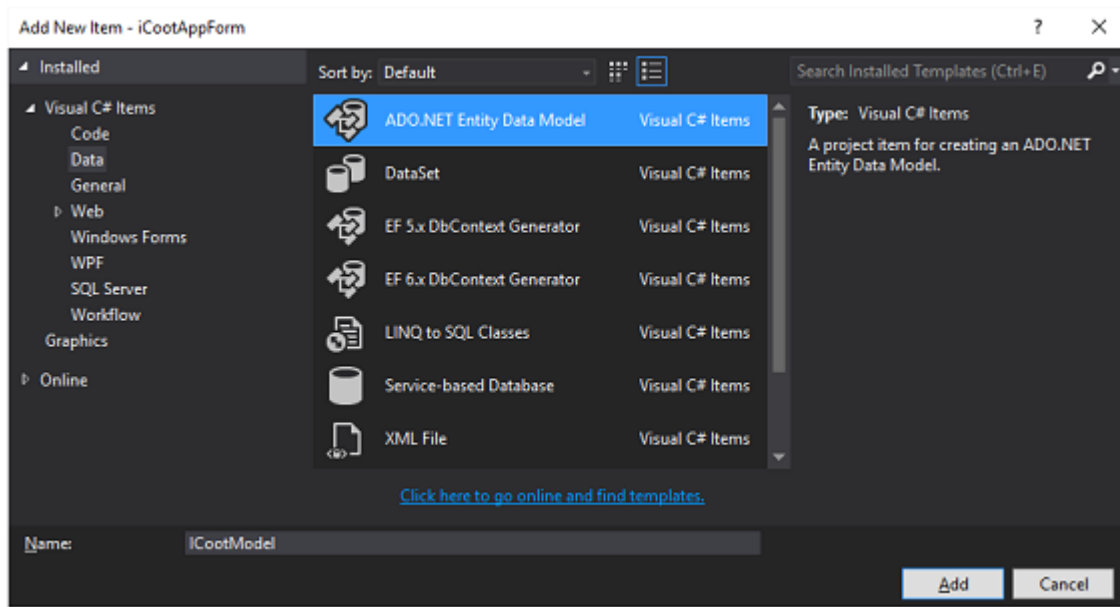


2. Ví dụ mô hình cơ sở dữ liệu (Tầng Access)

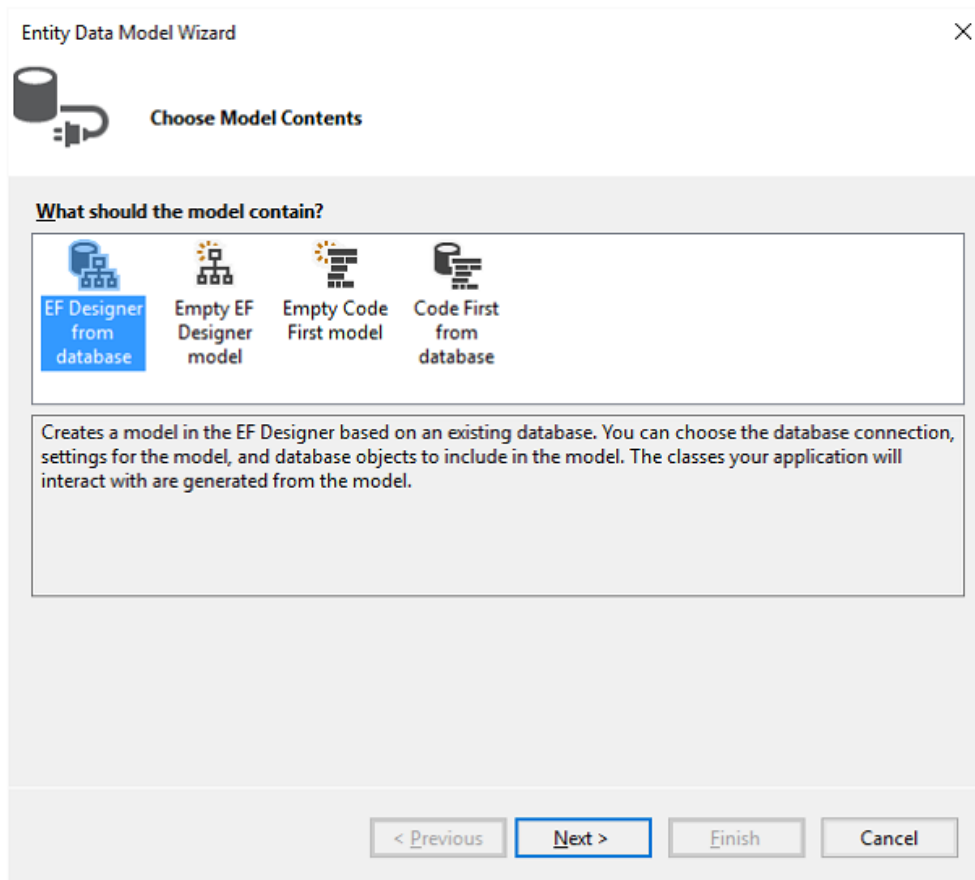


3. Ví dụ kết nối và thao tác dữ liệu sử dụng Entity Framework (Tầng Access và Business).

Dựa trên cơ sở dữ liệu được tạo ra, chúng ta sẽ kết nối trực tiếp trên bộ công cụ phát triển Visual Studio, để tạo mô hình Entity Framework mới ta sẽ thêm mới một item (Add->New Item) chọn ADO.NET Entity Data Model trong tab Data trong màn hình Wizard hỗ trợ như hình dưới:

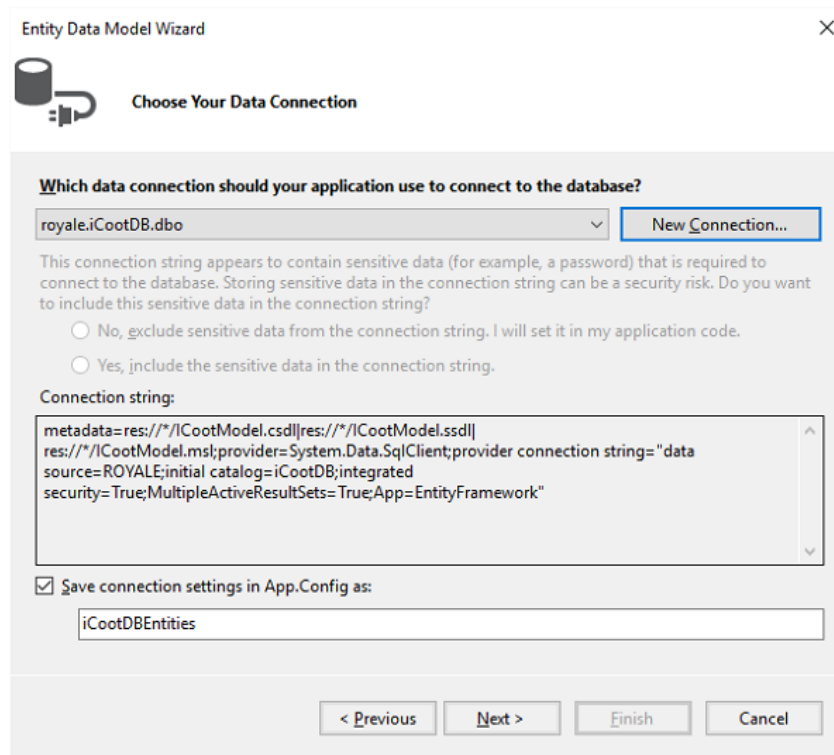


Đến giai đoạn này, ta chọn EF Designer from database để kết nối tới cơ sở dữ liệu sẵn có:



Sau khi chọn cơ sở dữ liệu và kết nối thành công, ta có màn hình Wizard hiển thị các thông số kết nối được thêm vào file cấu hình app.config:

THỰC HÀNH CÔNG NGHỆ PHẦN MỀM



Entity Data Model Wizard

Choose Your Data Connection

Which data connection should your application use to connect to the database?

royale.iCootDB.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Connection string:

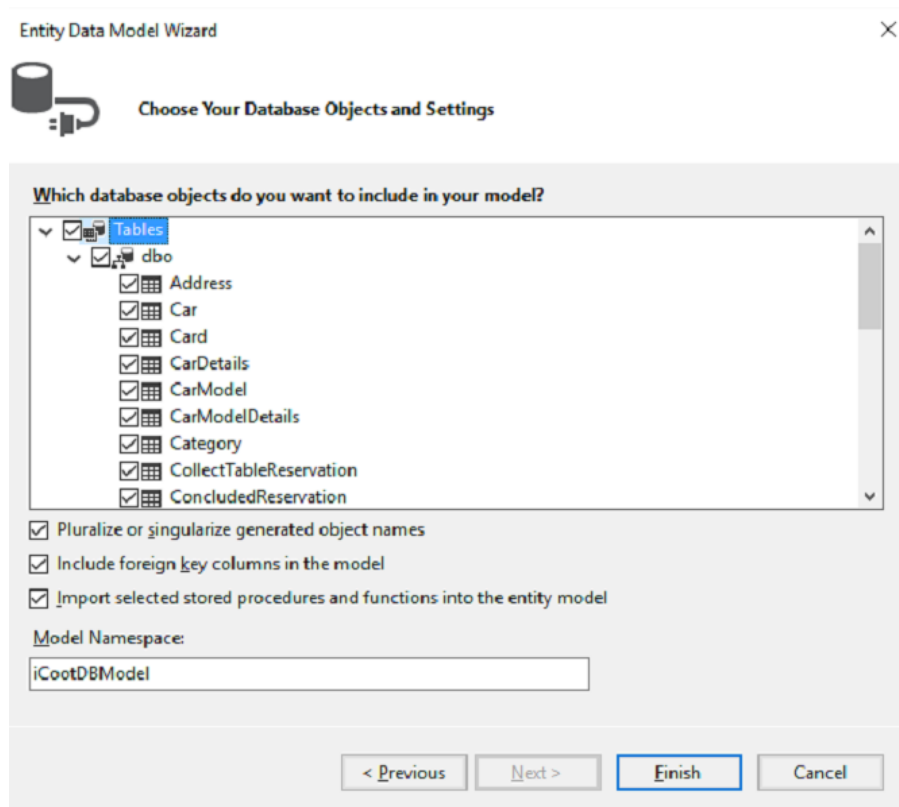
```
metadata=res://*/ICootModel.csdl|res://*/ICootModel.ssdl|
res://*/ICootModel.msl;provider=System.Data.SqlClient;provider connection string="data
source=ROYALE;initial catalog=iCootDB;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework"
```

☒ Save connection settings in App.Config as:

iCootDBEntities

< Previous Next > Finish Cancel

Màn hình tiếp theo, sau khi kết nối thành công, ta sẽ chọn các bảng dữ liệu cần ánh xạ vào mô hình trong tầng Business. Ở đây ta có thể chọn bảng, View và các hàm Store procedure:



Entity Data Model Wizard

Choose Your Database Objects and Settings

Which database objects do you want to include in your model?

☒ Tables

☒ dbo

- ☒ Address
- ☒ Car
- ☒ Card
- ☒ CarDetails
- ☒ CarModel
- ☒ CarModelDetails
- ☒ Category
- ☒ CollectTableReservation
- ☒ ConcludedReservation

☒ Pluralize or singularize generated object names

☒ Include foreign key columns in the model

☒ Import selected stored procedures and functions into the entity model

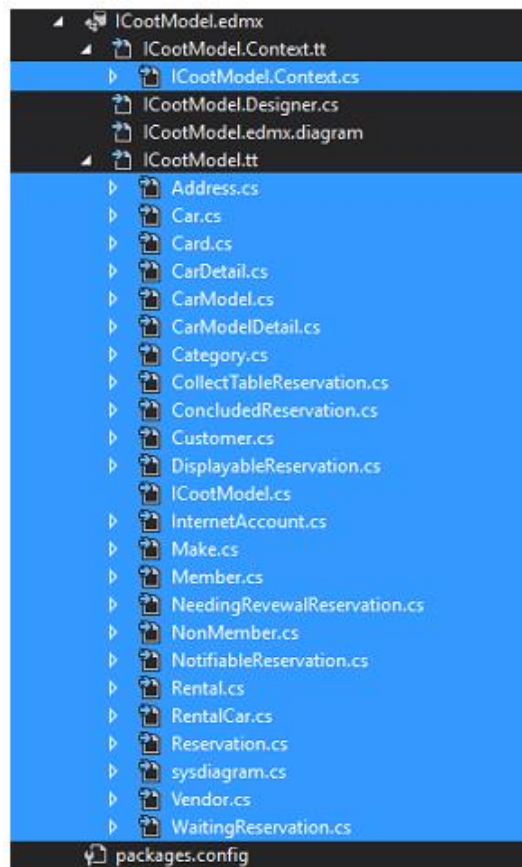
Model Namespace:

iCootDBModel

< Previous Next > Finish Cancel

THỰC HÀNH CÔNG NGHỆ PHẦN MỀM

Đến thời điểm này trong dự án sẽ xuất hiện tập tin với đuôi .EDMX chứa mô phỏng trực quan mối quan hệ giữa các bảng được ánh xạ thành các đối tượng.



Ví dụ một lớp được sinh ra tự động dựa trên bảng dữ liệu thực tế như sau:

```
public partial class CarModel
{
    public CarModel()
    {
        this.Reservations = new HashSet<Reservation>();
        this.Makes = new HashSet<Make>();
    }

    public int ID { get; set; }
    public string Name { get; set; }
    public Nullable<decimal> Price { get; set; }
    public Nullable<int> CarModelDetailsID { get; set; }
    public Nullable<int> CategoryID { get; set; }
    public Nullable<int> VendorID { get; set; }

    public virtual CarModelDetail CarModelDetail { get; set; }
    public virtual Category Category { get; set; }
    public virtual Vendor Vendor { get; set; }
    public virtual ICollection<Reservation> Reservations { get; set; }
    public virtual ICollection<Make> Makes { get; set; }
}
```

Tiếp theo ta sẽ thực hiện các thao tác dữ liệu có thể được thực hiện trong tầng Business như lấy tất cả dữ liệu, lấy theo khóa, thêm, xóa sửa và tìm kiếm.

```
public CarModel Single(int id)
{
    using (iCootDBEntities db = new iCootDBEntities())
    {
        CarModel carModel = db.CarModels.Find(id);
        return carModel;
    }
}
```

```
public List<CarModel> All()
{
    using (iCootDBEntities db = new iCootDBEntities())
    {
        List<CarModel> carModels = db.CarModels.ToList();
        return carModels;
    }
}
```

```
public int Add(CarModel carModel)
{
    using (iCootDBEntities db = new iCootDBEntities())
    {
        db.CarModels.Add(carModel);
        return db.SaveChanges();
    }
}
```

```
public int Update(CarModel updateCarModel, CarModel oldCarModel)
{
    using (iCootDBEntities db = new iCootDBEntities())
    {
        oldCarModel = db.CarModels.Find(oldCarModel.ID);
        oldCarModel.Name = updateCarModel.Name;
        return db.SaveChanges();
    }
}
```

```
public int Delete(int id)
{
    using (iCootDBEntities db = new iCootDBEntities())
    {
        CarModel carModel = this.Single(id);
        db.CarModels.Remove(carModel);
        return db.SaveChanges();
    }
}
```

Cuối cùng, sinh viên tạo thêm tầng giao diện người dùng (Tầng Presentation) sử dụng 2 tầng trình bày trên đây để xây dựng phần mềm của nhóm.

Kết quả

Sau khi thực hiện xong Lab 05, sinh viên có kết quả là phần mềm ứng dụng và kiểm thử (khuyến khích) cho phần mềm đó để báo cáo phần thực hành và thi cuối kỳ.

Kết thúc bài Lab