

LẬP TRÌNH WEB



Nguyễn Thị Mai Trang

1

Nội dung

- Chương 1: Tổng quan
- Chương 2: Ngôn ngữ PHP
- Chương 3: Controller – View – Model
- Chương 5: HTML Helpers
- **Chương 5: Làm việc với Cơ sở dữ liệu**

2

Chương 5

Làm việc với Cơ sở dữ liệu

3

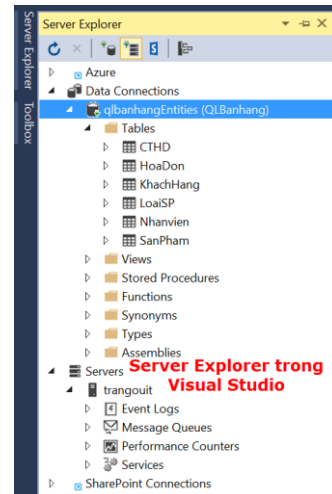
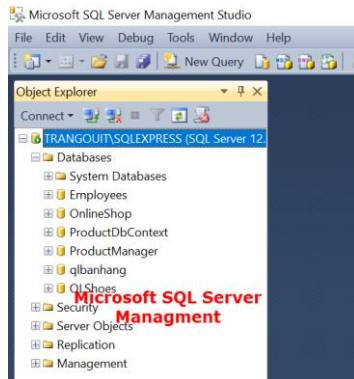
NỘI DUNG

-
- Microsoft SQL Server
 - Entity Framework
 - Lambda Expressions
 - LINQ

4

5.1 Microsoft SQL Server

- Hệ quản trị cơ sở dữ liệu Microsoft SQL Server



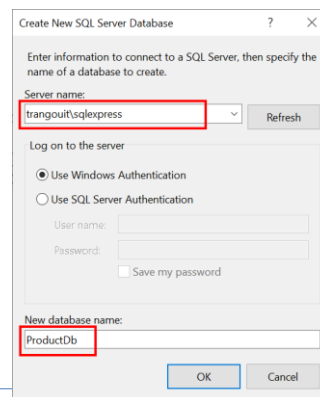
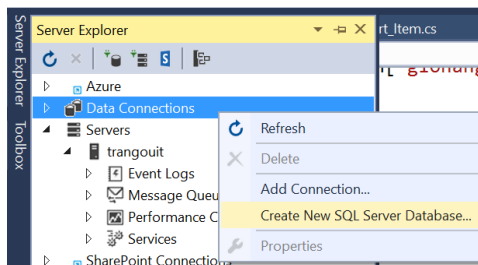
Lập trình web

5

5

5.1.1 Server database

- Cơ sở dữ liệu được lưu trữ trên Server của hệ quản trị cơ sở dữ liệu
- Tạo server database:
 - Sử dụng Microsoft SQL Server Management Studio
 - Tạo trực tiếp trong môi trường VS.NET:



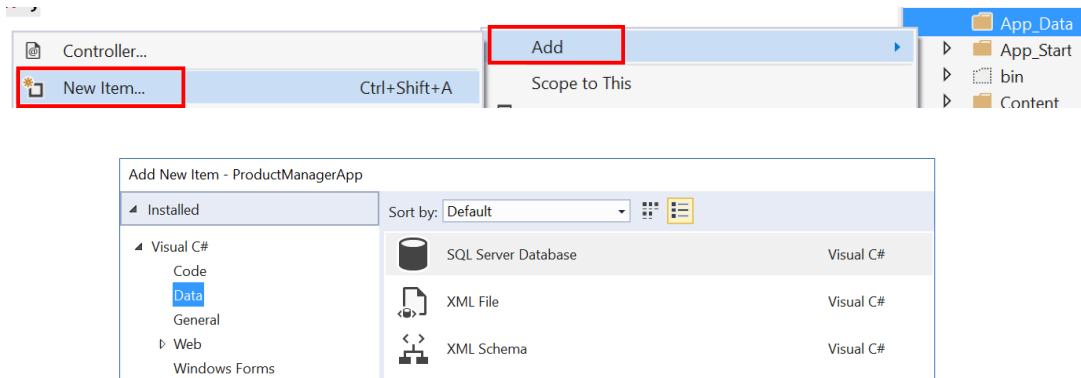
Lập trình web

6

6

5.1.2 Local database

- Cơ sở dữ liệu nằm trong thư mục App_Data của ứng dụng Web
- Tạo local database



Lập trình web

7

7

5.1.3 Chuỗi kết nối

- Chuỗi kết nối đến cơ sở dữ liệu nằm trong tập tin Web.config trong thư mục gốc của dự án web.

```
<configuration>
  <configSections>
    <!-- For more information on Entity Framework configuration, visit http://go.
    <section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.
  </configSections>
  <connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=(LocalDb)\MSSQLLo
    <add name="qlbanhangEntities" connectionString="metadata=res://*/Models.QLBar
  </connectionStrings>
```

- Chuỗi kết nối **DefaultConnection**: thường là chuỗi kết nối tới tập tin *.mdf chứa dữ liệu về user và role

Lập trình web

8

8

Chuỗi kết nối (tt)

- Chuỗi kết nối **qlbanhangEntities**: chuỗi kết nối đến EF SQL Server có dạng:

```
<add name="qlbanhangEntities"
connectionString="metadata=res://*/Models.QLBanhangEF.csdl|res://*/Models.QLBanhangEF.ss
dl|res://*/Models.QLBanhangEF.msl;provider=System.Data.SqlClient;provider connection
string=&quot;data source=TRANGOUI\SQLEXPRESS;initial catalog=qlbanhang;integrated
security=True;MultipleActiveResultSets=True;App=EntityFramework&quot;;"
providerName="System.Data.EntityClient" />
```

- data source=TRANGOUI\SQLEXPRESS: SQLServer mặc định cài trên máy tính
- initial catalog=qlbanhang: tên cơ sở dữ liệu
- integrated security=True: không cần tài khoản đăng nhập

Chuỗi kết nối (tt)

- Chuỗi kết nối đến LocalDatabase:

```
<add name="MyDbConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;
AttachDbFilename=|DataDirectory|\SchoolDB.mdf;
Initial Catalog=SchoolDB;Integrated Security=True"
providerName="System.Data.SqlClient" />
```

- Khi triển khai lên host, chỉ cần thay đổi chuỗi kết nối phù hợp với cấu hình trên máy chủ. Ví dụ:

```
<add name="qlbanhangEntities"
connectionString="metadata=res://*/Models.QLBanhangEF.csdl|res://*/Models.QLBanhangEF.ssdl|res:/
/*Models.QLBanhangEF.msl;provider=System.Data.SqlClient;provider connection string=&quot;data
source= 12.16.10.1;initial catalog=qlbanhang; User Id=myuser;Password=123456; integrated
security=False;MultipleActiveResultSets=True;App=EntityFramework&quot;;"
providerName="System.Data.EntityClient" />
```

5.2 Entity Framework

- Entity Framework (EF) là một thành phần của .NET Framework
- EF cho phép các nhà phát triển Web tương tác với dữ liệu quan hệ theo phương pháp hướng đối tượng.
- Giúp lập giảm thiểu việc viết mã nguồn lập trình để truy cập và tương tác với cơ sở dữ liệu.
- Trong ASP.NET MVC có ba cách sử dụng EF
 - Entity Framework Code First
 - Entity Framework Database First
 - Entity Framework Model First

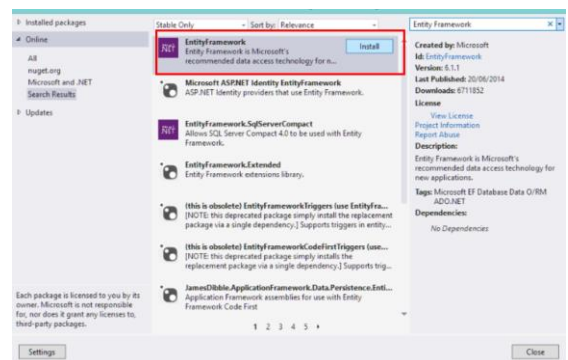
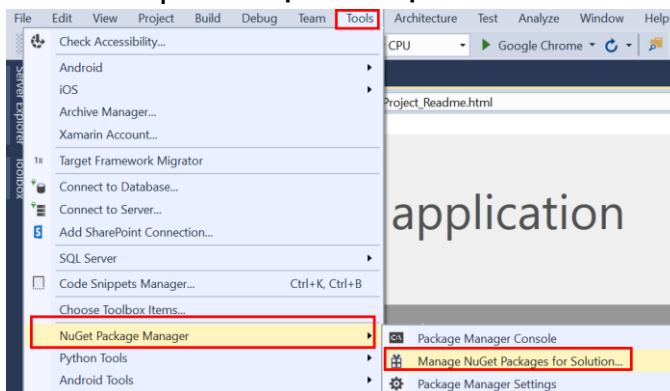
Lập trình web

11

11

Entity Framework

- Cài EF phải được cài đặt trước:



- Hoặc vào Package Manager Console, gõ **Install-Package EntityFramework**

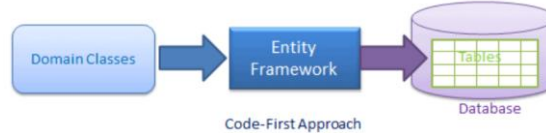
Lập trình web

12

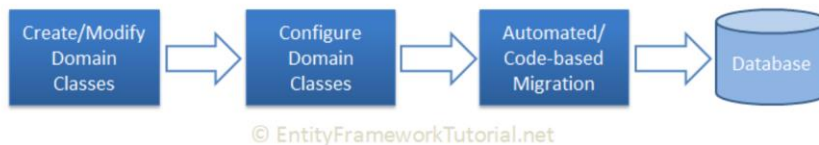
12

5.2.1 Entity Framework Code First

- EF tự động tạo các table trong cơ sở dữ liệu dựa trên các lớp trong model.
- EF phiên bản 6.1 về sau có thể được dùng với các cơ sở dữ liệu đã có sẵn bằng cách tạo ra các lớp dựa trên cơ sở dữ liệu đó.



- Các bước sinh database



Entity Framework Code First (tt)

- Một vài quy ước cấu hình mặc định của EF Code First:
 - Tên bảng: ClassName + s: ví dụ class Product → EF tạo ra table có tên Products.
 - Khóa chính:
 - Thuộc tính Id hoặc ClassNameld
 - Nếu kiểu dữ liệu là số nguyên: → auto-incrementing.
 - Khóa ngoại: EF tự động thiết lập mối quan hệ giữa các table trong database dựa vào các thuộc tính cùng tên và có kiểu dữ liệu giống nhau.
 - Cascade delete: mặc định cho tất cả các loại mối quan hệ.
 - Relationship:
 - Quan hệ One-to-Many được chuyển đổi tự động
 - Quan hệ One-to-One và Many-to-Many cần phải cấu hình

Entity Framework Code First (tt)

- Ánh xạ kiểu dữ liệu giữa C# và SQL Server:

C#	SQL Server
int	int
string	nvarchar(Max)
decimal	decimal(18,2)
float	real
byte[]	varbinary(Max)
datetime	datetime
bool	bit

C#	SQL Server
byte	tinyint
short	smallint
long	bigint
double	float
char	No mapping
sbyte	No mapping (throws exception)
object	No mapping

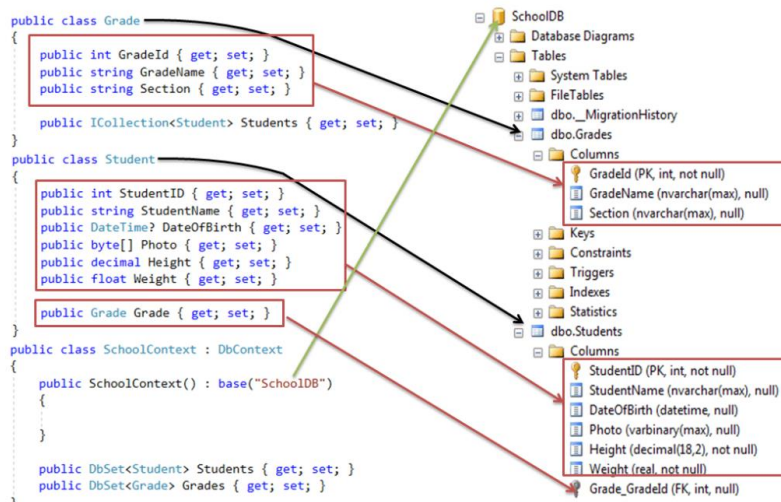
Lập trình web

15

15

Entity Framework Code First (tt)

- Ví dụ:



Lập trình web

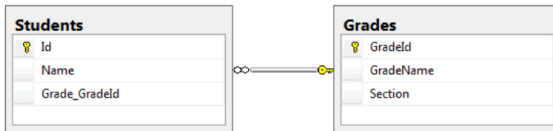
Nguồn: <https://www.entityframeworktutorial.net/>

16

16

Entity Framework Code First (tt)

- Relationship: One-to-Many
 - Cách 1:

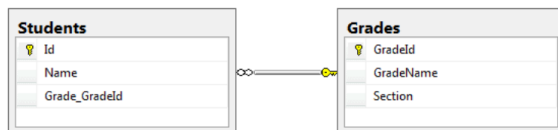


```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
    public Grade Grade { get; set; }
}

public class Grade
{
    public int GradeId { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }
}
```

Entity Framework Code First (tt)

- Relationship: One-to-Many
 - Cách 2



```
public class Student
{
    public int StudentId { get; set; }
    public string StudentName { get; set; }
}

public class Grade
{
    public int GradeId { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

Entity Framework Code First (tt)

- Relationship:
One-to-Zero-or-One

```
public class Student
{
    public int StudentId { get; set; }
    public string StudentName { get; set; }
    public virtual StudentAddress Address { get; set; }
}

public class StudentAddress
{
    public int StudentAddressId { get; set; }
    public string Address1 { get; set; }
    public string Address2 { get; set; }
    public string City { get; set; }
    public int Zipcode { get; set; }
    public string State { get; set; }
    public string Country { get; set; }
    public virtual Student Student { get; set; }
}
```

Lập trình web

19

19

Entity Framework Code First (tt)

- Relationship:
Many-to-Many

```
public class Student {
    public Student()
    {
        this.Courses = new HashSet<Course>();
    }
    public int StudentId { get; set; }
    [Required]
    public string StudentName { get; set; }
    public virtual ICollection<Course> Courses {get; set;}
}

public class Course {
    public Course()
    {
        this.Students = new HashSet<Student>();
    }
    public int CourseId { get; set; }
    public string CourseName { get; set; }
    public virtual ICollection<Student> Students {get; set;}
}
```

Lập trình web

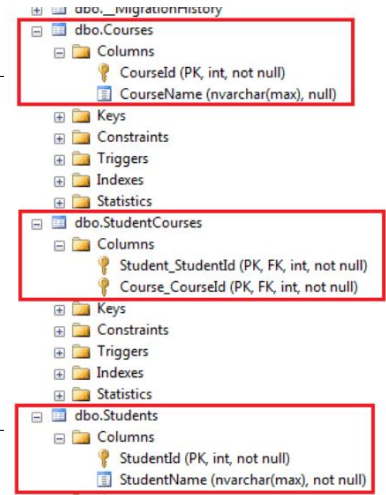
20

20

Entity Framework Code First (tt)

• Relationship: Many-to-Many

```
public class SchoolDbContext : DbContext
{
    public SchoolDbContext() : base("SchoolDB-DataAnnotations")
    {
    }
    public DbSet<Student> Students { get; set; }
    public DbSet<Course> Courses { get; set; }
    protected override void OnModelCreating(DbModelBuilder builder)
    {
        base.OnModelCreating(builder);
    }
}
```



Lập trình web

21

21

Entity Framework Code First (tt)

• Khởi tạo database trong class DbContext

– Gọi base constructor không tham số:

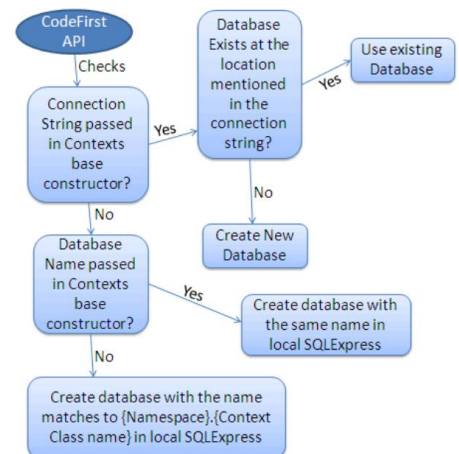
- Tạo database trong local SQLEXPRESS server có tên: {Namespace}. {Context class name}

```
namespace SchoolApp
{
    public class MyContext: DbContext {
        public MyContext(): base(){ }
    }
}
```

SchoolApp.MyContext

– Gọi base constructor có tham số là một chuỗi
→ tạo database với tên là chuỗi tham số

```
public class Context: DbContext
{
    public Context(): base("MySchoolDB") { }
}
```



Lập trình web

22

22

Entity Framework Code First (tt)

- Khởi tạo database trong class DbContext (tt)
 - Gọi base constructor có tham số là một chuỗi kết nối khai báo trong web.config
 - tạo database với tên tương ứng trong chuỗi connectionString

```
namespace SchoolApp
{
    public class Context: DbContext
    {
        public SchoolDbContext() : base("name=SchoolDBConnection") { }
    }
}
```

```
<connectionStrings>
  <add name="SchoolDBConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;
    AttachDbFilename=|DataDirectory|\SchoolDB.mdf;
    Initial Catalog=SchoolDB;Integrated Security=True"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Lập trình web

23

23

Entity Framework Code First (tt)

- Migration:
 - Công cụ tự động cập nhật lược đồ CSDL khi mô hình thay đổi mà không mất dữ liệu.
 - Có hai loại: Automated Migration, Code-based Migration
 - Enable-Migrations: sẽ sinh tự động file cấu hình Configuration.cs
 - Enable-Migrations: thiết lập Automated Migration mặc định
 - Enable-migrations –EnableAutomaticMigration:\$true: thiết lập Automated Migration tự động
 - Enable-Migrations -ContextTypeName ApplicationName.DirName.NamedDbContext
 - Với DirName là thư mục chứa class NamedDbContext
 - Khởi tạo và cập nhật dữ liệu:
 - add-migration initial
 - update-database -ConfigurationTypeName Configuration

Lập trình web

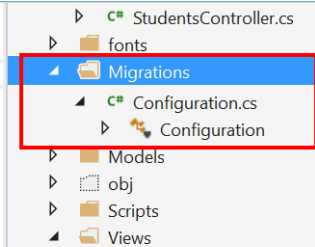
24

24

Entity Framework Code First (tt)

- enable-migrations –EnableAutomaticMigration:\$true

```
1 reference
internal sealed class Configuration :
    DbMigrationsConfiguration<MigrationDemo.Models.SchoolDbContext>
{
    0 references
    public Configuration()
    {
        AutomaticMigrationsEnabled = true;
    }
}
```



- Hoặc
 - enable-migrations -ContextTypeName ProjectName.Data. ContextClassName

Entity Framework Code First (tt)

- Các cách khởi tạo database với EF Code First
 - CreateDatabaseIfNotExists
 - DropCreateDatabaseIfModelChanges
 - DropCreateDatabaseAlways
 - Custom DB Initializer

```
public class SchoolDbContext: DbContext
{
    public SchoolDbContext(): base("SchoolDBConnectionString")
    {
        Database.SetInitializer<SchoolDbContext>(new CreateDatabaseIfNotExists<SchoolDbContext>());
        //Database.SetInitializer<SchoolDbContext>(new DropCreateDatabaseIfModelChanges<SchoolDbContext>());
        //Database.SetInitializer<SchoolDbContext>(new DropCreateDatabaseAlways<SchoolDbContext>());
        //Database.SetInitializer<SchoolDbContext>(new SchoolDBInitializer());
    }
    //...
}
```

Entity Framework Code First (tt)

- Trong file Configuration.cs, tạo dữ liệu tự động trong phương thức Seed

```
protected override void Seed(ContosoUniversity.DAL.SchoolContext context)
{
    // This method will be called after migrating to the latest version.
    // You can use the DbSet<T>.AddOrUpdate() helper extension method
    // to avoid creating duplicate seed data. E.g.
    // context.People.AddOrUpdate(
    //     p => p.FullName,
    //     new Person { FullName = "Andrew Peters" },
    //     new Person { FullName = "Brice Lambson" },
    //     new Person { FullName = "Rowan Miller" }
    // );
}
```

Lập trình web

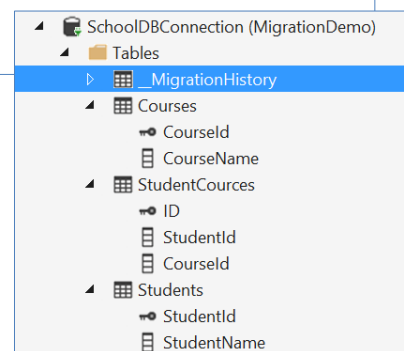
27

27

Entity Framework Code First – Ví dụ

- Tạo ứng dụng ASP.NET MVC
- Cấu hình chuỗi kết nối trong web.config

```
<add name="SchoolDBConnection" connectionString="Data Source=(LocalDb)\MSSQLLocalDB;
AttachDbFilename=|DataDirectory|\SchoolDB.mdf;
Initial Catalog=SchoolDB;Integrated Security=True"
providerName="System.Data.SqlClient" />
```



Lập trình web

28

28

Entity Framework Code First – Ví dụ

- Tạo các lớp trong Models, mỗi lớp tương ứng một table trong database

```
public class Course
{
    public int CourseId { get; set; }
    public string CourseName { get; set; }
    public virtual ICollection<StudentCourses> StudentCourses { get; set; }
}

public class Student
{
    public int StudentId { get; set; }
    [Required]
    public string StudentName { get; set; }
    public virtual ICollection<StudentCourses> StudentCourses { get; set; }
}

public class StudentCourses
{
    public int ID { get; set; }
    public int StudentId { get; set; }
    public int CourseId { get; set; }
    public virtual Student Student { get; set; }
    public virtual Course Course { get; set; }
}
```

Lập trình web

29

29

Entity Framework Code First – Ví dụ

- Trong Models, tạo class DbContext

```
using System.Data.Entity;
namespace MigrationDemo.Models {
    public class SchoolDbContext:DbContext {
        public SchoolDbContext() : base("name=SchoolDBConnection") { }
        public DbSet<Student> Students { get; set; }
        public DbSet<Course> Courses { get; set; }
        public DbSet<StudentCourses> StudentCourses { get; set; }
        protected override void OnModelCreating(DbModelBuilder modelBuilder) {
            base.OnModelCreating(modelBuilder);
        }
    }
}
```

- Trong Package Manager Console
 - Enable-Migrations -ContextTypeName MigrationDemo.Models.SchoolDbContext

Lập trình web

30

30

Entity Framework Code First – Ví dụ

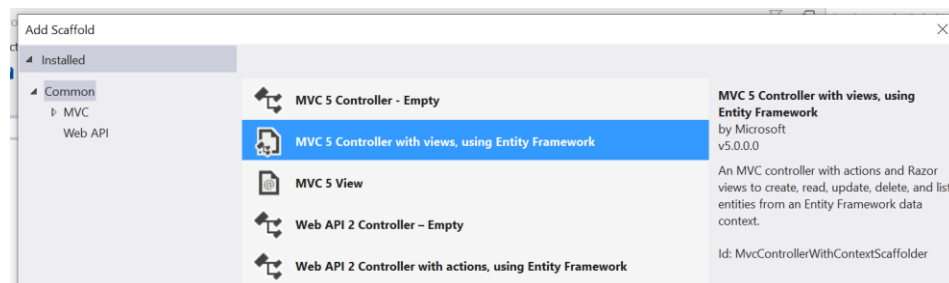
- Viết code khởi tạo dữ liệu trong Configuration.cs

```
public Configuration(){ AutomaticMigrationsEnabled = true; }
protected override void Seed(MigrationDemo.Models.SchoolDbContext context){
    context.Courses.AddOrUpdate(
        new Course { CourseName = "Course 01" },
        new Course { CourseName = "Course 02" } //...
    );
    context.Students.AddOrUpdate(
        new Student { StudentName = "Mary" },
        new Student { StudentName = "Tom" } //...
    );
    context.StudentCourses.AddOrUpdate(
        new StudentCourses { CourseId = 1, StudentId = 1 },
        new StudentCourses { CourseId = 1, StudentId = 2 } //...
    );
}
```

31

Entity Framework Code First – Ví dụ

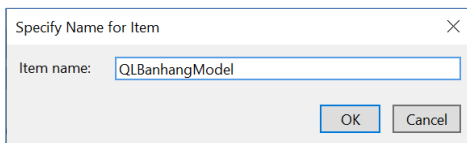
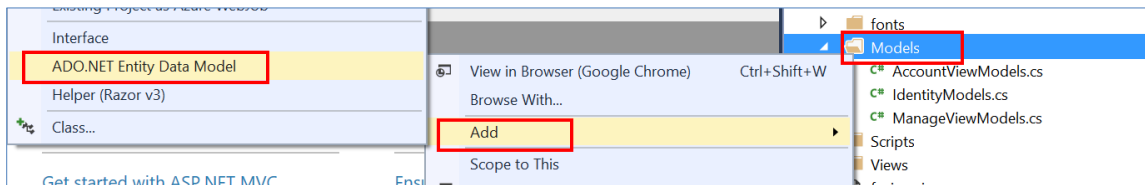
- Trong Package Manager Console
 - add-migration initial
 - update-database -ConfigurationTypeName Configuration
- Tạo các Scaffolded Controller và View



32

5.2.2 Entity Framework Database First

- Tạo ứng dụng MVC
- Trong Model, tạo ADO.NET Entity Data Model



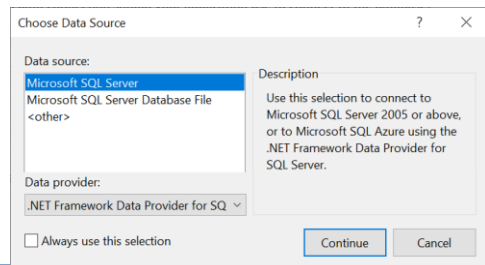
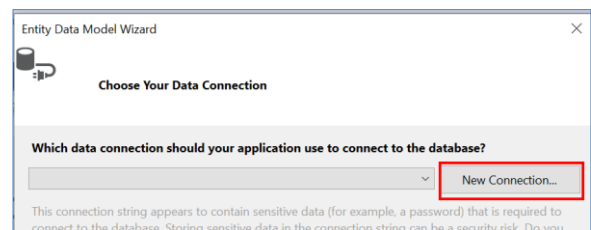
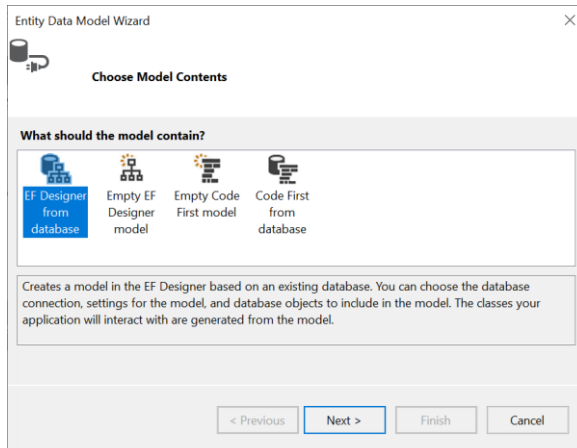
Lập trình web

33

33

Entity Framework Database First (tt)

- Trong Entity Data Model Wizard



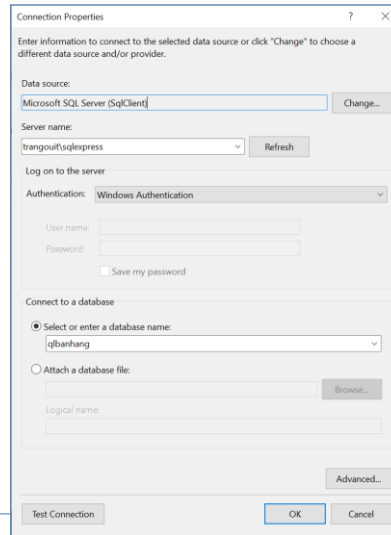
Lập trình web

34

34

Entity Framework Database First (tt)

- Chọn Server name
- Chọn database cần kết nối đến
- Next



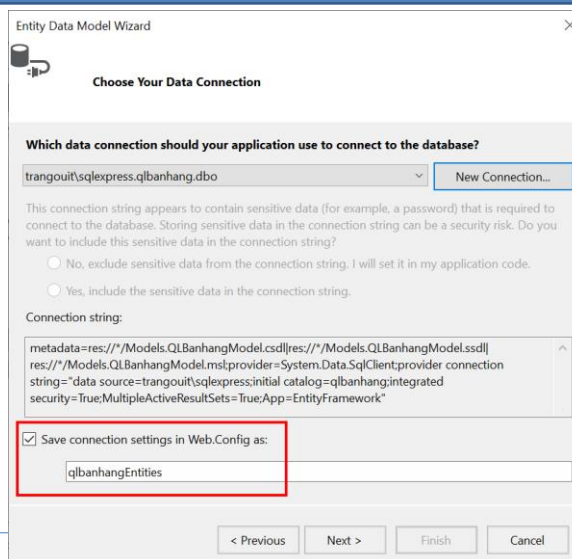
Lập trình web

35

35

Entity Framework Database First (tt)

- Đặt tên cho chuỗi kết nối sẽ lưu trong Web.config



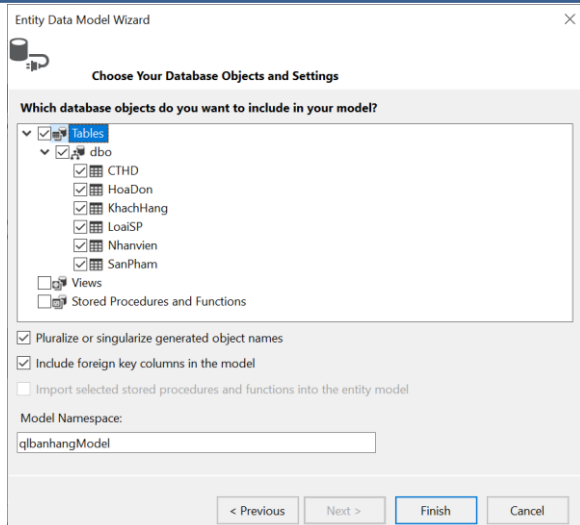
Lập trình web

36

36

Entity Framework Database First (tt)

- Chọn các table trong database
- Finish



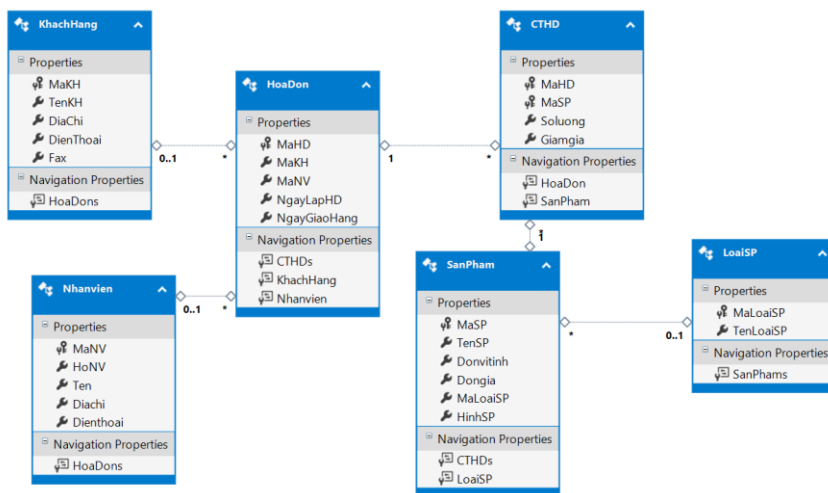
Lập trình web

37

37

Entity Framework Database First (tt)

- EF model được tạo tự động



Lập trình web

38

38

Entity Framework Database First (tt)

- Thư mục Models phát sinh các file:

- File QLBanhangModelContext.cs chứa một class kế thừa từ class DbContext, với các property tương ứng với các lớp model cho mỗi table trong database.
- Build project

```
public partial class qlbanhangEntities : DbContext
{
    0 references
    public qlbanhangEntities()
        : base("name=qlbanhangEntities")
    {
    }

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

    0 references
    public virtual DbSet<CTHD> CTHDs { get; set; }
    0 references
    public virtual DbSet<HoaDon> HoaDons { get; set; }
    0 references
    public virtual DbSet<KhachHang> KhachHangs { get; set; }
    0 references
    public virtual DbSet<LoaiSP> LoaiSPs { get; set; }
    0 references
    public virtual DbSet<Nhanvien> Nhanviens { get; set; }
    0 references
    public virtual DbSet<SanPham> SanPhams { get; set; }
}
```

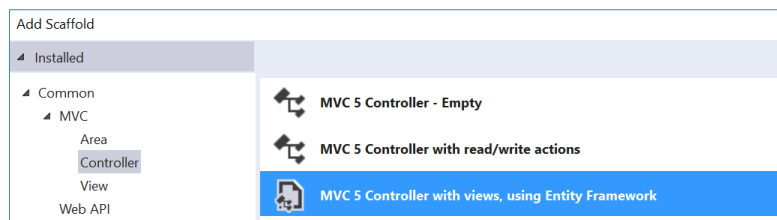
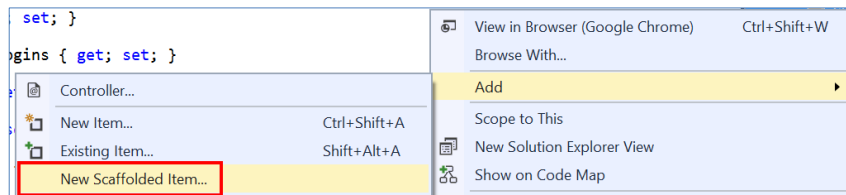
Lập trình web

39

39

Entity Framework Database First (tt)

- Tạo Controller: Add → New Scaffolded Item.



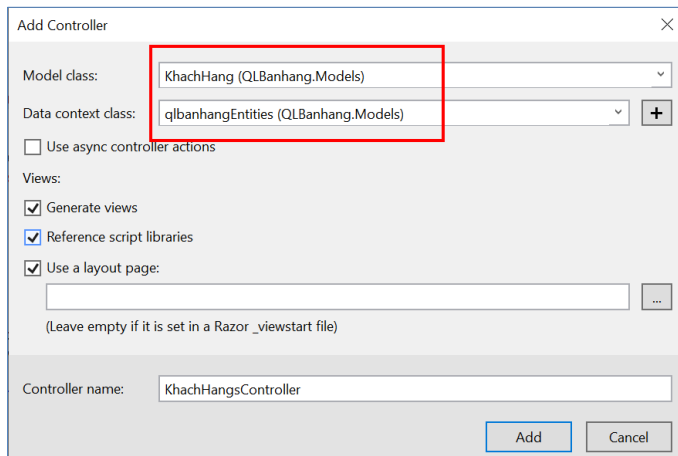
Lập trình web

40

40

Entity Framework Database First (tt)

- Chọn Model tương ứng với Controller
- Thực hiện tương tự với các bảng còn lại



Lập trình web

41

41

Entity Framework Database First (tt)

- Trong tập tin Home\Index.cshtml, bổ sung các ActionLink để mở các view tương ứng

```
<div>
    @Html.ActionLink("Danh sách sản phẩm", "Index", "Sanphams")<br />
    @Html.ActionLink("Danh sách khách hàng", "Index", "Khachangs")
</div>
```

- Chạy ứng dụng, truy cập các link xem kết quả

Index				
Create New				
TenKH	DiaChi	DienThoai	Fax	
Anh Hang	120 Ha Ton Quyen	8171717	084088171717	Edit Details Delete
Bong Hong	24 Ky Con		084088800256	Edit Details Delete
Em Anh	6 Ky Hoa		084088852258	Edit Details Delete
Ho Han	8 Pham van Khoe	8430156	084088430156	Edit Details Delete

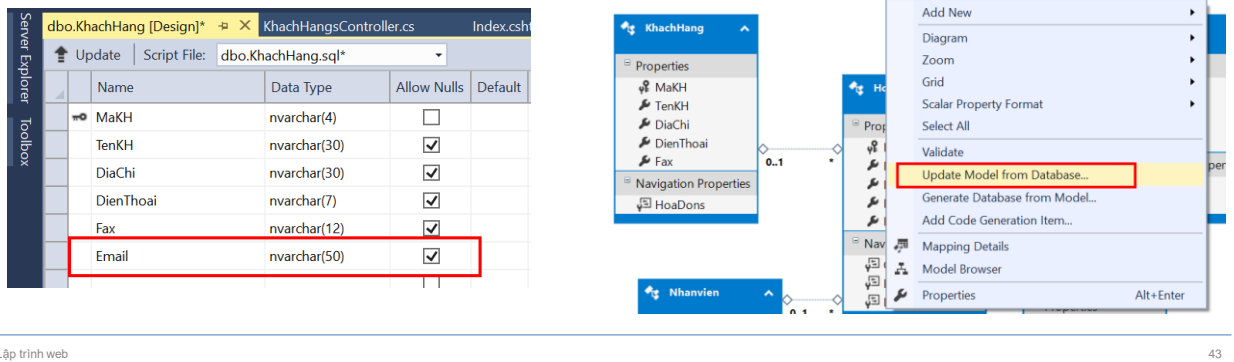
Lập trình web

42

42

Entity Framework Database First (tt)

- Thay đổi cấu trúc bảng trong CSDL:
 - Mở bảng trong chế độ thiết kế trong Server Explorer
 - Ví dụ thêm field Email trong bảng Khachhang
 - Mở tập tin .edmx → Update Model From Database



The screenshot shows the SQL Server Enterprise Designer interface. On the left, the 'Server Explorer' pane displays the 'dbo.KhachHang [Design]' table. The table has the following columns:

Name	Data Type	Allow Nulls	Default
MaKH	nvarchar(4)	<input type="checkbox"/>	
TenKH	nvarchar(30)	<input checked="" type="checkbox"/>	
DiaChi	nvarchar(30)	<input checked="" type="checkbox"/>	
DienThoai	nvarchar(7)	<input checked="" type="checkbox"/>	
Fax	nvarchar(12)	<input checked="" type="checkbox"/>	
Email	nvarchar(50)	<input checked="" type="checkbox"/>	

The 'Email' column is highlighted with a red rectangle. On the right, the 'Model Browser' pane shows the 'KhachHang' entity. A context menu is open, and the 'Update Model from Database...' option is highlighted with a red rectangle. The menu also includes options like 'Add New', 'Diagram', 'Zoom', 'Grid', 'Scalar Property Format', 'Select All', 'Validate', 'Generate Database from Model...', 'Add Code Generation Item...', 'Mapping Details', 'Model Browser', and 'Properties'.

43

Entity Framework Database First (tt)

- Thay đổi cấu trúc bảng trong CSDL:
 - Cập nhật field lên View
 - Xóa Controller và View để tạo lại
 - Hiệu chỉnh code trong View

44

5.3 Lambda Expressions

- Một biểu thức lambda (Lambda Expressions) là một hàm ẩn danh (anonymous function) được dùng để tạo các kiểu delegates hay cây biểu thức (expression tree)
- Với biểu thức lambda, ta có thể viết các hàm cục bộ có thể được dùng như các tham số hay được trả về như giá trị của hàm gọi.
- Biểu thức lambda rất hữu ích cho việc viết các biểu thức truy vấn LINQ.
- Có từ .NET 4.5
- Trong C#, lambda là một hàm sử dụng cú pháp đơn giản, có các đối số và trả về giá trị.

Lambda Expressions (tt)

- Cú pháp của biểu thức lambda:
(tham số đầu vào) => biểu thức
- **Vế trái:** dấu ngoặc đơn là tùy chọn, nếu tham số đầu vào có nhiều hơn một tham số, thì tất cả phải đặt trong () và các tham số cách nhau bởi dấu phẩy
 – Ví dụ:

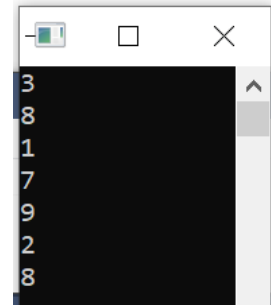
```
y => y * y
(x, y) => x == y
(int x, string s) => s.Length > x
() => SomeMethod()
```

```
static void Main(string[] args)
{
    List<int> elements = new List<int>() { 10, 20, 31, 40 };
    int oddIndex = elements.FindIndex(x => x % 2 != 0);
    Console.WriteLine(oddIndex); //2
}
```

Lambda Expressions (tt)

- **Về phải:** nếu trong biểu thức có nhiều lệnh thì phải đặt trong cặp ngoặc {}, khi đó biểu thức Lambda có dạng:
 (input parameters) => {statement;}

```
//in các số nhỏ <= 3 và >=7
int[] source = new[] { 3, 8, 4, 6, 1, 7, 9, 2, 4, 8 };
foreach (int i in source.Where(x => {
    if (x <= 3) return true;
    if (x >= 7) return true;
    return false;
})))
    Console.WriteLine(i);
```



Lambda Expressions (tt)

- Biểu thức Lambda có nhiều tham số
 – Các tham số đặt trong dấu ngoặc đơn

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

```
delegate bool IsYoungerThan(Student stud, int youngAge);
static void Main(string[] args)
{
    IsYoungerThan isYoungerThan = (s, youngAge) => s.Age < youngAge;
    Student stud = new Student() { Age = 25 };
    Console.WriteLine(isYoungerThan(stud, 26));
    Console.Read();
}
```


Lambda Expressions (tt)

- Quy tắc viết biểu thức lambda rút gọn:

- Có thể bỏ qua kiểu dữ liệu của tham số

`(string name) => {Console.WriteLine("Hello: " + name);}`

→ `(name) => {Console.WriteLine("Hello: " + name);}`

- Nếu chỉ có một tham số, có thể bỏ dấu **()**

`(x) => {Console.WriteLine("Hello " + x);}` → `x => {Console.WriteLine("Hello " + x);}`

- Nếu anonymous function chỉ có 1 câu lệnh, có thể bỏ dấu **{ }**

`x => { Console.WriteLine("Hello " + x); }` → `x => Console.WriteLine("Hello " + x)`

- Nếu chỉ return một giá trị, có thể bỏ lệnh **return**

• Ví dụ 4 lambda expression sau là tương đương nhau

`(x) => { return x > 4; }`

`x => { return x > 4; }`

`x => return x > 4`

`x => x > 4`

Lambda Expressions (tt)

- **delegate**: dùng để biểu diễn một anonymous function

- delegate không có tham số có thể bỏ **()**

```
delegate int PrintNumber();
static void Main(string[] args)
{
    Random rand = new Random();
    PrintNumber number = delegate { return rand.Next() % 100; };
    //hoặc PrintNumber number = () => rand.Next() % 100;
    Console.WriteLine(number());
    Console.Read();
}
```

Lambda Expressions (tt)

• delegate (tt):

– delegate có nhiều tham số: khai báo tham số như một phương thức

```
delegate int Sum (int x, int y);
static void Main(string[] args)
{
    Sum sum = delegate (int x, int y) { return x + y; };
    //hoặc Sum sum = (int x, int y) => x + y;
    Console.WriteLine(sum(10, 5));
    Console.Read();
}
```

Lambda Expressions (tt)

• delegate (tt)

```
class Program
{
    delegate bool IsTeenAger(Student stud);
    public static void Main()
    {
        IsTeenAger isTeenAger = delegate (Student s) {return s.Age > 12 && s.Age < 20; };
        Student stud = new Student() { Age = 15 };
        Console.WriteLine(isTeenAger(stud));
        Console.Read();
    }
}
```

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

Lambda Expressions (tt)

- **delegate:** gán biểu thức lambda đến delegate

– Ví dụ 1:

```
delegate bool IsTeenAger(Student stud);
public static void Main() {
//thay vì IsTeenAger isTeenAger = delegate (Student s) { return s.Age > 12 && s.Age < 20; };
IsTeenAger isTeenAger = s => s.Age > 12 && s.Age < 20;
Student stud = new Student() { Age = 25 };
Console.WriteLine(isTeenAger(stud));
Console.Read();
}
```

Lambda Expressions (tt)

- **delegate:** gán biểu thức lambda đến delegate

• Ví dụ 2:

```
delegate int Multiply(int i);
static void Main(string[] args)
{
//thay vì Multiply multiply = delegate (int y) { return y * y; };
Multiply multiply = y => y * y;
int j = multiply(5);
Console.WriteLine(j); // 25
Console.ReadLine();
}
```

Lambda Expressions (tt)

• Biểu thức lambda có nhiều câu lệnh

- Đặt các câu lệnh trong ngoặc {}:

```
delegate bool IsYoungerThan(Student stud, int youngAge);
static void Main(string[] args)
{
    IsYoungerThan isYoungerThan = (s, youngAge) => {
        Console.WriteLine("Lambda expression with multiple statements");
        return s.Age < youngAge;
    };
    Student stud = new Student() { Age = 25 };
    Console.WriteLine(isYoungerThan(stud, 26));
    Console.Read();
}
```

Lambda Expressions (tt)

• Sử dụng biến cục bộ bên trong biểu thức lambda

```
delegate bool IsAdult(Student stud);
static void Main(string[] args)
{
    IsAdult isAdult = (s) => {
        int adultAge = 18;
        return s.Age >= adultAge;
    };
    Student stud = new Student() { Age = 25 };
    Console.WriteLine(isAdult(stud));
    Console.Read();
}
```

Lambda Expressions (tt)

• delegate Func <...>:

- **Func** cho phép khai báo và tạo ra các dạng delegate với số lượng tham số và kiểu trả về khác nhau, tương tự như tạo ra một phương thức.
- **Func** được dùng để tạo và lưu trữ một anonymous method ngắn gọn bằng lambda expression và được sử dụng như những phương thức thông thường.
- **Func** cung cấp một cách thức để lưu trữ anonymous method cho phép sử dụng nhiều lần mà không cần phải khai báo và định nghĩa phương thức bên ngoài.
- Cú pháp: **Func <T1, T2, T3, TResult>**
 - T1, T2, T3: các kiểu của tham số cần truyền vào
 - TResult là kiểu của giá trị trả về
 - Func bắt buộc phải có kiểu trả về.

Lambda Expressions (tt)

• Delegate Func, ví dụ:

```
static void Main(string[] args)
{
    Func<string, int> stringLength = s => s.Length;
    Console.WriteLine(stringLength("Hello"));

    Func<float, int, float> multiply = (x, y) => x * y;
    Console.WriteLine(multiply(9f, 7));

    Func<int, int, int> max = (x, y) => x > y ? x : y;
    Console.WriteLine(max(9, 7));

    Console.Read();
}
```

Lambda Expressions (tt)

• Delegate Func, ví dụ:

```
Func<int, int> func1 = x => x + 1;
Console.WriteLine("FUNC1: {0}", func1.Invoke(200));

Func<int, int> func2 = x => { return x + 1; };
Console.WriteLine("FUNC2: {0}", func2.Invoke(200));

Func<int, int> func3 = (int x) => x + 1;
Console.WriteLine("FUNC3: {0}", func3.Invoke(200));

Func<int, int> func4 = (int x) => { return x + 1; };
Console.WriteLine("FUNC4: {0}", func4.Invoke(200));
```

Lambda Expressions (tt)

• Action Delegate

- Action delegate chỉ nhận danh sách các tham số
- Được sử dụng khi biểu thức lambda không trả về giá trị

```
Action<Student> PrintStudentDetail =
    s => Console.WriteLine("Name: {0}, Age: {1} ", s.Name, s.Age);
Student std = new Student() { Name = "Bill", Age = 21 };
PrintStudentDetail(std);
```

Lambda Expressions (tt)

• Lambda Expression và LINQ

- Sử dụng lambda expression để truyền một **anonymous function** vào các phương thức mở rộng **Where, First** của lớp **Enumerable**.

```
var students = studentList.Where(|
```

▲ 1 of 2 ▼ (extension) IEnumerable<Student> IEnumerable<Student>.Where(Func<Student,bool> predicate)
 Filters a sequence of values based on a predicate.
predicate: A function to test each element for a condition.

Func delegate

Lambda Expressions (tt)

• Lambda Expression và LINQ, ví dụ

```
IList<Student> studentList = new List<Student>() {
    new Student() { Id = 1, Name = "John", Age = 13 } ,
    new Student() { Id = 2, Name = "Moin", Age = 21 } ,
    new Student() { Id = 3, Name = "Bill", Age = 18 } ,
    new Student() { Id = 4, Name = "Ram", Age = 20 } ,
    new Student() { Id = 5, Name = "Ron", Age = 15 }
};
Func<Student, bool> isStudentTeenAger = s => s.Age > 12 && s.Age < 20;
var teenAgerStudent = studentList.Where(isStudentTeenAger);
Console.WriteLine("Teen age Students:");
foreach (Student std in teenAgerStudent)
    Console.WriteLine(std.Name);
```

Lambda Expressions (tt)

• Lambda Expression và LINQ, ví dụ

```

IList<Student> studentList = new List<Student>() {
    new Student() { Id = 1, Name = "John", Age = 13 } ,
    new Student() { Id = 2, Name = "Moin", Age = 21 } ,
    new Student() { Id = 3, Name = "Bill", Age = 18 } ,
    new Student() { Id = 4, Name = "Ram", Age = 20 } ,
    new Student() { Id = 5, Name = "Ron", Age = 15 }
};

Func<Student, bool> isStudentTeenAger = s => s.Age > 12 && s.Age < 20;
var teenAgerStudents = from s in studentList
                        where isStudentTeenAger(s)
                        select s;
Console.WriteLine("Teen age Students:");
foreach (Student std in teenAgerStudent)
    Console.WriteLine(std.Name);
  
```

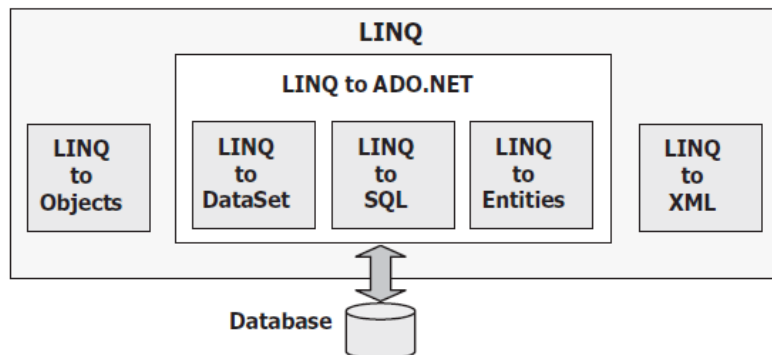
Lập trình web

63

63

5.4 LINQ (Language Integrated Query)

- Ngôn ngữ tích hợp truy vấn dữ liệu có trong thành phần .NET Framework
- Là một mô hình lập trình thống nhất cho bất kỳ loại dữ liệu nào, độc lập với nguồn dữ liệu.
- Cho phép viết các câu truy vấn dữ liệu ngay trong một ngôn ngữ lập trình, như C# hoặc VB.NET.
- Kiến trúc LINQ



Lập trình web

64

64

Linq To Entities Query

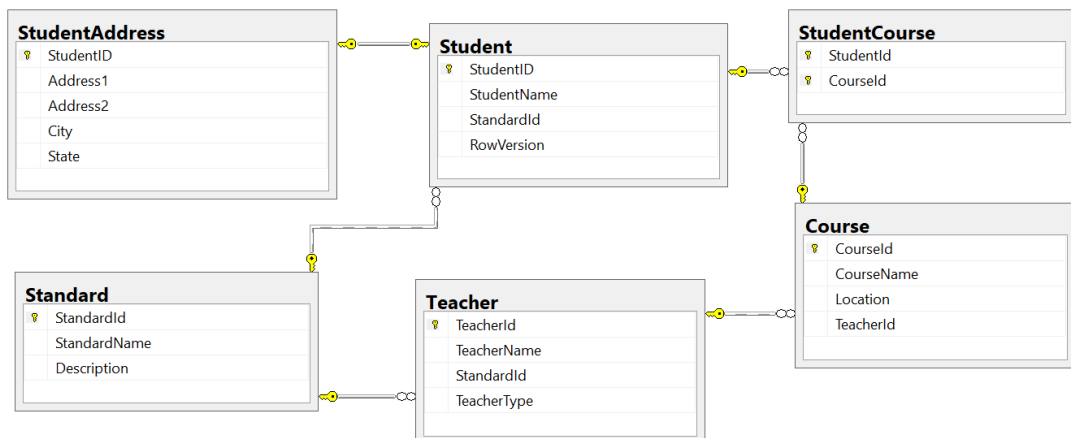
- Một số toán tử truy vấn tiêu chuẩn (hoặc phương thức mở rộng) có thể được sử dụng với các truy vấn LINQ-to-Entities.

- First()
- FirstOrDefault()
- Single()
- SingleOrDefault()
- ToList()
- Count()
- Min()
- Max()
- Last()
- LastOrDefault()
- Average()

```
//Example: Querying with LINQ to Entities
using (var objCtx = new SchoolDBEntities())
{
    var schoolCourse = from cs in objCtx.Courses
                        where cs.CourseName == "Course1" select cs;
    Course mathCourse = schoolCourse.FirstOrDefault<Course>();
    IList<Course> courseList = schoolCourse.ToList<Course>();
    string courseName = mathCourse.CourseName;
}
```

Linq To Entities Query

- Database cho các ví dụ



Linq-to-Entities Query (tt)

- **Find():** tìm kiếm thực thể dựa trên giá trị khóa chính.

```
//Tìm record có StudentId là 1
var ctx = new SchoolDBEntities();
var student = ctx.Students.Find(1);
```

- **First/FirstOrDefault:** trả về record đầu tiên thỏa điều kiện truy vấn

```
using (var ctx = new SchoolDBEntities()) {
    var student = (from s in ctx.Students
        where s.StudentName == "Bill"
        select s).FirstOrDefault<Student>();
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var student = ctx.Students
        .Where(s => s.StudentName == "Bill")
        .FirstOrDefault<Student>(); }
}
```

Linq-to-Entities Query (tt)

- **FirstOrDefault** có tham số

```
using (var ctx = new SchoolDBEntities()) {
    string name = "Bill";
    var student = ctx.Students
        .Where(s => s.StudentName == name)
        .FirstOrDefault<Student>(); }
}
```

- Điểm khác nhau giữa First và FirstOrDefault là First() sẽ trả về một exception nếu truy vấn không có kết quả trả về, còn FirstOrDefault() trả về null

- **ToList():** trả về danh sách các record của truy vấn.

```
using (var ctx = new SchoolDBEntities())
{
    var studentList = ctx.Students.Where(s => s.StudentName == "Bill").ToList();
}
```

- Có thể dùng tương tự với ToArray, ToDictionary, ToLookup

Linq-to-Entities Query (tt)

- **GroupBy**: truy vấn gom nhóm

```
using (var ctx = new SchoolDBEntities())
{
    var students = from s in ctx.Students
                   group s by s.StandardId into studentsByStandard
                   select studentsByStandard;
    foreach (var groupItem in students)
    {
        Console.WriteLine(groupItem.Key);
        foreach (var stud in groupItem)
        {
            Console.WriteLine(stud.StudentId);
        }
    }
}
```

Linq-to-Entities Query (tt)

- **GroupBy (tt)**

```
using (var ctx = new SchoolDBEntities())
{
    var students = ctx.Students.GroupBy(s => s.StandardId);
    foreach (var groupItem in students)
    {
        Console.WriteLine(groupItem.Key);
        foreach (var stud in groupItem)
        {
            Console.WriteLine(stud.StudentId);
        }
    }
}
```

Linq-to-Entities Query (tt)

• **OrderBy:** sắp xếp dữ liệu dùng OrderBy, OrderByDescending

```
using (var ctx = new SchoolDBEntities())
{
    var students = from s in ctx.Students
                   orderby s.StudentName ascending
                   select s;
}
```

```
using (var ctx = new SchoolDBEntities())
{
    var students = ctx.Students.OrderBy(s => s.StudentName).ToList();
    // descending order
    var descStudents = ctx.Students.OrderByDescending(s => s.StudentName).ToList();
}
```

Linq-to-Entities Query (tt)

• **Anonymous Object Result**

- Các truy vấn LINQ-to-Entity không phải luôn trả về các đối tượng thực thể mà có thể chọn để trả về một số thuộc tính của thực thể

```
using (var ctx = new SchoolDBEntities()) {
    var anonymousObjResult = from s in ctx.Students
                             where s.StandardId == 1
                             select new { Id = st.StudentId, Name = st.StudentName };
    foreach (var obj in anonymousObjResult)
        Console.WriteLine(obj.Name);
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var anonymousObjResult = ctx.Students .Where(st => st.StandardId == 1)
                                         .Select(st => new { Id = st.StudentId, Name = st.StudentName });
    foreach (var obj in anonymousObjResult)
        Console.WriteLine(obj.Name);
}
```

Linq-to-Entities Query (tt)

- **Include():** có thể nạp các thực thể liên quan trong cùng một câu truy vấn, sử dụng phương thức **Include()**

```
using (var ctx = new SchoolDBEntities()) {
    var stud1 = (from s in ctx.Students.Include("Standard")
        where s.StudentName == "Bill" select s).FirstOrDefault<Student>();
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var stud1 = ctx.Students .Include("Standard")
        .Where(s => s.StudentName == "Bill").FirstOrDefault<Student>();
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var stud1 = ctx.Students.Include(s => s.Standard)
        .Where(s => s.StudentName == "Bill").FirstOrDefault<Student>();
}
```

73

Linq-to-Entities Query (tt)

- Truy vấn nhiều thực thể có quan hệ phân cấp

```
using (var ctx = new SchoolDBEntities()) {
    var stud1 = ctx.Students.Include("Standard.Teachers")
        .Where(s => s.StudentName == "Bill") .FirstOrDefault<Student>();
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var stud1 = ctx.Students.Include(s => s.Standard.Teachers)
        .Where(s => s.StudentName == "Bill") .FirstOrDefault<Student>();
}
```

74

Linq-to-Entities Query (tt)

- **Raw SQL Queries:** sử dụng câu truy vấn SQL, hỗ trợ từ EF 6.x, sử dụng các phương thức `DbSet.SqlQuery()`, `DbContext.Database.SqlQuery()`, `DbContext.Database.ExecuteSqlCommand()`
 - **DbSet.SqlQuery():** thực thi truy vấn sql trả về các thực thể cho kết quả tương tự như sử dụng Linq

```
using (var ctx = new SchoolDBEntities()) {
    var studentList = ctx.Students.SqlQuery("Select * from Students")
        .ToList<Student>();
}
```

```
using (var ctx = new SchoolDBEntities()) {
    var student = ctx.Students
        .SqlQuery("Select * from Students where StudentId=@id",
            new SqlParameter("@id", 1)) .FirstOrDefault();
}
```

Lập trình web

75

75

Linq-to-Entities Query (tt)

- **Database.SqlQuery():** trả về giá trị của kiểu dữ liệu bất kỳ

```
using (var ctx = new SchoolDBEntities()) {
    string studentName = ctx.Database.SqlQuery<string>("Select studentname
from Student where studentid=1") .FirstOrDefault();
//or
    string studentName = ctx.Database.SqlQuery<string>("Select studentname
from Student where studentid=@id", new SqlParameter("@id", 1))
        .FirstOrDefault();
}
```

Lập trình web

76

76

Linq-to-Entities Query (tt)

– Database.ExecuteSqlCommand(): thực thi câu truy vấn hành động

```
using (var ctx = new SchoolDBEntities()) {
    int noOfRowUpdated = ctx.Database.ExecuteSqlCommand("Update student set
studentname = 'changed student by command' where studentid=1");
    int noOfRowInserted = ctx.Database.ExecuteSqlCommand("insert into
student(studentname) values('New Student')");
    int noOfRowDeleted = ctx.Database.ExecuteSqlCommand("delete from student
where studentid=1");
}
```

Linq-to-Entities Query (tt)

• Insert

```
var newStandard = new Standard();
newStandard.StandardName = "Standard 1";
using (var dbCtx = new SchoolDBEntities())
{
    dbCtx.Standards.Add(newStandard);
    dbCtx.SaveChanges();
}
```

• Update

```
using (SchoolEntities ctx = new SchoolDBEntities())
{
    var stud = (from s in ctx.Students
                where s.StudentName == "Student1" select s).FirstOrDefault();
    stud.StudentName = "Student2";
    int num = ctx.SaveChanges();
}
```

Linq-to-Entities Query (tt)

• Delete

```
using (var dbCtx = new SchoolDBEntities())
{
    //truy vấn đối tượng, sau đó sử dụng các phương thức Set().Remove(entity) để
    xóa đối tượng
    var newtchr = dbCtx.Teachers.Where(t => t.TeacherName ==
    "New teacher4").FirstOrDefault<Teacher>();
    dbCtx.Set(Teacher).Remove(newtchr);
    //có thể thiết lập trạng thái đã xóa cho đối tượng
    //dbCtx.Entry(newtchr).State = System.Data.EntityState.Deleted;
    // hoặc
    //dbCtx.Teachers.Remove(newtchr);
    dbCtx.SaveChanges();
}
```

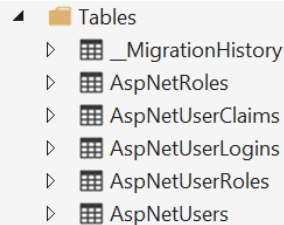
5.5 ASP.NET MVC Membership Provider

- Authentication (xác thực)
- Authorization (ủy quyền)
- Membership providers in ASP.NET MVC.
- Roles based authentication for user in ASP.NET MVC.

Tạo ASP.NET Membership Database

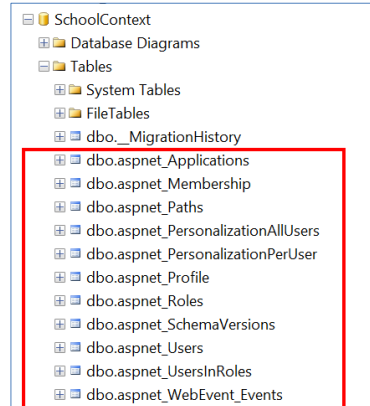
- Lưu ý: database aspnet-Membership nằm trong thư mục App_Data, khi đó chuỗi kết nối có dạng:

```
<connectionStrings>
  <add name="DefaultConnection" connectionString="Data
Source=(LocalDb)\MSSQLLocalDB;AttachDbFilename=|DataDirectory|\aspnet-
Membership-20210901035615.mdf;Initial Catalog=aspnet-Membership-
20210901035615;Integrated Security=True"
  providerName="System.Data.SqlClient" />
</connectionStrings>
```



Tạo ASP.NET Membership Database

- Nếu muốn tích hợp trong database khác, chạy file:
 C:\windows\Microsoft.NET\Framework\v4\aspnet_regsql.exe
 – Trong đó: Framework\v4 tùy theo phiên bản cài đặt



Authorization

- Bắt buộc đăng nhập/giới hạn quyền truy cập đối với mỗi người dùng/nhóm
- Sử dụng thuộc tính [Authorize]: ngăn chặn truy cập vào Controller/Action khi chưa đăng nhập
- Global Authorization Filter: bảo mật toàn bộ ứng dụng

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new AuthorizeAttribute());
        filters.Add(new HandleErrorAttribute());
    }
}
```

- Khi đó, muốn cho phép truy cập vào chức năng nào, bổ sung thuộc tính: **[AllowAnonymous]**

Authorization (tt)

- Cấp quyền truy cập cho user, role:
 - [Authorize(Users ="Tom,Mary")]: chỉ cho phép hai user là "Tom" và "Mary"
 - [Authorize(Roles ="Admin,WebMaster")]: chỉ cho phép các user thuộc Role "Admin" và "WebMaster"
 - [Authorize(Roles="Users", Users="Jon,Phil,Brad,David")]

Authorization (tt)

- Tạo Role và user:

```
ApplicationDbContext context = new ApplicationDbContext();  
var roleManager = new RoleManager<IdentityRole> (new  
    RoleStore<IdentityRole>(context));  
var role = new IdentityRole();  
role.Name = "Admin";  
roleManager.Create(role);  
var UserManager = new UserManager<ApplicationUser>(new  
    UserStore<ApplicationUser>(context));  
var user = new ApplicationUser();  
user.UserName = "admin@gmail.com";  
user.Email = "admin@gmail.com";  
string password = "123456";  
var result = UserManager.Create(user, password);  
if (result.Succeeded)  
    UserManager.AddToRole(user.Id, "Admin");
```

Lập trình web

85