**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**ĐỒ ÁN MÔN**

**CÔNG NGHỆ .NET**

**ĐỀ TÀI:**

**TÌM HIỂU VỀ ENTITY FRAMEWORK**

Giảng viên hướng dẫn: ThS. Phạm Thi Vương

Sinh viên thực hiện: Nguyễn Văn Quang 13520675

Bùi Đình Lộc Thọ 13520844

Đoàn Duy Phương 13520657

Nguyễn Thanh Hải 13520231

Lớp: SE310.H21

***TP. Hồ Chí Minh, ngày 21 tháng 06 năm 2017***

**Lời Cảm Ơn**

Đầu tiên, nhóm chúng em xin gởi lời cảm ơn chân thành đến tập thể quý Thầy Cô Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý Thầy Cô khoa Công Nghệ Phần Mềm đã giúp cho nhóm chúng em có những kiến thức cơ bản làm nền tảng để thực hiện đề tài này.

Đặc biệt, nhóm chúng em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới Thầy Phạm Thi Vương đã giúp nhóm chúng em hoàn thành tốt báo cáo môn học của mình.

Trong thời gian một học kỳ thực hiện đề tài, nhóm chúng em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới. Từ đó, nhóm chúng em vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, trong quá trình thực hiện, nhóm chúng em không tránh khỏi những thiếu sót. Chính vì vậy, nhóm chúng em rất mong nhận được những sự góp ý từ phía các Thầy Cô nhằm hoàn thiện những kiến thức mà nhóm chúng em đã học tập và là hành trang để nhóm chúng em thực hiện tiếp các đề tài khác trong tương lai.

Nhóm em xin chân thành cảm ơn quý Thầy Cô!

NHẬN XÉT CỦA GIẢNG VIÊN

………………………………………………………………………………………………....

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**MỤC LỤC**

[CHƯƠNG 1. NỀN TẢNG .NET 5](#_Toc485736808)

[1.1 Tổng quan .Net Framework 6](#_Toc485736809)

[1.2 Lịch sử phát triển 7](#_Toc485736810)

[1.2.1 .NET Framework Ver 1.0 8](#_Toc485736811)

[1.2.2 .NET Framework Ver 1.1 8](#_Toc485736812)

[1.2.3 .NET Framework Ver 2.0 9](#_Toc485736813)

[1.2.4 .NET Framework Ver 3.0 9](#_Toc485736814)

[1.2.5 .NET Framework Ver 4.0 10](#_Toc485736815)

[CHƯƠNG 2. ADO.NET 12](#_Toc485736816)

[2.1 Tổng quan về ADO.NET 12](#_Toc485736817)

[2.1.1 Giới thiệu 12](#_Toc485736818)

[2.1.2 Data Provider 12](#_Toc485736819)

[2.1.3 Các đối tượng của ADO.NET 13](#_Toc485736820)

[2.2 Entity Relationship Model 27](#_Toc485736821)

[2.2.1 Giới thiệu về Entity Relationship Model (mô hình thực thể mối kế hợp) 27](#_Toc485736822)

[2.2.2 Loại thực thể trong mô hình ER 27](#_Toc485736823)

[2.2.3 Thực thể (Entity) trong mô hình ER 28](#_Toc485736824)

[2.2.4 Thuộc tính của loại thực thể 28](#_Toc485736825)

[2.2.5 Các loại thuộc tính của loại thực thể 29](#_Toc485736826)

[2.2.6 Khóa của loại thực thể trong mô hình ER 30](#_Toc485736827)

[2.2.7 Các loại mối kết hợp trong mô hình ER 30](#_Toc485736828)

[2.2.8 Số ngôi của loại mối kết hợp 31](#_Toc485736829)

[2.2.9 Bản số 32](#_Toc485736830)

[2.3 Entity Data Model 33](#_Toc485736831)

[2.4 LINQ to SQL 35](#_Toc485736832)

[2.4.1 Tổng quan LINQ to SQL 35](#_Toc485736833)

[2.4.2 Sự ra đời của LINQ to SQL 36](#_Toc485736834)

[2.4.3 Công dụng của LINQ to SQL 37](#_Toc485736835)

[2.4 Hạn chế của LINQ to SQL 38](#_Toc485736836)

[CHƯƠNG 3. ENTITY FRAMEWORK 40](#_Toc485736837)

[3.1 Tổng quan về Entity Framework 40](#_Toc485736838)

[3.1.1 Giới thiệu 40](#_Toc485736839)

[3.1.2 Kiến trúc của Entity Framework 42](#_Toc485736840)

[3.2 Mô hình MVC 44](#_Toc485736841)

[3.2.1 Giới thiệu về mô hình MVC 44](#_Toc485736842)

[3.2.2 Các thành phần trong MVC 45](#_Toc485736843)

[3.2.3 Sự tương tác giữa các lớp trong mô hình MVC 45](#_Toc485736844)

[3.2.4 Ưu điểm 46](#_Toc485736845)

[3.2.5 Nhược điểm 47](#_Toc485736846)

[3.2.6 Liên hệ MVC và Entity Framework 48](#_Toc485736847)

[CHƯƠNG 4. PHÁT TRIỂN ỨNG DỤNG VỚI ENTITY FRAMEWORK 50](#_Toc485736848)

[4.1. Code first 50](#_Toc485736849)

[4.2. Model first 61](#_Toc485736850)

[4.3. Database first 67](#_Toc485736851)

[CHƯƠNG 5. TỔNG KẾT 73](#_Toc485736852)

[5.1. Môi trường phát triển và môi trường triển khai 73](#_Toc485736853)

[5.2. Kết quả đạt được 73](#_Toc485736854)

[5.3. Hạn chế 73](#_Toc485736855)

# 

# CHƯƠNG 1. NỀN TẢNG .NET

## 1.1 Tổng quan .Net Framework

**.NET Framework** là một nền tảng lập trình và cũng là một nền tảng thực thi ứng dụng chủ yếu trên hệ điều hành Microsoft Windows được phát triển bởi Microsoft. Các chương trình được viết trên nền .NET Framework sẽ được triển khai trong môi trường phần mềm (ngược lại với môi trường phần cứng) được biết đến với tên Common Language Runtime (CLR). Môi trường phần mềm này là một máy ảo trong đó cung cấp các dịch vụ như an ninh phần mềm (Security), quản lý bộ nhớ (Memory Management), và các xử lý lỗi ngoại lệ (Exception Handling).

.NET framework bao gồm tập các thư viện lập trình lớn, và những thư viện này hỗ trợ việc xây dựng các chương trình phần mềm như lập trình giao diện; truy cập, kết nối cơ sở dữ liệu; ứng dụng web; các giải thuật, cấu trúc dữ liệu; giao tiếp mạng... CLR cùng với bộ thư viện này là 2 thành phần chính của.NET framework.

.NET framework đơn giản hóa việc viết ứng dụng bằng cách cung cấp nhiều thành phần được thiết kế sẵn, người lập trình chỉ cần học cách sử dụng và tùy theo sự sáng tạo mà gắn kết các thành phần đó lại với nhau. Nhiều công cụ được tạo ra để hỗ trợ xây dựng ứng dụng.NET, và IDE (*Integrated Developement Environment*) được phát triển và hỗ trợ bởi chính Microsoft là Visual Studio.

## 1.2 Lịch sử phát triển

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phiên bản** | **Số hiệu phiên bản** | **Ngày phát hành** | **Visual Studio** | **Được phát hành kèm theo** |
| 1.0 | 1.0.3705.0 | 13 tháng 2 năm 2002 | Visual Studio.NET | Windows XP Tablet and Media Center Editions |
| 1.1 | 1.1.4322.573 | 24 tháng 4 năm 2003 | Visual Studio.NET 2003 | Windows Server 2003 |
| 2.0 | 2.0.50727.42 | 7 tháng 11 năm 2005 | Visual Studio 2005 | Windows Server 2003 R2 |
| 3.0 | 3.0.4506.30 | 6 tháng 11 năm 2006 |  | Windows Vista, Windows Server 2008 |
| 3.5 | 3.5.21022.8 | 19 tháng 11 năm 2007 | Visual Studio 2008 | Windows 7, Windows Server 2008 R2 |
| 4.0 | 4.0.30319.1 | 12 tháng 4 năm 2010 | Visual Studio 2010 |  |
| 4.5 | 4.5.50709 | 15 tháng 8 năm 2012 | Visual Studio 2012 | Windows 8, Windows Server 2012 |
| 4.6 |  | 20 tháng 7 năm 2015 | Visual Studio 2015 | Windows 10 |

### 1.2.1 .NET Framework Ver 1.0

Ngày 12/2/2002 đánh dấu bước quan trọng đầu tiên trong “cuộc đời” của .NET Framework, khi phiên bản 1.0 cùng với Visual Studio.NET 2002 được chính thức ra mắt. Chính .NET Framework 1.0 là điểm nhấn đáng chú ý nhất và làm cho Visual Studio. NET 2002 khác biệt hẳn với Visual Studio 6.0 đã phát hành năm 1998. Lần đầu tiên, Microsoft giới thiệu về “lập trình hợp nhất”, với việc lấy .NET Framework làm nền tảng.

### 1.2.2 .NET Framework Ver 1.1

Một năm sau ngày .NET Framework 1.0 ra đời, ngày 24/4/2003, Microsoft đã có ngay bản cập nhật 1.1 ra mắt cùng với Visual Studio.NET 2003. Không có nhiều nâng cấp đáng chú ý trong lần ra mắt này, đáng kể nhất là sự ra đời của .NET Compact Framework, phiên bản thu gọn của .NET Framework cho các thiết bị di động. Điều đáng tiếc là mặc dù có nền tảng rất tốt, cùng với sự hỗ trợ mạnh mẽ từ Microsoft, cho đến nay, .NET Compact Framework vẫn chưa phát triển như “lẽ ra nó phải thế”. Hiện nay số thiết bị di động chạy Windows Mobile/Windows Phone khá khiêm tốn so với các hệ điều hành (HĐH) còn lại.

.NET Framework 1.1 cũng mở ra một “truyền thống” là kể từ đây, các HĐH Windows đều được cài đặt sẵn phiên bản .NET Framework mới nhất. Windows Server 2003 tiên phong với phiên bản 1.1, sau đó là Windows Vista với .NET 3.0, và gần đây nhất là Windows 7/Server 2008 với .NET 3.5 SP1.

### 1.2.3 .NET Framework Ver 2.0

Microsoft mất đến hơn 2 năm để phát triển .NET Framework 2.0 và Visual Studio 2005, và thời gian bỏ ra là thật sự đáng giá. Tháng 11/2005, hai sản phẩm này ra mắt với hàng loạt tính năng mới, trong đó đáng kể nhất là việc hỗ trợ hoàn toàn cho tính toán 64-bit, .NET Micro Framework, bổ sung và nâng cấp nhiều control của ASP.NET và đặc biệt là hỗ trợ Generics. .NET 2.0 hoàn toàn khác biệt so với các phiên bản trước.

Generic cho phép chúng ta định kiểu an toàn (Type Safety). Chúng cho phép ta tạo ra một cấu trúc dữ liệu mà không cần phải xác định đó là kiểu dữ liệu gì. Tuy nhiên khi cấu trúc dữ liệu này được sử dụng, trình biên dịch phải đảm bảo rằng kiểu dữ liệu được sử dụng với nó là kiểu an toàn. Generic cũng tương đương vơi Template trong C++ tuy nhiên việc sử dụng Generic trong .NET dễ dàng hơn nhiều so với Template.

Phiên bản 1.0 và 1.1 của .NET Framework không hỗ trợ generics. Thay vào đó, lập trình viên sử dụng lớp Object với các tham số và thành viên sẽ phải chuyển đổi tới các lớp khác dựa trên lớp Object. Generics mang đến hai tính năng cải tiến đáng kể đối với việc sử dụng lớp Object: Giảm bớt lỗi vận hành (Reduced run-time errors), Hiệu suất được cải thiện (Improved performance).

### 1.2.4 .NET Framework Ver 3.0

Nếu như 3 phiên bản trước đó, .NET Framwork đều gắn liền với một phiên bản Visual Studio nào đó, thì.NET Framework 3.0 đã “phá” truyền thống này khi ra mắt cùng với hệ điều hành Windows Vista vào cuối năm 2006. Ba “điểm nhấn” trong lần nâng cấp này là thành phần được kỳ vọng thay thế Winform – Windows Presentation Foundation – WPF, Windows Communitcation Foundation – WCF, Windows Workflow Foundation – WF, và Windows Card Space.

.NET Framework 3.0 không phải là một phiên bản mới hoàn toàn, thực tế là một bản nâng cấp của .NET 2.0, hay đúng hơn là một bản nâng cấp cho thư viện của .NET 2.0. Chính vì không có Visual Studio “đi kèm”, mà .NET 3.0 đành phải “ký gửi” vào Visual Studio 2005 với một bộ công cụ mở rộng. Người dùng phải đợi đến tháng 11 năm 2007 mới được sử dụng một phiên bản Visual Studio hỗ trợ đầy đủ và toàn diện cho .NET 3.0, và hơn thế nữa. Vâng, chúng ta đang nói đến VS 2008 và .NET Frame work 3.5. Cũng như phiên bản 3.0, .NET 3.5, là một mở rộng trên nền .NET 2.0.

LINQ [LINQ: Language Integrated Query – đây là thư viện mở rộng cho các ngôn ngữ lập trình C# và Visual Basic.NET (có thể mở rộng cho các ngôn ngữ khác) cung cấp khả năng truy vấn trực tiếp dữ liệu đối tượng, CSDL và XML là phần nổi bật và đáng chú ý nhất trong .NET 3.5.

### 1.2.5 .NET Framework Ver 4.0

Đây là phiên bản đầu tiên sau .NET 2.0 kể từ 2005, có một CLR hoàn toàn mới: CLR 4.0. Cũng cần nhắc lại là cả .NET 3.0 và 3.5 đều sử dụng CLR 2.0, và không có CLR 3.0. Việc Microsoft chuyển thẳng lên 4.0 không chỉ để “đồng bộ” phiên bản, mà còn nhằm khẳng định đây là một bước tiến lớn.

.NET Framework 4 giới thiệu một model an ninh được cải thiện. Các tính năng mới và cải tiến trong .NET Framework 4 là:

* Application Compatibility and Deployment (Khả năng tương thích ứng dụng và triển khai).
* Core New Features and Improvements (Các tính năng mới và cải tiến của phần nhân).
* Managed Extensibility Framework (Quản lý mở rộng Framework).
* Parallel Computing (Điện toán song song).
* Networking.
* Web.
* Client.
* Data.
* Windows Communication Foundation (WCF).
* Windows Workflow Foundation (WF).

# CHƯƠNG 2. ADO.NET

## 2.1 Tổng quan về ADO.NET

### 2.1.1 Giới thiệu

**ADO.NET** tương tự với **ADO**, điểm khác biệt chính ở chỗ **ADO.NET** là một kiến trúc dữ liệu rời rạc, không kết nối (**Disconnected Data Architecture**). Với kiến trúc này, dữ liệu được nhận về từ cơ sở dữ liệu và được lưu trên vùng nhớ cache của máy người dùng. Người dùng có thể thao tác trên dữ liệu họ nhận về và chỉ kết nối đến cơ sở dữ liệu khi họ cần thay đổi các dòng dữ liệu hay yêu cầu dữ liệu mới.

Việc kết nối không liên tục đến cơ sở dữ liệu đã đem lại nhiều thuận lợi, trong đó điểm lợi nhất là việc giảm đi một lưu lượng lớn truy cập vào cơ sở dữ liệu cùng một lúc, tiết kiệm đáng kể tài nguyên bộ nhớ. Giảm thiểu đáng kể vấn đề hàng trăm ngàn kết nối cùng truy cập vào cơ sở dữ liệu cùng một lúc.

**ADO.NET** kết nối vào cơ sở dữ liệu để lấy dữ liệu và kết nối trở lại để cập nhật dữ liệu khi người dùng thay đổi chúng. Hầu hết mọi ứng dụng đều sử dụng nhiều thời gian cho việc đọc và hiển thị dữ liệu, vì thế **ADO.NET** đã cung cấp một tập hợp con các đối tượng dữ liệu không kết nối cho các ứng dụng để người dùng có thể đọc và hiển thị chúng mà không cần kết nối vào cơ sở dữ liệu.

### 2.1.2 Data Provider

ADO.NET cho phép tương tác với các loại dữ liệu và kiểu database. Mỗi loại dữ liệu cần một cách thức khác nhau để có thể truy xuất. Các loại dữ liệu cũ sử dụng giao thức ODBC, các loại dữ liệu mới hơn sử dụng giao thức OleDb. Vì vậy cần có một thư viện thống nhất để làm việc với chúng, đây chính là lý do mà ADO.NET được tạo ra.

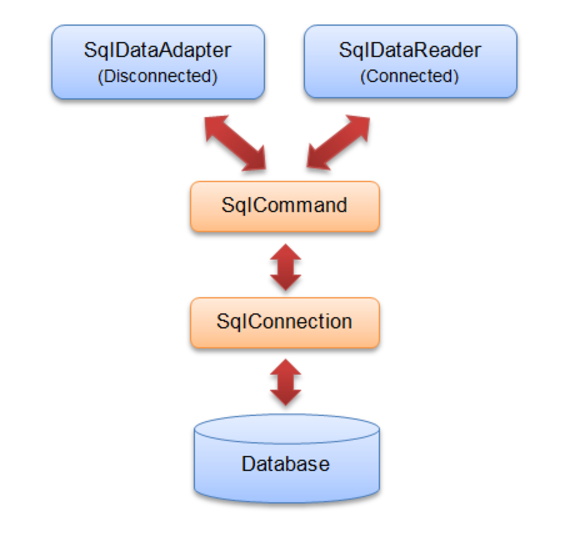
ADO.NET cung cấp một cách thức chung để tương tác với nguồn dữ liệu, nhưng với mỗi loại dữ liệu bạn phải sử dụng một thư viện khác nhau. Các thư viện này được gọi là Data Provider và thường được đặt tên theo giao thức hoặc loại dữ liệu mà chúng cho phép bạn truy xuất. Table 1 liệt kê các data provider phổ biến, tiền tố (prefix) API mà chúng sử dụng và kiểu dữ liệu mà bạn có thể làm việc.

### 2.1.3 Các đối tượng của ADO.NET

#### 2.1.3.1 SqlConnection

Để tương tác với database, bạn phải có một kết nối tới nó. Kết nối giúp xác định database server, database name, user name, password, và các tham số cần thiết để kết nối tới database. Một đối tượng connection được dùng bởi đối tượng command vì thế chúng sẽ biết database nào để thực thi lệnh.

Mô hình sau cho ta thấy cách mà SqlConnection được sử dụng bởi các đối tượng ADO.NET khác



**Tạo một đối tượng SqlConnection:**

Một đối tượng SqlConnection giống như các đối tượng khác trong C#. Chỉ cần khai báo một thể hiện của SqlConnection, như dưới đây:

“SqlConnection conn = new SqlConnection(

“Data Source = DatabaseServer;Initial Catalog=DatabaseName; User ID=YourUserID; Password=yourPassword”);”

ConnectionString chứa các cặp key/value để xác định cách tạo một kết nối đến database bao gồm:

|  |  |
| --- | --- |
| **Tên tham số** | **Mô tả** |
| Data Source | Xác định máy chủ. Có thể là máy địa phương, tên miền máy hoặc địa chỉ IP |
| Initial Catalog | Tên cơ sở dữ liệu |
| Integrated Security | Đặt SSPI để kết nối với đăng nhập Windows của người dùng |
| User ID | Tên người dùng được cấu hình trong SQL Server |
| Password | Mật khẩu khớp với ID người dùng SQL Server |

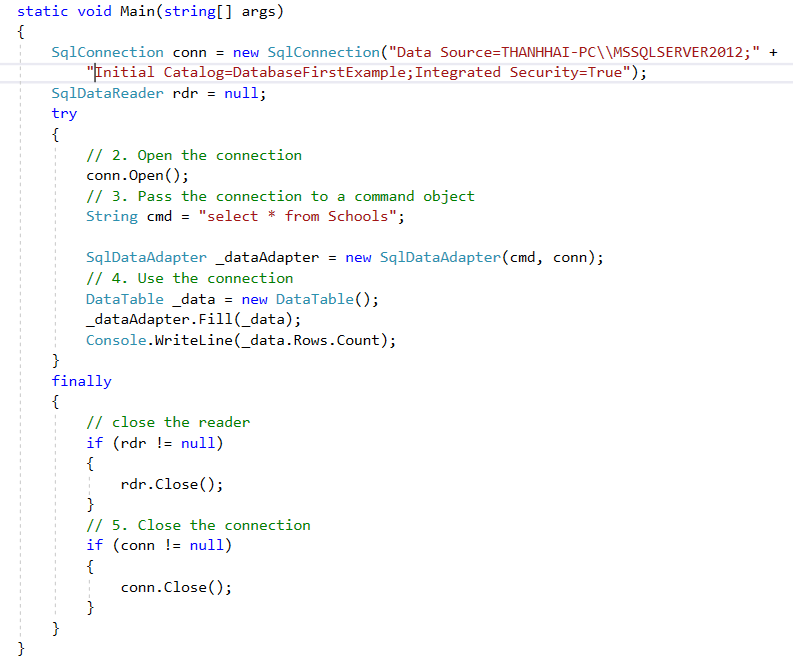
Integrated Security sẽ bảo mật khi bạn làm việc trên một máy tính đơn. Tuy nhiên, bạn sẽ thường xuyên cần phải xác định rõ mức bảo mật dựa trên SQL Server User ID với quyền hạn dược xác định cho ứng dụng của bạn.

**Ghi chú**: Data Source dược gán cho DatabaseServer để chỉ ra rằng bạn có thể định danh một database trên một máy khác, thông qua mạng LAN hoặc internet. Ngoài ra User ID và Password được thay thế cho tham số Integrated Security.

**Sử dụng SqlConnection:**

Mục đích của việc tạo một đối tượng SqlConnection là để các mã lệnh ADO.NET khác có thể làm việc được với database. Các đối tượng ADO.NET khác, như SqlCommand và SqlDataAdapter dùng một connection như một tham số. Quá trình sử dụng SqlConnection gồm các bước sau:

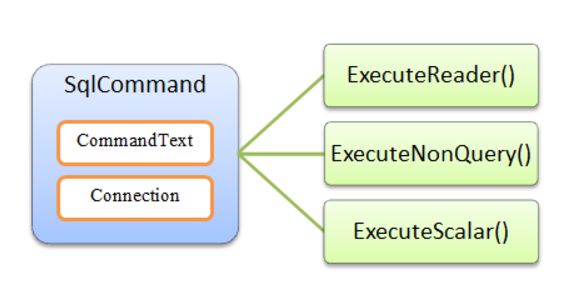
* Tạo một SqlConnection
* Mở connection
* Truyền connection cho các đối tượng ADO.Net khác
* Thực hiện các thao thác database với các đối tượng ADO.net này
* Đóng connection



**Lưu ý:** chúng ta bao đoạn mã ADO.Net trong một khối try/finally giúp đảm bảo rằng một phần mã lệnh sẽ được thực thi, bất kể một ngoại lệ có xảy ra hay không. Bới các connnection là một tài nguyên hệ thống ít ỏi, ta sẽ cần phải chắc chắc rằng chúng được đóng trong khối finally.

#### 2.1.3.2 SqlCommand

Quá trình tương tác với database cần phải biết hành động nào bạn muốn xảy ra. Điều này được thực hiện bởi đối tượng command. Bạn dùng đối tượng command để gửi một câu lệnh SQL tới database. Một đối tượng command dùng một đối tượng connection để xác định database nào sẽ được truy xuất. Bạn có thể dùng một đối tượng command riêng lẻ để thực thi lệnh trực tiếp, hoặc để gắn một tham chiếu của đối tượng command cho một SqlDataAdapter – đối tượng giữ các command sẽ làm việc trên một nhóm dữ liệu như sẽ đề cập tới trong phần dưới.



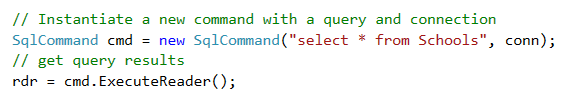
**Tạo một đối tượng SqlCommand:**

Tương tự như các đối tượng C# khác, tạo đối tượng SqlCommand bằng cách khai báo một thể hiện của nó:

*SqlCommand cmd= new SqlCommand(“select \* From Schools”,conn);*

**Querying Data – Truy vấn dữ liệu:**

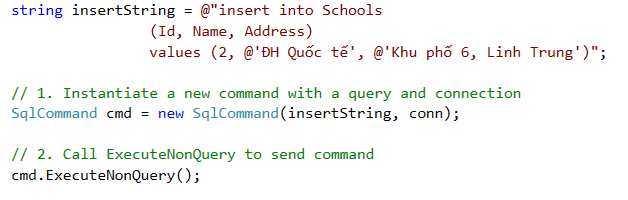
Khi dùng một lệnh SQL select, bạn lấy được một dữ liệu từ database để hiển thị. Để làm được điều này với SqlCommand, bạn cần dùng phương thức ExecuteReader() để trả về một đối tượng SqlDataReader. Chúng ta sẽ thảo luận về SqlDataReader trong phần tiếp theo. Ví dụ sau cho thấy cách dùng SqlCommand để lấy được một đối tượng SqlDataReader:



Trong ví dụ trên, chúng ta tạo một đối tượng SqlCommand, truyền chuỗi lệnh và đối tượng connection vào constructor. Sau đó chúng ta lấy về đối tượng SqlDataReader bằng cách gọi phương thức ExecuteReader() của đối tượng SqlCommand, cmd.

**Inserting Data – Chèn dữ liệu:**

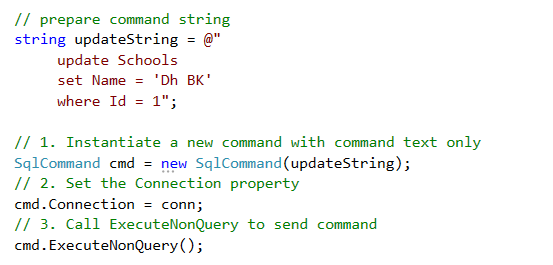
Để chèn dữ liệu vào database, dùng phương thức ExecuteNonQuery() của đối tượng SqlCommand. Đoạn code sau cho thấy cách chèn dữ liệu vào một bảng trong database:



Để thực thi lệnh này, chỉ cần đơn giản gọi phương thức ExecuteNonQuery() của đối tượng SqlCommand, cmd.

**Updating Data – Cập nhật dữ liệu**

Phương thức ExecuteNonQuery cũng được dùng để cập nhật dữ liệu, như đoạn mã sau:



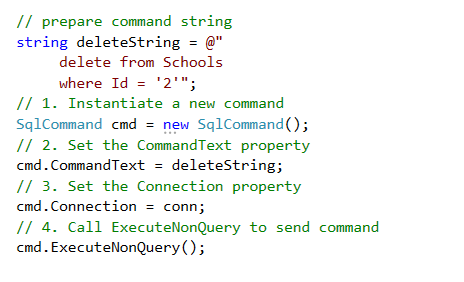
Chúng ta đặt câu SQL trong một biến string, nhưng lần này chúng ta dùng một constructor khác của SqlCommand, chỉ có một tham số là câu lệnh SQL. Trong bước 2, chúng ta gán đối tượng SqlConnection, conn, cho property Connection của đối tượng SqlCommand, cmd.

Thay vì cách này, bạn có thể sử dụng constructor với hai tham số tương tự như cho đoạn mã insert phần trên. Nó cho thấy rằng bạn có thể thay đổi đối tượng connection gán cho một command bất kì lúc nào.

Bước cuối cùng, phương thức ExecuteNonQuery() thực thi lệnh update

**Deleting Data – Xóa dữ liệu**

Bạn cũng có thể xóa dữ liệu bằng phương thức ExecuteNonQuery(). Ví dụ sau cho thấy cách xóa một dòng từ database với phương thức ExecuteNonQuery():

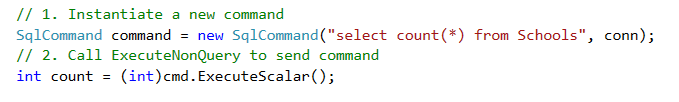


Ví dụ này sử dụng constructor không có tham số của SqlCommand. Thay vào đó, nó gán cho hai property CommandText và Connection của đối tượng SqlCommand, cmd.

Chúng ta cũng có thể dùng hai overload constructor trước đó của SqlCommand, dùng để insert và update, với kết quả tương tự. Điều này cho thấy rằng bạn có thể thay đôi cả chuỗi lệnh và đối tượng connection bất kì lúc nào. Lời gọi phương thức ExecuteNonQuery()  gửi lệnh đến database.

**Lấy một giá trị đơn**

Đôi lúc tất cả những gì bạn cần từ database chỉ là một giá trị đơn, đó có thể là một giá trị đếm, tổng, trung bình, hoặc các giá trị kết hợp khác từ dữ liệu. Thực thi phương thức ExecuteReader() để lấy về một đối tượng SqlDataReader và tính toán kết quả trong mã lệnh của bạn không phải cách tốt để làm điều này. Cách tốt nhất là để database làm công việc này và trả về giá trị bạn cần. Ví dụ sau cho thấy cách làm điều này với phương thức ExecuteScalar():



Câu truy vấn trong constructor của  SqlCommand sẽ lấy về giá trị đếm của tất cả dòng trong bảng Categories. Câu truy vấn này chỉ trả về một giá trị đơn. Phương thức ExecuteScalar() trong bước 2 trả về giá trị này. Bởi vì kiểu trả về của ExecuteScalar() là object, chúng ta cần ép kiểu để chuyển giá trị sang int.

#### 2.1.3.3 SqlDataReader

Một SqlDataReader là đối tượng phù hợp để đọc dữ liệu một cách hiệu quả nhất. Như tên gọi, bạn không thể dùng nó để ghi dữ liệu. SqlDataReader thường được mô tả là luồng dữ liệu fast-forward firehose-like (fire-hose: ống vòi rồng).

Bạn có thể đọc dữ liệu từ đối tượng SqlDataReader theo hướng forward-only trong một thứ tự nhất định. Mỗi lần đọc một vài dữ liệu, bạn phải lưu nó nếu cần thiết bởi vì bạn không thể quay trở lại và đọc nó một lần nữa.

Kiểu thiết kế forward-only của SqlDataReader để giúp nó hoạt động nhanh. Nó không thể di chuyển trực tiếp đến các dòng dữ liệu ở vị trí bất kì và không thể ghi vào dữ liệu nguồn. Do đó, nếu bạn chỉ yêu cầu đọc một nhóm dữ liệu một lần và cần phương pháp nhanh nhất, SqlDataReader là lựa chọn tốt nhất.

**Tạo một đối tượng SqlDataReader**

Có một chút khác biệt để lấy được một thể hiện của SqlDataReader so với các đối tượng ADO.NET khác. Bạn phải gọi phương thức ExecuteReader() của một đối tượng SqlCommand, như:

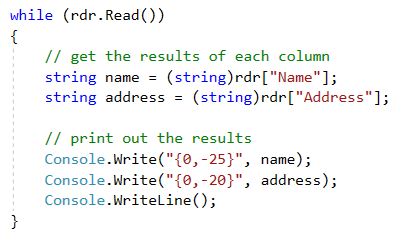
*“SqlDataReader rdr = cm.ExecuteReader()”*

Phương thức ExecuteReader() của đối tượng SqlCommand, cmd, trả về một thể hiện của SqlDataReader. Tạo một SqlDataReader với toán tử **new**không có tác dụng gì cho bạn cả. Như bạn đã học trong bài trước, đối tượng SqlCommand chứa tham chiếu đến connection và câu lệnh SQL cần thiết để SqlDataReader lấy được dữ liệu.

**Đọc dữ liệu**

SqlDataReader trả về dữ liệu qua một luồng liên tục. Để đọc dữ  liệu, bạn phải lấy dữ liệu từ một bảng từng dòng một (row-by-row). Mỗi lần một dòng được đọc, dòng trước đó sẽ không còn hiệu lực. Để đọc lại dòng đó, bạn cần phải tạo một thể hiện mới của SqlDataReader và đọc xuyên qua luồng dữ liệu một lần nữa.

Phương pháp điển hình để đọc từ luồng dữ liệu trả về bởi SqlDataReader là lặp qua mỗi dòng với một vòng lặp while. Đoạn code sau cho thấy cách làm điều này:



Phương thức Read() của SqlDataReader, rdr, trong điều kiện của vòng lặp while trong đoạn code trên. Giá trị trả về của Read() là kiểu bool và trả về **true** đến khi nào vẫn còn dòng để đọc. Sau khi dòng cuối được đọc trong luồng dữ liệu, Read() trả về **false.**

Luôn nhớ việc đóng SqlDataReader, giống như bạn cần phải đóng SqlConnection. Cho đoạn mã truy xuất dữ liệu trong một khối try và đặt lệnh đóng trong khối finally.

#### 2.1.3.4 DataSet

Đối tượng DataSet là một thể hiện của dữ liệu trong bộ nhớ. Chúng chứa nhiều đối tượng DataTable, bên trong DataTable lại có nhiều column và row, giống như các database table thông thường. Bạn thậm chí có thể định nghĩa dữ liệu giữa các table để tạo các quan hệ parent-child. DataSet được thiết kế đặc biệt để giúp quản lý dữ liệu trong bộ nhớ và để hỗ trợ các thao tác không cần kết nối (disconnected) trên dữ liệu. DataSet là một đối tượng được dùng bởi tất cả Data Provider, đó là lý do tại sao nó không có một Data Provider prefix trong tên gọi.

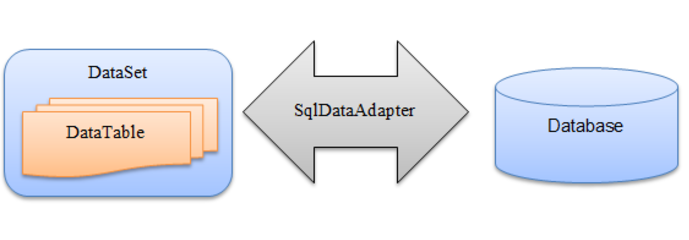
SqlDataAdapter thực hiện các tác vụ sau, khi đổ dữ liệu vào DataSet:

* Mở kết nối
* Đổ dữ liệu vào DataSet (Fill)
* Đóng kết nối

Và thực hiện các công việc sau, khi cập nhật dữ liệu nguồn với thay đổi của DataSet:

* Mở kết nối
* Ghi thay đổi từ DataSet vào dữ liệu nguồn (Update)
* Đóng kết nối

Giữa hai thao tác Fill và Update, các kết nối với nguồn dữ liệu được đóng lại và bạn có thể tự do ghi, đọc dữ liệu với DataSet. Đây chính là cơ chế của mô hình làm việc với disconnected data. Bởi vì ứng dụng sẵn sàng kết nối khi cần thiết, ứng dụng trở nên dễ phát triển hơn.



Dẫn chứng lý do tại sao bạn cần làm việc với disconnected data: giúp Web site dễ phát triển hơn.Với một SqlDataReader, bạn phải trở lại database lấy dữ liệu mỗi khi xem một trang. Điều này yêu cầu một kết nối mới cho mỗi lần tải trang, nó sẽ ảnh hưởng lớn đến hiệu suất khi số lượng người dùng tăng lên. Một cách để khắc phục điều này là dùng DataSet, chỉ cần cập nhật một lần và lưu trong bộ nhớ tạm (cache). Mỗi yêu cầu tải trang sẽ kiểm tra cache và nạp dữ liệu (từ database) nếu nó không tồn tại hoặc lấy dữ liệu ra khỏi cache và hiển thị nó. Điều này giúp hạn chế truy xuất database và tăng hiệu suất cho ứng dụng của bạn.

Các trường hợp ngoại lệ cho kịch bản trên bao gồm các trường hợp bạn cần cập nhật dữ liệu. Bạn phải quyết định, dựa trên cách dữ liệu sẽ được dùng và chiến lược của bạn. Dùng disconnected data khi thông tin của bạn thường là read-only, nhưng hãy xem xét các giải pháp thay thế (như dùng đối tượng SqlCommand để cập nhật tức thời) khi bạn yêu cầu một vài thứ linh hoạt hơn. Cũng vậy, nếu số lượng dữ liệu quá lớn để lưu trong bộ nhớ, bạn sẽ cần dùng SqlDataReader để đọc dữ liệu.

**Tạo đối tượng DataSet**

Không có gì đặc biệt khi tạo một DataSet. Bạn chỉ cần tạo một thể hiện mới, giống bất kì đối tượng nào:

*“DataSet dsSchools = new DataSet()”*

Constructor của DataSet không yêu cầu tham số. Tuy nhiên có một overload chấp nhận một chuỗi đại diện cho tên của DataSet, được dùng nếu bạn cần serialize dữ liệu thành XML. Bây giờ, ta có một DataSet rỗng và cần một SqlDataAdapter để nạp dữ liệu cho nó.

**Tạo một SqlDataAdapter**

SqlDataAdapter chứa các lệnh SQL và đối tượng connection để đọc và ghi dữ liệu. Bạn khởi tạo nó với câu SQL select và đối tượng connection:

*“SqlDataAdapter daSchools = new SqlDataAdapter (“select \* from Schools”, conn)”;*

Công việc của SqlDataAdapter để mở và đóng connection khi phương thức Fill() và Update() được gọi.

SqlDataAdapter tất cả lệnh cần thiết để tương tác với dữ liệu nguồn. Dòng mã trên xác định câu lệnh select, nhưng không cho thấy câu lệnh insert, update và delete. Chúng sẽ được thêm vào SqlDataAdapter sau khi nó được khởi tạo.

**Đổ dữ liệu vào DataSet**

Để đổ dữ liệu vào DataSet bạn cần dùng phương thức Fill() của SqlDataAdapter, như sau:

“*dsSchools.Fill(dsSchools, “Schools”);*”

Phương thức Fill(), trong dòng trên lấy hai tham số: một DataSet và một tên bảng.  DataSet phải được tạo trước khi bạn đổ dữ liệu vào nó. Tham số thứ hai là tên của bảng sẽ được tạo trong DataSet. Bạn có thể đặt bất kì tên gì cho bảng. Thông thường, tôi sẽ để tên bảng trùng với tên gốc của nó trong database. Tuy nhiên, nếu câu select của SqlDataAdapter chứa một lệnh join, bạn sẽ cần phải đặt một tên rõ ràng khác cho bảng.

Phương thức Fill() có một overload chấp nhận một tham số là DataSet. Trong trường hợp này, bảng được tạo sẽ có tên mặc định là “table1” cho bảng đầu tiên. Số này sẽ tăng dần (table2, tabl3,…,tableN) cho mỗi bảng thêm vào DataSet nếu như tên bảng không được chỉ ra trong phương thức Fill().

**Sử dụng DataSet**

Một DataSet sẽ gắn dữ liệu vào DataGrid của ASP.NET và Windows form. Đây là một ví dụ sẽ gán DataSet cho một Windows forms DataGrid

*“dgvSchools.DataSource = dsSchools;*

*dgvSchools.DataMember = “Schools”;*

**Cập nhật thay đổi**

Sau khi thay đổi được thực hiện trên dữ liệu, bạn sẽ cần ghi lại vào database. Dòng mã sau cho thấy cách dùng phương thức Update của SqlDataAdapter để cập nhật các thay đổi vào database.

*“daSchools.Update(dsSchools, “Schools”);”*

Phương thức Update() trên được gọi trên thể hiện của SqlDataAdapter có tham số đầu tiên là chính đối tượng gọi phương thức. Tham số thử hai của phương thức Update() chỉ ra bảng nào trong DataSet sẽ được cập nhật. Bảng chứa một danh sách các dòng dữ liệu đã bị thay đổi và các property Insert, Update, Delete của SqlDataAdapter chứa các lệnh SQL dùng để thực hiện thay đối database.

#### 2.1.3.5 SqlDataAdapter

Khi dữ liệu mà bạn làm việc là read-only và bạn ít khi cần thay đổi dữ liệu nguồn. Vài trường hợp cần lưu trữ tạm dữ liệu trong bộ nhớ để hạn chế truy xuất đến database. Data adapter làm điều này dễ dàng bằng cách giúp bạn quản lý dữ liệu trong chế độ ngắt kết nối. Data adapter sẽ đổ vào DataSet khi đọc dữ liệu và thực hiện thay đổi dữ liệu một lượt vào database.

Data adapter chứa một tham chiếu đến đối tượng connection và mở/đóng kết nối tự động khi đọc và ghi dữ liệu vào database. Hơn nữa, data adapter chứa đối tượng command cho những thao tác SELECT, INSERT, UPDATE và DELETE trên dữ liệu. Bạn sẽ có một data adapter được định nghĩa cho mỗi table trong một DataSet và nó sẽ quản lý các giao tiếp với database cho bạn. Tất cả những gì bạn cần làm là chỉ cho data adapter khi nào nạp hoặc ghi vào database.

## 2.2 Entity Relationship Model

### 2.2.1 Giới thiệu về Entity Relationship Model (mô hình thực thể mối kế hợp)

Mô hình thực thể kết hợp (Entity Relationship Model) viêt tắt là ER được Giáo sư Peter Pin-Shan Chen giới thiệu vào năm 1976. Đây là một mô hình được sử dụng rộng rãi trong các bản thiết kế cơ sở dữ liệu ở mức quan niệm

So với mô hình mạng thì mô hình ER có nhiều ưu điểm hơn và nó thể hiện rõ hơn các thanhhf phần trong thế giới thực. Ví dụ trong mô hình mạng ta chỉ biểu diễn các đối tượng chính chứ không mô tả được các đặc điểm trong đối tượng đó, vayajj thì trong mô hình ER sẽ khắc phục được những yếu điểm này. Chính vì vậy việc lực chọn mô hình này luôn là quyết định của các nhà phân tích thiết kế CSDL

### 2.2.2 Loại thực thể trong mô hình ER

**Loại thực thể** (Entity Type) là là những đối tượng hay sự vật của thế giới thực cần quản lý trong ứng dụng. Các đối tượng ở đây có thể sờ thấy và cũng có thể mang tính chất ảo tưởng.

**Ví dụ**: Khi thiết kế cơ sở dữ liệu quản lý học viên thì ta sẽ có một số thực thể như sau:

* + - **HOCVIEN**
    - **LOP**
    - **...**

Trong mô hình ER thì ta sẽ dùng ký hiệu sau để mô tả loại thực thể:





Như vậy ta dùng ký hiệu hình chữ nhật và bên trong chính là tên của loại thực thể.

### 2.2.3 Thực thể (Entity) trong mô hình ER

Trong lập trình hướng đối tượng thì khi bạn tạo mới một lớp thì tức là bạn đã tạo một thể hiện của đối tượng mà class đó đang mô tả. Vậy thì trong mô hình thực thể mối kết hợp thì **thực thể** chính là một thể hiện của **loại thực thể**.

**Ví dụ**:  Loại thực thể HOCVIEN gồm các thực thể như sau:

* + - Sinh viên Trần Văn A
    - Sinh viên Trần Văn B
    - Sinh viên Trần Văn C

Tất cả những **học viên** trên chính là thực thể thể hiện cho loại thực thể HOCVIEN.

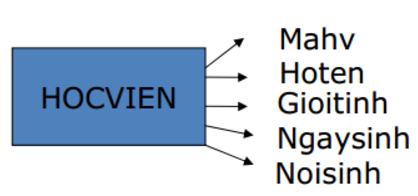
### 2.2.4 Thuộc tính của loại thực thể

Thuộc tính là những tính chất mô tả một đối tượng cụ thể nào đó. Ví dụ ta có chiếc mũ màu đỏ, chiếc áo màu vàng thì những màu sắc đỏ và vàng chính là những thuộc tính của mũ và áo.

Nếu ở đây ta nhắm tới ví dụ ở phần trên thì với loại thực thể HOCVIEN ta sẽ có một số thuộc tính như sau:

* Mahv
* Hoten
* Gioitinh
* Ngaysinh
* Noisinh

Và ta có ký hiệu như sau:



### 2.2.5 Các loại thuộc tính của loại thực thể

Có tổng cộng ba loại thuộc tính chính đó là **Đơn vị**, **Đa hợp** và **Đa trị**.

* **Đơn vị** (*simple*) là loại thuộc tính chỉ có một giá trị duy nhất. Ví dụ thuộc tính Noisinh, Mahv, ...
* **Đa hợp** (Composite) là thuộc tính có thể được tạo từ nhiều thành phần. Ví dụ thuộc tính Hoten có thể được tạo từ 3 thành phần đó là HỌ + TÊN LÓT + TÊN. Ký hiệu là HOTEN (HO, TENLOT, TEN)
* **Đa trị** (Multi valued) là loại thuộc tính có thể có nhiều giá trị đối với loại thực thể. Ví dụ bằng cấp thì ta có bằng loại tốt, loại khá, trung bình khá và giỏi. Ký hiệu là {BANGCAP}

### 2.2.6 Khóa của loại thực thể trong mô hình ER

Như định nghĩa trên thì mỗi **thực thể** sẽ là một thể hiện của **loại thực thể**, vậy thì làm sao để nhận diện các **thực thể** đó? Để nhận diện thì với mỗi **thực thể** chúng ta sẽ xác định cho nó một khóa duy nhất. Như vậy khóa của **loại thực thể** chính là dùng để nhận diện các **thực thể** của nó.

**Ví dụ**: trong **loại thực thể** HOCVIEN sẽ có nhiều **thực thể** học viên và mỗi học viên sẽ có một mã số duy nhất. Như vậy mã học viên chính là Key để ta nhận diện được học viên nào.

### 2.2.7 Các loại mối kết hợp trong mô hình ER

Trong mô hình mạng thì giữa hai đối tượng sẽ có một mối liên hệ và ta gọi nó là **loại liên hệ**. Vậy thì trong mô hình thực thể mối kết hợp ta gọi nó là loại mối kết hợp. Vậy loại mối kết hợp chính là mối liên hệ giữa hai loại thực thể.

**Ví dụ**: Giữa hai loại thực thể HOCVIEN và LOP ta sẽ có loại mối kết hợp THUOC, và ta biểu diễn như sau:

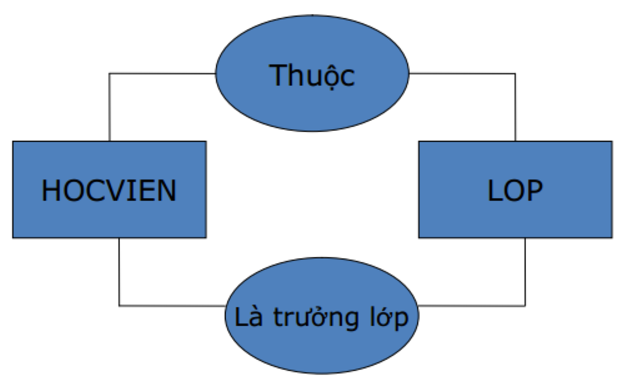


Biểu diễn bằng lời nói thì như sau "học viên sẽ thuộc lớp nào đó"

Ngoài ra giữa hai loại thực thể có thể tồn tại nhiều loại mối kế hợp.

**Ví dụ**:Giữa hai thực thể HOCVIEN và LOP sẽ có hai loại mối kế hợp nhu sau:

* + Học viên sẽ thuộc lớp nào đó
  + Học viên sẽ là lớp trường lớp nào đó



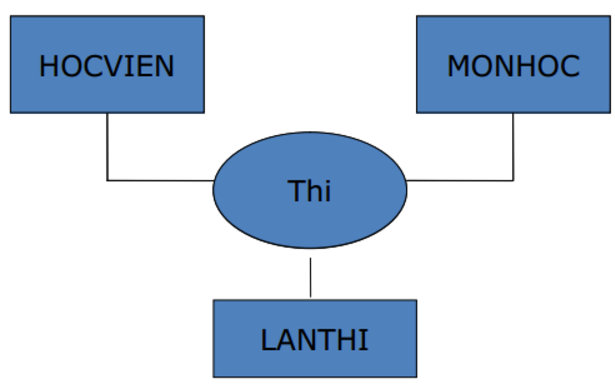
### 2.2.8 Số ngôi của loại mối kết hợp

**Số ngôi** chính là số **loại thực thể** mà **loại mối kết hợp** đó gắn kết. Như trong ví dụ 2 ở phần 7 thì **loại mối kết** hợp THUOC sẽ có số ngôi là 2 tại vì nó gắn kết giữa hai **loại thực thể** HOCVIEN và LOP.

**Ví dụ**: Giả sử ta có ba loại thực thể là:

* + **HOCVIEN**:học viên
  + MONHOC: môn học
  + LANTHI: lần thi

Ta sẽ có một **loại mối kết hợp** tên là THI để gắn ba **loại thực thể** này theo mô hình như sau:



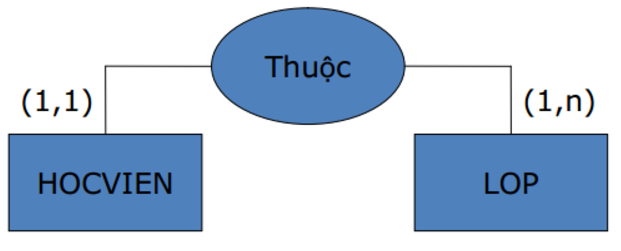
Như vậy **loại mối kết** hợp THI sẽ có số ngôi là 3

### 2.2.9 Bản số

Trong mô hình mạng thì ta cũng có sử dụng bản số để chỉ ra số lượng các mẫu tin tham gia vào các mối quan hệ, vậy thì trong mô hình ER cũng không có gì khắc biệt, chúng ta sử dụng những bản số để thể hiện số lượng tối thiểu và số lượng tối đa tham gia vào loại mối kết hợp.

Ký hiệu (min, max)

**Ví dụ**:  Loại mối kết hợp học viên (HOCVIEN) và lớp (LOP) có loại mối kết hợp thuộc:



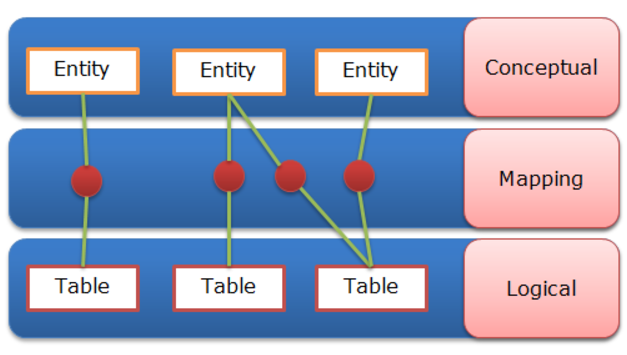
**Có nghĩa như sau:**

* Mỗi học viên sẽ thuộc một lớp
* Mỗi lớp có thể có 1 hoặc nhiều học viên

## 2.3 Entity Data Model

Entity Data Model (EDM) là mô hình dữ liệu được mô tả thông qua các ngôn ngữ theo chuẩn XML. EDM được chia làm 3 lớp là: Conceptual, Mapping và Logical. Mỗi lớp này được định nghĩa bởi ngôn ngữ riêng theo định dạng XML:

* **Conceptual – Conceptual Schema Definition Language (CSDL):** là ngôn ngữ định nghĩa các entity, relationship, hàm trong tập tin với phần mở rộng .csdl. Có thể tạo được các entity class (object layer)
* **Mapping – Mapping specification language (MSL)**: định nghĩa các ánh xạ giữa lớp conceptual và logical, nội dung này được lưu trong tập tin .msl
* **Logical – Store Schema Definition Language (SSDL):** định nghĩa mô hình lưu trữ của dữ liệu, lưu trữ trong tập tin .ssdl



Các nội dung của ba tập tin .csdl, .msl và .ssdl được lưu trữ trong cùng tập tin .**edmx** trong Visual Studio. Tập tin này có thể được tạo ra tự động từ database và công cụ Entity Framework Model Wizard của Visual Studio. Trong quá trình biên dịch, các tập tin .csdl, .msl và .ssdl sẽ được tạo ra dựa vào tập tin .edmx này.

Khi mở tập tin .edmx này, VS tự động hiển thị giao diện trực quan của nó bằng công cụ mặc định là ADO.NET Entity Data Model Designer. Để xem nội dung của tập tin này, bạn nhấn chuột phải, nhấn **Open with…**và chọn mục XML Editor hoặc một trình xem dưới dạng văn bản bất kì.

**Đoạn nội dung của CSDL**



**Đoạn nội dung của MSL**



**Đoạn nội dung của SSDL**



## 2.4 LINQ to SQL

### 2.4.1 Tổng quan LINQ to SQL

Language Integrated Query (LINQ) là một bộ thư viện được tích hợp vào nền tảng .NET 3.5 trở đi (19/11/2007). Mục đích ra đời của LINQ là cung cấp thêm một giải pháp truy vấn dữ liệu từ các nguồn dữ liệu như hệ quản trị cơ sở dữ liệu, file XML..., bên cạnh cách truyền thống thông qua các câu lệnh truy vấn SQL.

Trong phát triển phần mềm, nhu cầu truy xuất và thao tác dữ liệu là vô cùng cần thiết. Từ trước đến nay, để truy xuất và thao tác dữ liệu chúng ta sử dụng các truy vấn SQL. Tuy nhiên với những hệ quản trị cơ sở dữ liệu, file XML khác nhau các câu lệnh SQL sẽ khác biệt, do đó người lập trình phải nghiên cứu và chỉnh sửa lại đôi chút khi thao tác với từng loại cơ sở dữ liệu. Để tối ưu hóa qui trình phát triển phần mềm, Microsoft cho ra đời LINQ, một dạng truy vấn dữ liệu thông qua ngôn ngữ C#/Visual Basic .NET (VB.NET) kết hợp với những từ khóa, toán tử của LINQ là bạn dễ dàng truy vấn cũng như thao tác với dữ liệu theo phong cách C#/VB.NET. Điều này tạo cảm giác quen thuộc cho lập trình viên .NET.

### 2.4.2 Sự ra đời của LINQ to SQL

Mục đích ra đời của LINQ hay LINQ to SQL là để phục vụ cho hệ quản trị cơ sở dữ liệu SQL Server và nền tảng .NET nói chung hay ngôn ngữ lập trình C#/VB.NET nói riêng chứ không phải mục đích đại trà cho các ngôn ngữ lập trình hay các hệ quản trị cơ sở dữ liệu khác. Tôi thường sử dụng C# nên sử dụng C# làm ví dụ, ta biết rằng:

* C# là một ngôn ngữ lập trình thuần hướng đối tượng.
* SQL Server là một hệ quản trị cơ sở dữ liệu (DBMS) theo mô hình quan hệ, mô hình CSDL quan hệ ghi các dữ liệu theo dòng trong các bảng dữ liệu.

Sử dụng C# viết mã để lấy dữ liệu từ SQL Server nhưng tổ chức mô hình dữ liệu của C# và SQL Server là khác nhau. Để giải quyết tình trạng trên, một kỹ thuật gọi là ORM (Object Relational Mapping) ra đời nhằm mục đích chuyển đổi dữ liệu giữa các hệ thống khác (không phải là mô hình hướng đối tượng) sang các đối tượng trong ngôn ngữ lập trình hướng đối tượng. Tiếp đó LINQ to SQL ra đời dựa trên kỹ thuật ORM xóa bỏ khoảng cách giữa mô hình lập trình hướng đối tượng C#/VB.NET với hệ quản trị cơ sở dữ liệu SQL Server khi đã mô hình hóa theo hướng đối tượng các bảng trong Database thành các lớp tương ứng khi lập trình.

Sự ra đời của LINQ to SQL giúp các lập trình viên .NET bớt đi gánh nặng phụ thuộc bên thứ 3 (dùng SQL để truy vấn). LINQ to SQL đồng thời đồng bộ hóa dữ liệu lấy ra và trả về khi truy xuất dữ liệu bằng việc các Data Model hứng dữ liệu trả về được tạo tự động sao cho tương thích với kiểu dữ liệu tương ứng của chúng khi ánh xạ vào cơ sở dữ liệu. Điều này làm tránh tình trạng mất hoặc sai lệch dữ liệu khi truy xuất và thao tác với Database. Trên hết, một công cụ được phát hành cho nền tảng duy nhất là .NET với những anh em trong gia đình .NET sử dụng thì hiệu suất LINQ to SQL hằn là phải tốt hơn so với những công cụ bên thứ 3.

### 2.4.3 Công dụng của LINQ to SQL

#### 2.4.3.1 Hỗ trợ công cụ

Khi sử dụng LINQ to SQL với hệ quản trị cơ sở dữ liệu SQL Server, chúng ta không phải tạo các lớp Data Model để hứng dữ liệu trả về khi truy vấn dữ liệu vì LINQ to SQL đã tạo sẵn những lớp này với đầy đủ các thuộc tính và kiểu dữ liệu phù hợp với kiểu dữ liệu các cột bạn qui định trong Database (các thuộc tính của mỗi lớp ánh xạ vào các cột của bảng tương ứng trong CSDL).

#### 2.4.3.2 Giao diện trực quan và tự động

LINQ to SQL cung cấp giao diện trực quan về mối quan hệ các bảng dữ liệu sau khi được mô hình hóa. Các lớp DataContext sẽ được tạo ra tự động khi bạn Import file LINQ to SQL vào Project. Các lớp DataContext nhận nhiệm vụ mở kết nối đến cơ sở dữ liệu, thực hiện truy vấn hay thay đổi dữ liệu. Các lớp thuộc tính được mô hình hóa từ các bảng dữ liệu trong hệ quản trị cơ sở dữ liệu được truy cập thông qua các lớp DataContext. Lớp DataContext này gần như là một lớp bao (Wrapper Class), những thay đổi nếu có từ các bảng dữ liệu trong cơ sở dữ liệu thì lớp này sẽ cập nhật và thay đổi tương ứng (chúng cũng sẽ cập nhật vào các lớp Data Modal được tạo tự động). Điều này khiến cho việc thay đổi thuộc tính dữ liệu trong database diễn ra dễ dàng và người lập trình không mất quá nhiều công sức để chỉnh sửa lại code (do những thay đổi được tự động cập nhật lại).

#### 2.4.3.3 LINQ nhưng bản chất vẫn là SQL

Đúng như cái tên LINQ to SQL, các câu truy vấn LINQ sẽ được chuyển sang câu truy vấn SQL trước khi đưa vào SQL Server để truy vấn dữ liệu (LINQ to SQL giống như việc mang SQL vào và viết bằng C#, tuy vậy bản chất vẫn là SQL).

LINQ to SQL là một công cụ tốt đối với những lập trình viên .NET, nếu có thời gian bạn nên tìm hiểu công cụ này. Cấu trúc, câu lệnh của nó dễ học và quen thuộc (C#/VB.NET). LINQ to SQL rút ngắn thời gian phát triển phần mềm trên nền tảng .NET khi muốn thao tác và truy xuất dữ liệu từ cơ sở dữ liệu, hệ thống tạo vào hỗ trợ tất cả những thứ cơ bản như lớp DataContext, các lớp Data Model để hứng dữ liệu, lập trình viên chỉ cần truy xuất và thao tác với dữ liệu.

### 2.4 Hạn chế của LINQ to SQL

Dù là một công cụ khá hay trên nền tảng .NET, LINQ to SQL vẫn có hạn chế như:

* Chỉ thao tác duy nhất với hệ quản trị cơ sở dữ liệu SQL Server.
* Chỉ có thể tự động tạo Data Model từ Database chứ không thể tạo Database từ Data Model.
* Chỉ cho phép ánh xạ 1:1 giữa các Table trong Database với các lớp Data Model (tức là không thể tạo 1 Data Model là kết quả kết hợp từ 2 bảng dữ liệu trở lên).

# CHƯƠNG 3. ENTITY FRAMEWORK

## 3.1 Tổng quan về Entity Framework

### 3.1.1 Giới thiệu

Microsoft ADO.NET Entity Framework là một nền tảng sử dụng làm việc với cơ sở dữ liệu quan hệ thông qua cơ chế ánh xạ đối tượng Object / Relational Mapping (ORM).

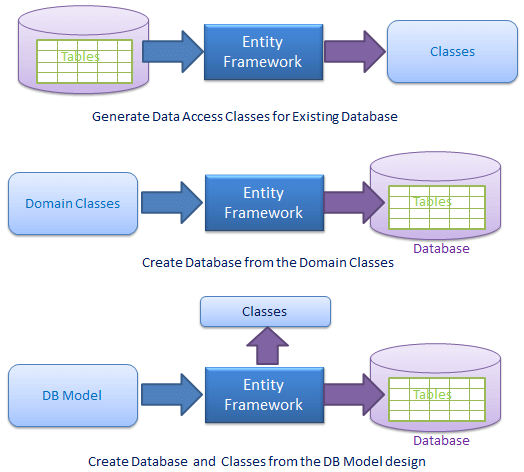
Entity Framework cho phép các nhà phát triển làm việc với cơ sở dữ liệu quan hệ trên mô hình các đối tượng không phải làm việc trực tiếp trên các bảng trong cơ sở dữ liệu, do đó người phát triển không phải viết các đoạn mã truy cập cơ sở dữ liệu thường cần phải viết.

Sử dụng Entity Framework, các nhà phát triển ứng dụng sử dụng các truy vấn LINQ, sau đó lấy ra và thao tác dữ liệu như các đối tượng. Ngoài ra  Entity Framework cung cấp các dịch vụ như theo dõi sự thay đổi, lazy loading, và biên dịch truy vấn để các nhà phát triển có thể tập trung vào xử lý nghiệp vụ ứng dụng, không phải tập trung vào các truy cập dữ liệu cơ bản.

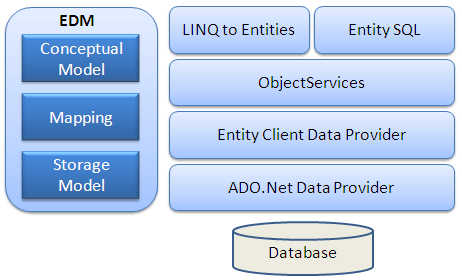
Entity framework là một nền tảng nâng cao của ADO.NET vì vậy Entity framework cung cấp cho các nhà phát triển ứng dụng một cơ chế tự động cho việc truy cập và lưu trữ dữ liệu trong cơ sở dữ liệu.

**Entity Framework sử dụng trong 3 trường hợp:**

* Thứ nhất, nếu đã có cơ sở dữ liệu hoặc muốn thiết kế cơ sở dữ liệu trước các phần khác của ứng dụng.
* Thứ hai, thiết kế các lớp domain trước và sau đó tạo ra cơ sở dữ liệu từ các lớp domain.
* Thứ ba, thiết kế lược đồ dữ liệu trên thiết kế trực quan và sau đó tạo ra các cơ sở dữ liệu và các lớp.



### 3.1.2 Kiến trúc của Entity Framework

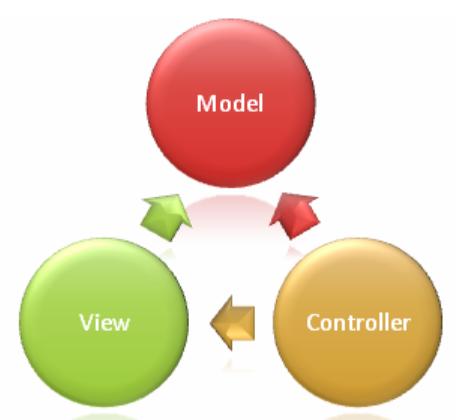


Kiến trúc Entity Framework bao gồm các thành phần sau:

* **EDM (Entity Data Model)**: EDM bao gồm ba phần chính – mô hình khái niệm, ánh xạ và mô hình lưu trữ.
* **Mô hình khái niệm (Conceptual Model)**: Mô hình khái niệm chứa các lớp model và các mối quan hệ.
* **Mô hình lưu trữ (Storage Model)**: Mô hình lưu trữ là mô hình thiết kế cơ sở dữ liệu bao gồm bảng, view, thủ tục lưu trữ, và các mối quan hệ và khóa chính.
* **Ánh xạ (Mapping)**: Ánh xạ bao gồm các thông tin về cách thức mô hình khái niệm được ánh xạ tới mô hình lưu trữ.
* **LINQ to Entities:** LINQ to Entities là ngôn ngữ truy vấn được sử dụng để viết các truy vấn trên mô hình đối tượng. Nó trả về các thực thể, được định nghĩa trong mô hình khái niệm. Chúng ta có thể sử dụng các câu lệnh LINQ ở đây.
* **Entity SQL**: Entity SQL là ngôn ngữ truy vấn giống như LINQ to Entities. Tuy nhiên, Entity SQL khó hơn LINQ to Entities và các nhà phát triển sẽ phải tìm hiểu nó một cách riêng biệt.
* **Object Service**: Object Service là một điểm vào chính cho việc truy cập dữ liệu từ cơ sở dữ liệu và trả lại Object Service trở lại. Object Service có nhiệm vụ chuyển đổi dữ liệu thực thể được cung cấp bởi client thành thực thể cấu trúc đối tượng.
* **Entity Client Data Provider**: Nhiệm vụ chính của lớp này là chuyển đổi L2E hoặc Entity SQL thành câu lệnh truy vấn SQL mà cơ sở dữ liệu thực thi được.
* **ADO.Net Data Provider**: Lớp này có nhiệm vụ giao tiếp với cơ sở dữ liệu bằng cách sử dụng ADO.Net chuẩn.

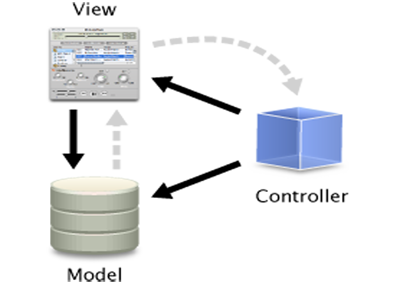
## 3.2 Mô hình MVC

### 3.2.1 Giới thiệu về mô hình MVC



Mô hình MVC (Model - View - Controller) là một kiến trúc phần mềm hay mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. Nó giúp cho các developer tách ứng dụng của họ ra 3 thành phần khác nhau Model, View và Controller. Mỗi thành phần có một nhiệm vụ riêng biệt và độc lập với các thành phần khác.

### 3.2.2 Các thành phần trong MVC



**Model:** Đây là thành phần chứa tất cả các nghiệp vụ logic, phương thức xử lý, truy xuất database, đối tượng mô tả dữ liệu như các Class, hàm xử lý...

**View:** Đảm nhận việc hiển thị thông tin, tương tác với người dùng, nơi chứa tất cả các đối tượng GUI như textbox, images...Hiểu một cách đơn giản, nó là tập hợp các form hoặc các file HTML.

**Controller:** Giữ nhiệm vụ nhận điều hướng các yêu cầu từ người dùng và gọi đúng những phương thức xử lý chúng... Chẳng hạn thành phần này sẽ nhận request từ url và form để thao tác trực tiếp với Model.

### 3.2.3 Sự tương tác giữa các lớp trong mô hình MVC

**Controller – View** sẽ lấy những hình ảnh, nút bấm…hoặc hiển thị dữ liệu được trả ra từ Controller để người dùng có thể quan sát và thao tác. Trong sự tương tác này cũng có thể không có dữ liệu được lấy từ Model và khi đó nó chỉ chịu trách nhiệm hiển thị đơn thuần như hình ảnh, nút bấm…

**Controller – Model** là luồng xử lý khi controller tiếp nhận yêu cầu và các tham số đầu vào từ người dùng, controller sẽ sử dụng các lớp/hàm trong Model cần thiết để lấy ra những dữ liệu chính xác.

**View – Model** có thể tương tác với nhau mà không qua Controller, nó chỉ đảm nhận hiển thị dữ liệu chứ không phải qua bất kỳ xử lý nghiệp vụ logics nào. Nó giống như các vùng dữ liệu hiển thị tĩnh trên các website như block slidebar…

Mô hình MVC có thể áp dụng cho nhiều loại dự án khác nhau : dự án website, dự án ứng dụng – phần mềm,… nhưng trong phạm vi bài viết chúng ta sẽ chỉ đi tìm hiểu mô hình này được áp dụng như nào trong các dự án xây dựng và phát triển website.

### 3.2.4 Ưu điểm

* Các dự án có thể áp dụng ngay mô hình MVC mà không phụ thuộc môi trường, nền tảng xây dựng hay ngôn ngữ lập trình phát triển;
* Quy hoạch các class/ function vào các thành phần riêng biệt Controller – Model – View, khi đó sẽ dễ dàng xây dựng – phát triển – quản lý – vận hành và bảo trì một dự án, tạo sự rõ ràng, trong sáng trong quá trình phát triển dự án, kiểm soát được các luồng xử lý và tạo ra các thành phần xử lý nghiệp vụ chuyên biệt hóa.
* Tạo thành mô hình chuẩn cho nhiều dự án, các chuyên gia sẽ tiếp cận – tìm hiểu những dự án đó một cách nhanh chóng và hiệu quả . Nếu bạn nắm rõ mô hình MVC của một dự án nào đó, thì khi tiếp cận với một dự án khác mà bạn chưa từng biết hoặc tiếp xúc, nhưng nó lại được xây dựng với mô hình MVC thì sẽ không khó khăn gì mà cực kỳ dễ dàng. Học một nhưng có thể hiểu và sử dụng được mười.
* Giúp các chuyên gia lập trình, nhà quản lý, nhà đầu tư, PM… có thể hiểu được dự án hoạt động ra sao hoặc giúp các lập trình viên dễ dàng quản lý – phát triển dự án. Nó không phải ngôn ngữ, nhưng khi họ cùng nhìn vào nó thì sẽ tự hiểu nó là gì, khi đó họ có thể trao đổi các yêu cầu và bàn bạc công việc.
* Đây là một mô hình chuẩn, nó tối ưu nhất hiện nay so với nhiều mô hình khác và được sử dụng trong nhiều dự án và nhiều lĩnh vực, đặc biệt trong công nghệ sản xuất ứng dụng – phần mềm. Các lập trình viên sử dụng mô hình chuẩn MVC để có thể dễ dàng phân phối và chuyển giao công nghệ.
* Đây là mô hình đơn giản, xử lý những nghiệp vụ đơn giản, và dễ dàng triển khai với các dự án nhỏ.

### 3.2.5 Nhược điểm

Yêu cầu về chuyên môn khá cao, có kiến thức vững về các mô hình chuẩn. Khó triển khai với những dự án yêu cầu phức tạp hơn. Hiện nay đang có một khái niệm mô hình mới đó là HMVC đang dần thay thế cho MVC

### 3.2.6 Liên hệ MVC và Entity Framework



Trong hình trên, Entity Framework dường như có vị trí rất tự do, xuất hiện ở cả 3 tầng Data Access Layer, Business Layer, và cả Presentation Layer. Có thể hình dung Entity Framework như là 1 thư viện, hễ tầng nào cần thì chỉ cần gọi đến. Mô hình này xuất phát có thể là sự cẩu thả của lập trình viên khi không thích tuân theo tiêu chuẩn quan hệ giữa các tầng hoặc cách giải thích khác là sự lỏng lẻo giúp cho việc lập trình thuận tiện hơn. Trong vài mô hình người ta gom tầng Data Access Layer và Business Layer chỉ làm tầng Data Model, và vị trí Entity Framework cũng bao hàm tầng Data Model và Presentation Layer như hình.

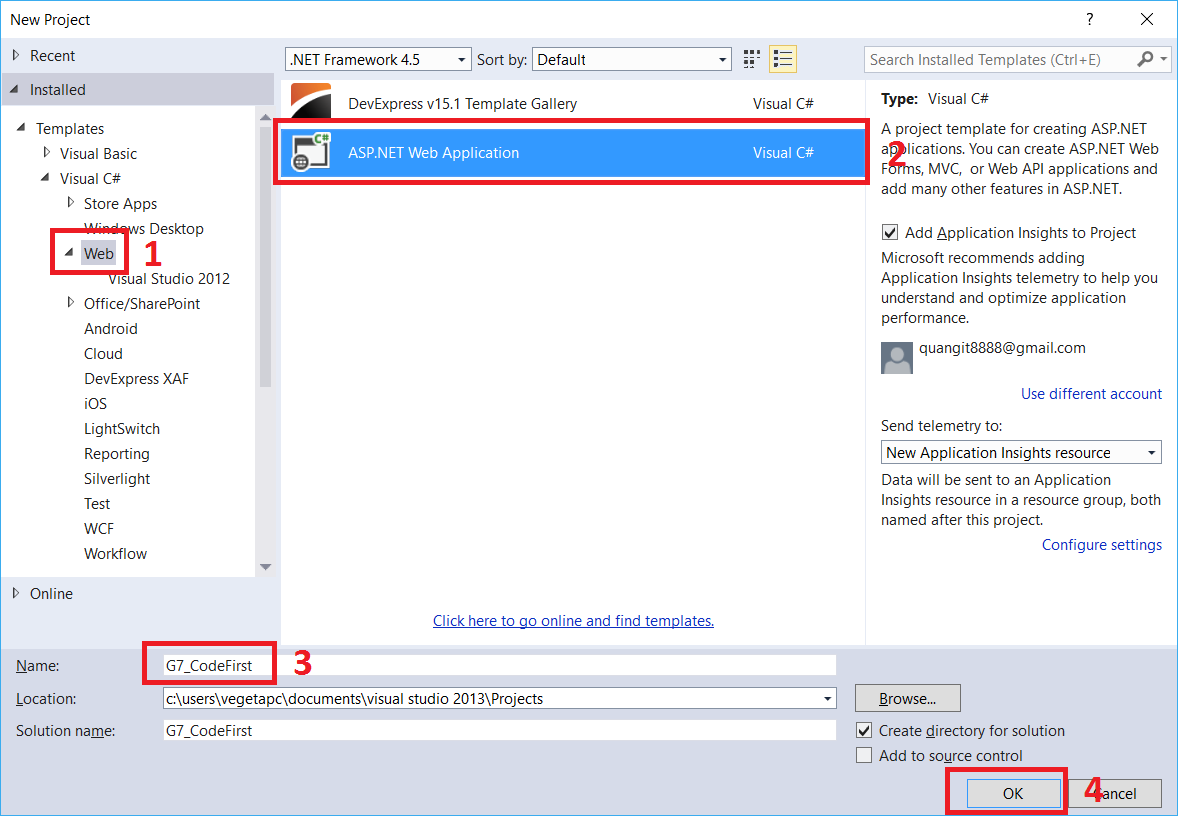
Có thể hiểu 1 cách chung nhất Entity Framework có vị trí trung gian, đóng vai trò kết nối giữa cơ sở dữ liệu và các thành phần khác của 1 dự án Web khi cần đến. Ngoài ra, có nhiều cách hiểu về vị trí của Entity Framework ở đâu trong mô hình Web, ta sẽ dần khám phá để đưa ra cách hiểu và định nghĩa riêng trong quá trình thiết kế và xây dựng dự án Web ASP.NET.

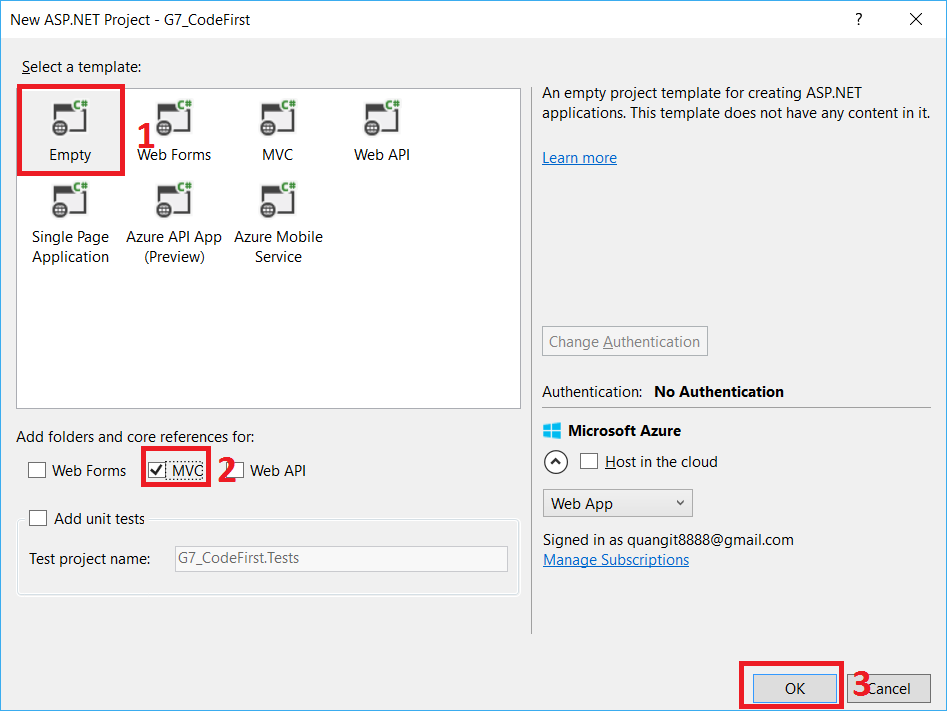
# CHƯƠNG 4. PHÁT TRIỂN ỨNG DỤNG VỚI ENTITY FRAMEWORK

Mục tiêu: Tạo website ASP.NET kết hợp MVC để đăng ký môn học cho sinh viên gồm các chức năng CRUID (Create – Read – Update – Insert – Delete) bằng 3 cách: Code first, model first và database first.

1. **Code first**

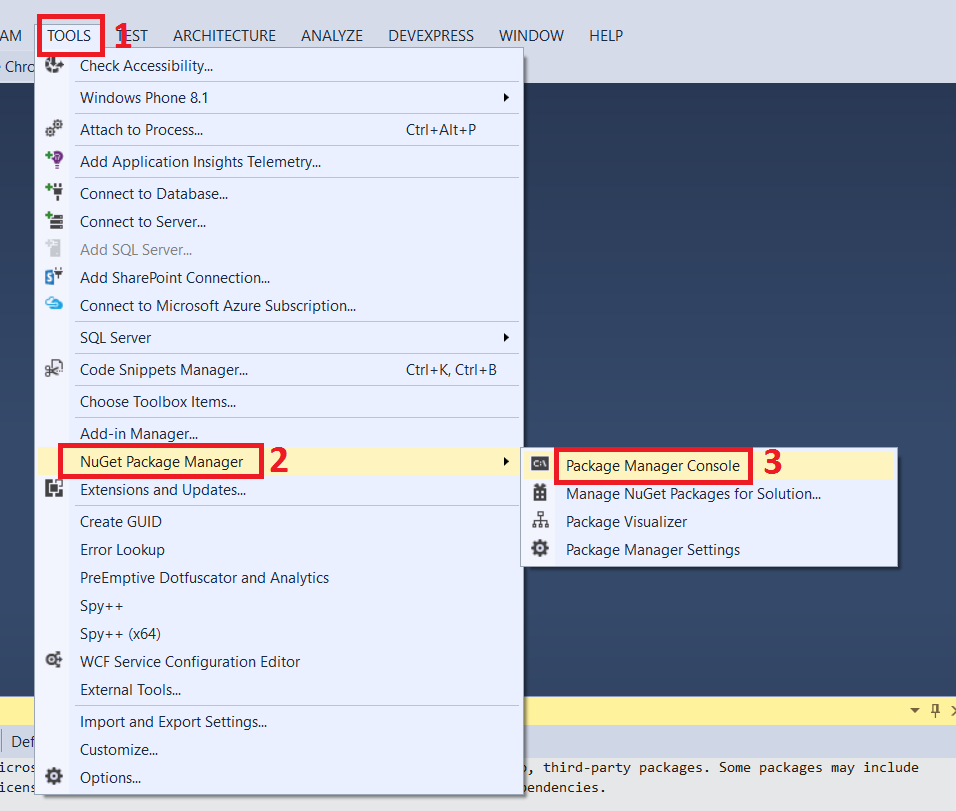
Tạo project web ASP.NET Web Application MVC:



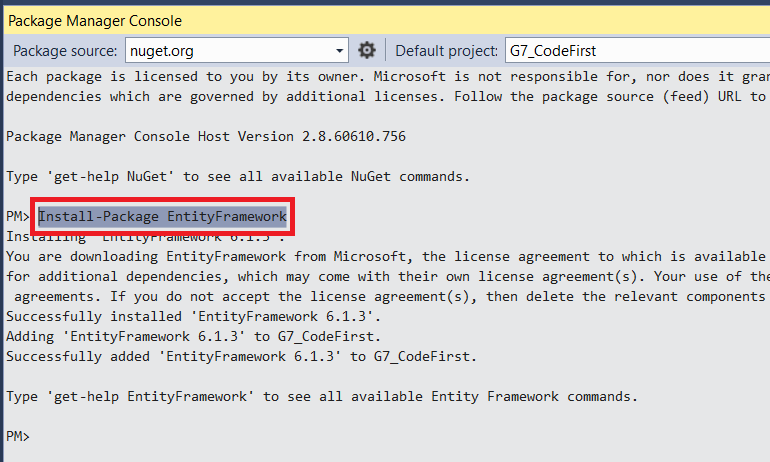


Cài đặt Entity Framework vào project.

Cách cài đặt bằng Package manager console:



Nhập “Install-Package EntityFramework” vào console sau đó enter.



Tạo 3 class như sau ở trong thư mục Models với các thuộc tính và phương thức get set như sau.

public class Student

{

[Key]

public int StudentID { get; set; }

public string StudentName { get; set; }

public int StudentAge { get; set; }

public float StudentGrade { get; set; }

public int StudentYearOfGraduation { get; set; }

}

public class Subject

{

[Key]

public int SubjectID { get; set; }

public string SubjectName { get; set; }

public int SubjectNumOfStudent { get; set; }

}

public class RegisterSubject

{

[Key]

public int RegSubID { get; set; }

public int? StudentID { get; set; }

[ForeignKey("StudentID")]

public virtual Student Student { get; set; }

public int? SubjectID { get; set; }

[ForeignKey("SubjectID")]

public virtual Subject Subject { get; set; }

}

Tạo class SchoolContext kế thừa từ class DbContext (cũng trong thư mục Models).

public class SchoolContext : DbContext

{

public SchoolContext()

: base("name=SchoolContext")

{

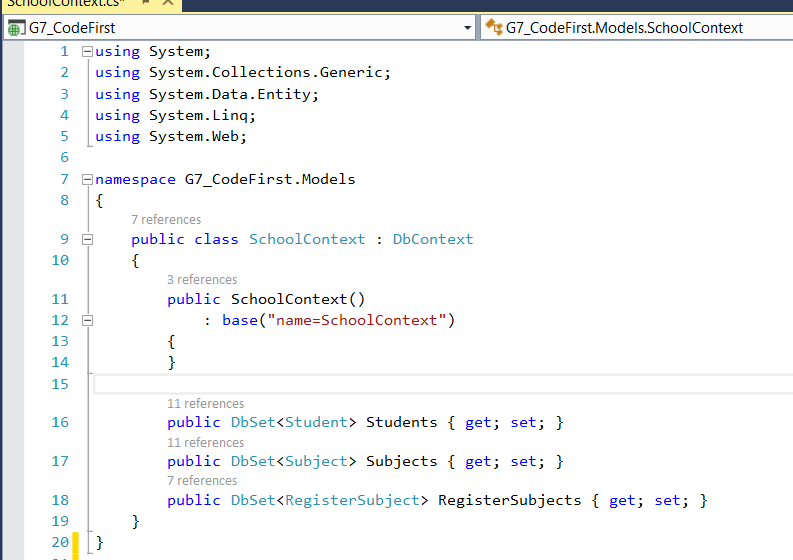
}

public DbSet<Student> Students { get; set; }

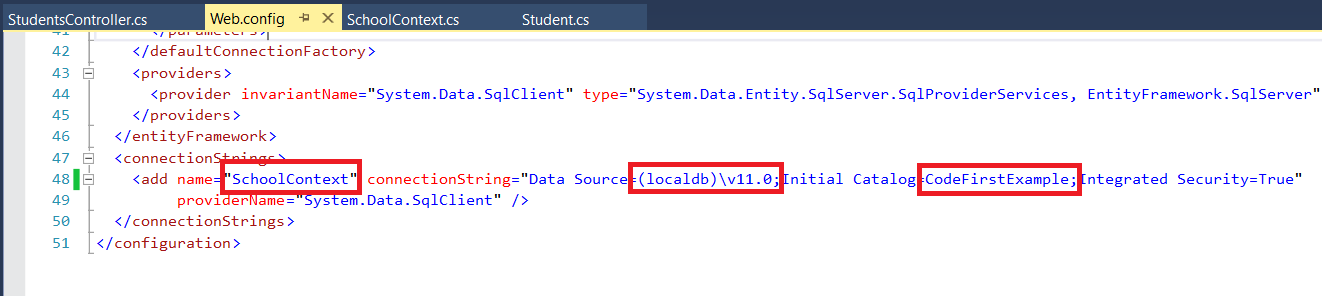
public DbSet<Subject> Subjects { get; set; }

public DbSet<RegisterSubject> RegisterSubjects { get; set; }

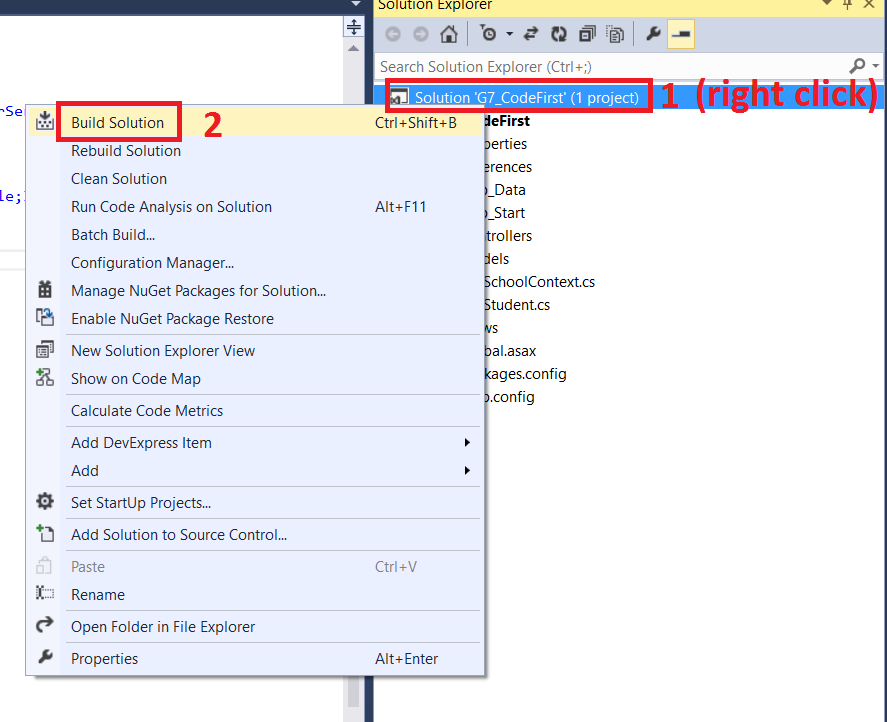
}



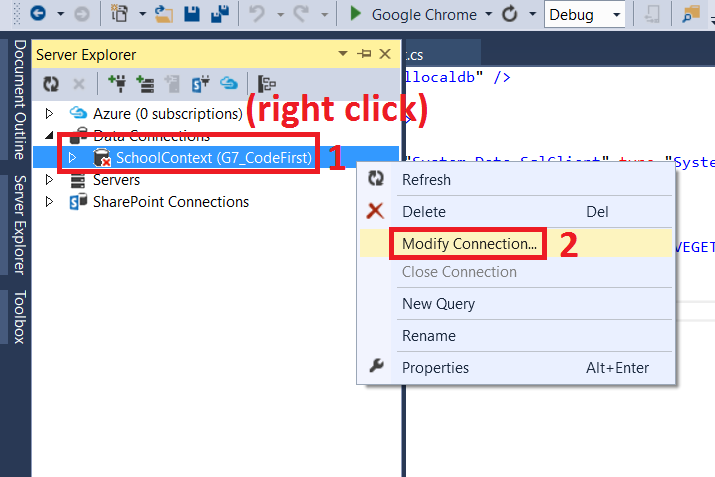
Khai báo chuỗi connectionStrings như sau để Code First làm việc được với SQL Server ở trong file Web.config.

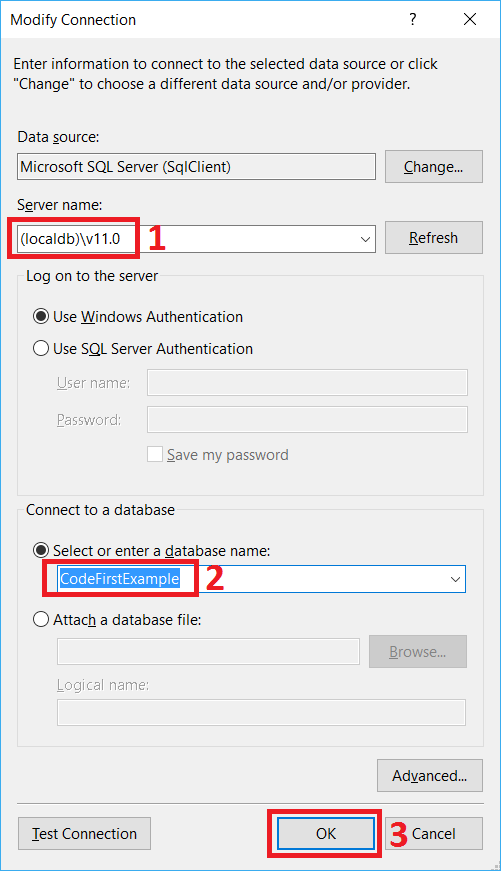


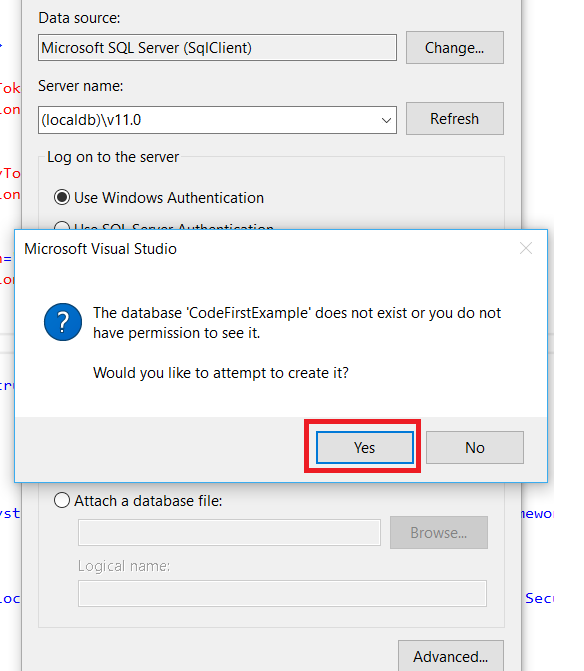
Build lại project.



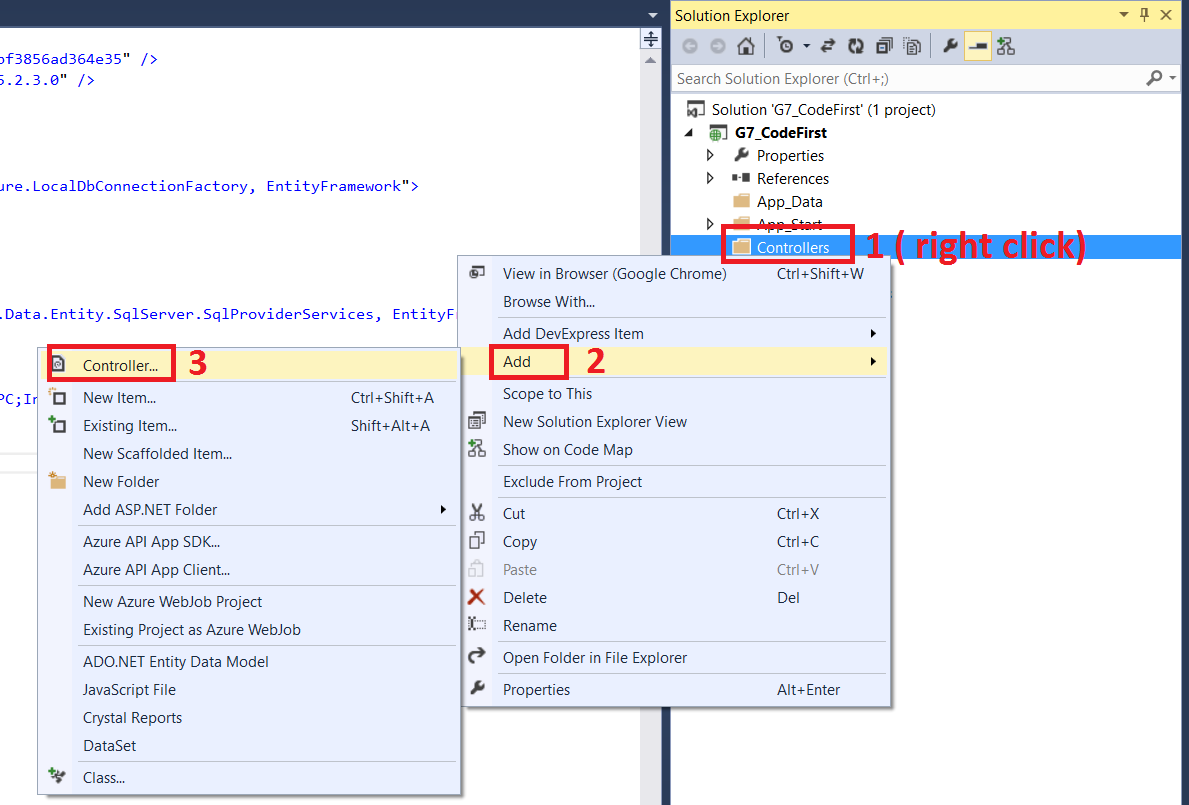
Tạo database (nếu chưa có)

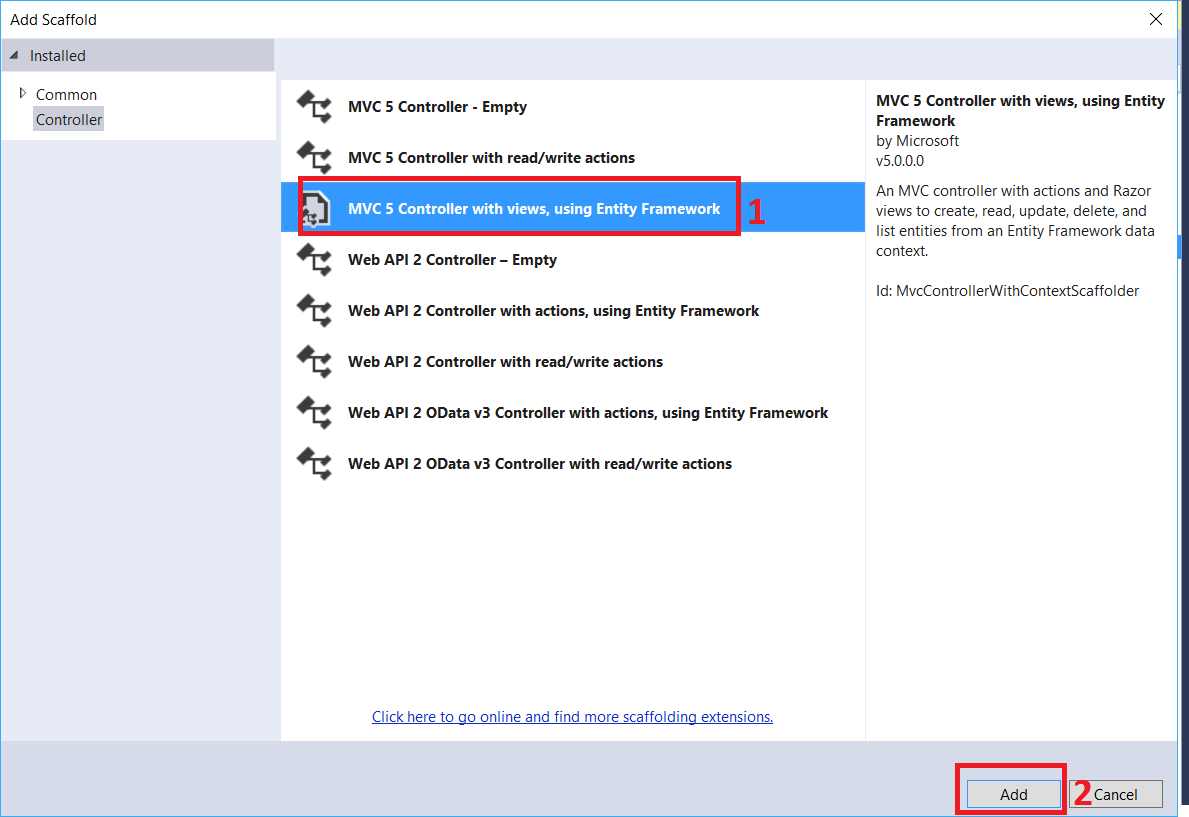


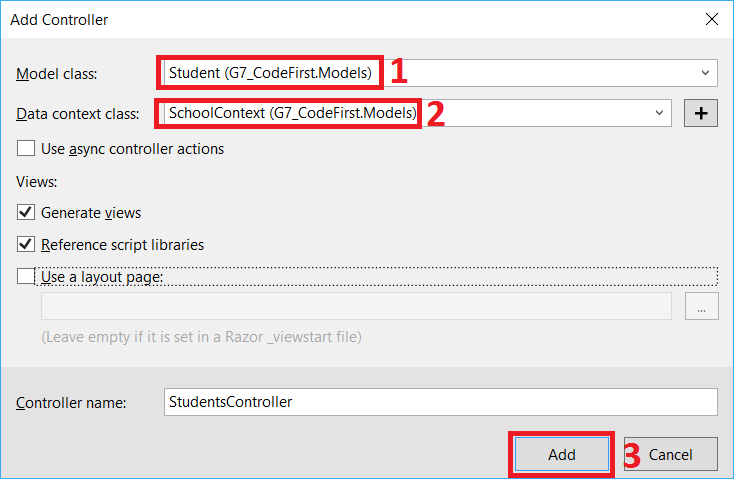




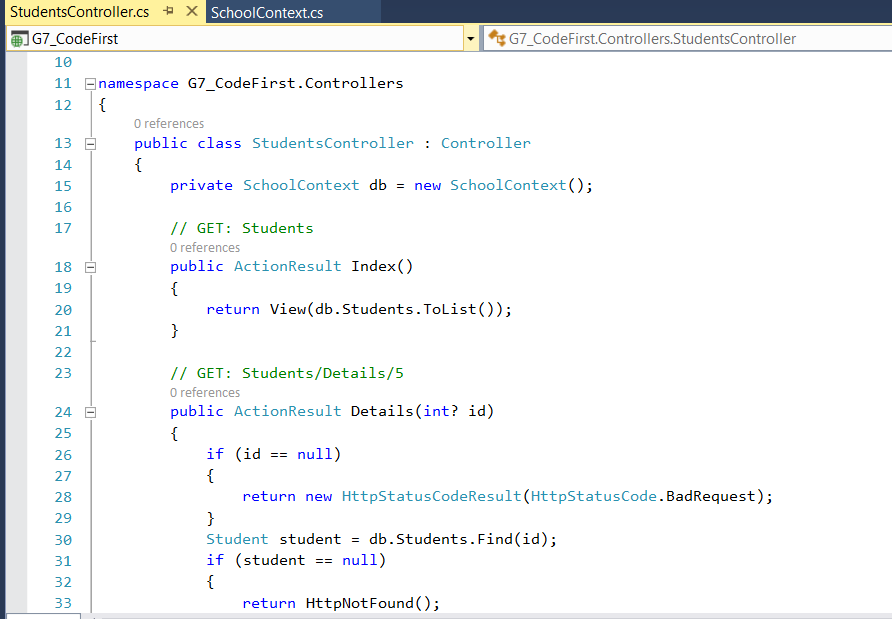
Thêm controller cho model vừa tạo.



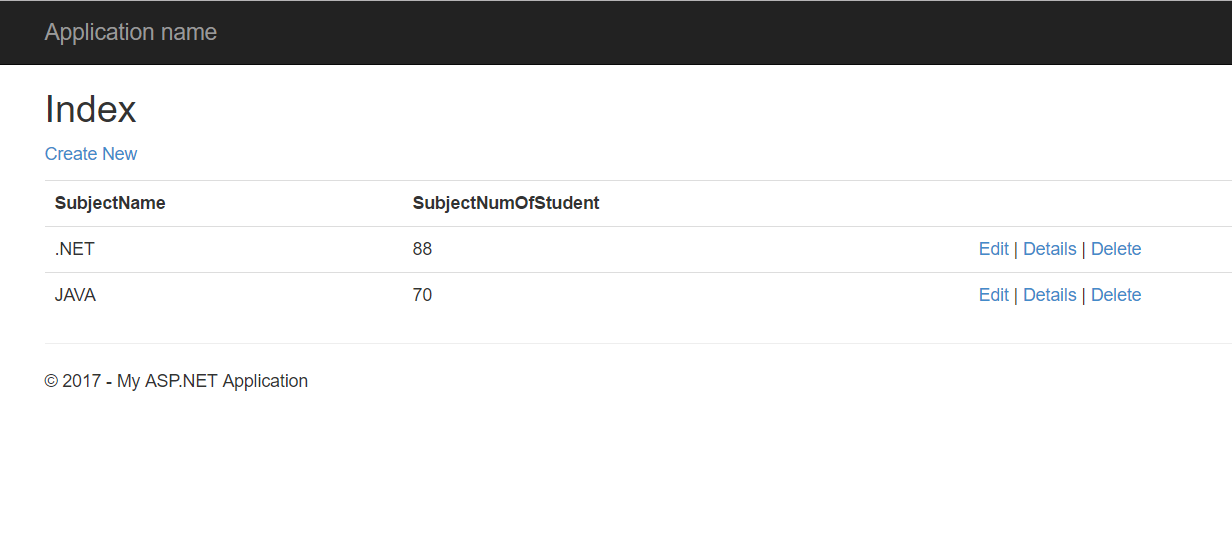




Controller sẽ được tự động gen ra các phương thức CRUD.



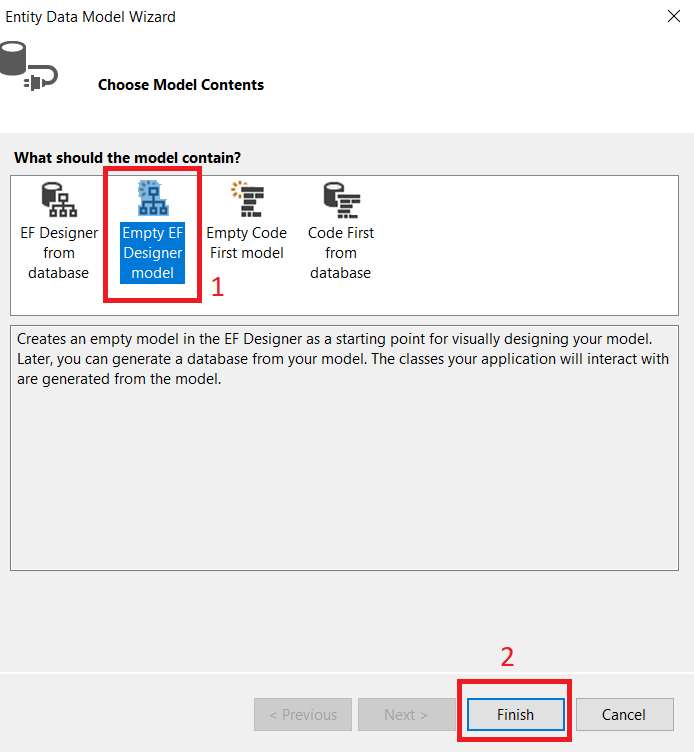
Làm tương tự với 2 class còn lại. Kết quả hiển thị.



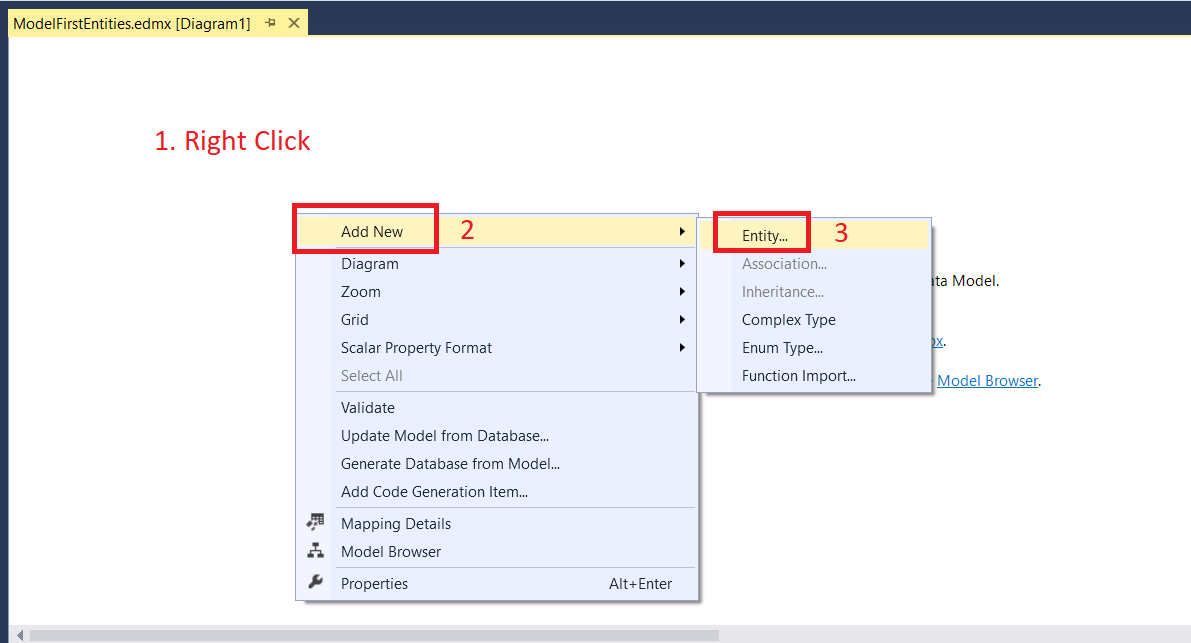
1. **Model first**

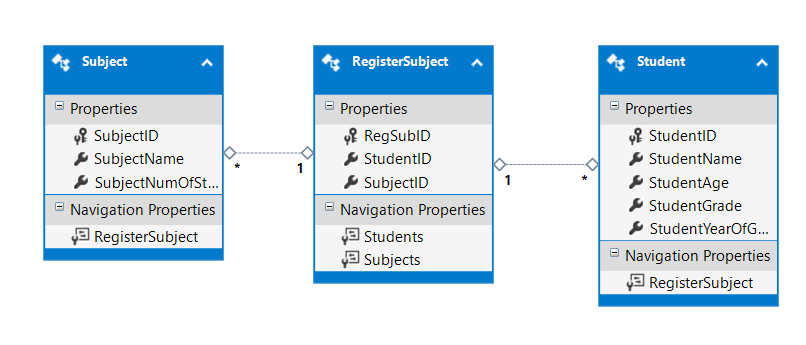
Tạo project mới giống như CodeFirst.

Sau đó thêm mới item EDM đặt tên là ModelFirstEntities trong thư mục Models như sau:

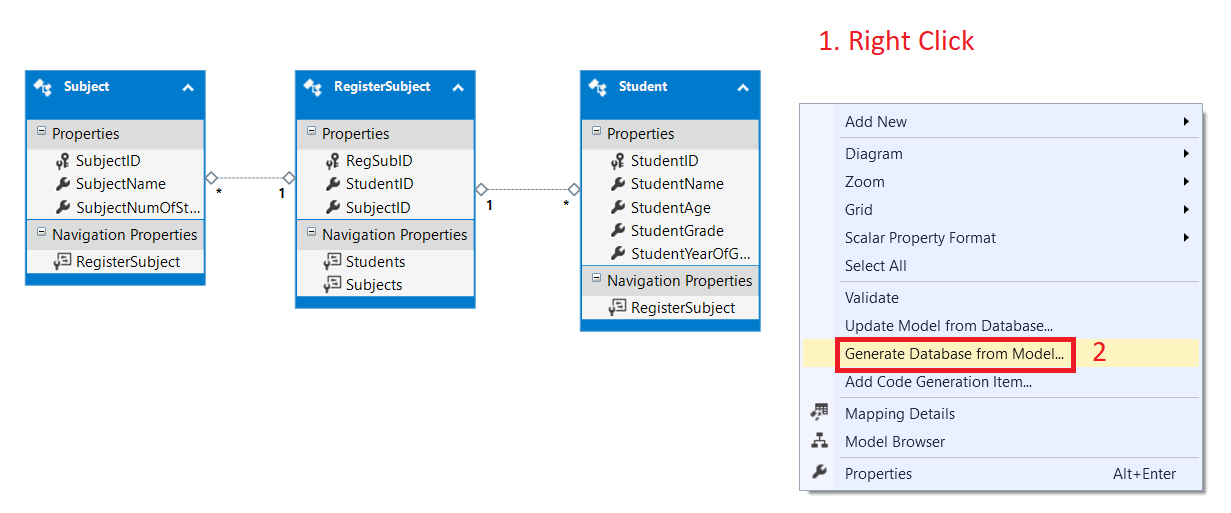


Sau đó tạo 3 entity gồm Student, Subject, RegisterSubject với khoá chính, khoá ngoại và các thuộc tính như sau:

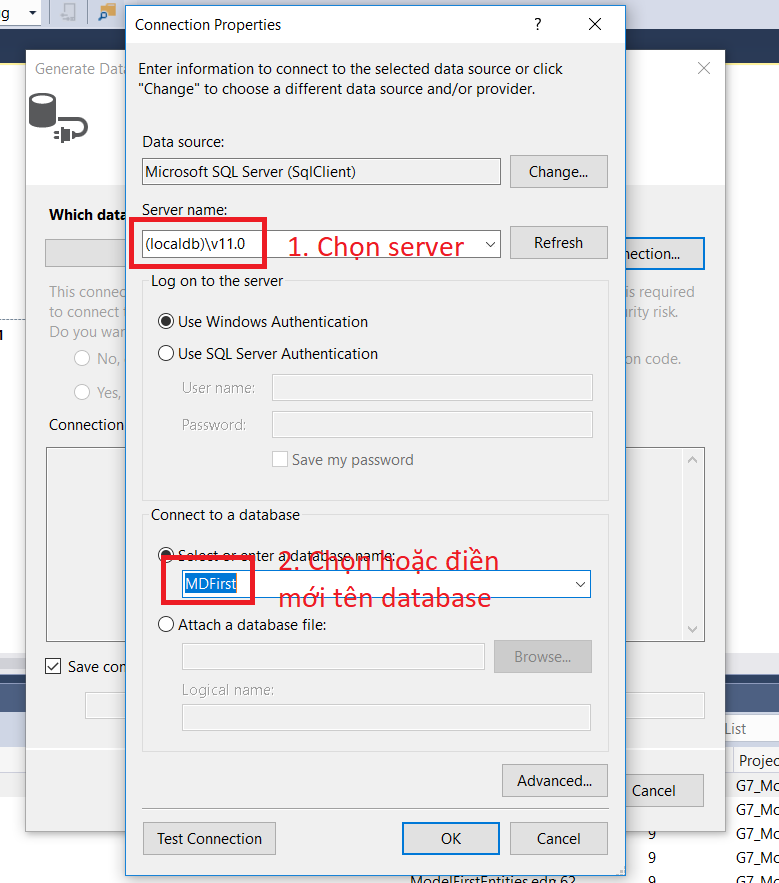




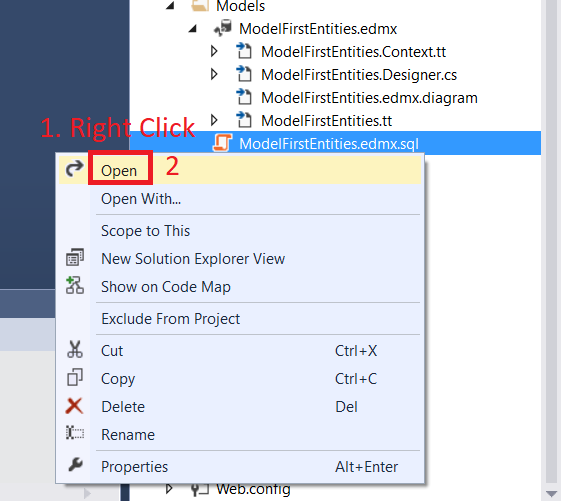
Sau khi tạo model, tiến hành generate database từ model:

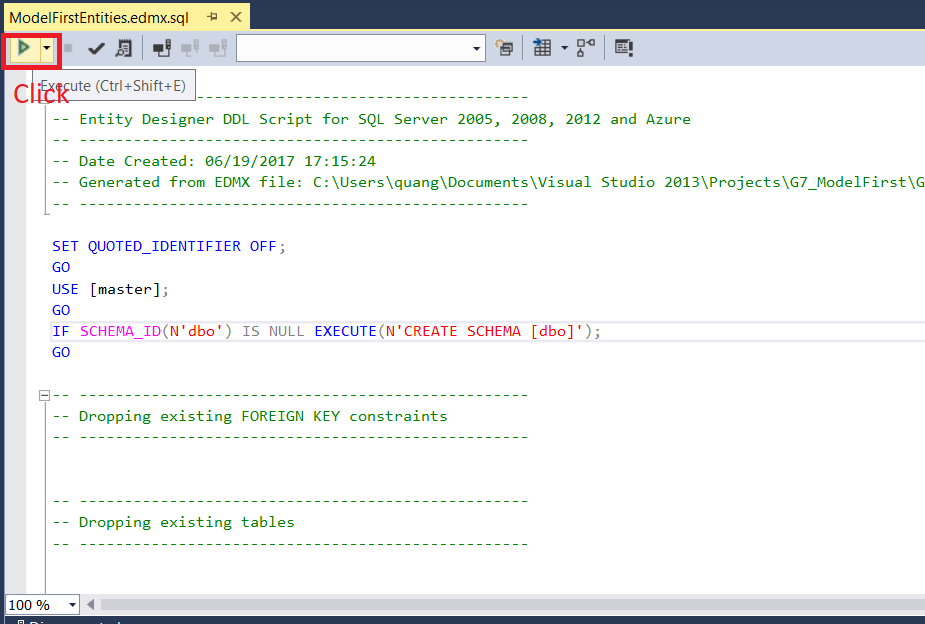


Chọn CSDL



Sau đó chạy file ModelFirstEntities.edmx.sql để thực thi các câu lệnh tạo bảng:





Các bước tiếp theo để tạo website tương tự CodeFirst.

Tiếp tục từ bước tạo Controller.

1. **Database first**

Tạo database như sau hoặc sử dụng file: /SOURCE CODE/dbfrist.sql

CREATE TABLE [dbo].[Students](

[StudentID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

[StudentName] [nvarchar](max) NULL,

[StudentAge] [int] NULL,

[StudentGrade] [real] NULL,

[StudentYearOfGraduation] [int] NULL

);

CREATE TABLE [dbo].[Subjects](

[SubjectID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

[SubjectName] [nvarchar](max) NULL,

[SubjectNumOfStudent] [int] NULL

);

CREATE TABLE [dbo].[RegisterSubjects](

[RegSubID] [int] IDENTITY(1,1) NOT NULL PRIMARY KEY,

[RegStudent\_StudentID] [int] NULL,

[RegSubject\_SubjectID] [int] NULL

);

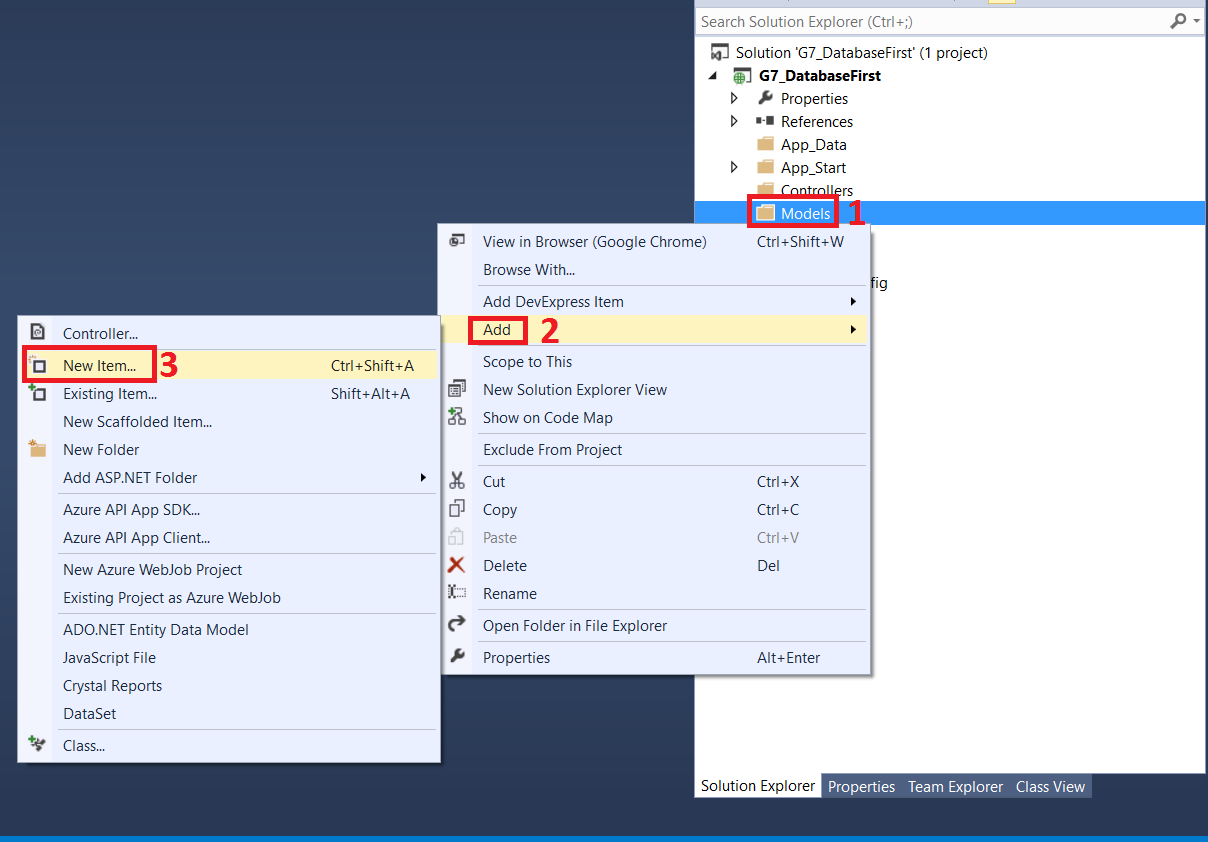
ALTER TABLE [dbo].[RegisterSubjects] ADD CONSTRAINT [FK\_RegisterSubjects\_Students] FOREIGN KEY([RegStudent\_StudentID])

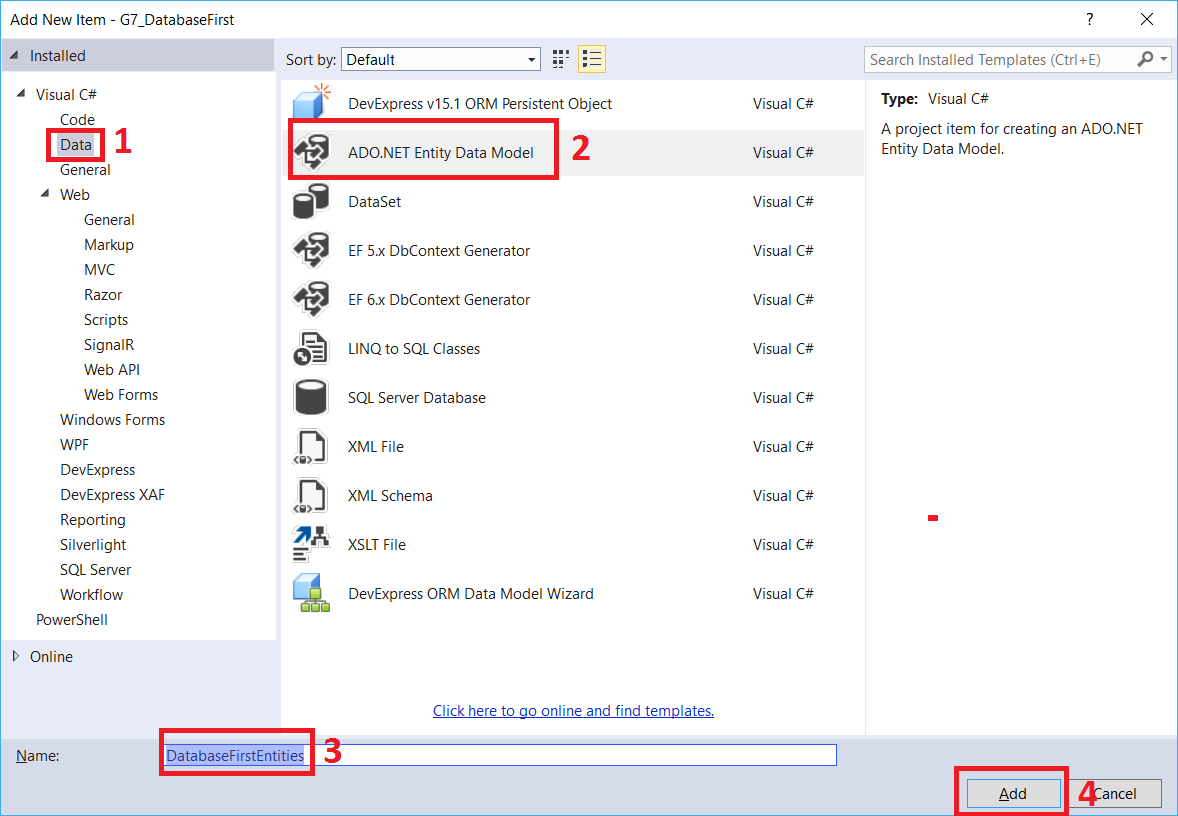
REFERENCES [dbo].[Students]([StudentID])

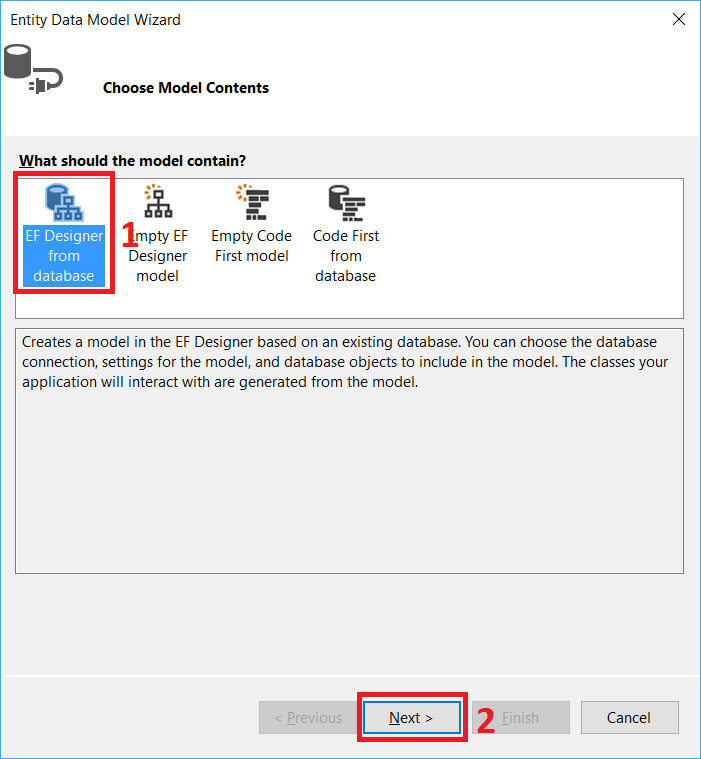
ALTER TABLE [dbo].[RegisterSubjects] ADD CONSTRAINT [FK\_RegisterSubjects\_Subjects] FOREIGN KEY([RegSubject\_SubjectID])

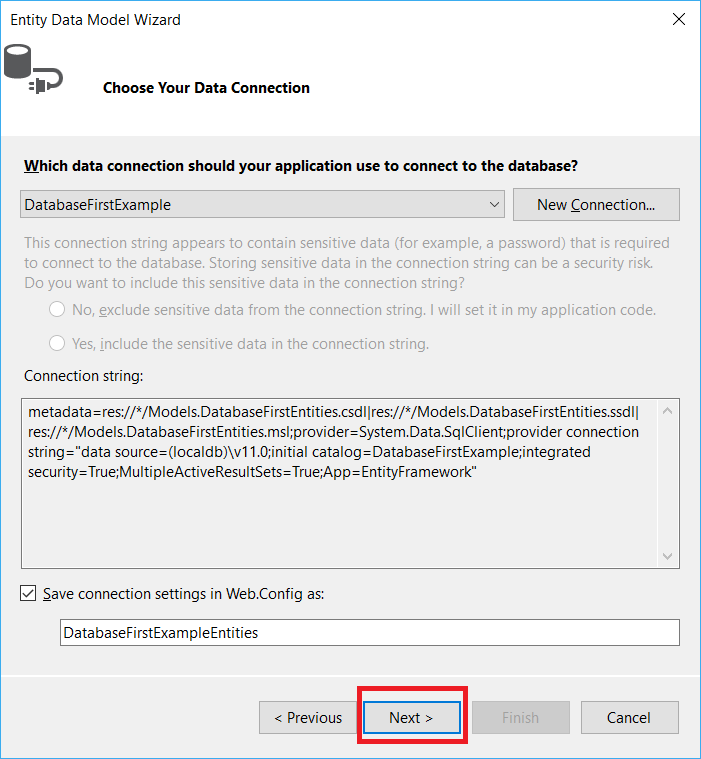
REFERENCES [dbo].[Subjects] ([SubjectID])

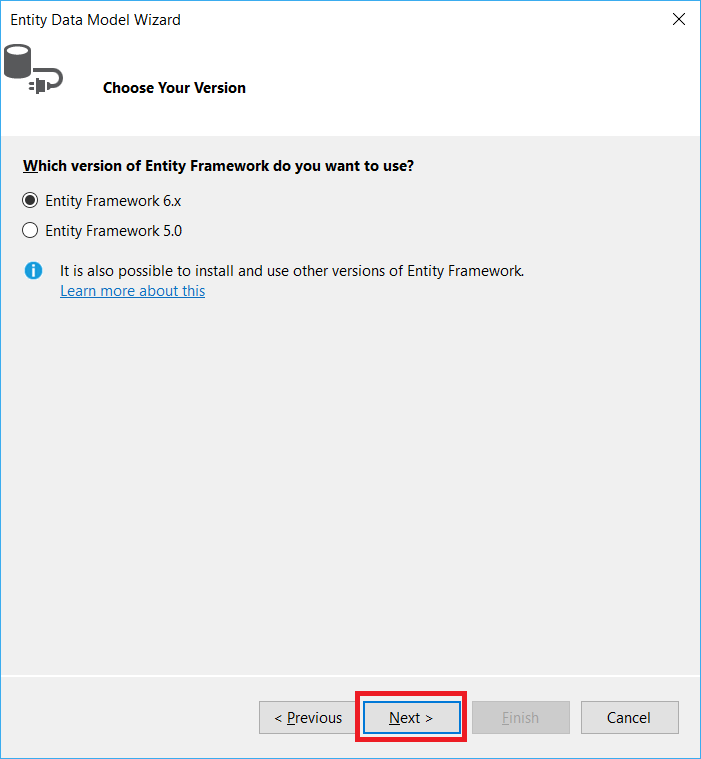
Thêm item EDM vào thư mục Models.



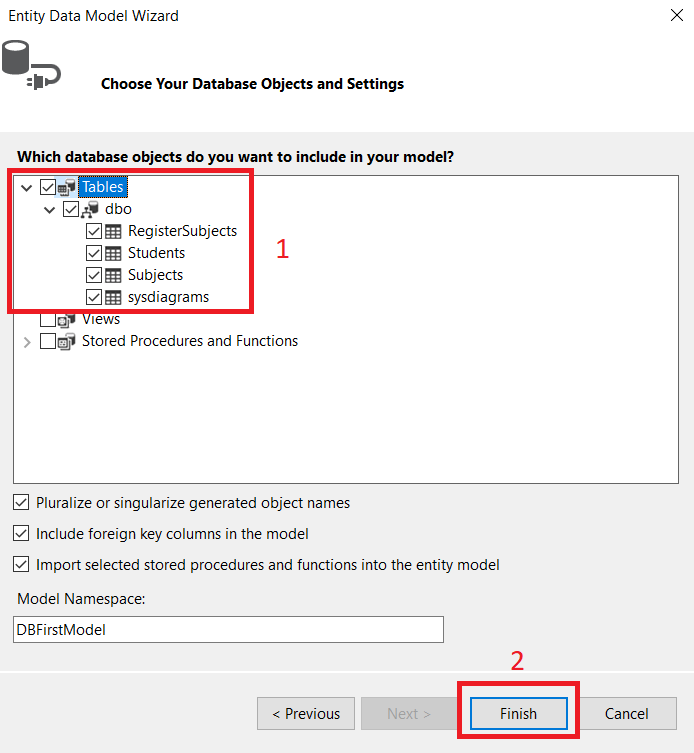








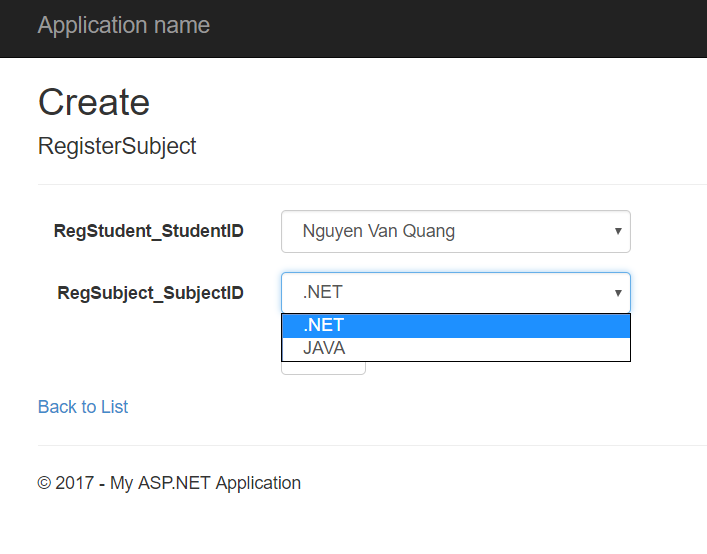
Chọn table cần sử dụng.



Build project.

Các bước thêm controller tương tự Code First.

Kết quả



# CHƯƠNG 5. TỔNG KẾT

## Môi trường phát triển và môi trường triển khai

* Các ví dụ chạy trên web browser.
* Yêu cầu .NET Framework 4.5, Microsoft SQL 2012, Visual Studio 2013.
* Sử dụng bộ thư viện giao diện Bootstrap.

## Kết quả đạt được

Sau thời gian thực hiện đề tài, các ví dụ đã hoàn thành và đạt được một số kết quả sau:

* + Biết cách xây dựng website sử dụng mô hình MVC và Entity Framework bằng cả 3 cách: code  first, model first, database first.
  + Các bước thực hiện dễ dàng, có hướng dẫn cụ thể, chi tiết.
  + Hiểu hơn về các kiến thức của MVC, EntityFramework.

## Hạn chế

Bên cạnh một số kết quả đạt được, vẫn còn một số hạn chế trong quá trình thực hiện:

* Một số lỗi trong quá trình sử dụng Entity Framework đối với các phiên bản khác nhau, tốn nhiều thời gian trong việc tìm hiểu, sửa lỗi.
* Ví dụ còn đơn giản, chưa thể hiện được vai trò của Entity Framework đối với các dự án lớn, bài tập lớn.
* Nội dung tìm hiểu vẫn chưa sâu, thiên về cách vận dụng để xây dựng chương trình.