**Bài giới thiệu: Cơ bản về Entity Framework**

10-02-2014 11:51:32 by 1

Trong năm nay 2014, tôi sẽ tập trung vào viết các bài viết chuyên về ASP.NET MVC, ASP.NET WEBFORM và đặc biệt là các buổi training miễn phí cho các bạn sinh viên đam mê lập trình cũng như đang có ý định trở thành 1 lập trình viên.

Bài viết đầu tiên của năm 2014 nói về Entity Framework – một nền tảng cơ bản cho việc phát triển các ứng dụng hiện nay. Trước khi bắt đầu bài viết, chúng ta sẽ tìm hiểu sơ lược về lịch sử của Entity Framework nhé.

|  |  |
| --- | --- |
| **Phiên bản** | **Tính năng** |
| EF 3.5 | Hỗ trợ O/RM cơ bản cho việc tiếp cận với Database |
| EF 4.0 | Hỗ trợ POCO, Lazy, cải thiện khả năng kiểm tra, tuỳ biến mã và tiếp cận với Model First |
| EF 4.1 | Gói NuGet, DBContext API, mã Code First và những bảng vá lỗi của EF 4.1.1 |
| EF 4.3 | Code First cho phép tạo Database bởi những dòng code cơ bản. EF 4.3.1 được phát hành với những bản vá lỗi EF 4.3.2n |
| EF 5.0 – hiện tại | Cung cấp EF dạng mã nguồn mỡ (open source). Hỗ trợ Enum, table-valued function, kiểu dữ liệu không gian spatial data types, multiple-diagrams der model, …. |
| EF 6.0 – sắp phát hành | EF 6.0 đang chuẩn bị phát hành. Chức năng sẽ gồm Task-based async, Store Procedure, Function trong Code First… |

Các bạn có thể [vào đây](http://msdn.microsoft.com/en-us/data/jj574253.aspx) để tìm hiểu về lịch sử Entity Framework hoặc tại [Blog](http://truongminhtuan.info/Article/Category?categoryId=51)

Các bài viết dưới đây, tôi đều dùng EF 4.3.1. Chúng ta sẽ hiểu ObjectContext và DBContext API

Về EF 5.0 tôi sẽ trình bày tiếp sau khi EF 4.3.1 được trình bày xong.

***Yêu cầu để trải nghiệm được EF 4.3.1 này:***

-          Có kiến thức về .NET Framework

-          C#

-          Visual Studio 2008/2010/2012

-          MS SQL Server 2008

**Bài 01: Entity Framework là gì?**

10-02-2014 12:26:46 by 1

ORM là một công cụ để lưu trữ dữ liệu từ các đối tượng cơ sở dữ liệu quan hệ miền như MS SQL Server. O/RM gồm 3 phần chính: các đối tượng miền dữ liệu – domain class object, các đối tượng cơ sở dữ liệu quan hệ - Relational database object và bản đồ thông tin về các miền đối tượng trong đối tượng cơ sở dữ liệu quan hệ (Table, View, Store Procedures)

***Chúng ta hãy xem định nghĩa Entity Framework của Microsoft***

Microsoft ADO.NET Entity Framework là một khuôn khổ đối tượng Object/ Bảng đồ quan hệ Relational Mapping (ORM) cho phép các nhà phát triển dể dàng làm việc với dữ liệu quan hệ như là các đối tượng domain-specific, loại bỏ đi sự khó khăn trong việc truy cập dữ liệu trước đây. Bằng cách sử dụng Entity Framework, truy vấn LINQ, thì việc lấy và thao tác dữ liệu như các đối tượng trở nên mạnh mẽ hơn. Entity Framework ORM cung cấp các dịch vụ như change tracking, idenity resolution, lazay loading, và truy vấn dữ liệu tập trung vào business logic của ứng dụng.

Đơn giản có thể định nghĩa Entity Framework như sau: Entity Framework là một khung Object/Relational Mapping (O/RM). Nó là một kỹ thuật mới cung cấp cho các nhà phát triển ADO.NET một cơ chế tự động để truy cập và lưu trữ dữ liệu trong cơ sở dữ liệu và làm việc với các kết quả ngoài DataReader và DataSet.

***Vấn đề tiếp theo là O/RM là gì và tại sao chúng ta lại cần nó?***

ORM là một công cụ để lưu trữ dữ liệu từ các đối tượng cơ sở dữ liệu quan hệ miền như MS SQL Server. O/RM gồm 3 phần chính: các đối tượng miền dữ liệu – domain class object, các đối tượng cơ sở dữ liệu quan hệ - Relational database object và bản đồ thông tin về các miền đối tượng trong đối tượng cơ sở dữ liệu quan hệ (Table, View, Store Procedures). ORM cho phép chúng ta giữ việc thiết kế tách biệt với thiết kế domain class. Điều này giúp ứng dụng được duy trì và mở rộng. Nó cũng tự động hoá các hoạt động CRUD như Create, Read, Update, Delete nhằm giúp nhà phát triển không phải viết nó bằng tay.

Có thể xem chi tiết tại <http://en.wikipedia.org/wiki/Object-relational_mapping>

Hiện nay có rất nhiều ORM framework như DataObject .Net, Nhibernate, OpenAccess, SubSonic…

**Bài 02: Entity Framework và LINQ-to-SQL (L2S)**

10-02-2014 13:13:48 by 1

Chúng ta có thể nghĩ rằng sự khác biệt giữa EF và L2S không có điều gì tương tự nhau. Dưới đây tôi đưa ra một số khác biệt giữa EF và L2S như sau:

- EF cung cấp một mô hình dịch vụ đầy đủ. Nó không chỉ hỗ trợ SQL Server mà còn các database khác nhau như Oracle, DB2, MySQL,….

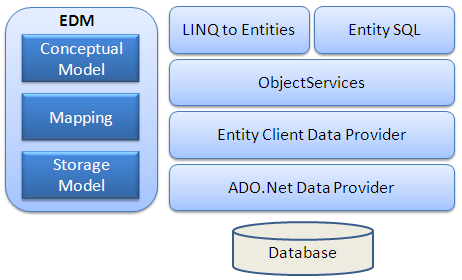
- Lớp L2S phải là one-to-one với cơ sở dữ liệu. Ví dụ lớp Nhân sự chỉ ánh xạ chỉ bảng Nhân sự, lớp Học sinh chỉ ánh xạ chỉ bảng Học sinh. Trong EF chúng ta có thể map (tạo) lớp miền (domain class) với nhiều bảng sử dụng theo cấu trúc database trong cơ sở dữ liệu quan hệ (phân cấp lớp, thừa kế)

- EF có nhiều cải tiến kỹ thuật để sử dụng trong EF 4.1 như Code First, Model First, Database First.

*Đây cũng là chiến lược dài hạn của Microsoft để tích hợp EF với nhiều sản phẩm khác của Microsoft.*

**Bài 03: Kiến trúc Entity Framework**

14-02-2014 11:26:44 by 1



Sơ đồ kiến trúc tổng thể của Entity Framework

**EDM** (Entity Data Model): bao gồm ba phần chính Conceptual model, Mapping and Storage model.

**Conceptual Model** - (khái niệm mô hình): là các lớp và mối quan hệ tươngứng với cơ sở dữ liệu. Điều này sẽ được độc lập từ cơ sở dữ liệu.

**Storage Model** - (mô hình lưu trữ): mô hình thiết kế cơ sở dữ liệu bao gồm Table, View, Store procedure, Relationship, Key,…

**Mapping** - (bảng đồ): gồm thông tin các khái niệm mô hình do developer ánh xạ tới mô hình lưu trữ hay cơ sở dữ liệu.

**LINQ to Entities**: là ngôn ngữ truy vấn được sử dụng để truy vấn với mô hình đối tượng Object model. Giá trị trả về tuỳ thuộc theo developer, theo Model.

**Entity SQL**: là ngôn ngữ truy vấn giống như LINQ to Entities. Nhưng nó hơi phức tạp hơn.

**Object Service**: phục vụ cho việc truy cập, trả gía trị dữ liệu từ cơ sở dữ liệu. Object Service cung cấp đầy đủ dịch vụ để quá trình chuyển đổi dữ liệu từ thực thể đến cấu trúc đối tượng dễ dàng hơn.

**Entity Client Data Provider**: giúp chuyển đổi L2E hoặc truy vấn Entity SQL vào truy vấn SQL trong cơ sở dữ liệu. Nó sẽ giao tiếp với ADO.NET data provider hoặc lấy dữ liệu từ cơ sở dữ liệu.

**ADO.Net Data Provider**: lớp này giao tiếp với cơ sở dữ liệu sử dụng theo chuẩn ADO.NET.

**Bài 04: Thiết lập môi trường làm việc EF 4.3.1**

14-02-2014 14:39:07 by 1

Các bước cài đặt gói EF4.3.1 và một số trở ngại khi cài đặt từ Visual Studio 2010

Để hoàn thành tốt bài viết Entity Framework, chúng ta cần cài đặt các gói sau

- .NET Framework 4.0/Visual Studio 2010 hoặc cao hơn

- ADO.Net Entity Framework 4.3 (you can install it using NuGet)

- SQL Server Express hoặc SQL Server 2005 hoặc cao hơn

Cài đặt **EntityFramework 4.3.1**, có **kết nối với Internet**:

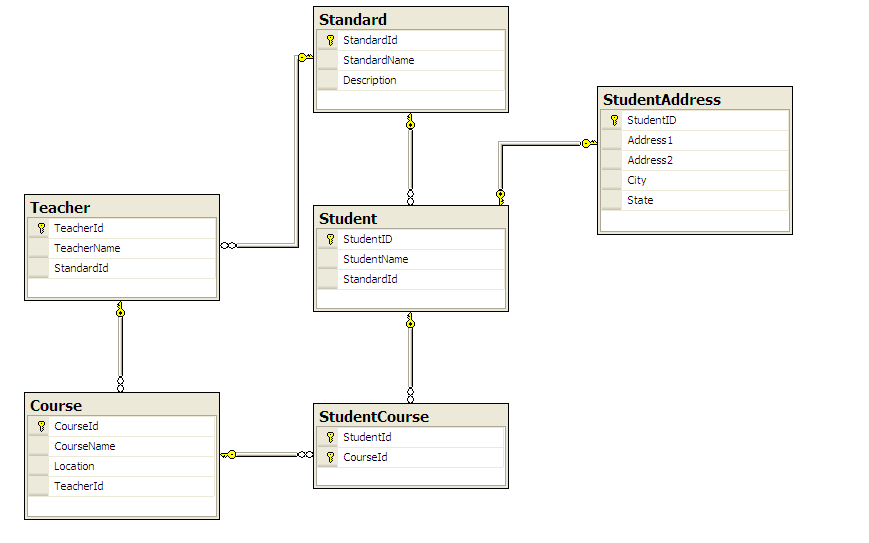
Mở **Visual Studio 2010 -> Tools -> Library Package Manager -> Package Manager Console**, gõ vào dòng lệnh

**PM> Install-Package EntityFramework -Version 4.3.1**

Nếu mở Visual Studio 2010 không có được Package Manager Console, thực hiện

Mở Visual Studio -> Tools -> Extension Manager -> Gõ NuGet Package Manager vào textbox để download từ NuGet, sau đó lập lại việc cài đặt EF 4.3.1

Tạo Database **SchoolDB** trong SQL Server, theo hình bên dưới. Các bạn cũng có thể tạo một Database khác để làm việc với loạt bài EF này.



*Việc hiểu được Database ở trên tôi không đề cập đến, các bạn có thể comment bên dưới để được trao đổi nhiều hơn.*

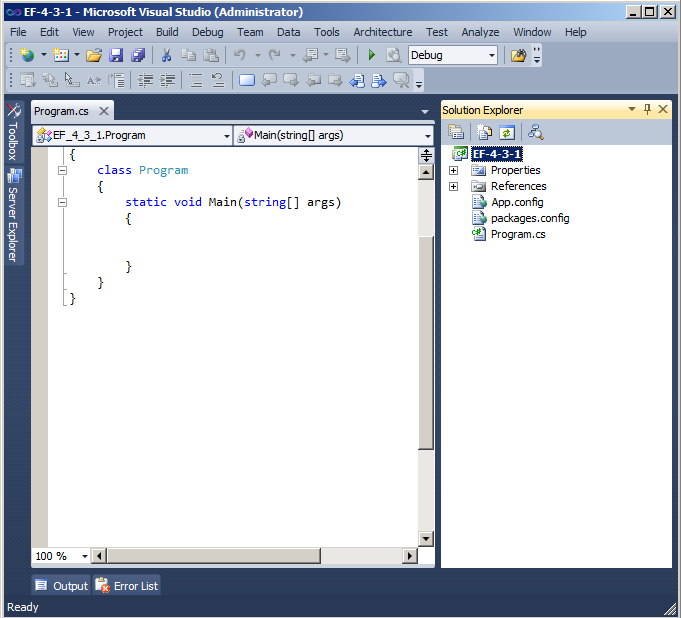
**Bài 05: Cách tạo một Entity Data Model (EDM)**

19-02-2014 11:42:03 by 1

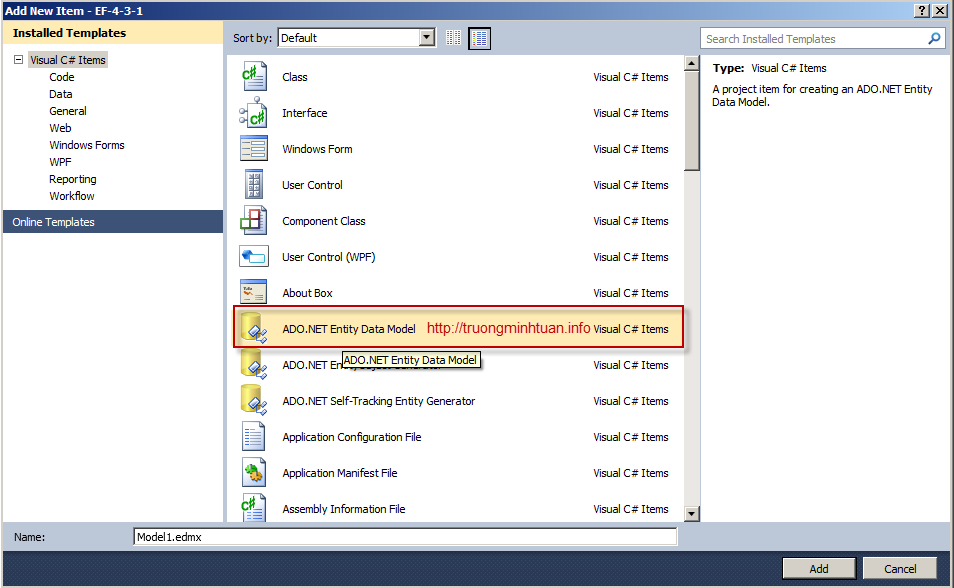
Các bước để thực hiện EDM và hiểu về cấu trúc EDM được tạo ra trong Visual Studio tương ứng với SQL Server

Tiếp theo, tôi hướng dẫn cách tạo một ADO.NET Entity Data Model cho cơ sở dữ liệu School DB.

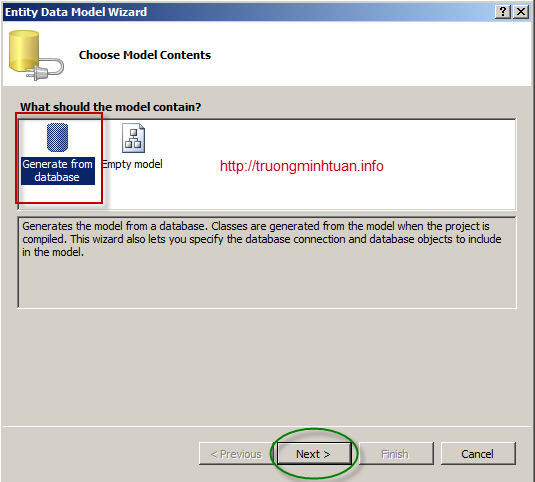
Bước 01: Mở Visual Studio 2010 và tạo một Project



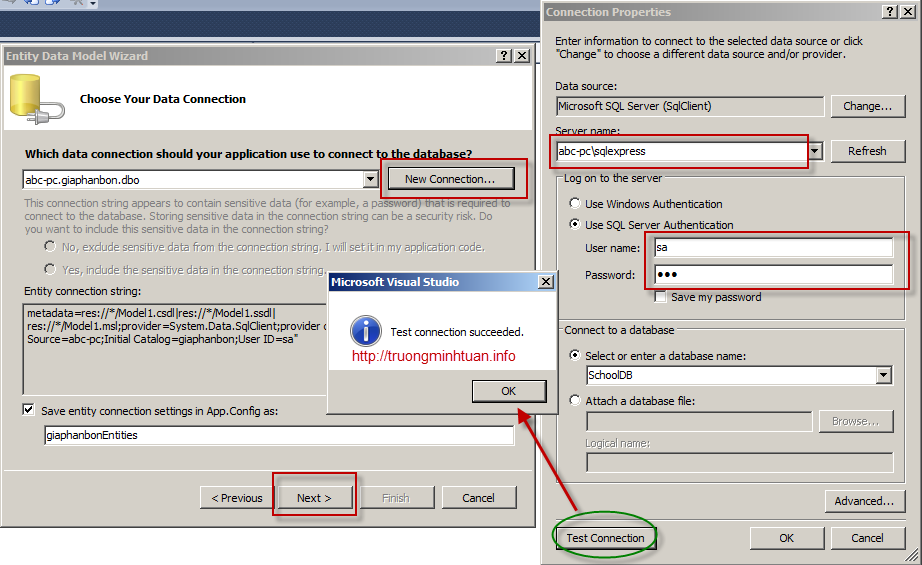
Bước 02: Click chuột phải vào Project EF-4-3-1, chọn Add -> New Item, chọn ADO.NET Entity Data Model



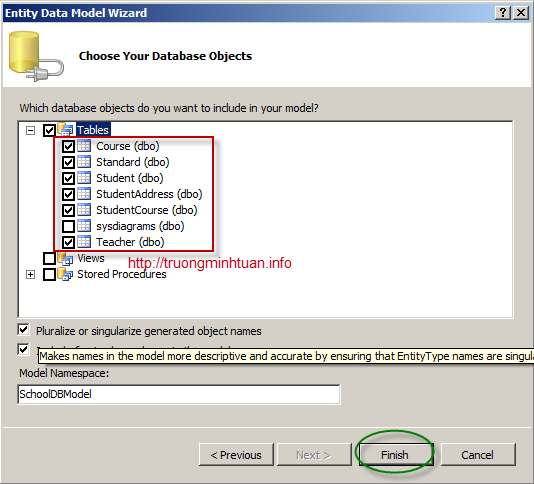
Bước 03: Hộp thoại xuất hiện, chọn Generate form database



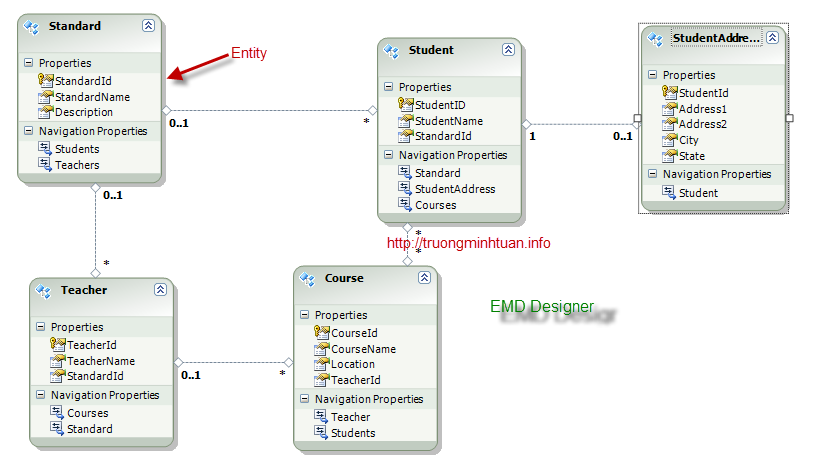
Bước 04: Hộp thoại Entity Data Model Wizard xuất hiện và chọn kết nối đến cơ sở dữ liệu SQL Server bằng cách nhấn New Connection hoặc chọn chuỗi kết nối đã tạo rùi ở ComboBox



Bước 05: Bước tiếp theo sẽ xuất hiện tất cả các Table, Views, Store procedures và chọn những Table theo hình bên dưới (Table này tương ứng với các Table trong database SchoolDB)



Bước 06: Nhấn Finish và chờ khoảng 10 giây để Visual General các Table vào trong Model.



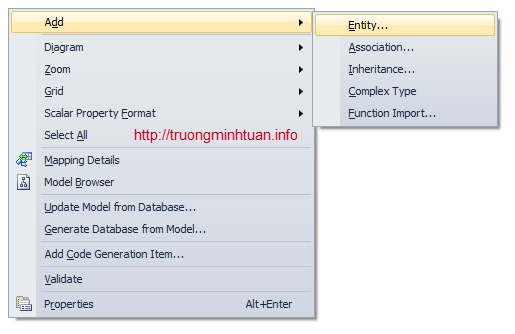
Sau đây chúng ta sẽ xem tất cả các blocks đã xây dựng nên EDM (Model1.edmx)

**EDM Designer:**

- Thiết kế EDM đại diện cho mô hình của chúng ta. Nó bao gồm Entities, associations và multiplicity giữa các thực thể.

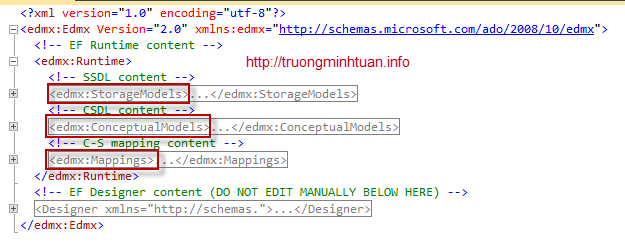
- Ban đầu nó giống như cấu trúc bảng (table) trong cơ sở dữ liệu, nhưng chúng ta có thể thêm hoặc trộn cột (merge column) hoặc xoá cột (remove column) mà không đòi hỏi phải từ ứng dụng của bạn thông qua thiết kế này. Thậm chí chúng ta có thể thêm đối tượng mới có các cột trong mô hình này.

Lưu ý, việc thay đổi sẽ ánh xạ với cơ sở dữ liệu, nên hết sức cẩn thận khi thay đổi trong thiết kế.



Chúng ta hãy mở EDM designer trong XML để xem 3 phần của giản đồ EDM: Conceptual schema (CSDL), Storage schema (SSDL) and mapping schema (MSL).

Tiếp theo để xem EDM XML, nhấn chuột phải vào Model1.edmx -> Open With -> xuất hiệp popup, chọn XML (text) Editor, chọn OK hoặc Yes để xuất hiện tập tin XML



Chúng ta có thể thấy nội dung của SSDL, nội dung CSDL và nội dung của bản đồ C-S.

**Entity Container**: là một wrapper cho EntitySets và AssociationSets. Đây là điểm khởi đầu quan trọng cho việc truy vấn của model

**EntitySet**: là một container cho EntityType. Nó được thiết lập cùng với EntityType. Chúng ta có thể hiểu nó như Table của database

**EntityType**: là một kiểu (type) của model. Mỗi EntityType tương ứng mô hình khái niệm trong XML (Key, Property)

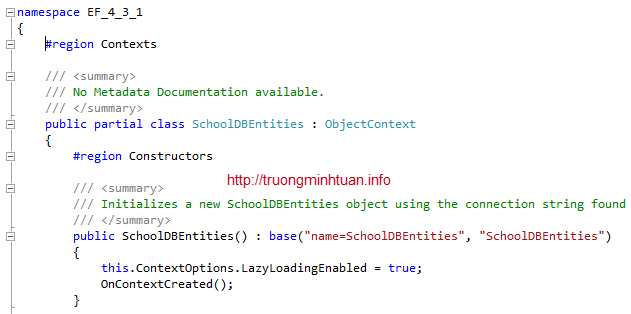
**AssociationSet**: xác định mối quan hệ giữa mỗi EntitySet

**ObjectContext**:

- Có Context và Entities. Bạn có thể thấy trong Class có hậu tố “Entities” và kế thừa từ lớp ObjectContext

- Giúp quá trình tương tác với các đối tượng thực thể và cơ sở dữ liệu

- Đối tượng ObjectStateManager theo dõi đối tượng trong quá trình các hoạt động create, update, delete



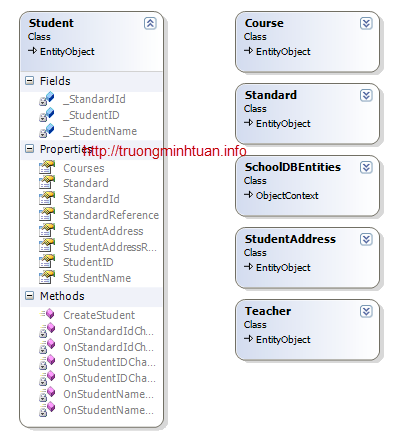
**ObjectSet**: Mỗi EntitySet trong lớp context là một loại ObjectSet<>.

**IObjectSet**: là một interface cho biết tập hợp collection như các chức năng. ObjectSet<> thực hiện IObjectSet.

**IObjectSet**: kiểm tra đơn vị (unit testing) nơi có thể tạo ra EntitySet kiểu IObjectSet<>.

Một điểm quan trọng nếu sử dụng IObjectSet sau đó chúng ta không nhận được một số chức năng của ObjectSet như include, except, method. Tuy nhiên chúng ta phải viết các extension method cho include, except method.

**EntityTypes**: click chuột phải vào Model1.Designer.cs chọn View Class Diagram sẽ xuất hiện rất nhiều partial class từ EntityObject. Các lớp này là EntityType của model.



**Bài 06: Truy vấn LINQ với Entity Data Model**

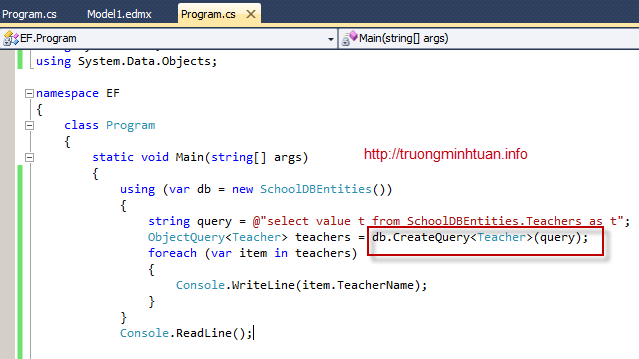
09-04-2014 10:58:20 by 1

Với EDM thì việc truy vấn (query) trở nên dễ dàng hơn. Hôm nay chúng ta sẽ biết cách truy vấn EDM với LINQ to Entities, Entity SQL, Native SQL

**LINQ to Entities:** Với L2E người phát triển truy vấn trực tiếp vào Entity và gọi Record trong Database một cách dễ dàng. Ở đây chúng ta có thể dùng từ khoá var, IEnumerable<T> hoặc IQueryable<T> để lấy dữ liệu thông qua cú pháp LINQ.



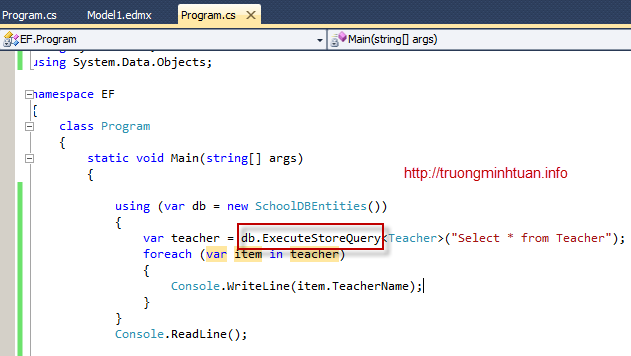
**Entity SQL:** Ngoài việc truy vấn dùng LINQ to Entities để truy vấn dữ liệu thì người phát triển có thể lựa chọn Entity Framework’s Object Services để truy vấn trực tiếp. Sự kết hợp giữa ObjectQuery<T> với Entity Framework’s T-SQL để xây dựng một biểu thức truy vấn .



Với ví dụ trên thì việc truy vấn dữ liệu phụ thuộc vào SchoolDBEntities() khi gọi trực tiếp đến một Object Teacher và kết quả trả về là một ObjectQuery<T> với cú pháp SQL để hiện thực bằng CreateQuery().

Người phát triển có thể dùng FirstOrDefault<T>() hoặc ToList() để gọi giá trị trả về.

**Native SQL:** Với Entity Framework thì trong lớp ObjectContext có thêm các phương thức ExecuteFunction(), ExecuteStoreQuery(), ExecuteStoreCommand().



Với EF thì một developer có thể tuỳ chọn một cách làm phù hợp trong lúc phát triển ứng dụng. Tuy nhiên thì đa phần developer đều dùng cách LINQ to Entities cho các ứng dụng sau này. Nó thay thế luôn cả một LINQ to SQL như trước đây.

**Bài 07: Các loại đối tượng trong Entity Framework**

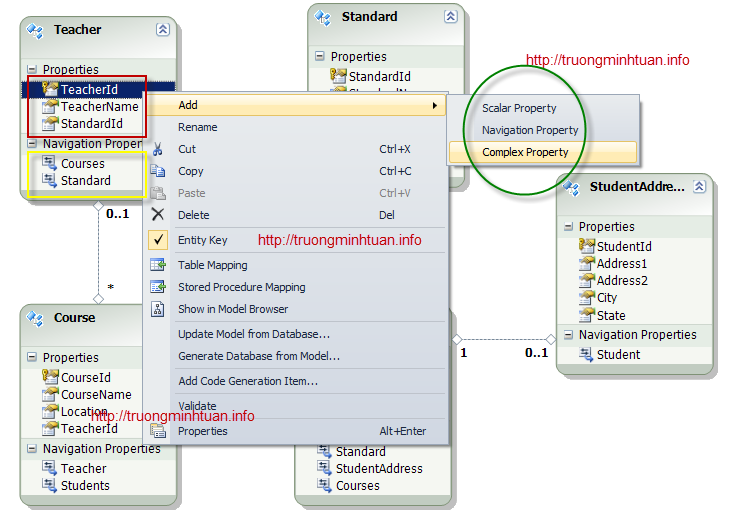
12-04-2014 14:31:44 by 1

Sự khác biệt giữa POCO và POCO Proxy nằm trong cơ chế tiết kiệm thay đổi. POCO sử dụng cơ chế Snapshot, vì thế phải sử dụng phương pháp DetechChanges của ObjectContext để đồng bộ hoá dữ liệu với ObjectStateManger. POCO Proxy sửg dụng thay đổi ngay lập tức. Nó đồng bộ với ObjectStateManager ngay sau khi POCO Proxy thay đổi, do đó POCO proxy có chức năng như EntityObject

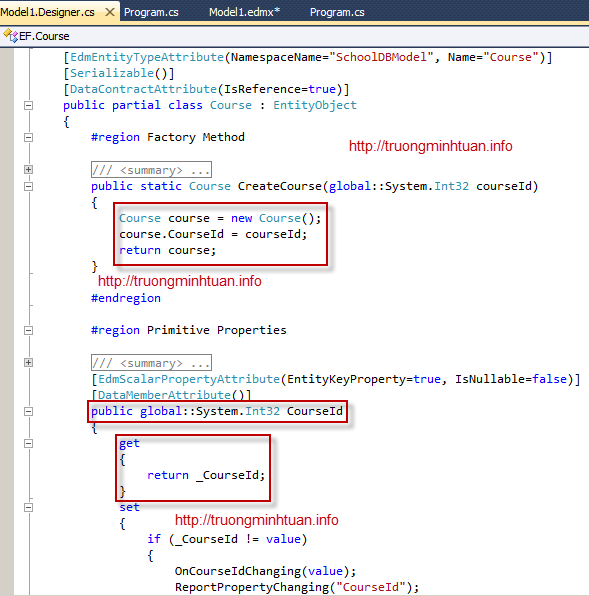
Trong EF có 4 loại: **EntityObject**, **POCO**, **POCO Proxy**, **Self-Tracking Entities**. Ứng với mỗi Entity đều có 2 loại của thuộc tính: Scalar properties, Navigation properties.

**Scalar properties** bao gồm các thuộc tính nằm trong một entity. Ứng với Entity Teacher có scalar properties là TeacherId, TeacherName, StandardId

**Navigation properties** là một con trỏ, trỏ đến một Entity khác. Hay nói cách khác nó hiện thực primary key – foreign key (1-n). Ứng với Entity Teacher có 2 navigation properties là Courses, Standard



**EntityObject:** mặc định ADO.NET Entity Data Model tạo một EntityObject từ database thông qua trình công cụ có sẵn của Visual Studio. Nó hiển thị đầy đủ cấu trúc của một Table (Fields Name, Data type,….)



**POCO (Plain Old CLR Object)**: đơn giản chỉ là một thực thể của database. Khi dùng Entity Framework nó sẽ tự động tạo ra các đối tượng ứng với database. POCO là một thực thể đơn giản. Nó cũng hỗ trợ cú pháp LINQ cho phép developer khi sử dụng và mang lại tất cả chức năng như dùng EntityObject.

public class Teacher

{

public int TeacherId { get; set; }

public string TeacherName { get; set; }

public Standard StandardId { get; set; }

public IList<Course> Courses { get; set; }

}

**POCO Proxy** là một lớp runtime proxy của POCO entity. POCO entity trở thành POCO proxy entity nếu đáp ứng được lazy loading proxy và change tracking.

POCO entity phải đáp ứng các yêu cầu sau để trở thành POCO proxy:

Một custom data phải được khai báo dạng public

Một custom data không phải là Abstract

Một custom data phải là public hoặc protected constructor

Lớp không thể hiện các giao diện IEntityWithChange Tracker hoặc IEntityWithRelationships vì các lớp proxy đã thực hiện

Các tuỳ chọn ProxyCreationEnabled phải được thiết lập đúng.

Mỗi navigation property phải được khai báo là public, virtual

Sự khác biệt giữa POCO và POCO Proxy nằm trong cơ chế tiết kiệm thay đổi. POCO sử dụng cơ chế Snapshot, vì thế phải sử dụng phương pháp DetechChanges của ObjectContext để đồng bộ hoá dữ liệu với ObjectStateManger. POCO Proxy sửg dụng thay đổi ngay lập tức. Nó đồng bộ với ObjectStateManager ngay sau khi POCO Proxy thay đổi, do đó POCO proxy có chức năng như EntityObject

public class Teacher

{

public virtual int TeacherId { get; set; }

public virtual string TeacherName { get; set; }

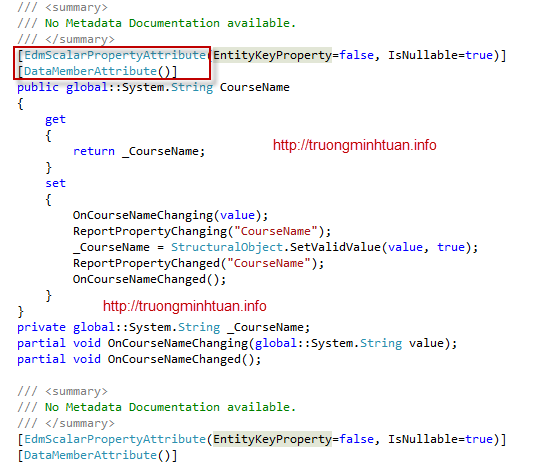
public virtual Standard StandardId { get; set; }

public virtual IList<Course> Courses { get; set; }

}

**Self-Tracking Entities:** là những EntityObject, POCO, POCO proxy entities làm việc tốt với ứng dụng. Tuy nhiên trong n-tier chúng ta phải chuyển các entities thành 1-tier nơi mà object context không có sẵn. Việc theo dõi những thay đổi trong object context đều do self-tracking entities đảm nhận. Với .NET Framework 4 thì self-tracking entities là name suggests, scalar, complex, navigation properties.

Ngoài ra thì Self-tracking entities có thêm chức năng tracking function và thực hiện IObjectWithChange Tracker và interface INotifyPropertyChanged. Nó cũng tự đánh dấu nó như DataContract khi sử dụng WCF services.



**Bài 08: Làm việc với EF dạng Database First**

21-04-2014 16:26:22 by 1

Thông thường thì một developer thường chọn Database First để triển khai ứng dụng, vì nó tương đối đơn giản và cách làm gần tương tự LINQ to SQL; Model First thì cũng tương đối khó hơn Database First, và Code First lại càng khó khăn hơn cho những bạn lập trình viên mới bắt đầu làm việc với ASP.NET

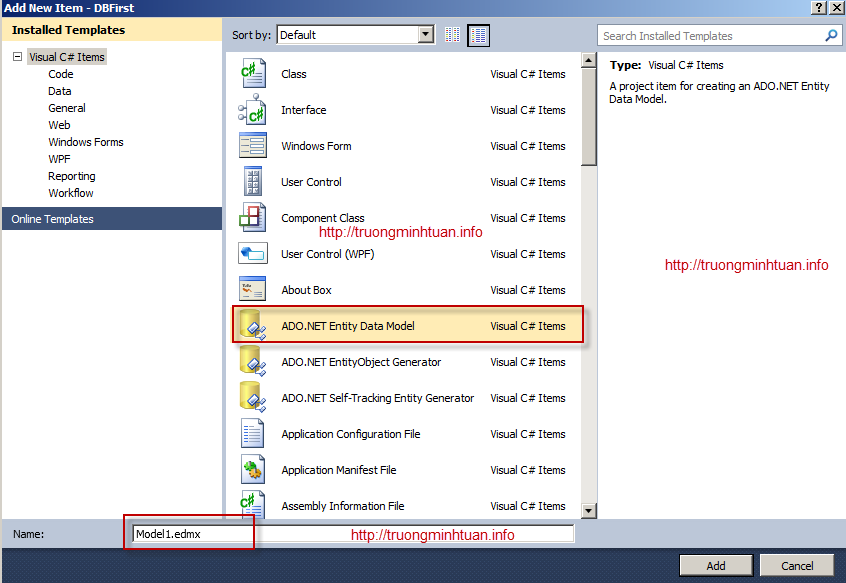
Với EF thì developer có thể chọn 1 trong 3 dạng để lập trình với Entity Data Model (EDM) đó là Database First, Model First, Code First.

Thông thường thì một developer thường chọn Database First để triển khai ứng dụng, vì nó tương đối đơn giản và cách làm gần tương tự LINQ to SQL; Model First thì cũng tương đối khó hơn Database First, và Code First lại càng khó khăn hơn cho những bạn lập trình viên mới bắt đầu làm việc với ASP.NET. Để hiểu rõ hơn, bạn tìm đọc 3 cách làm việc với database ở loạt bài nói về EF của Blog Trương Minh Tuấn.

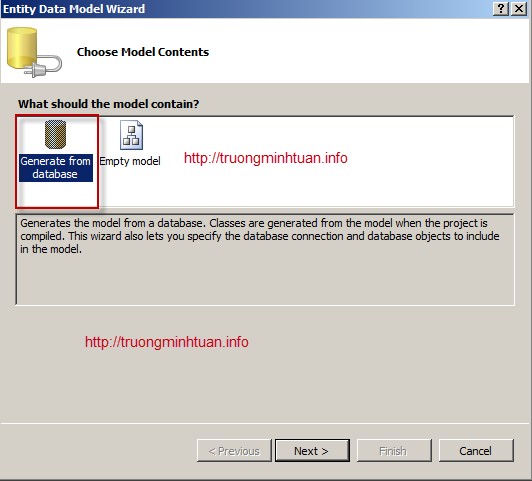
Đầu tiên, mình giới thiệu đến các bạn cách làm việc với database thông qua Entity Data Model dạng Database First,

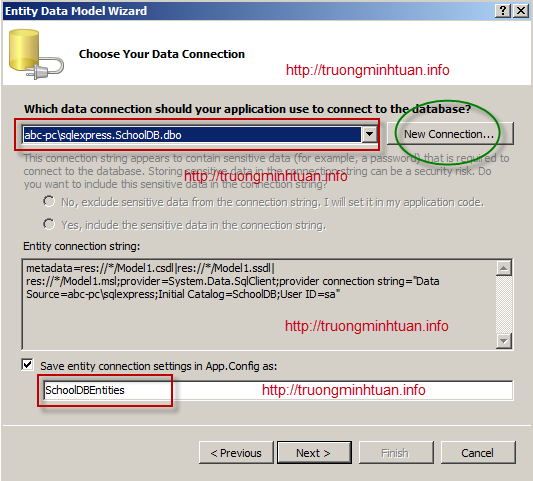
**Bước 01:** Bạn tạo một Project mới với tên **DBFirst**, dạng Console Application

**Bước 02:** Tạo một ADO.NET Entity Data Model bằng cách Right Project / Add / NewItem

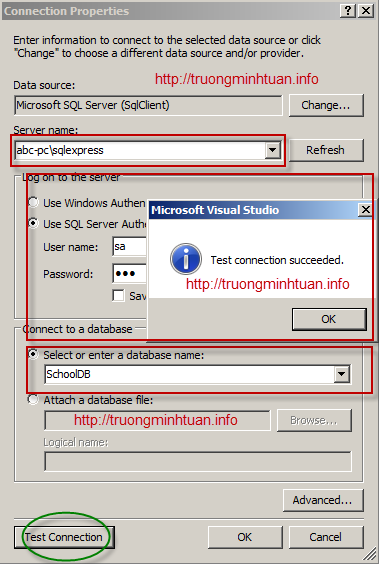


**Bước 03:** Xuất hiện hộp thoại, đây là nơi để chọn database từ phí SQL Server; ở đây bạn chọn chuỗi kết nối với Database hoặc tạo kết nối mới (New Connection)

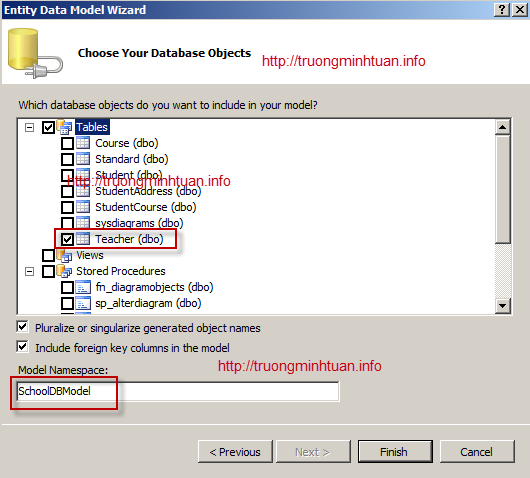




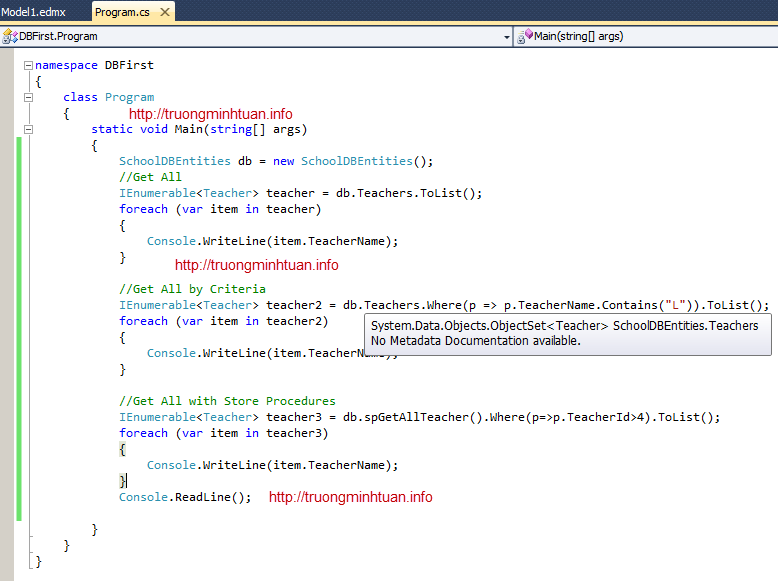
Nếu chọn New Connection thì hộp thoại Connection Properties xuất hiện, bạn chọn Server name, Log on to the server dạng Windows Authentication hoặc dạng SQL Server Authentication, tiếp theo chọn database name, sau đó nhấn nút Test Connection để kiểm tra chuỗi kết nối;



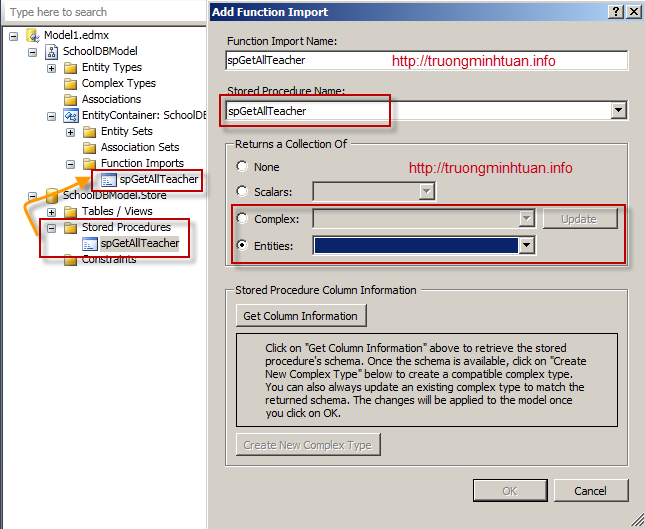
**Bước 04:** Đến bước này, việc connect đến với Cơ sở dữ liệu đã hoàn chỉnh, việc còn lại bạn tạo chọn Tables, Views, Stored Procedures



**Bước 05:** Tiếp theo chúng ta sẽ truy vấn dữ liệu EDM bằng cú pháp LINQ. Ngoài ra, bạn cũng làm việc tốt với Store Procedures cũng tương như như một Tables



**Lưu ý:** Khi bạn Update một Store Procedures cho EDM thì bạn cần Add Function Import cho ứng dụng; ở đây bạn sẽ có 2 kiểu để chọn: Entities, Complex Types (kiểu vô danh).



Chúc các bạn học tập tốt