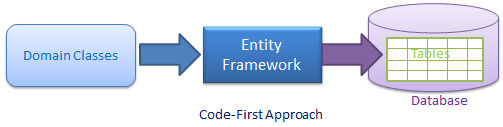
* Add tham chiếu ref Framwword
* Tạo Entity : để Public cho các class 🡪 add usingSystem.ComponentModel.DataAnnotations;
* Add connection string
* Tạo BLL,DAL,Giao diện
* Tạo class context (viết lớp DB Init)

<http://www.entityframeworktutorial.net/code-first/what-is-code-first.aspx>

# 1 / Code First là gì ?

Entity Framework đã giới thiệu cách tiếp cận Code-First với Entity Framework Code-First chủ yếu hữu ích trong [Thiết kế hướng tên miền](http://msdn.microsoft.com/en-us/magazine/dd419654.aspx) . Trong cách tiếp cận Code-First, bạn tập trung vào miền của ứng dụng của bạn và bắt đầu tạo các lớp cho thực thể miền của bạn thay vì thiết kế cơ sở dữ liệu của bạn trước rồi tạo các lớp khớp với thiết kế cơ sở dữ liệu của bạn. Hình dưới đây minh họa cách tiếp cận mã đầu tiên.



## Quy trình làm việc code first[quy trình phát triển mã đầu tiên](http://www.entityframeworktutorial.net/images/codefirst/dev-workflow.png)

Quy trình phát triển theo cách tiếp cận mã đầu tiên sẽ là: Tạo hoặc sửa đổi các lớp miền -> định cấu hình các lớp miền này bằng cách sử dụng Fluent-API hoặc các thuộc tính chú thích dữ liệu -> Tạo hoặc cập nhật lược đồ cơ sở dữ liệu bằng cách di chuyển tự động hoặc di chuyển dựa trên mã.

## \*DB Context

Lớp bối cảnh trong Entity Framework là một lớp xuất phát từ cả DbContextEF 6 và EF Core. Đây là một lớp quan trọng trong Entity Framework, đại diện cho một phiên với cơ sở dữ liệu cơ bản.

Trong ví dụ trên, SchoolContextlớp có nguồn gốc từ DbContextđó, làm cho nó trở thành một lớp bối cảnh. Nó cũng bao gồm một bộ thực thể cho Student, StudentAddressvà Gradecác thực thể.

Lớp bối cảnh được sử dụng để truy vấn hoặc lưu dữ liệu vào cơ sở dữ liệu. Nó cũng được sử dụng để cấu hình các lớp miền, ánh xạ liên quan đến cơ sở dữ liệu, thay đổi cài đặt theo dõi, bộ đệm, giao dịch, v.v.

Cách tiếp cận Code-First cũng yêu cầu một [context class](http://www.entityframeworktutorial.net/basics/context-class-in-entity-framework.aspx) nên được bắt nguồn từ lớp DbContext . Nó xuất phát từ DBContext class và hiển thị DbSet các thuộc tính cho các loại mà bạn muốn là một phần của mô hình, ví dụ Studentvà Gradecác lớp trong trường hợp này. Đây DbSetlà một tập hợp các lớp thực thể (còn gọi là tập thực thể), vì vậy chúng tôi đã đặt tên thuộc tính là số nhiều của tên thực thể như Studentsvà Grades.

**Nhưng cơ sở dữ liệu ở đâu và các bảng và cột của chúng ở đâu?**

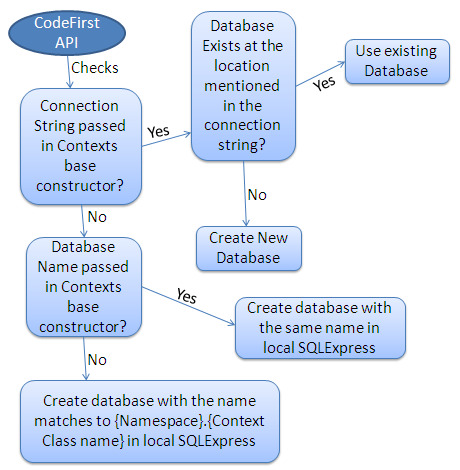
Do API Code-First của EF. Nó tạo ra cơ sở dữ liệu dựa trên tham số được truyền trong hàm tạo cơ sở của lớp ngữ cảnh của bạn. Vì chúng ta chưa truyền bất kỳ tham số nào trong hàm tạo của lớp ngữ cảnh, nên nó đã tạo cơ sở dữ liệu  EF6Console.SchoolContext  trong cơ sở dữ liệu SQLEXPRESS cục bộ

# Quy ước trong EF 6 Code-First

<http://www.entityframeworktutorial.net/code-first/code-first-conventions.aspx>

# 2.Database Initialization in Entity Framework 6

we will learn how EF decides the database name and server while initializing a database in code-first approach.



the base constructor of the context class can have the following parameter.

1. No Parameter
2. Database Name
3. Connection String Name

## 1.No Parameter

If you do not specify the parameter in the base constructor of the context class then it creates a database in your local SQLEXPRESS server with a name that matches your {Namespace}.{Context class name}. For example, EF will create a database named SchoolDataLayer.Context for the following context class:

namespace SchoolDataLayer

{

public class Context: DbContext

{

public Context(): base()

{

}

}

}

## 2. Database Name

namespace SchoolDataLayer

{

public class Context: DbContext

{

public Context(): base("MySchoolDB")

{

}

}

}

## 3 ConnectionString Name

namespace SchoolDataLayer

{

public class Context: DbContext

{

public SchoolDBContext() : base("name=SchoolDBConnectionString")

{

}

}

}

App.config:

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="SchoolDBConnectionString"

connectionString="Data Source=.;Initial Catalog=SchoolDB-ByConnectionString;Integrated Security=true"

providerName="System.Data.SqlClient"/>

</connectionStrings>

</configuration>

# 3.Database Initialization Strategies in EF 6 Code-First (chiến lược khởi tạo)

Bạn đã tạo cơ sở dữ liệu sau khi chạy ứng dụng Code-First lần đầu tiên, nhưng lần thứ hai trở đi thì sao? Nó sẽ tạo ra một cơ sở dữ liệu mới mỗi khi bạn chạy ứng dụng? Môi trường sản xuất thì sao? Làm thế nào để bạn thay đổi cơ sở dữ liệu khi bạn thay đổi mô hình miền của bạn? Để xử lý các kịch bản này, bạn phải sử dụng một trong các chiến lược khởi tạo cơ sở dữ liệu.

Có bốn chiến lược khởi tạo cơ sở dữ liệu khác nhau:

1. **CreateDatabase IfNotExists:** Đây là trình khởi tạo mặc định . Như tên cho thấy, nó sẽ tạo cơ sở dữ liệu nếu không tồn tại theo cấu hình. Tuy nhiên, nếu bạn thay đổi lớp mô hình và sau đó chạy ứng dụng với trình khởi tạo này, thì nó sẽ đưa ra một ngoại lệ.
2. **DropCreateDatabase IfModelChanges:**Trình khởi tạo này bỏ cơ sở dữ liệu hiện có và tạo cơ sở dữ liệu mới, nếu các lớp mô hình của bạn (các lớp thực thể) đã bị thay đổi. Vì vậy, bạn không phải lo lắng về việc duy trì lược đồ cơ sở dữ liệu của mình, khi các lớp mô hình của bạn thay đổi.
3. **DropCreateDatabaseAlways:** Như tên cho thấy, trình khởi tạo này sẽ loại bỏ cơ sở dữ liệu hiện có mỗi khi bạn chạy ứng dụng, bất kể các lớp mô hình của bạn có thay đổi hay không. Điều này sẽ hữu ích khi bạn muốn có một cơ sở dữ liệu mới mỗi khi bạn chạy ứng dụng, ví dụ như khi bạn đang phát triển ứng dụng.
4. **Custom DB Initializer:** Bạn cũng có thể tạo trình khởi tạo tùy chỉnh của riêng mình, nếu cách trên không thỏa mãn yêu cầu của bạn hoặc bạn muốn thực hiện một số quy trình khác khởi tạo cơ sở dữ liệu bằng trình khởi tạo ở trên.

# 4.Thắc mắc

Từ khóa vitural

<http://forum.tedu.com.vn/thread/thac-mac-ve-vitrual-va-nhung-quan-he-trong-entity-framework-code-first/>

<http://www.entityframeworktutorial.net/lazyloading-in-entity-framework.aspx>