|  |  |  |
| --- | --- | --- |
| Từ/Cụm từ viết tắt | Nội dung đầy đủ | Ghi chú khác |
| EF | Entity Framework |  |
| CSDL | Cơ sở dữ liệu |  |
| MVP | Model-View-Presenter |  |
| ORM | Object Relational Mapping |  |
| DB | Database |  |
| LINQ | Language Integrated Query |  |

I. ADO.NET Entity Framework

\*Definition:

-Là một framework giúp ánh xạ cơ sở dữ liệu quan hệ lên một khung nhìn các đối tượng và tập hợp các đối tượng.

-Là một DB ORM (Database Object-Relational Mapping) mã nguồn mở được chính Microsoft phát triển dựa trên nền tảng .NET.

-EF giúp người lập trình tiết kiệm được rất nhiều thời gian và công sức trong việc truy xuất và làm việc với các CSDL quan hệ, bởi vì bản thân EF đã cung cấp đầy đủ các tính năng và công nghệ tiên tiến giúp độc lập hóa khối CSDL ra khỏi lớp truy xuất.

\*Version: 6.1.1

\*Features:

- LINQ to Entities: EF sử dụng LINQ để truy vấn trên CSDL. LINQ là một công nghệ truy vấn CSDL không phụ thuộc vào dữ liệu nguồn, tức là có thể dùng LINQ để truy vấn và thực thi các thao tác dữ liệu trên: tập hợp/danh sách các đối tượng, tập tin XML hay các hệ quản trị CSDL khác nhau.

-LINQ IEnumberable và biểu thức chính quy Lambda

+LINQ IEnumberable: là một lớp giao diện (interface) định nghĩa trong LINQ, cho phép tạo ra các câu truy vấn lồng nhau trên một tập hợp có thực thi lớp giao diện này. Kết quả của một loạt các thực thi sẽ được trả về ngay sau khi có lời gọi để chuyển từ tập hợp không chính quy (non-generic) sang tập hợp chính quy (generic). Đây là tính năng rất hay trong LINQ, mà nhờ đó tiết kiệm được chi phí cũng như thời gian thực thi dữ liệu, vì câu truy vấn cuối cùng chỉ được kết lại và gọi chạy khi hoàn tất một loạt các truy vấn lồng nhau thay vì phải tải dữ liệu lên sau mỗi bước nạp điều kiện.

[Hình minh họa IEnumrable]

+Lambda là một dạng biểu thức chính quy được sử dụng trong các truy vấn đến các lớp dữ liệu có đăng ký thực thi lớp IEnumberable.

[Hình minh họa Lambda]

-Lazy loading: Trì hoãn load dữ liệu khi chưa cần thiết, dữ liệu chỉ được load khi có lời gọi tới lần đầu tiên (Giảm thời gian nạp dữ liệu ban đầu).

-Code first: là một cách tiếp cận mới trong các ứng dụng hướng CSDL, thay vì thiết kế cấu trúc CSDL trước thì Code first cung cấp các đặc tả để người lập trình có thể định nghĩa cấu trúc CSDL bằng các class và các logic, ràng buộc trên class, tận dụng được các tính năng kế thừa trên class, sau đó Entity Framework sẽ tự động ánh xạ xuống cấu trúc

CSDL.

+Sự dụng nhiều trong các ứng dụng dạng Domain-Driven Design, cho phép người lập trình ảo hóa CSDL vật lý, giảm sự phục thuộc vào CSDL vật lý (các vấn đề về tính tương thích trên các phiên bản Hệ quản trị).

+Code first to new Database

+Migration: dịch chuyển phiên bản trên CSDL

+Database Version (Upgrade/Downgrade/Rebase)

+Data Seeding

+Model cheking (Structure changed): Kiểm tra cấu trúc CSDL có bị thay đổi bởi các tác nhân ngoài hệ thống, đảm bảo hệ thống làm việc ổn định và dúng đắn.

+Custom Database Initalier: Chỉ định phương thức khởi tạo CSDL (tự động tạo CSDL nếu chưa có, tự động tạo bảng nếu chưa có, tự động tạo dữ liệu mẫu ban đầu,…)

a. CreateDatabaseIfNotExists: mặc định của EF. Tự động tạo CSDL nếu chưa có.

b. DropCreateDatabaseIfModelChanges: Tự động tạo lại CSDL khi cấu trúc bị thay đổi. Tuy nhiên không an toàn dữ liệu.

c. DropCreateDatabaseAlways: Luôn luôn xóa và tạo mới lại CSDL mỗi khi ứng dụng khởi chạy, phù hợp với các úng dụng sử dụng CSDL như là một Cache, và muốn CSDL rỗng cho mỗi phiên là việc.

d. Custom DB Initializer: Nếu các định nghĩa trên không đáp ứng được yêu cầu thì người lập trình có thể tự định nghĩa một trình khởi tạo dữ liệu riêng cho CSDL.

-BootStrapper (override some method for specific business process) vs Event Hooker: Cho phép đăng ký các sự kiện xảy ra trên Entity khi Entity Framework làm việc, sử dụng để thực hiện một vài hành động logic như: sau khi cập nhật thành công thì trường date\_modified sẽ có giá trị giờ hiện tại.

-Class proxy (override defined class): EF đòi hỏi thuộc tính khóa ngoại của các class phải được khai báo dạng virtual để EF override các kỹ thuật như Lazy loading lúc truy xuất các thuộc tính này.

-Transaction Manager (Rollback/commit): Đảm bảo tính đơn nguyên của giao dịch, hoặc là tất cả thao tác dữ liệu đều được thực thi hoặc là không. Nhằm đảm bảo tính nhất quán trên CSDL. EF cung cấp một cơ chế Transaction rất đơn giản và hiệu quả.

-Entity State tracking and “Filter Update”: Để có thể theo dõi được các sự thay đổi dữ liệu trên các Entity trong môi trường runtime, EF đưa ra định nghĩa về trạng thái của các Entity, trong đó một Entity có thể thuộc một trong các trạng thái sau:

a. Attached (Entity mới khoải tạo và được đưa và hệ thống tracking, tuy nhiên chưa được lưu xuống CSDL)

b. Detached (Entity đã bị loại khi hệ thống tracking)

c. Added (Entity được đưa vào hàng đợi chờ thêm vào CSDL)

d. Modified (Enity được đánh dấu là đã bị thay đổi ít nhất 1 thuộc tính được định nghĩa trong cấu trúc CSDL, được đưa vào hàng đợi chờ cập nhật xuống CSDL)

e. Unchange (Entity đượ đánh dấu là sạch, có thể là mới được khởi tạo hoặc là mới được load lên từ CSDL)

f. Deleted (Entity được đánh dấu là bị xóa, được đưa vào hàng đợi chờ xóa khởi CSDL)

-Entity framework kết hợp mô hình 3 lớp

-Manual media cached (class HinhAnh):

Với các ứng dụng hướng đa phương tiện thì việc tổ chứa lưu trữ và truy xuất các tài nguyên là quan trọng nhằm tiết kiệm bộ nhớ và thời gian truy xuất.

-Entity vs DataFilter:

+Entity là một thực thể mang các thuộc tính trực tiếp của bản thân nó, trong lập trình giao diện, thông thường khi hiển thị thông tin một đối tượng của một thực thể nào đó, ta thường hiển thị các thuộc tính gián tiếp (thuộc tính của khóa ngoại), do đó phần xử lý giao diện sẽ không làm việc trực tiếp với Entity mà làm việc thông qua một lớp mặt nạ gọi là DataFilter, nhiệm vụ của DataFilter là kết các Entity có liên quan lại với nhau sau đó chọn ra các thuộc tính cần hiển thị.

-Database Context vs Singleton Database Instance Provider:

+ EF xem Database Context là một ngữ cảnh truy xuất đến CSDL, trên đó chứa các lớp mặt nạ giúp truy xuất tài nguyên CSDL.

+Singleton giúp các lớp Entity nhìn thấy cùng một DbContext trong suốt phiên làm việc, bởi vì EF đòi hỏi các đối tượng sinh ra từ các class Entity phải thống nhất về DbContext, một đối tượng không thể được tracking bởi 2 DbContext khác nhau.

+Phiên làm việc được đánh dấu từ lúc DbContext được khởi tạo cho đến khi bị hủy bỏ (Dispose), các đối tượng có runtime nằm ngoài phiên làm việc được xem là không hợp lệ và không có ý nghĩa về mặt dữ liệu, muốn làm việc lại trên các đối tượng này nhất thiết phải load lại trong một phiên làm việc khác của DbContext.

-TPC (Table per Concreted class) pattern: tận dụng code logic bằng cách thiết kế mô hình kế thừa, đa hình. Code logic chung sẽ được đặt ở lớp trừu tượng.

+Trong TPC, các lớp ảo (abstract) sẽ không được ánh xạ xuống CSDL vật lý, các lớp này chỉ dùng cho mục đích kế thừa và đa hình. Muốn ánh xạ một Entity xuống CSDL vậy lý thì lớp Entity đó bắt buộc không được khai báo là một lớp ảo

-OOP:

+Interface: Định nghĩa các lớp giao diện phục vụ cho kế thừa và Entity Bootstrap

+Kế thừa:

+Đa hình (override) ở các lớp dẫn xuất để thay đổi hành vi ứng với các Rntity khác nhau.

\*References source:

<http://www.entityframeworktutorial.net/>

<http://msdn.microsoft.com/en-us/data/aa937723>

<https://github.com/tidyui/EFBootstrapper>

II. Sync Framework

\*Definition: Đồng bộ giữa các CSDL

\*Version: 2.1

\*Features:

-Đồng bộ dữ liệu giữa các CSDL

-Sử dụng kỹ thuật trigger trên từng Table, lưu thông tin về sự thay đổi trên CSDL trong các Table tracking mà Sync Framework tạo ra khi được cài đặt Scope

-Chỉ đồng bộ những dữ liệu mới dựa trên các tracking trên Scope đã có sẵn (tiết kiệm được thời gian và lượng dữ liệu phải xử lý)

-Sync Scope

+Define: là một định nghĩa về phiên đồng bộ trên một CSDL cụ thể, chứa tập hợp các thông tin về Table cần đồng bộ. Một CSDL có thể có nhiều Sync Scope.

+Set up scope to existed database: Chỉ định dãy tuyến tính các Table cần đồng bộ, sau đó gọi phương thức khởi tạo Scope, Sync Framework sẽ thực hiện nhiệm vụ còn lại.

+Remove scope from existed database: Chỉ định tên Scope muốn gõ bỏ, sau đó gọi phương thức gõ bỏ Scope, Sync Framework sẽ thực hiện nhiệm vụ còn lại.

+Fetching scope among databases: Đăng ký một Scope cho CSDL này (Scope ) từ CSDL khác (Scope provider)

+Sync Direction (Up/Down/Bidirectional link): chọn hướng đồng bộ, đáp ứng một số yêu cầu về bảo mật trên CSDL.

\*Challenge:

-Đụng độ dữ liệu:

+Đụng độ khóa chính: Xét 2 CSDL độc lập có cùng cấu trúc Table và dữ liệu, nếu khóa chính được thiết lập dạng Auto\_Increasement thì khi gọi Insert trên 2 CSDL, hệ quản trị CSDL địa phương ở cả 2 CSDL trên sẽ có khả năng tạo ra khóa chính trùng nhau (khả năng rất cao). Nên khi đồng bộ dữ liệu sẽ bị đụng độ khóa chính, 1 trong 2 dữ liệu mới Insert đó sẽ bị loại bỏ.

=> Sử dụng khóa chính Guid: Khóa chính Guid là khóa chính dạng “tự nhiên” (Natural), được tạo ra dựa trên các giá trị định danh (địa chỉ MAC của card mạng) và ngẫu nhiên (thời gian hiện tại trên Hệ quản trị CSDL), Hệ quản trị sẽ đảm bảo mỗi Guid được cấp phát sẽ là duy nhất trên toàn cầu (mặc dù khả năng trùng là có thể xảy ra trên lý thuyết nhưng có thể chấp nhận được).

- Database Schema “dbo” cho User (SQL Server): nếu sai khác giữa các Table sẽ gây lỗi do thiếu quyền chạy trigger trên các Table tracking.

=>Trước khi sử dụng User đó để sử dụng cho các lệnh thiết lập đồng bộ, nhất thiết phải chỉ định Default Database Schema là “dbo”.

-Ràng buộc thứ tự tiến trình trên Table: thứ tự các Table khi đồng bộ là rất quan trọng, vì nếu Table B có chứa khóa ngoại tham chiếu đến Table A thì Table A phải được đồng bộ trước Table B

=> Xây dựng thứ tự tiến trình trong đó luôn đảm bảo Table dạng B luôn nằm sau Table dạng A

\*References source:

<http://msdn.microsoft.com/en-us/library/bb902854(v=sql.110).aspx>

III. DevExpress

\*Definition:

\*Version:

\*Features:

-MVP (Model-View-Presenter):

+Tận dụng các giao diện sử dụng chung

+Tận dụng các logic về giao diện chung

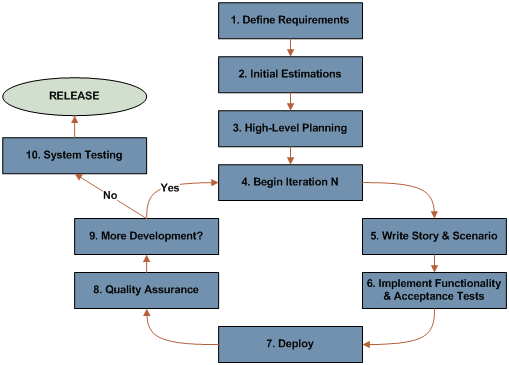
+Tạo sự thống nhất trong giao diện giữa các project GUI

-Custom User control:

+Coi như là một module MVP nhỏ sử dụng chung cho nhiều khu vục

IV. Mô hình phát triển phần mềm

-Nhóm áp dụng mô hình Agile trong toàn bộ quy trình phát triển phần mềm



[http://www.varsys.com/knowledgecenter\_WaterfallDownfall.html]