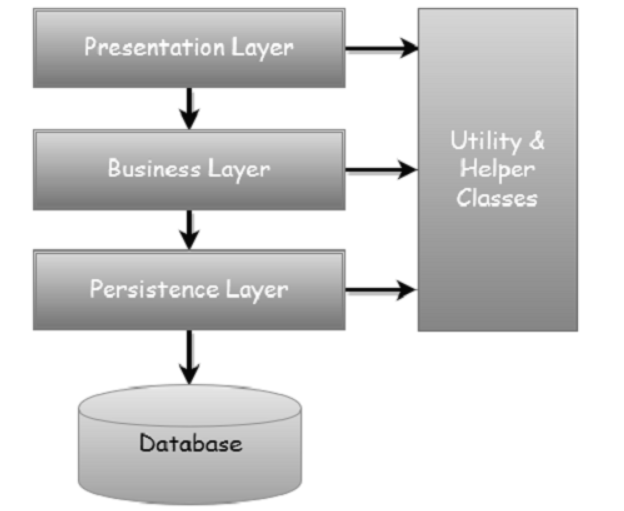
BÁO CÁO TASK 5a: Nhibernate Framework

Nhibernate là một trình ánh xạ object-relational mã nguồn mở. Giúp việc thao tác với dữ liệu trong database dễ dàng hơn.

# Layered Architecture

Cấu trúc ứng dụng cấp cao thường có 3 lớp:

* The Presentation layer
* The Business layer
* The Persistence layer



## Presentation layer

Đây là lớp trên cùng, chứa mã chịu trách nhiệm vẽ Giao diện người dùng, các trang, hộp thoại hoặc màn hình, thu thập thông tin đầu vào của người dùng và kiểm soát điều hướng.

## Business layer

* Business layer có nhiệm vụ thực hiện bất kỳ quy tắc nghiệp vụ hoặc yêu cầu hệ thống.
* Nó cũng sử dụng lại mô hình được xác định bởi Persistence layer.

## Persistence layer

* Persistence layer bao gồm các class và component chịu trách nhiệm lưu trữ và truy xuất dữ liệu ứng dụng.
* Layer này cũng xác định ánh xạ giữa model class và database. Nhibernate được sử dụng chủ yếu trong layer này.

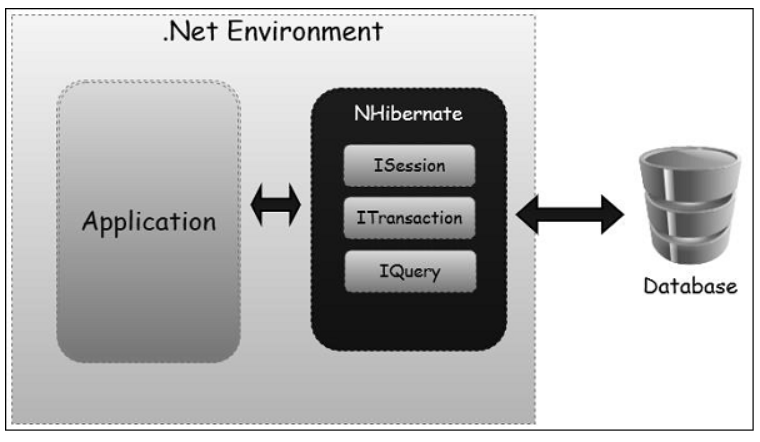
## Database

## Helper/Utility Classes

Mọi ứng dụng đều có một tập hợp các lớp trợ giúp hoặc tiện ích hỗ trợ các lớp khác.

ví dụ: UI widgets, messaging classes, Exception classes, and logging utilities.

# NHibernate - Architecture

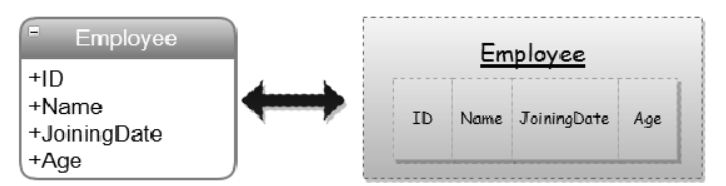


Code ứng dụng sử dụng API NHibernate ISession và IQuery cho các hoạt động persistence và chỉ phải quản lý các database transactions, ý tưởng là sử dụng API NHibernate Itransaction.

# ORM – Object-Relational Mapping

Là kỹ thuật lập trình để chuyển đổi dữ liệu giữa các hệ thống có kiểu không tương thích trong ngôn ngữ lập trình hướng đối tượng, ánh xạ các đối tượng nghiệp vụ của một ứng dụng tới các bảng cơ sở dữ liệu quan hệ, để dữ liệu có thể dễ dàng được truy cập và cập nhật hoàn toàn thông qua mô hình đối tượng của một ứng dụng.

ORM là một framework ánh xạ đến thông tin của các đối tượng trong ngôn ngữ hướng đối tượng thành các các hàng trong bảng quan hệ được tìm thấy trong cơ sở dữ liệu quan hệ.



Lợi ích:

* Hạn chế việc thao tác trực tiếp các câu lệnh SQL
* Tăng năng suất, rút ngắn thời gian và chi phí phát triển
* Giảm sự phụ thuộc của ứng dụng vào một hệ cơ sở dữ liệu cụ thể

# Ánh xạ

## File XML

File mapping được sử dụng để ánh xạ một hoặc nhiều lớp với các bảng trong database.



Các thẻ trong file mapping

* Tài liệu mapping là một tài liệu XML có <hibernate-mapping> là phần tử gốc chứa tất cả các phần tử <class>.
* Phần tử <meta> là thành phần tùy chọn và có thể được sử dụng để tạo ra mô tả lớp
* Phần tử **<id>** ánh xạ thuộc tính ID duy nhất trong lớp tới khóa chính của bảng cơ sở dữ liệu. Thuộc tính **name** của id đề cập đến thuộc tính trong lớp và thuộc tính **column** đề cập đến cột trong bảng cơ sở dữ liệu. Thuộc tính **type** giữ kiểu ánh xạ hibernate, các kiểu mapping này sẽ chuyển đổi từ kiểu dữ liệu C# sang SQL.
* Phần tử **<generator>** bên trong phần tử id được sử dụng để tự động tạo giá trị cho khóa chính. Thiết lập thuộc tính **class** của phần tử generator được đặt là **native** để cho phép hibernate chọn **identity**, **sequence** hoặc **hilo** để tạo khoá chính tùy thuộc vào khả năng của cơ sở dữ liệu.
* Phần tử **<property>** được sử dụng để ánh xạ một thuộc tính của lớp vào một cột trong bảng cơ sở dữ liệu. Thuộc tính **name** của phần tử đề cập đến thuộc tính trong lớp và thuộc tính **column** đề cập đến cột trong bảng cơ sở dữ liệu.

### Cấu hình

* Driver (driver\_class): Thuộc tính này dành riêng cho RDBMS được sử dụng và thường được đặt bởi phương ngữ.
* Dialect: Thuộc tính này là bắt buộc. Một lớp để xây dựng các chuỗi SQL dành riêng cho RDBMS. Thông thường, đây là một trong nhiều phương ngữ từ không gian tên NHibernate.Dialect. (tùy thuộc vào phiên bản SQL)
* Cấu hình bằng code



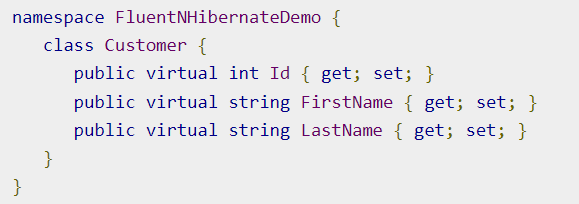
* Cấu hình bằng file XML: (\*.cfg.xml)

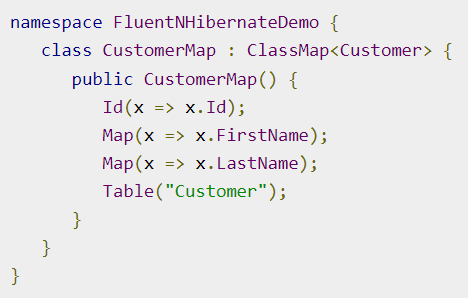


## Fluent Nhibernate

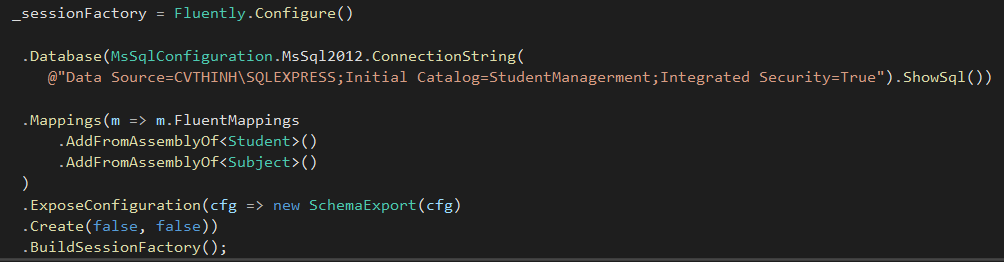
Fluent mapping là kiểu ánh xạ trùng tên mà Fluent NHibernate sử dụng. Đó là một fluent interface cho phép bạn lập bản đồ các thực thể của mình hoàn toàn bằng code, với tất cả sự an toàn trong thời gian biên dịch và khả năng tái cấu trúc mang lại.

* Bạn có thể dễ dàng thay đổi ánh xạ của mình giống như code của bạn và trình biên dịch sẽ không thành công với bất kỳ lỗi chính tả nào.
* Nó có một hệ thống cấu hình thông thường, nơi bạn có thể chỉ định patterns để ghi đè các quy ước đặt tên và nhiều thứ khác.





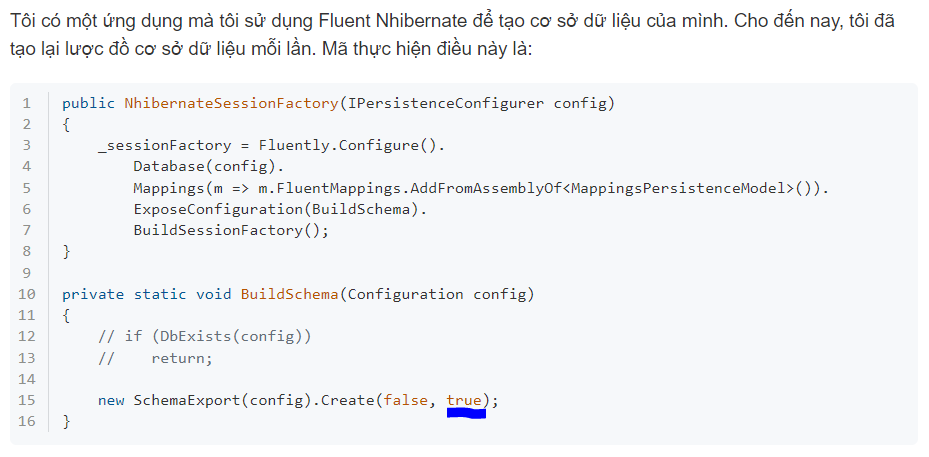
### Cấu hình



Database: thiết lập phiên bản và chuỗi kết nối vào.

Mappings: thiết lập ánh xạ cho đối tượng thông qua AddFromAssemblyOf.

ExposeConfiguration: thiết lập lược đồ.



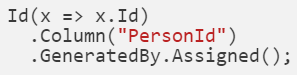
* do create true nên lược đồ sẽ tạo lại mỗi lần chạy.

### ClassMap

Mọi ánh xạ bên trong ClassMap <T> đều được xây dựng bằng cách sử dụng các biểu thức lambda, cho phép tham chiếu các thuộc tính trên các thực thể của bạn mà không ảnh hưởng đến sự an toàn trong thời gian biên dịch. Các lambdas thường có dạng x => x.Property; x ở bên trái là khai báo tham số, sẽ là thực thể mà bạn đang ánh xạ, trong khi x.Property đang truy cập một thuộc tính trên thực thể của bạn.

### Id

Mọi ánh xạ đều yêu cầu một loại Id nào đó, chúng được ánh xạ bằng phương pháp Id; phương thức này nhận một biểu thức lambda của thuộc tính bạn muốn sử dụng làm Id của mình.



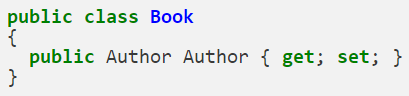
Nếu chúng ta cần thay đổi cột mà thuộc tính ánh xạ tới, chúng ta có thể sử dụng phương thức Column hoặc để chỉ định rõ ràng identity generator, bạn có thể sử dụng thuộc tính GeneratedBy.

### Properties



### Relationships

* References / many-to-one: Relationships là để tạo mối quan hệ nhiều-một giữa hai thực thể.





* HasMany / one-to-many: HasMany là mối quan hệ dựa trên collection-based mà bạn sẽ sử dụng



Để ánh xạ ta sử dụng:



* HasManyToMany / many-to-many: hoạt động như HasMany

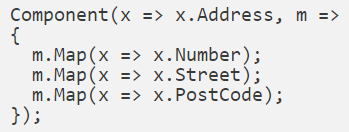


* HasOne / one-to-one: bạn thực sự muốn một-một, thì bạn có thể sử dụng phương pháp HasOne.



### Components

Components là một cách thông minh để ánh xạ một mô hình dữ liệu chuẩn hóa thành một mô hình đối tượng hợp lý hơn.



Phương thức Component nhận hai tham số. Tham số đầu tiên là một lambda truy cập thuộc tính, giống như tất cả các phương thức khác và tham số thứ hai là một lambda khác (thường được gọi là bao đóng lồng nhau) xác định một phạm vi mới để xác định ánh xạ của con cụ thể đó(trong trường hợp này là component).

# Truy vấn

## Linq

Cú pháp truy vấn:



có thể sử dụng câu lệnh Select tương tự.







## Hibernate Query Language (HQL)



Chuỗi HQL này giống SQL, sự khác biệt chính là FirstName là tên thuộc tính chứ không phải tên cột.

## Criteria Queries



Cho phép truy vấn động nhưng khó đọc hơn HQL.

## QueryOver Queries



Cũng như Criteria nhưng cú pháp đẹp hơn.

## Native SQL

