**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Môn: Mẫu thiết kế hướng đối tượng và ứng dụng**

**TÀI LIỆU HƯỚNG DẪN SỬ DỤNG SOURCE CODE**

DATABASE ACCESS MANAGEMENT (DAM) FRAMEWORK

**Giáo viên:** Nguyễn Đức Huy

Hồ Tuấn Thanh

**Sinh viên:** Trang Trung Hoàng Phúc 1612521

Tô Hiếu Minh 1612377

Lâm Cương Đạt 1612098

# Cài đặt các thư viện liên kết cơ sở dữ liệu

Đối với CSDL Postgres, vào thư mục project của source code và chạy câu lệnh trong command line

dotnet add package Npgsql –version 4.1.2

Đối với CSDL MySQL, add preferences đến thư mục MySql.Data.dll, thường nằm ở thư mục

C:\Program Files\MySQL\MySQL Installer for Windows\

# Tạo các lớp đê quản lý

Đầu tiên ta cần khai báo các lớp cần được quản lý. Để có thể sử dụng framework thì lớp được khai báo cần có constructor không tham số. Ngoài ra, cần khai báo các Attribute cho lớp để ORMManager biết để có thể thực hiện việc Mapping.

* Atribute TableName: Chứa thông tin về tên của bảng ứng với lớp đó

public TableName(string name)

{

    tableName = name;

}

* Atribute TableName: Chứa thông tin về tên của bảng ứng với lớp đó

public Column(string column, bool isKey = false, bool autoincrement = false)

{

    this.columnName = column;

    this.isKey = isKey;

    this.autoincrement = autoincrement;

}

* Atribute TableName: Chứa thông tin về tên của bảng ứng với lớp đó

public HasMany(string name, string inside, string fk)

{

    className = name;

    key = inside;

    foreignKey = fk;

}

Ví dụ sử dụng các Attribute:

[TableName("Classes")]

class Class

{

    [Column("ID", isKey: true, autoincrement: true)]

    public int Id { get; set; }

    [Column("Name")]

    public string tenLop { get; set; }

    [HasMany("Student", "Id", "classid")]

    public List<Student> dsHocSinh { get; set; }

}

# Tạo ORMManager

Để tạo ORMManager thì cần 2 thành phần là DBManager và Parser. Các thành phần này có thể được khởi tạo thông qua lớp DBFactory. Người dùng cần xác định loại cơ sở dữ liệu để tạo các thành phần cho chính xác.

DBFactory factory = new MySQLFactory();

DBManager dBManager = factory.CreateDBManager("localhost", 3306, "school", "root", "palo1234");

Parser parser = factory2.CreateParser();

ORMManager<Student> orm = new ORMManager<Class>(dBManager, parser);

Để thực hiện các câu lệnh Insert, Delete, Update cần tạo ra các câu lệnh đó và gọi Execute() để thực thi.

Đối với câu lệnh Select, ta cần gọi hàm ToList() để trả về List<Object>.

# Câu lệnh Insert

Chèn một dòng vào cơ sở dữ liệu

Ví dụ

Student student = new Student("Design Pattern", true, "123@gmail.com", "1234546", "Tp.HCM", new DateTime(1998, 5, 24), 2);

int numCol = orm.Insert(student).Execute();

# Câu lệnh Update

Chỉnh sửa các dòng dữ liệu đã có sẵn. Có thể chỉnh sửa bằng 2 cách:

* Truyền vào một đối tượng: chương trình sẽ tự động chỉnh sửa dòng dữ liệu ứng với đối tượng đó dựa vào khóa chính mà người dùng định nghĩa.

Student student = new Student(31, "Design Pattern", true, "123@gmail.com", "1234546", "Tp.HCM", new DateTime(1998, 5, 24), 2);

int numCol = orm.Update(student).Execute();

* Truyền vào điều kiện (dựa vào hàm Where) và thay đổi cần phải thực hiện (dựa vào hàm Set)

int numCol = orm.Update().Set("diaChi", "VietNam"). Where(Condition.Equal("sdt","1234546")).Execute();

# Câu lệnh Delete

Xóa các dòng dữ liệu có sẵn. Có thể xóa bằng 2 cách:

* Truyền vào một đối tượng: Chương trình sẽ tự động tìm dòng dữ liệu ứng với đối tượng đó để xóa.

Student student = new Student(31, "Design Pattern", true, "123@gmail.com", "1234546", "Tp.HCM", new DateTime(1998, 5, 24), 2);

int numCol = orm.Delete(student).Execute();

* Truyền vào một đối tượng: Chương trình sẽ tự động tìm dòng dữ liệu ứng với đối tượng đó để xóa.

int numCol = orm.Delete().Where(Condition.GreaterThan("lop", 2)).Execute();

# Câu lệnh Select

Chọn từ cơ sở dữ liệu dựa vào điều kiện người dùng định nghĩa và trả về môt List<Object> trong đó Object có thể là lớp template được định nghĩa khi tạo ORMManger (khi Select không có Projection) hoặc là Dictionary<string,Object> (từ điển map từ tên thuộc tính đến giá trị của thuộc tính đó).

Các hàm hỗ trợ trong việc chọn dữ liệu:

* Where(Condition condition): Điều kiện để chọn dữ liệu
* AddProjection(string attr, string alias = ""): Chọn thuộc tính cần lấy. Khi người dùng sử dụng câu lệnh GroupBy thì có thể thêm hàm tổng hợp trên thuộc tính đó bằng câu lệnh AddProjection(string attr, string aggFunc = "", string alias = "")
* GroupBy(string attr): Gom nhóm theo thuộc tính
* Having(Condition condition): Chọn dữ liệu dựa vào thuộc tính gom nhóm.

Ví dụ sử dụng

List<Object> res = orm.Select().AddProjection("ten").GroupBy("lop").AddProjection("ten", Aggregate.COUNT, "demten")

                .Having(Condition.LessThan("ten", 20, Aggregate.COUNT)).ToList();

Console.WriteLine(((Dictionary<string, Object>)res[0])["ten"]);