**Tìm Hiểu Về Mô Hình 3 Lớp (3 Layer)**

* Do dạo gần đây mình đang làm về một số app desktop sử dụng mô hình 3 lớp, nên mình sẽ note lại những điều cần chú ý trong mô hình 3 lớp.

**1. Mô hình 3 lớp là gì??**

**Khái niệm:**

* Mô hình 3 lớp hay còn được gọi là mô hình Three Layer(3-Layer), mô hình này ra đời nhằm phân chia các thành phần trong hệ thống, các thành phần cùng chức năng sẽ được nhóm lại với nhau và phân chia công việc cho từng nhóm để dữ liệu không bị chồng chéo và chạy lộn xộn.
* Mô hình này phát huy hiệu quả nhất khi bạn xây dựng một hệ thống lớn, việc quản lý code và xử lý dữ liệu lỗi dễ dàng hơn.

**Ưu điểm:**

* Phân loại rõ ràng các lớp có các nhiệm vụ khác nhau. Từ đó ta có thể quản lý và maintain project tốt hơn.
* Dễ dàng phân loại các hành động tại Business.
* Dễ dàng phân loại các hàm truy xuất tại Database, phân loại hàm theo table,…
* Ứng dụng được cho các project lớn ở bên ngoài.
* …

**Lưu ý khi xây dựng mô hình 3 lớp:**

* Cần một solution riêng cho project.
* Cần 3 project khác nhau để làm nên 3 lớp, tên Project đặt như sau:
* Lớp GUI: (VD: QuanLy\_GUI)
* Lớp Business: (VD: QuanLy\_BUS)
* Lớp Data Access: (VD: QuanLy\_DAL)
* Lớp DTO: (VD: QuanLy\_DTO)

**2. Giới thiệu về mô hình 3 lớp**

Mô hình 3-layer gồm có 3 phần chính:

Presentation Layer (GUI)

* Lớp này có nhiệm vụ chính là giao tiếp với người dùng. Nó gồm các thành phần giao diện ( winform, webform, …) và thực hiện các công việc như nhập liệu, hiển thị dữ liệu, kiểm tra tính đúng đắn dữ liệu trước khi gọi lớp Business Logic Layer (BLL).

Business Logic Layer (BLL) Layer này phân ra 2 thành nhiệm vụ:

* Đây là nơi đáp ứng các yêu cầu thao tác dữ liệu của GUI layer, xử lý chính nguồn dữ liệu từ Presentation Layer trước khi truyền xuống Data Access Layer và lưu xuống hệ quản trị CSDL.
* Đây còn là nơi kiểm tra các ràng buộc, tính toàn vẹn và hợp lệ dữ liệu, thực hiện tính toán và xử lý các yêu cầu nghiệp vụ, trước khi trả kết quả về Presentation Layer.

Data Access Layer (DAL)

* Lớp này có chức năng giao tiếp với hệ quản trị CSDL như thực hiện các công việc liên quan đến lưu trữ và truy vấn dữ liệu ( tìm kiếm, thêm, xóa, sửa,…).

**3. Các thành phần của từng lớp**

**Presentation Layer (GUI)**

Có hai thành phần chính sau đây với những tác vụ cụ thể :

* **UI Components** : gồm các thành phần tạo nên giao diện của ứng dụng (GUI). Chúng chịu trách nhiệm thu nhận và hiển thị dữ liệu cho người dùng… Ví dụ : textbox, button, combobox, …
* **UI Process Components** : là thành phần chịu trách nhiệm quản lý các quá trình chuyển đổi giữa các UI… Ví dụ : Sắp xếp quá trình kiểm tra thông tin khách hàng:
  1. Hiển thị màn hình tra cứu ID.
  2. Hiển thị màn hình thông tin chi tiết khách hàng tương ứng.
  3. Hiển thị màn hình liên lạc với khách hàng.

**Bussiness Layer (BLL)**

Lớp này gồm 4 thành phần:

* **Service Interface** : là thành phần giao diện lập trình mà lớp này cung cấp cho lớp Presentation sử dụng.
* **Bussiness Workflows** : chịu trách nhiệm xác định và điều phối các quy trình nghiệp vụ gồm nhiều bước và kéo dài. Những quy trình này phải được sắp xếp và thực hiện theo một thứ tự chính xác.
* **Bussiness Components** : chịu trách nhiệm kiểm tra các quy tắc nghiệp vụ, ràng buộc logic và thực hiện các công việc . Các thành phần này cũng thực hiện các dịch vụ mà Service Interface cung cấp và Business Workflows sẽ sử dụng nó.
* **Bussiness Entities** : thường được sử dụng như Data Transfer Objects ( DTO ) . Bạn có thể sử dụng để truyền dữ liệu giữa các lớp (Presentation và Data Layer). Chúng thường là cấu trúc dữ liệu ( DataSets, XML,… ) hay các lớp đối tượng đã được tùy chỉnh. Ví dụ : tạo 1 class Student lưu trữ các dữ liệu về tên, ngày sinh, ID, lớp.

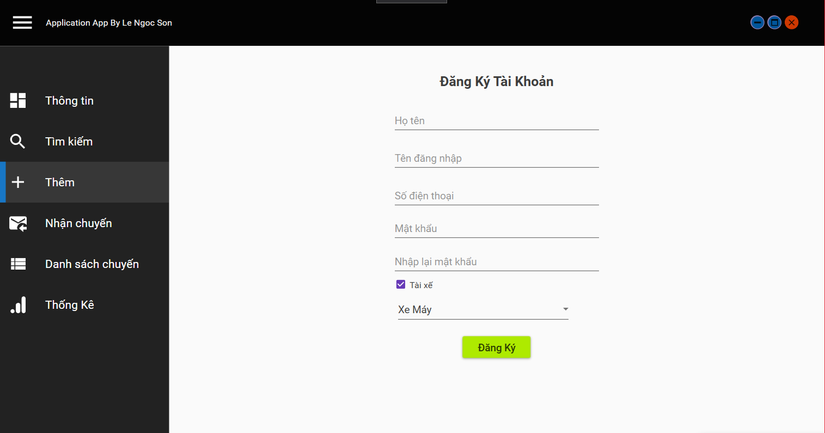
**Data Layer (DAL)**

* **Data Access Logic Components** : chịu trách nhiệm chính lưu trữ và truy xuất dữ liệu từ các nguồn dữ liệu (Data Sources) như XML, file system,… Hơn nữa còn tạo thuận lợi cho việc dễ cấu hình và bảo trì. Service Agents : giúp bạn gọi và tương tác với các dịch vụ từ bên ngoài một cách dễ dàng và đơn giản.

Để hiểu rõ hơn về cấu trúc và cách xây dựng của mô hình 3 lớp, chúng ta cùng tham khảo một ví dụ về mô hình quản lí Uber gồm các lớp BUS, DAO, DTO, GUI ( Lớp GUI sẽ là phần chương trình chính).

*Do ở đây mình lười nên mình gom lại thành các folder để dễ gọi nhau :v (Nếu làm đúng chuẩn thì phải là tạo ra các project trong Solution mới đúng)*

* Đầu tiên là GUI gồm các button, texbox, ... mà người dùng sẽ tương tác với màn hình giao diện này.



* Lớp DTO, chứa những dữ liệu được xây dựng dưới dạng lớp đối tượng. Mỗi một User sẽ mang những thuộc tính sau:

namespace UberManagerment\_WPF.DTO

{

public class Account

{

string name;

string userName;

string passWord;

string telephone;

string status;

public string Name { get => name; set => name = value; }

public string UserName { get => userName; set => userName = value; }

public string PassWord { get => passWord; set => passWord = value; }

public string Telephone { get => telephone; set => telephone = value; }

public string Status { get => status; set => status = value; }

//Hàm khởi tạo

public Account()

{

Name = "";

UserName = "";

PassWord = "";

Telephone = "";

Status = "0";

}

public Account(string Name, string UserName, string Password, string Telephone, string Status)

{

this.Name = Name;

this.UserName = UserName;

this.PassWord = PassWord;

this.Telephone = Telephone;

this.Status = Status;

}

}

Các nghiệp vụ xử lý chính sẽ được đặt ở lớp BUS (hay là BLL) gồm các nghiệp vụ insert, update, delete,...

namespace UberManagerment\_WPF.BUS

{

public class List\_Driver\_BUS

{

private static List\_Driver\_BUS instance;

public static List\_Driver\_BUS Instance

{

get

{

if (instance == null)

instance = new List\_Driver\_BUS();

return instance;

}

}

public void ShowListDriver(DataGrid data)

{

data.ItemsSource = List\_Driver\_DAO.Instance.ShowListDriver();

}

public void ShowListDriverMotobike(DataGrid data)

{

data.ItemsSource = List\_Driver\_DAO.Instance.ShowListDriver\_Motobike();

}

public void ShowListDriverCar(DataGrid data)

{

data.ItemsSource = List\_Driver\_DAO.Instance.ShowListDriver\_Car();

}

public void ShowListDriverTaxiCar(DataGrid data)

{

data.ItemsSource = List\_Driver\_DAO.Instance.ShowListDriver\_taxiCar();

}

}

}

Và cuối cùng là lớp DAO ( hay là DAL ). Truy vấn đến cơ sở dữ liệu

http://kitchencake.vn

namespace UberManagerment\_WPF.DAO

{

class List\_Driver\_DAO

{

private static List\_Driver\_DAO instance;

public static List\_Driver\_DAO Instance

{

get

{

if (instance == null)

instance = new List\_Driver\_DAO();

return instance;

}

}

public List<Driver\_Motobike\_DTO> LstDriver\_Motobike { get => lstDriver\_Motobike; set => lstDriver\_Motobike = value; }

public List<Driver\_Car\_DTO> LstDriver\_Car { get => lstDriver\_Car; set => lstDriver\_Car = value; }

public List<Driver\_TaxiCar\_DTO> LstDriver\_TaxiCar { get => lstDriver\_TaxiCar; set => lstDriver\_TaxiCar = value; }

public List<Travell\_DTO> LstTravel { get => lstTravel; set => lstTravel = value; }

public List\_Driver\_DAO() { }

List<Travell\_DTO> lstTravel;

private List<Driver\_Motobike\_DTO> lstDriver\_Motobike;

private List<Driver\_Car\_DTO> lstDriver\_Car;

private List<Driver\_TaxiCar\_DTO> lstDriver\_TaxiCar;

public List<Driver\_DTO> ShowListDriver()

{

string fileName = Static\_Instance.directory + "\\XML\\data\_Driver.xml";

List<Driver\_DTO> lstDriver = new List<Driver\_DTO>();

LstDriver\_Motobike = new List<Driver\_Motobike\_DTO>();

LstDriver\_Car = new List<Driver\_Car\_DTO>();

LstDriver\_TaxiCar = new List<Driver\_TaxiCar\_DTO>();

LstTravel = new List<Travell\_DTO>();

XmlDocument reader = new XmlDocument();

reader.Load(fileName);

foreach (XmlNode node in reader.DocumentElement.ChildNodes)

{

Driver\_DTO driver = new Driver\_DTO();

driver.Id = node["ID"].InnerText;

driver.Name = node["Name"].InnerText;

driver.UserName = node["UserName"].InnerText;

driver.PassWord = node["Password"].InnerText;

driver.Telephone = node["Telephone"].InnerText;

driver.Type\_Driver = Static\_Instance.checkTypeCar(node["Type\_Driver"].InnerText);

string type = node["Type\_Driver"].InnerText;

driver.Location\_Driver = node["Location"].InnerText;

driver.Status = Static\_Instance.checkStatus(node["Status"].InnerText);

Travell\_DTO travell = new Travell\_DTO(driver.Name, driver.UserName,

driver.PassWord, driver.Telephone, driver.Status, driver.Id,

driver.Type\_Driver, driver.Location\_Driver, "", "", "",

DateTime.Now.ToString(), DateTime.Today.ToString());

LstTravel.Add(travell);

lstDriver.Add(driver);

if (type == "0")

{

Driver\_Motobike\_DTO driver\_Motobike = new Driver\_Motobike\_DTO(driver.Name, driver.UserName,

driver.PassWord, driver.Telephone, driver.Status, driver.Id, driver.Type\_Driver, driver.Location\_Driver);

driver\_Motobike.Type\_Driver = "Xe máy";

LstDriver\_Motobike.Add(driver\_Motobike);

}

else if (type == "1")

{

Driver\_Car\_DTO driver\_Car = new Driver\_Car\_DTO(driver.Name, driver.UserName, driver.PassWord,

driver.Telephone, driver.Status, driver.Id, driver.Type\_Driver, driver.Location\_Driver,

int.Parse(node["Slot"].InnerText));

driver\_Car.Type\_Driver = "Xe ô tô";

LstDriver\_Car.Add(driver\_Car);

}

else

{

Driver\_TaxiCar\_DTO driver\_TaxiCar = new Driver\_TaxiCar\_DTO(driver.Name, driver.UserName,

driver.PassWord, driver.Telephone, driver.Status, driver.Id, driver.Type\_Driver,

driver.Location\_Driver, int.Parse(node["Weight"].InnerText));

driver\_TaxiCar.Type\_Driver = "Xe ô tô tải";

LstDriver\_TaxiCar.Add(driver\_TaxiCar);

}

}

return lstDriver;

}

//........................................................

//..............INSERT..............//

//..............UPDATE..............//

//..............DELETE..............//

//........................................................

Vậy là mình đã mô phỏng xong cách cấu trúc một chương trình sử dụng mô hình 3 lớp. Nếu có sai sót thì mong mọi người góp ý thêm ạ. Cảm ơn bạn đã đọc bài viết này.