Increasing Maintainability of Sale Charting Programs Using Fluent NHibernate ORM and MVP Pattern

Our daily need to computers and mechanizing most of our affairs is something we cannot escape of. In this situation programmers are also trying to develop programs that can meet most of the needs of users perfectly. In order of this purpose they are working on the process of designing a program to have a standard structure and be validated.

Mentioning all of the points of a good program is not going to be fitted in our article, so here are some of them:

- **Correctness:** How the program is going to meet the needs of its user.
- **Robustness & Reliability:** How it can treat bad situations and fault inputs.
- **Efficiency:** The way it optimizes the hardware and software sources.
- **Portability:** The functionality of it when switching through different platforms, for instance does it work on different operating systems or not.
- **Reusability:** Reinstalling a part and/or all of it on another system.
- **Usability:** How it's easy to use for its users and the guides are easy to understand.
- **Maintainability:** How much does it cost for a user to use and take care of the program in a recommended way.
- **Verifiability:** How easy is to test the program and make sure it's working in a proper way.
- **Compatibility or Interoperability:** Does the program make collision with other programs and can it handle other data with standard formats.
- …

If your program is written in a way that both database and the user interface are together in one place, you will face lots of difficulties. For instance, for your further developments, for a single change in an area you have to change most parts of your program. Even if it faces an error according to its chaos, the process of its debugging would be really hard and complicated. You may have difficulties in changing it or even this will not be possible for your program. Imagine you want to add some windows and forms to it, and the codes of different places are mixed together, you will definitely have a lot of difficulties in adding them.

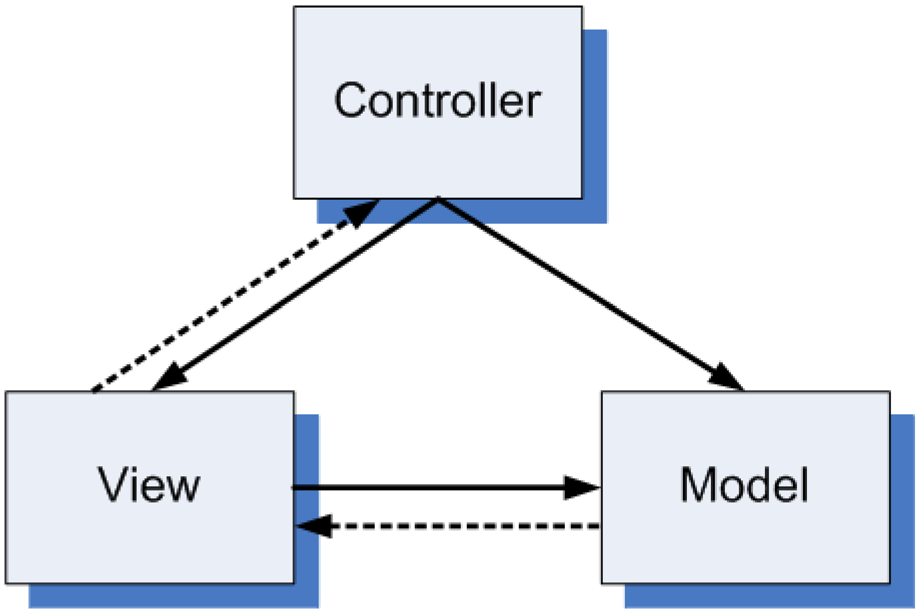
For increasing the value of a program mostly in maintainability part we can do lots of things. One of them that is common nowadays, is to make the program into layers, which means each part of it will have its own duties and other layers use the output of each layer.
Number of these layers depends on the program itself and its design, which can be 2, 3, 4, 5 or even more. But the standard and most usable number of them is 3 layers and the way that they plan and implement it is called "Three-Tier Architecture".

In theory these 3 layers are as below:

- **Data Access Layer**
- **Business Logic Layer**
- **Presentation layer**

The architecture of these 3 layers is an abstract architecture and although it tells about these 3 layers, it will not tell any specific thing about classes in these layers and the connection between them. 2 more accurate models are given for its implement:

- **Model-View-Controller (MVC)**
- **Model-View-Presenter (MVP)**

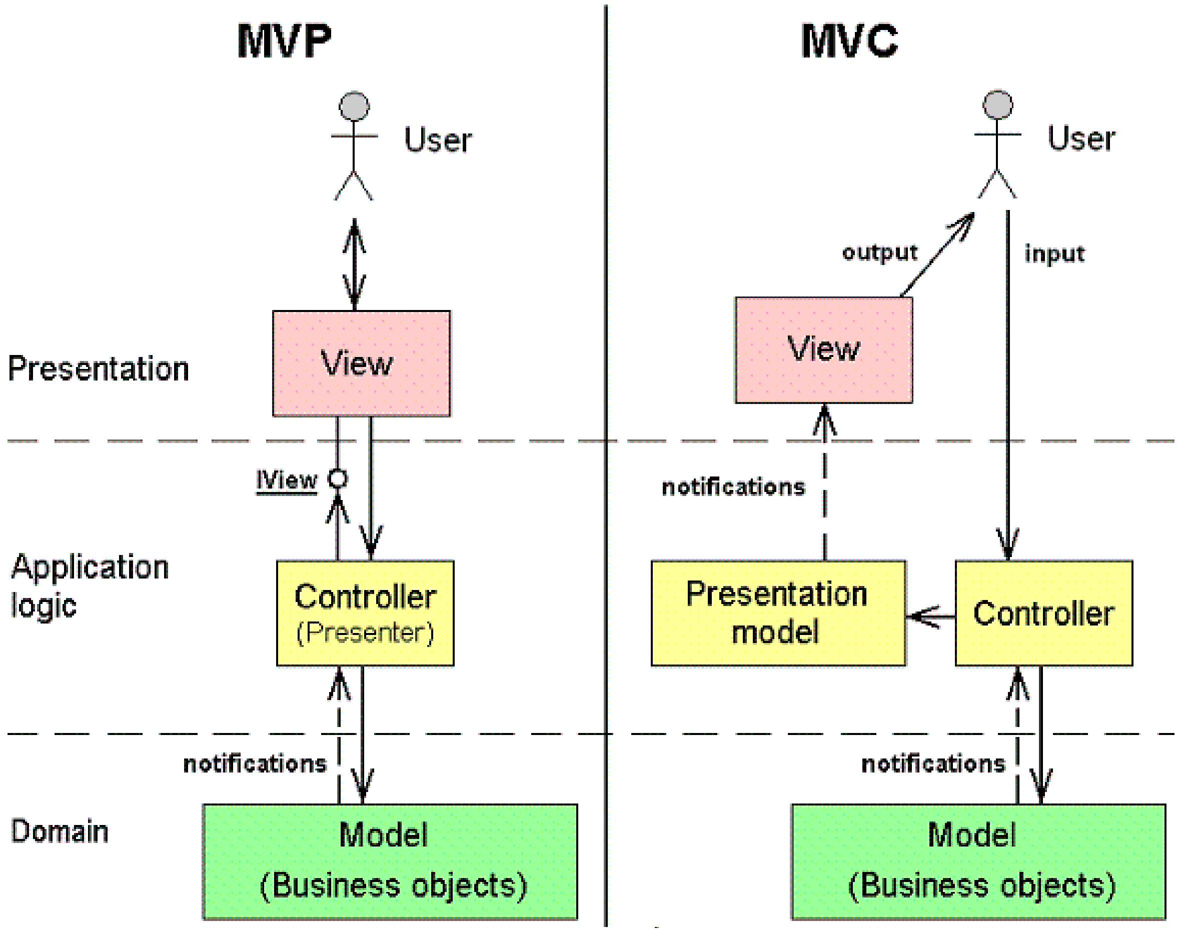


- **MODEL**
  It contains the main datas of the program, such as datas, databases, validation rules and the logic of complexing.
- **VIEW**
  This part that is also known as Presentation Layer in 3 Tier Architecture, should make the connection with the last user and get data from the user and show the prepared data to the user by making connection with the other parts. (Model and Controller)
  In fact the most important part that we should always keep in mind is that this layer shouldn't control the validity of the input datas from the user or even control the validity of the data that is showing. It just works with pure datas.
- **CONTROLLER**
  It controls the logical progress of the program. This part can implement and control the data progres of the program by interacting between Model and View. We can even say that controller is the connection between Model and View; it means that it works with Model and at last it will choose a monitor for Viewing the contents.
  It manages the input of the user, answers them and intracts with the user. For instance the controller manages the inquiry statements of the database and sends them to Model, and the Model should implement the inquiry.

MVP is the optimized version od MVC with 2 differences:

1. In MVC, the controllers get and process the users inputs, But in MVP, Views get the inputs and assign the process of them to its related controller. That's why MVP is more compatible with the modern user interfaces that get the users datas themselves and it's more popular because of this.

2. In MVC, the controllers will change themselves by changing the Intermediate Presentation Model. (By using the Observer Pattern)
In this way the Views will change to pure Observers without a direct access, but in MVP this problem is solved by a direct connection from the Controller to the Viewer. This makes MVP easier than MVC.



In fact Changing an application to different layers (Model-View-Controller) gives more speed and flexibility in developing, testing and …
Here is an example, you can make changes in View of your program without any need in changing the Model. You can even separate the duties and parts of your program to different people, without being worried about the connection between them, for instance designers can work on View, while the developers are working on the Model.

For using the MVP Pattern we should spend more time and cost and we may not even use it easily. In order to use it we have various frameworks, that the most usable one for .Net is the MVC# that is an Open Source program and has lots of advantages and it's easy to use that gives us more speed for developing the program.

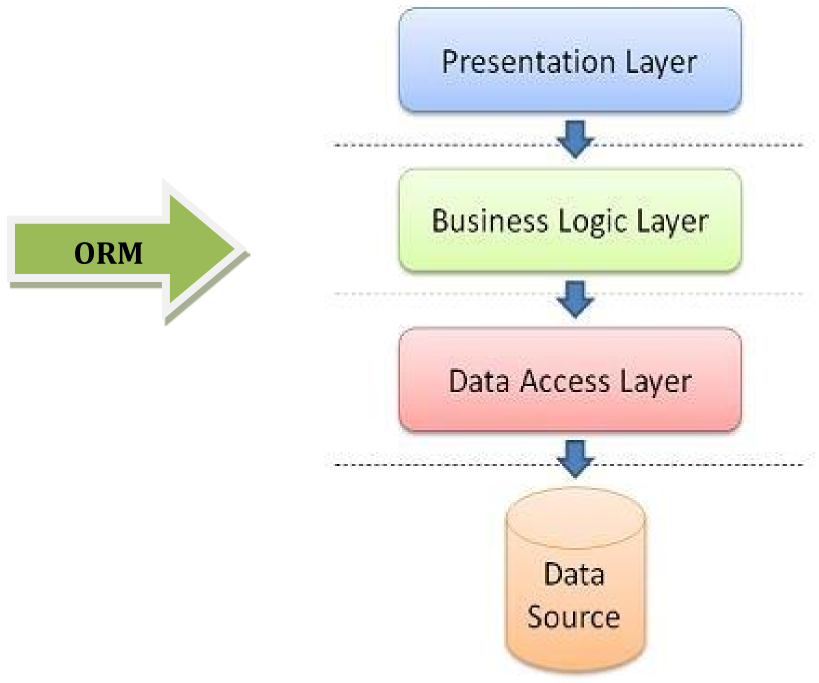
Preparing Data Layer Access part in a program usually contains %30 time of the whole prepration of it, and we should mention, this repeating process is not something valueable and will not add any value to the program. Most of the business programs need the Data Layer Access. So why do we need this boring and repeating proccess over and over again in each program?

The purpose of ORM (Object Relation Mapping) is to decrease this repeatation for a programmer. By using this library we will not have the Stored Procedure any more. You will not work with ADO.Net anymore. In this way we can spend more time on the main parts and designing the program instead of wasting it on codding a repeated layer. Even most of the professionals of these tools are believed that the doing this part by the programmers, is a kind of fraud. (Spending more time on a product and having some accidental bugs in the Data Access Layer by a not so professional programmer.)

ORM tools responsibility is to read your database structure and making some classes according to these structures, making connection between the objects and tables, Views, stored procedures and etc,. These tools can also define the *one-to-one, one-to-many, many-to-one* and *many-to-many* connections with objects according to your database structure.

NHibernate is an ORM that works with Microsoft.Net platform and suggests a framework for easier work with a database with less complexity. In this model our Entities in database map to obejcts for using in .Net platform. By this programmers will scape of the complexity of databases.

NHibernate is a free Open Source Program and is a part of Java ORM Hibernate that gives us XML codes, entities and relations, it automatically generates SQL codes for saving and recovering the datas, and you can also work with the meta data saved in the source code.



Another library for mapping is recently made that is called **Fluent NHibernate** which took the attention of the ones who are interested to .Net framework. With the help of this library the progress of writing objects into charts is done by the codes within the program, instead of the XML files.
This has lots of advantages, such as using a single and complete programming language for defining the maps, automatically checking of the data formats, and even the ability to define a specific logic for the mapping section of the program.

One of the important parts of this program that does the predications of the ERP program, is the Selling Diagrams. In most of the programs this diagram has a weak user interface and has low options for drawing. An ERP system uses different software and hardware sources to reach this massive set of datas. One key point in ERP systems is using a unique database to save the datas for different modules of the system. In most of the presented programs this database is not designed correctly and has a low efficiency. In this project we want to increase the efficiency of the selling diagram by using the ORM Fluent NHibernate, redesigning the database and user interface.

In order to do so we need a complete list of products, selling sites, site warehouses, features of the products in the site and selling informations. We definitly need the products and the sites to be categorized.
We need to take note from the number of items we have in our selling sites several times a year. Each product has an expiration date. In each static we have the sold amount is saved, which means how much of a product did we sell in a period of time. Each site has specific parameters such as address and other explanations and also each product in each site may contain parameters like price, availability and etc, with the choice of the user.
For predicting the sell these diagrams are needed. It is in a page with two trees for products and sites and two fields for time to make the period and an option menu.