Chương 4: Thiết kế và cài đặt

* 1. Nền tảng sử dụng
     1. Kiến trúc hệ thống

Presentation Layer

Service Layer

Business Layer

Data Access Layer

Database

* **Tầng Presentation**

Tầng này làm nhiệm vụ giao tiếp với người dùng cuối để thu thập dữ liệu và hiển thị kết quả hoặc dữ liệu thông qua các thành phần trong giao diện người dùng. Tầng này sử dụng các dịch vụ do tầng Business hoặc Service cung cấp.

* **Tầng service**

Tầng này chịu trách nhiệm liên kết các đối tượng các thành phần trong tầng Business với nhau. Tạo ra các dịch vụ để các thành phần trên tầng Presentation có thể sử dụng và hiển thị cho người dùng.

* **Tầng Business**

Tầng này thực hiện các nghiệp vụ chính của hệ thống, sử dụng các dịch vụ do tầng DataAcess cung cấp, và cung cấp các dịch vụ cho tầng Presentation và service. Tầng này cũng có thể sử dụng các dịch vụ của nhà cung cấp khác để thực hiện công việc của mình. Trong tầng này có các thành phần chính như là Business Entities và Business Component. Có thể xem tầng Service phía trên là một phần của tầng business.

* **Tầng Data Access**

Tầng này thực hiện các nghiệp vụ liên quan đến lưu trữ và truy xuất dữ liệu của ứng dụng, thường lớp này sẽ sử dụng các dịch vụ của các hệ quản trị cơ sở dữ liệu như SQL Server, Oracle, để thực hiện nhiệm vụ của mình.

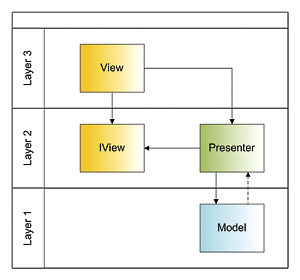
* **Cơ sở dữ liệu**

Mọi dữ liệu mọi thông tin của người dùng sẽ được lưu xuống cơ sở dữ liệu. Trong chương trình sẽ sử dụng hệ quản trị cơ sở dữ liệu SQL Server của Microsoft.

* + 1. Giới thiệu mô hình xây dựng website theo kiến trúc MVP

Mô hình MVP gồm ba thần phần chính View, Model và Presenter.

* Model chứa dữ liệu và các tính toán xử lý logic để giải quyết vấn đề mà Website hướng tới.
* View là thành phần đảm nhận việc thể hiện những dữ liệu của Model và là tổng hợp của các form, control được sử dụng.
* Presenter là thành phần đảm nhận các xử lý thể hiện cũng như tương tác đến dữ liệu bên dưới và có thể tương tác để thay đổi View trong quá trình xử lý.



Trong mô hình MVP, các lớp View sẽ được sử dụng thông qua một Interface được định nghĩa trong .NET. Các lớp Presenter tương ứng sẽ sử dụng Interface này để đọc và ghi dữ liệu lên trên các View. Trong đa số các cách implement, một View sẽ có một Presenter tương ứng của nó. View sẽ khởi tạo Presenter cho nó và truyền cho Presenter này tham chiếu đến chính nó. Khi một event nào trên view được kích hoạt chẳng hạn như button\_clicked, text\_changed,.. bản thân lớp View sẽ không làm gì cả mà sẽ để cho lớp Presenter xử lý những sự kiện đó. Presenter sẽ đọc dữ liệu từ View (vì nó giữ một instance của View như là một member trong class) thông qua View Interface, thực hiện những xử lý ứng với Event được kích hoạt và set những thay đổi từ dữ liệu Model lên trên View thông qua View Interface.  
Trong môi trường .NET, cùng một Presenter có thể được sử dụng cho View trên web như các trang ASP.NET hoặc được sử dụng cho các Form trong Windows Form Application. Các Presenter đọc và ghi dữ liệu thông qua một Interface của .NET nên nó hoàn toàn độc lập với layer View. Chính nhờ cách làm này mà ta có thể áp dụng Unit Test cho các lớp xử lý Presenter rất dễ dàng và nó cũng chính là một trong những lợi ích lớn nhất của MVP cho khả năng tái sử dụng.

**Các nhược điểm của MVP**

* + - * + Có nhiều lớp, nhiều layer phải thực thi; có nghĩa độ phức tạp của ứng dụng sẽ tăng
* Chúng ta phải nghĩ ra một cách để kết hợp giữa các Views và các Presenters. Trong ASP.NET thì các lớp Code-Behind sẽ implement một interface IView nào đó và sử dụng một member Presenter.
* Mô hình Model không biết gì về Presenter. Vì thế nếu dữ liệu Model bị thay đổi từ một chương trình khác thay vì bởi Presenter thì phải nghĩ ra một cách để Presenter biết. Các bạn có thể implement chức năng này nếu cần bằng cách sử dụng các events, hoặc tham khảo thiết kế Observer Pattern.
  1. Quá trình thiết kế
  2. Thiết kế database
     1. Mô hình đồ ER



* + 1. Mô hình quan hệ
  1. Thiết kế giao diện
  2. Thiết kế xử lý

1. Quá trình cài đặt