Model View Presenter (MVP)

Design Patterns

Motivating Example

```
protected void Page_Load(object sender, EventArgs e) {

    if (HttpContext.Current.User.IsInRole("User "))
    {
           FormView1.FindControl(“row").Visible = true;
           FormView1.FindControl("RequiredFieldValidator4").Visible = false;
           if (Request.Path.Contains(@"ClientProfile.aspx?Action=Add"))
           {
               FormView1.FindControl(“loginIDth").Visible = true;
               FormView1.FindControl(“loginIDtd").Visible = true;
               FormView1.FindControl(“loginIDtr").Visible = true;
           }
    }
    else
           FormView1.FindControl("noteRow").Visible = false;

    if (HttpContext.Current.User.IsInRole("Administrator"))
    { ... // Lots more code after this
```

Intent

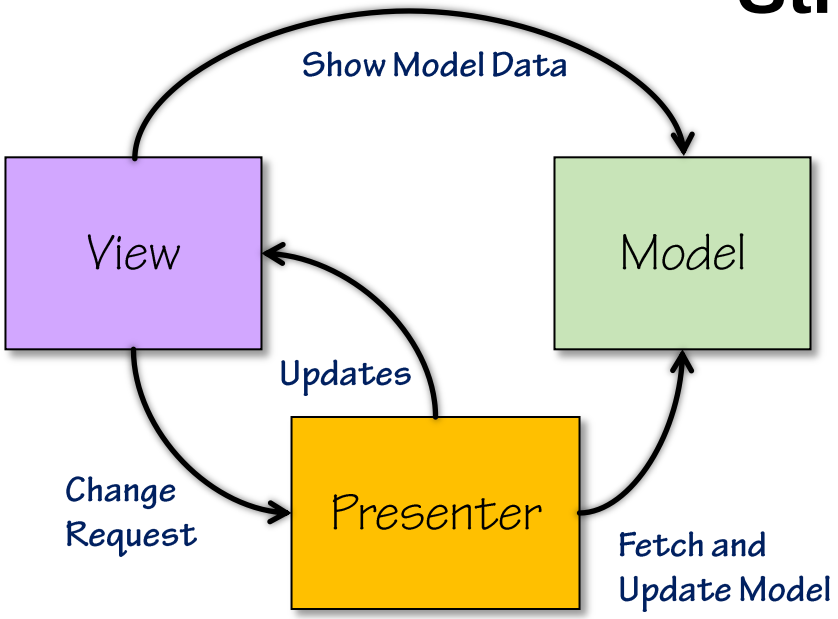
- **Provide clear separation between the following concerns**
  - The data to show
  - The business logic of the application
  - Display of data
- **Ensure each collaborator in the pattern has a single responsibility**
  - Model
  - View
  - Presenter
- **Facilitate isolated testing of each collaborator in the pattern**

Also Known As

Variations on MVP

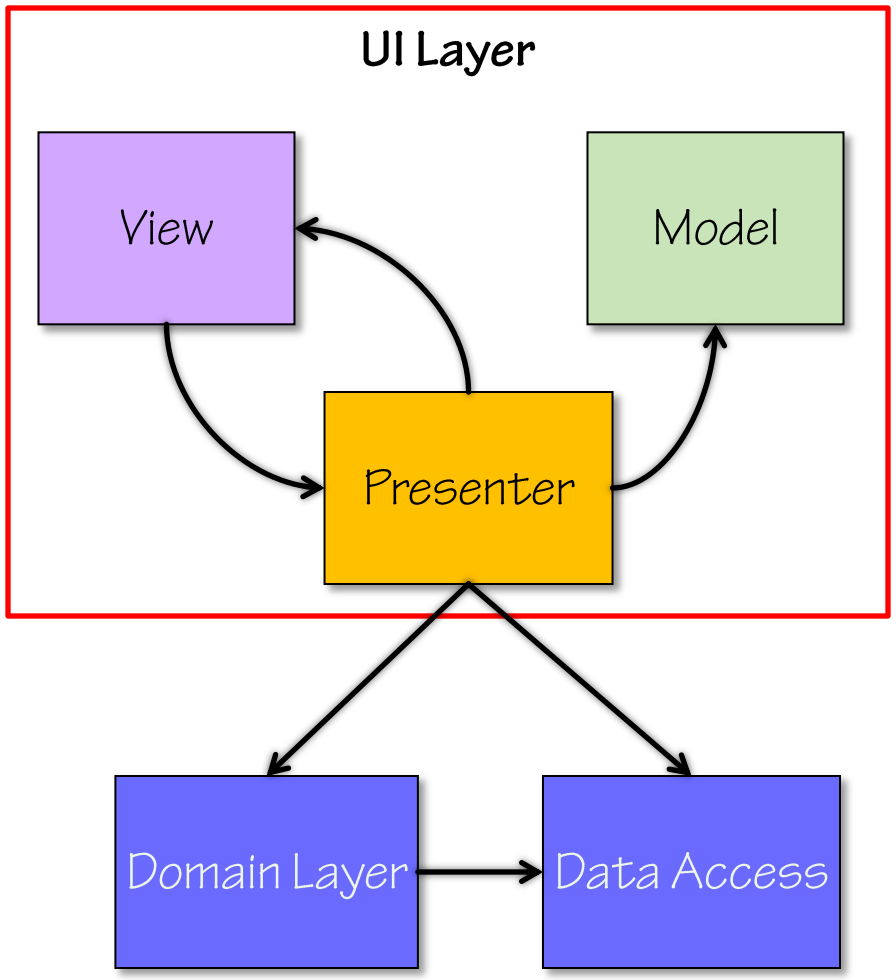
- **Martin Fowler's Take on It**
  - MVP Pattern Retirement Note
    http://link.pluralsight.com/cx3u69
  - Supervising Controller
    http://link.pluralsight.com/cx3u6a
  - Passive View
    http://link.pluralsight.com/cx3u6b
- **MVP Variations on MSDN**
  http://link.pluralsight.com/cx3u6c

Structure



- View and Model know nothing of each other
- Presenter coordinates requests and events between the Model and the View

Structure



- All of the MVP collaborators are part of the UI layer
- The Model isn't part of the Business Domain
- Model is a special purpose view of data just for by the view

Collaboration

- **Model**
  - Holds data destined to be surfaced by the View
- **View**
  - Instantiates the Presenter
  - Requests that the Presenter do work
  - May have a reference to the Model
    or
    May rely on the Presenter to set data on the View using the Model
- **Presenter**
  - Responds to View requests
  - May knows when data is updated
  - Updates the View with data from the Model

Consequences

- Each class is testable in isolation of the others
- Clear Separation of Concerns
- Each collaborator has a Single Responsibility
- Far more readable and maintainable than finding business logic
  - In WebForms code-behind files
  - In ASPX code-behind files

Known Uses

- Pluralsight.com
- MVP Frameworks for .NET
  - Claymore
  - MVC# Framework
  - Web Client Software Factory
  - Evolution.Net MVP Framework
  - ASP.NET Web Forms Model-View-Presenter
- Many, many WebForms websites
- Many, many Windows Forms applications

Related Patterns

- Model View Controller
- Model-View ViewModel
- Supervising Controller
- Passive View

Summary

- MVP has many variations and a long history
- Use it to create clean UI code
- All MVP collaborators are in the UI Layer
- Domain classes, services, and DAL do work under the UI
- There are different opinions on MVP implementations