# MVC and MVP and MVVM

In MVC, View knows the Model.

In WinForm app, the Winform itself it the View. The event handler code (let's say the .cs file instead of the .design.cs file) is the controller.

An alternative of MVC in Winform is a project in CodeComplex called MVCWin. Yet I think it is a little complex for Winform.

The Controller gets data from Model and sets them as session data, the View gets the session data for display. The Controller and the View holds the same session. When view changes, the view calls the Controller's method by reflection and the Controller then update the Model;  when the model changes, the Controller prepare the data and set them as session data and refresh the view, then the view displays using the session data.

In ASP.NET, tag file (e.g. aspx file) is for display and is the View. The code behind is the controller; what the controller calls is the Model. By the way, the asp.net runs in a route way instead of post way.

in MVP, View doesn't know the Model, the Presenter is the only bridge.

In Model-View-ViewModel

The **Model View ViewModel (MVVM)** is an architectural pattern used in software engineering that originated from Microsoft as a specialization of the [PresentationModel](http://en.wikipedia.org/w/index.php?title=PresentationModel&action=edit&redlink=1) design pattern.

The View-Model of MVVM is “basically a value converter on steroids” [[2]](http://en.wikipedia.org/wiki/Model-view-viewmodel#cite_note-1) meaning that the View-Model is responsible for exposing the data objects from the Model in such a way that those objects are easily managed and consumed. In this respect, the View-Model is more Model than View, and handles most if not all of the View’s display logic (though the demarcation between what functions are handled by which layer is a subject of ongoing discussion [[3]](http://en.wikipedia.org/wiki/Model-view-viewmodel#cite_note-2) and exploration).

**Model:** as in the classic MVC pattern, the model refers to the data access layer representing the real state content.

**View:** as in the classic MVC pattern, the view refers to all elements displayed by the GUI such as buttons, windows, graphics, and other controls.

**ViewModel:** the ViewModel is a “Model of the View” meaning it is an abstraction of the View that also serves in data binding between the View and the Model. It could be seen as a specialized aspect of what would be a Controller (in the MVC pattern) that acts as a data binder/converter that changes Model information into View information and passes commands from the View into the Model. The ViewModel exposes public properties, commands, and abstractions. The ViewModel has been likened to a conceptual state of the data as opposed to the real state of the data in the Model. [[7]](http://en.wikipedia.org/wiki/Model_View_ViewModel#cite_note-6)