

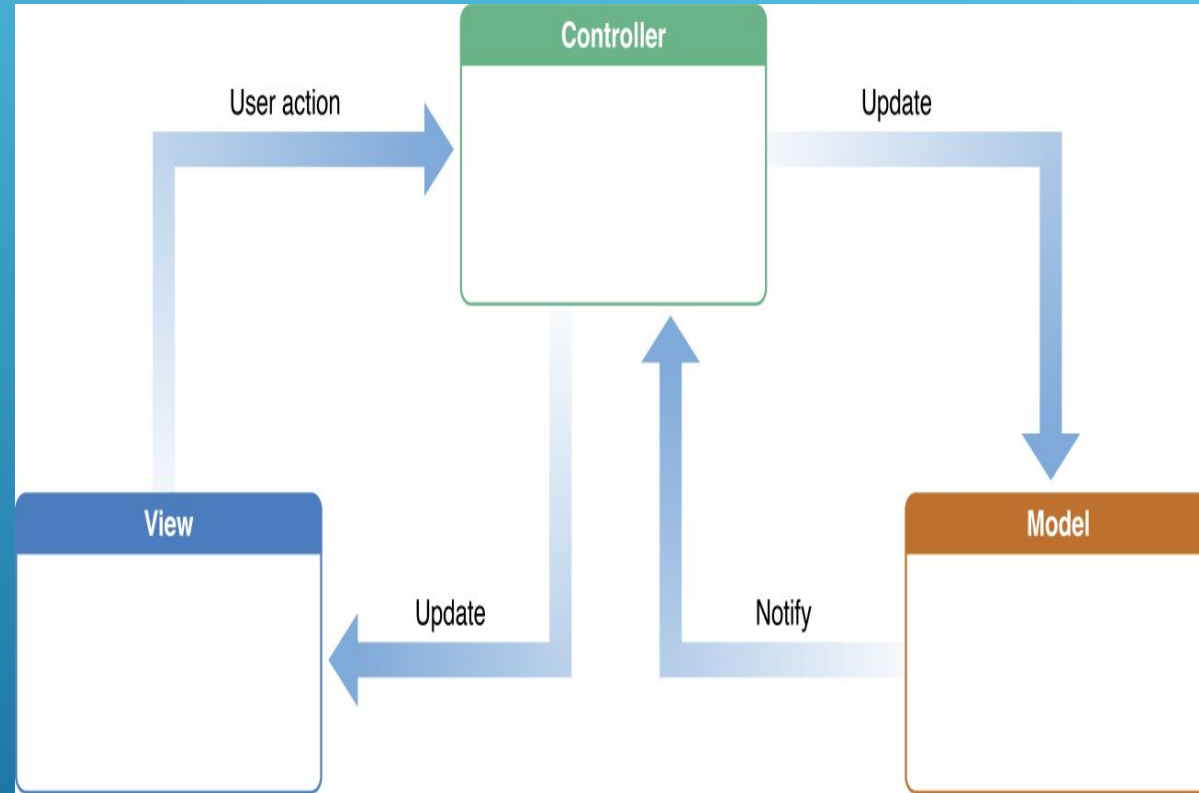
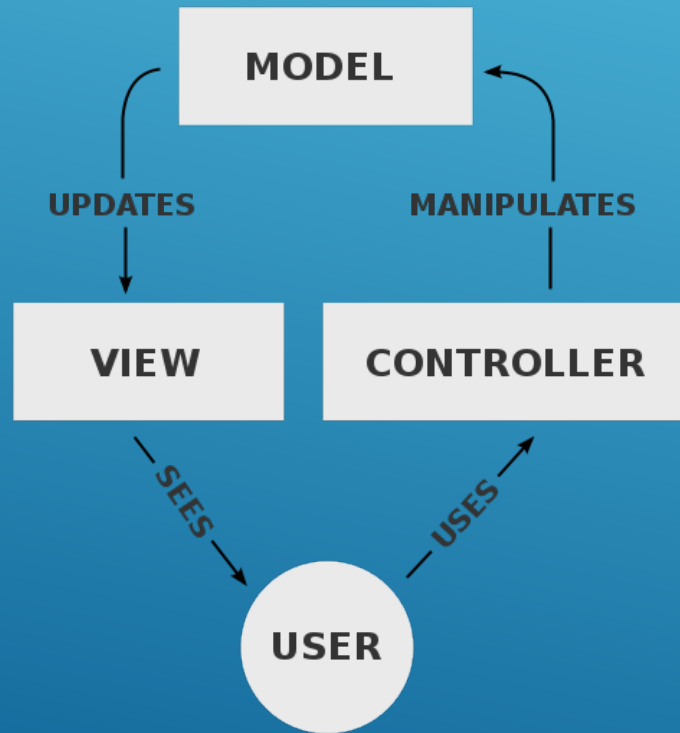
# MODEL VIEW CONTROLLER (MVC)

By

Jonathan Neronde 040172018

Ittipon Wonglertviriya 040035349

Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given application into three interconnected parts : Model object, view object, and Controller object.



# MODEL OBJECT

Model objects encapsulate the data specific to an application and define the logic and computation that manipulate and process that data. For example, a model object might represent a character in a game or a contact in an address book. A model object can have to-one and to-many relationships with other model objects, and so sometimes the model layer of an application effectively is one or more object graphs. Much of the data that is part of the persistent state of the application (whether that persistent state is stored in files or databases) should reside in the model objects after the data is loaded into the application. Because model objects represent knowledge and expertise related to a specific problem domain, they can be reused in similar problem domains. Ideally, a model object should have no explicit connection to the view objects that present its data and allow users to edit that data—it should not be concerned with user-interface and presentation issues.



# VIEW OBJECTS

A view object is an object in an application that users can see. A view object knows how to draw itself and can respond to user actions. A major purpose of view objects is to display data from the application's model objects and to enable the editing of that data. Despite this, view objects are typically decoupled from model objects in an MVC application.



# CONTROLLER OBJECTS

A controller object acts as an intermediary between one or more of an application's view objects and one or more of its model objects. Controller objects are thus a conduit through which view objects learn about changes in model objects and vice versa. Controller objects can also perform setup and coordinating tasks for an application and manage the life cycles of other objects.

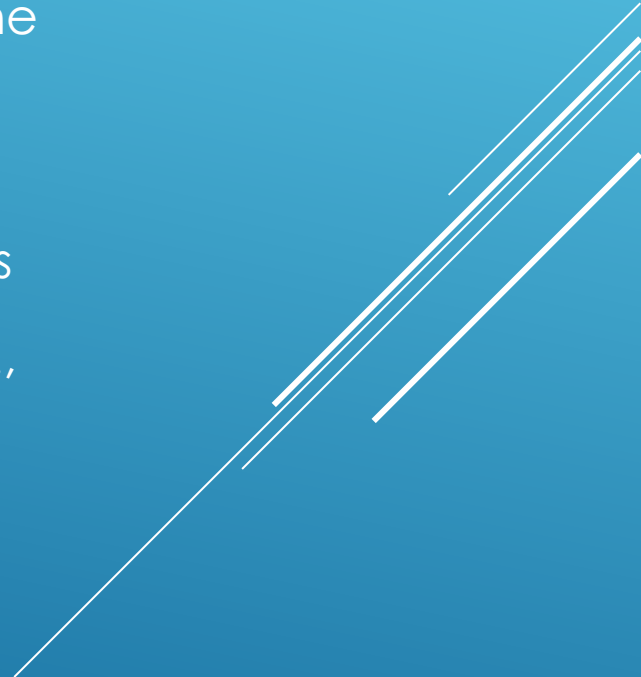


# COMPONENTS

- ▶ The *model* is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface.<sup>1</sup>It directly manages the data, logic and rules of the application.
- ▶ A *view* can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
- ▶ The third part, the *controller*, accepts input and converts it to commands for the model or view.



# INTERACTIONS

- ▶ In addition to dividing the application into three kinds of components, the model–view–controller design defines the interactions between them.
  - ▶ A *model* stores data that is retrieved according to commands from the controller and displayed in the view.
  - ▶ A *view* generates new output to the user based on changes in the model.
  - ▶ A *controller* can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., scrolling through a document, movement of document)
- 

# GOALS OF MVC

- ▶ Simultaneous development
  - ▶ Code reuse
- 
- A series of several parallel white diagonal lines of varying lengths, located in the bottom right corner of the slide, extending from the right edge towards the center.



# SIMULTANEOUS DEVELOPMENT

Because MVC decouples the various components of an application, developers are able to work in parallel on different components without impacting or blocking one another. For example, a team might divide their developers between the front-end and the back-end. The back-end developers can design the structure of the data and how the user interacts with it without requiring the user interface to be completed. Conversely, the front-end developers are able to design and test the layout of the application prior to the data structure being available.



# CODE REUSE

By creating components that are independent of one another, developers are able to reuse components quickly and easily in other applications. The same (or similar) view for one application can be refactored for another application with different data because the view is simply handling how the data is being displayed to the user.



# REFERENCES

- ▶ Burbeck (1992): "... the user input, the modeling of the external world, and the visual feedback to the user are explicitly separated and handled by three types of object."
- ▶ Buschmann, Frank (1996) *Pattern-Oriented Software Architecture*.
- ▶ Krasner, Glenn E.; Pope, Stephen T. (Aug-Sep 1988). "A cookbook for using the model-view controller user interface paradigm in Smalltalk-80". The Journal of Object Technology. SIGS Publications. Also published as "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System" (Report), ParcPlace Systems; Retrieved 2012-06-05.
- ▶ Leff, Avraham; Rayfield, James T. (September 2001). *Web-Application Development Using the Model/View/Controller Design Pattern*. IEEE Enterprise Distributed Object Computing Conference.

# REFERENCES

- ▶ Moore, Dana et al. (2007) *Professional Rich Internet Applications: Ajax and Beyond*: "Since the origin of MVC, there have been many interpretations of the pattern. The concept has been adapted and applied in very different ways to a wide variety of systems and architectures."
- ▶ Davis, Ian. "What Are The Benefits of MVC?". Internet Alchemy. Retrieved 2016-11-29

THAK YOU

