Michael Weyman Design Patterns

Design patterns are a formal way of documenting a solution to a design problem in a particular field of expertise. Design patterns exist to provide solutions to common problems in OOP that are easy to use and understand. Design patterns are split into three categories: Creational, Structural, and Behavioral.

Creational Design patterns are design patterns that create objects for you. Using a creational design pattern gives you more flexibility in deciding which objects are needed for a given case. The Creational Design patterns are: Abstract Factory, Builder, Factory Method, Prototype, and Singleton. Two of these patterns are Builder which constructs complex objects. It constructs complex objects by separating construction and representation. The Prototype pattern creates objects by cloning an existing object.

Structural Design patterns use inheritance to compose interfaces and define ways to compose objects to obtain new functionality. Structural Design patterns concern class and object composition. The Structural Design Patterns are: Adapter, Bridge, Composite, Decorator, Façade, Flyweight, and Proxy. Two of the Structural Patterns are Bridge which decouples an abstraction from its implementation so that the two can vary independently. The Flyweight pattern reduces the cost of creating and manipulating a large number of similar objects.

Finally Behavioral Design patterns are patterns specifically concerned with communication between objects. The List of Behavioral design patterns are: Chain of responsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template method, and Visitor. Two of these patterns are Mediator which allows loose coupling between classes by being the only class that has detailed knowledge of their methods, and Template method which defines the skeleton of an algorithm as an abstract class, allowing its subclasses to provide concrete behavior.

Modern View Controller is a software pattern for implementing user interfaces. This extremely popular method divides the software into three parts in order to separate internal representations of information from the ways that information is presented to the user. The three parts are the controller, the model, and the view. The Controller accepts input and converts it to commands for the model or view. The controller can send commands to the model in order to update the model’s state; it will also send commands to the view in order to change what the view is displaying.

The Model is the central component in the pattern and consists of application data, business rules, logic, and functions. The model’s job is to notify its associated views and controllers when there has been a change. This allows for the views to keep producing updated output, and allows the controllers to change the available set of commands.

A view is represented by any type of information output, including charts and diagrams. A program may even have multiple views of the same information. The view requests information from the model that it needs for generating an output representation to the user.

The Model View Controller solves many problems compared to the other design patterns. MVC is less complex than most software. Well defined chunks of code are easier to view and understand than thousands of lines of jumbled code. It allows for easier testing of your code, making debugging more time efficient. Finally MVC gives you a greater flexibility in designing your software. Since the code is split into chucks, switching out will take significantly less work.

The Model View Controller was originally developed for personal computing, but now MVC has been widely adopted as a pattern for the WWW (World Wide Web). It fits the WWW well because it allows for a hyperlink to be sent out and request a form displaying a web page. The model exists entirely on the server instead of on each individual PC.