APP Architectures

MVC, Apple’s MVC, MVP, MVVM, and Viper

Model View Controller is the traditional MVC. Within its architecture all three are constantly aware of what each other aspect is doing. This might not be a good thing since that would take up extra memory space, and require more coding. It may not be smart to have each piece constantly aware of what each other is doing. This also blurs the lines of what MVC is supposed to do, keeping each part separate from each other. They each have their own job.

Apple’s MVC somewhat takes care of this aspect by having some separation. The controller in their model is the mediator between the view and the model. This type of model isn’t very testable.

MVP is basically apple’s mvc but with a presenter. This presenter has nothing to do with the overall lifecycle of the view controller, but is instead responsible for the updates.

MVVM is the greatest of the MV(X) models, since it actually treats the view controller as the view controller. There is no longer any coupling between the view and the model. MVVM implements binding between the view and the view model to help keep each piece more separate. Thus resulting in better testability.

Lastly we have VIPER, which is much different than the prior architecture models since it has 5 layers. It has a router, presenter, view, interactor, and entity all working together.

Singleton, one of the creational architecturess ensures that the design pattern restricts object instantiation to only one. Sounds good at first until one looks further into it. Saving that information in one place might not be beneficial in the long run since it isn’t thread safe. It is mainly used for logging, driver objects, caching, and thread pool database connections.

Factory Method is another one of hte creational architectures. It creates objects without specifying the exact class to create. Its main purpose is to create objects without exposing the creation logic to users.

Prototype, creates new objects from an existing object. This approach saves resources and time by being reusable code. It has some advantages and disadvantages. A few advantages would be reduced subclassing, and being able to add and remove products at run time. One of the disadvantages of its design pattern is that it can be a bit overkill for small projects

Facade, is a structural architecture that is good to used for complex systems. It mainly deals with the interfaces, and hiding internal complexity behind a single interface.

Decorator allows for an objects behavior to be extended dynamically at run time. It’s classes should be open for extension and closed for modification. Any changes can lead to an alteration in the existing code

Adapter allows for two incompatible classes to work together by wrapping an interface around one of the existing classes. When two are incompatible it wraps an adapter around them.