Layering of Code

Topics of discussion:

**What code layering is**

* The definition of application layering
  + Dividing specific processes of an application into separate lateral segments
* Derived from N-Tier Architecture
  + Client-server hardware centered stages of an application
  + Three-Tier Architecture
* The difference between a tier and a layer
  + A layer is a logical structuring mechanism for elements of a software solution
  + A tier is a physical structuring mechanism for the system infrastructure

**Solid Principles**

* S.O.L.I.D. object-orientated design
  + introduced by Robert C. Martin
* Single Responsibility Principle
  + A responsibility is a reason to change (a model or business object)
  + correlates to separation of concerns pattern
  + A concern is any piece of interest or focus in a program
  + Loosely interpreted
* Liskov’s Substitution principle
  + ‘substitutability’ and behavioral subtypes
  + Allows for base classes and base functionality
  + Enables objects to ‘Pass through’ layers
* Interface Segregation Principle
  + Interface contracts
  + High Cohesion
  + Decoupling
  + Easily refactored

**How layering can improve an application**

* STOP SPAGHETTI CODE!!!
  + How many people got lost in an application?
* Code coordinate system
  + Cross section between object and task
* Happy Developers ☺
  + High Manageability, easy maintenance

**(Demonstrate simple architecture code sample here…)**

**Simple layering architecture**

* Domain Layer
  + Defines the business objects
* Repository Layer
  + Queries and persists business objects to a data source(s)
* Service Layer
  + Performs business logic on transient objects
* View Layer
  + Displays objects to the user

**Advanced layering architecture**

* MVVM design pattern
* View Model layer
  + A view model is tightly coupled to a user interface view
* Mapping layer
  + Converts domain models to view models (sometimes uses DTOs)
* View Model Manager layer
  + Queries and persists domain objects linked to view models
* Controller layer
  + Routes the user from view to view
  + Produces view model to be bound to a view
* View layer
  + Displays the UI to the user
  + Can use templates and consist of many sub-layers

**Summary**

* The purpose of layers in code
  + Coordinate system (object - action)
* The basic layers
* How to decide when and where to create a layer of code
  + Whenever a specific operation can be applied laterally across most objects
* Use layers to simplify code, not complicate it!
  + Still employ common sense