

SOLID SOFTWARE & DESIGN PATTERNS FOR MERE MORTALS

Philip Japikse (@skimedic)

phil.japikse@telerik.com

www.skimedic.com/blog

MVP, MCSD.Net, MCDBA, CSM, CSP
Patterns & Practices Evangelist, Telerik



TELERIK

DELIVER MORE THAN EXPECTED

One-stop shop for .NET tools & UI components



UI Components

Dev Tools

QA/PM Tools



WHO AM I?

- Patterns & Practices Evangelist, Telerik, Inc.
- Microsoft MVP
- MCSD, MCDBA, CSM, CSP
- Developer().Speaker().Writer()
- Lead Director, Cincinnati .NET User's Group
- Founder, Agile Conferences, Inc.
- Columnist, Developer.com



SOLID

- S – Single Responsibility Principle
 - There should never be more than one reason for a class or method to change
- O – Open/Closed Principle
 - Classes should be Open for extension, Closed for modification
- L - Liskov Substitution Principle
 - Derived classes should be substitutable for their base classes
- I - Interface Segregation Principle
 - Make fine grained interfaces that are client specific
- D - Dependency Inversion Principle
 - Depend on abstractions, not concrete implementations



“The goal is not to bend developers to the will of some specific patterns, but to get them to think about their work and what they are doing”

--Phil Haack



WHAT ARE DESIGN PATTERNS?

- General Reusable Solutions To A Common Problem
- Conceptual
- Defined by Purpose and Structure
- Method of Communication
- Support SOLID development
- NOT CODE!



TYPES OF DESIGN PATTERNS

- Creational
 - Deal with instantiation of objects (Singleton, Factories)
- Structural
 - Deal with Composition and Relations (Adapter, Decorator, Façade)
- Behavioral
 - Deal with responsibilities and communication between objects (Command, Strategy)



CREATIONAL

- Singleton
 - Ensures class has only one instance with a single access point
- Simple Factory (Not a “true” pattern)
 - Encapsulates Object Creation in one place
- Factory Method
 - Provides an interface to allow subclasses to determine what is instantiated
- Abstract Factory
 - Provides an interface for creating families of related objects without specifying their concrete class





DEMO

Singleton, Simple Factory, Factory Method,
Abstract Factory



STRUCTURAL

- Adapter
 - Converts the interface of a class into another interface the client expects
- Façade
 - Provides a unified interface to a set of interfaces
- Decorator
 - Attaches additional responsibilities to an object at runtime





Adapter, Façade, Decorator



BEHAVIORAL

- Command
 - Encapsulates a request as an object
- Strategy
 - Encapsulates an algorithm inside a class





DEMO

Command, Strategy



RESOURCES

- “Design Patterns: Elements of Reusable Object Oriented Design”
 - Eric Gamma, Richard Helm, Ralph Johnson, John Vlissides
- “Head First Design Patterns”
- www.dofactory.com



CONTACT ME

- phil@telerik.com
- www.skimedic.com/blog
- www.twitter.com/skimedic
- www.tinyurl.com/teleriktour

- Telerik
- www.twitter.com/Telerik
- www.facebook.com/Telerik

