## SOLID PRINCIPLES

SOLID are five basic principles whichhelp to create good software architecture. SOLID is an acronym where:-

* S stands for SRP (Single responsibility principle
* stands for OCP (Open closed principle)
* L stands for LSP (Liskov substitution principle)
* I stands for ISP ( Interface segregation principle)
* D stands for DIP ( Dependency inversion principle)

## What are Design Patterns in Software Programming?

Design Patterns in object oriented world is reusable solution to common software design problems which occur again and again in real world application development. It is a template or description for how to solve a problem which can be used in many different situations.

"A pattern is a recurring solution to a problem in a context."

"Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice."

-- Christopher Alexander - A Pattern Language

Patterns are used by developers to their particular design to solve their problems. Patterns usage and choice to choose among different design patterns is based on individual need and their problem.

Design patterns are the most powerful tool for software developer. It is important to understand design patterns rather than memorizing its classes, methods and properties. It is also important to learn how to apply pattern to specific problem to get the desired result. This will be required continuous practice of using and applying design patterns in day to day software development. First, identify the software design problem, then see how to address these problems using design patterns and find out the best suited design problem to solve the problem.

There are 23 design patterns also known as Gang of four design patterns (GoF). Gang of four are the authors of the book, “Design Patterns: Elements of Reusable Object Oriented Software”. These 23 patterns are grouped into three main categories based on their:

* Creational Design Pattern
  1. **Abstract Factory**:- Creates an instance of several families of classes
  2. **Builder**: - Separates object construction from its representation
  3. **Factory Method**:- Creates an instance of several derived classes
  4. **Prototype**:- A fully initialized instance to be copied or cloned
  5. **Singleton**:- A class in which only a single instance can exist

**Note**: - The best way to remember Creational pattern is by remembering ABFPS (Abraham Became First President of States).

* Structural Design Patterns

• **Adapter**:-Match interfaces of different classes.   
• **Bridge**:-Separates an object’s abstraction from its implementation.   
• **Composite**:-A tree structure of simple and composite objects.   
• **Decorator**:-Add responsibilities to objects dynamically.   
• **Façade**:-A single class that represents an entire subsystem.  
• **Flyweight**:-A fine-grained instance used for efficient sharing.   
• **Proxy**:-An object representing another object.

* Behavioral Design Patterns

• **Mediator**:-Defines simplified communication between classes.  
• **Memento**:-Capture and restore an object's internal state.   
• **Interpreter**:- A way to include language elements in a program.  
• **Iterator**:-Sequentially access the elements of a collection.   
• **Chain of Resp**: - A way of passing a request between a chain of objects.  
• **Command**:-Encapsulate a command request as an object.   
• **State**:-Alter an object's behavior when its state changes.   
• **Strategy**:-Encapsulates an algorithm inside a class.   
• **Observer**: - A way of notifying change to a number of classes.   
• **Template Method**:-Defer the exact steps of an algorithm to a subclass.   
• **Visitor**:-Defines a new operation to a class without change.

Singleton Design Pattern in Asp.net using C#

When we want to make a only one instance of a class and also making sure that there is a global access point to that object then the design pattern we user is called Singleton. The pattern ensures that the class is instantiated only once and that all requests are directed to that one and only object.

In asp.net you can use singleton through sessions or application variables easily however i will show you how to implement singleton object to User class.

In Asp.net to get user information like first name, surname or username through Memebership object does not look neat because to get UserID first you need to know user User.Identity.Name then use Membership.GetUser in order to get more detail for user.

I have created a interface called IUser which you can use anywhere on the page.

public interface IUser

{

string Email { get; set; }

string Firstname { get; set; }

string Surname { get; set; }

string UserID { get; set; }

string Username { get; set; }

}

Here is our Singleton class implementing IUser.

public sealed class Singleton

{

private const string IUserSessionName = "User";

private static readonly object padlock = new object();

private static IUser objUser ;

Singleton()

{

}

/// *<span* *class="code-SummaryComment"><summary>*

</span> /// Loading user information

/// *<span* *class="code-SummaryComment"></summary>*

</span> public static IUser IUserInstance

{

get

{

lock (padlock)

{

if (null == Session[IUserSessionName])

{

string userID = Membership.GetUser().ProviderUserKey.ToString();

// getting userinformation from database

objUser = new UserClass().GetUserInformation(userID);

Session[IUserSessionName] = objUser;

}

else

{

objUser = (IUser)Current.Session[IUserSessionName];

}

return objUser;

}

}

set

{

Session[IUserSessionName] = value;

}

}

public static void UserInstanceFlush()

{

Session[IUserSessionName] = null;

}

}

Because of IuserSessionName you are also making sure that you will keep this session name unique and at only one place.

IUser is a static property which can be used on any page from Singleton.IUserInstance.UserID.

UserInstanceFlush flushes the session if you want to remove the session value.