Go

C# : .NET : Class

.NET **Array** **Dictionary** **List** **String** 2D Async DataTable Dates DateTime Enum File For Foreach Format IEnumerable If IndexOf Lambda LINQ Parse Path Process Property Random Regex Replace Sort Split Static Substring Switch Tuple While

Factory Pattern. A factory creates objects. We implement the factory design pattern in a C# program. With this pattern, we develop an abstraction that isolates the logic for determining which type of class to create.

Object

Example. The factory design pattern relies on a type hierarchy. The classes must all implement an interface or derive from a base class. We use an abstract class as the base. The Manager, Clerk and Programmer classes derive from Position.

Interface

Abstract

```
    Based on:     .NET 4.5

    Factory pattern example: C#

using System;

class Program
```

```
{
    abstract class Position
    {
        public abstract string Title { get; }
    }

    class Manager : Position
    {
        public override string Title
        {
            get
            {
                return "Manager";
            }
        }
    }

    class Clerk : Position
    {
        public override string Title
        {
            get
            {
                return "Clerk";
            }
        }
    }

    class Programmer : Position
    {
        public override string Title
        {
            get
            {
                return "Programmer";
            }
        }
    }

    static class Factory
    {
        /// <summary>
        /// Decides which class to instantiate.
        /// </summary>
        public static Position Get(int id)
        {
            switch (id)
            {
```

```
                case 0:
                    return new Manager();
                case 1:
                case 2:
                    return new Clerk();
                case 3:
                default:
                    return new Programmer();
            }
        }
    }

    static void Main()
    {
        for (int i = 0; i <= 3; i++)
        {
            var position = Factory.Get(i);
            Console.WriteLine("Where id = {0}, position = {1} ", i,
position.Title);
        }
    }
}
```

Output

```
Where id = 0, position = Manager
Where id = 1, position = Clerk
Where id = 2, position = Clerk
Where id = 3, position = Programmer
```

The factory design pattern is found in the Factory class. The point of the Get method is to take a value and instantiate a class based on that value. It translates integers to objects with a switch statement.

Switch

Also:

Because Manager, Clerk, and Programmer all derive from the same abstract class, the return type Position can be used.

And:

An implicit cast automatically casts the Manager, Clerk and Programmer to

Position references.

The Main method serves to demonstrate the Factory class in action. We use as part of the demonstration the integers 0, 1, 2, and 3. We use the Get method with each of these values.

Then:

We show that the appropriate type of class was instantiated for each integer.

Discussion. Imagine you have a system that needs to create objects in many different places. Suppose the system has integers and you want objects for those integers. The factory pattern is ideal for this usage.

Tip:

You can use the Factory type to handle object creation in a uniform and concise way.

Summary. We looked the factory design pattern, which is used to instantiate objects based on another data type such as integers. Factories can be used to reduce code bloat and also make it easier to modify which objects need to be created.