C SHARP GENERIC DELEGATES FUNC, ACTION AND PREDICATE WITH ANONYMOUS METHOD

In .net 3.5 some new generic delegates -Func<T>, Action<T> and Predicate<T> were introduced. Using generic delegates, it is possible to concise delegate type means you don't have to define the delegate statement. These delegates are the Func<T>, Action<T> and Predicate<T> delegates and defined in the System namespace.

Action<T> performs an operation on the generic arguments. Func<T> performs an operationon the argument(s) and returns a value, and Predicate<T> is used to represent a set of criteria and determine if the argument matches the criteria.

```
1.  delegate TResult Func  ();
2. delegate TResult Func  (T arg);
3. delegate TResult Func  (T1 arg1, T2 arg2);
4. ... up to T16
5.  delegate void Action ();
6. delegate void Action  (T arg);
7. delegate void Action  (T1 arg1, T2 arg2);
8. ... up to T16
```

Here "in" shows the input parameters and "out" shows the return value by the delegate.

Generic delegate example

```
1.  using System;
2. class demo
3. {
4. delegate void MyDelegate(string str);
5. static void Main(string[] args)
6. {
7. MyDelegate d = show;
8. d("Hello World!");
9. Console.ReadLine();
10.   }
11.  static void show(string str)
```

```
12.  {
13.  Console.WriteLine(str);
14.  }
15.  }
```

Above code can be written as using generic delegate.

```
1.   using System;
2. class demo
3. {
4. static void Main(string[] args)
5. {
6. Action<string> d = show;
7. d("Hello World!");
8. Console.ReadLine();
9. }
10.  static void show(string str)
11.  {
12.  Console.WriteLine(str);
13.  }
14.  }
```

Generic delegate using anonymous method

```
1.   using System;
2. class demo
3. {
4. static void Main(string[] args)
5. {
6. Action<string> d = s => Console.WriteLine(s);
7. d("Hello World!");
8. }
9. }
```

Source : http://www.dotnet-tricks.com/Tutorial/csharp/67aF200612-C-Sharp-Generic-delegates-Func,-Action-and-Predicate-with-anonymous-method.html