Generic Delegates in C#

What are Generic Delegates in C#?

The Generic Delegates in C# were introduced as part of .NET Framework 3.5 which doesn't require defining the delegate instance in order to invoke the methods.

Types of Generic Delegates in C#

C# provides three built-in generic delegates, they are as follows:

1. **Func**
2. **Action**
3. **Predicate**
4.

Why do we need the Generic Delegates in C#?

In order to understand need for Generic Delegates in C#, let us first understand how we use delegates to invoke methods. Let us say we have the following three methods and we want to invoke these methods using delegates.

```
public static double AddNumber1(int no1, float no2, double no3)
{
    return no1 + no2 + no3;
}

public static void AddNumber2(int no1, float no2, double no3)
{
    Console.WriteLine(no1 + no2 + no3);
}

public static bool CheckLength(string name)
{
    if (name.Length > 5)
        return true;
    return false;
}
```

As you can see in the above code, the **AddNumber1** method takes three parameters and returns a value of double type. Similarly, the **AddNumber2** method takes three parameters but it does not return any value and here the return type is void. The third method i.e. the **CheckLength** method takes one string parameter and returns a Boolean value. If the string length is greater than 5, then it will return true else it will return false.

Now, if we want to invoke the above three methods using delegates in C#, then we need to create three delegates whose signatures should match the signatures of the above three methods as shown in the below image.

```
public delegate double AddNumber1Delegate(int no1, float no2, double no3);
public delegate void AddNumber2Delegate(int no1, float no2, double no3);
public delegate bool CheckLengthDelegate(string name);
```

As you can see in the above image, we create three delegates. Now, once we have created the delegates. then we can invoke the methods by creating instances of each delegate referring to the respective methods and then we can invoke the delegate as shown in the below code.

```
AddNumber1Delegate obj1 = new AddNumber1Delegate(AddNumber1);
double Result = obj1.Invoke(100, 125.45f, 456.789);
Console.WriteLine(Result);

AddNumber2Delegate obj2 = new AddNumber2Delegate(AddNumber2);
obj2.Invoke(50, 255.45f, 123.456);

CheckLengthDelegate obj3 = new CheckLengthDelegate(CheckLength);
bool Status = obj3.Invoke("Pranaya");
Console.WriteLine(Status);
```

Do we really need to create Custom Delegates to Invoke Methods in C#?

The answer is no. C#.NET Framework provides some generic delegates who can do the job for us. C# provides three Generic Delegates; they are as follows

1. **Func**
2. **Action**
3. **Predicate**

Now, let us proceed and try to understand all the above three generic delegates. Let us try to understand what are all these delegates, when, and how to use all these generic delegates in C# with examples.

What is Func Generic Delegate in C#?

The Func Generic Delegate in C# is present in the System namespace. This delegate takes one or more input parameters and returns one out parameter. The last parameter is considered as the return value. The Func Generic Delegate in C# can take up to 16 input parameters of different or the same data types. It must have one return type. The return type is mandatory but the input parameter is not mandatory.

Note: Whenever your delegate returns some value, whether by taking any input parameter or not, you need to use the Func Generic delegate in C#.

What is Action Generic Delegate in C#?

The Action Generic Delegate in C# is also present in the System namespace. It takes one or more input parameters and returns nothing. This delegate can take a maximum of 16 input parameters of the different or same data types.

Note: Whenever your delegate does not return any value, whether by taking any input parameter or not, then you need to use the Action Generic delegate in C#.

What is Predicate Generic Delegate in C#?

The Predicate Generic Delegate in C# is also present in the System namespace. This delegate is used to verify certain criteria of the method and returns the output as Boolean, either True or False. It takes one input parameter and always returns a Boolean value which is mandatory. This delegate can take a maximum of 1 input parameter and always return the value of the Boolean type.

Note: Whenever your delegate returns a Boolean value, by taking only one input parameter, then you need to use the Predicate Generic delegate in C#.