# 31. C# Delegate(델리게이트), C#델리게이트체인,delegate chain, 버블정렬,Bubble Sort

오라클자바 0 4,309 2014.02.16 08:43

**C# Delegate(델리게이트), C#델리게이트체인,delegate chain, 버블정렬,Bubble Sort**

델리게이트(Deligate)는 대리자라고 하는데 C의 함수 포인터와 유사한데 메소드의 참조를 포함한다. 즉, 자기 자신이 실제로 하는 일은 없고 단지 자기가 가리키고 있는 메서드(함수)를 호출하는 역할을 하는 것이다.

결국 델리게이트(Delegate)는 메소드에 대한 참조를 저장 하는 것인데 이렇게 메소드에 대한 참조를 가리키고 있다는 것으로 인해 불가능한 작업등이 가능해진다. 예를 들면 Delegete를 다른 함수의 인자로 넘겨주게 되면 그 함수는 델리게이트가 보내 주는 함수의 참조를 이용하여 실행 시점에 호출될 함수를 결정 할 수 있는 것이다.

델리게이트의 이해를 위해서는 콜백(CallBack)에 대해 알아야 하는데 어떤 일을 해줄 코드를 두고 실제 할 일은 컴파일 타임이 아닌 런타임에 부여하는 데 이 “일”을 콜백 이라 한다. C# 델리게이트는 이 콜백을 구현하기 위해 사용되는 것인 데 델리게이트에 메소드의 주소를 할당한 후 델리게이트를 호출하면 델리게이트가 메소드를 호출해 주는 것이다.

델리게이트는 쓰레드와 이벤트에서 주로 이용 할 수 있다는 것은 참고로 알아 두자.

선언방법

한정자 delegate 리턴형 델리게이트이름 ( 매개변수목록 );

델리게이트는 메소드에 대한 참조이므로 참조할 메소드의 반환형식과 매개변수를 정의해 주어야 한다. 구현부가 없는 이유는 delegate 자신은 아무것도 하지 않고, 자신이 참조하고 있는 메서드(함수)만 호출해 주기 때문 이다.

델리게이트를 이용하여 콜백을 구현하는 과정

1.     델리게이트 선언

2.     델리게이트의 인스턴스 생성(델리게이트가 참조 할 메소드를 매개변수로 넘긴다)

3.     델리게이트를 호출

아래의 코드를 보도록 하자…

int i = int.Parse(“5919”);

일반적으로 대부분의 메서드들이 인자 값을 받아 어떤 일 처리를 하게 된다. 위의 코드에서 Parse()메서드는 string형의 인자를 받아 int형으로 바꾸어 주는 역할을 하는데, 대부분의 메서드들이 인자를 받는 이유는 왜일까?

메서드들이 인자를 받는 이유는 처리해야 할 값을 runtime시에만 알 수 있기 때문이다. 예를 들어 숫자를 정렬하는 메서드를 만들었다고 가정하자  사용자가 숫자 몇에서 몇까지를 정렬하기를 원하는지 알 수가 없다. 따라서, 나중에 그 값을 입력 받도록 처리 하기 위해 인자를 쓰게 되는 것이다.

Delegate도 이와 같은 개념으로 생각하시면 되며 단지 다른 점이 있다면, value형인 매개변수 대신에 메서드를 인자로 받는다는 것이다.

아래의 예제를 보자

**[예제1]**

using System;

public class Delegate1

{

     private delegate string OnjString();

     public static void Main(string[] args)

     {

            int x = 4790;

            OnjString myMethod = new OnjString(x.ToString);

            Console.WriteLine("문자열 : {0}", myMethod());

      }

}

Delegate를 사용할 때 주의할 점은 delegate가 참조하는 메서드(여기서는 ToString) 의 인자 및 리턴형이 선언된 delegate와 같아야 한다는 것이다. 예제 1에서도 볼 수 있듯이, ToString() 메서드는 인자가 없고, string형을 리턴 한다. 따라서 선언된 delegate도 인자가 없이 string형을 리턴 하도록 선언되어 있지만 delegate는 참조하는 메서드가 어떤 타입이든 상관 하지 않는다. static이든 non static이든 관계없다. 즉, 참조하는 메서드의 인자형, 개수, 리턴 타입만 같으면 문제가 없다는 것 이다.

**[예제2]**

using System;

class OnjMath

{

    internal static double MultipleByTwo(double value)

    {

        return value \* 2;

    }

    internal static double Square(double value)

    {

        return value \* value;

    }

}

delegate double OnjDouble(double x);

class DelegateTest2

{

    static void Main(string[] args)

    {

        //Delegate를 배열로 지정

        OnjDouble[] op = {

new OnjDouble(OnjMath.MultipleByTwo),

                         new OnjDouble(OnjMath.Square)

                  };

        for (int i = 0; i < op.Length; i++)

        {

            Console.WriteLine("op[{0}] 호출:", i);

**CallDelegate**(op[i], 3.0);  //Delegate를 다른 메소드의 인자로 넘긴다.

            //Console.WriteLine("결과는 {0} 이다", op[i](3.0));

            Console.WriteLine();

        }

    }

    static void **CallDelegate**(OnjDouble func, double value)

    {

        //넘겨받은 Delegate를 실행 한다.

        double ret = func(value);

        Console.WriteLine("입력된 값은 {0}이고 결과는 {1} 이다", value, ret);

    }

}

 위의 예제에서 OnjDelegate라는 메서드를 호출할 필요 없이 곧바로 delegate가 참조하는 메서드를 실행 하려면 다음과 같이하면 된다.

op[i](2.0);

**[예제3]**

using System;

using System.Collections.Generic;

using System.Text;

namespace delegatetest

{

    delegate int Onjdelegate(int a, int b);

    {

**static** int Plus(int a, int b)

        {

            return a + b;

        }

        int Minus(int a, int b)

        {

            return a - b;

        }

        static void Main()

        {

            MainApp m = new MainApp();

            Onjdelegate CallBack = new Onjdelegate(MainApp.Plus);

            Console.WriteLine(CallBack(4,4));

            CallBack = new Onjdelegate(m.Minus);

            Console.WriteLine(CallBack(4, 4));

        }

    }

}

[결과]

8

0

이번에는 델리게이트 체인 예제이다.

**[예제4]**

using System;

using System.Collections.Generic;

using System.Text;

namespace delegatetest

{

    delegate void OnjDelegate(int a, int b);

    class MainApp

    {

        static void Plus(int a, int b)

        {

            Console.WriteLine("{0} + {1} = {2}", a, b, a+b);

        }

        static void Minus(int a, int b)

        {

            Console.WriteLine("{0} - {1} = {2}", a, b, a - b);

        }

        void Multiplication(int a, int b)

        {

            Console.WriteLine("{0} \* {1} = {2}", a, b, a \* b);

        }

        void Division(int a, int b)

        {

            Console.WriteLine("{0} / {1} = {2}", a, b, a / b);

        }

        static void Main()

        {

            MainApp m = new MainApp();

            OnjDelegate CallBack = (OnjDelegate)Delegate.Combine(new OnjDelegate(MainApp.Plus),

                                                           new OnjDelegate(MainApp.Minus),

                                                           new OnjDelegate(m.Multiplication),

                                                           new OnjDelegate(m.Division));

            CallBack(4, 3);

        }

    }

}

[결과]

4 + 3 = 7

4 - 3 = 1

4 \* 3 = 12

4 / 3 = 1

델리데이트는 언제 꼭 필요할까?

아래와 같은 버블정렬이 있다고 하자.

for (int i=0; i<sortArray.Length; i++)

{

     for (int j=0; j<i; j++)

     {

        if (j > i) // 문제의 소지가 있는 부분, 현재는 오름차순

          {

               int temp = sortArray[i]; // i 와 j 바꾸기

               sortArray[i] = sortArray[j];

               sortArray[j] = temp;

          }

     }

}

만약 이 버블정렬의 데이터가 위와 같은 숫자가 아니라 객체라면 어떻게 될까? 현재와 같은 구조에선 곤란하다. 이는 사용자가 어떤 식으로 클래스나 구조체 등을 정의해 놓았는지 알 수가 없기 때문이다. 사용자에게 여러분이 만든 버블정렬 컴포넌트를 판매한다고 가정한다면, 각 사용자를 위해 다른 프로그램을 만든다는 건 말이 안 되는 이야기 이다. 이럴 때 필요한 것이 바로 delegate다!!

using System;

//우리가 배포하는 버블정렬 컴포넌트

class **BubbleSorter**

{

     //RunTime에 수행할 함수를 isConvert 라는 Delegate를 통해서 받는다.

     static public void **Sort**(object [] sortArray, CompareOp isConvert) //①

     {

         for (int i=0; i<sortArray.Length; i++)

         {

             for (int j=0; j<i; j++)

             {

                  if (isConvert(sortArray[i], sortArray[j])) //②

                  {

                      object temp = sortArray[i];

                      sortArray[i] = sortArray[j];

                      sortArray[j] = temp;

                  }

              }

         }

     }

}

//사용자 정의 클래스

class **Employee**

{

             private string name;

             private decimal salary;

             public Employee(string name, decimal salary)

             {

                           this.name = name;

                           this.salary = salary;

             }

             public override string ToString()

             {

                           return string.Format(name + ", {0:C}", salary);

             }

             //사용자 정의 클래스 비교 메서드

             public static bool isConvert(object obj1, object obj2) //③

              {

                            Employee e1 = (Employee) obj1;

                            Employee e2 = (Employee) obj2;

                            return (e1.salary > e2.salary) ? true : false;

              }

}

delegate bool CompareOp (object lhs, object rhs);

class DelegateClass

{

             static void Main(string[] args)

             {

                  Employee [] employees =

                  {

                      new Employee("순돌이", 5000),

                      new Employee("순딩이", 1500),

                      new Employee("차돌이", 3000),

                      new Employee("공돌이",1000)

                  };

                  CompareOp EmployeeCompareOp = new CompareOp(Employee.isConvert); //④

                  BubbleSorter.Sort(employees, EmployeeCompareOp); //⑤

                  for (int i=0; i<employees.Length; i++)

                      Console.WriteLine(employees[i].ToString());

             }

}

①을 보시면 우리가 배포할 버블정렬 컴포넌트를 다시 편집한 것을 보실 수 있는데 , 컴포넌트는 두 개의 인자를 받는다, 첫번째는 정렬하게 될 object(즉, 모든 데이터 타입을 받을 수 있습니다. Object는 가장 상위클래스이기 때문이다.) 타입의 데이터이고, 두번 째 인자는 CompareOp delegate 이다. 다시 정리하면, object형 데이터와 delegate를 인자로 받는 버블정렬 컴포넌트 인 것이다.

②에서는 이전 소스코드에서 문제가 되었던 부분을 수정한 것으로 단지 숫자만 비교할 수 있게 만든 것이 아니라 delegate를 통해서 사용자 정의 비교 메서드를 호출하여 값을 비교하는 것이다. 여기서는 isConvert라는 delegate를 사용하고 있는 것이다.

③은 사용자가 정의한 Employee라는 클래스를 비교하는 메서드

④에서는 CompareOp delegate가 ③에서 정의한 사용자 정의 비교 메서드를 참조하도록 하고 있다.

⑤에서는 employees라는 배열 데이터와 EmployeeCompareOp라는 delegate를 Sort() 메서드의 인자로 넘겨 정렬하고 있다.

위의 예제를 보시면, 이제 우리가 만든 버블정렬 컴포넌트는 어떤 상황에도 문제없이 사용될 수 있다는 것을 알 수 있다. 버블정렬 컴포넌트의 핵심기술(?)은 두 데이터의 값을 비교하는 것인데, 이 부분을 delegate로 바꿈으로써, 사용자가 정의할 수 있도록 만든 것이다. 그 만큼 컴포넌트 유연성 혹은 확장성이 확대된 것이다.

우리는 delegate도 배열로 선언되어 사용될 수 있음을 이전 예제에서 보았다. 하지만, 꼭 배열로 선언하지 않아도 하나의 delegate에 여러 개의 메서드를 넣을 수가 있는데 다음과 같이 쓸 수 있다.

delegate void DoubleOp(double value);

class MainEntryPoint

{

     static void Main(string[] args)

     {

          DoubleOp operations = new DoubleOp(MathOperations.MultiplyByTwo);

          operations += new DoubleOp(MathOperations.Square);

…

여러분이 원하시면, 다음과 같이 하는 것도 똑 같은 효과를 낸다.

DoubleOp operation1 = new DoubleOp(MathOperations.MultiplyByTwo);

DoubleOp operation2 = new DoubleOp(MathOperations.Square);

DoubleOp operations = operation1 + operation2;



**Torrent search results**

**Checked Sites**

**0**

No items to list   
Use the search bar above for instant results

To see search results, type here and hit `Enter`

**No results**

**Try our Pro Versions to unlock:**

Faster Results

Secure Torrenting

Unlimited Search Results with detailed torrent info

1-YR Subscription to CyberGhost VPN (PRO+VPN only)

[BUY PRO + VPNBUY PRO](https://www.utorrent.com/webpro-offer/?utm_source=Lavasoft&utm_medium=version_1.0&utm_campaign=Scanner)



One more step to go before you start torrenting!

This extension can sync results with BitTorrent and/or uTorrent for instant downloading.

To activate this feature, please click on the button below, and then on the Chrome message to activate the 'Messaging Permission'.

Activate Messaging Permission



Syncing...

Please allow Messaging Permissions in the proceeding Chrome message.



Sync Complete

You have successfully activated the “Messaging Permission” feature. All your search results will sync with BitTorrent and/or uTorrent.



Enter License Key

Enter your license key and click on the activate button to start using µTorrent Web Pro.



Activate

Don't have a license key? Click here



Happy Torrenting!

You are now an active PRO user

Your key is valid until



Your license key has expired

Looks like your license key has expired, to renew your PRO license key, please select a license type:

[Buy Pro + VPNBuy Pro](https://www.utorrent.com/webpro-offer/?utm_source=Lavasoft&utm_medium=version_1.0&utm_campaign=Scanner)

Already have a license key? Click here



Pro User

Your key is valid until

Your License Key:

Switch back to Torrent Scanner Free?

Revert to Free Version



Feedback

Help us improve Torrent Scanner, send us comments, bugs, feedback, and suggestions.

Send Feedback

**Settings**

**FAQ**

FAQ: [Click here](chrome-extension://aegnopegbbhjeeiganiajffnalhlkkjb/faq.html)

**Rate the extension**

How did you like the extension experience?

★★★★★

Submit

**About**

**Version 1.3.0   
  
What's New**

* Experience a complete new User Interface of the extension. It is enhanced and user friendly now.
* Squashed some bugs.

**Privacy Policy**

Adaware Software (7270356 Canada Inc.) is the operator of the Adaware products suites and related services (the “**Company**”, ”**we**” or “**us**”). We respect your privacy rights and we are committed to protecting them. This privacy policy (“**Privacy Policy**” or simply “**policy**”) governs our products, services and websites that link to this Privacy Policy, and describes our practices of processing data from you. By “**you**”, we refer to either or all of the following: (i) visitors to our websites that links to this Privacy Policy (“**Visitor**” and “**Website**”, respectively); (ii) our customers using our software products and Services (“**User**”); and (c) a business customer, a business partner that has a contractual relationship with us or a prospective customer that is yet to be engaged in a contract with us (“Business Customer”). Unless explicitly mentioned otherwise, the information in this Privacy Policy refers to any and all data subject types (“you” or “your’).   
  
For the purpose of this policy, the “**Service(s)**” shall include any software licensed by the Company, including features offered by or within the installed software or additional software scripts available therein (either downloaded from one of our websites, pre-installed on your device, downloaded through a third party website, obtained on a physical medium, or otherwise), or services provided through and/or on top such software, services offered on our websites, communication forums, support services, account operation, updates, enhancements, new features, premium support, extended guarantees, online version and free versions of a software or additional services or features as we ay make available from time to time.   
  
If you are a California resident, please also see our [CCPA Notice](https://www.adaware.com/CCPA/).   
  
[Read more](https://www.adaware.com/privacy-policy/)

**Contact Us**

For any payment and order-related support, please contact us at Email: [support@torrentscanner.zendesk.com](mailto:support@torrentscanner.zendesk.com) or [pcsoftwareinfo.com](mailto:pcsoftwareinfo.com)  
  
Phone: [Click here](https://pcsoftwareinfo.com/contact.aspx)

SyncLicenseHomeFeedbackSettings