.Net Garbage Collector

Nir Adar
Email: underwar@hotmail.com
Home Page: http://underwar.livedns.co.il

.Net Garbage Collector

סביבת Net. ובפרט שפת C# מציגים גישה של **זיכרון מנוהל**, בה אנו, המתכנתים, רק מקצים זיכרון דינמי, והסביבה עצמה אחראית לשחרורו.
ה-Garbage Collector **אוסף** את הזבל מהזיכרון. הוא לוקח את האובייקטים שאינם נמצאים בשימוש יותר, קורא ל-destructor שלהם ולאחר מכן משחרר את הזיכרון שהוקצה להם.

כאשר אנו אומרים ש**אובייקט אינו נמצא בשימוש יותר** אנו מתכוונים שאין אף references המצביעים אליו. (עם יוצא דופן הנקרא WeekReference המאפשר את השימוש באובייקט וגם מרשה ל-GC לאסוף את האובייקט.

בעיה ראשונה

הבעיה הראשונה שנציג היא המקרה בו אנו חושבים שהאובייקט אינו בשימוש יותר, אולם ה-GC אינו חושב כך.
נביט בקוד הבא:

```
using System;

namespace Bad
{
	class Class1
	{
		private void EventHandler(string input)
		{
			int hash=this.GetHashCode();
			System.Console.WriteLine("This is a class2 instance with hashcode={0}",hash);
			System.Console.WriteLine("I got this input in an event:\t{0}\n", input);
		}

		public Class1(Class2 c)
		{
			c.SomethingWasRead+=new Class2.SomethingWasReadHandler(EventHandler);
		}
	}

	public class Class2
	{
		public delegate void SomethingWasReadHandler(string input);
		public event SomethingWasReadHandler SomethingWasRead;

		public void RaiseThatEvent(string input)
		{
			if(SomethingWasRead!=null)
				SomethingWasRead(input);
		}
	}

	class Program
	{
		static void Main(string[] args)
		{
			Class2 EventRaiser=new Class2();
			Class1 HandlesEvent1=new Class1(EventRaiser);
			Class1 HandlesEvent2=new Class1(EventRaiser);

			System.Console.WriteLine("Please enter something");
```

```
                        string input=System.Console.ReadLine();
                        System.Console.WriteLine();

                        //raise the event again
                        EventRaiser.RaiseThatEvent(input);

                        // you probably know what result we get by now
                        // now we set the two references HandlesEvent1
                        // and HandlesEvent2 to null, so there is nothing
                        // else referencing to those objects

                        HandlesEvent1=HandlesEvent2=null; // GC will send both of them to hell, right?
                        GC.Collect(); // let's call GC.Collect to make sure

                        System.Console.WriteLine("Again please enter something");
                        input=System.Console.ReadLine();
                        System.Console.WriteLine("Press any key to quit...");

                        //raise the event again
                        EventRaiser.RaiseThatEvent(input);
                        // now we are almost sure we won't get any output
                        // since nothing else is handling the events but...
                        // "Damm , what's wrong? why are we getting results
                        // from those classes again?!"
                        // Read the following to see why.

                        System.Console.WriteLine("Press enter to quit...");
                        System.Console.ReadLine();
                }
        }
}
```

פלט לדוגמא של התוכנית:

```
Please enter something
a

This is a class2 instance with hash code=20
I got this input in an event: a

This is a class2 instance with hash code=21
I got this input in an event: a

Again please enter something
a
Press any key to quit...
This is a class2 instance with hash code=20
I got this input in an event: a

This is a class2 instance with hash code=21
I got this input in an event: a

Press enter to quit...
```

פלט זה עלול להפתיע אם לא שמים לב, כי לכאורה אחרי שהשמנו NULL במשתני ההתייחסות הם היו אמורים להיעלם.

הסיבה לתופעה שראינו היא השורה הבאה:

```
c.SomethingWasRead+=new Class2.SomethingWasReadHandler(EventHandler);
```

שורה זו גורמת ל-Class2.SomethingWasReadHandler להתייחס אל Class1.SomthingWasRead. כאלו מכיוון שיש אל Class1 התייחסות, ה-GC לא ישחרר אותו.

פתרון

הפתרון הטוב ביותר לבעיה זו הוא להגדיר פונקצית Dispose עבור Class1 ולהשתמש בה כדי למנוע את המצב שתואר.

נציג את הקוד לפתרון אפשרי זה:

```
using System;

namespace Good
{
	class Class1 : IDisposable
	{
		private Class2.SomethingWasReadHandler eventDelegate;
		private Class2 cl;

		private void EventHandler(string input)
		{
			int hash = this.GetHashCode();
			System.Console.WriteLine("This is a class2 instance with hashcode={0}",hash);
			System.Console.WriteLine("I got this input in an event:\t{0}\n", input);
		}

		public Class1(Class2 c)
		{
			eventDelegate=new Class2.SomethingWasReadHandler(EventHandler);
			c.SomethingWasRead+=eventDelegate;
			cl=c;
		}

		public void Dispose()
		{
			cl.SomethingWasRead-=eventDelegate;
		}
	}

	public class Class2
	{
		public delegate void SomethingWasReadHandler(string input);
		public event SomethingWasReadHandler SomethingWasRead;

		public void RaiseThatEvent(string input)
		{
			if(SomethingWasRead!=null)
				SomethingWasRead(input);
		}
	}

	class Program
	{
```

```
                static void Main(string[] args)
                {
                        Class2 EventRaiser=new Class2();
                        Class1 HandlesEvent1=new Class1(EventRaiser);
                        Class1 HandlesEvent2=new Class1(EventRaiser);

                        System.Console.WriteLine("Please enter something");
                        string input=System.Console.ReadLine();
                        System.Console.WriteLine();

                        EventRaiser.RaiseThatEvent(input);

                        HandlesEvent1.Dispose();
                        HandlesEvent2.Dispose();
                        HandlesEvent1=HandlesEvent2=null;

                        System.Console.WriteLine("Again please enter something");
                        input=System.Console.ReadLine();

                        EventRaiser.RaiseThatEvent(input);

                        System.Console.WriteLine("Press enter to quit...");
                        System.Console.ReadLine();
                }
        }
}
```

בעיה שניה

מצביעים חלשים (week references) הם התייחסויות לאובייקטים אשר ל-GC מותר לאסוף אותם בכל רגע.
דוגמא לשימוש במצביעים חלשים:

```
using System;

namespace WeakRef
{
        class Program
        {
                static void Main(string[] args)
                {
                        WeakReference r=new WeakReference(new byte[100]);
                        //making sure GC collects objects without a strong reference at this time
                        GC.Collect();
                        System.Console.WriteLine("Type of r.Target={0}\n\nNow trying an object with
both a strong reference and weak reference\n",r.Target==null?"null":r.Target.GetType().ToString());

                        byte []b=new byte[100];
                        r=new WeakReference(b);
                        System.Console.WriteLine("Calling GC.Collect\n");
                        GC.Collect();
                        System.Console.WriteLine("This time type of
r.Target={0}",r.Target==null?"null":r.Target.GetType().ToString());

                        System.Console.WriteLine("Press enter to quit...");
                        System.Console.ReadLine();
                }
        }
}
```

הפלט יהיה:

```
Type of r.Target=null

Now trying an object with both a strong reference and weak reference

Calling GC.Collect

This time type of r.Target=System.Byte[]
Press enter to quit...
```

את בעיה זו לא ניתן לפתור בעזרת ארועים, כי כאשר נוסיף מצביע אל delegate יווצר תמיד מצביע חזק. הפתרון המוצע הוא פשוט להיות מודע לנושא ולהזהר בעבודה עם מצביעים חלשים.

EOF