الباب العاشر

المناديب والأحداث

(Delegates and Events)

محتويات الباب:

- المناديب (Delegates)
- إعلان واستخدام المندوب
- تجميع وحذف المناديب (Multicast)
- الأحداث (Events)
- استخدام الأحداث فى التطبيقات

المناديب (Delegates)

إن المـــندوب (**delegate**) هو أحد أنماط المرجع بلغة سى#. ويستخدم المندوب فى كبسلة (احتواء) أسلوب ذى توقيع معين وتمريره كبارامتر. وهـــو فى ذلك يماثل المؤشرات إلى الدوال فى لغة سى++ فيما عدا أن المـــندوب يتمتع بصفات الكود المأمون كسائر عناصر اللغات المحكومة التى تسيطر عليها مظلة دوت.نت.

وكمـــا يوحـــى اسم المندوب فهو لا يعمل بمفرده ، وإنما يوكّل عنه فى العمل الأسلوب الذى يرتبط به.

إعلان واستخدام المندوب

تستخدم الصورة التالية فى إعلان المندوب:

```
[attributes] [modifiers] delegate result identifier ([parameters]);
```

حيث:

- ❖ attributes: صفات إضافية ــ اختيارية.
- ❖ modifiers: المعـــدّلات المســـموح بها وتشمل توليفة جائزة من معدّلات التوصل والمعدّل new.
- ❖ result: نمط المندوب ، وهو يماثل نمط الأسلوب الذى يرتبط به المندوب.
- ❖ identifier: اسم المندوب.

❖ parameters: قائمة اختيارية من البارامترات.

مثال لخطوات إعلان واستدعاء المندوب:

١. أعلن المندوب فى حيز الاسم (أو فى المكان الذى تعلن فيه الفصائل):

```
delegate void MyDelegate(object o1, object o2);
```

٢. أعلن الأسلوب المرتبط بهذا المندوب:

```
public static void MyMethod(object id, object name)
```

ونلاحظ أن كلاً من نمط القيمة المرتجعة والبارامترات التى يستخدمها الأسلوب تماثل تلك التى يستخدمها المندوب وهى موضحة بالبنط الأسود (الثقيل).

٣. لاستدعاء المندوب ابدأ بخلق هدف منه:

```
MyDelegate delgObj = new MyDelegate(MyMethod);
```

ونلاحظ أن بارامتر هدف المندوب عبارة عن اسم الأسلوب المرتبط به (MyMethod).

٤. ثم استدع هدف المندوب كما لوكنت تستدعى الأسلوب MyMethod المرتبط به:

```
delgObj(425, "Moustafa Amin");
```

أى أن البارامترات المستخدمة عبارة عن بارامترات الأسلوب المرتبط بالمندوب.

كما يمكنك ، كبديل ، استدعاء المندوب من خلال أسلوب كالأسلوب التالى:

```
public static void CallDelegat e(MyDelegate meth)
{
    meth(425, "Moustafa Amin");
```

}

ونلاحظ أن الأسلوب يستخدم بارامتراً من النمط MyDelegate بالاسم meth. وبطبيعة الحال فإن اسم البارامتر لايهم ولكن المهم أن يكون اسماً لأسلوب له نفس توقيع الأسلوب MyMethod . (وهذه هى نفس القاعدة المستخدمة مع أسماء البارامترات عامة.)

لاحظ أيضاً أننا قد استخدمنا المعدّل static حتى نتجنب خلق الأهداف بغرض تبسيط المثال. ولكن المندوب يمكنه استدعاء الدوال الأمثلة والدوال الإستاتيكية سواء بسواء.

مثال (10-1)

فى هذا المثال نضع الخطوات السابقة فى برنامج واحد لإعلان وتنفيذ مندوب.

```
// Example 10-1.cs
// Using gelegates

using System;

// Declare a delegate:
delegate void MyDelegate(int n, string s);

class MainClass
{
  static void Main()
  {
    // Instantiate the class:
    MyClass mc = new MyClass();

    //  Intantiate the delegate:
    MyDelegate d = new MyDelegate(mc.MyMethod);
```

```
    // Invoke the delegate:
    mc.CallDelegate(d);
  }
}

class MyClass
{
// A method to invoke the delegate:
  public void CallDelegate(MyDelegate meth)
  {
    m eth(425, "Moustafa Amin");
  }

// The encapsulated Method:
  public void MyMethod(int id, string name)
  {
    Console.WriteLine("ID = {0}  \nName = {1}", id, name);
  }
}
```

تنفيذ البرنامج:

```
ID = 425
Name = Moustafa Amin
```

تدريب (10-1)

أعد كتابة البرنامج السابق بدون استخدام الأسلوب الإضافى:

CallDelegate(MyDelegate meth)

بدلاً من ذلك استدع المندوب مباشرة باستخدام بارامترات الأسلوب المرتبط به.

مقارنة بين المندوب والوصلة البينية

يتشابه المندوب مع الوصلة البينية من حيث أن كليهما يمثل عقداً بين طرفين. والوصلة البينية ، كما رأينا من قبل ، عبارة عن تعاقد مع الفصائل التى تطبق الوصلة. أما المندوب فهو تعاقد مع أسلوب ما.

أمـــا الفارق الرئيسى بينهما فهو أن الوصلة يتم تعريفها أثناء الترجمة ، أمـــا المندوب قيتم تعريفه أثناء التشغيل ، أى ديناميكياً. بمعنى آخر فإن المندوب لا "يعرف" ــ أثناء الترجمة ــ الأسلوب الذى يشير إليه. إن ما يهم هو توقيع الأسلوب والنمط المرتجع منه.

مثال (2-10)

فـــى هذا المثال نستخدم مندوباً من النمط double ، وهو بالطبع يرتبط بأساليب من نفس النمط. الأسلوب الأول لحـــساب المتوسط (Average) والـــثانى لحساب المجموع (Sum). والمثال تطبيق مباشر على استخدام المناديب.

```
// Example 10-2.cs
// Delegate example

using System;

public class Calc
{
  // Declare a delegate:
  public delegate double Cal culation(int x, int y, int z);
    // Declare methods:
       public static double Sum(int n1, int n2, int n3)
  {
    return n1 + n2 + n3;
  }
     public static double Average(int n1, int n2, int n3)
    {
    return (n1 + n2 + n3)/3;
     }

   public static void Main()
   {
  double result;
```

```
    // Instantiate the delegate, associate it with Average:
  Calculation myCalc = new Calculation(Average);
    // Invoke the delegate:
    result = myCalc(3,6,9);
  Console.WriteLin e("Average: {0}", result);

  // Instantiate another object, associate it with Sum:
  myCalc = new Calculation(Sum);
    // Invoke the delegate:
  result = myCalc(3,6,9);
  Console.WriteLine("Sum: {0}", result);
   }
}
```

تنفيذ البرنامج:

```
Average: 6
Sum: 18
```

ملاحظات على البرنامج السابق:

لاحظ فى هذا البرنامج أننا استخدمنا هدفين من المندوب ، وربطنا الهدف الأول بالأسلوب Average والثانى بالأسلوب Sum. وعند استدعاء الأهداف قام كل هدف بدوره المحدد فى استدعاء الأسلوب المناسب.

تجميع وحذف المناديب (Multicast)

تتميز المناديب بأن يمكن تجميعها معاً باستخدام المؤثر "+" لإنتاج مندوب مركب. ويقوم المندوب المركب الناتج باستدعاء المناديب المكونة له. كما يجوز حذف مندوب من المندوب المركب باستخدام المؤثر "-". ويجوز أيضاً استخدام المؤثرات "+=" و "-=" لأداء نفس المهام.

كما يجوز تجميع المناديب وحذفها باستخدام أساليب دوت.نت **Combine()** و **Remove()** وهى أعضاء بالفصيلة **System.Delegate** كما سيلى شرحه.

مثال (10-3) المناديب المركبة

يوضح المثال التالى عملية تجميع المناديب بغرض كبسلة أكثر من أسلوب واحد بنفس المندوب. كما يوضح عملية طرح مندوب من المناديب المركبة.

```
// Example 10-3.cs
// Adding and removing delegates

using System;

// Declare a delegate:
delegate void MyDelegate();

class MyClass
{
  public void MyMethod1()
  {
    Console.Write("MyMethod #1 ");
  }

  public void MyM ethod2()
  {
    Console.Write("MyMethod #2 ");
  }
}

class MainClass
{
  static void Main()
  {
    // Instantiate MyClass:
    MyClass mc = new MyClass();
```

```
    // Declare delegate object and reference MyMehod1:
       MyDelegate d1   = new MyDelegate(mc.MyMethod1);

    // Declare delegate object and reference MMehod2:
       MyDelegate d2 = new MyDelegate(mc.MyMethod2);

       // Declare delegate d3 by adding d1 and d2. This will invoke
both MyMehod1 and MyMehod2:
       MyDelegat  e d3 = d1 + d2;    // جمع المناديب

       // Declare delegate d4 by removing d1 from d3. This will
invoke MyMehod2 only:
       MyDelegate d4 = d3    - d1;  // حذف مندوب من المجموعة

       Console.Write("Invoking d1, referencing ");
       d1();
        Console.Write("\nInvoking d2, referencing ");
       d2();
       Console.Write("  \nInvoking d3, referencing ");
       d3();
       Console.Write("  \nInvoking d4, referencing ");
       d4();
    }
}
```

تنفيذ البرنامج:

```
Invoking d1, referencing MyMethod #1
Invoking d2, referencing MyMethod #2
Invoking d3, referencing MyMethod #1 MyMethod #2
Invoking d4, referencing MyMethod #2
```

ملاحظات على البرنامج السابق:

فـــى هـــذا البرنامج قد استخدمنا المندوب الأول (d1) لكبسلة الأسلوب MyMethod1 وكذلـــك اســـتخدمنا المندوب الثانى (d2) لكبسلة الأسلوب MyMethod2.

لاحـــظ أن إضافة المندوب d1 إلى d2 لتكوين المندوب المركب d3 أدى إلى استدعاء الأسلوبين MyMethod1 و MyMethod2 عند استدعاء d3.

لاحظ أيضاً أن حذف المندوب d1 من المندوب المركب d3 قد أدى إلى استدعاء الأسلوب MyMethod2 فقط عند إستدعاء d3.

لاحـــظ أنـــه كان من الممكن إجراء عمليات الجمع والإضافة باستخدام المؤثر المركب =+ و =- مثل:

```
d2 += d1;  // إضافة المندوب ١ و ٢ ووضع الناتج فى ٢
d3 -= d1;  // طرح المندوب ١ من المندوب المركب ٣ ووضع الناتج فى ٣
```

استخدام أساليب دوت.نت لتجميع وحذف المناديب

كان من الممكن أيضاً إجراء العمليات السابقة باستخدام أساليب دوت.نت كالآتى.

❖ الإضافة:

```
MyDelegate d3 = (MyDelegate)Delegate.Combine(d1, d2);
```

❖ الحذف:

```
MyDelegate d4 = (MyDelegate)Delegate.Remove(d3, d1);
```

وتســـتلزم أساليب دوت.نت للحذف والإضافة استخدام الاسقاط لتحويل نمـــط الناتج إلى نمط المندوب. ومن البديهى أن سى# قد قدمت تسهيلاً لهذه العمليات.

ملاحظة:

إن اســـتخدام الـــنمط void مع المناديب المركبة لم يكن مصادفة ، بل إنه لا يوصـــى باســـتخدام أى نمط آخر طالما أن المندوب سوف يرتبط بأكثر من

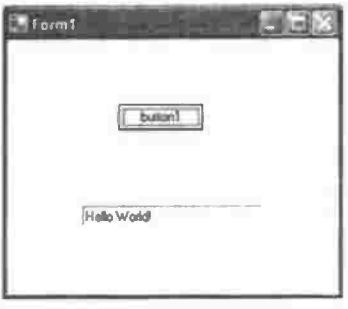
أسلوب. إن المشكلة المتوقع حدوثها فى هذه الحالة هى عدم إمكان متابعة أو التحكم فى القيم المرتجعة من الأساليب عند استدعاء المندوب.

تدريب (2-10)

أعد كتابة البرنامج السابق باستخدام الدوال الإستاتيكية ، وتأكد من حصولك على نفس النتيجة عند تشغيل البرنامج.

الأحداث (Events)

من أهم استخدامات المناديب هى برمجة الأحداث ولاسيما فى بيئة النوافذ. فأنت عندما تضغط على أحد الأزرار بنافذة ما فإن هذا يسمى حدثاً. أما الاستجابة لهذا الحدث فقد تأخذ صورا متعددة ، وعليك كمبرمج أن تكتب الاستجابة الملائمة للحدث فى للتطبيق. وكمثال بسيط لو أنك أنشأت تطبيقاً نوفذياً (Windows Forms application) وبدأت بإضافة زر واحد إلى النموذج (نافذة التطبيق) فإنك يمكن أن تربط حدث الضغط على الزر بنتيجة ما مثل كتابة عبارة ما فى صندوق نصوص. وفى الشكل التالى نرى العبارة Hello World! مكتوبة فى صندوق النصوص كنتيجة لحدث الضغط على الزر.



أما الكود المكتوب فى خلفية هذه العملية فهو الأسلوب التالى:

```
private void button1_Click(object sender, System.EventArgs e)
{
  textBox1.Text = "Hello World!";
}
```

وعـندما تخطو الخطوات الأولى فى برمجة النوافذ ، سوف تعرف أن الكود المطلوب إضافته هنا هو:

```
textBox1.Text = "Hello World!";
```

أمـا الأسلوب نفسه فهو قياسى وتقوم البيئة المجمعة للإستوديو بإضافته لـك. وما يعنيه هذا الكود هو: "عند الضغط على الزر button1 اكتب الـنص "Hello World!" فـى صـندوق النصوص المسمى textBox1".

وسـوف نصـطلح فـى مجال الأحداث على تعبيرى **إرسال** الحدث ، و**اسـتقبال** الحـدث (قـد توجـد ترجمات أخرى مثل **إشعال** الحدث ، و**معالجة** الحدث):

إرسال (أو إشعال) الحدث: Firing an event
استقبال (أو معالجة) الحدث: Handling an event

ويوجد العديد من الأحداث فى بيئة النوافذ التى يمكنك إرسالها مثل حدث تغيير النص بصندوق نصوص أو حدث مغادرة الصندوق إلى أداة تحكم أخـرى وهكذا. أما استقبال هذه الأحداث فهو يخضع للتطبيق الذى تقوم ببرمجـته. فـربما يؤدى استقبال الحدث إلى عزف نغمة ما أو إرسال رسالة ما بداخل صندوق رسالة.

استخدام الأحداث فى التطبيقات

يمكنك ، فى بيئة الأوامر ، استخدام الأحداث فى تطبيقات مختلفة نبدأها بتطبيق محاكاة الضغط على الزر. والمحاكاة تعنى أنه ليس هناك زر حقيقى ولكننا سوف نقوم بتمثيل العمليات المرتبطة بإرسال الحدث عند الضغط على الزر ، ثم استقباله ومعالجته بكتابة عبارة ما على الشاشة.

ولتحقيق ذلك سوف نستخدم المندوب (delegate) كالمثال الآتى:

```
public delegate void ClickHandler(object sender, EventArgs e)
```

والاسم الذى منحناه للمندوب هنا هو ClickHandler بمعنى مقبض المعالجة. ويستخدم هذا الاسم كنمط للحدث الذى نعلنه باستخدام الكلمة المفتاحية **event** كالآتى:

```
public event ClickHandler Click;
```

وبذلك يتم ربط الحدث بالمندوب.

أما بارامترات المندوب فسوف نناقشها مع مناقشة المثال.

مثال (10-4) محاكاة الضغط على الزر

فى هذا المثال تفترض وجود الزر ، ونربطه بحدث الضغط على الزر. وعند إرسال هذا الحدث فإن البرنامج يستقبله ويستجيب له بطباعة الرسالة:

```
Message from myButton: 'He y, you clicked me.'
```

وهذا هو البرنامج.

```
// Example 10-3.cs
// Event example

using System;
```

```
public class ButtonClass
{
  public delegate void ClickHandler(object sender, EventArgs e);
  public event ClickHandler Click;

  public void OnClick()
  {
     if (Click != null)
    Click(this, null);
  }
}

class MyClass
{
  public static void ButtonHandler(object sender, EventArgs e)
  {
    Console.WriteLine("Message from myButton: 'Hey, you clicked
me.'");
  }

  static void Main()
  {
    Button  Class myButton = new ButtonClass();
    myButton.Click += new
              ButtonClass.ClickHandler(ButtonHandler);

     myButton.OnClick();

     myButton.Click  -= new
               ButtonClass.ClickHandler(ButtonH     andler);

  }
}
```

تنفيذ البرنامج:

```
Message from myButton: 'Hey, you clicked me.'
```

ملاحظات على البرنامج السابق:

أول مـــا تلاحـــظ علـــى صـــيغة الحـــدث أنـــه يستخدم اسم المندوب ClickHandler كنمط أى:

```
public event ClickHandler Click;
```

لاحـــظ أيضاً أنه فى الأسلوب ()Main يتم التوصل إلى الحدث وإرساله وكأنه مندوب مع كبسلة الأسلوب ()ButtonHandler كبارامتر للمندوب ، أى:

```
myButton.Click += new
            ButtonClass.ClickHandler(ButtonHandler);
```

إن المؤشـــرات المسموح باستخدامها لإضافة وطرح الأحداث هى =+ و =-. بمعنى أنه لا يجوز استخدام عبارة مثل:

```
myButton.Click = new
      ButtonClass.ClickHandler(ButtonHandler);
```

إن هـــذه العبارة تؤدى إلى فك أى ارتباط سابق للحدث ، بينما المقصود هو إضافة إرتباط جديد.

أما بارمترات المندوب (object sender, EventArgs e) فوظائفها كالآتى:

- sender: يسمح لمستخدم الحدث أن يعرف أى الأهداف قد أرسل هذا الحدث.
- e: يحـــتوى علـــى المعلومات المرتبطة بالحدث. ولأنه لا توجد معلومات فى هذا المثال فقد استخدمنا النمط EventArgs مباشرة. وفـــى حالـــة إذا مـــا أردنـــا تمرير بعض المعلومات فلابد من تضمينها فى فصيلة مشتقة من EventArgs كما سنرى فى المثال التالى.

مثال (5-10) محاكاة مبيعات دار نشر

فى هذا المثال نمثل مبيعات دار نشر للكتب ، والبرنامج يقوم باستشعار حـــركة البيع والشراء عن طريق قياس كمية المخزون (stock) والتغير فـــى المخزون (change) بحيث يعطينا فى كل لحظة موقف المبيعات.

ونتيجة البرنامج عبارة عن رسالة عن كل صنف كالمثال الآتى:

```
2004 Sales Rate of:
    C# Books is up by 20
    C++ Books is down by 11
....
```

إن هذا التقرير قد بنى على نقصان مخزون كتب C# بمقدار 20 وزيادة مخزون كتب C++ بمقدار 11.

```
// Example 10-5.cs
// Book Store Simulation

using System;
class MyEventArgs: EventArgs
{
  // Fields:
  private string stock;
  private int change;
     // Properties:
  public string MyStock
  {
    get { return stock; }
  }
  public int MyChange
  {
    get { return change; }
  }
     // Constructor:
  public MyEventArgs(string s, int c)
  {
    stock = s;
    change = c;
  }
}
```

```
class Sender
{
  public delegate void EventHandler(object source, MyEventArgs
e);
  public event EventHandler OnChange;

       // Update database:
  public void Update(string s, int c)
  {
    MyEventArgs e = new MyEventArgs(s,c);
    if (OnChange != null)
      OnChange(this, e);
  }
}

class Receiver
{
  Sender senderObj;
  public Receiver(Sender s)
  {
    sender Obj = s;
       // Add the event:
    senderObj.OnChange += new
Sender.EventHandler(OnInventoryChange);
  }
  void OnInventoryChange(object source, MyEventArgs e)
  {
    string upOrDown;
          if (e.MyChange > 0)
            u    pOrDown = "down";
          else upOrDown = "up";
          int ch = Math.Abs(e.MyChange);

          Console.WriteLine("{0} {1} by {2}",e.MyStock,
upOrDown, ch);
  }
}

class MyClass
{
```

```
  public static void Main()
  {
    Sender s = new Sender();
    Receiver r = new Receiver(s);
    // Print Results:
    Console.WriteLine("2004 Sales Rate of:");
    s.Update(" \tC# Books:", -20);
    s.Update(" \tC++ Books:", 11);
          s.Update("   \tVB .NET Books:", -15);
          s.Update ("   \tScience Fiction Books:", 120);
  }

}
```

تنفيذ البرنامج:

```
Output:
2004 Sales Rate of:
        C# Books: up by 20
        C++ Books: down by 11
        VB .NET Books: up by 15
        Science Fiction Books: down by 120
```

ملاحظات على البرنامج السابق:

أول مـــا نلاحظ هنا أننا قد ورثنا الفصيلة EventArgs باستخدام الفصيلة MyEventArgs كالآتى:

```
class MyEventArgs: EventArgs
{
  private string stock;
  private int change;
  ...
}
```

وقـــد كان من اللازم عند بناء هذه الفصيلة تمـــرير بيانات المخزون (stock) والتغير فى المخزون (change) كما موضح فى أسلوب البناء:

```
public MyEventArgs(string s, int c)
```

نلاحظ أيضاً تقسيم فصائل البرنامج إلى مُرسل (Sender) ومُستقبل (Receiver) علاوة على الفصيلة MyEventArgs.

تم إعلان المندوب فى هذا المثال كالآتى:

```
public delegate void EventHandler(object source, MyEventArgs e);
```

وبذلك فقد أصبح البارامتر e يحمل البيانات اللازمة للمرسل عندما يحدث تغير فى المخزون. أما الحدث فقد تم إعلانه باستخدام اسم المندوب (كنمط) كالآتى:

```
public event EventHandler OnChange;
```

نلاحظ أيضاً أن فصيلة المُستقبل تحتوى على هدف من فصيلة المُرسل senderObj استخدمناه فى إضافة الحدث كالآتى:

```
senderObj.OnChange += new
        Sender.EventHandler(OnInventoryChange);
```

أما الأسلوب OnInventoryChange فهو المسئول عن استقبال وتحليل معلومات المخزون ، وهو أيضاً يستخدم البارمترات source و e:

```
void OnInventoryChange(object source, MyEventArgs e)
```

أما الأسلوب الرئيسى فكل دوره ينحصر فى تمرير التغيرات الحادثة فى المخزون إلى الأسلوب Update بفصيلة المرسل التى تتولى إشعال الحدث ومعالجته.