[00_1 var]

VAR

עד הכרזת גרסה 3 של #C הגדרת טיפוסים חייבה הגדרת טיפוס ספציפי int, string ...

מ 3.0 #C ניתן להגדיר טיפוסים בצורה מרומזת באמצעות השימוש במילה השמורה החדשה var.

הקומפיילר מזהה אוטומטית את הטיפוס על פי הערך המוגדר. הטיפוס האמיתי מזוהה בזמן קומפילציה ולא ניתן לשינוי במהלך הריצה.

```
var num = 7;
var num2 = 1.23;
var name = "chipopo";
Console.WriteLine(num.GetType());
Console.WriteLine(num2.GetType());
Console.WriteLine(name.GetType());
```

פלט:

System.Int32

System.Double

System.String

Press any key to continue. . .

הקומפיילר מזהה ש-num הוא למעשה int על פי הערך המושם, ש- name הוא למעשה string וש-num2 הוא למעשה double ומקצה את המשתנים מטיפוס הנכון.

ב- var ניתן להשתמש גם בסריקת מערכים

```
//pay attention that the IDE already recognizes the item as int
//move the mouse over the item and see that it is an int:)
foreach (var item in new []{1,2,3,4})
{
	Console.WriteLine(item.GetType());
}

//here we have to use the object type!
foreach (var item in new object[] { 1.1, "", 2, 3, 4 })
{
	Console.WriteLine(item.GetType());
}
```

System.Int32

```
System.Int32

System.Int32

System.Int32


System.Double

System.String

System.Int32

System.Int32

System.Int32

Press any key to continue. . .
```

מגבלות של שימוש ב- var

- חייבים לאתחל אותו בשורת ההגדרה.
- לא ניתן להגדירו כמאפיין PROP.
- לא ניתן להגדיר var כפרמטר למתודה או כערך מוחזר.
- לא ניתן לאתחל אותו ב- null.
- ניתן להגדרה רק כמשתנה לוקלי (לא ניתן להגדיר כשדה של מחלקה למשל!)

למה זה טוב ?

- בחלק הבא נילמד anonymous types אשר ביצירתם ניתן לקבל ייחוס רק מסוג var.

- לעיתים לא נוכל לדעת מראש איזה אובייקטים יחזרו מהשאילתה (כשנגיע ל LINQ) ולכן נהיה חייבים לשים את התוצאה ב var.

[00_2 anonymous types]

ANONYMOUS TYPES

טיפוסים אנונימיים (Anonymous types) מאפשרים לרכז אוסף של ערכים <u>לקריאה בלבד</u> (Read Only) באובייקט יחיד מבלי שיהיה צורך להגדיר מחלקה בצורה מפורשת.

לעיתים נוצר צורך לרכז מספר ערכים זמניים החיוניים לביצוע מטלה מסוימת שעם סיומה אין בהם כל צורך. החל מגרסה 3.0 של #C ניתן להגדיר טיפוסים אנונימיים (Anonymous types).
לעיתים זה יהיה פיתרון הולם משום שהנתונים הם זמניים, שאין משמעות להגדרת מחלקות, מתודות או אירועים, ושימוש חוזר בנתונים הללו כנראה לא יהיה מרובה.
טיפוסים אנונימיים הם דרך נוחה, מהירה ונטולת קוד מיותר לריכוז אוסף נתונים זמניים ליחידה אחת.

```
var myAnonymous = new { Age = 7, Name = "chipopo", Gender = true };
Console.WriteLine(myAnonymous.Age);
//myAnonymous.Age = 9;  //ERROR - read only!
Console.WriteLine(myAnonymous);
```

פלט:

```
7
{ Age = 7, Name = chipopo, Gender = True }
Press any key to continue . . .
```

שם הטיפוס ממנו נוצר האובייקט ניתן על ידי הקומפיילר ואינו ניתן לגישה מהקוד, ממילא אין בו צורך.

טיפוס המאפיינים של הטיפוס האנונימי אינו מוגדר במפורש אלא מוגדר על ידי הקומפיילר, בדומה למנגנון של var.

הייחוס, myAnonymous, המכיל את המופע של ה- Anonymous Type חייב להיות var משום שאין אנו יודעים את שם הטיפוס ובכל מקרה הוא יווצר מאוחר יותר רק בזמן הקומפילציה, כפי שמדגימה הדוגמה הבאה

```
Console.WriteLine("myAnonymous.GetType() : " + myAnonymous.GetType());
Console.WriteLine("myAnonymous.GetType().BaseType : " +
myAnonymous.GetType().BaseType);
Console.WriteLine("myAnonymous.Age.GetType() : " + myAnonymous.Age.GetType());
Console.WriteLine("myAnonymous.Name.GetType() : " +
myAnonymous.Name.GetType());
Console.WriteLine("myAnonymous.Gender.GetType() : " +
myAnonymous.Gender.GetType());
```

פלט:

```
myAnonymous.GetType() :
<>f__AnonymousType0`3[System.Int32,System.String,System.Boolean]

myAnonymous.GetType().BaseType : System.Object

myAnonymous.Age.GetType() : System.Int32

myAnonymous.Name.GetType() : System.String

myAnonymous.Gender.GetType() : System.Boolean

Press any key to continue . . .
```

נסתכל על הדוגמה הבאה ונבין ש:

1. obj1 ו-obj2 מאותו הטיפוס האנונימי, הקומפיילר יודע לזהות שהטיפוס כבר נוצר ופשוט מקצה ממנו אובייקט נוסף.
2. שהייחוסים obj1 ו- obj2 מצביעים לאובייקטים שונים.
3. שהתכונות של האובייקטים הללו מכילות ערכים זהים. Equals מתנהג כמו ב struct משווה כל אחד מהשדות

```
var myAnonymous = new { Age = 7, Name = "chipopo", Gender = true };

var myAnonymous2 = new { Age = 7, Name = "chipopo", Gender = true };

if (myAnonymous == myAnonymous2)
       Console.WriteLine("the same object/ref");
else
       Console.WriteLine("not the same object/ref");

if (myAnonymous.GetType() == myAnonymous2.GetType())
       Console.WriteLine("the same Type");
else
       Console.WriteLine("not the same Type");

if (myAnonymous.Equals(myAnonymous2))
       Console.WriteLine("the same Values");
else
       Console.WriteLine("not the same Values");
```

פלט:

not the same object/ref

the same Type

the same Values

ומה לגבי אוסף של טיפוסים אנונימיים?

```
 //anonymous collection
var myCollection = new[]
{
       new {num1=8, num2=1.1},
       new {num1=9, num2=2.2},
       new {num1=10, num2=3.3},
       //new {num1="a", num2="a"} //Error type mismatch
};

Console.WriteLine(myCollection.GetType());
Console.WriteLine(myCollection[0].GetType());
```

```
foreach (var item in myCollection)
{
        Console.WriteLine(item);
}
```

פלט:

<>f__AnonymousType1`2[System.Int32,System.Double][]

<>f__AnonymousType1`2[System.Int32,System.Double]

{ num1 = 8, num2 = 1.1 }

{ num1 = 9, num2 = 2.2 }

{ num1 = 10, num2 = 3.3 }

Press any key to continue . . .