# **Boxing and Unboxing**

**Boxing** - це процес перетворення типу-значення в тип об'єкта або в будь-який тип інтерфейсу, реалізований цим типом-значенням. Коли середовище виконання звичайної мови (CLR) боксує тип значення, воно обгортає значення всередині екземпляра System.Object і зберігає його у керованій купі. Розпакування витягує тип значення з об'єкта. Боксирування є неявним, а розпаковування - явним. Концепція боксування та розпакування лежить в основі уніфікованого погляду на систему типів C#, в якому значення будь-якого типу може розглядатися як об'єкт.

У наступному прикладі цілочисельну змінну ***i*** взято в блок і присвоєно об'єкту ***o***.

int i = 123;

// The following line boxes i.

object o = i;

Після цього об'єкт ***o*** можна розпакувати і присвоїти цілочисельній змінній ***i***:

o = 123;

i = (int)o; // unboxing

Наступні приклади ілюструють, як boxing використовується в C#.

// String.Concat example.

// String.Concat has many versions. Rest the mouse pointer on

// Concat in the following statement to verify that the version

// that is used here takes three object arguments. Both 42 and

// true must be boxed.

Console.WriteLine(String.Concat("Answer", 42, true));

// List example.

// Create a list of objects to hold a heterogeneous collection

// of elements.

List<object> mixedList = new List<object>();

// Add a string element to the list.

mixedList.Add("First Group:");

// Add some integers to the list.

for (int j = 1; j < 5; j++)

{

// Rest the mouse pointer over j to verify that you are adding

// an int to a list of objects. Each element j is boxed when

// you add j to mixedList.

mixedList.Add(j);

}

// Add another string and more integers.

mixedList.Add("Second Group:");

for (int j = 5; j < 10; j++)

{

mixedList.Add(j);

}

// Display the elements in the list. Declare the loop variable by

// using var, so that the compiler assigns its type.

foreach (var item in mixedList)

{

// Rest the mouse pointer over item to verify that the elements

// of mixedList are objects.

Console.WriteLine(item);

}

// The following loop sums the squares of the first group of boxed

// integers in mixedList. The list elements are objects, and cannot

// be multiplied or added to the sum until they are unboxed. The

// unboxing must be done explicitly.

var sum = 0;

for (var j = 1; j < 5; j++)

{

// The following statement causes a compiler error: Operator

// '\*' cannot be applied to operands of type 'object' and

// 'object'.

//sum += mixedList[j] \* mixedList[j]);

// After the list elements are unboxed, the computation does

// not cause a compiler error.

sum += (int)mixedList[j] \* (int)mixedList[j];

}

// The sum displayed is 30, the sum of 1 + 4 + 9 + 16.

Console.WriteLine("Sum: " + sum);

// Output:

// Answer42True

// First Group:

// 1

// 2

// 3

// 4

// Second Group:

// 5

// 6

// 7

// 8

// 9

// Sum: 30

## **Boxing**

Boxing використовується для зберігання типів значень у купі сміття, що збирається. Boxing - це неявне перетворення типу значення до типу об'єкта або до будь-якого типу інтерфейсу, реалізованого цим типом значення. Boxing типу-значення виділяє екземпляр об'єкта у купі і копіює значення у новий об'єкт.

Розглянемо наступне оголошення змінної типу-значення:

int i = 123;

Наступний оператор неявно застосовує операцію вставки до змінної ***i***:

// Boxing copies the value of i into object o.

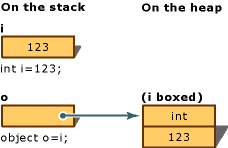
object o = i;

Результатом виконання цього оператора є створення посилання на об'єкт o у стеку, яке посилається на значення типу int у купі. Це значення є копією значення типу int, присвоєного змінній i. Різниця між двома змінними, i та o, проілюстрована на наступному зображенні перетворення **boxing**.

Також можна виконати **boxing** явно, як у наступному прикладі, але явне виконання **boxing** ніколи не є обов'язковим:

int i = 123;

object o = (object)i; // explicit boxing



### **Example**

This example converts an integer variable i to an object o by using boxing. Then, the value stored in the variable i is changed from 123 to 456. The example shows that the original value type and the boxed object use separate memory locations, and therefore can store different values.

class TestBoxing

{

static void Main()

{

int i = 123;

// Boxing copies the value of i into object o.

object o = i;

// Change the value of i.

i = 456;

// The change in i doesn't affect the value stored in o.

System.Console.WriteLine("The value-type value = {0}", i);

System.Console.WriteLine("The object-type value = {0}", o);

}

}

/\* Output:

The value-type value = 456

The object-type value = 123

\*/

У цьому прикладі ціла змінна ***i*** перетворюється на об'єкт o за допомогою боксу. Потім значення, що зберігається у змінній ***i***, змінюється зі **123** на **456**. Приклад показує, що вихідний тип значення і об'єкт, який перетворюється, використовують різні ділянки пам'яті, а отже, можуть зберігати різні значення.

class TestBoxing

{

static void Main()

{

int i = 123;

// Boxing copies the value of i into object o.

object o = i;

// Change the value of i.

i = 456;

// The change in i doesn't affect the value stored in o.

System.Console.WriteLine("The value-type value = {0}", i);

System.Console.WriteLine("The object-type value = {0}", o);

}

}

/\* Output:

The value-type value = 456

The object-type value = 123

\*/

## **Unboxing**

**Unboxing** - це явне перетворення з об'єкта типу у тип-значення або з типу інтерфейсу у тип-значення, що реалізує інтерфейс. Операція розпакування складається з наступних дій:

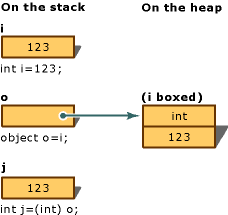
* Перевірка екземпляру об'єкта на предмет того, що він є значенням типу значення, що розпаковується.
* Копіювання значення з екземпляру у змінну типу-значення.

Наступні оператори демонструють як операцію boxing, так і unboxing:

int i = 123; // a value type

object o = i; // boxing

int j = (int)o; // unboxing



Щоб розблокування типів значень було успішним під час виконання, елемент, який розблоковується, має бути посиланням на об'єкт, який було створено раніше шляхом блокування екземпляра цього типу значень. Спроба розпакувати нуль викликає виключення ***NullReferenceException***. Спроба розпакувати посилання на несумісний тип значення спричиняє виключення ***InvalidCastException***.

### **Example**

The following example demonstrates a case of invalid unboxing and the resulting InvalidCastException. Using try and catch, an error message is displayed when the error occurs.

class TestUnboxing

{

static void Main()

{

int i = 123;

object o = i; // implicit boxing

try

{

int j = (short)o; // attempt to unbox

System.Console.WriteLine("Unboxing OK.");

}

catch (System.InvalidCastException e)

{

System.Console.WriteLine("{0} Error: Incorrect unboxing.", e.Message);

}

}

}

This program outputs:

Specified cast is not valid. Error: Incorrect unboxing.

If you change the statement:

int j = (short)o;

to:

int j = (int)o;

the conversion will be performed, and you will get the output:

Unboxing OK.