Bài 11: KIỂU DỮ LIỆU OBJECT TRONG C#.

Xem bài học trên website để ủng hộ Kteam: Kiểu dữ liệu Object trong C#.

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Ở bài KIỂU DỮ LIỆU TRONG C# chúng ta có đề cập đến một kiểu dữ liệu tham chiếu đó là **kiểu dữ liệu object**. Trong bài học này chúng ta sẽ cùng tìm hiểu chi tiết về kiểu dữ liệu này.

Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về các phần:

- Cấu trúc lệnh của C# viết trên nền Console Application.
- Cấu trúc nhập xuất của C# trên nền Console Application.
- BIẾN, KIỂU DỮ LIỆU, TOÁN TỬ trong C#.
- ÉP KIỂU TRONG C#.

Trong bài học này, chúng ta sẽ cùng tìm hiểu các vấn đề:

- Khái niệm về kiểu dữ liệu object.
- Boxing và unboxing trong C#.
- Từ khóa var trong C#.

Khái niệm về kiểu dữ liệu object

Kiểu dữ liệu object:

- Là một kiểu dữ liệu cơ bản của tất cả các kiểu dữ liệu trong .NET.
- Mọi kiểu dữ liệu đều được kế thừa từ System.**Object** (khái niệm về kế thừa sẽ được trình bày trong bài TÍNH KẾ THỪA TRONG C#).
- Thuộc kiểu dữ liệu tham chiếu (đã trình bày ở bài KIỂU DỮ LIỆU TRONG C#).

Kiểu dữ liệu object cung cấp một số phương thức ảo cho phép mình overload để sử dụng (những khái niệm này sẽ được trình bày trong bài TÍNH KẾ THỪA TRONG C#)

- Một số phương thức tiêu biểu nằm trong object:

Phương thức	Ý nghĩa
ToString()	Trả về kiểu chuỗi của đối tượng (chuyển từ kiểu dữ liệu nào đó về kiểu chuỗi)
GetHashCode()	Trả về mã băm của đối tượng.
Equals()	So sánh 2 đối tượng và trả về true khi 2 đối tượng có giá trị bằng nhau, ngược lại trả về false.
GetType()	Trả về kiểu dữ liệu của đối tượng.

- Ở bài học này chúng ta chỉ tìm hiểu các phương thức trên ở mức khái niệm. Trong các bài học sau sẽ ứng dụng vào ví dụ, khi đó sẽ có giải thích chi tiết về các phương thức trên.

Boxing và unboxing trong C#

Boxing là quá trình chuyển dữ liệu từ kiểu dữ liệu giá trị sang kiểu dữ liệu tham chiếu.

- Quá trình boxing:
  - Khởi tạo một đối tượng trong vùng nhớ Heap (đã được trình bày trong bài KIỂU DỮ LIỆU TRONG C#).
  - Copy giá trị của biến có kiểu dữ liệu giá trị vào đối tượng này.

Quá trình boxing được thực hiện ngầm định.

Ví dụ:

```
// Khởi tạo biến Value kiểu int (kiểu dữ liệu giá trị)
        int Value = 109;

        /* thực hiện boxing bằng cách:
         * Khởi tạo đối tượng ObjectValue kiểu object
         * Gán giá trị của biến Value vào ObjectValue */
        object ObjectValue = Value;
```

- Như phần trước mình đã trình bày là mọi kiểu dữ liệu đều kế thừa từ lớp **System.Object** nên có thể gán giá trị của kiểu dẫn xuất cho kiểu dữ liệu cha (sẽ được trình bày trong bài TÍNH KẾ THỪA TRONG C#). Trong ví dụ là biến kiểu int gán giá trị cho biến kiểu object.

Unboxing là quá trình ngược lại với boxing, tức là đưa dữ liệu từ kiểu dữ liệu tham chiếu về kiểu dữ liệu giá trị.

- Unboxing được thực hiện tường minh và thông qua cách ép kiểu tường minh (đã được trình bày trong bài ÉP KIỂU TRONG C#).
- Phải chắc chắn rằng đối tượng cần boxing thuộc đúng kiểu dữ liệu đưa ra. Nếu không việc unboxing sẽ báo lỗi chương trình.
- Quá trình unboxing:
  - Kiểm tra xem đối tượng cần un-boxing có thuộc đúng kiểu dữ liệu đưa ra hay không.
  - Nếu đúng thì thực hiện copy giá trị của đối tượng sang biến dữ liệu kiểu giá trị. Ngược lại thì thông báo lỗi.

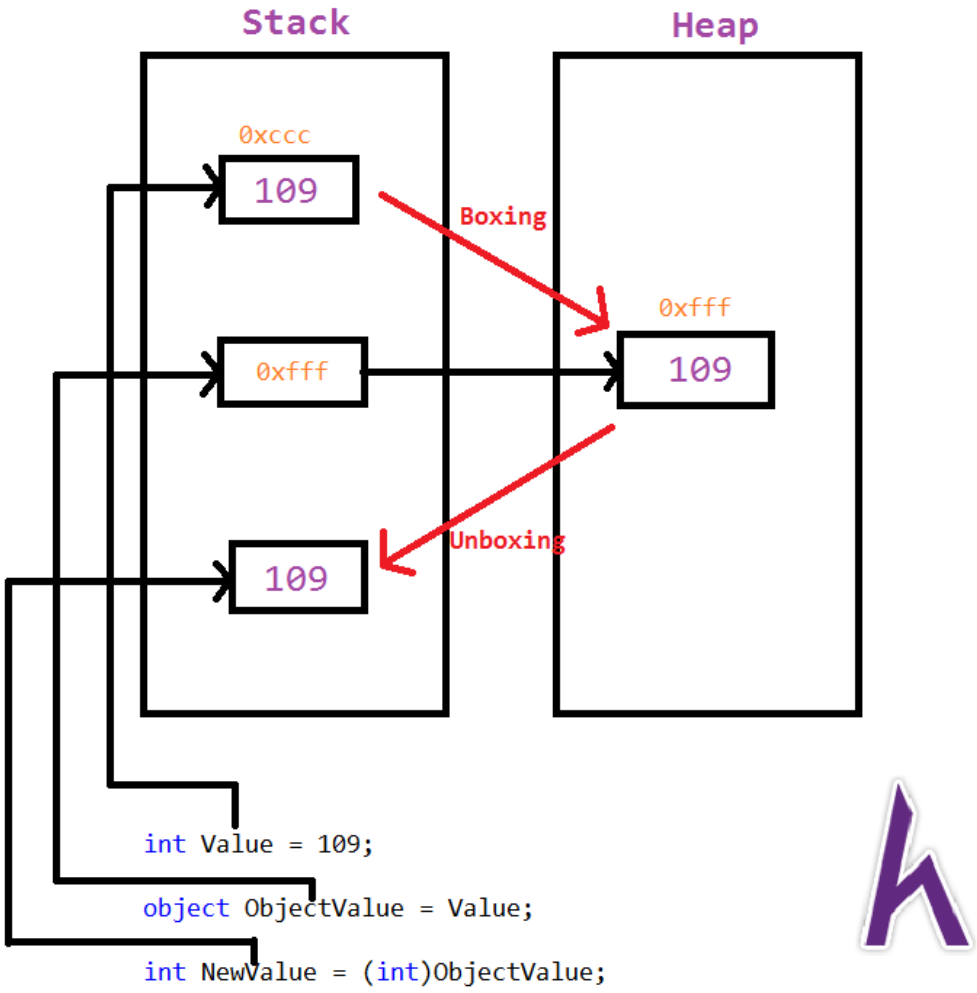
Ví dụ:

```
int Value = 109;

        // Boxing
        object ObjectValue = Value;

        /* thực hiện unboxing bằng cách:
         * Kiểm tra dữ liệu biến ObjectValue thấy thuộc đúng kiểu int.
         * Gán giá trị của biến ObjectValue vào biến NewValue bằng cách ép kiểu
tường minh.
         * Biến NewValue sẽ mang giá trị là 109*/
        int NewValue = (int)ObjectValue;
```

Việc boxing và unboxing có thể mình họa bằng hình ảnh sau:



Từ khóa var trong C#

var là từ khóa hỗ trợ khai báo biến mà không cần kiểu dữ liệu, kiểu dữ liệu sẽ được xác định khi gán giá trị cho biến, lúc đó chương trình sẽ tự ép kiểu cho biến.

Những lưu ý khi sử dụng từ khóa var:

- Bắt buộc phải gán giá trị ngay khi khởi tạo biến và không thể khởi tạo giá trị null cho biến var.

```
// Lỗi vì chưa khởi tạo giá trị cho biến varInt.
```

```
        var varInt;

        // Lỗi vì không được khởi tạo giá trị null cho biến varString.
        var varString = null;

        // Lỗi vì phải khởi tạo giá trị ngay khi khai báo
        var varLong;
        varLong = 109;

        // Khai báo đúng!
        var varBool = true;
```

- var chỉ là từ khóa dùng để khai báo biến không phải là một kiểu dữ liệu.

Khai báo biến bằng từ khóa var thường được ứng dụng trong:

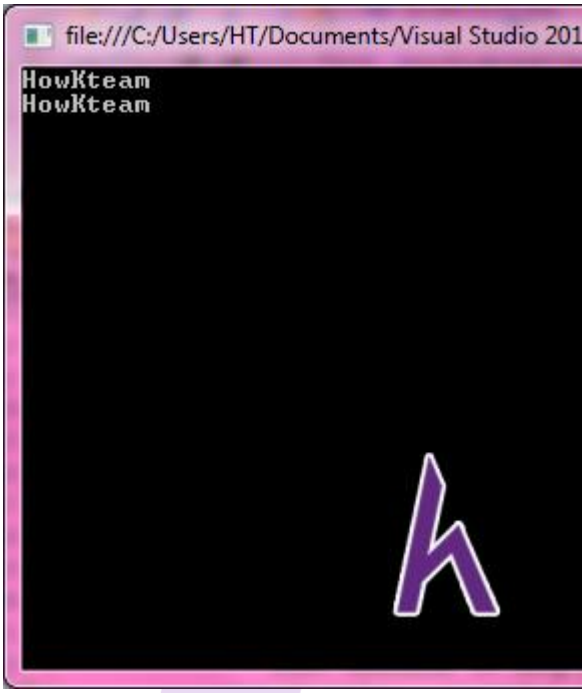
- Duyệt mảng bằng foreach (sẽ trình bày trong bài FOREACH TRONG C#).
- Truy vấn LinQ (sẽ trình bày trong bài LINQ TRONG C#).

Ví dụ minh họa sử dụng từ khóa var:

```
/* Vì biến StringVariable được khởi tạo giá trị kiểu chuỗi
         * nên trình biên dịch sẽ hiểu biến này như là biến kiểu string.
         */
        var varString = "HowKteam";
        // Khai báo tường minh biến kiểu string
        string Content = "HowKteam";

        // In giá trị của biến StringVariable và biến Content
        Console.WriteLine(varString);
        Console.WriteLine(Content);
```

- Kết quả khi chạy chương trình trên:



> Ở bài này mình chỉ ví dụ đơn giản cho các bạn hiểu về var. Ứng dụng thực sự của nó sẽ được trình bày ở các bài sau.

Kết luận

Nội dung bài này giúp các bạn nắm được:

- Khái niệm về kiểu dữ liệu object.
- Boxing và unboxing trong C#.
- Từ khóa var trong C#.

Bài học sau chúng ta sẽ cùng tìm hiểu một khái niệm tiếp theo đó là KIỂU DỮ LIỆU DYNAMIC TRONG C#

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".