Bài: HashTable trong C#

Xem bài học trên website để ủng hộ Kteam: HashTable 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 các bài học trước, chúng ta đã cùng nhau tìm hiểu về ARRAYLIST TRONG C#. Hôm nay chúng ta sẽ cùng tìm hiểu về **Hashtable trong C#**.

Nội dung

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

- BIẾN , KIỂU DỮ LIỆU, TOÁN TỬ trong C#
- CÂU ĐIỀU KIỆN trong C#
- Cấu trúc cơ bản của VÒNG LẶP, HÀM trong C#
- MẢNG trong C#
- LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG TRONG C#

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

- Hashtable là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Hashtable.

Hashtable là gì?

Hashtable trong C#:

Là một Collections lưu trữ dữ liệu dưới dạng cặp **Key - Value**.

- **Key** đại diện cho 1 khoá giống như chỉ số phần tử của mảng và **Value** chính là giá trị tương ứng của khoá đó.
- Ta sẽ dử dụng **Key** để truy cập đến **Value** tương ứng.

Vì **Key** và **Value** đều là kiểu object nên ta có thể lưu trữ được mọi kiểu dữ liệu từ những kiểu cơ sở đến kiểu phức tạp (**class**).

Nếu các **Key** của Hashtable là các số nguyên tăng dần từ **0** thì Hashtable nhìn trông giống ArrayList (chỉ là giống về bề ngoài thôi chứ bên trong rất khác nhau).

Do Hashtable là 1 Collections nên để sử dụng ta cần thêm thư viện System.Collections bằng câu lệnh:

> using **System.Collections;**

Vì Hashtable là một lớp nên trước khi sử dụng ta cần khởi tạo vùng nhớ bằng toán tử new:

C#:

```
// khởi tạo 1 Hashtable rỗng
Hashtable MyHash = new Hashtable();
```

Bạn cũng có chỉ định sức chứa (Capacity) ngay lúc khởi tạo bằng cách thông qua **constructor** được hỗ trợ sẵn:

C#:

```
// khởi tạo 1 Hashtable và chỉ định Capacity ban đầu là 5
Hashtable MyHash2 = new Hashtable(5);
```

Ngoài ra bạn cũng có thể khởi tạo 1 Hashtable chứa các phần tử được sao chép từ một Hashtable khác:

C#:

```
/*
 * Khởi tạo 1 Hashtable có kích thước bằng với MyHash2.
 * Sao chép toàn độ phần tử trong MyHash2 vào MyHash3.
 */
 Hashtable MyHash3 = new Hashtable(MyHash2);
```

Một số thuộc tính và phương thức hỗ trợ sẵn trong Hashtable

Một số thuộc tính thông dụng trong Hashtable:

TÊN THUỘC TÍNH	Ý NGHĨA
Count	Trả về 1 số nguyên là **số phần tử hiện có** trong Hashtable.
Keys	Trả về 1 danh sách chứa các **Key** trong Hashtable.
Values	Trả về 1 danh sách chứa các **Value** trong Hashtable.

Một số phương thức thông dụng trong Hashtable:

TÊN PHƯƠNG THỨC	Ý NGHĨA
Add(object Key, object Value)	Thêm 1 cặp **Key - Value** vào Hashtable.
Clear()	Xoá tất cả các phần tử trong Hashtable.
Clone()	Tạo 1 bản sao từ Hashtable hiện tại.
ContainsKey(object Key)	Kiểm tra đối tượng **Key** có tồn tại trong Hashtable hay không.
ContainsValue(object Value)	Kiểm tra đối tượng **Value** có tồn tại trong Hashtable hay không.
CopyTo(Array array, int Index)	Thực hiện sao chép tất cả phần tử trong Hashtable sang mảng một chiều array từ vị trí **Index** của array. **Lưu ý:** array phải là mảng các object hoặc mảng các **DictionaryEntry**.
Remove(object Key)	Xoá đối tượng có **Key** xuất hiện đầu tiên trong Hashtable.

Một số lưu ý về Hashtable

Mỗi một phần tử trong **Hashtable** (bao gồm 1 cặp **Key - Value**) được C# định nghĩa là 1 đối tượng có kiểu **DictionaryEntry**. Trong **DictionaryEntry** có 2 thuộc tính chính:

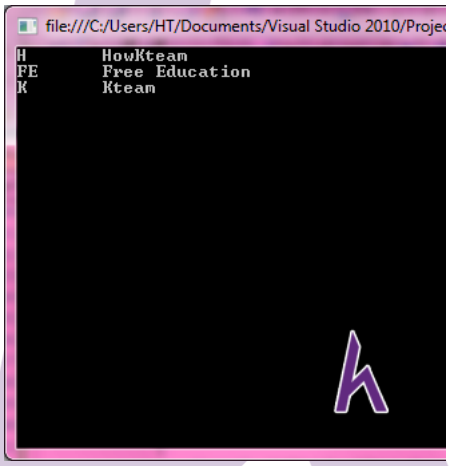
- **Key**: trả về giá trị Key của phần tử hiện tại.
- **Value**: trả về giá trị Value của phần tử hiện tại.

Ví dụ: mình thử dùng foreach duyệt 1 Hashtable và in ra giá trị **Key – Value** của mỗi phần tử:

C#:

```
// Tạo một Hashtable đơn giản với 3 phần tử
            Hashtable hash = new Hashtable();
            hash.Add("K", "Kteam");
            hash.Add("H", "HowKteam");
            hash.Add("FE", "Free Education");

            /*
             * Duyệt qua các phần tử trong Hashtable.
 * Vì mỗi phần tử là 1 DictionaryEntry nên ta chỉ định kiểu dữ liệu cho item là DictionaryEntry luôn.
             * Thử in ra màn hình cặp Key - Value của mỗi phần tử được duyệt.
             */
            foreach (DictionaryEntry item in hash)
            {
                Console.WriteLine(item.Key + "\t" + item.Value);
            }
```

Kết quả:



Việc truy xuất các phần tử trong Hashtable giống như truy xuất các phần tử mảng nhưng thông qua **Key**. Ví dụ:

C#:

```
Console.WriteLine(MyHash["One"]);
```

Trong đó:

- **MyHash** là tên của Hashtable.
- **One** là tên Key cần truy xuất.
- **MyHash["One"]** sẽ lấy ra giá trị **Value** tương ứng với **Key** trên.

Ta nên cẩn thận khi truy xuất các phần tử trong Hashtable thông qua **Key**:

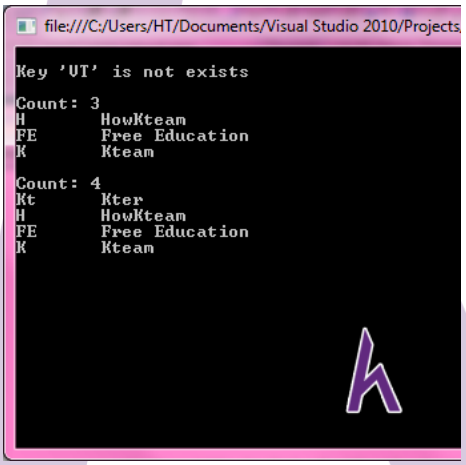
- Nếu ta thực hiện lấy giá trị 1 phần tử trong Hashtable với **Key** không tồn tại thì sẽ ra giá trị null và không báo lỗi.
- Nếu ta thực hiện gán giá trị cho 1 phần tử trong Hashtable tại vị trí **Key** không tồn tại thì Hashtable sẽ tự thêm 1 phần tử mới với **Key** và **Value** như trên. Điều này có thể làm phát sinh thêm các phần tử không mong muốn trong danh sách.

Ví dụ:

C#:

```
// In ra màn hình giá trị Value trong 1 Key không tồn tại.
            Console.WriteLine(hash["VT"]);

            // Để chắc chắn là null ta thử kiểm tra bằng điều kiện if.
            if (hash["VT"] == null)
            {
                Console.WriteLine("Key 'VT' is not exists");
            }

            // Thử in ra số phần tử ban đầu của Hashtable
            Console.WriteLine("\nCount: " + hash.Count);
            foreach (DictionaryEntry item in hash)
            {
                Console.WriteLine(item.Key + "\t" + item.Value);
            }

            // thực hiện gán giá trị cho 1 Key không tồn tại.
            hash["Kt"] = "Kter";

            // thực hiện in lại số phần tử của Hashtable để thấy sự thay đổi.
            Console.WriteLine("\nCount: " + hash.Count);
            foreach (DictionaryEntry item in hash)
            {
                Console.WriteLine(item.Key + "\t" + item.Value);
            }
```

Kết quả:



Kết luận

Nội dung bài này giúp các bạn nắm được:

- Hashtable là gì?
- Một số thuộc tính và phương thức hỗ trợ sẵn trong Hashtable.

Bài học sau chúng ta sẽ cùng tìm hiểu về SORTEDLIST 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**".