IEnumerable trong C#

Trong C#, IEnumerable là một interface trong namespace System.Collections (cho kiểu non-generic) hoặc System.Collections.Generic (cho kiểu generic). Interface này được sử dụng để cung cấp khả năng lặp qua một tập hợp (collection) các phần tử.

**1. Ý nghĩa của IEnumerable**

* **Non-generic IEnumerable**: Làm việc với các đối tượng kiểu object. Được sử dụng trong các collection cũ như ArrayList.
* **Generic IEnumerable<T>**: Làm việc với các đối tượng kiểu cụ thể (generic). Đây là cách phổ biến hơn khi làm việc với collection hiện đại, như List<T>, Queue<T>, hoặc HashSet<T>.

**2. Phương thức chính**

* GetEnumerator(): Phương thức này trả về một **enumerator** (đối tượng thực thi interface IEnumerator) để lặp qua các phần tử của collection.

**3. Ứng dụng**

* Cho phép sử dụng **foreach loop** để duyệt qua các phần tử trong collection.
* Dễ dàng thực hiện truy vấn LINQ trên các tập hợp dữ liệu.
* Hỗ trợ khả năng duyệt qua các tập hợp có kích thước lớn hoặc có thể thay đổi mà không cần biết chi tiết cách thức chúng được lưu trữ.

**Khi nào IEnumerable hữu ích?**

1. **Duyệt qua tập hợp không xác định kích thước**: Ví dụ, dữ liệu được tải từ file hoặc cơ sở dữ liệu.
2. **Khi không cần sửa đổi tập hợp**: Nếu chỉ đọc dữ liệu, IEnumerable rất tiện dụng và nhẹ nhàng.
3. **Kết hợp với LINQ**: IEnumerable cho phép thực hiện các truy vấn LINQ như lọc, sắp xếp.

Ví dụ minh họa :

foreach trong C# **hoạt động dựa trên IEnumerable**.

Khi bạn sử dụng foreach để lặp qua một tập hợp, C# sẽ ngầm gọi phương thức GetEnumerator() của IEnumerable để lấy một **bộ liệt kê (enumerator)**, sau đó sử dụng bộ liệt kê này để duyệt qua các phần tử.

Giả sử bạn viết một vòng lặp foreach như sau:

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };

foreach (int number in numbers)

{

Console.WriteLine(number);

}

Phía sau, foreach sẽ được biên dịch tương đương với mã sau:

using System.Collections.Generic;

List<int> numbers = new List<int> { 1, 2, 3, 4, 5 };

// Lấy enumerator từ tập hợp

IEnumerator<int> enumerator = numbers.GetEnumerator();

try

{

// Lặp qua từng phần tử

while (enumerator.MoveNext())

{

int number = enumerator.Current; // Lấy phần tử hiện tại

Console.WriteLine(number); // In ra phần tử

}

}

finally

{

// Giải phóng bộ nhớ nếu cần thiết

if (enumerator != null)

{

enumerator.Dispose();

}

}