Namespace (không gian tên) trong C#

Namespace (Không gian tên) trong C# có vai trò rất quan trọng khi bạn định nghĩa và sử dụng các kiểu dữ liệu. Namespace có nhiệm vụ phân nhóm toàn bộ các kiểu dữ liệu của C# và .NET theo một cấu trúc phân cấp. Nhờ có namespace, kiểu dữ liệu được quản lý tốt hơn và tránh được hiện tượng xung đột tên.

Bài học này sẽ cung cấp cho bạn những thông tin cơ bản về namespace trong C#.

NỘI DUNG CỦA BÀI [Ẩn]

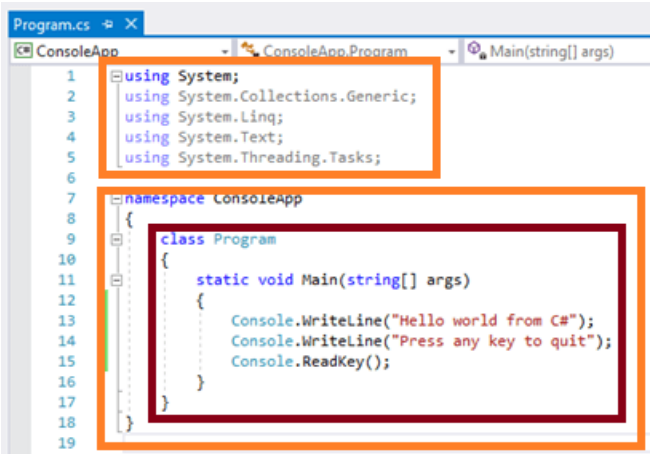
Namespace trong C#

Khi bạn bắt đầu có thể định nghĩa ra các kiểu dữ liệu của riêng mình (enum, struct), một vấn đề bắt đầu phát sinh: khả năng đặt tên kiểu trùng nhau. C# không cho phép đặt tên kiểu trùng nhau. Dễ hiểu thôi. Nếu tên kiểu trùng nhau, vậy sẽ dùng cái nào?

Lấy ví dụ về struct Complex mà bạn đã xây dựng trong bài học trước. Thực tế, C# (.NET) đã định nghĩa sẵn một kiểu struct Complex với cùng mục đích. Nhưng tại sao bạn vẫn xây dựng được struct Complex mà không bị lỗi.

Câu trả lời là, mặc dù có cùng tên nhưng chúng được đặt trong các **namespace** khác nhau.

Khái niệm Namespace trong C#



Các khối cơ bản trong file mã nguồn C#

Không gian tên (namespace) trong C# được dùng để tổ chức quản lý code. Có thể tưởng tượng namespace như một cái hộp chứa, trong đó có thể chứa những chiếc hộp khác (namespace con) và các vật dụng (các kiểu dữ liệu như struct, class, interface). Namespace đóng vai trò đặc biệt quan trọng giúp viết code rõ ràng rành mạch, cũng như hỗ trợ đắc lực trong việc quản lý các dự án lớn.

> Namespace mặc dù có thể dịch sang tiếng Việt là Không gian tên. Tuy nhiên nó không được sử dụng rộng rãi. Vì vậy, chúng ta sẽ gắn bó với từ gốc tiếng Anh **namespace**.

Namespace có thể trực tiếp chứa các đơn vị sau:

- Namespace khác (namespace con)
- Cấu trúc (struct)
- Lớp (class)
- Giao diện (interface)
- Kiểu liệt kê (enum)
- Kiểu ủy nhiệm (delegate)

> Khái niệm và cách sử dụng các loại đơn vị trên sẽ lần lượt được xem xét kỹ trong các bài tương ứng.

Namespace không thể chứa trực tiếp phương thức (method), biến, hằng, v.v.. Nói một cách khác, trong namespace chỉ có thể chứa định nghĩa kiểu dữ liệu (data type) và namespace con.

Namespace và thư mục

Để so sánh, chúng ta có thể hình dung cấu trúc namespace tương tự như cấu trúc thư mục và file trong hệ điều hành.

Giả sử chúng ta có một lượng lớn file, thư mục cho phép chúng ta nhóm các file có liên quan vào cùng một thư mục, nhóm các thư mục có liên quan vào một thư mục lớn hơn, v.v.. Cách tổ chức như vậy giúp chúng ta quản lý file và thư mục hiệu quả hơn.

Trong so sánh đó, không gian tên có vai trò tương tự thư mục, các định nghĩa kiểu dữ liệu có vai trò tương tự file. Sự khác biệt ở chỗ, nếu trong cấu trúc thư mục (Windows) sử dụng ký tự "" để phân tách các thư mục chứa nhau thì trong cấu trúc không gian tên sử dụng dấu chấm ".” để phân tách các không gian tên lồng nhau.

Nếu trong Windows không cho phép trong một thư mục có 2 file trùng tên nhau thì trong C# cũng không cho phép trong một không gian tên có hai đơn vị code trùng tên. Không gian tên cũng giúp giải quyết tình trạng trùng tên gọi (naming conflict).

Quy tắc đặt tên namespace trong C#

Không gian tên có thể được lựa chọn tùy ý, với các cấp độ lồng nhau (giống như đặt tên thư mục) theo tư tưởng phân chia code của người lập trình nhưng phải tuân thủ quy tắc đặt tên của C#:

- các tên gọi chỉ được bắt đầu bằng ký tự chữ cái hoặc ký tự gạch chân (ký tự underscore "_"),
- chỉ được chứa chữ cái, chữ số và ký tự gạch chân,
- phân biệt chữ hoa/thường.

Ngoài quy tắc bắt buộc trên, việc đặt tên không gian tên thường tuân thủ quy ước (không bắt buộc):

- không gian tên gốc nên đặt trùng với tên project;
- nếu trong project có thêm các thư mục thì file mã nguồn đặt trong các thư mục này sẽ có không gian tên con trùng với tên thư mục.

Khi đó, cấu trúc không gian tên sẽ đồng nhất với cấu trúc thư mục của project, giúp chúng ta quản lý code dễ dàng hơn. Nếu chúng ta thêm các file mã nguồn mới, Visual Studio sẽ tự động đặt không gian tên theo quy ước này. Ngoài ra, Visual Studio mặc định sẽ lấy tên của project làm namespace cho tất cả file mã nguồn trong project.

Trong file code dưới đây

File mã nguồn đầu tiên (Program.cs)

chứa 1 class (`Program`) đặt trong không gian tên `ConsoleApp` , trùng với tên project. File mã nguồn này nằm trực tiếp trong thư mục dự án nên không đặt thêm không gian con.

Cấu trúc using trong C#

Tên ngắn gọn và tên đầy đủ của kiểu dữ liệu

Với ví dụ về hệ thống file, ta thấy rằng mỗi file có thể có hai tên gọi: tên ngắn gọn (chỉ chứa tên của bản thân file) và tên đầy đủ (chứa thêm đường dẫn từ thư mục gốc tới thư mục chứa file đó).

Khi làm việc với hệ thống file của Windows (ví dụ, khi chạy một chương trình cần truy cập tới file), chúng ta gặp tình huống: nếu đang ở trong một thư mục nào đó, ta có thể truy xuất tới các file trong thư mục đó dùng tên ngắn gọn; nếu muốn truy xuất tới file nằm trong thư mục khác, ta phải dùng tên đầy đủ (rất dài dòng).

Tình huống tương tự xảy ra với không gian tên: mỗi kiểu dữ liệu cũng có hai tên gọi, *tên ngắn gọn* (short name) và *tên đầy đủ* (fully-qualified name). Tên ngắn gọn là tên (định danh) của kiểu do người lập trình đặt; tên đầy đủ là tên ngắn gọn cộng thêm cấu trúc không gian tên chứa định nghĩa kiểu này.

Nếu các định nghĩa kiểu nằm trong cùng một không gian tên, bạn có thể truy xuất tới nó trực tiếp thông qua tên ngắn gọn. Nếu các định nghĩa kiểu nằm trong các không gian tên khác nhau, bạn phải sử dụng tên đầy đủ.

Ví dụ khi học về console trong C#, bạn đã làm việc với class Console. Console là tên "cúng cơm" (ngắn gọn). Do lớp `Console` nằm trong không gian tên `System`, do đó nó có họ tên đầy đủ là `System.Console`.

Cấu trúc using

Theo nguyên tắc trên, nếu bạn muốn sử dụng lớp `Console` thì phải dùng tên đầy đủ của nó là `System.Console`. Cách viết như vậy rất dài dòng nếu như cấu trúc không gian tên phức tạp.

Cấu trúc **using** cho phép sử dụng tất cả các kiểu trong một namespace mà không cần sử dụng tên đầy đủ của chúng.

Ví dụ, khi chúng ta sử dụng `using System;`, tất cả các kiểu trong không gian tên này có thể được sử dụng thông qua tên ngắn gọn. Các lệnh using thường đường đặt cùng nhau thành một khối ở đầu mỗi file mã nguồn nhưng không bắt buộc. Khối using cũng có thể nằm luôn trong khối namespace.

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace ConsoleApp
8.  {
9.  // khối using cũng có thể để ở đây
10.     class Program
11.     {
12.         static void Main(string[] args)
13.         {
14.             Console.WriteLine("Hello world from C#");
15.             Console.WriteLine("Press any key to quit");
16.             Console.ReadKey();
17.         }
18.     }
```

```
19.  }
```

Cấu trúc using static

C# 7 đưa thêm vào cấu trúc using static giúp đơn giản hóa hơn nữa khi làm việc với các thành viên static của struct hoặc class.

Hãy xem lại một đoạn code bạn đã viết trong bài về enum trong C#:

```
1.  namespace P01_EnumVar
2.  {
3.      using System;
4.      using static System.Console;
5.      class Program
6.      {
7.          static void Main(string[] args)
8.          {
9.              // In biến enum ra console
10.             var gender = Gender.Male;
11.             WriteLine($"My gender is {gender} ({(int) gender})");
12.             var day = DayOfWeek.Tuesday;
13.             WriteLine($"Today is {day} ({(int)day})");
14. ...
```

Hãy để ý tới cấu trúc lạ `using static System.Console;`. Cấu trúc này cho phép bạn sử dụng trực tiếp tất cả các thành viên static của lớp Console. Nghĩa là giờ bạn có thể gọi trực tiếp Write, WriteLine (static method), thay vì Console.Write, Console.WriteLine. Bạn có thể sử dụng thẳng ForegroundColor (static property) thay vì phải viết Console.ForegroundColor.

Kết luận

Bài học này đã giúp bạn hiểu một khái niệm hơi mới lạ trong C# – namespace, cũng như vai trò quan trọng của nó khi khai báo và sử dụng kiểu dữ liệu trong C#.

Đến bài học về assembly và nội dung thực hành, bạn sẽ gặp lại namespace và cách vận dụng của nó.

> + Nếu bạn thấy site hữu ích, trước khi rời đi hãy **giúp đỡ** site bằng một hành động nhỏ để site có thể phát triển và phục vụ bạn tốt hơn.
> + Nếu bạn thấy bài viết hữu ích, hãy giúp **chia sẻ** tới mọi người.
> + Nếu có thắc mắc hoặc cần trao đổi thêm, mời bạn viết trong phần **thảo luận** cuối trang.
> Cảm ơn bạn!