Partial class và Partial method

Partial class

Khái niệm partial class

Trong các bài học từ đầu đến giờ, bạn xây dựng mỗi class trong một file đặt trùng tên với class. Đây là cách thức tổ chức mã nguồn của class bình thường và phổ biến trong tất cả các ngôn ngữ lập trình hướng đối tượng.

Giờ hãy hình dung một số tình huống khác.

Visual studio có khả năng sinh code tự động để tạo ra nhiều class khác nhau. Ví dụ, nếu bạn học lập trình với windows forms sẽ thấy, khi đặt một điều khiển lên form, trình thiết kế của Visual studio sẽ tự động sinh code tương ứng. Vậy giờ nếu bạn tự viết thêm code của mình vào file code được sinh tự động đó, code của bạn có thể bị mất đi nếu form thay đổi (vì bạn không thể kiểm soát việc sinh code tự động). Ngoài ra, code sinh tự động thường rất phức tạp. Bạn rất khó theo dõi các code sinh tự động này.

Đây là tình hình khi lập trình windows forms trong C# 1.0, khi chưa có khái niệm partial class.

Visual studio cũng có một công cụ riêng giúp sinh code tự động theo mẫu, gọi là T4 Text Template. Bộ sinh code tự động này được sử dụng rất nhiều, ví dụ, cho asp.net, entity framework. Nếu viết code vào file code sinh tự động, mỗi lần chạy lại bạn có thể mất hết code của mình.

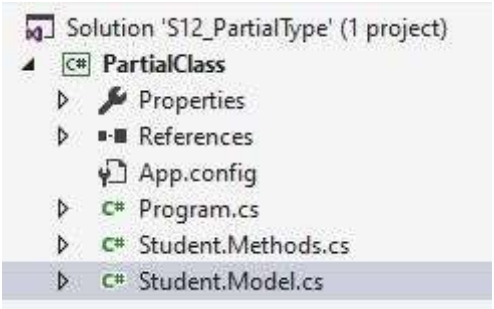
Từ những tình huống trên dẫn đến một nhu cầu đặc biệt: xây dựng MỘT class trên NHIỀU file vật lý. Điều này giúp giải quyết những vấn đề vừa nêu. Phần sinh tự động đặt ở một file độc lập. Phần bạn tự viết nằm trên một file khác. Thay đổi anh này không ảnh hưởng đến anh kia. C# gọi loại class xây dựng trên nhiều file vật lý như vậy là partial class.

Partial class là một tính năng của C# cho phép định nghĩa một class trên nhiều file vật lý khác nhau. C# compiler sẽ tự động ghép nối các file mã nguồn này trong quá trình biên dịch.

Partial class là tính năng phục vụ cho các công cụ hỗ trợ thiết kế và sinh code tự động.

Sử dụng partial class project

Hãy cùng thực hiện ví dụ sau. Thêm project tên PartialClass vào solution. Thêm hai file mã nguồn Student.Model.cs và Student.Methods.cs vào project.



partial class trong c# project

Lần lượt viết code cho các file như sau:

```
using System;
namespace PartialClass
{
    public partial class Student
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public DateTime DateOfBirth { get; set; }
        public string Major { get; set; }
        public string Specialization { get; set; }
    }
}
```

```
namespace PartialClass
{
    public partial class Student
    {
        public override string ToString()
        {
            return $"{FirstName} {LastName} ({DateOfBirth.ToShortDateString()})";
        }
    }
}
```

```
namespace PartialClass
{
    using static System.Console;
    class Program
    {
        static void Main(string[] args)
        {
            var student = new Student
            {
                Id = 1,
                FirstName = "Donald",
                LastName = "Trump",
```

```
                DateOfBirth = new System.DateTime(1990, 12, 31),
                Major = "Computer Science",
                Specialization = "Information Systems"
            };
            WriteLine(student);
            ReadKey();
        }
    }
}
```

Bạn để ý một số vấn đề sau:

Hai file mã nguồn Student.Model.cs và Student.Methods.cs đều chứa khai báo class Student trong cùng namespace PartialClass. Rõ ràng, điều này nhẽ ra phải gây ra xung đột định danh (name conflict). Tuy nhiên, khi bạn dịch chương trình sẽ thành công.

Vấn đề là ở chỗ, trước từ khóa **class** bạn gặp thêm từ khóa **partial**. Từ khóa này báo cho C# compiler rằng hai khai báo này thuộc về cùng một class Student. Chỉ là code được đặt trên hai file mã nguồn khác nhau. Khi dịch chương trình, C# compiler sẽ tự động ghép nối chúng lại thành một class duy nhất.

Do C# hiểu rằng hai khai báo thuộc về một class duy nhất, trong client code (phương thức Main), bạn sử dụng được những thành viên khai báo trên cả hai file. Nói cách khác, client code không phân biệt đây là partial class hay class thông thường.

Một số vấn đề khi sử dụng partial class

Như vậy có thể thấy, việc khai báo và sử dụng partial class thực ra rất đơn giản về cú pháp cũng như ý tưởng:

- Nếu trong class có nhiều thành phần khác nhau nhưng bắt buộc phải thuộc về 1 class, bạn có thể sử dụng partial class để tách các phần đó sang các file riêng rẽ. Ví dụ, phần thiết kế của form và xử lý sự kiện vốn có bản chất khác nhau nhưng phải thuộc về cùng class. Visual studio tự động tạo form làm partial class.
- Nếu trong một class có những thành phần code thường biến động theo quá trình phát triển và có những thành phần ổn định thì cũng có thể xem xét tách chúng ra các file khác nhau để dễ quản lý. Trong đó, thành phần biến động ra một file riêng, thành phần ổn định để lại một file riêng. Ví dụ, các thành phần field, property, constructor thường cố định. Trong khi đó các member method thường biến động nhiều hơn. Bạn có thể tách code thành hai phần riêng biệt.
- Nếu bạn sử dụng công cụ sinh code tự động, hãy để class đó làm partial class. Biết đâu về sau bạn cần bổ sung code tự viết.

Ngoài ra, khi sử dụng partial class cần lưu ý các vấn đề sau:

Thứ nhất, các file mã nguồn của partial class phải nằm trong cùng một assembly (cùng trong một project). Nếu bạn viết thư viện class và biên dịch để người khác sử dụng, trong đó có xây dựng một partial class. Người dùng class đó không thể dùng cơ chế partial class để mở rộng tiếp partial class của bạn được. Lý do rất đơn giản, partial class chỉ là tách code ra nhiều file riêng rẽ để sau compiler tự mình gom lại. Một partial class đã được biên dịch thì compiler không thể gom vào cùng code khác được nữa.

Thứ hai, trong mỗi phần code của partial class, bạn không được khai báo các thành viên trùng nhau. Lý do là, mặc dù trải rộng trên nhiều file mã nguồn khác nhau nhưng các phần của partial class thuộc về cùng một class. C# không cho phép khai báo cùng một thành viên nhiều lần.

Thứ ba, mỗi phần của class bắt buộc phải có đủ hai từ khóa **partial class**. Tuy nhiên, modifier (**public | internal**) thì chỉ cần viết một lần. Lớp cơ sở hoặc interface mà class này thừa kế cũng chỉ cần viết một lần.

Partial method

Khái niệm partial method

Trong partial class có thể chứa một loại thành viên đặc biệt mà class thông thường không có: **partial method**. Tương tự như partial class, partial method cũng hướng tới hỗ trợ sinh code tự động. Partial method xuất hiện từ C# 3.0 và được xem như thành phần mở rộng cho partial class (xuất hiện từ C# 2.0).

Partial method cho phép code sinh tự động gọi phương thức nhưng không nhất thiết phải xây dựng (implement) phương thức đó. Do vậy, partial method chỉ chứa signature (mô tả) mà không có phần implementation (phần thân, phần thực thi). Nếu không tìm thấy phần thực thi của partial method, compiler sẽ bỏ qua lệnh gọi partial method.

Người ta cũng thường gọi lời gọi partial method là **hook**. Hook nếu gắn với phần thực thi sẽ được gọi như phương thức bình thường. Nếu không có phần thực thi, hook sẽ được compiler bỏ qua.

Cơ chế trên giúp giữ khối lượng code nhỏ nhưng vẫn đảm bảo tính linh hoạt.

Sử dụng partial method

Hãy cùng thực hiện một ví dụ với partial method trong partial class. Chúng ta lặp lại ví dụ trên nhưng với một số thay đổi nhỏ:

```
using System;
namespace PartialMethod
{
    public partial class Student
    {
        private string _firstName;
        private string _lastName;
        partial void OnSettingFirstName(string value);
        partial void OnSettingLastName(string value);
        public int Id { get; set; }
        public string FirstName
        {
            get => _firstName;
            set {
                OnSettingFirstName(value);
                _firstName = value;
            }
        }
```

```
        public string LastName
        {
            get => _lastName;
            set {
                OnSettingLastName(value);
                _lastName = value;
            }
        }
        public DateTime DateOfBirth { get; set; }
        public string Major { get; set; }
        public string Specialization { get; set; }
    }
}
```

```
namespace PartialMethod
{
    public partial class Student
    {
        public override string ToString()
        {
            return $"{FirstName} {LastName} ({DateOfBirth.ToShortDateString()})";
        }
        private void CheckName(string value)
        {
            var temp = value.Trim();
            if (string.IsNullOrEmpty(temp) || temp.Contains(" "))
                throw new System.Exception("Tên sai quy cách");
        }
        partial void OnSettingFirstName(string value)
        {
            CheckName(value);
        }
        partial void OnSettingLastName(string value)
        {
            CheckName(value);
        }
    }
}
```

```
namespace PartialMethod
{
    using static System.Console;
    class Program
    {
        static void Main(string[] args)
        {
            var student = new Student
            {
                Id = 1,
```

```
                FirstName = "",
                LastName = "Trump",
                DateOfBirth = new System.DateTime(1990, 12, 31),
                Major = "Computer Science",
                Specialization = "Information Systems"
            };
            WriteLine(student);
            ReadKey();
        }
    }
}
```

Khi dịch và chạy thử, chương trình sẽ báo lỗi khi gặp lệnh gán chuỗi rỗng cho FirstName khi khởi tạo object student.

Trong ví dụ trên, ở file Student.Model.cs chúng ta đã chuyển FirstName và LastName thành full property với hai backed field lần lượt là _firstName và _lastName.

Bạn **khai báo** hai **partial method** OnSettingFirstName và OnSettingLastName:

```
partial void OnSettingFirstName(string value);
partial void OnSettingLastName(string value);
```

Bạn gọi hai partial method trong setter của hai property tương ứng:

```
public string FirstName
 {
     get => _firstName; set {
         OnSettingFirstName(value);
         _firstName = value;
     }
 }
 public string LastName
 {
     get => _lastName; set {
         OnSettingLastName(value);
         _lastName = value;
     }
 }
```

Trong file Student.Methods.cs bạn **thực thi** hai **partial method** này:

```
private void CheckName(string value)
 {
     var temp = value.Trim();
     if (string.IsNullOrEmpty(temp) || temp.Contains(" "))
         throw new System.Exception("Tên sai quy cách");
 }
 partial void OnSettingFirstName(string value)
```

```
{
    CheckName(value);
}
partial void OnSettingLastName(string value)
{
    CheckName(value);
}
```

Logic của các phương thức này rất đơn giản: nếu giá trị gán cho FirstName hoặc LastName chứa dấu cách, hoặc là xâu rỗng thì phát ra exception. Vì lí do này, nếu bạn chạy client code như trên thì chương trình sẽ báo lỗi và mở lại giao diện code ngay.

Trong ví dụ trên, giả sử Student.Model.cs được sinh tự động. Rõ ràng, bạn không muốn gán cứng logic kiểm tra tính hợp lệ của FirstName và LastName mà muốn để cho người dùng tự mình thực hiện logic riêng.

Do đó, bạn khai báo hai partial method và đặt sẵn hai hook (hai lời gọi partial method) ở những vị trí phù hợp. Phần thân của partial method (chứa logic kiểm tra giá trị của FirstName và LastName) để dành cho người lập trình tự viết trong file Student.Methods.cs.

Nếu người lập trình không tự viết phần thực thi thì hook không có ý nghĩa. Compiler bỏ qua lời gọi hook kia. Nếu người lập trình viết phần thực thi, hook sẽ được thực thi như phương thức bình thường.

Các lưu ý khi sử dụng partial method

Partial method chỉ có thể sử dụng bên trong partial class. Không thể sử dụng partial method trong class thông thường.

Partial method bao gồm ba phần (và thường nằm trong các file khác nhau): phần khai báo, phần sử dụng (gọi phương thức), phần thực thi.

Khai báo partial method bắt buộc phải bắt đầu bằng từ khóa **partial**, kết thúc là dấu chấm phẩy sau danh sách tham số và không có thân.

Lời gọi partial method không có gì khác biệt với member method thông thường.

Phần thực thi thường đặt trong một file code khác. Phần thực thi giống hệt như xây dựng một phương thức thông thường nhưng có từ khóa **partial** ở đầu. Ngoài ra, phần thực thi và phần khai báo phải có signature giống hệt nhau.

Partial method bắt buộc phải có return type là **void** và không được có tham số **out**. Tuy nhiên, tham số **ref** vẫn sử dụng được nếu bạn cần giữ lại thay đổi của tham số. Partial method không được sử dụng access modifier như public, private, protected. Nó cũng không được sử dụng các modifier khác như virtual, abstract, sealed, v.v..

Nếu không tìm thấy phần thực thi, compiler sẽ bỏ qua lời gọi partial method (coi như không có gì ở đó!).