View xuất dữ liệu: phương thức, hàm tạo, xâu, xuất thông tin, enum

Trong bài này chúng ta tiếp tục xây dựng class C# giúp xuất thông tin về một cuốn sách ra màn hình console. Qua bài thực hành này bạn sẽ học cách áp dụng phương thức thành viên, xây dựng hàm tạo, xử lý xâu ký tự, làm việc với console, sử dụng kiểu enum.

NỘI DUNG CỦA BÀI [Ẩn]

Phân tích yêu cầu

Trong bài học trước chúng ta đã cùng xây dựng hoàn thiện một class đầu tiên dùng để mô tả thông tin của các cuốn sách điện tử. Đây là một class đặc biệt chỉ mô tả dữ liệu mà không có phương thức xử lý nào.

Trong kiểu tiếp cận hướng nghiệp vụ (domain-driven), loại class này có tên gọi là lớp thực thể (entity class) hay domain model class. Class này sẽ là khuôn mẫu để chúng ta tạo ra các object là dữ liệu về các cuốn sách điện tử cụ thể.

Với cách phân chia code theo mô hình MVC, chúng ta sẽ tạo ra các class riêng giúp hiển thị dữ liệu hoặc nhập dữ liệu (được gọi chung là các lớp view).

Ở bài thực hành đầu tiên chúng ta đã phân tích bài toán và các ca sử dụng. Chúng ta thấy, trước mắt phần mềm này phải có khả năng tương tác với người dùng ở các mặt sau:

1. Hiển thị một cuốn sách cụ thể;
2. Hiển thị một danh sách các cuốn sách điện tử;
3. Nhập thông tin cho một cuốn sách mới;
4. Cập nhật thông tin cho một cuốn sách đang có sẵn;
5. Nhận câu lệnh bất kỳ từ người dùng.

Ứng với mỗi yêu cầu về hiển thị thông tin chúng ta sẽ xây dựng một class riêng biệt. Như vậy chúng ta sẽ phải xây dựng ít nhất bốn class giao diện khác nhau cho các mục đích trên.

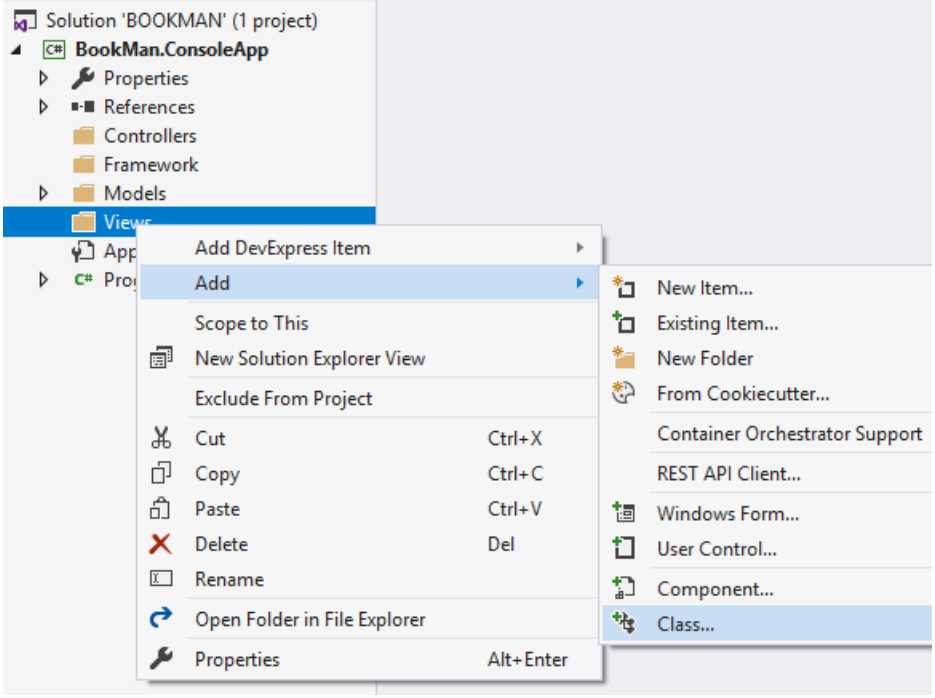
Trong bài này chúng ta sẽ bắt đầu xây dựng class giúp hiển thị thông tin về một cuốn sách cụ thể.

Thực hành 1: xây dựng class hiển thị thông tin về một cuốn sách

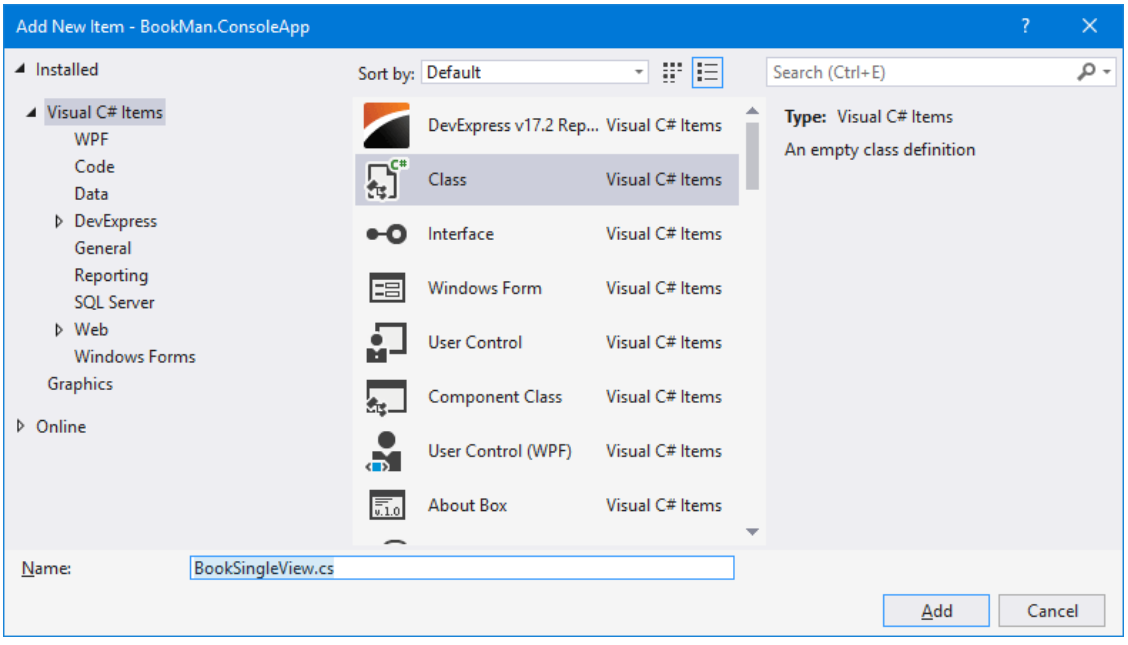
Trong phần thực hành này chúng ta bước đầu tạo ra một class mới có khả năng hiển thị thông tin chi tiết của một cuốn sách.

Tạo một file mã nguồn mới cho class

Click phải vào thư mục Views, chọn Add => Class …



Trong hộp thoại "Add New Item" chọn loại file là "Class", mục "Name" nhập tên "BookSingleView.cs" Ấn nút "Add" để hoàn tất.



Tạo file mã nguồn mới cho class BookSingleView

Một file mã nguồn "BookSingleView.cs" đã được tạo ra trong thư mục Views. Trong file mã nguồn này Visual Studio đã sinh sẵn code cho một class có cùng tên với file:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BookMan.ConsoleApp.Views
{
    class BookSingleView
    {
    }
}
```

*** Lưu ý khi đặt tên file và class**

Khi đặt tên file/class chúng ta nên tự đặt ra một quy luật nào đó để nếu sau này xuất hiện class mới cùng loại chúng ta sẽ dễ dàng chọn được tên phù hợp. Chúng ta đặt ra quy ước sau đối với tên các lớp view. Tên của mỗi lớp view gồm 3 phần:

- tên của các class (và cũng là tên file) của các lớp view bắt đầu bằng tên loại dữ liệu mà nó hiển thị (trong trường hợp này là Book),
- phần tiếp theo chứa một từ mô tả đặc thù của giao diện (ví dụ, lớp view hiển thị một object của Book sẽ có chứa từ Single),
- tên gọi kết thúc là View (để sau này sử dụng nếu nhìn thấy class tận cùng là View, chúng ta sẽ hình dung ra ngay nhiệm vụ của nó).

Như vậy, sau này nếu cần xây dựng lớp view để hiển thị danh sách các cuốn sách, có thể dễ dàng nghĩ ngay tới tên gọi "BookListView", class để tạo cuốn sách mới sẽ là "BookAddView" hay "BookCreateView", v.v..

Viết code cho class BookSingleView

Bổ sung code cho lớp `BookSingleView` như sau (chú ý đọc kỹ các chú thích)

```
using System;
// bốn dòng using dưới đây là thừa và có thể xóa đi (sử dụng Quick Action)
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BookMan.ConsoleApp.Views // chú ý cách Visual Studio đặt tên namespace
{
    using Models; // chú ý cách dùng using bên trong namespace

    /// <summary>
    /// class để hiển thị một cuốn sách, chỉ sử dụng trong dự án (internal)
    /// </summary>
    internal class BookSingleView
    {
        protected Book Model; // biến này để lưu trữ thông tin cuốn sách đang cần hiển t

        /// <summary>
        /// đây là hàm tạo, sẽ được gọi đầu tiên khi tạo object
        /// </summary>
        /// <param name="model">cuốn sách cụ thể sẽ được hiển thị</param>
        public BookSingleView(Book model)
        {
            Model = model; // chuyển dữ liệu từ tham số sang biến thành viên để sử dụng
        }
```

```
27.
28.          /// <summary>
29.          /// thực hiện in thông tin ra màn hình console
30.          /// </summary>
31.          public void Render()
32.          {
33.              if (Model == null) // kiểm tra xem object có dữ liệu không
34.              {
35.                  Console.ForegroundColor = ConsoleColor.Red; // đổi màu chữ sang đỏ
36.                  Console.WriteLine("NO BOOK FOUND. SORRY!"); // in ra dòng thông báo
37.                  Console.ResetColor(); // trả lại màu chữ mặc định
38.                  return; // kết thúc thực hiện phương thức (bỏ qua phần còn lại)
39.              }
40.
41.              Console.ForegroundColor = ConsoleColor.Green;
42.              Console.WriteLine("BOOK DETAIL INFORMATION");
43.              Console.ResetColor();
44.
45.              /* các dòng dưới đây viết ra thông tin cụ thể theo từng dòng
46.               * sử dụng cách tạo xâu kiểu "interpolation"
47.               * và dùng dấu cách để căn chỉnh tạo thẩm mỹ
48.               */
49.              Console.WriteLine($"Authors:     {Model.Authors}");
50.              Console.WriteLine($"Title:       {Model.Title}");
51.              Console.WriteLine($"Publisher:   {Model.Publisher}");
52.              Console.WriteLine($"Year:        {Model.Year}");
53.              Console.WriteLine($"Edition:     {Model.Edition}");
54.              Console.WriteLine($"Isbn:        {Model.Isbn}");
55.              Console.WriteLine($"Tags:        {Model.Tags}");
56.              Console.WriteLine($"Description: {Model.Description}");
57.              Console.WriteLine($"Rating:      {Model.Rating}");
58.              Console.WriteLine($"Reading:     {Model.Reading}");
59.              Console.WriteLine($"File:        {Model.File}");
60.              Console.WriteLine($"File Name:   {Model.FileName}");
61.          }
62.      }
63.  }
```

Ở đây bạn đã xây dựng một class C# "bình thường" và quen thuộc hơn, bao gồm biến thành viên (biến `Model`), phương thức thành viên (phương thức `Render`), và hàm tạo (constructor) của class (`BookSingleView(Book model)`).

Trong thân phương thức `Render()` bạn cũng xây dựng một số property và sử dụng các phương thức xuất nhập với giao diện dòng lệnh.

Sự tồn tại của cả dữ liệu (biến thành viên) và phương thức xử lý dữ liệu (phương thức thành viên) trong khuôn khổ một class là biểu hiện của một nguyên lý trong lập trình hướng đối tượng: *đóng gói* (encapsulation).

Trong class này chúng ta bắt đầu sử dụng class "`Book`" được xây dựng ở bài trước để khai báo object. Vì "`Book`" là một class – một kiểu dữ liệu người dùng tự định nghĩa, nó có thể được sử dụng như bất kỳ kiểu dữ liệu nào (như `int`, `bool`, `string` mà ta đã biết).

Logic của class này như sau:

1. Khi người sử dụng class khởi tạo object (sẽ xem xét ở bài sau) của lớp `BookSingleView` sẽ phải cung cấp một object của lớp `Book` (tức là một cuốn sách cụ thể);
2. Hàm tạo (sẽ xem xét ở bài sau) sẽ chuyển object này sang một biến cục bộ (biến `Model`) để sử dụng trong toàn class `BookSingleView`;
3. Phương thức `Render()` sẽ kiểm tra xem object có dữ liệu hay không.
4. Nếu object không chứa dữ liệu (thực tế phải hiểu object này không chỉ tới một vùng nhớ nào, tức nó là một object trống, C# gọi những object như vậy là null object), phương thức sẽ viết dòng thông báo màu đỏ.
5. Nếu object chứa dữ liệu, giá trị của từng trường dữ liệu (là property) sẽ được viết ra màn hình.

Thực hành 2: xây dựng thêm phương thức mới

Trong phương thức `Render()` ở trên chúng ta thấy có nhóm code sau bị trùng

```
Console.ForegroundColor = ConsoleColor.Red; // đổi màu chữ sang đỏ
Console.WriteLine("NO BOOK FOUND. SORRY!"); // in ra dòng thông báo
Console.ResetColor(); // trả lại màu chữ mặc định
và
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("BOOK DETAIL INFORMATION");
Console.ResetColor();
```

Hai nhóm code này chỉ khác nhau về thông tin và màu sắc của chữ hiển thị ra.

Giả sử chúng ta cần viết thêm một số thông tin nữa với màu sắc, 3 đoạn code như trên sẽ lặp lại thêm nhiều lần. Điều này vi phạm nguyên lý DRY (Don't Repeat Yourself) – không lặp code.

Trong phạm vị một class, nếu có nhiều dòng code giống nhau về logic như vậy và chỉ khác về tham số, chúng ta có thể nghĩ tới xây dựng một phương thức nội bộ để tránh lặp code.

Xây dựng phương thức in ra console với màu sắc

Bổ sung thêm phương thức sau vào cuối class `BookSingleView`:

```
/// <summary>
/// in thông báo ra màn hình console với chữ màu
/// </summary>
/// <param name="message">thông báo</param>
/// <param name="color">màu</param>
protected void WriteLine(string message, ConsoleColor color)
{
    Console.ForegroundColor = color;
    Console.WriteLine(message);
    Console.ResetColor();
}
```

Sử dụng phương thức mới xây dựng

Gọi phương thức mới thay cho các nhóm code cũ trong phương thức Render:

*Thay đoạn code

```
Console.ForegroundColor = ConsoleColor.Red; // đổi màu chữ sang đỏ
Console.WriteLine("NO BOOK FOUND. SORRY!"); // in ra dòng thông báo
Console.ResetColor(); // trả lại màu chữ mặc định
```

bằng

```
// sử dụng phương thức WriteLine vừa tạo thay cho đoạn code cũ
WriteLine("NO BOOK FOUND. SORRY!", ConsoleColor.Red);
```

*Thay đoạn code

```
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("BOOK DETAIL INFORMATION");
Console.ResetColor();
```

Bằng

```
// sử dụng phương thức WriteLine vừa tạo thay cho đoạn code cũ
WriteLine("BOOK DETAIL INFORMATION", ConsoleColor.Green);
```

Code của cả lớp BookSingleView giờ trở thành như sau:

BookSingleView.cs

```
using System;

namespace BookMan.ConsoleApp.Views // chú ý cách Visual Studio đặt tên namespace
{
    using Models; // chú ý cách dùng using bên trong namespace

    /// <summary>
    /// class để hiển thị một cuốn sách
    /// </summary>
    internal class BookSingleView
    {
        protected Book Model; // biến này để lưu trữ thông tin cuốn sách đang cần hiển t

        /// <summary>
        /// đây là hàm tạo, sẽ được gọi đầu tiên khi tạo object
        /// </summary>
        /// <param name="model">cuốn sách cụ thể sẽ được hiển thị</param>
        public BookSingleView(Book model)
        {
            Model = model; // chuyển dữ liệu từ tham số sang biến thành viên để sử dụng
        }

        /// <summary>
        /// thực hiện in thông tin ra màn hình console
        /// </summary>
        public void Render()
        {
            if (Model == null) // kiểm tra xem có dữ liệu không
            {
                // sử dụng phương thức WriteLine vừa tạo thay cho đoạn code cũ
                WriteLine("NO BOOK FOUND. SORRY!", ConsoleColor.Red);
                return; // kết thúc thực hiện phương thức (bỏ qua phần còn lại)
            }

            // sử dụng phương thức WriteLine vừa tạo thay cho đoạn code cũ
            WriteLine("BOOK DETAIL INFORMATION", ConsoleColor.Green);

            /* các dòng dưới đây viết ra thông tin cụ thể theo từng dòng
             * sử dụng cách tạo xâu kiểu "interpolation"
             * và dùng dấu cách để căn chỉnh tạo thẩm mỹ
             */
            Console.WriteLine($"Authors:     {Model.Authors}");
            Console.WriteLine($"Title:       {Model.Title}");
            Console.WriteLine($"Publisher:   {Model.Publisher}");
            Console.WriteLine($"Year:        {Model.Year}");
            Console.WriteLine($"Edition:     {Model.Edition}");
            Console.WriteLine($"Isbn:        {Model.Isbn}");
            Console.WriteLine($"Tags:        {Model.Tags}");
            Console.WriteLine($"Description: {Model.Description}");
```

```
            Console.WriteLine($"Rating:      {Model.Rating}");
            Console.WriteLine($"Reading:     {Model.Reading}");
            Console.WriteLine($"File:        {Model.File}");
            Console.WriteLine($"File Name:   {Model.FileName}");
        }

        /// <summary>
        /// in thông báo ra màn hình console với chữ màu
        /// </summary>
        /// <param name="message">thông báo</param>
        /// <param name="color">màu</param>
        protected void WriteLine(string message, ConsoleColor color)
        {
            Console.ForegroundColor = color;
            Console.WriteLine(message);
            Console.ResetColor();
        }
    }
}
```

Kết luận

Trong bài này chúng ta đã xây dựng một class giao diện giúp hiển thị thông tin của một cuốn sách. Qua đây chúng ta áp dụng cách xây dựng phương thức thành viên trong C#, cách xuất thông tin ra giao diện console, cách định dạng chuỗi ký tự và kiểu liệt kê. Chúng ta đã áp dụng các kiến thức này để xây dựng một class trọn vẹn giúp hiển thị thông tin chi tiết một cuốn sách ra console.