

Hoàn thiện (1): nhập mới, cập nhật, partial class

Hướng dẫn tự học lập trình C# toàn tập > Hoàn thiện (1): nhập mới, cập nhật, partial class

Ở phần trước chúng ta đã xây dựng hoàn chỉnh tất cả các lớp hỗ trợ của chương trình. Trong bài này chúng ta sẽ áp dụng để hoàn thiện các chức năng hiện có: hiển thị (single, list), nhập, cập nhật.

NỘI DUNG CỦA BÀI [Ấn]

- Thực hành 1: hoàn thiện chức năng nhập dữ liệu
 - Bước 1. Thay đổi nội dung của lớp BookCreateView
 - Bước 2. Thay đổi phương thức Create của BookController
 - Bước 3. Điều chỉnh phương thức Main
 - Bước 4. Dịch và chạy thử chương trình với chức năng thêm dữ liệu
- Thực hành 2: cải tiến lớp Program với partial class
 - Bước 1. Thêm mới file Program.Config.cs
 - Bước 2. Viết code cho file Program.Config.cs
 - Bước 3. Điều chỉnh file Program.cs
- Thực hành 3: hoàn thiện chức năng cập nhật dữ liệu
 - Bước 1. Điều chỉnh lớp BookUpdateView
 - Bước 2. Điều chỉnh phương thức Update của BookController
 - Bước 3. Bổ sung route vào ConfigRouter
 - Bước 4. Dịch và chạy thử chương trình với lệnh update
- Kết luận

Thực hành 1: hoàn thiện chức năng nhập dữ liệu

Trong các bài trước chúng ta đang làm dở chức năng nhập dữ liệu vì chưa thể truyền được thông tin người dùng nhập về controller. MVC quy định rằng các view không trực tiếp khởi tạo object. Khởi tạo object là nhiệm vụ của controller.

Do vậy, lớp `BookCreateView` sau khi nhận thông tin từ người dùng phải chuyển thông tin đó về cho `BookController` để xử lý.

Bước 1. Thay đổi nội dung của lớp BookCreateView

BookCreateView.cs

```
1. using System;
2.
3. namespace BookMan.ConsoleApp.Views
4. {
5.     using Framework;
6.
7.     internal class BookCreateView : ViewBase
8.     {
9.         public BookCreateView() { }
10.
11.         public override void Render()
12.         {
13.             ViewHelp.WriteLine("CREATE A NEW BOOK", ConsoleColor.Green);
14.
15.             var title = ViewHelp.InputString("Title");
16.             var authors = ViewHelp.InputString("Authors");
17.             var publisher = ViewHelp.InputString("Publisher");
18.             var year = ViewHelp.InputInt("Year");
19.             var edition = ViewHelp.InputInt("Edition");
20.             var tags = ViewHelp.InputString("Tags");
```

```

21.     var description = ViewHelp.InputString("Description");
22.     var rate = ViewHelp.InputInt("Rate");
23.     var reading = ViewHelp.InputBool("Reading");
24.     var file = ViewHelp.InputString("File");
25.
26.     var request =
27.         "do create ? " +
28.         $"title = {title}" +
29.         $" & authors = {authors}" +
30.         $" & publisher = {publisher}" +
31.         $" & year = {year}" +
32.         $" & edition = {edition}" +
33.         $" & tags = {tags}" +
34.         $" & description = {description}" +
35.         $" & rate = {rate}" +
36.         $" & reading = {reading}" +
37.         $" & file = {file}";
38.     Router.Forward(request);
39. }
40. }
41. }

```

Ở bước này chúng ta tạo ra một truy vấn với route là "do create" và danh sách tham số chứa tất cả giá trị mà người dùng nhập. Truy vấn này sẽ được router gửi ngược về controller.

Bước 2. Thay đổi phương thức Create của BookController

```

1.     public void Create(Book book = null)
2.     {
3.         if (book == null)
4.         {
5.             Render(new BookCreateView());
6.             return;
7.         }
8.
9.         Repository.Insert(book);
10.        Success("Book created!");
11.    }

```

Ở bước này chúng ta thay đổi phương thức Create để có thể thực hiện hai nhiệm vụ:

1. nếu tham số book nhận giá trị null thì sẽ kích hoạt `BookCreateView` để người dùng nhập thông tin;
2. nếu book khác null thì yêu cầu Repository thêm dữ liệu và hiển thị thông báo thành công.

Lưu ý bổ sung `using Models;` ở ngay sau khai báo namespace.

```

1.     namespace BookMan.ConsoleApp.Controllers
2.     {
3.         using Models;
4.         using DataServices;
5.         using Framework;
6.         using Views;
7.         ...

```

Bước 3. Điều chỉnh phương thức Main

```

1.     private static void Main(string[] args)
2.     {
3.         Console.OutputEncoding = System.Text.Encoding.UTF8;
4.
5.         SimpleDataAccess context = new SimpleDataAccess();
6.         BookController controller = new BookController(context);
7.
8.         Router r = Router.Instance;
9.
10.        r.Register("about", About);

```

```

11.     r.Register("help", Help);
12.     r.Register(route: "create",
13.         action: p => controller.Create(),
14.         help: "[create]\r\nnhập sách mới");
15.     r.Register(route: "do create",
16.         action: p => controller.Create(toBook(p)),
17.         help: "this route should be used only in code");
18.     r.Register(route: "update",
19.         action: p => controller.Update(p["id"].ToInt()),
20.         help: "[update ? id = <value>]\r\ntim và cập nhật sách");
21.     r.Register(route: "list",
22.         action: p => controller.List(),
23.         help: "[list]\r\nhiển thị tất cả sách");
24.     r.Register(route: "single",
25.         action: p => controller.Single(p["id"].ToInt()),
26.         help: "[single ? id = <value>]\r\nhiển thị một cuốn sách theo id");
27.
28.     r.Register(route: "list file",
29.         action: p => controller.List(p["path"]),
30.         help: "[list file ? path = <value>]\r\nhiển thị tất cả sách");
31.     r.Register(route: "single file",
32.         action: p => controller.Single(p["id"].ToInt(), p["path"]),
33.         help: "[single file ? id = <value> & path = <value>]");
34.
35.     //hàm cục bộ để chuyển object của Parameter sang object của Book
36.     Models.Book toBook(Parameter p)
37.     {
38.         var b = new Models.Book();
39.         if (p.ContainsKey("id")) b.Id = p["id"].ToInt();
40.         if (p.ContainsKey("authors")) b.Authors = p["authors"];
41.         if (p.ContainsKey("title")) b.Title = p["title"];
42.         if (p.ContainsKey("publisher")) b.Publisher = p["publisher"];
43.         if (p.ContainsKey("year")) b.Year = p["year"].ToInt();
44.         if (p.ContainsKey("edition")) b.Edition = p["edition"].ToInt();
45.         if (p.ContainsKey("isbn")) b.Isbn = p["isbn"];
46.         if (p.ContainsKey("tags")) b.Tags = p["tags"];
47.         if (p.ContainsKey("description")) b.Description = p["description"];
48.         if (p.ContainsKey("file")) b.File = p["file"];
49.         if (p.ContainsKey("rate")) b.Rating = p["rate"].ToInt();
50.         if (p.ContainsKey("reading")) b.Reading = p["reading"].ToBool();
51.         return b;
52.     }
53.
54.     while (true)
55.     {
56.         ViewHelp.Write("# Request >>> ", ConsoleColor.Green);
57.         string request = Console.ReadLine();
58.         Router.Instance.Forward(request);
59.
60.         Console.WriteLine();
61.     }
62. }

```

Ở bước này chúng ta thêm một route nữa để gọi phương thức Create của BookController.

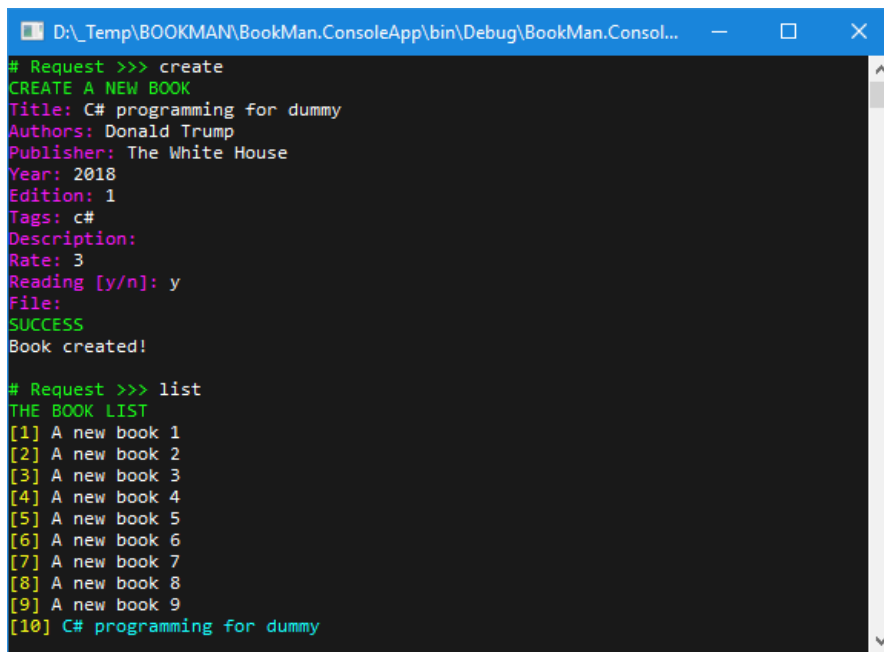
```

1.     r.Register(route: "do create",
2.         action: p => controller.Create(toBook(p)),
3.         help: "this route should be used only in code");

```

Để đơn giản hóa việc chuyển object của Parameter sang object của Book, chúng ta viết thêm một hàm cục bộ `toBook` ở ngay trong thân của Main.

Bước 4. Dịch và chạy thử chương trình với chức năng thêm dữ liệu



```
# Request >>> create
CREATE A NEW BOOK
Title: C# programming for dummy
Authors: Donald Trump
Publisher: The White House
Year: 2018
Edition: 1
Tags: c#
Description:
Rate: 3
Reading [y/n]: y
File:
SUCCESS
Book created!

# Request >>> list
THE BOOK LIST
[1] A new book 1
[2] A new book 2
[3] A new book 3
[4] A new book 4
[5] A new book 5
[6] A new book 6
[7] A new book 7
[8] A new book 8
[9] A new book 9
[10] C# programming for dummy
```

Kết quả chạy chương trình với lệnh `create`

Thực hành 2: cải tiến lớp Program với partial class

Trong phần thực hành 1 chúng ta nhận thấy rằng phương thức Main đang mở rộng ra rất nhanh khi chúng ta bổ sung thêm các route mới để hoàn thiện các chức năng xử lý dữ liệu.

Trong các phần thực hành tiếp theo, số lượng code ở phương thức này còn tăng lên nữa. Code được bổ sung đều là code để đăng ký route cho các chức năng mới. Chúng ta sẽ tìm cách tách rời phần đăng ký route này sang một file riêng để dễ dàng thay đổi về sau.

Ở đây chúng ta áp dụng một giải pháp đơn giản: sử dụng *partial class*.

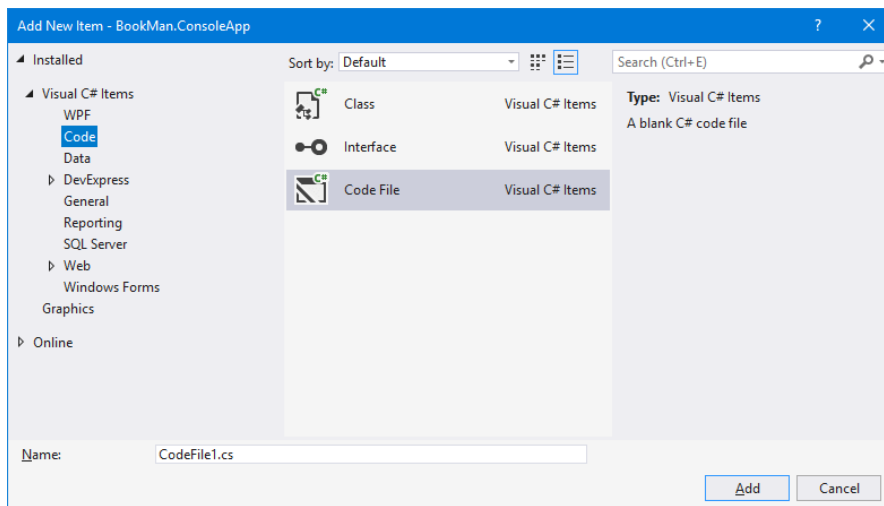
Partial class là một tính năng của C# cho phép định nghĩa một class trên nhiều file vật lý khác nhau.

Từ đầu đến giờ, mỗi class đều được xây dựng trong một file đặt trùng tên với class. Nếu trong class có nhiều logic khác nhau, chúng ta hoàn toàn có thể xem xét tách các logic đó sang các file riêng rẽ sử dụng 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.

Bước 1. Thêm mới file Program.Config.cs

1. Click phải vào project BookMan.ConsoleApp, chọn Add => New Item;
2. Trong cửa sổ Add New Item, chọn nhánh Code bên trong Visual C# Items (bên tay trái);
3. Chọn mục Code File trong danh sách bên tay phải



Tạo code file mới

Đặt tên file là Program.Config.cs và ấn nút Add.

Bước 2. Viết code cho file Program.Config.cs

```
1. namespace BookMan.ConsoleApp
2. {
3.     using Models;
4.     using Controllers;
5.     using DataServices;
6.     using Framework;
7.
8.     internal partial class Program
9.     {
10.         private static void ConfigRouter()
11.         {
12.             SimpleDataAccess context = new SimpleDataAccess();
13.
14.             BookController controller = new BookController(context);
15.
16.             Router r = Router.Instance;
17.
18.             r.Register("about", About);
19.             r.Register("help", Help);
20.             r.Register(route: "create",
21.                 action: p => controller.Create(),
22.                 help: "[create]rnnhập sách mới");
23.             r.Register(route: "do create",
24.                 action: p => controller.Create(toBook(p)),
25.                 help: "this route should be used only in code");
26.
27.             r.Register(route: "list",
28.                 action: p => controller.List(),
29.                 help: "[list]rnnhiên thị tất cả sách");
30.             r.Register(route: "list file",
31.                 action: p => controller.List(p["path"]),
32.                 help: "[list file ? path = <value>]rnnhiên thị tất cả sách");
33.
34.             r.Register(route: "single",
35.                 action: p => controller.Single(p["id"].ToInt()),
36.                 help: "[single ? id = <value >]rnnhiên thị một cuốn sách theo id");
37.             r.Register(route: "single file",
38.                 action: p => controller.Single(p["id"].ToInt(), p["path"]),
39.                 help: "[single file ? id = <value> & path = <value>]");
40.
41.
42.             //r.Register(route: "",
43.             //    action: null,
44.             //    help: "");
45.
46.             #region helper
47.
48.             //local function to convert parameter to book object
49.             Book toBook(Parameter p)
```

```

50.     {
51.         var b = new Book();
52.         if (p.ContainsKey("id")) b.Id = p["id"].ToInt();
53.         if (p.ContainsKey("authors")) b.Authors = p["authors"];
54.         if (p.ContainsKey("title")) b.Title = p["title"];
55.         if (p.ContainsKey("publisher")) b.Publisher = p["publisher"];
56.         if (p.ContainsKey("year")) b.Year = p["year"].ToInt();
57.         if (p.ContainsKey("edition")) b.Edition = p["edition"].ToInt();
58.         if (p.ContainsKey("isbn")) b.Isbn = p["isbn"];
59.         if (p.ContainsKey("tags")) b.Tags = p["tags"];
60.         if (p.ContainsKey("description")) b.Description = p["description"];
61.         if (p.ContainsKey("file")) b.File = p["file"];
62.         if (p.ContainsKey("rate")) b.Rating = p["rate"].ToInt();
63.         if (p.ContainsKey("reading")) b.Reading = p["reading"].ToBool();
64.         return b;
65.     }
66.
67.     #endregion
68. }
69.
70. }

```

Ở bước này chúng ta khai báo thêm một class Program nữa và nằm trong cùng không gian tên với class Program đã có sẵn trong file Program.cs.

Chúng ta thêm từ khóa `partial` vào trước định nghĩa của lớp Program. Nếu không có từ khóa `partial` ở đây thì C# sẽ báo lỗi vì có hai class trùng tên nhau trong cùng một không gian tên.

Trong phần này của lớp Program này định nghĩa thêm phương thức static `ConfigRouter` và di chuyển một phần của phương thức Main sang phương thức mới `ConfigRouter`.

Bước 3. Điều chỉnh file Program.cs

```

1.  using System;
2.
3.  namespace BookMan.ConsoleApp
4.  {
5.      using Framework;
6.
7.      internal partial class Program
8.      {
9.          private static void Main(string[] args)
10.         {
11.             Console.OutputEncoding = System.Text.Encoding.UTF8;
12.
13.             ConfigRouter();
14.
15.             while (true)
16.             {
17.                 ViewHelp.Write("# Request >>> ", ConsoleColor.Green);
18.                 string request = Console.ReadLine();
19.                 Router.Instance.Forward(request);
20.
21.                 Console.WriteLine();
22.             }
23.         }
24.
25.         private static void Help(Parameter parameter)
26.         {
27.             if (parameter == null)
28.             {
29.                 ViewHelp.WriteLine("SUPPORTED COMMANDS:", ConsoleColor.Green);
30.                 ViewHelp.WriteLine(Router.Instance.GetRoutes(), ConsoleColor.Yellow);
31.                 ViewHelp.WriteLine("type: help ? cmd= <command> to get command details",
32.                                     return;
33.             }
34.             Console.BackgroundColor = ConsoleColor.DarkBlue;
35.             var command = parameter["cmd"].ToLower();
36.             ViewHelp.WriteLine(Router.Instance.GetHelp(command));
37.         }
38.
39.         private static void About(Parameter parameter)

```

```

40.         {
41.             ViewHelp.WriteLine("BOOK MANAGER version 1.0", ConsoleColor.Green);
42.             ViewHelp.WriteLine("by Mype Nguyen @ ictu.edu.vn", ConsoleColor.Magenta);
43.         }
44.     }
45. }

```

Ở bước này chúng ta thêm từ khóa `partial` vào trước định nghĩa của class `Program`, đồng thời gọi phương thức `ConfigureRouter` đã định nghĩa ở phần khác của lớp `Program` trong file `Program.Config.cs`.

Đây có thể xem như thành phần “tĩnh” hơn của `Program`. Toàn bộ những code khác của `Main` đã di chuyển sang phương thức `ConfigureRouter` (thành phần “động” của `Program`).

Partial class đặc biệt hữu ích khi sử dụng chung với các công cụ sinh code tự động. Phần class sinh ra tự động thường được định nghĩa sẵn với từ khóa `partial`. Người dùng có thể tự định nghĩa phần của riêng mình mà không ảnh hưởng đến phần sinh tự động. Khi phần sinh tự động thay đổi, phần do người dùng định nghĩa không bị ảnh hưởng. Khi học đến công nghệ windows forms chúng ta sẽ thường xuyên gặp partial class.

Như vậy, qua phần thực hành vừa rồi chúng ta đã điều chỉnh lớp `Program` thành dạng partial class nằm trên hai file: `Program.cs` và `Program.Config.cs`. Mặc dù nằm ở hai file khác nhau nhưng lại là cùng một class. File `Program.cs` chứa thành phần ít biến đổi (thành phần tĩnh hơn); file `Program.Config.cs` chứa thành phần thường biến động (các route được liên tục bổ sung khi phát triển các chức năng của ứng dụng).

Thực hành 3: hoàn thiện chức năng cập nhật dữ liệu

Tương tự như chức năng thêm dữ liệu, chức năng cập nhật dữ liệu trước đây cũng không truyền được thông tin về controller. Trong phần thực hành này chúng ta sẽ xây dựng hoàn thiện.

Bước 1. Điều chỉnh lớp `BookUpdateView`

```

1.  using System;
2.
3.  namespace BookMan.ConsoleApp.Views
4.  {
5.      using Framework;
6.      using Models;
7.
8.      internal class BookUpdateView : ViewBase<Book>
9.      {
10.         public BookUpdateView(Book model) : base(model)
11.         {
12.         }
13.
14.         public override void Render()
15.         {
16.             ViewHelp.WriteLine("UPDATE BOOK INFORMATION", ConsoleColor.Green);
17.
18.             var authors = ViewHelp.InputString("Authors", Model.Authors);
19.             var title = ViewHelp.InputString("Title", Model.Title);
20.             var publisher = ViewHelp.InputString("Publisher", Model.Publisher);
21.             var isbn = ViewHelp.InputString("Isbn", Model.Isbn);
22.             var tags = ViewHelp.InputString("Tags", Model.Tags);
23.             var description = ViewHelp.InputString("Description", Model.Description);
24.             var file = ViewHelp.InputString("File", Model.File);
25.             var year = ViewHelp.InputInt("Year", Model.Year);

```

```

26.         var edition = ViewHelp.InputInt("Edition", Model.Edition);
27.         var rating = ViewHelp.InputInt("Rate", Model.Rating);
28.         var reading = ViewHelp.InputBool("Reading", Model.Reading);
29.
30.         var request =
31.             "do update ? " +
32.             $"id = {Model.Id}" +
33.             $"& title = {title}" +
34.             $"& authors = {authors}" +
35.             $"& publisher = {publisher}" +
36.             $"& year = {year} &" +
37.             $"& edition = {edition}" +
38.             $"& tags = {tags}" +
39.             $"& description = {description}" +
40.             $"& rate = {rating}" +
41.             $"& reading = {reading}" +
42.             $"& file = {file}";
43.         Router.Forward(request);
44.     }
45. }
46. }

```

Bước điều chỉnh này tương tự như đối với lớp BookCreateView: tạo và gửi truy vấn về controller.

Bước 2. Điều chỉnh phương thức Update của BookController

```

1. public void Update(int id, Book book = null)
2. {
3.     if (book == null)
4.     {
5.         var model = Repository.Select(id);
6.         var view = new BookUpdateView(model);
7.         Render(view);
8.         return;
9.     }
10.
11.     Repository.Update(id, book);
12.     Success("Book updated!");
13. }

```

Tương tự với việc điều chỉnh phương thức Create, bước điều chỉnh này giúp phương thức Update làm việc với hai giai đoạn của việc cập nhật:

1. nếu chỉ cung cấp id mà không cung cấp một object của lớp Book thì chỉ kích hoạt giao diện BookUpdateView;
2. nếu cung cấp cả một object của lớp Book thì sẽ thực sự cập nhật thông tin sách và thông báo kết quả lại cho người dùng.

Bước 3. Bổ sung route vào ConfigRouter

```

1. r.Register(route: "update",
2.     action: p => controller.Update(p["id"].ToInt()),
3.     help: "[update ? id = <value>]rntim và cập nhật sách");
4. r.Register(route: "do update",
5.     action: p => controller.Update(p["id"].ToInt(), toBook(p)),
6.     help: "this route should be used only in code");

```

Bước 4. Dịch và chạy thử chương trình với lệnh update

Kết quả
thực
hiện

Kết luận

Trong bài học này chúng ta đã vận dụng các lớp hỗ trợ xây dựng trong bài trước để hoàn thành hai chức năng chính của ứng dụng: Thêm mới và cập nhật thông tin sách. Chúng ta cũng vận dụng partial class để tách nội dung lớp Program ra hai file vật lý khác nhau.

Trong bài tiếp theo chúng ta sẽ tiếp tục hoàn thiện và bổ sung một số chức năng theo phân tích.

- + 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!