

# Hoàn thiện (2): xóa, lọc, tìm kiếm, xử lý file

Hướng dẫn tự học lập trình C# toàn tập > Hoàn thiện (2): xóa, lọc, tìm kiếm, xử lý file

Trong bài học này chúng ta tiếp tục hoàn thiện các chức năng chính như đã phân tích, bao gồm: bổ sung chức năng xóa dữ liệu, lọc dữ liệu, tự động tìm sách trong thư mục, mở file pdf từ chương trình, đánh dấu (bookmark) các cuốn sách đang đọc.

## NỘI DUNG CỦA BÀI [ Ấn ]

- Thực hành 1: thêm chức năng xóa dữ liệu
  - Bước 1. Tạo phương thức Delete
  - Bước 2. Bổ sung thêm route
  - Bước 3. Dịch và chạy thử với lệnh delete
- Thực hành 2: thêm chức năng lọc dữ liệu
  - Bước 1. Thêm phương thức Filter vào lớp BookController
  - Bước 2. Bổ sung thêm route
  - Bước 3. Dịch và chạy thử chương trình với lệnh filter
- Thực hành 3: thêm chức năng tìm sách trong thư mục
  - Bước 1. Xây dựng lớp ShellController
  - Bước 2. Bổ sung code cho ConfigRouter
  - Bước 3. Bổ sung route mới cho ConfigRouter
  - Bước 4. Dịch và chạy thử chương trình
  - Lớp Directory
  - Lớp Path
- Thực hành 4: thêm chức năng đọc sách từ chương trình
  - Bước 1. Thêm phương thức vào ShellController
  - Bước 2. Bổ sung route vào ConfigRouter
  - Bước 3. Dịch và chạy thử chương trình
- Thực hành 5: đánh dấu những cuốn sách đang đọc
  - Bước 1. Bổ sung phương thức vào lớp Repository
  - Bước 2. Bổ sung hai phương thức vào lớp BookController
  - Bước 3. Bổ sung route vào ConfigRouter
  - Bước 4. Dịch và chạy thử chương trình
- Thực hành 6: bổ sung khả năng xóa toàn bộ dữ liệu
  - Bước 1. Thêm phương thức vào lớp Repository
  - Bước 2. Thêm phương thức Clear vào ShellController
  - Bước 3. Bổ sung các route
  - Bước 4. Dịch và chạy thử chương trình với lệnh clear
- Kết luận

## Thực hành 1: thêm chức năng xóa dữ liệu

Chức năng xóa dữ liệu chưa được thực hiện ở các bài trước đây. Ở chức năng này chúng ta không xây dựng một lớp view riêng mà tận dụng khả năng của lớp `MessageView` (đã xây dựng ở [bài này](#)).

### Bước 1. Tạo phương thức Delete

Xây dựng phương thức `Delete` trong lớp `BookController` như sau:

```
1. public void Delete(int id, bool process = false)
2. {
3.     if (process == false)
4.     {
5.         var b = Repository.Select(id);
```

```

6.         Confirm($"Do you want to delete this book ({b.Title})? ", $"do delete?id={b.Id} ")
7.     }
8.     else
9.     {
10.         Repository.Delete(id);
11.         Success("Book deleted!");
12.     }
13. }

```

Trong phương thức `Delete` chúng ta vận dụng lớp `Message` và `MessageView` đã xây dựng ở bài trước để đưa ra các thông báo ngăn cho người dùng mà không cần xây dựng một lớp view riêng cho phương thức `Delete`.

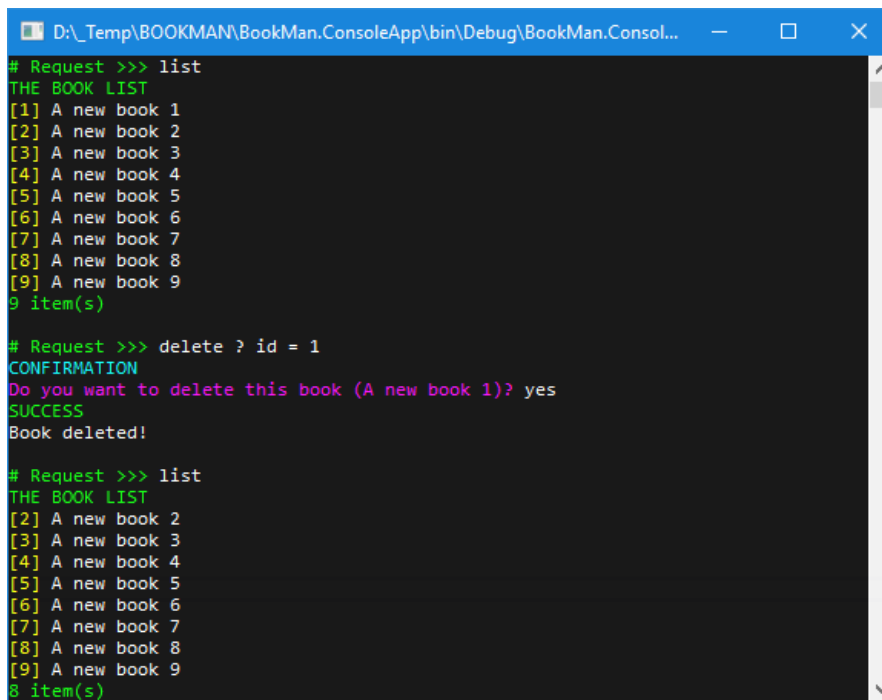
## Bước 2. Bổ sung thêm route

```

1. r.Register(route: "delete",
2.     action: p => controller.Delete(p["id"].ToInt()),
3.     help: "[delete ? id = <value>]");
4. r.Register(route: "do delete",
5.     action: p => controller.Delete(p["id"].ToInt(), true),
6.     help: "this route should be used only in code");

```

## Bước 3. Dịch và chạy thử với lệnh delete



```

# 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
9 item(s)

# Request >>> delete ? id = 1
CONFIRMATION
Do you want to delete this book (A new book 1)? yes
SUCCESS
Book deleted!

# Request >>> list
THE BOOK LIST
[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
8 item(s)

```

Kết quả thực hiện lệnh `delete`

## Thực hành 2: thêm chức năng lọc dữ liệu

Ở phần thực hành này chúng ta bổ sung thêm một tính năng mới: lọc dữ liệu. Ở chức năng này, người dùng cung cấp một từ khóa bất kỳ. Chương trình sẽ tìm trong danh sách dữ liệu tất cả những cuốn sách mà tiêu đề, tác giả, nhà xuất bản, tag và mô tả có chứa từ khóa này.

### Bước 1. Thêm phương thức Filter vào lớp BookController

```

1. public void Filter(string key)
2. {
3.     var model = Repository.Select(key);
4.     if (model.Length == 0)
5.         Inform("No matched book found!");
6.     else
7.         Render(new BookListView(model));
8. }

```

## Bước 2. Bổ sung thêm route

```

1. r.Register(route: "filter",
2.     action: p => controller.Filter(p["key"]),
3.     help: "[filter ? key = <value>]rntim sách theo từ khóa");

```

## Bước 3. Dịch và chạy thử chương trình với lệnh filter

```

# 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
9 item(s)

# Request >>> filter ? key = book 1
THE BOOK LIST
[1] A new book 1
1 item(s)

# Request >>> _

```

Kết quả thực hiện lệnh `filter`

Khi lọc dữ liệu, bạn cũng có thể muốn sắp xếp dữ liệu theo tiêu chí nào đó, ví dụ theo tựa sách, theo tác giả, theo năm xuất bản. Bạn có thể đọc thêm về [các thuật toán sắp xếp](#) để tự thực hiện.

## Thực hành 3: thêm chức năng tìm sách trong thư mục

Như đã mô tả ở bài 1, đây là chức năng giúp xây dựng dữ liệu sách tự động. Người dùng cung cấp một đường dẫn tới thư mục chứa các file sách, chương trình sẽ phát hiện tất cả các file pdf có trong đó và sử dụng tên và đường dẫn của các file này để tạo ra dữ liệu cơ bản về kho sách.

Trong phần thực hành này, chúng ta sẽ tạo ra thêm một lớp điều khiển mới để thực hiện các chức năng liên quan tới file, thư mục.

### Bước 1. Xây dựng lớp ShellController

Tạo mới file `ShellController.cs` trong thư mục `Controllers` dành cho lớp `ShellController` và viết code cho lớp `ShellController` như sau:

```
ShellController.cs
1. using System.Diagnostics;
2. using System.IO;
3. namespace BookMan.ConsoleApp.Controllers
4. {
5.     using DataServices;
6.     using Models;
7.     using Views;
8.     using Framework;
9.
10.    internal class ShellController : ControllerBase
11.    {
12.        protected Repository Repository;
13.        public ShellController(SimpleDataAccess context)
14.        {
15.            Repository = new Repository(context);
16.        }
17.
18.        public void Shell(string folder, string ext = "*.pdf")
19.        {
20.            if (!Directory.Exists(folder))
21.            {
22.                Error("Folder not found!");
23.                return;
24.            }
25.            var files = Directory.GetFiles(folder, ext ?? "*.pdf", SearchOption.AllDirectories);
26.            foreach (var f in files)
27.            {
28.                Repository.Insert(new Book { Title = Path.GetFileNameWithoutExtension(f) });
29.            }
30.            if (files.Length > 0)
31.            {
32.                //Render(new BookListView(Repository.Select()));
33.                Success($"{files.Length} item(s) found!");
34.                return;
35.            }
36.            Inform("No item found!", "Sorry!");
37.        }
38.    }
39. }
```

Ở bước này chúng ta xây dựng lớp `ShellController` kế thừa từ lớp `ControllerBase` đã xây dựng ở các bài trước.

## Bước 2. Bổ sung code cho ConfigRouter

```
1. BookController controller = new BookController(context);
2. ShellController shell = new ShellController(context);
```

Bước này chúng ta chỉ bổ sung khai báo và khởi tạo một object của `ShellController`. Object này dùng để thực hiện các chức năng mới liên quan đến file và thư mục.

## Bước 3. Bổ sung route mới cho ConfigRouter

```
1. r.Register(route: "add shell",
2.    action: p => shell.Shell(p["path"], p["ext"]),
3.    help: "[add shell ? path = <value>]");
```

## Bước 4. Dịch và chạy thử chương trình

```
G:\OneDrive\filecodes\Learn C# in mini-project\BOOKMAN\BookMan.ConsoleApp\bin\Debug\BookMan.ConsoleApp.exe
# Request >>> add shell ? path = E:\OneDrive - ictu.edu.vn\BOOKS\MATH\APPLIED MATHEMATICS
SUCCESS
12 item(s) found!
# Request >>> add shell ? path = E:\OneDrive - ictu.edu.vn\BOOKS\MATH\CALCULUS
SUCCESS
10 item(s) found!
# Request >>> add shell ? path = E:\OneDrive - ictu.edu.vn\BOOKS\MATH\DIFFERENTIAL EQUATIONS
SUCCESS
7 item(s) found!
# Request >>> add shell ? path = E:\OneDrive - ictu.edu.vn\BOOKS\MATH\DISCRETE MATHEMATICS
SUCCESS
0 item(s) found!
# Request >>> list
# new 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] [Duffy Dean G.] Advanced Engineering Mathematics - CRC Press, 1997
[11] [Gupta C.B., Malik A.K., Kumar V.] Advanced Mathematics - New Age International, 2009
[12] [John Bird] Engineering Mathematics - Elsevier, 2007
[13] [Lee de Haan, Toon Koppelaar] Applied Mathematics For Database Professionals - Apress, 2007
[14] [Peter J. Olver, Chehrzad Shakiban] Applied Mathematics - John Wiley & Sons, 2004
[15] [Sean Mauch] Advanced Mathematical Methods for Scientists and Engineers - Mauch Publishing, 2003
[16] [Sean Mauch] Introduction to Methods of Applied Mathematics - Mauch Publishing, 2004
[17] [Sean Mauch] Methods of Applied Mathematics - Mauch Publishing, 2004
[18] [Steven Ian Barry, Stephen Alan Davis] Essential mathematical skills for engineering, science and applied mathematics - UNSW Press, 2002. 636683257882243472
[19] [Steven Ian Barry, Stephen Alan Davis] Essential Mathematical Skills For Engineering, Science And Applied Mathematics - UNSW Press, 2002. 636683257900380734
[20] [Steven Ian Barry, Stephen Alan Davis] Essential mathematical skills for engineering, science and applied mathematics - UNSW Press, 2002
[21] [Tang K.T.] Mathematical Methods for Engineers and Scientists 2 - Springer, 2007
[22] Calculus_Dr_Gilbert_Strang
[23] Calculus_Know-It-ALL - Beginne
[24] Edwin Herman, Gilbert Strang - Calculus Volume 1- Openstax
[25] Edwin Herman, Gilbert Strang - Calculus Volume 2, Openstax
[26] Elementary_Calculus - An Appro
[27] Gilbert Strang - Elementary Calculus
[28] Mathematics - Vector Calculus
[29] Techniques of doing Calculus
[30] [Stan Gibilisco] Calculus Know-It-ALL - Beginner to Advanced, and Everything in Between - McGraw-Hill, 2009
[31] [Stroyan K. D.] Mathematical Background Foundations of Infinitesimal Calculus - Academic Press, 1997
[32] Differential Equations - Crash Course
[33] Differential Equations - Linear, Nonlinear, Ordinary, Partial
[34] Differential and Integral Calc
[35] Introduction to Partial Differential Equations - A Computational Approach - A. Tveito, R. Winther
[36] Ordinary and Partial Differential Equations - Agarwal, Regan
[37] THEORY_AND_PROBLEMS_OF_DIFFERE
[38] [Weinberger] A First Course In Partial Differential Equations - Dover Publications, 1995
[39] A_First_Course_in_Discrete_Mat
[40] discrete_mathematics
[41] Discrete_Mathematics_-_Dr_J_Sa
[42] Discrete_Mathematics_-_Element
[43] Discrete_Mathematics_and_Its_A
[44] Discrete_Math_-_Mathematics_Fo
44 item(s)
# Request >>> list -DBooks()
```

Kết quả thực hiện lệnh `add shell`

Trong phần thực hành trên chúng ta lần đầu áp dụng các lớp .NET hỗ trợ làm việc với hệ thống file của windows.

Tất cả các lớp để làm việc với file trong .NET nằm trong không gian tên `System.IO`. Ba class chính để làm việc với hệ thống file là `Directory` (làm việc với thư mục), `File` (làm việc với file), `Path` (làm việc với đường dẫn).

## Lớp Directory

Lớp `Directory` chứa hầu hết các phương thức tĩnh giúp làm việc với file và thư mục. Trong phần thực hành chúng ta đã sử dụng một số phương thức của lớp này giúp kiểm tra đường dẫn và giúp lấy danh sách file trong một thư mục.

- Phương thức tĩnh `Exists` của lớp `Directory`: kiểm tra xem một đường dẫn tới thư mục có tồn tại hoặc chính xác không.

```
1. if (!Directory.Exists(folder))
2. {
3.     Error("Folder not found!");
4.     return;
5. }
```

- Phương thức tĩnh `GetFiles` của lớp `Directory`: tìm tất cả các file trong thư mục có phần mở rộng là `pdf`:

```
1. Directory.GetFiles(folder, ext ?? "*.pdf", SearchOption.AllDirectories);
```

Phương thức này sử dụng ba tham số:

1. đường dẫn tới thư mục;
2. mẫu tìm kiếm: mẫu văn bản mà phương thức `GetFiles` sử dụng trong quá trình tìm kiếm. `GetFiles` chỉ trả lại những file mà tên phù hợp với mẫu văn bản của tham số này.
3. phạm vi tìm kiếm: xác định xem phương thức `GetFiles` chỉ tìm trong thư mục được chỉ định ( `TopDirectoryOnly` ) hay tìm cả trong các thư mục con của nó ( `AllDirectories` ).

Kết quả thực hiện của phương thức này là một mảng string chứa tên đầy đủ (bao gồm cả đường dẫn) của các file tìm thấy.

## Lớp Path

Lớp Path cũng chứa hầu hết các phương thức tính giúp phân tích đường dẫn tới file hoặc thư mục. Ở phần thực hành chúng ta đã sử dụng một phương thức của lớp này để phân tách tên file khỏi đường dẫn và phần mở rộng.

Phương thức `GetFileNameWithoutExtension` của lớp `Path` trích ra phần tên của file, bỏ phần đường dẫn và phần mở rộng.

```
1. Repository.Insert(new Book { Title = Path.GetFileNameWithoutExtension(f), File = f });
```

Tên file này được sử dụng tạm thời làm tiêu đề của sách (vì thường sách điện tử đặt tên file trùng với tiêu đề sách).

Các phương thức của hai class `Directory` và `Path` đều tương đối dễ sử dụng. Bạn đọc có thể tự mình tìm hiểu các phương thức khác.

## Thực hành 4: thêm chức năng đọc sách từ chương trình

Chức năng này cho phép mở file pdf bằng chương trình đọc pdf mặc định của windows (Acrobat Reader, Foxit Reader, v.v.). Chức năng này tiện lợi cho người sử dụng vì không cần phải mở các thư mục để tìm đến file.

### Bước 1. Thêm phương thức vào ShellController

```
1. public void Read(int id)
2. {
3.     var book = Repository.Select(id);
4.     if (book == null)
5.     {
6.         Error("Book not found!");
7.         return;
8.     }
9.     if (!File.Exists(book.File))
10.    {
11.        Error("File not found!");
12.        return;
13.    }
14.    Process.Start(book.File);
15.    Success($"You are reading the book '{book.Title}'");
16. }
```

### Bước 2. Bổ sung route vào ConfigRouter

```
1. r.Register(route: "read",
```

```
2. action: p => shell.Read(p["id"].ToInt()),
3. help: "[read ? id = <value>]";
```

### Bước 3. Dịch và chạy thử chương trình

Kết  
quả  
thực  
hiện  
lệnh  
read

Sau lệnh này, cuốn "A first course in discrete mathematics" sẽ được mở ra bằng chương trình đọc pdf mặc định trên windows.

## Thực hành 5: đánh dấu những cuốn sách đang đọc

Chức năng này cho phép đánh dấu những cuốn sách đang đọc để có thể dễ dàng tìm đọc tiếp. Khi xây dựng lớp Book chúng ta có thuộc tính Reading kiểu bool dành cho chức năng này.

### Bước 1. Bổ sung phương thức vào lớp Repository

```
1. public Book[] SelectMarked()
2. {
3.     var list = new List<Book>();
4.     foreach (var b in Books)
5.     {
6.         if (b.Reading) list.Add(b);
7.     }
8.     return list.ToArray();
9. }
```

### Bước 2. Bổ sung hai phương thức vào lớp BookController

```
1. public void Mark(int id, bool read = true)
2. {
3.     var book = Repository.Select(id);
4.     if (book == null)
5.     {
6.         Error("Book not found!");
7.         return;
8.     }
9.     book.Reading = read;
10.    Success($"The book '{book.Title}' are marked as { (read ? "READ" : "UNREAD")}");
11. }
12.
13. public void ShowMarks()
14. {
15.     var model = Repository.SelectMarked();
16.     var view = new BookListView(model);
17.     Render(view);
18. }
```

### Bước 3. Bổ sung route vào ConfigRouter

```
1. r.Register(route: "mark",
2.     action: p => controller.Mark(p["id"].ToInt()),
3.     help: "[mark ? id = <value>]");
4. r.Register(route: "unmark",
5.     action: p => controller.Mark(p["id"].ToInt(), false),
```

```

6.     help: "[unmark ? id = <value>]";
7.     r.Register(route: "show marks",
8.         action: p => controller.ShowMarks(),
9.         help: "[show marks]");

```

## Bước 4. Dịch và chạy thử chương trình

Kết  
 quả  
 thực  
 hiện  
 lệnh  
 mark  
 và  
 show  
 marks

Chức năng này tận dụng lại lớp BookListView đã xây dựng từ trước. Lớp này có thể hiển thị các cuốn sách đang đọc bằng màu Cyan, còn các cuốn sách khác hiện màu trắng.

## Thực hành 6: bổ sung khả năng xóa toàn bộ dữ liệu

### Bước 1. Thêm phương thức vào lớp Repository

```

1.     public void Clear()
2.     {
3.         _context.Books.Clear();
4.     }

```

### Bước 2. Thêm phương thức Clear vào ShellController

```

1.     public void Clear(bool process = false)
2.     {
3.         if (!process)
4.         {
5.             Confirm("Do you really want to clear the shell? ", "do clear");
6.             return;
7.         }
8.         Repository.Clear();
9.         Inform("The shell has been cleared");
10.    }

```

### Bước 3. Bổ sung các route

```

1.     r.Register(route: "clear",
2.         action: p => shell.Clear(),
3.         help: "[clear]\r\nUse with care");
4.     r.Register(route: "do clear",
5.         action: p => shell.Clear(true),
6.         help: "[clear]\r\nUse with care");

```

### Bước 4. Dịch và chạy thử chương trình với lệnh clear

Kết  
 quả  
 thực  
 hiện



## Kết luận

---

Trong bài này chúng ta vận dụng các class xây dựng trong các bài trước để hoàn thiện hầu hết các chức năng đã phân tích. Qua bài này chúng ta có thể thấy được khả năng mở rộng của ứng dụng và sự rành mạch trong phân chia code. Khi cần bổ sung thêm chức năng mới, chúng ta xác định được ngay các code mới cần thêm vào đâu.

Trong các bài sau chúng ta sẽ hoàn thiện nốt chức năng cuối cùng: lưu trữ dữ liệu.

- + 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!