Tạo cấu trúc quản lý mã nguồn cho dự án

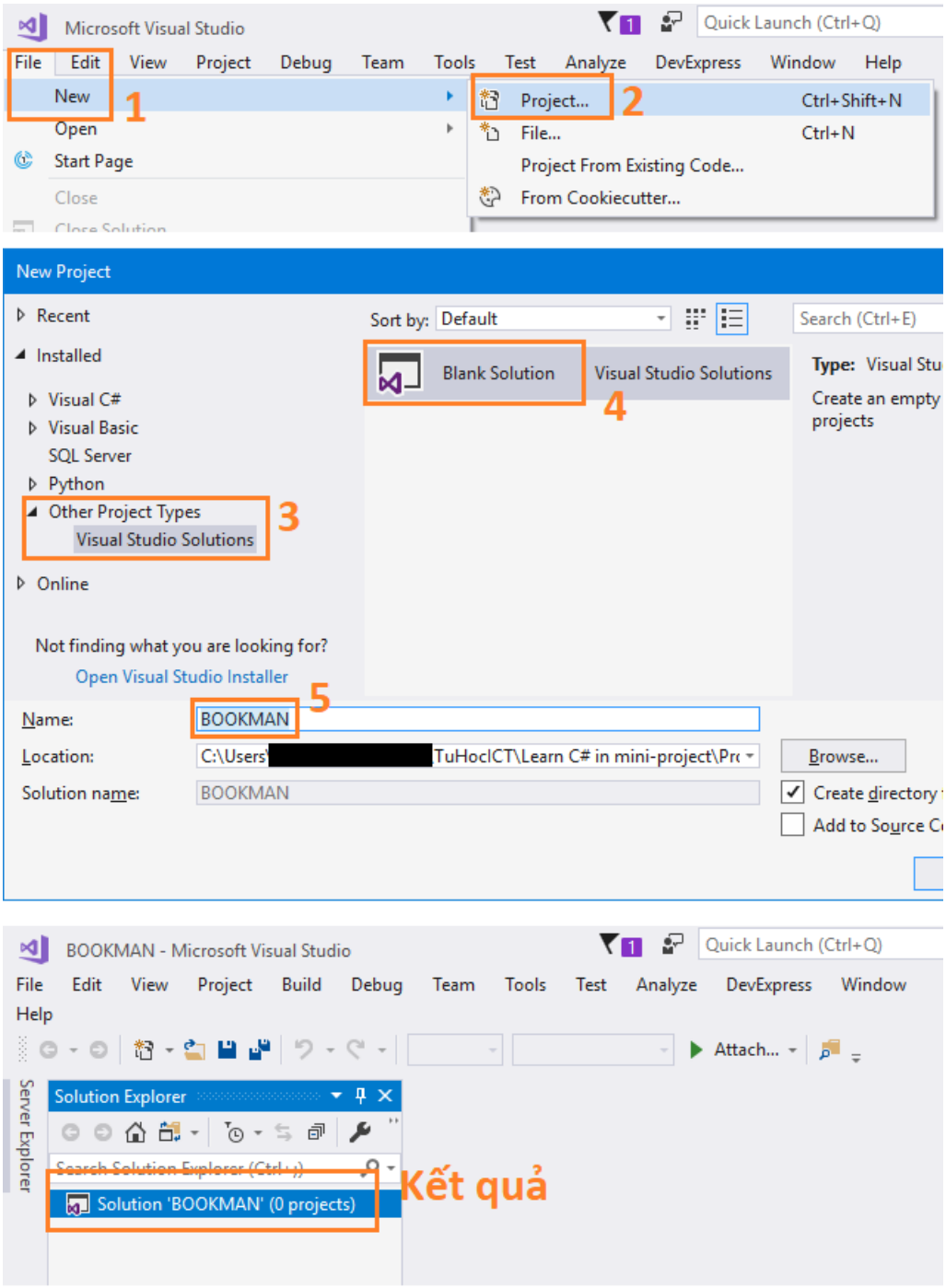
Bài thực hành này sẽ trình bày sơ lược về cách thức tổ chức quản lý mã nguồn theo mô hình MVC, sau đó sẽ tạo project chính thức sử dụng trong dự án với các file và thư mục theo yêu cầu của MVC.

NỘI DUNG CỦA BÀI [Ẩn]

Tạo project

Chuẩn bị: tạo solution BOOKMAN

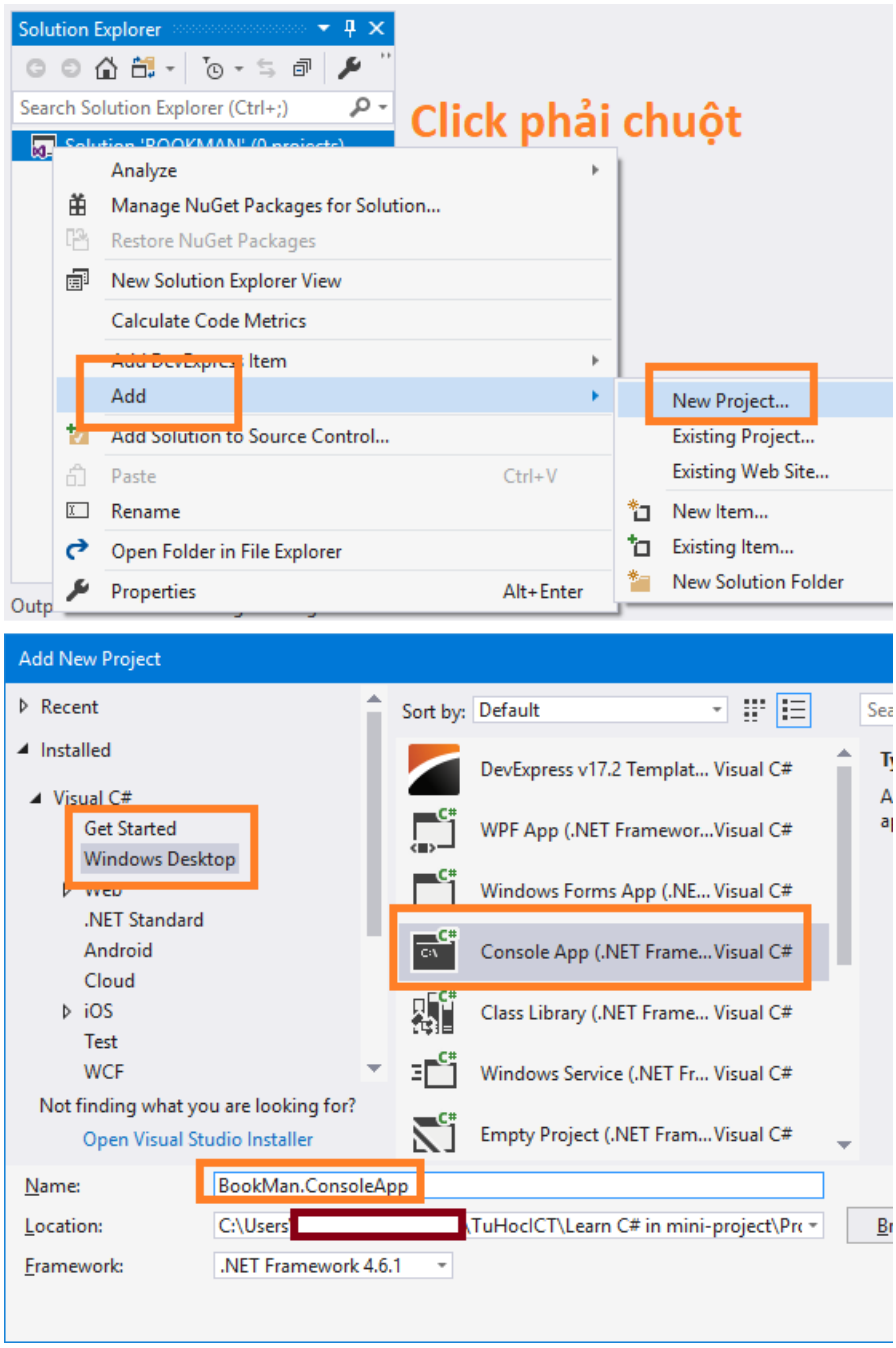
Tạo blank solution BOOKMAN như sau:



Thêm project “BookMan.ConsoleApp”

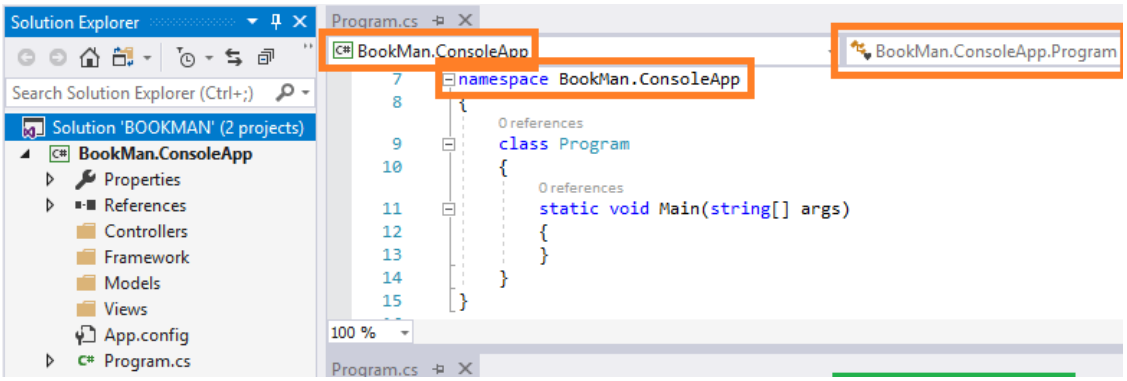
Tạo thêm một project thuộc loại ConsoleApp trong solution “BOOKMAN”:

1. Click phải vào tên Solution, chọn Add => New Project …
2. Trong hộp thoại “Add New Project” chọn loại project là “Console App (.NET Framework);
3. Trong hộp văn bản “Name” nhập “BookMan.ConsoleApp”;
4. Ấn nút “Add” hoặc phím “Enter” để hoàn thành.



> Nếu không nhớ các thao tác, bạn hãy đọc lại phần nội dung về tạo project mới trong bài Cài đặt Visual Studio.

Kết quả thu được như sau:



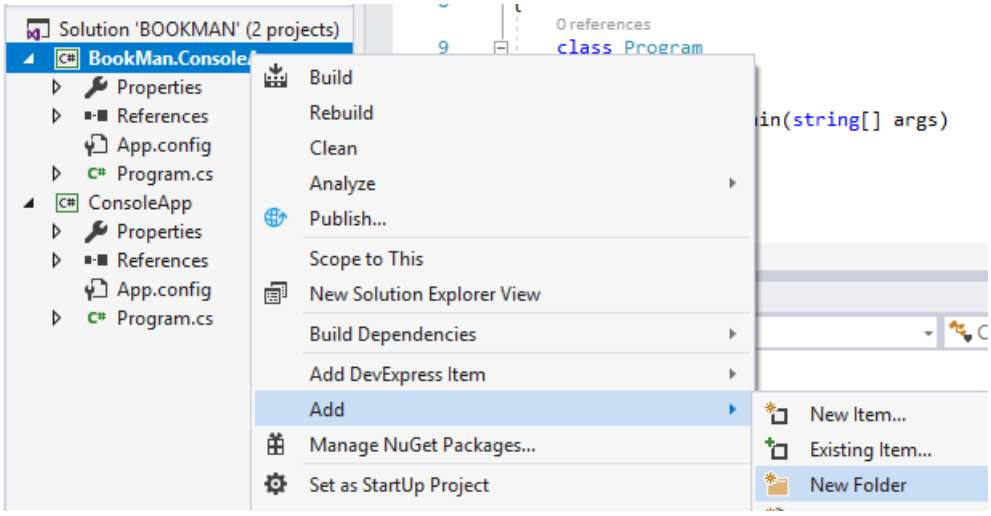
> Bạn hãy tự nhận xét sự liên quan giữa tên project, cấu trúc namespace và tên đầy đủ của lớp Program trong project.

Có thể để ý thấy rằng, tên của project "BookMan.ConsoleApp" được tô màu đậm, vì Visual Studio đang đặt nó làm *project khởi động*. Project khởi động (StartUp Project) là project được chạy mặc định (khi ấn F5 hoặc Ctrl+F5).

> Nếu trong solution có những project khác, bạn có thể thiết lập project khởi động bằng cách Click phải chuột vào project tương ứng và chọn "Set as StartUp Project". Tên project khởi động trong solution được Visual studio hiển thị bằng chữ đậm.

Tạo thư mục trong project

Click phải vào tên project => Add => New Folder

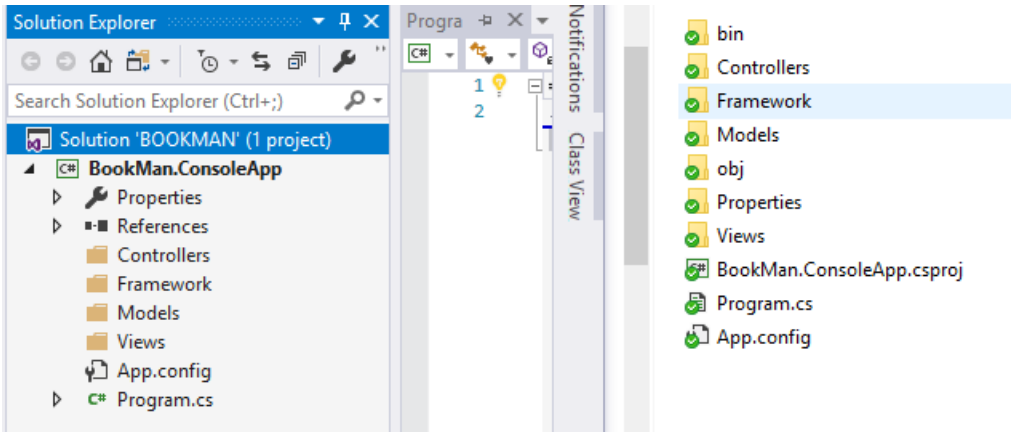


Trong project xuất hiện một thư mục mới "NewFolder". Đổi tên thư mục này thành Models.

Nếu thực hiện sai, có thể đổi tên thư mục, hoặc xóa bỏ thư mục bằng cách click phải chuột vào tên thư mục và chọn lệnh tương ứng (Rename, Delete). Bạn cũng có thể dùng phím tắt F2 để đổi tên thư mục.

Lặp lại bước trên để lần lượt tạo ra các thư mục **Views**, **Controllers**, **Framework**.

Bạn thu được cấu trúc project và cấu trúc thư mục tương ứng trên ổ cứng như sau:



Cấu trúc mã nguồn theo mô hình MVC

Người mới học lập trình hướng đối tượng thường có xu hướng trộn lẫn code xử lý dữ liệu với code xử lý giao diện.

Ví dụ, khi tạo lớp ma trận thường trộn luôn code nhập/xuất dữ liệu cùng với code xử lý ma trận (cộng, tích vô hướng, nhân ma trận, v.v.).

Hoặc, khi tạo lớp mô tả sinh viên thường viết luôn trong đó code xuất/nhập thông tin sinh viên, nhưng lại không biết nên lưu trữ danh sách sinh viên, viết các phương thức thêm mới/sửa/xóa/lọc ở đâu.

Vấn đề tổ chức mã nguồn của project

Phân chia code xử lý dữ liệu và code xử lý giao diện là một trong những yêu cầu đầu tiên và đặc biệt quan trọng, giúp việc quản lý code và kiểm thử ứng dụng dễ dàng hơn.

Các *ứng dụng quản lý* (Line-of-Business, LOB) đều có xu hướng phân chia code thành các phần: giao diện (gọi là các view), các lớp mô tả các đối tượng được quản lý (các model), các lớp hỗ trợ xử lý/truy xuất/lưu trữ dữ liệu (thường gọi là khối data service).

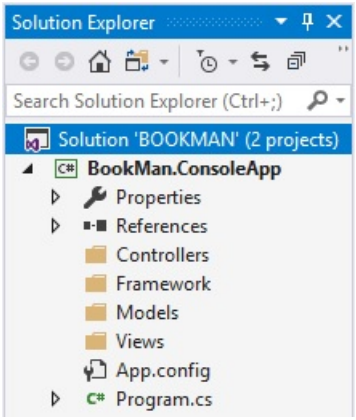
Để ghép nối giao diện với dữ liệu, người ta đưa ra nhiều loại lớp trung gian khác nhau. Từ sự khác nhau về đặc điểm và chức năng của các lớp trung gian xuất hiện một số *mô hình kiến trúc* (architectural pattern) thường được sử dụng trong phát triển các ứng dụng quản lý: mô hình MVC (Model – View – Controller), mô hình MVP (Model – View – Presenter), mô hình MVVM (Model – View – View Model), v.v..

Chúng ta có thể nhận thấy ngay là trong các mô hình này đều tồn tại hai thành phần chung: View và Model. Việc lựa chọn mô hình nào sẽ do nhiều yếu tố khác nhau quyết định. Ở đây chúng ta sẽ không đi vào phân tích các mô hình này vì đây là nội dung của một môn học khác.

Mô hình MVC

Trong dự án này, chúng ta sẽ vận dụng mô hình MVC, với mục đích chính là giúp phân chia code một cách rõ ràng, cụ thể như sau:

1. Tất cả code dùng để xuất nhập dữ liệu trên giao diện dòng lệnh (gọi chung là các lớp giao diện hoặc các lớp view) sẽ đặt trong thư mục Views;
2. Code dùng để mô tả thực thể (gọi chung là các lớp dữ liệu hoặc các lớp model) được quản lý (như sách và giá sách) được đặt trong thư mục Models;
3. Trong thư mục Controllers sẽ chứa code giúp ghép nối dữ liệu và giao diện (gọi chung là các lớp điều khiển hay lớp controller).



Cấu trúc mã nguồn theo mô hình MVC

> Ngoài các vấn đề nêu trên, khi phân chia code theo mô hình MVC cũng đặt thêm các yêu cầu về sự phụ thuộc giữa các class, cụ thể như sau:
> (1) Các lớp giao diện chỉ được biết đến sự tồn tại của các lớp dữ liệu, nói cách khác, các lớp giao diện chỉ phụ thuộc vào các lớp giao diện;
> (2) Các lớp giao diện không được biết đến sự tồn tại của các lớp điều khiển;
> (3) Các lớp điều khiển có thể phụ thuộc vào các lớp dữ liệu và các lớp giao diện.
>
> Giới hạn này cũng đưa ra thứ tự xây dựng các lớp trong mô hình MVC: dữ liệu => giao diện => điều khiển.
>
> Trong các bài thực hành, bạn sẽ làm quen với các vấn đề của thể của mô hình này. Nó sẽ rất có ích để sau bạn chuyển sang học công nghệ ASP.NET MVC và Web API.

Kết luận

Trong bài này chúng ta đã xem xét một số vấn đề về cách tổ chức mã nguồn của dự án. Chúng ta cũng xác định cấu trúc thư mục của dự án này theo cách phân chia code của mô hình MVC.

Nội dung thực hành của bài tương đối đơn giản, tuy nhiên vấn đề phân chia code của dự án có thể hơi xa lạ với bạn. Tuy nhiên, đây lại là vấn đề gần như bắt buộc phải biết với lập trình viên.

+ 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!