MusicBookMovieStore – mandatory assignment 1

This is the first of two mandatory assignments in backend programming that focus on the development of a website for *MusicBookMovieStore*.

The web application must present *MusicBookMovieStore* for customers and support online shopping. The reservation system must keep track of customers and their lodged animals. Upon collection of pets, it must be possible to print an invoice. Moreover, it should be possible on daily basis to generate a list with name, species, and age on lodged animals.

You are supposed to build an early prototype in this assignment. In the assignment, you must demonstrate that you are able to write and instantiate classes and display data in a basic way inside an ASP.NET MVC web application. At this point data are stored as *non-persistent sample data* inside the program itself. In addition, there will be no administration part with HTML forms for data input and maintenance.

In assignment two, we will design a great looking website for *MusicBookMovieStore* with an online basket.

You can do the assignment individually or in small groups of two or three persons. You must upload the assignment to Fronter (in the Hand-in folder) as a .zip file containing all project files and a UML-diagram. The default page of the website – which loads when you run the project – must have links to the webpages that are part of the solution.

The assignment must be approved in order to be recommended for examination in the *Backend Programming* module.

Deadline

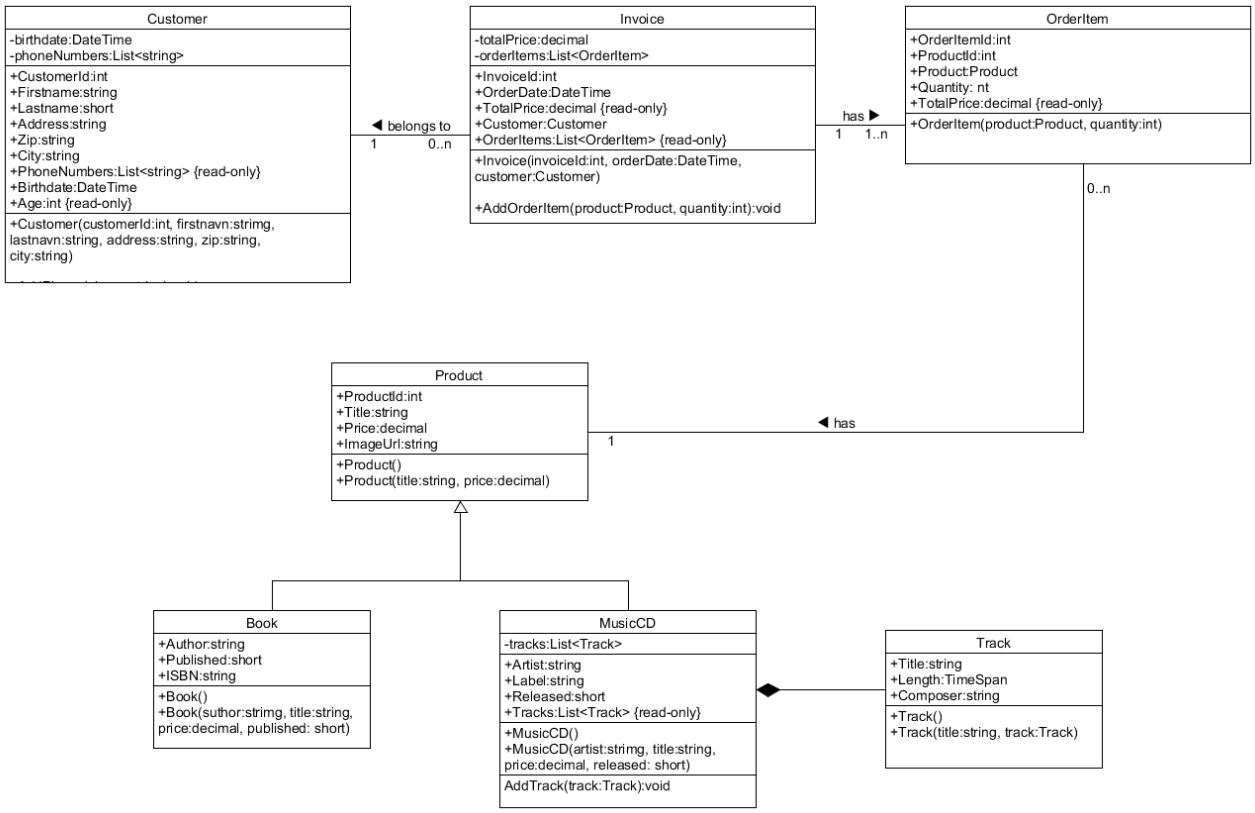
Sunday 2/10 at 24:00.

Precondition

Before you start, be sure that you have done *exercise 1-4* in lesson 2, and *exercise 1-4* in Lesson 3.

Exercise 1

Add the `Invoice` and `OrderItem` classes to the Models folder and program them as specified by the UML diagram below:



Review all you classes to make sure that they reflect the UML diagram above.

Exercise 2

In all previous exercises during this course, we have instantiated our classes inside controllers. Normally you will do it elsewhere. A common way of doing it is to store the data inside of a database, and writing a *Data Access Layer* (DAL) component, that handles access to the database.

The Repository Class

For this exercise however, we will create a new `Infrastructure` folder with a `Repository` class that we will use to instantiate classes with example data that we will use inside the application.

Open the `Repository` class and declare a `Products` property and instantiate it as a list of Products and likewise an `Invoices` property and instantiated it as a list of Invoices and:

```
public List<Product> Products = new List<Product>();
public List<Invoice> Invoices = new List<Invoice>();
```

You should also create an empty constructor with no parameters.

We now want to store our product objects as separate objects in the Products list. To do that, you must move all objects created in the Movie and Catalogue controllers into the `Repository` constructor and add each product to the list of products, like for example this book object:

```
// Book no 2
Book myBook = new Book("Georg Martin", "With a Little Help from My Friends: The
```

```
                                Making of Sgt. Pepper", 1800M, 1995);
        myBook2.Publisher = "Little Brown & Co";
        myBook.ISBN = "0316547832";
        myBook.ImageUrl = "The Making of Sgt. Pepper.jpg";
        Products.Add(myBook);
```

The Catalogue Controller

After you have created all product object inside the controller of the repository class, you can fetch this list into the Catalogue controller by creating an instance of the `Repository` class in its `Index` action method. To do that you must add this line of code:

```
// create a Reposptory object
private Repository repository = new Repository();
```

By saving the product list to a ViewBack property, you have access to the full products list in inside the view:

```
ViewBag.Products = repository.Products;
```

The View

The next and final step is to display the full products list from inside the view grouped by categories.

You can now loop through the list and display products for each categories by selecting procucts by object type:

```
<h2>The Movies</h2>
@foreach (Product product in ViewBag.Products)
{
    if (product is Movie)
    {
        Movie movie = (Movie)product;

        <div>
            <strong>Title:</strong> @movie.Title<br />
            Director: @movie.Director<br>
            Price: @String.Format("{0:0.00}", movie.Price) <br /><br />
            <img src="/Content/Images/@movie.ImageUrl" alt="@movie.Title" />
        </div>

    }
}
```

Tip

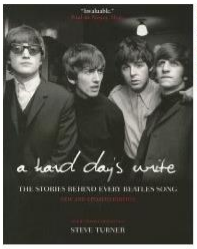
Alternatively, you also can send sepetate Book, MusicCD and Movie lists to the view by using LINQ (Language-Integrated Query) inside the controller to save each product type in its own category list, as in this example of the book list:

```
Repository repository = new Repository();
IList<Book> books = new List<Book>();
books = repository.Products.OfType<Book>().ToList();
```

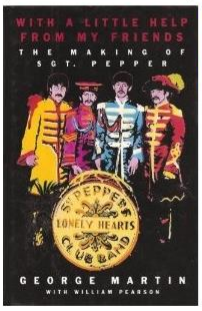
```
ViewBag.Books = books;
```

The view must generate an output similar to this display of products:

The Books

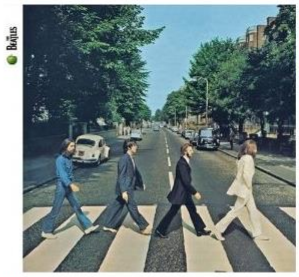


Title: A Hard Day's Write: The Stories Behind Every Beatles Song
Author: Steve Turner
Price: 150,00
Publisher: It Books (2005)
ISBN: 978-0060844097



Title: With a Little Help from My Friends: The Making of Sgt. Pepper
Author: Georg Martin
Price: 1800
Publisher: Little Brown & Co (1995)
ISBN: 0316547832

The Music CDs



Album: Abbey Road (Remastered)
Artist: Beatles
Price: 128,00
Publisher: EMI (2009)

Tracks:

1. Come Together (Lennon, McCartney) 4:20
2. Something (Harrison) 3:3
3. Maxwell's Silver Hammer (Lennon, McCartney) 3:29
4. Oh! Darling (Lennon, McCartney) 3:26
5. Octopus's Garden (Starkey) 2:51
6. I Want You (She's So Heavy) (Lennon, McCartney) 7:47
7. Here Comes The Sun (Harrison) 3:5
8. Because (Lennon, McCartney) 2:45
9. You Never Give Me Your Money (Lennon, McCartney) 4:2
10. Sun King (Lennon, McCartney) 2:26
11. Mean Mr. Mustard (Lennon, McCartney) 1:6
12. Polythene Pam (Lennon, McCartney) 1:12
13. She Came In Through The Bathroom Window (Lennon, McCartney) 1:57
14. Golden Slumbers (Lennon, McCartney) 1:31
15. Carry That Weight (Lennon, McCartney) 1:36
16. The End (Lennon, McCartney) 2:19
17. Her Majesty (Lennon, McCartney) 0:23

Total playing time: 47:18

The Movies

Title: Jungle Book
Director: Jon Favreau
Price: 160,50

Title: Gladiator
Director: Ridley Scott
Price: 49,95

Exercise 3

Create a couple of `Invoice` objects with `OrderItem` and `Customer` object references. Add these objects to the `Invoices` list.

To to that follow these steps:

- Create (at least) two new `Customer` objects
- Create (at least) two new `Invoice` objects
- Create (at least) four new `OrderItem` objects
- Add two `OrderItem` objects to the first `Invoice` object
- Add two `OrderItem` objects to the second `Invoice` object
- Add each `Invoice` object to the `Invoices` list.

Exercise 4

Create an `Invoice` controller class, instantiale the `Repository` class and return the list of `Invoices` to as a ViewBag property to a view.

Use razor code inside the view to generate a display similar to this (or at least with the same information):

Invoices

Customer	Product	Price
Tina Petterson	Forrest Gump	154,50
	With a Little Help from My Friends: The Making of Sgt. Pepper	180,00
Thomas Larsson	A Hard Day's Write: The Stories Behind Every Beatles Song	150,00
	Revolver (Remastered)	128,00

Exercise 5

Enhance the display with an output similar to this:

Invoices

Customer	Product	Price
Tina Petterson	Forrest Gump (Movie)	154,50
	With a Little Help from My Friends: The Making of Sgt. Pepper (Book)	180,00
	Total	**334,50**
Thomas Larsson	A Hard Day's Write: The Stories Behind Every Beatles Song (Book)	150,00
	Revolver (Remastered) (MusicCD)	128,00
	Total	**278,00**

Tip 1: In the Razor view you can declare variables by placing them inside a code block, like this:

```
@{decimal price = 0M;}
```

Tip 2: If you call the `ToString` method on an object, you'll get the namespace and the class name. To substract the class name from the string you can use the method `.Substring`. It has a overload with one parameter of type int that skips the number of characters you give from the beginning of the string and returns the remaining characters of the string.

You can use that for extracting the classname which is the same as the category. In an upcomming lesson we'll create a new `Category` class for holding product categories.

Read more: http://www.dotnetperls.com/substring

/Jes, 2016-08-09.