

ASP.NET Core Practice- Library Due Date Tracker Day 1

Deadline: Tuesday, September 22nd 2020, 09:00 AM

[GitHub Classroom Link](#)

Introduction

This assignment is meant to challenge your mastery of ASP.NET Web Application (Model - View - Controller) and how well you are able to use MVC to create a CRUD application. Your goal in this assignment is to create a tool that will help you keep track of all the books you have checked out of the library. This is a cumulative activity. You will be adding onto this code for Day 2 and Day 3.

Requirements (See Rubric for Details)

- ☐ Book class (Model):
 - ☐ string "Title"
 - ☐ This property should be readOnly (getter only, backing variable initialized via constructor)
 - ☐ DateTime "PublicationDate"
 - ☐ This property should be readOnly (getter only, backing variable initialized via constructor)
 - ☐ DateTime "CheckedOutDate"
 - ☐ DateTime "DueDate"
 - ☐ This will equal "CheckedOutDate" + 14 days (set in constructor)
 - ☐ This will update with each extension (via the ExtendDueDateForBookByID() method)
 - ☐ DateTime "ReturnedDate"
 - ☐ Default value should be null (set in constructor)
 - ☐ string "Author"
 - ☐ This property should be readOnly (getter only, backing variable initialized via constructor)
 - ☐ int "ID"
 - ☐ This property should be readOnly (getter only, backing variable initialized via constructor)
 - ☐ This property will be auto-incremented by the database in tomorrow's practice
 - ☐ User will have to add "ID" on Day 1 and the code will have to validate that the "ID" is unique (in the CreateBook() method)
 - ☐ Constructor accepting the ID, Title, Author, PublicationDate and CheckedOutDate as parameters
 - ☐ "DueDate" will be set to 14 days after "CheckedOutDate"
 - ☐ "ReturnedDate" will be set to null
- ☐ BookController : Controller class:
 - ☐ Action/View "Create"
 - ☐ Will display the form to create an object.
 - ☐ If the appropriate parameters are supplied in the query, attempt creation of the "Book" and add it to the list.
 - ☐ If the ID is already in the list, throw an exception.

- ☐ Success message: "You have successfully checked out {title} until {DueDate}."
- ☐ Error Message: "Unable to check out book: {Exception.Message}."
- ☐ Action/View "List"
 - ☐ Render a list of all books as links that will load the "Details" Action/View.
- ☐ Action/View "Details"
 - ☐ If no get parameter "id" was supplied, render "No book selected."
 - ☐ If an "id" get parameter was supplied, use GetBookByID() and render:
 - ☐ "You checked out {title} on {CheckedOutDate}, and it {is/was} due on {DueDate}."
 - ☐ Use conditional rendering to make a choice about using 'is' or 'was' based on today's date.
 - ☐ A button that will call ExtendDueDateForBookByID().
 - ☐ A button that will call DeleteBookByID().
- ☐ Method "CreateBook()".
 - ☐ Accepts the same parameters as the "Book" constructor.
 - ☐ Creates and adds a "Book" to the "Books" list.
 - ☐ Ensures the provided ID is unique in the list.
 - ☐ Throw an exception if the ID already exists.
- ☐ Method "GetBookByID()".
 - ☐ Returns the book with the given ID from the "Books" list.
- ☐ Method "ExtendDueDateForBookByID()".
 - ☐ Extensions are 7 days from the current date (7 days from when the user requests the extension, not 7 days past the "DueDate").
- ☐ Method "DeleteBookByID()".
 - ☐ Removes the book with the given ID from the "Books" list.
- ☐ A public static "Books" property which is a list of "Book" objects.
 - ☐ This will be replaced by a proper database on {Day 2 assignment title}.

Challenges (See Rubric for Details)

- ☐ Make it look nice with CSS
- ☐ Have an unexpected feature

Hints

- Under new project: ASP.NET Core Web Application > Web Application (Model-View-Controller).
- Ensure your project isn't running when modifying code.
- General Hints:
 - Focus on the requirements first, challenges are extra!
 - This kind of project has been done by many others in the past! Don't hesitate to use your google-fu skills if you don't know how to implement certain features!
 - Please include source citations in your code and README.md
- Day 1 Hints:
 - If you are struggling with the Book class, look back at other class examples done during C# (Such as the Car and Pen classes during OOP)
 - Look up how the DateTime class works for C#, this will help you easily keep track of dates
 - The Book class has properties defined, the BookController : Controller class is where all your data manipulation methods will be contained

Rubric

Requirement	Points
<ul style="list-style-type: none"> Book class: <ul style="list-style-type: none"> string "Title" <ul style="list-style-type: none"> This property should be readOnly DateTime "PublicationDate" <ul style="list-style-type: none"> This property should be readOnly DateTime "CheckedOutDate" DateTime "DueDate" <ul style="list-style-type: none"> This will equal "CheckedOutDate" + 14 days This will update with each extension DateTime "ReturnedDate" <ul style="list-style-type: none"> Default value should be null string "Author" <ul style="list-style-type: none"> This property should be readOnly int "ID" <ul style="list-style-type: none"> This property should be readOnly This property will be auto-incremented by the database on Day 2 User will have to add ID on Day 1 and the code will have to validate that the ID is unique 	17
<ul style="list-style-type: none"> Author class: <ul style="list-style-type: none"> int "ID" <ul style="list-style-type: none"> This property should be readOnly string "Name" <ul style="list-style-type: none"> This property should be readOnly 	5
<ul style="list-style-type: none"> BookController : Controller class: <ul style="list-style-type: none"> ActionResult "CreateBook" <ul style="list-style-type: none"> Success message: "you have successfully checked out {title} until {DueDate}." ActionResult "ExtendDueDateForBookByID" <ul style="list-style-type: none"> Extensions are 7 days from the current date (7 days from when the user requests the extension, not 7 days past the "DueDate") ActionResult "DeleteBookByID" ActionResult "ViewBooks" <ul style="list-style-type: none"> Display: "You checked out {title} on {CheckedOutDate}, and it {is/was} due on {DueDate}." Use conditional rendering to make a choice about using 'is' or 'was' based on today's date. ActionResult "ViewBookByID" The BookController class must have a "Books" property which is just a list of "Book" objects <ul style="list-style-type: none"> This will be replaced by a proper database on {Day 2 assignment title}. 	10
<p>CHALLENGE:</p> <ul style="list-style-type: none"> Make it look nice with CSS Have an unexpected feature 	2
Total:	17

Citation Guide for Borrowed Code

Whenever you borrow code, the following information must be included:

- Comments to indicate both where the borrowed code begins and ends.
- A source linking to where you found the code (URL, book, example, etc.).
- Your reason for adding the code to your assignment or project instead of writing it out yourself.
- Explain to us how the code is supposed to work, include links to documentation and articles you read to help you understand.
- A small demonstration to prove you understand how the code works.

```
1  const inputArr = [5,1,3,4,2];
2
3  /*Borrowed code for bubbleSort starts*/
4  let bubbleSort = (inputArr) => {
5      let len = inputArr.length;
6      for (let i = 0; i < len; i++) {
7          for (let j = 0; j < len; j++) {
8              if (inputArr[j] > inputArr[j + 1]) {
9                  let tmp = inputArr[j];
10                 inputArr[j] = inputArr[j + 1];
11                 inputArr[j + 1] = tmp;
12             }
13         }
14     }
15     return inputArr;
16 };
17
18 /*Borrowed code from bubbleSort ends*/
19
20 //Source: bubbleSort function obtained from https://medium.com/javascript-algorithms/javascript-algorithms-bubble-sort-3d27f285c3b2
21 //Reason to add: implementing bubble sort can be tedious and bug prone, it would be better to use a proven version than to write my own
22 //How it works: I read the following article to understand how bubble sorts work (http://www.pkirs.utep.edu/CIS3355/Tutorials/chapter9/tutorial19A/bubblesort.htm)
23 //Demonstration of understanding:
24 //Example array: [3,1,2]
25 //Step 1: Compare 3 and 1. Since 1 is smaller, swap places.
26 //Array: [1,3,2]
27 //Step 2: Compare 3 and 2. Since 2 is smaller, swap places.
28 //Array: [1,2,3]
29 //Step 3: Compare 1 and 2. No need to swap.
30 //Array: [1,2,3]
31 //Step 4: Compare 2 and 3. No need to swap.
32 //Array: [1,2,3]
33 //Function complete.
34 console.log(bubbleSort(inputArr));
```