



ASP.NET Core Assignment - Library Due Date Tracker Day 3

Deadline: Thursday, September 24 2020, 09:00AM

[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. Use your code from **ASP.NET Core Assignment - Library Due Date Tracker Day 2** as a starting point.

Requirements

- ☐ Modify "Book" (Model):
 - ☐ Add a property "ExtensionCount" - int(10), not nullable.
 - ☐ Update your seed data for this table to include values for this field.
 - ☐ Add a migration.
 - ☐ Update the database.
- ☐ Modify "BookController" (Controller):
 - ☐ Modify "CreateBook()".
 - ☐ Check that "Title" is unique before saving books to the database.
 - ☐ If "Title" is not unique do not add the new book.
 - ☐ Ensure this comparison is case insensitive and trimmed.
 - ☐ Set "CheckedOutDate" to today's date.
 - ☐ Set "ExtensionCount" to 0.
 - ☐ Keep the logic to set DueDate and ReturnedDate.
 - ☐ "PublishedDate" cannot be in the future.
 - ☐ "Title" cannot exceed its size in the database.
 - ☐ "Title" cannot be empty or whitespace.
 - ☐ Trim "Title" before saving it's value.
 - ☐ Display itemized errors for every field that has an issue.
 - ☐ Add a "GetOverdueBooks()" method.
 - ☐ Return a list of books with "DueDate" in the past, that have no "ReturnedDate".
 - ☐ Add a "ReturnBookByID()" method.
 - ☐ Set the returned date of the specified book to today's date.
 - ☐ Overdue books cannot be returned.
 - ☐ Display an error on the page calling the method informing the user they will have to speak to the librarian.

- ❑ Modify “ExtendDueDateForBookByID”.
 - ❑ A book can only be extended a maximum of 3 times.
 - ❑ If a user tries to extend a book a fourth time do not update the database
 - ❑ Display an error on the page calling the method informing the user they will have to speak to the librarian.
 - ❑ Overdue books cannot be extended.
 - ❑ Display an error on the page calling the method informing the user they will have to speak to the librarian.
- ❑ Modify “List” (View / Action):
 - ❑ Create a form with a checkbox “Filter to Overdue”.
 - ❑ When the page loads with the checkbox checked (GET parameter), call the “GetOverdueBooks()” method instead of the “GetBooks()” method.
- ❑ Modify “Create” (View / Action):
 - ❑ Remove the “Checked Out Date” form input.
- ❑ Modify “Details” (View / Action):
 - ❑ Add a “Return Book” button.
 - ❑ The button will call the “ReturnBookByID()” method in the action.
 - ❑ Add a “Number of Extensions” line / output.

Challenges (See Rubric for Details)

- ❑ Make it look nice with CSS
- ❑ Have an unexpected feature
- ❑ Modify “List” (View) to show the user how many days a book is overdue, and make the text dark red.
- ❑ Write a new Action and View called “Report”.
 - ❑ Display which author’s books have the longest cumulative checked-out time.
 - ❑ This should work on books that haven’t been returned, as well as on books that have been returned
 - ❑ Display which author is most likely to be overdue.
 - ❑ Display any other fun facts you can think of.

Hints

- 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
- Day 2 Hints:
 - Remember you must use a database to store all information, the information should not change if a session is switched or the page is refreshed
 - Your Book ID should no longer be user defined, but be generated by the database, search up how to use the “auto increment” attribute if you are struggling
- Day 3 Hints:

- All changes outlined in day 3 requirements should be done server side before passing the information off to the database; existing book records should be edited by ID
- Draw up existing records by ID in order to match information against one another (example: recall CheckoutDate from the database to match it against ReturnDate)

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/tutorial9A/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));

```