Adding Search Features To Our.NET MVC Web Application

Adding Search To Our Application (Using LINQ)

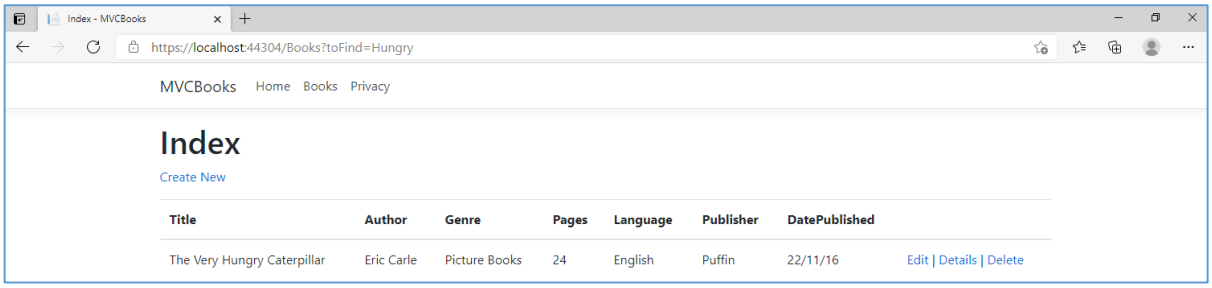
Here we are going to use LINQ to add a search feature to the Index action method.

```
// GET: Books

// Adding a search string to this method
public async Task<IActionResult> Index(string toFind)
{
    // Obtain the books from the database (using LINQ)
    var books = from b in _context.Books select b;

    // If the search string (passed to the method) is not empty
    // Return the books whose title contains the search string
    if (!String.IsNullOrEmpty(toFind))
    {
        books = books.Where(b => b.Title.Contains(toFind));
    }

    // Call the view with the relevant books (all or ones matching search string)
    return View(await books.ToListAsync());
}
```

The first line creates a LINQ query to get the books from the database.
If the search string contains a string then the query is modified to filter on the title field using the search string entered.

LINQ queries are not executed when they are created or modified. They are only actually valuated when the data is iterated over, or the ToListAsync method is called.

Run the application and enter some examples of search strings in the URL.
e.g.
https://localhost:44304/Books?toFind=Hungry
https://localhost:44304/Books?toFind=ll



So this works but we want to give the users an easy way to enter a search string (not in the URL like we just did), so we can add a form to do this. Let's add a form in the Index view.

```
<form asp-controller="Books" asp-action="Index">
    <p>
        Title: <input type="text" name="toFind" /><input type="submit" value="Search" />
    </p>
</form>
```

This allows users to enter what they want to search for into the form field as shown.

You may notice that you cannot see the search string in the URL.
If we want to see this (or want to be able to bookmark this, or send the link) then we need to use the GET method in the form.

Adding More Search Features To Our Application (Using LINQ)

Let's add a filter for Genre to our search features.
In the Models folder, add a new class (I called it GenreViewModel).
Add the following properties.

```
public class GenreViewModel
{
    // List of books
    public List<Book> Books { get; set; }

    // SelectList (to allow user to select from the list)
    public SelectList Genres {get; set;}

    // Property to contain the selected genre
    public string BookGenre { get; set; }

    // Property to contain the search string
    public string SearchString { get; set; }

}
```

We can then edit the Index method in the BooksController.cs as shown:

```
// Adding a search string and genre filter to this method
public async Task<IActionResult> Index(string SearchString, string BookGenre)
{
    // Use LINQ to get list of genres
    IQueryable<string> genreQuery = from b in _context.Books orderby b.Genre select b.Genre;

    // Obtain the books from the database (using LINQ)
    var books = from b in _context.Books select b;

    // If the search string (passed to the method) is not empty
    // Return the books whose title contains the search string
    if (!string.IsNullOrEmpty(SearchString))
    {
        books = books.Where(f => f.Title.Contains(SearchString));
    }

    // Now check for the genre selected
    if (!string.IsNullOrEmpty(BookGenre))
    {
        books = books.Where(k => k.Genre == BookGenre);
    }

    //
    var bookGenreVM = new GenreViewModel
    {
        Genres = new SelectList(await genreQuery.Distinct().ToListAsync()),
        Books = await books.ToListAsync()
    };

    // Call the view with the view model
    return View(bookGenreVM);
}
```

The SelectList of genres is create by getting the list of genres and removing duplicates.

In the Index.cs view, we need to make several changes.

We are going to use the new model we created, so we need to change the @Model directive.

```
@model MVCBooks.Models.GenreViewModel
```

We also need to add a dropdown (select) at the top of the page with the list of genres.
We are using tag helpers for the dropdown and the textbox.

```
<select asp-for="BookGenre" asp-items="Model.Genres">
    <option value="">All</option>
</select>

Title: <input type="text" asp-for="SearchString" />

<input type="submit" value="Search" />
```

We then change the table header row, and we are using HTML helpers.
We use the lambda expression so that we do not get any errors if the model is empty.

```
<tr>
    <th>
        @Html.DisplayNameFor(model => model.Books[0].Title)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Books[0].Author)
    </th>
    <th>
        @Html.DisplayNameFor(model => model.Books[0].Genre)
    </th>
```

We also have to change the Model that is used in the for each

```
@foreach (var item in Model.Books)
{

        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Genre)
            </td>
```

If we run the application,
We can now search for title and filter the list by genre, as seen in the screenshots below.

Picture Books | Title: | Search

Title	Author	Genre	Pages	Language	Publisher	DatePublished	
The Very Hungry Caterpillar	Eric Carle	Picture Books	24	English	Puffin	22/11/16	Edit | Details | Delete
Tiddler	Julia Donaldson	Picture Books	32	English	Alison Green	7/7/16	Edit | Details | Delete
Little Blue And Little Yellow	Leo Leonni	Picture Books	48	English	Dragonfly Books	17/1/17	Edit | Details | Delete

Picture Books | Title: Little | Search

Title	Author	Genre	Pages	Language	Publisher	DatePublished	
Little Blue And Little Yellow	Leo Leonni	Picture Books	48	English	Dragonfly Books	17/1/17	Edit | Details | Delete

All | Title: The | Search

Title	Author	Genre	Pages	Language	Publisher	DatePublished	
The Very Hungry Caterpillar	Eric Carle	Picture Books	24	English	Puffin	22/11/16	Edit | Details | Delete
The Pillars Of The Earth	Ken Follett	Historical Fiction	1105	English	Pan	4/9/08	Edit | Details | Delete
Astrosaurs: The Riddle Of The Raptors	Steve Cole	Science Fiction For Children	144	English	Red Fox	28/1/10	Edit | Details | Delete

Useful links:

Understanding tag helpers
https://docs.microsoft.com/en-us/aspnet/core/mvc/views/working-with-forms?view=aspnetcore-5.0