LAB 2
ASP.NET CORE MVC (P2)

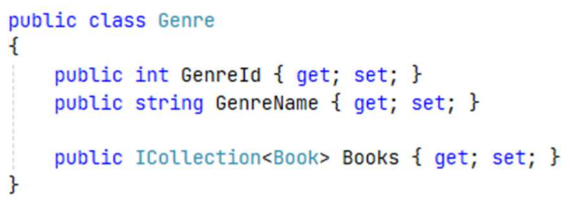
❖ CONTENT

- Setup table relationship
- Add data (model) validation
- Upload image
- Filter data
- Search & Sort

❖ INSTRUCTION

1. Add new model *Genre* and setup relationship with model *Book*

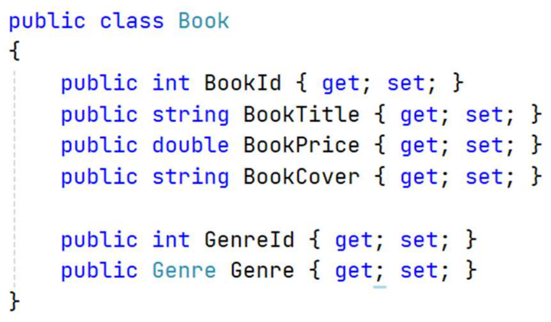
   Relationship: One Genre – Many Books



```
public class Genre
{
    public int GenreId { get; set; }
    public string GenreName { get; set; }

    public ICollection<Book> Books { get; set; }
}
```

*Figure 1 - **Models/Genre.cs***

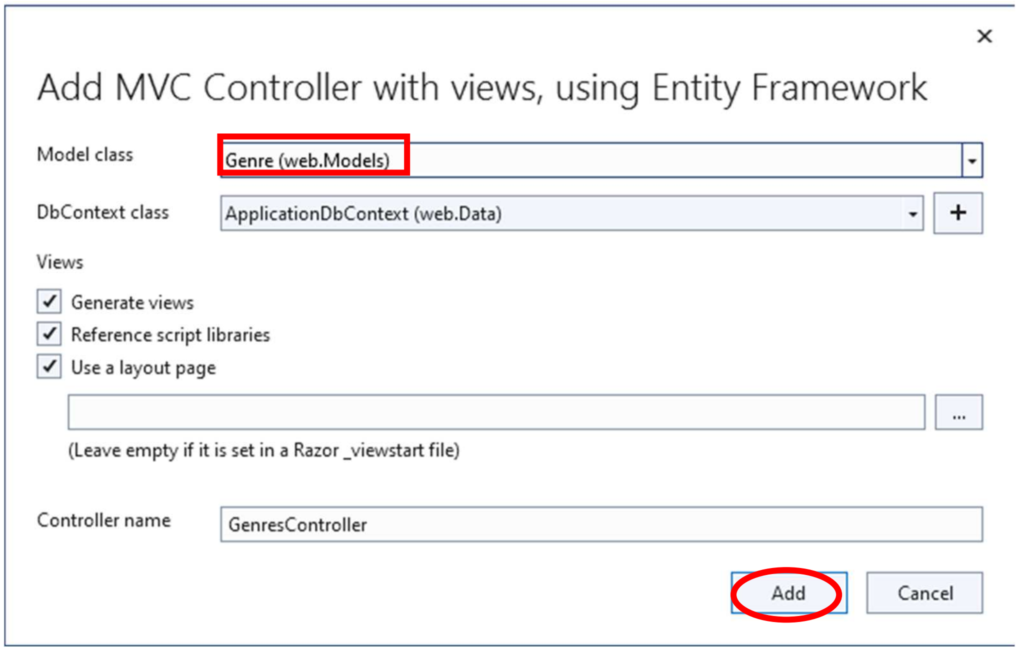
2. Update model *Book* to setup relationship with model *Genre*



```
public class Book
{
    public int BookId { get; set; }
    public string BookTitle { get; set; }
    public double BookPrice { get; set; }
    public string BookCover { get; set; }

    public int GenreId { get; set; }
    public Genre Genre { get; set; }
}
```

*Figure 2 - **Models/Book.cs***

3. Generate Controller & Views for model *Genre* with Scaffolding technique



Add MVC Controller with views, using Entity Framework

Model class: Genre (web.Models)

DbContext class: ApplicationDbContext (web.Data)

Views

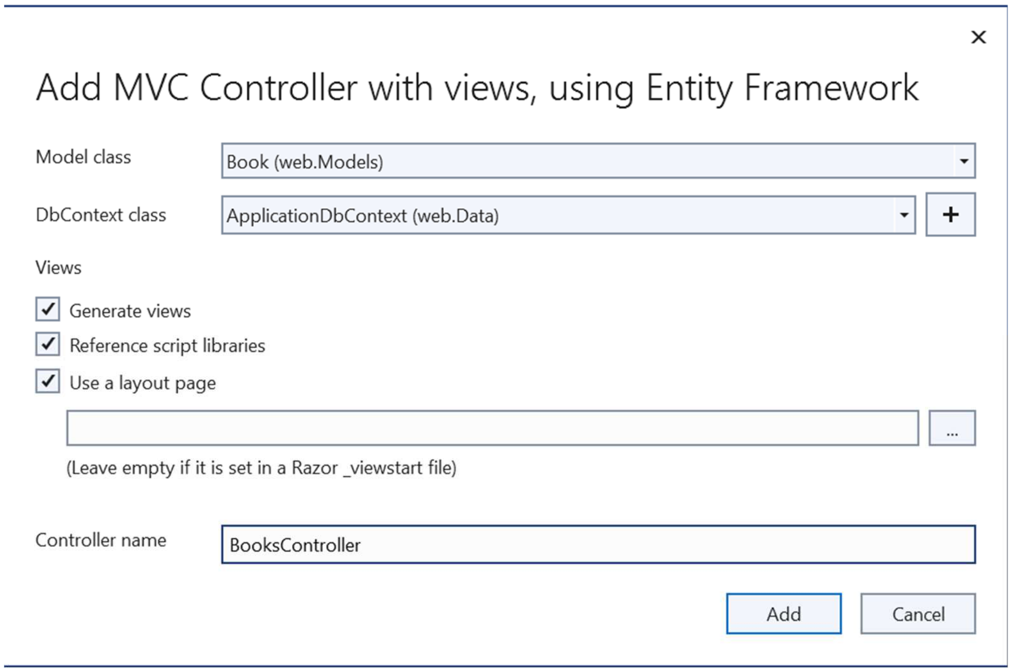
☑ Generate views
☑ Reference script libraries
☑ Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

Controller name: GenresController

Figure 3 - Generate ***Controller & Views*** *for model* ***Genre***

4. Re-generate Controller & Views for model *Book* to update *Genre* data



Add MVC Controller with views, using Entity Framework

Model class: Book (web.Models)

DbContext class: ApplicationDbContext (web.Data)

Views

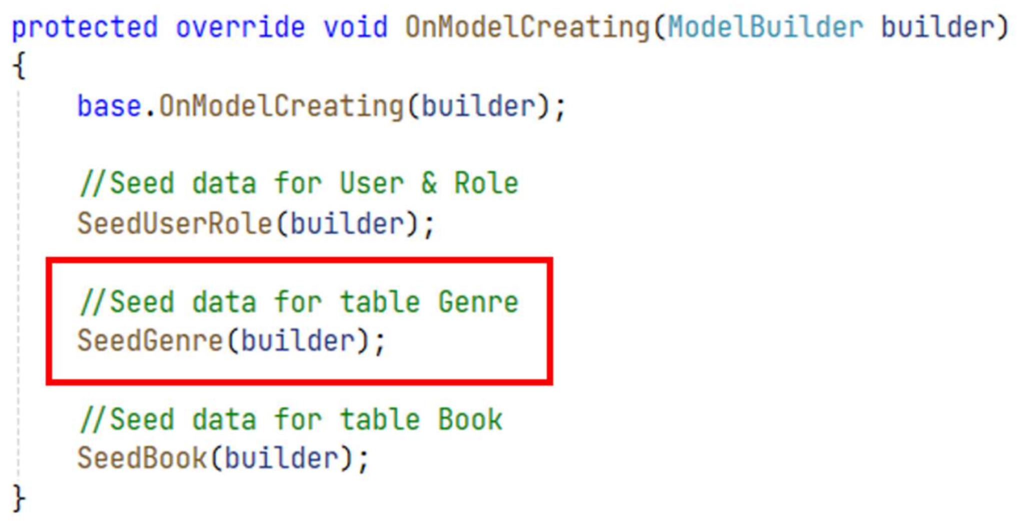
☑ Generate views
☑ Reference script libraries
☑ Use a layout page

(Leave empty if it is set in a Razor _viewstart file)

Controller name: BooksController

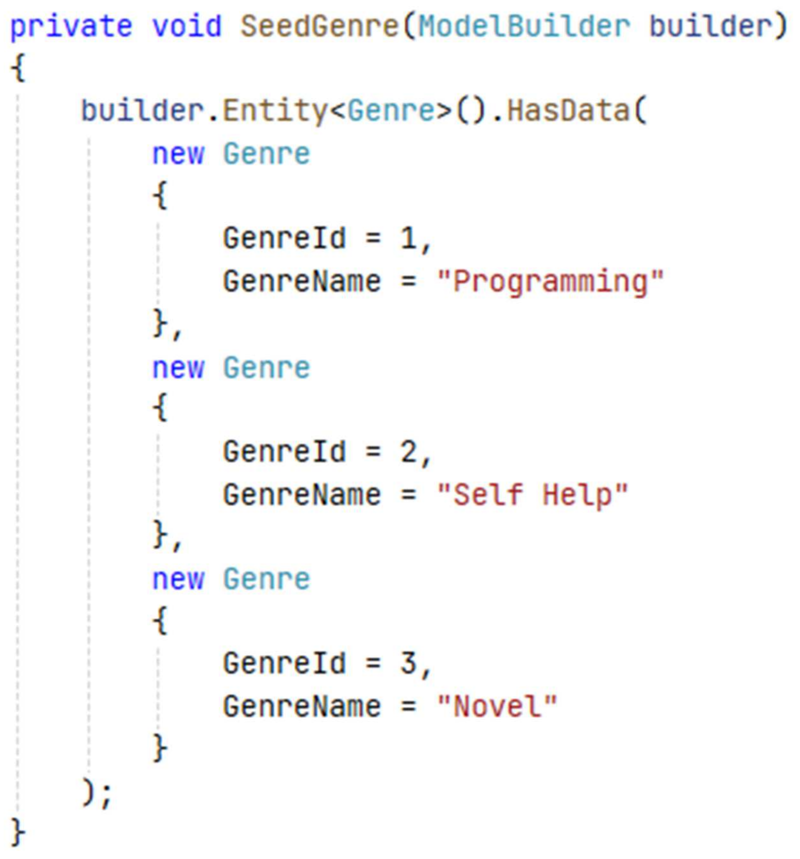
Figure 4 - Re-generate ***Controller & Views*** *for model* ***Book***

5. Update data seeding code for 2 models: *Genre* & *Book*



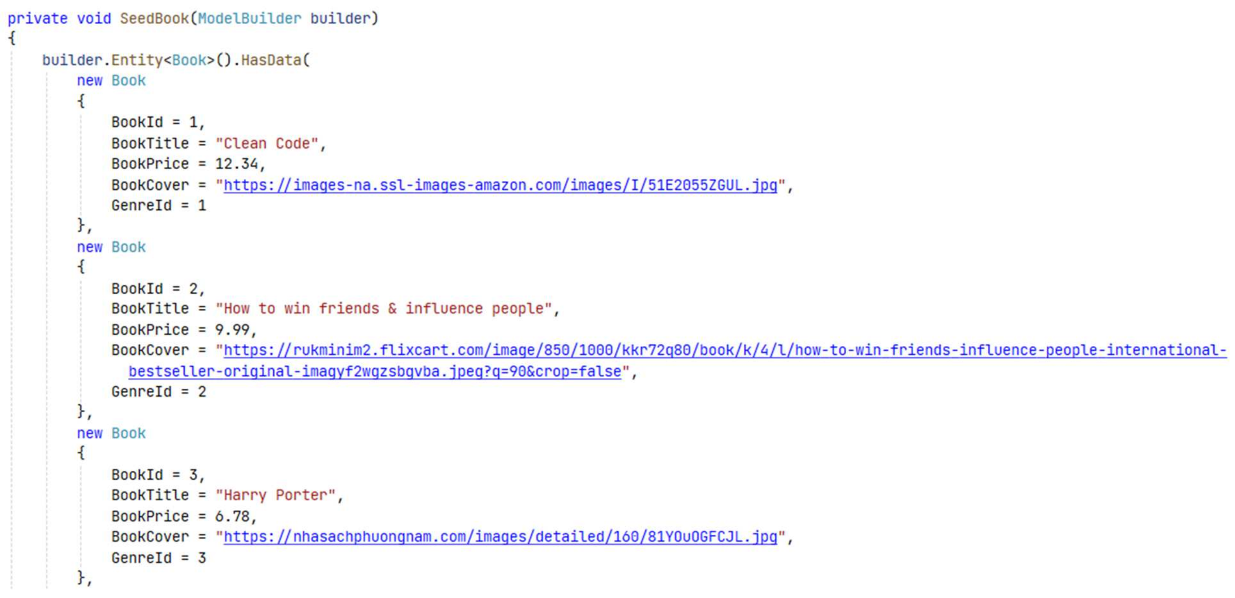
```
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);

    //Seed data for User & Role
    SeedUserRole(builder);

    //Seed data for table Genre
    SeedGenre(builder);

    //Seed data for table Book
    SeedBook(builder);
}
```

*Figure 5 - **Data/ApplicationDbContext.cs** (1)*



```
private void SeedGenre(ModelBuilder builder)
{
    builder.Entity<Genre>().HasData(
        new Genre
        {
            GenreId = 1,
            GenreName = "Programming"
        },
        new Genre
        {
            GenreId = 2,
            GenreName = "Self Help"
        },
        new Genre
        {
            GenreId = 3,
            GenreName = "Novel"
        }
    );
}
```

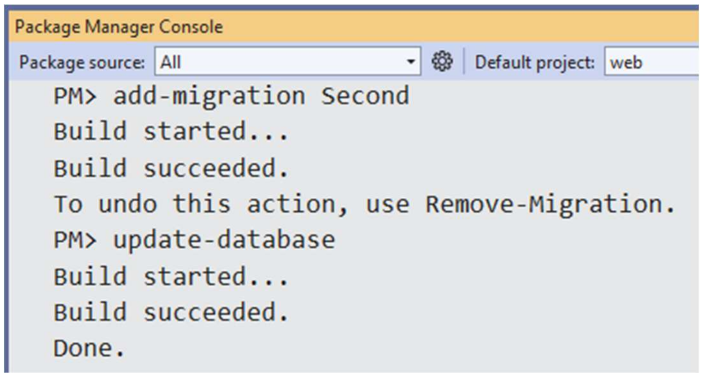
*Figure 6 - **Data/ApplicationDbContext.cs** (2)*



```
private void SeedBook(ModelBuilder builder)
{
    builder.Entity<Book>().HasData(
        new Book
        {
            BookId = 1,
            BookTitle = "Clean Code",
            BookPrice = 12.34,
            BookCover = "https://images-na.ssl-images-amazon.com/images/I/51E2055ZGUL.jpg",
            GenreId = 1
        },
        new Book
        {
            BookId = 2,
            BookTitle = "How to win friends & influence people",
            BookPrice = 9.99,
            BookCover = "https://rukminim2.flixcart.com/image/850/1000/kkr72q80/book/k/4/l/how-to-win-friends-influence-people-international-
              bestseller-original-imagyf2wgzsbgvba.jpeg?q=90&crop=false",
            GenreId = 2
        },
        new Book
        {
            BookId = 3,
            BookTitle = "Harry Porter",
            BookPrice = 6.78,
            BookCover = "https://nhasachphuongnam.com/images/detailed/160/81YOuOGFCJL.jpg",
            GenreId = 3
        },
```

*Figure 7 - **Data/ApplicationDbContext.cs** (3)*

6. Add new migration and update database again

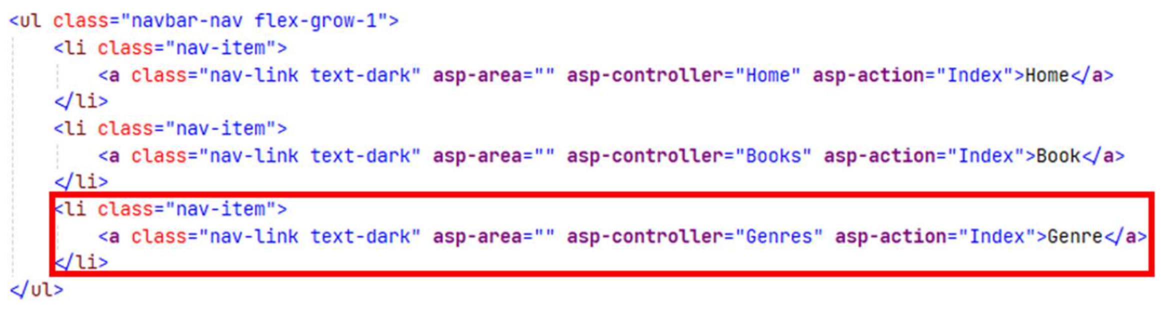


Package Manager Console
Package source: All | Default project: web

```
PM> add-migration Second
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
Build succeeded.
Done.
```

*Figure 8 - Use **PMC** to update database*

7. Update code of views



```
<ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Books" asp-action="Index">Book</a>
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Genres" asp-action="Index">Genre</a>
    </li>
</ul>
```

Figure 9 – Layout (Code)

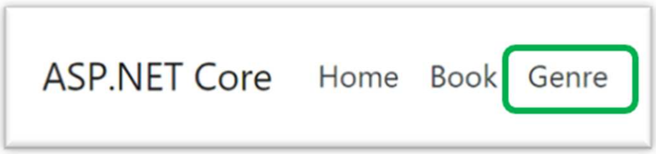
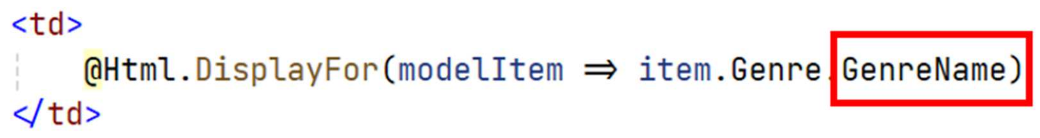


Figure 10 - Layout (Web)



```
<td>
    @Html.DisplayFor(modelItem => item.Genre.GenreName)
</td>
```

Figure 11 – Book Index (Code)

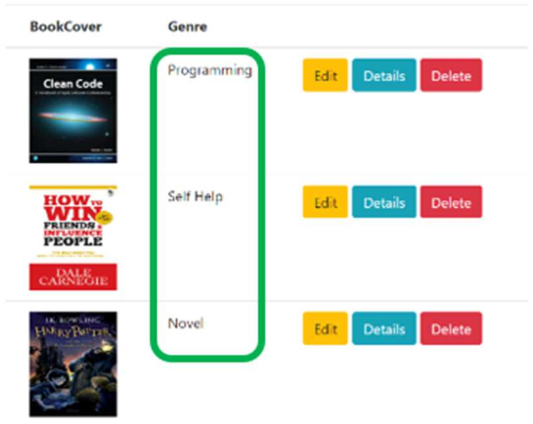
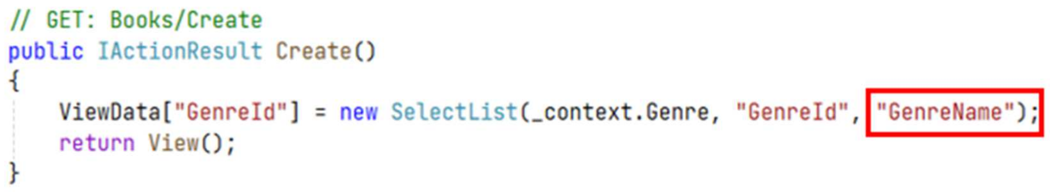


Figure 12 - Book Index (Web)



```
// GET: Books/Create
public IActionResult Create()
{
    ViewData["GenreId"] = new SelectList(_context.Genre, "GenreId", "GenreName");
    return View();
}
```

Figure 13 – Add Book (Code)

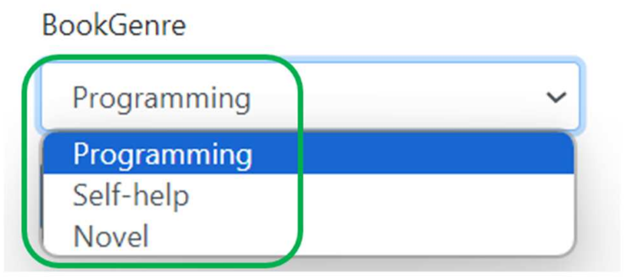
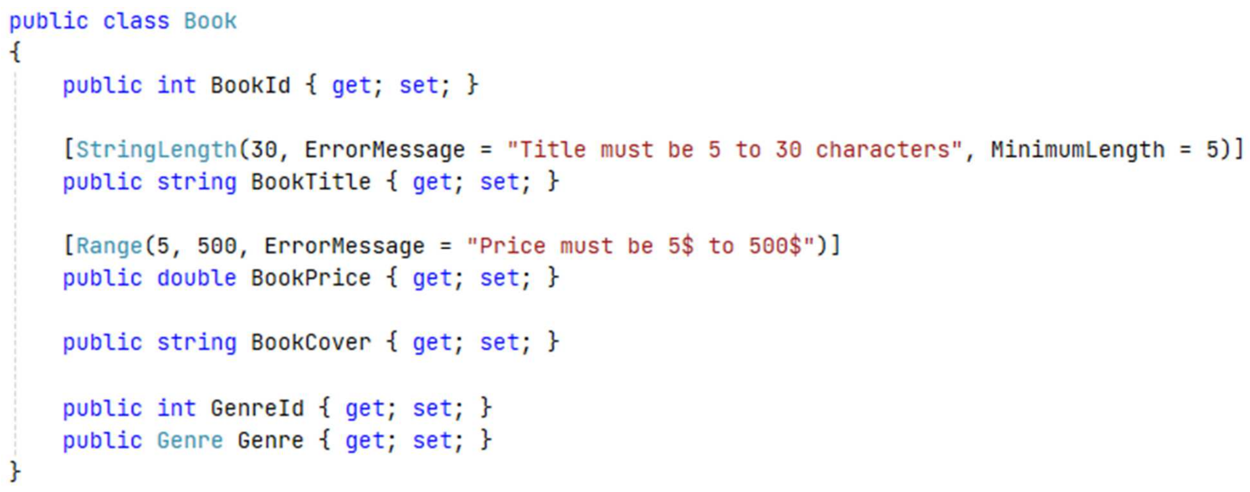


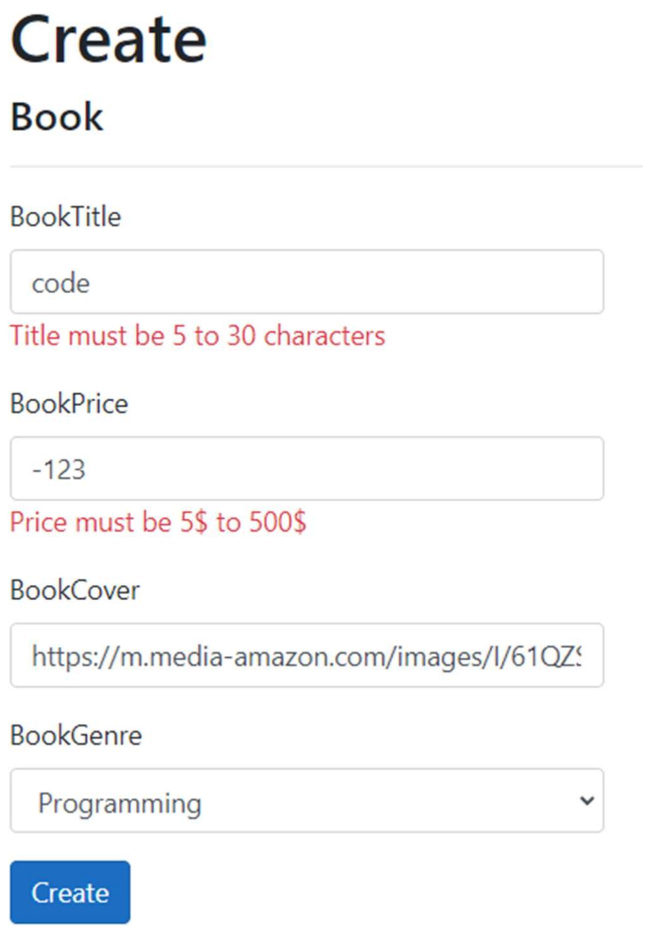
Figure 14 - Add Book (View)

8. Add model validation



```
public class Book
{
    public int BookId { get; set; }

    [StringLength(30, ErrorMessage = "Title must be 5 to 30 characters", MinimumLength = 5)]
    public string BookTitle { get; set; }

    [Range(5, 500, ErrorMessage = "Price must be 5$ to 500$")]
    public double BookPrice { get; set; }

    public string BookCover { get; set; }

    public int GenreId { get; set; }
    public Genre Genre { get; set; }
}
```

*Figure 15 - **Models/Book.cs (2)***



Create

Book

BookTitle

code

Title must be 5 to 30 characters

BookPrice

-123

Price must be 5$ to 500$

BookCover

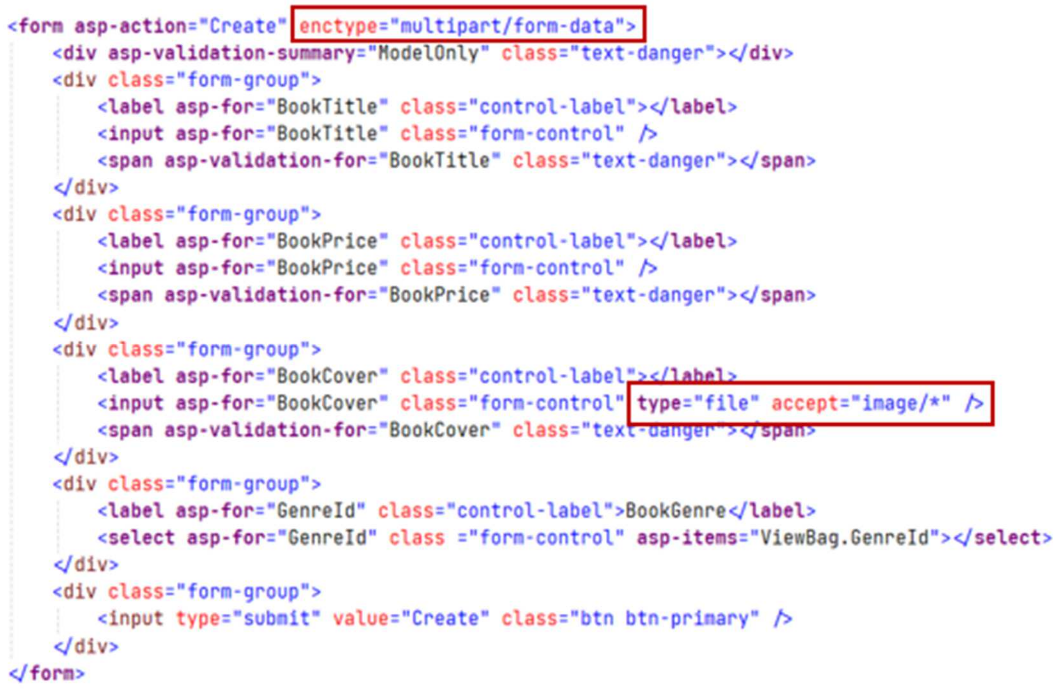
https://m.media-amazon.com/images/I/61QZ

BookGenre

Programming

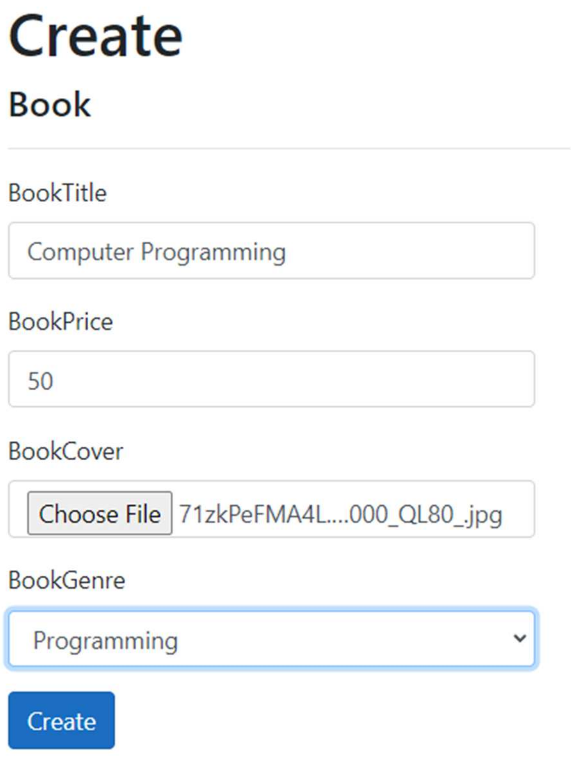
Figure 16 - Create Book (View)

9. Implement Upload image feature



```
<form asp-action="Create" enctype="multipart/form-data">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="BookTitle" class="control-label"></label>
        <input asp-for="BookTitle" class="form-control" />
        <span asp-validation-for="BookTitle" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="BookPrice" class="control-label"></label>
        <input asp-for="BookPrice" class="form-control" />
        <span asp-validation-for="BookPrice" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="BookCover" class="control-label"></label>
        <input asp-for="BookCover" class="form-control" type="file" accept="image/*" />
        <span asp-validation-for="BookCover" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="GenreId" class="control-label">BookGenre</label>
        <select asp-for="GenreId" class ="form-control" asp-items="ViewBag.GenreId"></select>
    </div>
    <div class="form-group">
        <input type="submit" value="Create" class="btn btn-primary" />
    </div>
</form>
```

*Figure 17 - **Views/Books/Create.cshtml***



Create

Book

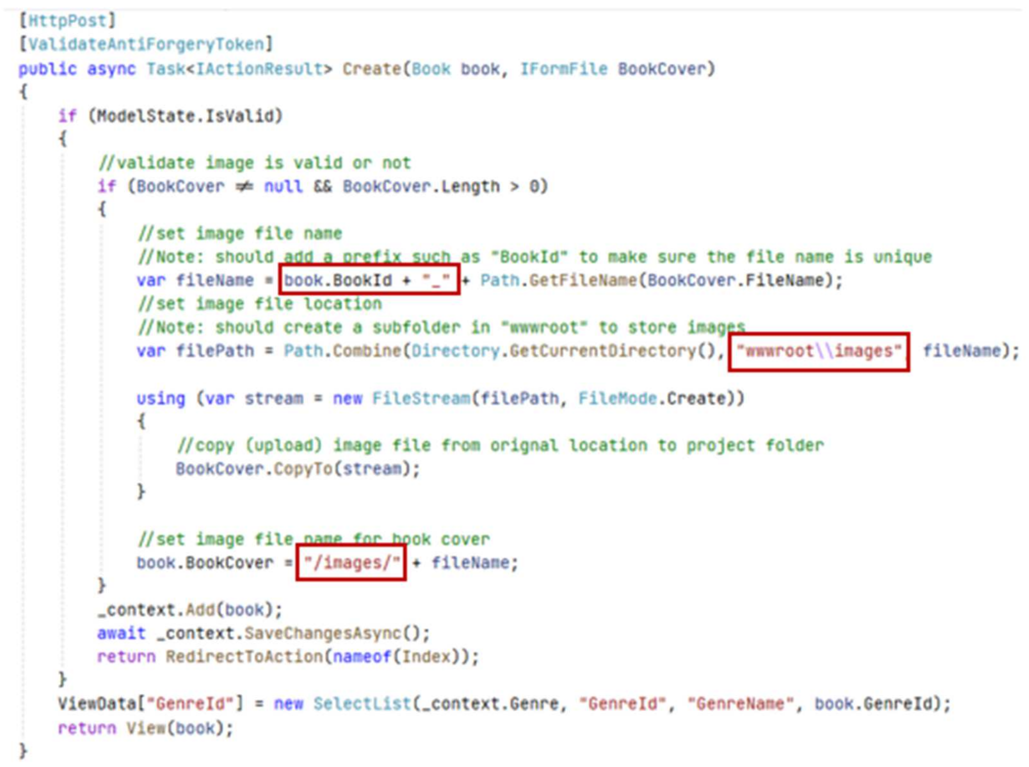
BookTitle: Computer Programming

BookPrice: 50

BookCover: Choose File 71zkPeFMA4L....000_QL80_.jpg

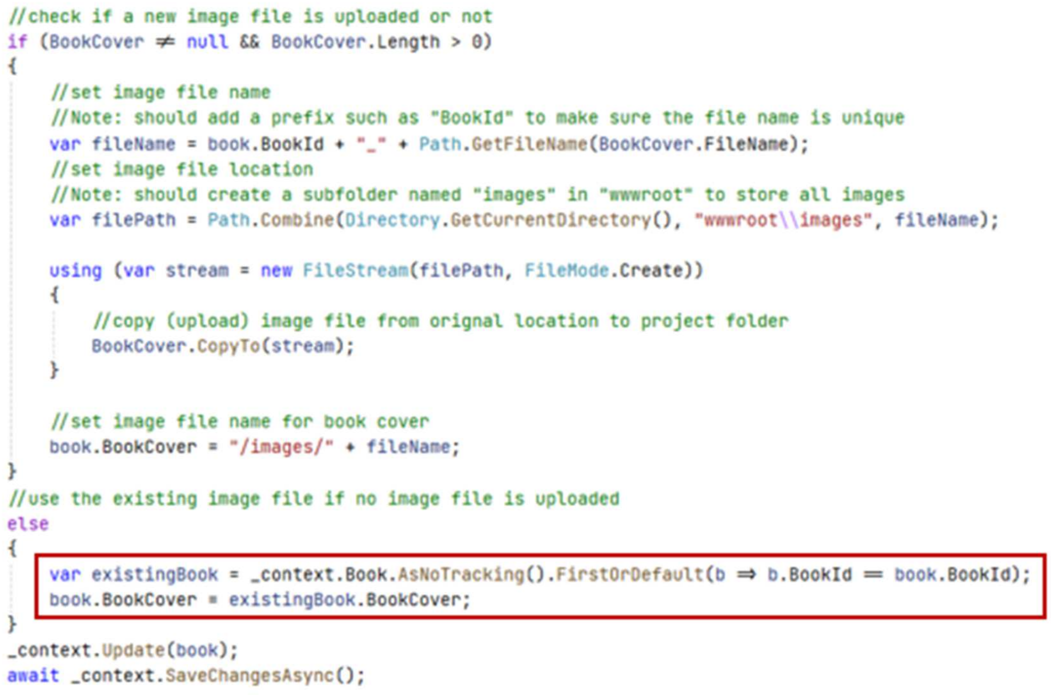
BookGenre: Programming

Figure 18 - Create Book (View)



```
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(Book book, IFormFile BookCover)
{
    if (ModelState.IsValid)
    {
        //validate image is valid or not
        if (BookCover != null && BookCover.Length > 0)
        {
            //set image file name
            //Note: should add a prefix such as "BookId" to make sure the file name is unique
            var fileName = book.BookId + "_" + Path.GetFileName(BookCover.FileName);
            //set image file location
            //Note: should create a subfolder in "wwwroot" to store images
            var filePath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\images", fileName);

            using (var stream = new FileStream(filePath, FileMode.Create))
            {
                //copy (upload) image file from orignal location to project folder
                BookCover.CopyTo(stream);
            }

            //set image file name for book cover
            book.BookCover = "/images/" + fileName;
        }
        _context.Add(book);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["GenreId"] = new SelectList(_context.Genre, "GenreId", "GenreName", book.GenreId);
    return View(book);
}
```

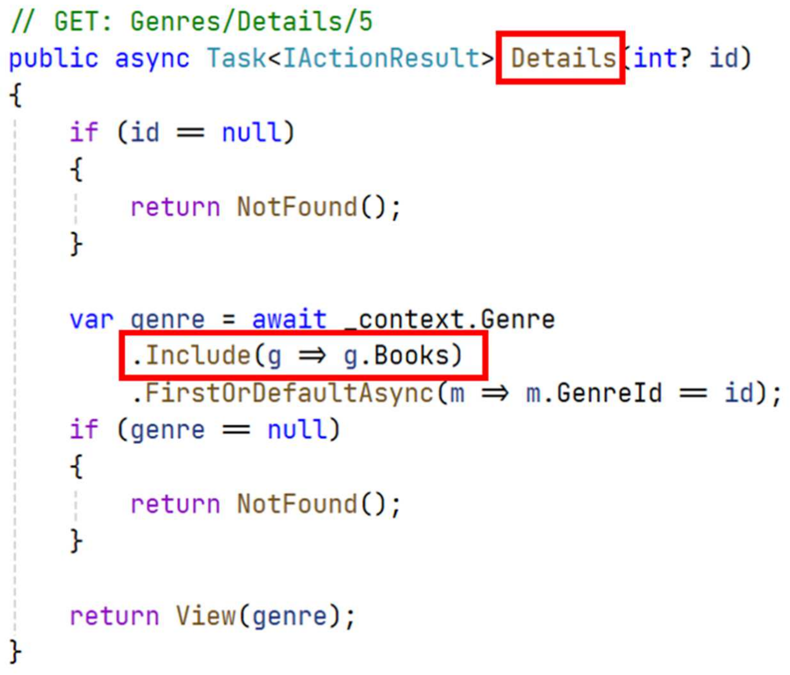
Figure 19 - ***Controllers/BooksController.cs – Create()***



```
//check if a new image file is uploaded or not
if (BookCover != null && BookCover.Length > 0)
{
    //set image file name
    //Note: should add a prefix such as "BookId" to make sure the file name is unique
    var fileName = book.BookId + "_" + Path.GetFileName(BookCover.FileName);
    //set image file location
    //Note: should create a subfolder named "images" in "wwwroot" to store all images
    var filePath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot\\images", fileName);

    using (var stream = new FileStream(filePath, FileMode.Create))
    {
        //copy (upload) image file from orignal location to project folder
        BookCover.CopyTo(stream);
    }

    //set image file name for book cover
    book.BookCover = "/images/" + fileName;
}
//use the existing image file if no image file is uploaded
else
{
    var existingBook = _context.Book.AsNoTracking().FirstOrDefault(b => b.BookId == book.BookId);
    book.BookCover = existingBook.BookCover;
}
_context.Update(book);
await _context.SaveChangesAsync();
```

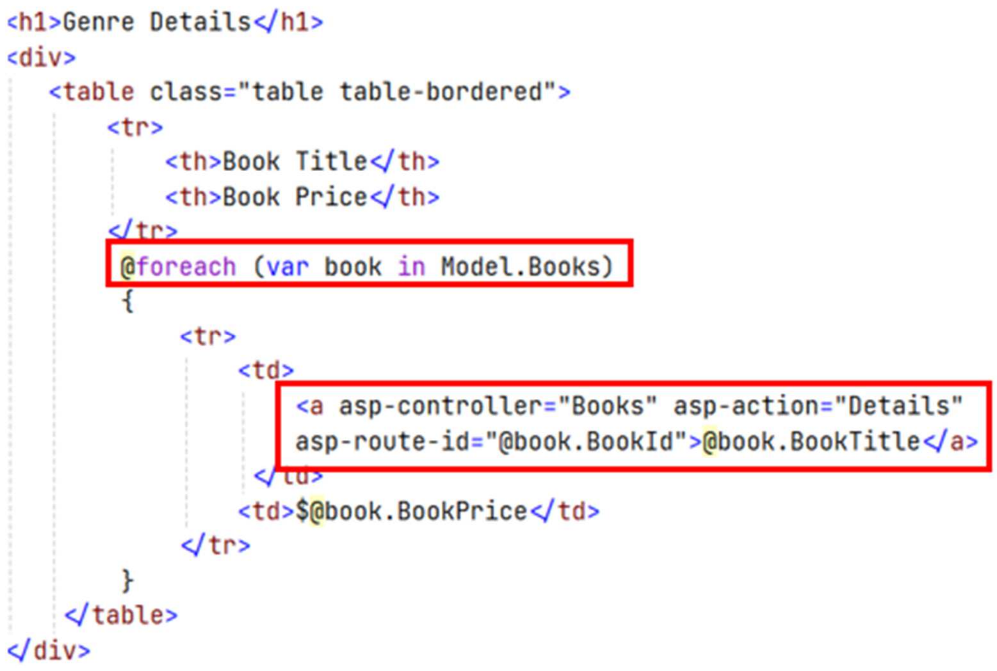
Figure 20 ***- Controllers/BooksController.cs – Edit()***

10.Implement Filter feature: filter books by genre



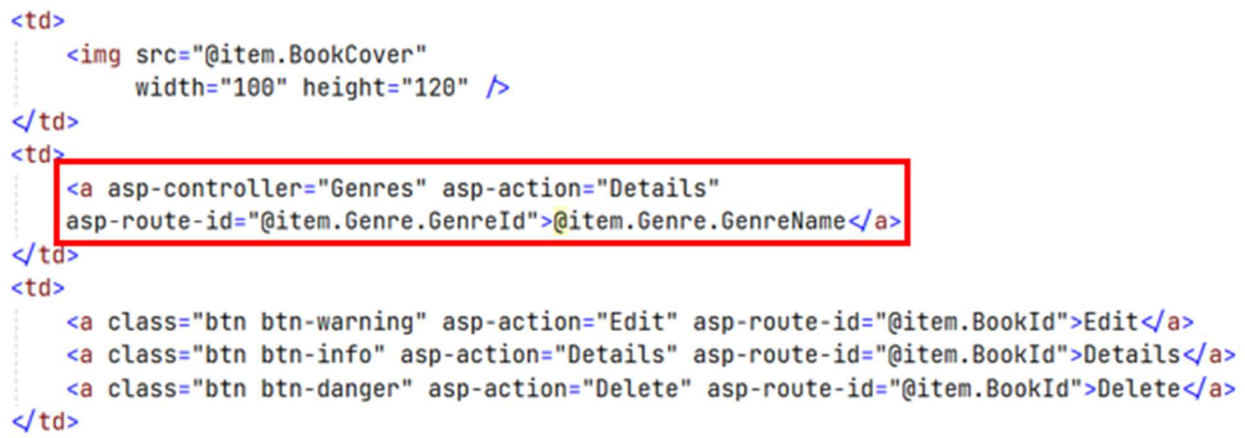
```
// GET: Genres/Details/5
public async Task<IActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var genre = await _context.Genre
        .Include(g => g.Books)
        .FirstOrDefaultAsync(m => m.GenreId == id);
    if (genre == null)
    {
        return NotFound();
    }

    return View(genre);
}
```

*Figure 21 - **Controllers/GenresController.cs***



```
<h1>Genre Details</h1>
<div>
   <table class="table table-bordered">
       <tr>
           <th>Book Title</th>
           <th>Book Price</th>
       </tr>
       @foreach (var book in Model.Books)
       {
           <tr>
               <td>
                   <a asp-controller="Books" asp-action="Details"
                   asp-route-id="@book.BookId">@book.BookTitle</a>
               </td>
               <td>$@book.BookPrice</td>
           </tr>
       }
   </table>
</div>
```

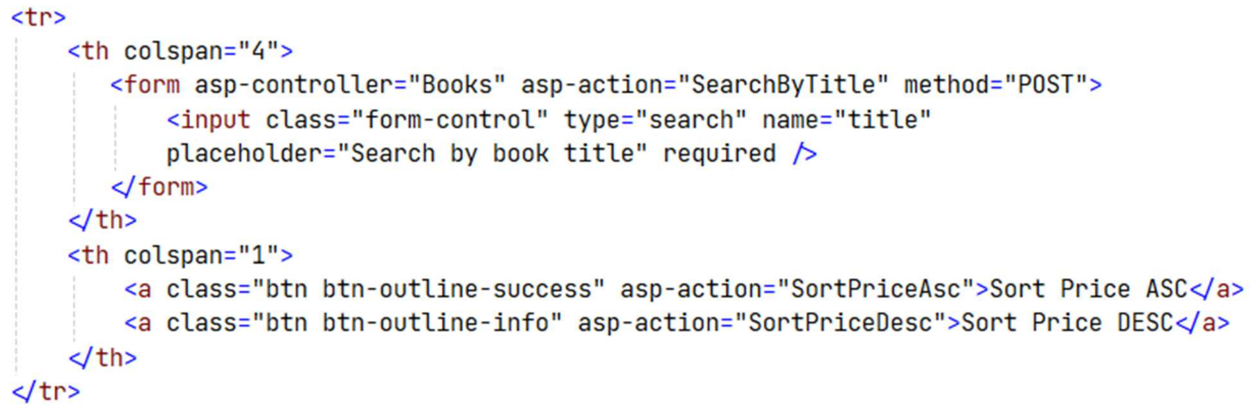
*Figure 22 - **Views/Genres/Detail.cshtml***



```
<td>
    <img src="@item.BookCover"
         width="100" height="120" />
</td>
<td>
    <a asp-controller="Genres" asp-action="Details"
    asp-route-id="@item.Genre.GenreId">@item.Genre.GenreName</a>
</td>
<td>
    <a class="btn btn-warning" asp-action="Edit" asp-route-id="@item.BookId">Edit</a>
    <a class="btn btn-info" asp-action="Details" asp-route-id="@item.BookId">Details</a>
    <a class="btn btn-danger" asp-action="Delete" asp-route-id="@item.BookId">Delete</a>
</td>
```

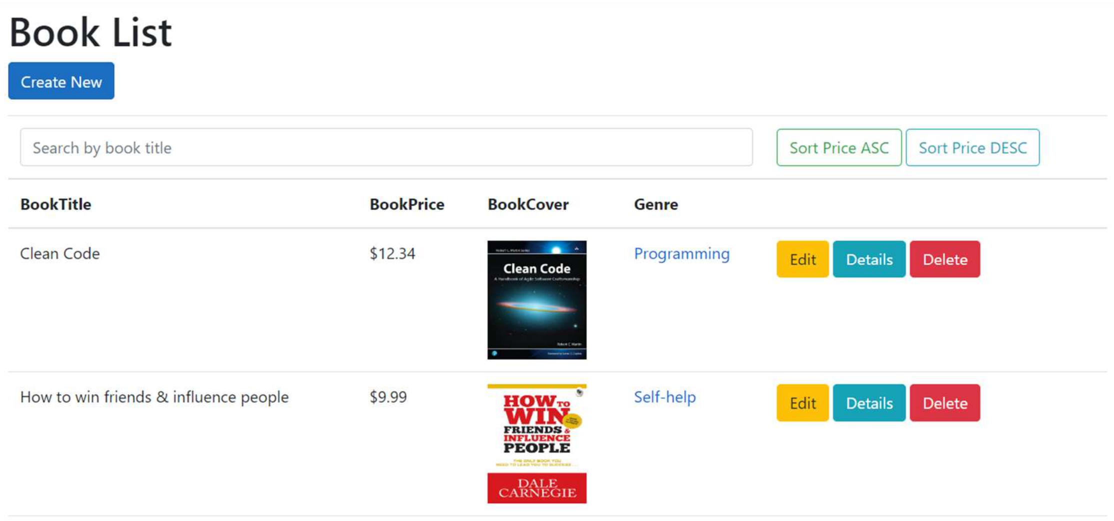
*Figure 23 - **Views/Books/Index.cshtml (1)***

11. Implement Search & Sort feature: search by title, sort by price



```
<tr>
    <th colspan="4">
        <form asp-controller="Books" asp-action="SearchByTitle" method="POST">
            <input class="form-control" type="search" name="title"
            placeholder="Search by book title" required />
        </form>
    </th>
    <th colspan="1">
         <a class="btn btn-outline-success" asp-action="SortPriceAsc">Sort Price ASC</a>
         <a class="btn btn-outline-info" asp-action="SortPriceDesc">Sort Price DESC</a>
    </th>
</tr>
```

*Figure 24 – **Views/Books/Index.cshtml***



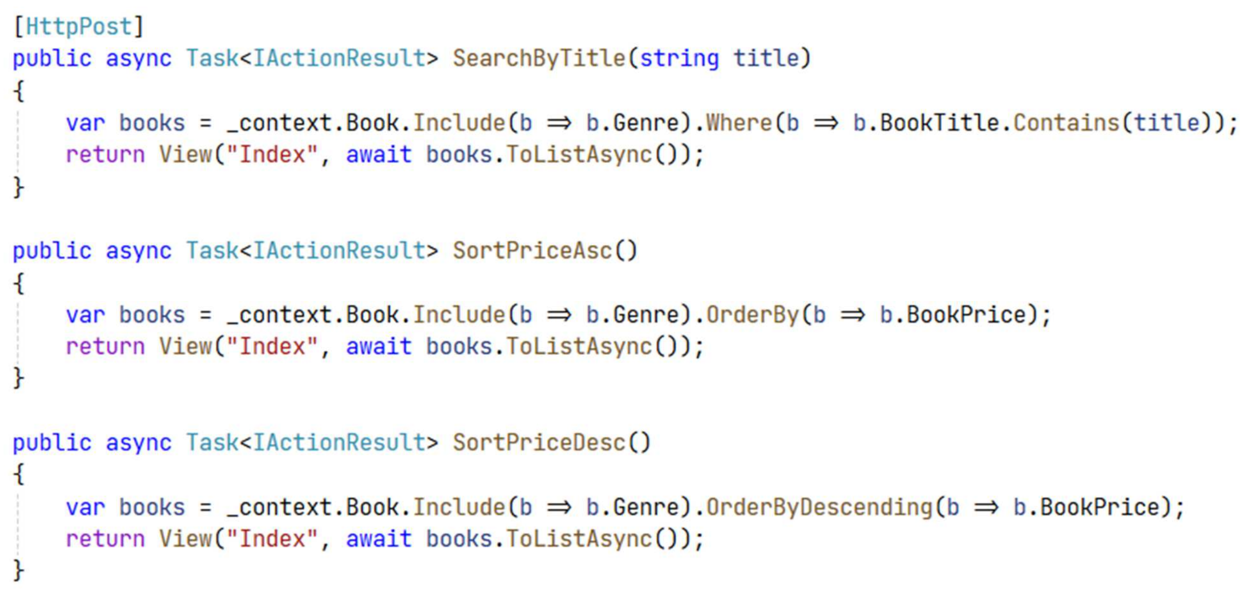
Book List

Create New

Search by book title | Sort Price ASC | Sort Price DESC

BookTitle	BookPrice	BookCover	Genre	
Clean Code	$12.34		Programming	Edit Details Delete
How to win friends & influence people	$9.99		Self-help	Edit Details Delete

Figure 25 - Book Index (Web)



```
[HttpPost]
public async Task<IActionResult> SearchByTitle(string title)
{
    var books = _context.Book.Include(b ⇒ b.Genre).Where(b ⇒ b.BookTitle.Contains(title));
    return View("Index", await books.ToListAsync());
}

public async Task<IActionResult> SortPriceAsc()
{
    var books = _context.Book.Include(b ⇒ b.Genre).OrderBy(b ⇒ b.BookPrice);
    return View("Index", await books.ToListAsync());
}

public async Task<IActionResult> SortPriceDesc()
{
    var books = _context.Book.Include(b ⇒ b.Genre).OrderByDescending(b ⇒ b.BookPrice);
    return View("Index", await books.ToListAsync());
}
```

Figure 26 - ***Controllers/BooksController.cs – SearchByTitle() + SortPriceAsc() + SortPriceAsc()***