

# LAB 2

## DEVELOP ASP.NET CORE WEB APP (P2)

### ❖ CONTENT

- Setup relationship between tables
- Add data (model) validation
- Implement Filter, Search & Sort features

### ❖ 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

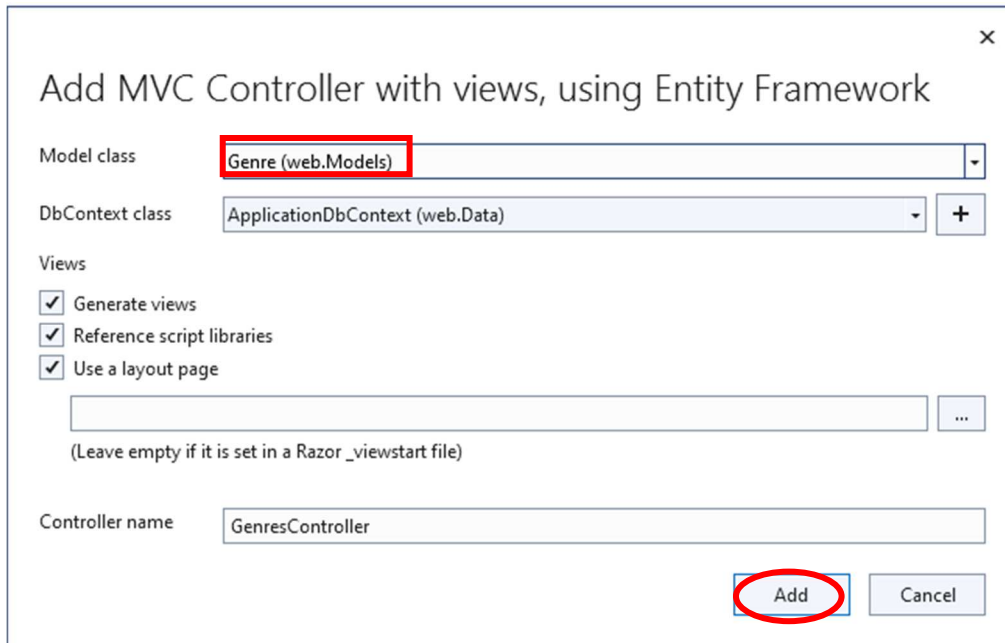
2. Update model *Book* to setup relationship with model *Genre*

```
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

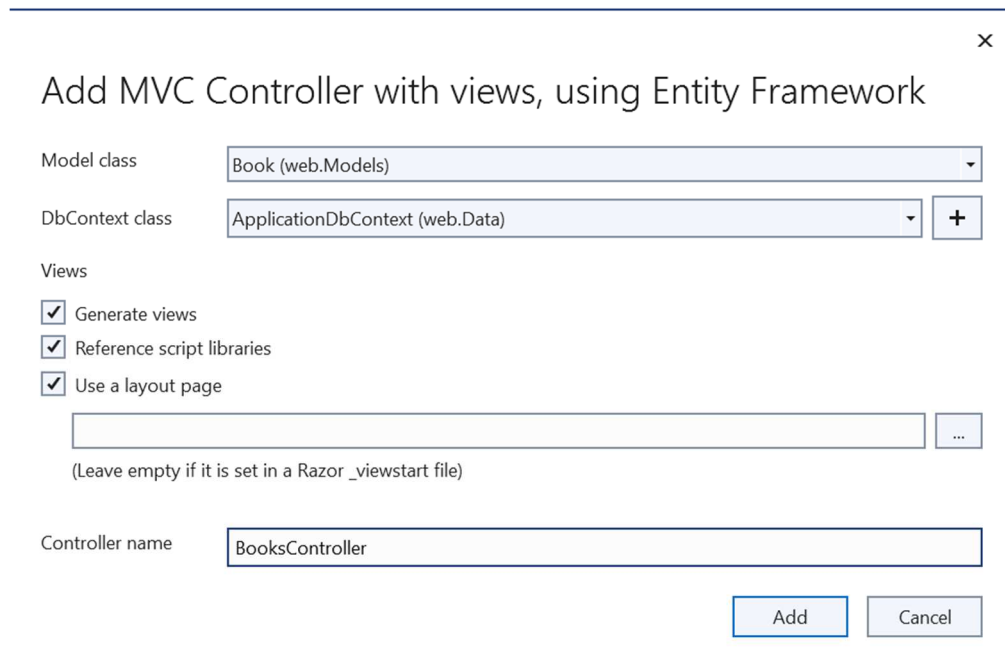
(Leave empty if it is set in a Razor \_viewstart file)

Controller name: GenresController

Add Cancel

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

Add Cancel

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)

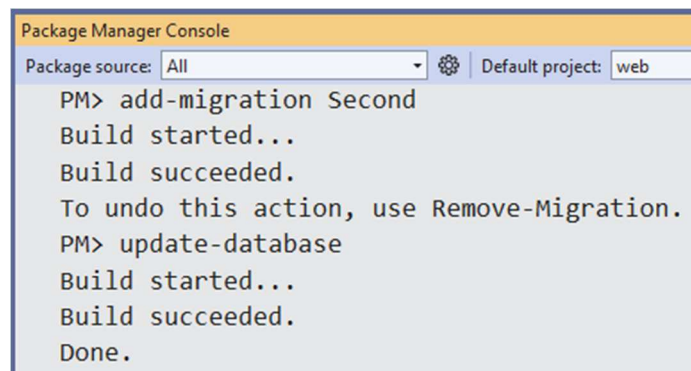
```

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-imageyf2wqzsbqvba.jpeg?q=90&crop=false",
            GenreId = 2
        },
        new Book
        {
            BookId = 3,
            BookTitle = "Harry Porter",
            BookPrice = 6.78,
            BookCover = "https://nhasachphuongnam.com/images/detailed/160/81Y0u0GFCJL.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)

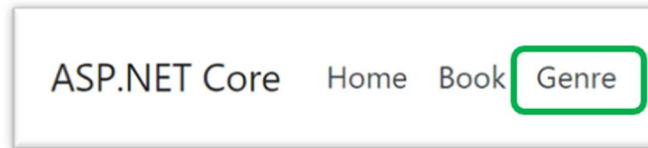


Figure 10 - Layout (Web)

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

Figure 11 – Book Index (Code)

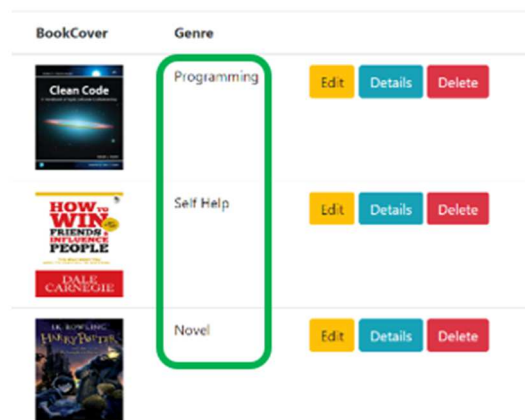


Figure 12 - Book Index (Web)

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

Figure 13 – Add Book (Code)

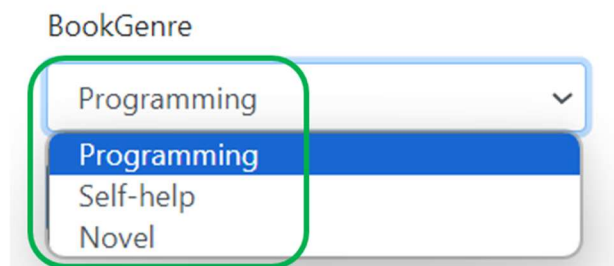


Figure 14 - Add Book (View)

## 8. Add model validation

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

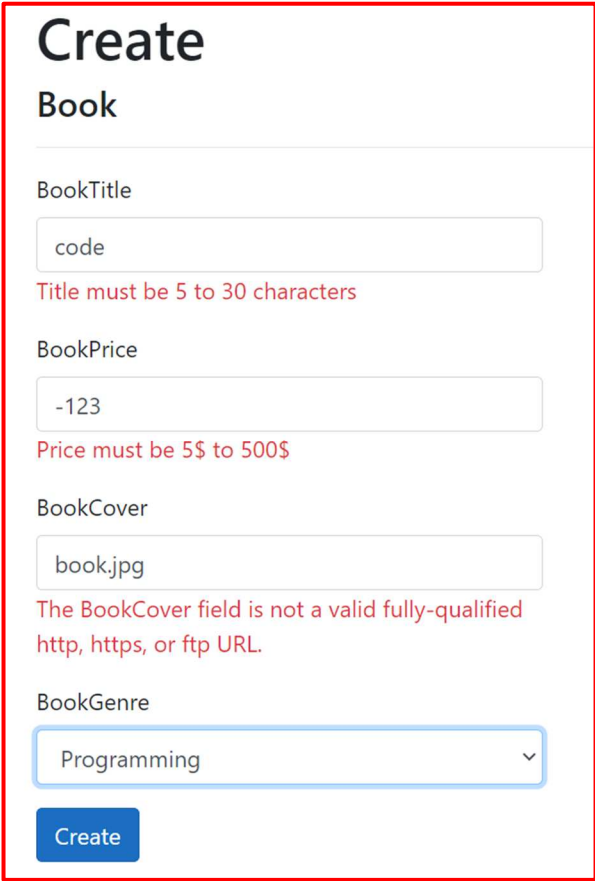
    [StringLength(30, ErrorMessage = "Title must be 5 to 30 characters", MinimumLength = 5)]
    public string BookTitle { get; set; }

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

    [Url]
    public string BookCover { get; set; }

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

Figure 15 - Models/Book.cs (2)



The screenshot shows a web form titled "Create Book". It contains five input fields: "BookTitle" (text), "BookPrice" (text), "BookCover" (text), "BookGenre" (dropdown), and a "Create" button. The "BookTitle" field contains the text "code" and has a red error message below it: "Title must be 5 to 30 characters". The "BookPrice" field contains the text "-123" and has a red error message below it: "Price must be 5\$ to 500\$". The "BookCover" field contains the text "book.jpg" and has a red error message below it: "The BookCover field is not a valid fully-qualified http, https, or ftp URL." The "BookGenre" dropdown menu is open, showing "Programming" as the selected option. The "Create" button is a blue button with white text.

Figure 16 - Create Book (View)

## 9. 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 17 - 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 18 - Views/Genres/Detail.cshtml



```

<td>
    
</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 19 - Views/Books/Index.cshtml (1)

## 10. 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 20 – Views/Books/BooksController.cs

## Book List

Create New

Search by book title

Sort Price ASC

Sort Price DESC


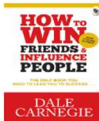
BookTitle	BookPrice	BookCover	Genre	
Clean Code	\$12.34		Programming	<a>Edit</a> <a>Details</a> <a>Delete</a>
How to win friends & influence people	\$9.99		Self-help	<a>Edit</a> <a>Details</a> <a>Delete</a>

Figure 21 - 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 22 - *Controllers/BooksController.cs*