**Where we are:**

|  |  |
| --- | --- |
| Intro to MVC | Navigation, Searching and Filtering |
| Controllers and Views | Forms and Tag Helpers |
| Unit Testing, Publishing to Azure | Annotation and validation |
| Dependency Injection, Models | Authentication |
| Entity Framework and Scaffolding | Authorization |

Intro

Filtering, page 236

* Products = repository.Products  
  Where(p => category == null || p.Category == category)  
  OrderBy(p => p.ProductID)
* Unit testing the filter
* Ch. 19 – Filters

Views, Ch. 21

Page 669, Adding Dynamic Content to a Razor View

Using the BookInfo Project:

1. Modify the Book Model
   1. Add a Genre property to the Book model
   2. Add genres to the books in SeedData
   3. Delete the database and run Update-Database
2. Add a search method to the Book Controller
   1. Add a BooksByGenre method to the BookController, it will use the Index view.  
      Test it using a parameter on the URL: Book/BooksByGenre?genre=”Fantasy”
   2. Add new ActionLinks on the Home/Index page to invoke Book/BooksByGenre with several different genre arguments. For the route parameter use: new {Genre = genre}
3. Move the menu to a shared layout
   1. Create \_Layout in the Shared folder and add the HTML boiler plate and menu code from the Home/Index view to it.
   2. Add a \_ViewStart file to the View folder and set \_Layout to be the layout for all views

Notes:

* We had to add .Include(b => b.Author) to GetAllBooks in the repository I should move that to the BookController method, but first I have to change GetAllBooks to return an IQueryable
* You can keep VS running and edit .cs code and then just refresh the browser without stopping and restarting execution in VS