Book Store: Book Tea

Index:

Project Name: Book Tea

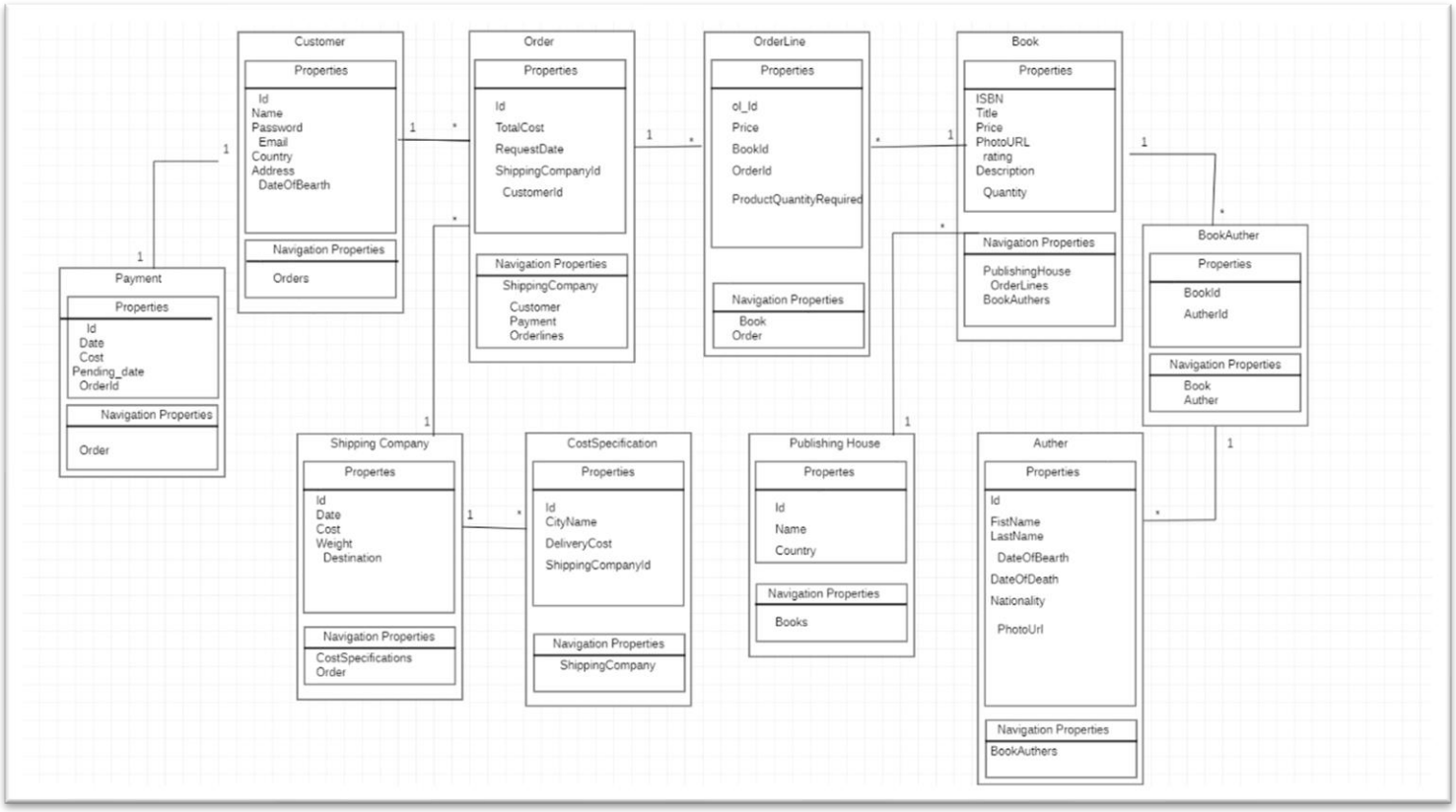
Dr: Hossam El-Adlly

Eng. Mohamed Abo-Suarie, **Eng.** Mohamed Saeed

Team Names:

1. Rawan Hassan Fathy
2. Roaa Bassam Naser
3. Menatallah Essam Anter
4. Mahmoud Atef Abdelaziz
5. Belal Samir Nasirallah

Data Model



1. Introduction to Library Store MVC .NET Project

This project is reflective of a simplified web store. A web store, like any complex system, requires the transmission of data between many discrete parts of the system. These parts include the user of the system (who visits the web store), the server (that the web store exists upon), and the data store (that logs and dispatches the state of the system). These communications between a user, a server, and a data store occur in response to actions performed against the web store. Each action has a request object, and this request object is dispatched to the parts of the system necessary to act upon that request. In an object-oriented system, the parts of the system able to dispatch instructions are called the controllers. The sum of this system is called a Model-View-Controller (MVC) project.

1.1. Overview of MVC Architecture

The four layers of the MVC pattern should facilitate the use of any interface technology we use (Forms, UWP, Xamarin, ASP.NET Core, etc.) throughout the application. The entire system should be able to pull data from the same session from any view. Utility will be created to assist with data classification, which will be discussed in the final chapters.

Therefore, in our system, Lectures Store, we will use the MC architecture developed for the application and regulate the administrative functions. Its structural representation will be: - Model: Responsible for the business logic of the activity, be it data delivery or processing of rules. - View: Glues the user interface to the application business models, sharing display information or receiving user data. In general, MVC builds two types of views: user view and admin view. These will be distinct. - Controller: Accountable for dealing with user requests, either displaying information or processing control and manipulating information execution. The objective of the controller will be to pass the addresses of information or actions that will be executed on the Web.

Employing an MVC architecture allows for better presentation, facilitates testing, designs software, and promotes modularity. Model-View-Controller permits multiple instances of View and calls the direction of needs from the controller's function. This assembly can include coordinating code for views and models.

MVC architecture takes into account the following three components: - Model: Refers to the layer of the application involved in managing all the rules, process logic, and data. - View: This layer represents the output of the user interface that efficiently uses user input, requests, and responses. The View visualizes the Model and collects the necessary information for the user interface. - Controller: Refers to the user input/control and is responsible for controlling the channel to user input and directing requests.

Model-View-Controller (MVC) is a well-known architectural pattern to build applications in Microsoft .NET. MVC is distinguished by its decentralized and completely interchangeable elements. The system architecture becomes flexible, maintainable, and robust. The developed application can be tested and developed independently. Data analysis and classification are utilized to describe and separate module functions. It is created with four layers: presentation, business module, data modules, and utility.

3. Controllers

3.1. Authors

Actions

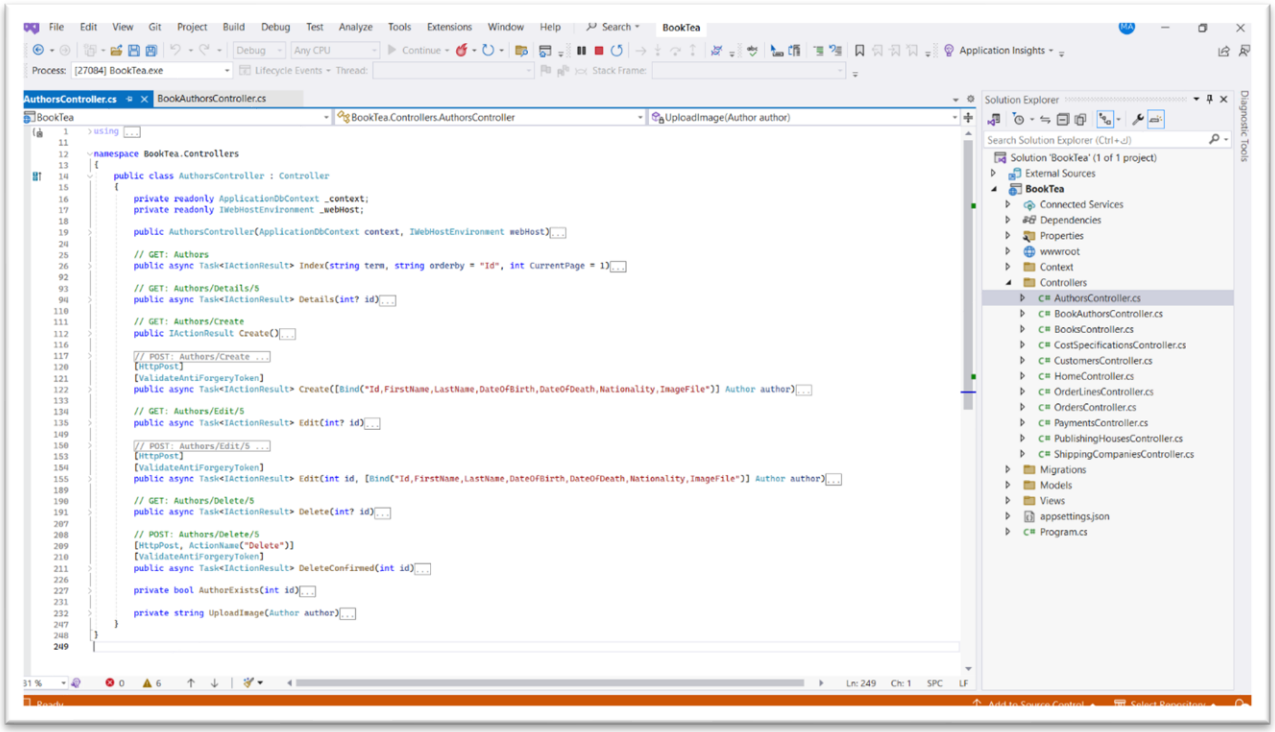
The “AuthorsController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null or an author is not found, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.2. Book Authors

Actions

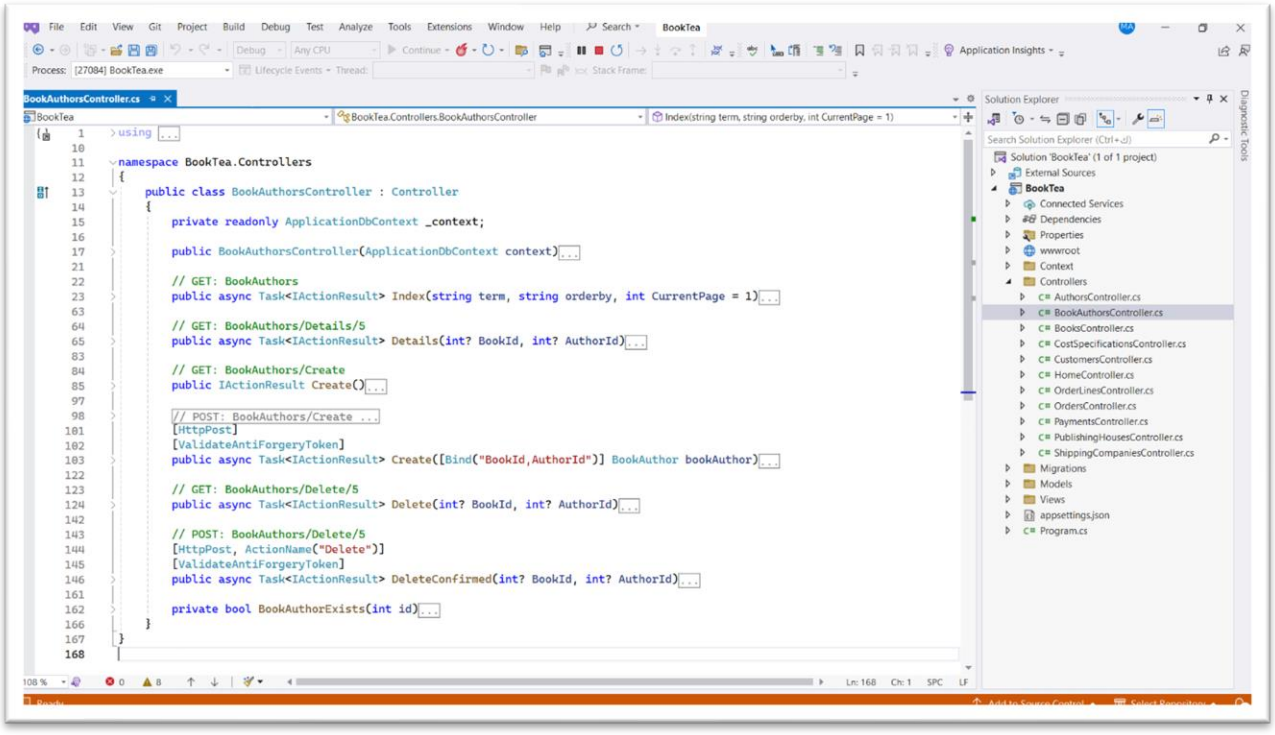
The "BookAuthorsController" includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.3. Books

Actions

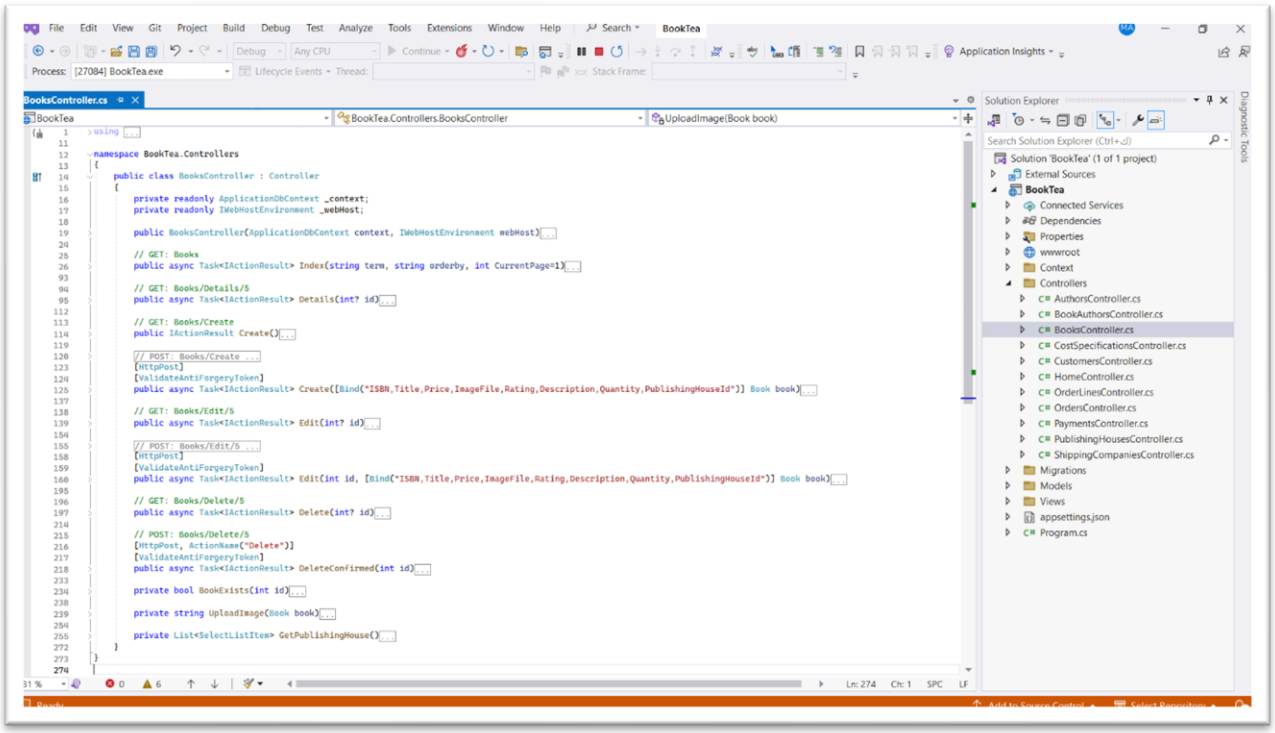
The “BooksController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.4. Cost Specifications

Actions

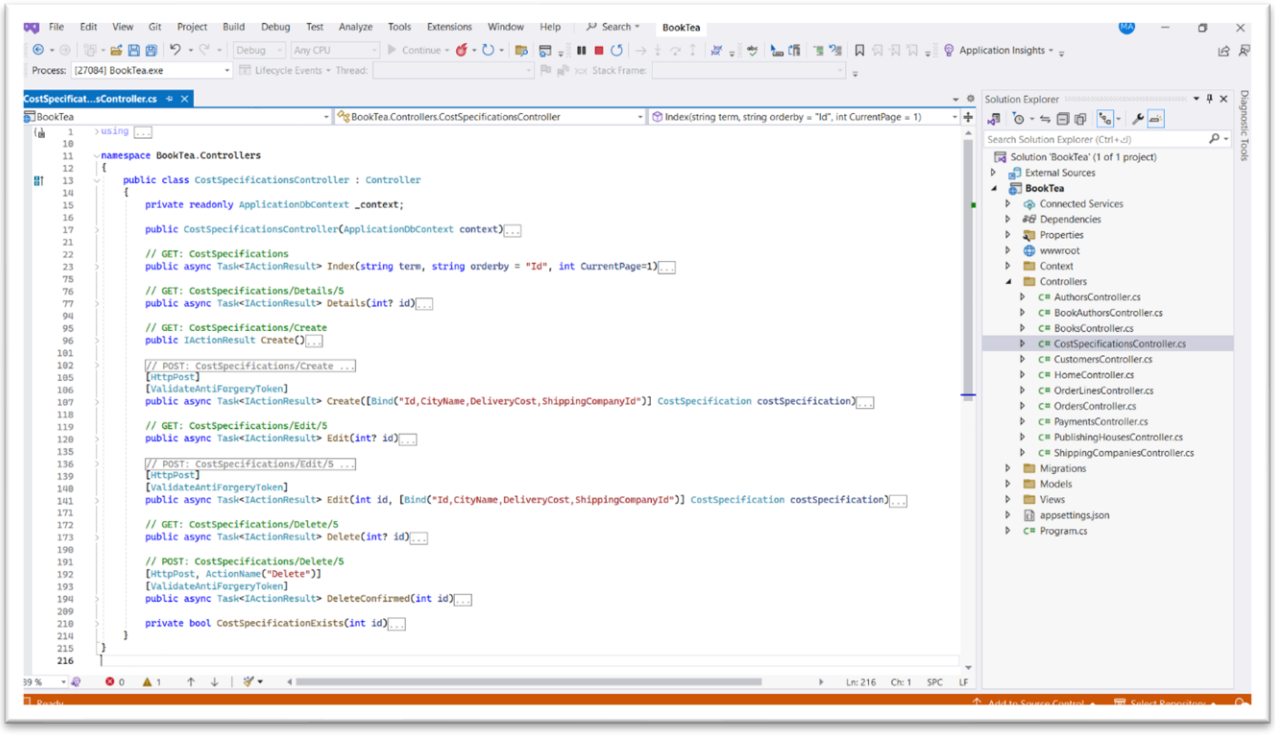
The “CostSpecificationsController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.5. Customers

Actions

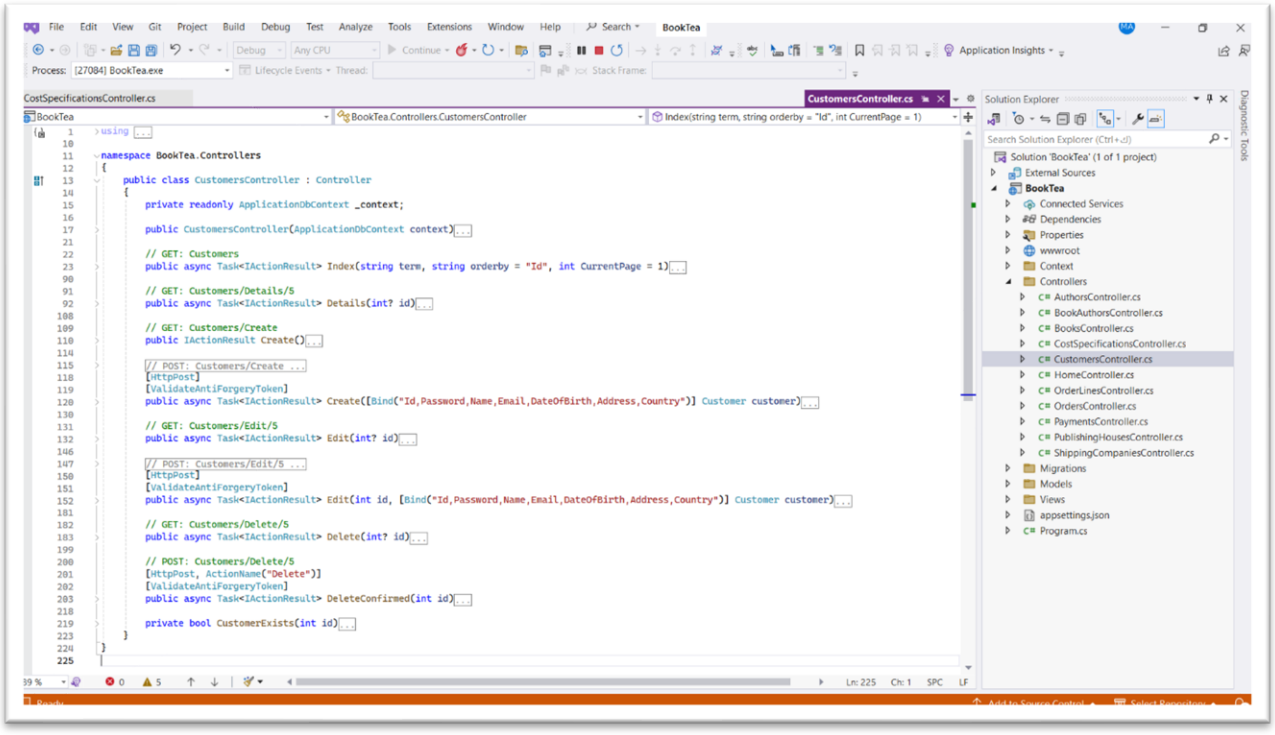
The “CustomersController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.6. Home

Actions

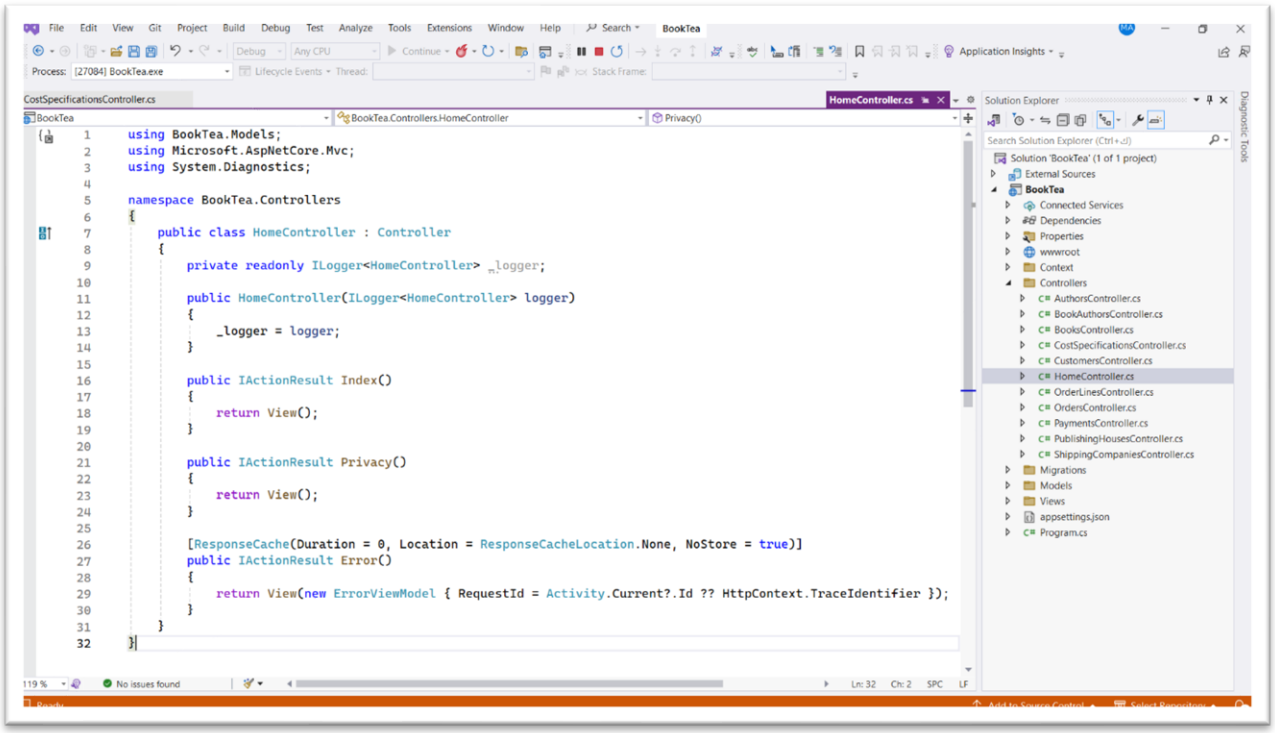
The “HomeController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.7. Order Lines

Actions

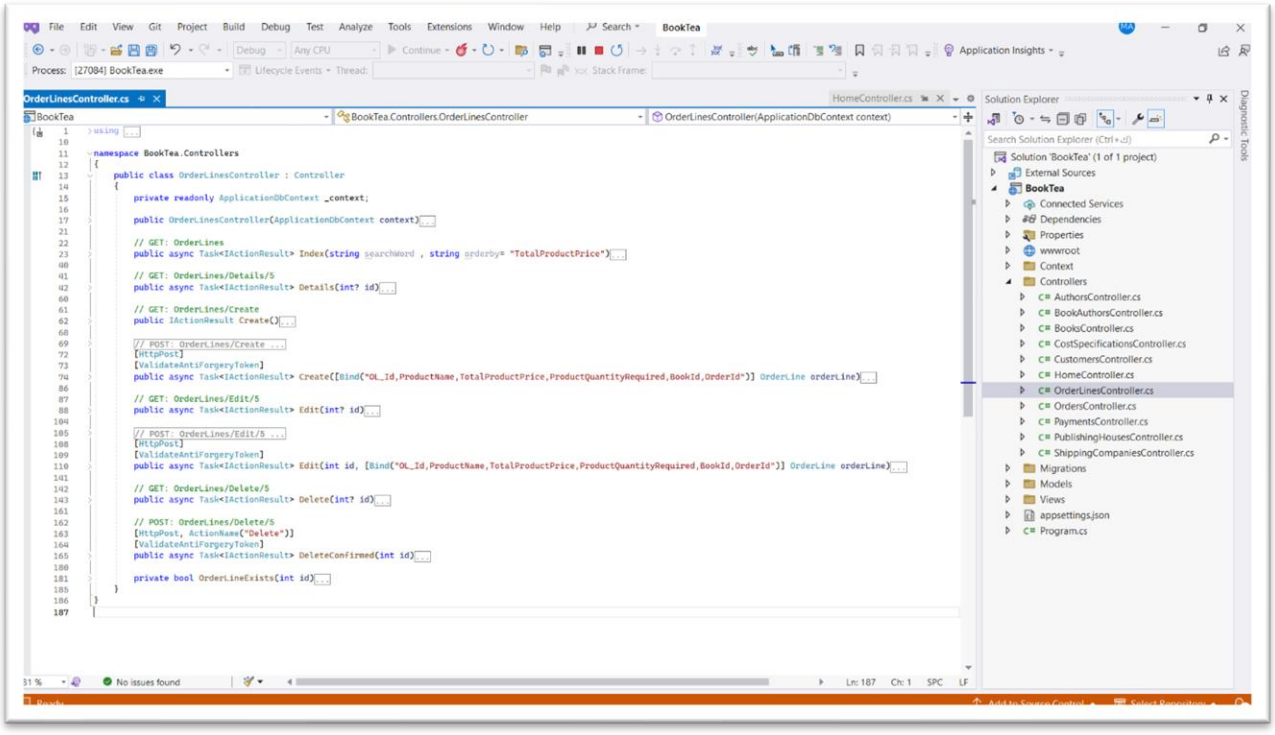
The “OrderLinesController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.8. Orders

Actions

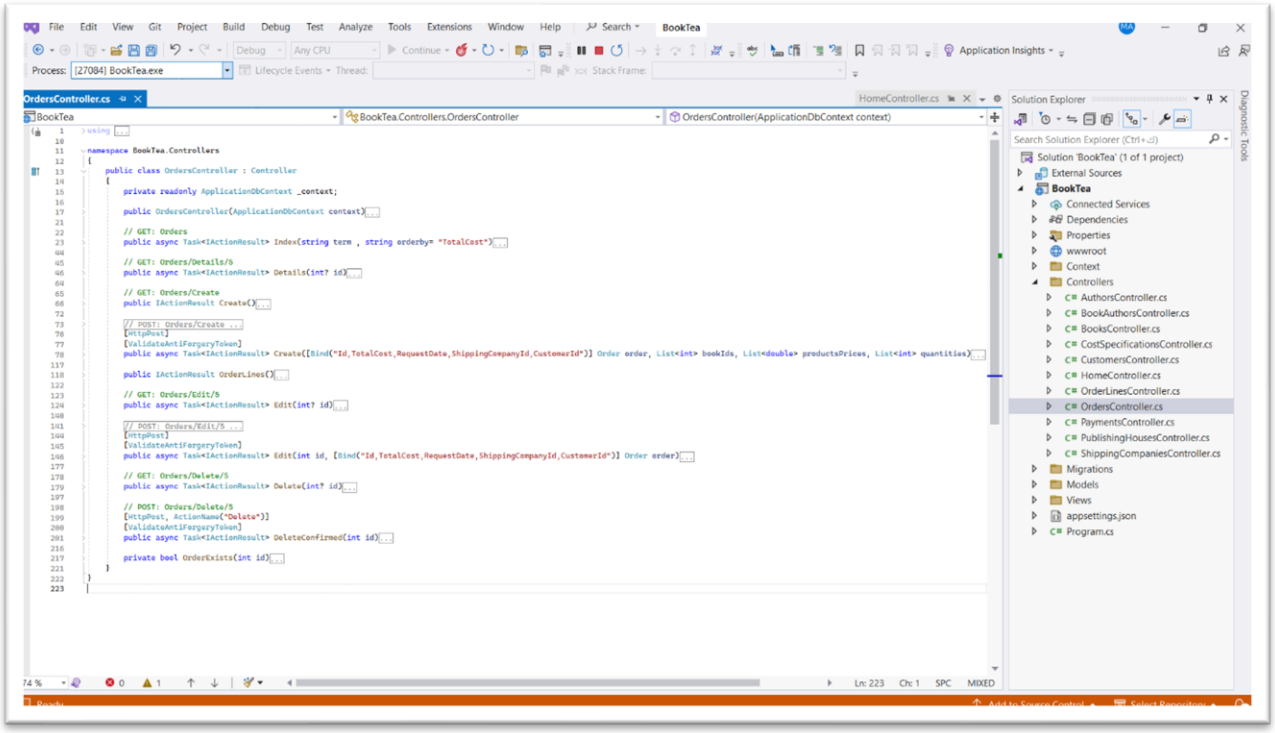
The “OrdersController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.9. Payments

Actions

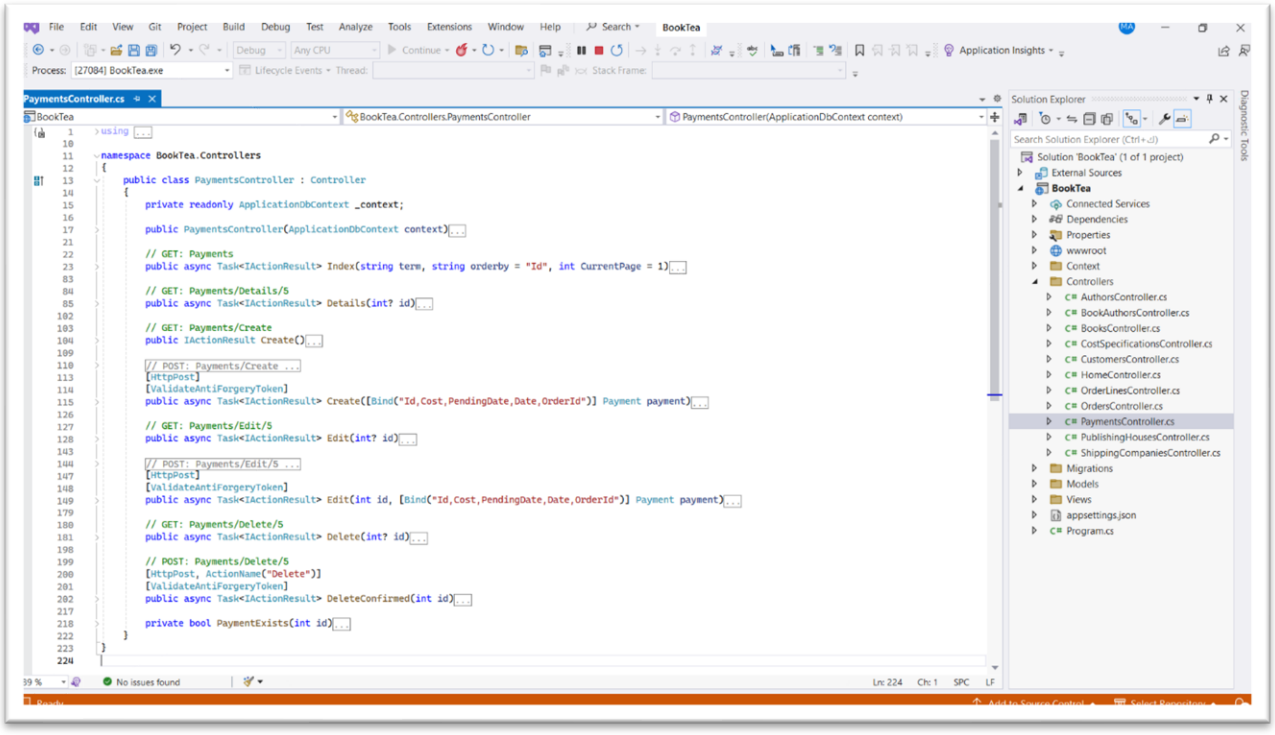
The “PaymentsController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.10. Publishing Houses

Actions

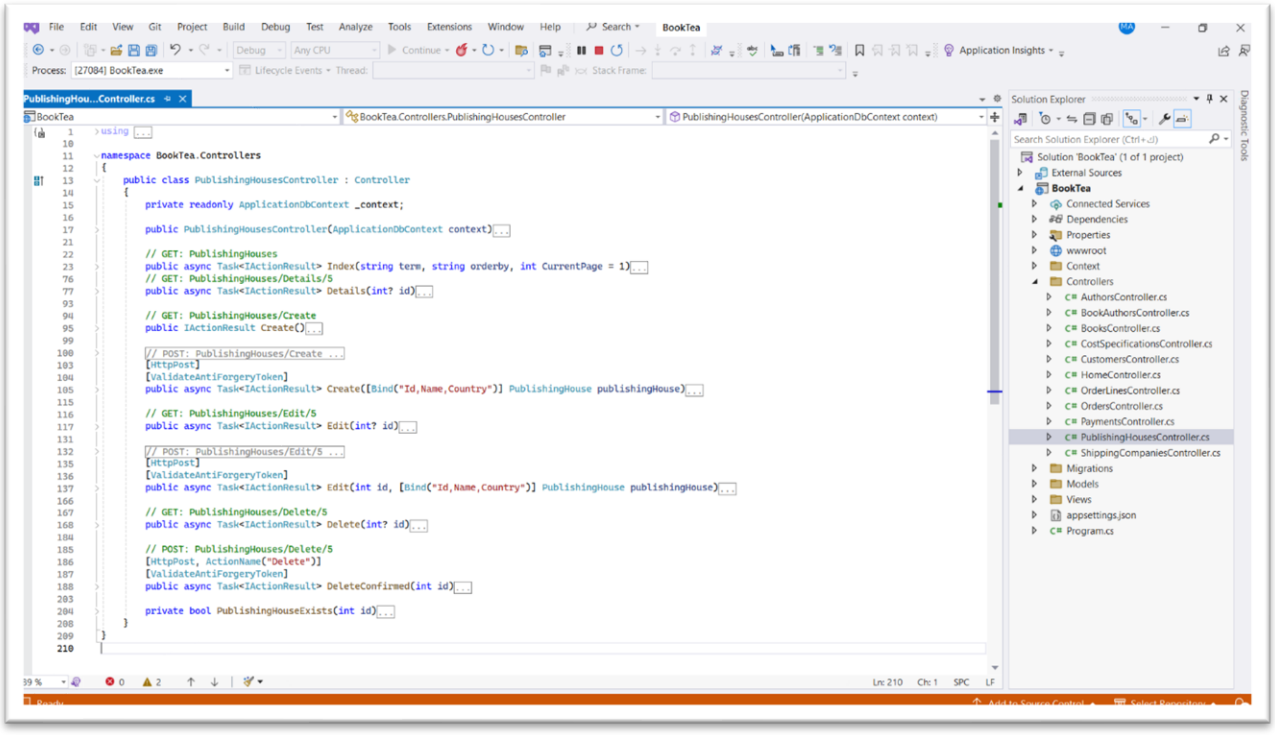
The “PublishingHousesController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



3.11. Shipping Companies

Actions

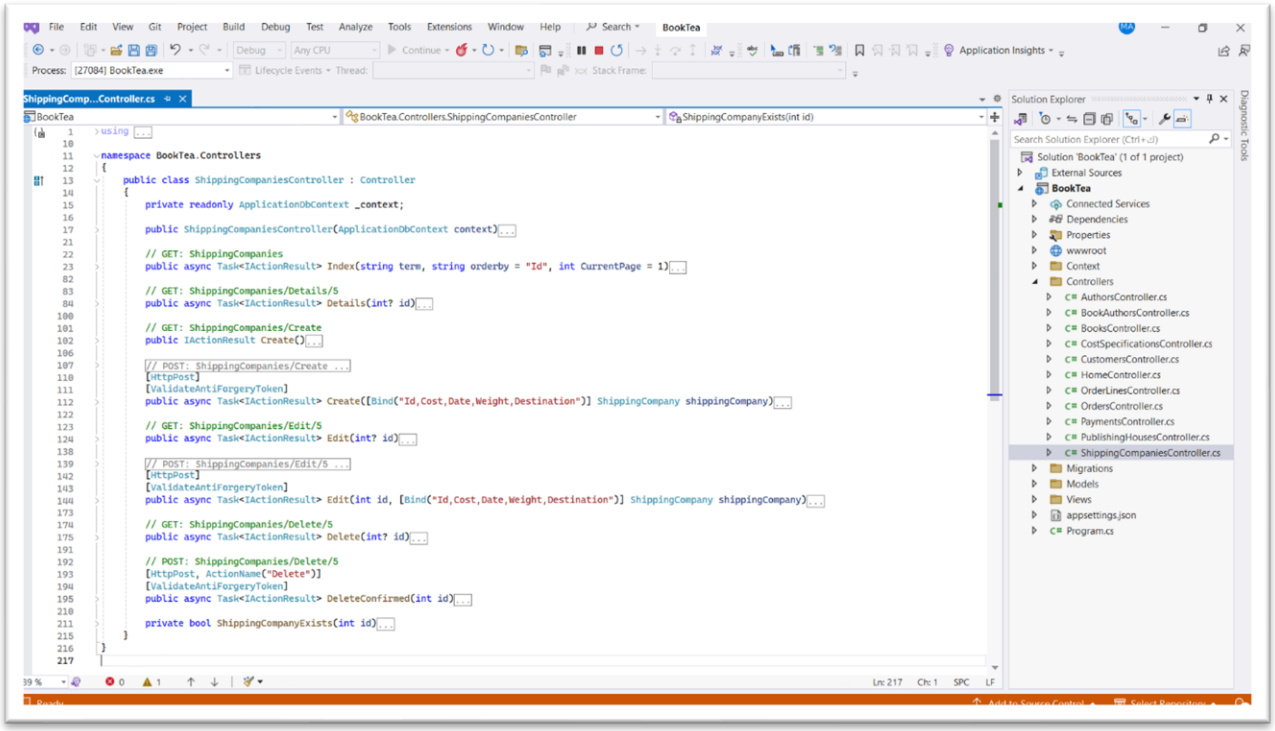
The “ShippingCompaniesController” includes several standard actions that correspond to common CRUD (Create, Read, Update, Delete) operations:

1. Index
2. Details
3. Create
4. Edit
5. Delete

Error Handling

The controller includes basic error handling using HTTP status codes. For example, if an id is null, the controller returns a NotFound status. Additional error handling can be added as needed to manage more complex scenarios.

Implementation in C#



4. Models

4.1. Authors

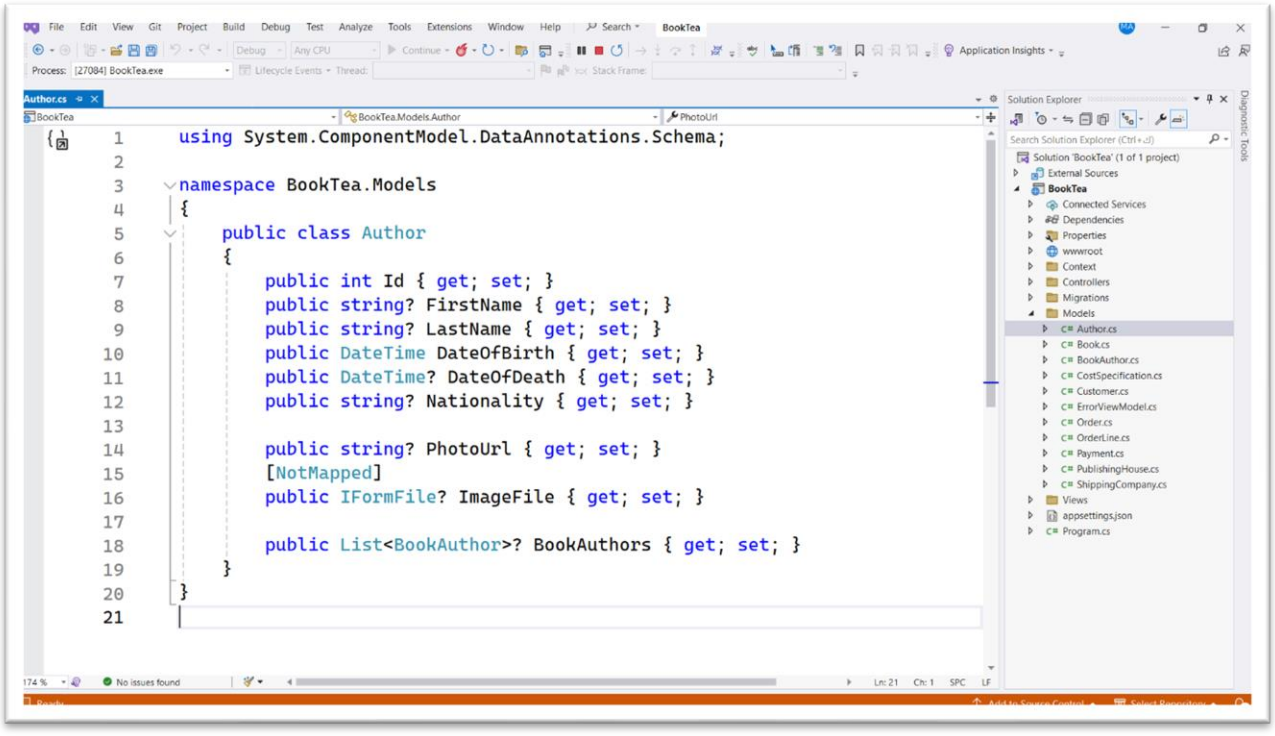
Overview

The Author model represents the data structure for authors within the application and maps to the Authors table in the database. This model is used throughout the application to store and retrieve information about authors, ensuring that the data is managed consistently and efficiently.

Properties

The Author model typically includes several key properties:
Id (int): The unique identifier for the author.
FirstName (string): The first name of the author.
LastName (string): The last name of the author.
DateOfBirth (DateTime): The birth date of the author.
DateOfDeath (DateTime): The death date of the author.
Nationality (string): The nationality of the author.
PhotoURL (string): The photo url of the author.
ImageFile (string): The Image File for the author.
Books (List<BookAuthor>): A collection of books written by the author, establishing a relationship between the Author and BookAuthor models‘a mediate table’.

Implementation in C#



```
using System.ComponentModel.DataAnnotations.Schema;

namespace BookTea.Models
{
    public class Author
    {
        public int Id { get; set; }
        public string? FirstName { get; set; }
        public string? LastName { get; set; }
        public DateTime DateOfBirth { get; set; }
        public DateTime? DateOfDeath { get; set; }
        public string? Nationality { get; set; }

        public string? PhotoUrl { get; set; }
        [NotMapped]
        public IFormFile? ImageFile { get; set; }

        public List<BookAuthor>? BookAuthors { get; set; }
    }
}
```

Relationships

The Author model has a one-to-many relationship with the BookAuthor model, represented by the BooksAuthors property. This relationship is established using Entity Framework's navigation properties

4.2. Book Authors

Overview

The Book Author model represents the data structure for book authors within the application and maps to the BookAuthor table in the database. This model is used throughout the application to store and retrieve information about book author, ensuring that the data is managed consistently and efficiently.

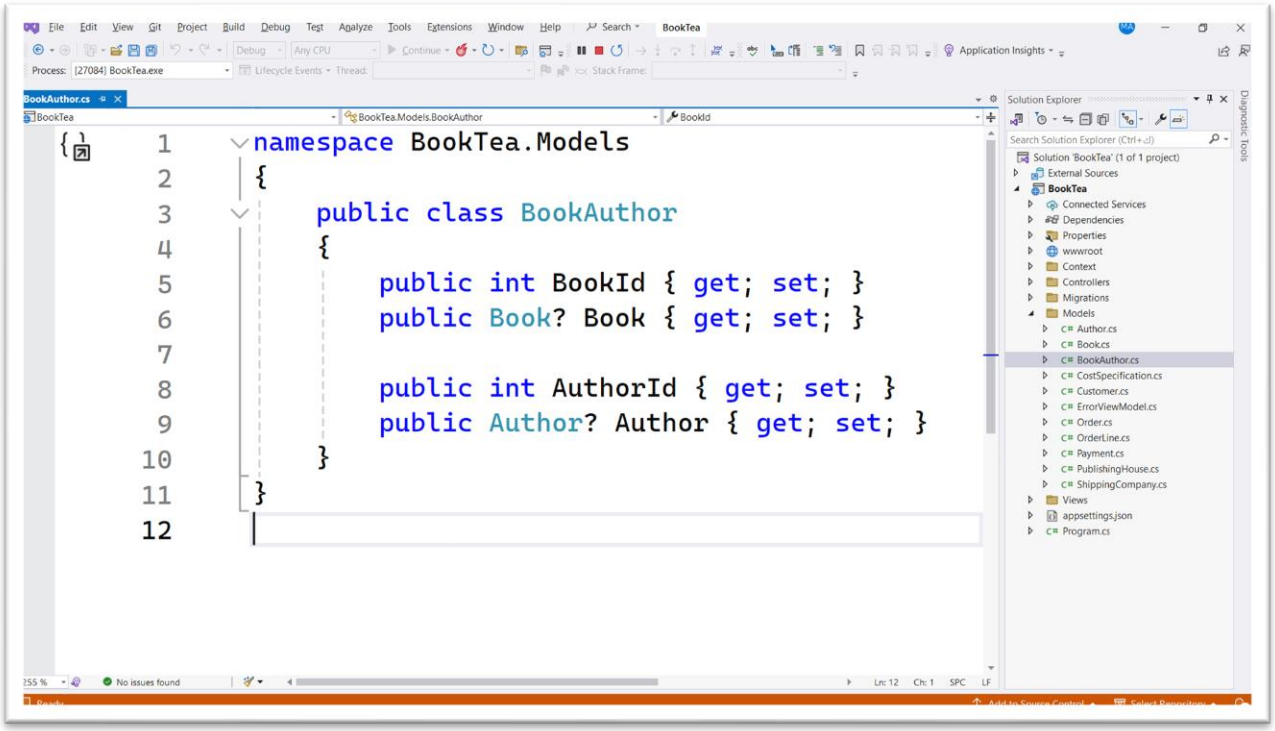
Properties

The BookAuthor model typically includes several key properties:

BookId (int): A foreign key of Book, establishing a relationship between the Book and BookAuthor models.

AuthorId (int): A foreign key of Author, establishing a relationship between the Book and BookAuthor models.

Implementation in C#



Relationships

The BookAuthor model has a one-to-many relationship with:

1. the Book model, represented by the BookId property.
2. the Author model, represented by the AuthorId property.

These relationships are established using Entity Framework's navigation properties.

4.3. Books

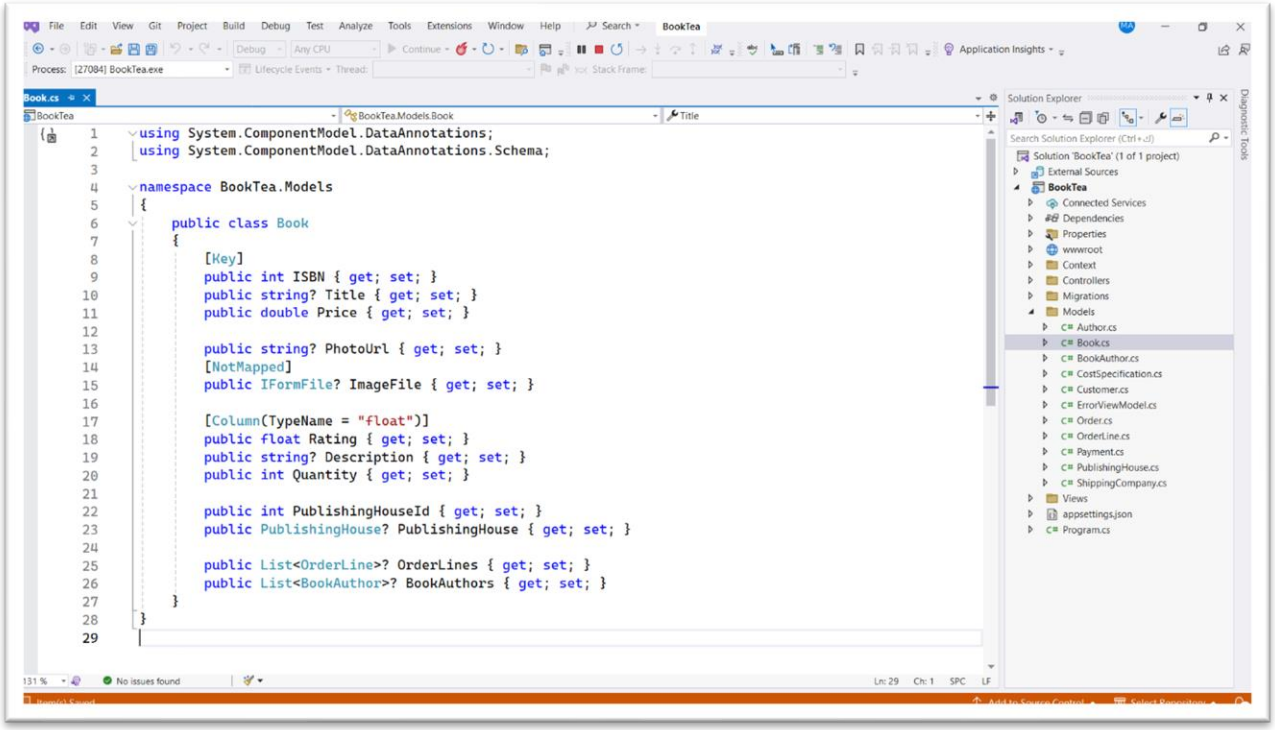
Overview

The Book model represents the data structure for books within the application and maps to the Books table in the database. This model is used throughout the application to store and retrieve information about books, ensuring that the data is managed consistently and efficiently.

Properties

The Book model typically includes several key properties:
ISBN (int): The unique identifier for the author.
Title (string): The title of the book.
Price (double): The price of the book.
PhotoURL (string): The photo url of the book.
ImageFile (string): The Image File for the book.
Rating (float): The rating for the book.
Description (string): The description of the book.
Quantity (int): The quantity of the book.
PublishingHouseId (int): A foreign key of Publishing House, establishing a relationship between the Book and PublishingHouse models.
OrderLines (List<OrderLine>): A collection of Order Lines contains books, establishing a relationship between the Book and OrderLine models.
BookAuthors (List<BookAuthor>): A collection of books written by the author, establishing a relationship between Book and BookAuthor models‘a mediate table’.

Implementation in C#



```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace BookTea.Models
{
    public class Book
    {
        [Key]
        public int ISBN { get; set; }
        public string? Title { get; set; }
        public double Price { get; set; }

        public string? PhotoUrl { get; set; }
        [NotMapped]
        public IFormFile? ImageFile { get; set; }

        [Column(TypeName = "float")]
        public float Rating { get; set; }
        public string? Description { get; set; }
        public int Quantity { get; set; }

        public int PublishingHouseId { get; set; }
        public PublishingHouse? PublishingHouse { get; set; }

        public List<OrderLine>? OrderLines { get; set; }
        public List<BookAuthor>? BookAuthors { get; set; }
    }
}
```

Relationships

The Book model has a one-to-many relationship with:

1. the PublishingHouse model, represented by the PublishingHouseId property.
2. the OrderLine model, represented by the OrderLines property.
3. the BookAuthor model, represented by the BooksAuthors property.

These relationships are established using Entity Framework's navigation properties.

4.4. Cost Specifications

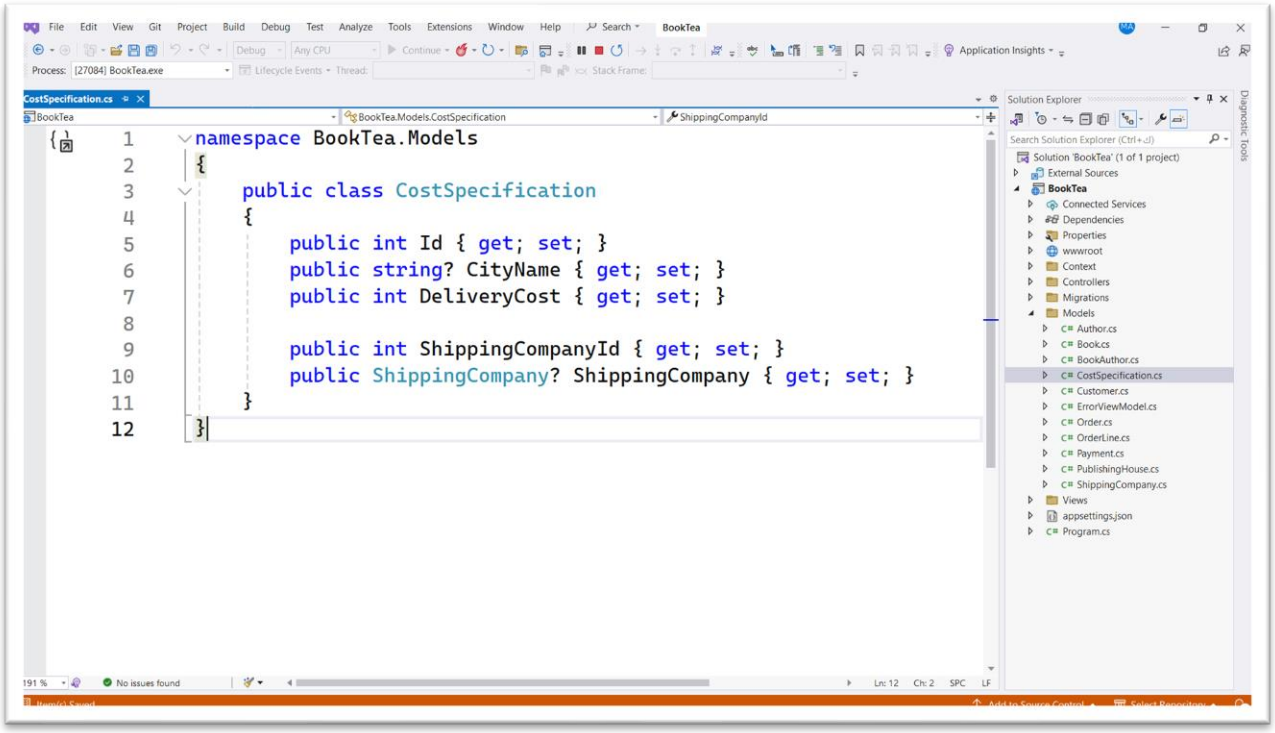
Overview

The Cost Specifications model represents the data structure for cost specifications within the application and maps to the CostSpecifications table in the database. This model is used throughout the application to store and retrieve information about cost specifications, ensuring that the data is managed consistently and efficiently.

Properties

The CostSpecifications model typically includes several key properties:
Id (int): The unique identifier for the cost specifications.
CityName (string): The city name for the cost specifications.
DeliveryCost (int): The delivery cost for the cost specifications.
ShippingCompanyId (int): A foreign key of Shipping Company, establishing a relationship between the CostSpecifications and ShippingCompany models.

Implementation in C#



Relationships

The CostSpecifications model has a one-to-many relationship with:

1. the ShippingCompany model, represented by the ShippingCompanyId property.

These relationships are established using Entity Framework's navigation properties.

4.5. Customers

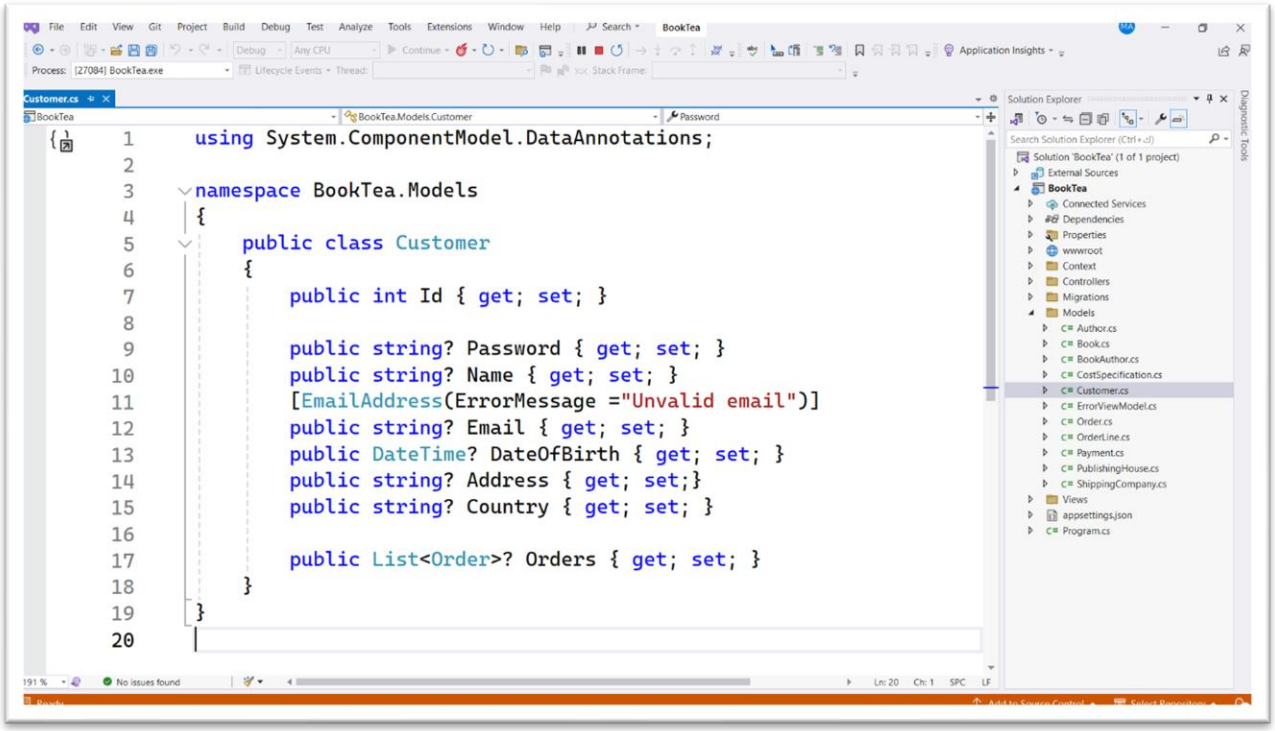
Overview

The Customer model represents the data structure for customers within the application and maps to the Customer table in the database. This model is used throughout the application to store and retrieve information about customer, ensuring that the data is managed consistently and efficiently.

Properties

The Customer model typically includes several key properties:
Id (int): The unique identifier for the customer.
Password (string): The password for the customer.
Name (string): The name of the customer.
Email (string): The email of the customer.
DateOfBirth (DateTime): The date of birth of the customer.
Address (string): The address of the customer.
Country (string): The country of the customer.
Orders (List< Order>): A collection of orders done by the customer, establishing a relationship between the Customer and Order models.

Implementation in C#



Relationships

The Customer model has a one-to-many relationship with:

1. the Order model, represented by the Orders property.

These relationships are established using Entity Framework's navigation properties.

4.6. Order Lines

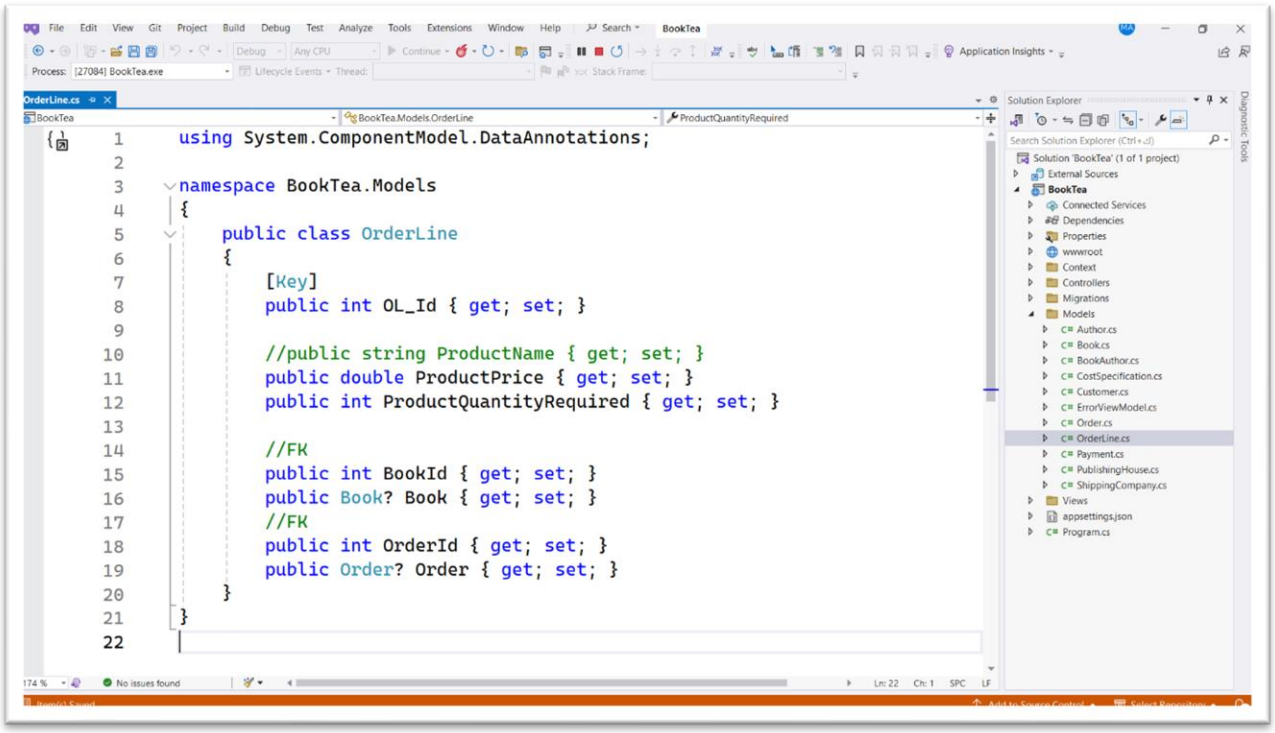
Overview

The OrderLine model represents the data structure for order line within the application and maps to the OrderLine table in the database. This model is used throughout the application to store and retrieve information about order line, ensuring that the data is managed consistently and efficiently.

Properties

The OrderLine model typically includes several key properties:
OL_Id (int): The unique identifier for the order line.
ProductName (string): The product name for the order line.
TotalProductPrice (int): The total product price if there is quantitiy of the order line.
ProductQuantityRequired (int): The product quantity required for the order line.
BookId (int): A foreign key of Book, establishing a relationship between the OrderLine and Book models.
OrderId (int): A foreign key of Order, establishing a relationship between the OrderLine and Order models.

Implementation in C#



```
using System.ComponentModel.DataAnnotations;

namespace BookTea.Models
{
    public class OrderLine
    {
        [Key]
        public int OL_Id { get; set; }

        //public string ProductName { get; set; }
        public double ProductPrice { get; set; }
        public int ProductQuantityRequired { get; set; }

        //FK
        public int BookId { get; set; }
        public Book? Book { get; set; }
        //FK
        public int OrderId { get; set; }
        public Order? Order { get; set; }
    }
}
```

Relationships

The OrderLine model has a one-to-many relationship with:

1. the Book model, represented by the BookId property.
2. the Order model, represented by the OrderId property.

These relationships are established using Entity Framework's navigation properties.

4.7. Orders

Overview

The Order model represents the data structure for order within the application and maps to the Order table in the database. This model is used throughout the application to store and retrieve information about order, ensuring that the data is managed consistently and efficiently.

Properties

The Order model typically includes several key properties:

Id (int): The unique identifier for the order.

TotalCost (int): The total cost for the order.

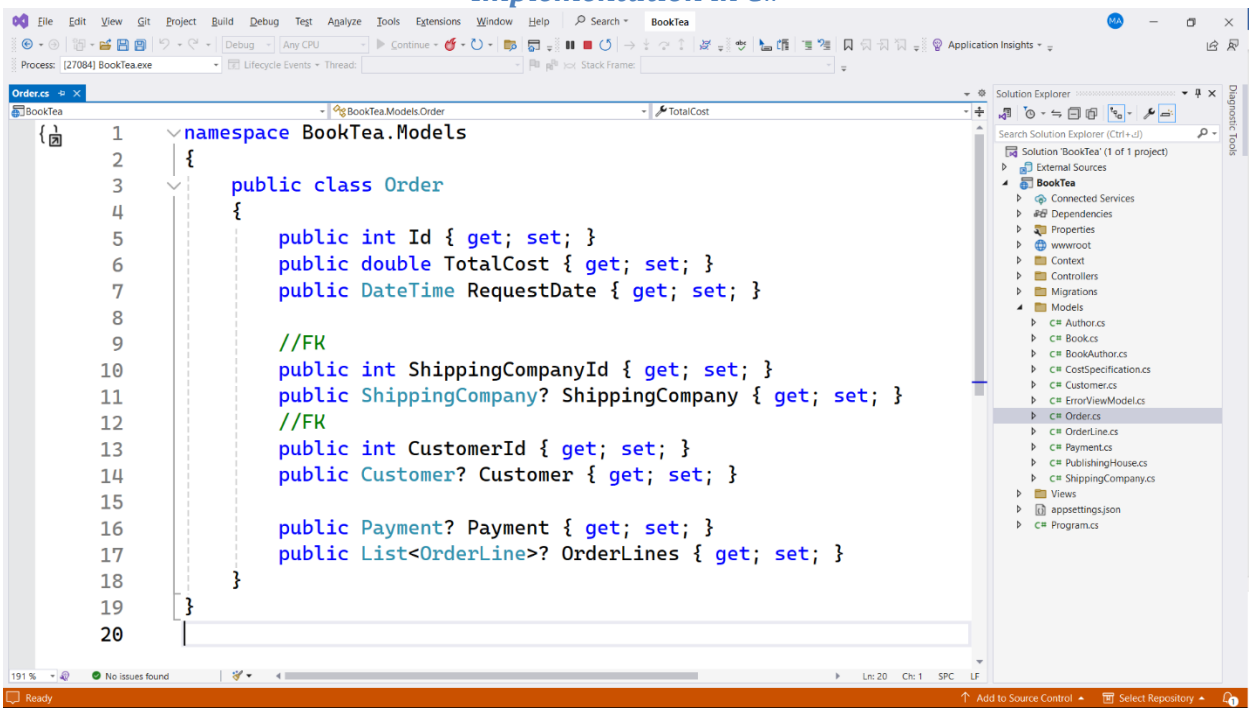
RequestDate (DateTime): The required date to be delivered of the order.

ShippingCompanyId (int): A foreign key of shipping company, establishing a relationship between the Order and ShippingCompany models.

CustomerId (int): A foreign key of customer, establishing a relationship between the Order and Customer models.

OrderLines (List<OrderLine>): A collection of order lines included inside order, establishing a relationship between the Order and OrderLine models.

Implementation in C#



```
namespace BookTea.Models
{
    public class Order
    {
        public int Id { get; set; }
        public double TotalCost { get; set; }
        public DateTime RequestDate { get; set; }

        //FK
        public int ShippingCompanyId { get; set; }
        public ShippingCompany? ShippingCompany { get; set; }
        //FK
        public int CustomerId { get; set; }
        public Customer? Customer { get; set; }

        public Payment? Payment { get; set; }
        public List<OrderLine>? OrderLines { get; set; }
    }
}
```

Relationships

The Order model has a one-to-many relationship with:

1. the ShippingCompany model, represented by the ShippingCompanyId property.
2. the Customer model, represented by the CustomerId property.
3. the OrderLine model, represented by the OrderLines property.

These relationships are established using Entity Framework's navigation properties.

4.8. Payments

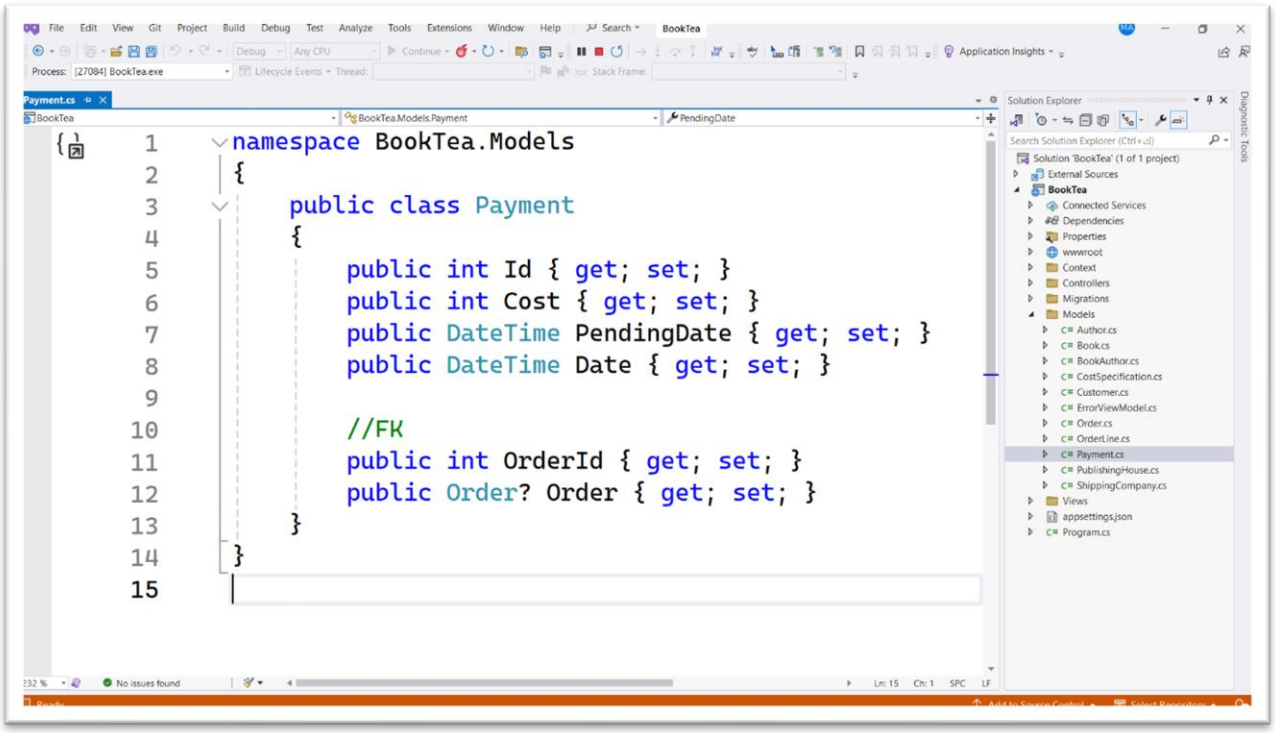
Overview

The Payment model represents the data structure for payment within the application and maps to the payment table in the database. This model is used throughout the application to store and retrieve information about payment, ensuring that the data is managed consistently and efficiently.

Properties

The Payment model typically includes several key properties:
Id (int): The unique identifier for the payment.
Cost (int): The cost for the payment.
PendingDate (DateTime): The pending date the limit time of the payment.
Date (DateTime): The date for pay of the payment.
OrderId (int): A foreign key of order, establishing a relationship between the Payment and Order models.

Implementation in C#



Relationships

The Payment model has a one-to-many relationship with:

1. the Order model, represented by the OrderId property.

These relationships are established using Entity Framework's navigation properties.

4.9. Publishing Houses

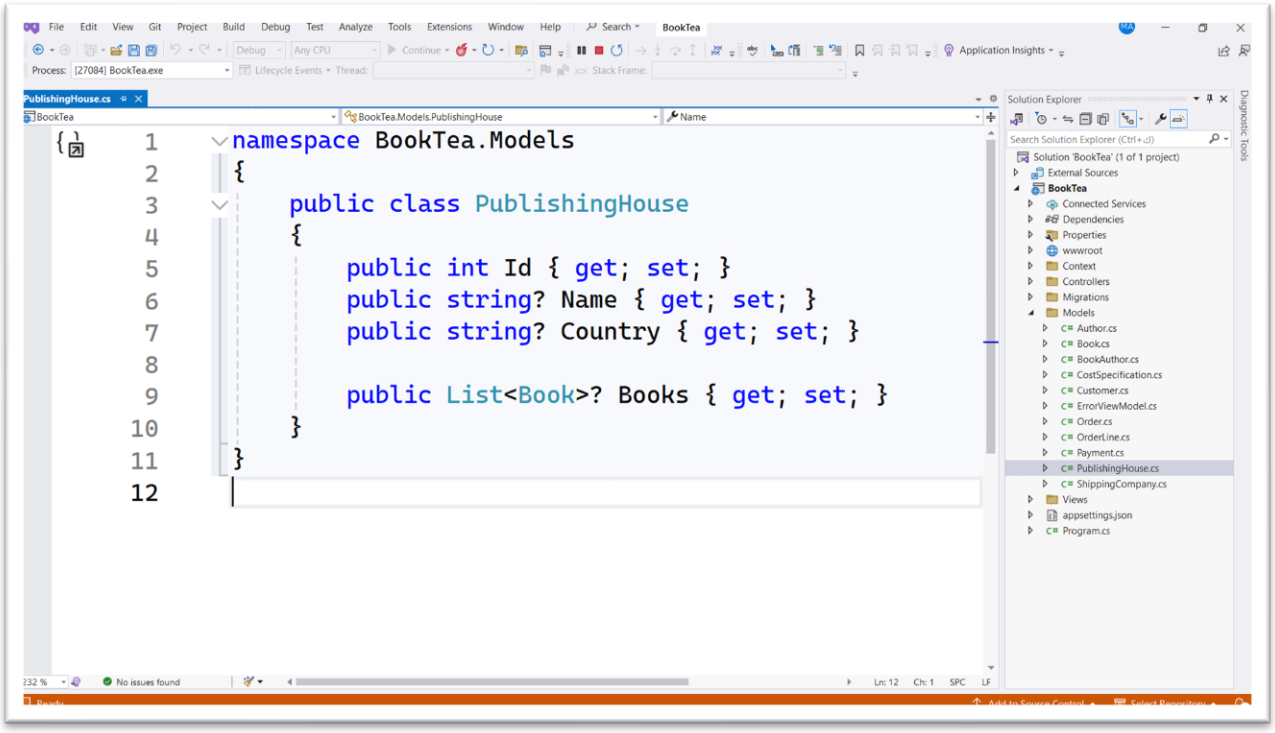
Overview

The PublishingHouse model represents the data structure for publishing house within the application and maps to the PublishingHouse table in the database. This model is used throughout the application to store and retrieve information about publishing house, ensuring that the data is managed consistently and efficiently.

Properties

The PublishingHouse model typically includes several key properties:
Id (int): The unique identifier for the publishing house.
Name (string): The name for the publishing house.
Country (string): The country of the publishing house.
Books (List<Book>): A collection of books published by publishing house, establishing a relationship between the PublishingHouse and Book models.

Implementation in C#



Relationships

The PublishingHouse model has a one-to-many relationship with:

1. the Book model, represented by the Books property.

These relationships are established using Entity Framework's navigation properties.

4.10. Shipping Companies

Overview

The ShippingCompany model represents the data structure for shipping company within the application and maps to the ShippingCompany table in the database. This model is used throughout the application to store and retrieve information about shipping company, ensuring that the data is managed consistently and efficiently.

Properties

The ShippingCompany model typically includes several key properties:

Id (int): The unique identifier for the shipping company.

Cost (int): The all costs due to the city of the shipping company.

Date (DateTime): The date to deliver for the shipping company.

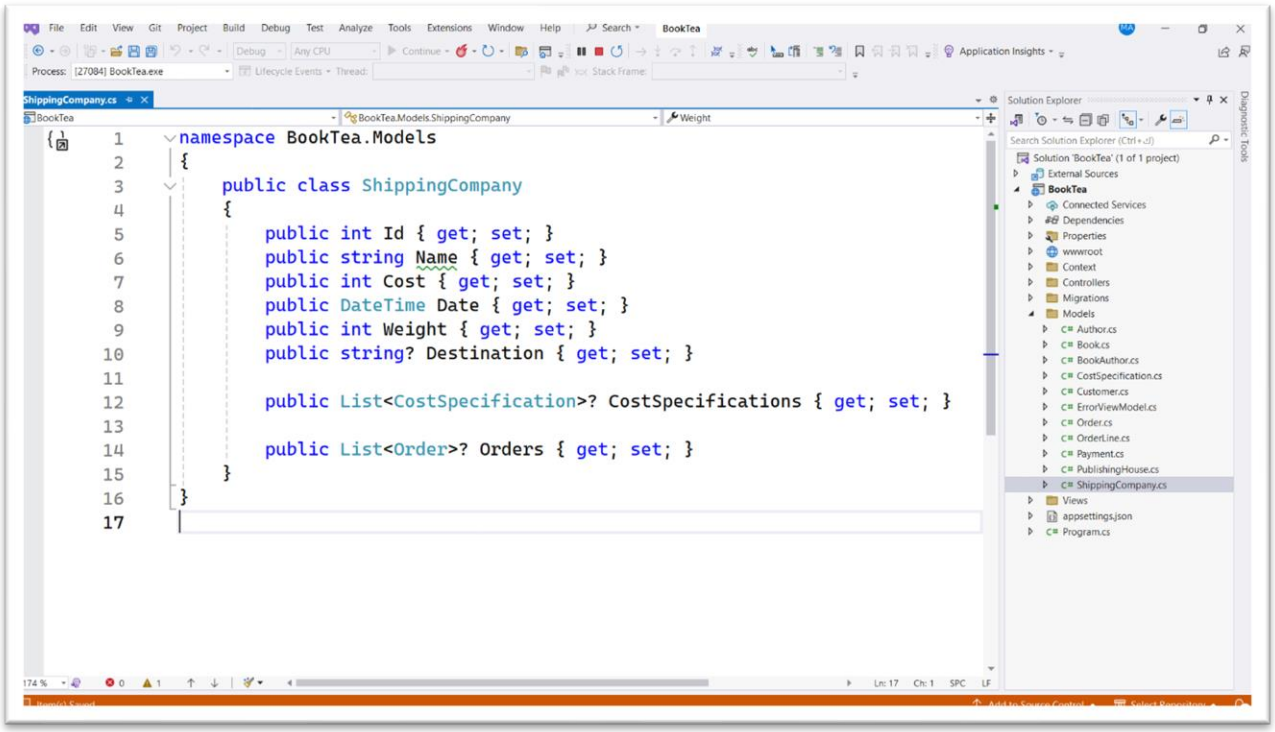
Weight (int): The weight to take for the shipping company.

Destination (string): The destination to go for the shipping company.

CostSpecifications (List<CostSpecification >): A collection of cost specifications that will be cheched from shipping company after deliver the order, establishing a relationship between the ShippingCompany and CostSpecification models.

Orders (List<Order>): A collection of orders that will be deliverd by shipping company, establishing a relationship between the ShippingCompany and Order models.

Implementation in C#



Relationships

The ShippingCompany model has a one-to-many relationship with:

1. the CostSpecification model, represented by the CostSpecifications property.
2. the Order model, represented by the Orders property.

These relationships are established using Entity Framework's navigation properties.

5. Views

5.1. Authors

Implementation in C#

Create

```
@model BookTea.Models.Author

@{
    ViewData["Title"] = "Create";
}

<style>
    .custom-file-button input[type=file]{
        margin-left: -2px !important;
    }
    .custom-file-button input[type=file]::-webkit-file-upload-button{
        display: none;
    }
    .custom-file-button input[type=file]::file-selector-button{
        display: none;
    }
    .custom-file-button:hover label{
        background-color: #dde0e3;
        cursor: pointer;
    }
</style>

<h1>Create</h1>

<h4>Author</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="FirstName" class="control-label"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName" class="control-label"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DateOfBirth" class="control-label"></label>
                <input asp-for="DateOfBirth" class="form-control" />
                <span asp-validation-for="DateOfBirth" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DateOfDeath" class="control-label"></label>
                <input asp-for="DateOfDeath" class="form-control" />
                <span asp-validation-for="DateOfDeath" class="text-danger"></span>
```

```
            </div>
            <div class="form-group">
                <label asp-for="Nationality" class="control-label"></label>
                <input asp-for="Nationality" class="form-control" />
                <span asp-validation-for="Nationality" class="text-danger"></span>
            </div>

            <div class="col-md-12 btn-group pb-2">
                <div class="form-group col-4 p-0">
                    <label asp-for="ImageFile" class="control-label"></label>
                    <img id="PreviewPhoto" src="~/images/images.png" alt="Logo
Image" width="125" height="125" />
                    <span asp-validation-for="PhotoUrl" class="text-danger"></span>
                </div>
                <div class="col-1 p-0">
                </div>
                <div class="form-group col-7 p-0">
                    <div class="input-group custom-file-button mt-5 pt-5">
                        <input asp-for="ImageFile" class="form-control custom-file-
input" id="customFile" />
                        <label class="input-group-text"
for="customFile">Browse</label>
                    </div>
                </div>
            </div>

            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>

        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Delete

```
@model BookTea.Models.Author

@{
    ViewData["Title"] = "Delete";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Author</h4>
    <hr />
```

```
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.FirstName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.FirstName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.LastName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.LastName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DateOfBirth)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DateOfBirth)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DateOfDeath)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DateOfDeath)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Nationality)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Nationality)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PhotoUrl)
        </dt>
        <dd class = "col-sm-10">
            <img src="~/Images/@Model.PhotoUrl" alt="Logo Image" width="125"
height="125" />
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="Id" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

Details

```
@model BookTea.Models.Author

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>
```

```
<div>
    <h4>Author</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.FirstName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.FirstName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.LastName)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.LastName)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DateOfBirth)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DateOfBirth)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.DateOfDeath)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.DateOfDeath)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Nationality)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Nationality)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PhotoUrl)
        </dt>
        <dd class = "col-sm-10">
            <img src="~/Images/@Model.PhotoUrl" alt="Logo Image" width="125"
height="125" />
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.Id">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

Edit

```
@model BookTea.Models.Author

@{
    ViewData["Title"] = "Edit";
}

<style>
```

```
    .custom-file-button input[type=file] {
        margin-left: -2px !important;
    }

        .custom-file-button input[type=file]::-webkit-file-upload-button {
            display: none;
        }

        .custom-file-button input[type=file]::file-selector-button {
            display: none;
        }

    .custom-file-button:hover label {
        background-color: #dde0e3;
        cursor: pointer;
    }
</style>

<h1>Edit</h1>

<h4>Author</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="Id" />
            <div class="form-group">
                <label asp-for="FirstName" class="control-label"></label>
                <input asp-for="FirstName" class="form-control" />
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="LastName" class="control-label"></label>
                <input asp-for="LastName" class="form-control" />
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DateOfBirth" class="control-label"></label>
                <input asp-for="DateOfBirth" class="form-control" />
                <span asp-validation-for="DateOfBirth" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="DateOfDeath" class="control-label"></label>
                <input asp-for="DateOfDeath" class="form-control" />
                <span asp-validation-for="DateOfDeath" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Nationality" class="control-label"></label>
                <input asp-for="Nationality" class="form-control" />
                <span asp-validation-for="Nationality" class="text-danger"></span>
            </div>
            <div class="col-md-12 btn-group pb-2">
                <div class="form-group col-4 p-0">
                    <label asp-for="ImageFile" class="control-label"></label>
                    <img id="PreviewPhoto" src="~/Images/@Model.PhotoUrl" alt="Logo
Image" width="125" height="125"/>
                    <span asp-validation-for="PhotoUrl" class="text-danger"></span>
```

```
                </div>
                <div class="col-1 p-0">
                </div>
                <div class="form-group col-7 p-0">
                    <div class="input-group custom-file-button mt-5 pt-5">
                        <input asp-for="ImageFile" class="form-control custom-file-
input" id="customFile" />
                        <label class="input-group-text"
for="customFile">Browse</label>
                    </div>
                </div>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index

```
@model IEnumerable<BookTea.Models.Author>

@{
    ViewData["Title"] = "Index";
}

@if (Model == null || Model.Count() == 0)
{
    <div class="alert alert-warning" role="alert">
        <h4 class="alert-heading">Oh no!</h4>
        <p>Sorry! There is no authors yet!</p>
        <hr>
        <a asp-action="Create">Create Authors</a>
    </div>
}
else{
<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
    <form asp-action="Index" method="get">
        <input style="width:400px" name="term" class="d-inline form-control"
placeholder="Search...." type="text" />
        <button type="submit" class="d-inline btn btn-outline-
primary">Search</button>
    </form>
</p>
<table class="table">
```

```
    <thead>
        <tr>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderFirstName">@Html.DisplayNameFor(model =>
model.FirstName)</a>
            </th>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderLastName">@Html.DisplayNameFor(model =>
model.LastName)</a>
            </th>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderDateOfBirth">@Html.DisplayNameFor(model =>
model.DateOfBirth)</a>
            </th>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderDateOfDeath">@Html.DisplayNameFor(model =>
model.DateOfDeath)</a>
            </th>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderNationality">@Html.DisplayNameFor(model =>
model.Nationality)</a>
            </th>
            <th>
                @Html.DisplayNameFor(model => model.PhotoUrl)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.FirstName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LastName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateOfBirth)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.DateOfDeath)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Nationality)
            </td>
            <td>
                <img src="~/Images/@item.PhotoUrl" width="60px" height="60px" />
            </td>
            <td>
                <a asp-action="Edit" asp-route-id="@item.Id">Edit</a> |
                <a asp-action="Details" asp-route-id="@item.Id">Details</a> |
                <a asp-action="Delete" asp-route-id="@item.Id">Delete</a>
```

```
            </td>
        </tr>
}
    </tbody>
</table>

    <div class="d-flex justify-content-end">
        <nav aria-label="Page navigation example">
            <ul class="pagination">
                <li class="page-item @(ViewBag.CurrentPage==1?"disabled":"")"><a
class="page-link asp-action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage-1)"
asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">Previous</a></li>
                @for (int i = 1; i <= ViewBag.NumPages; i++)
                {
                    <li class="page-item @(i==ViewBag.CurrentPage?"active":"")">
                        <a class="page-link" asp-action="Index" asp-route-
CurrentPage="@i" asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">@i</a>
                    </li>
                }
                <li class="page-item
@(ViewBag.CurrentPage==ViewBag.NumPages?"disabled":"")"><a class="page-link" asp-
action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage+1)" asp-route-
term="@ViewBag.term" asp-route-orderby="@ViewBag.orderby">Next</a></li>
            </ul>
        </nav>
    </div>
}
```

5.2. Book Authors

Implementation in C#

Create

```
@model BookTea.Models.BookAuthor

@{
    ViewData["Title"] = "Create";
}

<h1>Create</h1>

<h4>BookAuthor</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="BookId" class="control-label"></label>
                <select asp-for="BookId" class="form-control" asp-
items="ViewBag.BookId"><option>-----Select Book-----</option></select>
            </div>
            <div class="form-group">
                <label asp-for="AuthorId" class="control-label"></label>
                <select asp-for="AuthorId" class ="form-control" asp-
items="ViewBag.AuthorId"><option>-----Select Author-----</option></select>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Delete

```
@model BookTea.Models.BookAuthor

@{
    ViewData["Title"] = "Delete";
}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
```

```
<div>
    <h4>BookAuthor</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Book)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Book.Title)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Author)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Author.FirstName) @Html.DisplayFor(model
=> model.Author.LastName)
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="BookId" />
        <input type="hidden" asp-for="AuthorId" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

Details

```
@model BookTea.Models.BookAuthor

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>BookAuthor</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Book)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Book.Title)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Author)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Author.FirstName) @Html.DisplayFor(model
=> model.Author.LastName)
        </dd>
    </dl>
</div>
<div>
```

```
    @* @Html.ActionLink("Edit", "Edit", new { /* id = Model.PrimaryKey */ }) | *@
    <a asp-action="Index">Back to List</a>
</div>
```

Index

```
@model IEnumerable<BookTea.Models.BookAuthor>

@{
    ViewData["Title"] = "Index";
}

@if (Model == null || Model.Count() == 0)
{
    <div class="alert alert-warning" role="alert">
        <h4 class="alert-heading">Empty!!!</h4>
        <hr>
        <a asp-action="Create">Create New</a>
    </div>
}
else{
<h1>Index</h1>

    <form asp-action="Index" method="get">
        <input style="width:400px" name="term" class="d-inline form-control"
placeholder="Search...." type="text" />
        <button type="submit" class="d-inline btn btn-outline-
primary">Search</button>
    </form>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderBook">@Html.DisplayNameFor(model => model.Book)</a>
            </th>
            <th>
                <a class="text-reset text-decoration-none" asp-action="Index" asp-
route-orderby="@ViewBag.OrderAuthor">@Html.DisplayNameFor(model => model.Author)</a>
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.Book.Title)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Author.FirstName)
@Html.DisplayFor(modelItem => item.Author.LastName)
            </td>
```

```
            <td>
                        @Html.ActionLink("Details", "Details", new { BookId =
item.BookId, AuthorId = item.AuthorId }) |
                        @Html.ActionLink("Delete", "Delete", new { BookId =
item.BookId, AuthorId = item.AuthorId })
            </td>
        </tr>
}
    </tbody>
</table>

    <div class="d-flex justify-content-end">
        <nav aria-label="Page navigation example">
            <ul class="pagination">
                <li class="page-item @(ViewBag.CurrentPage==1?"disabled":"")"><a
class="page-link asp-action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage-1)"
asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">Previous</a></li>
                @for (int i = 1; i <= ViewBag.NumPages; i++)
                {
                    <li class="page-item @(i==ViewBag.CurrentPage?"active":"")">
                        <a class="page-link" asp-action="Index" asp-route-
CurrentPage="@i" asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">@i</a>
                    </li>
                }
                <li class="page-item
@(ViewBag.CurrentPage==ViewBag.NumPages?"disabled":"")"><a class="page-link" asp-
action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage+1)" asp-route-
term="@ViewBag.term" asp-route-orderby="@ViewBag.orderby">Next</a></li>
            </ul>
        </nav>
    </div>
}
```

5.3. Books

Implementation in C#

Create

```
@model BookTea.Models.Book

@{
    ViewData["Title"] = "Create";
}

<style>
    .custom-file-button input[type=file] {
        margin-left: -2px !important;
    }

        .custom-file-button input[type=file]::-webkit-file-upload-button {
            display: none;
        }

        .custom-file-button input[type=file]::file-selector-button {
            display: none;
        }

    .custom-file-button:hover label {
        background-color: #dde0e3;
        cursor: pointer;
    }
</style>

<h1>Create</h1>

<h4>Book</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Rating" class="control-label"></label>
                <input asp-for="Rating" class="form-control" />
                <span asp-validation-for="Rating" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
```

```
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PublishingHouseId" class="control-label"></label>
                <select asp-for="PublishingHouseId" class="form-control" asp-
items="ViewBag.PublishingHouse"><option>------Select Publishing house-----
</option></select>

            </div>
            <div class="col-md-12 btn-group pb-2">
                <div class="form-group col-4 p-0">
                    <label asp-for="ImageFile" class="control-label"></label>
                    <img id="PreviewPhoto" src="~/images/images.png" alt="Logo
Image" width="125" height="125" />
                    <span asp-validation-for="PhotoUrl" class="text-danger"></span>
                </div>
                <div class="col-1 p-0">
                </div>
                <div class="form-group col-7 p-0">
                    <div class="input-group custom-file-button mt-5 pt-5">
                        <input asp-for="ImageFile" class="form-control custom-file-
input" id="customFile" />
                        <label class="input-group-text"
for="customFile">Browse</label>
                    </div>
                </div>
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Delete

```
@model BookTea.Models.Book

@{
    ViewData["Title"] = "Delete";
}
```

```
<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>
<div>
    <h4>Book</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Title)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Title)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Rating)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Rating)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Description)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Quantity)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Quantity)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PublishingHouse)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.PublishingHouse.Id)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.PhotoUrl)
        </dt>
        <dd class="col-sm-10">
            <img src="~/Images/@Model.PhotoUrl" alt="Logo Image" width="125"
height="125" />
        </dd>
    </dl>

    <form asp-action="Delete">
        <input type="hidden" asp-for="ISBN" />
        <input type="submit" value="Delete" class="btn btn-danger" /> |
        <a asp-action="Index">Back to List</a>
    </form>
</div>
```

Details

```
@model BookTea.Models.Book

@{
    ViewData["Title"] = "Details";
}

<h1>Details</h1>

<div>
    <h4>Book</h4>
    <hr />
    <dl class="row">
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Title)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Title)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Price)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Price)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Rating)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Rating)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Description)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Description)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.Quantity)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.Quantity)
        </dd>
        <dt class = "col-sm-2">
            @Html.DisplayNameFor(model => model.PublishingHouse)
        </dt>
        <dd class = "col-sm-10">
            @Html.DisplayFor(model => model.PublishingHouse.Id)
        </dd>
        <dt class="col-sm-2">
            @Html.DisplayNameFor(model => model.PhotoUrl)
        </dt>
        <dd class="col-sm-10">
            <img src="~/Images/@Model.PhotoUrl" alt="Logo Image" width="125"
height="125" />
```

```
        </dd>
    </dl>
</div>
<div>
    <a asp-action="Edit" asp-route-id="@Model?.ISBN">Edit</a> |
    <a asp-action="Index">Back to List</a>
</div>
```

Edit

```
@model BookTea.Models.Book

@{
    ViewData["Title"] = "Edit";
}
<style>
    .custom-file-button input[type=file] {
        margin-left: -2px !important;
    }

        .custom-file-button input[type=file]::-webkit-file-upload-button {
            display: none;
        }

        .custom-file-button input[type=file]::file-selector-button {
            display: none;
        }

    .custom-file-button:hover label {
        background-color: #dde0e3;
        cursor: pointer;
    }
</style>

<h1>Edit</h1>

<h4>Book</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Edit" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <input type="hidden" asp-for="ISBN" />
            <div class="form-group">
                <label asp-for="Title" class="control-label"></label>
                <input asp-for="Title" class="form-control" />
                <span asp-validation-for="Title" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Price" class="control-label"></label>
                <input asp-for="Price" class="form-control" />
                <span asp-validation-for="Price" class="text-danger"></span>
            </div>
            <div class="form-group">
```

```
                <label asp-for="Rating" class="control-label"></label>
                <input asp-for="Rating" class="form-control" />
                <span asp-validation-for="Rating" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Description" class="control-label"></label>
                <input asp-for="Description" class="form-control" />
                <span asp-validation-for="Description" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="Quantity" class="control-label"></label>
                <input asp-for="Quantity" class="form-control" />
                <span asp-validation-for="Quantity" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label asp-for="PublishingHouseId" class="control-label"></label>
                <select asp-for="PublishingHouseId" class="form-control" asp-
items="ViewBag.PublishingHouseId"></select>
                <span asp-validation-for="PublishingHouseId" class="text-
danger"></span>
            </div>
            <div class="col-md-12 btn-group pb-2">
                <div class="form-group col-4 p-0">
                    <label asp-for="ImageFile" class="control-label"></label>
                    <img id="PreviewPhoto" src="~/Images/@Model.PhotoUrl" alt="Logo
Image" width="125" height="125" />
                    <span asp-validation-for="PhotoUrl" class="text-danger"></span>
                </div>
                <div class="col-1 p-0">
                </div>
                <div class="form-group col-7 p-0">
                    <div class="input-group custom-file-button mt-5 pt-5">
                        <input asp-for="ImageFile" class="form-control custom-file-
input" id="customFile" />
                        <label class="input-group-text"
for="customFile">Browse</label>
                    </div>
                </div>
            </div>
            <div class="form-group">
                <input type="submit" value="Save" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-action="Index">Back to List</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

Index

```
@model IEnumerable<BookTea.Models.Book>

@{
    ViewData["Title"] = "Index";
}

@if (Model == null || !Model.Any())
{
    <div class="alert alert-warning" role="alert">
        <h4 class="alert-heading">Oh no!!!</h4>
        <p>There are no books yet!</p>
        <hr>
        <a asp-action="Create">Create Books</a>
    </div>
}
else
{
    <h1>Index</h1>

    <p>
        <a asp-action="Create">Create New</a>
        <form asp-action="Index" method="get" class="form-inline">
            <input style="width:400px" name="term" class="form-control mr-sm-2"
placeholder="Search...." type="text" />
            <button type="submit" class="btn btn-outline-primary my-2 my-sm-
0">Search</button>
        </form>
    </p>

    <table class="table">
        <thead>
            <tr>
                <th>
                    <a class="text-reset text-decoration-none" asp-action="Index"
asp-route-orderby="@ViewBag.OrderTitle">@Html.DisplayNameFor(model =>
model.First().Title)</a>
                </th>
                <th>
                    <a class="text-reset text-decoration-none" asp-action="Index"
asp-route-orderby="@ViewBag.OrderPrice">@Html.DisplayNameFor(model =>
model.First().Price)</a>
                </th>
                <th>
                    @Html.DisplayNameFor(model => model.First().PhotoUrl)
                </th>
                <th>
                    <a class="text-reset text-decoration-none" asp-action="Index"
asp-route-orderby="@ViewBag.OrderRating">@Html.DisplayNameFor(model =>
model.First().Rating)</a>
                </th>
                <th>
                    @Html.DisplayNameFor(model => model.First().Description)
                </th>
                <th>
                    <a class="text-reset text-decoration-none" asp-action="Index"
asp-route-orderby="@ViewBag.OrderQuantity">@Html.DisplayNameFor(model =>
```

```
model.First().Quantity)</a>
                </th>
                <th>
                    <a class="text-reset text-decoration-none" asp-action="Index"
asp-route-orderby="@ViewBag.OrderPublishingHouse">@Html.DisplayNameFor(model =>
model.First().PublishingHouse.Name)</a>
                </th>
                <th></th>
            </tr>
        </thead>
        <tbody>
            @foreach (var item in Model)
            {
                <tr>
                    <td>@Html.DisplayFor(modelItem => item.Title)</td>
                    <td>@Html.DisplayFor(modelItem => item.Price)</td>
                    <td><img src="~/Images/@item.PhotoUrl" width="60px"
height="60px" /></td>
                    <td>@Html.DisplayFor(modelItem => item.Rating)</td>
                    <td>@Html.DisplayFor(modelItem => item.Description)</td>
                    <td>@Html.DisplayFor(modelItem => item.Quantity)</td>
                    <td>@Html.DisplayFor(modelItem =>
item.PublishingHouse.Name)</td>
                    <td>
                        <a asp-action="Edit" asp-route-id="@item.ISBN">Edit</a> |
                        <a asp-action="Details" asp-route-
id="@item.ISBN">Details</a> |
                        <a asp-action="Delete" asp-route-id="@item.ISBN">Delete</a>
                    </td>
                </tr>
            }
        </tbody>
    </table>

    <div class="d-flex justify-content-end">
        <nav aria-label="Page navigation example">
            <ul class="pagination">
                <li class="page-item @(ViewBag.CurrentPage==1?"disabled":"")"><a
class="page-link asp-action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage-1)"
asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">Previous</a></li>
                @for (int i = 1; i <= ViewBag.NumPages; i++)
                {
                    <li class="page-item @(i==ViewBag.CurrentPage?"active":"")">
                        <a class="page-link" asp-action="Index" asp-route-
CurrentPage="@i" asp-route-term="@ViewBag.term" asp-route-
orderby="@ViewBag.orderby">@i</a>
                    </li>
                }
                <li class="page-item
@(ViewBag.CurrentPage==ViewBag.NumPages?"disabled":"")"><a class="page-link" asp-
action="Index" asp-route-CurrentPage="@(ViewBag.CurrentPage+1)" asp-route-
term="@ViewBag.term" asp-route-orderby="@ViewBag.orderby">Next</a></li>
            </ul>
        </nav>
    </div>
}
```