**Building Book Management Application Using OData**

# Introduction

Imagine you're a librarian of an university, your leader has asked you to develop an application for book management especially support the querying data in many options. The book information includes *id, ISBN, title, author, price, press information and address information*. The press information includes *press id, press name*, the press information also has category information such as *book, ebook, magazine*. The address information will have *city and country*. The application has to support adding, viewing, modifying, and removing books - a standardized usage action verbs better known as Create, Read, Update, Delete (CRUD).

OData is an open protocol for operating data over HTTP. It also follows REST architecture. Currently OData is widely used for exposing data, so this is the suitable method to implement Book Management application. This lab explores creating an application using OData to create service, and ASP.NET Core Web Application with Model-View-Controller. An **In-Memory** **Database** will be created to persist the book data that will be used for reading and managing book data by **Entity Framework Core.**

# Lab Objectives

In this lab, you will:

* Use the Visual Studio.NET to create OData Service using ASP.NET Core Web Web API Project.
* Develop Web application using MVC Pattern.
* Use Entity Framework Core to create a In-Memory database
* Develop Entity classes, DBContext class, DataSource class to perform CRUD actions using Entity Framework Core
* Run the project and test the services using Postman.
* Run the project and test the application actions.

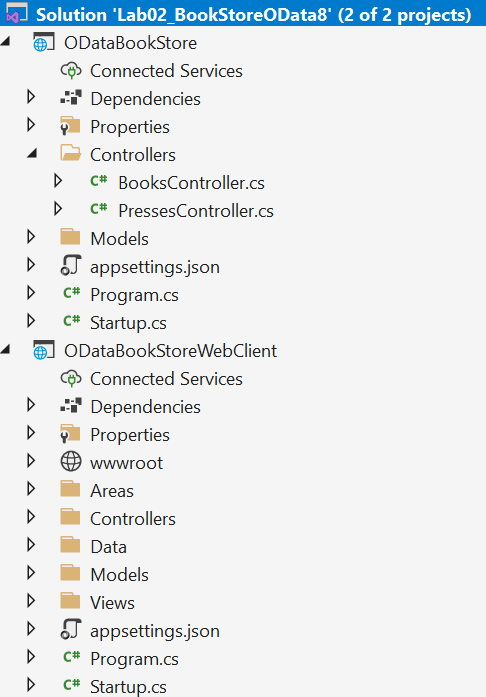
# Guidelines

# Activity 01: Create a Blank Solution

**Step 01**. Create a Solution named **Lab01\_BookStoreOData8**.

**Step 05**. Create ASP.NET Core Web Web API Project for OData Service.

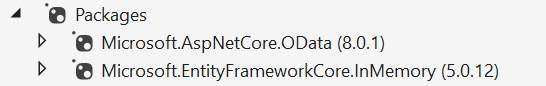
**Step 06**. Create ASP.NET Core Web Application (Model-View-Controller) Project.



# Activity 02: Creating an OData Service

**Step 01**. Create a skeleton of the ASP.NET Core OData service using ASP.NET Core Web API.

**Step 02**. Install the following packages from NuGet:

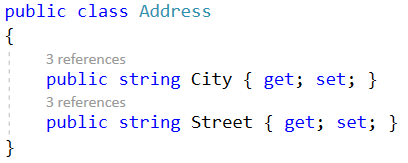


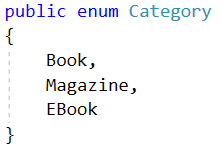
*dotnet package add Microsoft.AspNetCore.OData --version 8.0.1*

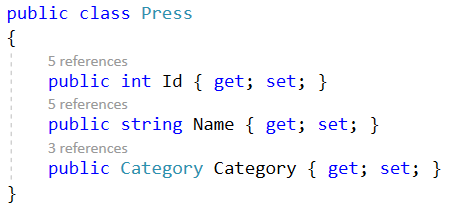
*dotnet package add Microsoft.EntityFrameworkCore.InMemory --version 5.0.12* (for simplicity, using the version with the In-Memory data source, in the real application should use SqlServer)

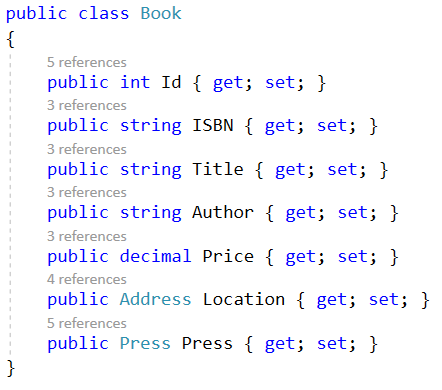
**Step 03**. Add the model classes for building the Entity Data Model (EDM)

This steps will create 1 enum data named Category and 3 classes named Address, Press and Book.



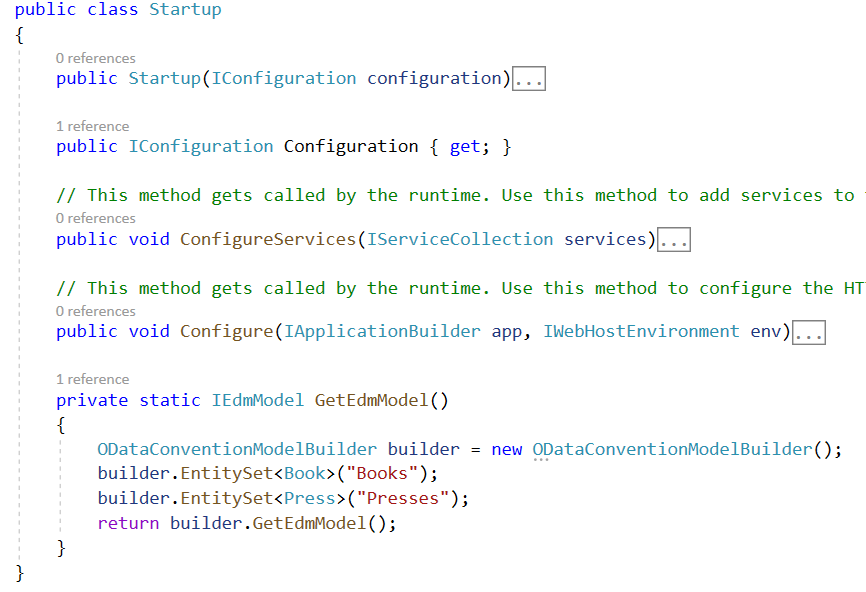






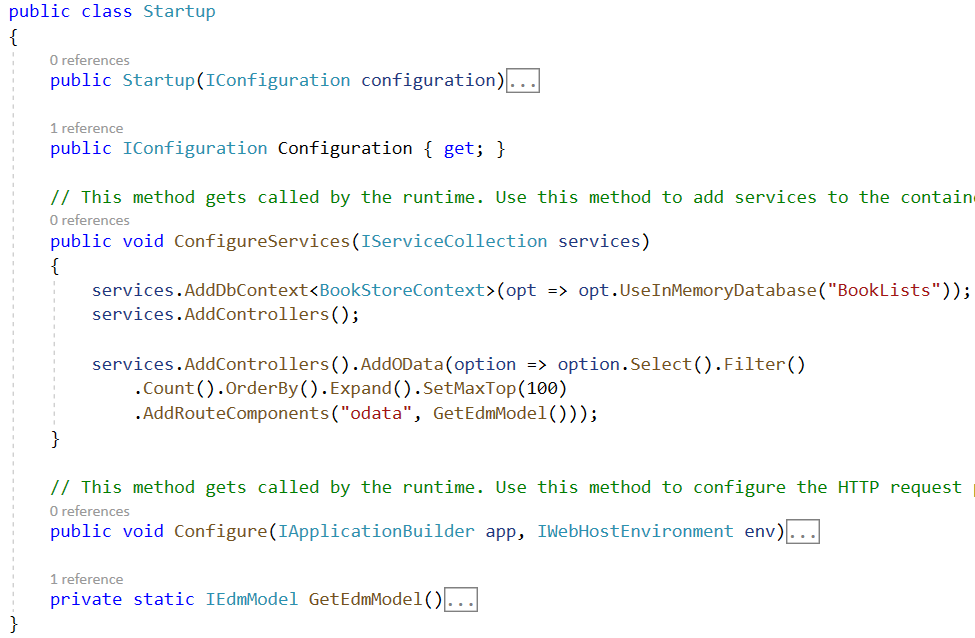
**Step 04**. Build the Entity Data Model

OData uses the *Entity Data Model (EDM)* to describe the structure of data. To build the EDM add a method in *Startup.cs* class.



**Step 05**. Register the OData Services through Dependency Injection

Register the OData Services in *ConfigureServices()* of Startup.cs



Register the OData Endpoint in *Configure()* method of Startup.cs



Note: Can use the below option *Configure()* method

*app.UseEndpoints(endpoints =>*

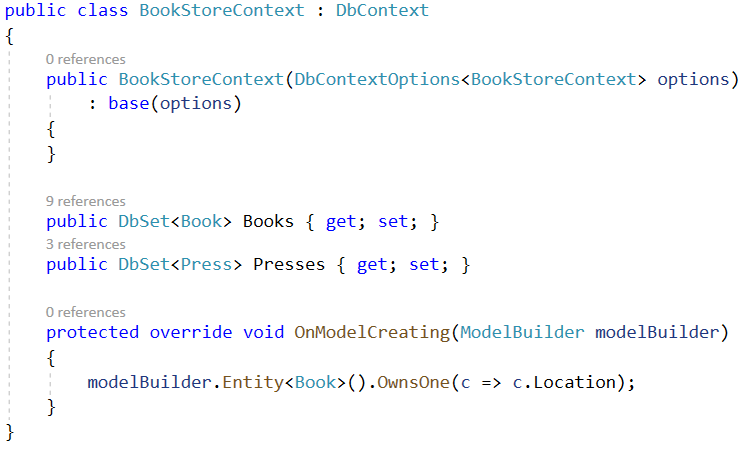
*{*

*endpoints.MapODataRoute("odata", "odata", GetEdmModel());*

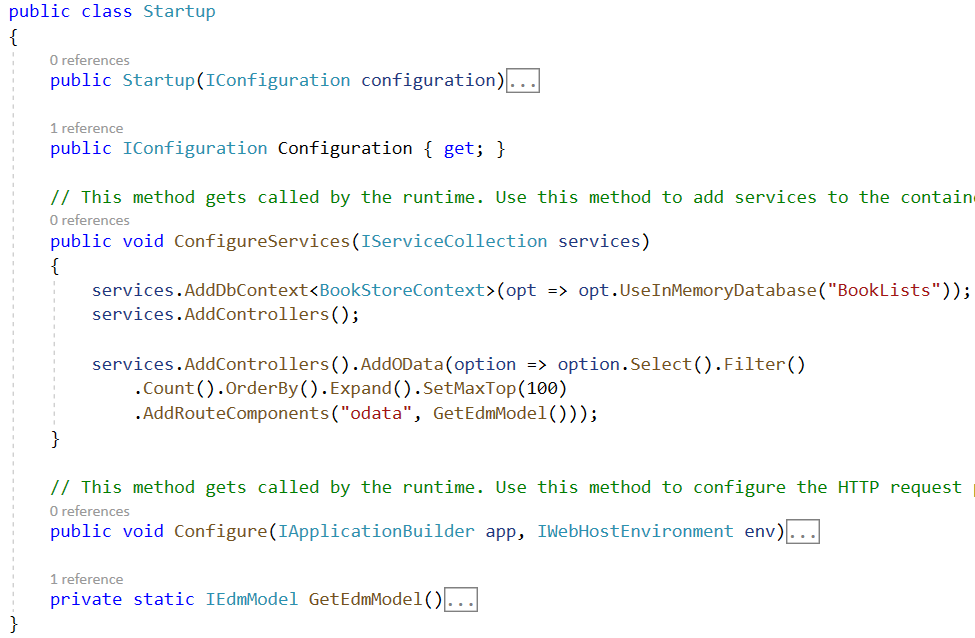
*});*

**Step 06**. Create the Data Source

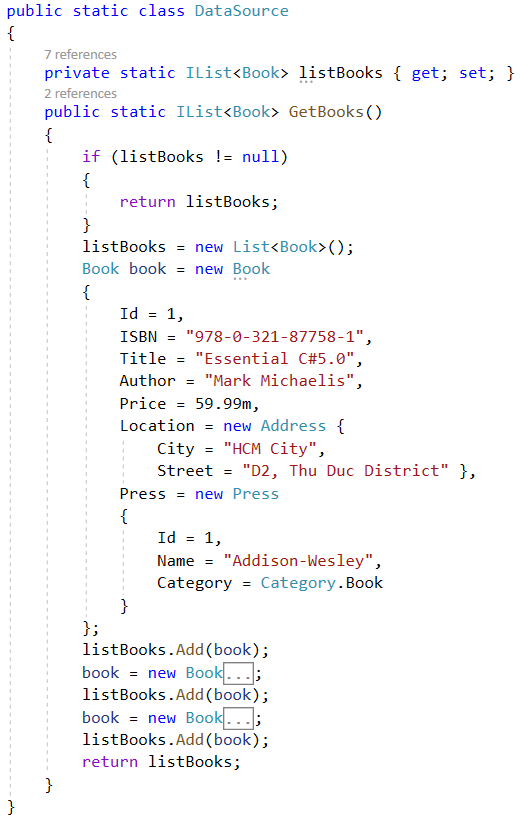
Create the data context class (this class extends DbContext class)



Add service with In-Memory Database to ConfigureServices() method of Startup.cs

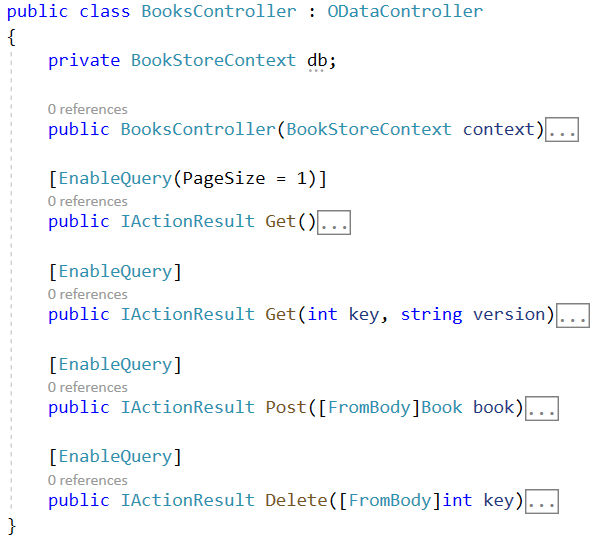


Add sample data

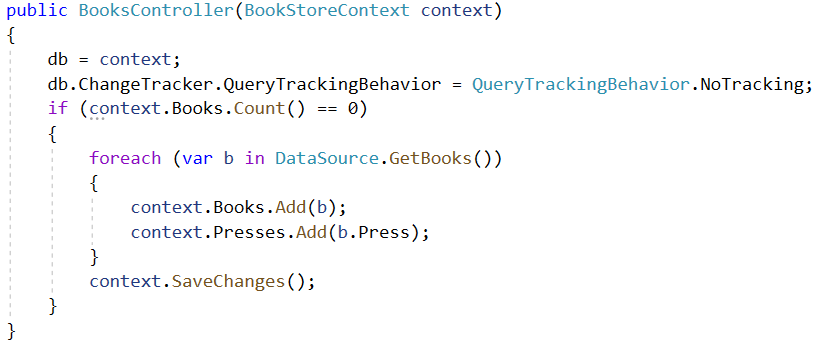


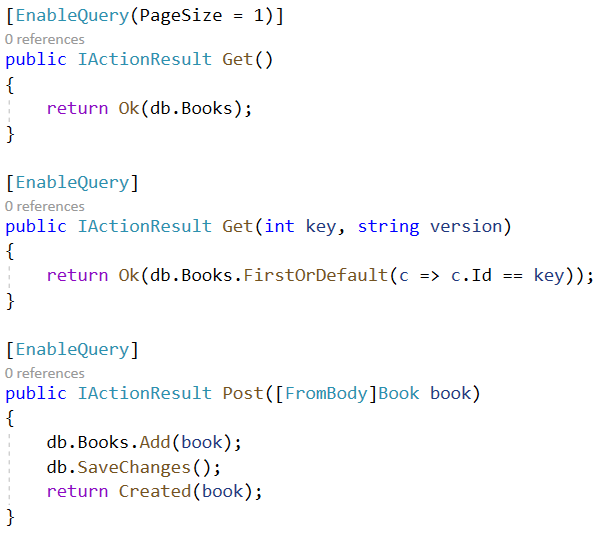
**Step 07**. Add Controllers

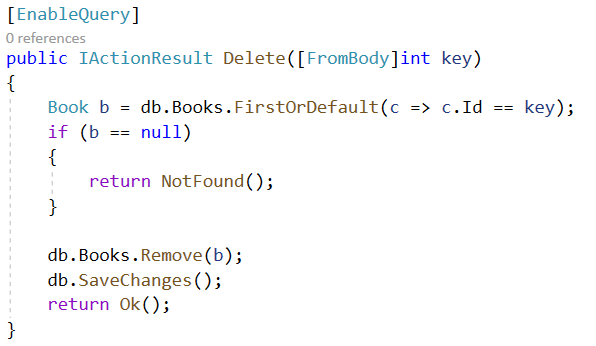
Create 2 Controllers named *BooksController, PressesController* extends ODataController.



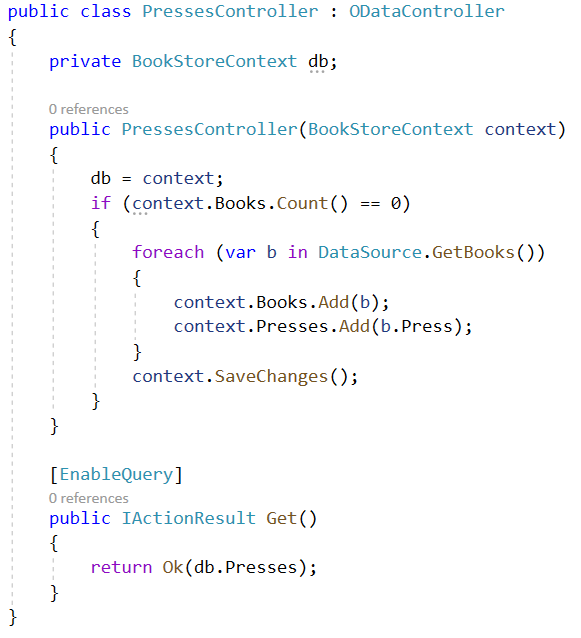
The details of constructor and functions in BooksController.







The details of functions in PressesController.

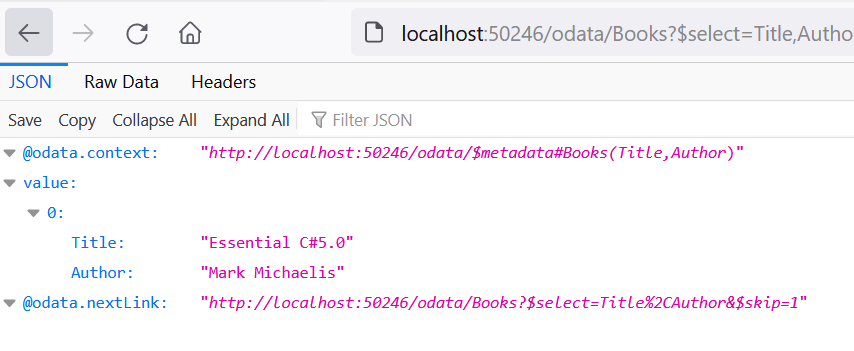


# Activity 03: Testing OData Service

**Step 01**. Test *$select* option

The $select system query option allows clients to request a specific set of properties for each entity or complex type. The set of properties will be comma-separated while requesting.

*http://localhost:50246/odata/Books?$select=Title,Author*

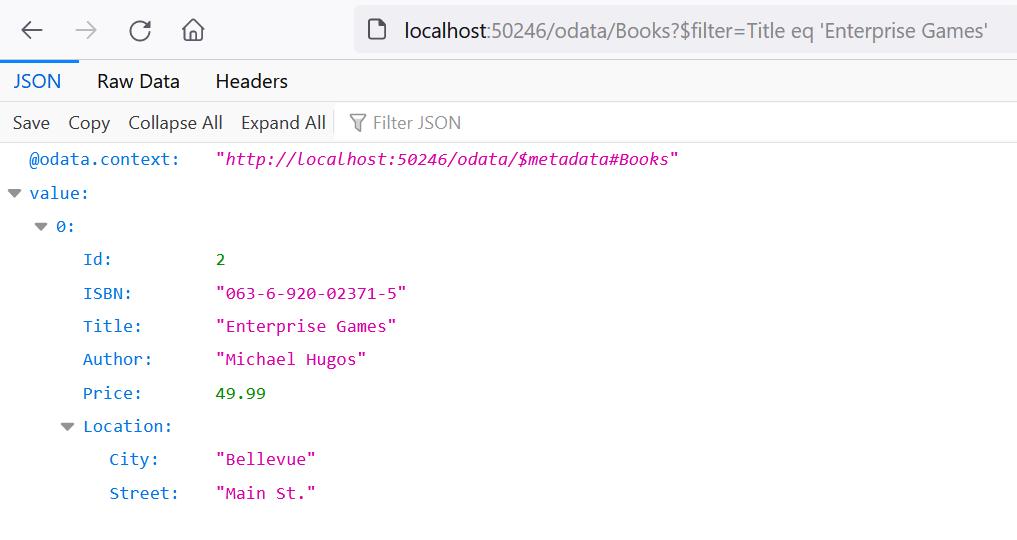


**Step 02**. Test *$filter* option

The $filter filters data based on a boolean condition. The following are conditional operators that have to be used in URLs.

* *eq - equals to.*
* *ne - not equals to*
* *gt - greater than*
* *ge - greater than or equal*
* *lt - less than*
* *le - less than or equal*

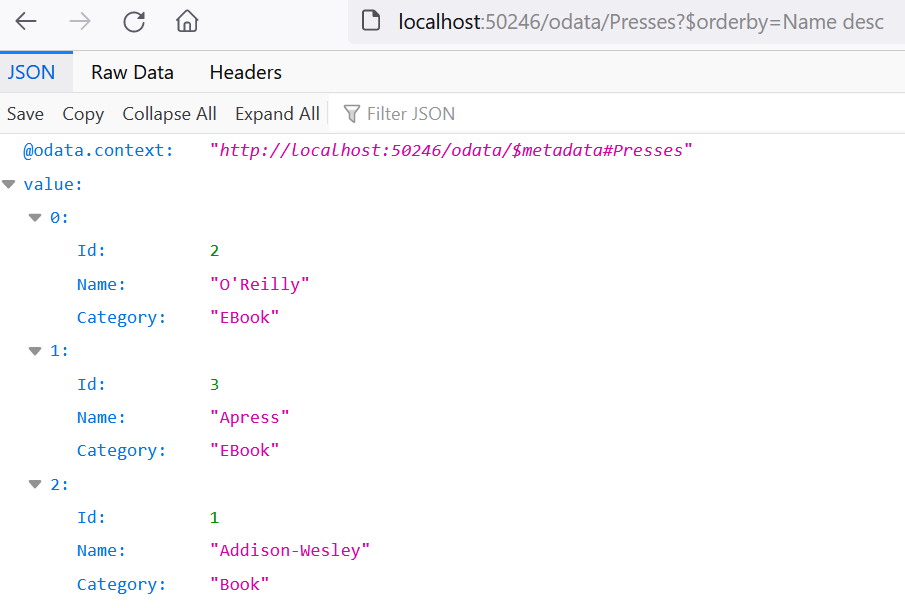
[*http://localhost:50246/odata/Books?$filter=Title*](http://localhost:50246/odata/Books?$filter=Title) *eq ‘Enterprise Games’*



**Step 03.** Test *$orderby* option

The $orderby sorts the data using 'asc' and 'desc' keywords. We can do sorting on multiple properties using comma separation.

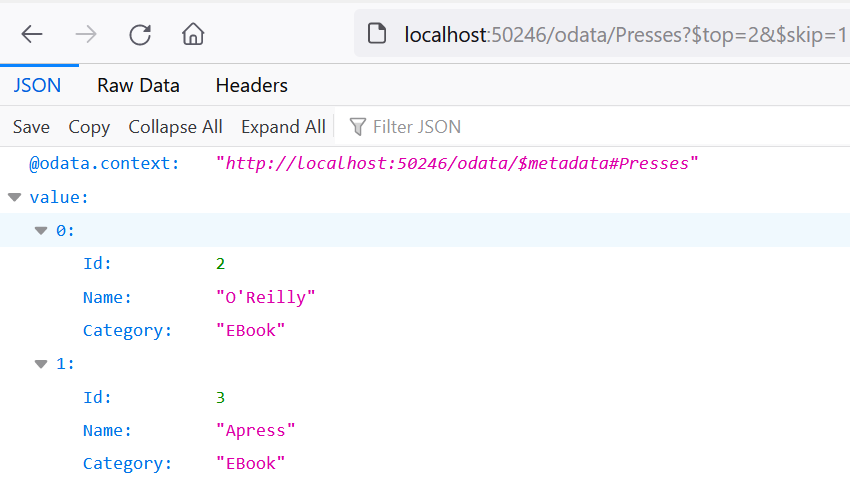
[*http://localhost:50246/odata/Presses?$orderby=Name*](http://localhost:50246/odata/Presses?$orderby=Name) *desc*



**Step 04.** Test *$top and $skip* options

The $top fetches specified the count of top records in the collection. So to work this operator, we must specify an extension method like 'SetMaxTo(specify\_max\_number)'.

<http://localhost:50246/odata/Presses?$top=2&$skip=1>

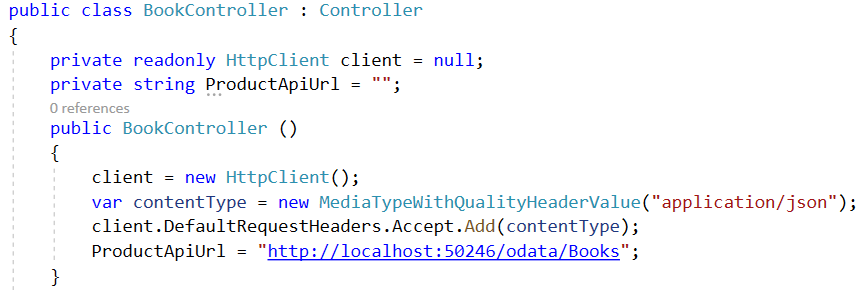


# Activity 04: ASP.NET Core Web Application with Model-View-Controller Project to get data from OData Service

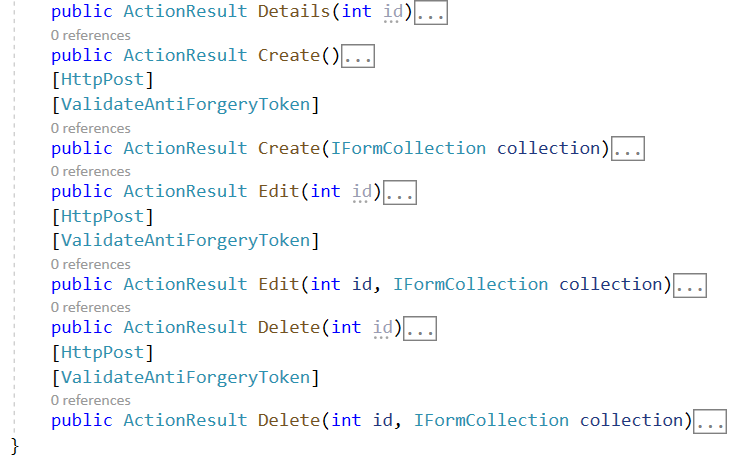
**Step 01**. Create ASP.NET Core Web App (Model-View-Controller) named ODataBookStoreWebClient

**Step 02**. Create Controller to connect to OData Service

* Use HttpClient, C# HttpClient creates HTTP requests.
* The GetAsync method sends a GET request to the specified Uri as an asynchronous operation. The await operator suspends the evaluation of the enclosing async method until the asynchronous operation completes. When the asynchronous operation completes, the await operator returns the result of the operation, if any.



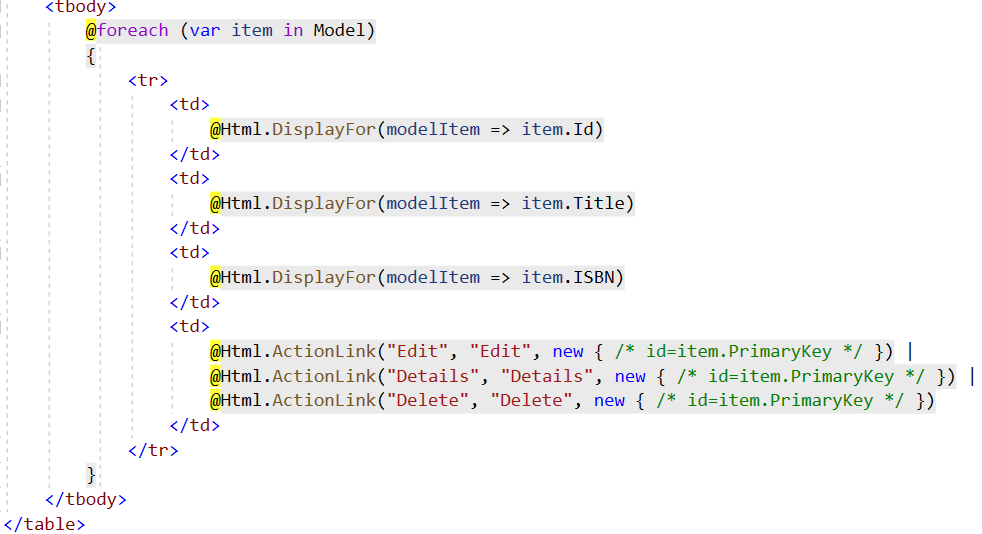




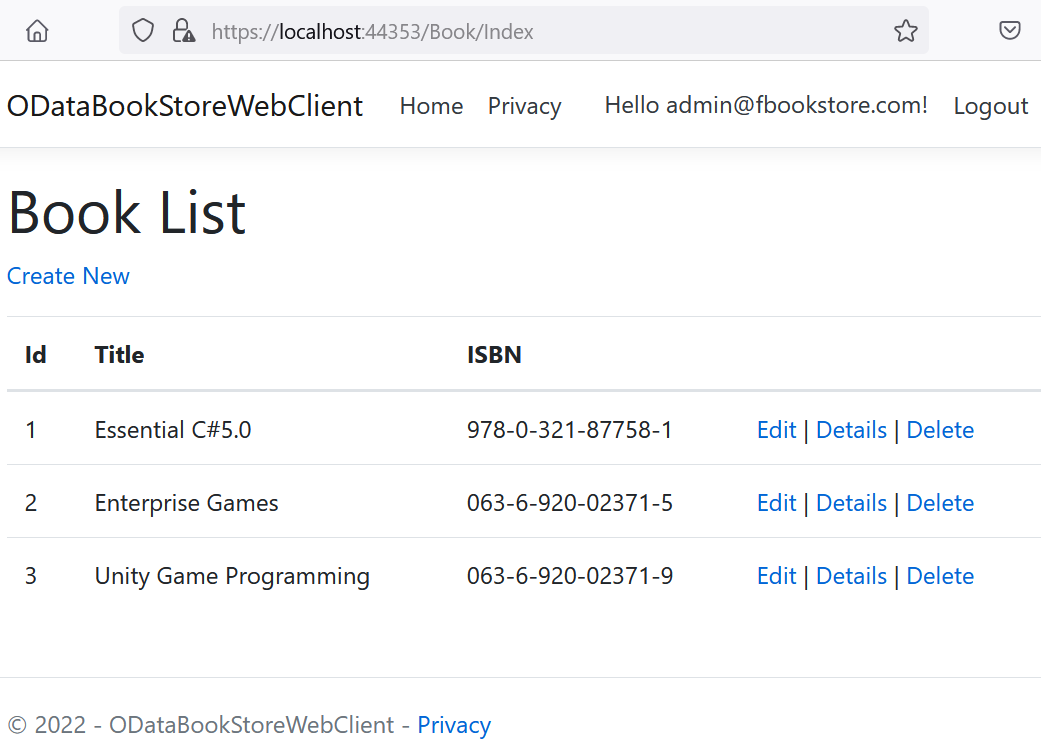


**Step 03**. Create View





**Step 04**. Test the function of Web Client



# Activity 05: Build and run Project. Test all CRUD actions

Note: Choose the option for multiple startup projects.

# 