CQRS Pattern With MediatR

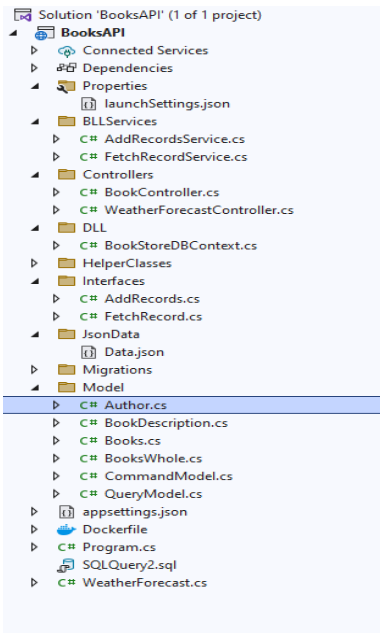
This is a web API where we are working on a Book Store, We have two methods fetch book details by **id** or **title** and another method is to add a **book details** to the database.

Step 1: Folder structure

This is a typical web API folders structure

Folders

1. Controller: we have book controller to in which acts like façade of an API
    a. Books : Fetch Book details using title or id.
    b. addAllBooks : we need to run this method to add all our book details to Database
    c. addBooks : used to add book details to Database.
2. Models : Created different models where we can find properties of books and request response.
3. Interface: we have two contract structures for adding book record and fetching book records
4. BLLServices : Business logic for adding book record and fetching book records
5. DLL : Context file
6. JSON Data: initial json files of 100 books details. Which is to be store in DB first as a sample data.

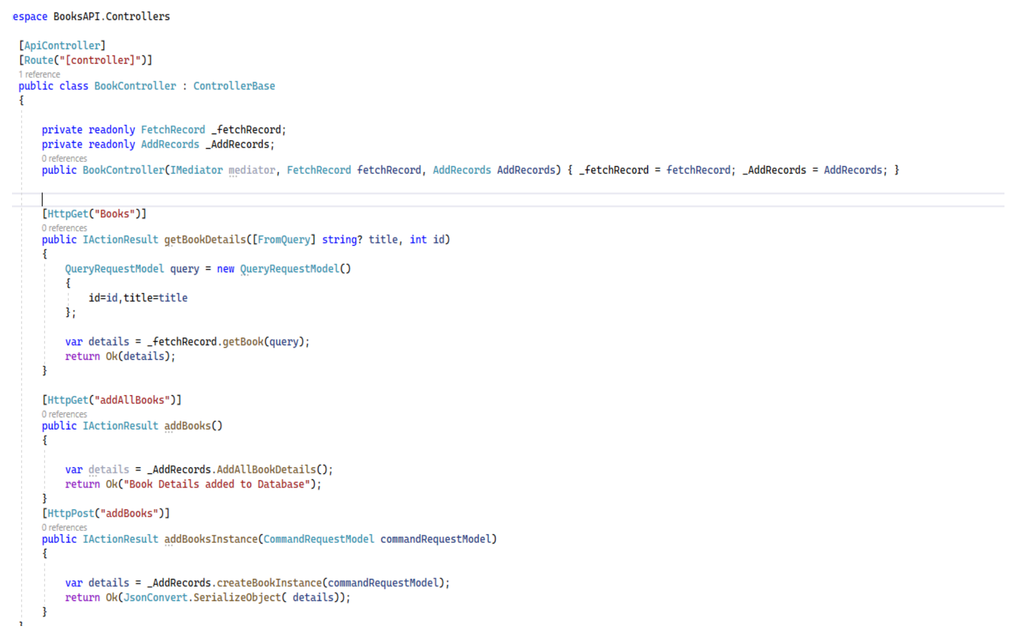


```
Solution 'BooksAPI' (1 of 1 project)
  BooksAPI
    Connected Services
    Dependencies
    Properties
      launchSettings.json
    BLLServices
      AddRecordsService.cs
      FetchRecordService.cs
    Controllers
      BookController.cs
      WeatherForecastController.cs
    DLL
      BookStoreDBContext.cs
    HelperClasses
    Interfaces
      AddRecords.cs
      FetchRecord.cs
    JsonData
      Data.json
    Migrations
    Model
      Author.cs
      BookDescription.cs
      Books.cs
      BooksWhole.cs
      CommandModel.cs
      QueryModel.cs
    appsettings.json
    Dockerfile
    Program.cs
    SQLQuery2.sql
    WeatherForecast.cs
```

Dependencies Libraries:

```
<PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.32" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Abstractions" Version="6.0.32" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Relational" Version="6.0.32" />
<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.32" />
<PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.32">
```

Controller: this is our previous controller where we have not implemented MediatR lib.



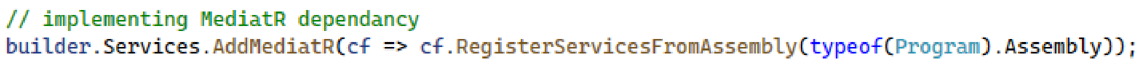
```
espace BooksAPI.Controllers

 [ApiController]
 [Route("[controller]")]
 1 reference
 public class BookController : ControllerBase
 {

     private readonly FetchRecord _fetchRecord;
     private readonly AddRecords _AddRecords;
     0 references
     public BookController(IMediator mediator, FetchRecord fetchRecord, AddRecords AddRecords) { _fetchRecord = fetchRecord; _AddRecords = AddRecords; }

     |
     [HttpGet("Books")]
     0 references
     public IActionResult getBookDetails([FromQuery] string? title, int id)
     {
         QueryRequestModel query = new QueryRequestModel()
         {
             id=id,title=title
         };

         var details = _fetchRecord.getBook(query);
         return Ok(details);
     }

     [HttpGet("addAllBooks")]
     0 references
     public IActionResult addBooks()
     {

         var details = _AddRecords.AddAllBookDetails();
         return Ok("Book Details added to Database");
     }
     [HttpPost("addBooks")]
     0 references
     public IActionResult addBooksInstance(CommandRequestModel commandRequestModel)
     {

         var details = _AddRecords.createBookInstance(commandRequestModel);
         return Ok(JsonConvert.SerializeObject( details));
     }
```

Steps to use CQRS with MediatR pattern

Step 1: Adding required Library

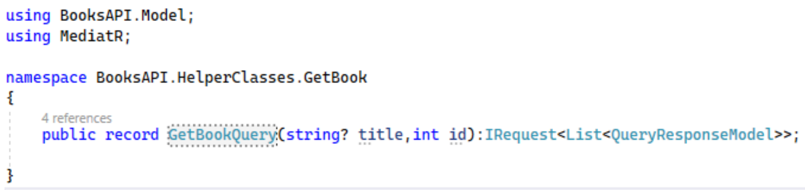
MediatR by Jimmy Bogard, **218M** downloads 12.4.0
Simple, unambitious mediator implementation in .NET

Step 2: Adding dependency injection

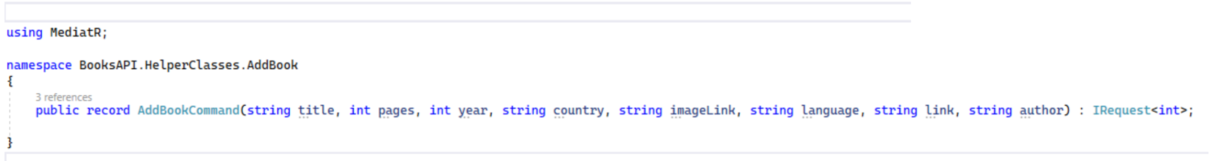


```
// implementing MediatR dependancy
builder.Services.AddMediatR(cf => cf.RegisterServicesFromAssembly(typeof(Program).Assembly));
```

Step 3: Creating request models

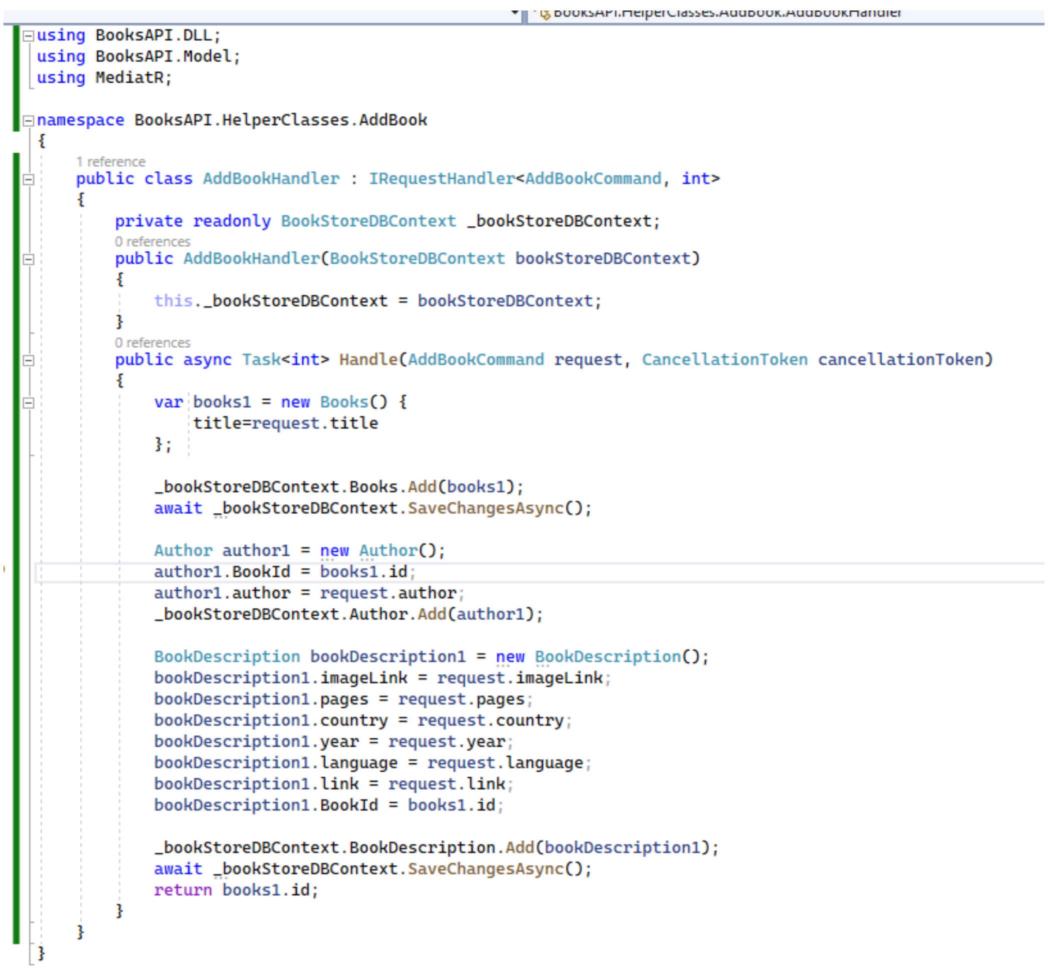


```
using BooksAPI.Model;
using MediatR;

namespace BooksAPI.HelperClasses.GetBook
{
    4 references
    public record GetBookQuery(string? title,int id):IRequest<List<QueryResponseModel>>;

}
```

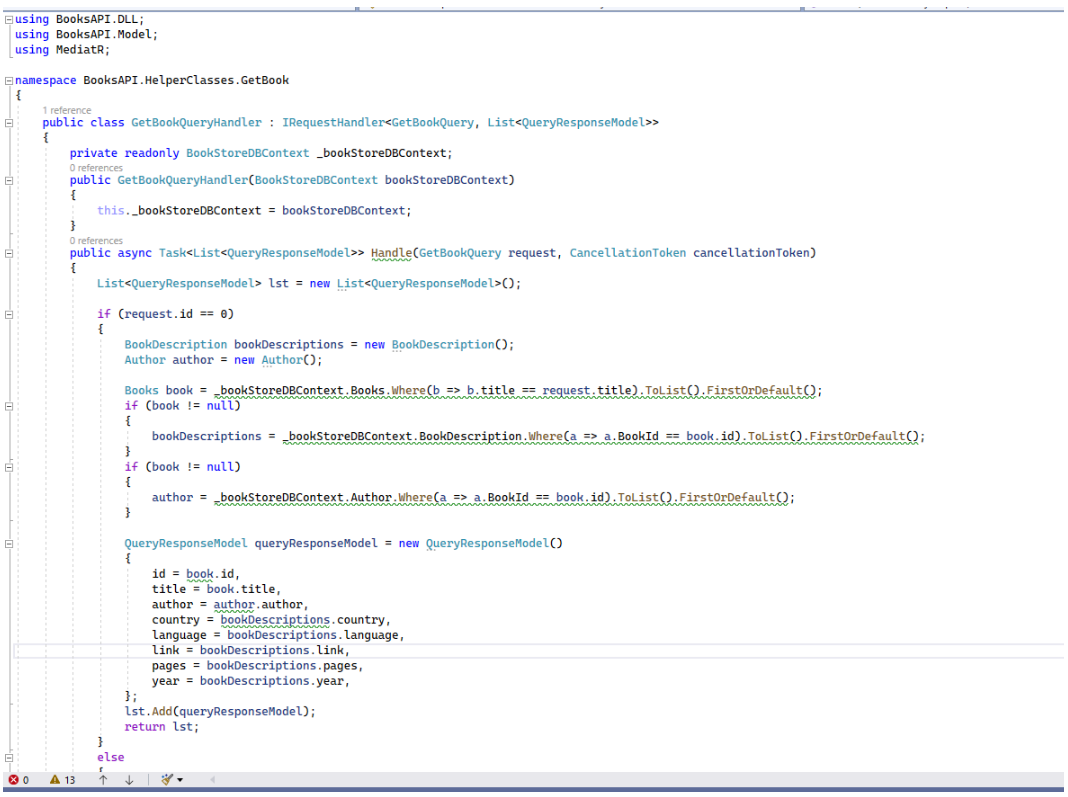


```
using MediatR;

namespace BooksAPI.HelperClasses.AddBook
{
    3 references
    public record AddBookCommand(string title, int pages, int year, string country, string imageLink, string language, string link, string author) : IRequest<int>;

}
```

Step 4: Creating Handler classes



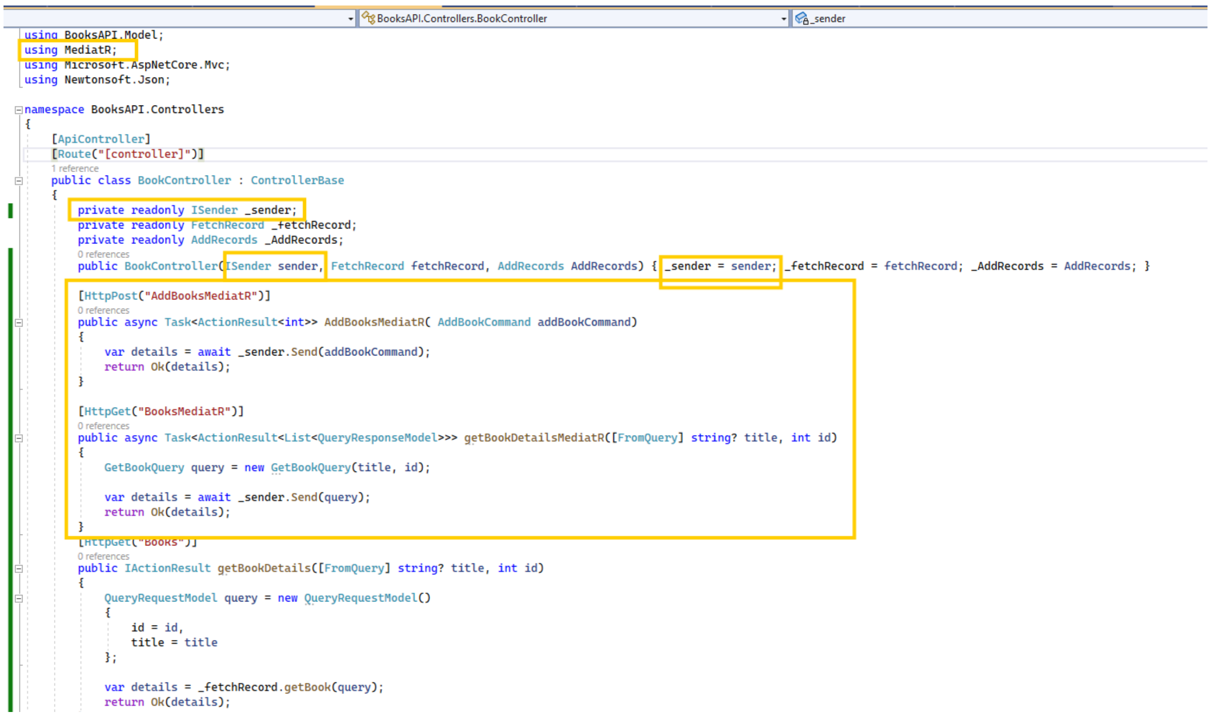
```
using BooksAPI.DLL;
using BooksAPI.Model;
using MediatR;

namespace BooksAPI.HelperClasses.AddBook
{
    1 reference
    public class AddBookHandler : IRequestHandler<AddBookCommand, int>
    {
        private readonly BookStoreDBContext _bookStoreDBContext;
        0 references
        public AddBookHandler(BookStoreDBContext bookStoreDBContext)
        {
            this._bookStoreDBContext = bookStoreDBContext;
        }
        0 references
        public async Task<int> Handle(AddBookCommand request, CancellationToken cancellationToken)
        {
            var books1 = new Books() {
                title=request.title
            };

            _bookStoreDBContext.Books.Add(books1);
            await _bookStoreDBContext.SaveChangesAsync();

            Author author1 = new Author();
            author1.BookId = books1.id;
            author1.author = request.author;
            _bookStoreDBContext.Author.Add(author1);

            BookDescription bookDescription1 = new BookDescription();
            bookDescription1.imageLink = request.imageLink;
            bookDescription1.pages = request.pages;
            bookDescription1.country = request.country;
            bookDescription1.year = request.year;
            bookDescription1.language = request.language;
            bookDescription1.link = request.link;
            bookDescription1.BookId = books1.id;

            _bookStoreDBContext.BookDescription.Add(bookDescription1);
            await _bookStoreDBContext.SaveChangesAsync();
            return books1.id;
        }
    }
}
```



```
using BooksAPI.DLL;
using BooksAPI.Model;
using MediatR;

namespace BooksAPI.HelperClasses.GetBook
{
    1 reference
    public class GetBookQueryHandler : IRequestHandler<GetBookQuery, List<QueryResponseModel>>
    {
        private readonly BookStoreDBContext _bookStoreDBContext;
        0 references
        public GetBookQueryHandler(BookStoreDBContext bookStoreDBContext)
        {
            this._bookStoreDBContext = bookStoreDBContext;
        }
        0 references
        public async Task<List<QueryResponseModel>> Handle(GetBookQuery request, CancellationToken cancellationToken)
        {
            List<QueryResponseModel> lst = new List<QueryResponseModel>();

            if (request.id == 0)
            {
                BookDescription bookDescriptions = new BookDescription();
                Author author = new Author();

                Books book = _bookStoreDBContext.Books.Where(b => b.title == request.title).ToList().FirstOrDefault();
                if (book != null)
                {
                    bookDescriptions = _bookStoreDBContext.BookDescription.Where(a => a.BookId == book.id).ToList().FirstOrDefault();
                }
                if (book != null)
                {
                    author = _bookStoreDBContext.Author.Where(a => a.BookId == book.id).ToList().FirstOrDefault();
                }

                QueryResponseModel queryResponseModel = new QueryResponseModel()
                {
                    id = book.id,
                    title = book.title,
                    author = author.author,
                    country = bookDescriptions.country,
                    language = bookDescriptions.language,
                    link = bookDescriptions.link,
                    pages = bookDescriptions.pages,
                    year = bookDescriptions.year,
                };
                lst.Add(queryResponseModel);
                return lst;
            }
            else
```

0 13

Step 5: Implementing MediatR to controller



```
using BooksAPI.Model;
using MediatR;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;

namespace BooksAPI.Controllers
{
    [ApiController]
    [Route("[controller]")]
    1 reference
    public class BookController : ControllerBase
    {
        private readonly ISender _sender;
        private readonly FetchRecord _fetchRecord;
        private readonly AddRecords _AddRecords;
        0 references
        public BookController(ISender sender, FetchRecord fetchRecord, AddRecords AddRecords) { _sender = sender; _fetchRecord = fetchRecord; _AddRecords = AddRecords; }

        [HttpPost("AddBooksMediatR")]
        0 references
        public async Task<ActionResult<int>> AddBooksMediatR( AddBookCommand addBookCommand)
        {
            var details = await _sender.Send(addBookCommand);
            return Ok(details);
        }

        [HttpGet("BooksMediatR")]
        0 references
        public async Task<ActionResult<List<QueryResponseModel>>> getBookDetailsMediatR([FromQuery] string? title, int id)
        {
            GetBookQuery query = new GetBookQuery(title, id);

            var details = await _sender.Send(query);
            return Ok(details);
        }
        [HttpGet("Books")]
        0 references
        public IActionResult getBookDetails([FromQuery] string? title, int id)
        {
            QueryRequestModel query = new QueryRequestModel()
            {
                id = id,
                title = title
            };

            var details = _fetchRecord.getBook(query);
            return Ok(details);
```