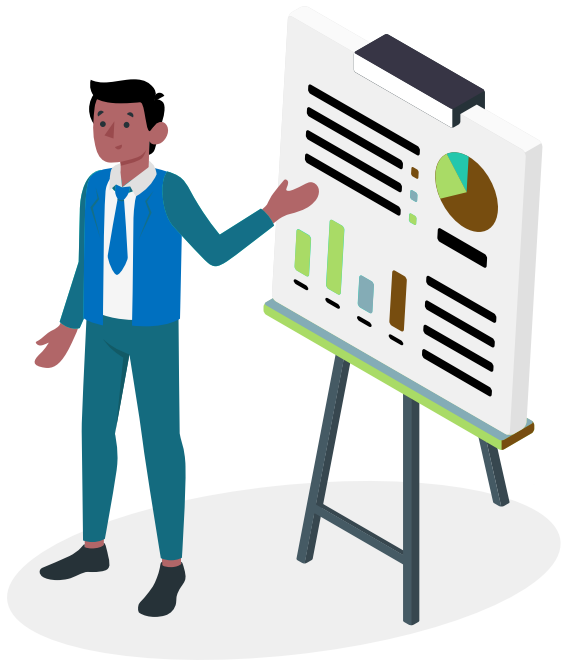
Working with data using EntityFramework

Agenda

1. • **Config Entity Framework Core and Seeding Data**
2. • **Add new Data, Update data, getting data, and deleting data with WebAPI Controller**
3. • **Working with relational data in Entity Framework**



Lesson Objectives

- Understand the role of Entity Framework Core in data access and ORM (Object-Relational Mapping).
- Learn how to configure Entity Framework Core in an ASP.NET Web API application.
- Explore different approaches for database initialization and seeding data.
- Understand how to define and configure database models using Entity Framework Core.
- Learn how to perform migrations and update the database schema using Entity Framework Core.
- Understand the basic CRUD operations (Create, Read, Update, Delete) in Web API.
- Understand the concept of relational data and its representation in Entity Framework.
- Learn how to define relationships between database entities using Entity Framework Core.
- Explore various relationship types, such as one-to-one, one-to-many, and many-to-many, in EF.
- Understand how to perform CRUD operations on related entities using Entity Framework.

Section 1

Config Entity Framework Core and Seeding Data

Adding Data Model

*Adding a **Book** Entity Model Class:*

```
public class Book
{
    public int Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public int? Rate { get; set; }
    public string Genre { get; set; }
    public string Author { get; set; }
    public DateTime DateAdded { get; set; }
}
```

Entity Framework DBContext

- Install **Microsoft.EntityFrameworkCore.SqlServer**
- Install **Microsoft.EntityFrameworkCore.Tools**
- Add Class **Context**:

```
using Microsoft.EntityFrameworkCore;
using MyBookApp.Data.Models;

namespace MyBookApp.Data
{
    public class AppDbContext: DbContext
    {
        public DbSet<Book> Books { get; set; }
        public AppDbContext(DbContextOptions options) : base(options)
        {

        }
    }
}
```

Seeding data

- In method **OnModelCreating** of class **AppDbContext**:

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Book>().HasData(new Book()
            {
                Id=1,
                Title = "1st book title",
                Description = "1st book description",
                Rate = 5,
                Genre = "Biography",
                Author = "First Author",
                DateAdded = DateTime.Now
            });
            modelBuilder.Entity<Book>().HasData(new Book()
            {
                Id=2,
                Title = "2nd book title",
                Description = "2nd book description",
                Rate = 4,
                Genre = "Biography",
                Author = "Second Author",
                DateAdded = DateTime.Now
            });
        }
```

Add Connection string

- Add connection in **appsettings.json** file

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection": "server=localhost;database=LibraryDb;Trusted_Connection=true;TrustServerCertificate=True"
  }
}
```

Register DbContext as a service

- In **Program.cs** file register **AppDbContext**:

```
var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<AppDbContext>(options =>
{
    options.UseSqlServer(connectionString);
});
```

- Adding your first EF core migration and update database:

```
PM> add-migration InitialDatabase
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
Build succeeded.
```

Lesson Summary

- *Entity data model*
- *Entity framework core*
- *Added your first controller and service*
- *HttpPost – Add New Data*
- *HttpGet – Get Data*
- *HttpPut – Update Existing Data*
- *HttpDelete – Delete Existing Data*

Section 2

CRUD with WebAPI Controller

Adding Web Api Controller

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;

namespace MyBookApp.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    0 references
    public class BooksController : ControllerBase
    {
    }
}
```

Add class Service and class ViewModel

- Add class ViewModel

```
public class BookVm
    {
        public string Title { get; set; }
        public string Description { get; set; }
        public int? Rate { get; set; }
        public string Genre { get; set; }
        public string Author { get; set; }
    }
```

Add class Service and class ViewModel

- Add class **BookService**
- Register **BookService** in **program.cs** file:

```
builder.Services.AddScoped<BookService>()
;
```

```
public class BookService
  {
    private readonly AppDbContext _context;

    public BookService(AppDbContext context)
    {
      _context = context;
    }

    public void AddBook(BookVm book)
    {
      _context.Books.Add(new Book()
      {
        Title= book.Title,
        Description= book.Description,
        Rate= book.Rate,
        Genre= book.Genre,
        Author= book.Author,
      });

      _context.SaveChanges();
    }
  }
```

Adding a new Book [HttpPost]

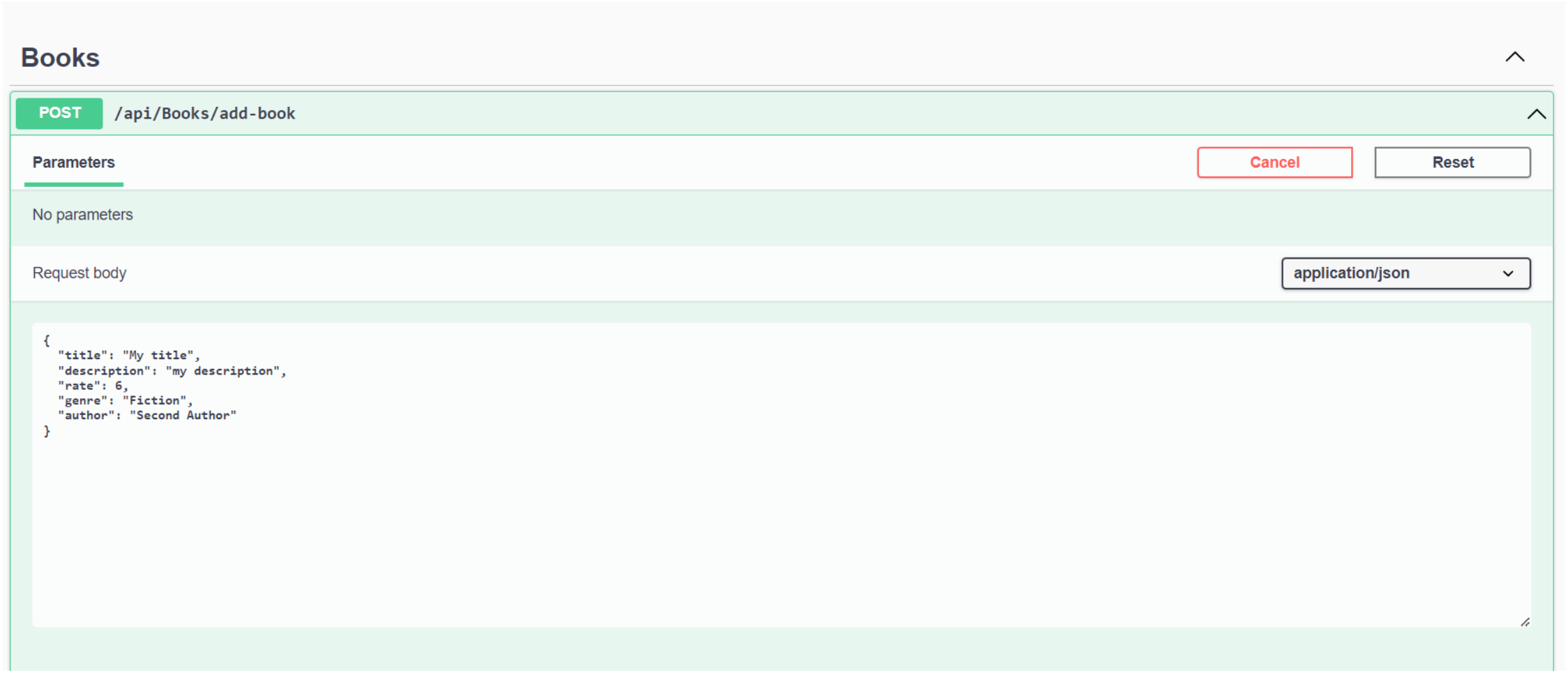
- Add Method **AddBook()** in **BooksController**

```
public class BooksController : ControllerBase
    {
        private readonly BookService _bookService;
        public BooksController(BookService bookService)
        {
            _bookService= bookService;
        }

        [HttpPost("add-book")]
        public IActionResult AddBook([FromBody] BookVm book)
        {
            _bookService.AddBook(book);

            return Ok();
        }
    }
```

Testing with add book



Listing all Book [HttpGet]

- Update **BookService** add method **GetBooks()** and **GetBook(int id)**

```
public List<Book> GetBooks()
        {
            return _context.Books.ToList();
        }

        public Book GetBook(int id)
        {
            return _context.Books.Find(id);
        }
```

- Add method in controller:

```
[HttpGet("get-all-books")]
        public IActionResult GetAllBooks()
        {
            var books = _bookService.GetBooks();

            return Ok(books);
        }

        [HttpGet("get-book-by-id/{id}")]
        public IActionResult GetBook(int id)
        {
            var book = _bookService.GetBook(id);

            return Ok(book);
        }
```

Updating an existing book

- Update **BookService**:

```
public Book UpdateBook(int id, BookVm book)
        {
            var _book = _context.Books.Find(id);

            if (_book != null)
            {
                _book.Title = book.Title;
                _book.Description = book.Description;
                _book.Rate = book.Rate;
                _book.Genre = book.Genre;
                _book.Author = book.Author;

                _context.SaveChanges();
            }

            return _book;
        }
```

Updating an existing book

- Add method **UpdateBookById()**:

```
[HttpPut("update-book-by-id/{id}")]
        public IActionResult UpdateBookById(int id,[FromBody] BookVm bookVm)
        {
            var book = _bookService.UpdateBook(id,bookVm);

            return Ok(book);
        }
```

Deleting an existing book [HttpDelete]

- Update **BookService**:

```
public void DeleteBook(int id)
        {
            var book = _context.Books.Find(id);

            if (book != null)
            {
                _context.Books.Remove(book);
                _context.SaveChanges();
            }
        }
```

- Update **Controller**:

```
[HttpDelete("delete-book-by-id/{id}")]
        public IActionResult DeleteBookById(int id)
        {
            _bookService.DeleteBook(id);

            return Ok();
        }
```

Section 3

Working with relational data in EF

Add model class Publisher

- Add model class **Publisher** and create relationship with **Book** is one to many:

```
public class Publisher
   {
      public int Id { get; set; }
      public string Name { get; set; }

      public List<Book> Books { get; set;
}
   }
```

```
public class Book
   {
      public int Id { get; set; }
      public string Title { get; set; }
      public string Description { get; set; }
      public int? Rate { get; set; }
      public string Genre { get; set; }
      public string Author { get; set; }
      public DateTime DateAdded { get; set; }

      public int PublisherId { get; set; }
      public Publisher Publisher { get; set; }
   }
```

Add model class Author, BookAuthor

- Add model class **Author** and create relationship with **Book** is many to many:

```
public class Author
{
  public int Id { get; set; }
  public string FullName { get; set; }
  public List<BookAuthor> BookAuthors { get; set; }
}
```

```
public class Book
{
  public int Id { get; set; }
  public string Title { get; set; }
  public string Description { get; set; }
  public int? Rate { get; set; }
  public string Genre { get; set; }
  //public string Author { get; set; }
  public DateTime DateAdded { get; set; }

  public int PublisherId { get; set; }
  public Publisher Publisher { get; set; }

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

}
```

```
public class BookAuthor
{
  public int AuthorId { get; set; }
  public Author Author { get; set; }
  public int BookId { get; set; }
  public Book Book { get; set; }
}
```

Config relationship many to many with Book and Author

- Update **AppDbContext**:

```
protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            modelBuilder.Entity<BookAuthor>()
                .HasKey(e => new { e.AuthorId, e.BookId });

            modelBuilder.Entity<BookAuthor>()
                .HasOne(b => b.Book)
                .WithMany(ba => ba.BookAuthors)
                .HasForeignKey(ba => ba.BookId);

            modelBuilder.Entity<BookAuthor>()
                .HasOne(b => b.Author)
                .WithMany(ba => ba.BookAuthors)
                .HasForeignKey(ba => ba.AuthorId);

        }
```

Add Publisher Service and AuthorService

- Publisher
  - Add Publisher Service
  - Add PublisherVm
  - Add PublisherController
- Author
  - Add Author Service
  - Add AuthorVm
  - Add AuthorController
- Configure Services

Add Book with Publisher and Authors

- Update **BookVM**:

```
public class BookVm
{
    public string Title { get; set; }
    public string Description { get; set; }
    public int? Rate { get; set; }
    public string Genre { get; set; }
    //public string Author { get; set; }
    public int PublisherId { get; set; }
    public List<int> AuthorId { get; set; }
}
```

Add Book with Publisher and Authors

- Change method name **AddBook** to **AddBookWithAuthors**:

```
public void AddBookWithAuthors(BookVm book)
        {
            var _book =new Book()
            {
                Title = book.Title,
                Description = book.Description,
                Rate = book.Rate,
                Genre = book.Genre,
                //Author = book.Author,
                PublisherId= book.PublisherId,
                DateAdded = DateTime.Now
            };

            _context.Books.Add(_book);
            _context.SaveChanges();

            foreach (var authorId in book.AuthorId)
            {
                _context.BookAuthors.Add(new BookAuthor() { BookId = _book.Id, AuthorId = authorId });
            }

            _context.SaveChanges();
        }
```

- Update attribute of method **AddBook**: `[HttpPost("add-book-with-authors")]`

Get Book with publisherName and Authors

- Add class **BookWithAuthorsVM**:

```
public class BookWithAuthorsVm
    {
        public string Title { get; set; }
        public string Description { get; set; }
        public int? Rate { get; set; }
        public string Genre { get; set; }
        public string PublisherName { get; set; }
        public List<string> AuthorNames { get; set; }
    }
```

- Add method in **BookService** and add method **GetBookWithAuthors** in controller:

```
public BookWithAuthorsVm GetBookWithAuthors(int id)
        {
            var bookWithAuthors = _context.Books.Where(b => b.Id == id)
                .Select(book => new BookWithAuthorsVm()
                {
                    Title = book.Title,
                    Description = book.Description,
                    Rate = book.Rate,
                    Genre = book.Genre,
                    PublisherName = book.Publisher.Name,
                    AuthorNames = book.BookAuthors.Select(a => a.Author.FullName).ToList()
                }).FirstOrDefault();

            return bookWithAuthors;
        }
```

Lesson Summary

- *Configure relationship type: one to many, many to many*
- *Add Relational Data*
- *Get Relational Data*

THANK YOU!