<connectionStrings>

<add name="BookServiceContext"

connectionString="Data Source=(localdb)\v11.0; Initial Catalog=BookDB; Integrated Security=True;"

providerName="System.Data.SqlClient"/>

</connectionStrings>

CONTEXT

using System;

using System.Collections.Generic;

using System.Data.Entity;

using System.Linq;

using System.Web;

namespace BookService.Models

{

public class **BookServiceContext** : DbContext

{

// If you want Entity Framework to drop and regenerate your database

// automatically whenever you change your model schema, please use data migrations.

// http://msdn.microsoft.com/en-us/data/jj591621.aspx

public BookServiceContext() : base("**name=BookServiceContext**")

{

this.Database.Log = s => System.Diagnostics.Debug.WriteLine(s);

}

public System.Data.Entity.DbSet<BookService.Models.Book> Books { get; set; }

}

}

CONTROLLER

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Threading.Tasks;

using System.Web.Http;

using System.Web.Http.Description;

using BookService.Models;

namespace BookService.Controllers

{

public class BooksController : ApiController

{

private **BookServiceContext** db = new **BookServiceContext**();

// GET: api/Books

public IQueryable<BookDTO> **GetBooks()**

{

var books = from b in db.Books

select new BookDTO()

{

Id = b.Id,

Title = b.Title,

AuthorName = b.Author.Name

};

return books;

}

// GET: api/Books/5

[ResponseType(typeof(BookDetailDTO))]

public async Task<IHttpActionResult**> GetBook(int id)**

{

var book = await db.Books.Include(b => b.Author).Select(b =>

new BookDetailDTO()

{

Id = b.Id,

Title = b.Title,

Year = b.Year,

Price = b.Price,

AuthorName = b.Author.Name,

Genre = b.Genre

}).SingleOrDefaultAsync(b => b.Id == id);

if (book == null)

{

return NotFound();

}

return Ok(book);

}

// PUT: api/Books/5

[ResponseType(typeof(void))]

public async Task<IHttpActionResult> **PutBook(int id, Book book)**

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

if (id != book.Id)

{

return BadRequest();

}

db.Entry(book).State = EntityState.Modified;

try

{

await db.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!BookExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return StatusCode(HttpStatusCode.NoContent);

}

// POST: api/Books

[ResponseType(typeof(Book))]

public async Task<IHttpActionResult> **PostBook(Book book)**

{

if (!ModelState.IsValid)

{

return BadRequest(ModelState);

}

db.Books.Add(book);

await db.SaveChangesAsync();

// Load author name

db.Entry(book).Reference(x => x.Author).Load();

var dto = new BookDTO()

{

Id = book.Id,

Title = book.Title,

AuthorName = book.Author.Name

};

return CreatedAtRoute("DefaultApi", new { id = book.Id }, dto);

}

// DELETE: api/Books/5

[ResponseType(typeof(Book))]

public async Task<IHttpActionResult> **DeleteBook(int id)**

{

Book book = await db.Books.FindAsync(id);

if (book == null)

{

return NotFound();

}

db.Books.Remove(book);

await db.SaveChangesAsync();

return Ok(book);

}

protected override void **Dispose(bool disposing)**

{

if (disposing)

{

db.Dispose();

}

base.Dispose(disposing);

}

private bool **BookExists(int id)**

{

return db.Books.Count(e => e.Id == id) > 0;

}

}

}