WEB PROGRAMMING ASP.NET MVC CORE

BOOK MODEL

```
namespace Books.Models {
    public class Book {
        public int BookId { get; set; }
        public string Title { get; set; }
        public Author Author { get; set; }
        public int AuthorId { get; set; }
        public ICollection<BookCategory> Categories { get; set; }
    }
}
```

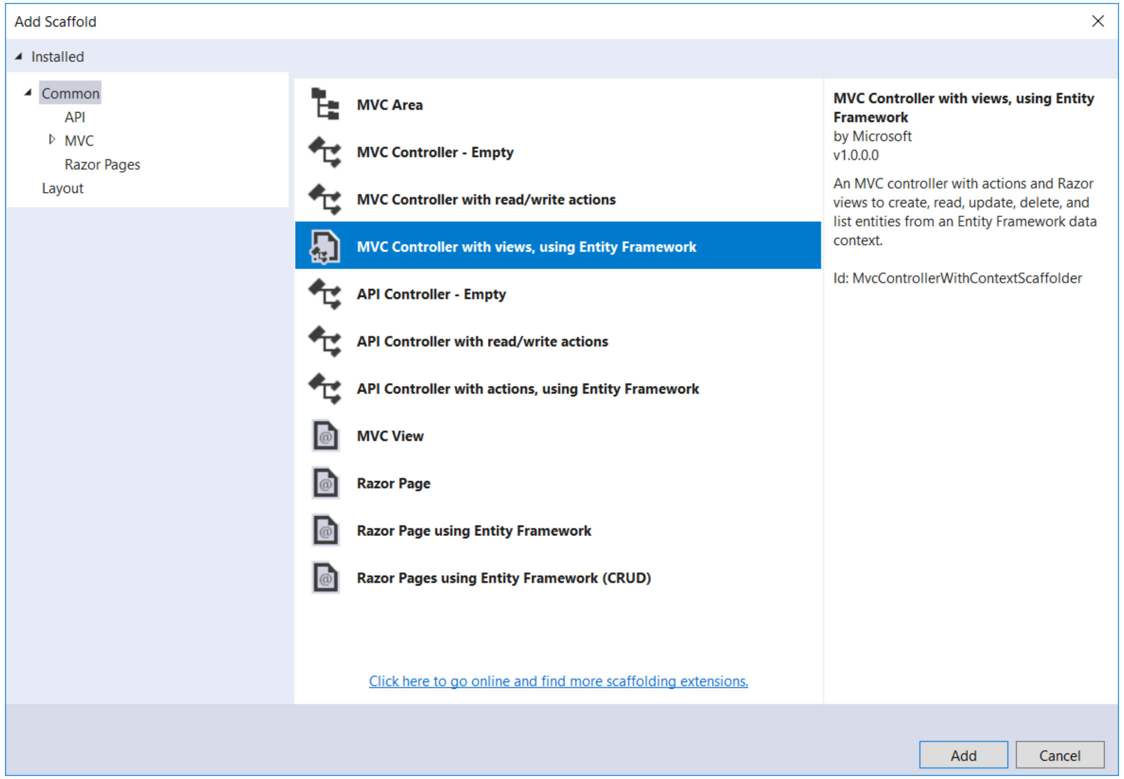
CATEGORY MODEL

```
namespace Books.Models {
    public class Category {
        public int CategoryId { get; set; }
        public string Name { get; set; }
        public ICollection<BookCategory> Books { get; set; }
    }
}
```

BOOKCATEGORY MODEL

```
namespace Books.Models {
    public class BookCategory {
        public int BookId { get; set; }
        public Book Book { get; set; }
        public int CategoryId { get; set; }
        public Category Category { get; set; }
    }
}
```

SCAFFOLDING

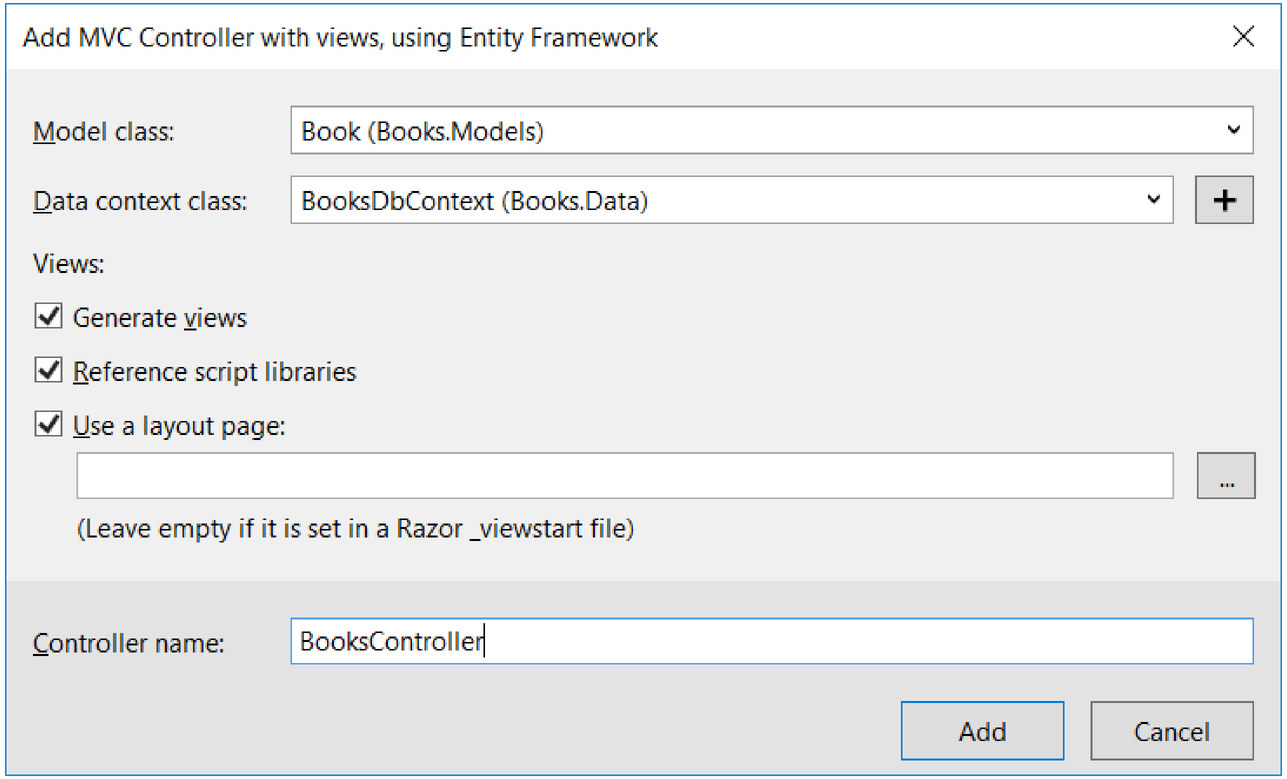

Add Scaffold
Installed
Common
API
MVC
Razor Pages
Layout
MVC Area
MVC Controller - Empty
MVC Controller with read/write actions
MVC Controller with views, using Entity Framework
API Controller - Empty
API Controller with read/write actions
API Controller with actions, using Entity Framework
MVC View
Razor Page
Razor Page using Entity Framework
Razor Pages using Entity Framework (CRUD)
Click here to go online and find more scaffolding extensions.
Add
Cancel

MVC Controller with views, using Entity Framework
by Microsoft
v1.0.0.0

An MVC controller with actions and Razor views to create, read, update, delete, and list entities from an Entity Framework data context.

Id: MvcControllerWithContextScaffolder

SCAFFOLDING



Add MVC Controller with views, using Entity Framework

Model class: Book (Books.Models)

Data context class: BooksDbContext (Books.Data)

Views:

☑ Generate views

☑ Reference script libraries

☑ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Controller name: BooksController

DATABASE CONTEXT

```
public class BooksDbContext : DbContext {
    public BooksDbContext(
            DbContextOptions<BooksDbContext> options) : base(options) {
    }

    public DbSet<Author> Authors;
    public DbSet<Book> Books;
    public DbSet<Category> Categories;
    public DbSet<BookCategory> BooksCategories;
}
```

DATABASE CONTEXT: FLUENT API FOR EF CORE

```
public class BooksDbContext : DbContext {
    // ...

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        // ...
    }
}
```

COMPOSITE PRIMARY KEY

```
public class BooksDbContext : DbContext {
    // ...

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<BookCategory>()
                .HasKey(bc => new { bc.BookId, bc.CategoryId });
    }
}
```

MANY TO MANY RELATIONSHIP

```
public class BooksDbContext : DbContext {
        // ...

        protected override void OnModelCreating(ModelBuilder modelBuilder) {
            modelBuilder.Entity<BookCategory>()
                .HasKey(bc => new { bc.BookId, bc.CategoryId });

            modelBuilder.Entity<BookCategory>()
                .HasOne(bc => bc.Book)
                .WithMany(b => b.BookCategories)
                .HasForeignKey(bc => bc.BookId);

            modelBuilder.Entity<BookCategory>()
                .HasOne(bc => bc.Category)
                .WithMany(c => c.BookCategories)
                .HasForeignKey(bc => bc.CategoryId);
        }
}
```

CASCADE DELETE

```
public class BooksDbContext : DbContext {
    // ...

    protected override void OnModelCreating(ModelBuilder modelBuilder) {
        modelBuilder.Entity<BookCategory>()
            .HasKey(bc => new { bc.BookId, bc.CategoryId });

        modelBuilder.Entity<BookCategory>()
            .HasOne(bc => bc.Book)
            .WithMany(b => b.BookCategories)
            .HasForeignKey(bc => bc.BookId)
            .OnDelete(DeleteBehavior.Restrict);

        modelBuilder.Entity<BookCategory>()
            .HasOne(bc => bc.Category)
            .WithMany(c => c.BookCategories)
            .HasForeignKey(bc => bc.CategoryId)
            .OnDelete(DeleteBehavior.Restrict);
    }
}
```

DISABLE CASCADE DELETE GLOBALLY

```
protected override void OnModelCreating(ModelBuilder modelBuilder) {
    // https://stackoverflow.com/questions/46526230/disable-cascade-delete-on-ef-core-2-globally
    var cascadeFKs = modelBuilder.Model.GetEntityTypes()
                    .SelectMany(t => t.GetForeignKeys())
                    .Where(fk => !fk.IsOwnership && fk.DeleteBehavior == DeleteBehavior.Cascade);

    foreach (var fk in cascadeFKs) {
        fk.DeleteBehavior = DeleteBehavior.Restrict;
    }
}
```

CONNECTION STRING (APPSETTINGS.JSON)

```
{
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  },
  "ConnectionStrings": {
    "ConnectionStringBooks":
     "Server=(localdb)\\mssqllocaldb;Database=Books;Trusted_Connection=True;MultipleActiveResultSets=true"
  }
}
```

CREATE DATABASE

```
Add-Migration initial -Context BooksDbContext
Update-Database -Context BooksDbContext
```