BaseEntity..all entities will have these…

```
public class BaseEntity
    {
        public Int64 Id
        {
            get;
            set;
        }
        public DateTime AddedDate
        {
            get;
            set;
        }
        public DateTime ModifiedDate
        {
            get;
            set;
        }
        public string IPAddress
        {
            get;
            set;
        }
```

Repository Interface

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace GR.Data
{
    public interface IRepository<T> where T : BaseEntity
    {
        IEnumerable<T> GetAll();
        T Get(long id);
        void Insert(T entity);
        void Update(T entity);
        void Delete(T entity);
    }
}
```

Repository Implementation

```
sing Microsoft.EntityFrameworkCore;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
```

```
public class Repository<T> : IRepository<T> where T : BaseEntity
    {
        private readonly ApplicationContext context;
        private DbSet<T> entities;
        string errorMessage = string.Empty;

        public Repository(ApplicationContext context)
        {
            this.context = context;
            entities = context.Set<T>();
        }
        public IEnumerable<T> GetAll()
        {
            return entities.AsEnumerable();
        }

        public T Get(long id)
        {
            return entities.SingleOrDefault(s => s.Id == id);
        }
        public void Insert(T entity)
        {
            if (entity == null)
            {
                throw new ArgumentNullException("entity");
            }
            entities.Add(entity);
            context.SaveChanges();
        }

        public void Update(T entity)
        {
            if (entity == null)
            {
                throw new ArgumentNullException("entity");
            }
            context.SaveChanges();
        }

        public void Delete(T entity)
        {
            if (entity == null)
            {
                throw new ArgumentNullException("entity");
            }
            entities.Remove(entity);
            context.SaveChanges();
        }

    }
```

Startup Class registrations

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Logging;
using GR.Data;
using Microsoft.EntityFrameworkCore;
```

```
services.AddMvc();
            services.AddDbContext<ApplicationContext>(options =>
options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
            services.AddScoped(typeof(IRepository<>), typeof(Repository<>));
```

AppSettings

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Data Source=(local);Initial Catalog=GRepoDb"
  },
  "ApplicationInsights": {
    "InstrumentationKey": ""
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Debug",
      "System": "Information",
      "Microsoft": "Information"
    }
  }
}
```

Using in a controller

```
private IRepository<Author> repoAuthor;
        private IRepository<Book> repoBook;
        public AuthorController(IRepository<Author> repoAuthor, IRepository<Book>
repoBook)
        {
```

```
            this.repoAuthor = repoAuthor;
            this.repoBook = repoBook;
        }



public IActionResult Index()
        {
            List<AuthorListingViewModel> model = new List<AuthorListingViewModel>();

            repoAuthor.GetAll().ToList().ForEach(a =>
            {
                AuthorListingViewModel author = new AuthorListingViewModel
                {
                    Id = a.Id,
                    Name = $"{a.FirstName} {a.LastName}",
                    Email = a.Email
                };
                author.TotalBooks = repoBook.GetAll().Count(x => x.AuthorId == a.Id);
                model.Add(author);
            });
            return View("Index", model);

        }




using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using GR.Web.Models;
using GR.Data;


namespace GR.Web.Controllers
{
    public class AuthorController : Controller
    {
        private IRepository<Author> repoAuthor;
        private IRepository<Book> repoBook;
        public AuthorController(IRepository<Author> repoAuthor, IRepository<Book>
repoBook)
        {
            this.repoAuthor = repoAuthor;
            this.repoBook = repoBook;
        }

        [HttpGet]
        public IActionResult Index()
        {
            List<AuthorListingViewModel> model = new List<AuthorListingViewModel>();

            repoAuthor.GetAll().ToList().ForEach(a =>
```

```
            {
                AuthorListingViewModel author = new AuthorListingViewModel
                {
                    Id = a.Id,
                    Name = $"{a.FirstName} {a.LastName}",
                    Email = a.Email
                };
                author.TotalBooks = repoBook.GetAll().Count(x => x.AuthorId == a.Id);
                model.Add(author);
            });
            return View("Index", model);

        }

        [HttpGet]
        public PartialViewResult AddAuthor()
        {
            AuthorBookViewModel model = new AuthorBookViewModel();
            return PartialView("_AddAuthor", model);
        }

        [HttpPost]
        public ActionResult AddAuthor(AuthorBookViewModel model)
        {
            Author author = new Author
            {
                FirstName = model.FirstName,
                LastName = model.LastName,
                Email = model.Email,
                AddedDate = DateTime.UtcNow,
                IPAddress = Request.HttpContext.Connection.RemoteIpAddress.ToString(),
                ModifiedDate = DateTime.UtcNow,
                Books = new List<Book>
                {
                    new Book
                    {
                        Name = model.BookName,
                        ISBN= model.ISBN,
                        Publisher = model.Publisher,
                        IPAddress =
Request.HttpContext.Connection.RemoteIpAddress.ToString(),
                        AddedDate = DateTime.UtcNow,
                        ModifiedDate = DateTime.UtcNow
                    }
                }
            };
            repoAuthor.Insert(author);
            return RedirectToAction("Index");
        }

        [HttpGet]
        public IActionResult EditAuthor(long id)
        {
            AuthorViewModel model = new AuthorViewModel();
            Author author = repoAuthor.Get(id);
            if (author != null)
            {
                model.FirstName = author.FirstName;
```

```
                model.LastName = author.LastName;
                model.Email = author.Email;
            }
            return PartialView("_EditAuthor", model);
        }
        [HttpPost]
        public IActionResult EditAuthor(long id, AuthorViewModel model)
        {
            Author author = repoAuthor.Get(id);
            if (author != null)
            {
                author.FirstName = model.FirstName;
                author.LastName = model.LastName;
                author.Email = model.Email;
                author.IPAddress =
Request.HttpContext.Connection.RemoteIpAddress.ToString();
                author.ModifiedDate = DateTime.UtcNow;
                repoAuthor.Update(author);
            }
            return RedirectToAction("Index");
        }


        [HttpGet]
        public PartialViewResult AddBook(long id)
        {
            BookViewModel model = new BookViewModel();
            return PartialView("_AddBook", model);
        }
        [HttpPost]
        public IActionResult AddBook(long id, BookViewModel model)
        {
            Book book = new Book
            {
                AuthorId = id,
                Name = model.BookName,
                ISBN = model.ISBN,
                Publisher = model.Publisher,
                IPAddress = Request.HttpContext.Connection.RemoteIpAddress.ToString(),
                AddedDate = DateTime.UtcNow,
                ModifiedDate = DateTime.UtcNow
            };
            repoBook.Insert(book);
            return RedirectToAction("Index");
        }
    }
}
```

Using in a controller…part 2

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using GR.Data;
```

```
using GR.Web.Models;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.AspNetCore.Http;

namespace GR.Web.Controllers
{
    public class BookController : Controller
    {
        private IRepository<Author> repoAuthor;
        private IRepository<Book> repoBook;
        public BookController(IRepository<Author> repoAuthor, IRepository<Book> repoBook)
        {
            this.repoAuthor = repoAuthor;
            this.repoBook = repoBook;
        }

        public IActionResult Index()
        {
            List<BookListingViewModel> model = new List<BookListingViewModel>();
            repoBook.GetAll().ToList().ForEach(b =>
            {
                BookListingViewModel book = new BookListingViewModel
                {
                    Id = b.Id,
                    BookName = b.Name,
                    Publisher = b.Publisher,
                    ISBN=b.ISBN
                };
                Author author = repoAuthor.Get(b.AuthorId);
                book.AuthorName = $"{author.FirstName} {author.LastName}";
                model.Add(book);
            });
            return View("Index", model);
        }

        public PartialViewResult EditBook(long id)
        {
            EditBookViewModel model = new EditBookViewModel();
            model.Authors = repoAuthor.GetAll().Select(a => new SelectListItem
            {
                Text = $"{a.FirstName} {a.LastName}",
                Value = a.Id.ToString()
            }).ToList();
            Book book = repoBook.Get(id);
            if(book != null)
            {
                model.BookName = book.Name;
                model.ISBN = book.ISBN;
                model.Publisher = book.Publisher;
                model.AuthorId = book.AuthorId;
            }
            return PartialView("_EditBook",model);
        }
        [HttpPost]
        public ActionResult EditBook(long id, EditBookViewModel model)
        {
            Book book = repoBook.Get(id);
            if (book != null)
```

```
            {
                book.Name = model.BookName;
                book.ISBN = model.ISBN;
                book.Publisher = model.Publisher;
                book.AuthorId = model.AuthorId;
                book.IPAddress =
Request.HttpContext.Connection.RemoteIpAddress.ToString();
                book.ModifiedDate = DateTime.UtcNow;
                repoBook.Update(book);
            }
            return RedirectToAction("Index");
        }
        [HttpGet]
        public PartialViewResult DeleteBook(long id)
        {
            Book book = repoBook.Get(id);
            return PartialView("_DeleteBook",book?.Name);
        }
        [HttpPost]
        public ActionResult DeleteBook(long id, FormCollection form)
        {
            Book book = repoBook.Get(id);
            if(book != null)
            {
                repoBook.Delete(book);
            }
            return RedirectToAction("Index");
        }

    }
}
```