Repos/.netcore/DropDownDB\_Core – this demo is working

https://www.c-sharpcorner.com/article/binding-dropdown-list-with-database-in-asp-net-core-mvc/#:~:text=For%20adding%20model%2C%20just%20right,going%20to%20inherit%20DbContext%20class.

DropDpwnList creation – by getting data from database – ASP.NET Core MVC:

1. **Installing Package for Entity framework core From NuGet -** Microsoft.EntityFrameworkCore.SqlServer
2. **Adding Connection string and Setting up DbContext -** {

"ConnectionStrings": {

"LibConStr": "Data Source=(localdb)\\MSSQLLocalDB;Database=Library;Integrated security=SSPI;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

1. Add Model class – Book.cs, BookContext.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace DropDownDB\_Core.Models

{

public class Book

{

[Key]

public int bookid { get; set; }

public string booknm { get; set; }

public string authnm { get; set; }

public int price { get; set; }

public int qty { get; set; }

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

namespace DropDownDB\_Core.Models

{

public class BookContext:DbContext

{

public BookContext(DbContextOptions<BookContext> options) : base(options)

{

}

public DbSet<Book> Books { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Book>().ToTable("Book");

}

}

}

1. we are going to add new Service in Startup.cs class to inject dependency.

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using DropDownDB\_Core.Models;

using Microsoft.EntityFrameworkCore.SqlServer;

namespace DropDownDB\_Core

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

//Fetching Connection string from APPSETTINGS.JSON

var ConnectionString = Configuration.GetConnectionString("LibConStr");

//Entity Framework

IServiceCollection serviceCollections = services.AddDbContext<BookContext>(options => options.UseSqlServer(ConnectionString));

// services.AddDatabaseDeveloperPageExceptionFilter();

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Book}/{action=Index}/{id?}");

});

}

}

}

1. Adding controller

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using DropDownDB\_Core.Models;

using System.Data.Entity;

using System.Data.Entity.Infrastructure;

namespace DropDownDB\_Core.Controllers

{

public class BookController : Controller

{

private readonly BookContext \_dbContext;

public BookController(BookContext dbContext)

{

\_dbContext = dbContext;

}

public IActionResult Index()

{

return View(\_dbContext.Books.ToList());

}

[HttpGet]

public IActionResult Create()

{

List<Book> bkls = new List<Book>();

bkls = (from b in \_dbContext.Books

select b).ToList();

bkls.Insert(0, new Book { bookid = 0, booknm = "select" });

ViewBag.listofbooks = bkls;

return View();

}

[HttpPost]

public IActionResult Create([Bind("bookid,booknm,authnm,price,qty")] Book bks)

{

try

{

if (ModelState.IsValid)

{

\_dbContext.Add(bks);

\_dbContext.SaveChanges();

return RedirectToAction("Index");

}

}

catch (DbUpdateException /\* ex \*/)

{

ModelState.AddModelError("", "Unable to save changes. " +

"Try again, and if the problem persists " +

"see your system administrator.");

}

return View(bks);

}

}

}

1. Adding view:

Create.cshtml

@model DropDownDB\_Core.Models.Book

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>Book</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form **asp-action**="Create">

<div **asp-validation-summary**="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="bookid" class="control-label"></label>

<input **asp-for**="bookid" class="form-control" />

<span **asp-validation-for**="bookid" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="booknm" class="control-label"></label>

<select **asp-for**="booknm" class="form-control"

**asp-items**="@(new SelectList (@ViewBag.listofbooks,"booknm","booknm"))"></select>

<span **asp-validation-for**="booknm" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="authnm" class="control-label"></label>

<input **asp-for**="authnm" class="form-control" />

<span **asp-validation-for**="authnm" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="price" class="control-label"></label>

<input **asp-for**="price" class="form-control" />

<span **asp-validation-for**="price" class="text-danger"></span>

</div>

<div class="form-group">

<label **asp-for**="qty" class="control-label"></label>

<input **asp-for**="qty" class="form-control" />

<span **asp-validation-for**="qty" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a **asp-action**="Index">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

1. Goto appsettings.json and establish connectionstring
2. Install Microsoft.EntityFrameworkCore
3. Install Microsoft.EntityFrameworkCore.SqlServer
4. Install the above two in startup.cs + using <projectname>.Models
5. As usual in controller add models folder, controller, views