Contents

[Adding Database 2](#_Toc123868046)

[Data Annotations 3](#_Toc123868047)

[Creating Foreign Key Implementation 4](#_Toc123868048)

[Routing 6](#_Toc123868049)

[Layout Pages 7](#_Toc123868050)

## Adding Database

1. Create project without authentication
2. Add folder called Data
3. Install efcore tools and efcore sqlserver
4. View installed packages by right clicking project->Edit Project File
5. Add the following code for packages if working offline

<ItemGroup>

    <PackageReference Include="Codehaks.Pagination" Version="1.0.0" />

    <PackageReference Include="FluentAssertions" Version="6.8.0" />

    <PackageReference Include="Galactic.ActiveDirectory" Version="1.3.0.499" />

    <PackageReference Include="Microsoft.AspNet.Identity.Core" Version="2.2.3" />

    <PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="6.0.11" />

    <PackageReference Include="Microsoft.AspNetCore.Identity.UI" Version="6.0.10" />

    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="6.0.12" />

    <PackageReference Include="Microsoft.EntityFrameworkCore.Tools" Version="6.0.12">

      <PrivateAssets>all</PrivateAssets>

      <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>

    </PackageReference>

    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration" Version="6.0.11" />

    <PackageReference Include="Microsoft.VisualStudio.Web.CodeGeneration.Design" Version="6.0.10" />

    <PackageReference Include="System.Configuration.ConfigurationManager" Version="7.0.0" />

    <PackageReference Include="System.DirectoryServices" Version="7.0.0" />

    <PackageReference Include="System.DirectoryServices.AccountManagement" Version="7.0.0" />

  </ItemGroup>

1. Add class with name of context: ApplicationDbContext, and inherits from IdentityDbContext
2. Add constructor

public class ApplicationDbContext : IdentityDbContext

{

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

    {

    }

}

1. Add to Program.cs:

var connectionString = builder.Configuration.GetConnectionString("TIMBContextConnection") ?? throw new InvalidOperationException("Connection string 'TIMBContextConnection' not found.");

builder.Services.AddDbContext<ZltInventoryContext>(options =>options.UseSqlServer(connectionString));

1. Add to appsettings.json:

"ConnectionStrings":{"ZltContextConnection":"Server=(localdb)\\mssqllocaldb;Database=Zlt;Trusted\_Connection=True;MultipleActiveResultSets=true"},

1. Create a model

using System.ComponentModel.DataAnnotations;

namespace ZltInventory.Models

{

    public class Page

    {

        public int Id { get; set; }

        [Required]

        public string Title { get; set; }

        [Required]

        public string Slug { get; set; }

        [Required]

        public string Content { get; set; }

        public int Sorting { get; set; }

    }

}

1. Add Dbset property to context class
2. public DbSet<Page> Pages { get; set; }

### Adding Seed Data to Database

1. Create model SeedData

using Microsoft.EntityFrameworkCore;

using ZltInventory.Infrastructure;

namespace ZltInventory.Models

{

    public class SeedData

    {

        public static void Initialize(IServiceProvider serviceProvider)

        {

            using(var context = new ZltInventoryContext

                (serviceProvider.GetRequiredService<DbContextOptions<ZltInventoryContext>>()))

            {

                if (context.Pages.Any())

                {

                    return;

                }

                context.Pages.AddRange(

                    new Page

                    {

                        Title = "Home",

                        Slug = "home",

                        Content = "home page",

                        Sorting = 0

                    },

                    new Page

                    {

                        Title = "Devices",

                        Slug = "devices",

                        Content = "devices page",

                        Sorting = 0

                    }

                    );

                context.SaveChanges();

            }

        }

    }

}

1. In program.cs add these lines under var app = builder.Build();

using (var scope = app.Services.CreateScope())

{

    var services = scope.ServiceProvider;

    try

    {

        SeedData.Initialize(services);

    }

    catch (Exception ex)

    {

        throw ex;

    }

}

Creating Admin Area

1. Create new Folder called Areas
2. Right click on the folder and add area called admin

## Creating a select dropdown list

In the Controller

public IActionResult Create()

{

ViewBag.CustomerId = new SelectList(\_context.Customer.OrderBy(x => x.Name), "Id", "Name");

return View();

}

public async Task<IActionResult> Create([Bind("Id,BaleNumber,Mass,PeelTime,ClientName")] BalePeeler balePeeler)

{

    ViewBag.CustomerId = new SelectList(\_context.Customer.OrderBy(x => x.Name), "Id", "Name");

    if (ModelState.IsValid)

    {

        \_context.Add(balePeeler);

        await \_context.SaveChangesAsync();

        return RedirectToAction(nameof(Index));

    }

    return View(balePeeler);

}

## Data Annotations

1. Display Name

[Display(Name = "Brand")]

public int BrandId { get; set; }

2. Minimum Length

[MinLength(2, ErrorMessage = "Minimum length is 2")]

3. Maximum Length

[MaxLength(10, ErrorMessage = " Maximum length is 10")]

4. Regular Expression/Default characters

[RegularExpression(@"^[a-zA-Z-]+$", ErrorMessage = "Only Letters are allowed")]

5. Foreign Key

[ForeignKey("BrandId")]

6. Range

[Range(1, int.MaxValue, ErrorMessage = "You must choose a category")]

public int CategoryId { get; set; }

7. Not Mapped

[NotMapped]

public IFormFile? ImageUpload { get; set; }

8. File Extension

[FileExtension]

public IFormFile? ImageUpload { get; set; }

## Creating Foreign Key Implementation

Create Model 1

namespace Zlt.Models

{

    public class Customer

    {

        public int Id { get; set; }

        public string Name { get; set; }

    }

}

Create Model 2

using System.ComponentModel;

using System.ComponentModel.DataAnnotations.Schema;

namespace Zlt.Models

{

    public class BalePeeler

    {

        public int Id { get; set; }

        [DisplayName("Bale Number")]

        public string BaleNumber { get; set; }

        public int Mass { get; set; }

        [DisplayName("Peel Time")]

        public DateTime PeelTime { get; set; }

        //Foreign Key Implementation

        [DisplayName("Customer Name")]

        public int CustomerId { get; set; }

        [ForeignKey("ClientId")]

        public Customer Customer { get; set; }

    }

}

In the Controller

 // GET: BalePeelers/Create

        public IActionResult Create()

        {

            ViewBag.CustomerId = new SelectList(\_context.Customer.OrderBy(x => x.Name), "Id", "Name");

            return View();

        }

        // POST: BalePeelers/Create

        // To protect from overposting attacks, enable the specific properties you want to bind to.

        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

        [HttpPost]

        [ValidateAntiForgeryToken]

        public async Task<IActionResult> Create([Bind("Id,BaleNumber,Mass,PeelTime,ClientName")] BalePeeler balePeeler)

        {

            ViewBag.CustomerId = new SelectList(\_context.Customer.OrderBy(x => x.Name), "Id", "Name");

            if (ModelState.IsValid)

            {

                \_context.Add(balePeeler);

                await \_context.SaveChangesAsync();

                return RedirectToAction(nameof(Index));

            }

            return View(balePeeler);

        }

In the Create View

<div class="form-group">

    <label asp-for="CustomerId" class="control-label"></label>

    <select asp-for="CustomerId" asp-items="@ViewBag.CustomerId" class="form-control">

        <option value = 0>Choose Category</option>

    </select>

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

</div>

## Routing

Done in the program.cs file

app.MapControllerRoute(

    name: "default",

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

## Layout Pages

You can create many layout pages and then specify the layout page to use in a view in the \_ViewStart.cshtml

@{

    Layout = "\_Layout";

}