Rigit Zoo - Full Guide

Step 1: Create the Project

Create a new project:

```
ASP.NET Core Web App (Model-View-Controller)
```

Project Settings

```
Project Name: RigitZoo
Framework: .NET 8+
Authentication: Individual Accounts
Configure HTTPS:
```

Step 2: Run the Project Once

Press:

```
F5
```

Confirm that:

- The site runs
- Register/Login appears
- You can create an account

  Fix any errors before continuing.

Step 3: Complete `appsettings.json`

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "ConnectionStrings": {
    "DefaultConnection": "Server=name;Database=dbname;TrustServerCertificate=True;Trusted_Co
  },
  "AllowedHosts": "*"
}
```

Step 4: Create Animal Model

Create file:

`Models/Animal.cs`

```
using System.ComponentModel.DataAnnotations;

namespace RigitZoo.Models
{
    public class Animal
    {
        public int Id { get; set; }

        [Required]
        public required string Name { get; set; }

        [Required]
        public required string Description { get; set; }

        public string? ImageUrl { get; set; }
    }
}
```

> **Note:** If you see "Non-nullable property must contain a non-null value", use `required` or make properties nullable.

Step 5: Add Animal to DbContext

Open `Data/ApplicationDbContext.cs` and add:

```
using RigitZoo.Models;
```

Inside the class:

```
public DbSet<Animal> Animals { get; set; }
```

Full example:

```
public class ApplicationDbContext : IdentityDbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options) { }

    public DbSet<Animal> Animals { get; set; }
}
```

Step 6: Fix `Program.cs`

All services must be registered **before** `builder.Build()`.

```
using Microsoft.EntityFrameworkCore;
using Microsoft.AspNetCore.Identity;
using RigitZoo.Data;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

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

builder.Services.AddDefaultIdentity<IdentityUser>()
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<ApplicationDbContext>();

var app = builder.Build();

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.MapRazorPages();

app.Run();
```

Common Errors:

- DbContext registered after Build() → move all `builder.Services.Add...` above Build()
- Connection string mismatch → check `GetConnectionString("DefaultConnection")`

Step 7: Install Required Packages

Use Package Manager Console:

```
Install-Package Microsoft.AspNetCore.Identity.EntityFrameworkCore
Install-Package Microsoft.AspNetCore.Identity.UI
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore.Tools
```

> **UIFrameworkAttribute missing** → install `Microsoft.AspNetCore.Identity.UI`

Step 8: Create Database Migration

```
Add-Migration AddAnimals
Update-Database
```

Should create tables: `AspNetUsers`, `AspNetRoles`, `AspNetUserRoles`, `Animals`

> **Tip:** If migration fails, rebuild the solution.

Step 9: Scaffold Animal Controller + Views

Right click:

```
Controllers → Add → Controller
```

Choose:

```
MVC Controller with views, using Entity Framework
```

Settings:

```
Model: Animal
DbContext: ApplicationDbContext
```

Click Add.

Step 10: Add Navbar Link

Edit `_Layout.cshtml`:

```
<li class="nav-item">
    <a class="nav-link" asp-controller="Animals" asp-action="Index">Animals</a>
</li>
```

Step 11: Scaffold Identity UI

Right click project:

`Add → New Scaffolded Item → Identity`

Select:

```
Account/Login
Account/Register
Manage/*
```

Data context: `ApplicationDbContext`

Step 12: Setup Login/Register Navbar

Add to `_Layout.cshtml`:

```
<ul class="navbar-nav">
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Login</a
    </li>
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Regis
    </li>
</ul>
```

Step 13: Add Roles Support

Ensure `Program.cs` contains:

```
.AddRoles<IdentityRole>()
```

Step 14: Seed Admin Role + User

Before `app.Run()`, add:

```
using (var scope = app.Services.CreateScope())
{
    var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>()
    var userManager = scope.ServiceProvider.GetRequiredService<UserManager<IdentityUser>>()

    if (!await roleManager.RoleExistsAsync("Admin"))
        await roleManager.CreateAsync(new IdentityRole("Admin"));
    var email = "admin@rigitzoo.com";
    var user = await userManager.FindByEmailAsync(email);

    if (user == null)
    {
        user = new IdentityUser { UserName = email, Email = email, EmailConfirmed = true };
        await userManager.CreateAsync(user, "Admin123!");
    }

    await userManager.AddToRoleAsync(user, "Admin");
}
```

Step 15: Hide Admin Buttons in Views

Wrap buttons in Razor:

```
@if (User.IsInRole("Admin"))
{
    <a asp-action="Edit">Edit</a>
}
```

Step 16: Error Checks / Common Fixes

- Build failed → check **Error List**
- Namespace mismatch → ensure `RigitZoo.Models` and `RigitZoo.Data` match
- Identity UI missing → install `Microsoft.AspNetCore.Identity.UI`
- Identity pages 404 → ensure `app.MapRazorPages();`