User Registration

Sunday, November 26, 2023 9:21 PM

Regsitration.html

```
@page
@model RegistrationModel
@{
}
<div class="container border" style="padding:20px">
    <form method="post">
        <div class="text-danger" asp-validation-summary=All></div>
        <div class="mb-3 row">
            <div class="col-2">
                <label asp-for="RegistrationViewModel.Email"></label>
            </div>
            <div class="col-5">
                <input type="text" asp-for="RegistrationViewModel.Email" class="form-control" />
                <span class="text-danger" asp-validation-for="RegistrationViewModel.Email"></span>
            </div>
        </div>

        <div class="mb-3 row">
            <div class="col-2">
                <label asp-for="RegistrationViewModel.Password"></label>
            </div>
            <div class="col-5">
                <input type="password" asp-for="RegistrationViewModel.Password" class="form-control" />
                <span class="text-danger" asp-validation-for="RegistrationViewModel.Email"></span>

            </div>
        </div>

        <div class="mb-3 row">
            <div class="col-2">
                <input type="submit" class="btn btn-primary" value="Register" />
            </div>
            <div class="col-5">
            </div>
        </div>


    </form>
</div>
```

Registration.html.cs

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.ModelBinding;
using Microsoft.AspNetCore.Mvc.RazorPages;
using System.ComponentModel.DataAnnotations;

namespace WebApp.Pages.Account
{
    public class RegistrationModel : PageModel
    {

        public RegistrationModel(UserManager<IdentityUser> userManager)
        {
            UserManager = userManager;
        }
        [BindProperty]
        public RegistrationViewModel RegistrationViewModel { get; set; } = new
RegistrationViewModel();
        public UserManager<IdentityUser> UserManager { get; }

        public void OnGet()
        {
        }

        public async Task<IActionResult> OnPostAsync()
        {
```

```
            if (!ModelState.IsValid) return Page();

            var user = new IdentityUser()
            {
                Email = RegistrationViewModel.Email,
                UserName = RegistrationViewModel.Email
            };

            var result = await UserManager.CreateAsync(user,
RegistrationViewModel.Password);

            if (result.Succeeded)
            {
                return RedirectToPage("/Account/Login");
            }
            else
            {
                foreach (var error in result.Errors)
                {
                    ModelState.AddModelError("Register", error.Description);
                }

                return Page();
            }
        }
    }

    public class RegistrationViewModel
    {

        [Required]
        [EmailAddress(ErrorMessage = "Invalid Email Address.")]
        public string Email { get; set; } = string.Empty;

        [Required]
        [DataType(dataType: DataType.Password)]
        public string Password { get; set; } = string.Empty;
    }
}
```

Program.cs

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

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorPages();

builder.Services.AddDbContext<ApplicationDbContext>(options => {
    options.UseSqlServer(builder.Configuration.GetConnectionString("Default"));
});


builder.Services.AddIdentity<IdentityUser, IdentityRole>(options => {
    options.Password.RequiredLength = 8;
    options.Password.RequireLowercase = true;
    options.Password.RequireUppercase = true;
    options.Lockout.MaxFailedAccessAttempts= 5; ;
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(15);
    options.User.RequireUniqueEmail = true;

})
  .AddEntityFrameworkStores<ApplicationDbContext>();

builder.Services.ConfigureApplicationCookie(options => {
    options.LoginPath = "/Account/Login";
    options.AccessDeniedPath = "/Account/AccessDenied";
});

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/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.MapRazorPages();

app.Run();
```