05 Create Product Form

Goal

Create a form that submits a new Product to the database.
This test the full flow. Form, validation, saving, redirect.

THINGS TO HAVE

A GET method to show the form
A POST method to process the form
Model validation using ModelState
Saving to database using EF Core
Redirect after success

Step 1. Create a DTO for the form

Why use a DTO
It stops users posting fields you do not want them to set.
It helps with file upload because Product model is for database, not for upload handling.

Create a folder named DTOs.
Create ProductCreateDTO.cs.
Use a clear name like ProductCreateDTO.

```
1. using System.ComponentModel.DataAnnotations;
2. using Microsoft.AspNetCore.Http;
3.
4. namespace RTWebApplication.DTOs
5. {
6.     public class ProductCreateDTO
7.     {
8.         [Required]
9.         [MaxLength(100)]
10.        public string Name { get; set; } = "";
11.
12.        [Required]
13.        [MaxLength(500)]
14.        public string Description { get; set; } = "";
15.
16.        [Range(0.01, 1000000)]
17.        public decimal Price { get; set; }
18.
19.        public IFormFile? ImageFile { get; set; }
20.    }
21. }
```

Explanation

This DTO matches what the user fills in.
IFormFile holds the uploaded image.

Step 2. Add Create actions in controller

Add actions to AdminProductController.

```
 1.   [HttpGet]
 2.   public IActionResult Create()
 3.   {
 4.       return View();
 5.   }
 6.
 7.   [HttpPost]
 8.   public IActionResult Create(GEProductDTO gEProductDTO)
 9.   {
10.       if (gEProductDTO.ImageFile == null)
11.       {
12.           ModelState.AddModelError("ImageFile", "Product Image is required");
13.           return View(gEProductDTO);
14.       }
15.
16.       if (!ModelState.IsValid)
17.       {
18.           return View(gEProductDTO);
19.       }
20.
21.       string folderpath = Path.Combine(webHostEnvironment.WebRootPath, "geproducts");
22.       string fullFilePath = Path.Combine(folderpath, gEProductDTO.ImageFile.FileName);
23.
24.       using var stream = new FileStream(fullFilePath, FileMode.Create);
25.       gEProductDTO.ImageFile.CopyTo(stream);
26.
27.       var gEProduct = new GEProduct
28.       {
29.           Name = gEProductDTO.Name,
30.           Brand = gEProductDTO.Brand,
31.           Category = gEProductDTO.Category,
32.           Description = gEProductDTO.Description,
33.           Price = gEProductDTO.Price,
34.           ImageFileName = gEProductDTO.ImageFile.FileName,
35.           CreatedAt = DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss")
36.       };
37.
38.       context.GEProducts.Add(gEProduct);
39.       context.SaveChanges();
40.       return RedirectToAction("Index", "Product");
41.   }
42.
```

Explanation in detail

1. GET Create shows the form page.
2. POST Create receives the form data.
3. ValidateAntiForgeryToken protects against CSRF.

4. ModelState.IsValid checks DataAnnotations on the DTO.
5. Files are stored in wwwroot/geproducts so the browser can access them.
6. Product object is created from DTO values, then saved to database.
5. RedirectToAction avoids resubmitting the form on refresh.

Step 3. Create the Create view

RIGHT CLICK ON view from return View() in the GET create method

Then click on Add view and give it the same name as the method which will be CREATE

Then get form code snippet from boostrap

@model RTWebApplication.DTOs.ProductCreateDTO

```
<h2>Create Product</h2>

<form asp-action="Create" method="post" enctype="multipart/form-data">
    <div>
        <label>Name</label>
        <input asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>

    <div>
        <label>Description</label>
        <textarea asp-for="Description"></textarea>
        <span asp-validation-for="Description"></span>
    </div>

    <div>
        <label>Price</label>
        <input asp-for="Price" />
        <span asp-validation-for="Price"></span>
    </div>

    <div>
        <label>Image</label>
        <input asp-for="ImageFile" type="file" />
    </div>

    <button type="submit">Save</button>
</form>

```

Explanation

enctype multipart/form-data is required for file uploads.
asp-for binds inputs to DTO properties.
asp-validation-for displays validation messages.

Step 4. Add Create link on Index page

In Index.cshtml inside Product folder in view add this above the table:

```
1. <a asp-action="Create">Create New Product</a>
2.
```