

Almacenar una imagen (o un archivo cualquiera subido al servidor)

1. Crear una aplicación web asp.net core MVC

Seleccionar la plantilla para crear una aplicación Web Asp.net Core como lo hace habitualmente.

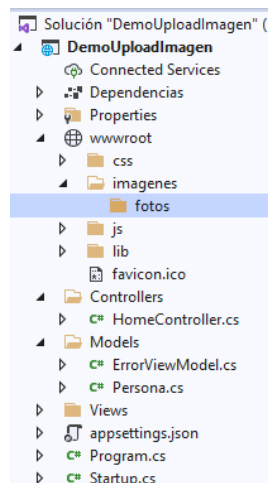
2. Crear el modelo

Crear en la carpeta Models una clase Persona con dos atributos, el nombre y el nombre de su Foto. Tomar en cuenta que esta es una demo, probablemente en el contexto de una aplicación con Entidades de dominio lo resolveríamos ubicando el objeto Persona en un proyecto Biblioteca de Clases por separado:

```
namespace DemoUploadImagen.Models
{
    public class Persona
    {
        public string Nombre { get; set; }
        public string NombreArchivoFoto { get; set; }
    }
}
```

3. Preparar la carpeta para almacenar las imágenes.

Crear bajo la carpeta **wwwroot** de la aplicación MVC un directorio **imagenes** que incluya un subdirectorio **fotos**. No es obligatorio almacenar en esta ubicación las imágenes, pero es la apropiada para los archivos estáticos (ej. Css, js, imágenes, etc.)



4. Preparar la aplicación para procesar archivos subidos desde la web.

Utilizaremos la interfaz IFormFile incluida en una petición HttpRequest de tipo POST para manejar la imagen subida. La interfaz (integrada en el namespace Microsoft.AspNetCore.Http) es:

```

public interface IFormFile
{
    string ContentType { get; }
    string ContentDisposition { get; }
    IDictionary Headers { get; }
    long Length { get; }
    string Name { get; }
    string FileName { get; }
    Stream OpenReadStream();
    void CopyTo(Stream target);
    Task CopyToAsync(Stream target, CancellationToken
cancellationToken = null);
}

```

En HomeController agregar el using al namespace que incluye IFormFile:

```
using Microsoft.AspNetCore.Http;
```

Incluir también el using para acceder al sistema de archivos:

```
using System.IO;
```

Para acceder a las variables de ambiente (por ejemplo la ruta física de wwwroot) se debe incluir:

```
using Microsoft.AspNetCore.Hosting;
```

Para utilizar el modelo Persona se debe incluir (si no está incluido):

```
using DemoUploadImagen.Models;
```

Para obtener la ruta física en que está ubicado wwwroot usaremos IWebHostEnvironment. Inyectaremos la dependencia en el constructor del controller:

```

public class HomeController : Controller
{
    private IWebHostEnvironment _environment;

    public HomeController(IWebHostEnvironment environment)
    {
        _environment = environment;
    }
}

```

5. Implementar los métodos para crear un objeto, setearle el nombre de la imagen y guardar la imagen en el servidor.

- a. Incluir 2 métodos Create, uno para desplegar el formulario de registro – que incluye el upload de un archivo – y el método HttpPost para recibir la información HTTP mediante post.

```
[HttpGet]
public IActionResult Create()
{
    return View();
}

[HttpPost]
public IActionResult Create(Persona p, IFormFile imagen)
{
    if (p == null || imagen == null || !ModelState.IsValid)
        return View();
    //ruta física donde está ubicada wwroot en el servidor

    if (GuardarImagen(imagen,p))
    {
        return RedirectToAction("Visualizar",p);
    }
    return View(p);
}
```

b. Implementar el método que guarda la imagen¹

```
private bool GuardarImagen(IFormFile imagen, Persona p)
{
    if (imagen == null || p == null)
        return false;

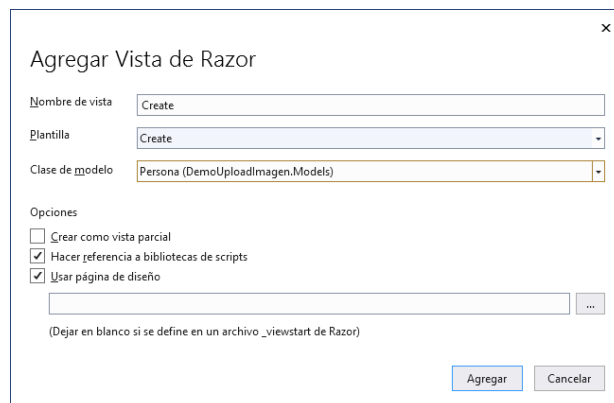
    // SUBIR LA IMAGEN
    string rutaFisicaWwwRoot = _environment.WebRootPath;
    //ruta donde se guardan las fotos de las personas
    string nombreImagen = imagen.FileName;
    string rutaFisicaFoto = Path.Combine
        (rutaFisicaWwwRoot, "imagenes", "fotos", nombreImagen);
    //FileStream permite manejar archivos
    try
    {
        //el método using libera los recursos del objeto FileStream al finalizar
        using (FileStream f = new FileStream(rutaFisicaFoto, FileMode.Create))
        {
            //si fueran archivos grandes o si fueran varios, deberíamos usar la versión
            //asincrónica de CopyTo, aquí no es necesario.
            //sería: await imagen.CopyToAsync (f);
            imagen.CopyTo(f);
        }
        //GUARDAR EL NOMBRE DE LA IMAGEN SUBIDA EN EL OBJETO
        p.NombreArchivoFoto = nombreImagen;
        return true;
    }
    catch (Exception ex)
    {
        return false;
    }
}
```

6. Preparar la vista para realizar el upload.

Utilizaremos un elemento html <input type = "file" />.

IMPORTANTE: El tag <form> debe incluir el atributo enctype="multipart/form-data" y method="post" de otro modo el archivo no subirá.

Agregar una vista Razor (no la vacía) con la plantilla Create y modelo Persona:



¹ Se podría incluir una clase que se ocupara del manejo del archivo subido para no incluir este código en el controller. Se optó por esta solución por su simplicidad. También se podría mejorar utilizando métodos de la clase System.IO.File, para crear las carpetas si no existen, verificar las excepciones producidas, etc.

Cambiar en la vista los tags form y el input para el archivo (líneas recuadradas):

```
@model DemoUploadImagen.Models.Persona
@{
    ViewData["Title"] = "Create";
}
<h4>Crear persona </h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form asp-action="Create" enctype="multipart/form-data">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="Nombre" class="control-label"></label>
                <input asp-for="Nombre" class="form-control" />
                <span asp-validation-for="Nombre" class="text-danger"></span>
            </div>
            <div class="form-group">
                <label for="imagen" class="control-label"></label>
                <input type="file" name="imagen" class="form-control" />
                <span asp-validation-for="imagen" 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">Lista de personas</a>
</div>

@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

7. IMPLEMENTAR LA VISTA PARA VISUALIZAR EL OBJETO CREADO Y SU IMAGEN SUBIDA.

Agregar una vista llamada Visualizar usando la plantilla Edit y como modelo utilizar un objeto de la clase Persona.

En esa vista cambiar el tag generado para incluir el nombre de la imagen, sustituyéndolo por el tag .

```
@model DemoUploadImagen.Models.Persona
```

```
@{  
    ViewData["Title"] = "Visualizar";  
}
```

```
<h1>Visualizar</h1>
```

```
<div>
```

```
    <h4>Persona</h4>
```

```
    <hr />
```

```
    <dl class="row">
```

```
        <dt class = "col-sm-2">
```

```
            @Html.DisplayNameFor(model => model.Nombre)
```

```
        </dt>
```

```
        <dd class = "col-sm-10">
```

```
            @Html.DisplayFor(model => model.Nombre)
```

```
        </dd>
```

```
        <dt class = "col-sm-2">
```

```
            Foto
```

```
        </dt>
```

```
        <dd class = "col-sm-10">
```

```
            
```

```
        </dd>
```

```
    </dl>
```

```
</div>
```

```
<div>
```

```
    <a asp-action="Index">Inicio</a>
```

```
</div>
```

8. IMPLEMENTAR LA VISTA INDEX

La vista Index solamente contendrá un link que lleva a /home/crea y despliega el form para crear el objeto.

```
@{  
    ViewData["Title"] = "View";  
}
```

```
<h1>Inicio</h1>
```

```
<a asp-action="Create" asp-controller="Home" class="btn btn-primary">Crear persona</a>
```

9. PROBAR LA APLICACIÓN

Ejecutar la aplicación y presionar el link para crear un objeto Persona.

Ingresar el nombre, y seleccionar cualquier archivo de imagen del disco duro y presionar submit.

Ese archivo se copiará a la carpeta wwwroot/imágenes/fotos y se guardará su nombre en el objeto Persona.

Inmediatamente redirigirá a visualizar el objeto con su imagen.

Nota: Como está utilizando la misma máquina como servidor (IISExpress de desarrollo) y como cliente (navegador) puede ocurrir que su navegador impida la subida del archivo desde localhost. Pruebe cambiando de navegador en la barra del editor de Visual Studio:

