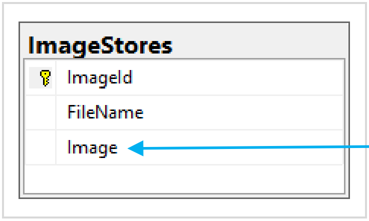
Using .NET To Store Images in SQL Server

Create a new database and call it `ImageStoreDB`, then run the following script to create the image table:

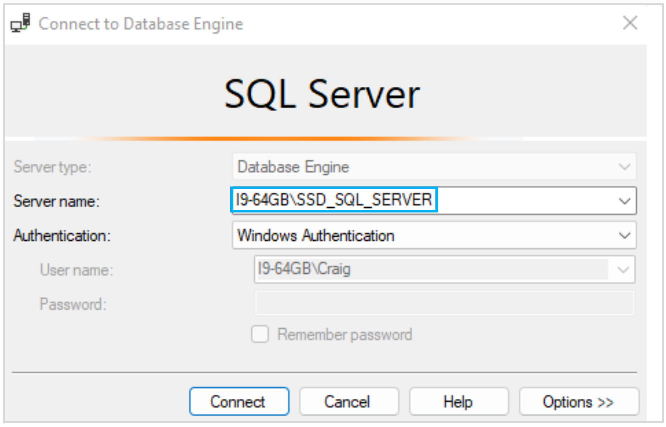
```
DROP TABLE IF EXISTS ImageStores;

CREATE TABLE ImageStores( ImageId    INT IDENTITY(1,1) NOT NULL
                        , [FileName] VARCHAR(20)       NOT NULL
                        , [Image]    VARBINARY(MAX)    NOT NULL
     , CONSTRAINT PK_ImageStores PRIMARY KEY CLUSTERED ( ImageId ASC ));
```

The `ImageStores` entity consists of an `INT IDENTITY` `Id` column, a `VARCHAR(20)` `FileName` column and a `VARBINARY(MAX)` `Image` column.



The `Image` column stores the bite array (`byte[]`) created in .Net.

Obtain your database server name from the dialog that appears when launching **SQL Server Management Studio**.



Server type: Database Engine
Server name: I9-64GB\SSD_SQL_SERVER
Authentication: Windows Authentication
User name: I9-64GB\Craig
Password:
Remember password

Visual Studio Web Application

Create a new `.NET MVC C#` web application then open the *Package Manager Console* by clicking:

Tools→NuGet Package Manager→Package Manager Console

Run the following script in the *Package Manager Console* to enable model/class creation which is scaffolded on the `ImageStoreDB` database.

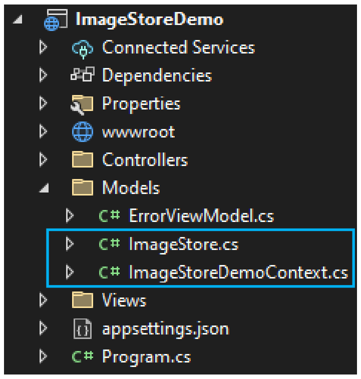
```
Install-Package Microsoft.EntityFrameworkCore.Tools
Install-Package Microsoft.EntityFrameworkCore.SqlServer
Install-Package Microsoft.EntityFrameworkCore
```

Next in the **Package Manager Console**, build the Entity classes by running the following command.

Important: Be sure to update your server name and double check the database name.

```
Scaffold-DbContext "Server=your server name Database=ImageStoreDB;
Trusted_Connection=True; TrustServerCertificate=True"
Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models
```

When you finish running the script you will notice the **Models** folder now contains the new context file and the ImageStore entity class.



Database Context

The `ImageStoreDbContext` class inherits from .Net's `DbContext` class and contains information about the database connection and schema. This class is referenced whenever a query is made on the database.

Connection String

Add the following connection string to the appsettings.json file.

Important: Be sure to update your server name and double check the database name.

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

Remove the following connection string from the context file.

```
~~protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)~~
~~=> optionsBuilder.UseSqlServer("Server=CAS-Y58388\\SQLSERVER2019;~~
~~Database=BridgeExampleDb; Trusted_Connection=True; TrustServerCertificate=True");~~
```

Add the following code to the `Program.cs` file to allow dependency injection for database access.

```
builder.Services.AddControllersWithViews();

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");
builder.Services.AddDbContext<ImageStoreDbContext>(options =>
        options.UseSqlServer(connectionString));

var app = builder.Build();
```

New File Upload Model

Create new file in the `Models` folder and name it `UploadModel.cs`. Add the following contents to the file.

```
using System.ComponentModel.DataAnnotations;

namespace ImageStoreDemo.Models
{
    public class UploadModel
    {
        [Required(ErrorMessage = "Please select a file.")]
        public IFormFile ImageFile { get; set; }
    }
}
```

Home Controller

Replace the `HomeController.cs` file contents with the following code.

```
using ImageStoreDemo.Models;
using Microsoft.AspNetCore.Mvc;

namespace ImageStoreDemo.Controllers
{
    public class HomeController : Controller
    {
        private readonly ImageStoreDbContext _db;

        public HomeController(ImageStoreDbContext db)
        {
            _db = db;
        }

        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Images()
        {
            IEnumerable<ImageStore> images = _db.ImageStores;
            return View(images);
        }

        public IActionResult SaveImage()
        {
            return View();
        }

        [HttpPost]
        public async Task<IActionResult> SaveImage(UploadModel uploadModel)
        {
            if (ModelState.IsValid)
            {
                if (uploadModel.ImageFile != null && uploadModel.ImageFile.Length > 0)
                {
                    string contentType = uploadModel.ImageFile.ContentType;

                    if (contentType == "image/png" ||
                        contentType == "image/jpeg" ||
                        contentType == "image/jpg")
                    {
                        try
                        {
                            byte[] imageData;

                            using (var memoryStream = new MemoryStream())
                            {
                                await uploadModel.ImageFile.CopyToAsync(memoryStream);
                                imageData = memoryStream.ToArray();
                            }

                            var image = new ImageStore
                            {
                                FileName = Path.
                                   GetFileNameWithoutExtension(uploadModel.ImageFile.FileName),
                                Image = imageData
                            };

                            _db.ImageStores.Add(image);
                            await _db.SaveChangesAsync();

                            return RedirectToAction("Index", "Images");
                        }
                        catch (Exception ex)
                        {
                            ModelState.AddModelError("imageUpload"
                                                    , "An error occured uploading your image."
                                                    + " Please try again.");
                            System.Diagnostics.Debug.WriteLine(ex.Message);
                        }
                    }
                    else
                    {
                        ModelState.AddModelError("imageUpload", "Please upload a PNG, " +
                                                                "JPG, or JPEG file.");
                    }
                }
                else
                {
                    ModelState.AddModelError("imageUpload", "Please select an " +
                                                            " image to upload.");
                }
            }

            return View(uploadModel);
        }
    }
}
```

Home\Index View

Replace the contents in the `Home\Index` view with the following code.

```
@{
    ViewData["Title"] = "Home Page";
}

<div class="container my-5">
    <div class="jumbotron text-center">
        <h1 class="display-4" style="color:blue">Welcome to SecureShots</h1>
        <p class="lead">
            If you're looking for a secure and reliable way to
            store your images, you've come to the right place!
        </p>
        <hr class="my-4">
        <p class="text-center">
            Click the button below to navigate
            To the image file management area
        </p>
        <a class="btn btn-primary btn-lg" href="home/Images"
           role="button">All Images</a>&nbsp;&nbsp;&nbsp;
        <a class="btn btn-primary btn-lg" href="home/SaveImage"
           role="button">Save Image</a>
    </div>
</div>
```

Home\SaveImage View

Create an empty `Home\SaveImage` view and replace the contents with the following code.

```
@model ImageStoreDemo.Models.UploadModel

@{  ViewData["Title"] = "Images";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="container my-5">
    <div class="jumbotron text-center">
        <h2 class="display-4" style="color:blue">Save Image</h2><br />

        @if (ViewData["SuccessMessage"] != null)
        {
            <div class="alert alert-success" role="alert">
                @ViewData["SuccessMessage"]
            </div>
        }

        <div class="row">
            <div class="col-sm-6 offset-3">
                <form method="post" enctype="multipart/form-data">
                    <div class="form-group">
                        <input asp-for="ImageFile" class="form-control"
                               type="file" accept="image/*" />
                        <span asp-validation-for="ImageFile"
                              class="text-danger"></span>
                    </div><br /><br />
                    <button type="submit" class="uploadButton">
                        <img src="~/Images/SaveImage.png" alt="Upload Image" />
                    </button>
                </form>
            </div>
        </div>
    </div>
</div>
```

Home\Images View

Create an empty `Home\Images` view and replace the contents with the following code.

```
@model IEnumerable<ImageStoreDemo.Models.ImageStore>

@{  ViewData["Title"] = "Images";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<div class="container my-5">
    <div class="jumbotron text-center">
        <h2 class="display-4" style="color:blue">Images</h2><br />

        <div class="row">
            @foreach (var image in Model)
            {
                <div class="col-lg-4 col-md-6 col-sm-12 mb-4">
                    <div class="card">
                        <img src="data:image/jpeg;base64,
                                 @Convert.ToBase64String(image.Image)"
                             alt="@image.FileName" />
                        <div class="card-body">
                            <h5 class="card-title">@image.FileName</h5>
                        </div>
                    </div>
                </div>
            }
        </div>
    </div>
</div>
```

Image Folder

Create an Image folder in the `wwwroot` directory and copy in the following files:

- cassettes.jpeg
- saveImage.png

Layout View

Replace the `Home` and `Privacy` links with `Images` and `Save Image` links that redirect back the matching action methods in the Home controller. Also, replace the inner html on the `<title>` and `header/footer` `<a>` elements to match the application name.

Web Page Layouts

Landing Page:

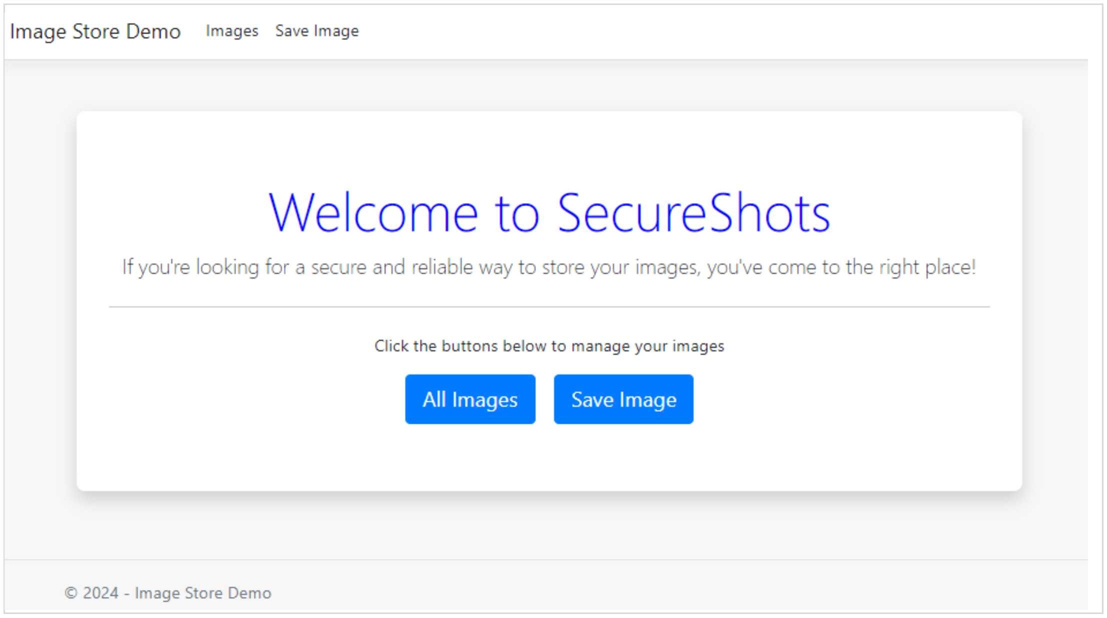


Image Store Demo Images Save Image

Welcome to SecureShots

If you're looking for a secure and reliable way to store your images, you've come to the right place!

Click the buttons below to manage your images

© 2024 - Image Store Demo

Save Image:

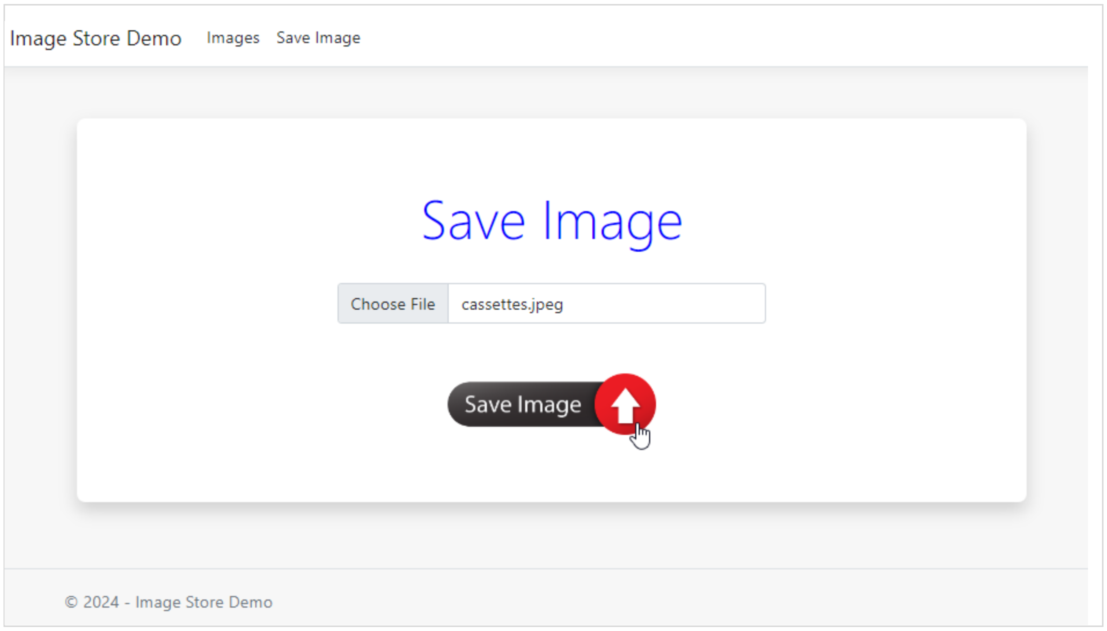


Image Store Demo Images Save Image

Save Image

© 2024 - Image Store Demo

Images:

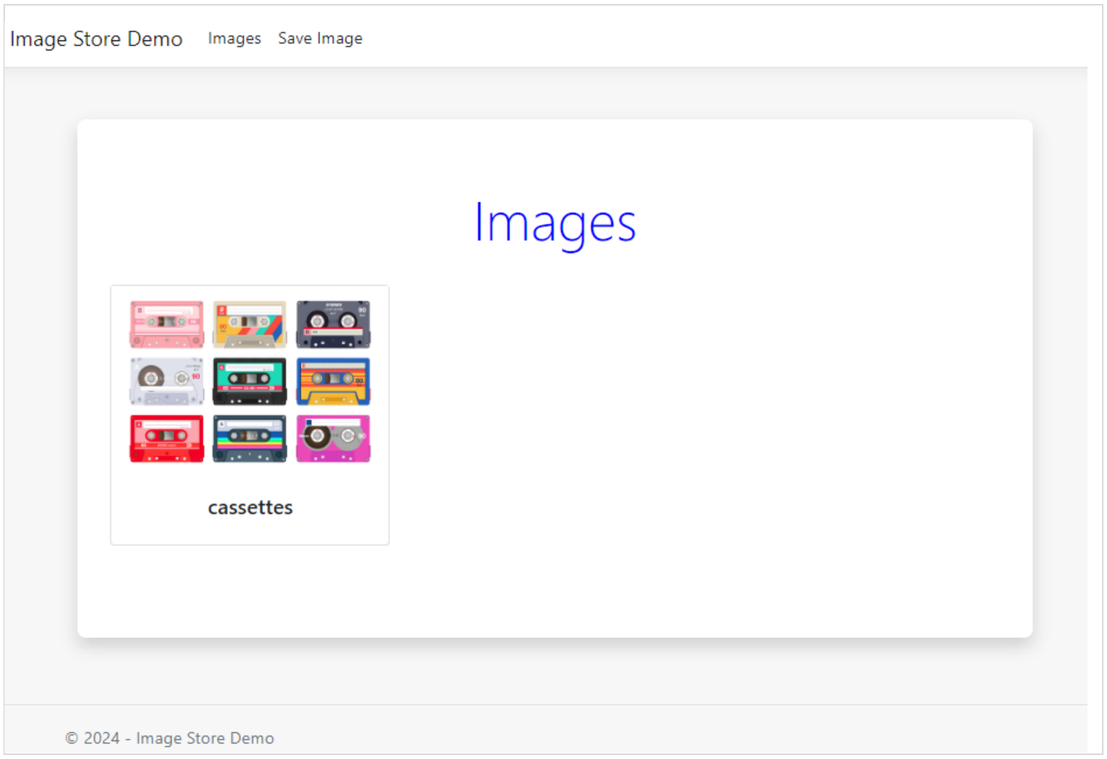


Image Store Demo Images Save Image

Images

cassettes

© 2024 - Image Store Demo