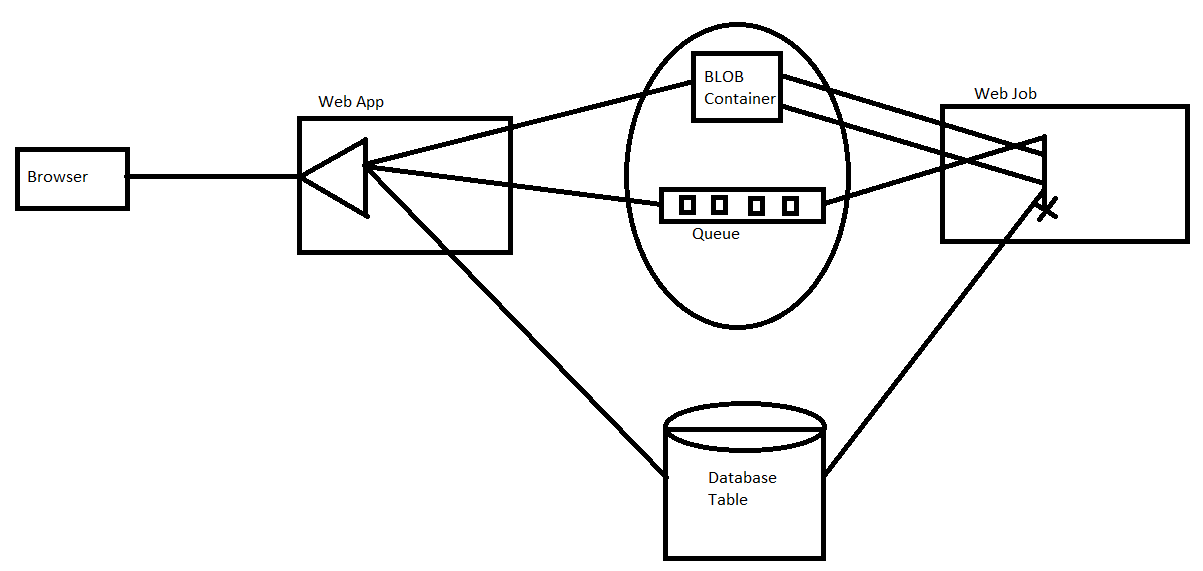
**Example for Creating Image Thumbnail Through Web Jobs (Record this)**



**Create a New ASP.NET MVC Application**

1. Add the MainWeb Application
   1. File 🡪 New 🡪 Project, Template = ASP.NET Core Web App (Model-View-Controller) 🡪 Next
   2. Project name= **EmployeeManagement**.**Web**, Solution Name="**EmployeeManagementSolution**" 🡪 Next
   3. Framework = .NET 6.0 (LTS) 🡪 Create

**Programming the Class Library Project**

1. Add New Project to Solution for sharing code between Web Application and Azure Function project.
   1. File 🡪 Add 🡪 New Project 🡪 Class Library 🡪 Name="**EmployeeManagement**.**SharedLibrary**" 🡪 Next 🡪 Create
   2. Add NuGet Package Reference to **Microsoft.EntityFrameworkCore**
2. Add the following to **EmployeeManagement.SharedLibrary**

**Employee.cs**

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

[Table("Employee")]

public class Employee

{

[Key]

public int Id { get; set; }

public string? EmpName { get; set; }

public decimal Salary { get; set; }

[StringLength(2083)]

[DisplayName("Full-size Image")]

public string? ImageURL { get; set; }

[StringLength(2083)]

[DisplayName("Thumbnail")]

public string? ThumbnailURL { get; set; }

}

public class **BlobInformation**

{

public Uri? BlobUri { get; set; }

public string BlobName

{

get

{

return BlobUri!.Segments[BlobUri.Segments.Length - 1];

}

}

public string BlobNameWithoutExtension

{

get

{

return Path.GetFileNameWithoutExtension(BlobName);

}

}

public int EmpId { get; set; }

}

**Programming Web Application**

1. In Web Application add reference to class library project "**EmployeeManagement**.**SharedLibrary**"
2. Using Scaffolding generate the Controller and DataContext class
   1. In **Solution Explorer**, right click on the Controllers Folder of Web Application Project 🡪 **Add** 🡪 **Controller** 🡪 Select **MVC Controller with views, using Entity Framework** 🡪 Add
   2. Select the Model class = "Employee", Data context class: Click +, New data context type: **EmployeeDataContext** 🡪 Add
   3. Go to Data\**EmployeeDataContext.cs and edit as below:**

public DbSet<EmployeeManagement.SharedLibrary.Employee> Employees { get; set; } = default!;

* 1. Move the Data\EmployeeDataContext to EmployeeManagement.SharedLibrary Project and delete from EmployeeManagement.Web Project.
  2. Change the Namespace of EmployeeDataContext from EmployeeManagement.Web.Data to EmployeeManagement.SharedLibrary
  3. Use Ctrl+H and do Replace All from EmployeeManagement.Web.Data to EmployeeManagement.SharedLibrary

1. Create the Initial Migration and Update the database.
   1. Open Tools 🡪 NuGet Package Manager 🡪 **Package Manager Console,** Default Project= **EmployeeManagement**.**Web**
   2. Run **Add-Migration InitialSchema** to scaffold a migration to create the initial set of tables for your model.
   3. Edit the Code in **Program.cs** as below

var app = builder.Build();

var context = app.Services.CreateScope().ServiceProvider.GetRequiredService<**EmployeeDataContext**>();

context.Database.Migrate();

1. Azure Portal 🡪 Create SQL Database and copy its connection string to be used in next step.
2. In WebApplication 🡪 appSettings.json , Edit the database connectionstring as required.

"ConnectionStrings": {

"EmployeeDataContext": "<sql database connectionstring>",

}

1. Build the Solution
2. In Views/Employees/**Create.cshtml** and **Edit.cshtml** replace the HTML for **ImageURL and ThumbnailUrl** with the following (FileUpload element)

<form **asp-action**="Create" method="post" enctype="multipart/form-data">

. . .

<div class="form-group">

<**label** **asp-for**="ImageURL" class="control-label"></**label**>

<input type="file" name="ImageUrl" class="form-control" />

~~<~~**~~span~~****~~asp-validation-for~~**~~="ImageURL" class="text-danger"></~~**~~span~~**~~>~~

</div>

~~<div class="form-group">~~

~~<~~**~~label~~****~~asp-for~~**~~="ThumbnailURL" class="control-label"></~~**~~label~~**~~>~~

~~<~~**~~input~~****~~asp-for~~**~~="ThumbnailURL" class="form-control" />~~

~~<~~**~~span~~****~~asp-validation-for~~**~~="ThumbnailURL" class="text-danger"></~~**~~span~~**~~>~~

~~</div>~~

1. Run and test the application by inserting Employee. **Note that ImageURL and ThumbNailURL are blank.**

**Storing Image as Blob in Container and posting a message in Azure Storage:**

1. Create **Storage Account** using Azure Portal.

Set Name = "dssemployeestorage" (all lowercase)

1. Edit the appsettings.json and add connectionstring of Storage Account.

"StorageConnectionString": "<Storage connection string>",

1. Tools 🡪 NuGet Package Manager 🡪 Manage NuGet Packages for Solution
   1. Azure.Storage.Queues
   2. Azure.Storage.Blobs
2. Update EmployeesController Construtor as below

private readonly IConfiguration \_config;

public EmployeesController(EmployeeDataContext context, IConfiguration config)

{

\_config = config;

\_context = context;

}

1. Update EmployeesController 🡪 **Create** method to as below.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> **Create**([Bind("EmpName,Salary,ImageURL,ThumbnailURL")] Employee employee, IFormFile imageURL)

{

if (ModelState.IsValid)

{

//Code to Upload the image into Blob Container.

string cs = \_config["StorageConnectionString"];

BlobServiceClient blobServiceClient = new BlobServiceClient(cs);

BlobContainerClient blobClient = blobServiceClient.GetBlobContainerClient("empimages");

string blobName = Guid.NewGuid().ToString() + Path.GetExtension(imageURL.FileName);

var blob = await blobClient.UploadBlobAsync(blobName, imageURL.OpenReadStream());

employee.ImageURL = blobClient.Uri.ToString() + "/" + blobName;

\_context.Add(employee);

await \_context.SaveChangesAsync();

//Posting message to Queue.

QueueClient queueClient = new QueueClient(cs, "thumbnail-queue");

BlobInformation blobInformation = new BlobInformation();

blobInformation.BlobUri = new Uri(blobClient.Uri.ToString() + "/" + blobName);

blobInformation.EmpId = employee.Id;

string blobString = **ToBase64**(blobInformation);

Console.WriteLine(blobString);

await queueClient.SendMessageAsync(blobString);

return RedirectToAction(nameof(Index));

}

return View(employee);

}

public string **ToBase64**(object obj)

{

string json = JsonConvert.SerializeObject(obj);

byte[] bytes = Encoding.Default.GetBytes(json);

return Convert.ToBase64String(bytes);

}

1. **Build and run the application and open the url: http://locahost:5000/employees**

**Note that this time FullSizeImage is updated with BLOB URL and ThumbNailUrl is null.**

**Programming Azure Function**

1. Add Function App Project to the Solution
   1. File 🡪 Add 🡪 New Project 🡪 Azure Functions 🡪 Next
   2. Project name = **EmployeeManagement.ThumbnailGenerator** 🡪 Next
   3. Functions worker = .NET 6.0 (LTS), Function = Queue trigger, **Uncheck** Use Azurite for runtime storage account 🡪 Connection strings settings name = "StorageConnectionString", Queue-name=**thumbnail-queue** 🡪 **UnCheck** configure dependencies 🡪 Create
2. Add reference to NuGet package **Microsoft.EntityFrameworkCore.SqlServer**
3. Add reference to Project **EmpoyeeManagement.SharedLibrary**
4. Add to Project appsettings.json

{

"IsEncrypted": false,

"**ConnectionStrings**": {

"EmployeeDataContext": "<Database connection string>"

},

"Values": {

"AzureWebJobsStorage": "<Storage connection string>",

"FUNCTIONS\_WORKER\_RUNTIME": "dotnet",

}

}

1. Edit Function1.cs

using Microsoft.Azure.Storage.Blob;

using Microsoft.Azure.WebJobs;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Logging;

using System;

using System.Drawing;

using System.IO;

using ThumbnailLibrary;

public class Function1

{

private IConfiguration Configuration;

public Function1(IConfiguration \_configuration)

{

Configuration = \_configuration;

}

[FunctionName("GenerateThumnail")]

public void Run([**QueueTrigger**("thumbnail-queue")] BlobInformation blobInfo,

[Blob("empimages/{BlobName}", FileAccess.Read)] Stream input,

[Blob("empimages/{BlobNameWithoutExtension}\_thumbnail.jpg")] CloudBlockBlob outputBlob, ILogger log)

{

using (Stream output = outputBlob.OpenWrite())

{

// ConvertImageToThumbnailJPG(input, output);

Image image = Image.FromStream(input);

var thumb = image.GetThumbnailImage(32, 32, () => false, IntPtr.Zero);

outputBlob.Properties.ContentType = "image/jpeg";

thumb.Save(output, System.Drawing.Imaging.ImageFormat.Jpeg);

}

var options = new DbContextOptionsBuilder<EmployeeDataContext>();

options.UseSqlServer(Configuration.GetConnectionString("EmployeeDataContext"));

var db = new EmployeeDataContext(options.Options);

var id = blobInfo.EmpId;

Employee emp = db.Employees.Find(id);

emp.ThumbnailURL = outputBlob.Uri.ToString();

db.SaveChanges();

}

}

1. Run the Web Application and Add an Employee
2. Run the Azure Function and Note that the **GenerateImage** method is called when a new message is found in the queue.