Demo 1: Azure Web Job

Web Job to create a PDF from the data files in the storage.

1. Create a Blob Storage Account in Azure Portal.
   1. Create a container and name it **data**. If you plan to change the name of the container, make sure you also change it in the code.
2. Create a Web App with App Service Plan S1.
3. Create a new console application and add the below dependencies using NuGet command:
   1. **Install-Package iTextSharp**
   2. **Install-Package WindowsAzure.Storage**
4. Go to the storage account and get the connection string from the **Access Keys** under the **Settings** section.
5. Open the **App.Config** file and add the below application setting:

<appSettings>

<add key="StorageConnectionString" value="Your Connection String Here"/>

</appSettings>

1. Create a new class and call it **Invoice**.

public class Invoice

{

[JsonProperty("id")]

public string Id { get; set; }

[JsonProperty("name")]

public string Name { get; set; }

[JsonProperty("address")]

public string Address { get; set; }

[JsonProperty("amount")]

public string Amount { get; set; }

}

1. Replace the below code in the **Program.cs** file with the below:

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Blob;

using iTextSharp.text;

using iTextSharp.text.pdf;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.IO;

using System.Linq;

using System.Net;

using Newtonsoft.Json;

namespace InvoiceGenerator

{

class Program

{

static string storageConnection = ConfigurationManager.AppSettings.Get("StorageConnectionString");

static CloudStorageAccount storageAccount = CloudStorageAccount.Parse(storageConnection);

static CloudBlobClient blobClient;

static CloudBlobContainer blobContainer;

private static List<Uri> GetDataFiles()

{

var list = blobContainer.ListBlobs();

return list.OfType<CloudBlockBlob>().Select(b => b.StorageUri.PrimaryUri).ToList();

}

private static string GetInvoiceData(Uri uri)

{

return new WebClient().DownloadString(uri.OriginalString);

}

private static void GeneratePDF(string InvoiceData)

{

Invoice inv = JsonConvert.DeserializeObject<Invoice>(InvoiceData);

FileStream fs = new FileStream(inv.Id + ".pdf", FileMode.Create);

Document doc = new Document(PageSize.A4, 25, 25, 30, 30);

PdfWriter writer = PdfWriter.GetInstance(doc, fs);

doc.AddTitle("Invoice - " + inv.Name + " - " + inv.Id);

doc.Open();

doc.Add(new Paragraph("Invoice # - " + inv.Id));

doc.Add(new Paragraph("Company Name - " + inv.Name));

doc.Add(new Paragraph("Address - " + inv.Address));

doc.Add(new Paragraph("Amount - " + inv.Amount));

doc.Close();

writer.Close();

fs.Close();

CloudBlockBlob blockBlob = blobContainer.GetBlockBlobReference("pdf/" + inv.Id + ".pdf");

using (var fileStream = File.OpenRead(inv.Id + ".pdf"))

{

blockBlob.UploadFromStream(fileStream);

}

}

static void Main(string[] args)

{

blobClient = storageAccount.CreateCloudBlobClient();

blobContainer = blobClient.GetContainerReference("data");

foreach (var dataFile in GetDataFiles())

{

//Read the data from the JSON file and generates the PDF in the storage account.

GeneratePDF(GetInvoiceData(dataFile));

}

}

}

}

1. All the PDFs are created from a JSON files which will be placed in the storage. Create multiple JSON files and use the below format.

{

"id": "I201701022",

"name": "Contoso Inc.",

"address": "Microsoft Corporation, 1 Microsoft Way, Redmond, WA, 98052",

"amount": "$12.10"

}

1. Build the project.
2. Once the build is successful, go to the Release/Debug folder.
   1. Select all the files by pressing **CTRL+A**.
   2. Right-click and select **Send to** **> Compressed (zipped) folder**.
3. In the Azure Portal, go to a web app and click **WebJobs** under **Settings** section.
4. Click Add to create/add a new WebJob. In the Add WebJob blade:
   1. **Name:** This will be the name of the WebJob.
   2. **File Upload:** Select the zip file.
   3. **Type:** Set the type as **Triggered**. This will enable the **Triggers** and **CRON Expression**.
   4. Set **Triggers** as **Scheduled** because we want to run it automatically.
   5. Set the CRON Expression as per your demo need. You can check the CRON expression with the help of [this](https://www.freeformatter.com/cron-expression-generator-quartz.html) website.
   6. Click **OK**
5. Check after the specified time that the PDFs are generated in the storage account under the **pdf** folder.

Demo 2: Azure Batch

Please follow the steps mentioned in the documentation here: <https://docs.microsoft.com/en-us/azure/batch/batch-dotnet-get-started>

Github: <https://github.com/Azure/azure-batch-samples/tree/master/CSharp>