**Background**

Large file uploads over HTTP can be challenging due to the stateless nature of the HTTP protocol and the necessity to maintain an open connection to the server for a long period of time. To avoid timeout issues the file upload method must be able to split the file into smaller chunks that can be uploaded in short bursts that do not result in server timeout errors. The Azure Blob Storage API includes support for uploading large files to Azure blob storage using a “chunked” upload mechanism. The Azure Blob Storage API is available for a variety of programming languages and the provided sample code demonstrates how to use the JavaScript Azure Blob Storage API to upload a large file to Azure blob storage using the “chunked” upload approach. The sample also includes a console application that demonstrates how to create a SAS token for an Azure blob storage container.

**Solution Structure**

ChunkedBlobUpload.sln (Visual Studio 2017 solution)

* ChunkedBlobUpload (.Net Framework 4.7 ASP.Net MVC project)
* CreateContainerSASUrl (.Net Framework 4.7 console application project)

**Project Structure**

ChunkedBlobUpload

*Controllers*

* HomeController.cs
  + The Index method stores the name of the Azure storage account and the blob container where the files will be uploaded in the ViewBag for transmission to the view along with the application root URL and then displays the Index page.
  + The GetSAS method returns a SAS token that will allow write access to the blob container. This SAS token was generated using the CreateContainerSASUrl console application. If a dynamically generated SAS token is needed, the code from the CreateContainerSASUrl console application can be included in the GetSAS method.

*Views*

* Index.cshtml
  + Displays the HTML page for the file upload application and includes all of the client-side Javascript code necessary to trigger the file upload and report on the progress of the upload.
  + Registers the Azure storage common JavaScript library.
  + Registers the Azure storage blob JavaScript library.
  + Function FileStream creates a file stream object used to read the local file to upload from the file system.
  + Loads the storage account name, container, and application root URL into JavaScript variables from the ViewBag.
  + Function checkParameters validates the input parameters needed to upload a file to Azure blob storage.
  + Function getBlobService connects to the Azure storage account using the provided SAS token and gets a reference to the blob service.
  + Function displayProcess updates the percent complete on the upload progress bar.
  + Function uploadBlobByStream calculates the size of the blocks (or chunks) to upload. It then calls the createBlockBlobFromStream method of the Azure Blob Service to upload the file. The createBlockBlobFromStream method returns a SpeedSummary object that will report the status of the upload as it is executing. When the upload completes it calls the provided callback function that will indicate if the upload was successful or failed.
  + Function refreshProgress calls the getCompletePercent method of the SpeedSummary object to get the percent complete of the in progress file upload. This value is used to update the percent complete on the progress bar.
  + On document ready it calls the GetSAS method of the HomeController to retrieve the container SAS token and populate the sas JavaScript variable.

CreateContainerSASUrl

* Program.cs
  + Creates a CloudStorageAccount for the specified Azure storage account.
  + Creates a CloudBloblClient for the specified blob.
  + Creates a CloudBlobContainer for the specified blob container and creates the container if it does not exist.
  + Calls GetContainerSasUri to create a SAS Uri for the container
  + Method GetContainerSasUri creates a SharedAccessBlobPolicy valid for 24 hours and specifies List, Write, and Read access to the blob container. It then calls the GetSharedAccessSignature method of the CloudBlobContainer object passing it the SharedAccessBlobPolicy. The result contains the SAS token for the blob container.