Assignment 9 – Cloud Storage  
COS318 – FA2018

Due Date: November 1st, 2018  
Turn in all files using Moodle

You’re at tribal council, not sure if you can trust everyone who is sitting around you. Perhaps some of them really are on your side, or perhaps they are about to blindside you. Whatever the outcome, it is the people who learn to outwit, outplay, and outlast who will be at the final tribal council. In this ninth assignment, you will be using azure cloud storage to save table entities and images. Here’s to hoping that Jeff does not snuff out your torch.

1. **(30 Points) Html/Javascript**
   1. **HTML:** Create an HTML page that will display a series of images. The list of images should be laid out horizontally. If the list of images is too long to fit horizontally, they should automatically wrap onto more lines (and update dynamically if the browser window is resized). These image elements should be created dynamically as you loop through a JSON response of image entities from the server.
   2. **img:** Each image should be enforced to have a reasonable max height and width, such as 300px and should display their title when the mouse hovers over them.
   3. **form:** Add a form field where a user can select a file to upload and a title for the image. The file selection dialog should be scoped to only show image files and the title should be validated to be at least three characters. Also add an upload button to begin the upload of the image file.
   4. **upload:** After the upload is complete, send a request to notify the server, and then display the new image on the page automatically (using a download URL obtained from the server).
2. **(30 Points)** **ImagesController**
   1. **GET:** Return a JSON response with all the images on the server.
   2. **GET (id):** Return a URL to an image as a temporary redirect. Set the cache to be at least six hours.
   3. **Post:** Accept an ImageEntity JSON that allows the client to specific the image name to be created. The response JSON should have a URL and SAS token that can be used to upload an image to an azure storage account.
   4. **UploadComplete (Post):** Accept a string id that is an id of an image previously created. Update the image that is already stored and save that the upload has completed.
3. **(20 Points) ImageTableStorage**
   1. **GetUploadSas:** Change the GetUploadSas method so that SAS that is returned only has permissions to a single blob file, rather than the entire blob container.
   2. **GetDownloadUrl:** Implement this method so that it gives a URL that can be used to view the image passed into the method. The URL should last for at least six hours, and only have access to that specific image, not the entire container. (Hint: This method will be called by your controller GET (id) method from step 2b.)
4. **(20 Points)** Code style, formatting, completeness, and quality.

Stretch Levels

If you already have a lot of experience with azure storage accounts, or if you just won an immunity challenge, try to complete these stretch levels for a reputation bonus. If you try for the stretch levels, make sure to type it in the comments on Moodle so I don’t miss it.

**Cirie Level**

Add some CSS to your page to make it look nicer. Background colors, font colors, or anything that looks good.

**Rupert Level**

Add a progress bar to your page that updates as the image is being uploaded. Bonus if the progress bar is smoothly animated!

**Boston Rob Level**

Add a slider to the page that controls image size.

The Rules

1. No inline styles or javascript.
2. Error messages must be “in-page” i.e. no pop-ups or alerts.
3. Any resources not created by you (images, javascript libraries, etc.) must be referenced using a CDN or URL, not directly included in your assignment submission.
4. Service/data/model classes must not have any http, request, or response references.
5. Controller entity classes must not be used directly to store data on the server; translate them into a model (data storage) class before saving the data. Conversely, controllers must not send any model classes to the user; translate them into controller entity classes before sending the response.
6. You may not use any synchronous methods in your C# code wherever there is an async option.
7. All service class instances must be obtained using dependency injection.
8. All requests that submit a body to your server must have their JSON structure validated with ModelState. The controller is not allowed to validate the ModelState directly; this must be done in a filter.