Assignment 11 – Api Versioning  
COS318 – Web Programming

It’s time to return to Survivor. Many have had the opportunity to play more than once, and there are some who have played four times. Just like the returning players of Survivor have learned from the mistakes of their first plays, hopefully you have learned from your past assignments. In this revisiting of the cloud storage assignment, you will add a new field to your controller and html, but must also support the existing version from the previous assignment at the same time.

1. **(40 Points) Html/Javascript**
   1. **HTML:** Make a copy of the html page from the cloud storage assignment. The existing html page will be known as 1.0 and the new html will be 1.1. Add three things to this new page. First, a new field called “description.” Second, put some indication on the page that the user is on version 1.1 of the assignment (and add 1.0 to the original html.) Lastly, add a link to both pages that takes the user to the other page.
   2. **Javascript:** Update the javascript to send the api-version query parameter with all requests. The api-version that is sent should match the version displayed on the html page. If the description field is available (from the 1.1 page version), then also send that field in the JSON body to the server when posting new images. On the 1.1 page, display an error message if the description field is not at least 5 characters long and then don’t make a request to the server.
      1. There should only be a single javascript code file, not one per version.
2. **(40 Points)** **Server**
   1. **Controller:** Update your folder structure and namespaces to add a second ImagesController. They should be tagged with the api versions of 1.0 and 1.1. All new changes will be done to the 1.1 version of the controller.
   2. **Entities:** Update your folder structure and namespaces to add a second ImageEntity. The new ImageEntity should add a Description string that is required and has a minimum length of 5. Hint: Make sure each controller is using the correct matching version of ImageEntity.
   3. **ImageModel:** Modify ImageModel to also store the description field. The conversion methods to and from ImageEntity and ImageModel will also need to copy the description field (in the 1.1 version).
3. **(20 Points)** Code style, formatting, completeness, and quality.

Stretch Levels

If you already have a lot of experience with api versioning, or if you just won Survivor for the second time (only one person has ever done this!), 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.

**Sandra Level**

Also support date versions on your controllers. Version 1.0 would map to “2019-11-01” and version 1.1 would map to “2019-11-15”. Specifying the versions as 1.0 and 1.1 must continue to work.

**Cochran Level**

Support the date versions like you did in the Sandra stretch level, but instead of using annotations on the controller, specify the versions in Startup.cs using Conventions. If you do this stretch level, you will automatically also receive credit for the Sandra level, since doing this stretch level replaces that work. Hint: Be careful with your using statements for your controllers at the top of Startup.cs! It will probably be easiest to fully qualify your controller names (i.e. full namespace path) at the place you reference them and skip the using at the top of the file altogether.

The Rules

1. No inline styles or inline 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 entities validated with appropriate annotations, such as MinLength, Range, or Required.