Assignment 7 – TFH, async, await  
COS318 – FA2017

Due Date: October 26th, 2017  
Turn in all files using Moodle

Grab a star and run like crazy. Assignment seven is full of koopas, goombas, and pirana plants. In this assignment you’ll be creating a controller that is dependent on another service. There is a problem though. This service doesn’t allow more than about one request per second. That is certainly going to be annoying.

1. **(30 Points)** **MarioLevelController**
   1. Create a controller with a single endpoint which will accept GET requests and one string URL parameter. This endpoint will return a JSON document with one field, “message”. The message should be a short string explaining the action Mario just took.
   2. The URL parameter sent to MarioLevelController must be one of the four valid actions, “walk”, “jump”, “wait”, or “run.”
   3. MarioLevelController should use a service to make a GET request to the external server (see below for more information). This request must be wrapped in a Retry Policy with a maximum of 10 retries.
      1. The URL parameter accepted by your MarioLevelController is the same value you will send to the external server.
      2. The message that MarioLevelController returns can optionally be read directly from the external server response. Otherwise, you can create your own messages based on the action.
2. **(20 Points) MarioService and IMarioService**
   1. Create a service that will make requests to the external server.
   2. The IMarioService should not expose any fields or methods that would imply that it is making requests to a server to perform its work.
3. **(30 Points) Html and Javascript**
   1. Create an html page with javascript that will communicate with your MarioLevelController.
   2. Add an image of Mario on the left side of the screen.
   3. Add a button to the HTML page that begins the level.
      1. The button should not be active while the level is in progress.
   4. When the level begins, make a request to your MarioLevelController with a random value selected from the four actions. On a successful response, move Mario along the screen based on the action selected.
      1. Response: Walk: 5%, Jump: 5%, Wait: 0%, Run: 10%
      2. Repeat the request until Mario dies or reaches the far right of the screen.
   5. Display the Message that comes from the MarioLevelController response somewhere on the page.
4. **(20 Points)** Code style, formatting, completeness, and quality.
   1. The javascript code should never be exposed to any errors from the external service, nor should it have any knowledge that the external service is being used by MarioLevelController

The external endpoint URL is of the form http://webprogrammingassignment7.azurewebsites.net/api/mario/{move} where {move} is one of the four valid actions, “walk”, “jump”, “wait”, or “run”. The endpoint has three possible responses.

1. **200 – OK:** A JSON document with Message and NextStep keys. Message is a friendly string of what happened to Mario that can be directly returned by the MarioLevelController. NextStep is used in the silver stretch level to indicate the next action Mario should take.
2. **503 – Service Unavailable:** The server can’t process the request because another request was processed too recently. This is a transient failure.
3. **500 – Internal Server Error:** Mario died. Any request to the server has a small chance for this response. This is NOT a transient failure. If this is encountered, your MarioLevelController should return 200, but with a message indicating that Mario died.

The Rules

1. You may not use any synchronous methods in your C# code wherever there is an async option.
2. All service class instances must be obtained using dependency injection.
3. The controller is not allowed to validate the ModelState. This must be done in a filter.
4. All rules are cumulative, so all past rules also apply to this assignment.

Stretch Levels

If you already have a lot of experience transient fault handling or just really want to rescue the princess, try to complete these stretch levels for extra credit. If you try for the stretch levels, make sure to type it in the comments on Moodle so I don’t miss it.

**Peaches’ Castle Level**

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

**Toad House Level**

Use a RetryPolicy that has a minimum and maximum backoff with a delta time increase for each retry.

**Luigi’s Mansion Level**

The external service returns a NextStep in addition to a message. Modify your MarioLevelController to pass along NextStep to your html page. Then instead of randomly choosing the next action Mario will take, use the value from NextStep.

**Bowser’s Castle Level**

MarioLevelController is an external service to your javascript code. Because of that, it really should have transient fault handling also. Design and implement transient fault handling in your javascript code that retries failed requests at least five times before it gives up.