Assignment – Dependency Injection  
COS318 – Web Programming

Your next assignment is full of dependencies and hobbits. Frodo and Sam are dependent on Aragorn and the humans distracting Sauron. Gondor is dependent on help from Rohan. You are dependent on the code from GitHub. You will need to download all of the code in the assignment folder. The server folder contains all of the code you will be editing for this assignment. The wwwroot folder contains some helper html and javascript to test your server code.

There are four bad service dependencies in the assignment code that you will need to update.

1. **(20 Points)** **MemoryDatabase**
   1. HobbitController keeps track of a list of hobbits in a database. However, it is currently coupled with MemoryDatabase. Update the controller so that it receives an instance of an IDatabase in the constructor.
   2. Create an IDatabase interface based on the MemoryDatabase class.
   3. There should only ever be one instance of MemoryDatabase created for the lifetime of the application.
   4. Don’t create an instance of MemoryDatabase yourself. Let it to be created for you by the framework.
2. **(20 Points)** **ConsoleLogger**
   1. ConsoleLogger is a singleton class. It creates its own instance that is then used everywhere else in the application to write debug messages.
   2. Create an ILogger interface based on the ConsoleLogger class.
   3. Update all references to ConsoleLogger to depend on and use ILogger instead.
   4. There should only ever be one instance of ConsoleLogger created for the lifetime of the application.
   5. The only change you should make to ConsoleLogger is to implement an interface.
3. **(20 Points) RequestIdGenerator**
   1. Create an IRequestIdGenerator interface based on the RequestIdGenerator class.
   2. A new IRequestIdGenerator instance should be created once per request.
   3. RequestIdFilter is supposed to add a unique request-id header to each response. However, currently it only is using the static string “local-id.” Update the RequestIdFilter so it depends on an IRequestIdGenerator and uses it to add a real request id to the request-id header.
4. **(20 Points) StopWatchService**
   1. StopwatchFilter tracks the amount of time each step of the request is taking. It then outputs the result into a response header.
   2. Update all references to StopWatchService so that during a request all references share the same instance.
   3. You do not need to create an interface for StopWatchService. You may use it directly.
5. **(20 Points)** Code style, formatting, completeness, and quality.
   1. Running all the tests on the assignment6.html page will help a lot in making sure your updates are working correctly. The tests are not exhaustive to all possible scenarios. Make sure to double check your work even if all the tests are passing.
   2. ConsoleLogger doesn’t have any tests in the provided html or javascript. You’ll have to ensure this is working on your own.

Stretch Levels

If you already have a lot of experience with dependency injection, or if you just want to go hunt some orc, 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.

**Gandalf Level**

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

**Legolas Level**

ConsoleLogger (ILogger) shouldn’t be logging requests directly in the controller. Create a new filter or middleware that logs the request path, query, and method, then remove all ILogger references from HobbitController.

**Aragorn Level**

Update StopWatchFilter to also add the total request time in a different header called ‘totaltime.’ Be sure this header is useable by the browser. (Hint: CORS!)

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. All requests that submit a body to your server must have their entities validated with appropriate annotations, such as MinLength, Range, or Required.
5. The root path of your server must display the main page of your application.
6. Service/data/model classes must not have any http, request, or response references.
7. 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.
8. **New Rule:** All service class instances must be obtained using dependency injection.