INFO5125

Assignment 3

You will be creating a fake network server cluster. It will work like a real server cluster, except you decide when requests are processed.

In computer networking, servers typically handle requests and provide responses. A remote computer will make a request to a server's IP address and port (the IP address is like a building's mailing address and a port is like an apartment or unit number). The server will read the request, process it, and send some data back to the sender.

Typically with REST-specification servers, each request is routed to an endpoint: for example, if I wanted to retrieve all Widgets, I might send a request of type "GET" to "/api/widgets". If I wanted to get a specific widget, I would send a request of type "GET" to "/api/widgets/5", where 5 is the ID for a widget. When the server receives a request, it will check all of its **routes** to see if that route's that matches the request's path.

For example if a route has path "/api/widgets/:id", a request for "/api/widgets/5" will match it, and as such that route will handle the request (i.e. call a function with it) and not let any other requests handle it. **This is exactly the Chain of Responsibility pattern** discussed in class.

You will be implementing this server using microservices. Microservices are a very new server set-up where multiple copies of the server program are running over multiple machines. When a request comes in, it is sent to a free machine in order to spread the work over multiple CPUs and network widths.

Details

Request

- Each HTTP request is a Command.
- It should contain a body (an integer) and a string Route to use.

AbstractServer

- Superclass of Server.

Server

- Servers have functionality to handle a Request.
- Your Server class should have the following "endpoints":
    - /mul
    - /mul/4
    - /add
- "/mul" takes an argument of a number and returns that number multiplied by 2.
- "/mul/4" takes an argument of a number and returns that number multiplied by 4.
- "/add" takes an argument of a number and returns that number plus 8.
- Server should store a reference to the first Route in your Chain of Responsibility

Route

- Each Route is comprised of just a handler method which takes an integer argument and returns another integer.
- See the Server details on which routes you need to implement.

ServerQuery

- A ServerQuery is a Visitor.
- It visits subclasses of AbstractServer and checks to see if they have a pending request or not.
- Use a ServerQuery to check if a Server is busy or not.

Main Method

- Your main method has a list of Servers.
- You can create Requests here and send them to Servers.

Your server will not process requests immediately. You will use commands and user input to manually process them.

Command	Effect
createserver	Creates a new Server.
deleteserver:1	Deletes server #1.
listservers	List all servers.
new:/widgets/abc:5	Creates a new request with payload 5, routing to /widgets/abc.
dispatch	Dispatch a pending request to an available Server. Available servers are ones without a request currently in them. Check for an available server with a ServerQuery. If there are no available servers, print a message.
server:2	Has server 2 process work. Should print a route's response, or "404" if no routes were matched.
quit	Exit the program.

Starter Content

You are provided with a starter project containing many empty classes and interfaces and a basic main method including some input parsing functionality.

You should not have to add or remove any classes or interfaces.

You are also provided with an executable demo of the final program.

The starter project was written using .NET Core 3.0 or .Net Framework 6.0. If you want to deviate from this, feel free to create a new project in Visual Studio and copy over the files.

Rubric

Detail	**Points (%)**
Command ● Requests are encapsulated as Commands.	 15
Visitor ● ServerQuery, Server, and AbstractServer are set up to use the Visitor pattern. ● ServerQuery returns if a Server can accept requests.	 15 10
Chain of Responsibility ● Users can create requests, send requests to servers, and make servers do work. ● Routes correctly implement the CoR pattern. ● Servers correctly process work. ● All routes as described function as expected (e.g. /mul returns payload multiplied by 2)	 10 15 10 5
Misc. ● Program runs correctly without errors or warnings (correct output) ● Program adequately commented	 15 5
Total	**100**

NOTES:

- The rubric is a general heading for grades. Your code should follow the code standard. The code standard is part of each line of the rubric.