Assignment Report
Distributed Computing

4/06/2020

by:

Kyle Kriskovich, 18831117

Contents

1. Three Tier .Net Remoting Application

1.1 Summary

The solution consists of separate projects DC_lab which holds the database dll, DC_Server which is the data tier application, WebServiceBuisness which is the business tier of the application and DC_Client which is the presentation tier of the application. When starting the application begin the instances in this order first the data tier, then business tier then finally the client.

1.2 Implementation Choices

After implementing the presentation tier the user has the ability to access the database via two possible methods. Firstly entering a desired index and the program will change the textboxes to hold the information of the account at that index. Secondly the user can search the database for a last name and the client will output the information that matches the first instance of that last name in the database if it can find it. There is also a progress bar that should count up as the request for the search is being performed by the data tier such that the client does not freeze.

The logging keeps track of the programs progress and is implemented in the business tier of the application which this keeps track of every request to the data server and if it was successful or not. Lastly the data tier holds the instance of the database from the DC_Lab dll. Exception handling for the client application means the client is able to continue to run without crashing but the business tier is where the logging occurs.

2. Three Tier Web Service Application

2.1 Summary

The three tier web service application consists of two projects one web api data server and one business server that has the presentation tier implemented as webpage in the browser. To start the application you must have both server instances running.

2.2 Implementation Choices

The presentation tier of the application is a single web page that allows the user to load various html sections of the application onto the home page using AJAX. The user is able to create and get Users, Accounts and Transactions from the underlying database behind the data tier. The Home page also has a hot bar that allows the user access to the individual pages if they want. Exception handling primarily occurs on the web api's of each application as each function try's to catch each all of exception that they can throw. Once an exception is caught the applications appends what function and what controller it occurred at then outputs it to the console.

3. Peer to Peer Application

3.1 Summary

The Peer to Peer application only holds two projects the client application and the webserver. The webserver must be running before the client applications can communicate.

A client CANNOT execute its own submitted job so to test you need two client instances.

3.2 Implementation Choices

Each client has the ability to launch itself on a desired IP and Port once it's done this it will register that port to the webserver which holds a list of clients. The application allows the user to submit snippets of python code to each other but CANNOT execute its own submitted code. Thus to use this application minimum of two client instances need to be running so they can send and execute each other's code. Once the code has been submitted and the get results button has been pressed then code box will display the result when its ready and the completed jobs box will update. I've added default job option buttons that fill the python box with some default jobs. This allows the user to send some quick default jobs. When it comes to exception handling most functions that can potentially throw an exception will try to catch each exception that can be thrown. Once an exception is caught the application will log the exception message to the console as well as the IP and Port of the client so the console can determine the instance which the exception is thrown from.

4. Block chain Application

4.1 Summary

The block chain application consists of client application and a web server. The webserver must be started so the clients can function properly.

To test the client application each submitted transaction waiting to be executed must differ in content to the other waiting transactions eg("1+1", "2+2", "3+3", "4+4", "5+5") is a valid transaction list.

4.2 Implementation Choices

Each client has a static class that holds their instance of the block chain and their client details (ip and port) that they are currently running on. The application allows the user to select the IP and Port that they desire to run on but will not allow them to do anything else before doing so. Once the application has successfully opened on an IP and port it registers itself with the server. Now the user is able to submit bits of python code to be mined into blocks for the block chain. The get results button will fill the results tab once the block chain is valid and holds the results of the submitted code. The web service only holds a list of all the clients that have registered themselves with the server and does not have any form of computation other than get of set. The application will only begin adding blocks to the chain once it has 5 transactions waiting to be added and each transaction MUST be different to the other waiting transactions to be a valid transaction. Exception handling occurs in each function that can throw exceptions so that the two threads can continue running even if one has occurred. If an exception occurs when a block is being mined the block that was being worked on is discarded. All Exceptions are logged to the console from the client that the exception occurred in including the IP and Port of the current client.