BLCN532 Lab 3
Middleware

Introduction

This lab introduces you to the layer of functional code that exists between smart contract chaincode and the user-focused application. Middleware acts as both a service and interface layer. In this lab you'll examine source code and configuration files that make up the middleware layer and then invoke functionality through an exposed service.

Lab Deliverables:

To complete this lab, you must create a **Lab Report file** and submit the file in iLearn. The Lab Report file must be a Microsoft Word format (.docx), and have the filename with the following format:

BLCN532_SECTION_STUDENTID_LASTNAME_FIRSTNAME_Lab03.docx

- SECTION is the section number of your current course (2 digits)
- STUDENTID is your student ID number (with leading zeros)
- LASTNAME is your last name
- FIRSTNAME is your first name

To get started, create a Microsoft Word document (.docx) with the correct filename for this lab. You'll be asked to enter text and paste screenshots into the lab report file.

NOTE: All screenshots MUST be readable. A screenshot that I cannot read (i.e. cannot read the text you are capturing) will not count for any points.

SECTION 1: Installing tools and dependencies

Step 1.1: Building an application: Installing tools and dependencies

First, you'll log into your development environment and install a few tools and dependencies.

1. Open PowerShell
2. PS %HOME%> **cd vagrant\hyperledger**
3. PS %HOME%\vagrant\Hyperledger> **vagrant up**
4. Launch PuTTY -> open connection (Vagrant Hyperledger)
5. $ **cd $GOPATH/src/trade-finance-logistics/middleware**
6. Good news!!! The virtual machine we're using already has node.js and npm installed!!
7. $ **node -v**
8. $ **npm -v**
9. Create a screenshot of the results of steps 5-8, and paste that screenshot into your Lab Report File.
10. Install fabric-ca-client and fabric-client
    a. Two methods:
        i. $ **npm install fabric-ca-client**
        ii. $ **npm install fabric-client**
    b. OR, just $ **npm install**
11. Create a screenshot of the results of **npm install**, and paste that screenshot into your Lab Report File.

Step 1.2: Configuring the network and application

Next, you'll examine the network configuration.

1. Open the **config.json** file ($ **nano config.json**)
2. Scroll down to the entry that defines the **Carrier** organization. (**carrierorg**)
3. Create a screenshot that shows the **url, requests,** and **events** attributes of the **carrierorg** definition, and paste that screenshot into your Lab Report File.
4. Exit from the **nano** editor.
5. Open the **clientUtils.js** file ($ **nano clientUtils.js**)
6. Scroll down to the **function** named **registerAndEnrollUser**.
7. Briefly read through the code and comments to see how the JavaScript program carries out the tasks of registering and enrolling a new user.
8. Create a screenshot that shows the first several lines of the **registerAndEnrollUser** function, and paste that screenshot into your Lab Report File.
9. Exit from the **nano** editor.

10. Open the **createTradeApp.js** file ($ **nano createTradeApp.js**)
11. Scroll down to the line in which "Wood for Toys" requests a new trade action.
12. Create a screenshot that shows the JavaScript code from step 11, and paste that screenshot into your Lab Report File.
13. Exit from the **nano** editor.

Step 1.3: Install and run, and invoke the chaincode

1. $ **cd $GOPATH/src/trade-finance-logistics/network**
2. Remove previous cryptographic material
    a. $ **rm -rf client-certs**
3. $ **./trade.sh up**
4. Open a New PuTTY connection (Vagrant Hyperledger)

   NOTE: This is the SECOND PuTTY CONNECTION for screenshots in later steps.

5. $ **cd $GOPATH/src/trade-finance-logistics/middleware**
6. $ **node createTradeApp.js**
7. Create a screenshot that shows results from step 5, and paste that screenshot into your Lab Report File.
8. (In the FIRST PuTTY CONNECTION window) $ **docker ps -a**
    a. Scroll to display the dev-peers
9. Create a screenshot that shows results from step 7, and paste that screenshot into your Lab Report File.
10. $ **node runTradeScenarioApp.js**
11. Create a screenshot that shows results from step 9, and paste that screenshot into your Lab Report File.

Step 1.4: Run the blockchain client application

1. $ **cd $GOPATH/src/trade-finance-logistics/application**
2. $ **npm install**
3. $ **node app.js**
4. Create a screenshot that shows results from step 5, and paste that screenshot into your Lab Report File.
5. Open a New PuTTY connection (Vagrant Hyperledger)

   NOTE: This is the THIRD PuTTY CONNECTION for screenshots in later steps.

6. $ **cd $GOPATH/src/trade-finance-logistics/application**
7. $ **curl -s -X POST http://localhost:4000/login -H "content-type: application/x-www-form-urlencoded" -d 'username=Jim&orgName=importerorg'**

8. $ **curl -s -X POST http://localhost:4000/login -H "content-type: application/x-www-form-urlencoded" -d 'username=admin&orgName=importerorg&password=adminpw'**
9. **Copy the value of the token you got back from the results of step 8 to your clipboard.**
    a. In Linux, just highlighting text at the command line copies that text to your clipboard.
10. $ **curl -s -X POST http://localhost:4000/channel/create -H "authorization: Bearer PASTE THE TOKEN VALUE FROM STEP 8 HERE"**
11. Create a screenshot that shows results from step 10, and paste that screenshot into your Lab Report File.

Section 2: Wrapping up

Now that you have installed, run, and tested your chaincode, you need to properly shutdown your business network.

1. In the FIRST PuTTY CONNECTION $ **./trade.sh down**
2. Exit from your Linux sessions
3. $ **exit**
4. In Windows PowerShell, shut down your Hyperledger virtual machine:
1. PS %HOME%\vagrant\Hyperledger> **vagrant halt**

You should have 10 screenshots in your Lab Report File. Save your file and submit it in iLearn as a file attachment for the Lab 3 assignment.

Congratulations! You have complete lab 3.