PROJECT COURSE
Demo Client-Server Setup
Installation and Setup

Overview

The client-server demo serves two key purposes. First it illustrates how to use a client to access a node server to create, read, update, and delete (CRUD) data from a database. Secondly, is to give you a head start on building your servers to accept orders and perform all the other order fulfillment duties described in the project requirements document. Every attempt was made to keep this system small and manageable. This document will describe how to download the required technologies and install the systems on your machine.

Client-Server Demo Technology Stack

This assignment makes use of the following technologies:

- MySQL Community Server – Open source database that stores all of the order and inventory data.
- J Connector – This is the Java Database Connectivity standard that allows Java programmers to access data in databases from within a java application.
- Node.js/Javascript – Node.js is used as a server for the web-services system. Of course Javascript is the language that Node.js servers are written in.
- Java - This is the language that is used on the client side for both systems and on the server side for the micro-services systems. We will utilize RMI once again (hopefully leveraging what you learned in A2).

Unpacking the Archive

Create a working folder on your computer. Since you will have to use the command window (windows) or the terminal (MacOS), put the folder somewhere where it is easy to navigate to on the command line. It's a good idea to give it a nice short name too. Unzip the client-server.zip file into the working folder. You should see…

`package.json` - contains the application dependencies
`REST.js` – Javascript/node.js module for the restful services
`Server.js` - Javascript/node.js server module
`OrdersUI.java` – orders database user interface
`WSClient.java` – web services client used by OrdersUI.java
`orderdb_schema.sql` – orders database schema template

Installation of MySQL

If you already have MySQL installed on your system, you can skip this step. If not, download the MySQL Database that matches your OS from *https://www.mysql.com/downloads/*

Follow the installation guidelines provided. You will not need any other data base components other than the database unless you want to load them. You will have to add the path to the MySQL executables in your path, otherwise you will have to use the fully qualified pathname to run any of the executables. You should use a typical installation and all defaults. Make sure to record the default password for root. You can change your password if you like by following the instructions here:

https://dev.mysql.com/doc/refman/5.7/en/resetting-permissions.html

Testing MySQL

To test MySQL installation, start it up. On a Windows operating system…

- open a command window
- at the prompt type `start mysqld &` (note that this starts MySQL and runs it as a background process. Omit the '&' to run it in the foreground.)
- to stop MySQL type mysqladmin type `mysqladmin –u rootpw shutdown` (note rootpw is the MySQL root password).

On MacOs…

- open a terminal window
- at the prompt type `mysql.server start` (note that this is a script that automatically starts mysqld and runs it in the background for you).
- to stop MySQL type `mysql.server stop` (again this is a script that automatically stops mysqld).

Creating the Databases

Make sure that MySQL is running per the instructions above. If you were able to start the MySQL database, you should be able to start the command interface to MySQL by typing:
`mysql --user=root --password=YourRootPassword`

If you successfully started the command interface, you should see the command prompt as follows:

```
mysql>
```

Next we will create the database…

```
mysql>create database orderdb;
```

Now exit mysql by typing, `exit` or `quit` at the command prompt.

Next, we will copy the schema of the database from the template provided. At the operating system prompt, type the following:

```
mysql --user=root --password=YourRootPassword –D orderdb <
orderdb_schema.sql
```

Note that orderdb_schema.sql is in your working folder. This command copies the schema information from the orderdb_schema.sql file into the orderdb database. Note that it does not copy data.

Checking the Database Schema

Now start up mysql again and verify that both databases exist by typing:

```
mysql> show databases;
```

You should see the orderdb database listed. Now verify that the tables are correct by typing:

```
mysql> use orderdb;
mysql> show tables;
```

You should see the "orders" table. Next check the table schema by typing:

```
mysql> describe orders;
```

You should see the following:

Setting up the Server

If Node.js is installed on your system, you can skip this step. If not, download Node.js from: *https://nodejs.org/en/download/*

Install Node.js on your system per the instructions provided. To verify that the installation worked, start a command window (Windows OS) or a terminal window (MacOS) and type `node –v` at the prompt. You should see the version number of Node.js displayed on your terminal.

In the working folder, we first install all the modules required for the node server. Type the following at a command or terminal window prompt:

```
npm install express --save
npm install mysql --save
npm install body-parser --save
npm init
```

Accept all the defaults after you type `npm init`

After you type each command, you will see npm downloading modules and installing them automatically. Next we need to build the `package.json` file. This contains the dependencies

required to run your web-server and is built for you automatically based on the modules you just down loaded. To build the `package.json` file type `npm install` at the command/terminal prompt. You can accept the defaults. If all this works, you should see the `node_modules` and the `package-lock.json` file added to the working folder.

You will need to change the root password for the MySQL instance on your machine. Open the Server.js file with an editor. You should see the following code snipit.

```
REST.prototype.connectMysql = function() {
    var self = this;
    var pool = mysql.createPool({
        connectionLimit : 100,
        host     : 'localhost',
        user     : '<username>',
        password : '<password>',
        database : 'orderdb',
        debug    :  false
    });
```

Change the user variable from <username> to your username (I use root) and change password variable from <password> to your MySQL password for the username. Save the file and exit the editor.

Building the Client Application

The client is a java program that consists of the following files.

- WSClientAPI.java – Abstracts the underlying construction of URLs and communication with the Node.js server from the user interface.
- OrdersUI.java – Application user interface.

To compile the source code, in the working directory type the following:

```
javac *.java
```

This will compile all of the source code required for the client. At this point the client is ready and you should be ready to test the system.

Starting the Web-Services System

Open two command/terminal windows and in both navigate to the working folder. In one window type: `node Server.js`

This will start the server. You will see a message indicating that the server started and is bound to port 3000.

In the other window, start the client by typing: `java OrdersUI`

This will start the user interface. Test the system by trying all the options.