CS 580 Client-Server Programming
Fall Semester, 2012
Assignment 2

San Diego State University -- This page last updated 9/13/12

Assignment 2 - Mars

Due Sept 14 23:59

Version 1.3

The goal of this assignment is to write a small client API that talks to a server. The API will be use by a GUI client in assignment 3. We are not building a GUI client in this assignment. That will be done in assignment 3.

The subject is motived by the recent rover on Mars and the subsequent interest in sending people to mars. Using the path of least energy consumption a round trip to mars takes nearly 3 years, which includes a 1.5 year wait for the right time to return. Our server helps answer the question of how much energy would it take to send people on the round trip. The server estimates how much energy it would take to send people and the needed food on a round trip to mars from the international space station. It does not address energy required to move the needed air, water and fuel. It also does not address the ship needed to house the people nor the energy to get everything from earth to the international space station. Using current rockets it will take more energy to get to the international space station than to get to mars from the space station. Instead of dealing with gallons of fuel the server computes how many years you will have to drive your car to equal the amount of energy to go to mars.

Server

The server protocol is text based. It assumes all text is encoded using UTF 8. The server uses TCP. Our test server address is: bismarck.sdsu.edu on port 8009. The server responses to two messages: quit and trip.

Messages

Quit.

The quit message is sent to the server when the client wants to end the connection. The message is:

```
quit;;
```

The server response with the message **quit;;** and then closes the connection

Trip.

A sample of the trip message is:

```
trip;destination:mars;people:1;weight:175;mpg:32;milesperyear:12000;;
```

The message starts with **trip;** and is followed by information. The message ends with ';' (a semi-colon). Given the format of the key-value pairs this means finds two semi-colons at the end of the message. The message contains five pieces of information: destination, people, weight, mpg, milesperyear. The information is sent as key-value pairs. The format is

key:value;

Keys are not case sensitive. So the key **milesperyear** is the same as **milesPerYear**. The values are case sensitive. The order of the key-value pairs does not matter. So the following message is the same as the one above.

```
trip;mpg:32;people:1;weight:175; destination:mars;milesperyear:12000;;
```

Destination. Currently the only valid value for destination is **mars.**

People (optional). The represents the number of people in your party going to mars. The value must be an integer. This key-value is optional. If **people** is not part of the message the default of one is used.

Weight (required). The value is the total weight of the people in your party going to mars plus the weight of there luggage. The value is an integer or a float. So 12, 12.34 and 2.34e2 are valid.

Mpg (required). The miles per gallon you get on your car. The value is an integer or a float. So 12, 12.34 and 2.34e2 are valid.

milesPerYear (required). The miles you drive your car. The value is an integer or a float. So 12, 12.34 and 2.34e2 are valid.

The server responds with the message

```
food:5376.4;weight:6.1;;
```

Where the value following the key **food** is the number of years you need to drive your car to use the amount of gasoline to transport your food for the mars trip. The value following the key **weight** is the number of years you need to drive your car to use the amount of gasoline to transport the given weight on roundtrip to mars.

Whitespace

The server allows whitespace (spaces and tabs) before and after the tokens ':' and ';'. For example the following is valid:

```
trip; mpg : 32 ; people : 1 ; weight:175; destination:mars;milesperyear:12000 ; ;
```

Errors

If the message you send to the server is not well-formed or there is an error on the server the server will respond with an error. A sample error is:

```
error: Invalid request starting with x;;
```

The format is **error:** followed by text making attempt to explain the error. Depending on the error the server may or may not be able to recover. If it can not recover the next message sent on the same connection will result in an error. After three errors the server sends the quit message and closes the connection.

Timeouts

The server will close connections if the delay between opening the connection and the first message being received is longer than 10 seconds.

Sample Conversation

Client Request	Server Response
trip;destination:mars;people:1;mpg:1;milesperyear:10;;	error: Missing weight;;
trip;destination:mars;people:1;weight:1;mpg:1;milesperyear:10;;	food:5376.4;weight:6.1;;
trip;destination:mars;people:3;weight:500;mpg:32;milesperyear:12000;;	food:430.1;weight:81.5;;
quit;;	quit;; Closes connection

How to turn in your Assignment

We will use the same process as assignment one. Upload your mercurial repository to bitbucket and submit the url to the course repository. If you use eclipse make sure that you include all the eclipse files in your repository.

Grading

The assignment will be graded based on being able connect to the server and properly handle the response.

Document History

Version 1.3 Added Grading and turning instructions.

Version 1.2 Added the Timeout section.

Version 1.1 Corrected typo in sample conversation. Removed quit;; from first and third client requests.