

rpc service

Recursive Tree Parsing



December 2, 2015

**1. Introduction**

The purpose of this project is to parse the trees using recursive decent. The communication between client and server is asynchronous over message bus.

**2. Design**

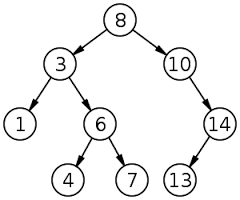
a) Binary Search Tree is used on server side to parse based on client input

b) ActiveMQ is used as a message broker for client and server communication.

c) The client and server uses “*JMS Queue & Topic*” to send requests and responses.

d) Request and Response are tracked by correlation ID. Each time client checks the correlation ID when it receives the message.

e) Below tree is used in this project.



f) Tree operations implemented on server side are

- Pre Order Traversal

- In Order Traversal

- Post Order Traversal

- Search Node

- Size of Binary Search Tree

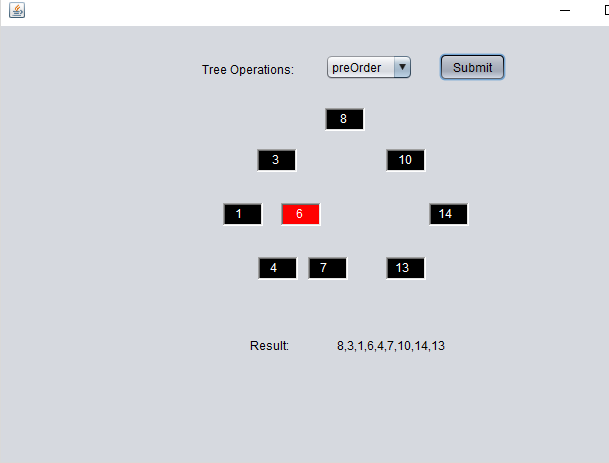
- Sum of all nodes in Binary Search Tree

- Maximum Node

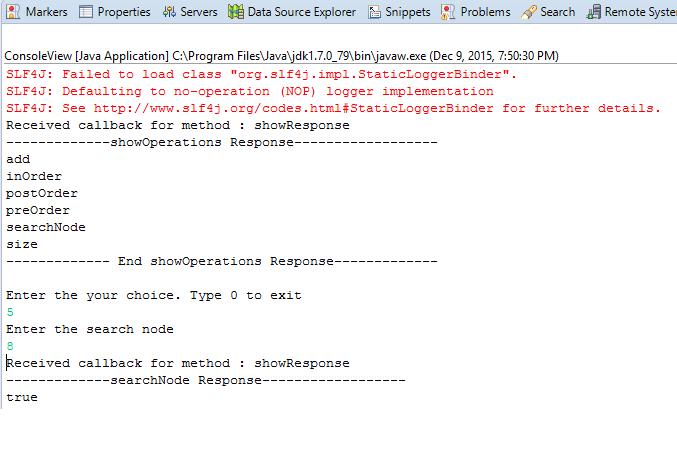
- Minimum Node

g) Two different views are implemented on client side.

- Swing View: Client can select the type of traversal from the drop down. On submit, server processes the request & sends back the response.



- Console View: Client will be listed with different tree operations from the server. Client can select any of those tree operations. On submit, server processes the request & sends back the response.



**3. Project Execution**

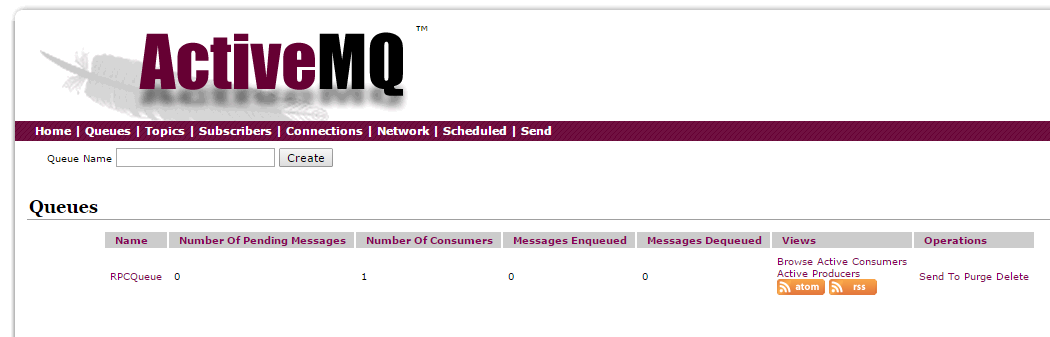
a) Install ActiveMQ

b) Import Client & Tree Parser Projects into eclipse.

c) ActiveMQ Connection URL, Queue & topic names are maintained in app.properties file. Modify if required

d) Build both projects using maven

e) Run com.rpc.server.Server.java class. On startup, binary search tree will be initialized and server starts listening to the queue. On execution queue will be created with 1 consumer. Below is the screen shot for reference.



In above screen shot, queue (RPCQueue) got created with 1 consumer & 0 messages.

f) Run com.rpc.client.Client.java class. It sends all the tree operations as requests. Below is the screen shot for reference.



In above screen shots, messages sent from the client are 7 and server processes messages are 7.

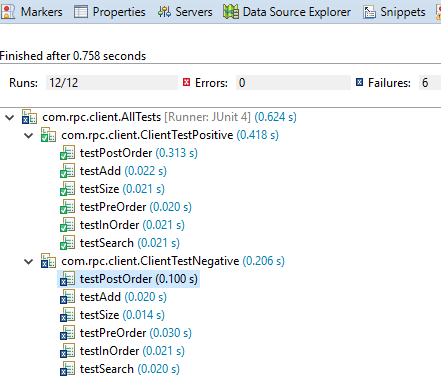
**3. Test Case Execution**

a) Client

- Positive & Negative test cases are written to the test request & response.

- Before running the client test cases, run the com.rpc.server.Server.java

- Once the server is started. Run com.rpc.client.AllTests.java. Below is the screen shot for reference.



b) Server

- Positive & Negative test cases are written to test the binary search tree operations

- Run com.rpc.server.Client.AllTests.java. Below is the screen shot for reference.

