CS5300 Project 1b - Submitted March 31 2013

Chantelle Farmer (csf25), Ben Perry (bap63) & Matthew Green (mcg67)

Running the Project

You can run our assignment by deploying the compiled WAR file to a tomcat 7 server on the Amazon Elastic Beanstalk. It is important to note that you will have to adjust the environment for this project to work properly by changing the min and max server instances to 4 or some number greater than 1.

You may also view the project by visiting our elastic beanstalk environment:
http://sandbox-env2.elasticbeanstalk.com/Controller

This link will take you to our most recent version, the one we submitted.

Overall Structure

- The session ID is generated from a timestamp appended to the client's IP address, with all special characters removed. This format guarantees that it will be unique even across multiple session stores.
- Cookies contain this session ID so the server can access an existing session.
- Sessions are stored in a ConcurrentHashMap structure, keyed on session ID.
- Sessions are timed out after 10 minutes of inactivity.
- Every 30 minutes, a cleanup daemon removes expired entries from the session table.

Cookie Format

The data stored in our cookie consists of 4 elements, delimited by a "#": sessionID, version, locations
sessionID = the unique ID for the session, created as described above
version = the session version #, which starts with 1 and is incremented with each request
locations = server(s) which store the session data.

Session Table Design

The session table is stored in a ConcurrentHashMap structure, keyed on session ID. This structure is threadsafe, so does not require us to provide any external locking mechanisms.
The format of our session table data is an array of strings: data, version, expires, expiresTimestamp
data = the message that is displayed to the user at the top of the page
version = the session version ID, consistent with what is stored in the cookie
expires = a string representation of the session expiration time
expiresTime = a timestamp representation of the expiration

Description of the Files

We have seven files that are used in this Dynamic Web Project separated into three packages. The first is our
session package which is handles the HTTP requests, the creation and storage of sessions and the cleanup of old sessions. The second is our serverblocks package which handles the creation of server objects and their management. Finally, there is the remote procedure call class, rpc that facilitates communication to and from server instances of session data. Each file is detailed further below.

[session] Controller.java

The controller in this assignment takes on the function of both the View and the Controller (from MVC architecture). This class is what receives GET and POST requests from a client, creates the session object(our storage mechanism), as well as handles the modifications and issue of our custom cookie 'CS5300PROJ1SESSION'. This class is also responsible for the generation of HTML (aka the web form / UI) and for invoking the daemon cleanup process for the hashmap (see Cleanup.java).

[session] Session.java

Session defines our object whereby client sessions are stored. This class creates our session table as well as session objects. When a user issues a GET request to our site, the controller creates a new cookie that is passed back and forth between client and server containing a unique session ID. That session ID is used as a key to store and retrieve the 'message' from the ConcurrentHashMap in this class. The Session class handles all major operations relating to the table except for cleanup, which is defined below.

[session] Cleanup.java

This file essentially extends the timer class in Java and sets up a service which checks our ConcurrentHashMap, reads through it and compares the time NOW to the timestamp in the table. If the difference is greater than 600 seconds (10 minutes), the entry is removed and space in the table is freed. This process is invoked by the constructor of the Controller.

[serverblocks] Server.java

This is just a template for a Server instance, containing the fields IP address and port. Server.java also has some comparison, and utility functions to handle objects.

[serverblocks] ServerManager.java

This class handles the insertion, maintained and removal of servers in the server list. Sends a ping to the servers every 10 seconds to ensure they're still online.

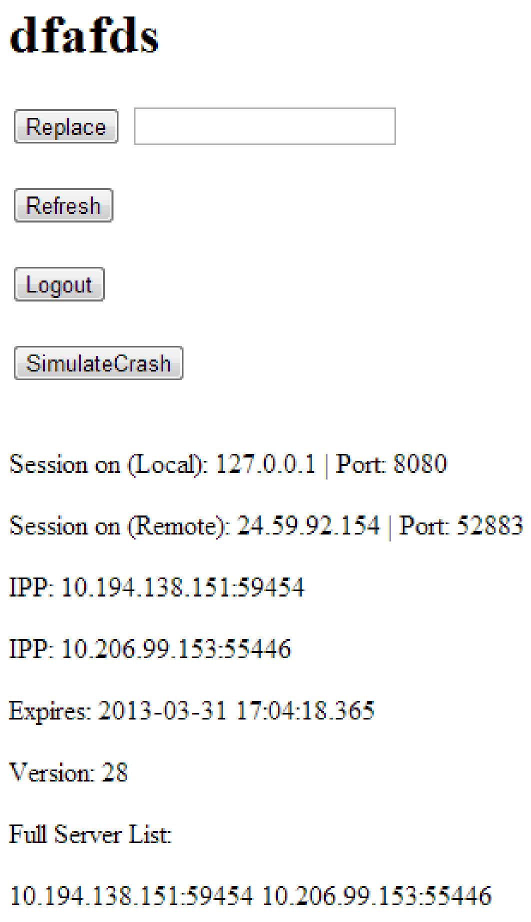
[rpc] rpcClient.java

This class handles RPC calls from the client side with GET and PUT operations.

[rpc] rpcServer.java

This class handles RPC calls from the server side with GET, PROBE, and PUT operations.

Screenshot of Working System



dfafds

Session on (Local): 127.0.0.1 | Port: 8080

Session on (Remote): 24.59.92.154 | Port: 52883

IPP: 10.194.138.151:59454

IPP: 10.206.99.153:55446

Expires: 2013-03-31 17:04:18.365

Version: 28

Full Server List:

10.194.138.151:59454 10.206.99.153:55446

Two servers are currently up. One is listed as the primary for the session, and one is listed as the backup.