CS5300-Project1a - Session Management

Name: FNU Arpana Hosabettu, Net-id : fa97

Run the assignment

- The solution folder contains a war file. Deploying it to tomcat and accessing the url - **http://localhost:8080/SessionManagement/** will display page with a session state and three buttons for Replace, Refresh and Logout.
- Also, for the purpose of viewing, cookie version number and session id and expiration time stamp is displayed at the end of the jsp.
- The events on the button clicks are as mentioned in the assignment description. The screenshots for the same are part of the solution archive.

Overall Structure:

The structure of the solution looks like this:

Package ***edu.cornell.cs5300.fa97.proj1.core*** contains

CustomCookie class - This is the cookie class for this project. It extends http.Cookie. This is explained in details in the below sections.

CustomSession class - This is the session class for this project. This is explained in details in the below sections.

LocationData class - This is for the purpose of having location addresses in the later extensions of the project in partb. Currently does not do anything.

SessionTable - This is the class for maintaining session table. This extends ConcurrentHashMap for now. This is explained in detail below section.

Package ***edu.cornell.cs5300.fa97.proj1.servlet*** contains

SessionManagementServlet - this is the servlet class that extends HttpServlet.

- This includes method *doGet*, *doPost* and a daemon method *startSessionCleanupDaemon*

A jsp file called customSession.jsp inside WebContent

Design of Session Table

Session table is designed as a class that extends *ConcurrentHashMap* which store session id as key and the Session object as its value.

```
public class SessionTable extends ConcurrentHashMap<String, CustomSession>
```

This class also implements functions like

- *getSession(String sessionID)* - which returns the session object.
- getSessionCookie(String sessionID) - return cookie object corresponding to the session represented by session id.

- *invalidate(String sessionID)* - which calls the invalidation on session object corresponding to session id which sets the max age of cookie as 0. It then deletes the session entry from session table.
- *cleanExpiredSessions()*- this method iterates through all the session entries and checks if the expiration time stamp has expired for any session, if so it removes the entry from session table. This is called from daemon.

It also implements various other functions to update session state, refresh session, create new session etc.

Design of Session

CustomSession is the name of the class that represents the Session.

It contains fields *sessionId, sessionState, sessionVersion, creationTime, lastAccessedTime, expirationTimeStamp*. It also encompasses *CustomCookie* which is the custom cookie class for this project.

It includes methods like :

- *constructor* - which creates a new session object. This method also create a new cookie. The session id is generated as *UUID.randomUUID()* and retains on numeric value. Version number is set to 1 and session state to "Hello User". It sets the creation time, last accessed time and also sets the expiration time stamp which is used to check if the session ahs expired.
- *refresh()* - Resets cookie max age, resets the expiration time stamp by adding cookie max age, increments version number.

It includes few other method to update session state and getters.

Design of Cookie

CustomCookie class extend http.Cookie. This class has fields - sessionID, version number, LocationData[] which represents location metadata for use in later part. This is the class that represent the tuple

public class CustomCookie **extends** Cookie

Cookie max age is 180 seconds.

Garbage Collection of timed out session

The expired sessions are removed from session table by calling *cleanExpiredSessions()* method (explained above) on session table.

This method is called every 3 minutes by a daemon method - *startSessionCleanupDaemon()* which is present in servlet class. This is implemented as Runnable thread to invoke *cleanExpiredSessions()* on session table.

Name: **FNU, Arpana Hosabettu** Net-id: **fa97**

CS5300-Project1a - Session Management

Working

- When the server is stared, ***doGet*** method is called on the servlet. This method checks if the request contains the cookie by name - **CS5300PROJ1.** As it is not, it calls sessionTable.createNewSession() to create a new session and returns the jsp page along with new cookie.
- The text box next to Replace button is limited to 200 characters(nearly 512 bytes). This when populated and Replace is hit, it calls the **doPost** method. It gets the session id from cookie and gets the session object form session table and replaces the session state. It also refreshes the session.
- The refresh button just refreshes the session(increment version number in session and cookie, update last accessed time, set max cookie age.)
- The logout button invokes invalidate on session table which reset cookie max age to 0, deletes the session from session table and send back the user the cookie with max age set to 0.
- If the session is idle, then the session expires. This happens as the daemon thread that runs every 3 minutes checks for expired session by calling method *cleanExpiredSessions() on session table.*

Name: **FNU, Arpana Hosabettu** Net-id: **fa97**