TASK 1

How did you use connection pooling?

Our project uses connection pooling by initializing our connection resource in our \META-INF\context.xml file and \WEB-INF\web.xml file. We call one resource our “jdbc/TestDB” which is used for reading from either master or slave’s local database. The second resource is “jdbc/MasterDB” this resource is used for writing to the master database. By using connection pooling we have our connections cached for easy reuse for further use in the project rather than closing and reopening a connection to a database over and over. The following images are snapshots of where are connection and prepared statements can be found.

context.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <Context>
4
5
6
7     <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
8               maxTotal="100" maxIdle="30"
9               maxWaitMillis="10000"
10              username="mytestuser"
11              password="mypassword"
12              driverClassName="com.mysql.jdbc.Driver"
13              url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
14
15    <Resource name="jdbc/MasterDB" auth="Container" type="javax.sql.DataSource"
16              maxTotal="100" maxIdle="30"
17              maxWaitMillis="10000"
18              username="mytestuser"
19              password="mypassword"
20              driverClassName="com.mysql.jdbc.Driver"
21              url="jdbc:mysql://52.53.250.44:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
22
23 </Context>
```

src/default/BrowseByGenre.java - line 56-63

BrowseByGenre.java

```
42      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
43      */
44     protected void doGet(HttpServletRequest request, HttpServletResponse response) thro
45         // TODO Auto-generated method stub
46         response.getWriter().append("Served at: ").append(request.getContextPath());
47         String genre= (String)request.getParameter("genre");
48         String genreId="";
49
50         //Connection con=DBConnection.getConnection();
51
52         PreparedStatement statement = null;
53         try {
54             //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
55
56             Context initCtx = new InitialContext();
57
58             Context envCtx = (Context) initCtx.lookup("java:comp/env");
59
60             // Look up our data source
61             DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
62
63             Connection con=ds.getConnection();
64
65             //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
66
67             String query="";
68             ResultSet result = null;
```

src/default/BrowseByTitle - line 53-60

BrowseByTitle.java

```
46
47
48          PreparedStatement statement = null;
49          try {
50
51              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
52
53              Context initCtx = new InitialContext();
54
55              Context envCtx = (Context) initCtx.lookup("java:comp/env");
56
57              // Look up our data source
58              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
59
60              Connection con=ds.getConnection();
61            //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
62
63
64              String query="";
```

src/default/GenreServlet - lines 73-80

GenreServlet.java

```
68                  // create database connection
69                  //Connection connection = DBConnection.getConnection();
70
71              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
72
73              Context initCtx = new InitialContext();
74
75              Context envCtx = (Context) initCtx.lookup("java:comp/env");
76
77              // Look up our data source
78              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
79
80              Connection con=ds.getConnection();
81
82              //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
83
```

src/default/insertMovie - lines 58-65

insertMovie.java | insertStar.java

```
38
39    /**
40     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
41     */
42    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExcepti
43        // TODO Auto-generated method stub
44        String name= (String)request.getParameter("title");
45        String year=(String)request.getParameter("year");
46        String director= (String)request.getParameter("director");
47        String star=(String)request.getParameter("star");
48        String genre=(String)request.getParameter("genre");
49
50
51        //Connection con=DBConnection.getConnection();
52        Constants cons= new Constants();
53
54        CallableStatement csmt;
55        try {
56            //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
57
58            Context initCtx = new InitialContext();
59
60            Context envCtx = (Context) initCtx.lookup("java:comp/env");
61
62            // Look up our data source
63            DataSource ds = (DataSource) envCtx.lookup("jdbc/MasterDB");
64
65            Connection con=ds.getConnection();
66
67            //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
68
```

src/default/insertStar
lines 57-64

insertMovie.java | insertStar.java

```
50
51
52        PreparedStatement statement = null;
53
54        try {
55            //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
56
57            Context initCtx = new InitialContext();
58
59            Context envCtx = (Context) initCtx.lookup("java:comp/env");
60
61            // Look up our data source
62            DataSource ds = (DataSource) envCtx.lookup("jdbc/MasterDB");
63
64            Connection con=ds.getConnection();
65
66            //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
67
68
```

src/default/Login - 77-84

Login.java

```
70          try {
71
72              //Class.forName("com.mysql.jdbc.Driver").newInstance();
73
74              //Connection connection = DriverManager.getConnection(loginUrl,loginUser,lo
75              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
76
77              Context initCtx = new InitialContext();
78
79              Context envCtx = (Context) initCtx.lookup("java:comp/env");
80
81              // Look up our data source
82              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
83
84              Connection con=ds.getConnection();
85
86              //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
87
88
```

src/default/LoginServletAndroid
61-68

Login.java | LoginServletAndroid.java

```
52          LoginResponse auth = new  LoginResponse();
53          try {
54
55              //Class.forName("com.mysql.jdbc.Driver").newInstance();
56
57              //Connection connection = DriverManager.getConnection(loginUrl,loginUser,logir
58
59              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
60
61              Context initCtx = new InitialContext();
62
63              Context envCtx = (Context) initCtx.lookup("java:comp/env");
64
65              // Look up our data source
66              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
67
68              Connection con=ds.getConnection();
69
70              //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
71
72
```

src/default/SearchResults - lines 91-98

SearchResults.java

```
83          // Building page body
84          out.println("<body><h1>Movie Titles Found:</h1>");
85
86
87          try {
88
89              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
90
91              Context initCtx = new InitialContext();
92
93              Context envCtx = (Context) initCtx.lookup("java:comp/env");
94
95              // Look up our data source
96              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
97
98              Connection con=ds.getConnection();
99
100             //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
101
102
```

src/default/StarServlet - lines 63-70

StarServlet.java

```
58          out.println("<%@include file=\"navbar.jsp\" %>");
59
60          try {
61              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
62
63              Context initCtx = new InitialContext();
64
65              Context envCtx = (Context) initCtx.lookup("java:comp/env");
66
67              // Look up our data source
68              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
69
70              Connection con=ds.getConnection();
71
72              //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
73
74
```

src/helper/SaleActions - lines 51-58

SaleActions.java

```
42          try {
43
44              java.util.Date date = new java.util.Date();
45
46              //Class.forName("com.mysql.jdbc.Driver").newInstance();
47              //Connection connection = DriverManager.getConnection(loginUrl,loginUser,loginPa
48
49              //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
50
51              Context initCtx = new InitialContext();
52
53              Context envCtx = (Context) initCtx.lookup("java:comp/env");
54
55              // Look up our data source
56              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
57
58              Connection con=ds.getConnection();
59
60              //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
61
```

src/helper/SearchManager - lines 26-33

SearchManager.java

```
1  package helper;
2
3  import java.sql.Connection;
17
18 public class SearchManager {
19     public static ArrayList<MovieListing> getSearchResults(MovieListing movie, String sn
20         ArrayList<MovieListing> list=new ArrayList<>();
21         try{
22             //Connection con=DBConnection.getConnection();
23
24             //P5 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
25
26             Context initCtx = new InitialContext();
27
28             Context envCtx = (Context) initCtx.lookup("java:comp/env");
29
30             // Look up our data source
31             DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
32
33             Connection con=ds.getConnection();
34
35             //P5~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

How did you use Prepared Statements?

Our prepared Statement can be found in our SearchManager as well, lines 43-51

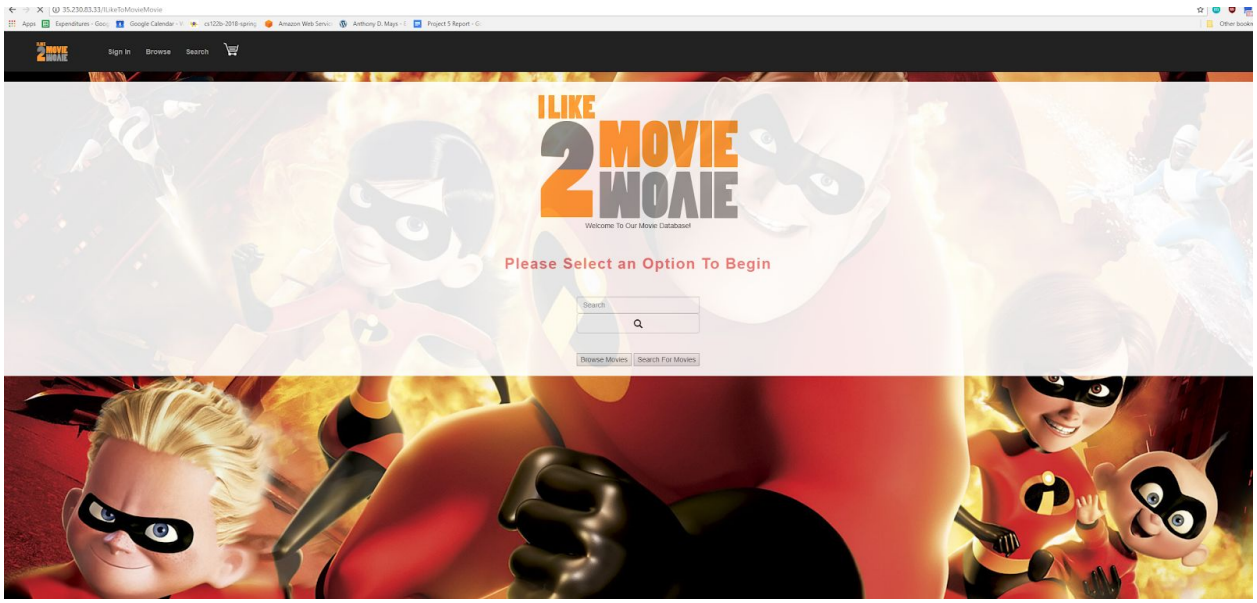
```
43                  PreparedStatement pst=con.prepareStatement(query);
44                  pst.setString(1, "%"+movie.getTitle()+"%");
45                  if(movie.getYear()!=null){
46                          pst.setString(2, "%"+movie.getYear().toString()+"%");
47                  }else{
48                          pst.setString(2, "%%");
49                  }
50                  pst.setString(3, "%"+movie.getDirector()+"%");
51                  pst.setString(4, "%"+sn+"%");
```

Task 2
Address of our AWS and Google Instance

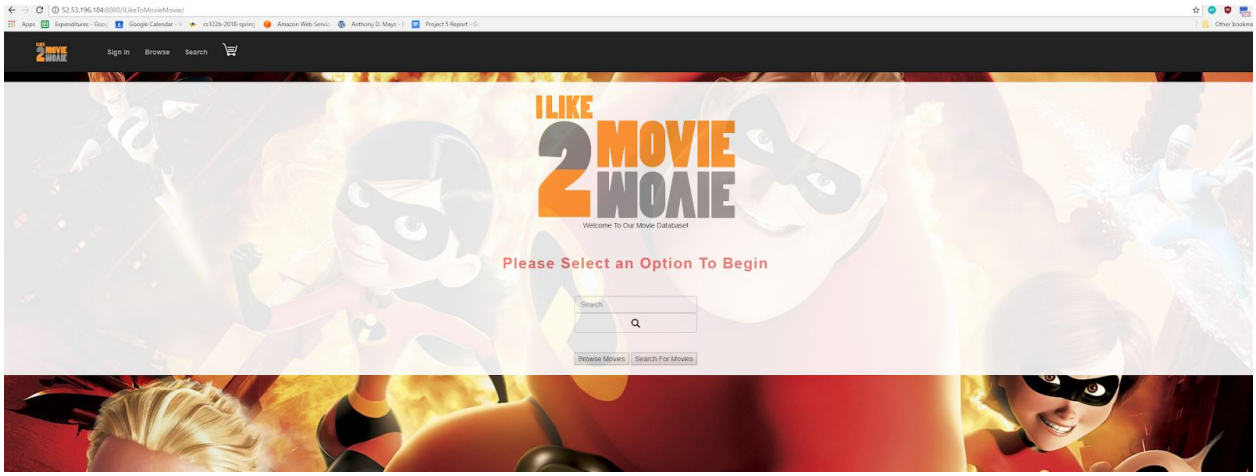
Project(Original)	52.53.196.184
Master	13.57.223.173
Slave	54.183.155.49
Google Cloud	35.230.83.33

Have you verified that they are accessible? Does Fablix site get opened both on Google's 80 port and AWS' 8080 port?

Fablix on Google 80: http://35.230.83.33/ILikeToMovieMovie



Fablix on AWS 8080: http://52.53.196.184:8080/ILikeToMovieMovie/



Explain how connection pooling works with two backend SQL (in your code)?

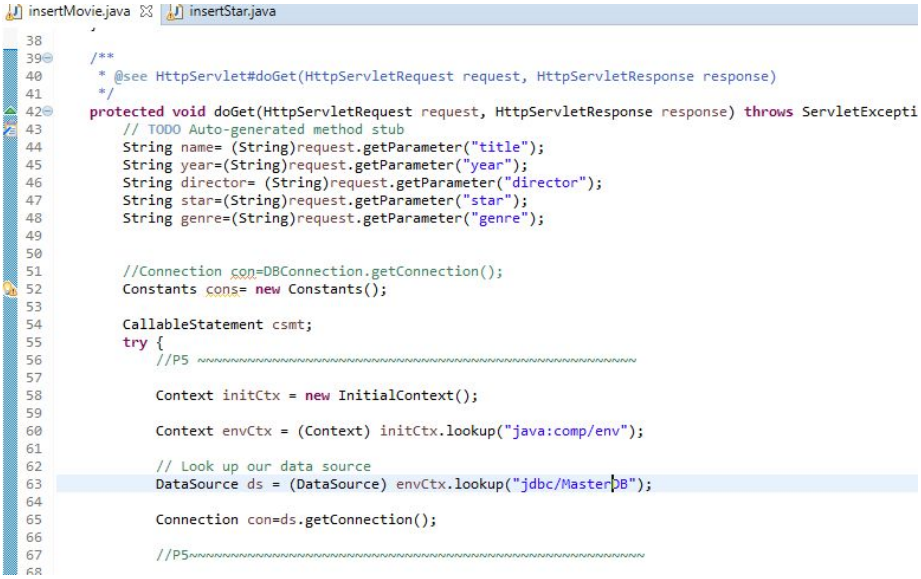
Connection pooling works by our load balancer which switches queries to the database from master and slave in order to avoid all queries being made to a single database.The balancer can be found in our instance's etc/apache2/sites-enabled/000-default.conf folder

```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_
ROUTE_CHANGED

<Proxy "balancer://Session_balancer">
    #Master
    BalancerMember "http://13.57.223.173:8080/Session" route=1

    #Slave
    BalancerMember "http://54.183.155.49:8080/Session" route=2
ProxySet stickysession=ROUTEID
</Proxy>

<Proxy "balancer://TomcatTest_balancer">
    #master
    BalancerMember "http://13.57.223.173:8080/TomcatTest/"

    #slave
    BalancerMember "http://54.183.155.49:8080/TomcatTest/"
</Proxy>

<Proxy "balancer://ILikeToMovieMovie_balancer">
    #Master
    BalancerMember "http://13.57.223.173:8080/ILikeToMovieMovie" route=1

    #Slave
    BalancerMember "http://54.183.155.49:8080/ILikeToMovieMovie" route=2
ProxySet stickysession=ROUTEID
</Proxy>

<VirtualHost *:80>

        ProxyPass /TomcatTest balancer://TomcatTest_balancer
        ProxyPassReverse /TomcatTest balancer://TomcatTest_balancer

        ProxyPass /Session balancer://Session_balancer
        ProxyPassReverse /Session balancer://Session_balancer

        ProxyPass /ILikeToMovieMovie balancer://ILikeToMovieMovie_balancer
        ProxyPassReverse /ILikeToMovieMovie balancer://ILikeToMovieMovie_balance
r
```

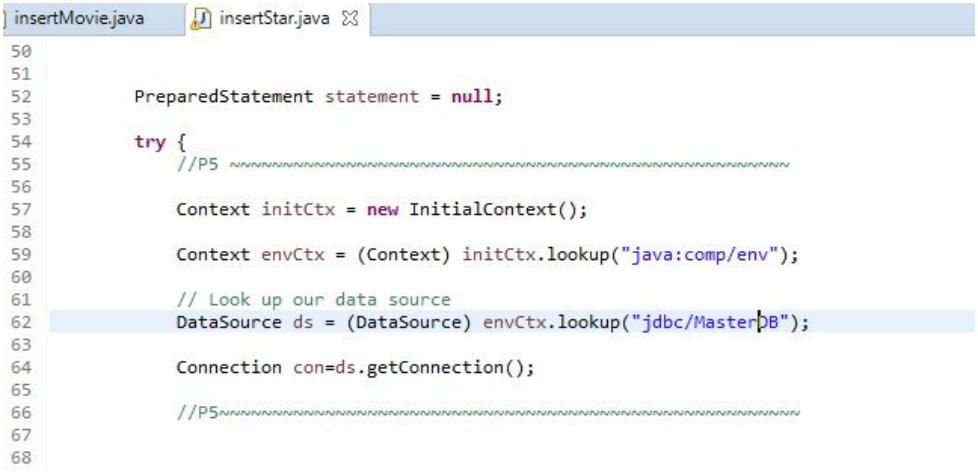
How read/write requests were routed?

Write requests were routed to the MASTERDB while reads were routed to TestDB these can be found in our insertMovie.java and insertStar.java

src/default/insertMovie - lines 58-65



src/default/insertStar
lines 57-64



TASK 3

Have you uploaded the log files to Github? Where is it located?
The log files are found in the folder Reports. The text files are the logs for each of the tests.

Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
The HTML file is also located in the Reports folder called jmeter-report.html

Have you uploaded the script to Github? Where is it located?
Our log script is found in the src/helper folder called LogFileParser.java.

Have you uploaded the WAR file and README to Github? Where is it located?
The WAR file can be found in the default folder, it is called ILikeToMovieMovie.war

In this task, we prepared (1) the average time it takes for the search servlet to run completely for a query (called TS), and (2) the average time spent on the parts that use JDBC, per query (called TJ)

- We prepared log statements for our main search servlet.

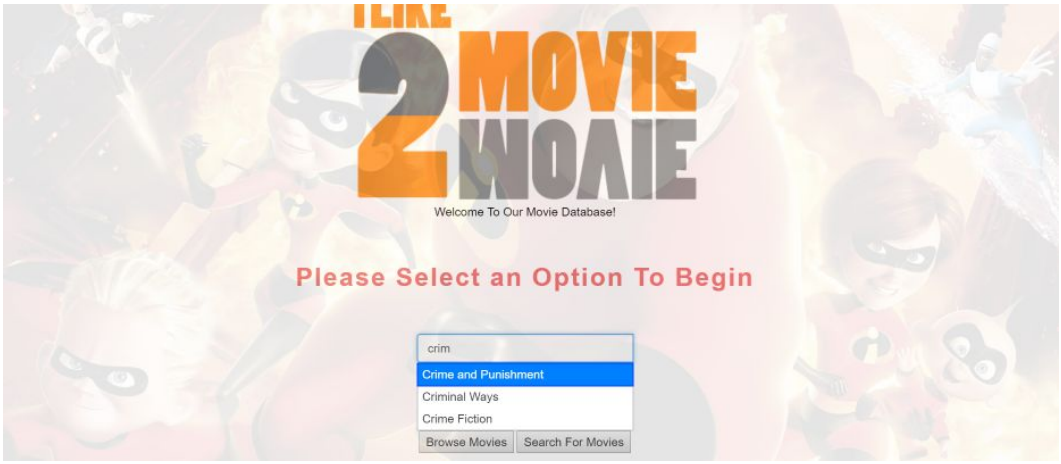
AdvanceSearch.java | LogFileParser.java

```
 1
 2 import java.io.BufferedWriter;
26
27
28
29 /**
30  * Servlet implementation class AdvanceSearch
31  */
32 @WebServlet("/AdvanceSearch")
33 public class AdvanceSearch extends HttpServlet {
34
35     long startTimeTS = System.nanoTime();
36     private static final long serialVersionUID = 1L;
37
38     /**
39      * @see HttpServlet#HttpServlet()
40      */
41
```

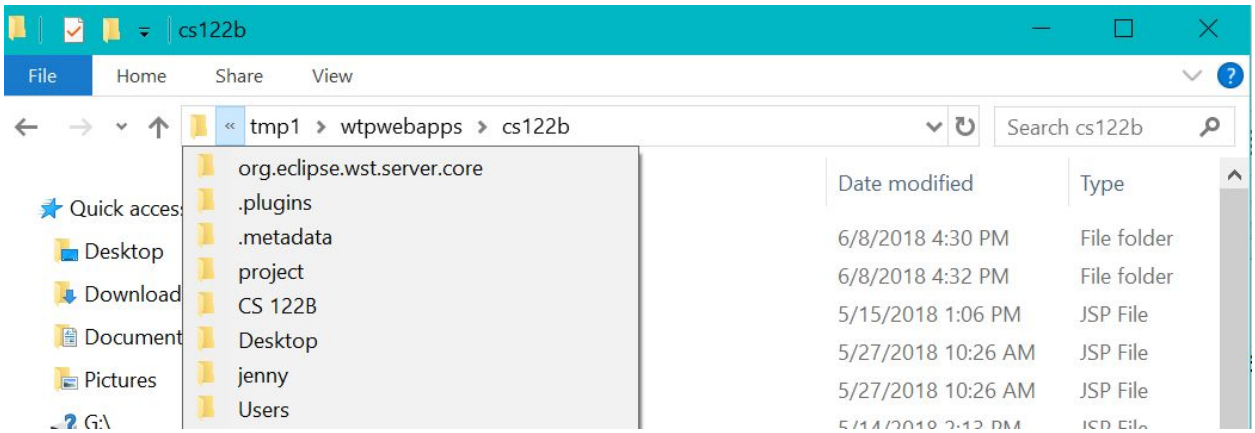
AdvanceSearch.java | LogFileParser.java

```
68         query = query.toLowerCase();
69         long startTimeTJ = System.nanoTime();
70         ArrayList<MovieResult> list=SearchManager.getAdvanceSearchResults(query);
71         long endTimeTJ = System.nanoTime();
72         long TJ = endTimeTJ - startTimeTJ;
73         String json = new Gson().toJson(list);
74
75         out.println(json);
76         out.close();
77         long endTimeTS = System.nanoTime();
78         long TS = endTimeTS - startTimeTS;
79
80         String contextPath = getServletContext().getRealPath("/");
81         String xmlFilePath = contextPath + "\\log.txt";
82         System.out.println(xmlFilePath);
83         File logFile = new File(xmlFilePath);
84         Writer writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(logFile, true), "UTF-8"));
85         if(logFile.exists()) {
86             writer.append(TS+","+TJ+"\n");
87             writer.close();
88         }else {
89             logFile.createNewFile();
90             writer.write(TS+","+TJ+"\n");
91             writer.close();
92         }
```

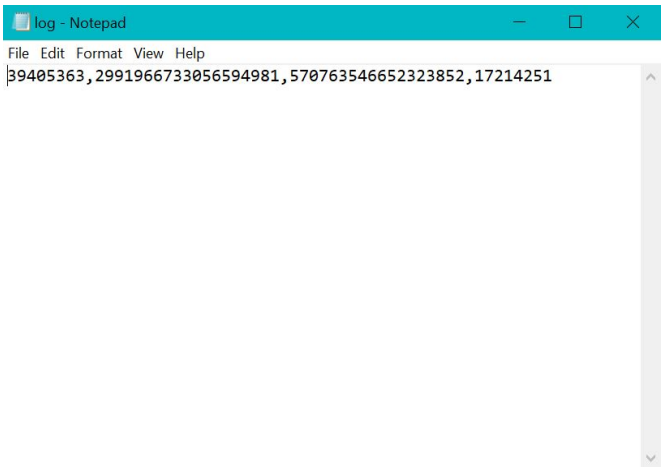
- To write to a log.txt,launch the webapp and attempt search activities.



- When running from eclipse, the log.txt file gets written to a temporary location. In this specific case, this is where my log.txt was located:



- However, once you deploy your application to the Apache Tomcat servlet container everything will work as expected and file will be created in the root folder of the webapp.
- Once the log is located, it should contain sets of numbers (TS + TJ) , similar to the follow:



- We have also generated a Log Parser as shown:

- On line 15 of the servlet, you must insert the path which your log.txt file is located at

- The console will generate the average TS and TJ

INSTRUCTIONS FOR README:

To run the scripts to parse the log files:

1) Run the webapp and attempt search activities, which should be the search bar located on index.jsp.
2) Locate the log.txt file, which should be somewhere within \workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp1\wtpwebapps\ If you're running the files using Eclipse, or in the WebContent folder (root) when deployed on a Tomcat servlet container.
3) Copy the directory path and paste it in LogFileParser.java, on line 15, where File file = new File("INSERT_PATH_HERE\\log.txt")
4) Run LogFileParser.java to generate Average TS and Average TJ

Refer to "TASK 3" of the report for snapshots and more detailed instructions.