**Task 1**

- How did you use connection pooling?
  We used connection pooling in all our servlets where we make a connection to the database. Now there will be only one connection to our database when the website is used, instead of before open a new connection each time we needed to connect to the database.

  - File name, line numbers as in Github

| | |
|---|---|
| /cs122bproject/src/AddMovie.java | lines 65-78 |
| /cs122bproject/src/AddNewStar.java | lines 62-75 |
| /cs122bproject/src/AdvSearch.java | lines 49-62 |
| /cs122bproject/src/AutoComplete.java | lines 53-66 |
| /cs122bproject/src/CustomerInfo.java | lines 66-79 |
| /cs122bproject/src/EmployeeDashboard.java | lines 58-71 |
| /cs122bproject/src/GenreList.java | lines 58-71 |
| /cs122bproject/src/Login.java | lines 71-84 |
| /cs122bproject/src/MovieList.java | lines 58-71 |
| /cs122bproject/src/MoviePage.java | lines 42-55 |
| /cs122bproject/src/Search.java | lines 46-59 |
| /cs122bproject/src/ShowMovies.java | lines 65-78 |
| /cs122bproject/src/StarPage.java | lines 42-55 |

  - Snapshots

```
64
65                      //CONNECTION POOLING
66                      Context initCtx = new InitialContext();
67                      Context envCtx = (Context) initCtx.lookup("java:comp/env");
68              if (envCtx == null)
69                  out.println("envCtx is NULL");
70
71              DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
72
73              if (ds == null)
74                  out.println("ds is null.");
75
76              Connection dbcon = ds.getConnection();
77              if (dbcon == null)
78                  out.println("dbcon is null.");
79
```

- How did you use Prepared Statements?
  We changed our regular Statements to Prepared Statements. Since the website used the search feature the most, Prepared Statements would make the search faster, since there is already a template in the database.

    - File name, line numbers as in Github
      /cs122bproject/src/AdvSearch.java lines 43-45, 64-74, 118-134, 136-187
      /cs122bproject/src/AutoComplete.java lines 26-27, 49-50, 69-74, 100-112, 134-146
      /cs122bproject/src/Search.java lines 25-27, 47-49, 69-78, 192-202

    - Snapshots

AdvSearch.java

```
64              dbcon.setAutoCommit(false);
65              statement = dbcon.prepareStatement(selectString);
66              genreStatement = dbcon.prepareStatement(genreString);
67              starStatement = dbcon.prepareStatement(starString);
68
69              int newItems = createItems(items);
70                          int offset = createOffset(newItems, page);
71                          setQuery(newItems, offset);
72
73                          ResultSet rs = statement.executeQuery();
74                          dbcon.commit();
75                          JsonArray jsonArray = new JsonArray();
76
77                          while (rs.next()) {
78                                  String m_id = rs.getString("M.id");
79                                  String m_title = rs.getString("M.title");
80                                  int m_year  = rs.getInt("M.year");
81                                  String m_director = rs.getString("M.director");
82                                  String genreList = createQueryList(genreStatement, m_title, "G.name");
83                                  String starList = createQueryList(starStatement, m_title, "S.name");
84
85                                  JsonObject jsonObject = new JsonObject();
86                                  jsonObject.addProperty("movie_id", m_id);
87                                  jsonObject.addProperty("movie_title", m_title);
88                                  jsonObject.addProperty("movie_year", m_year);
89                                  jsonObject.addProperty("movie_director", m_director);
90                                  jsonObject.addProperty("genre_list", genreList);
91                                  jsonObject.addProperty("star_list", starList);
92
93                                  jsonArray.add(jsonObject);
94                          }
```

```
118         public String createStatementQuery() {
119                 String query = "SELECT DISTINCT M.id, M.title, M.year, M.director ";
120                 if (!name.isEmpty()) {
121                         query += "FROM movies M, stars S, stars_in_movies SIM WHERE ";
122                 }
123                 else {
124                         query += "FROM movies M WHERE ";
125                 }
126
127                 query += appendQuery(title, year, director, name);
128
129                 if(sortByTitle != null || sortByYear != null) {
130                         query += appendOrderBy(sortByTitle, sortByYear);
131                 }
132                 query += "LIMIT ? OFFSET ?";
133                 return query;
134         }
135
136         public String appendQuery(String myTitle, String myYear, String myDir, String myName) {
137                 String query = "";
138                 if (!myTitle.isEmpty()) {
139                         String titleConstraint = " M.title = ?";
140                         query += titleConstraint;
141                 }
142                 if (!myYear.isEmpty()) {
143                         String yearConstraint = " M.year = ?";
144                         if (myTitle != "")
145                                 query += " AND";
146                         query += yearConstraint;
147                 }
148                 if (!myDir.isEmpty()) {
149                         String directorConstraint = " M.director = ?";
150                         if (myTitle != "" || myYear != "")
151                                 query += " AND";
152                         query += directorConstraint;
153                 }
154                 if (!myName.isEmpty()) {
155                         String nameConstraint = " S.name = ?";
156                         if (myTitle != "" || myYear != "" || myDir != "")
157                                 query += " AND";
158                         query += nameConstraint;
159                         query += "AND SIM.movieId = M.id AND SIM.starId = S.id";
160                 }
161                 return query;
162         }
163
164         public void setQuery(int items, int offset) throws SQLException {
165                 int count = 0;
166                 if (!title.isEmpty()) {
167                         count++;
168                         statement.setString(count, title);
169                 }
170                 if (!year.isEmpty()) {
171                         count++;
172                         int intYear = Integer.parseInt(year);
173                         statement.setInt(count, intYear);
174                 }
175                 if (!director.isEmpty()) {
176                         count++;
177                         statement.setString(count, director);
178                 }
179                 if (!name.isEmpty()) {
180                         count++;
181                         statement.setString(count, name);
182                 }
183                 count++;
184                 statement.setInt(count, items);
185                 count++;
186                 statement.setInt(count, offset);
187         }
```

AutoComplete.java

```
26          private PreparedStatement movieStatement = null;
27          private PreparedStatement starStatement = null;

49                  String movieString = createMovieQuery();
50                  String starString = createStarQuery();

69                dbcon.setAutoCommit(false);
70                            movieStatement = dbcon.prepareStatement(movieString);
71                            starStatement = dbcon.prepareStatement(starString);
72
73                            createJson(query, movieStatement, jsonArray, "Movie");
74                            createJson(query, starStatement, jsonArray, "Star");

100         public String createMovieQuery() {
101                 return "SELECT id, title " +
102                             "FROM movies " +
103                             "WHERE MATCH(id, title) AGAINST(? IN BOOLEAN MODE) " +
104                             "LIMIT 5";
105         }
106
107         public String createStarQuery() {
108                 return "SELECT id, name " +
109                             "FROM stars " +
110                             "WHERE MATCH(id, name) AGAINST(? IN BOOLEAN MODE) " +
111                             "LIMIT 5";
112         }

134     public void createJson(String query, PreparedStatement statement, JsonArray jArray, String category) throws SQLException {
135             String FTQuery = createFullTextQuery(query);
136             statement.setString(1, FTQuery);
137             ResultSet rs = statement.executeQuery();
138             dbcon.commit();
139             String name = makeName(category);
140             while (rs.next()) {
141                     String m_id = rs.getString("id");
142                     String m_name = rs.getString(name);
143                     jArray.add(generateJsonObject(m_id, m_name, category));
144             }
145             rs.close();
146     }
```

Search.java

```
84              dbcon.setAutoCommit(false);
85                      statement = dbcon.prepareStatement(selectString);
86                      genreStatement = dbcon.prepareStatement(genreString);
87                      starStatement = dbcon.prepareStatement(starString);
88
89                      String query = createFullTextQuery(searchParam);
90                      int newItems = createItems(items);
91                      int offset = createOffset(newItems, page);
92                      statement.setString(1, query);
93                      statement.setInt(2,  newItems);
94                      statement.setInt(3, offset);
95
96                      ResultSet rs = statement.executeQuery();
97                      dbcon.commit();
98                      endTime = System.nanoTime();
99                      long psElaspedTime = endTime - startTime;
100                     JsonArray jsonArray = new JsonArray();
101                     int count = 0;
102                     while (rs.next()) {
103                             count++;
104                             String m_id = rs.getString("id");
105                             String m_title = rs.getString("title");
106                             int m_year  = rs.getInt("year");
107                             String m_director = rs.getString("director");
108                             String genreList = createQueryList(genreStatement, m_title, "G.name");
109                             String starList = createQueryList(starStatement, m_title, "S.name");
110
111                             JsonObject jsonObject = new JsonObject();
112                             jsonObject.addProperty("movie_id", m_id);
113                             jsonObject.addProperty("movie_title", m_title);
114                             jsonObject.addProperty("movie_year", m_year);
115                             jsonObject.addProperty("movie_director", m_director);
116                             jsonObject.addProperty("genre_list", genreList);
117                             jsonObject.addProperty("star_list", starList);
118
119                             jsonArray.add(jsonObject);
120                     }
```

```
192         public String createStatementQuery() {
193                 String query = "SELECT id, title, year, director " +
194                                                 "FROM movies " +
195                                                 "WHERE MATCH(id, title) AGAINST(? IN BOOLEAN MODE) ";
196
197                 if(sortByTitle != null || sortByYear != null) {
198                         query += appendOrderBy(sortByTitle, sortByYear);
199                 }
200                 query += "LIMIT ? OFFSET ?";
201                 return query;
202         }
```

**Task 2**

- Address of AWS and Google instances

AWS: http://18.144.8.219/cs122bproject/
Google: http://35.230.34.254/cs122bproject/

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

- How connection pooling works with two backend SQL?
  We added another datasource, in addition to the localhost, associated with the master's IP. In the servlets where writing is required, we changed the datasource to use the master's IP. In all the other servlets where only reads are being done we kept the datasource as localhost, which with the load balancer will distribute the reads to the master and the slave.

  - File name, line numbers as in Github
    /cs122bproject/WebContent/META-INF/context.xml lines 8-11
    /cs122bproject/WebContent/WEB-INF/web.xml lines 40-54
    /cs122bproject/src/AddMovie.java lines 65-78
    /cs122bproject/src/AddNewStar.java lines 62-75

  - Snapshots

context.xml

```
8         <Resource name="jdbc/MasterDB" auth="Container" type="javax.sql.DataSource"
9                 maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
10                password="mypassword" driverClassName="com.mysql.jdbc.Driver"
11                url="jdbc:mysql://172.31.10.118:3306/MovieDB?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true" />
12
```

web.xml

```
40      <resource-ref>
41                  <description>
42                          Resource reference to a factory for java.sql.Connection
43                          instances that may be used for talking to a particular
44                          database that
45                          is configured in the server.xml file.
46                  </description>
47      <res-ref-name>
48                          jdbc/MasterDB
49                  </res-ref-name>
50      <res-type>
51                          javax.sql.DataSource
52                  </res-type>
53      <res-auth>Container</res-auth>
54      </resource-ref>
```

AddMovie.java & AddNewStar.java

```
65                          //CONNECTION POOLING
66                          Context initCtx = new InitialContext();
67                          Context envCtx = (Context) initCtx.lookup("java:comp/env");
68              if (envCtx == null)
69                  out.println("envCtx is NULL");
70
71              DataSource ds = (DataSource) envCtx.lookup("jdbc/MasterDB");
72
73              if (ds == null)
74                  out.println("ds is null.");
75
76              Connection dbcon = ds.getConnection();
77              if (dbcon == null)
78                  out.println("dbcon is null.");
```

- How read/write requests were routed?
  For our website we separated each function into their own servlet, so in servlets where we write to the database, we changed the datasource to the one associated with the master IP, so the writes will only be done by the master. In the servlets where reads were being done we kept the datasource as localhost, so the read will be done by either the master or the slave.

    - File name, line numbers as in Github
      /cs122bproject/src/AddMovie.java lines 65-78
      /cs122bproject/src/AddNewStar.java lines 62-75
    - Snapshots

AddMovie.java & AddNewStar.java

```
65                          //CONNECTION POOLING
66                          Context initCtx = new InitialContext();
67                          Context envCtx = (Context) initCtx.lookup("java:comp/env");
68              if (envCtx == null)
69                  out.println("envCtx is NULL");
70
71              DataSource ds = (DataSource) envCtx.lookup("jdbc/MasterDB");
72
73              if (ds == null)
74                  out.println("ds is null.");
75
76              Connection dbcon = ds.getConnection();
77              if (dbcon == null)
78                  out.println("dbcon is null.");
```

**Task 3**

- Have you uploaded the log file to Github? Where is it located?
  Yes, it is located at /cs122bproject/WebContent/TimeResults/SearchLog<#>.txt

- Have you uploaded the HTML file to Github? Where is it located?
  Yes, it is located at /cs122bproject/WebContent/TimeResults/jmeter_report.html

- Have you uploaded the script to Github? Where is it located?
  Yes, it is located at /cs122bproject/WebContent/TimeResults/GetTimeResults.java

- Have you uploaded the WAR file and README to Github? Where is it located?
  Yes, the WAR file is located at /cs122bproject/WebContent/cs122bproject.war and the README is located at /cs122bproject/WebContent/README.txt