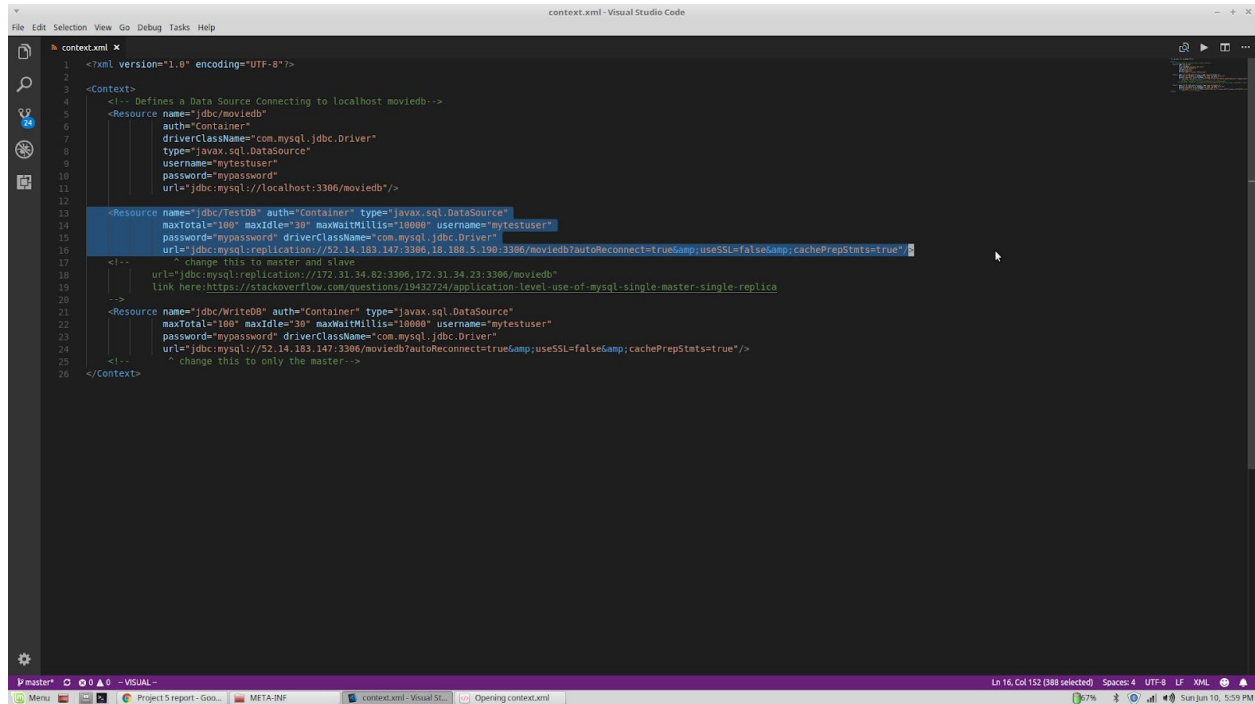


Task 1

- How did you use connection pooling?

I used connection pooling in to distribute the load on the mySQL transaction. The read is distribute to my master and my slave mySQL instances. In my context.xml, i used a replication in order to distribute the load to the slave and master. Shown here:



```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <!-- Defines a Data Source Connecting to localhost moviedb-->
  <Resource name="jdbc/moviedb"
    auth="Container"
    driverClassName="com.mysql.jdbc.Driver"
    type="javax.sql.DataSource"
    username="mytestuser"
    password="mypassword"
    url="jdbc:mysql://localhost:3306/moviedb"/>

  <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
    password="mypassword" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql:replication://52.14.183.147:3306,18.188.5.196:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"
    <!-- change this to master and slave -->
    url="jdbc:mysql:replication://172.31.34.62:3306,172.31.34.23:3306/moviedb"
    <!-- change this to master and slave -->
    link href="https://stackoverflow.com/questions/19432724/application-level-use-of-mysql-single-master-single-replica"
    <!-- change this to master and slave -->
  </Resource>

  <Resource name="jdbc/WriteDB" auth="Container" type="javax.sql.DataSource"
    maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
    password="mypassword" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://52.14.183.147:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
    <!-- change this to only the master-->
  </Resource>
</Context>
```

- File name, line numbers as in Github

AddInfo - 63

AddMovie - 73

AddStar - 56

CreditCard - 51

Login - 76

MovieAutoComplete - 95

MovieListServlet - 84

MovieListNoPS - 83

MovieListNoPool - 87

SalesInsert - 64

Stars - 51

- Snapshots showing use in your code

```
context.xml - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

context.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <Context>
4 <!-- Defines a Data Source Connecting to localhost moviedb-->
5 <Resource name="jdbc/moviedb"
6   auth="Container"
7   driverClassName="com.mysql.jdbc.Driver"
8   type="javax.sql.DataSource"
9   username="mytestuser"
10  password="mypassword"
11  url="jdbc:mysql://localhost:3306/moviedb"/>
12
13 <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
14   maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
15   password="mypassword" driverClassName="com.mysql.jdbc.Driver"
16   url="jdbc:mysql:replication://52.14.183.147:3306,18.188.5.190:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"
17   ^ change this to master and slave
18   url="jdbc:mysql:replication://172.31.34.82:3306,172.31.34.23:3306/moviedb"
19   link href="https://stackoverflow.com/questions/19432724/application-level-use-of-mysql-single-master-single-replica"
20   -->
21 <Resource name="jdbc/WriteDB" auth="Container" type="javax.sql.DataSource"
22   maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
23   password="mypassword" driverClassName="com.mysql.jdbc.Driver"
24   url="jdbc:mysql://52.14.183.147:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
25   ^ change this to only the master-->
26 </Context>
```

```
MovieListServlet.java - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

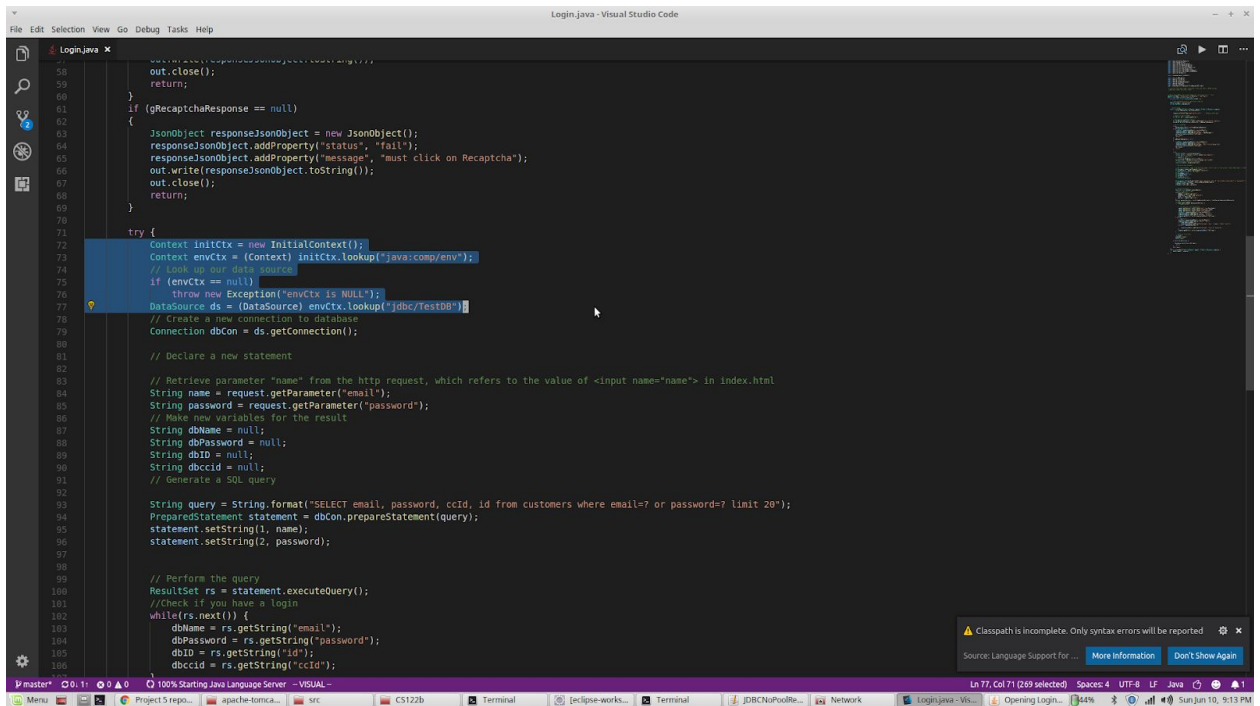
MovieListServlet.java x
69 if (title != null && !search.isEmpty())
70     title = "%" + title;
71     title = (title != null) ? title : "";
72
73 System.out.println("TitleTemp = " + title);
74 year = (year != null) ? year : "";
75 director = (director != null) ? director : "";
76 star_name = (star_name != null) ? star_name : "";
77
78 System.out.println("going to movieServlet");
79
80 // Output stream to STDOUT
81 PrintWriter out = response.getWriter();
82
83 try {
84     Context initCtx = new InitialContext();
85     Context envCtx = (Context) initCtx.lookup("java:comp/env");
86     // look up our data source
87     if (envCtx == null)
88         throw new Exception("envCtx is NULL");
89     DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
90     // get a connection from data source
91     Connection dbcon = ds.getConnection();
92     String query = "";
93     if (genre != null) {
94         System.out.println("going into Genre!!!!");
95         query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating\n" +
96             "FROM movies m, stars_in_movies sim, stars s, genres g, genres_in_movies gim, ratings r\n" +
97             "WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = gim.movieid AND m.id = r.movieid\n" +
98             "AND g.name LIKE ?\n" +
99             "GROUP BY m.id, m.title, m.year, m.director, r.rating\n" +
100             "LIMIT 1000");
101     } else {
102         query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating\n" +
103             "FROM movies m, stars_in_movies sim, stars s, genres g, genres_in_movies gim, ratings r\n" +
104             "WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = gim.movieid AND m.id = r.movieid\n" +
105             "AND (match (m.title) against ( ? in boolean mode) OR ( ? LIKE m.title) OR ed(m.title, ?) <= 3) AND m.director LIKE ? AND m.year LIKE ?\n" +
106             "GROUP BY m.id, m.title, m.year, m.director, r.rating\n" +
107             "order by (ed(?, m.title)) asc\n" +
108             "LIMIT 1000");
109     }
110     // Declare our statement
111     PreparedStatement statement = dbcon.prepareStatement(query);
112
113     // Set the parameter represented by "?" in the query to the id we get from url,
114     // num 1 indicates the first "?" in the query
115     if (genre != null) {
116         genre = "%" + genre + "%";
117     }
118 }
```

```
MovieListServletNoPool.java - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

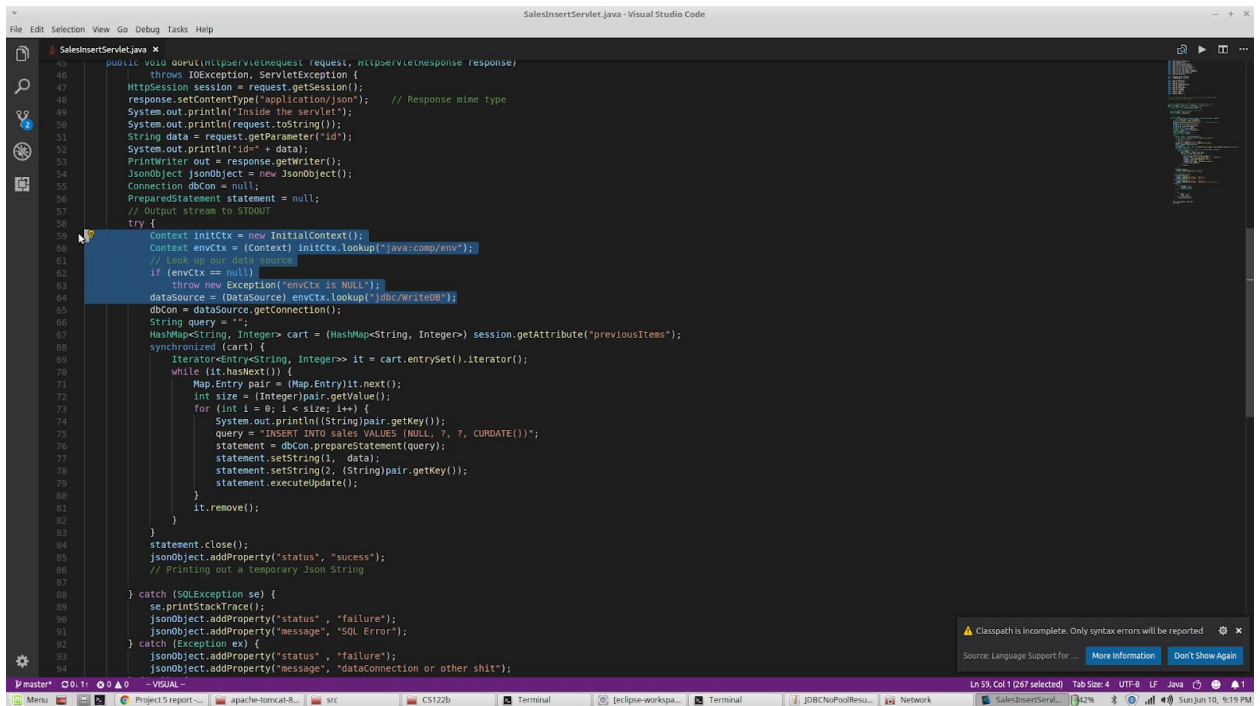
MovieListServletNoPool.java
57 // Retrieve parameter id from url request.
58 String title = request.getParameter("movie_title");
59 String year = request.getParameter("movie_year");
60 String director = request.getParameter("director");
61 String star_name = request.getParameter("star_name");
62 String genre = request.getParameter("genre");
63 String orderby = request.getParameter("order by");
64 String frmSearch = request.getParameter("s");
65
66 String terms = returnQueryString(title);
67 if (title != null && frmSearch != null)
68     title = "%" + title;
69     title = (title != null) ? title : "";
70
71 System.out.println("TitleTemp = " + title);
72 year = (year != null) ? year : "";
73 director = (director != null) ? director : "";
74 star_name = (star_name != null) ? star_name : "";
75
76 System.out.println("going to movieServlet");
77
78 // Output stream to STDOUT
79 PrintWriter out = response.getWriter();
80
81 try {
82     Context initCtx = new InitialContext();
83     Context envCtx = (Context) initCtx.lookup("java:comp/env");
84     // Look up our data source
85     if (envCtx == null)
86         throw new Exception("envctx is NULL");
87     DataSource ds = (DataSource) envCtx.lookup("jdbc/movieadb");
88     // Get a connection from dataSource
89     Connection dbcon = ds.getConnection();
90     String query = "";
91     if (genre != null) {
92         System.out.println("going into Genre!!!!!!");
93         query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating
94             * FROM movies m, stars in movies sim, stars s, genres g, genres_in_movies gim, ratings \r\n" +
95             * WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = r.movieid \r\n" +
96             * AND g.name LIKE ? \r\n" +
97             * GROUP BY m.id, m.title, m.year, m.director, r.rating \r\n" +
98             * LIMIT 1000");
99     }
100 } else {
101     query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating
102         * FROM movies m, stars in movies sim, stars s, genres g, genres_in_movies gim, ratings \r\n" +
103         * WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = r.movieid \r\n" +
104         * AND (match (m.title) against ( ? in boolean mode) OR ( ? LIKE m.title) OR ed(m.title, ?) <= 3) AND m.director LIKE ? AND m.year LIKE ?
105         * AND s.name LIKE ? \r\n" +
106         * ORDER BY ed( ? , m.title) asc \r\n" +
107         * LIMIT 1000");
108 }
109
110 // Declare our statement
111 Statement statement = dbcon.createStatement();
112
113 // Set the parameter represented by "?" in the query to the id we get from url.
114 // num 3 indicates the first "?" in the query
115 statement.setString(1, terms);
116 statement.setString(2, title);
117 statement.setString(3, title);
118 statement.setString(4, director);
119 statement.setString(5, year);
120 statement.setString(6, star_name);
121
122 ResultSet rs = statement.executeQuery(query);
123 while (rs.next()) {
124     System.out.println("Movie: " + rs.getString(1) + " Title: " + rs.getString(2) + " Year: " + rs.getString(3) + " Director: " + rs.getString(4) + " Genres: " + rs.getString(5) + " StarNameID: " + rs.getString(6) + " Rating: " + rs.getString(7));
125 }
126 out.println("Movie List Servlet Results");
127 out.close();
128 } catch (Exception e) {
129     e.printStackTrace();
130 }
131 }
```

```
MovieListServletNoPS.java - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

MovieListServletNoPS.java
64 String frmSearch = request.getParameter("s");
65 String title_cpy = title;
66 String terms = returnQueryString(title);
67 if (title != null && frmSearch != null)
68     title = "%" + title;
69     title = (title != null) ? title : "";
70
71 System.out.println("TitleTemp = " + title);
72 year = (year != null) ? year : "";
73 director = (director != null) ? director : "";
74 star_name = (star_name != null) ? star_name : "";
75
76 System.out.println("going to movieServlet");
77
78 // Output stream to STDOUT
79 PrintWriter out = response.getWriter();
80
81 try {
82     Context initCtx = new InitialContext();
83     Context envCtx = (Context) initCtx.lookup("java:comp/env");
84     // Look up our data source
85     if (envCtx == null)
86         throw new Exception("envctx is NULL");
87     DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
88     // Get a connection from dataSource
89     Connection dbcon = ds.getConnection();
90     String query = "";
91     title = title + "%";
92     director = "%" + director + "%";
93     year = "%" + year + "%";
94     star_name = "%" + star_name + "%";
95     query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating
96         * FROM movies m, stars in movies sim, stars s, genres g, genres_in_movies gim, ratings \r\n" +
97         * WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = r.movieid \r\n" +
98         * AND (match (m.title) against ( ?+ terms +? in boolean mode) OR ( ?+ title +? LIKE m.title) OR ed(m.title, ?+ title_cpy +?) <= 3) AND m.director LIKE ?+ director +?
99         * AND s.name LIKE ?+ star_name +? \r\n" +
100         * GROUP BY m.id, m.title, m.year, m.director, r.rating \r\n" +
101         * order by ed( ?+ title +? , m.title) asc \r\n" +
102         * LIMIT 1000");
103
104     // Declare our statement
105     Statement statement = dbcon.createStatement();
106
107     // Set the parameter represented by "?" in the query to the id we get from url.
108     // num 3 indicates the first "?" in the query
109     statement.setString(1, terms);
110     statement.setString(2, title);
111     statement.setString(3, title);
112     statement.setString(4, director);
113     statement.setString(5, year);
114     statement.setString(6, star_name);
115
116     ResultSet rs = statement.executeQuery(query);
117     while (rs.next()) {
118         System.out.println("Movie: " + rs.getString(1) + " Title: " + rs.getString(2) + " Year: " + rs.getString(3) + " Director: " + rs.getString(4) + " Genres: " + rs.getString(5) + " StarNameID: " + rs.getString(6) + " Rating: " + rs.getString(7));
119     }
120     out.println("Movie List Servlet Results");
121     out.close();
122 } catch (Exception e) {
123     e.printStackTrace();
124 }
125 }
```



```
58 out.close();
59 return;
60 }
61 if (gRecaptchaResponse == null)
62 {
63     JSONObject responseObject = new JSONObject();
64     responseObject.addProperty("status", "fail");
65     responseObject.addProperty("message", "must click on Recaptcha");
66     out.write(responseObject.toString());
67     out.close();
68     return;
69 }
70
71 try {
72     Context initCtx = new InitialContext();
73     Context envCtx = (Context) initCtx.lookup("java:comp/env");
74     // Look up our data source
75     if (envCtx == null)
76         throw new Exception("envCtx is NULL");
77     DataSource ds = (DataSource) envCtx.lookup("jdbc/testDB");
78     // Create a new connection to database
79     Connection dbCon = ds.getConnection();
80
81     // Declare a new statement
82
83     // Retrieve parameter "name" from the http request, which refers to the value of <input name="name"> in index.html
84     String name = request.getParameter("email");
85     String password = request.getParameter("password");
86     // Make new variables for the result
87     String dbName = null;
88     String dbPassword = null;
89     String dbID = null;
90     String dbccid = null;
91     // Generate a SQL query
92
93     String query = String.format("SELECT email, password, ccid, id from customers where email=? or password=? limit 20");
94     PreparedStatement statement = dbCon.prepareStatement(query);
95     statement.setString(1, name);
96     statement.setString(2, password);
97
98
99
100 // Perform the query
101 ResultSet rs = statement.executeQuery();
102 //Check if you have a login
103 while(rs.next()) {
104     dbName = rs.getString("email");
105     dbPassword = rs.getString("password");
106     dbID = rs.getString("id");
107     dbccid = rs.getString("ccid");
108 }
```



```
46 public void doPut(HttpServletRequest request, HttpServletResponse response)
47     throws IOException, ServletException {
48     HttpSession session = request.getSession();
49     response.setContentType("application/json"); // Response mime type
50     System.out.println("Inside the servlet");
51     System.out.println(request.toString());
52     String data = request.getParameter("id");
53     System.out.println("id = " + data);
54     PrintWriter out = response.getWriter();
55     JSONObject jsonObject = new JSONObject();
56     Connection dbCon = null;
57     PreparedStatement statement = null;
58     // Output stream to STDOUT
59     try {
60         Context initCtx = new InitialContext();
61         Context envCtx = (Context) initCtx.lookup("java:comp/env");
62         // Look up our data source
63         if (envCtx == null)
64             throw new Exception("envCtx is NULL");
65         DataSource ds = (DataSource) envCtx.lookup("jdbc/WriteDB");
66         dbCon = ds.getConnection();
67         String query = "";
68         HashMap<String, Integer> cart = (HashMap<String, Integer>) session.getAttribute("previousItems");
69         synchronized (cart) {
70             Iterator<Entry<String, Integer>> it = cart.entrySet().iterator();
71             while (it.hasNext()) {
72                 Map.Entry pair = (Map.Entry) it.next();
73                 int size = (Integer) pair.getValue();
74                 for (int i = 0; i < size; i++) {
75                     System.out.println(String.format("%s", pair.getKey()));
76                     query = "INSERT INTO sales VALUES (NULL, ?, ?, CURDATE())";
77                     statement = dbCon.prepareStatement(query);
78                     statement.setString(1, data);
79                     statement.setString(2, (String) pair.getKey());
80                     statement.executeUpdate();
81                 }
82                 it.remove();
83             }
84             statement.close();
85             jsonObject.addProperty("status", "success");
86             // Printing out a temporary Json String
87
88         } catch (SQLException se) {
89             se.printStackTrace();
90             jsonObject.addProperty("status", "failure");
91             jsonObject.addProperty("message", "SQL Error");
92         } catch (Exception ex) {
93             jsonObject.addProperty("status", "failure");
94             jsonObject.addProperty("message", "dataConnection or other shit");
95 }
```

- How did you use Prepared Statements?

Prepared Statements are used by putting ? in the MYSQL statement. Depending on how many ? there are in the mySQL statement, you have to fill in the ? by putting in putString in it or the associated correct value.

- File name, line numbers as in Github

[cs122b-spring18-team-63/project1/tomcatServlet/src/](https://github.com/cs122b-spring18-team-63/project1/tomcatServlet/src/)

AddInfo - 66

AddMovie - 76

AddStar - 58

Credit Card - 71

Login - 93

MovieAutoComplete - 114

MovieListServlet - 103

MovieListNoPS - 95

MovieListNoPool - 101

SalesInsert - 75

Stars - 58

- Snapshots showing use in your code

```
41 // Extract request parameter id from request object
42 String id = request.getParameter("movie_id");
43 String star = request.getParameter("movie_star");
44 String genre = request.getParameter("movie_genre");
45 //String rating = request.getParameter("rating");
46
47 if (id == "")
48     id = null;
49 if (star == "")
50     star = null;
51 if (genre == "")
52     genre = null;
53 //rating = Objects.toString(rating, "");
54
55 // Output stream to STDOUT
56 PrintWriter out = response.getWriter();
57
58 try {
59     Context initCtx = new InitialContext();
60     Context envCtx = (Context) initCtx.lookup("java:comp/env");
61     // Look up our data source
62     if (envCtx == null)
63         throw new Exception("envCtx is NULL");
64     DataSource envCtxLookup = (DataSource) envCtx.lookup("jdbc/WriteDB");
65     Connection dbCon = envCtxLookup.getConnection();
66
67     String query = "call add_info(?, ?, ?)";
68     PreparedStatement preparedStatement = dbCon.prepareStatement(query);
69     if (id == null)
70         preparedStatement.setString(1, null);
71     else
72         preparedStatement.setString(1, id);
73     if (genre == null)
74         preparedStatement.setString(2, null);
75     else
76         preparedStatement.setString(2, genre);
77     if (star == null)
78         preparedStatement.setString(3, null);
79     else
80         preparedStatement.setString(3, star);
81     preparedStatement.executeUpdate();
82
83     JSONObject jsonObject = new JSONObject();
84     jsonObject.addProperty("status", "success");
85     jsonObject.addProperty("message", "Successful insertion.");
86     response.getWriter().write(jsonObject.toString());
87     response.setStatus(200);
88
89 } catch (SQLException ex) {
90     // write error message JSON object to output
91 }
```



```

SalesInsertServlet.java - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

SalesInsertServlet.java
56: PreparedStatement statement = null;
57: // Output stream to STDOUT
58: try {
59:     Context initCtx = new InitialContext();
60:     Context envCtx = (Context) initCtx.lookup("java:comp/env");
61:     // Look up our data source
62:     if (envCtx == null)
63:         throw new Exception("envCtx is NULL");
64:     DataSource ds = (DataSource) envCtx.lookup("jdbc/WriteDB");
65:     dbCon = ds.getConnection();
66:     String query = "";
67:     HashMap<String, Integer> cart = (HashMap<String, Integer>) session.getAttribute("previousItems");
68:     synchronized (cart) {
69:         Iterator<Entry<String, Integer>> it = cart.entrySet().iterator();
70:         while (it.hasNext()) {
71:             Map.Entry pair = (Map.Entry) it.next();
72:             int size = (Integer) pair.getValue();
73:             for (int i = 0; i < size; i++) {
74:                 System.out.println(String.format("%s", pair.getKey()));
75:                 query = "INSERT INTO sales VALUES (NULL, ?, ?, CURDATE())";
76:                 statement = dbCon.prepareStatement(query);
77:                 statement.setString(1, data);
78:                 statement.setString(2, (String) pair.getKey());
79:                 statement.executeUpdate();
80:             }
81:             it.remove();
82:         }
83:     }
84:     statement.close();
85:     JSONObject.addProperty("status", "success");
86:     // Printing out a temporary Json String
87:
88: } catch (SQLException se) {
89:     se.printStackTrace();
90:     JSONObject.addProperty("status", "failure");
91:     JSONObject.addProperty("message", "SQL Error");
92: } catch (Exception ex) {
93:     JSONObject.addProperty("status", "failure");
94:     JSONObject.addProperty("message", "dataConnection or other shit");
95: } finally {
96:     try {
97:         if (statement != null)
98:             statement.close();
99:     } catch (SQLException se) {
100:
101:     }
102:     try {
103:         if (dbCon != null)
104:             dbCon.close();
105:     } catch (SQLException se) {
106:
107:     }
108: }

```

```

MovieListServlet.java - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

MovieListServlet.java
85: Context envCtx = (Context) initCtx.lookup("java:comp/env");
86: // Look up our data source
87: if (envCtx == null)
88:     throw new Exception("envCtx is NULL");
89: DataSource ds = (DataSource) envCtx.lookup("jdbc/TestDB");
90: // Get a connection from DataSource
91: Connection dbcon = ds.getConnection();
92: String query = "";
93: if (genre != null) {
94:     System.out.println("going into Genre!!!!");
95:     query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating, r.starNameID, r.rating FROM movies m, stars_in_movies sim, stars s, genres g, genres_in_movies gim, ratings r\n" +
96:         "WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = gim.movieid AND m.id = r.movieid\n" +
97:         "AND g.name LIKE ?\n" +
98:         "GROUP BY m.id, m.title, m.year, m.director, r.rating\n" +
99:         "LIMIT 1000");
100: } else {
101:     query = ("SELECT m.id, m.title, m.year, m.director, GROUP_CONCAT(DISTINCT g.name separator ',') AS genres, GROUP_CONCAT(DISTINCT s.name, ',', s.id separator ',') AS starNameID, r.rating, r.starNameID, r.rating FROM movies m, stars_in_movies sim, stars s, genres g, genres_in_movies gim, ratings r\n" +
102:         "WHERE m.id = sim.movieid AND s.id = sim.starid AND g.id = gim.genreid AND m.id = gim.movieid AND m.id = r.movieid\n" +
103:         "AND (match (m.title) against ( ? in boolean mode) OR ( ? LIKE m.title) OR ed(m.title, ?) <= 3) AND m.director LIKE ? AND m.year LIKE ?\n" +
104:         "GROUP BY m.id, m.title, m.year, m.director, r.rating\n" +
105:         "order by (ed( ? , m.title) asc)\n" +
106:         "LIMIT 1000");
107: }
108: // Declare our statement
109: PreparedStatement statement = dbcon.prepareStatement(query);
110: // Set the parameter represented by "?" in the query to the id we get from url
111: // num indicates the first "?" in the query
112: if (genre != null) {
113:     genre = "%" + genre + "%";
114:     statement.setString(1, genre);
115: } else {
116:     title = title + "%";
117:     director = "%" + director + "%";
118:     year = "%" + year + "%";
119:     star_name = "%" + star_name + "%";
120:     statement.setString(1, title);
121:     statement.setString(2, director);
122:     statement.setString(3, year);
123:     statement.setString(4, star_name);
124:     statement.setString(5, title);
125:     statement.setString(6, star_name);
126:     statement.setString(7, title);
127: }
128: // Set the parameter represented by "?" in the query to the id we get from url

```

Task 2

- Address of AWS and Google instances

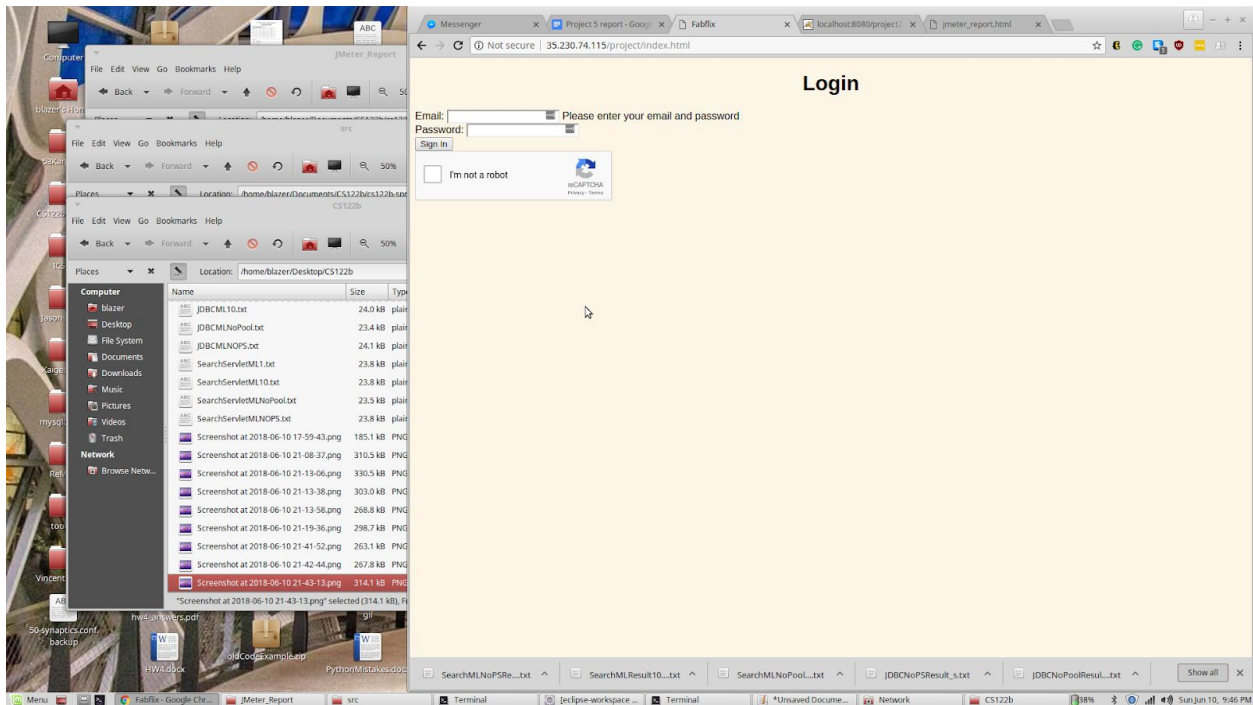
Google	Original(Load)	Master	Slave
--------	----------------	--------	-------

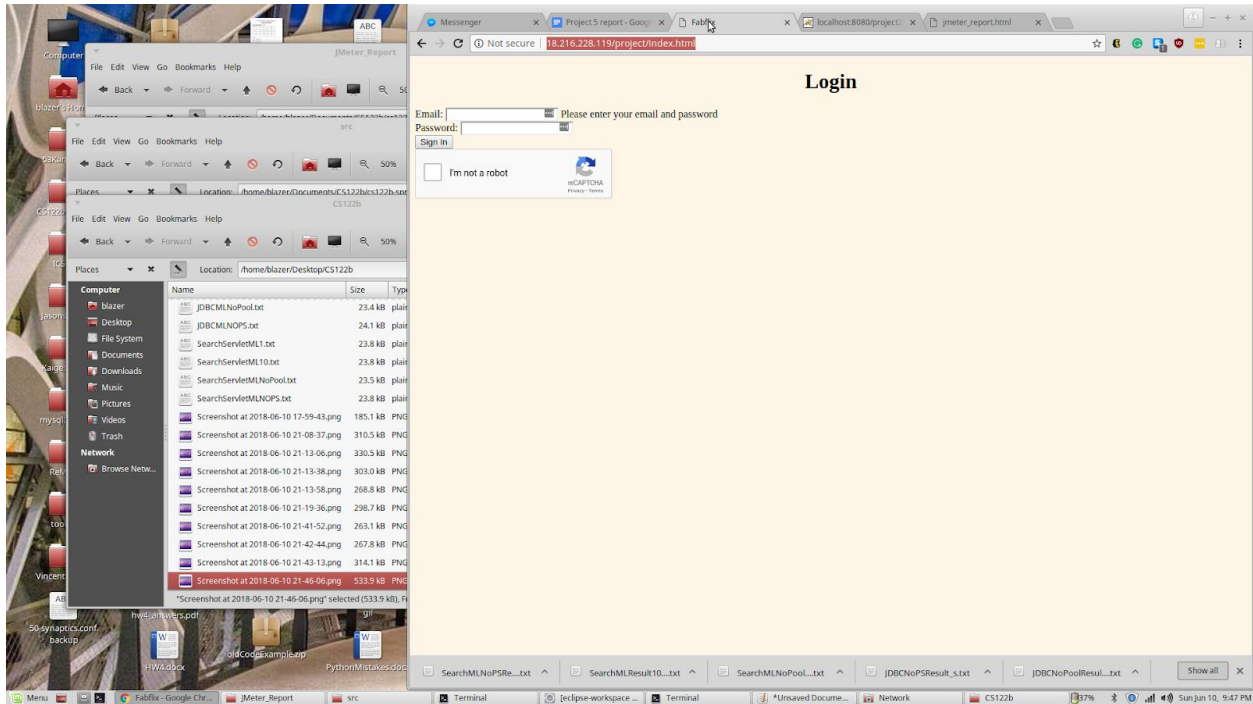
35.230.74.115	18.216.228.119	52.14.183.147	18.188.5.190
---------------	----------------	---------------	--------------

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

The Fablix instance is open on both Google 80, and AWS is working on 8080 for the slave and master.

Google instance:





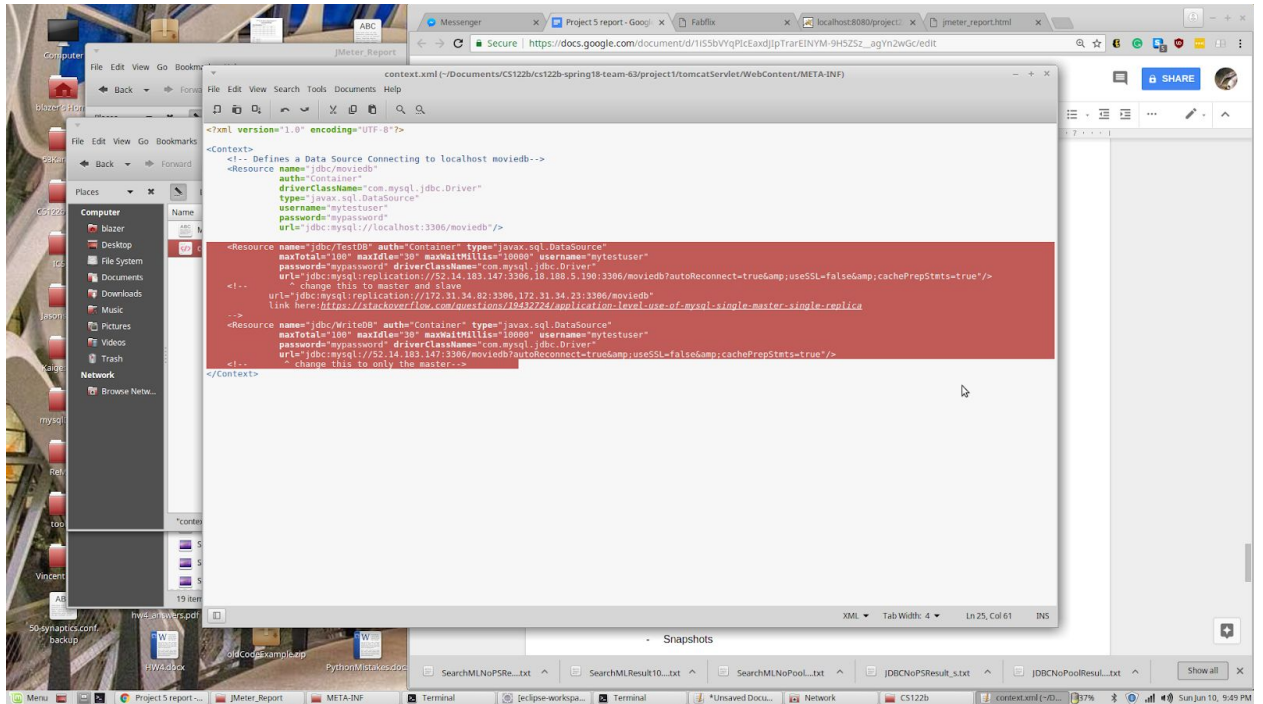
AWS Load Balancer

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

Each tomcat instance can talk to the master mysql instance and also the slave mysql instance. In your context.xml, you have to make sure the read settings are replicated for the read.

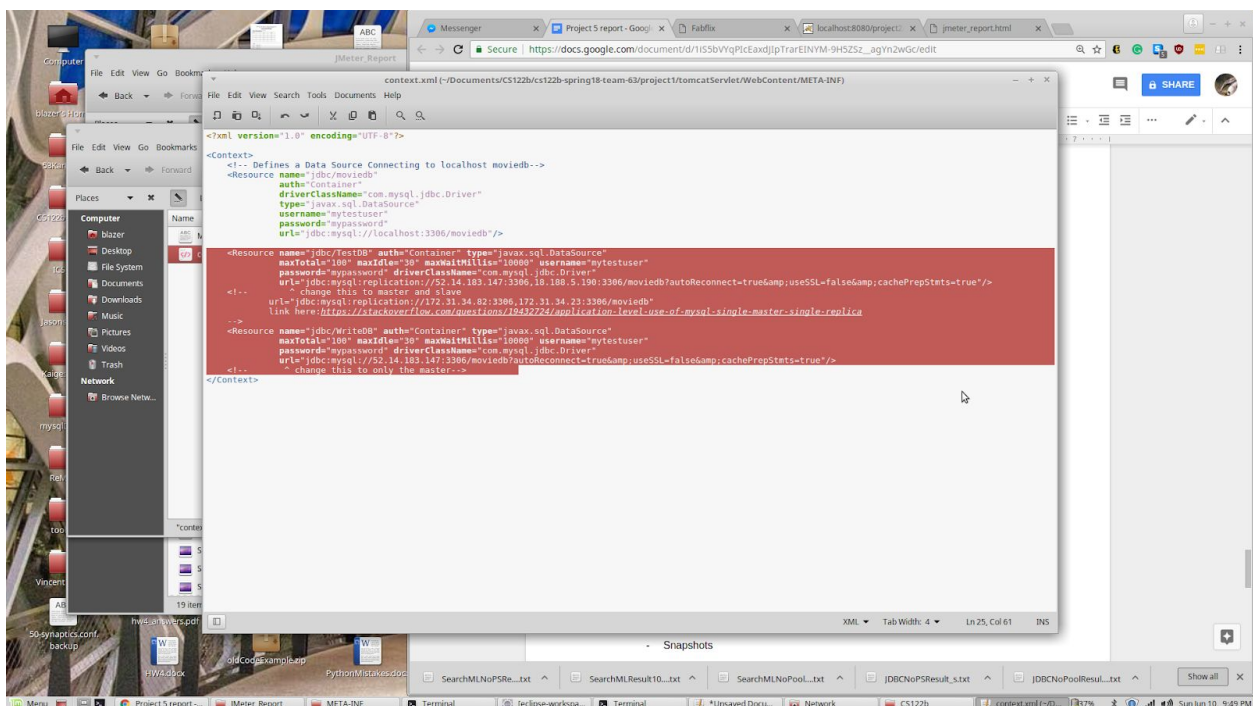
- File name, line numbers as in Github
[project1/tomcatServlet/WebContent/META-INF/context.xml](#) - 13 -24

- Snapshots



- How read/write requests were routed?
Read uses mysql replication while the write is only routed to the master.

- File name, line numbers as in Github
[project1/tomcatServlet/WebContent/META-INF/context.xml](#) - 13-24
- Snapshots



Task 3

- Have you uploaded the log files to Github? Where is it located?
The logfile are in the root directory of the Github file. It is called 80 Logs and 8080 Logs
- Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
The Jmeter Report is in the folder JMeter_report.
- Have you uploaded the script to Github? Where is it located?
The Jmeter Scripts are on the folder Jmeter Scripts
- Have you uploaded the WAR file and README to Github? Where is it located?
The Readme are in the in root folder. The war file is in
/home/blazer/Documents/CS122b/cs122b-spring18-team-63/project1/tomcatServlet/target