

Task 1

- How did you use connection pooling?

Line 3-6 in the picture below is about connection pooling in **context.xml**

[cs122b-winter18-team-17 / project2-login-example-master / WebContent / META-INF / context.xml](https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/META-INF/context.xml)
<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/META-INF/context.xml>

```
1 <Context>
2
3     <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
4         maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
5         password="wei123456" driverClassName="com.mysql.jdbc.Driver"
6         url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&useSSL=false" />
7
8     <Resource name="jdbc/insert" auth="Container" type="javax.sql.DataSource"
9         maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
10        password="wei123456" driverClassName="com.mysql.jdbc.Driver"
11        url="jdbc:mysql://172.31.30.134:3306/moviedb?autoReconnect=true&useSSL=false" />
12
13 </Context>
14
```

Create the first resource, which use the localhost on the instance. Therefore, read operation will go into either master instance or slave instance.

Line 13-27 in the picture below **web.xml**

[cs122b-winter18-team-17 / project2-login-example-master / WebContent / WEB-INF / web.xml](https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/WEB-INF/web.xml)
<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/WEB-INF/web.xml>

```

13  <resource-ref>
14    <description>
15      Resource reference to a factory for java.sql.Connection
16      instances that may be used for talking to a particular
17      database that
18      is configured in the server.xml file.
19    </description>
20    <res-ref-name>
21      jdbc/moviedb
22    </res-ref-name>
23    <res-type>
24      javax.sql.DataSource
25    </res-type>
26    <res-auth>Container</res-auth>
27  </resource-ref>
28
29  <resource-ref>
30    <description>
31      Resource reference to a factory for java.sql.Connection
32      instances that may be used for talking to a particular
33      database that
34      is configured in the server.xml file.
35    </description>
36    <res-ref-name>
37      jdbc/insert
38    </res-ref-name>
39    <res-type>
40      javax.sql.DataSource
41    </res-type>
42    <res-auth>Container</res-auth>
43  </resource-ref>
44

```

cs122b-winter18-team-17 / project2-login-example-master / src /

In the following file, change how to connect the database, using connection pooling now:

In **AdvancedSearch.java**: line 64 - line 85

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AdvancedSearch.java>

In **AutoComplete.java**: line 63 - line 84

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AutoComplete.java>

In **BrowseByGenre.java**: line 64 - line 84

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/BrowseByGenre.java>

In **BrowseByTitle.java**: line 56 - line 77

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/BrowseByTitle.java>

In **CheckOut.java**: line 141 - line 162

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/CheckOut.java>

In **EmployeeLogin.java**: line 83 - line 104

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/EmployeeLogin.java>

In **Login.java**: line 84 - line 105

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Login.java>

In **Metadata.java**: line 58 - line 79

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Metadata.java>

In **Search.java**: line 51 - line 74

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Search.java>

In **SingleMovie.java**: line 64 - line 84

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/SingleMovie.java>

In **SingleStar.java**: line 62 - line 83

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/SingleStar.java>

- Snapshots

```
9
0      long startTime1 = System.nanoTime();
1      // the following few lines are for connection pooling
2      // Obtain our environment naming context
3
4      Context initCtx = new InitialContext();
5      if (initCtx == null)
6          out.println("initCtx is NULL");
7
8      Context envCtx = (Context) initCtx.lookup("java:comp/env");
9      if (envCtx == null)
0          out.println("envCtx is NULL");
1
2      // Look up our data source
3      DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
4
5      // the following commented lines are direct connections without pooling
6      //Class.forName("org.gjt.mm.mysql.Driver");
7      //Class.forName("com.mysql.jdbc.Driver").newInstance();
8      //Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
9
0      if (ds == null)
1          out.println("ds is null.");
2
3      Connection dbcon = ds.getConnection();
4      if (dbcon == null)
5          out.println("dbcon is null.");
6      /
7      /      //Class.forName("org.gjt.mm.mysql.Driver");
8      /      Class.forName("com.mysql.jdbc.Driver").newInstance();
9      /
0      /      Connection dbcon = DriverManager.getConnection(loginUrl, loginUser, loginPasswd);
1      // Declare our statement
2
```

- How did you use Prepared Statements?

cs122b-winter18-team-17 / project2-login-example-master / src /

All java file under project2-login-example-master/src

In **Search.java** (where I use fulltext search): line 99-106, using "?" in query

In line 133, create PreparedStatement

In line 135 - 139 put value into PreparedStatement

In line 147, run PreparedStatement

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/Search.java>

```
98
99
100     for (int i=0; i<splited.length; i++)
101     {
102         total_input = total_input + "MATCH (title) AGAINST (? IN BOOLEAN MODE) ";
103         if (i != splited.length-1)
104         {
105             total_input+="AND ";
106         }
107     }
108
109     System.out.println(total_input);
110
111
112
113
114     // String query = ""
115     // + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ' ) AS stars, GROUP_CONCAT(DISTINCT genres.name ORDER BY genres.name SEPARATOR ', ' ) AS genres"
116     // + "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
117     // + "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND ratings.movieId=movies.id\n" +
118     // + "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
119     // + "ORDER BY ratings.rating DESC;" ;
120
121     String query = ""
122     + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ' ) AS stars, GROUP_CONCAT(DISTINCT genres.name ORDER BY genres.name SEPARATOR ', ' ) AS genres"
123     + "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
124     + "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND ratings.movieId=movies.id\n" +
125     + "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
126     + "ORDER BY ratings.rating DESC;" ;
127
128     System.out.println("query111111 = "+query);
129
130
131     //movies.title LIKE 'XhomeX'
132     //movies.title LIKE '%'+input+'%\n" +
133     PreparedStatement pstmt = dbcon.prepareStatement( query );
134     System.out.println("after preparement statement = "+query);
135     for (int i=1; i<=splited.length; i++)
136     {
137         pstmt.setString( i,splited[i-1]+"*");
138     }
139
140     System.out.println("query222222 = "+query);
141
142
143
144     long startTime2 = System.nanoTime();
145
146     // Perform the query
147     ResultSet rs = pstmt.executeQuery();
148
149     long endTime2 = System.nanoTime();
150
151
152     JSONArray jsonArray = new JSONArray();
153     while (rs.next())
154     {
155         String movie_id = rs.getString(1);
156         String movie_title = rs.getString(2);
157         int movie_year = rs.getInt(3);
158         String movie_director = rs.getString(4);
159         String star_name = rs.getString(5);
160         String genre_type = rs.getString(6);
```

In **AdvancedSearch.java**: line 136-135, using “?” in query

In line 154, create PreparedStatement

In line 157 - 178 put value into PreparedStatement

In line 185, run PreparedStatement

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AdvancedSearch.java>

```
115     String total_input = "";
116     if (title.length() > 0)
117     {
118         total_input += "AND movies.title LIKE ? ";
119         hm.put("title", title);
120     }
121     if (year.length() > 0)
122     {
123         total_input += "AND movies.year=? ";
124         hm.put("year", year);
125     }
126     if (director.length() > 0)
127     {
128         total_input += "AND movies.director LIKE ? ";
129         hm.put("director", director);
130     }
131     if (input_star_name.length() > 0)
132     {
133         total_input += "AND stars.name LIKE ? ";
134         hm.put("input_star_name", input_star_name);
135     }
136
137     System.out.println(" total input : " + total_input);
138
139
140
141     System.out.println("before query ");
142     //genre = "drama";
143     String query = "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DISTINCT genres.name ORDER BY genres.name SEPARATOR ', ') AS genres, ratings.rating\n" +
144         "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
145         "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id\n" +
146         "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
147         "ORDER BY ratings.rating DESC" ; ////
148     //LIMIT 100";
149
150     System.out.println("parameterrrrrrrrr " + title);
151     System.out.println("go after query " + query);
152     //total_input[1]
153     // Perform the query
154     PreparedStatement pstmt = dbcon.prepareStatement( query );
155     //pstmt.setString( 1, MovieList.get(i).getTitle());
156
157     int index=1;
158
159     if(hm.get("title")!=null)
160     {
161         pstmt.setString( index, "K"+title+"K");
162         index++;
163     }
164     if(hm.get("year")!=null)
165     {
166         pstmt.setString( index, year);
167         index++;
168     }
169     if(hm.get("director")!=null)
170     {
171         pstmt.setString( index, "K"+director+"K");
172         index++;
173     }
174     if(hm.get("input_star_name")!=null)
175     {
176         pstmt.setString( index, "K"+input_star_name+"K");
177         index++;
178     }
179
180
181
182
183
184     long startTime2 = System.nanoTime();
185     ResultSet rs = pstmt.executeQuery();
186     long endTime2 = System.nanoTime();
187
188     JSONArray jsonArray = new JSONArray();
189
```

In **AutoComplete.java**:

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/AutoComplete.java>

For **search Movie title part**:

line 118-125, using “?” in query

In line 149, create PreparedStatement

In line 151 - 155 put value into PreparedStatement

In line 162, run PreparedStatement

```
115     String total_input="";
116
117
118     for (int i=0; i<splited.length; i++)
119     {
120         total_input = total_input + "MATCH (title) AGAINST ( ? IN BOOLEAN MODE) ";
121         if (i != splited.length-1)
122         {
123             total_input+="AND ";
124         }
125     }
126
127
128     System.out.println(total_input);
129
130
131
132     String query = ""
133         + "SELECT movies.id, movies.title, movies.year, movies.director, GROUP_CONCAT(DISTINCT stars.name ORDER BY stars.name SEPARATOR ', ') AS stars, GROUP_CONCAT(DIS"
134         + "FROM movies, genres, stars, stars_in_movies, genres_in_movies, ratings\n" +
135         + "WHERE movies.id=stars_in_movies.movieId AND stars_in_movies.starId=stars.id AND movies.id=genres_in_movies.movieId AND genres_in_movies.genreId=genres.id AND r"
136         + "GROUP BY movies.id, movies.title, movies.year, movies.director, ratings.rating\n" +
137         + "ORDER BY ratings.rating DESC\n" +
138         + "LIMIT 5;";
139
140
141
142     System.out.println("query = "+query);
143
144     //movies.title LIKE '%home%'
145     //movies.title LIKE '%'+input+'%\n" +
146
147     // Perform the query
148
149     PreparedStatement pstmt = dbcon.prepareStatement( query );
150
151     for (int i=1; i<=splited.length; i++)
152     {
153         pstmt.setString( i,splited[i-1]+"*" );
154     }
155
156
157
158
159
160     // Perform the query
161     long startTime2 = System.nanoTime();
162     ResultSet rs = pstmt.executeQuery();
163     long endTime2 = System.nanoTime();
164
```

For **search Star name part**:


line 205-209, using “?” in query

In line 215, create PreparedStatement

In line 217 - 221 put value into PreparedStatement

In line 256, run PreparedStatement

```

204
205     String query2 = ""
206         + "SELECT name\n" +
207         "FROM stars\n" +
208         "WHERE " + total_input2+"\n" +
209         "LIMIT 5;";
210
211
212
213     System.out.println("query2 = "+query2);
214
215     PreparedStatement pstmt2 = dbcon.prepareStatement( query2 );
216
217     for (int i=1; i<=splited.length; i++)
218     {
219         pstmt2.setString( i,splited[i-1]+"*" );
220
221         
222
223         // Perform the query
224         long startTime3 = System.nanoTime();
225         rs = pstmt2.executeQuery();
226         long endTime3 = System.nanoTime();
227
228
229

```

Task 2

- Address of AWS and Google instances

AWS Address: 18.219.36.92 <http://18.219.36.92/project2-login-example/>

Google Address: on google spread sheet

When you go to website, you need to enter like: **xx.xxx.xx.xx/project2-login-example/**
 There is a slash at the end, and it should be there in order to work correctly (I don't know why :(

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

I comment out the login filter portion, which makes me easier to go to my website every time, but you can check my sticky session using shopping cart.

- How connection pooling works with two backend SQL?

I create second resource in context.xml file, in this resource, it connect to master's instance database. Therefore, all the write operation will pass to master's database.

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/META-INF/context.xml>

Line 8-11

```
1 <Context>
2
3     <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
4         maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
5         password="wei123456" driverClassName="com.mysql.jdbc.Driver"
6         url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&useSSL=false" />
7
8     <Resource name="jdbc/insert" auth="Container" type="javax.sql.DataSource"
9         maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="root"
10        password="wei123456" driverClassName="com.mysql.jdbc.Driver"
11        url="jdbc:mysql://172.31.30.134:3306/moviedb?autoReconnect=true&useSSL=false" />
12
13 </Context>
14
```

Also map the resource in web.xml below

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/WebContent/WEB-INF/web.xml>

```
13 <resource-ref>
14     <description>
15         Resource reference to a factory for java.sql.Connection
16         instances that may be used for talking to a particular
17         database that
18         is configured in the server.xml file.
19     </description>
20     <res-ref-name>
21         jdbc/moviedb
22     </res-ref-name>
23     <res-type>
24         javax.sql.DataSource
25     </res-type>
26     <res-auth>Container</res-auth>
27 </resource-ref>
28
29 <resource-ref>
30     <description>
31         Resource reference to a factory for java.sql.Connection
32         instances that may be used for talking to a particular
33         database that
34         is configured in the server.xml file.
35     </description>
36     <res-ref-name>
37         jdbc/insert
38     </res-ref-name>
39     <res-type>
40         javax.sql.DataSource
41     </res-type>
42     <res-auth>Container</res-auth>
43 </resource-ref>
44
```



```

Downloads — ubuntu@ip-172-31-42-18: ~ — ssh -i project11.pem ubuntu@ec2-18-219-36-92.us-east-2.compute.amazona...
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED

#<Proxy "balancer://TomcatTest_balancer">
#   BalancerMember "http://172.31.30.134:8080/project2-login-example/"
#   BalancerMember "http://172.31.16.103:8080/project2-login-example/"
#</Proxy>

<Proxy "balancer://Session_balancer">
    BalancerMember "http://172.31.30.134:8080/project2-login-example/" route=1
    BalancerMember "http://172.31.16.103:8080/project2-login-example/" route=2
ProxySet stickysession=ROUTEID
</Proxy>

<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    # For most configuration files from conf-available/, which are
    # enabled or disabled at a global level, it is possible to
    # include a line for only one particular virtual host. For example the
    # following line enables the CGI configuration for this host only
    # after it has been globally disabled with "a2disconf".
    #Include conf-available/serve-cgi-bin.conf

#   ProxyPass /project2-login-example balancer://TomcatTest_balancer
#   ProxyPassReverse /project2-login-example balancer://TomcatTest_balancer

    ProxyPass /project2-login-example balancer://Session_balancer
    ProxyPassReverse /project2-login-example balancer://Session_balancer
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
~

```

Above is my /etc/apache2/sites-enabled/000-default.conf file.

Configure the proxy on instance 1 to handle sessions properly (sticky session).

One is private address of master instance, and another one is private address of slave address

- How read/write requests were routed?

In **InsertMovie.java** and **InsertStar.java**

In **InsertMovie.java**, it is line 65-81

In **InsertStar.java**, it is line 63-80

The difference between this one and the one in task 1 is in line 74, we now look up (jdbc/insert), which is the database that run writing operation.

```

61     try {
62         // the following few lines are for connection pooling
63         // Obtain our environment naming context
64
65         Context initCtx = new InitialContext();
66         if (initCtx == null)
67             out.println("initCtx is NULL");
68
69         Context envCtx = (Context) initCtx.lookup("java:comp/env");
70         if (envCtx == null)
71             out.println("envCtx is NULL");
72
73         // Look up our data source
74         DataSource ds = (DataSource) envCtx.lookup("jdbc/insert");
75
76         if (ds == null)
77             out.println("ds is null.");
78
79         Connection dbcon = ds.getConnection();
80         if (dbcon == null)
81             out.println("dbcon is null.");
82     }

```

Task 3

- Have you uploaded the **log file** to Github? Where is it located?

There are 12 log files for me, 5 for single-instance case, and 2*4-1 for scaled version, because sticky session makes one thread case always go to one instance:

All log file are under this:

UCI-Chenli-teaching / cs122b-winter18-team-17

Single-instance case:

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single1.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single2.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single3.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single4.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single5.txt>

Scaled version case:

Case1: Use HTTP, without using prepared statements, 10 threads in JMeter.

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled1-master-real.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled1-slave-real.txt>

Case2: Use HTTP, without using prepared statements, 10 threads in JMeter.

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled2-master.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled2-slave.txt>

Case3: Use HTTP, 1 thread in JMeter.

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/single3.txt>

Case4: Use HTTP, 10 threads in JMeter.

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled4-master.txt>

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/scaled4-slave.txt>

- Have you uploaded the **HTML** file to Github? Where is it located?

UCI-Chenli-teaching / **cs122b-winter18-team-17**

https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/jmeter_report.html

- Have you uploaded the **script** to Github? Where is it located?

cs122b-winter18-team-17 / project2-login-example-master / **src** /

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example-master/src/CalculateAverage.java>

- Have you uploaded the WAR file and README to Github? Where is it located?

UCI-Chenli-teaching / **cs122b-winter18-team-17**

README is on the home page of our repository (at the bottom):

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17>

The **WAR file** include the script is:

<https://github.com/UCI-Chenli-teaching/cs122b-winter18-team-17/blob/master/project2-login-example.war>