Project 5

Task 1

- **How did you use connection pooling?**

It involves 3 configuration files.

I need to set up a database in the context.xml and specify the total number of connections and the number of idle connections. Then we need to set the factory for connection instances to talk to the particular database, TestDB. Then in the servlet, we use resource TestDB to establish connections to enable connection pooling.

- **File name, line numbers as in Github**

cs122b-spring18-team-104/project2/WebContent/WEB-INF/web.xml // line 13-18

cs122b-spring18-team-104/project2/WebContent/META-INF/context.xml // line 22-25
And in every servlet, I establish connection by connecting to TestDB (in my case, I call it jdbc/Master as reference to the snapshots below).

- **Snapshots showing use in your code**

```
<Resource name="jdbc/Master" auth="Container" type="javax.sql.DataSource"
        maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
        password="mypassword" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://172.31.7.41:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;cachePrepStmts=true"/>
```

```
13    <resource-ref>
14      <description>MySQL DataSource Master</description>
15      <res-ref-name>jdbc/Master</res-ref-name>
16      <res-type>javax.sql.DataSource</res-type>
17      <res-auth>Container</res-auth>
18    </resource-ref>
```

```
public class AddMovieServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
    @Resource(name="jdbc/Master")
    DataSource dataSource;
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
```

- **How did you use Prepared Statements?**

For every Servlet, I use prepared Statements to prepare each query. In this case, only then the statement is compiled just once and can be used repeatedly with different parameters which results in better performance.

After we applying the connection pooling, we need to set the url in the context.xml to url="jdbc:mysql://172.31.7.41:3306/moviedb?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>

Then we can make sure that prepared statements work well with the connection pooing.

- **File name, line numbers as in Github**

In all the servlets which involve query, we used prepared statements.

cs122b-spring18-team-104/project2/src/MovieListServlet.java

```
214                     PreparedStatement preparedStatement = database.prepareStatement(query);
215                     if(isLetter)
216                             preparedStatement.setString(1, queries.get(0) + "%");
217                     else {
218                             for(int x = 0; x < queries.size(); x++)
219                             {
220                                     preparedStatement.setString(x + 1, "%" +queries.get(x)+ "%");
221                             }
222                     }
223                     ResultSet rs = preparedStatement.executeQuery();
224                     System.out.println(preparedStatement);
```

cs122b-spring18-team-104/project2/src/InsertStarServlet.java

```
62                      query = "insert into stars(id, name, birthYear) values(?, ?, ?);";
63                      PreparedStatement preparedStatement = database.prepareStatement(query);
64                      for(int x = 0; x < parameters.size(); x++)
65                      {
66                              preparedStatement.setString(x+1, parameters.get(x));
67                              System.out.println(parameters.get(x));
68                      }
69                      System.out.println(preparedStatement);
70                      preparedStatement.executeUpdate();
71
```

cs122b-spring18-team-104/project2/src/EmployeeSearchServlet.java

```
47                      query += "select movies.id, movies.title, movies.year, movies.director, stars.name as starName, genres.
48                      query += "join stars_in_movies on movies.id = stars_in_movies.movieId ";
49                      query += "join stars on stars.id = stars_in_movies.starId ";
50                      query += "join genres_in_movies on movies.id = genres_in_movies.movieId ";
51                      query += "join genres on genres_in_movies.genreId = genres.id where ";
52                      query += "movies.id =? and genres.name=? and stars.name=?";
53
54                      System.out.println(query);
55
56                      PreparedStatement prepareStatement = database.prepareStatement(query);
57
58                      prepareStatement.setString(1, id);
59                      prepareStatement.setString(2, genreName);
60                      prepareStatement.setString(3, starName);
61
62                      ResultSet rs = prepareStatement.executeQuery();
63
```

cs122b-spring18-team-104/project2/src/LoginServlet.java

```
70                      query += "SELECT * from employees where email=?";
71                      username = employee;
72                      System.out.println(query);
73
74              }
75              //else if(!username.equals("null") && !username.isEmpty())
76              else
77              {
78                      query += "SELECT * from customers where email=?";
79                      System.out.println(query);
80              }
```

Task 2

- **Address of AWS and Google instances**

AWS:
Instance 1: 54.153.105.163
Master: 54.183.219.216
Slave: 54.67.73.197

Google Cloud: 35.196.57.81

- **How does load balancing work?**

I did MySQL master/slave replication. Configure the original instance properly to enable load balancing, connection pooling, sticky sessions.
For master/slave replication, I already checked with pet example provided by professor.
To enable load balancing, Configure the Apache2 web server to use its balancer to the url of fabflix. For the sticky session, we should add a line to the 000-default.conf:
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED

The configuration file is as below.

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

<Proxy "balancer://Session_balancer">
    BalancerMember "http://172.31.7.41:8080/Session" route=1
    BalancerMember "http://172.31.4.105:8080/Session" route=2
    ProxySet stickysession=ROUTEID
</Proxy>
<Proxy "balancer://Fabflix_balancer">
    BalancerMember "http://172.31.7.41:8080/project2-api-example/" route=1
    BalancerMember "http://172.31.4.105:8080/project2-api-example/" route=2
    ProxySet stickysession=ROUTEID
</Proxy>
<Proxy "balancer://TomcatTest_balancer">
    BalancerMember "http://172.31.7.41:8080/TomcatTest/"
    BalancerMember "http://172.31.4.105:8080/TomcatTest/"
</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 /TomcatTest balancer://TomcatTest_balancer
        ProxyPassReverse /TomcatTest balancer://TomcatTest_balancer
        ProxyPass /Session balancer://Session_balancer
        ProxyPassReverse /Session balancer://Session_balancer
        ProxyPass /project2-api-example balancer://Fabflix_balancer
        ProxyPassReverse /project2-api-example balancer://Fabflix_balancer
</VirtualHost>

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

- **How read/write requests were routed?**

When accessing a 'read' page. The read page servlet then connects with 'jdbc/moviedb', which sets url 'localhost'.

As localhost here is original instance public IP, the proxy will then route requests to either of the new instances, master or slave.

When accessing a 'write' page. The write page servlet then connects with 'jdbc/Master', which sets url to my master server public IP. Then the writing operation will be sent to the master replication.

In the servlet which will do writes to the database, I will set the resource to 'jdbc/Master'. Like

cs122b-spring18-team-104/project2/src/AddMovieServlet.java

cs122b-spring18-team-104/project2/src/AddServlet.java

cs122b-spring18-team-104/project2/src/InsertStarServlet.java

I will set the resource to jdbc/Master

```
@WebServlet("/AddMovieServlet")
public class AddMovieServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
    @Resource(name="jdbc/Master")
```

```
@WebServlet("/InsertStarServlet")
public class InsertStarServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        @Resource(name="jdbc/Master")
    private DataSource dataSource;
```

```
*/
@WebServlet("/AddMovieServlet")
public class AddMovieServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
    @Resource(name="jdbc/Master")
    DataSource dataSource;
```

Task 3

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

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

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

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