Task 1

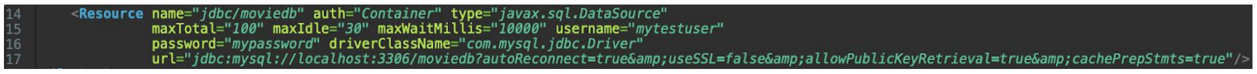
- How did you use connection pooling?
    - I made a copy of the JDBC Driver jar and placed it inside my Tomcat/lib folder. Then I edited my web.xml and added a resource-reference to a factory of java.sql.Connection instances from a server that is configured in my server.xml file.
    - Once connection pooling was configured I implemented it in my code by adding a few lines to obtain the environment naming context by using the Context interface. I used “initCtx.lookup("java:comp/env");” to lookup the environment and “envCtx.lookup("jdbc/moviedb")” to lookup the datasource.

    - File name, line numbers as in Github

    1) cs122b-winter19-team-24/WebContent/META-INF/context.xml (lines 14-17)
    2) cs122b-winter19-team-24/WebContent/WEB-INF/web.xml (lines 22-31)
    3) cs122b-winter19-team-24/src/MovieServlet.java (lines 104-118)
    4) cs122b-winter19-team-24/src/LoginServlet.java (lines 53-67)

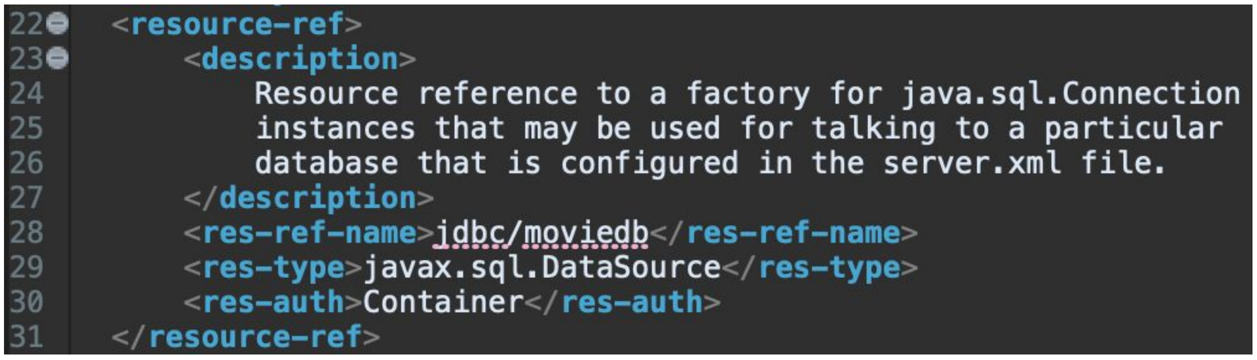
    - Snapshots showing use in your code

#1



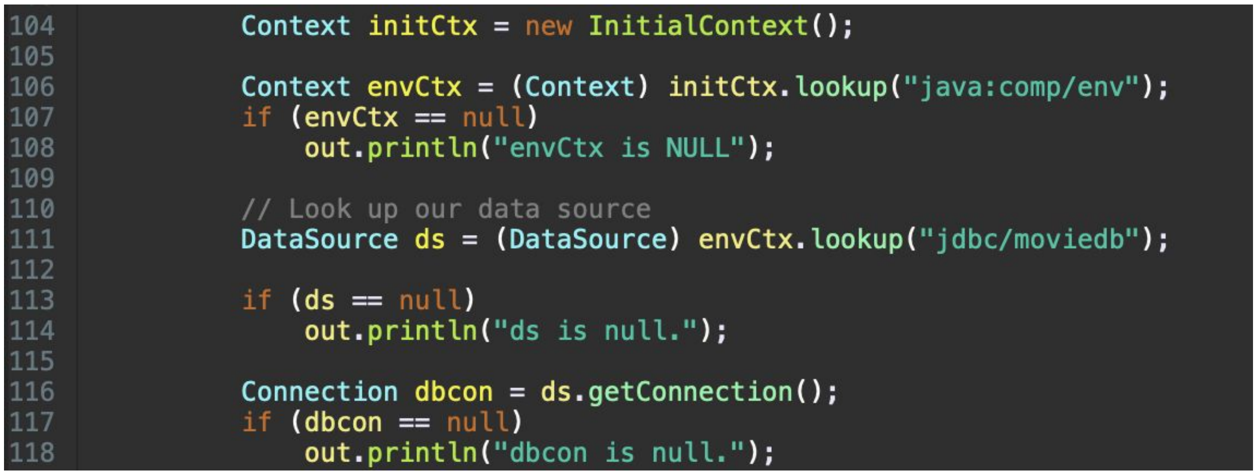
```
14      <Resource name="jdbc/moviedb" auth="Container" type="javax.sql.DataSource"
15                maxTotal="100" maxIdle="30" maxWaitMillis="10000" username="mytestuser"
16                password="mypassword" driverClassName="com.mysql.jdbc.Driver"
17                url="jdbc:mysql://localhost:3306/moviedb?autoReconnect=true&amp;useSSL=false&amp;allowPub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

#2



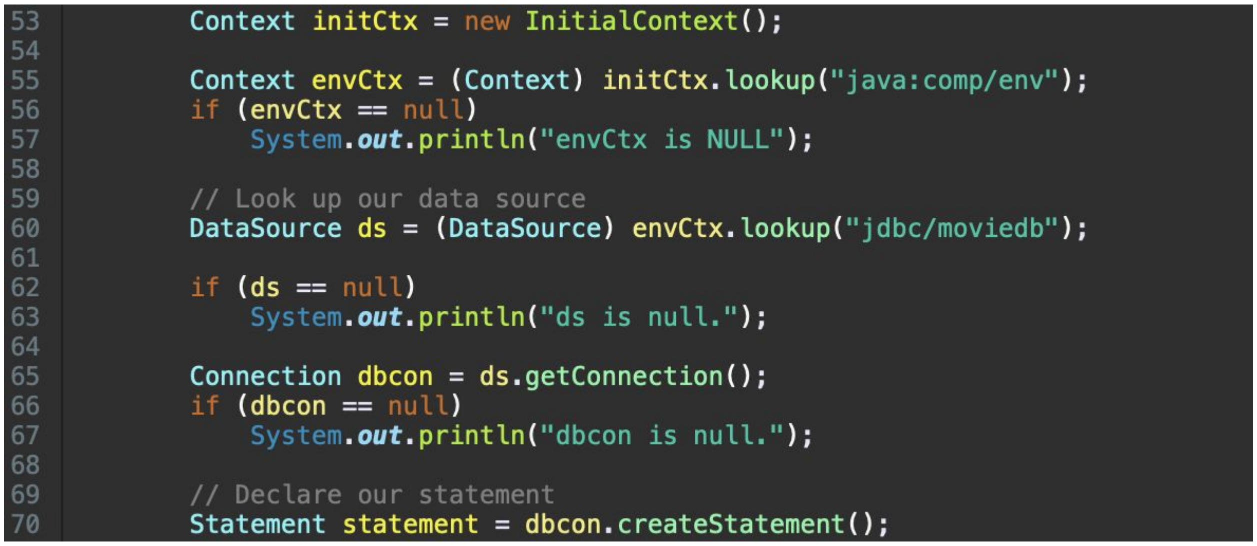
```
22    <resource-ref>
23        <description>
24            Resource reference to a factory for java.sql.Connection
25            instances that may be used for talking to a particular
26            database that is configured in the server.xml file.
27        </description>
28        <res-ref-name>jdbc/moviedb</res-ref-name>
29        <res-type>javax.sql.DataSource</res-type>
30        <res-auth>Container</res-auth>
31    </resource-ref>
```

#3



```
            Context initCtx = new InitialContext();

            Context envCtx = (Context) initCtx.lookup("java:comp/env");
            if (envCtx == null)
                out.println("envCtx is NULL");

            // Look up our data source
            DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

            if (ds == null)
                out.println("ds is null.");

            Connection dbcon = ds.getConnection();
            if (dbcon == null)
                out.println("dbcon is null.");
```

#4

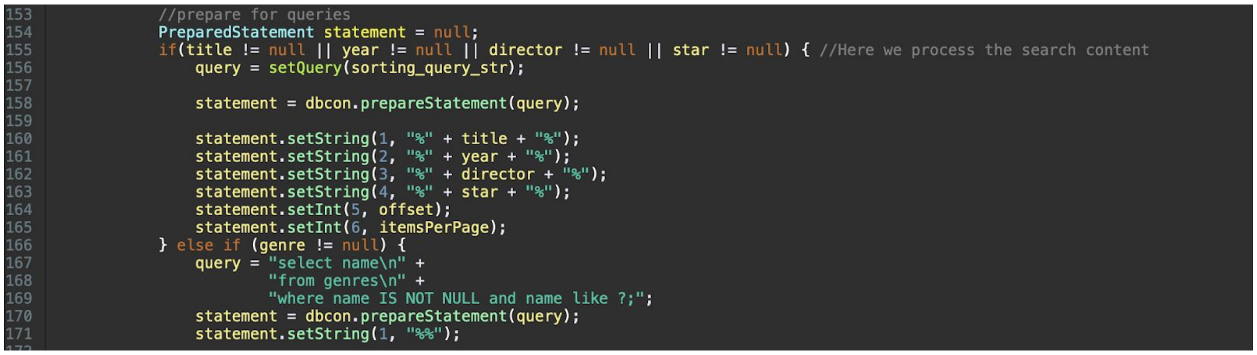


```
        Context initCtx = new InitialContext();

        Context envCtx = (Context) initCtx.lookup("java:comp/env");
        if (envCtx == null)
            System.out.println("envCtx is NULL");

        // Look up our data source
        DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");

        if (ds == null)
            System.out.println("ds is null.");

        Connection dbcon = ds.getConnection();
        if (dbcon == null)
            System.out.println("dbcon is null.");

        // Declare our statement
        Statement statement = dbcon.createStatement();
```

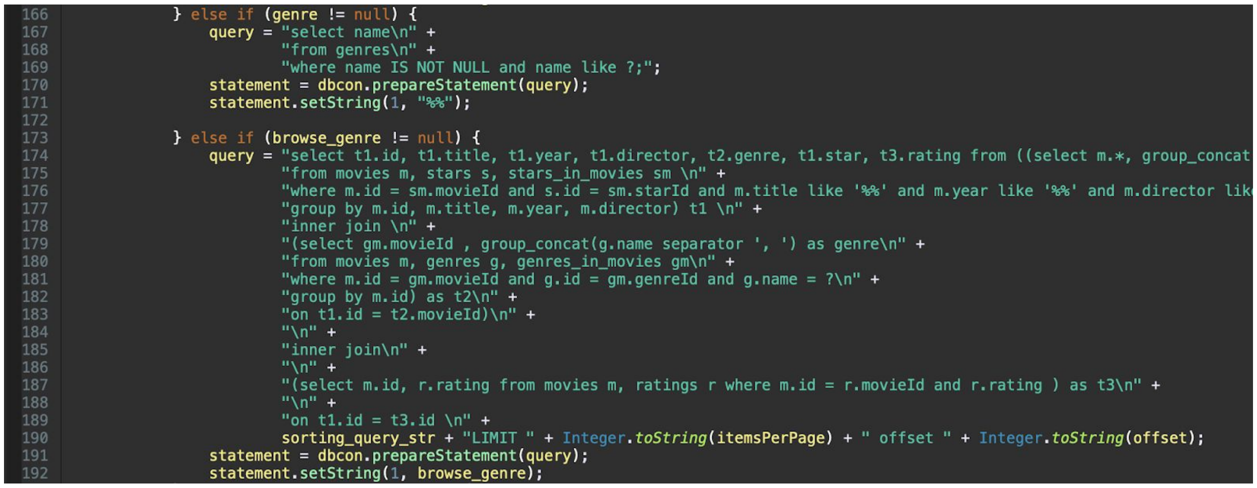
- How did you use Prepared Statements?
    - Prepared statements were used throughout the whole web application whenever a query needed to be compiled and executed to prevent mysql-injections. There is a single servlet called “MovieServlet” which handles any type of searching that the user wants to do, either from the search bar, search by title/genre, or filter. I use the url parameters to determine which parameters will go in the query in order to get the correct results from mysql.

    - File name, line numbers as in Github
      cs122b-winter19-team-24/src/MovieServlet.java (lines 155-282)

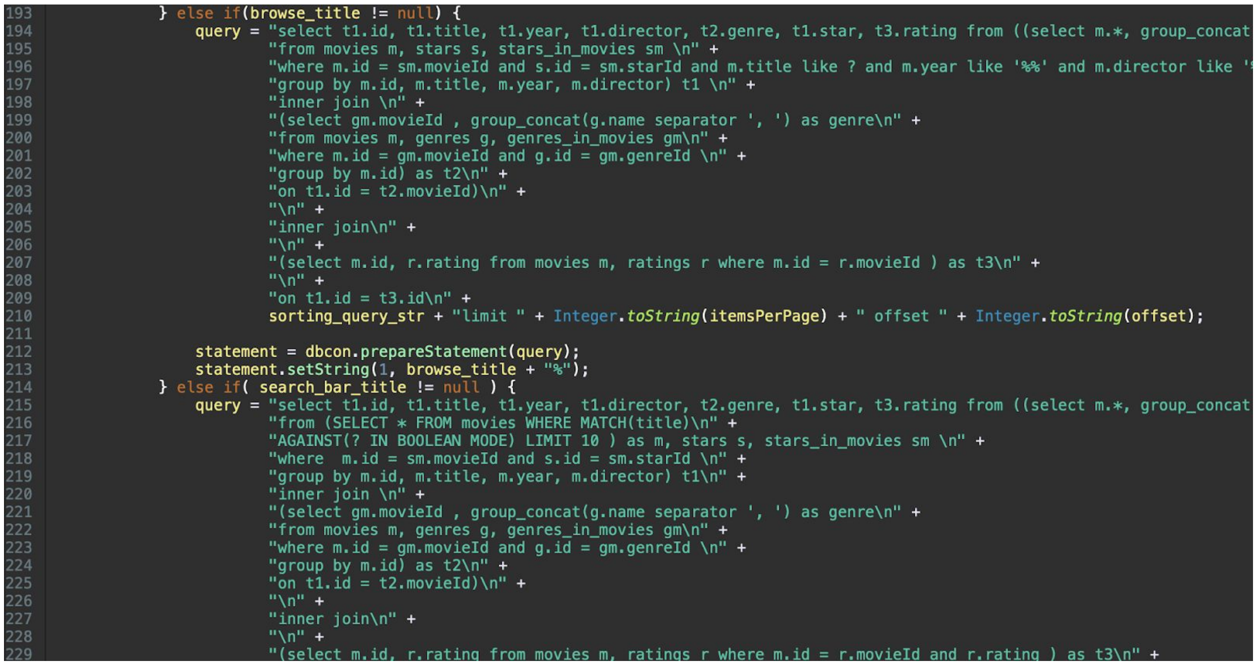
    - Snapshots showing use in your code



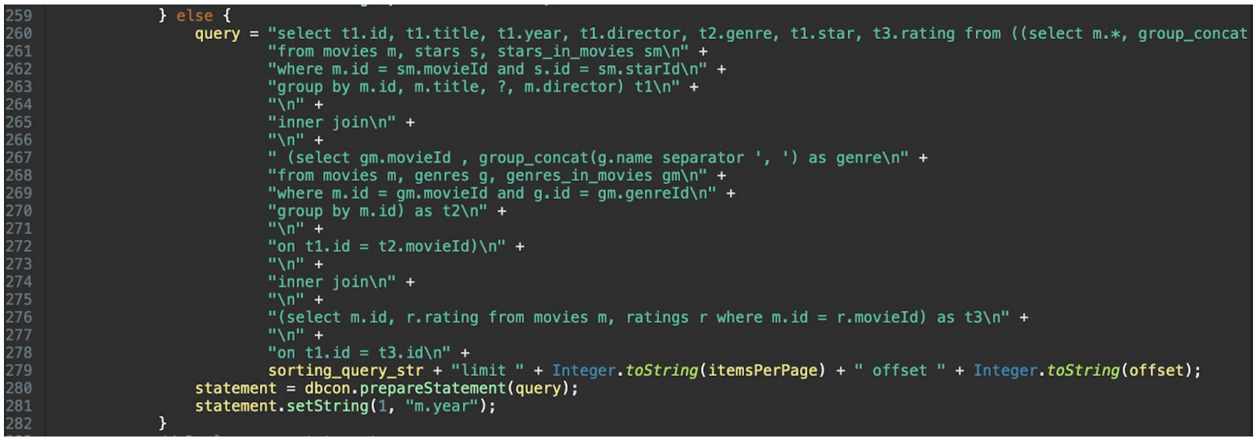
```java
            //prepare for queries
            PreparedStatement statement = null;
            if(title != null || year != null || director != null || star != null) { //Here we process the search content
                query = setQuery(sorting_query_str);

                statement = dbcon.prepareStatement(query);

                statement.setString(1, "%" + title + "%");
                statement.setString(2, "%" + year + "%");
                statement.setString(3, "%" + director + "%");
                statement.setString(4, "%" + star + "%");
                statement.setInt(5, offset);
                statement.setInt(6, itemsPerPage);
            } else if (genre != null) {
                query = "select name\n" +
                        "from genres\n" +
                        "where name IS NOT NULL and name like ?;";
                statement = dbcon.prepareStatement(query);
                statement.setString(1, "%%");
```



```java
            } else if (genre != null) {
                query = "select name\n" +
                        "from genres\n" +
                        "where name IS NOT NULL and name like ?;";
                statement = dbcon.prepareStatement(query);
                statement.setString(1, "%%");

            } else if (browse_genre != null) {
                query = "select t1.id, t1.title, t1.year, t1.director, t2.genre, t1.star, t3.rating from ((select m.*, group_concat
                        "from movies m, stars s, stars_in_movies sm \n" +
                        "where m.id = sm.movieId and s.id = sm.starId and m.title like '%%' and m.year like '%%' and m.director lik
                        "group by m.id, m.title, m.year, m.director) t1 \n" +
                        "inner join \n" +
                        "(select gm.movieId , group_concat(g.name separator ', ') as genre\n" +
                        "from movies m, genres g, genres_in_movies gm\n" +
                        "where m.id = gm.movieId and g.id = gm.genreId and g.name = ?\n" +
                        "group by m.id) as t2\n" +
                        "on t1.id = t2.movieId)\n" +
                        "\n" +
                        "inner join\n" +
                        "\n" +
                        "(select m.id, r.rating from movies m, ratings r where m.id = r.movieId and r.rating ) as t3\n" +
                        "\n" +
                        "on t1.id = t3.id \n" +
                        sorting_query_str + "LIMIT " + Integer.toString(itemsPerPage) + " offset " + Integer.toString(offset);
                statement = dbcon.prepareStatement(query);
                statement.setString(1, browse_genre);
```



```java
        } else if(browse_title != null) {
            query = "select t1.id, t1.title, t1.year, t1.director, t2.genre, t1.star, t3.rating from ((select m.*, group_concat
                    "from movies m, stars s, stars_in_movies sm \n" +
                    "where m.id = sm.movieId and s.id = sm.starId and m.title like ? and m.year like '%%' and m.director like '
                    "group by m.id, m.title, m.year, m.director) t1 \n" +
                    "inner join \n" +
                    "(select gm.movieId , group_concat(g.name separator ', ') as genre\n" +
                    "from movies m, genres g, genres_in_movies gm\n" +
                    "where m.id = gm.movieId and g.id = gm.genreId \n" +
                    "group by m.id) as t2\n" +
                    "on t1.id = t2.movieId)\n" +
                    "\n" +
                    "inner join\n" +
                    "\n" +
                    "(select m.id, r.rating from movies m, ratings r where m.id = r.movieId ) as t3\n" +
                    "\n" +
                    "on t1.id = t3.id\n" +
                    sorting_query_str + "limit " + Integer.toString(itemsPerPage) + " offset " + Integer.toString(offset);

            statement = dbcon.prepareStatement(query);
            statement.setString(1, browse_title + "%");
        } else if( search_bar_title != null ) {
            query = "select t1.id, t1.title, t1.year, t1.director, t2.genre, t1.star, t3.rating from ((select m.*, group_concat
                    "from (SELECT * FROM movies WHERE MATCH(title)\n" +
                    "AGAINST(? IN BOOLEAN MODE) LIMIT 10 ) as m, stars s, stars_in_movies sm \n" +
                    "where  m.id = sm.movieId and s.id = sm.starId \n" +
                    "group by m.id, m.title, m.year, m.director) t1\n" +
                    "inner join \n" +
                    "(select gm.movieId , group_concat(g.name separator ', ') as genre\n" +
                    "from movies m, genres g, genres_in_movies gm\n" +
                    "where m.id = gm.movieId and g.id = gm.genreId \n" +
                    "group by m.id) as t2\n" +
                    "on t1.id = t2.movieId)\n" +
                    "\n" +
                    "inner join\n" +
                    "\n" +
                    "(select m.id, r.rating from movies m, ratings r where m.id = r.movieId and r.rating ) as t3\n" +
```



```java
        } else {
            query = "select t1.id, t1.title, t1.year, t1.director, t2.genre, t1.star, t3.rating from ((select m.*, group_concat
                    "from movies m, stars s, stars_in_movies sm\n" +
                    "where m.id = sm.movieId and s.id = sm.starId\n" +
                    "group by m.id, m.title, ?, m.director) t1\n" +
                    "\n" +
                    "inner join\n" +
                    "\n" +
                    " (select gm.movieId , group_concat(g.name separator ', ') as genre\n" +
                    "from movies m, genres g, genres_in_movies gm\n" +
                    "where m.id = gm.movieId and g.id = gm.genreId\n" +
                    "group by m.id) as t2\n" +
                    "\n" +
                    "on t1.id = t2.movieId)\n" +
                    "\n" +
                    "inner join\n" +
                    "\n" +
                    "(select m.id, r.rating from movies m, ratings r where m.id = r.movieId) as t3\n" +
                    "\n" +
                    "on t1.id = t3.id\n" +
                    sorting_query_str + "limit " + Integer.toString(itemsPerPage) + " offset " + Integer.toString(offset);
            statement = dbcon.prepareStatement(query);
            statement.setString(1, "m.year");
        }
```

Task 2

- Address of AWS and Google instances
  Instance1: http://18.222.177.195:8080/cs122b-winter19-team-24/index.html
  Instance2: http://52.15.74.2:8080/cs122b-winter19-team-24/index.html
  Instance3: http://3.17.70.3:8080/cs122b-winter19-team-24/index.html

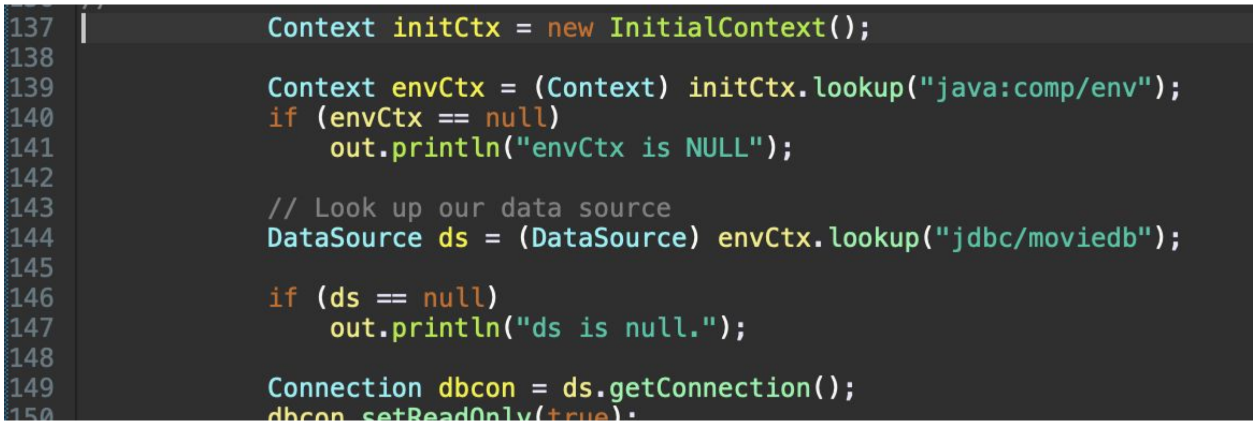
  Google: http://http://35.236.85.61/cs122b-winter19-team-24/index.html

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

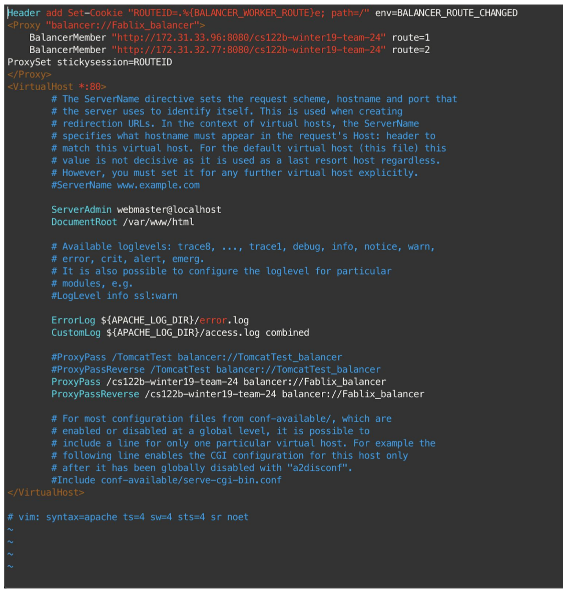
- Explain how connection pooling works with two backend SQL (in your code)?
    - I defined a resource which used the replication driver provided by mysql and jdbc and added both the master and slave mysql instances. And with sticky sessions, each user will only go to one of their specified tomcat instances.

    - File name, line numbers as in Github
      cs122b-winter19-team-24/src/MovieServlet.java (lines 137-150)

    - Snapshots



```
137             Context initCtx = new InitialContext();
138
139             Context envCtx = (Context) initCtx.lookup("java:comp/env");
140             if (envCtx == null)
141                 out.println("envCtx is NULL");
142
143             // Look up our data source
144             DataSource ds = (DataSource) envCtx.lookup("jdbc/moviedb");
145
146             if (ds == null)
147                 out.println("ds is null.");
148
149             Connection dbcon = ds.getConnection();
```



```
Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy "balancer://Fablix_balancer">
    BalancerMember "http://172.31.33.96:8080/cs122b-winter19-team-24" route=1
    BalancerMember "http://172.31.32.77:8080/cs122b-winter19-team-24" 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

        #ProxyPass /TomcatTest balancer://TomcatTest_balancer
        #ProxyPassReverse /TomcatTest balancer://TomcatTest_balancer
        ProxyPass /cs122b-winter19-team-24 balancer://Fablix_balancer
        ProxyPassReverse /cs122b-winter19-team-24 balancer://Fablix_balancer

        # 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
</VirtualHost>

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

- How read/write requests were routed?

    - File name, line numbers as in Github

    - Snapshots

Task 3

- Have you uploaded the log files to Github? Where is it located?
  No

- Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
  Yes
  cs122b-winter19-team-24/WebContent/jmeter_report.html

- Have you uploaded the script to Github? Where is it located?
  Yes
  cs122b-winter19-team-24/read_values.py

- Have you uploaded the WAR file and README to Github? Where is it located?
  Yes
  cs122b-winter19-team-24/cs122b-winter19-team-24.war