

Task 1

- How did you use connection pooling?

The connections to MySQL instance is defined in /webapp/META-INF/context.xml for both master and slave. Inside servlets, I use Context.lookup to find and declare MySQL instance I use (defined in the context.xml), and use datasource.getConnection() method to get the connection from the MySQL and close the connection after request end.

- File name, line numbers as in Github

File name: context.xml(/Fablix_Web/src/main/webapp/META-INF/)

Line numbers: main instance: line 6-12; master instance: line 14-20; slave

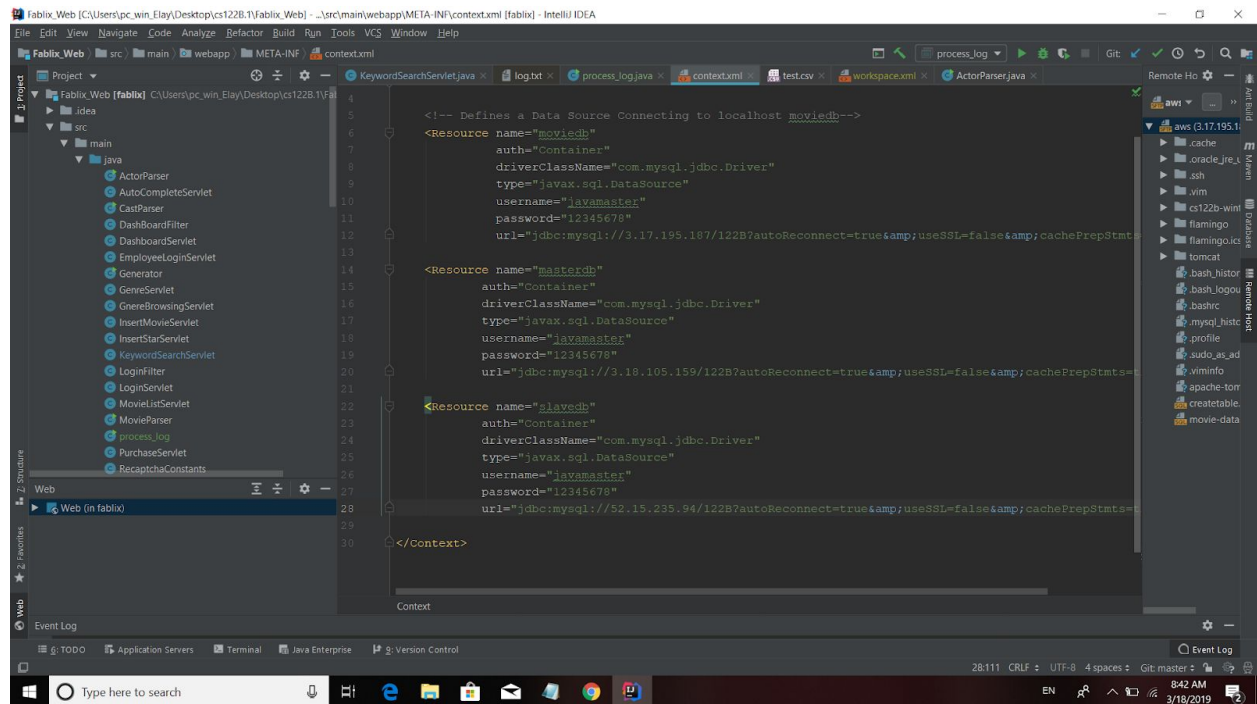
Instance:22-28

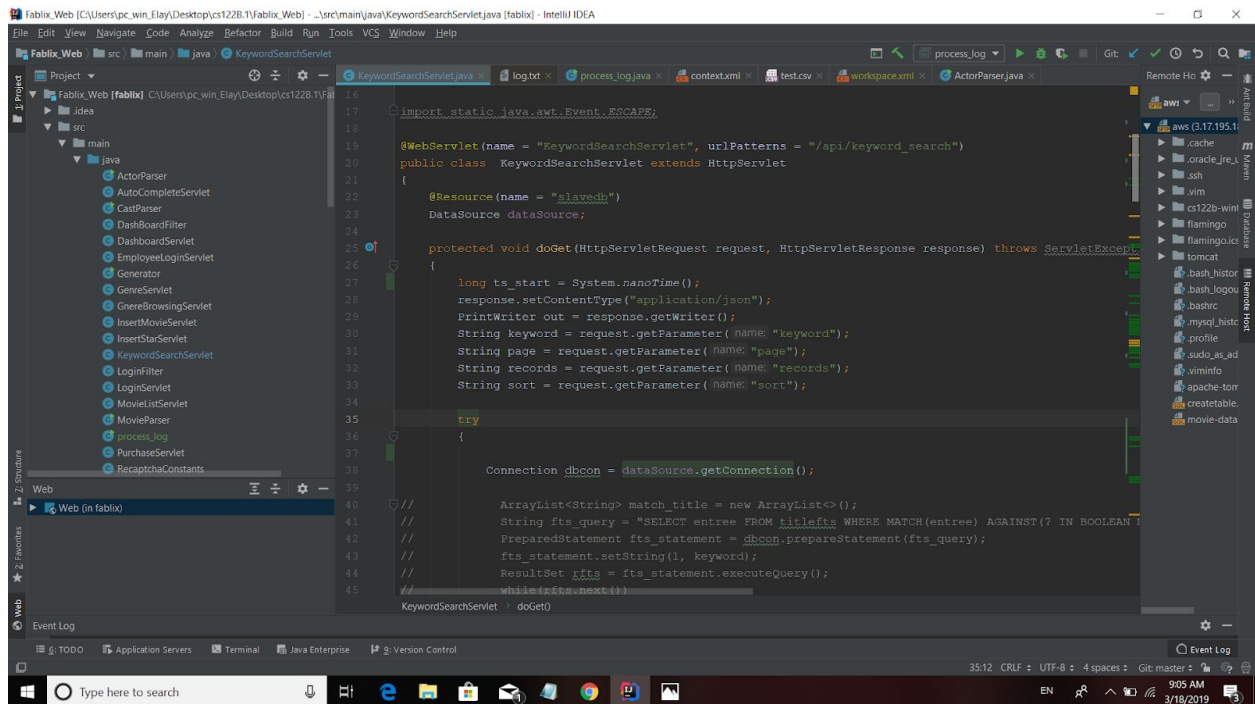
File name: KeywordSearchServlet.java(/Fablix_Web/src/main/java/)

Line numbers: declare the connection to the MySQL instance: line 67-75

(I applied connection pooling to all servlets which have interaction with MySQL instance)

- Snapshots showing use in your code





- How did you use Prepared Statements?

For all servlets involved in search, I use the prepared statement. I construct basically the complete query and replace all user input parts with placeholders("?"), and after I receive the user input, I use the setString(the_index_of_placeholder, user_input) to fill the SQL query.

- File name, line numbers as in Github

File name: KeywordSerachServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 51-64, line 104-107

File name: SesarServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 42-71, line 96-101

File name: AutoCompleteServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 35-44

File name: EmployeeLoginServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 70-72, line 93-95

File name: GnerServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 30-31(no placeholder)

File name: GnerBrowsingServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 38-49, line 78-82

File name: InsertMovieServlet(/Fablix_Web/src/main/java/)

Line numbers: line 42-48

File name: InsertStarServlet(/Fablix_Web/src/main/java/)

Line numbers: line 35-39

File name: LoginServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 117-119

File name: SingleMovieServlet.java(/Fablix_Web/src/main/java/)

Line numbers: line 33-42, line 60-64

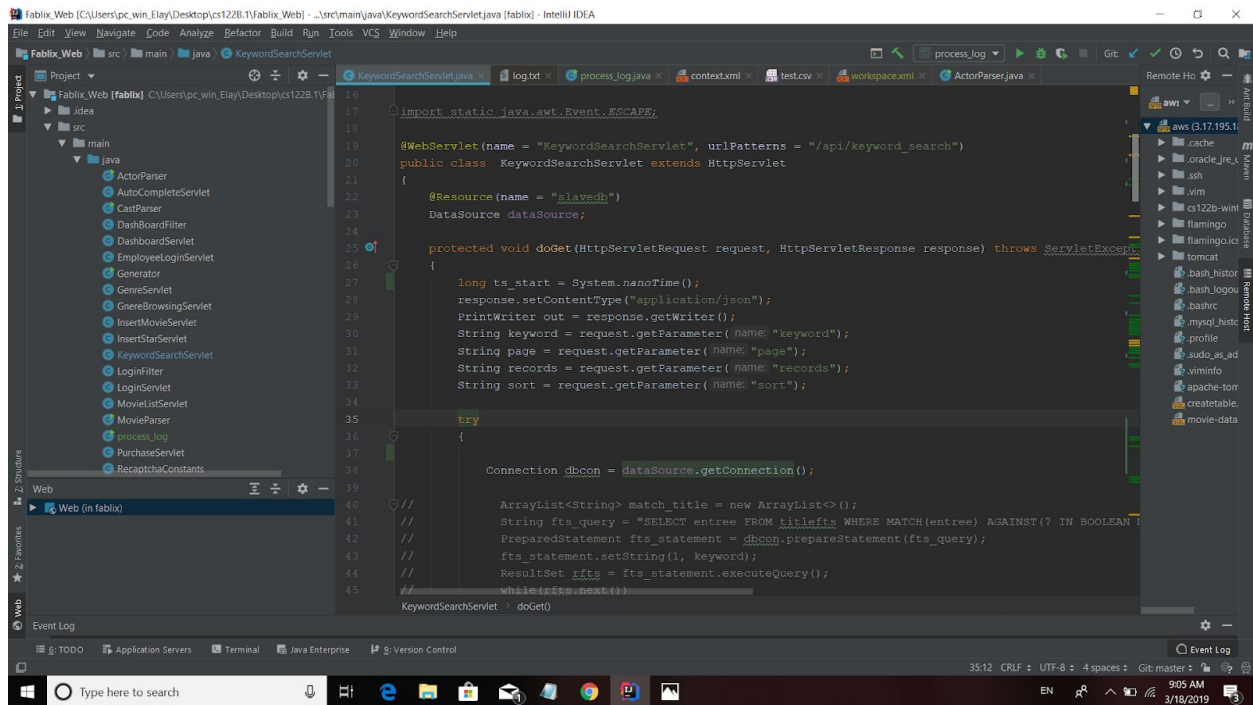
File name: SingleStarServlet.java(/Fablix_Web/src/main/java/)

Line number: line 32-39, line 52-56

File name: TitleBrowsingServlet.java(/Fablix_Web/src/main/java/)

Line number: line 38-51

- Snapshots showing use in your code



File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project: Fablix_Web [C:\Users\pc_w... \Fablix_Web] - \src\main\java\KeywordSearchServlet.java [fablix] - IntelliJ IDEA

KeywordSearchServlet.java

```
49 String select query = "SELECT movies.id, title, `year`, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR '
50 String from_query = "FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_movies
51 String where_join = "WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movies
52 String title_condition = "AND MATCH(movies.title) AGAINST(? IN BOOLEAN MODE) AND ed(?,movies.title) <= 10 ";
53 String group_clause = "GROUP BY id";
54 String order_clause = get_sort_clause(sort);
55 String offset_clause = get_offset_clause(page, records);
56
57 String query = select_query + from_query + where_join + title_condition + group_clause + order_clause + offset_clause;
58
59 String queryt = "Select count(distinct movies.id) as a" + from_query + where_join + title_condition;
60 String fuzzy_query = "SELECT * FROM [" + query + "] WHERE ed(title,?) <= 3";
61 String fuzzy_count = "SELECT COUNT(*) AS a FROM " + fuzzy_query;
62 PreparedStatement statement = dbcon.prepareStatement(query);
63 PreparedStatement statement1 = dbcon.prepareStatement(queryt);
64 String[] con = keyword.split(regex "\\s+");
65 String f = "";
66 for(String each: con)
67 {
68     f += (" " + each + " ");
69 }
70 statement.setString( parameterIndex 1, f );
71 statement.setString( parameterIndex 2, keyword);
72 statement1.setString( parameterIndex 1, f );
73 statement1.setString( parameterIndex 2, keyword);
74 long tj_start = System.nanoTime();
75 ResultSet resultSet = statement.executeQuery();
76 ResultSet w = statement1.executeQuery();
77
78 KeywordSearchServlet.doGet()
```

Run: Tomcat 8.5.37

127:1 CRLF : UTF-8 : 4 spaces : Git: master : 11:46 AM 3/18/2019

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

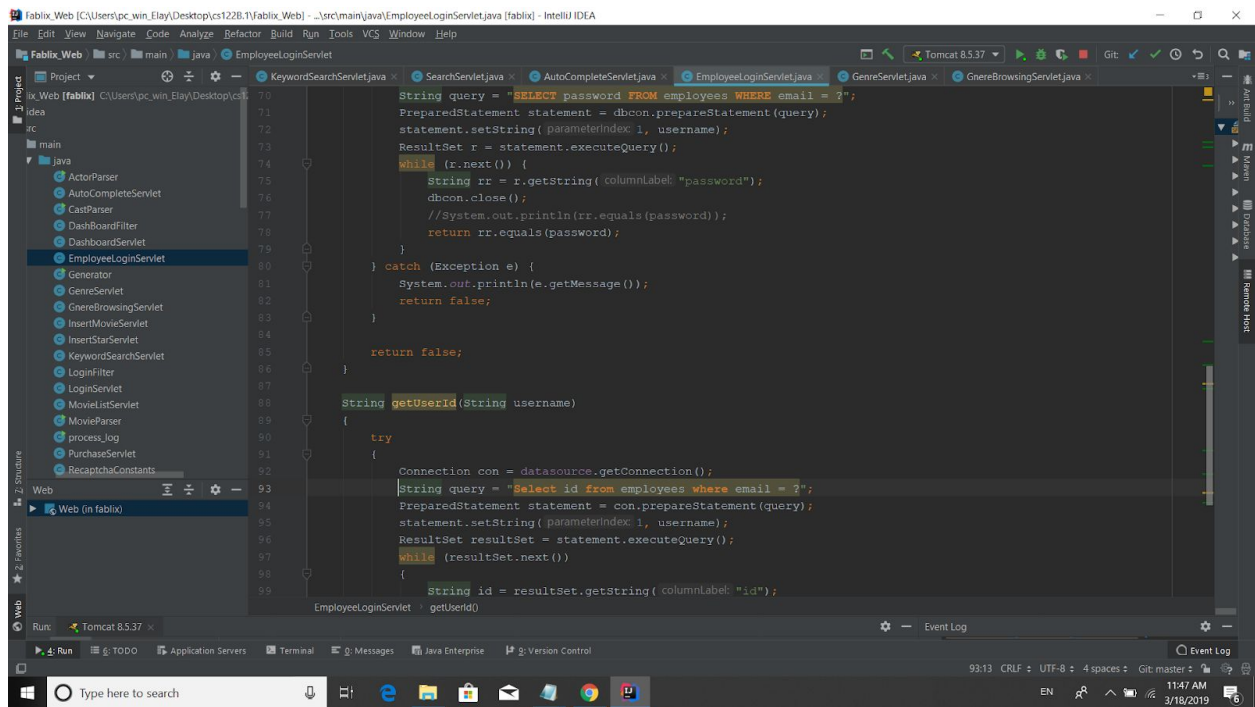
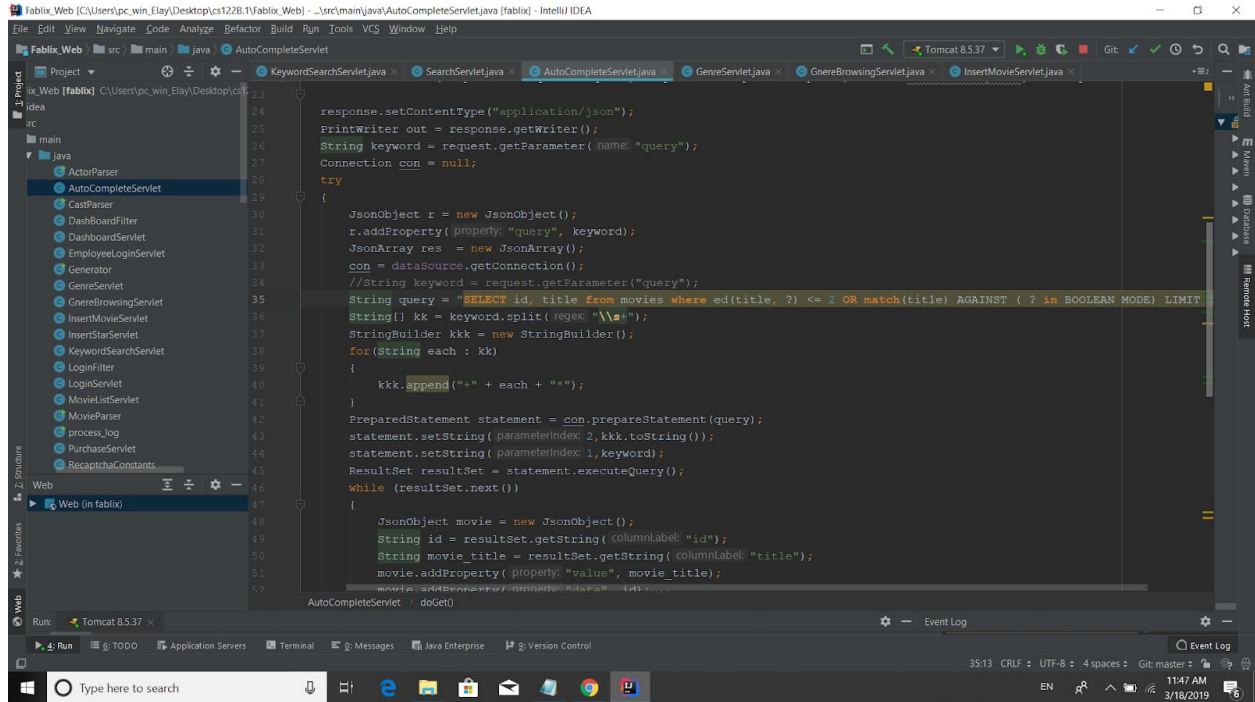
Project: Fablix_Web [C:\Users\pc_w... \Fablix_Web] - \src\main\java\SearchServlet.java [fablix] - IntelliJ IDEA

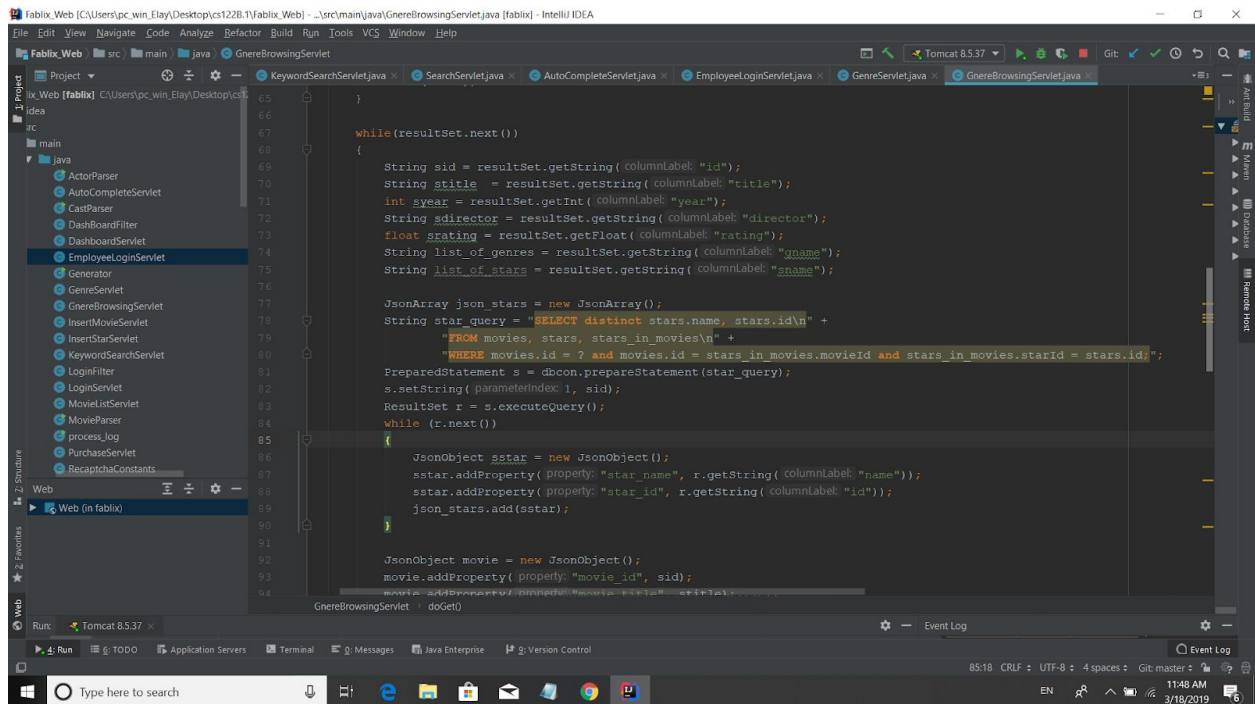
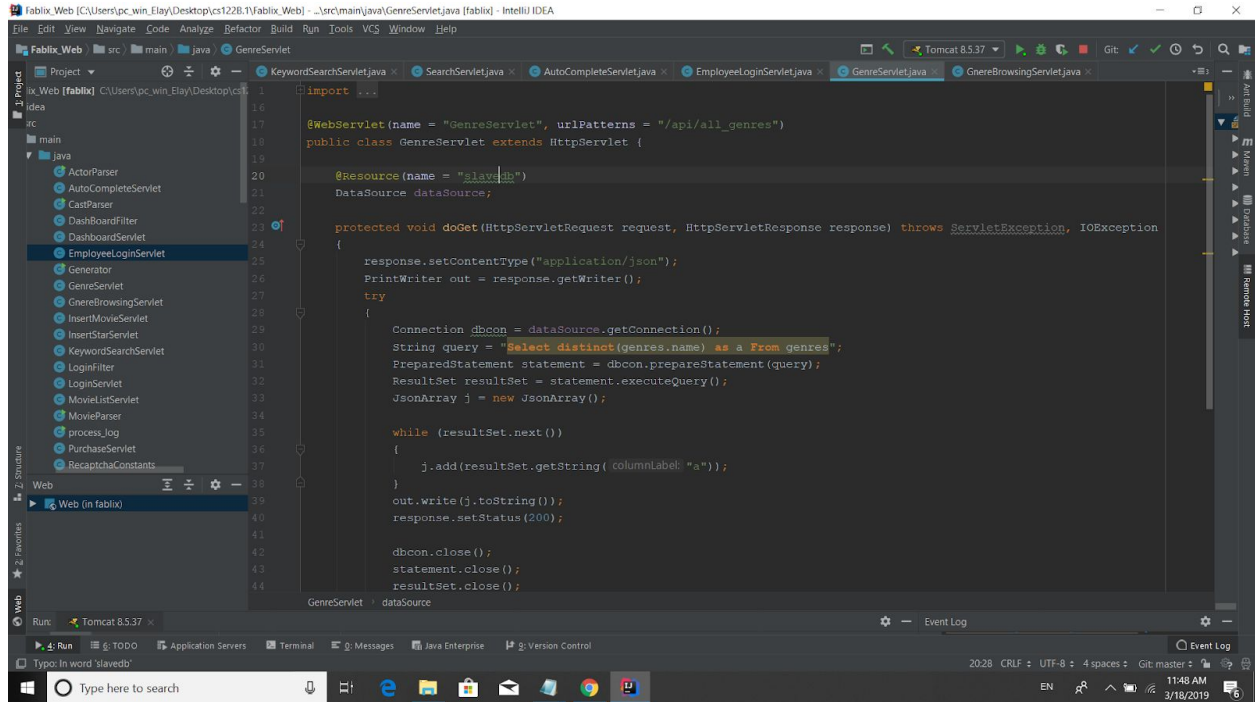
SearchServlet.java

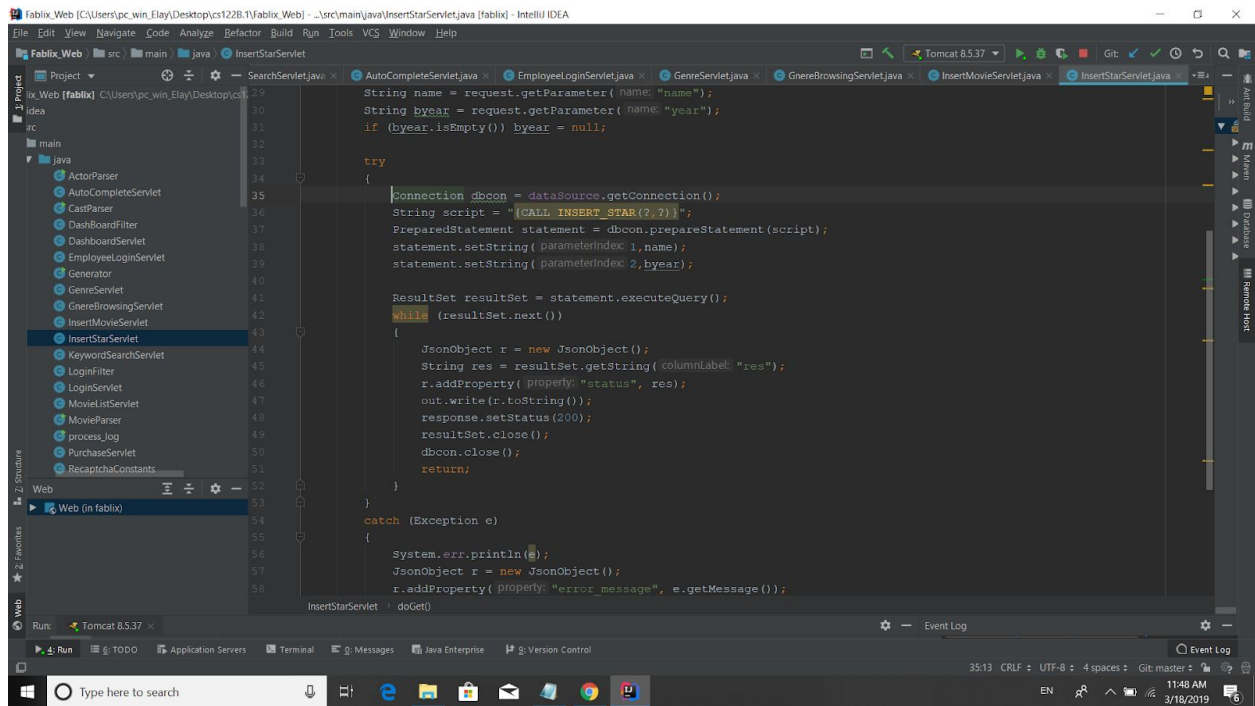
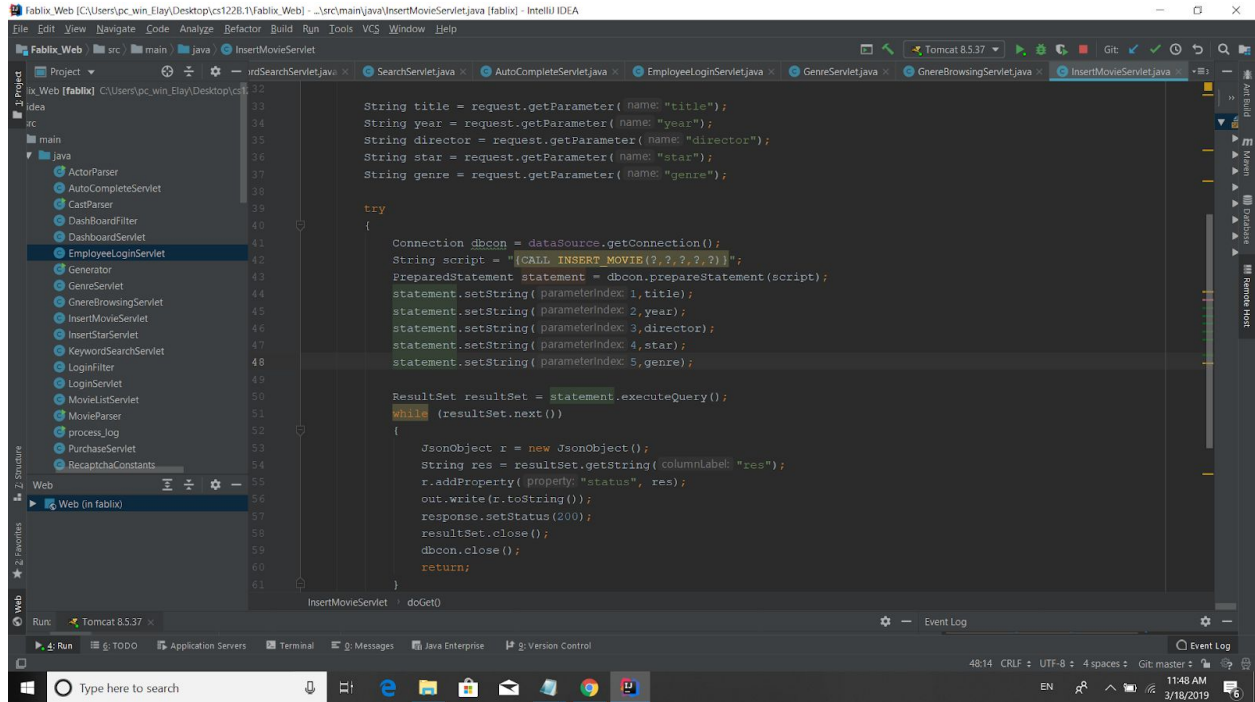
```
43 String from_query = "FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_movies
44 String where_join = "WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movies
45 String title_condition = "and movies.title like ? \n";
46 String year_condition = "and movies.year = ? \n";
47 String directory_condition = "and movies.director like ? \n";
48 String star_condition = "and stars.name like ? \n";
49
50 String group_clause = "GROUP BY title ";
51 String order_clause = get_sort_clause(sort);
52 String offset_clause = get_offset_clause(page, records);
53 if (year.isEmpty())(year = "%"; year_condition = "and movies.year like ? ";)
54 String query = select_query + from_query + where_join + title_condition + year_condition
55 + directory_condition + star_condition + group_clause + order_clause + offset_clause;
56
57 String queryt = "SELECT COUNT(distinct movies.id) as a" + from_query + where_join + title_condition + year_condition
58 + directory_condition + star_condition;
59
60 System.out.println(queryt);
61
62 PreparedStatement statement = dbcon.prepareStatement(query);
63 PreparedStatement statement1 = dbcon.prepareStatement(queryt);
64 statement.setString( parameterIndex 1, X: "%" + title + "%");
65 statement1.setString( parameterIndex 1, X: "%" + title + "%");
66 statement.setString( parameterIndex 2, year);
67 statement1.setString( parameterIndex 2, year);
68 statement.setString( parameterIndex 3, X: "%" + director + "%");
69 statement1.setString( parameterIndex 3, X: "%" + director + "%");
70 statement.setString( parameterIndex 4, X: "%" + star + "%");
71 statement1.setString( parameterIndex 4, X: "%" + star + "%");
72
73 SearchServlet
```

Run: Tomcat 8.5.37

96:1 CRLF : UTF-8 : 4 spaces : Git: master : 11:47 AM 3/18/2019







File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project: Fablix_Web [C:\Users\pc_w... Desktop\cs122b\1\Fablix_Web] - IntelliJ IDEA

src \ main \ java \ LoginServlet

```
71 out.write(responseJsonObject.toString());
72 }
73 }
74 out.close();
75 }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }
```

String getUserId(String username)

```
{
    Connection con = null;
    try
    {
        con = datasource.getConnection();
        String query = "Select id from customers where email = ? ";
        PreparedStatement statement = con.prepareStatement(query);
        statement.setString(1, username);
        ResultSet resultSet = statement.executeQuery();
        while (resultSet.next())
        {
            String id = resultSet.getString("id");
            con.close();
            statement.close();
            resultSet.close();
            return id;
        }
    }
    catch (Exception e)
    {
        System.out.println(e.getMessage());
        return "";
    }
}
```

Run: Tomcat 8.5.37

Event Log

11:13 CRLF : UTF-8 : 4 spaces : Git: master : 3/18/2019

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Project: Fablix_Web [C:\Users\pc_w... Desktop\cs122b\1\Fablix_Web] - IntelliJ IDEA

src \ main \ java \ SingleMovieServlet

```
44 ResultSet rs = statement.executeQuery();
45
46 JSONArray movie_list = new JSONArray();
47
48 while(rs.next())
49 {
50     String movie_id = rs.getString("id");
51     String title = rs.getString("title");
52     int year = rs.getInt("year");
53     String director = rs.getString("director");
54     float rating = rs.getFloat("rating");
55     if(Float.compare(rating, 0f) == 0)(rating = -1);
56     String list_of_genres = rs.getString("genres");
57     String list_of_stars = rs.getString("stars");
58
59     JSONArray json_stars = new JSONArray();
60     String star_query = "SELECT distinct stars.name, stars.id\n" +
61         "FROM movies, stars, stars_in_movies\n" +
62         "WHERE movies.id = ? and movies.id = stars_in_movies.movieid and stars_in_movies.starid = stars.id";
63     PreparedStatement s = dbcon.prepareStatement(star_query);
64     s.setString(1, movie_id);
65     ResultSet r = s.executeQuery();
66     while (r.next())
67     {
68         JSONObject star = new JSONObject();
69         star.addProperty("star_name", r.getString("name"));
70         star.addProperty("star_id", r.getString("id"));
71         json_stars.add(star);
72     }
73 }
```

SingleMovieServlet.doGet()

Run: Tomcat 8.5.37

Event Log

6:14 CRLF : UTF-8 : 4 spaces : Git: master : 3/18/2019


```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("application/json");
    String id = request.getParameter("id");
    PrintWriter out = response.getWriter();

    try {
        Connection dbcon = dataSource.getConnection();

        String query = "SELECT stars.id, stars.birthYear, stars.name, stars.birthYear, GROUP_CONCAT(distinct movies.title SEPARATOR ',' ) as movie_list\n"
            + "FROM stars_in_movies INNER JOIN movies ON stars_in_movies.movieId = movies.id\n"
            + "INNER JOIN stars ON stars_in_movies.starId = stars.id\n"
            + "WHERE stars.id = ?\n"
            + "GROUP BY name;";

        PreparedStatement statement = dbcon.prepareStatement(query);
        statement.setString(1, id);

        ResultSet rs = statement.executeQuery();
        JSONArray star = new JSONArray();

        while (rs.next()) {
            String name = rs.getString("name");
            String star_id = rs.getString("id");
            String birth_year = rs.getString("birthYear");
            String movie_list = rs.getString("movie_list");

            JSONObject json_movie = new JSONObject();
            json_movie.put("id", star_id);
            json_movie.put("name", name);
            json_movie.put("birthYear", birth_year);
            json_movie.put("movie_list", movie_list);
            star.add(json_movie);
        }

        out.print(star.toString());
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
try {
    Connection dbcon = dataSource.getConnection();
    String fc = request.getParameter("fc");
    String sort = request.getParameter("sort");
    String page = request.getParameter("page");
    String records = request.getParameter("records");

    String select_query = "SELECT movies.id, title, year, director, rating, GROUP_CONCAT(distinct genres.name SEPARATOR ',' ) as genres\n"
        + "FROM movies left join ratings r on movies.id = r.movieId, genres, genres_in_movies, stars, stars_in_movies\n"
        + "WHERE movies.id = genres_in_movies.movieId and genres_in_movies.genreId = genres.id and stars_in_movies.starId = stars.id\n"
        + "AND movies.title like ? ";
    String group_clause = "GROUP BY title";
    String order_clause = get_sort_clause(sort);
    String offset_clause = get_offset_clause(page, records);

    String query = select_query + from_query + where_join + title_condition + group_clause + order_clause + offset_clause;
    String queryt = "select count(distinct movies.id) as a" + from_query + where_join + title_condition;
    PreparedStatement statement = dbcon.prepareStatement(query);
    PreparedStatement statementt = dbcon.prepareStatement(queryt);
    statement.setString(1, fc + "%");
    statementt.setString(1, fc + "%");

    ResultSet resultSet = statement.executeQuery();
    ResultSet w = statementt.executeQuery();

    System.out.println(queryt);

    JSONArray movie_list = new JSONArray();
    while (resultSet.next()) {
        JSONObject json_movie = new JSONObject();
        json_movie.put("id", resultSet.getString("id"));
        json_movie.put("title", resultSet.getString("title"));
        json_movie.put("year", resultSet.getString("year"));
        json_movie.put("director", resultSet.getString("director"));
        json_movie.put("rating", resultSet.getString("rating"));
        json_movie.put("genres", resultSet.getString("genres"));
        movie_list.add(json_movie);
    }

    out.print(movie_list.toString());
} catch (SQLException e) {
    e.printStackTrace();
}
```

Task 2

- Address of AWS and Google instances
AWS Balancer: 3.17.195.187/fablix/
AWS Master: 52.15.235.94
AWS Slave: 3.18.105.159
GCP Balancer: 34.73.131.140/fablix/

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

Yes

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

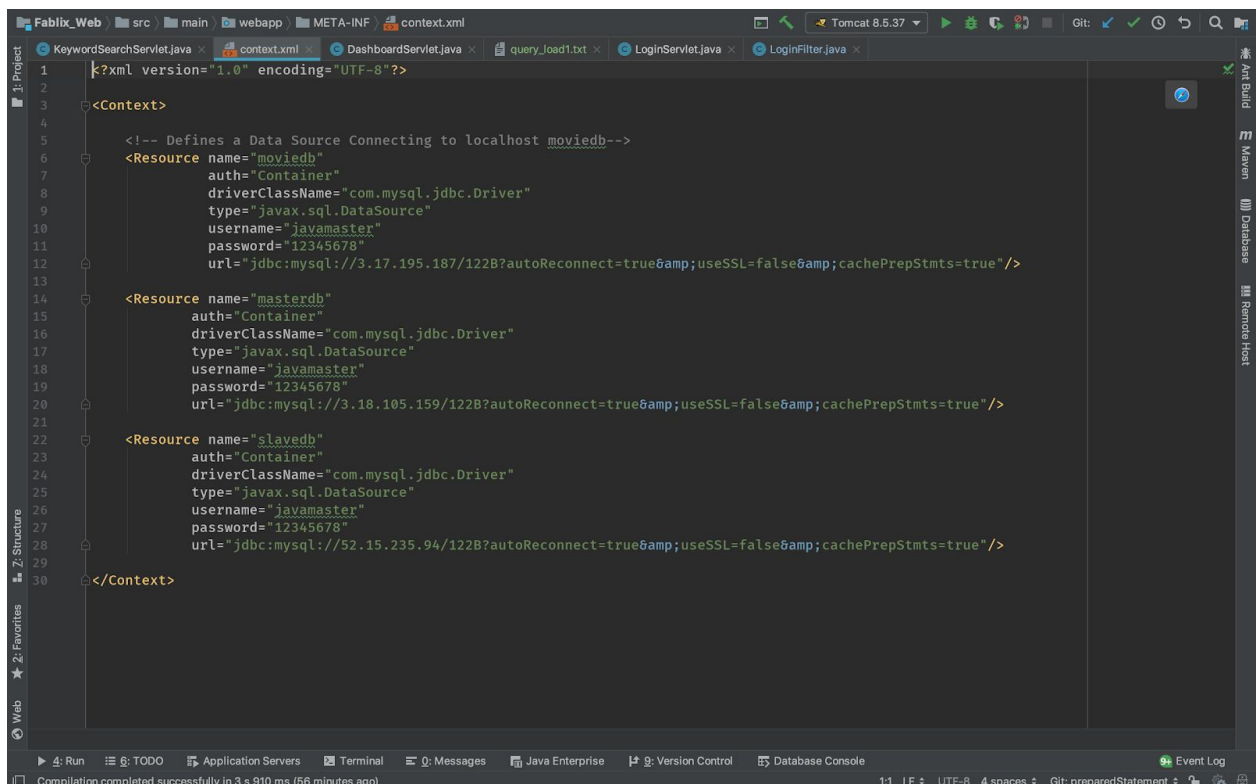
There are two database resources written in context.xml and we'll choose one to use based on where it was used.

- File name, line numbers as in Github

Fablix_Web/src/main/webapp/META-INF/context.xml

Line 14-28

- Snapshots



```

1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <Context>
4
5      <!-- Defines a Data Source Connecting to localhost moviedb-->
6      <Resource name="moviedb"
7              auth="Container"
8              driverClassName="com.mysql.jdbc.Driver"
9              type="javax.sql.DataSource"
10             username="javamaster"
11             password="12345678"
12             url="jdbc:mysql://3.17.195.187/122B?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
13
14      <Resource name="masterdb"
15              auth="Container"
16              driverClassName="com.mysql.jdbc.Driver"
17              type="javax.sql.DataSource"
18              username="javamaster"
19              password="12345678"
20              url="jdbc:mysql://3.18.105.159/122B?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
21
22      <Resource name="slavedb"
23              auth="Container"
24              driverClassName="com.mysql.jdbc.Driver"
25              type="javax.sql.DataSource"
26              username="javamaster"
27              password="12345678"
28              url="jdbc:mysql://52.15.235.94/122B?autoReconnect=true&useSSL=false&cachePrepStmts=true"/>
29
30  </Context>
  
```

- How read/write requests were routed?

We randomly choose which db(master or slave) will be used in the servlet that only reads the db. We choose the master db in the servlet that writes to db

- File name, line numbers as in Github

All *****Servlet.java** uses random db except for **PurchaseServlet.java**, **InsertMovieServlet.java** and **InsertStarServlet.java**, which are using master db only.

- Snapshots

Task 3

- Have you uploaded the log files to Github? Where is it located?
All log files have been uploaded to the github.
Path: /Fablix_Wed/src/main/jmeter_log/
- Have you uploaded the HTML file (with all sections including analysis, written up) to Github? Where is it located?
Yes the file have been uploaded to the project.
Path: /Fablix_Web/jmeter_report.html
- Have you uploaded the script to Github? Where is it located?
Yes:
Path: /Fablix_Web/src/main/java/process_log.java
- Have you uploaded the WAR file and README to Github? Where is it located?
The war file is /Fablix_Web/fablix.war